March 2015

WanderLog Travel Application

Benjamin Onil Senecal
Worcester Polytechnic Institute

Cassandra Lynne Hamlin
Worcester Polytechnic Institute

Christina Jane Aiello
Worcester Polytechnic Institute

Project Number. GFP1404

WANDERLOG
A Major Qualifying Project Report:

submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

_____

Christina Aiello

_____

Cassandra Hamlin

_____

Benjamin Senecal

Date: March 6th, 2015

Approved:

_____

Professor Gary F. Pollice, Major Advisor

Abstract

WanderLog is a travel journal/scrapbook application developed for Android™ Devices. This is an application that can be used by study-abroad students or other travelers to record their experiences. The application gives users the ability to create “timelines,” which are the equivalent to journals or scrapbooks. Each timeline can contain one or more “entries,” which are the equivalent of pages in a journal. Users then can insert photographs, videos, audio recordings, and text into entries.

Acknowledgements

We would like to thank our advisor, Professor Gary F. Pollice, for guiding us throughout this project.

We would also like to thank all of the individuals who took time out of their schedules to test and provide feedback about our project.

Lastly, we would like to thank the authors of all projects and other documents that we used as references throughout this project (seen in our references).

Table of Contents

List of Figures

List of Tables

1 Introduction

Fifty-six percent of American adults owned smartphones in the year 2013, according to the Pew Research Center (Smith, 2013). People now more than ever are looking to access the Internet, take photographs, record videos, write notes, and much more while on-the-go, making mobile applications for smartphones increasingly prevalent in today's society. Many popular websites today either have mobile-friendly websites and/or specifically have created applications for mobile users, further adding use to mobile devices.

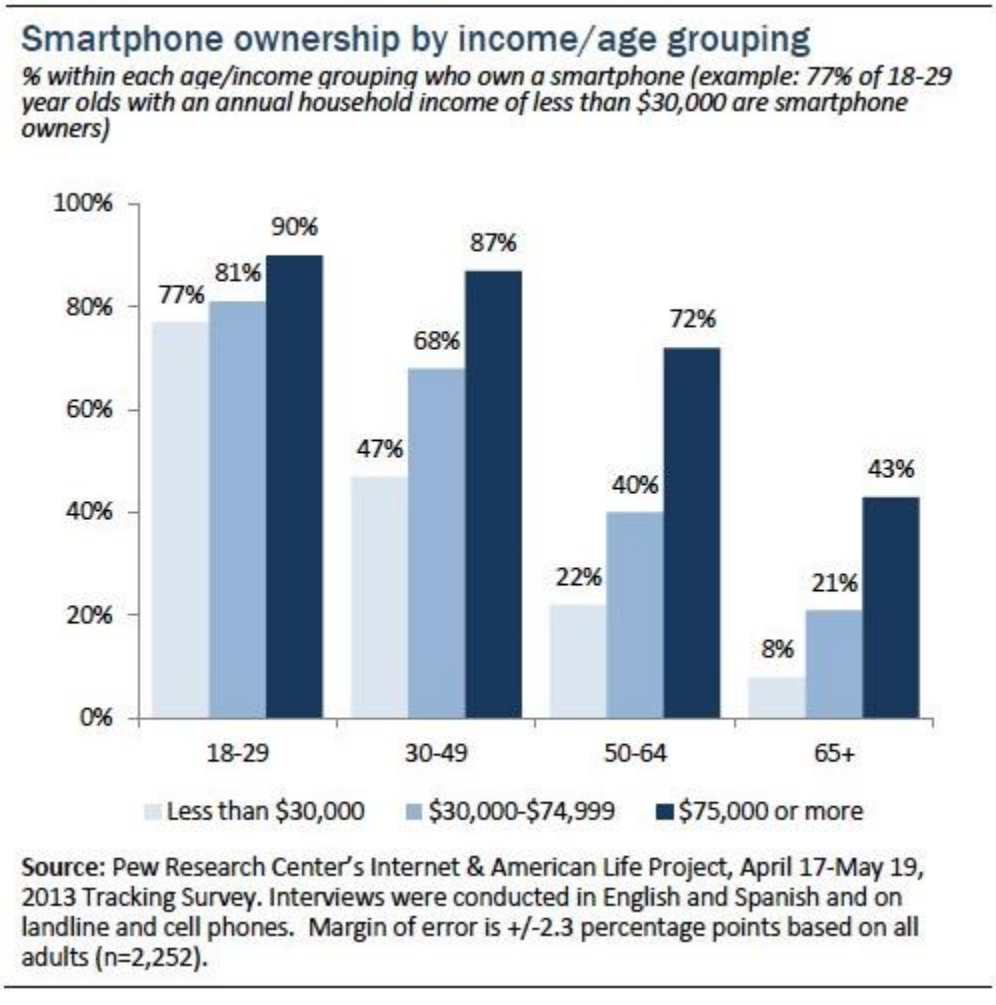


Figure 1.1: This graphic shows the percentage of individuals in various age groups and yearly income groups who own smartphones (Smith, 2013).

Smartphones serve as an ‘all-in-one’ device, offering the functionality of a camera, journal, notebook, and audio recorder all in one unit. Individuals who travel can use smartphones to make notes, take photographs, record videos, and record sound clips of the world around them. This additional functionality provided by smartphones is convenient for travelers, as it eliminates the need to carry around several different devices.

Travelers want to preserve the sights, sounds, and experiences for personal viewing. Travelers also want to share their experiences with friends and family. Students at Worcester Polytechnic Institute who participate in Major Qualifying Projects (MQPs), Interactive Qualifying Projects (IQPs), and Humanities projects have the opportunity to travel to off campus locations and are often required to keep a travel journal of their experiences.

WanderLog provides the ability to combine numerous types of media into a clean, organized, travel-focused record that allows for easy and enjoyable reflection upon one’s travels. The application lets the user create what are called “timelines”, where each timeline is roughly equivalent to a journal or scrapbook. Like any good journal, the timeline has many pages, called “entries”, and each entry can be filled with various types of media including photographs, videos, text, and sound clips. By defining timelines, entries, and media, WanderLog provides an easy-to-navigate structure for recounting a user’s explorations.

Presently, people often use websites and applications such as Flickr/Flicka™, Facebook™, or Twitter™ to share their travel experiences. While these options let user share media and text, chronologically grouping all the relevant content is not always possible in these applications. In addition, these applications put more focus on adding information and less focus on organizing the information. When creating media, a lot of current applications only offer one or two types (usually pictures and text) By having the ability to include more types of media, WanderLog has greater flexibility in how time spent traveling is recorded by the application. Our project group took notes on advantages and disadvantages of current options when designing our application.

2 Background

Prior to beginning development, we researched existing travel oriented applications and the processes and standards of Android™ programming. This section of our paper outlines our findings regarding similar applications and major competitors, and then continues on to explain our choice of the Android™ platform. After choosing the Android™ platform, we conducted more research into the Android™ lifecycle and application testing on Android™ devices. We also include in this section research relating to media types available to use on Android™ devices.

2.1 Similar applications

Numerous applications currently exist that offer the functionality of storing pictures, video, and text. However, none of the current options offer a consolidated, trip-focused, offline method to display information. Applications similar to our project can be grouped into the following categories:

Table 1: This table contains various categories and example applications that are similar to our project.

Category	Example
Photographs, videos, and text	Facebook, Tumblr
Note-taking	Evernote™
Scrapbooking applications	Google+™ gallery, ScrapPad

The categories listed above fall short in that they provide the ability for a user to preserve and manage only a couple of types of media.

In contrast to note-taking applications that are used for lists and reminders, we created our application for individuals who travel and want to document their trips. In comparison to popular social media tools such as Facebook and Twitter, WanderLog presents media such as photographs, videos, and text in a more organized in a more hierarchically structured fashion. The purpose of this application is solely for documenting individual traveling experiences rather than being used to post any and all unrelated media. Below we assess major competitors for our application.

2.1.1 Major competitors

Many applications exist on both Web and mobile platforms that let users post and share different types of media. The two major services that provide functionality similar to our application are Facebook and Tumblr™: social media websites with mobile applications. Both Facebook and Tumblr are widely used websites that allow users to post, group, and connect several types of media; however, both of these services require an Internet connection to create posts, which can be problematic for someone who wants to make a post while not connected to the Internet.

With 1.15 billion active users per month, Facebook is the largest existing social media network. Its users can post text, photos, and videos to their timeline, where they are displayed in chronological order. Though it includes the types of media also found in our application, Facebook does not offer means of organization beyond the timeline. Videos are displayed independently of everything else, though they can have text descriptions. Photos can be grouped into albums and can have captions, but there is no way to group individual posts into a cohesive “article.”

The dilemma of organizing posts into a larger group also exists in Tumblr. There are several media options available for a post on Tumblr including text, video, photos, and audio. Like a Facebook timeline, posts on Tumblr blogs are often displayed chronologically with the most recent content first. Tumblr offers slightly more flexibility for organization than Facebook. A handful of photos can be grouped into a photoset post, as opposed to keeping all photos in one all-encompassing album. The Tumblr blog can also be manipulated by putting a customized theme on your blog that influences how posts are displayed. Though there are more ways for a user to manipulate how their posts are displayed, there still no way to group all content posted into an organized hierarchy.

Evernote is a mobile application with a Web interface centered on creating notes. This application excels in its note taking capabilities, letting users put text, pictures, video, audio recordings, and attach files into a note. It also has the ability to write a note with a stylus or one's finger. Evernote also offers the ability to organize notes within collections called notebooks. Many users of Evernote like the large set of features that application has; however, some users do not like how the features seem to make simple tasks harder as more features are added ("Reviews"). Evernote also does not offer as many features for organizing within the note. All forms of information within the note are placed in the note and left there for the user to rearrange things within the note manually. Drag and drop features do not exist for simple rearranging of information within a note.

ScrapPad is a mobile application that allows a user to create his or her own scrapbook using media from his or her phone. The application offers options to apply borders to pictures, use multiple font styles for text, and lets the user create multiple scrapbooks within the application. Reviewers of this application cited the application's ease of use and multiple options for designing ("ScrapPad - Scrapbook for IPad"). There were qualities of this application that users did not enjoy, such as not having a way to back up these scrapbooks created within the application, in addition to not having an auto save feature that prevents data from being lost if the application crashes before the user can save his or her work ("ScrapPad - Scrapbook for IPad").

Figure 2.1: Shown here is the ScrapPad application ("ScrapPad - Scrapbook for IPad").

A summary of the positives and negatives of each aforementioned application can be seen below:

Table 2: This table summarizes positives and negatives of each aforementioned application.

Application	Pros	Cons
Facebook	Allows for text, photo, video	Cannot organize different types of media together. Photographs can be grouped into albums, but photos and videos (for example) cannot be grouped together.
Tumblr	Allows for text, photo, video, audio	Does not have a means of grouping all content posted into an organized hierarchy
Evernote	Allows for text, photo, video, audio, attaching of files. Users can handwrite notes. Can organize notes into "notebooks."	Does not support drag-and-drop reordering of a note's content.
ScrapPad	Allows for text and photos Can add borders and stylized text to pictures Can create multiple scrapbooks	No way to back up scrapbooks (if device is damaged, cannot retrieve data) No automatic saving feature

2.2 Why our application?

Our application does what current applications already on the Google Play Store do not do quite as well – act as a multimedia travel journal for people to record their experiences in one organized, focused place. This application supports common forms of media such as text, pictures, audio, and video. It also supports organization by date and trip, allowing users to go back through an organized account of everything they did to find exactly what they are looking for with ease. The application also focuses on a

simple and efficient user experience so the user spends less time on organization and more time logging travel experiences.

One of the groups that this application is targeting is Worcester Polytechnic Institute (WPI) students completing an Interdisciplinary Qualifying Project (IQP). The IQP is a project where WPI students try to solve social issues using the technical and research skills they have learned. For this project, students have the option to go abroad to an off campus location to complete their project. A student who chooses this option would benefit from an application that allows for easy logging and sharing of travel experiences. In addition, every IQP team must write a paper detailing their experience, and this application can help students easily recall their travels.

2.3 Android™ smartphone background

2.3.1 Choosing a Mobile Operating System

In order to create an application, a mobile operating system (OS) has to be chosen. The ideal choice of a mobile OS would have a large amount of people who use it, so the application would be easily available to a large audience, and the OS would also have to be low development cost.

The first criterion desired from a mobile OS is a large user base. Since this application was designed to log travel experiences, it would most likely be run on a smaller, more portable device that can easily be carried with the user. The device that best fits this category is the smartphone or tablet. We considered two popular mobile operating systems, Android™ and iOS™, for the development because any other options would have too low of a presence in the smartphone market to be easily deployable to a large

group of people. In Q2 2014, smartphones running the Android™ OS made up the largest smartphone OS market share with 84.6% of smartphones shipped running Android™, while smartphones running iOS™ made up 11.9% of the smartphone market (Mawston, "Wireless Smartphone Strategies"). Therefore, an application that is easily accessible to the majority of smartphone users must run on Android™.

The second criterion is a low development cost. Starting with the cost of making an application public, the cost to host an Android™ application on the Google Play Store is a one-time fee of $25 to create an account ("Get Started with Publishing"). The cost to develop in Android™ is free because Android™ development is done in either Android™ Studio or Eclipse, both of which have free downloads for Windows, OSX, and Linux. In comparison, iOS™ application development costs $100 a year for a developer program to host applications on the Apple Application Store. In addition, an iOS™ application can only be programmed on an Apple computer, so there would be additional costs getting equipment necessary to produce an iOS™ application even though that software for it is a free download ("Apple Developer"). When considering costs, Android™ still comes out on top as the OS to use for mobile application development and was, therefore, the mobile OS chosen for our application.

2.3.2 The Android™ Life Cycle

The Android™ life cycle is a series of stages that all Android™ activities go through at different points when being using on a device. An Android™ activity is essentially a window that the user can interact with. The stages an activity goes through are accompanied by a series of methods that are called at each stage in the life cycle. The

different stages and methods that are called to transition from one stage to another are displayed in the figure below.

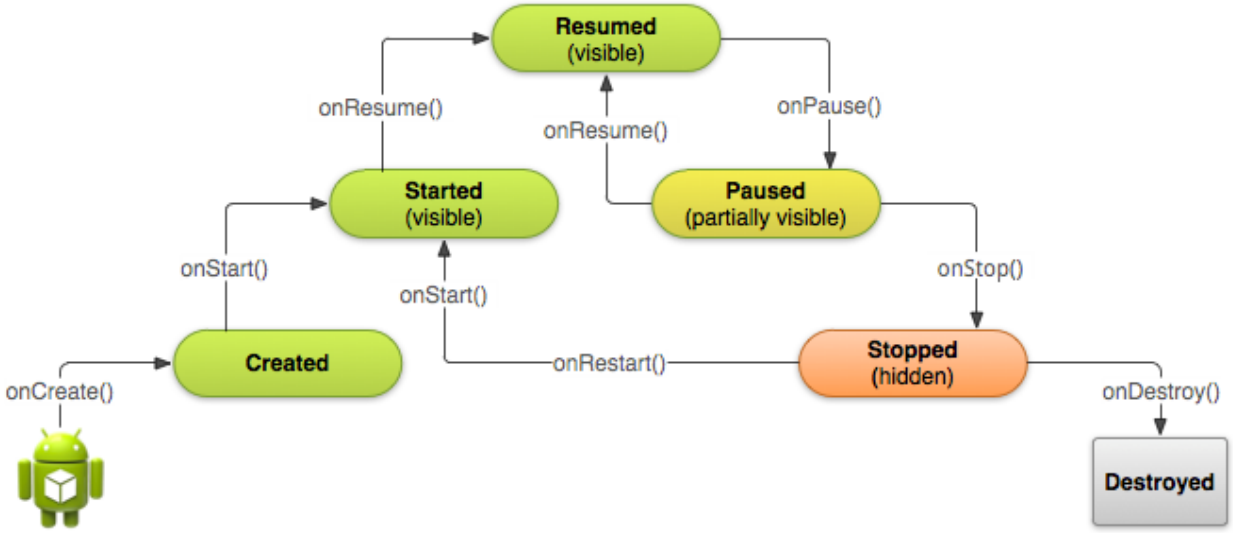


Figure 2.2: This diagram explains the Android™ life cycle ("Starting an Activity").

At the very beginning of an app's life cycle, the onCreate() method is called when the application is being opened. The body of this method is where some of the initial one-time things get done like creating or linking to a database. After onCreate() is the onStart() method. The onStart() method is called before the application becomes visible, so a user interface should be created during this method call. After onStart() is the onResume() method which mainly recreates any saved states that the user may have been in beforehand. At this point in the application lifecycle, the application is fully visible and the user can freely interact with it.

While the activity is visible to the user, there are actions the user can take to make the activity not be visible anymore. If the user does something that causes another activity to be displayed in front of the current activity, then the onPause() method will be called on the activity that is moving into the background. Since the activity is losing focus, pieces of information necessary to save state should be saved, like form entries or

unsaved draft emails. If the user goes back into the application, the onResume() method would get called to restore the saved state. On the other hand, if the user decides to do something like switch applications or go to the home screen, then the onStop() method will be called on the current activity. At this point, the application may never come back into the foreground, so all states should be saved here, and all resources that are not necessary should be released.

In addition to the user, the Android™ OS can also affect the activity life cycle. One situation where this is the case is if the OS ever runs low on memory and needs to free some up. it may call onDestroy() on the application which would completely close the application and its activities. Everything should be closed and released during the onDestroy() method call. If the onDestroy() method is not called on the application, and the user goes back into the application, the onRestart() method will get called followed by the onStart() method, then the onResume() method. The only new step here is the onResume() method which will handle things that may need to be restored from a previous state as a result of calling onStop().

2.4 Media on the Android™ Smartphone

There are multiple types of media available on Android™ devices: Users are able to record audio, take photographs, record videos, and write text. Each type of media provides a different way to record an experience.

Video and photographs can be taken with the default camera application on an Android™ device, in addition to being able to download pictures and video from websites, email, and text messages. Android™ developers have the option to either allow users to make use of the existing camera application on the Android™ device or directly access the camera hardware to create one's own camera application. Text can be written in numerous applications, one of which is the default note-taking application available on Android™ devices.

There is no default sound-recording application on Android™ devices. There are, however, ways for developers to allow users to record audio. In addition, there are options that allow applications to control an application’s volume, respond to hardware audio key presses, and manage audio focus ("Building Applications with Multimedia").

2.5 Testing Android™ Applications

Testing is an important part of the software development lifecycle and is integral to the success of an application. For Android™, a particularly valuable method of testing is a unit testing framework called Robotium. Unit tests are small tests that allow a programmer to individually test parts of his or her code. Unit tests can detect regressions, which are software bugs found after modifying code, in addition to

continuously asserting that previously-written code functions in an expected manner.. Don Wells, author of an Extreme Programming website, says that "during the life of a project an automated test can save you a hundred times the cost to create it by finding and guarding against bugs" (Wells, 2013). The purpose of unit testing is to "guard your functionality from being accidentally harmed" (Wells, 2013), if either the author of the tests or another individual modifies the code being tested by the unit tests during the refactoring process. Unit tests can help a programmer see if changes to the structure of a program also affected the program's functionality.

The testing framework called "Robotium" is an automated unit testing framework for Android™ Applications. This testing framework allows for function-testing scenarios, system-testing scenarios, and user acceptance-scenarios (Reda, "User Scenario Testing for Android™"). Some benefits of using this framework are fast test case execution, the ability to handle multiple Android™ activities automatically, and easy integration with various build automation tools such as Ant, Gradle, and Maven (Reda, "User Scenario Testing for Android™"). This testing framework supports Android™ API level 8 and above, and it can be used both for testing on real devices and on the Android™ emulator (Reda, "User Scenario Testing for Android™"). Tests can be run from the command line or within a design studio such as the Android™ Design Studio.

2.6 Conclusion

Competing applications, reasons individuals choose Android™ devices, and components of the Android™ device are all pieces that were taken into account when designing this project. While numerous competing applications exist, no application

provides the exact functionality desired for our needs. The successes of competitors, in addition to their pitfalls, were considered when designing this project. After deciding on the Android™ platform, we researched several hardware and software components of Android™ devices in order to familiarize ourselves with the methods necessary to design the structure of our application.

3 Methodology

In designing and developing this application, three areas of importance needed to be researched in detail and put into practice within our project: the user interface, the structure of the database, and the ability to implement testing. Android™ applications put a heavy emphasis on certain important user interface design concepts, which we researched and implemented within our project. In addition, choosing a proper database and structure for that database was important not only for the current development for our application, but future development as well. The chosen structure could affect the difficulty of developing a website to accompany the mobile application in the future. Last, testing is an integral part in identifying and preventing bugs in software, so we researched multiple methods for testing our product.

This chapter outlines which processes and procedures we used in development and how we used them. We start by discussing our user interface design process and Android™ design principles, and then move on to outline the development of the database. Finally, we describe our methods of testing different portions of our application.

3.1 User Interface

The design principles for Android™ applications centralize around three goals: Enchant Me, Simplify My Life, and Make Me Amazing ("Design | Android™ Developers"). The objectives of these goals exist to enhance the experience of Android™ users and ensure that an application is simple and easy for them to use. People will be more

responsive and more accepting of applications that involve less cognitive work and make them feel like they know what they are doing. Therefore, the easiest way to create a positive user experience is to keep screen layouts as simple as possible and to only present information necessary to the user at the time. The Android™ design principles suggest breaking information into small tasks and chunks to avoid overloading the user with too much information. In order to decide how to split up tasks into different screens, we first drafted user stories and use cases based on the desired functionality of our application, such as adding a photo to an entry from the photo gallery or titling a timeline. We then determined which tasks would be performed on each screen based on a logical flow (entries that are part of timelines are obviously created from within a timeline), and the structure of similar applications discussed in the background.

The analysis of several applications that have similar functionality to this project led to several design decisions and helped us in developing an information flow that would make sense to users. While determining how to group tasks, we looked to applications that have similar information grouping, such as Evernote and Facebook, to see what organizational mechanisms were commonly used.

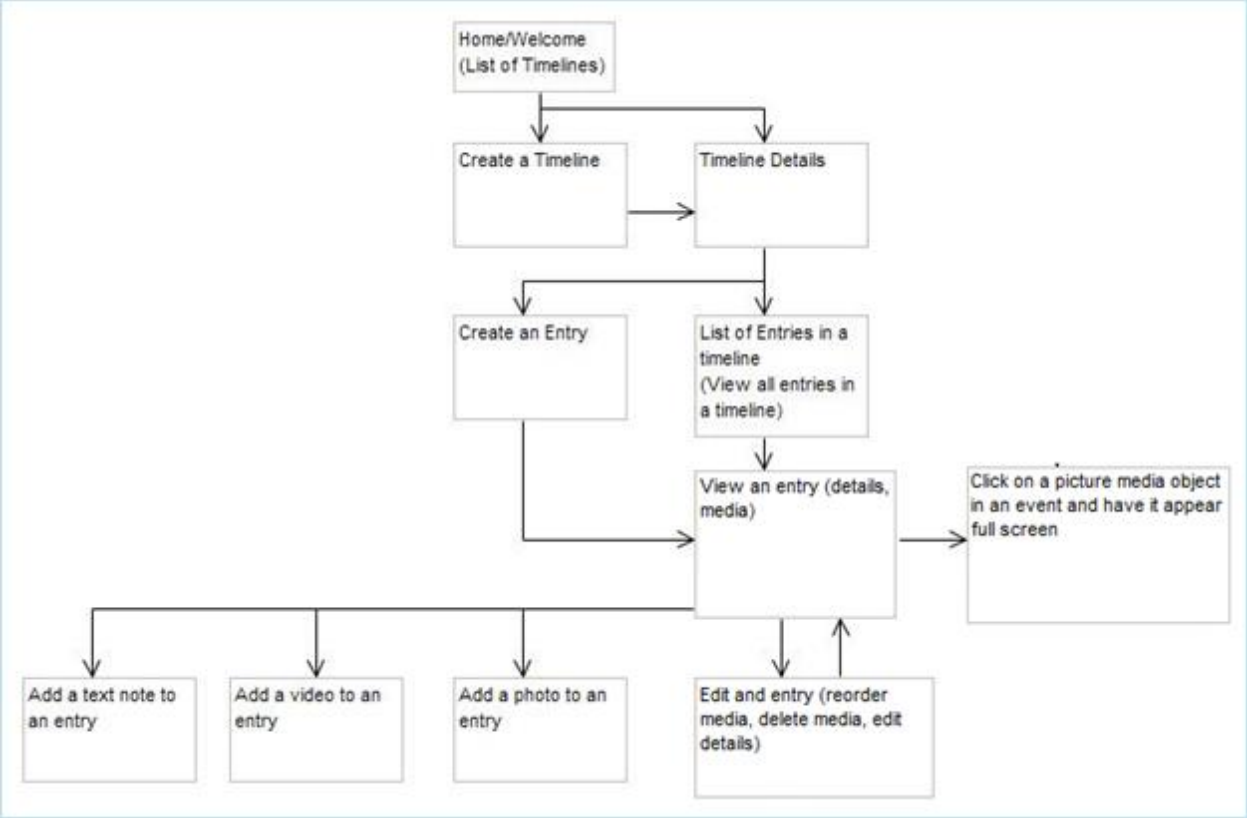


Figure 3.1: Flow diagram for our application.

The Android™ Developers Guide describes relationships between screens as descendent or lateral. Descendent screens involved navigating down a screen hierarchy, or in the case of our application, navigating to an entry from a timeline. Lateral navigation is navigating between a collection of sibling screens of the same type (i.e. a group of timelines or a group of entries).

One of the suggestions in the Android™ Developers Guide, and also the most common way we found both lateral and decedent navigation implemented, was in a list view. Thus, our timelines and entries are displayed in vertical lists, and the users navigate down the information hierarchy by selecting one of the child items in the list. To handle up navigation between entries and timelines, we rely on Android™'s back button, which

was also a common trend we saw in many Android™ applications, as the developers guide heavily stresses the importance of maintaining look and feel of the Android™ design standards.

We also made use of Android™'s standard action bar for additional navigation. Instead of using buttons on the screen for common or universal tasks, Android™ suggests using icons in the action bar, especially when tasks require moving to a new screen. We put the functionality of creating, editing, and adding content to timelines and entries in the action bar in order to remove clutter from the content screens and to prevent users from having to search for a button to accomplish one of these tasks. We include a cancel button on the action bar whenever a user is in the edit screen for a timeline or entry. This eliminates the necessity for the user to use the back button to reach a previous screen, as it is mostly used for navigating up a hierarchy, which is not the case in the instance of creating a new object or editing an existing one.

Android™ user interface design should not only implement familiar navigation to standard Android™ applications, but should also incorporate a similar look and style. There are two standard Android™ themes: Holo Light and Holo Dark. We started with the Holo Light theme and customized our color scheme from there, keeping the application design simple by minimizing color selection. According to Google design guides, good design uses no more than three hues of a primary color, along with an accent color ("Google Design Guidelines"). In addition to adhering to design principles, we also wanted to create a brand for our product. While color psychology is an important aspect of branding a product, research has shown that a person's personal

experience has too much impact on how they perceive a specific color to confidently conclude that any one color will evoke a particular emotion. It is instead most important to create a brand that people will recognize and that includes colors that are appropriate for the brand (e.g. something that is supposed to be seen as rugged will likely not be associated with pink). (Bottomley, "The Interactive Effects of Colors and Products on Perceptions of Brand Logo Appropriateness."). The color pallet we chose uses blue as the primary color with green as an accent. Since our application is geared towards travel and our logo incorporates an image of a globe, we used colors that are most often associated with maps of the earth.

In order to ensure that our product was usable in practice, we conducted user studies. Usability testing is essential to software design, as it ensures that the people who will be using a product are able to figure out how it works and use it with ease. The study conducted consisted of nine tasks (listed in Appendix B) that walked the user through the process of creating a timeline, creating an entry, and adding media to that entry. Users also completed tasks relating to editing and deleting various items. Each task was timed in order to gauge how difficult it was for users to complete it. After a user completed a task, they gave any relevant feedback they had, such as if the task was easy to complete or if it was confusing, and why.

3.2 Database

Data storage in an Android™ application is vital to the success of many applications, and with users of WanderLog recording travel experiences, storage of data is an important aspect of this application. We were then faced with two questions: which database should we use and how should we structure it? The chosen database would have to be well-suited for the kind of data that will be stored on the device, which is partially dependent by how much information the user wishes to put in it. After choosing a database, the structure would be decided based on the information to be stored in it. In this case, our application has timelines, entries, and media. In addition, entries are linked to a timeline, and media to an entry. The relationships between elements in the database must also be reflected in the structure. A properly-structured database allows for ease of an application's development. Two popular databases for mobile applications are SQLite and CouchbaseLite. The Android™ Developer Website provides extensive documentation for integrating SQLite into an Android™ application. CouchbaseLite is a more popular noSQL database that is capable of running on mobile devices with limited resources.

We first researched advantages of an SQLite database. The extensive support on the Android™ Developers Website was a big advantage over CouchbaseLite. SQLite would also be easier because of our group members' past experience with SQL. These benefits, however, do not come without drawbacks. One of the first drawbacks is that SQLite would require a more effort to set up than the CouchbaseLite database. Set-up would require setting up all the tables in the database. Since creation would be through

Java, that would involve the creation of a contract class which essentially sets up the structure for that database. Like the name suggests, the contract class would set up a contract that the database tables and the user queries would have to follow. One of the other drawbacks to using SQLite is that the database does not come with any built in functionality for syncing with external SQL databases.

We then looked at CouchbaseLite. Being a noSQL database, CouchbaseLite would offer a structure-free implementation of a database. A lack of structure is advantageous because allows for the insertion of free-form data without much change to the structure of the database. Another advantage of using CouchbaseLite is that it comes with built in support for syncing with other Couchbase databases, which would be ideal if a Web application is to be made in the future to work with the mobile application. Like with SQLite, the advantages come with a cost. One of these costs is the lack of information available on the Android™ Developer Website. All support for using Couchbase would have to come from the Couchbase website or other sources on the Internet. Another disadvantage is that Couchbase would require more organization. The lack of structure makes the database very adaptable, but it is also easy to make mistakes due to the lack of imposed structure in the database. Lastly, none of our group members have had previous experience with noSQL databases, meaning it would be slow to get started with Couchbase (but it would be a much better learning experience than SQLite).

When it came time to make a decision, we decided to use CouchbaseLite as the database that would back our application. The advantage of it not being innately structured made the database well suited for the storage of timelines, entries, and

media which are more free-form types of data. On the other hand, the cons for using CouchbaseLite would not delay our project by an unreasonable amount of time. Once we decided on CouchbaseLite, the next challenge was to learn about document-based databases and figure out how to structure the database.

After deciding on CouchbaseLite, we researched and familiarized ourselves with it. The Couchbase online documentation for Android™ devices turned out to be the most useful resource for setting up the database. It offered detailed instructions for including CouchbaseLite in an Android™ project in eclipse as well as some code to get a database started. The Couchbase online documentation also provided us with general tips for using a noSQL database, which was helpful after we set up our database.

One of the more challenging aspects of this project was choosing the best structure for the database. The structure for the database went through an iterative process, starting with a basic structure and being continuously revised until the final structure was made. The first appearance of the database in the application was with a basic text application – all this application did was let a user store text and then display a list of all the stored text in the database. This initial database had very similarly structured documents. They all had a key, “text”, which corresponded to a value which was the text inserted by the user. Such a simple database structure did not last long, but it did accomplish the goal of incorporating CouchbaseLite into the application.

The next step was adding support for pictures. In order to support different stored content, the “TYPE” key was introduced. This key would have a value of either picture

or text and could be compared to determine what type of content is stored in that database document. Once the document type is known, the calling method would know whether to look at the “text” key or the “picture” key and whether text or a picture would be returned. We also had to figure out how to store pictures in the database. The resulting process was to store a Uniform Resource Identifier (URI) of the picture and use that to load the picture when it needs to be displayed. By using the URI for the picture, the application can drastically reduce the size of the database by storing something that points to where the picture can be found rather than storing the whole picture. Further in this paper, we will discuss how these images are displayed.

With more than one type of document in the database, it became necessary to start creating views on the database. Views are similar to SQL indexes – They are a subset of the data in the database that can be queried. To look up data more easily, a view for text documents and a view for picture documents were created by coding map functions that will look at the value corresponding to the type key and add appropriate content to the views. The keys in the views consisted of either the text or the picture URI depending on the view. This way, the content could be displayed without any processing of documents since the desired content was already stored in the key. With these views, a list that displays all the text in the database would just have to query the view with all the text documents in the database and display the keys of all the results. After basic types of media could be successfully inserted and read from the database, the next step was to incorporate timelines and entries into the database.

Timelines are the largest structure that represents a whole trip. Within a timeline would be several entries with each entry corresponding to some event in the timeline, and there would be media such as text or pictures inside an entry. The next challenge was to make the database reflect this relationship between timelines, entries, and media. First, we created timeline documents which stored the general data a timeline would have like a title. Next, we created a view over all the timelines so timelines could be easily queried. Entries in media followed media similar process. Now that all the content could be stored in the database, the relationships between timelines and entries, and entries and media were next. We added a timeline key to each entry document to make an entry part of a timeline. The value of this key would correspond to the timeline that the entry is part of. Adding the timeline key to entries created a link between entries and timeline, but the link was not reflected in the views. We used the timeline unique identifier as the key in a view of entries to fix this. This way, the view could be queried for a specific timeline unique identifier, and the results from the query would be all of the entries that are part of the given timeline. To make the rest of the entry data accessible, the values returned from a query consisted of the documents containing the rest of the information for the entries.

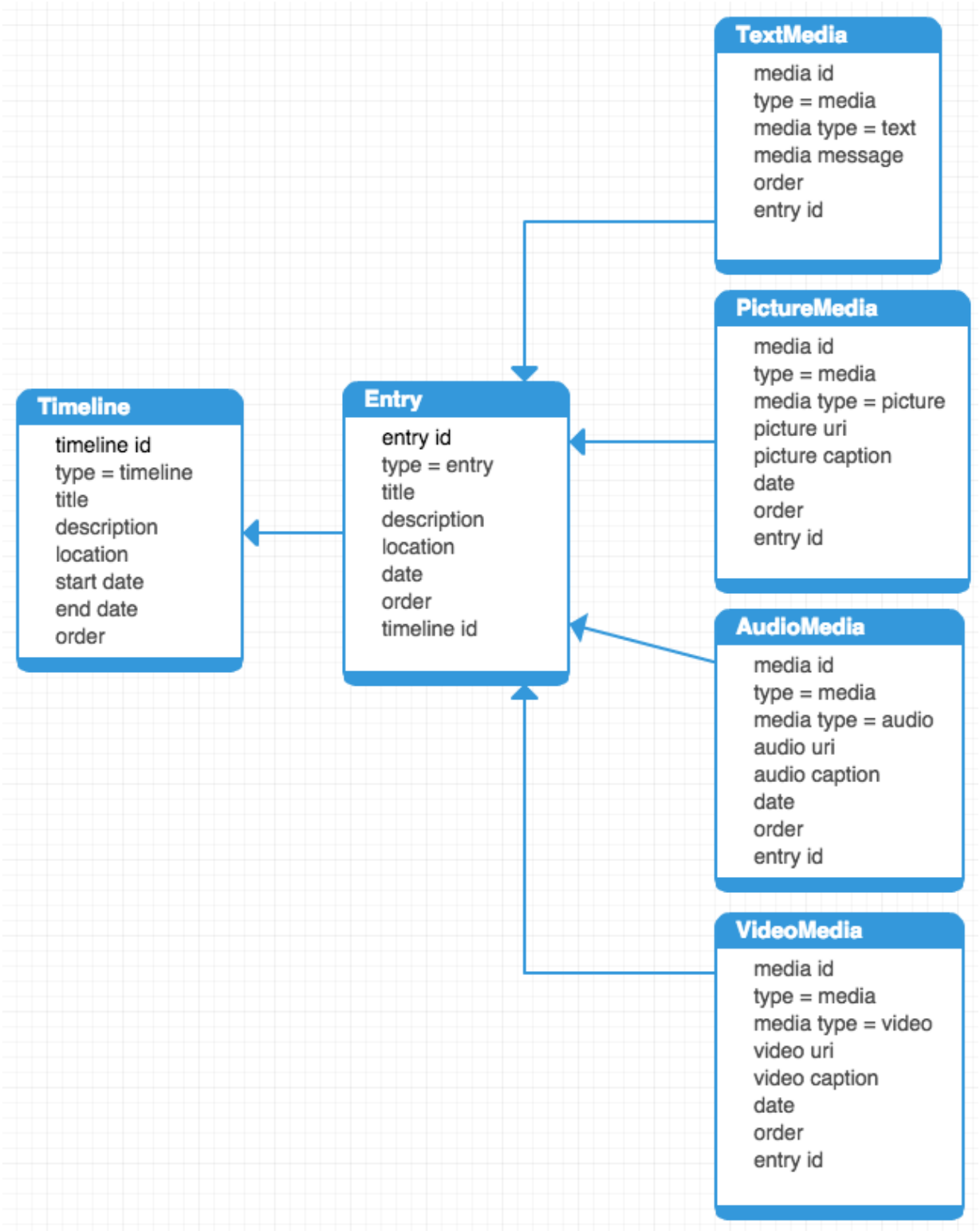


Figure 3.2: The database structure for WanderLog

After creating an entry view, a media view had to be created. We originally created a view that consisted of both text and picture documents, but then realized this was not an accurate enough representation of the data. Both text and pictures are media, so entries should contain media, which could be either text or a picture. To accomplish this, we changed the type for text and picture documents to media and added a media type key to these documents to distinguish between text and pictures. With text and pictures encapsulated into the media document type, coding the view for media became very similar to the view for an entry; there would be a key for the entry that the media belongs in. The unique identifier of the entry that the media belongs to would then be used as the key for the view of media and the document would be used as the value.

The database structure at this point in time held up well for a couple of development cycles and worked well when we added the picture with caption type. However, when we decided to add ordering to timeline, entries, and media, an order had to be added to all the database documents. At first, adding an order to the document seemed sufficient, but then a way to order the information in the list view would have to be created. To solve this issue, we took advantage of the default ordering behavior of CouchbaseLite views. The views automatically order by the key. To order documents, we had to change the key to something that would represent the order. For timelines, we changed the key from the timeline name to the timeline order, which lead to timelines being ordered by creation order.

After ordering timelines, the process of ordering entries became a little more complicated. For entries, we wanted to order them by the date given to that entry.

Unfortunately, the keys for the entry view already serve the purpose of identifying the timeline each entry is part of. Swapping out the key for the order number was not be an option because then the entries would not be organized by timeline. We needed the entries to be ordered by both the timeline information and the date. To solve this, we used compound keys: The compound key is a list of items, and the keys in the view are first ordered by the first item in the list, then, in the event of ties, the second item is used to order the keys further. This process continues until the whole list has been used to order the keys. To take advantage of this, we used a list with two items as the key. The timeline unique identifier in the first item in this list, and the entry date formatted to be in the form year-month-day is the second item in the list. This made the entries ordered first by timeline and then by date if there is more than one entry in a timeline.

In order to support reordering media, a different process was necessary. Users have the ability to reorder media, so the order would have to be based off of an order number. After adding an order number to all media documents, the same process we used to order entries could be used. The key would be a list where the entry the media is part of is the first item in the list, and the order number for the media in the form of a zero-padded string is the second item in the list. With this system in place, the database supported reordering of media, and the application had to make sure changing the order numbers of documents was done correctly.

Table 3: A table of all the views WanderLog uses

View Name	View Key	View Value
timeline View	[timeline order]	timeline document
entry View	[timeline id, entry date]	entry document
media view	[entry id, media order]	media document

3.3 Testing

Testing is an integral part of software development. Testing is, as described in the Microsoft Developer Network guide on testing, “a proxy for the customer” (“Testing in the Software Lifecycle”). It is a method of extensively testing each component to assure that the product aligns with the expectations held by project stakeholders. Microsoft supports this by describing tests as “executable requirements” (“Testing in the Software Lifecycle”). Unit testing, one method used in software development, is used to test methods, classes, or procedures. Unit tests allow a developer to test the smallest possible pieces of code and monitor those pieces’ behavior.

One successful method for unit testing software is JUnit, described as a “simple framework to write repeatable tests” (“JUnit Wiki”). Repeatable tests allow for continuous assessment of functionality, letting developers see if any additions or alterations made to existing code have caused a difference between what is expected of the code and what the code does. This can help developers identify problems and begin plans to resolve them.

A major component of JUnit tests are assertions: These are the “bread and butter” for unit testing (“JUnit Wiki”). Assertions can access whether something is true, false, null, or not null, for example. These assertions can be used on primitive types, objects, and arrays. JUnit also allows developers to create Exception Tests. These tests “specify expected exceptions in unit tests” (“JUnit Wiki”). Our project group has chosen to make

use of JUnit testing for any classes within our project that are not classes for the user interface of the application.

One unit testing suite that exists for Android™ devices is the Robotium Test Automation Framework. As described on their project homepage, “Robotium makes it easy to write powerful and robust automatic black-box UI tests for Android™ applications” (Reda, “User Scenario Testing for Android™”). This framework allows developers to write automated user-interface tests for major Android™ components, such as action bars, activities, buttons, date pickers, lists, and text boxes. These tests can be function tests, system acceptance tests, and user acceptance tests. The framework allows for unit testing on multiple Android™ devices, which benefits our project because our application is meant to be used on any Android™ device (phone or tablet). This particular framework also allows for multiple Android™ activities to be tested automatically, which is beneficial for our project because we have multiple activities running within our application. We have implemented Robotium tests within our project, which allows us to repeatedly test the functionality of our application from a user’s perspective. These tests allow us to not only test the activity that is presently being viewed, but also to test that activities successfully transition from one to the other by clicking certain elements on the screen.

In conclusion, our project group researched three areas of importance in detail to make the best decisions for the application in its present and potential future form. We conducted research related to the user interface, the structure of the database, and the

ability to implement testing. Once we conducted this research, we implemented these practices and ideas within our project.

4 Results and Analysis

The resulting application of this Major Qualifying Project (MQP) lets a user compile a series of events using a variety of media types (picture, audio, notes, and more) into a simple event timeline that can be navigated with ease. In this section, we discuss the metrics we used to analyze our final project, starting with the usability testing results, and then different statistics pertaining to our code.

4.1 Usability Results

The results from our usability testing indicate that the overall structure of the application is logical to users and that many of the features function as expected. In general, once users figured out how using edit mode and the information structure worked, they found it easy to navigate the given tasks. One issue that all users made clear was that their first instinct for editing a piece of media, specifically adding a caption to a picture that is in an entry, is that their first instinct was to tap on a picture to caption it. This, among other results from our user study, can be seen in the table below.

Table 4: Summarization of users' problems or suggestions for our application

Problem or Suggestion	Number of Users
User clicked on a full-screened image to try to add a caption	13 out of 13
Clicked start and end date text when editing entry (rather than calendar icon)	1 out of 13
Did not know how to save an edited timeline	1 out of 13
Thought that edit mode would allow user to add media items to a timeline	1 out of 13
Desired a confirmation message when deleting a timeline	4 out of 13

This user testing allowed for us to access our current product and acquire suggestions from users as well. One repeatedly-made suggestion from users was to have confirmation for deleting timelines, which we have incorporated into our code. The raw usability data is included in Appendix B.

4.2 Source Code Statistics

By the conclusion of the project, our project group wrote 11,174 lines of code. 9,960 of these lines are project code and 1,214 lines are test code. Our project contains four packages, 31 classes, and 54 tests. While working on our project, our group made 301 commits to Git.

5 Future Work and Conclusions

Future development for this project might involve further improving the application itself or creating a website that would work alongside the application.

One possibility for future work would be the creation of a website interface that would sync with the Android™ application. A Web interface would allow for easier sharing, a popular idea in today's culture. This would require insertion of pictures into the database (rather than picture URIs, which are what is currently stored in the database) and the creation of user accounts. In addition to allowing for syncing, a Web interface would allow users to use desktop or laptop computers which would allow for easier typing.

In addition to creating a Web application that would be an extension of our application. Some ideas include GPS location tagging, an "audio bank," and more, which are described below:

- Implementing Global Positioning System (GPS) tagging for locations could be useful because it would allow a user's location to be immediately loaded into an entry or timeline. This would act like the default date for timelines and entries being set to the current date.
- The creation of an activity that lets a user record any sound clip and later attach it to an entry (an "audio bank," so to speak) would allow a user to record an audio clip without having already created an entry for that audio to go into. At the

moment, a user must have already created an entry before recording audio, and audio clips cannot be moved between entries.

- Adding the ability to take pictures or videos from within the application would prevent users from needing to open the camera application on their devices to take a picture or record a video.
- Allowing for the insertion of pictures, videos, or audio from various locations such as Dropbox, Facebook, and Google Drive would give users more flexibility with our application. Rather than being limited to using images saved on one's device, this would let a user use images hosted on a website.
- For ease of reading, we feel that "next entry" and "previous entry" buttons located within each entry would be valuable to the user. This is a small addition that could improve the project.
- An improvement that could be made to backing up the application's data would be inserting the actual images into the database. Our project's database only stores Uniform Resource Identifiers (URIs) rather than the actual images themselves, meaning that images cannot be shared between two devices at the moment.
- To provide a more responsive user experience, bitmap caching should be implemented in the app. By caching picture date, the images would load faster when they must be reloaded after being scrolled back on screen because the images could be read from the app's ram rather than from disk.
- Another improvement that could be made to the application is adding full screen rotation support. As of now, the application does not crash during screen rotation,

but it will not preserve state from one screen orientation to another. If the user is typing some text and rotates the screen, that text will be lost.

- In addition to improving or adding features, some more essential fixes that should also be made to the application could make it run more smoothly. The first of these fixes is loading an image from disk. The current temporary image is not sized correctly, so when the picture is loaded, the image either jumps up or down in size suddenly which can disorient a user. To prevent this, the temporary image should be sized to the same dimensions as the picture that will replace it.
- The second fix major fix is merging edit mode into view mode. When a user wants to edit something, he or she will want to do it right away, not have to switch into edit mode and find where they wanted to edit all over again. Editing should be done by tapping on the item in the list to bring up editing options like changing or deleting content. Then, reordering could be done with drag handles in the list.
- In conclusion, our project team successfully created an application that allows users to document their experiences in a journaling/scrapbooking form. Our team surpassed our base goals of adding picture, text, and video to entries, and we dove into our stretch goals of adding audio recordings and syncing with Google Drive.

References

“Accessing Google APIs.” *Android™ Developers Guide.* N.p., n.d. Web. 14 Oct. 2014. <https://developer.android.com/google/auth/api-client.html>

“Android™ Audio Capture.” *Android™ Developers Guide.* N.p., n.d. Web. 29 Dec. 2014. <http://developer.android.com/guide/topics/media/audio-capture.html>

“Android™ Processing Bitmaps Off the UI Thread.” *Android™ Developers Guide.* N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/training/displaying-bitmaps/process-bitmap.html>

"Apple Developer." *iOS™ Developer Program*. 2015. Web. 14 Oct. 2014. <https://developer.apple.com/programs/ios/>.

Bauer, Carl. “Drag-Sort-ListView.” N.p. 2 April 2013. https://github.com/bauerca/drag-sort-listview

Bottomley, P. A., and J. R. Doyle. "The Interactive Effects of Colors and Products on Perceptions of Brand Logo Appropriateness." Marketing Theory 6.1 (2006): 63-83. Web.

"Building Applications with Multimedia." *Android™ Developers Guide*. N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/training/building-multimedia.html>

“DatePickerFragment.” *Android™ Developers Guide.* N.p., n.d. Web. 14 Oct. 2014. http://developer.android.com/guide/topics/ui/controls/pickers.html#DatePicker

"Design | Android™ Developers." Design | Android™ Developers. Google, n.d. Web. 7 Dec. 2014.

Dextor. “Get filename and path from URI from mediastore.” N.p., n.d. <http://stackoverflow.com/questions/3401579/get-filename-and-path-from-uri-from-mediastore>

"Get Started with Publishing." *Android™ Developers Guide*. N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/distribute/googleplay/start.html>.

"Google Design Guidelines." Google Design Guidelines. Google, n.d. Web. 7 Dec. 2014.

“Grid View.” *Android™ Developers Guide.* N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/guide/topics/ui/layout/gridview.html>

“JUnit Wiki.” *JUnit*, Github., n.d. Web. 12 Dec. 2014. <https://github.com/junit-team/junit/wiki>.

“Loading Large Bitmaps Efficiently.” *Android™ Developers Guide.* N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/training/displaying-bitmaps/load-bitmap.html>

Mawston, Neil. "Wireless Smartphone Strategies." *Android™ Captured Record 85 Percent Share of Global Smartphone Shipments in Q2 2014*. Web. 14 Oct. 2014. <http://blogs.strategyanalytics.com/WSS/post/2014/07/30/Android™-Captured-Record-85-Percent-Share-of-Global-Smartphone-Shipments-in-Q2-2014.aspx>.

“Navigation Drawer Fragment.” *Android™ Developers Guide.* N.p., n.d. Web. 16 Oct. 2014. <https://developer.android.com/design/patterns/navigation-drawer.html#Interaction>

"Reviews." *Evernote*. 3 March 2015. Np.p, n.d. Web. 14 Oct. 2014. <https://play.google.com/store/apps/details?id=com.evernote&hl=en>.

"ScrapPad - Scrapbook for IPad." Apple Application Store. N.p., n.d. Web. 13 Oct. 2014.<https://itunes.apple.com/us/app/scrappad-scrapbook-for-ipad/id353143273?mt=8>

Mayani, Paresh. “Select Multiple Photos from Gallery.” N.p., 18 Oct. 2012. <http://www.technotalkative.com/android-select-multiple-photos-from-gallery/>

Smith, Aaron. "Smartphone Ownership 2013." Pew Research Centers Internet American Life Project RSS. Pew Research Center, 5 June 2013. Web. 15 Oct. 2014. <http://www.pewinternet.org/2013/06/05/smartphone-ownership-2013/>

"Starting an Activity." *Android™ Developers*. N.p., n.d. Web. 14 Oct. 2014. <http://developer.android.com/training/basics/activity-lifecycle/starting.html>.

“Testing in the Software Lifecycle.” Microsoft Developer Network. Microsoft, n.d. Web. 12 Dec. 2014. <http://msdn.microsoft.com/en-us/library/jj159342.aspx>.

Thompson, Arthur. "Copying database to Google Drive.” N.p., 8 Dec 2014. <https://github.com/googledrive/android-demos/blob/master/src/com/google/android/gms/drive/sample/demo/CreateFileActivity.java>

Reda, Renas. "User Scenario Testing for Android™." Robotium - The World's Leading Android™ Test Automation Framework. Google, n.d. Web. 13 Oct. 2014. https://code.google.com/p/robotium/

Wells, Don. "A Gentle Introduction." Extreme Programming. N.p., 8 Oct. 2013. Web. 13 Oct. 2014.<http://www.extremeprogramming.org/>

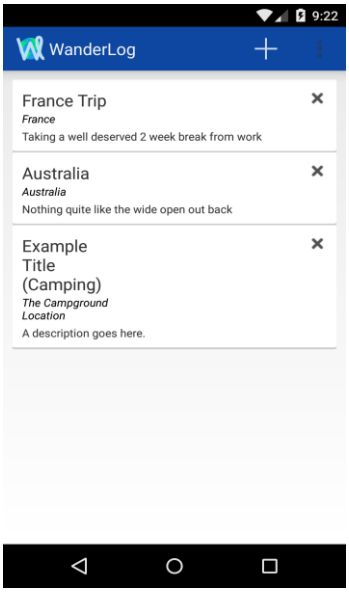
Appendix A:

WanderLog: What It Does And How To Use It

The original purpose for creating WanderLog is to make an application for travelers. This application allows a travel to chronicle events as they happen and maintain an electronic journal of their travels. WanderLog can support multiple different trips in what are called Timelines. Each timeline describes a single trip taken by the traveler and can have several events that take place within it. While traveling, a person might go to a restaurant, meet some local residents, go to public events, and many more possibilities. All these possible events are stored in what is called an Entry within a Timeline. Within these Entries, the traveler can add text, pictures, videos, and audio to the Entry. By combining all of these pieces of media, a blog style page can be created to recount what happened at that event.

Timelines

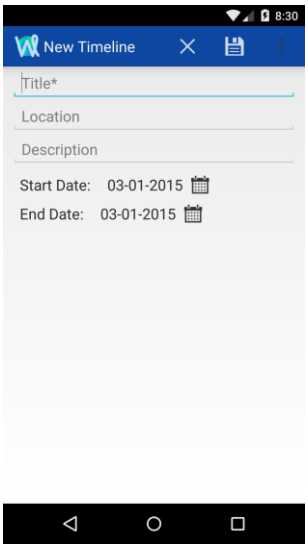
Upon launching the application for the first time, the user will view a screen that will display all the timelines stored in the application in a list.



This screen will look empty at first, but will fill up as more timelines are added to the app.

Creating a Timeline

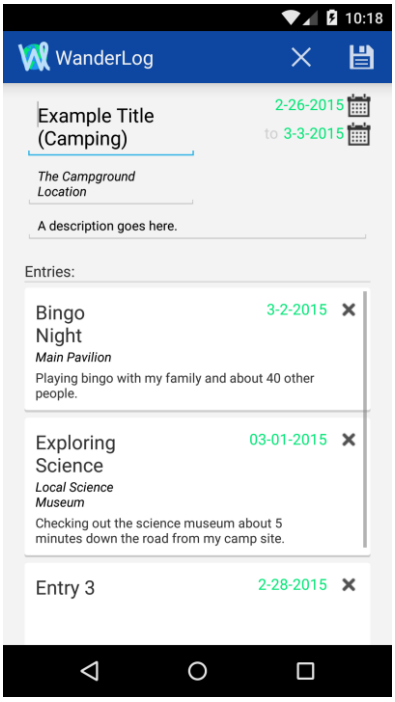
At the top of the main screen, there will be a button with a plus symbol. This symbol stands adding a timeline. If the user is unsure what the symbol means, he or she can hold down on the button, and a pop up message will describe the functionality of the button. Upon tapping on the new timeline button, the user will then be presented with a screen where a new timeline can be created.



The add timeline screen will have text areas where the user can tap on to bring up a keyboard and enter information like a timeline's title, location, and description. After adding all this information, the user can then tap on the start and end date text to bring up a date picker for selecting the date. After all the information has been added, the user can select the button with a check on top to save the new timeline, or, at any time, the user can tap the "x" button on top to cancel creating a timeline. After a new timeline has been created, the main screen will be shown again with the new timeline added.

Editing a Timeline

From the main timeline viewing screen, a user can tap on any of the timelines listed to display the contents of that timeline. The new screen that will appear contains all the information inserted into the timeline upon creation as well as a list of entries that are a part of that timeline.

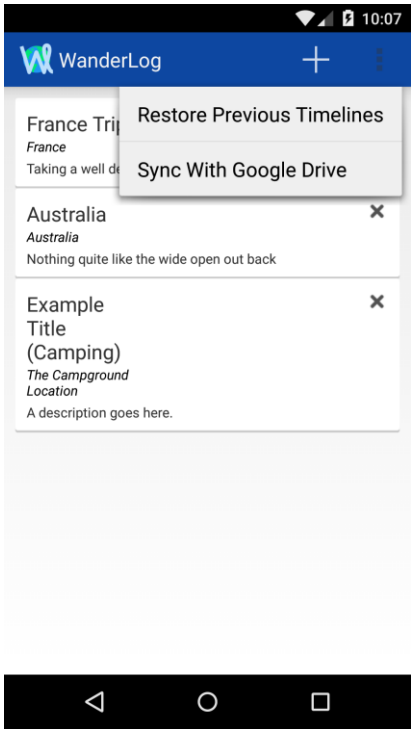


All the entries in this timeline will be sorted by the date given to them with the most recent ones being on the top of the list. If the user wishes to edit any of the timeline data, he or she must tap on the edit button at the top which looks like a pencil. This enables all the edit options for the timeline data being displayed. After editing, the user can then tap the “X” to cancel changes or the save button represented by a floppy disk. After saving or editing, the user will go back into view mode with all the editing options disabled.

Syncing with Google Drive:

Saving to and Downloading from Google Drive

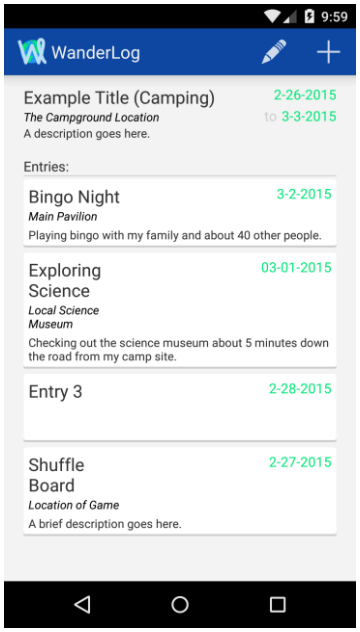
To sync with Google Drive, the “Sync with Google Drive” option in the overflow menu (of the main timeline screen) can be clicked.



An Internet connection (3G/4G or wifi) is required to sync with Google Drive. Once synced with Google Drive, the application allows a user to save all of his or her timelines from Google Drive or download a copy of timelines from Google Drive. To save to google drive, a user clicks the “Save Timelines to Google Drive” button. The file can be given any name (but must end in “.cblite”). To download timelines from Google Drive (which will set the downloaded file to the user’s database, overwriting any timelines on the device), the user can click “Download Timelines from Google Drive.” The user must choose a database file (ending in “.cblite”).

Creating an Entry

From inside of a timeline, the button with a plus sign on it at the top can be tapped on to add a new entry to the current timeline.



The next screen can then be used to enter information for a new entry like a title, description, more specific location, and date. After entering the information, the cancel and save buttons at the top of the screen can be used to go back to the viewing timeline screen and either discard or save the changes made.

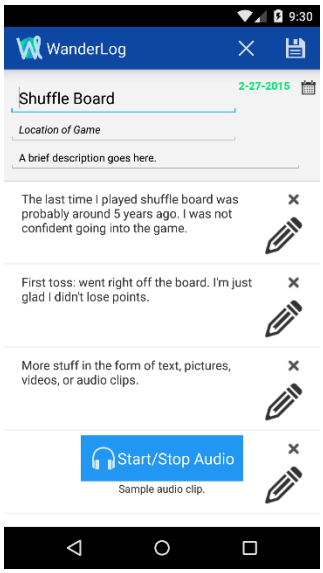
Deleting Entries

If ever a user wishes to delete an entry, this can be done in the edit window of the timeline the entry is within. From the timeline detail screen, the screen where the list of entries can be viewed, the edit button that looks like a pencil in the top of the screen can be used to enter edit mode. Once in edit mode, all of the entries in the list will now display a delete button in the upper right corner of the entry. Once done deleting, the cancel and save buttons at the top can be used to confirm or undo the deletions.

Editing an Entry

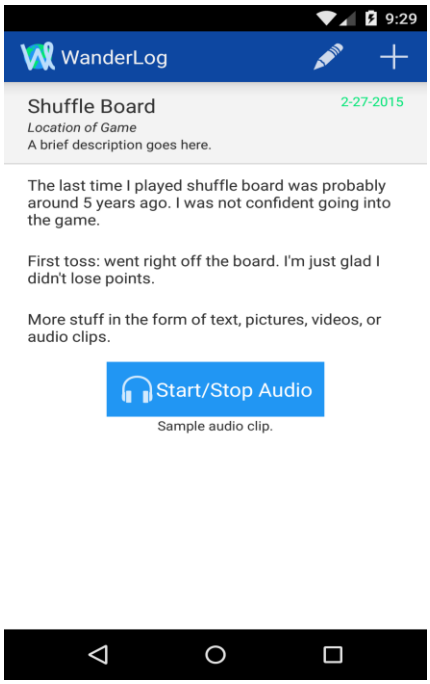
After an entry is created, the details of that entry can be displayed by tapping on the entry in the list of entries present in the timeline details page. The user will transition to

the entry details page where all of the entry information as well as any media within the entry will be displayed. In order to change any of the information about the entry that was inserted during creating, the user must first tap on the edit button, the one that

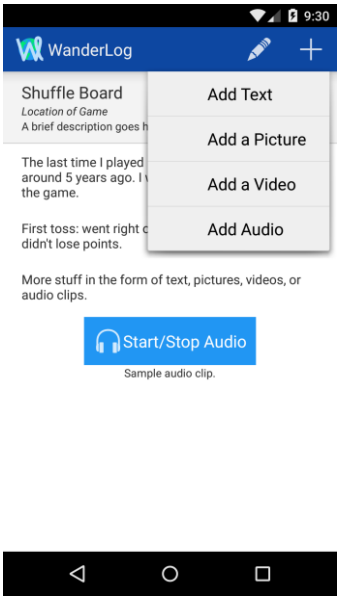


looks like a pencil, in the top of the entry details screen. Once in edit mode, tapping on the title, description, or location will bring up a keyboard to edit the information. The date can also be tapped on to bring the datepicker back up to select a new date. When edits are done, the cancel and save buttons at the top can be used to return to the entry detail screen and either discard or save the changes.

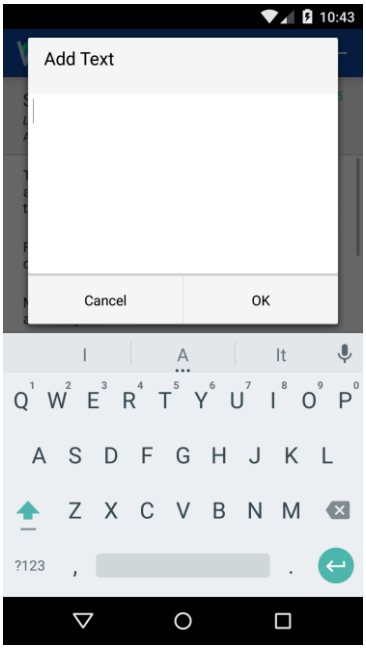
Creating Media



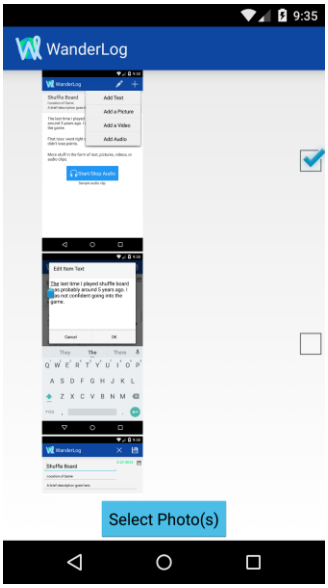
The essence of the WanderLog application is the ability to create media that recounts the events of an event. When at the entry detail screen, the add media button (plus icon) in the top can be used to add media. Upon tapping this button, a drop down menu will appear with options to add text, pictures, videos, or audio.



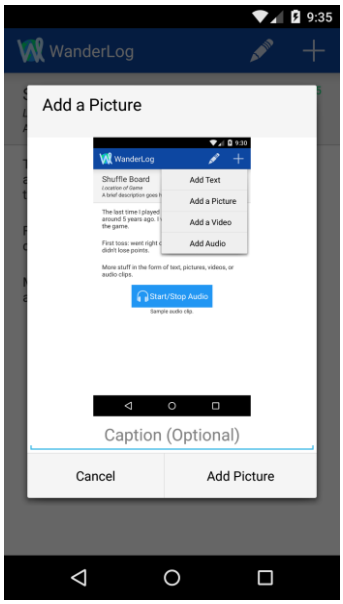
From the drop down menu, tapping “Add Text” from this menu will bring up an area to type out some text and buttons at the bottom to add the typed text or cancel adding the current text.



The “Add a Picture” button can be tapped from the add media drop down menu to bring up the photo selector.

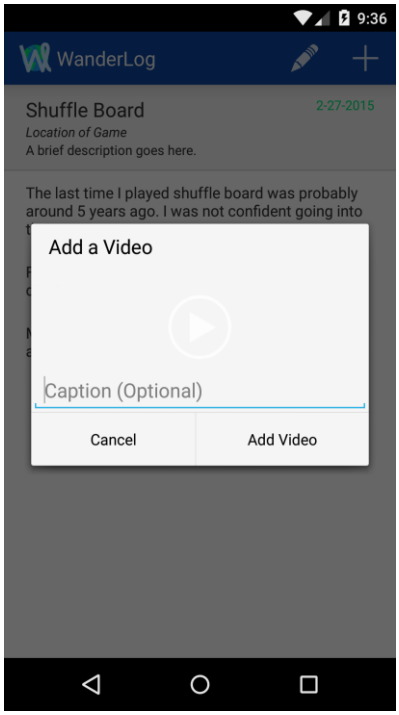


Multiple photos can be checked off before adding them. The back arrow can be used to cancel from the photo selector. If a single photo is selected to add, a confirmation screen will display the picture and give the option to add a caption before adding the picture.

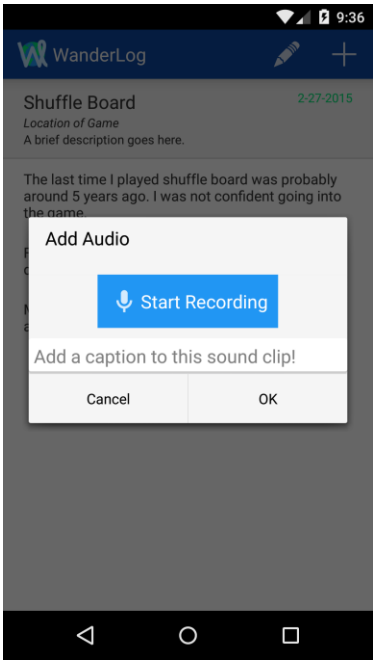


If multiple photos are selected, a confirmation screen will also display to confirm or cancel the addition of multiple photos. Captions can be added after importing multiple photos by editing the photo in edit mode.

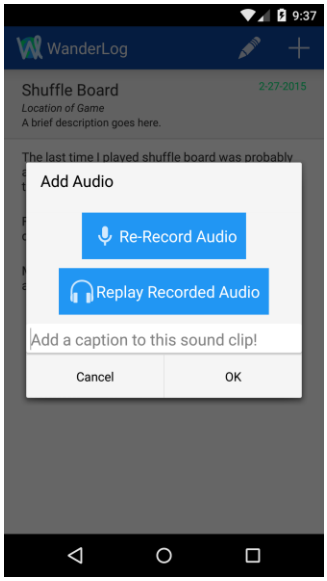
To add a video, the “Add a Video” button can be tapped in the drop down menu for adding media. This will bring up the video selector where a single video can be selected to add to that activity. Tapping a video will select it for adding to the entry while tapping the back button will cancel adding a video. If a video is selected, a confirmation screen will appear where the video will be previewed and a caption can be added before confirming or rejecting the addition of the video.



Lastly, audio can be added to the entry by tapping the “Add Audio” button in the add media drop down menu. This will bring up an area where audio can be recorded by tapping the “Start Recording” button.



After tapping the button, it will change to a button to stop recording audio. After tapping the button to stop audio recording, a new button will appear to replay the recorded audio.

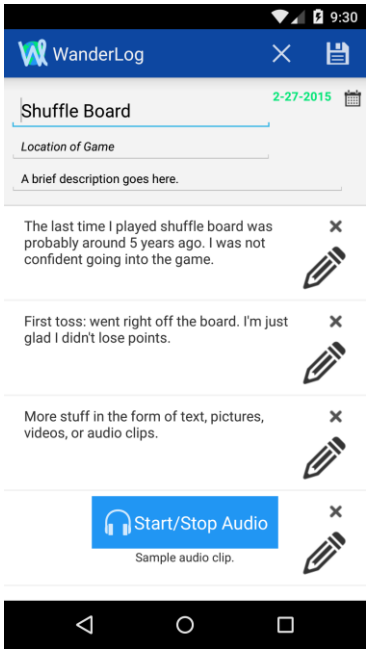


At this point, the audio can be re-recorded if it did not come out well the first time. Before saving the audio, a caption can be added to the audio to distinguish it from other

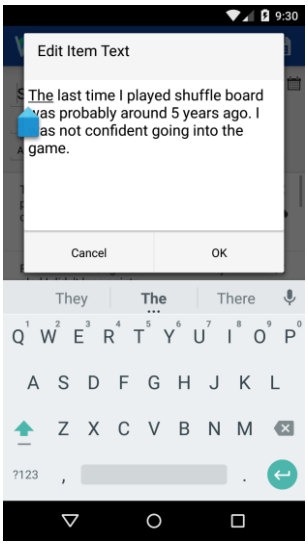
audio recordings. After everything is good to save, the buttons at the bottom of the audio recording area can be used to save or cancel adding the audio.

Editing, Reordering, and Deleting Media

After media has been added to an entry, it can be edited, reordered, or deleted all through the use of edit mode. Edit mode can be entered by tapping on the edit button (pencil icon) at the top of the entry detail screen. Once in edit mode, edit and delete buttons will appear to the right of the media.



The delete button is an “X” that will remove the corresponding piece of media from the entry. The edit button is a pencil icon to the right every piece of media that can be used to edit the text in that media, which means the entire content of text media and the captions to pictures, videos, or audio clips.



Media can also be reordered in edit mode by dragging media up and down. Once all the editing, reordering, and deleting has been completed, it can all be confirmed by tapping the check mark icon at the top, or it can all be undone by tapping the “X” button at the top. Tapping either of these two buttons will also exit edit mode and resume view mode.

How To Install Our Project

Download the .apk file to your Android device from *https://drive.google.com/file/d/0B-_RoHL8_wb5VUd1NUFQd3ppNGM/view?usp=sharing*. Once it has been downloaded, locate your downloads on your device and click the file. This will prompt you to install our application.

Appendix B

Usability tasks

Task 1	**Create a Timeline**
Task 2	**Edit Timeline Title**
Task 3	**Create Entry**
Task 4	**Add multiple pictures to an entry**
Task 5	**Caption a picture**
Task 6	**Edit the date of an entry**
Task 7	**Delete media from an entry**
Task 8	**Delete an entry**
Task 9	**Delete a timeline**

Usability Data

Numbers indicate time in seconds taken to complete task

User ID	Class year	Gender	Android™ user?	Task 1	Task 2	Task 3
1	2015	M	no	44.56	11	30.81
				Didn't take time to figure out what to do...time used was for typing	Checkmark more obvious than saving...maybe make them the same button	made sense...I hate typing on this tablet
2	2018	F	yes - for about a month	29.9	26.08	31.2
				straight forward	-	entry list heading looks like input for creating an entry
3	2015	F	no	37.71	13.33	33.68
				tapped start date and end date text first rather than icon	easy	put stuff in at the same time as creating an entry
4	2015	M	never owned a smartphone	30.83	25.8	35.33
				easy	flip icons around	add entry button was last button pressed - thought it made

						another timeline. Put add entry button on entries heading line
5	2015	M	no	45.88	10	20.55
				didn't know what button to press to finish	easy	why set location for timeline and entries..might want a timeline with new york and boston
6	2017	F	yes	40.43	12.11	47.4
				very easy to use	-	thought would be entering an entry in edit timeline screen
7	2017	F	no	31.42	15.12	34.75
				straight forward	-	tapped on entry heading first
8	2018	F	yes	35.53	12.53	31.84
				-	-	made sense
9	2016	F	yes	41.02	11.98	32.71
				easy	easy	adding in action bar made sense
10	2016	F	no	39.03	18.7	33.28
				-	-	-
11	2015	M	no	38.1	13.57	22.69
				easy	easy	easy
12	2015	M	yes	32.92	12.78	29.04
				-	-	wasn't confusing
13	2016	M	yes	28.13	13.98	24.01
				made sense	-	made sense to edit in action bar

User ID	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9
1	25.28	49.76	15.5	9	13.31	6.7
	make checkboxes to the left of the pictures they're with	first instinct was to press and hold on the picture to get a menu. First thing looked for was edit key on bottom of picture	maybe have tapping on date bring up date picker as well as icon	wasn't obvious that button was pressed	same as other things...made sense	there should be confirmation on that
2	18	19.36	14.4	3.55	19.55	10.23
	tried to hit the overflow button which is what some	click on picture first. edit button makes sense	-	-	usually swipe to go back	-

	android apps use					
3	31.31	27.88	6.73	9.88	6.28	3.9
	maybe clearer icon	click on picture from view and then edit picture	-	makes sense after editing	follows same format as rest of deleting	-
4	23.85	41.59	29.35	4.61	9.71	4.41
	trouble differentiating between plus buttons on different screens	tap on picture to add caption maybe	didn't state task well enough	didn't know if hitting x button	wasn't obvious to hit save	put in confirmation on that
5	13.71	25.36	5.63	5.1	5.65	7.46
	nothing confusing	tap on the picture...didn't realize to save	that was really easy		interactions are similar in different levels	-
6	48.9	16.37	6.33	5.83	7.13	4.11
	went into edit screen first	tapped on picture first	-	-	-	-
7	32.33	18.93	9.64	6.21	7.37	4.59
	-	tapped on picture first	-	-	easy to do given previous deleting	-
8	22.73	23.93	7.14	3.95	6.32	3.92
	also made sense	tapped on picture first	-	straightforward	knew what to do from previous task	simple
9	18.62	30.41	6.86	4.17	8.12	4.91
	didn't realize checkboxes immediately	tapped on picture first...figured out to use editmode	-	saw delete button when editing	-	-
10	24.95	26.29	10.83	5.92	10.39	5.19
	make checkboxes more clearly attached to pictures	tapped on picture first...expected to edit from there	-	-	-	-
11	20.77	29.94	8.98	6.12	8.24	4.98
	didn't think to use plus immediately but figured it out quickly	tapped on picture first	-	-	not clear it was necessary to hit save to actually delete	easy but might misclick and accidentally delete
12	20.12	25.81	5.41	6.08	7.16	4.11
	tapped on entry heading first	tapped on picture first...editmode	-	-	-	-

	but figured out to go to action bar quickly	made since once realized to click on pencil				
13	14.87	19.82	6.28	5.97	6.23	3.93
	-	tapped on picture first	knew to use edit button immediately	button lagged...wasn't sure if pressed at first	-	maybe confirm deleting timeline

Appendix C: Glossary

- **Timeline**: The object that contains a set of events for some trip (set of entries).
- **Entry**: An entry in the timeline. This will have a title and media (one or more media items) within it.
- **Abstract Media**: An abstract class that will contain shared methods between the types of media:
    - **Text**: Written type of media (could be written in an external note or within the application)
    - **Photo**: Visual media (taken either within application or externally)
    - **Video**: Visual media (taken either within application or externally)
    - **Audio**: Sound media (taken within application)
- **Create Entry View**: Screen that allows user to create an entry
- **Entry View**: Full screen view for an entry that displays all media in an entry
- **Edit Entry View**: Content view that shows what media is in an entry and allows user to edit entries details/what media in an entry
- **Create Timeline View**: Screen that allows user to create a new timeline
- **Edit Timeline View**: Content view that shows what entries are in a timeline and allows the use to edit timeline details/what entries are in a timeline
- **Media View:** View of a fragment containing a media object on top an Entry Detail View or a Present Entry View