CULTIVATING COMMUNITY INTERACTIONS IN CITIZEN SCIENCE: CONNECTING PEOPLE TO EACH OTHER AND THE ENVIRONMENT

by

Bret Allen Finley

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

December 2017

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Bret Allen Finley

Thesis Title: Cultivating Community Interactions in Citizen Science: Connecting People to Each Other and the Environment

Date of Final Oral Examination: 23rd October 2017

The following individuals read and discussed the thesis submitted by student Bret Allen Finley, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Jerry Alan Fails, Ph.D.	Chair, Supervisory Committee
Bogdan Dit, Ph.D.	Member, Supervisory Committee
Maria Soledad Pera, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Jerry Alan Fails, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ACKNOWLEDGMENTS

This work would not have been possible without the thoughtful support of many who took a personal interest in its completion. The author would like to thank his parents, who have supported him every step of the way. The author would also like to thank God, for the good mental and physical health required for finishing this thesis. The author is also very grateful for the immense time and effort contributed by Dr. Jerry Fails. His guidance and insight were a driving force in completing this work. The author would also like to thank Boise State University and the faculty of the Computer Science department for awarding an assistantship, without which this work would not have been possible.

ABSTRACT

Citizen science leverages a distributed user-base which participates in crowd-sourced scientific inquiry. Geotagger is a citizen science project that allows people to collaboratively investigate the natural world around them and share their findings. Citizens are rarely compensated for their work and individual contributors can feel isolated which leads to motivation problems. This thesis focuses on engaging citizen scientists and motivating their contributions via social interaction and engagement. As a part of this work, a number of social enhancements have been developed as extensions to the existing Geotagger project. These enhancements and their effect on social engagement were evaluated using in-field studies and design investigations with children. In the studies, children engaged effectively with each other using the social enhancements in Geotagger, and showed a preference for the application that included these social enhancements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AJAX – Asynchronous JavaScript and XML

ADB – Android Device Bridge

API – Application Programming Interface

CHI – Computer Human Interaction

CLI – Command Line Commander

CRUD – Create Read Update Delete

CSS – Cascading Style Sheet

DDL – Data Definition Language

DML – Data Manipulation Language

DOM – Document Object Model

GIF – Graphics Interchange Format

GPS – Global Positioning System

HCI – Human Computer Interaction

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IM – Instant Message

IRB – Institutional Review Board

JDK – Java Development Kit

JPG – Joint Photographic Experts Group

JS – JavaScript

JSON – JavaScript Object Notation

MVC – Model View Controller

NPM – Node Package Manager

ORM – Object Relational Mapper

PHP – PHP HyperText Preprocessor

PNG – Portable Network Graphics

RDBMS – Relational Database Management System

REST – Representational State Transfer

SASS – Syntactically Awesome Style Sheets

SCSS – Sassy Cascading Style Sheet

SDK – Software Development Kit

SQL – Structured Query Language

SMS – Short Message Service

SSL – Secure Sockets Layer

UI – User Interface

URL – Uniform Resource Locator

XML – Extensible Markup Language

CHAPTER 1

INTRODUCTION

The aim of citizen science is to create a distributed community of data collectors and scientists. Citizen science projects oftentimes do not include social features to connect citizen scientists with one another. Those that do, do not make social interaction an integral part of the project workflow. Furthermore, citizen science projects do not usually provide a monetary incentive for contribution, so scientists must volunteer their own time. This research investigates social enhancements to create connections between citizen scientists, motivating further contribution. It builds on past citizen science research, specifically, the Geotagger project by creating and evaluating a suite of social enhancements. This chapter presents additional motivations for this work, familiarizing the reader with the field of citizen science and the Geotagger project.

1.1 Motivation

Citizen science is a field of scientific inquiry where research is conducted and data is collected by amateur or non-traditional scientists. The primary goal of citizen science is to enlist groups of people to record observations concerning a specific subject. In this way, citizen science seeks to connect people - who are sometimes distributed around the world - with a similar interest to accomplish a larger goal. Most citizen science projects do this by creating bite-sized, research-oriented tasks which users can accomplish quickly and easily. This encourages frequent and regular contributions by the user-base. Some projects even incorporate gamification elements like milestone rewards, user leaderboards, and forum badges to entice users to contribute more seriously.

Citizen science projects span many disciplines and interests, ranging from plants and animals to cosmic bodies. For example, Project Budburst [1] is a citizen science project which asks citizens to record observations concerning plant life found around their community. In this way, Project Budburst seeks to create a digital ecological record which educators and citizens the world over can download and use in their projects. The Old Weather Project [4] tasks citizens with reading scans of ship logs from old whaling vessels. The citizens then transcribe these documents digitally and upload the transcript to the website. Having these logs in a digital format makes them readily available to meteorologists and other researchers who have an interest in historical weather patterns. In the eBird Project [2], citizens are asked to observe bird life and record the time, place, and what species of bird was observed. This comprehensive archive of observations allows ornithologists and bird enthusiasts to gain a deeper understanding on migration and distribution patterns of different species of birds. Galaxy Zoo [27] is a project which focuses on the identification and classification of galaxies. Digitally classifying these galaxies allows scientists to sift through the large amount of galactic data available to them.

Citizen science is inherently very distributed. Because of this, smart devices are seeing increased use in this context as they are a good fit for the workflow of various citizen science projects. Smart devices can be used to take pictures and upload them to the Internet. They can send messages across the world in an instant and see daily usage in social media and network applications. They can be used to record high resolution video and audio, and can be used to share these artifacts with people around the world on social networking platforms like Facebook, Instagram, and Twitter. Smart devices have changed the nature of data recording, and in doing so, have made some specialized field equipment unnecessary. Most smart devices come standard with a high resolution camera with audio/video capabilities, this means that a considerable amount of equipment relating to this kind of data capture is not needed. While special equipment will still be required when the situation calls for it (ultra high resolution images, satellite imaging, atmospheric readings, etc.), smart devices account for a large number of citizen science use cases. Citizen science projects are starting to find that mobile applications are not only a promising means to record data, but also to share this data [36, 6]. Users can log findings directly on their mobile device, and upload video, images, audio, and textual data from wherever they are. These observations can be directly uploaded to remote servers provided the device has an Internet connection. With all of these observations in a single place, one would think that sharing these findings with the rest of the project community would be a relatively simple task and a natural flow to support user community. However, many citizen science projects [2, 4, 30] do not readily allow users to share and discuss their findings, resulting in users contributing in isolation.

Few citizen science projects encourage sharing discoveries and interacting with other project members as much as they do contributing to the project directly. The eButterfly Project [3] encourages users to submit data concerning when and where users observe various species of butterflies. However, this project has no way for users to interact with one another and share the different species of butterflies they have found. The Old Weather Project [4] deals with digitally transcribing scans of weather logs found on antique whaling vessels. Users are free to look over documents they transcribed, but are unable to view what other users have done. Many projects include a forum system in one form or another, but do not allow users to engage in one-on-one or small group discussions. Few projects seek to provide a sense of community and promote user interaction, especially on an individual scale. Social media websites like Facebook and Twitter enable users to communicate and interact with each-other and small-groups of individuals. These sites allow users to connect and engage in various discussions with other members. Users can gain a broad sense of available discussions by filtering content, usually with a search bar. They are then presented with an overview of related discussions and can choose to participate in individual discussions that interest them. This research focused on bringing these same principles to the world of citizen science, to give users a platform with which they can communicate with other members and engage in discussions surrounding the project at hand.

Contributors to citizen science projects are doing so on their own time. Very rarely is any conventional incentive offered to compensate the users for their time. Rather, project designers must leverage other motivations in order to motivate users to contribute. Giving users a sense of place or purpose is one way to motivate them [21]. It has been found that citizens will more readily contribute to projects if they feel attached to that project in some way [15]. Whether it is the subject matter of the project which interests them, or if they have friends or family who also contribute to the project, users need to feel that the project is worth their time. This work builds on this by putting emphasis on sharing discoveries and facilitating interactions between members of the project community. This is accomplished by a number of social features which built into the existing application in order to make these interactions feel like an integral part of the project, and more specifically, the user's workflow. A principle goal of this project is to make these enhancements easily available to the users. Interaction can be catalyzed by bringing presently available social features to the forefront of the application. Other features include creating spaces in which users may explore the project data in order to find other contributors with similar interests. Over the course of this thesis, incorporating these features into the existing Geotagger application increases both user interactions, as well as the user contribution rate.

1.2 Geotagger

Geotagger is a "collaborative participatory environmental inquiry system" [14]. Geotagger is focused on developing software tools for children to encourage informal scientific inquiry and environmental investigation. At its core, Geotagger is a citizen science project with a focus on environmental observations called tags. Geotagger was designed to interest children in the outside world around them, though individuals of all ages can use Geotagger. The Geotagger system workflow is made up of several key elements: tags, comments, adventures, and collections [12]. The main collective unit of Geotagger is the adventure. A tag represents the basic interactive unit of Geotagger. A tag is simply an environmental observation which can be described in terms of a picture, a name, textual description, and geographic coordinates. Users can share their thoughts about a singular tag by commenting on the tag. These are visible to other users and help to generate a discussion about what the tag represents. Adventures are groups of tags contributed by "members" of the adventure. Adventures may have certain themes, or can represent a scientific outing by a research team or class of students. Adventures consist of metadata, collections of tags, and collections of users. Tag collections serve to organize various tags, allowing users to create distinct areas for certain kinds of tags. User collections allow adventure organizers to group users into different teams. These features combine to create an environment that enables learning and collaboration within the context of the natural world.

Geotagger was created with the intention of connecting children with nature. This thesis research has broadened the focus of Geotagger to allow users to better connect with each other, as well as the environment. This was accomplished by adding social enhancements to the existing Geotagger platform which allows users to more readily explore and collaborate with one another. Previously, the only way for users to interact with one another is by sharing comments on tags. The only way to view and respond to these comments was by viewing the comments directly within the *tag view*. This work involved creating new and interesting ways for users to engage with one-another within the Geotagger application itself. The goal is to create micro-communities within the Geotagger application via small-scale, individual interactions that users will have with one another. These interactions come about by way of the social enhancements that have been added to Geotagger. Making comments more visible in the *tag list view* allows users to quickly view and make comments at a glance. Aggregating available tags into a global map-view was originally proposed to give a novel and visual way for users to explore data which other adventure members have submitted. However, this feature remains unfinished, and exists primarily as a proof of concept. Additionally, a common chat-area for each Adventure has been implemented. This gives members of a particular adventure a dedicated space for conversation. These enhancements, have helped Geotagger to build a sense of community to motivate new users to contribute more regularly and achieve a common goal.

The Geotagger project consists of a variety of software technologies that provide an online, collaborative space for scientific environmental inquiry. User generated content, like tags, adventures, and comments are kept in a long term data store for further analysis and retrieval. This creates a remote data repository, making Geotagger data available to a multitude of users through a variety of in-house developed mobile and web client applications. The REST API rounds out the suite of software, providing the connective tissue between local clients and the remote data store.

1.3 Thesis Statement

This thesis work aims to bring community members together in order to cultivate user interest and engagement in citizen science. This was realized by creating and evaluating social enhancements to the Geotagger project. Making tag comments a more integral part of the Geotagger user's workflow allows users to more readily take part in tag-related discussion. User interface improvements make it easier to learn more about community members and the content they contribute to the Geotagger project. The addition of an adventure chat screen, which provides a space within the application to allow users to engage with one another and share their ideas. Finally, creating geographical groupings for tag data grants users a novel means of visualizing tags authored by the community, and how these tags relate to one another. These enhancements facilitate social interactions and allow users to engage with one another in meaningful ways.

CHAPTER 2

RELATED WORK

In order to situate the reader as to the relevance of this work, this chapter will focus on the surveyal of previous work in the field of citizen science, as well as the Geotagger project. Firstly, the reader will become familiar with the field of citizen science and the relevancy of this work. Immediately following is a sample of several contemporary citizen science projects in order to give broader understanding as to the practical state of the field. Lastly, the user will be acquainted with various components of the Geotagger ecosystem, and where this work fits in amongst them.

2.1 Overview of Related Citizen Science Projects

In the past, citizen science projects have utilized different technologies to facilitate data collection and recording. Mobile applications have become increasingly attractive as the presence of smartphones and other "smart devices" have become extremely prevalent [23]. In fact, smart devices have become so common that it is an oddity to find an individual without one. Smart devices have been shown to provide many of the same features and capabilities of more specialized devices which have been traditionally used in the field. This research introduces a framework for developing mobile-based citizen science applications with a focus on user-authored data collection. Teacher highlights the advantages of using mobile applications for data collection and inquiry [37]. The paper notes that smart-devices are becoming increasingly ubiquitous. High-resolution displays, professional quality cameras, and Internet connectivity have led to smartphones becoming an attractive alternative to dedicated recording equipment. As smartphones mature, traditional recording hardware is on the brink of obsolescence. Smartphones are increasingly capable of accurately and effectively recording and uploading audio, visual, and place-based observations. Teacher and Kim propose ways to design, develop and integrate mobile applications into citizen science projects, in order to leverage the advantages afforded by smart devices. However, they do not offer advice on how these applications can develop and nurture a growing community around the project.

The Creek Watch tool is an example of a citizen science project which is bolstered by the presence of both online and mobile tools. Like Geotagger, Creek Watch facilitates data recording and discovery. Kim [24] suggests that citizen science projects can only succeed if the data they collect is actually used. The application provides features for viewing the findings of other users of the project. The Creek Watch tool represents a significant resource for local projects, for example, water and trash management programs can utilize the data provided by the application in day-to-day activities. Boston describes an online tool to be used in the collection and dissemination of volunteer recorded observations [7]. The focus of this project is Sligo Creek and the wildlife that live along its bank. Participants were especially interested in the community involvement aspect of the project. Volunteers were able to share their discoveries with fellow community members, which led to an added layer of discovery and inquiry. This project noted the positive effects that direct social interactions had on contributors, leading them to answer questions, comment on other observations, and further provide a source of discussion and enthusiasm for the project. Sullivan proposed the eBird Project [35]. This project uses a tool that allows users to record bird sightings. The tool leverages a distributed citizen science user base in order to facilitate the observation and collection of a wide variety and high volume of data. These observations are archived and made available to all users of the service, thereby facilitating inquiry and discovery. While many of these tools are able to successfully utilize citizen scientists for data collection, none of these projects made use of the full power of social and community interaction. Rather, these projects are examples of user bases which are largely isolated from one another. Geotagger seeks to cultivate real, meaningful, and organic interactions between community members. These interactions help to cultivate excitement for the project, as well as give participants a sense of purpose.

The lifeblood of any citizen science project are the citizens themselves. Citizen sciences greatest strength is the ability to collect large amounts of data almost effortlessly via a dedicated volunteer-base. These participant bases are not built overnight and require a significant amount of thought and planning. Citizen scientists often perform scientific activities on their own time and are not usually compensated monetarily. Much research has been done on how citizen science projects can motivate and engage user participation. In her dissertation, Curtis performed a case study in which three separate citizen science projects were investigated concerning how they sought to engage new volunteers and motivate existing volunteers [8]. This study found that direct interaction with project researchers and scientists greatly reinforces the resolve that participants have concerning the project. Being able to see and directly interact with authority figures in the project greatly boosted morale and gave the participants a sense of direction and importance. Creating an environment in which users feel like they are learning and making a difference has been a big part of engaging participants. Fischer looks at the advantages afforded by creating "cultures of participation" in which learning and contribution are actively encouraged by both participants and supervisors [16]. Project volunteers directly interact with their peers as well as project coordinators, in an environment which encourages collaboration, learning, inquiry, and discovery. Many "cultures of participation" or "affinity spaces" can be observed in online culture. Gee discusses various "affinity spaces" which can be found surrounding computer games. He studied how the users interact with one another and discuss objectives of the game [19]. Users provided helpful advice for difficult sections of the game, and challenged one another to play the game in new and interesting ways. Community members contribute and interact with one another on a voluntary basis. These communities have been shown to consist of thousands of participants which make contributions over several years. This is a very desirable trait that citizen science projects should seek to emulate. Raddick's work also covered the formation and further motivation of a prospective user base [31]. This paper attempts to discover why citizen scientists are essentially spending their free time contributing to a particular scientific pursuit. Unfortunately, this study did not draw any concrete conclusions, rather it posits possible answers to the question. More often than not, users contributed to the project because they were simply interested in it or they thought the activities involved in the project were "fun".

As with all citizen science projects, the overall health of the project relies almost exclusively on the health of its user base. Motivating community contribution is extremely important to develop a healthy community. Similarly important is being able to motivate community interaction. These interactions run parallel to the overall goals of the project, while not strictly contributing to the overall goal. Rather, these community interactions are about the project and can help users learn and stimulate enthusiasm about the project. Aristeidou models the growth of a newly created citizen science project [5]. This project sought to stimulate community interaction through the nQuire site, as well as launching its own social media campaigns. However, project researchers noted a distinct subset of users did not feel a sense of community due to the lack of a centralized goal. Tinati avoids this problem by incorporating community interaction into the very foundation of the project [38]. This paper studies a citizen science project know as EyeWire. The EyeWire project maps the structure of neurons in the brain. Users are presented with a game screen and must select portions of a cross-section which they believe to be a part of the current neuron string. Users are furnished with a real-time chat that allows them to interact with the EyeWire community. Mugar also describes different types of community interaction in his paper [28]. Mugar enumerates various ways of learning. The most effective form involves direct interaction with the project community at-large through forum posts, social media, or direct messages to users. Newer users of the community are able to learn from more experienced users, while more experienced users are able to discuss more difficult problems with their peers. This approach leads to a more tight-knit community which is more deeply invested in the goal of project and the community built around it.

This paper is not a comprehensive survey of citizen science, however, these projects have particularly interesting goals and act as examples of contemporary citizen science projects. Researching citizen science projects provided the background and context necessary to proceed with this work as a contribution to citizen science. Going forward, this work researches the current state of the Geotagger project and its technologies. Backend technologies (e.g., data store and the REST API) were directly used by this work. Frontend technologies provided ideas and concepts that influenced this new work.

2.2 Previous Work on the Geotagger Project

The mobile application implemented in this work is not the first technology developed for the Geotagger Project. The Geotagger platform has been developed over the last five years and includes software components that work together to provide collaborative technologies for environmental inquiry. These technologies are designed to allow users to create and experience observations with others who share a similar interest in the environment and the world around them. The Geotagger project is comprised of a number of individual programming projects. Together they have developed Geotagger into an extensible and versatile cloud platform. The aim of these projects is to give users the tools to create and share their own Geotagger findings, as well as view the contributions of others. Geotagger is a remote application, therefore citizen scientists can access Geotagger data from a variety of devices. The ecosystem includes various clients that interact with the Geotagger platform to reach as many prospective users as possible. The Geotagger platform consists of: a backend database, a REST API, with user clients that include a web portal, an iOS mobile application, and an android mobile application.

The MySQL RDBMS was selected to provide long term data storage and relational table operations for this project. The underlying database schema was designed to provide a flexible and durable data storage strategy for use with the REST API and primarily JavaScript-based clients. The RESTful API provides a secure data access interface, that allows secure viewing and altering of user-generated Geotagger data. This interface can be accessed with a wide range of clients, giving users the ability to access Geotagger data from virtually anywhere in the world. In order to provide useful functionality, this interface must connect to a long term data store. Here, the system stores user-created data which can be retrieved and updated at a later time. A web portal was developed with the intent of offering users access to the Geotagger system from a standard web-browser. Due to the geographical nature of Geotagger, this is perhaps not an ideal interaction when compared with a mobile client. It does, however, allow for rapid browsing of others' contributions which may not be afforded on a mobile device. The Android mobile application provides a similar suite of functionality to the web portal. It also allows the Geotagger project to target a wider audience. The added mobility of the Android application allows users to create tags as they discover them. Finally, the iOS application was meant to extend the reach of the Geotagger project with Apple device users in mind.

A brief discussion of each component and its impact on the ecosystem as a whole will now follow. It will contextualize this work relative to previous Geotagger projects. This work represents a foray into this space. In the past, a number of projects served to provide key pieces of infrastructure which allowed for the development of future Geotagger applications which can take advantage of this infrastructure, this work in particular.

2.2.1 Backend Data Store

As with all citizen science projects, the core of the project is the data collected by its members. Geotagger is no exception. The database is the “heart” of the Geotagger system, as it holds all of the pertinent data for the project. It is the very reason the project exists. In order to consolidate this data, a centralized, remote data store is required.

Riad Jeradeh [22] developed a backend data store using MySQL RDBMS. MySQL provides a relational SQL implementation useful for storing structured and related data. The Geotagger database follows the ACID principles (Atomicity Consistency Isolation Durability) regarding database transactions and storage. It ensures that stored data is consistent, durable, and non-volatile. It was created with the intention of implementing a flexible, yet durable, schema specification that would allow for storage of “diverse and large amounts of data”. In designing the database, Jeradah recognized that the schema needed to be both flexible and extensible [22].

The intent of the Geotagger project is to make contribution and data analysis available to members who utilize this data in a variety of ways. To create a flexible schema, Jeradah added several columns to existing tables to allow for user-driven extensions of the schema, without the need for additional tables [22]. Data privacy features like access levels and user roles give adventure coordinators tools to manage access to portions of data collected during an adventure. This function allows adventure coordinators to partition members into separate groups to accomplish different tasks. Furthermore, for the sake of posterity, records are never completely removed from the database; rather a “deleted_at” column is updated with a date and time representing when the row was removed. This column serves to exclude “deleted” rows from further querying and allows for continued storage.

The Geotagger project is meant to evolve and grow with the needs of its users. This necessitates an extensible data store which can incorporate these changes. Custom tag attributes allow for users to append extra data to the tag entity, so users can share interesting data about their tag. Following this project, Geotagger was afforded a versatile data store. As Geotagger is a collaborative project, it would not be appropriate to simply store user-generated data in an inaccessible database. So arises the need for an intuitive and comprehensive interface by which users may access this data. A REST API was chosen to provide this functionality for the Geotagger project.

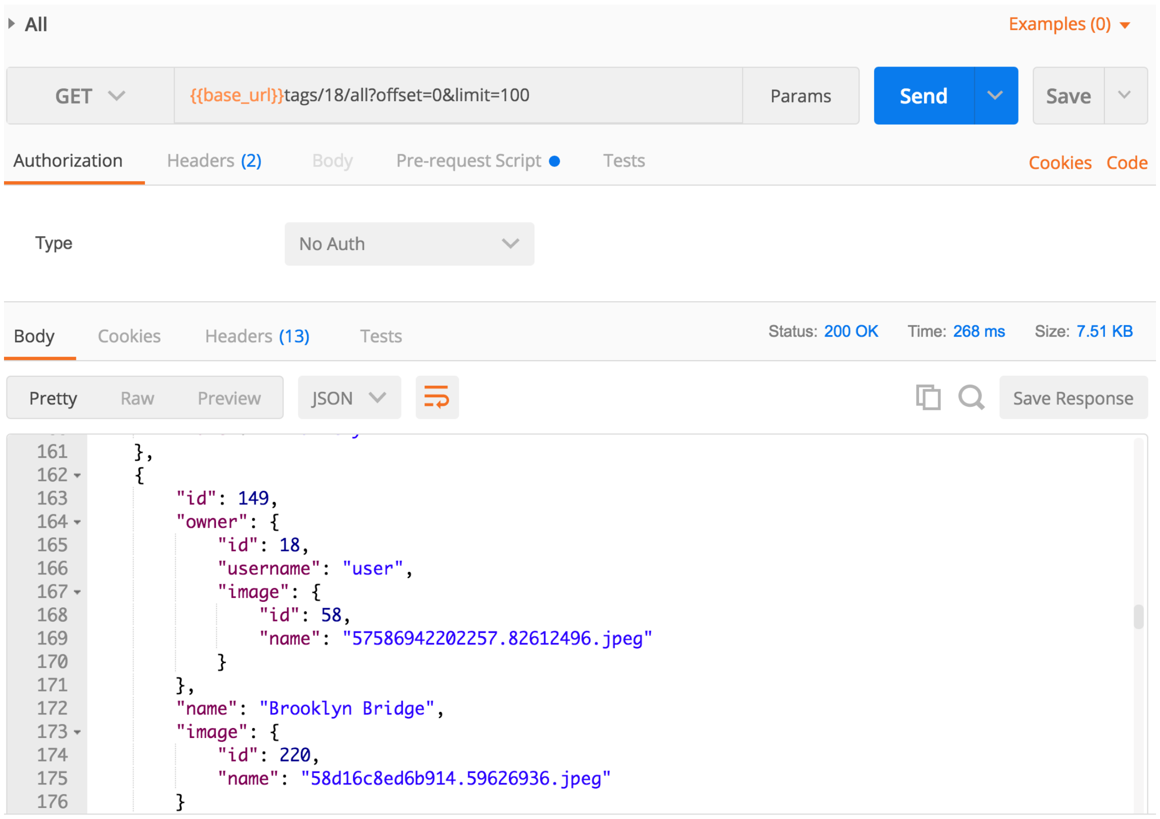
The aim of this project was to provide a robust, extensible schema and implementation which could provide data storage according to the needs of the Geotagger project. With a working store, the Geotagger project was now in need of a means of accessing this data. In order to retrieve Geotagger data, an API was required that could be utilized by a variety of frontend clients, allowing Geotagger to engage a wide range of users. The database and the API would work hand-in-hand to serve relevant Geotagger data to interested clients while providing a much needed "collaboration" element to the project.

2.2.2 REST API

A REST API allows for secure database querying and data transmission. With a large number of clients wanting to access the data, it is not feasible to grant every client direct access to the database system. Rather than allowing unfettered database access to these anonymous clients, REST APIs grant authorized access to a subset of the available data. The Geotagger REST API was developed by Andrew De Stefano [10] and provides a unified data endpoint which can be used by present and future Geotagger clients. All CRUD operations performed on Geotagger data is sent through this interface, granting well-formed and legal requests and defending against malformed and malicious requests. Authentication is handled with OAuth 2.0 tokens, which must be sent with every information-sensitive API request. To retrieve the data proper, Geotagger's REST API works hand-in-hand with the MySQL database. The HTTP data transmission protocol is used to make API requests, which supports *GET*, *POST*, *PUT*, and *DELETE* methods. HTTP requests are deserialized into SQL queries using an object relational mapper (ORM). The ORM returns a well-formatted JSON response to the API endpoint, which in turn sends the data to the client. API routes follow a semantic naming scheme that produces straightforward and meaningful data interactions. Routes are authorized on a case-by-case basis, thus disallowing access to data to which the user does not have rights. Finally, API responses are returned in the JSON data format. Given the popularity and ubiquity of JSON, this approach allows for a very diverse set of clients operating on a wide variety of platforms.

This project was accomplished using the Symfony framework. Symfony is a server-side MVC framework which ties URL routes to logic controllers. Procedures can be defined to run each time a client requests a certain route. The ORM-based data connector ensures that the API is oblivious of the store implementation and allows developers to easily change the database vendor at a later date. Furthermore, this project removes coupling between frontend clients and the data interface, thereby allowing any kind of client that implements the HTTP standard. This is further reinforced by JSON responses, a common data format which can be easily adapted to a wide variety of clients.

A REST API defines a number of parameterized URL "routes" which allow for explicit and semantic access to data. A URL with the form `domain.com/user/19` could be used to request access to pertinent data regarding a user with the ID: "19". Since REST APIs commonly use HTTP as a transfer protocol, HTTP methods are leveraged to grant even more meaning to API routes. For example, the request: `DELETE domain.com/user/19` deletes a user and would give a much different result than the request: `POST domain.com/user/19` which might creates a user. Figure 2.1 depicts an example of a `GET` method call and its response.



{{base_url}}tags/18/all?offset=0&limit=100

Type: No Auth

Status: 200 OK Time: 268 ms Size: 7.51 KB

```
    },
    {
        "id": 149,
        "owner": {
            "id": 18,
            "username": "user",
            "image": {
                "id": 58,
                "name": "57586942202257.82612496.jpeg"
            }
        },
        "name": "Brooklyn Bridge",
        "image": {
            "id": 220,
            "name": "58d16c8ed6b914.59626936.jpeg"
        }
```

Figure 2.1: REST API Request/Response

The Geotagger REST API allows users to create, view, edit and remove Geotagger content (e.g., tags, comments, adventures) while remaining ignorant of client architecture or platform. The REST API enables user authentication and authorization. Authentication is required to ensure that any client making a request for Geotagger data has created an account and is registered within the Geotagger system. Authorization allows the project designer to hide certain details from users, while exposing other details. Finally, the REST API is responsible for serializing and deserializing data through the use of an ORM. Whenever a user creates an object through the API, the API must serialize the data into a model object format which the API recognizes. Once this is done, the object is given to the ORM which uses the structure and class of the object to determine the database table and columns where the data should be placed. Similarly, when the user requests data from the API, the ORM must retrieve the data from the database. It then converts the retrieved data it into a model object and serializes it into JSON format. This JSON data is finally attached to an HTTP response. Geotagger is now equipped with a data storage mechanism, as well as a means of accessing this data. Further Geotagger projects focused on providing users with engaging tools through which they could contribute to and interact with the Geotagger system.

2.2.3 Web Portal

Web portals represent an classic user interface which provides users with an abstracted view of the data and functionalities available to them. In designing a web portal, care must be taken in giving users interface elements and calls to action which are intuitive and logical. A major strength of websites is that they allow users to interface with a more powerful server without additional software. Alicina Mumar [29] developed a web front-end, providing Geotagger users an intuitive way of interacting with the REST API. This online presence acts as an informative brochure for the uninitiated, and a system interface for members of the project. Users can log into their Geotagger account granting access to the Geotagger ecosystem. This interface allows for adventure, tag, and comment creation. The web front-end is compatible with all modern browsers. The web front-end simply makes calls to the REST API allowing for fast and secure data transfer. This client gives users the opportunity for rich interactions using the Geotagger system without the need to download additional software. Web pages provide a low barrier to entry for users and allow for quick and easy access to a web application. The web portal provides an intuitive and comprehensive interface for the Geotagger system, as the portal currently implements a large amount of the Geotagger system's available functionality.

This project uses the JQuery and Bootstrap libraries. JQuery is a popular JavaScript library that provides a number of features for controlling the look and feel of web applications. JQuery was instrumental in the user interface design as this project made extensive use of JQuery interface elements and the animation API. Interface colors denote different users and privileges. Scientists and citizens will see different shades of blue, while administrators will see a black color scheme. JQuery also has an API for changing the user interface dynamically, like UI element animation and dynamic color pallets. AJAX requests enable loading of new content without the need to refresh the page. Bootstrap's primary feature is a CSS grid-system that allows for precise placement of screen elements. Bootstrap is also used in creating "responsive" interfaces, this allows clients to use the web application on almost any screen size as Bootstrap will resize view elements automatically. Finally, SASS was used to provide a modular, extensible stylesheet for customizing the look and feel of the application. Figure 2.2 depicts the web portal displaying a user's tag view.

Although the web portal is a fully-featured Geotagger client, it does not provide the inherent convenience found in mobile applications. Laptops and desktop computers are inconvenient for field-related project use due to the device size. These devices also lack a high-quality front-facing camera for capturing environmental observations. Mobile web-portal clients are indeed more flexible, but access to native device features like GPS and the camera can prove difficult to implement. Though the flexibility and mobility of smartphone applications lend themselves well to the field-intensive nature of the Geotagger project.

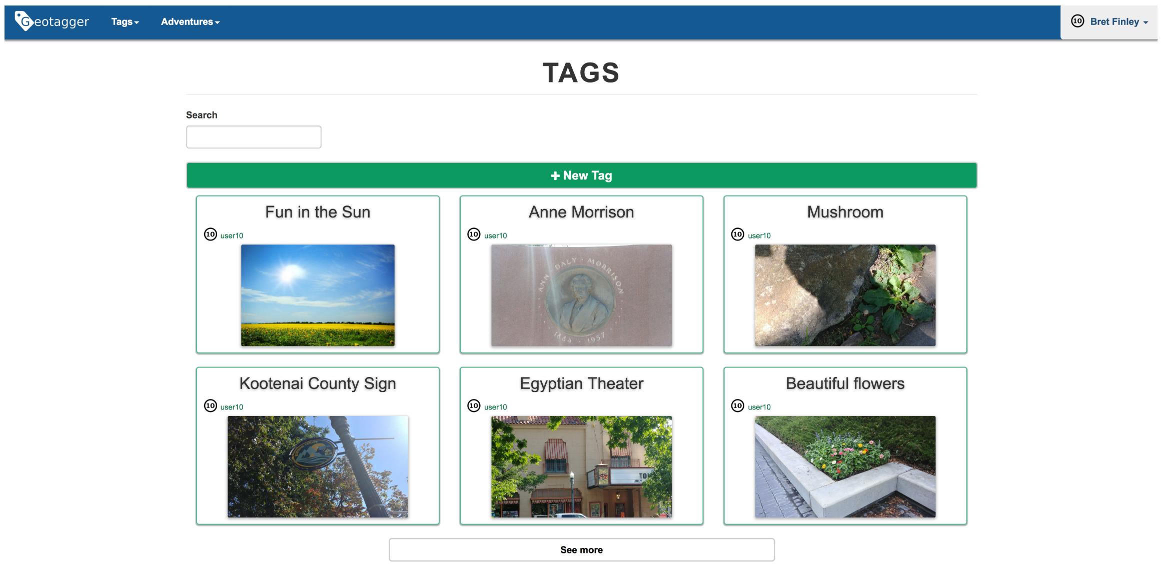


Figure 2.2: Geotagger Web Interface

2.2.4 Android Application

The Android platform is the most popular mobile device operating system. Therefore, the Geotagger project made it a priority to build an Android application to reach this growing userbase. The Android application provides a frontend environment for querying and updating the REST API. Due to the popularity and flexibility of Android, the Geotagger project wished to provide a user friendly, intuitive experience to assist the capture of environmental data. Paul Cushman [9] created the Android application, with a focus on offline functionality and local data caching. Because of the Geotagger project's subject matter, a mobile application is a fitting client; as users move around the environment, capturing any features they find to be interesting. Most modern smart-devices are equipped with hardware necessary to collect project data. Users can directly contribute to the project from their device. Figure 2.3 depicts a tag screen displaying some environmental data. The largest difference between the

Android application and the web frontend is the Android application's offline-usage functionality that is realized in the local caching scheme.

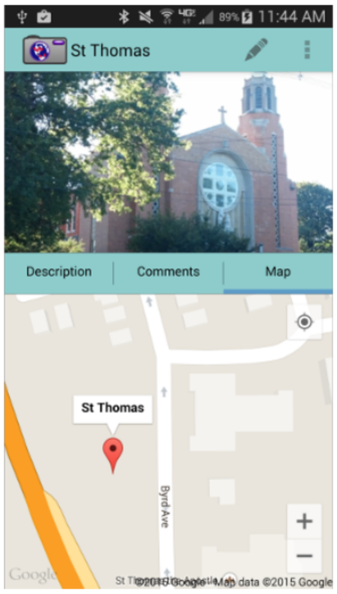


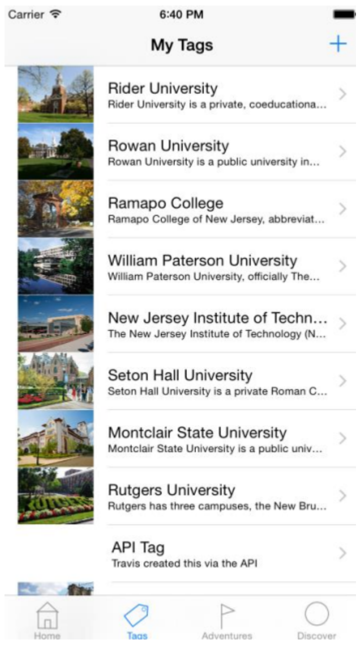
Figure 2.3: Geotagger Android Application

The action cache utilizes SQLite, a lightweight, local relational database system intended for client-side or local application use. SQLite allows for storage and retrieval of records without the need for an Internet connection. Research teams and other scientific endeavors are not always guaranteed to have a strong network connection in the field. It is with this in mind that the local cache was designed. While the REST API requires an Internet connection to be used, the Android application caches records retrieved from the REST API while the device is connected to the network. In this way, when the device loses connection, it can still provide functionality to the user (albeit a subset). When network connection is re-established, the application will synchronize with the REST API and update the local SQLite cache in the process. Furthermore, the caching scheme allows for "action caching". That is, the user is still able to make changes to cached data while offline. Actions for creating data, updating data, and removing data are cached locally in queue-like fashion. Once network connection is established, actions are serviced in order of arrival. The results of these actions are then propagated throughout the local cache, thus ensuring data consistency with the remote data store.

At the close of the project, the Android application implemented a large amount of Geotagger functionality. While this application has room for improvement, it interfaced well with the second iteration of the Geotagger API and provides a novel means of interacting with the Geotagger system. The work contained in this thesis built off of the Android application by leveraging several features and design paradigms found in this project.

2.2.5 iOS Application

While not as prevalent as the Android operating system, Apple devices are nonetheless a very popular mobile platform. In terms of scope and intent, the iOS application project is similar to the Android application. Developing a Geotagger client for iOS could only serve to increase the potential userbase for the project. Travis Gantt [18] developed a Geotagger iOS application. An effort was made to incorporate as many features of the Geotagger project into the application as was feasible during the project's time frame. However, due to the increased difficulty of developing for the platform, the iOS application is perhaps the least feature-complete Geotagger client. While the application lacked several key elements at the time of the project's completion, nonetheless, the project serves as a proof-of-concept for a Geotagger application in the iOS space, and has laid the groundwork, should development continue at a later date.



Rider University
Rider University is a private, coeducationa...

Rowan University
Rowan University is a public university in...

Ramapo College
Ramapo College of New Jersey, abbreviat...

William Paterson University
William Paterson University, officially The...

New Jersey Institute of Techn...
The New Jersey Institute of Technology (N...

Seton Hall University
Seton Hall University is a private Roman C...

Montclair State University
Montclair State University is a public univ...

Rutgers University
Rutgers has three campuses, the New Bru...

API Tag
Travis created this via the API

Figure 2.4: Geotagger iOS Application

Difficulties experienced by this project were due to the "greenfield" nature of the project, with little code brought over from the existing Android application. The difference in platforms meant that little code could be shared between the two projects. Additionally, Apple requires substantially more documentation and scrutiny when it comes to developing iOS applications, which gives iOS developers a much higher barrier to entry than that of the Android development environment.

This thesis work builds on the Geotagger iOS application by providing a cross-platform codebase which may be deployed on a variety of platforms to instantly afford the Geotagger project much greater reach than a single native application. This work updates features found in both mobile applications and is responsible for adding a suite of social enhancements targeted at engaging and motivating citizen scientists.

CHAPTER 3

DESCRIPTION OF WORK

The work implemented in this thesis builds on previous work; it improves, updates, and enhances the Geotagger ecosystem. This chapter presents the influence of previous Geotagger software projects on this work and how this work differentiates itself from past efforts. This work was implemented using web development technologies. An effort was made to become acquainted with contemporary web technologies used in developing modern web applications. This chapter will feature a discussion that will cover the research process that led to the Ionic framework being selected. The Ionic framework is a cross-platform library that allows a single codebase to be deployed across multiple platforms. This chapter will also give an overview of additional software libraries that were used and their significance to the project. Finally, this chapter will conclude with a visual walkthrough of both versions of the application implemented in this work. The first version was intended to instrument existing Geotagger functionality. The second version builds on it by extending the application with social enhancements discussed previously. Special attention will be given to those aspects which serve to realize the proposed social enhancements of this work. These enhancements have been enumerated and codified below. The remainder of the chapter will use these codes when referring to a specific social enhancement.

- **SE1** - Making comments more prevalent

- **SE2** - Connecting people through their contributions
- **SE3** - Creating a shared social space for adventures
- **SE4** - Geographic overview of tags

3.1 Influence of Previous Work

This work takes inspiration from previous Geotagger projects. The most notable inspirations originated in the Android application [9] and the MySQL data store implementation [22]. The Android application provides the most obvious inspirations, since that project and this work fill similar roles. The Android application made many considerations based on the specificity of its platform. This work is concerned with a broader range of devices. The MySQL database's impact is not to be discounted. The schema found in that project largely served as the blueprint by which the local SQLite cache was built. Both projects served to provide critical insight into the design and implementation of this application.

3.1.1 Android Application

The Geotagger Android application is the most fully-featured Geotagger client. It was here that researchers first looked for design inspiration, specifically regarding application data flow. The data flow structure determines how non-view portions of the application are structured and how data is made available to the rest of the application. At the source code level, both applications use very different methods for designing and structuring the user interface and handling user-generated events. Inspirations were taken from the user interface itself, but the ways that various application frameworks express these views are very different. Finally, the Android application provided the leading influence in the design of the action cache.

To overcome the limitations of a single platform code-base, a cross-platform mobile application framework was chosen. The *Ionic Framework* contains tools to develop and deploy a mobile application to a variety of mobile platforms. Ionic leverages web technologies to create a web application development environment. The resulting application is executable by the target platform. Research was conducted as to which of the existing cross-platform frameworks would best suit this project. This research is discussed in section 3.2. Additionally, other software used in the development of this application can be found in section 3.2.

Application Data Flow

One of the most important aspects of this application is the way data is shared between various modules. This flow would largely determine how the application's concerns were separated and how they communicate with each other. It is possible to include the entire application in two separate files (for the view markup and the programming logic). However, this architecture would be unwieldy and unmaintainable. Thus, the method in which data flowed through the Android application was analyzed. Data flows from the user creating local data to a permanent store in the cache or the API. Conversely, data may flow from the permanent store to be displayed in the view. In the Android application, if the local model data changes, then view activities (the controller in MVC) call the "Application" class to dispatch handler functions based on the calling record's type (e.g., tag, comment, adventure, etc.). These handler functions are called "connectors". Connectors provide an interface between the application's business logic and data stores. They include the local

SQLite cache and the remote API. Beyond this, cache and API handlers interact with the stores directly. Connectors make specialized requests that format and serialize the data in preparation for transmission to either the cache or the API. The API and cache-specific handlers appear as mirror images of each other, as both are performing very similar actions on different storage locations. Figure 3.1 shows the data flow in the Android application.

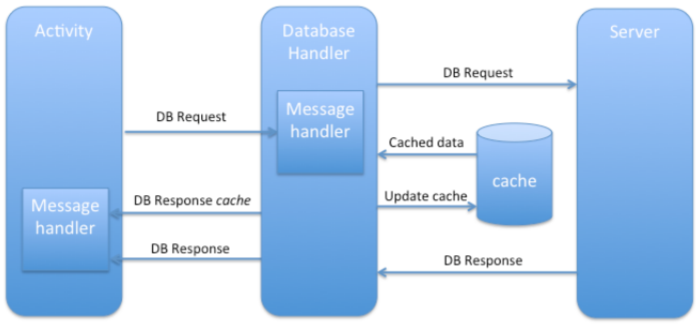


Figure 3.1: Android Data Flow

The cross-platform application is broken up into a similar hierarchy of modules. Controllers, provide logical hooks which can be referenced in the user interface. Controllers also make data available to the user interface. If a controller needs to make a data request such as reading or modifying data, then it does so with the “data-handler” class of modules. The data handler provides an interface between controllers (which are closely coupled to the views) and the storage locations. Depending on whether the data needs to be transmitted or requested from the cache or API, the handler will make the appropriate call to corresponding module. Cache and API modules operate very similarly. However, the cache contains an instance of the “CacheDatabase” service, while the API module makes HTTP requests to the server directly. One major difference between the two applications is that the Android application defines distinct routines within the connectors for every type of model object. In designing the caching mechanism, a more generalized approach was chosen that would work with any model object. Model objects only require an analogous cache table. In this application, object structure determines the structure of data within the cache itself. In this way, data is simply objects that are deconstructed into their properties, and placed in corresponding columns. Figure 3.2 depicts the flow of data through the new Ionic application.

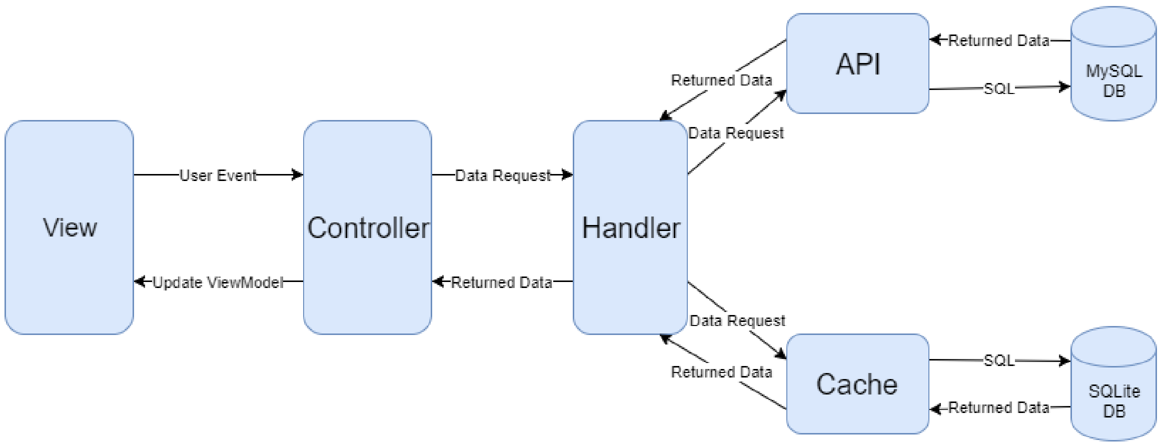


Figure 3.2: Cross-Platform Data Flow

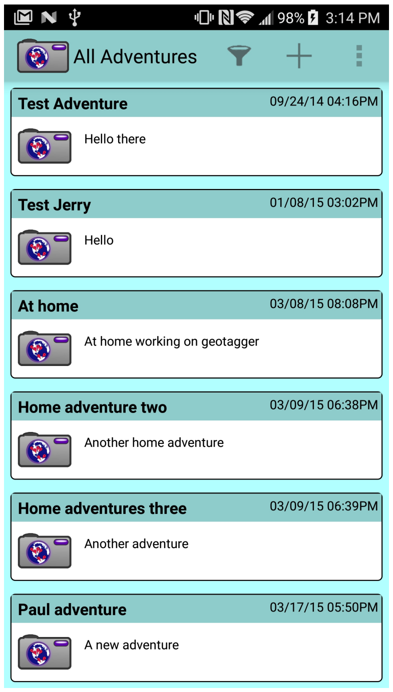
User Interface Elements

The Android application provided many interesting insights as to what the application's user interface needed to accomplish. Having implemented a considerable amount of the functionality of the Geotagger system, there were many views that could be considered. The most notable inspirations were those found in the views for: adventure list, tag list, tag detail, and the comment list. The tag and adventure list views are very similar in nature. The list is easy to read as the user scans their eyes down the page. Both lists contain "cards", which are distinct, visually self-contained objects, that usually have a square or rectangle-like shape. These cards display pertinent information regarding their entity: name, time of creation, time of last modification, and brief description of the entity. It helps to give a brief overview of the data, so the user can decide if they would like to learn more by clicking or pressing on the card. A comparison of these lists can be seen in figure 3.3. The tag detail view is decidedly more complex and provides the user with information specific to the selected tag.

The adventure list view within the Ionic application remains very similar to that of the Android application. Adventure "cards" contain similar information, including: the adventure's title, the time at which it was created (in a relative format, contrasting with the Android application's absolute format), and the description of the adventure. The tag list has undergone several changes. It was the subject of some of the largest modifications when moving between the two versions of the application. The first version of the application is more similar than the second version, so it will be discussed here.

The tag list is another example of a similar user interface view between the two applications. Both applications use lists to display the tag items. However, to produce action buttons, the Android application requires the user to long-press the list item. The analogous action in the Ionic application requires a left-swipe to slide the list item, thus revealing the buttons. Both items show tag name, description, and creation date. The Ionic application uniquely displays the location description. These similarities can be seen in figure 3.4.

Both versions of the application display comments in a similar list view. Comments contain: name of the authoring user, comment text, and time the comment was created. The Ionic application makes two improvements. User profile images are



All Adventures

Test Adventure 09/24/14 04:16PM
Hello there

Test Jerry 01/08/15 03:02PM
Hello

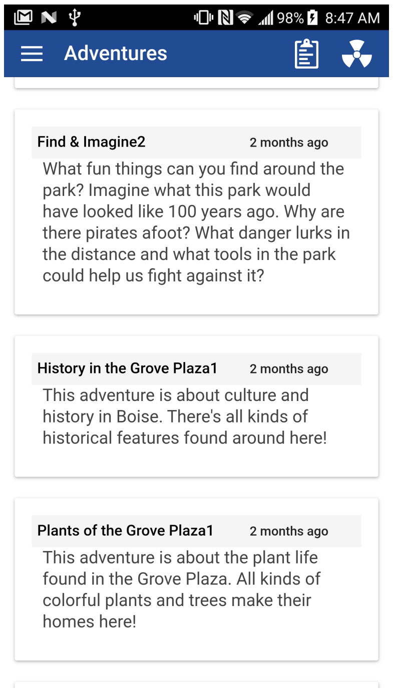
At home 03/08/15 08:08PM
At home working on geotagger

Home adventure two 03/09/15 06:38PM
Another home adventure

Home adventures three 03/09/15 06:39PM
Another adventure

Paul adventure 03/17/15 05:50PM
A new adventure

(a) Android



Adventures

Find & Imagine2 2 months ago
What fun things can you find around the park? Imagine what this park would have looked like 100 years ago. Why are there pirates afoot? What danger lurks in the distance and what tools in the park could help us fight against it?

History in the Grove Plaza1 2 months ago
This adventure is about culture and history in Boise. There's all kinds of historical features found around here!

Plants of the Grove Plaza1 2 months ago
This adventure is about the plant life found in the Grove Plaza. All kinds of colorful plants and trees make their homes here!

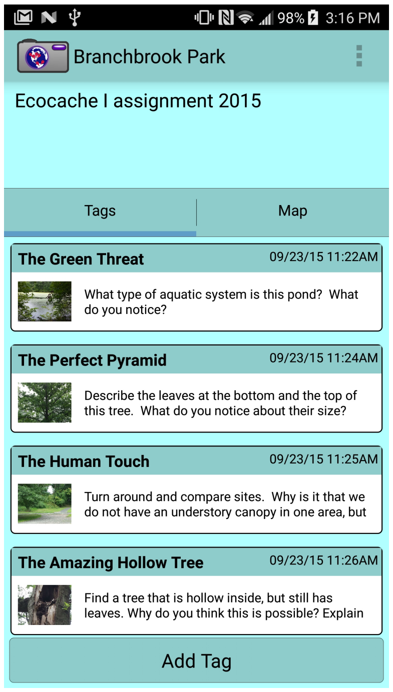
(b) Ionic

Figure 3.3: Adventure List Views

placed to the left of the comment and give users a visual identifier that allows them to easily associate users with comments. Additionally, comment images are much more prominent in the Ionic application. The Android application displays images in a small rectangular box on the side of the image, while the Ionic application utilizes more screen area to display a larger preview of the image. This preview can be pressed in order to bring up a full-size, zoom-able version of the image. Figure 3.5 highlights this comparison.

Action Cache Design

The action cache provides facilities for caching a queue of data modification "actions" which are resolved when the device is connected to the Internet. The Android applica-


98% 3:16 PM

Branchbrook Park

Ecocache I assignment 2015

Tags | Map

The Green Threat 09/23/15 11:22AM
What type of aquatic system is this pond? What do you notice?

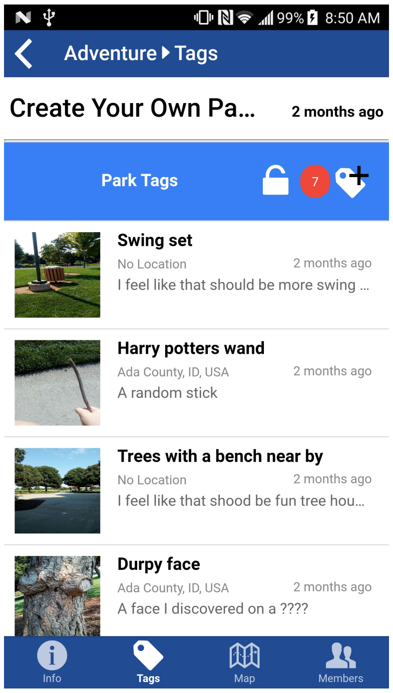
The Perfect Pyramid 09/23/15 11:24AM
Describe the leaves at the bottom and the top of this tree. What do you notice about their size?

The Human Touch 09/23/15 11:25AM
Turn around and compare sites. Why is it that we do not have an understory canopy in one area, but

The Amazing Hollow Tree 09/23/15 11:26AM
Find a tree that is hollow inside, but still has leaves. Why do you think this is possible? Explain

Add Tag

(a) Android


99% 8:50 AM

Adventure ▸ Tags

Create Your Own Pa... 2 months ago

Park Tags 7

Swing set
No Location 2 months ago
I feel like that should be more swing ...

Harry potters wand
Ada County, ID, USA 2 months ago
A random stick

Trees with a bench near by
No Location 2 months ago
I feel like that shood be fun tree hou...

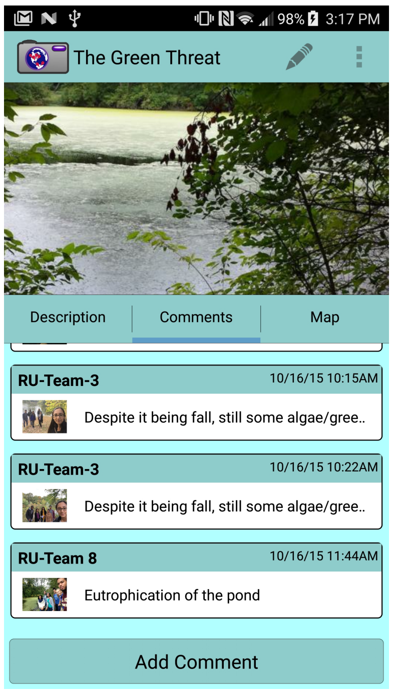
Durpy face
Ada County, ID, USA 2 months ago
A face I discovered on a ????

Info | Tags | Map | Members

(b) Ionic

Figure 3.4: Tag List Views

tion accomplishes this by defining a SQLite table that is responsible for maintaining an ordered list of unresolved actions. These actions are ordered by arrival time, where the earliest arrivals will be resolved first. New Geotagger objects are immediately stored in the local cache for further retrieval. These objects are given a negative ID in order to differentiate them from objects which have been created with the API. Once a record is ready to be resolved to the API, the resolver uses the ID stored in the action table to directly address the corresponding record. This record holds all of the pertinent data that must be transmitted to the API. Once this data is retrieved, a specialized data handler is responsible for affecting the data remotely. Each model entity has its own data handler. Once the data has been sent in the form of an API request, the API will respond with an updated version of the record that is

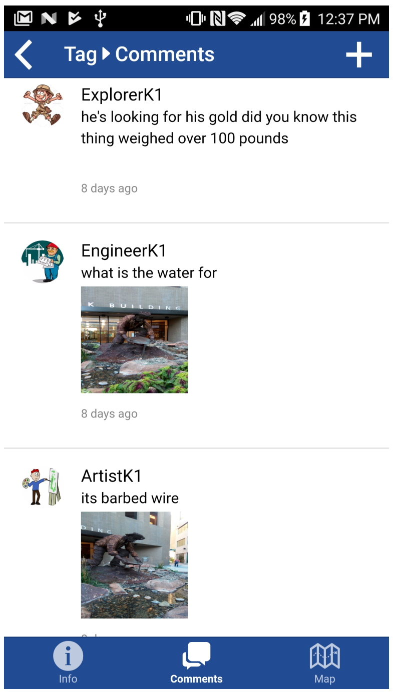


RU-Team-3 10/16/15 10:15AM
Despite it being fall, still some algae/gree..

RU-Team-3 10/16/15 10:22AM
Despite it being fall, still some algae/gree..

RU-Team 8 10/16/15 11:44AM
Eutrophication of the pond

(a) Android



ExplorerK1
he's looking for his gold did you know this thing weighed over 100 pounds
8 days ago

EngineerK1
what is the water for
8 days ago

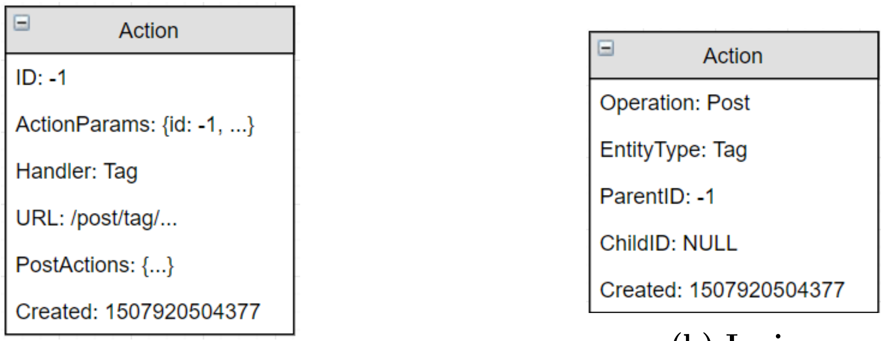
ArtistK1
its barbed wire

(b) Ionic

Figure 3.5: Comment Views

inserted into the local cache. The action resolver will read the “PostOperation” field to determine a series of further actions that must be taken to ensure data consistency throughout the cache. Finally, the action cache itself must be updated to reflect this change, as there may be actions within the action table that depend upon the new data.

The Ionic action cache improves on many of the ideas presented in the original action cache, yet preserves the overall algorithm. Like the Android application, the Ionic version utilizes a SQLite table to queue modification actions. The action table maintains an ID reference to the corresponding data. Rather than storing an array of “PostOperations”, this project favored a more normalized approach did not allow composite data to be stored. Rather than create individual handler objects for each type of model object, this approach created a series of callback functions that relied on the “EntityType” field to specialize operations. These operations act much in the same way as “PostOperations”, since they are responsible for calling functions on either the cache or API services. Finally, “ParentID” and “ChildID” fields were added to accommodate junction tables, which were not present in the previous version of the Geotagger API. Unlike other tables, these do not contain a unique ID field. To circumvent this shortcoming, the entire junction record is stored in the action table. For example, the AdventureTag object connects a tag to an adventure collection. In this case the “ParentID” refers to the parent of the relationship: the collection. The “ChildID” refers to the child of the relationship: the tag. Figure 3.6 provides a comparison of the action table schema.



Action
ID: -1
ActionParams: {id: -1, ...}
Handler: Tag
URL: /post/tag/...
PostActions: {...}
Created: 1507920504377

Action
Operation: Post
EntityType: Tag
ParentID: -1
ChildID: NULL
Created: 1507920504377

(a) Android **(b) Ionic**

Figure 3.6: Action Cache

3.1.2 MySQL Database

The primary inspiration taken from the MySQL database was its schema. The structure of the remote data store is often mirrored in the HTTP responses returned by the REST API. Because of this, model objects were designed to resemble the structure of the MySQL database tables. This provides the advantage of easily serializing and deserializing model objects. However, one serious disadvantage is that all model objects must be normalized. Normalized model objects cannot have object or array properties. This could have simplified processing data that contains child records. Rather than place children in an array property, child objects are placed into model objects of their own, using a referencing property (an integer that refers to the parent's own ID value) to associate parents and children. This design choice has allowed for the database to be ignorant of data types when performing cache operations. Model objects contain properties which correspond exactly to columns found in the analogous cache table (e.g., TagID). Model objects feature a `getTable` function which returns a string value representing the name of the corresponding cache table. The CacheDatabase service uses this information to make decisions about database queries (including select, insert, update, and delete) to determine the destination table and the data being inserted. Hence, the model objects act as a "blueprint" which informs the cache on how each model object should be handled

3.2 Survey of Cross-Platform Frameworks

Creating this application required selecting a build environment that would allow this project to target a broad and potentially architecture-diverse user-base. Previous development efforts focused on single platforms. This platform restriction inevitably restricted the number of potential users as well. Therefore, a framework which would allow developers to quickly and easily target multiple platform environments was vital to the realization of the proposed social enhancements.

Because mobile application developers are accustomed to streamlined build environments, developing a cross-platform mobile application is an extremely common software use-case. As such, there is an abundance of available frameworks to facilitate such a task. And so, preceding any concrete code implementation, a survey of available technologies would be conducted in order to find a framework that would best suit the project's needs. Table 3.1 gives a selected overview of several frameworks surveyed during the course of research.

Web frameworks use existing technologies (HTML, CSS, and JS) to create a "mobile-like" web page that will provide a native mobile application user experience. These frameworks will oftentimes look to a third party library (e.g., Cordova) to leverage various native device features like: camera, GPS, or local file access. "Native" frameworks provide a lower level of abstraction concerning the hardware of the device. Some native frameworks also leverage web technologies. These technologies are used to create cross-platform user interfaces that are compiled into native device elements. However, in order to make native device calls, the developer must create software hooks in the device's native language. Web-based frameworks provide a uniform and simplified development experience. Web-based frameworks can achieve nearly total code reuse between platforms. Conversely, native-based frameworks are more performant relying on native interface elements and device calls. Native frameworks lack extra layers of abstraction needed by their web-based counterparts. This leads to extra engineering effort, as each target platform must be accounted for when accessing device features. Ultimately, each of these frameworks represents a significant developer effort, backed by communities of passionate users.

In selecting the appropriate framework there were many factors to consider. A platform was needed that was easy to use, well documented, and had good community support. An active developer community could be used as a resource when problems were encountered. Licensing fees were to be avoided, as there are very strong open

Table 3.1: Cross Platform Frameworks

Rank	Framework	Type	Price	Language	GitHub Stars
1	Ionic	Web	Free	HTML, CSS, JS	31102
2	Xamarin	Native	Free	C#, .NET	804
3	Supersonic	Web	Free	HTML, CSS, JS	153
4	Trigger.io	Native	$50/month	HTML, CSS, JS	N/A
5	Sencha	Web	$900/year	HTML, CSS, JS	N/A
6	Onsen UI	Web	Free	HTML, CSS, JS	5267
7	React Native	Native	Free	HTML, CSS, JS	52905
8	M Project	Web	Free	HTML, CSS, JS	778

source options. Both Sencha and Trigger.io require expensive licensing fees and were eliminated. OnsenUI has several very strong features, but is very similar to Ionic without the active community of Ionic. At the time of this project's creation, React Native was still a beta product and was not selected. Finally the M Project has little developer and community support and was not suitable to the needs of this project. In narrowing down the list, the top three frameworks provided these qualifications.

3.2.1 Selected Frameworks

In order to select the framework which would best fit this project's needs, a short trial was conducted. This involved hands-on development with all three prospective frameworks. This involved setting up each of the development environments and creating a "Hello World" application that allowed each framework to be evaluated in terms of setup difficulty, integration with other libraries, overall health of the development community, and give experience diagnosing and repairing any errors that arose.

Supersonic

Supersonic projects are “multi-page” applications, holding multiple screens in memory to allow for fast navigation. Most other web-based frameworks develop “single-page” applications. Single page application consist of a single HTML page which is injected with HTML “partials” whenever the user navigates to a new screen. This can create opportunities for elusive bugs as elements must not be left in the DOM when a new page is injected. Multi page applications also provide a greater level of modularization than single page applications, as page logic and elements are separated. Supersonic applications are written in HTML, CSS, and JS, which was the preferred software stack for this work.

Supersonic provided a number of admirable qualities, but its community was found to be not as active as that of Xamarin or Ionic. Examples were fairly difficult to find, leaving doubt as to whether problems encountered during the development process could be easily rectified. Additionally, several problems were encountered setting up and installing the development environment. These difficulties prevented the development of even a simple proof-of-concept application. Because of these difficulties, Supersonic was not selected for this work.

Xamarin

Xamarin is an extremely well known and mature framework developed by the Xamarin group. Xamarin’s intent is to provide cross-platform APIs using XAML for building uniform interfaces and C# based operating system hooks for integrating native device features. Xamarin features APIs for Android, iOS, and Windows phone platforms, seamless integration with a mature software toolchain (e.g., Visual Studio, Xamarin Android player), and a robust user community. Xamarin seeks to provide a native experience for application users with native interface elements and device calls. Developing an application with Xamarin would undoubtedly yield a more performant program than that of Ionic.

Although Xamarin presents a very robust and feature complete ecosystem of technologies, there were a number of reasons why it was not selected for development of this application. While versatile, the overall build system for Xamarin applications is very fragile. Several applications were built that tested camera and GPS functionality, and it was fond that slight, seemingly inconsequential changes to the application would break the build process entirely. Large changes would have to be built incrementally until the problem was found. And although Xamarin does have a large community, many of the best tutorials and resources are a part of Xamarin University, a premium tutorial website. Furthermore, Xamarin applications are built using XAML and C#. Development APIs are extensively documented, but team development experience consisted mostly of HTML, CSS, and JS. One major disadvantage of Xamarin is that user-provided examples are not easily accessed. In this way, most Xamarin examples live in their own GitHub repositories, which require building of their own. There are hundreds, if not thousands of Ionic examples hosted on Codepen, which can be viewed and run instantly from within the browser. In the end, the Ionic Framework includes a number of extremely valuable features, making it suitable for this project's needs.

Ionic

Ionic is a web-based cross-platform mobile application framework which leverages AngularJS and Cordova. AngularJS is used to manage separating concerns in order to modularize things like view definition, UI hooks, and data flow. Like Supersonic, Ionic makes use of well known web technologies like NPM, HTML, CSS, JS, and Gulp to build web applications which run locally on the mobile device. Due to Ionic's popularity, Ionic has an extremely active and helpful development community. This, coupled with the feature allowing Ionic applications to be built directly in the browser (provided that there are no device API calls), means that debugging errors is by far the simplest in Ionic. Previous experience with AngularJS and Ionic also supported the prospect of utilizing Ionic in this project. Furthermore, because Ionic leverages Cordova for device API calls, Ionic is able to build executable binary files for a wide variety of platforms, including iOS, Android, Windows phone, Blackberry, and Ubuntu. This allows mobile applications to be built for a large number of devices. Being able to target more devices broadens allows Geotagger to target and engage a much larger audience to grow the Geotagger community.

While Ionic is an attractive web framework, it is not without its caveats. At the time of development, Ionic was experiencing a major version re-write. The move from version 1.X to 2.X was very tumultuous, leaving virtually no backwards compatible code between the two versions. At the start of this project, version 2.X was still a beta product. Application stability is of great importance in this work, so the more mature version 1.x was chosen.

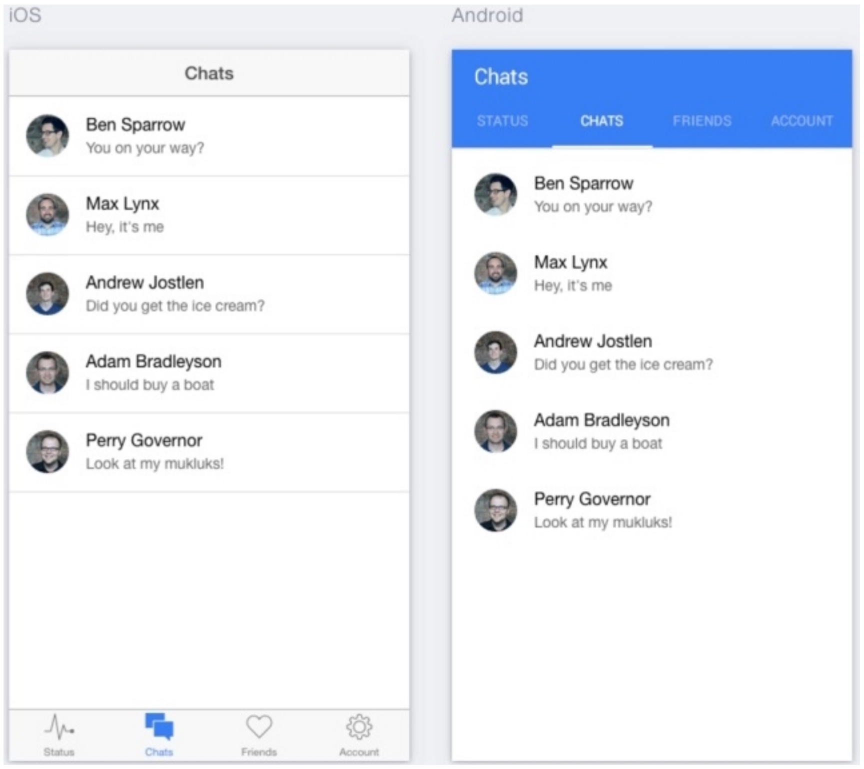
3.3 Overview of Employed Software Technologies

Due to the size and scope of the programming portion of this work, several third-party technologies were employed to abstract and provide support for many different features of this application. This section will discuss the main libraries that were used in developing the Geotagger cross-platform mobile application. While not every software library used during development will be covered, the reader should gain a high-level understanding of the technology used in the application to grant further context for the application's operation.

3.3.1 Ionic Framework and Cordova Runtime

Ionic is a cross-platform mobile application framework. While some cross-platform frameworks define their own MVC facilities for separating the various concerns of the application, Ionic utilizes AngularJS to provide data-binding between view elements and the main event loop. AngularJS creates programmatic hooks which can be referenced in the user interface markup code. Ionic provides a large number of AngularJS directives and services which can be used to develop a mobile-like user-experience and affords the developer the simplicity of web technologies. Ionic directives include pre-defined templates for standard mobile interface elements: cards, list views, modal dialog windows, and buttons. Ionic services define ways for programmatically interacting with view elements: dismissing a popup window from, scrolling the current view to the bottom, enabling zoom functionality for images, and overriding the default back button behavior. Web applications are rendered in a browser, this means that the application developer need not be aware of the hardware specifics of target platform. Because of this, the developer is able to deploy a single codebase to a variety of mobile platforms. AngularJS and Ionic account for a large number of mobile use-cases on their own, but they do not provide perhaps the most useful feature of a mobile application: the ability to access device hardware. Figure 3.7 gives an example of an Ionic application running on both iOS and Android platforms.

The Cordova project provides a JavaScript API for accessing commonly used



Ben Sparrow
You on your way?

Max Lynx
Hey, it's me

Andrew Jostlen
Did you get the ice cream?

Adam Bradleyson
I should buy a boat

Perry Governor
Look at my mukluks!

Figure 3.7: Ionic Application on Different Platforms

device features. In general, native device functions (image picker, camera, flash lens, etc.) are accessed through a device specific interface provided by the vendor. These interfaces are written in the native device language (e.g., Swift, Java, C#) and are distinct from one another in both implementation and structure. The Apache Cordova project provides a runtime environment which allows a single JavaScript API call to be deployed to several different mobile platforms. Cordova is also responsible for packaging the Ionic web application into an executable binary file that can run on the smart device. Once this file has been generated, developers can easily upload it to a popular "App Store" and distribute the application publicly. Cordova automatically supports iOS, Android, Windows phone, Blackberry, Fire OS, and Ubuntu Mobile, meaning that Geotagger can be deployed on any of these platforms at a later date. This can potentially give way to a dramatic increase in the number of people the project can reach. A number of features built into the Geotagger application would have been impossible without Cordova: GPS locations, native navigation, photos taken by the camera, as well as chosen from the file system, toast notifications, and SQLite. Figure 3.8 illustrates the architecture of Cordova.

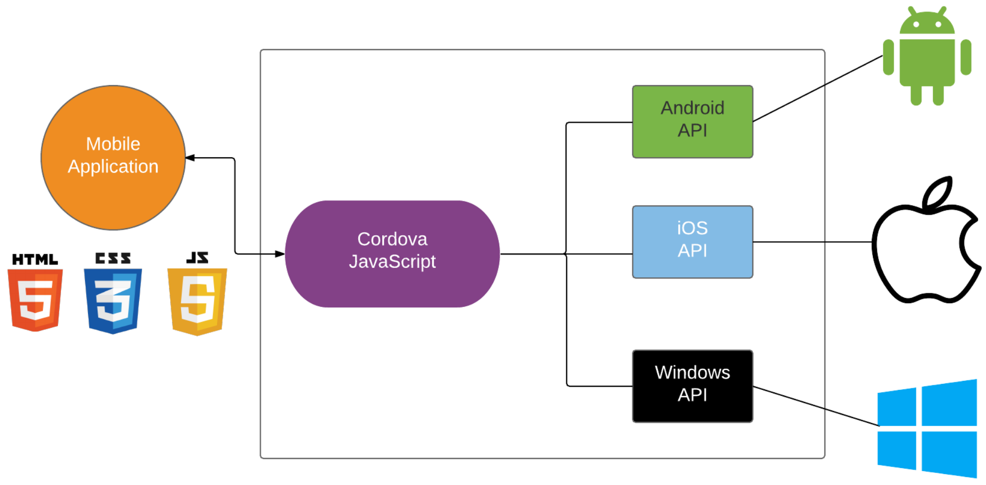


Figure 3.8: Cordova Architecture

In implementing Ionic and Cordova in the project, several considerations were made in order to keep these elements as loosely coupled as possible. While Cordova does provide a wealth of APIs from the start, there are a good number of "plugins" which others have developed for the platform. In this way, one particular Cordova API may prove to be more desirable than another. However, if this API is heavily integrated into the application, then changing the API calls would have far-reaching implications. The application was designed with a number of modules in mind, and any plugin that accomplished a certain use case was placed in its own Angular service.

For example, image picker and camera APIs were placed in a single “ImageService”, which was then referenced throughout the application. If at some point the camera API needed to be replaced, then the implementation with the ImageService could be changed, without modifying the publicly exposed functions on the service. This process was repeated for other concerns such as logging, local database querying, toast notifications, and geo-location services. Unfortunately, Ionic APIs must be coupled more closely with the application. Ionic is more a framework upon which the application is built and Cordova is a library. In any case, Ionic elements were made to be more simple and reusable. Due to Angular’s concept of “directives”, Ionic components could be broken up into their own components, which then could be brought together to compose larger elements. This helped minimize code-repetition and served to make the application’s source code more readable and maintainable.

3.3.2 SQLite

SQLite is a self-contained SQL engine for use with localized applications. It is generally used when a network connection is not required, otherwise unavailable, or where a more fully featured SQL implementation (e.g., MySQL) would create unnecessary bulk for the project’s scope. The role of SQLite in this project was to maintain a local store of data objects that could be queried for rapid user interface displays, as well as offline application functionality. Data is stored as the user navigates between views. When the user returns to a view, the application checks the cache and immediately displays relevant data if it is found. This provides the user with a fluid user experience, even if the connection to the remote database is lost.

SQLite maintains simple database text-files which are highly portable and allow for a fast, transaction-based queries one might find in a heavier-weight implementation. Due to its small installation size, SQLite is oftentimes favored for embedded applications where storage space is limited. For this project, SQLite took the shape of a Cordova plugin, as most major vendors ship their devices with built-in SQLite support. Cordova provides a transaction-based interface to the existing SQLite software. While the Cordova SQLite implementation lacks several features found in traditional SQLite (most notably foreign key support), the Cordova plugin has the facilities needed by this application in order to implement an effective caching solution.

The SQLite Cordova plugin provides the most crucial functionality of any third-party software included in this project. Functions regarding SQLite database operations appear in a single file - the "CacheDatabase". The CacheDatabase exposes a public functions which are used to read and manipulate data in the cache. This file contains both DDL and DML queries. It is responsible for creating database tables, dropping tables, and reading data from those tables. In this way, whenever the application is launched, the entry script runs an initialization function to start a Cordova SQLite instance and readies it for further querying. SQLite provides long term data storage on the device and serves two main functions for this project: record caching and action caching. Record caching concerns itself with storing data retrieved from the REST API. This is most useful in readily showing interesting data to the user, as the cache is checked each time the controller makes a request for data. If the cache contains relevant data, then it is given to the controller and immediately displayed to the user. This gives the application a more fluid user experience as local cache queries are oftentimes faster than an API request. Once the API returns the requested data, this newly retrieved data must be compared and merged with data taken from the cache. It is done to ensure that the local SQLite cache is as up-to-date as possible. Additionally, record caching helps to reduce redundant API calls, as resources (e.g., images) are checked and only requested from the API when they are needed. Furthermore, record caching gives a rudimentary offline viewing experience. If the user loses connection to the Internet, they are still able to view a subset of the available data (i.e., data stored in the local cache). As the Geotagger project largely deals with natural phenomena, members may oftentimes find themselves in locations where a network connection is not available. By locally caching every record the application retrieves, users are provided a useful, limited, experience even when an Internet connection is not available. While record caching deals with storing retrieved data locally, action caching, on the other hand, is more concerned with the modification of the local data.

Action caching enables a mechanism for affecting immediate remote changes if an Internet connection is available or as soon as one is established. Traditional web applications require a constant Internet connection as they are usually interacted with via an Internet browser. However, mobile applications are expected to give some kind of user-experience, even when a network connection is not available. While record caching affords the user the ability to view data they've already visited, there is no way to update this data. Action caching provides a way to "cache actions" in order that they may be resolved either immediately or at a later time. In this way, every "action" that the user takes (e.g., update, add, or delete data) is "pushed" onto the action cache and resolved as soon as the device detects an available network connection. When a user makes an action, this action is reflected in the local cache to allow the user to immediately view the changed data. A timestamp and reference to this data is stored in the "action cache". Once a connection is established, the "ActionService" removes the action, reads the referenced data, and sends an HTTP request to the API so that the change may be affected on the Geotagger server. Once the API responds with the result of the change, the action cache uses the data to synchronize the cache to any changes that took place on the server (e.g., replace the new records old ID value with one assigned to it by the remote database). Indeed, SQLite is an extremely important piece of the application structure which reduces redundant data, allows for fluid online viewing, and affords users an application-like offline experience that is missing in most traditional web applications.

3.3.3 Google Maps

Google Maps is a popular mapping API available to a number of clients. Google Maps supports iOS, Android, and Web clients. Google Maps also provides a number of location APIs that can be used for geocoding, location searching, or mapping street directions between a series of points. Google Maps hooks into an existing application, and upon instantiation, draws the map area. Google Maps was a natural fit for this application as Geotagger data deals with "tagging" locations. Google Maps was used to create a novel visualization for tag data, giving a different way to view this tags. This is especially interesting when viewing a map with multiple tags, as users are able to see tag locations in relation to one another.

Google Maps was selected for its staggering popularity, safe in the knowledge that there would be abundant tutorials and examples should trouble arise. While Google does provide a script file for use with the Google Maps API, an Angular plugin wrapper called "ngMaps" was chosen. This allowed the current project to follow the standard AngularJS dependency injection convention, making it easier to instantiate Google Maps object instances and pass them throughout the application. Figure 3.9 gives an example of a Google Map screen.

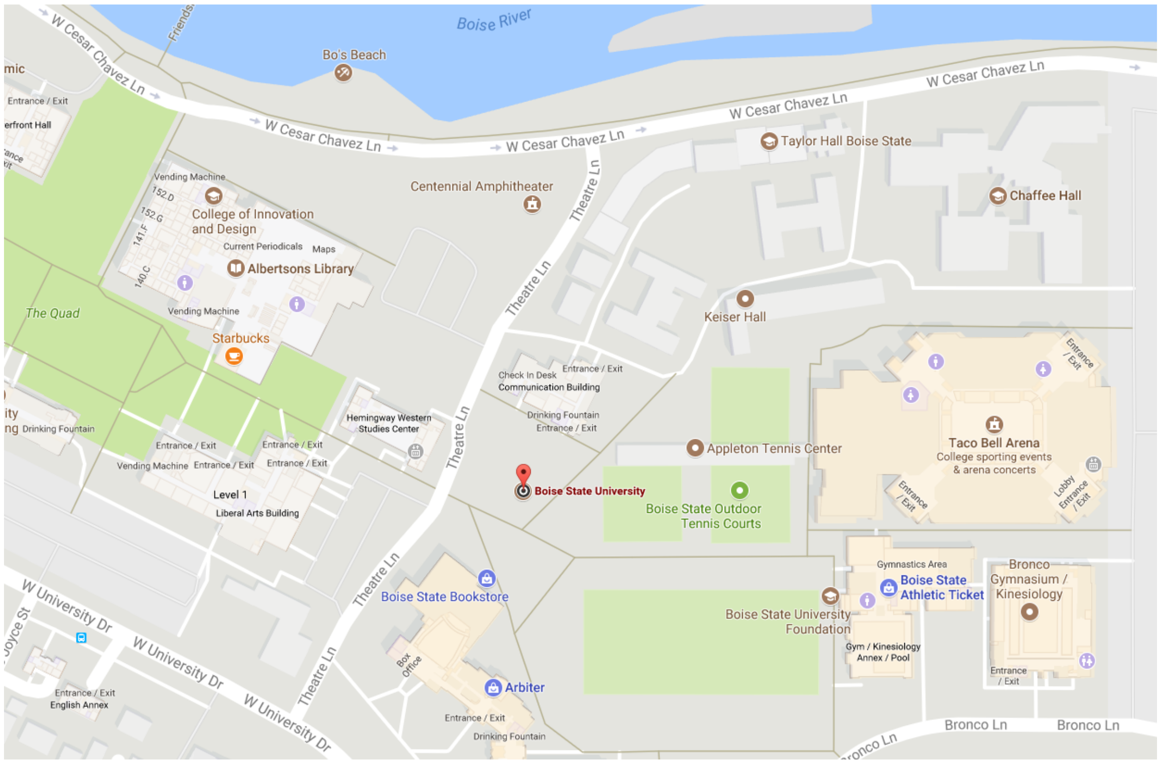


Figure 3.9: Google Map View

A Google Maps window was implemented as an AngularJS "directive" to allow for simple and efficient reuse throughout the application. A flexible directive was created that could handle single locations and a list of locations. This directive could easily be integrated into other areas of the application should the developer want to incorporate further map views; as all that is needed is a list of latitude and longitude locations to display. This interface provides an intuitive way to view tag location data, since the user can simply look at the map view to determine the tag and their own location relative to the tag. From here, users can press a button which opens the native navigation application in order to safely travel from their current location to a desired tag. Maps were also added to the adventure view to provide context to the user about tags created by other members of that particular adventure. The goal was to create an informative application that would feature intuitive interaction. As most users will likely be familiar with geographical maps, using this feature should be a simple and enjoyable experience.

3.3.4 Socket.io

Socket.io is a JavaScript library that allows for real time, bi-directional communication between clients. Socket.io passes "messages" between clients and server with events which can be hooked-into to provide a wide range of contextual actions. Current Geotagger software facilities were not suited to real-time communication between devices. As a result, the Socket.io library was used to implement a real-time messaging function for implementing the shared social space enhancement (**SE3**).

A server application is needed to implement Socket.io. The server defines the transportation protocol with Websockets serving as the default. In the event that the server does not support websockets, Socket.io is able to downgrade the connection to a protocol the server supports. One of the greatest strengths of Socket.io is the documentation and overall simplicity of use. The library is "event-based". It simply calls a number of events and provides "callback" routines to develop a minimally functioning Socket.io application. The above technologies were incorporated into the Geotagger application to realize functionality that has been present in other forms of the Geotagger client. Socket.io was particularly leveraged in order to provide data transmission of the new "adventure chat" feature. It allows members of the same adventure to visit a shared space where they can communicate ideas and discuss the adventure. Figure 3.10 depicts the Socket.io application architecture.

Traditionally, data sent between a Geotagger client and the Geotagger server has strictly involved a client request, followed by the server's response. This scheme

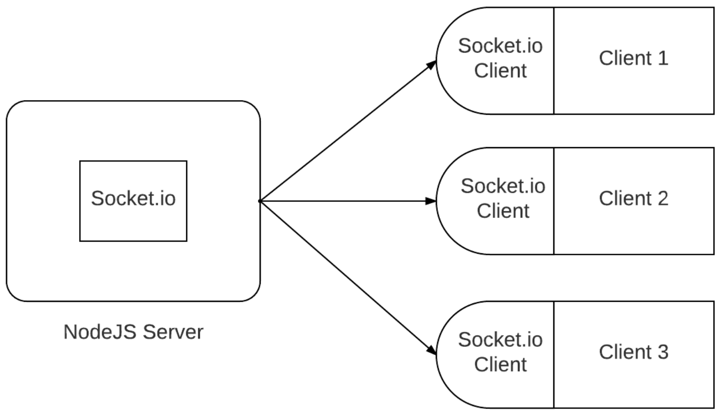


Figure 3.10: Socket.io Architecture

was unacceptable, as the vision for an adventure chat area would involve a real-time chatting implementation that is similar to features found in major social networks. Socket.io consists of two primary components: client and server. Both the client and server listen for events caused by each other. The Socket.io server runs on a NodeJS instance. Server developers can listen for different kinds of events which are triggered by client messages and vice-versa. If the server receives a message relating to one of its events, it will "emit" the message to all of the clients that are listening to the same event. Figure 3.10 demonstrates the Socket.io server emitting an event.

Socket.io provides an intuitive API for developing a real-time chat application which integrated easily into the existing codebase. This is a new feature to the Geotagger project, so it was one of the final social enhancements to be implemented for this work. It must work seamlessly with existing features. In order to continue using AngularJS's dependency injection system, Socket.io would have to be brought into a service of its own. So, Socket.io was relegated to its own AngularJS service which was responsible for maintaining a single Socket.io client object. Other controllers that needed to send messages to the Socket.io server would need to refer to public functions on this service if they wished to transmit data. However, due to AngularJS services implementing the singleton pattern, creating and managing library instances is handled within the framework itself. Incorporating this service into a controller allows the developer to immediately reflect data updates in the view. This gives the user interface a fluid, real-time feel that users of social media have come to expect. Socket.io has given Geotagger the capabilities that it needs to provide an interactive and hopefully inviting social space that will allow users to quickly and informatively interact with one another.

3.4 Walkthrough of Social Enhancements

Development for the application took place in two stages. The first stage of development was concerned with implementing the core Geotagger functionality. This included the ability to log into the system and use the returned authentication token to perform further actions with the Geotagger API. Special attention was paid toward adding, updating, and deleting tags, adventures, and comments. These features would be most heavily used during the evaluative portion of this work. The second stage of development focused on implementing the proposed social enhancements. These enhancements have been given an abbreviated code which will be referred to in the remainder of the chapter.

- **SE1** - Making comments more prevalent
- **SE2** - Connecting people through their contributions
- **SE3** - Creating a shared social space for adventures
- **SE4** - Geographic overview of tags

The remainder of this section will walk the reader through the various screens in the application. This will give context to the operation of these social enhancements within the application. Two images will be shown for screens that differ between the two applications.

3.4.1 Login View

It is common practice for the initial screen of the application to allow the user to log into the system. Users enter the username and password they registered during the sign-up process. If an incorrect username/password combination is entered, the system will provide a "toast" popup informing the user and allowing them to try again. Figure 3.11 depicts the login screen.

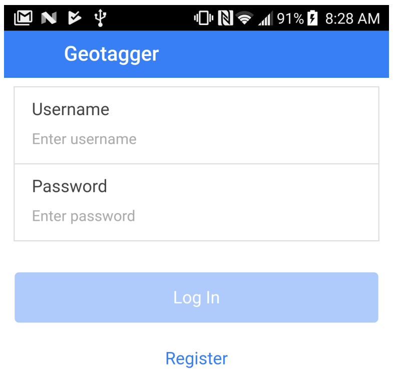
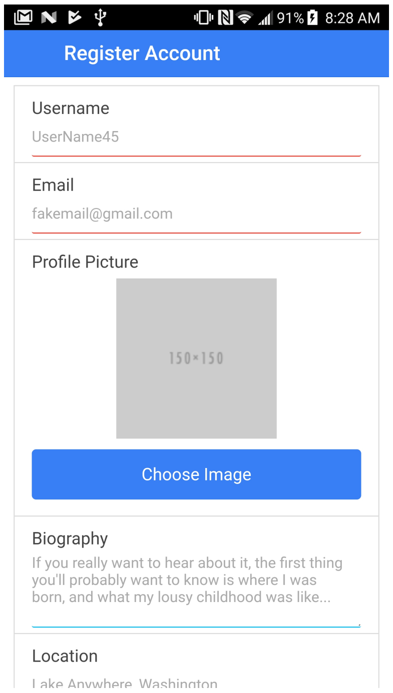


Figure 3.11: Geotagger Login View

Uninitiated users may register with the Geotagger system by pressing the "Register" button. This will launch a "modal" form with the necessary fields for registering a new account. Figure 3.12 shows the register view.


91% 8:28 AM

Register Account

Username

UserName45

Email

fakemail@gmail.com

Profile Picture

Biography

If you really want to hear about it, the first thing you'll probably want to know is where I was born, and what my lousy childhood was like...

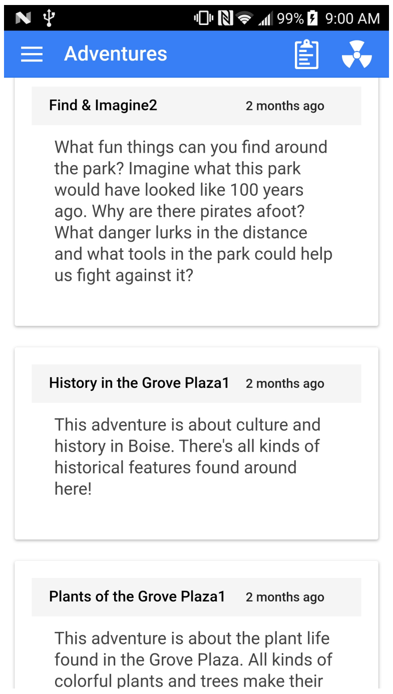
Location

Lake Anywhere, Washington

Figure 3.12: Geotagger Register View

3.4.2 Adventure List View

Upon successfully logging into the system, the user is greeted with a list of adventures which they own or are a member. Each unit in the list is a "card" that represents an adventure. These cards display: name of the adventure, description of the adventure, and relative time at which the adventure was created. This view serves to centralize a user's activity, giving an overview of all of the adventures available to them. Additionally, users may slide the cards to the left in order to access contextual "edit" and "delete" actions for that particular adventure. Figure 3.13 shows the adventure list view.



Find & Imagine2 2 months ago

What fun things can you find around the park? Imagine what this park would have looked like 100 years ago. Why are there pirates afoot? What danger lurks in the distance and what tools in the park could help us fight against it?

History in the Grove Plaza1 2 months ago

This adventure is about culture and history in Boise. There's all kinds of historical features found around here!

Plants of the Grove Plaza1 2 months ago

This adventure is about the plant life found in the Grove Plaza. All kinds of colorful plants and trees make their

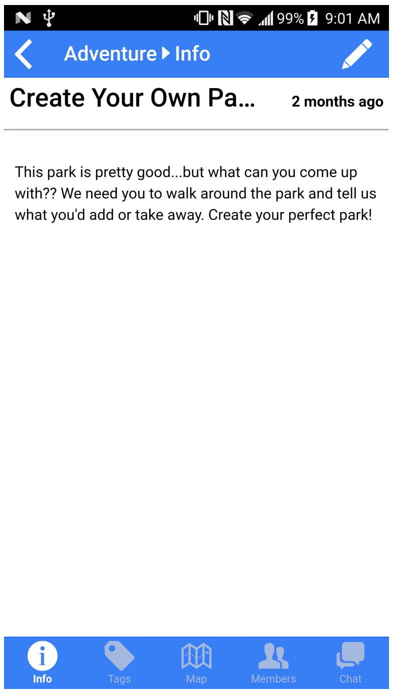
Figure 3.13: Adventure List View

3.4.3 Adventure Detail View

Pressing a finger ("tapping") on one of the adventure cards will bring the user to a detail view for that adventure. Detail views are divided into several different tabs, each tab pertains to a different perspective of that adventure. Adventure detail tabs include: "info", "tags", "map", and "members". The enhanced version of the application also contains a "chat" tab.

Adventure Info

The info screen serves as an initial landing page for the adventure. The information contained on this screen is identical to the information present on the card representation. The purpose of this screen is to orient users, acquaint them with the tabbed layout and grant them context for a new set of screens. Figure 3.14 shows the info screen.



Create Your Own Pa... 2 months ago

This park is pretty good...but what can you come up with?? We need you to walk around the park and tell us what you'd add or take away. Create your perfect park!

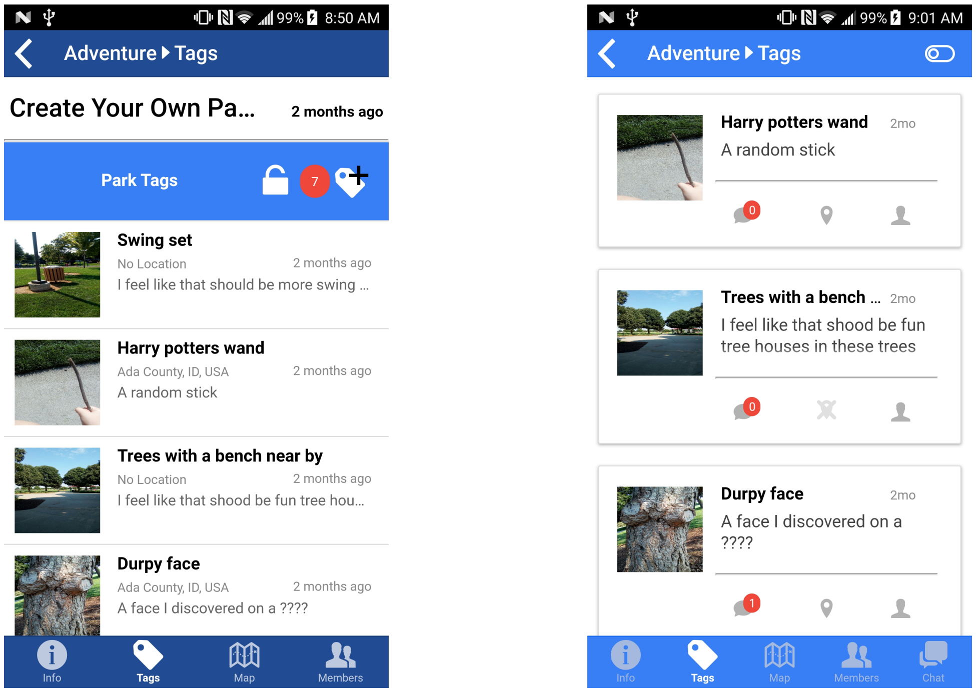
Figure 3.14: Adventure Info Screen

Adventure Tags

The adventure tags screen shows all of the available tags for an adventure. This screen is different between the two application versions. This screen represents enhancements **SE1** and **SE2** as users are able to comment more effectively and directly navigate to user profiles (figure 3.15). The screen on the left (dark blue trim) represents the original development effort, while the screen on the right (light blue trim) represents the enhanced application. Tag items on the left represent a “list-like” group, while tag items on the right are more “card-like”. List items provide a more compact representation, allowing more items to fit on one screen. The enhanced “card” design opts for more screen space, affording additional functionality to users. This card representation is the result of several design iterations. Earlier prototypes of these iterations can be found in appendix B.

The list items of the original application have no visible action buttons. Users may slide the list items to reveal “edit” and “delete” buttons. Beyond these actions a user is only allowed to press on the “list” item. Upon pressing the list item, the user is taken to the tag detail view. This is another tabbed view displaying different aspects of the tag. This view will be discussed in further detail in section 3.4.4. Tag cards supersede the need for a tag detail view, as all pertinent tag data is found directly on the card (figure 3.16). Tag cards contain a row of action buttons which are absent from the original application. Buttons included are: “comments”, “location”, and “user”. These buttons allow users to view detailed information about the tag without having to leave the tag list view.

The comment button is represented by the chat-bubble icon on the center-left of the card and serves two primary purposes in making comments a more prevalent part of the user workflow - the intent of **SE1**. Firstly, it informs the user about comment activity on that particular card and allows users to see the number of comments on a tag at a glance. Secondly, users are able to navigate to the tag comments view with a single button press. This view is depicted in figure 3.23. These two features serve to make comments more integrated and engaging to users. Users now have



Swing set
No Location — 2 months ago
I feel like that should be more swing …

Harry potters wand
Ada County, ID, USA — 2 months ago
A random stick

Trees with a bench near by
No Location — 2 months ago
I feel like that shood be fun tree hou…

Durpy face
Ada County, ID, USA — 2 months ago
A face I discovered on a ????

Harry potters wand — 2mo
A random stick

Trees with a bench … — 2mo
I feel like that shood be fun tree houses in these trees

Durpy face — 2mo
A face I discovered on a ????

(a) Original (b) Enhanced

Figure 3.15: Adventure Tag Views

an easier workflow for monitoring comment activity and adding comments. In the original version, there is no indication of the number of comments on a tag. Rather, a user must navigate to the tag's comment screen and view the actual comments in order to get a sense of comment activity. Adding a comment takes place in the same screen in the enhanced version of the application to enable users to contribute more efficiently.

The button in the center of the card is a "location" button. Pressing it opens the map view for that tag. This view can be seen in figure 3.24. The tag map view gives users a visual representation of a tag's location, given that the tag has been provided with latitude and longitude data. This view is implemented using Google Maps, so the user is granted additional built-in functionality. This functionality will be discussed in section 3.4.4.

The third button present on the tag card is the “user” button; it introduces the functionality proposed by **SE2**. This button serves to tie authoring users more closely to their contributions. Pressing this button will bring users to the author’s profile detail view. This is another tabbed view which details a user’s registered information, as well as work they contributed. The profile detail view is discussed further in section 3.4.5.

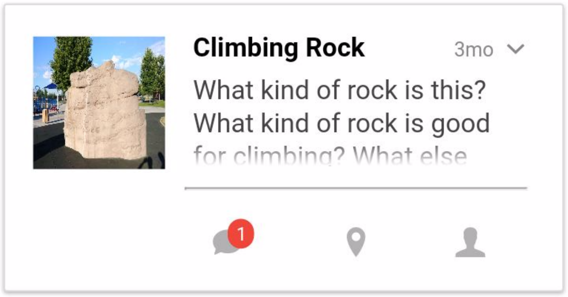

Climbing Rock
3mo
What kind of rock is this? What kind of rock is good for climbing? What else
1

Figure 3.16: The Tag Card

Adventure Map

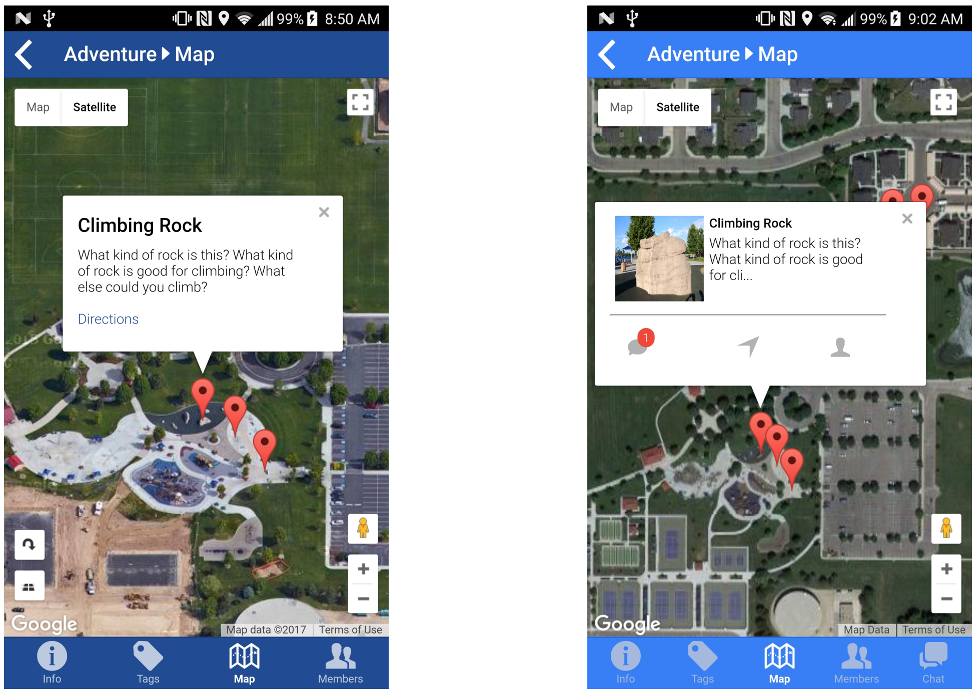
The map view contextualizes all tags contributed to the currently selected adventure. Both versions of the application display markers relating to the adventure’s tags, as long as the tag has latitude and longitude information. To bolster this geographical representation, Google Maps provides several built-in map controls. The top left of the map houses a toggle button which allows users to change between a more realistic “Satellite” tile-set or a stylized “Map” tile-set. The square in the upper right corner is a “fullscreen” button for the map view, while the “plus” and “minus” buttons in the bottom right allow for manual zooming. Finally, the yellow character above the "zoom" buttons allows users to drag this icon onto a city street in order to display Google Map's "Street View". Figure 3.17 shows the adventure maps.

Each of the markers displayed on the map is a click-able button. Pressing on one of these markers brings up a contextual "info window" which provides relevant tag information. The original version of the application displays the name and description of the tag. The "Directions" link which will launch the device's native navigation application in order to safely direct users to the tag. The enhanced application also displays the tag name, description, image and a suite of action buttons. These buttons perform identical operations to that of the buttons on the tag card with one exception. The center "location" button has been replaced with a "navigation" button which acts identically to the "Directions" link.

These additional action buttons facilitate enhancements **SE1** and **SE2**, much the same as the tag cards. The "comment" button takes users directly to the comment screen for the selected tag, allowing them to quickly and efficiently continue to contribute. The "profile" button serves to further tie users to their contributions, even on the map screen. The included image also grants users greater geographical context, especially when used with the satellite view where as large environmental features may be seen from the default zoom level.

Adventure Members

The fourth adventure view displays the users that are a part of the currently selected adventure. From here, users can be added or removed from the adventure. Adventure coordinators can create "member collections" which group users by roles. Scientists and adventure personnel can be grouped into one collection, while students or citizens can be placed into another collection. As with other lists, items act as links to



Climbing Rock

What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

Directions

Climbing Rock

What kind of rock is this? What kind of rock is good for cli...

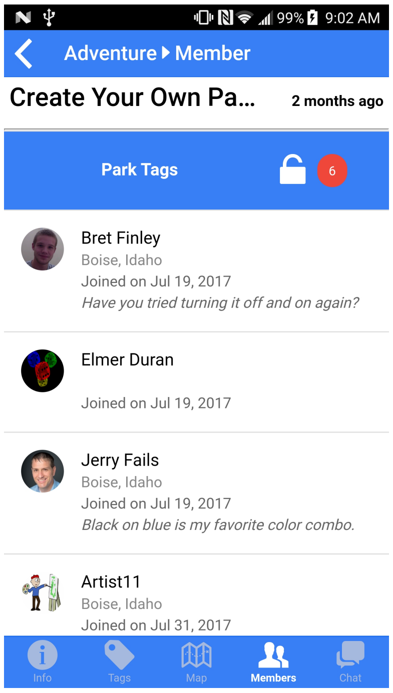
(a) Original (b) Enhanced

Figure 3.17: Adventure Maps

further information about the item. Pressing on a member item takes the user to the member's profile detail view. In the enhanced application, this flow provides a connection between users and the content they contribute (e.g., tags). This view will be discussed further in section 3.4.5. Figure 3.18 shows the adventure member screen for both applications.

Adventure Chat

The adventure chat embodies enhancement **SE3** by creating a shared social space for members of an adventure. The adventure chat represents an all new feature for the Geotagger project. This tab acts as an instant-messaging system between members of an adventure. Adventure chat areas are local to each adventure, so the conversation


99% 9:02 AM

Adventure ▸ Member

Create Your Own Pa... 2 months ago

Park Tags 6

Bret Finley
Boise, Idaho
Joined on Jul 19, 2017
Have you tried turning it off and on again?

Elmer Duran
Joined on Jul 19, 2017

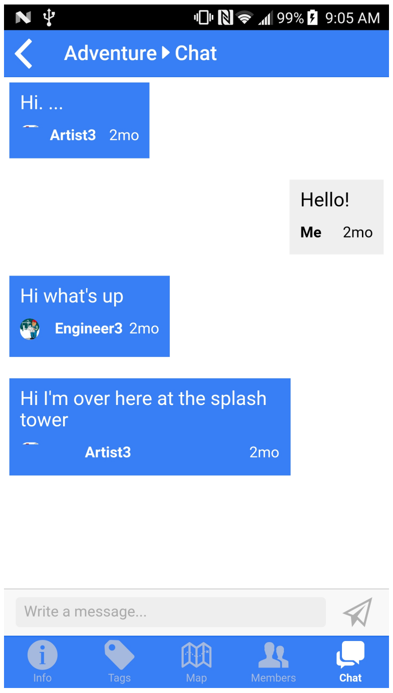
Jerry Fails
Boise, Idaho
Joined on Jul 19, 2017
Black on blue is my favorite color combo.

Artist11
Boise, Idaho
Joined on Jul 31, 2017

Figure 3.18: Adventure Members

stays focused on the adventure at hand. In a similar fashion to popular SMS and IM applications, user-sent messages appear on the right side of the screen (e.g., black text with a gray background). Other messages are visible on the left side of the screen (e.g., white text with a blue background). Each chat message is appended with the authoring user's username, profile image, and the time at which the message was sent. To send a message, the user presses on the "input bar" at the bottom of the screen and enters the text on the dynamic keyboard. Pressing the "paper airplane" icon sends the message to the Socket.io server, which then "emits" the message to the other clients that are members of this adventure. The chat window can be seen in figure 3.19.

When a user is not currently on the chat screen, any chat messages received during that time will increment a small badge icon on the "chat" tab. This enables users



Hi. ...
Artist3 2mo

Hello!
Me 2mo

Hi what's up
Engineer3 2mo

Hi I'm over here at the splash tower
Artist3 2mo

Figure 3.19: Adventure Chat

to become aware that other users are contributing to the chat while they are away. This badge serves a similar purpose to the badge seen on the "tag" card, giving users further context as to the discussion happening around them. The chat badge is shown in figure 3.20.

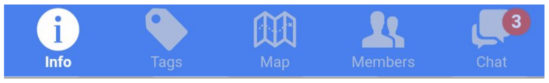


Figure 3.20: Adventure Chat Badge

3.4.4 Tag Detail View

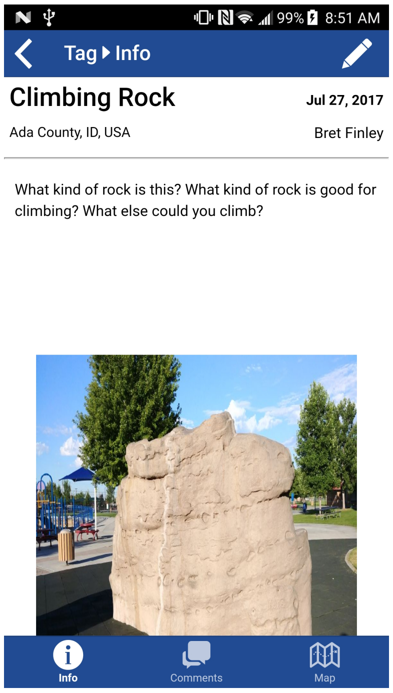
The tag detail grants users detailed information concerning a selected tag. Users can navigate to the tabbed view by pressing on a tag list item in the adventure tag view (section 3.4.3). The enhanced application makes use of the comment and map views. However, the tabbed version of this view is only available to the original version of the application, since the enhanced version replaced this view with the tag card representation. The tabbed view is comprised of three sub-views: “tag info”, “tag comment”, and “tag map”.

Tag Info

This view grants the users some contextual information about the currently selected tag. This view acts as a “landing page” after a user pressed a tag list item (figure 3.21). Users can make edits to the tag with the “pencil” button in the upper-right corner. Tag information is kept in a section below the action bar. Here the view displays: the name of the tag, date it was authored, location description, and username of the author. Farther below this is the tag’s description and image, if one exists. If a user presses on the displayed image, a modal window showing a zoomed version of the image will appear (figure 3.22).

Tag Comments

Users are able to view and add their own comments in the tag’s “comment” view. Figure 3.23 shows the tag comment view for both applications. Note the presence of a tab-bar at the bottom of the original version. The enhanced version appears when the user presses the “chat-bubble” button on the tag card. Comments are the primary form of interaction between Geotagger members. As such, it is important that the process of adding a comment be simple and expedient. The “input bar” at the bottom of the comment screen represents a further step toward expedience (**SE1**). In the original application, users need to press the “plus” button in the upper-right



Climbing Rock Jul 27, 2017

Ada County, ID, USA Bret Finley

What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

Figure 3.21: Tag Info

corner of the screen. This action produces a form used for adding comments. The enhanced application streamlines this workflow as users can type a comment without having to leave the screen. The "camera" icon allows users to add an image by uploading one from the device or by taking a new photograph. A further enhancement focused on implementing profile links for user profile images. User profile icons can be found on the left side of each comment item; these icons provide a direct link to the commenting user's profile. This approach creates a stronger connection between users and the content that they contribute (**SE2**), as users are able to further explore user-created content within the profile view. The profile view is discussed in detail in section 3.4.5.

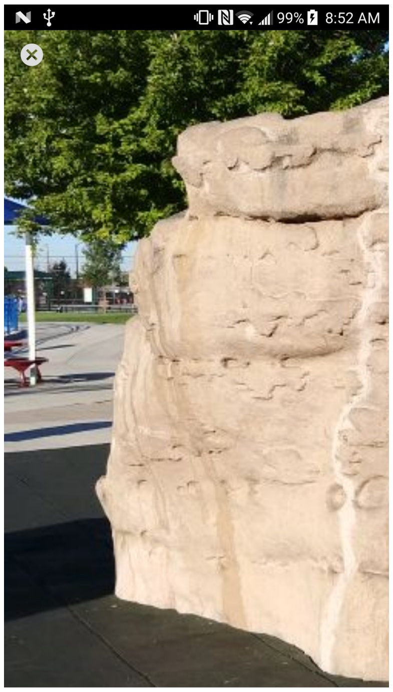
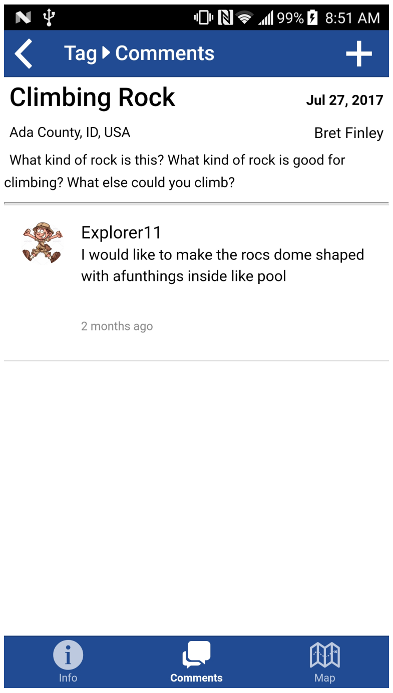


Figure 3.22: Zoomed Image

Tag Map

The tag map displays tag's geographical location according to the tag's latitude and longitude metadata. Similarly to the comment view, the tag map view is accessible via the "location" button on the tag card, rather than through a tabbed interface. The absence of a tab bar in figure 3.24 is noticeable. The tag map view also displays the user's location as a blue marker, as opposed to the orange tag marker. This screen is very similar to the adventure map view of section 3.4.3, with a few exceptions.

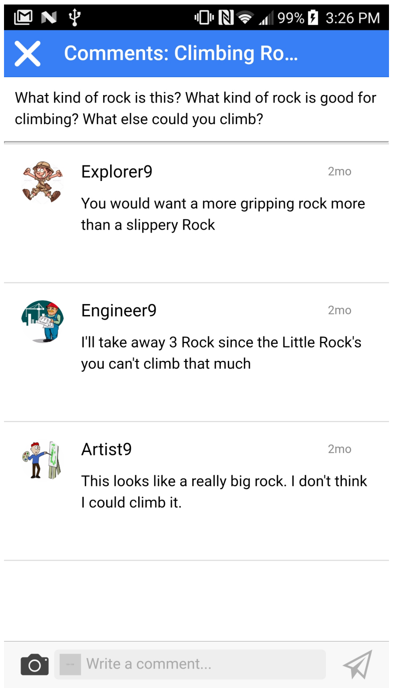
The first difference is that the tag map only reveals the location of the currently selected tag and the user's current location. The tag map screen contains several map controls not found on the adventure map screen. Both versions contain buttons in the bottom-left of the screen to allow the user to rotate and enlarge the map.



Climbing Rock
Jul 27, 2017
Ada County, ID, USA
Bret Finley
What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

Explorer11
I would like to make the rocs dome shaped with afunthings inside like pool
2 months ago

(a) Original



What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

Explorer9 2mo
You would want a more gripping rock more than a slippery Rock

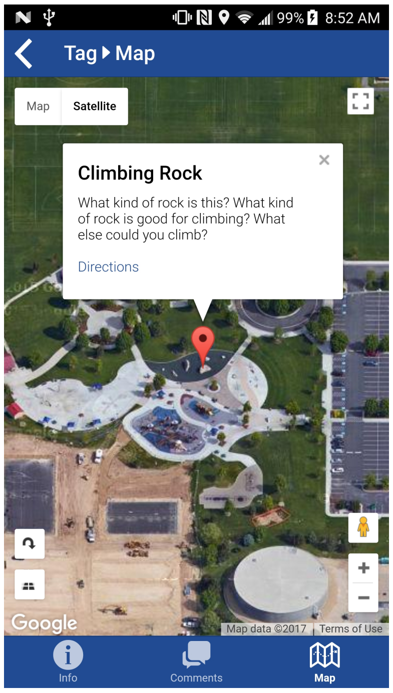
Engineer9 2mo
I'll take away 3 Rock since the Little Rock's you can't climb that much

Artist9 2mo
This looks like a really big rock. I don't think I could climb it.

(b) Enhanced

Figure 3.23: Tag Comments

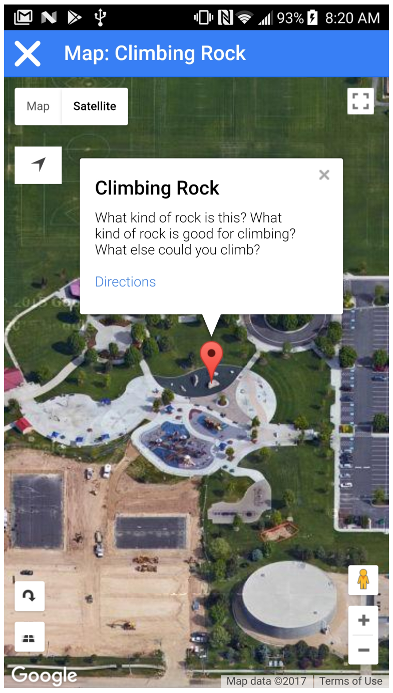
The enhanced version of the application also contains a “navigate” button located under the Map/Satellite toggle button. It provides identical functionality to the “Directions” link. Finally, when the user presses a “marker” icon, it invokes an “info window” design similar to that of the original application, as opposed to the “card” design found in the adventure map. When a user brings up the tag’s map, they have navigated “into” the tag. Providing the user with the same functionality as the adventure map cards would create the opportunity for confusing UI interactions. A tag card existing within a tag card may be confusing to the user, since the tag map is accessed from the tag card. Therefore, it was decided to maintain the original design; providing user’s with on the the tag’s name, description, and a button for getting directions to the tag.



Climbing Rock

What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

(a) Original



Climbing Rock

What kind of rock is this? What kind of rock is good for climbing? What else could you climb?

(b) Enhanced

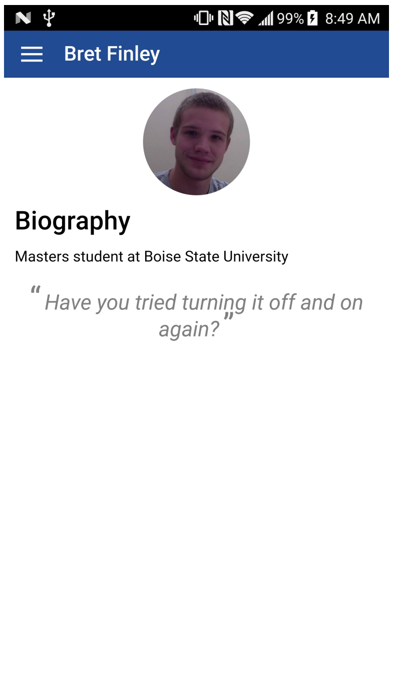
Figure 3.24: Tag Map

3.4.5 Profile Detail View

The profile view is a means for users to become more familiar with one another, giving a "face to the name" and helping to humanize contributors. This was the intent of enhancement **SE2**. User accounts contain some interesting personal account data designed to engage other users. This data includes a personalized "quote" and a short "biography". This was designed to give users a sense of personality and allow for a degree of personal expression. This information is coupled with project data, creating an association between the users themselves and their contributions to the project. In developing the social enhancements, the profile screen was converted to a tabbed layout, thus allowing visiting users to explore the user's other contributions to the project.

Profile Info

The profile "info" screen acts as the initial screen for the enhanced application. This screen is reached by pressing the "user" button on the tag card (figure 3.16). Figure 3.25 shows that in the original application the profile view is a single screen that consists of the user's: name, profile image, biography, and quote. This same information remains unchanged in the enhanced version.

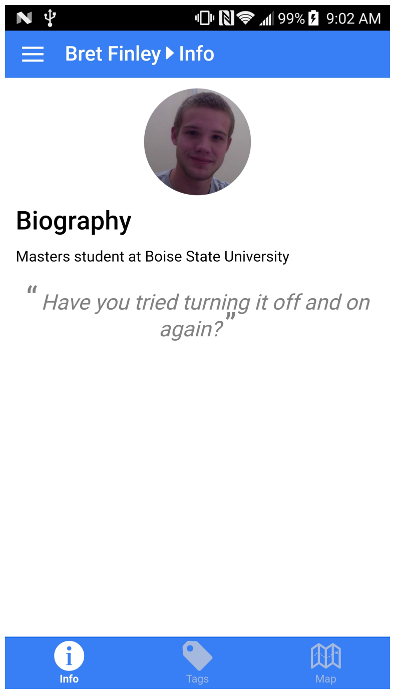


Biography

Masters student at Boise State University

"Have you tried turning it off and on again?"

(a) Original



Biography

Masters student at Boise State University

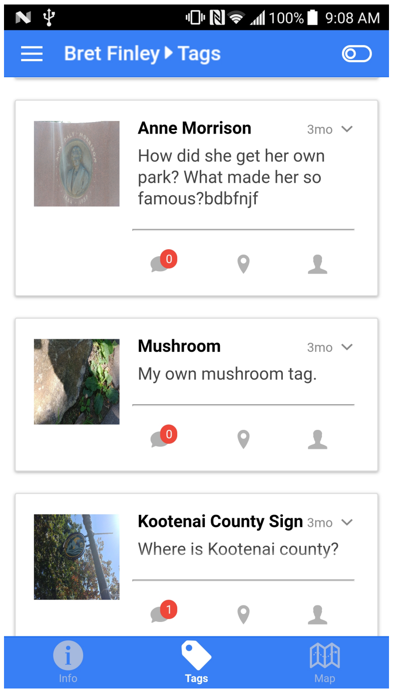
"Have you tried turning it off and on again?"

(b) Enhanced

Figure 3.25: Profile Info

Profile Tags

The "profile tags" view serves to connect users to their contributions. In the original application, there is no way for users to see the contributions of other's. The aim of this enhancement is to give users greater awareness of other members of the project, and foster a growing community by creating personal connections between users. The profile tag list contains identical functionality to that of the adventure tag list with the exception that every tag seen here has been contributed by the same user. This view allows visiting users to directly comment and get to a user's tags (figure 3.26). In hindsight, the "user" button may be somewhat redundant. This serves to create a kind of rapport between users as they can see what kind of tags interest them.


9:08 AM

Bret Finley ▸ Tags

Anne Morrison 3mo

How did she get her own park? What made her so famous?bdbfnjf

Mushroom 3mo

My own mushroom tag.

Kootenai County Sign 3mo

Where is Kootenai county?

Figure 3.26: Profile Tags

Profile Map

The profile map gives visiting users a visual representation of all of the tags found in the profile tag list, shown in figure 3.27. This serves to implement enhancement **SE4** to contextualize the data from the previous screen and give a novel depiction of a user's contributed work. Controls present on this view are identical to those in the tag map view, yet this map view has the return of the tag card info windows found in the adventure map view (section 3.4.3). This allows users to make comments on a user's tag within the context of the tag's location. In hindsight, the presence of the "user" button is perhaps redundant.

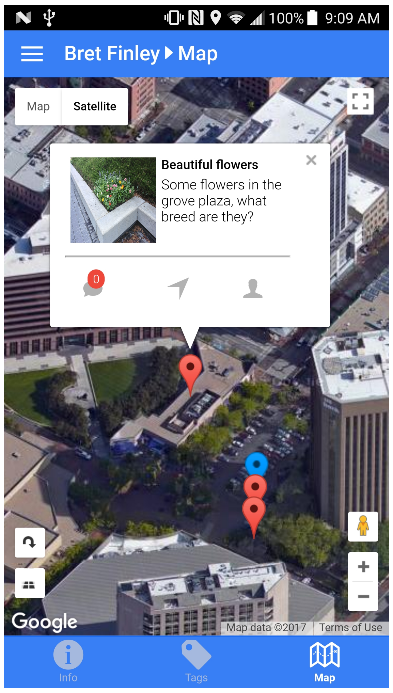


Figure 3.27: Profile Map

3.4.6 Tag Overview

Though these views were not part of the evaluative study, they still represent a step toward realizing the proposed social enhancements. A geographic overview of a single tag is available in the tag detail view (figure 3.24) and an overview of tags in an adventure are visible in the adventure tags view (figure 3.17). The aim of this view was to further implement enhancement **SE4** and provide an overview of all tags that a user could view. The result is to further contextualize the available work and provide the user with a novel domain-centric visualization.

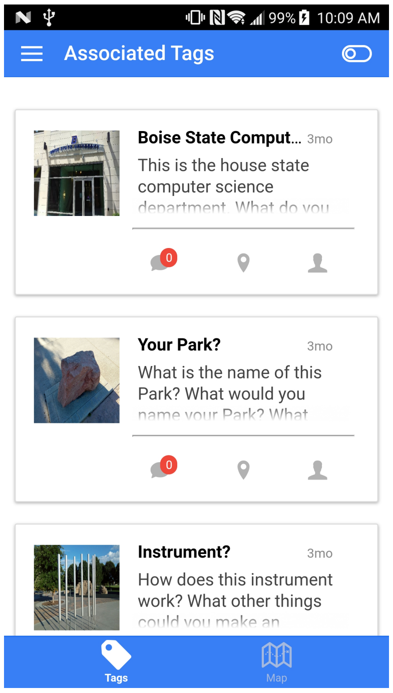
This enhancement was not completed due to time constraints and difficulties encountered during development. Much of the functionality for the enhancement was implemented. However, due to performance problems with large sets of tags, the enhancement did not appear in the user study. This tag overview contains two tabs: the tag list and the tag map. The list displays the card representation of the tags, allowing users to comment on and locate interesting tags quickly. The map view is very similar to the adventure map view, displaying all of the tags from the list view geographically.

Aggregate Tag List

The intent of the aggregate tag list is to gather tags from all adventures that a user has membership and display them in a list form for rapid browsing. In previous versions of the application, this tag list mirrored the content found in the user's profile tag list and displayed only tags that the user had created themselves. Figure 3.28 shows the aggregated list. This list contains tags from several different authors. Note the same card representation seen in other list views.

3.4.7 Aggregate Tag Map

The aggregate tag map gives users a geographic representation of all of the tags available to them. These tags are displayed on a familiar Google Map that has the same features as the tag map mentioned earlier. Tags are partitioned by assigning different color markers depending on each tags' adventure. Tags with similar color


10:09 AM

Associated Tags

Boise State Comput... 3mo
This is the house state computer science department. What do you

Your Park? 3mo
What is the name of this Park? What would you name your Park? What

Instrument? 3mo
How does this instrument work? What other things could you make an

Figure 3.28: Aggregate Tag List

markers are from the same adventure, allowing users to mentally group these tags. Figure 3.29 shows the aggregated map.

This chapter has provided an overview of the engineering effort completed in this thesis. Two working Geotagger applications were completed, one application with and one without the social enhancements, studies were conducted to understand the effectiveness of these social enhancements.

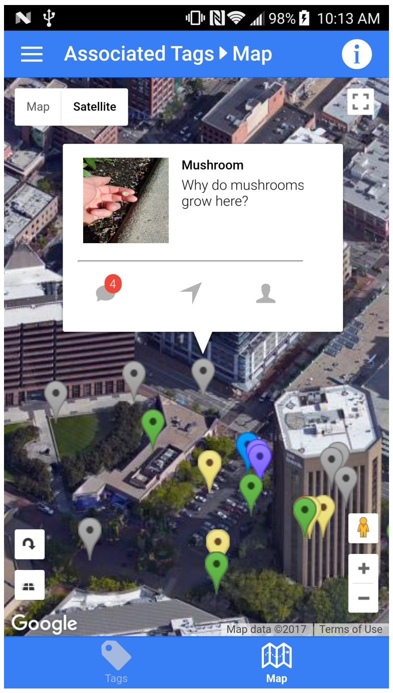


Figure 3.29: Aggregate Tag Map

CHAPTER 4

EVALUATION

Having introduced the social enhancements previously, this chapter will discuss the means by which those enhancements were evaluated. The evaluative study took place in two stages. The first stage was a comprehensive real-world use scenario involving children at a local park. The second stage was a more focused study of select enhancements involving the Kidsteam group. Approach, results, and limitations will be discussed for each study in order to provide a comprehensive understanding of the evaluation techniques and the resulting data.

4.1 Field Studies

The first step in evaluating the social enhancements was to design a study that would effectively exercise and assess them. This design was passed on to the IRB in the form of an expedited review application. This study was relatively non-invasive. The primary evaluation method involved simply asking children a series of questions concerning the application and surveying their responses. Once the application had been accepted, a pilot study was then conducted. This pilot study did not produce usable data, but allowed for changes to be made in the overall approach. After the approach was modified the user study began in earnest, boosting the scale of the pilot study in order to obtain more data. Unfortunately, the data gathered from the initial user study was not as demonstrative as originally expected. This outcome necessitated a second, more focused user study that would allow for a fine-grained evaluation of select enhancements.

4.1.1 Method

Two study treatments were conducted, one for each version of the application. The original application, not containing the new social enhancements was named "Geotagger X". The second iteration, this containing the social enhancements, was called "Geotagger Z". The user tasks in this study focused on the strengths of the Geotagger project, to give a users a more clear idea of what it was like to contribute to the project. In this way, tasks were created that emulated real-world usage of the application. To find a suitable location for the study, a number of public, open locations were investigated. Since this study involved surveying children, a space was selected that would make parents and children as comfortable as possible. Furthermore, the study's goal was to observe application-based interaction, not than group-based interaction. Thus children could not be grouped together, otherwise they would likely collaborate outside of the application. To alleviate parental concern, study personnel would accompany each child during the treatments. Study personnel offered guidance and support where needed, but served a primarily observational role.

Once the tasks had been designed, a survey using a five-point Likert scale was created to quantify the children's thoughts. The version of the application and survey given to the children would depend on the treatment scheme used in this study. Study designers choose between a "within-subject" or "between-subject" study. Both designs have their own strengths and weaknesses which needed to be weighed in order to determine which design would be appropriate for this study.

Tasks

Study tasks involved children interacting with the Geotagger application in a real-world use-case scenario. Allowing children to interact naturally with the application provides a more realistic representation as to how users would operate the application outside of the study. Each child was furnished with an Android device with the treatment application pre-installed. As tags and comments are the main collaborative unit of the Geotagger project, it was appropriate that they be placed at the focus of the study.

In partitioning the treatments, two different adventures were created. This kept the data organized, and allowed investigators to theme tags in order to make the treatments more varied and interesting. Study personnel created a number of "starter" tags for both of these adventures in order to orient them and gain familiarity with tags. Starting tags gave children ideas and helped generate some initial discussions. These tags were designed to be fairly obvious and easy to reach with children being able to instantly recognize the location and make their way to it without outside help. From here, children were able to explore the environment and create tags of their own.

Study designers devised three main tasks that children were asked to complete during each treatment: visit tags that others created, create tags, and leave comments on tags. Tasks functioned as a series of suggestions in the event that the children needed some guidance. Children were meant to feel like they were in control of what they were doing without becoming overwhelmed. No quota or set of ordered steps was provided, as the aim of the study was to foster open-ended interaction with the application in order to make the collected data as natural as possible.

Evaluative Artifacts

In order to evaluate the children's interaction with the application, a number of evaluative artifacts would be required. Surveys are a very common evaluative artifact, as they allow users to codify their thoughts on paper. These quantified and qualified thoughts and experiences can be analyzed later by the research team in order to draw conclusions. However, special care must be taken when the subjects of the study are children. Children do not have the cognitive ability nor the reading comprehension of adults, so special allowances must be made. Particular consideration must be given to the wording of survey queries. Query text must be written in such a way that a child can unambiguously understand the intent of the question. Ambiguous or advanced wording can lead to the child becoming confused or misinterpreting the questions. A flawed understanding of the question will lead to a flawed answer. Thought must also be given to the exact domain of answers available to the children. A survey consisting of only short answer "what do you think" type questions will likely be time consuming and fatiguing to the children.

Read [33] outlines a series of methods known as the "Fun Toolkit". The Fun Toolkit contains three evaluative techniques for use in child-centered research projects: smileyometer, again-again table, and the fun sorter. The smileyometer is a five point Likert scale, where the numbers (one to five) have been replaced with smiley-faces displaying increasing degrees of happiness. This work made use of the smileyometer, while omitting the again-again table and the fun sorter, as these methods are primarily focused on drawing conclusions from multiple treatments.

This study made use of an updated version of the smileyometer. In its initial form, the smileyometer features a range of faces, starting at "awful" (a very unhappy face representing a “1”) and ending at “brilliant” (a very happy face representing a “5”). Hall [20] suggests that this particular ordering and representation of the smileyometer can lead to a skewing of the data, as children are more likely to select a smiley face over that of a frowning face. The original form of the smileyometer can be seen in figure 4.1.

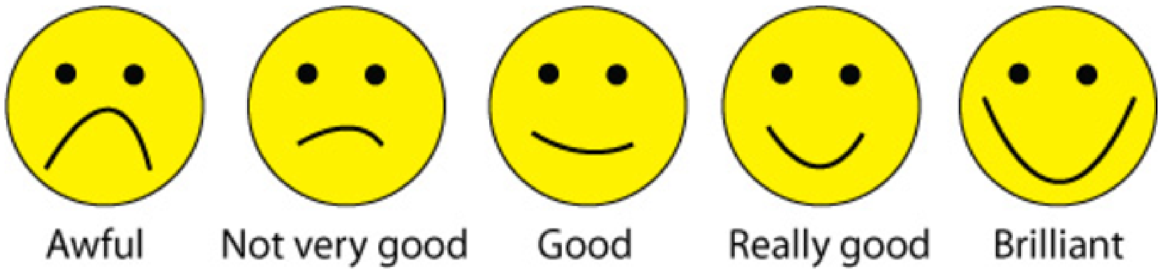


Figure 4.1: Original Smileyometer

To remedy this, Hall suggested a reworking of the smileyometer, in order to modernize it and to achieve a more uniform distribution of responses across the scale. Rather than start the scale at a decidedly negative emotion (awful), Hall started the scale at a more ambivalent emotion (ok). Oftentimes, children did not display strong negative emotion towards a poor user experience. In these cases, the center of the scale was favored, even though the application was specifically designed to incur negative responses. In this way, a middling emotion was chosen to be the negative end of the range. Also note that all faces in the new smileyometer depict smiles. Hall found that even with the same wording, children were not as likely to select an option that was nonplussed or displayed a slight frown. Figure 4.2 shows the final version of the new smileyometer.

In addition to the smileyometer, this study also made use of the “this-or-that” evaluation technique. This-or-that is a pairwise comparison among several different

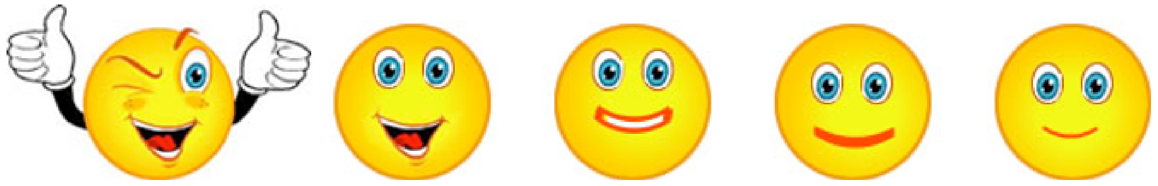


Figure 4.2: Improved Smileyometer

categories or queries. An example of a query could ask which option they favored (e.g., which application was most fun) or which option they disliked (e.g., which application was hard to use). This-or-that is useful for comparing two distinct entities or features. Sim [34] notes that when compared with other evaluative techniques, like the smileyometer, this-or-that lacks the ability to provide a nuanced or reasoned response. For this reason, this-or-that was limited to strict comparisons, which were backed by ancillary questions to provide triangulation for the children's thoughts.

The above techniques were used to develop a treatment survey and a comparative survey. Treatment surveys pertained to one version of the application. These surveys consisted entirely of smileyometer questions and were administered following the treatment. Treatment surveys were designed to be easily digested and completed quickly. Smileyometer questions were designed to require little cognitive effort on the child's part to avoid fatigue. Table 4.1 shows each question from the treatment survey. At the close of the study, the final comparative survey was administered. This survey contained a mix of this-or-that, smileyometer, and qualitative questions. Comparative survey questions may be found in table 4.2. Questions **CS1** through **CS4** were this-or-that questions, simply asking children to compare their experience with both versions of the application. Questions **CS5**, **CS6**, and **CS8** were qualitative, allowing children to expound on their feelings. Question **CS7** consisted of a smileyometer, asking children to gauge the amount of enjoyment they had during the study.

Table 4.1: Treatment Survey

Code	Query
TS1	Do you think adding a tag was...
TS2	Do you think adding a comment was...
TS3	Do you think looking at what other people did with the app was...
TS4	Do you think looking at other people's tags was...
TS5	Do you think looking at other people's comments was...
TS6	Did you connect and interact with others using the app?
TS7	Did seeing their tags help you connect with the other kids using the app?
TS8	Did seeing their comments help you connect with other kids using the app?

Table 4.2: Comparative Survey

Code	Query
CS1	Overall, which version of the app did you like better?
CS2	Which version of the app made it easier to see what other people were contributing?
CS3	Which version of the app made it easier to add comments?
CS4	Which version of the app made it easier to explore tags?
CS5	What parts of the app helped you connect with others?
CS6	What parts of the app made it hard to connect with others?
CS7	How fun or boring was the app?
CS8	What made the app fun or boring?

Investigators provided assistance where needed, especially if a child required help writing during the qualitative questions. Investigators also carefully read and explained the meaning of each question, in order to elicit an accurate response from the child.

Experimental Design

With the artifacts and evaluation methods developed, researchers were still torn on whether the treatments were to be administered in a between-subject or within-subject fashion. A within-subject study would allow users to provide direct comparisons between the two applications. At the same time, a between-subject study was likely to be more timely, and therefore make it more likely that a parent would consent to their children participating in the study.

In an evaluative sense, the between-subject is perhaps the less-desirable of the two options, yet it is not without benefits. Between-subject treatments would only require the participant to interact with a single version of the application. This would make it harder to collect interesting data, as the children could not compare their experiences with the two applications. Rather comparisons would have to be drawn from the disparate data. Furthermore, this variety of study would require an increased number of participants in order to achieve the same treatment coverage as a within-subject study. On the other hand, a between-subject study would likely allow for faster, more iterative study sessions. Children would be exposed to only a single treatment, lessening the likelihood that the child would become fatigued during the study. It was estimated that a single treatment would require approximately 25 minutes, this figure includes the time it takes to complete the tasks listed above, as well as the surveys. This would be a more palatable request to parents passing through the area, as they would be less likely to agree to a substantial time commitment off-hand.

The data extracted from this study was be primarily comparative, as the two applications in question needed to be contrasted in order to ascertain which application allowed users to interact socially. A within-subject study would make the data easier to reconcile to this conclusion, because users would be made aware of both versions of the application. It would give more context to their thoughts. The limiting factor for this type of study would be the amount of time required to administer each treatment of the study. Surveys would be administered at the close of each treatment, in order to solidify children's thoughts before interacting with the next application. After the second treatment survey, the comparative survey would be administered, asking the children to compare and contrast the two applications, as well as what they thought of the process overall. All of these artifacts and tasks would likely require forty-five to sixty minutes. It would be a much less desirable request to parents in the area. Due to the comparative nature of the study, investigators greatly preferred a within-subject study over a between-subject study. To combat the lengthy study time, participants were compensated monetarily for their time. An amount of $10.00 in the form of an Amazon gift card was given to each child participant. Investigators felt that this was a competitive offering for the time and amount of work required.

Within-subject studies are prone to incurring an "order bias", whereby participants exhibit a preferential bias depending on which treatment was first administered. To avoid this bias, treatments were administered to participants in a counter-balanced fashion. So the mapping between adventures and applications versions would change between treatments. Given two versions of the application and two adventures, there are four pairings of application-to-adventure treatments. Each one is given a reference code. Each pairing was visited session-by-session in a round robin fashion. The first session utilized session "A", the second session utilized "B", and so on. Table 4.3 demonstrates this counterbalancing scheme. Please note that "application" is shortened to "app" in the table.

Study Location

In choosing a location for the user study, a number of aspects had to be taken into account to ensure that the environment was suitable for children. Firstly, the location must be public or similarly trustworthy. Initially, the user study was meant to be con-

Table 4.3: Treatment Counterbalance Scheme

	First		Second		
Session Code	Adventure	App	Adventure	App	Count
A	Find	Z	Create	X	8
B	Create	X	Find	Z	7
C	Find	X	Create	Z	8
D	Create	Z	Find	X	7

ducted on Boise State University property, in the Human Computer Interaction Lab. Boise State University is a trustworthy institution where parents would understand that their children are safe. However, due to unforeseen complications, the study would need to be opened up to additional participants. Certainly, the best place to find willing participants would be in a public, outdoor setting, which could also serve as the recruitment ground and environment for the study. Outdoor areas would need to be open and public to guarantee parents that their children would be safe. Care would be taken to ensure that “tagged” areas were safe, easily reachable, and highly visible from multiple vantage points. Children would not be taken into enclosed or obscured areas. Additionally, prospective locations should contain a number of interesting environmental features to discover and explore. Though a soccer field may be open and public, it provides few opportunities for creating interesting tags.

The initial location chosen for this study was the Grove Plaza. The Grove Plaza contains many interesting cultural and historical locations which would make for engaging adventures and tags. Additionally, the Grove Plaza is located near the Boise State University campus, providing support in case difficulties were encountered during the study. However, concerns were expressed that the Grove Plaza might be an unsuitable location, as oftentimes visitors to the Grove Plaza do not expect to spend an extended amount of time there. Alternate locations were surveyed, including several public parks. Public parks featuring playgrounds were a particularly good fit for these criteria. Several potential parks in the Boise area were surveyed, including Settler's Park, Story Park, Kleiner Park, and Ann Morrison Park

IRB Application

This study involves child subjects, therefore extra precautions and considerations were needed to ensure IRB approval. The study was conducted in a public, open area with many witnesses. Participating children were accompanied at all times by study personnel. Study personnel were also instructed to help children where needed, but to not direct the children in their tasks. The child was the driving force when moving from location to location. Additionally, parents were encouraged to accompany the children, if they felt their presence was required. Children would also be informed that they were not obligated to participate in the study and encouraged to simply tell an investigator or their parents if the they became uncomfortable at any time. Children were free to opt-out of the study at any time and for any reason.

At the commencement of each session, the parent or parents were furnished with a consent form detailing the expectations of the study. Consent forms contained sections for the child's name, age, and gender. In collecting data, the child's name will be omitted entirely, the remaining fields will only be utilized in gathering demographic data. Once the parent gave consent for their children to participate, the children were asked for verbal assent concerning their participation. If the child did not wish to give their assent, then they would not be obligated to participate in the study. Upon granting assent, the child would be given an Android device running the treatment-specific version of the Geotagger application. User accounts have been previously created and do not include any data relating to the child.

Furthermore, in an effort to attract potential participants, investigators offered parents one $10.00 Amazon gift card for every child that agreed to and completed the study. These gift cards were intended for the children, but were furnished to the parents for safekeeping.

These provisions were submitted to the IRB in the form of an Expedited Review application. The application was accepted and assigned the IRB Protocol Number: `131-SB17-101`.

4.1.2 Study 0: Grove Plaza Pilot Study

An initial study was conducted in the Grove Plaza, in Boise, Idaho. The objective of this study was to observe how participants interacted with each other, the application, and study investigators. Data gathered from this study session was unusable, however this study afforded investigators a clearer idea as to the difficulties that might be encountered during a larger study.

Approach

Following the counterbalancing scheme seen in table 4.3, two adventures were created in preparation for this study. The Grove Plaza is located in the center of the downtown Boise area. As such, it is near interesting historic sites, as well as unusually shaped buildings. This location would be the basis for two adventures: “Buildings of the Grove Plaza” and “History in the Grove Plaza”. “Buildings of the Grove Plaza” focused on large and unusual buildings in the downtown area. These tags asked children to identify the uses of these buildings and the businesses housed in them. “History in the Grove Plaza” explored local history and culture, as children were asked to identify key points in Idaho’s history and what made these events significant.

Three tags were created for each adventure. These initial tags served as a starting point for the children. The intent was to include several obvious locations, while purposefully omitting others. This interested the children as they felt like they had discovered the locations of the starting tags and then turned to explore new tags on their own. The descriptions of each tag provided little information, instead they acted as discussion starters with questions such as: "What would you want to go to this building for?" and "What company is in this building and what do they do?". Children were expected to provide their own thoughts to these questions, as well as pose some questions of their own. This would provide a foundation of discussion and interaction within the application.

Investigators stationed themselves at the center of the plaza and made requests passersby for participation in the study. Special attention was paid to groups containing a legal guardian and three children. Few groups of people fit this description and only a single group consented to the study. This group contained three children between the ages of six and twelve years.

The children were each separated and accompanied by an investigator. The children were given Android devices with the original version (Geotagger X) and were asked to lead investigators through the plaza, while finding tags, adding comments, and creating tags. Fifteen minutes was allotted for this treatment of the study. When time expired, investigators led the children to a table and asked them to complete the first treatment survey. Children completed the surveys in isolation from one another, in order to avoid a cross-contamination of ideas. Once complete, investigators changed the Geotagger application (Geotagger Z) and started the second adventure. The children were given another fifteen minutes to explore this adventure. Help was provided if the children became frustrated, needed assistance typing or thinking of a word. At the close of the second fifteen minute period, the children were again taken to a table and asked to complete the second treatment survey. Immediately following the completion of the second survey, children were given the comparative survey. The exit survey signaled the end of the treatment, so the legal guardians were furnished with one $10.00 Amazon gift card for each child that completed the study and thanked for their time.

Results and Discussion

Due to the size and scope of this session, the data acquired during this session was mostly anecdotal in nature, relying upon investigators observations. This information was used to make changes to the approach in the proceeding study. This study provided a base for how children interacted with both versions of the Geotagger application and gave investigators an idea as to how subsequent sessions might unfold.

Investigators noted that children seemed to enjoy themselves during both treatments of the session and that finding other tags was a favored activity. Children were especially excited at the prospect that the other children could see and respond to their contributions. When the children were reunited at the close of the study phrases like “Did you see my tag?” and “I saw your comment!” were exchanged between the children.

Children were quick to comment on and discuss tags, though the discussion sometimes took place outside of the application. In this study the participants were siblings, so they were all very familiar and comfortable with each other. This is likely the cause for some of this impromptu discussion, as children would sometimes walk over to their sibling to discuss a tag or a comment. However, children became very excited when they realized that others could view and comment on their tags.

Participants made little use of the profile pages. Even with the anonymized usernames, children had little difficulty identifying which of their group-mates created a particular tag. This was most likely due to the small group size and participant familiarity with one another. Furthermore, time constraints led to children focusing their time on finding tags created by others, leaving comments, and adding their own tags, without room for organic exploration of the application. Map screens saw limited use. Children seemed to focus primarily on just the adventure tags and tag comments screens. These two views held the primary workflow for the study and the brisk pace of the study did not incentivize navigation to other screens. In the case of the enhanced application, the chat window was used only sparingly. Children were highly receptive to the chat window, but this was not a part of the three primary goals of the study and it was largely unused.

Limitations

This study was intended to function as a pilot study from which the approach could be scaled. No generalized data was taken from this study as participants only consisted of three children in total. A subsequent study with more participants was conducted to provide more empirical results.

Difficulties were encountered when administering surveys. Members of the research team were distributed throughout the Grove Plaza, yet the transition between the study treatments had to be synchronized. During this session, treatments were slightly staggered due to some miscommunication. Later sessions involved a single investigator timing each session with a stopwatch, then informing other investigators via text message.

Researchers found that the Grove Plaza was an inappropriate location to conduct this study. The Grove Plaza was well populated, but groups of people containing children were somewhat seldom. Furthermore, most visitors to the plaza only intended to spend a limited amount of time there or were in transit to a different location. Many people that were approached said that they did not have time to participate in the study. In order to bring in more participants, it was decided that the study be moved to a playground area in public park area. Visitors to playgrounds already have an expectation that they will be spending an extended period of time there. Playgrounds are usually very public and open, which allowed parents to watch their children at all times during the study.

4.1.3 Study 1: Settler's Park Study

In response to the difficulties encountered during the Grove Plaza study, a new study was devised that allowed investigators to find prospective participants much more quickly. Investigating local parks used the same criteria as before, yet with an emphasis on network connectivity. A data plan was procured so that the devices could connect to the Internet while at the park. Yet some parks experienced data coverage issues, requiring the application to be tested on-site before the park was considered a suitable location. Ann Morrison Park, in particular, had poor data connection and was not considered for this study, though it successfully met all other criteria.

Settler's Park was eventually chosen as the replacement location, containing a large play structure and water park area. This park was also very popular during the late morning and early afternoon. This study took place during August, so children were out of school for summer vacation. However, hot weather conditions drove parents and children from the park as the afternoon progressed. Study activities largely took place between 9:00 AM and 2:00 PM before the weather became too uncomfortable.

Approach

Adventures centered around the playground and the park, tasking kids to rely on their imagination. “Create Your Own Park” instructed children to visit existing playground structures and “tag” other structures, detailing how they would change or incorporate them into their own park. “Find and Imagine” involved the children finding and creating tags with fantastical elements. Pirates, explorers, and castles were the subjects of the adventure and children were tasked with filling in the blanks to create their own fantastical scenarios. Three tags were created for each adventure. These tags were very conspicuous and usually recognized almost immediately. Similar prompts were provided for each tag and the children were invited to comment on a prompt, if they felt they had an interesting answer.

In preparation for the larger study, a logging mechanism was added to both versions of the application. This logging feature was created to monitor a users activity with the application. The logging module was responsible for time-stamping each action taken in the application. This included creation, updating, and deletion of Geotagger entities (primarily tags and comments), as well as any textual data that might help identify the specific record in the store. View changes were also logged in an effort to track how much time users were spending on each view. Time-stamps served to track user activity and action frequency. This data served to bolster the data taken from the surveys and questionnaire.

Willing participants were difficult to find at first, as many children at the playground were below the age of six years, or did not have siblings who could participate in the study. Due to the lack of participants, an alternate means of proposition was pursued. An advertisement was posted on “Nextdoor”, a neighborhood oriented social media network. This advertisement informed homeowners of the time requirements of the study, as well as the monetary incentive. A link was also provided to the IRB approved consent form. This advertisement proved to be very effective at informing potential participants, eschewing the need to further proposition park-goers.

Much of the approach for this section mirrored that of the previous study. Treatments were allotted fifteen minutes each, allowing five minutes following each treatment for the completion of the treatment survey. Five more minutes was allowed to complete the final comparative survey. These artifacts remained unchanged from the previous study, though an effort was made to streamline the treatments and the surveys.

Results and Discussion

At the close of the study, researchers collected participant surveys and log files generated by the application. Survey questions which involved Likert scales (e.g., smileyometer questions) were coded from one to five. One being the least favorable answer and five being the most favorable. Anonymized surveys were cross-referenced with consent forms in order to gather demographic data (e.g., age and gender). Children’s names were not used in the analysis of this data.

A total of thirty children between the ages of five and twelve years participated in this study. This study took place over the course of twelve sessions consisting of groups of two or three children. Groups of two participants were considered in the interest of time, though these sessions were greatly disfavored compared to sessions of three children. Session treatments followed the round-robin counterbalancing scheme described in table 4.3. Toward the close of the study, the round-robin scheme was changed slightly, so that each treatment type received an even distribution of participants. Table 4.4 details the coverage of each session type.

The first metric analyzed was children's preference of one application over the other application. These social enhancements were designed to encourage continued contribution and make the overall user experience more enjoyable. Researchers expected children to greatly prefer the usage of Geotagger Z over Geotagger X. However, the comparative survey showed children did not show strong feelings towards one or the other. Ultimately, children's preferences between the two applications were extremely similar. Results taken from **CS1** were largely inconclusive in demonstrating which of the two applications the children favored. Of the thirty children surveyed, 9 preferred Geotagger X, 14 preferred Geotagger Z, and 7 children were undecided in choosing between the two. Though children favored Geotagger Z over Geotagger X, this is not by a very large margin. This information coupled with the proportionally significant number of undecided answers (~23% of participants) indicates that children likely had difficulties discerning between the two, or at the very least, felt the applications were very similar.

Participant gender was greatly skewed towards males as this study consisted of 7 females and 23 males. Most of the children playing in and around the park were male and this same bias was reflected in the participants generated from the Nextdoor advertisement. Further studies should take care in creating a more balanced gender distribution. However, this finding did not present an obvious correlation between gender and application preference. Of the 7 girls surveyed, 4 stated that they preferred the original application (Geotagger X) over that of the enhanced application. The other 3 stating that they could not decide between the two.

Table 4.4: Participant Age Distribution

Age	Count
5	2
6	3
7	6
8	3
9	7
10	3
11	5
12	1

The treatment surveys also showed fairly even or neutral feelings between the applications. Researchers aggregated the smileyometer questions from the treatment surveys. Figure 4.3 shows the median responses given by children in each of the treatment surveys. Note that the x-axis represents each question in the treatment survey. Responses between the two applications were extremely similar. Geotagger Z had higher median answers for **TS3** and **TS5**. Question **TS5** in particular asks about discovering other member's comments, a question designed to evaluate the effectiveness of **SE1**. This may be related to how accessing comments in the enhanced application required less button presses than the original application. Geotagger X was more favored in question **TS7**, which asked about exploring the tags of others. This may be related to the more compact look of the tag list item in the original application, allowing more tags to be visible on the screen. In any case, the differences in the median responses for these questions was too minuscule to draw concrete conclusions.

In addition to the median responses, mean responses were also calculated, though

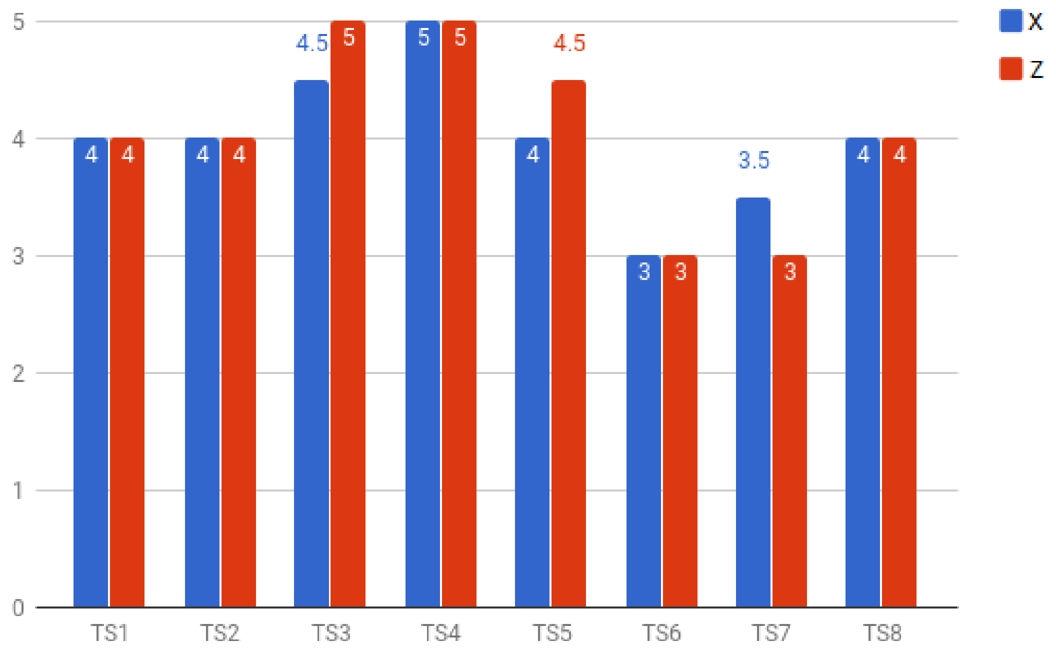


Figure 4.3: Treatment Survey Median Response

the results were extremely similar to those found in the figure 4.3. Children's responses were again extremely similar between the two applications. The mean graph shown in figure 4.4 shows slight differences in the average response for each question, yet these values are extremely close and do not vary greater than a value of 0.5. Graphs for both mean and median responses show that **TS6** and **TS7** are the questions which had the lowest responses. Both of these questions ask how children felt they connected with one another. This lower average response is probably due to the small session size; the largest of which was just three children. This aspect combined with the relatively short amount of time children had to adjust to the applications was likely the cause for the lower response rate in how children interacted with one another.

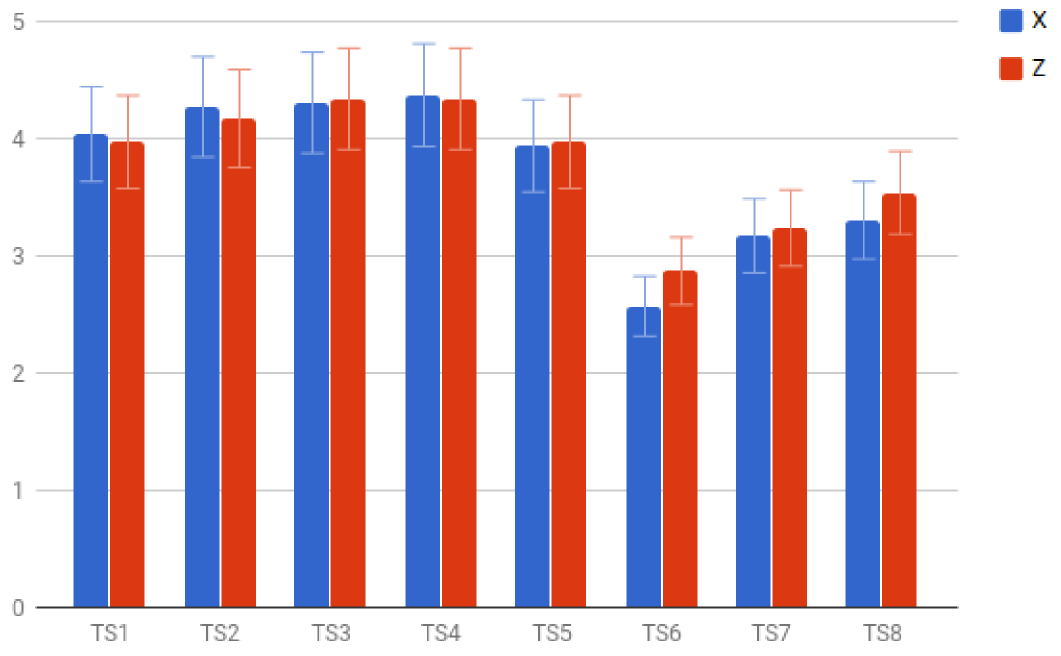


Figure 4.4: Treatment Survey Mean Response

The concrete data collected from the study was unexpected and did not draw a strong correlation between the presence of social enhancements and increased user interaction. Participants did not show a strong preference toward one application over the other, though the enhanced application garnered slightly more favorable responses. Due to the unexpected data returned by this study, further investigations were made to more directly target specific social enhancements in an attempt to study their social effectiveness.

Limitations

This study experienced several issues that prevented the resulting data from being strongly unbiased. Results collected from the data were unexpected and did not provide a clear correlation between user preference and the introduction of social enhancements. This problem may have been caused by the scope of this study, which was too broad and tried to accomplish too much. Users were unrestricted in their use of Geotagger, as social enhancements are interspersed throughout the whole application. This made it difficult to pinpoint the effectiveness of any one social enhancement, instead it relied on the enhancements as a whole. The abundance of features and lack of time led to the full functionality of features not being exercised properly. Logging systems were included in order to monitor user activity and identify those enhancements which were most used. However, the breadth of the application, combined with the limited time available to the study, made it difficult to delineate the effectiveness of each enhancement.

Session size and user availability was of particular concern during this study. The size of the session group would affect the number of tags and comments created during the session. Time constraints and participant availability forced the research team to accept groups when just two participants were available. Groups with two participants created markedly less tags and comments, though the average number of comments and tags created per user was comparable to groups of three. This provided fewer opportunities for interaction between participants, rendering the social enhancements ineffectual. Ideally, a user study would involve a group size closer to that of a classroom in order to have more potential for user interaction. Likewise, groups of three people were thought to be too small to foster social interaction between users. Available resources made it impossible to increase the group size beyond this study. In conclusion, the lack of variety of participants to interact with likely decreased the amount of interaction between participants.

Environmental elements also posed a problem in conducting this study. The geographical nature of Geotagger dictated that a user study take place outdoors. This study was conducted during the first week of August at a particularly hot time of the year for this area. Children and researchers often experienced fatigue as the day progressed, especially during the later stages of the day. This variable may have had an impact on how children interacted with the application or when it was time to test the second application. Fatigue from the heat may have affected user enthusiasm, causing them to participate less effectively than they would have in a more comfortable climate.

Logging and data collection were implemented during a later stage of this project. Effort was made to revise how activity was logged in order to provide finer grain reporting on how users were interacting with the application. Unfortunately, this added logging merely showed how much time users spent on each view. It was largely unsupported by follow-up questions in the treatment and comparative surveys. Some values taken from the logs showed a very large amount of time spent on certain screens without creating further content. This was likely caused by the child locking the device or simply walking around with that particular screen open. The comparative survey was a source of problems, as "this-or-that" questions left little room for little nuance or explanation. The comparative survey contained four such questions, asking children which version of the application was preferred for different aspects of the application. Additional or revised questions were needed to help reconcile this data and the data collected from the questionnaire. This would have helped to explain children's reasoning as to why they preferred one aspect of an application over another.

Participants in this study occupied an age range between ages of six and twelve years. While many children in this day and age have been exposed to mobile technology at some point, it is likely that they do not have the expertise or the critical skills needed to analyze the differences between the two applications. On the surface, both applications appear to be very similar. This was likely exacerbated by the fact that children were asked to perform identical tasks between the two applications. In this way, the changes between the two applications may have been too nuanced for the study age range. They may be more suitable for an older, more technologically experienced user-base.

To facilitate interaction through the application, participants were divided amongst investigative personnel. This was done to ensure the safety of participants, as well as allow an opportunity for investigators to suggest areas of the park that were not occupied by other study participants. Investigators attempted to ensure that participants worked on their own, but the modest size of the park led to unavoidable verbal interactions between participants. Investigators observed that children were enthusiastic to talk with one another about their experience with the application. These types of external interactions were to be avoided as the aim of this study was to investigate the ways children interacted using the application. These external collaborations may have effected the results of this study as children interacted verbally, rather than using the application.

In designing this study, investigators worked to maximize the amount of time children spent with each version of the application. Allowing for too much time might have caused children to become disinterested in the application and led parents to become frustrated with the protracted pace of the study. Ultimately, fifteen minutes was allotted for each treatment, though this time may not have been sufficient to fully exercise the functionality present in the application. Furthermore, participants were administered treatments in quick succession, allowing them little time for reflection.

This lack of contemplation between both versions may have led to confusion during the comparative survey (table 4.2).

Child participants require extra considerations compared to adult participants. Children are cognitively and emotionally different from adults. Obtaining user feedback from children is more involved as these aspects must be taken into account. Preceding this study, investigators researched methods by which children's thoughts and opinions could be accurately elicited. To that effect, this study made use of the "smileyometer" and "this-or-that" survey techniques [32]. These techniques are designed to provide children with a simple means for communicating ideas. Due to the some of the nuanced changes in this application, a survey may have been an inappropriate method in evaluating enhancement effectiveness. Children may have had more to say than what was prompted in the survey. Surveys provided the most efficient means of gather user data, which is a useful feature due to the time sensitive nature of this study. Yet the rigidity of the questions leaves little room for supplementary ideas. This study may have been improved with open-ended evaluation techniques that allow for children to express their ideas more naturally.

The application developed in this work is by no means free of software defects. As such, several bugs were encountered during the course of this study. Thankfully, these defects were encountered only very occasionally, and were usually remedied by simply restarting the application. A "panic" button was made available to researchers, in the event that any serious difficulties were encountered. This button simply cleared the local cache and restored the application to its initial state. This whole process would take the investigator approximately fifteen seconds and having likely little impact on the child's interaction with the application. Faults were seldom encountered and no fault was found to stop the session completely. Application difficulties likely had little effect on survey results, but it is worth noting just the same.

Tags are usually furnished with a GPS coordinate consisting of a longitude value and a latitude value. This allows other users to exactly pin-point the tag's location and make their way to it. When adding a tag, a user is able to automatically call the GPS system which provides the user with their current coordinate location. However, the remote location of this study interfered with the GPS found on the Android devices. This issue caused automatically positioned tags to place map markers on the nearest street addresses which could be as much as one hundred yards away, making the adventure and tag map views wildly inaccurate. This difficulty was discovered by researchers when creating the initial set of tags, a problem that was not present during the Grove Plaza study (see section 4.1.2). To avoid confusing the participants, researchers manually set each tag's GPS location. However, this could not be done for tags created by study participants, so participants were informed that tags placed outside of the map were inaccurate. Fortunately, children exclusively added tags with images. Due to the distinct appearance of structures within the park, children were able to find other's tags almost entirely without issue.

4.1.4 Results and Discussion

Results in this study were rather unexpected and did not serve to highlight the strengths and impact of the proposed social enhancements. Participants in this study were split in their preference of one application over the other. Though Geotagger Z was somewhat preferred over Geotagger X, a large number of indecisive answers likely show that children were confused, or could otherwise not readily distinguish the applications. Limited treatment time led to tasks feeling rushed and a diminished amount of social interaction taking place. The decision to study the application as a whole weakened the correlation between user interaction and the presence of social enhancements. This was only exacerbated by the fact that the largest group consisted of three participants that created few opportunities for social interaction.

4.2 Investigations of Specific Social Enhancements

An additional round of investigations were conducted to research the social enhancements in isolation. These investigations were conducted in response to the somewhat confusing and muted results discovered in the initial study. These investigations were designed to provide more focused results. The obscured results from the first study were found to be caused by the unwieldy and sizable scope of the study. Further inquiry provided a smaller, more concentrated view of individual enhancements, contrasting with the initial study's survey of the application as a whole. Third party participants were not utilized for this investigation, rather members of the Kidsteam design group participants for this study. This study took place in two stages, with each stage focusing on a different social enhancement. These social enhancements included:

- SE1 - Making comments more prevalent
- SE3 - Creating a shared social space for adventures

The inherent differences between these two enhancements called for different approaches. In this way, specialized examinations were constructed in order to take advantages of the strengths of each enhancement. Each inquiry was designed to give participants a more precise set of tasks that would highlight differences between the two versions of the application. Children could then evaluate these differences on their own, then draw conclusions and make suggestions where needed. These evaluations were captured by lab personnel and used to draw conclusions concerning the effectiveness of these enhancements.

4.2.1 Method

This investigation did not fit as neatly into two distinct treatments as did the previous study. Application enhancements were studied on a case-by-case basis using multiple treatments where required. Tasks did not focus on real-world use cases for the application. This investigation focused on children directly interacting with the social enhancements. They provided feedback on features they liked or disliked, as well as other ideas they had for the application. Evaluative artifacts took on a very different form due to the size of the participant group. This study took place in the Human Computer Interaction lab located at Boise State University. Parents were initially invited to the lab space and laboratory personnel were on-hand to ensure parents of their children's safety. Furthermore, an amended application was submitted to the IRB reflecting the changes to personnel and environment.

Kidsteam Design Group

The Kidsteam design group is an intergenerational technology design team that uses cooperative inquiry to elicit new and innovative ways of creating and interacting with technology. Kidsteam personnel work closely with children to research, model, and implement technologies targeted at child users. Children are often marginalized when it comes to providing informative feedback, yet they can offer a unique point of view which may not be immediately obvious to adult users [11]. This unique perspective can give rise to interesting and effective user interfaces which would otherwise not be possible. Participatory design techniques provide a means for children to express their thoughts. Oftentimes these techniques involve hands-on design work consisting of art and craft activities with illustrative group discussions [13]. These techniques are designed to allow children to express ideas other than verbally, as children may not be well spoken at such a young age. Techniques include: *Bags of Stuff*, *Sticky Notes*, *Journals*, *Mixing Ideas*, and *Layered Elaboration*. These techniques are designed to create avenues of *idea elaboration* between the different age demographics present in the group. Many of these techniques are employed directly in investigations involving the Kidsteam group.

Tasks

A major shortcoming of the previous study is that it tried to evaluate all of the social enhancements at once. The social enhancements implemented during this work are found throughout the entirety of the application. With limited time for each treatment, there was not enough data taken from the study to draw concrete conclusions on any one enhancement's effectiveness. This investigation instead focused on individual social enhancements. Feedback was taken from children who were directly interacting with a small number of systems in the application. In this way, children were instructed to only utilize very specific aspects of the Geotagger application. This focus would allow researchers to directly draw conclusions concerning these enhancements, providing a clearer, albeit smaller, picture of how these enhancements affected user interaction and discussion. This would not provide a mirror to real-world usage of the application, yet it would provide targeted insight into how children viewed and interacted with the application.

Evaluative Artifacts

Evaluative artifacts took on a very different form for this investigation. Previously, surveys were used to gather data from children in an effort to solidify their thoughts about the application. These surveys attempted to gather children's thoughts about multiple aspects of the application, where as this study focused few aspects to achieve a finer grain of detail. This fine-grained look would be better suited toward a more expressive and creative form of evaluation. Surveys were also better suited to large groups of participants where statistically significant data could be taken from these surveys. However, due to the small group size, data taken from this study could not be generalized to other children. Nevertheless descriptive information could still be drawn from this study.

In order to effectively leverage the Kidsteam design group, a different evaluation technique was required. Knudzton [25] noted the importance of "cooperative inquiry activities" that allow children to come up with ideas and solidify thoughts as a group. These cooperative activities can take many different forms to serve many different purposes. This includes critiquing existing software and using three-dimensional objects to prototype new or future software. Whatever the activity, these methods are usually very hands-on and require creative thinking. As the Geotagger application was already in a finished state prior to this study, an evaluation method was chosen that would allow children to focus on specific aspects of the existing application. This study employed cooperative inquiry techniques known as "sticky notes", "big paper", and "big ideas".

Sticky notes [13, 26] is an evaluation technique that provides an avenue for children to share their thoughts through creative and physical means. Children are paired with team members and are supplied with "sticky notes" and a pen or pencil. Ideas are recorded on sticky notes as the children interact with the technology. It is important to note that each note represents a single idea, so multiple ideas should not be combined on one note. This approach enables the notes to be collected and arranged on a whiteboard to be categorized. These ideas are then grouped by subject matter using a marker. This process allows researchers to draw common themes and insight from the exercise. Figure 4.5 gives an example of a completed sticky note exercise.

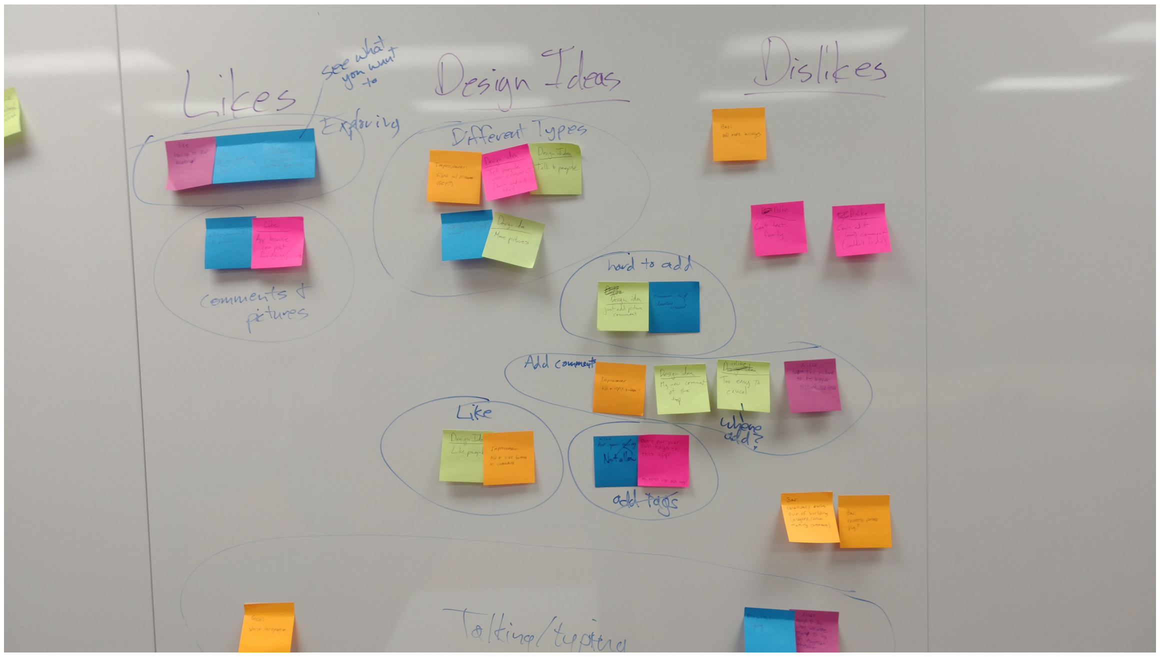


Figure 4.5: Sticky Notes Cooperative Inquiry

The big paper [17, 39] evaluation technique gives children the opportunity to build a low-tech prototype from the ground up. Children are divided into small groups and given a large piece of paper and colored markers. They are then supplied with a limited amount of time and an objective to redesign some technology in any way they see fit. Children then create a mock drawing of their version of the technology. Once time is called, the groups come together to discuss their creations. This technique is low-cost and requires very few supplies, yet it has the potential for generating rich intra-group discussion during the activity and inter-group discussion following the activity.

Big ideas [26, 13] is an evaluation technique that allows children time to verbally share their ideas with the rest of the group. Big ideas often takes place at the end of an activity and provides an opportunity for retrospective on what was accomplished during the activity. Oftentimes these ideas are recorded on a whiteboard or other large, centralized surface that is easily read by all participants. Children can monitor the recording of their ideas and make corrections where needed. Once complete, investigators note commonalities or similar themes that occur between various observations. Big ideas can show where development effort needs to be targeted or where further testing and analysis may be required.

Study Location

Evaluative portions of these investigations took place in the Human Computer Interaction lab at Boise State University. The initial session of Kidsteam allowed parents to become better acquainted with the lab environment and lab personnel that would be interacting with their children during the design sessions. The HCI lab is equipped with computers and other tools used in learning and technology design. Additionally, the HCI lab is an open space with many windows providing an unobstructed view of lab activity. Children were safe at all times and were asked to inform personnel if they ever became uncomfortable.

Practical portions of this study took place in the Grove Plaza in Boise, Idaho. As previously noted, the Grove Plaza provides many opportunities to create interesting tags and explore local culture. Due to the colder fall weather, parents were informed in advance of outdoor activities to provide their children with warm clothing. During the activities, children were paired with lab personnel and supervised at all times.

IRB Application

A new application was submitted to the Institutional Review Board in preparation for Kidsteam design sessions. All investigators that worked directly with children are recorded as personnel on the IRB approved application. This application was accepted and assigned the IRB Protocol Number: `131-SB17-027`.

4.2.2 Investigation 1: Tag Cards and Comment Flow

The objective of this aspect study was to identify effective and ineffective UI elements for comments and tag cards. Attention was paid to how users created comments with the application and what sort of tag-related discussion this caused. This investigation also focused on how users navigated between tags and how tag information was organized, as well as study the effect of user interface choices that attempted to make comments more prevalent (**SE1**). This investigation required that children be exposed to both versions of the application so they could make comparisons between. The study took place in two stages with evaluations following each stage.

Approach

To prepare for this study, two adventures were created which would be used in concert with the two different application versions. The “Buildings of the Grove Plaza” and “History in the Grove Plaza” adventures were reused, as they already contained a number of interesting tags. Tags were left mostly unchanged, except for a few minor updates that were made to the tag descriptions to allow for further discussion. Creating tags was not the focus of this study, so study activities hinged on participants discussing the provided tags.

Prior to any study activities taking place, children were oriented as to the first version of the application (Geotagger X). A short demonstration was given as to how tags operated and how the children could leave comments on tags. Children were then instructed to focus on leaving comments on the provided tags, responding to prompts given in the tag description, as well as comments left by their peers. As before, children were paired with lab personnel to facilitate interaction exclusively through the application. The participants were then lead outside and the practical portion of the study began in earnest.

Children led lab personnel between tags in a fashion similar to the previous study. Investigators provided support to children when prompted, but served to primarily observe and write down the child's thoughts. These thoughts were recorded on colored "sticky notes", where the color of each note corresponded to a different child. These thoughts were collected and divided into three categories: **Likes**, **Dislikes**, and **Design Ideas**. After collecting the initial thoughts recorded by researchers, the children were asked to record their own ideas fill in any gaps. These were further grouped by the subject of the thought, allowing researchers to better organize the participant's opinions.

This process was repeated for the enhanced application: Geotagger Z. To diversify their experience with the application, children were supplied with a different adventure with which to find tags. As in the first adventure, sticky notes were recorded and categorized by investigators at the close of the adventure.

Results and Discussion

Due to time constraints, this study took place over two sessions of the Kidsteam design group. Both sessions consisted of five children and allotted approximately twenty minutes for each adventure. The amount of time it took to conduct the adventures, evaluate the sticky notes, and reflect in journals was too much for a single Kidsteam session. Thus, adventures took place on two different days during the same week. Figure 4.6 depicts the results of the sticky notes session for the first adventure.

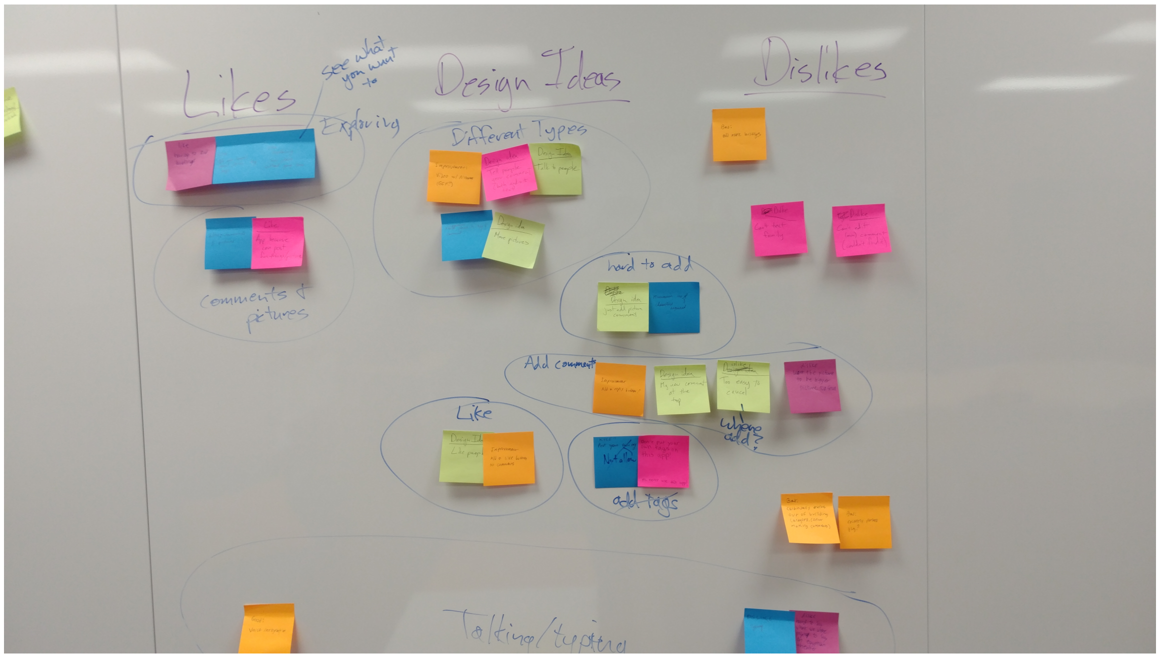


Figure 4.6: Geotagger X Sticky Notes

Geotagger X was the subject of the first adventure, "Buildings of the Grove Plaza". After collecting notes from Kidsteam members, investigators noted several similarities between them. Children especially liked exploring the environment for tags as did the participants of the previous study found in section 4.1). The Kidsteam group also liked the idea of tags and being able to post comments about "fun things" and attach images to them. However, children wanted to extend this function with additional media types that could also be attached to tags. Also, participants felt that adding a comment was frustrating, because it was too easy to cancel the comment and lose your work. Furthermore, they wanted to be able to add a comment which consisted only of an image without the need for a textual comment.

This was Kidsteam's first experience with the Geotagger application and many of the sticky notes fall within *Design Ideas* and *Dislikes*. Children had many new extensions that they wanted to see added to the application and were quick to point out any frustrations they felt. Following this investigation, a second adventure was created with the express purpose of testing Geotagger Z in a similar manner. Geotagger Z was tested with the "History in the Grove Plaza" adventure and sticky notes were once again collected (figure 4.7).

Anecdotally, one can immediately see that Geotagger Z has a higher concentration of sticky notes in the *Likes* category. Several children liked the flow of adding a comment, saying that they liked how "you don't go into the tag". Many children were able to easily identify the difference in flow when it came to adding a comment. The goal of **SE1** is making comments more prevalent. A further "like" was the slight UI change to comment cards in Geotagger Z which removed the slide function in favor of an explicit options "chevron". Members also liked being able to immediately bring up a larger version of a tag image by pressing on the image present on the card. Note that Geotagger X required the user to navigate to the tag-detail view first. However, participants noted the small size of the comment input bar made it difficult to spell-check comments, as you could not see the entire comment at once. Some verbiage when editing a comment was also found to be unclear. "Update"

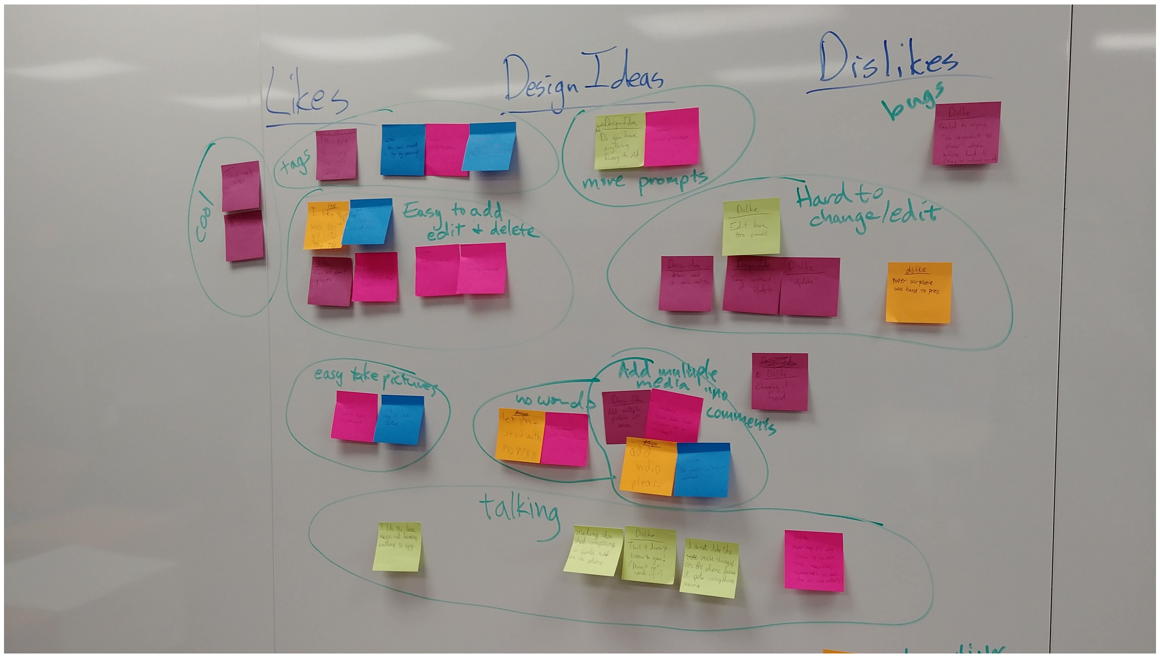


Figure 4.7: Geotagger Z Sticky Notes

was thought to be a confusing word. This issue coupled with a small input box made editing comments a frustrating experience to some members. Many *Design Ideas* were holdovers from the previous adventure, as image comments and additional media types made a repeat appearance.

At the close of the second treatment, children were administered a survey that asked them to compare the two applications. This survey was meant to provide an informal evaluation of the applications. Of the five children surveyed, one child had not been exposed to the Geotagger Z version of the application. So they could not make accurate comparisons between the two versions. Table 4.5 lists the questions found in the survey.

Many of these questions were meant to mirror the questions found in the comparative survey for the initial study (see table 4.2). However, extra steps were taken to

Table 4.5: Kidsteam Survey

Code	Query
KS1	Overall, which version of the app did you like better?
KS2	Which list of tags did you like better?
KS3	Which version of the app made it easier to get to comments?
KS4	Which version of the app made it easier to add comments?
KS5	What worked and what didn't work about the first app?
KS6	What worked and what didn't work about the second app?
KS7	Which list of tags did you like better and why?
KS8	How fun or boring was the app?
KS9	What made the app fun or boring?

ensure that the intent of these questions were clear to the children. Questions **KS1** through **KS4** were, once again, "this-or-that" questions. In addition, children were provided with full color printouts of each view that the question referred to, which gave children an unambiguous context for each question. Questions **KS5**, **KS6**, **KS7** and **KS9** were qualitative, and meant to provide a space for children to list any other thoughts they had about the applications. Question **KS8** is the sole smileyometer question, a hold-over from the previous survey.

Compared to previous surveys in this study, the results taken from this survey were much more demonstrative. Although these responses may not be generalized, they still provide compelling outcomes on a much smaller scale. Firstly, every child surveyed selected Geotagger Z as their preferred application in **KS1**, as well as **KS3**. Children enjoyed using the enhanced application more than the original application. They felt that the enhanced application made it easier to get to tag comments (e.g., part of the enhancement proposed in **SE3**). Investigators noted that several children identified the number of button presses as the cause for preferring Geotagger Z. Geotagger Z requires a single button press to navigate to comments, while Geotagger X requires several button presses. One child also commented that "the second one

(Z) was really much faster than the first one (X)".

Questions **KS2** and **KS4** saw 3 out of 5 children preferring Geotagger Z over Geotagger X. These questions were a little more indecisive, yet comments by participants reveal some interesting insights. Several children mentioned off-hand that they liked the smaller tag list item than the larger tag list card. This allowed for more tags to be visible on-screen at any one time in Geotagger Z. It was easier to get an informed interview of the adventure for this tag list, because list items require less screen space than cards. One participant mentioned that they liked how "tags and photos of places" were arranged for Geotagger X. Regarding comments, children mentioned that writing a comment in Geotagger Z was a little difficult, because the input box was smaller and more difficult to proofread. Geotagger X has a larger input box, but the proximity of this box to the "cancel" button caused problems for several participants. However, several children clearly indicated that it was "easier to add comments" with Geotagger Z.

Questions **KS5** and **KS6** served to fill in gaps and triangulate responses to other questions. They allowed children to express further thoughts that had not been covered elsewhere. One child liked how "you don't have to open the tag to read about it" and noted that "you have to go through three screens to comment (on version X)". Other participants said that Geotagger Z was "easier" and "faster" to comment.

This investigation was able to show that the Kidsteam design group preferred the enhanced Geotagger Z application to the original Geotagger X application. Geotagger Z received more favorable responses from the "sticky notes" session and the survey. Though only five children participated in this activity, this fine-grain look at tags and comments was able to focus children's attention and garner more interesting results.

This investigation spent twenty minutes on finding tags and creating comments alone, whereas the initial study spent fifteen minutes per application, and utilized both applications in the same day. This investigation focused on quality discussion around a small subset of Geotagger features over the course of two sessions. Narrowing the overall scope served to allow children to concentrate on just two main aspects. The focus led to children having more concrete, well formed thoughts as to their interactions with the application. The intention of this investigation was to take a closer look at the effect of more prevalent comments (**SE1**) on children's thoughts on the application. While this investigation was not able to measure the effect of the enhancement directly, participants were able to pinpoint this enhancement as a reason for preferring the enhanced application's comment-adding workflow over that of the original version. It was obvious that making comments more prevalent led to richer, more empowering application usage.

Limitations

The more focused, smaller scale of this investigation meant that results found during these sessions could not be generalized to other children in the same demographic. With a population of five, the results found here could only be used in a descriptive way, having no empirical value.

4.2.3 Investigation 2: Practical Adventure Chat

The previous investigation utilized past adventures in order to provide a real-world experience of the Geotagger application. Due to time constraints, the previous study focused very little on the adventure chat, opting instead to focus on tags and comments. This investigation focused on communicating almost exclusively through comments and sought to garner feedback from children on which application provided a better, more social user experience. Creating a shared social space for adventure members was a part of **SE2** and this investigation ssought to provide a more focused look on how children interact with this space. This investigation largely mirrored the workflow of the previous study (section 4.1), but distinguished itself in the evaluation portion. This investigation would require somewhat more creativity in order to effectively exercise and accurately reflect the usefulness of the adventure chat.

Approach

The adventure chat functionality is an all new feature to the Geotagger project. As such, there is no analogous functionality present in the original application (Geotagger X). The previous investigation furnished children with both versions of the application to allow children to draw comparisons between the two. This investigative technique would be inappropriate for this aspect. Children were furnished only with Geotagger Z, since it is the only version of the application that contained the adventure chat.

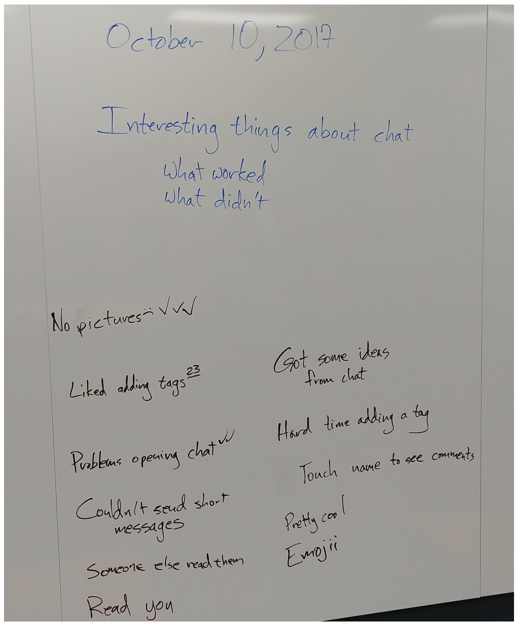
Playing once again to the strengths of the Geotagger project, this investigation revolved around a Geotagger adventure. Though only a single adventure was needed, care was taken to ensure that this adventure was easily distinguished from the other adventures. If a similar adventure was created, children may become bored dealing with the same tags and themes from previous Kidsteam sessions. This adventure, entitled “Signs of Fall”, focused on environmental features that indicated the arrival of the autumn season. Previous Kidsteam sessions focused on finding tags and commenting on them. Instead, this adventure focused on children creating their own tags and discussing the adventure as a whole using the shared chat area. Children were given the task of collectively creating several different tags which indicated the changing of the season. Investigators added some initial tags which covered the most obvious indicators (e.g., leaves changing color). This incentivized the children to communicate with one another using the adventure chat function.

Unlike the previous investigation, children were furnished with Geotagger accounts which contained their first name. This was done to allow children to better identify their peers who created interesting tags. Though profile images were not provided, children were able to identify the author of a tag by pressing the "user profile" button located on the tag card. Children were again paired with investigators to ensure child safety and provide support in case the children encountered difficulties with the application. Approximately thirty minutes was allotted for the children to navigate the Grove Plaza and create their own tags. During this time, children led investigators around the immediate area, while discussing with investigators and peers using the adventure chat function. Investigators helped guide discussion using devices of their own and pointing out duplicate tags when they arose.

Results and Discussion

Children seemed to respond very well to this investigation's activity. Child interest was held well throughout the activity and a good amount of discussion occurred in the chat screen. To evaluate these interactions, children's "big ideas" were written on a whiteboard. Each child was given an opportunity to express aspects they liked and disliked about the chat feature. Figure 4.8 shows the resulting ideas,

In the twenty minutes allotted for this investigation, children created a total of 23 tags and sent 19 messages. Initial tags centered on fairly overt signs of Fall weather. Topics such as dead or colored leaves were the main subjects. Then, toward the close



October 10, 2017

Interesting things about chat
What worked
What didn't

No pictures ✓✓✓

Liked adding tags 23

Problems opening chat ✓✓

Couldn't send short messages

Someone else read them

Read you

Got some ideas from chat

Hard time adding a tag

Touch name to see comments

Pretty cool

Emojii

Figure 4.8: Adventure Chat Big Ideas

of the study, the idea of Fall weather became more abstract and children started creating tags which did not have to do with natural features. Objects like Autumn sales, dates on signs, and Fall icons such as pumpkins became subjects for tags. Chat messages largely consisted of children discussing their ideas for further tags. Messages like "I am going to take pictures of fall weather", "We have added backpacks", and "We are making a tag about closing patios" alerted other participants of potential tag conflicts. Due to the coupling of tags and chat, children made considerable use of the chat badge, which alerts users (not currently in the chat) to the presence of new chat messages. Investigators observed multiple instances where children were looking at the tags list, then the chat badge popped up. At that time they would navigate to the chat to read the new messages.

Participants were very excited about the idea of the adventure chat, so ideas taken from this activity were generally focused on extending the existing functionality of the adventure chat. During this investigation, the chat only supported the transmission of text-based messages. Children were especially excited at the prospect of sending pictures in the chat. One key advantage of the chat function is that messages arrived in real-time. This contrasts with the comments which require manually sending HTTP requests in order to refresh the view using the "pull-to-refresh" function. Participants experienced some difficulty with the minimum length requirement imposed when sending a message. Chat messages can be no shorter than five characters, yet some participants were frustrated with this limitation and wanted to send messages of any length.

Participants also wanted to see an aggregation of a users chat messages when pressing on a chat message. At the time of this investigation, pressing on a chat message would bring the user to the author's profile page which displays the tags. This was done to create a connection between users and their contributions (**SE2**). However, these contributions need not be limited to tags only, as comments and messages represent legitimate contributions to the project. Future features could provide aggregations of different types of contributions and make them available on demand.

A particular highlight of this investigation was tag creation. The previous investigation tasked children with finding existing tags and had little emphasis on creating tags. This investigation focused on collaborative tag creation, which the participants highly enjoyed. Children especially liked finding new locations for potential tags and finding tags that others had created. Due to the influx of child-created tags, children were observed making greater use of the profile button located on the tag card. Previously, all tags added to the adventure were created by a single investigator, but this investigation added an extra dimension to tag exploration by allowing others to contribute tags.

Overall, children enjoyed this investigation, but deficiencies in the chat feature led to some children focusing exclusively on tag creation. This investigation found that the adventure chat function proposed in **SE2** is an interesting and useful functionality. While children did not make constant use of this function during the investigation, enriching discussion still took place in the limited amount of time the function was used. This function is relatively untried when compared to other Geotagger facilities and bears much room for improvement as a result. In an effort to address these shortcomings, a third and final investigation was designed to strengthen the chat feature.

Limitations

The real-time chat functionality is the least mature feature of the Geotagger system. As such, children encountered multiple difficulties in its operation during this investigation. One child in particular was unable to utilize the chat at all, while other children mentioned that these difficulties resulted in them moving away from the chat screen in favor of the more reliable comment feature.

4.2.4 Investigation 3: Adventure Chat Redesign

This investigation focused on redesigning and reworking elements present in the adventure chat. The adventure chat feature was meant to provide a shared social space and a forum for discussion centering around each adventure. The previous investigation also dealt with a practical use-case of the chat feature. This investigation showed that the adventure chat did indeed provide the features needed for adventure-related and meta-tag discussion. However, this investigation also pointed out several shortcomings of the adventure chat in its current implementation. It indicated that the adventure chat yet lacks functionality required for it to be considered an effective part of the Geotagger system. This outcome spurred the decision to conduct a secondary investigation into the adventure chat. In order to address the issues of the previous investigation, a collaborative redesign of adventure chat visuals and functionality was conducted.

Approach

The Kidsteam group was tasked with overhauling and redesigning user interface elements and chat functionality. Previous investigations conducted a practical assessment of specific Geotagger views and functions. This investigation was not concerned with real-world usage. It was concerned with creating new technology, rather than critiquing existing technology. The cooperative inquiry technique "big paper" was utilized in collecting children's ideas and identifying common themes.

Implementing the big paper technique, children were divided into three groups of two members each. Each group was accompanied by an investigator whose role was to keep the group on topic and give direction when needed. Groups were supplied with large canvas-sized pieces of paper and colored markers. Children were then tasked with completely redesigning the user interface for the adventure chat. This included the addition of any extra functionality that the children desired. Paper and markers were used to create a mock representation of the adventure chat. Groups were given thirty minutes to complete their drawings and were asked to present their work to the rest of Kidsteam.

Results and Discussion

Each group presented their mock-up and the main points from each mock-up were recorded on a whiteboard. Each team was assigned a different marker color (e.g., orange, green, and blue) in order to keep ideas separate. Investigators spent time focusing on each idea, confirming the intent of the idea before moving to the next one. Figure 4.9 displays the result of this activity.

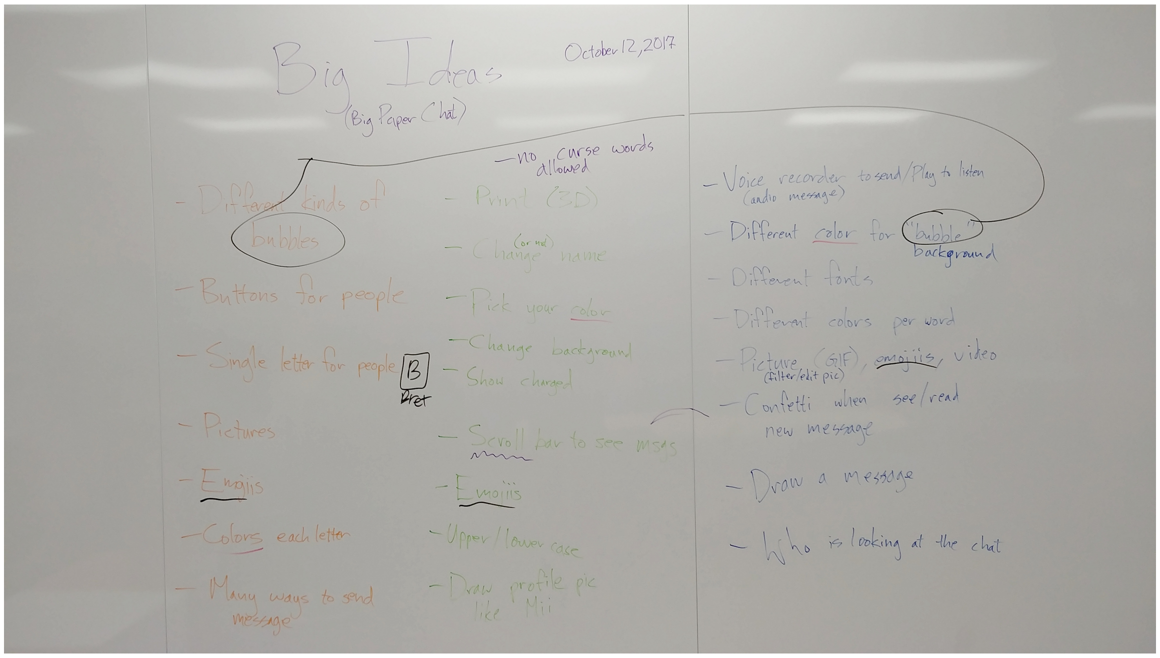

Big Ideas
(Big Paper Chat)
October 12, 2017
– Different kinds of bubbles
– Buttons for people
– Single letter for people
B
Bret
– Pictures
– Emojiis
– Colors each letter
– Many ways to send message
– no curse words allowed
– Print (3D)
– Change (or not) name
– Pick your color
– Change background
– Show changed
– Scroll bar to see msgs
– Emojiis
– Upper/lower case
– Draw profile pic like Mii
– Voice recorder to send/Play to listen (audio message)
– Different color for "bubble" background
– Different fonts
– Different colors per word
– Picture (GIF), emojiis, video (filter/edit pic)
– Confetti when see/read new message
– Draw a message
– Who is looking at the chat

Figure 4.9: Chat Redesign Big Ideas

While collating these ideas, researchers noted several common themes. These themes are made up of several disparate ideas contributed by all three groups. These themes helped to categorize participants needs and were used to identify the underlying intention of each feature within the theme. The "Big Ideas" on the whiteboard did not just represent simple feature requests, but represented a need for: additional interface feedback, enhancement of existing communication, additional forms of communication, and personal expression through interface customization. Further details are below.

- Informative feedback - More information as to what is happening within the chat.
- Communication enhancement - Additional features for chat messages.
- Further channels of communication - Multi-media gives way to richer, more diverse expression.
- Interface customization - Personalization of interface elements as a means of self-expression.

Participants came up with several ideas that centered around *informative feedback*. It is the idea that user interface elements provide additional information as to events and developments within the adventure chat. The blue team developed the idea to specify different message background colors for users. During the investigation, message backgrounds consisted of blue (for other users' messages) and gray (for current users' messages). These colors helped differentiate between messages that the user authored and messages that other users authored. However, message color does not differentiate between other messages. Allowing users to assign this color themselves would provide more feedback as to which users authored a particular message. The green team added to this idea by indicating that a user should be able to specify the bubble color themselves, lending itself to an extra dimension of customization. The blue team was interested in a mechanism that would inform authors when users read their messages. This might take the form of a confetti explosion or some other creative visual that would provide additional feedback as to who is reading messages and of what information they are aware. A somewhat similar idea taken from both orange and blue teams is to have a list of users who are currently viewing the adventure chat. This feature would afford the user greater awareness as to who they could address in chat messages. Finally, the orange team felt that the adventure chat needed different "types" of bubbles. These types of bubbles might include a *thought* bubble for idea messages, an *assertive* bubble for direct commands or assertions, and an *inquisitive* bubble for questions. These types of bubbles inform other users as to the intent of the message, allowing users to glean information about the message before they have even read them.

The adventure chat implemented in this work allowed users to only send messages to all members of an adventure. This restriction was meant to create a space of community for users to share their ideas. However, several participants of this investigation expressed an interest in sending messages to a subset of recipients or even a single recipient. Different types of messages were also considered. The type of message would be represented in the "bubble" interface element, providing users a quick overview as to the intent of the message. In reality, this feature describes the semantics of a message, a function not currently present in the adventure chat.

Perhaps the most common feature type was the addition of further channels of communication. The implementation phase for this chat feature was primarily concerned with creating a minimum viable product that would allow users to communicate in real-time. Certainly, simple, text-based messages were by far the most straightforward way to realize this functionality. As a result, children were keen to introduce different types of media into the adventure chat that would serve to create richer interactions with other users. "Emojis", a small, character-sized image used to express thoughts or emotion, were a particular favorite suggested by all three teams.

Emojis could add a playful element to communication and allow users to express themselves in non-conventional and creative ways. In addition, children requested increased media support beyond that of text. Suggested formats included audio with voice recording though the application, video, still image (e.g., PNG, JPG), and animated image (GIF). These additional media types enhance communication directly by extending the domain of expression that is available to the user. The blue team devised another extension with the concept of *drawing* a message. The user would be provided with some kind of canvas to allow them to select different colors and draw images with their finger. This idea grants yet another medium of communication, allowing users to express their thoughts in a way which is not covered by other forms of expression discussed in this section.

Finally, many participants expressed a desire for user interface customization features. Ideas included changing message font colors, font size and family, as well as creating a personalized "emoji" that would be associated with a specific user account. These features would allow users to express themselves and grant a sense of ownership to their user account.

Though the adventure chat proved to be a worthwhile feature, it is a very new feature that requires more development. Kidsteam members designed a number of improvements that should be considered when development resumes on the Geotagger application. This work laid a solid foundation for the current state of the adventure chat. The adventure chat represents a beneficial addition to the Geotagger system as it was able to facilitate project discussion between adventure members. Kidsteam used this chat feature in a previous investigation to communicate with one another about an adventure. However, participants were able to identify several shortcomings of the feature, as well as suggest many worthwhile extensions for future improvement.

CHAPTER 5

FUTURE WORK AND CONCLUSIONS

The closing chapter will contain future improvements for this work and final thoughts. Each enumerated feature would be a welcome asset upon implementation. This document closes with conclusions drawn from this work and how they affect citizen science and the Geotagger project.

5.1 Future Work

The development undertaken during this work represents a sizable engineering effort. Even so, further work is required to polish and complete some of the functionality required for a fully-featured Geotagger client. Though this application implements a considerable amount of Geotagger functionality, choices were made to focus more closely on certain aspects. These neglected aspects would benefit from further work and provide the Geotagger application with novel, useful, and interesting functionality.

5.1.1 Design Team Iteration and Development

Design teams are a part of the iterative development process whereby an application is improved by continuous design, development, and revision. In the past, the Geotagger project has leveraged design teams in order to improve the user interface and various features of the system. This work made limited use of a design team, and would likely benefit greatly from continued iterative design sessions. It would benefit future researchers to incorporate this feedback into forthcoming features, as well as provide a retrospective on previous features and design. Focusing on engaging outside users will grant a different perspective and invariably improve aspects that are currently inadequate, while reinforcing those aspects which are found to be worthwhile.

5.1.2 Application Look and Feel Refresh

During the course of the application's development, only cursory attention was paid to the application's look and feel. Most of the development effort was spent in ensuring that the desired application functionality was implemented in a timely manner, in preparation for the user study. Going forward, greater emphasis should be placed on the look and feel of user interface elements of the application. Potential changes include: updating to the overall color scheme of the application, reworking font size and style, widget placement and sizing, as well as including an application icon and splashscreen. All of these improvements would serve to give the application a more complete feel. They would go hand-in-hand with design team integration. Designers could work closely with children to ensure that the user interface reflects their needs. As noted previously, children had difficulty understanding certain prompts and buttons. Further effort put into revitalizing the application's interface would certainly improve overall usability.

5.1.3 iOS Application Testing and Further Development

This project was primarily concerned with the development of the Android version of the application. Developers were able to build and deploy an iOS version of the application, however this version was not the focus of the project. Android devices were more common and more economical to acquire for the user study, so Android became the target platform. At the time of development, no effort was made to create an Apple Developer account - which is required to test and develop for the iOS platform. Additionally, most of the user interface elements were created for the Android user experience. In actuality, there are many subtle interface queues which serve to differentiate the two platforms. Care should be taken to provide iOS users with an interface that is familiar to them. Ultimately, the development team would like to make the Geotagger application publicly available on both major platforms, yet further development would be required to realize this goal for the iOS version. Due to the difficulties experienced by previous attempts at developing a Geotagger iOS application, such a project would likely be a significant undertaking.

5.1.4 Performance Enhancements

Emphasis was placed on creating a fully-featured, functioning software product which could be utilized during the user study. Compromises were made relating to the application's overall performance. While care was taken to ensure that these performance shortcomings did not interfere with user interaction, improving upon these weaknesses would greatly add to the application's usability.

Infinite Scroll Content Loading Scheme

The developed application contains a number of "list" views. These list views serve to display a compact representation of aggregated data. Currently, there are six main list views in the application: general tag list view, profile tag list view, adventure list, comment view, adventure member view, and tag list within an adventure. During development, testing, and ultimately the user study, none of these views contained a cumbersome amount of content. Loading pages was relatively expedient. Data loaded as it was introduced to the application, lending itself to a more responsive user experience. However, these trials were conducted with a limited number of users. If an adventure were to contain a large number of concurrent users, tag and comment views would likely experience performance issues. The API specification defines route parameters which can be used to limit the amount of data being returned. This can be used to "paginate" results in a similar fashion to a Google search results page. This way the application would not have to load the entirety of the available data to result in increasing performance. Instead of presenting the data as discrete pages (in the case of Google), most modern mobile applications will instead choose to implement "infinite scrolling". Infinite scrolling happens when a user scrolls to the bottom of the current "page" of content. When the device detects that the user has reached the bottom, the application will automatically make a request for more data to present the next page. This allows users to scroll "infinitely", reading new content without having to change pages. Ionic provides simple hooks for implementing this functionality. Data is currently loaded into the application by requesting all available data. As data is pulled into the application, a merge is conducted between the data returned by the cache and the data returned by the API. Changing the amount of data loaded from each data source would affect this merge. Due to this complexity, adding infinite scrolling functionality would be a fairly involved addition.

Local Cache Tuning

In its final form, the local cache is functional for the purposes of this project. Yet problems may be encountered when scaling this project for wider use. As of now, the cache lacks several key features that would serve to tune its performance.

The local cache currently lacks database indexes. When searching for data the DBMS performs a linear search on the table in question. This is acceptable for tables with few columns. However, as a table continues to grow, this operation can become quite costly. Indexes prevent linear table scans by providing a secondary means of traversing table entries. Choosing clever columns to place in an index can lead to drastically improved query performance, thus leading to improved application performance. This change could improve search performance, but more can be done to ensure that only pertinent data is being stored within the cache.

Data that has been inserted within the local cache will continue to exist until one of two actions occurs: the entity is deleted (e.g., deleting a tag) or if the user logs out of the application. This situation can lead to stale or unused data taking up space in the cache. Creating a scheme for cache eviction would ensure that only relevant data stays in the cache. This is especially important if a user is a member of many different adventures that possess their own tags and comments. Caching chat messages further compounds this issue as chat threads can quickly grow, especially in adventures with many users. This cache eviction scheme could take the form of a limited cache that evicts the least recently used data in favor of new data. However, this could result in a high number of write statements as the cache fills, possibly requiring some kind of "batch" eviction. Clearly, more and effort time needs to be spent in designing an appropriate eviction scheme that would allow the application to scale reliably.

5.1.5 Automated Testing

Testing was a major concern in this application's development. In the weeks preceding the user study, group trials were conducted in an effort to eliminate the most pressing deficiencies. During development, testing primarily consisted of localized beta tests and end-to-end tests. However, no effort was made to integrate automated or unit testing into the project. The development team had relatively little experience in creating unit tests for a GUI application. Engineering effort was instead directed toward designing and developing individual application features and localized testing of these features. This paradigm allowed for expedited prototyping and testing of features, at the risk of more error-prone code. Indeed, further code refactoring will be a difficult process, as all changes will need to be tested completely to ensure that no faults were introduced. This process could be obviated through the introduction of automated unit tests. With automated tests, refactoring and other code changes could be tested efficiently without the need for full end-to-end testing. However, integrating a suitable testing framework into the project at this late stage would certainly be a difficult undertaking. Yet future peace of mind and easing of development difficulties afforded by such tests would justify the investment of time and effort.

5.1.6 Additional Geotagger Features

On the whole, the social features developed for this work are fairly complete and representative of the abilities of the Geotagger system. However, certain features were disregarded because they were either deemed to have lower research potential for the user study or these features are not fully realized within the Geotagger system. Going forward, an effort should be made to integrate these features into the API and database backend prior to their addition to the mobile client.

Custom Tag Attributes

Custom tag attributes were introduced during the design and implementation of the MySQL data store. Custom attributes allow for users to easily add their own attributes to the tag entity. This feature is primarily aimed at scientist users who may need to extend tags with additional information in order to provide a complete picture of the data being collected. This was not implemented as the standard fields found on a tag proved sufficient for the work accomplished in this thesis. Future work incorporating this functionality should devise a novel way for custom attributes to be displayed in the user interface. Special attention should be given to displaying these attributes on tag cards, giving users a concise, yet complete, representation of the tag. Lengthy lists of custom tag attributes could exist in some kind of popup or popover menu with visual context of the attributes.

User Roles and Collection Visibility

At present, user roles are largely unused in the Geotagger application. Visibility and user roles were statically assigned in an attempt to simplify that particular aspect of application development. For the purposes of testing and field research, tags and adventures were made available to all members. While this simplified development, it robs the application of a powerful feature. One that should certainly be present at the time of a full release. The database and API backend fully support the use of user roles and collection visibility, yet the documentation surrounding these features can be improved. Future emphasis should also be placed on more completely explaining the implications of these aspects to allow for easier development. Documentation should include a clear definition of each user role and visibility option, detailing their use and effect.

Thought should be given to creating different views of the application that will only be accessible to adventure coordinators. A kind of administration panel may be helpful in setting up user groups, permissions, tag visibility, as well as moving tags between collections. This could be particularly useful in a classroom setting here a teacher may have to manage a group of children. Quickly and easily navigating between the various collections and groups available in the application would greatly ease a busy teacher's workflow.

Groups

As a Geotagger feature, groups have yet to be implemented in any form. On the surface, groups appear to be similar to adventure member collections that are an existing feature. Adventure member collections serve to divide members based on their "role". That is, scientist members may occupy one collection, while citizen members occupy a different collection. It is possible the types of tags between the two collections have different intentions. Groups, on the other hand, provide a way to group users within collections. This can be useful in a classroom setting where a teacher needs to separate children into different teams. These different teams might have separate objectives. One team might be tasked with finding tags around a body of water, while another team finds tags in a wooded area. Groups represent another tool that can be used by adventure coordinators to further organize and create interesting scenarios for adventures. Once groups are implemented in the system's backend, the user interface should keep groups separate from member collections. These two concepts are easily confused and steps should be taken to alleviate this confusion.

Friends

Although a simple concept, friends can prove to be difficult to implement. Geotagger has the concept of users, but currently there is no backend support for relationships between these users. Adding the a friends feature would further connect users to one another, thereby motivating collaboration. Tags and comments which have been authored by “friends” could have a different look or a visual affordance that would inform the user that this content has been contributed by a friend. Fischer [15] shows that users are more likely to continually contribute to a citizen science project if they feel that they have some kind of “stake” in the project. The friends system would make strides in providing further opportunities to grant users a stake in the Geotagger project.

Activity Page

An activity page would give an overview of relevant Geotagger activity to the currently logged-in user. This could include tags and comments posted by friends, as well as other users which are members of any of the user’s adventures. This page would largely act as a sort of “news feed” common in social networks like Facebook and Twitter. Giving users a synopsis of user activity in the Geotagger project would show them, at a glance, what people are doing, seeing, and going. This page could also include buttons to add a tag of their own, reply to comments, or view other locations that a user may have been. An activity page would serve to give users a sense of “community”. Fischer [15] shows that users greatly value feeling a sense of belonging and community. This activity page would afford users instant context as to what has been happening in the project during their absence. This page could aid users in feeling that they are a part of a larger community, all of whom share an interest in the natural world around them.

Enhanced Chat Features

At the time of this writing, the adventure chat functionality is somewhat limited. The adventure chat provides a real-time instant messaging experience, but users are limited to sending only text-based messages. It is not uncommon for modern instant messaging applications to allow users to send a variety of media to other recipients. For example, Facebook Messenger allows users to send animated images, video, and even money. Slack, another popular messaging system, allows for the transmission of zipped archive files, source code snippets, and video. While it is not expected that Geotagger account for all of these features, extending the text-based functionality would give users tools for novel and more meaningful interactions. Adding the ability to send images would allow users to share things that they see in nature without the need to create a different tag. Images could also help users provide others with illustrations of where they are. Sending locations would help users communicate their current location or the location of an interesting landmark. Additionally, a user could send an “inline tag” which could combine these elements and create a compact tag representation within chat. A further feature could include a button that would allow the authoring user to automatically create a tag from the inline one present in the chat. These features would serve to give users interesting interactions with the Geotagger project, thus making their conversations more environmentally-focused than what can be accomplished with simple text-based messaging.

5.1.7 Source Code Maintenance

Over the course of the coding portion of this work, maintaining clean and readable source code was not the frontal concern. While the code is modularized and contains external documentation, these artifacts can be improved. Improving documentation and overall readability of this project would allow future developers to more easily begin work contributing to this codebase.

Improved Documentation

The Geotagger mobile application leverages a documentation library called "ngDocs". This library operates similarly to other documentation libraries like JavaDoc and Doxygen. These libraries express documentation as annotations that describe how classes, methods, and properties are used. These documentation utilities then compile the provided source code, stripping away the annotations and building a deployable static website which may be further referenced by developers. In its current iteration, the documentation for this project is fairly complete, however certain processes (e.g., adding a table to the local cache) are abstracted from the new developer. This would likely not require a significant amount of work on its own. It would also be useful to bring in an inexperienced developer, in order to see how newcomers to the project interact with the documentation and what questions that they might raise.

Implement CSS Preprocessor

CSS is a styling language commonly used in web application development. CSS "styles" are applied directly in the web browser client. At its core, the Geotagger mobile application is a web application. All of the styling, component coloring, and layout is accomplished via CSS. CSS defines a number of very powerful features that developers can leverage to style their application. However, as a web application grows, so does the size of its stylesheet. CSS stylesheets simply define a series of rules which are applied to the various elements in the page. Growing stylesheets can cause problems when targeting specific view elements, and can lead to inconsistent styling. This means that the process of updating and maintaining large stylesheets can become a challenging task in its own right.

CSS preprocessors like Sass and Less provide a way to combine standard CSS rules with control flow constructs such as variables, condition blocks, and subroutines. Rather than manually setting a background color or font family, these can be easily assigned with a variable. Subroutines, commonly called “mixins”, can be used when a large block of styling must be repeated for different rules. Once these constructs are added to the CSS code, the preprocessor is run and the code is “compiled” to a standard CSS file that can be interpreted by the browser. All of these features lend themselves well toward cleaning up the CSS rules and making the style hierarchy more readable.

5.2 Conclusions

Citizen science strives to create tight-knit communities of individuals who share a common interest in scientific inquiry. This thesis set about designing, implementing, and evaluating four social enhancements with the intent of strengthening the collaborative aspect of the Geotagger project. Over the course of this thesis, a new, cross-platform mobile client was created to allow a wide variety of users to interface with the Geotagger project. This thesis proposed a set of social enhancements designed to impact the way users interact with the project and with each other.

- **SE1** - Making comments more prevalent
- **SE2** - Connecting people through their contributions
- **SE3** - Creating a shared social space for adventures
- **SE4** - Geographic overview of tags

An evaluative study was conducted to investigate the social impact of these enhancements. The first study was a practical, real-world scenario intended to exercise all four social enhancements in a natural setting. However, due to a large number of variables and the sizable scope of the study, data taken from this study was unexpectedly ambiguous. It did not draw strong parallels between increased social interaction and the presence of social enhancements. Further investigations were created to provide small-scale, focused research of at the individual impact of a selection of social enhancements. The initial study took a holistic approach to assessing the impact of these enhancements. This led to the results being inconclusive and difficult to understand. Additional investigations were designed to understand the potential nuance present in participant feedback.

These investigations could not focus on every social enhancement, but a selection of enhancements was made to provide a deeper understanding of the impact of these enhancements. **SE1** was the subject of the first investigation. It dealt with adding comments to existing tags. Special attention was given to observe how users navigated to tag comments and how they created tag comments. This study sought to measure the impact of more prevalent comments (**SE1**). This investigation utilized sticky notes to gather participant feedback and draw commonalities between responses.

Children preferred the enhanced Geotagger Z to original Geotagger X, citing the ease of navigation to the comments screen. Children felt that the flow in Geotagger Z "easier" and "faster" than the analogous flow of Geotagger X.

The second investigation targeted the adventure chat function. This was an all new feature to the Geotagger project and received little attention in the initial study. This feature seeks to create a shared social space for adventure members where discussion about adventures and tags can take place (**SE3**). Although technical difficulties were encountered during this investigation, children were still able to make considerable use of the adventure chat. Here, children discussed creating tags about "signs of Fall", in which duplicate tags had to be avoided. Children participants frequently mentioned tags they were making or what might make for a good tag subject. Several children encountered difficulties in using the chat. Participants' big ideas were recorded on a whiteboard. These big ideas indicated that the adventure chat was an important feature, yet lacked significant core functionality that would allow for it to reach its potential of driving social interaction. A third investigation was designed to further investigate the adventure chat, this time focusing on features that would improve the user experience using the adventure chat.

The final investigation tasked Kidsteam members with completely redesigning the adventure chat function. Supplied with large pieces of paper and colored markers, these children set about designing new user interface elements and functionality for the adventure chat. Once they presented their designs to the rest of the group, big ideas were again recorded on the whiteboard. Investigators identified four primary themes in the children's designs: *informative feedback*, *communication enhancement*, *additional ways to communicate*, and *interface customization.* The children wanted more informative feedback to tell them what was happening and what other users were doing in the adventure chat. Visual feedback for when a message is read, different background colors for messages to differentiate message authors, and a list of members currently viewing the chat give users greater context as to the state of the adventure chat. Communication enhancement ideas included semantic bubble types to increase the methods that users can communicate with the existing chat functionality. The design team also wanted additional ways to talk to each other, including multi-media support, canvas messages, and emoji support. Finally, they wanted to be able to customize parts of the application. Different font sizes, styles, and user interface colors would help to give children a sense of ownership in the application.

Due to their smaller size, these investigations did not provide generalizable data, but were able to provide descriptions as to how and why enhancements were successful or unsuccessful. These investigations showed that children were able to utilize the enhancements to effectively socialize. Study participants noted a difference in interaction when comparing the two applications, favoring the enhanced application Geotagger Z. While this document may be at an end, work on the Geotagger project is far from finished. Many more features may be added to the mobile application to ensure rich interactions with the Geotagger project. This thesis has shown that these social enhancements have a direct impact on how users interact with the application and each other.

REFERENCES

[1] Bud Burst. http://budburst.org/, 2017. [Online; accessed 14-February-2017].

[2] eBird. http://ebird.org/content/ebird/, 2017. [Online; accessed 14-February-2017].

[3] eButterfly. http://www.e-butterfly.org/, 2017. [Online; accessed 14-February-2017].

[4] Old Weather. https://www.oldweather.org/, 2017. [Online; accessed 14-February-2017].

[5] Maria Aristeidou, Eileen Scanlon, and Mike Sharples. Weather-it: Evolution of an Online Community for Citizen Inquiry. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, i-KNOW '15, pages 13:1–13:8, New York, NY, USA, 2015. ACM.

[6] Frederic Bartumeus, Aitana Oltra, John Palmer, and Joan Garriga. Mosquito Alert. http://www.mosquitoalert.com/en/, 2017. [Online; accessed 14-February-2017].

[7] Carol Boston, Marshini Chetty, and Jennifer Preece. Understanding and Supporting Community Exploration of Local Green Spaces through Technology.

[8] Vickie Curtis. *Online citizen science projects: an exploration of motivation, contribution and participation.* PhD thesis, The Open University, February 2015.

[9] Paul Cushman. Geotagger Caching. Master's thesis, Montclair State University, Montclair, New Jersey, 2015.

[10] Andrew DeStefano. Geotagger: A Multi-Platform Citizen Science Application. Master's thesis, Montclair State University, Montclair, New Jersey, 2016.

[11] Allison Druin. Cooperative Inquiry: Developing New Technologies for Children with Children. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 592–599, New York, NY, USA, 1999. ACM.

[12] Jerry Fails, Katherine Herbert, Elizabeth Hill, Andrew DeStefano, Brandon Hesse, Paul Cushman, Travis Gant, Syed Shah, Aliet Abreu-Cruz, Nikita Panchariya, and Varsha Nimbagal. Geotagger: A collaborative environmental inquiry platform. In *2015 International Conference on Collaboration Technologies and Systems (CTS)*, pages 383–390, June 2015.

[13] Jerry Alan Fails, Mona Leigh Guha, and Allison Druin. *Methods and Techniques for Involving Children in the Design of New Technology for Children.* Now Publishers Inc., Hanover, MA, USA, 2013.

[14] Jerry Alan Fails, Katherine G. Herbert, Emily Hill, Christopher Loeschorn, Spencer Kordecki, David Dymko, Andrew DeStefano, and Zill Christian. GeoTagger: A Collaborative and Participatory Environmental Inquiry System. In *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW Companion '14, pages 157–160, New York, NY, USA, 2014. ACM.

[15] Gerhard Fischer. End User Development and Meta-Design: Foundations for Cultures of Participation. *Journal of Organizational and End User Computing (JOEUC)*, 22(1):52–82, 2010.

[16] Gerhard Fischer. Understanding, Fostering, and Supporting Cultures of Participation. *interactions*, 18(3):42–53, May 2011.

[17] Elizabeth Foss, Mona Leigh Guha, Panagis Papadatos, Tamara Clegg, Jason Yip, and Greg Walsh. Cooperative Inquiry Extended: Creating Technology with Middle School Students with Learning Differences. *Journal of Special Education Technology*, 28(3):33–46, September 2013.

[18] Travis Gant. Developing and Deploying a Mobile Application Platform to Facilitate Environmental Research and Education. Master's thesis, Montclair State University, Montclair, New Jersey, 2014.

[19] James P. Gee. Semiotic social spaces and affinity spaces. January 2005.

[20] Lynne Hall, Colette Hume, and Sarah Tazzyman. Five Degrees of Happiness: Effective Smiley Face Likert Scales for Evaluating with Children. In *Proceedings of the The 15th International Conference on Interaction Design and Children*, IDC '16, pages 311–321, New York, NY, USA, 2016. ACM.

[21] Benjamin K. Haywood. A Sense of Place in Public Participation in Scientific Research. *Science Education*, 98(1):64–83, January 2014.

[22] Riad Jeradeh. Integrating Scientific Data Analysis Methodologies into a Citizen Science Mobile Platform. Master's thesis, Montclair State University, Montclair, New Jersey, 2016.

[23] Sunyoung Kim, Jennifer Mankoff, and Eric Paulos. Sensr: Evaluating a Flexible Framework for Authoring Mobile Data-collection Tools for Citizen Science. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1453–1462, New York, NY, USA, 2013. ACM.

[24] Sunyoung Kim, Christine Robson, Thomas Zimmerman, Jeffrey Pierce, and Eben M. Haber. Creek Watch: Pairing Usefulness and Usability for Successful Citizen Science. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2125–2134, New York, NY, USA, 2011. ACM.

[25] Kendra Knudtzon, Allison Druin, Nancy Kaplan, Kathryn Summers, Yoram Chisik, Rahul Kulkarni, Stuart Moulthrop, Holly Weeks, and Ben Bederson. Starting an Intergenerational Technology Design Team: A Case Study. In *Proceedings of the 2003 Conference on Interaction Design and Children*, IDC '03, pages 51–58, New York, NY, USA, 2003. ACM.

[26] Mona Leigh Guha, Allison Druin, and Jerry Fails. Cooperative Inquiry revisited: Reflections of the past and guidelines for the future of intergenerational co-design. *International Journal of Child-Computer Interaction*, 1:14–23, January 2013.

[27] Karen Masters. Galaxy Zoo. `https://www.galaxyzoo.org/`, 2017. [Online; accessed 14-February-2017].

[28] Gabriel Mugar, Carsten sterlund, Corey Brian Jackson, and Kevin Crowston. Being Present in Online Communities: Learning in Citizen Science. In *Proceedings of the 7th International Conference on Communities and Technologies*, C&T '15, pages 129–138, New York, NY, USA, 2015. ACM.

[29] Alicina Mumar. A Dynamic and Interactive Citizen Science Client. Master's thesis, Montclair State University, Montclair, New Jersey, 2016.

[30] Max Nanis, Ginger Tsueng, and Andrew Su. Mark2cure. `http://mark2cure.org`, 2017. [Online; accessed 14-February-2017].

[31] Jordan Raddick, Georgia Bracey, Pamela L. Gay, Chris J. Lintott, Phil Murray, Kevin Schawinski, Alexander S. Szalay, and Jan Vandenberg. Galaxy Zoo: Exploring the Motivations of Citizen Science Volunteers. *Astronomy Education Review*, 9(1), December 2010. arXiv: 0909.2925.

[32] Janet C. Read. Evaluating Artefacts with Children: Age and Technology Effects in the Reporting of Expected and Experienced Fun. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 241–248, New York, NY, USA, 2012. ACM.

[33] Janet C. Read and Stuart MacFarlane. Using the Fun Toolkit and Other Survey Methods to Gather Opinions in Child Computer Interaction. In *Proceedings of the 2006 Conference on Interaction Design and Children*, IDC '06, pages 81–88, New York, NY, USA, 2006. ACM.

[34] Gavin Sim and Matthew Horton. Investigating Children's Opinions of Games: Fun Toolkit vs. This or That. In *Proceedings of the 11th International Conference on Interaction Design and Children*, IDC '12, pages 70–77, New York, NY, USA, 2012. ACM.

[35] Brian L. Sullivan, Christopher L. Wood, Marshall J. Iliff, Rick E. Bonney, Daniel Fink, and Steve Kelling. eBird: A citizen-based bird observation network in the biological sciences. *Biological Conservation*, 142(10):2282–2292, October 2009.

[36] Jan Sveide, Alexandre Antonelli, Edler Daniel, and Johannes Klein. BioNote - Identify the World's Species, Together. `http://bionote.xyz/`, 2017. [Online; accessed 14-February-2017].

[37] Amber G. F. Teacher, David J. Griffiths, David J. Hodgson, and Richard Inger. Smartphones in ecology and evolution: a guide for the app-rehensive. *Ecology and Evolution*, 3(16):5268–5278, December 2013.

[38] Ramine Tinati, Markus Luczak-Roesch, Elena Simperl, Nigel Shadbolt, and Wendy Hall. '/Command' and Conquer: Analysing Discussion in a Citizen Science Game. In *Proceedings of the ACM Web Science Conference*, WebSci '15, pages 26:1–26:10, New York, NY, USA, 2015. ACM.

[39] Greg Walsh, Alison Druin, Mona Leigh Guha, Elizabeth Foss, Evan Golub, Leshell Hatley, Elizabeth Bonsignore, and Sonia Franckel. Layered Elaboration: A New Technique for Co-design with Children. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1237–1240, New York, NY, USA, 2010. ACM.

APPENDIX A

DESIGN DOCUMENTS

A.1 Project Structure

Due to the overall size of the application codebase, a robust project file structure was required to keep code as modular and understandable as possible. The Angular framework was used to separate the concerns of this application. The modules of this project fall into one of three Angular entities:

- Directives - HTML elements which act as a container for other HTML elements.
- Controllers - JavaScript functions which are visible to the view (HTML) and provide hooks for UI interaction.
- Services - JavaScript modules which define reusable functions and properties.

Code entities are divided into directories based on their role in the application. Views and controllers have a one to one relationship, a single controller is attached to a single view, both of which have similar names. Directives are found in the same directory as their functionality, in general, and can apply to the whole application. Element directives abstract other HTML elements and require an HTML template file. These can be found in the "templates" directory. Services may also be used anywhere, yet they are subdivided based on which part of the application concerns them. Services comprise the majority of the application and can provide a wide range of reusable functionality. Please note that all code is contained in the "www" directory of the repository. Cordova pulls source-code directly from this directory to compile the application into an executable binary. The project structure can be seen in the itemized list below.

- **css/** - Contains Cascading Style Sheet files.
- **img/** - Contains local image resources.
- **js/** - Contains the majority of the JavaScript source code.
    - **controllers/**
        - **adventure/** - Contains tabbed adventure controllers.
        - **profile/** - Contains tabbed profile controllers.
        - **tag/** - Contains tag comment and map controllers.
    - **directives/** - Contains template directives.
    - **services/**
        - **api/** - Handles requests to remote API endpoints.
        - **cache/** - Defines cache schema and handles local cache requests.
        - **constants/** - Defines configurable constants.
        - **data-handlers/** - Handles requests between controllers and API/cache endpoints.
        - **models/** - Object classes that make up the application's data model.
        - **requests/** - Hooks for use when the application makes a "request" to modify remote data.

    * *account-srvc.js* - Handles user sessions and action permissions.
    * *action-srvc.js* - The action cache.
    * *geo-logger-srvc.js* - Provides logging capabilities.
    * *geocode-srvc.js* - Provides geo-location and geo-coding functions.
    * *image-srvc.js* - Handles requests for image resources, both local and remote.
    * *network-srvc.js* - Provides hooks for network events.
    * *notify-srvc.js* - Provides hooks for changes in the underlying data model.
    * *utility-srvc.js* - Provides general utility functions.
  - *app.js* - Angular application entry point.
  - *config.js* - Configuration script for setting default Ionic options.
  - *routes.js* - Angular UI Router route definition file.

- **lib/** - External library directory.

- **templates/** - Directive template directory.

- **views/** - Contains HTML frontend view definitions.

  - **adventure/** - Contains tabbed adventure views.
  - **partials/** - Contains all partial views (e.g., modals, popups, actionsheets, etc.).
  - **profile/** - Contains tabbed profile views.
  - **tag/** - Contains comment and map views for tags.

– *index.html* - Sole HTML page responsible for bootstrapping the application.

A.2 Cache Schema

In designing a database schema for the local cache, a schema was needed that would be flexible enough to deal with the dynamic nature of JavaScript objects and work in concert with data retrieved from the existing remote database. Much of the structure for the local cache was taken directly from Cushman's thesis [9] on the topic. However, a number of changes were necessary to the existing schema to reflect the differing application domain.

A.2.1 Schema Overview

Several other caching solutions were considered, such as MongoDB and CouchDB/PouchDB. However, the data contained in this application was too relational to be easily stored with a NoSQL database, thus spurring the decision to use SQLite. Schema tables were designed so that each represents a single type of data entity. This model was passed between various modules in the application. Unlike the Android and remote databases, tables are normalized so that each column refers to a single point of data. This design makes the data more structured but less flexible, while it disallows the application from taking full advantage of JavaScript objects. To circumvent this shortcoming, CRUD operations are handled by an ORM module which maps table columns to object properties. Figure A.1 gives a relational overview of the cache schema.

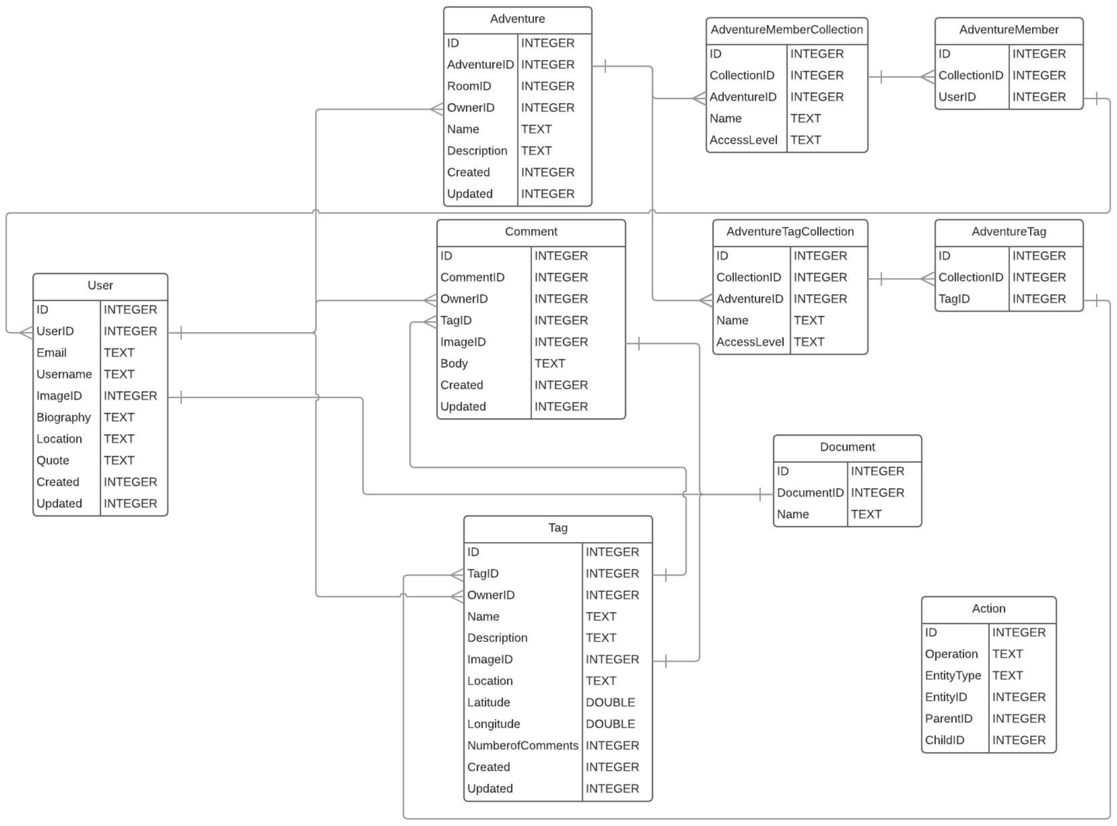


Figure A.1: Cache Schema Diagram

A.2.2 Overview of Database Tables

All tables present in the schema have an “ID” field which is used to insert new elements into tables. This value is created locally by SQLite, as opposed to local ID fields on tables (e.g., DocumentID) which are inserted into the table directly from the API.

Adventure Table

The adventure table holds records relating to Geotagger Adventures. An adventure is a grouping of users and tags where tags contributed by members can be seen and responded to by all other members of the adventure. This table structure is described below.

- AdventureID: `INTEGER` - A primary key value used to uniquely identify adventures.
- OwnerID: `INTEGER` - A reference column used to relate an adventure to its creating user.
- Name: `TEXT` - The name or title given to the adventure.
- Description: `TEXT` - Text describing the adventure, its aim, location, etc.
- AccessLevel: `TEXT` - Visibility level of this adventure (e.g., protected or private).
- RoomID: `INTEGER` - Unique room key used for joining socket.io rooms.
- Created: `INTEGER` - Millisecond value representing when this record was created.
- Updated: `INTEGER` - Millisecond value representing when this record was last modified.

Collection Tables

The collection tables are junction tables and their children which are used to relate entities to collections. AdventureMemberCollection and AdventureTagCollection make up the collection tables, while AdventureMember and AdventureTag represent the junction tables.

Collection tables represent a collection of entities (e.g., users and tags) which can be grouped together at the adventure level. This provides adventure coordinators with fine-grain control over what tags are available to the various members of the adventure. Collection tables take the following form.

- CollectionID: `INTEGER` - A primary key value used to uniquely identify collections.
- AdventureID: `INTEGER` - A reference column used to relate this collection to its parent adventure.
- Name: `TEXT` - A name or title given to the collection.
- AccessLevel: `TEXT` - Visibility level of this adventure (e.g., protected or private).

Junction tables are used to map between the collections and the individual entity rows that are a part of the collections. In this way, a many-to-many relationship is achieved. Here multiple collections may contain multiple entities, likewise multiple entities may be a part of multiple collections. Junction tables take the following form.

- CollectionID: `INTEGER` - A reference column used to relate this mapping to the parent collection.
- EntityID: `INTEGER` - A reference column used to relate this mapping to the child entity (e.g., UserID, TagID).

Comment Table

The comment table represents records pertaining to responses left on tags. These comments consist of textual data, as well as optional image data.

- CommentID: `INTEGER` - A primary key value used to uniquely identify comments.

- TagID: `INTEGER` - A reference column used to relate this comment to its parent tag.
- OwnerID: `INTEGER` - A reference column used to relate a comment to its creating user.
- ImageID: `INTEGER` - A reference column relating a comment to image data found in the Document table (optional field).
- Body: `TEXT` - The textual body of the comment.
- Created: `INTEGER` - Millisecond value representing when this record was created.
- Updated: `INTEGER` - Millisecond value representing when this record was last modified.

Document Table

The document table represents all of the local "document" resources available to the application. This may include profile pictures, tag images, and comment images. Users are able to upload their own images via the application. A record of every image is kept in this table. This avoids storing multiple copies of the same image, thereby improving application speed and reducing network load. Table fields are described below.

- DocumentID: `INTEGER` - A primary key value used to identify documents.
- Name: `TEXT` - A local file path pointing to the image file's location on disk.

Message Table

The message table was introduced with the real-time chat feature. Chat messages retrieved from the API, as well as messages created via the application are recorded here. Message table fields are described in the following list.

- MessageID: `INTEGER` - A primary key value used to uniquely identify messages.
- RoomID: `INTEGER` - The room to which this message record belongs.
- OwnerID: `INTEGER` - A reference column used to relate a message to its creating user.
- Body: `TEXT` - The textual body of the message.
- Created: `INTEGER` - Millisecond value representing when this record was created.

Tag Table

Tags represent an interesting landmark or environmental feature. Tags contain an image to visually identify the feature, as well as latitude and longitude coordinates so that other users may also find the feature. Tag table columns are described below.

- TagID: `INTEGER` - A primary key value used to uniquely identify tags.
- OwnerID: `INTEGER` - A reference column used to relate a tag to its creating user.
- Name: `TEXT` - The name or title given to the tag.
- Description: `TEXT` - Text describing the tag's location, interesting features, etc.

- ImageID: `INTEGER` - A reference column relating a tag to image data found in the Document table.
- Location: `TEXT` - A textual representation of a tag's location (e.g., Boise, Idaho).
- Latitude: `DOUBLE` - Latitude coordinate value between -90 and 90 degrees.
- Longitude: `DOUBLE` - Longitude coordinate value between -180 and 180 degrees.
- NumberOfComments: `INTEGER` - The number of comments responding to this particular tag.
- Created: `INTEGER` - Millisecond value representing when this record was created.
- Updated: `INTEGER` - Millisecond value representing when this record was last modified.

User Table

The user table stores metadata for user accounts. Some of these fields are made available through the application to enable further exploration of that data, as well as the contributors of the data.

- UserID: `INTEGER` - A primary key value used to uniquely identify users.
- Username: `TEXT` - An identifying name for this user.
- Email: `TEXT` - Email address used in registering.
- ImageID: `INTEGER` - A reference column relating a user record to image data representing the user's profile picture (optional field).

- Location: `TEXT` - A textual representation of the user's location (e.g., Boise, Idaho).
- Biography: `TEXT` - A short biography describing the user's background and interests.
- Quote: `TEXT` - A quote which the user found particularly inspirational.
- Role: `TEXT` - The user group that this record falls into (e.g., role-scientist, role-citizen).
- Created: `INTEGER` - Millisecond value representing when this record was created.
- Updated: `INTEGER` - Millisecond value representing when this record was last modified.

A.2.3 Changes From Android Application Cache Schema

The Android application made frequent use of SQLite foreign keys to ensure data consistency and referential integrity. Unfortunately, the capabilities of this particular implementation of SQLite did not allow for foreign key functionality. Even with the usage of foreign keys, the order in which data enters the application is not guaranteed. Therefore, foreign keys were removed to make the data more flexible. Yet this approach came with the added complexity of ensuring data at the application level.

With the absence of foreign keys, facilities were added to manually ensure data consistency throughout the application. When a parent record is removed from the cache, any child records from other tables relying on that record have to be removed themselves. For instance, if a tag was removed from the local cache, the cache would also have to remove any comment records with a TagID field corresponding to the removed tag. Data consistency problems arise when a record is modified by a user. In this way, much of this functionality was consigned to the action cache which would be able to update ancillary cache records when needed.

Another change from the Android schema is the addition of a RoomID column on the Adventure table. In implementing the social enhancements outlined in the proposal for this thesis, a social space was created for adventure members to discuss the adventure itself. This took the form a “chat” tab which includes a real-time chat feature. A JavaScript library called “socket.io” was used to implement this feature. Socket.io uses the concept of “rooms” to organize the messages being sent to the socket.io server.

Users can enter any number of rooms in order to receive messages sent in that room. Each adventure acts as its own “room”. Retroactive changes were made to both the remote and local databases to accommodate this feature, ensuring that every row in the adventure table would be given its own room. A room is a unique identifier which allows the socket.io client to connect to the server as soon as the adventure is entered. While the user is present in that adventure, they will receive any messages sent within that adventure’s room. The number of received messages will display on the badge icon on the “chat” tab.

One of the planned social enhancements called for making tag comments more prevalent. The previous version of the application required users to navigate to the tag’s detail view, then the comment view. This enhancement allowed users to view comments on tags with a single button press. A secondary goal was to provide the user with added context as to the activity on each tag. A user could see which tags had more comments by looking at the tag list view, rather than navigating to the comments each time. Even displaying the comments with a single button press could be a tedious process as the user could find themselves frequently opening the tag comments.

To provide the user with instantaneous information about comment activity, a small badge icon was added to the "comment" button. This badge tracks the number of comments relating to that tag. This enhancement was integrated into the card representation so that users could scroll through available tags, aware of the number of comments on each tag. To achieve this result, a new field was added to all tag related API responses. When the API receives a request for a list of tags, the number of comments that reference those tags is calculated and returned with the rest of the HTTP response. A new field called "NumberOfComments" was added to the local cache database. Rather than having to make an API call each time, a previously accurate representation of the number of comments on a tag is stored. This is updated each time the page is refreshed.

The final change from the Android application was to the action cache. While the application developed in this thesis relies on the same principles of the action cache designed by Cushman [9], there were significant changes in the implementation of the action cache. The previous action cache implemented the following fields.

- CacheTime: `DATETIME` - Time of cache, used in ordering records.
- ActionString: `TEXT` - Action to perform on the database.
- ActionHandler: `TEXT` - Identifies a handler to perform the action.
- Url: `TEXT` - A remote resource location to which the data should be sent.
- PostActions: `TEXT` - Further actions to be performed following the action.

The algorithm for registering and resolving applications has been overhauled. As before, actions are serviced in the order that they are inserted in the action cache. Though the data required to service and propagate these changes to the API has changed. The action schema is as follows:

- Operation: `TEXT` - The type of operation performed by this action (e.g., Post, Put, or Del).
- EntityType: `TEXT` - The type of object being operated on (e.g., Tag, Comment, Adventure).
- EntityID: `INTEGER` - Row identifier for the object being operated on.
- ParentID: `INTEGER` - Used in place of EntityID for referring to junction records (e.g., AdventureTag, AdventureMember).
- ChildID: `INTEGER` - Used in place of EntityID and in conjunction with ParentID.
- Created: `INTEGER` - Millisecond value representing when the row was inserted into the action cache.

Note that there are similarities between the two implementations as both keep track of the type of action being propagated and the time at which the action was registered. This work's implementation removes URL strings as API routes could change, leading to unresolved action records being invalidated. Additionally, "handlers" have been replaced by local resolver functions in the ActionService. The ActionService will resolve any data inconsistencies that might arise when side-effecting data in the local cache, thereby removing the need for the PostActions. More information about how actions are handled and resolved can be found in section A.3.

A.3 Action Cache

The scheme for the action cache was originally taken from Cushman's work on the Geotagger Android application [9]. The local cache allows for rudimentary data retrieval and modification in the event that the mobile device loses a data collection. This is a feasible use scenario, especially in the case of scientific or field research teams where a network connection is not always available. While the device is connected to the network, all data brought in from the REST API is stored in the local SQLite database. As changes are made to the in-memory data model, they are also propagated to both the local cache and the API. When the device is disconnected from the Internet, these modifications are queued in the action cache until a data connection is restored. This is vital to maintaining consistency with the remote data store.

When the device is offline, users are able to view and update cached data, though they will not be able to load views that they have not visited. Actions are enqueued as they are created via the user interface. Newly created records are given negative ID values. Assigning a positive, auto-incrementing identifier value could conflict with a preexisting, unrelated record. As such, ID values of increasing negativity are used to denote records which do not exist in the remote data store. Once a network connection has been established, actions stored in the action table will be resolved in a "first-in, first-out" fashion. As actions are resolved, data returned by the API response propagates through all tables in the cache, including the action table. Once an action is resolved, its row is removed from the action table and the servicing routine continues to the next row until the table is empty.

A.3.1 Maintaining Synchronicity with Remote Data Store

In order to provide a consistent offline experience, data must be cached and kept up-to-date as it is returned by the API. It would be wasteful to simply overwrite all similar records in the local cache. Instead the results of the API call must be merged with that of the equivalent cache call. This merge algorithm abides by the following rules:

1. Unmodified, up-to-date local data should not be overwritten.
2. API data that has no equivalent in the cache result should be inserted freshly into the cache.
3. API data that differs from its cache equivalent should overwrite its analogue.
4. Once all data in the API resolved has been resolved, data in the cache result which has no analogue in the API should be removed from the local cache.

Listing A.2 depicts a simplified version of the algorithm as it is implemented in both versions of the application. The algorithm takes the shape of a subroutine called: *reconcile*. This function "reconciles" the results of the cache to that of the remote API. The *reconcile* function takes a number of arguments:

- cache - An array of objects which represents the results of a local cache query.
- api - An array of objects which represents the results of a REST API response.
- addHandler - A function handle used to create new records and update existing records within the local cache.

- delHandler - A function handle used to delete existing records from the local cache.

For simplicity, results from both the cache and the API are stored in JavaScript objects, using the record IDs as object keys. In JavaScript, objects function similarly to *hash tables* or *associative arrays* in other languages. Properties may be appended or removed from objects dynamically and are accessed via standard dot notation (e.g., `object.property`) or with a string via bracket notation (e.g., `object['property']`). Once results have been placed in objects, the contents of the API results are iterated and compared with the analogous results from the cache. If the API item does not exist in the cache, then it must be inserted. If it does exist in the cache, then it must be compared with the similar item and overwrite the cache item when the objects are not equal. Whether or not the API item is equal to the cache item, the corresponding cache item is "deleted" from the cache result object, because the cache result object keeps an account of objects from the cache which have been serviced. Once all of the API items have been serviced, the remaining items of the cache result are checked. If there are no more remaining items in the cache results array, then all cache items have been accounted. Remaining items in the cache results array are obsolete as they no longer exist in the remote store. This is feasible if a record is removed from the API and the change has not been reflected in the local cache (e.g., manually deleted from the database or removed via a separate REST client). In this case, the record must be deleted from the cache.

Negative IDs must be separated during this comparison. While it is unlikely that records with negative IDs will be merged (this merge only takes place when there is a network connection and actions are resolved as soon as a network is established), there

```
reconcile(cache,api,addHandler,delHandler) {
  var cheRes = {}; // cache result
  var apiRes = {}; // api result
  cache.forEach( function(item) {
    cheRes[ item[item.getIDField()] ] = item;
  });
  api.forEach( function(item) {
    apiRes[ item[item.getIDField()] ] = item;
  });
  for(var apiProp in apiRes) {
    if(cheRes[apiProp] != null) {
      if(!compareObjs(cheRes[apiProp],apiRes[apiProp]))
          addHandler(apiRes[apiProp]);
      delete cheRes[apiProp];
    } else
        addHandler(apiRes[apiProp]);
  }
  for(var cheProp in cheRes)
    delHandler(cheRes[cheProp]);
}
```

Figure A.2: Reconcile Algorithm

is still a possibility that a negative ID record may be present, due to the asynchronous nature of API requests and responses.

A.3.2 Storing Actions

The local cache allows for offline viewing of Geotagger data. Further facilities are required to allow the user to modify this data while the device is offline. When the user modifies local data this is referred to as an “action”. Actions are stored to the action table of the local cache regardless of the presence of a data connection. This ensures a uniform flow of data, allowing for easier code comprehension and maintenance.

Each action is divided into one of three types: *post*, *put*, and *del*. These actions correspond to the public functions on the ActionService module: *create*, *update*, and *remove*. Retrieval actions do not side-affect cache data, so they do not need to be a part of the action cache. "Create" actions immediately affect the cache, inserting a new element into the database. In order to separate local records (e.g., those that have not been sent to the API yet), the newly inserted record is given a negative ID. Negative IDs start at negative one (-1). Negative IDs are local to tables (e.g., there may exist a comment with an ID of -1 and a tag of ID -1). Each time a new record is inserted, the lowest ID is found, decremented, then given to the new record at time of insertion. Once the record is inserted, a corresponding record is added to the action cache. Here, the record's negative ID, its table name, the type of action (post), and the time of insertion is recorded. Updates are fairly simple: the updated record immediately overwrites the old record. Subsequent local queries will refer to the updated record. Updates are recorded by inserting the entity's ID (this can be negative in the event that an unresolved record is updated), the entity's table name, the operation on the entity (put) and the time at which the entity was updated. Deleted records work in a similar fashion: the entity is immediately removed from the local cache, then the entity's ID, table name, and action type (del) are placed in the action cache. Once an action is stored, if a data connection is available, then the action service can immediately begin resolving the stored actions.

A.3.3 Servicing Actions

Whenever a user creates, updates, or deletes record, the corresponding action function is called to affect the cache and queue the action. Once that function has returned, a call is made to the *start* function. The *start* function commences the servicing routine and resolves the first action in the queue. Once the action has been serviced it will continue until there are no more actions in the queue. The servicing routine begins by “popping” the action table and removing the first (oldest) element. The “Operation” field is read to determine the next steps to take. In the case of either “post” or “put”, the servicing routine proceeds as normal. The order of operations for “del” actions is explained later in this section. Using the operation type, the servicing routine indexes into one of three map objects (one for each action type). Each action map contains a number of properties which correspond to entities existing within the database (e.g., comment, tag, adventure). The servicing routine takes the “EntityType” field and uses it to index into this second level of maps. This second level returns three functions: *affect*, *proceed*, and *rectify*. *Affect* provides the proper call to the API which will finally persist the data. *Proceed* determines whether or not the “rectify” step is necessary. Rectify performs a series of operations in order to maintain the referential integrity of the cache (which does not utilize foreign keys). Below is the resolutio9n algorithm followed by the servicing routine.

1. `Enqueue` - Pushes the action onto the action cache, so it is ready for servicing.

2. `Execute` - Entry function that starts servicing actions.

3. `Proceed` - A boolean operation that determines whether or not the “affect” and “rectify” need to execute. Usually this function returns a boolean “false” value in the case that there are no further records referring to this record.

4. `Get` - Retrieve the referenced record from the cache so that it may be serialized into an HTTP request.

5. `Affect` - This function persists the change to remote data store via the REST API.

6. `Rectify` - Function that ensures data consistency by updating any records which depend on the updated record.

7. `Dequeue` - Remove the action completely from the action cache.

8. `Notify` - Observer pattern function which serves to provide a hook to view controllers watching particular data models. It notifies any observers that the data has changed and the view should be updated.

9. `ContinueActions` - Check for further records and return to "Execute" if needed.

This is not a complete walkthrough of the action cache procedure. The *dequeue* function is not a conditional operation. Actions will be removed from the action cache even if an error is encountered. Furthermore, the *rectify* routine may affect the action cache itself. For example, adding a tag and then updating the same tag will result in the update action referring to a negative ID. Once the "post" operation for the tag has been resolved, the *rectify* routine must pass through the action cache and update the EntityID reference for the "put" operation.

Finally, the *notify* operation is required due to asynchronous HTTP requests and responses. If the user adds a tag in the adventure tag view, the data in the view model will reflect the state of the tag at the time of creation. This includes the tag's negative ID value. Adding comments or making changes to the tag will use the tag's current ID value in order to link the record to these actions. If the user remains on the tag-list view, then the in-memory tag ID will continue to be negative, even after the cache has been "rectified" with the new ID. The NotifyService provides a push-based, observer pattern allowing the ActionService to notify observing controllers that a view element has been updated. In this way, the underlying data model for view elements may change, even though it might not affect a visual change.

Deletion actions perform somewhat differently from other actions. These actions must be immediately "rectified" within the cache, because the record itself should no longer exist in the cache. It would have been possible to mirror the remote store's schema, adding a "deleted_at" field. However, the development team chose the simplicity of removing the deleted record. Furthermore, all that is needed to "affect" a delete action on the server is the unique identifier for the deleted record. There is no need to maintain the corresponding record in the local cache. From here, the delete operations have a very similar flow to that of the other actions. The flow is described below.

1. `Rectify` - Since deleted records must be removed immediately, all children and other references must be immediately updated as if the change has been persisted remotely.

2. `Execute` - Once the change has been persisted, further steps may now be taken to fully resolve the delete action.

3. `Proceed` - Determines whether or not the delete action needs to be "affected" on the remote store.

4. `Affect` - Removes the corresponding record from the remote store.

5. `Dequeue` - As with other actions, the delete action must be removed from the cache.

6. `ContinueActions` - Continue execution, if further actions remain in the cache.

This algorithm does not include the notify *notify* step, because Angular provides an API for removing visual elements from the user interface. When a data model entity is deleted, then the visual representation of that entity is deleted as well. For this application it fell to the controller to persist the removal of both the user interface element and the corresponding model entity. These changes are affected as soon as the user presses the “delete” button. This gives the interface a more fluid feel and grants the illusion that the entity was immediately removed. For this reason, the controller does not need a hook to know when the deletion action was persisted.

APPENDIX B

CARD DESIGN PROTOTYPES

The “tag card” representation was largely responsible for implementing **SE1** and **SE2** throughout several views in the resulting application. This appendix will illustrate a sampling of the prototype designs proposed by researchers. These prototypes were sketched on pieces of paper and not implemented in the application. This allowed for faster iteration on these designs and the ability to express features that would otherwise require significant engineering effort.

Figure B.1 depicts a sample of earlier card prototypes. Initial prototypes utilized the “item” representation found in Geotagger X. The original goal was to add buttons to this more compact design to realize the social enhancements. Researchers found that this compact design made it difficult to accommodate all of the pertinent tag data. Later iterations would prefer a more spacious layout. The right side of the figure depicts larger cards that allow for more tag-related information and larger button targets.

Figure B.2 displays later examples of card prototype iterations. These iterations focused on more liberal use of screen space, using a large tag image to capture the user’s attention. Since tags represent a physical landmark, making the image a relatively large part of the card would give the user a better connection to the place that tag is representing. Though this would eventually take less screen space,

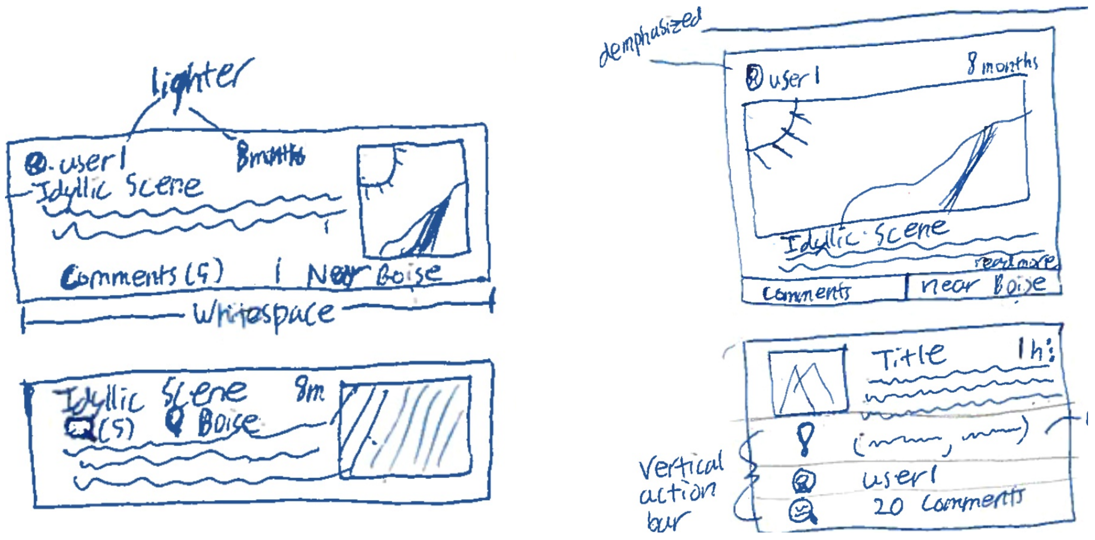
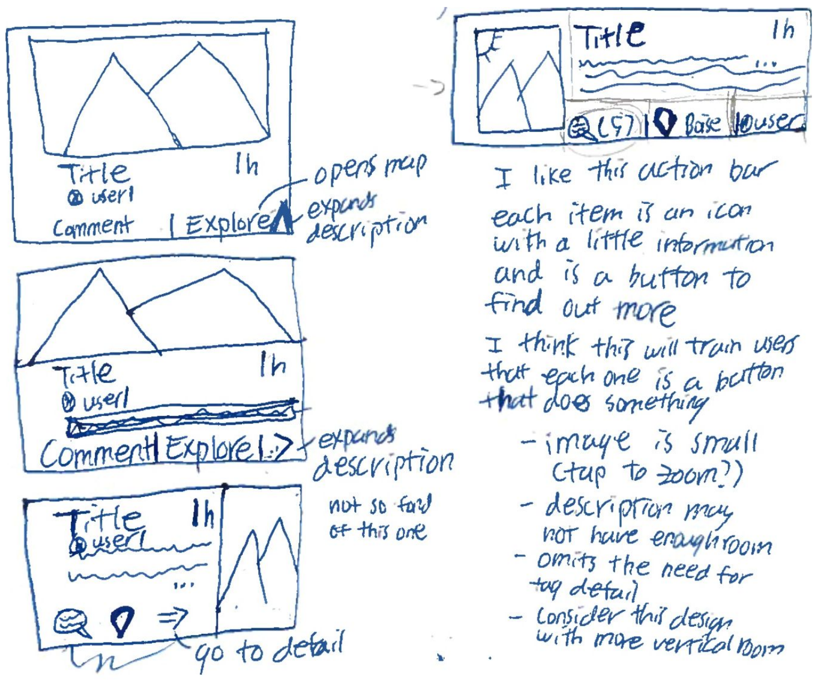


Figure B.1: Card Prototypes

and instead allow the user to expand the image by pressing on the smaller image. This idea of on-demand expansion was also used in displaying the tag's description. Researchers wanted to place the entire tag description on the card, to prevent the user from navigating away from the tag list. An shortened description appeared on each card which could be expanded by pressing a button located on the card.



I like this action bar

each item is an icon with a little information and is a button to find out more

I think this will train users that each one is a button that does something

- image is small (tap to zoom?)
- description may not have enough room
- omits the need for tag detail
- consider this design with more vertical room

Figure B.2: Additional Card Prototypes