2012

Simplifying the Campus Experience through Mobility

Corrine Erin Olson
The University of Montana

SIMPLIFYING THE CAMPUS EXPERIENCE THROUGH MOBILITY

By

CORRINE ERIN OLSON

Bachelor of Science in Computer Science, University of Montana, Missoula, MT, 2010

Professional Paper

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

May 2012

Approved by:

Sandy Ross, Associate Dean of The Graduate School
Graduate School

Doctor Min Chen, Chair
Computer Science

Professor Michael Cassens
Computer Science

Doctor Martin Fromm
Media Arts

Olson, Corrine, Master of Science, May 2012 Computer Science

Abstract

Chairperson: Dr Min Chen

Committee: Michael Cassens, Martin Fromm

The student experience at the University of Montana is rewarding, but uniquely challenging due to the size of the campus and the nature of many of the buildings. My goal is to design a project to simplify the life of the average student by providing a mobile application built for the Android platform. The project will provide students with tools to facilitate their educational experience and life on campus.

This application will display the campus map and the user's location based on the phone's GPS or wireless controller using the Google Maps API. The map will allow users to pull different points of interest from web services that already exist and display them on the mobile device. The campus news feed will be consumed by the application and displayed in a mobile-friendly way for users to peruse as well. Students will have access to different modes of transportation to and from the University, such as bus routes, bike/walk trails, and parking for personal vehicles. The final product will be an application supporting the daily navigation and transportation needs of the University of Montana student, allowing them to find campus information at the touch of a finger. This project is a progressive step towards increasing the mobile nature of our society, and bringing education into the mobile revolution through the use of open source tools and frameworks.

Contents

Figures

Tables

Introduction

Cell phone use is on the rise, with more than 83 percent of American adults owning some kind of cell phone, and 35 percent owning smartphones, according to a recent survey performed by the Pew Research Center. Smartphones have become increasingly popular worldwide, with the number of high-tech gadgets available increasing with every passing year. Between June 2010 to June 2012, the smartphone audience increased by more than 53 percent according to comScore (Figure 1). With this in mind, I wanted to build something that the students, faculty, and visitors to the University of Montana could use whenever they felt the need.

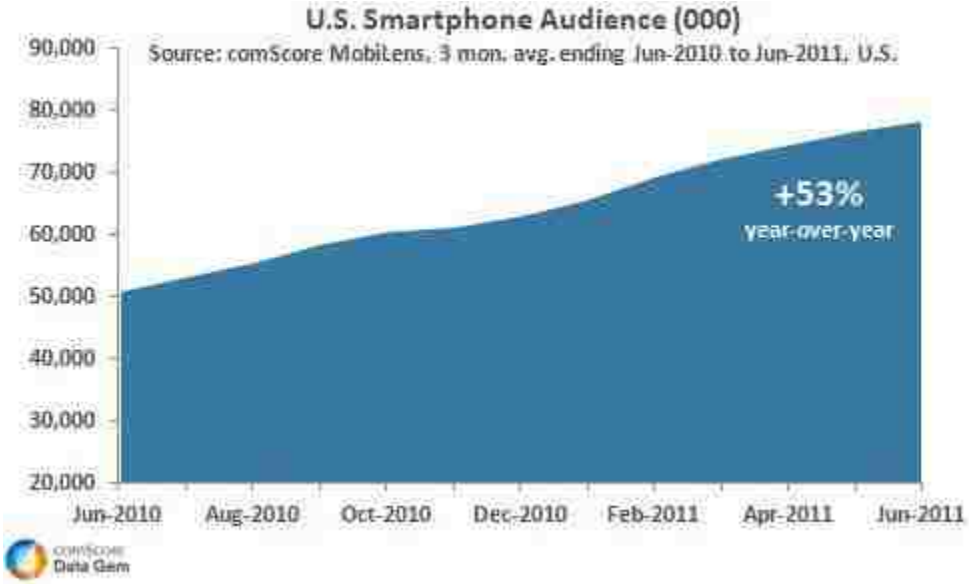


Figure 1: Graph showing growth of smartphone audience.

The University of Montana (UM) in Missoula has a very large and moderately complex campus ecosystem. Our 220-acre campus serves more than 15,000 students, both undergraduate and graduate alike, not to mention the many employees, faculty, and visitors that spend time here. With over 60 buildings on campus, finding needed information if unfamiliar with UM can be a daunting task. I distinctly remember wandering around looking for my next class as a freshman, with a paper map clutched in my hands my only guide. If an application incorporating a campus

map, points of interest, and a building search had existed, it would have greatly simplified the initial campus experience.

The need for an application to address the challenges of campus evolved into a plan of implementation for a mobile application, with the ultimate goal of creating a product ready for public release after approval. To this end, the Information Technology department decided to sponsor the project, while offering the full support of their very intelligent and experienced team of developers.

The Problem

A significant problem for people who spend time at the University of Montana is the abundance of information, juxtaposed with the relative lack of easy access to most of it. Finding needed information on the mobile web is daunting, especially on the University of Montana website, which is not optimized for mobile viewing. This information is gleaned not only from personal experience, but from conversations with other students. This presents a challenge – which information is most useful and frequently used, and how can it be presented in a mobile format which is both easy to use and easy to understand?

Determining the most useful information to include in the application required more than a bit of thought, as well as a good deal of discussion with the IT department. Some of the functionality originally envisioned, such as implementing a mobile version of Academic Planner, was deemed outside the scope of the project's current iteration due to the security considerations involved with the NetID system (although the planner system is something they wish to support in the future). After a great deal of discussion and reviewing feasibility of implementation on a number of features, and going through cycles of review, the main components of the mobile application were solidified:

- Academic Planner
- Campus map
- Transportation
- News
- Search

Once these components were identified, the next goal became paring down which information to display for these topics, and how much of the data was available in a mobile friendly form (which will be discussed in the Web Services section).

Now that the problem was defined, the issue of which mobile operating system (OS) to develop became tantamount. Unlike some of the more challenging questions in the design process, this one did not require an undue amount of time to answer. The Android platform, an open source operating system designed by the search giant Google, became the target.

Why Android?

Part of the reason Android was chosen is its developer-friendly, open-source nature. It costs nothing to set up a development environment, and that includes free emulator capabilities, and the capacity to launch development applications to actual Android devices. No upfront costs meant less time waiting for funding, and more time becoming familiar with the SDK and the IDE the default IDE for Android is Eclipse, the very popular and open source IDE, with around 250,000 users currently according the Eclipse usage statistics.

The other reason Android was chosen is market presence. Very recently, Android took over 50 percent of the smartphone market in February of 2012 (Wasserman). More than 200 million activated devices are running Android, with more than 550,000 new devices coming online every day. With this kind of market saturation, and with the only true competitor, Apple, having

upfront costs associated with development, Android became the obvious choice. Taking a closer look at the inner workings of the OS gives a feel for its structure and the services available for developers.

The Android OS is built on a Linux kernel, and includes not only the operating system, but also other key applications and middleware. The features, as detailed in the Android Developer Guide, are as follows:

- **Application framework** enabling reuse and replacement of components, which is the core of all Android applications
- **Dalvik virtual machine** optimized for mobile devices, the virtual machine is what actually runs applications on Android device
- **Integrated browser** based on the open source WebKit engine, which is used to view information not consumable in any other form
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification, allowing applications to display user interfaces
- **SQLite** for structured data storage, which will be needed for the next iteration of this project
- **Media support** for common audio, video, and still image formats, which the application will take advantage of for its GUI
- **GSM Telephony** which has no effect on this application, as the development is taking place on a Verizon phone (CDMA, not GSM)
- **Bluetooth, EDGE, 3G, and Wi-Fi** without which the web services the application relies upon would not function

- **Camera, GPS, compass, and accelerometer**, the application uses the GPS to show the user's current position on campus
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE, all of which were integral in the development of the application

The overall system architecture (Figure 2) of the Android system makes development an involved, but enjoyable process. The available libraries and key applications (such as the phone or browser) make incorporating functionality into implementation very straightforward.

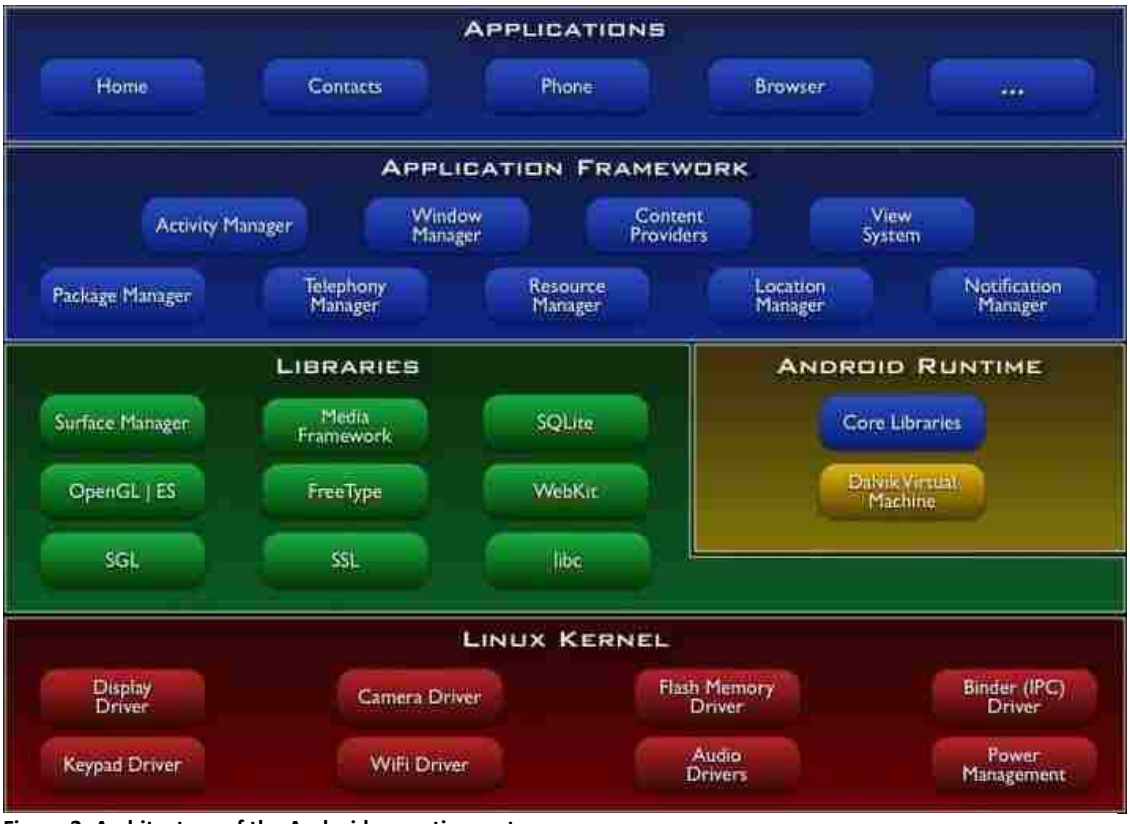


Figure 2: Architecture of the Android operating system.

Requirements

This section defines the requirements for the system, including the functional, data, usability (non-functional), and hardware requirements. All of these requirements are integral to the design of the system, and must be formed before implementation.

Functional Requirements

Functional requirements are tasks the application should allow the user to perform. The users' should be able to perform the following actions:

- Launch application
- View recent headlines
- Access Academic Planner
- View campus map
    - See current location on map
    - View points of interest
        - Art
        - Bus stops
        - Food
        - Printing
        - Wi-Fi
- View transportation resources
    - View ASUM bus locations
    - View Mountain Line bus information
    - View bike information
    - View parking information

- View news
    - Change news subscriptions
    - Refresh news
    - View original news story
- View information about application
- Search
    - People
    - Places
        - View place on map
    - Acronyms
    - Featured Links

Data Requirements

By using content provided in the form of web services, which could be queried or pulled much like a database, the data requirements for the project were minimal. The use of web services – online content providers – at least in this stage of development, did not require a database, although there is certainly functionality that may benefit from database support in the future. The current design would not be difficult to modify in order to accommodate future database integration.

The only data consideration needed is the use of Shared Preferences within the Android framework. Shared Preferences can be used by any application on the Android phone and are stored on the phone's internal memory in key value pairs. The Shared Preferences should only be used to store a small amount of data to limit the footprint of the application on the user's device. The only data being stored using this method is the user's preference for news

subscriptions. Accessing this data is easy, as the Android SDK allows specification of preference pages in XML and handles all user input behind the scenes. Retrieving preference data is also simple by requesting a Shared Preferences object and specifying the key. Shared preference data is preserved between sessions and is only removed if the application is uninstalled.

Another detail worth mentioning is the nature of the application (web service based) requires an internet connection (3G or Wi-Fi) to use the features. While the application does pull most of its data from the internet, it also caches all but the search results locally, and only replaces them if the application is stopped or the user chooses to refresh them.

Usability Requirements

Usability requirements, or non-functional requirements, define how an application should behave. They also reflect how well the application allows users to complete tasks and how fully functional the system is. The application should:

- Allow first time users to identify key features
- Not overly drain the battery
- Respond in a timely manner to user interaction
- Limit errors the user can make
- Provide feedback when things are loading to prevent user confusion
- Allow users to quickly learn how to accomplish tasks
- Alleviate user stress
- Display informative message for user if the application fails to pull information
- Be robust

Hardware Requirements

The hardware must follow a number of minimum specifications in order to work with the implementation of the application:

- Running Android 2.2 (Froyo) or newer
- GPS capabilities
- Wi-Fi or 3G connectivity
- Valid email account setup
- Google Maps installed

Design

The design of the application is essentially what is known a *mashup*. A mashup is an application that combines multiple pre-existing services to create something new, and more useful. The fundamental design of the application included the implementation of multiple University web services and data adapted to the Android environment. With the cooperation of the IT department, the application design went very smoothly. The first issue was dividing the application into components or modules. In that way, individual parts of the application could be implemented on a piecewise basis.

Figure 3: Twitter Android application dashboard design.

Application Structure

When designing the format of the application, the evolving Android standards were taken into account. With the introduction of Android 4.0, also known as Ice Cream Sandwich, Google has released a set of standards to bring

some clarity to the Google Marketplace, including design patterns for popular user interfaces.

Android 4.0 includes several design patterns that became popular in older versions, making them much easier to implement. Among these design patterns is the dashboard design. The simple dashboard design is extremely versatile with a high level of usability, being employed by such apps as Twitter (Figure 3) and Facebook. The mobile application was designed to use the same pattern, as users would most likely already be familiar with it, and the design provides a clean interface.

The first goal of designing a dashboard is deciding the most important functions of the application. The most used functionality should be visible to the user right away on the home screen (dashboard) of the application. The most prominent features of the application are Academic Planner, Campus Map, Transportation, and News. Because there are only four main

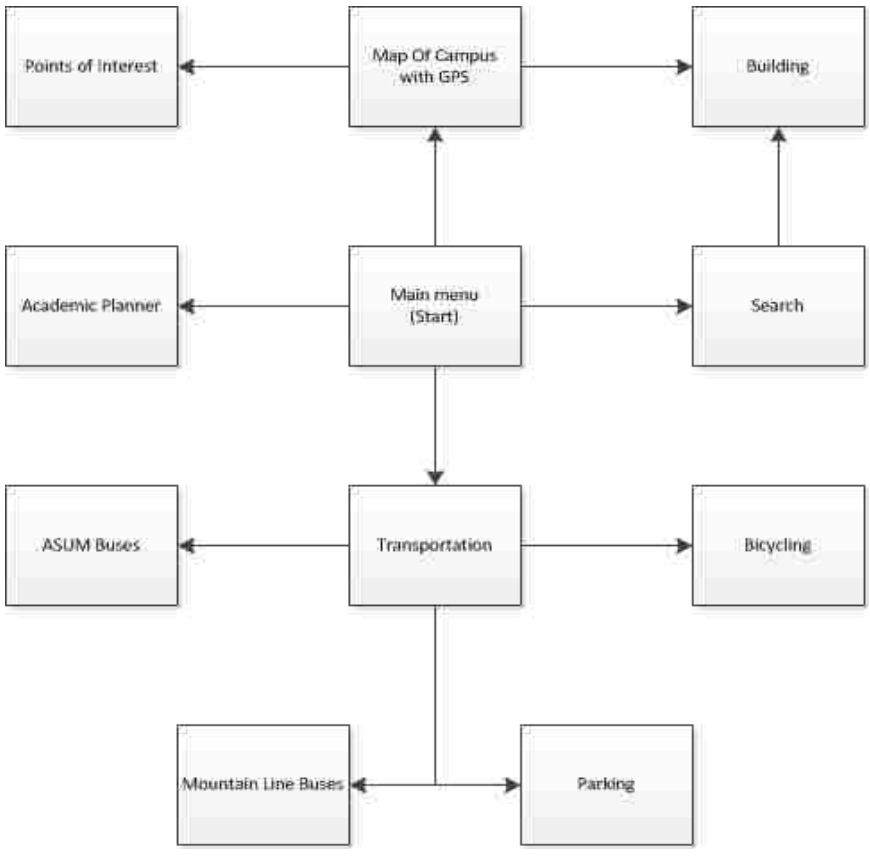


Figure 4: Flow of the user interface.

features, the dashboard design was amended to include a number of recent headlines from the news feeds as well.

Documentation

The first design deliverable created was a flow chart (Figure 4) showing the general steps in the interface, most of which are activities or intents (Android intents start activities or applications, activities are the basic screen class in Android). The flow chart is representative of the large picture as the user navigates away from the main dashboard screen.

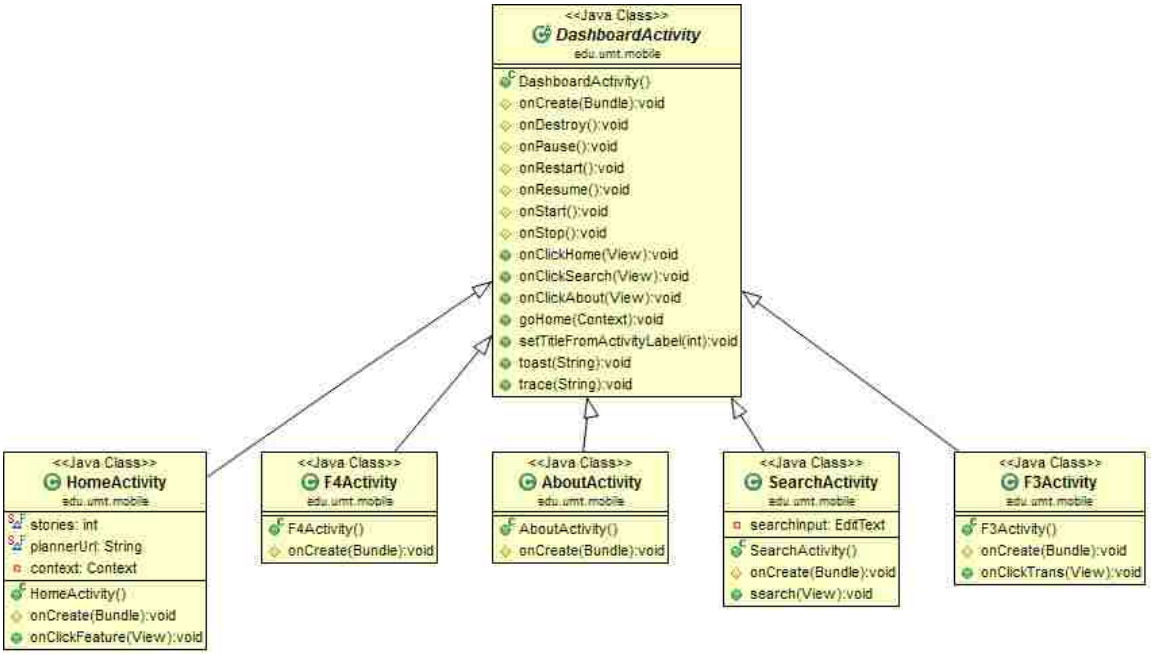


Figure 5: UML Diagram for DashboardActivity and the activities that inherit from it.

A very detailed complete UML diagram is included in the Appendices; however there are certain pieces of the design and diagram worth discussing. In particular, the structure of the Dashboard, and its respective screens are interesting. Another point of interest is dashboard activity is the main driver for the user interface, from which all the other screen activities (home, features, search, about) are derived.

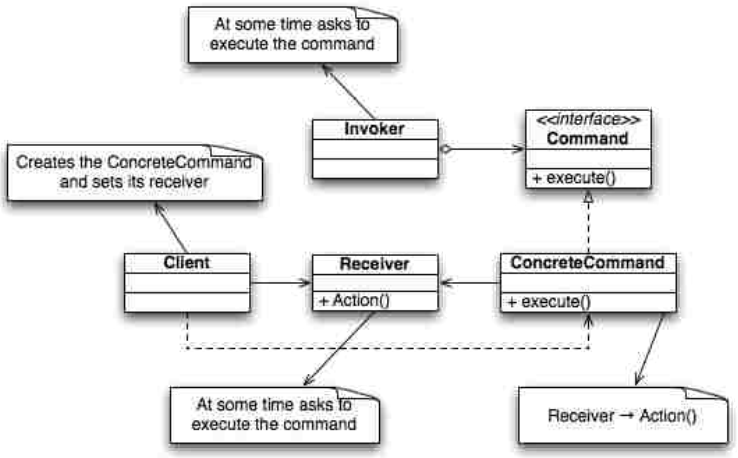


Figure 6: Command design pattern UML representation (Johns).

There is also a Custom Application replacing the default application as the basis of the entire system. Android allows writing a custom implementation in order to use the singleton design pattern while programming. Singleton patterns allow the application to maintain single copies of certain variables in order to share them between screens without having to pass the data back and forth.

Finally, another important design decision was choosing to implement the same event handling system the IT department employs in their own Java programming solutions: the command pattern (Figure 6). Using the command pattern allows the system to handle actions and responses using handlers through the command interface. While the pattern makes the system more complicated at first, the code maintainability and reliability is greatly enhanced.

Web Services

The use of web services required a few custom objects necessary to contain results from parsing some of the feeds, such as the News item, which contains multiple objects of the Item class. These objects are how the parsing responses of the RSS feed XML are stored for the application to display.

Another object of similar structure is the SearchResult object, which contains Places. The search web service is a brilliant piece of code where a query is simply made a part of the URL, along

with a preference for the return of a JSON or XML object. The simplicity of this design makes implementation much easier, as all it requires is a simple HTTP call to the server.

The final object class is the CampusOverlay, which extends the Android ItemizedOverlay object for the parsing of the points of interest KML files from the University Map Beta. The object was necessary due to the lack of Google Maps API being able to handle KML or KMZ files. Hopefully in the future, this functionality will be added, so custom parsing is not necessary on the part of the developer.

Risks

The main risk of the design comes from its dependence on web services. Without an internet connection, either Wi-Fi or 3G, the application will cease to function. Another possible issue is the web services' host server becoming unresponsive. If the application is unable to contact the server, then the application is useless. To ensure the user has feedback in this case, the application notifies the user if they lose network connection, informing them the application requires internet access.

While loss of service is likely to happen, as web servers notoriously crash, this loss is currently the only foreseeable risk with the chosen design. The risk is the same challenge that many web-driven applications face, and must address by notifying the user in the most informative way possible. For example, do not allow the application to crash and require a Force Close, as this makes the user think something is wrong with the application, not the services it depends upon to execute.

Another possible risk is any change to the way the data is released by the web service. For example, if the format of the XML in the RSS feeds is suddenly changed, this change could

possibly affect the way the news displays. Most of these feeds, however, are stable, so risk is minimal at this time.

Implementation

The first step in implementing the system was to choose a target Android API level. The API level indicates what framework the application can use to interact with the underlying Android system. As defined by the Google Android Developer's documentation, a framework API includes the following components:

- A core set of packages and classes
- A set of XML elements and attributes for declaring a manifest file
- A set of XML elements and attributes for declaring and accessing resources
- A set of Intents
- A set of permissions that applications can request, as well as permission enforcements included in the system

One consideration easing the process of selecting an API was Google's retention of backwards compatible components. Backwards compatibility allowed the selection of an older version of Android while retaining a significant portion of the market share. Table 1 contains all of the API levels and the Android versions they represent.

Before making the decision as to what version of Android to design for, the current market share of the different releases had to be taken into consideration. To this end, Google has a platform distribution page showing the number of active devices using different versions of Android.

Table 1: Google Android API levels along with corresponding versions of Android.

Platform Version	API Level	VERSION_CODE
Android 4.0.3	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4	10	GINGERBREAD_MR1
Android 2.3.3		
Android 2.3.2	9	GINGERBREAD
Android 2.3.1		
Android 2.3		
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

When the original research was done in August of 2011, the pie chart showed Froyo (Android 2.2) as the dominant platform, with more than two thirds of active devices using it. Currently, the share of Froyo is much lower, as Figure 7 shows, but is still a dominant platform.

Due to the backwards compatibility of the Android API's, the decision was made to implement the application using API level 8. To ensure the compatibility with the new platform leader, the project was also tested on devices running Android 2.3.3.

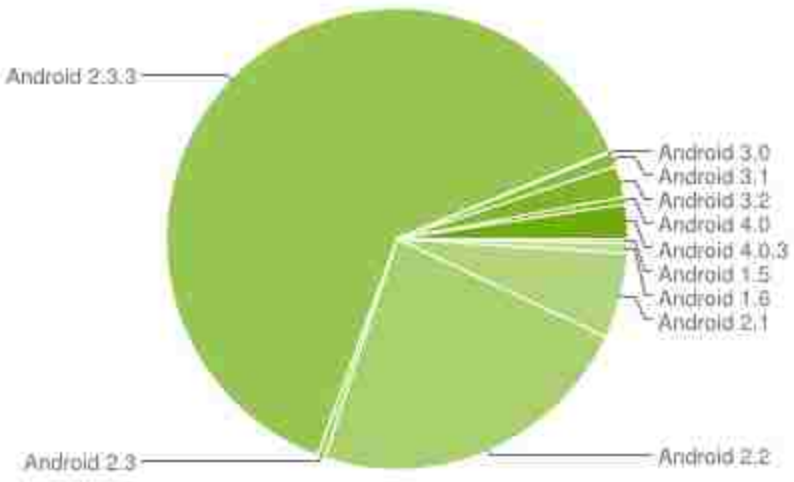


Figure 8: Pie chart showing platform share of active Android devices.

Development Environment

Once the API decision was made, the development environment had to be set up. Developing for Android can be done in a number of ways, but the option chosen for the application is considered the standard by most mobile developers. The first step is downloading the Android SDK, which allows the download of the frameworks to use, and the Google API's that correspond to them (such as Google Maps). The SDK also includes the device manager where emulators can be setup and run for testing of applications.

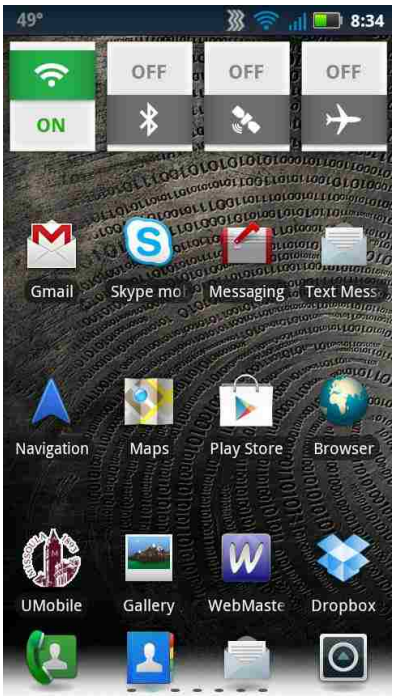


Figure 7: Shows a home screen on an Android device showing launcher icon for UMobile application.

Once the Android SDK is installed, a version of Eclipse, an open-source IDE available in a number of different versions and flavors, needs to be downloaded. For Android development either the Classic or Java version is recommended, and both have been tested with the application.

Once the steps have been followed, the final piece of the

Android development environment is the Android ADT plugin for Eclipse. The plugin is downloaded within eclipse, and allows the development, debugging, and execution of Android applications.

Figure 9: Close up of UMobile launcher icon.

The programming language for Android development is, by default, Java. Eclipse is a very useful IDE for Java development, and once the pieces are assembled, it also performs well as an Android development environment.

Results

The resulting application from the implementation is a useful, light-weight system providing support for students, employees, and visitors to the University of Montana, with a focus on students. The current section will walk through the application and talk about navigating through the application and will give an overview of the application's functionality.

A not about the application worth mentioning is the name UMobile will most likely not be the permanent name of the application. The question has been put to the University administration, but the response is still being awaited.

The first interaction with the program comes when the user wants to launch the application (Figure 8). The University of Montana clock tower logo was used in accordance with the publication standards.

Although care was take n to only use the correct coloring of the logo, its appearance on a dark background or light

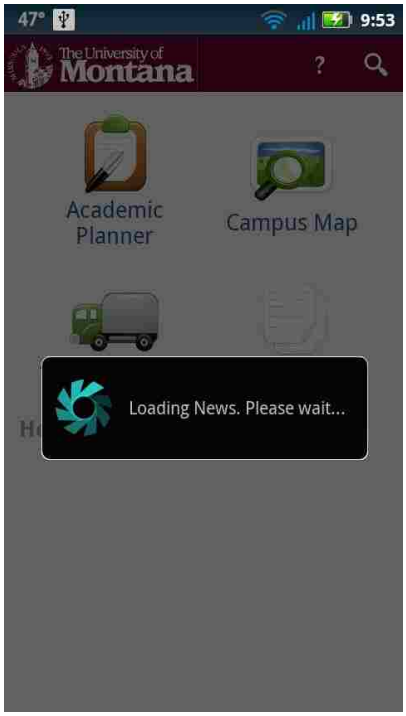


Figure 10: Home screen loading dialog for news headlines.

Figure 11: Home screen after headlines are loaded.

background had to be taken into account. By highlighting the logo in white, the launcher icon shows up well on both light and dark backgrounds, which, when colored plain maroon, was not the case (Figure 9).

Tapping the icon, if added to a home screen on the device or from the application menu, launches the main activity of the application. The main activity is the dashboard activity, which takes time to load the news items in order to display the top stories (Figure 10).

Whenever there is a loading dialog, a new thread has been spawned in order to handle the fetching of external data. If this is not done, Android runs everything on the UI thread and – if a process takes a long time – the UI hangs until completion, giving the user the feeling that the application is broken or has crashed. By separating the longer processes, such as grabbing and parsing the RSS feeds, the user gets some feedback as to why they cannot interact with UI right away.

The other choice for implementation of the task would have been to use the AsynchronousTask class and completely unblock the UI while it is pulling the data. However, using the asynchronous method caused other concerns, as the UI was dependent upon the information coming from the processes needing to be asynchronously run; thus blocking the UI makes more sense to reduce errors based on multiple threads trying to access the same information.

Once the headlines have loaded, the user is presented with the dashboard of the home screen (Figure 11). This interface is very clean, and all of the functionality is plain for the user to see. As

previously discussed in the design section, the home screen uses a classic dashboard design pattern, with an action bar at the top of the application perpetuated throughout the system.

Throughout this project, there are places where Android intents call outside Android core functionality, such as phone dialer, email services, and internet browser. These intents call outside activities the same way internal activities are called, allowing the user to go back to application from an external intent easily using the back button on the device.

An example of an *intent* is the browser, which is used throughout the application. It is executed when the Academic Planner icon is tapped, as there is no mobile support currently for the application. The browser intent is also executed when the user taps a headline on the home screen or the news screen (Figure 12).

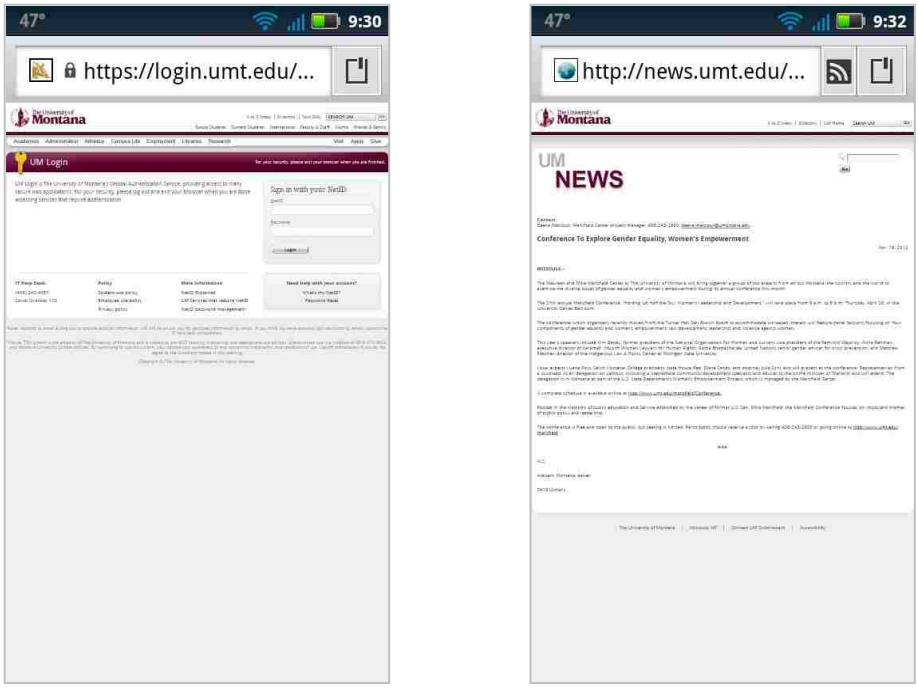


Figure 12: Examples of browser intents for Academic Planner and a News story example.

The campus map activity is one of the more complex screens in the application. This activity extends the Google Maps API MapActivity class, allowing the application to draw maps using Google's data.

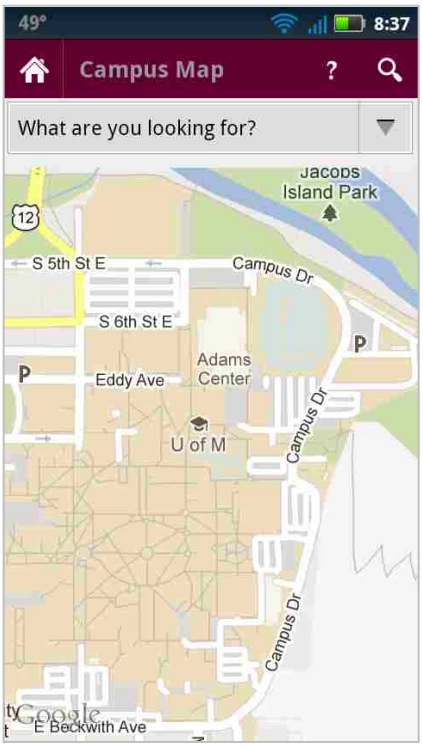


Figure 13: Campus activity screen when first opened.

The main object of the map screen (Figure 13) is to allow the user to see where they are on campus using the standard location symbol on mobile, a blue dot with a blue circle indicating the level of accuracy. Location is only displayed if the user has their GPS turned on.

The other goal the map screen was to provide the user a way to look at points of interest, as stated in the requirements section. All this data was available in KML format on UM's beta website on the beta campus map. The problem, however, arises from the mobile Google Maps API being unable to handle KML files like it's larger, internet-based relative. Because of this lack of functionality, the KML files had to be parsed much like the RSS feeds and fed into CampusOverlay object. When the user selects a category from the "What are you looking for?" dropdown menu, the overlay is displayed.

The application also employs the icons the beta campus map uses to represent the placemark categories (Figure 14). When a placemark is tapped, the information that the University has about the location is displayed in a window (Figure 14). Some placemarks, like the art category, also had images in their descriptions. Due to the limitations of the placemark windows, however, the images could not be displayed, and had to parsed out of the descriptions.

The campus map activity also has another function that will be discussed when the search activity is later discussed. The map defaults to fill the device screen with the campus map, whatever size that may be. It also defaults to fit all of the placemarks into the device screen, making sure the user can always see the points of interest regardless of device screen density or resolution.

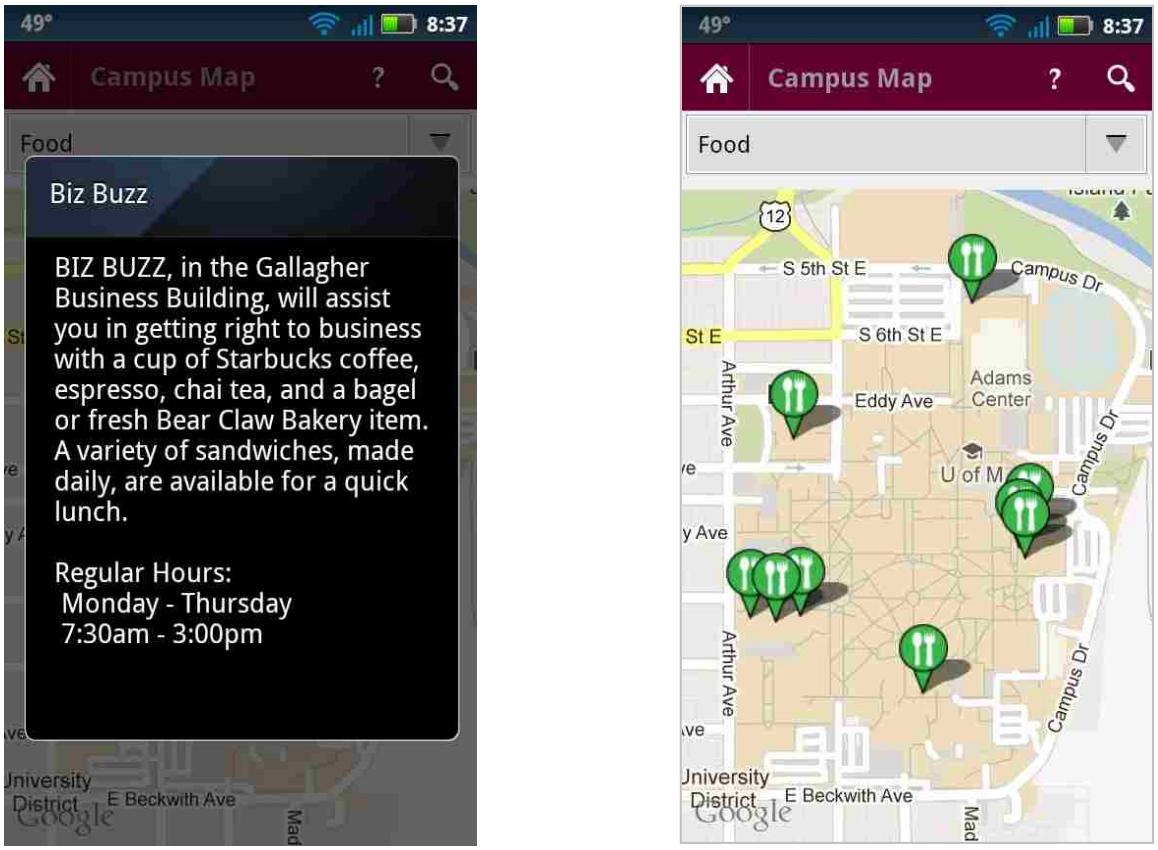


Figure 14: Example of KML overlay displaying food on campus and showing place mark information.

The next screen to discuss is the transportation screen (Figure 15). Getting to and from the Univeristy of Montana is a challenge that everyone who has spent time on campus has run into at one time or another. There are many ways to get to and from campus and all of them can be trouble for those unfamiliar with them. The application provides information for some of the more popular transportation methods to or from campus, including ASUM and Mountain Line bus information, parking information, and bicycle information.

Figure 15: Transportation screen displaying popular commuter options.

The ASUM us system is University run, and as such, there was already some support for commuters. The IT department had already designed a bus tracking system enabling users to see the route and location of buses, including the Park n' Ride and the U Dash. The public bus system in Missoula, the Mountain Line bus system, has also implemented a small mobile site for users to check on the status of buses, lookup routes, and other useful functionality.

Figure 16: News screen showing campus new stories.

Both the parking and bicycling information are located on the University of Montana website, as there are no mobile resources (yet) for the information. As such, all of the transportation screen icons launch the Android browser intent, sending the user to the respective resource and allowing easy return to the application by use of the back button.

The news screen (Figure 16) has a loading dialog just like the home screen for news downloading and parsing. Each story displays a date, a headline that links to the story's page, and gives a description of the story.

The news page also has a settings button and a refresh button in the action bar. The settings button takes the user to the shared preferences (Figure 17) that allow the user to choose which RSS feeds to subscribe to for news stories, while the refresh button allows the application to check for new news stories.

The preference page is generated from an XML shared preferences file. It automatically saves the user's preferences when the user checks or unchecks boxes.

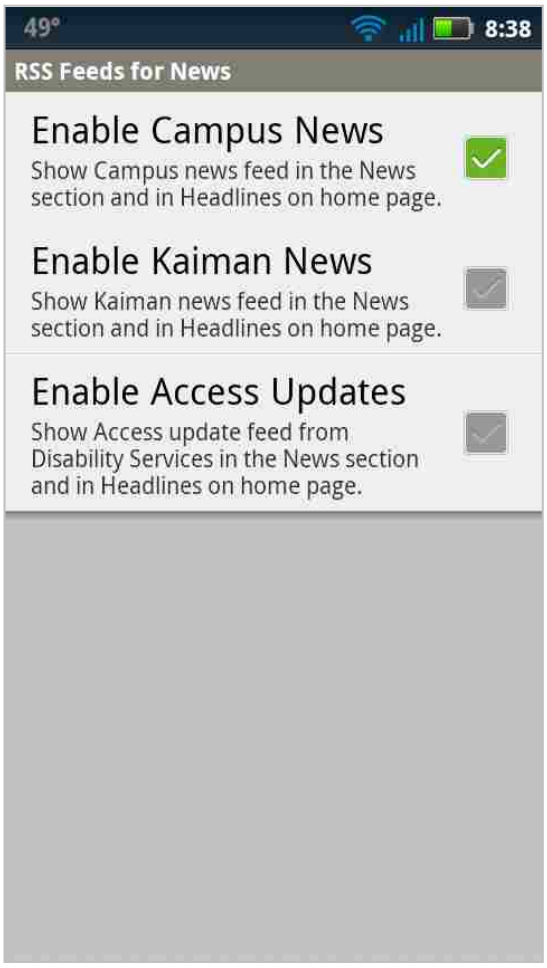


Figure 17: Shared Preferences for RSS Feed subscription choices.

One choice that should be emphasized is the subscription to Disability Services' (DSS) Access Updates feed, which gives updates on outages that effect campus accessibility, such as elevator outages.

Due to the multiple formats of date used between the different RSS feeds, custom date format parsing must be done for each type of news feed in order to be converted to the cohesive date format of choice. The news screen is the most data intensive of all of the activities in the application, and as such can take longer to load than the others. For this reason, the news object is stored in the application so it does not need to be re-pulled and re-parsed everytime the user enters the home screen or news screen.

The final piece of functionality in the application is extremely useful in multiple situations. There is a search button in the action bar opening the search screen, where the user may search for anything on campus whether it is a person, place, or thing. This search is implemented using the campus web service, as previosuly discussed in the design section, and makes use of a JSON object which the search service returns.

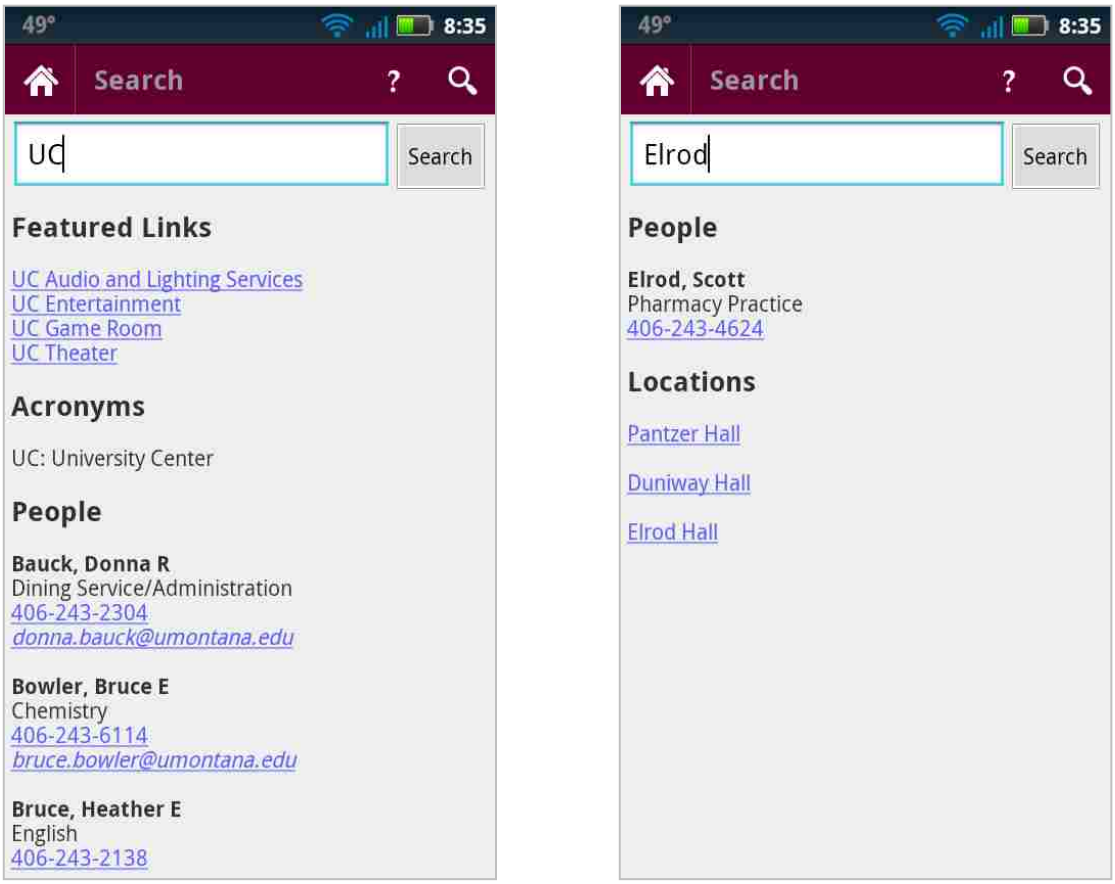


Figure 18: Search results demonstrating all available sections.

The service divides the results into sections: featured links, acronyms, people, and locations. With individual results such as people, email addresses are linked to messaging intents, while phone numbers are linked to the phone dialer (Figure 18). Locations, on the other hand, load the campus map with a maroon placemark on the location the user selected. The location intent allows the user to look for a particular location and see where they are in relation to it.

The remaining screen is the About screen, which explains the purpose of the application and gives credit to the people and organizations that made this project possible. The About screen is located on the action bar and represented by the ? icon.

This completes the discussion of the implementation. The remaining factor to discuss allowing users to test the implentation and see how successfully the application performs as a usable and useful tool for students of the University of Montana.

Testing

The following section explores the testing done on the UMobile application. In order to get optimal feedback on the system, the testing focused on the functionality and requirements from the design phase. In order to emphasize these aspects of testing, usability and black box testing were chosen in order to showcase the uses of the project.

Modular testing was also employed throughout the development process, with the code being tested for functionality, robustness, and maintainability. This allowed the application to be built in functioning blocks independent of one another. Black-box testing, on the other hand, allowed an overall functional look at the entirety of the application, including modular interaction, while usability testing put the application into the hands of actual users in order to determine its functionality.

Usability Testing

Usability testing uses testers to find issues with the functionality and ease of use. In this case, the testing was done with the help of five student volunteers. The goal of the testing was to determine whether the application was usable by students with no previous experience with the application, and variable experience with Android devices.

Testing Plan

The following questions identify the problems the testing hopes to find:

- Is the control scheme obvious to the user?

- Does the user understand how to complete simple tasks?
- Do the menus make sense to the user?
- Can the user find what they are looking for?

In order to perform the tests, the research had to be approved by the Institute Review Board (IRB) due to the testing involving human research subjects. The approval for the study and the informed consent form (ICF), and other testing materials provided to each subject can be found in the Appendices.

The tests were predominately performed in SS 402. The procedure for the testing is as follows:

1. Introduce the study, and allow the test subject to read over the ICF and sign.
2. Explain the test procedure and provide task sheet.
3. Fasten the Go Pro camera to the test subjects head and begin recording.
4. Allow user to complete tasks, talking out loud during the session.
5. Complete post-test interview.
6. Ask the user to complete a questionnaire.

After the session is complete and the tester has been thanked, the video is copied from the camera to make room for the next session, and the application is reset. The following sections explore the results of the usability testing.

Recording Transcriptions

Notes were taken during testing while observing the subject and while viewing the recordings further ensured no usability issues were overlooked by seeing exactly how the user interacted

with the application. Using the Go Pro camera for testing was an ingenious idea introduced was discussed in Yolanda Reimer's Human Computer Interaction course.

Table 2 shows each task and each participant's performance of the task including any issues or comments they had during use. The total time to complete tasks is also included for reference. For more details on the tasks performed, please reference Appendix A.

Table 2: Data capture from video transcription and testing session notes.

Task	Description	Participant Performance				
		1	2	3	4	5
1	Start App	Easy to recognize	Easy to recognize	Took a while to find icon	Easy to recognize	Easy to recognize
2	Explore News	No trouble	Found easily, back button to return made sense	Found easily, had to be prompted on use of back button	Found easily	Disliked refresh, counter-intuitive, could not select story
3	Search ISB	Looked for search on home screen, took a moment to find in action bar	Thought the search icon in action bar was obvious	Easily found	Went to campus map before search	Took a while to find search
4	Find SS	Recognized location link in search and found SS on map	Found building and recognized location link	Recognized location links	Used map to find	Recognized location links in search
5	Academic Planner	Easily found	Easily found	Easily found	Easily found	Easily found

6	Find Foley	Had trouble pressing the phone number instead of the email, links a little small, found bug in search (space causes error)	Found bug in search with spaces	Easily found	Easily found	Easily found
7	Find Broadway Park n' Ride	Easily found, wanted to zoom in but couldn't	Easily found	Tried searching before transportation	Went to transportation, didn't see Park n' Ride, did not complete task	Easily found
Time (min:sec)		4:48	5:58	6:20	5:20	4:23

The testing went extremely well, and a great deal of information was gathered from the recordings. The overall trends were encouraging: everyone had positive experiences with the application, even if they had some suggestions for improvement. There was also a wide variety of experience amongst the testers, which was a goal of the usability testing. Two had Android phones, one had a Windows 7 phone, one had an iPhone, and one had no mobile platform experience at all. All the testers were current students of the University of Montana, including both undergraduate and graduate level students. Despite this attempt at wide sampling, the testing showed a very small amount of variance in the time it took to complete the tasks, with a range of four minutes and 23 seconds to six minutes and 20 seconds. With a difference of just under two minutes for such a wide variety of mobile device experience, the application demonstrated itself to be very usable.

Also included in the recordings were the post-test interviews. While most of the feedback was very positive, there were some ideas for improvements worth mentioning:

- Add search to main page as a feature
- Improve the news loading
- Do not lock the user out of the UI on the main and news pages during loading
- Refreshing news page should be done automatically, especially if the user updates the subscription preferences (counter-intuitive)
- Hitting enter button on the search page should be an alternative to tapping the search button

The feedback ascertained from the users was very valuable, and pointed out some usability issues that would need to be addressed before public release of the application. The testers also found a minor bug that escaped the initial application testing: when a space is entered into the search box, the application force-closes (crashes) with no warning or feedback for the user. While the crash happened in the first tests, the task was changed in order to avoid spaces in the search box for future tests.

In summary, the testing was extremely useful in determining where the application still needed work. The test sessions revealed a small bug and some improvements that could be made, both of which will be discussed in the Conclusions section. After the interview was complete, the tester was allowed to remove the camera for the final part of the study, the questionnaire.

Questionnaire Results

After the recording session was complete, the testers were asked to complete a short survey about the application. Please refer to Appendix B for the questionnaire. Table 3 shows the answers to each question and the average score of each question. The questionnaire uses a scale from one to six, with one being the worst and six being the best. There was also a single

yes or no answer question asking whether the user felt they accomplished all of the tasks successfully. All of the testers answered the question in the affirmative.

Table 3: Tester responses to the post-test questionnaire.

Question	Participants					Averages
	1	2	3	4	5	
1	Yes	Yes	Yes	Yes	Yes	Yes
2	6	5	6	5	6	5.6
3	6	6	6	6	6	6.0
4	6	5	6	-	6	5.8
5	5	6	6	6	6	5.8
6	3	5	4	5	6	4.6
7	5	5	6	5	6	5.4
8	4	5	6	4	6	5.0
9	4	2	6	6	4	4.4
10	6	6	6	6	6	6.0
11	5	6	6	5	6	5.6

The questionnaire results were also exceedingly positive. The average score overall was 5.4 out of 6, which extremely encouraging in terms of the testers' enjoyment of the application. The lowest score on the questionnaire was number nine, which asked the user if all functions worked without error. Due to the discovery of the search bug during usability testing, the lower score is explainable, and thus not truly relevant. The highest scores were on questions three and ten – six and six respectively – both of which related to the ease of use and navigation around the application. This indicates that the choice of the dashboard design pattern was well conceived and resulted in a very user-friendly application.

The combination of usability testing and committee commentary led to a number of fixes and updates being implemented before black-box testing:

- Adding a notification for no network access on the home screen informing the user the application is web based and will not function without internet access

    - Required adding new permission to the Android manifest in order to access the network state
- Re-evaluating and changing the threading for the news page and the headlines on the main page to a safely interruptible and superior coding strategy
- News now refreshes if the user changes their preferences (able to implement due to new thread handling)
- Search bug fixed
    - The Apache Common Lang 3.1 library has a string utility allowing HTML encoding, although this did not include spaces (had to be handled separately)
- Search event now fires if the user clicks done on the keyboard, not just if the button is clicked

Black-box Testing

Black-box testing is integral to the software development process. This form of testing is also known as functional or behavioral testing, and focuses on the programs ability to accomplish the tasks it is required to do. While typically this testing is best done by someone with no knowledge of the code and knows nothing about the structure of the underlying architecture, as long as the programmer remains unbiased and does not just test for what they programmed the application to do, black-box testing is possible. The tester should only be concerned with the input and output of a given task, with the program itself being an impenetrable black-box.

The goal of this testing is to determine if the project meets the initial requirements laid out in the design phase of the project. To this end, tests must be designed, expected results identified, and actual results received through interaction with the application. The first step in black-box

testing is determining the test cases. These should be derived directly from the requirements.

Once this is done, a formatted table should be used to record the test cases and their results:

Table 4: Black-box test cases and results.

Test #	Description	Expected Results	Actual Results
1	Preconditions: Application is installed on Android device User taps on the application icon	Application launches, user sees the home screen (dashboard)	Pass
2	Preconditions: Test case 1 successful, user has network connection User can see recent headlines on home screen and clicks on one	News headlines and the selected story load successfully	Pass
3	Preconditions: User is on home screen, user has network connection User taps Academic Planner icon	Academic Planner loads	Pass
4	Preconditions: User is on home screen, user has network connection and GPS turned on User taps the Campus Map icon	Campus map loads and shows current location of user	Pass
5	Preconditions: Test case 4 successful User selects Art from points of interest dropdown	Campus map shows art place marks	Pass
6	Preconditions: Test case 4 successful User selects Bus stops from points of interest dropdown	Campus map shows bus stop place marks	Pass
7	Preconditions: Test case 4 successful User selects Food from points of interest dropdown	Campus map shows food place marks	Pass
8	Preconditions: Test case 4 successful User selects Printing from points of interest dropdown	Campus map shows printing place marks	Pass
9	Preconditions: Test case 4 successful User selects Wi-Fi from points of interest dropdown	Campus map shows Wi-Fi place marks	Pass
10	Preconditions: User is on home screen, user has network connection User taps Transportation icon	Transportation dashboard loads	Pass
11	Preconditions: Test case 10 successful User taps ASUM bus icon	ASUM bus tracker loads	Pass
12	Preconditions: Test case 10 successful User taps Mountain Line bus icon	Mountain Line mobile site loads	Pass
13	Preconditions: Test case 10 successful User taps Bike icon	UM bike info page loads	Pass

14	Preconditions: Test case 10 successful User taps Parking icon	UM parking info page loads	Pass
15	Preconditions: User is on home screen, user has network connection User taps News icon	News page loads	Pass
16	Preconditions: Test case 15 successful User taps refresh button	News page reloads	Pass
17	Preconditions: Test case 15 successful User taps settings button	News preference page opens	Pass
18	Preconditions: Test case 17 successful User changes preferences and returns to News page	News page reloads with news preferences	Pass
19	Preconditions: Test case 15 successful User taps on story link	Story loads	Pass
20	Preconditions: Test case 15 successful and there is more than one page of stories User swipes up or down on screen	News page should scroll	Pass
21	Preconditions: User is on a screen with the action bar User taps the question mark button	About page loads	Pass
22	Preconditions: User is on a screen with the action bar, user has network connection User taps the magnifying glass button	Search page loads	Pass
23	Preconditions: Test case 22 successful User taps the text box and types a search term then hits enter or the search button	Soft keyboard pops up allowing the user to enter their search, then when user hits done or the search button the keyboard disappears and the search results load	Pass
24	Preconditions: Test case 23 User taps on a person's email	Email application opens with message to person chosen	Pass
25	Preconditions: Test case 23 User taps on a person's phone number	Phone dialer opens with number in the dial field	Pass
26	Preconditions: Test case 23 User taps on a location link	Campus map loads, zooms in and place marks the building	Pass
27	Preconditions: User is on a screen with the action bar other than home User taps home button	Home screen loads	Pass
28	Preconditions: Test case 2 User taps back button (on device)	Returns to last screen	Pass
29	Preconditions: Test case 1 and no network connection	Notifies user the application will not function properly	Pass

The results of the testing were positive. While black-box testing is usually done after usability testing, this testing was done after in order to incorporate the user feedback before doing a final functionality testing. This means that the bugs found during usability testing were found and fixed before doing this iteration of testing.

Conclusions

Throughout the paper, the entire process of developing a mobile application has been discussed from defining a problem to be solved, to implementing and testing the resulting system. The discussion allows an understanding of the UMobile project from inception to completion. While this application has evolved in a workable solution to a very real problem, there are still issues to address.

Future of the Project

As revealed in testing, there are some changes that need to be made, and others that, upon consideration of the user feedback during usability testing *should* be made. The main recommendations for changes are as follows:

- Remove headlines from the home screen and add a search feature icon (keep search n the action bar as well)
- Add another feature to pair with the search feature on the dashboard
- Change the news loading to an asynchronous task that does not lock the user out the interface while updating
- Add user-defined news refresh interval to the shared preferences
- Add text size shared preference for search and news screens so users with bigger fingers can have more room to click on links.

While the original intent was to have the Academic Planner available in a more mobile friendly way, the Information Technology Department stated at the current stage, Academic Planner was far outside the scope of the project. There is almost nothing to make this easier (no available web services) and just getting the security approval to login with the NetID system employed by the University is extremely difficult. A requirement for future development would be the implementation of a system for mobile devices – such as a mobile web version – that can be accessed from multiple mobile platforms.

Another notable suggestion would be the employment of a campus-wide survey asking students what information they access most frequently, and which information they would like to have in a mobile application. Having the development of this application be driven by student needs would be the perfect realization of the project's goal.

Assessment of Solution

Overall, the project developed offers a functional proof-of-concept to the University of Montana, showing the administration that not only is a mobile application possible, the results were well received by a small, diverse group of student. The application displays all of the functionality defined in the requirements, and exemplifies user-centered design in its usability and practicality.

This application successfully provides answers to numerous challenges that confront students of the University of Montana. The IT department will be continuing work on this project, and will hopefully implement some the future recommendations mentioned in next section. The application allows users to find people, places, and things on campus, while providing a centralized location to access information most needed on a regular basis.

During development, many students expressed a need for just such an application not only on Android, but all mobile devices. This indicates the success of the endeavor undertaken by the implementing this project – to provide an application to simplify the campus experience through mobility.

References

"Android - Developers ." Android . Google, n.d. Web. 17 Apr. 2012. <http://www.android.com/developers>.

"Android Developers." Android Developers. Google, n.d. Web. 1 Sept. 2011. <http://developer.android.com/index.html>.

"comScore Reports February 2012 U.S. Mobile Subscriber Market Share - comScore, Inc." comScore, Inc. - Measuring the Digital World. N.p., 3 Apr. 2012. Web. 17 Apr. 2012. <http://www.comscore.com/Press_Events/Press_Releases/2012/4/comScore_Reports_February_2012_U.S._Mobile_Subscriber_Market_Share>.Felker, Donn, and Joshua Dobbs. Android application development for dummies. Hoboken, N.J.: Wiley, 2011. Print.

Johns, Trevor . "File:Command Design Pattern Class Diagram.png - Wikipedia, the free encyclopedia." Wikipedia, the free encyclopedia. N.p., 21 Mar. 2008. Web. 23 Apr. 2012. <http://en.wikipedia.org/wiki/File:Command_Design_Pattern_Class_Diagram.png>.

Komatineni, Satya, and Dave MacLean. Pro Android 3. New York: Apress :, 2011. Print.

Lidwell, William, Kritina Holden, and Jill Butler. Universal principles of design. Gloucester, Mass.: Rockport, 2003. Print.

Mednieks, Zigurd R., and Laird Dornin. Programming Android. Sebastopol, Calif.: O'Reilly, 2011. Print.

Meike, Blake. "Programming Android Examples." github. O'Reilly's, n.d. Web. 1 Sept. 2011. <https://github.com/bmeike/ProgrammingAndroidExamples>.

Reimer, Yolanda. "User-Centered Design." Human Computer Interaction. University of Montana. Social Sciences 362, Missoula, MT. 28 Feb. 2012. Class lecture.

Reimer, Yolanda. "User-Centered Design – Testing." Human Computer Interaction. University of Montana. Social Sciences 362, Missoula, MT. 10 Apr. 2012. Class lecture.

Smith, Aaron. "Americans and Their Cell Phones | Pew Research Center's Internet & American Life Project." Pew Research Center's Internet & American Life Project. N.p., 15 Aug. 2011. Web. 9 May 2012. <http://pewinternet.org/Reports/2011/Cell-Phones.aspx>.

"U.S. Smartphone Audience Growth ." The comScore Data Mine | Colorful, bite-sized graphical representations of the best discoveries we unearth from our data.. N.p., 8 Aug. 2012. Web. 25 Feb. 2012. <http://www.comscoredatamine.com/2011/08/u-s-smartphone-audience-growth/>.

"Usage Data Collector Usage Report." Eclipse - The Eclipse Foundation open source community website.. N.p., 21 Feb. 2011. Web. 9 May 2012. <http://www.eclipse.org/org/usagedata/usage.php>

Wasserman, Todd. "Android Tops 50% Market Share in the U.S. [STUDY] ." Social Media News and Web Tips – Mashable – The Social Media Guide. Mashable Tech, 4 Apr. 2012. Web. 1 Mar. 2012. <http://mashable.com/2012/04/04/android-breaks-50-market-share/>.

Williams, Laurie. "Testing Overview and Black-box Testing." A (Partial) Introduction to Software Engineering Practices and Methods. 2008-2009 (Fifth) Edition ed. N/A: NCSU CSC326 Course Pack, 2006. 33-59. Print.

Appendix A: Testing Task List

Task 1

For the first task, please open the application with the launcher icon.

Task 2

Please explore the news section. Try selecting a story for more information.

Task 3

You need to know what the acronym ISB stands for. Try searching for it to find out.

Task 4

Cyberbear tells you your class takes place in SS. Find out what this building is and where it is located.

Task 5

Please try entering academic planner and logging in. When done, log out and return to the application.

Task 6

You need to find the phone number for someone named Foley. Please try this now and select it as if you were making a call. When done, hit back to return to the application.

Task 7

You need to find out where the Broadway Park n' Ride is currently. Please try this and return to the application when done.

Complete!

Please feel free to play around with application before filling out the questionnaire. If you have comments, please write them on the form. If you find any bugs, please write those down in the comments section as well, along with any information on the nature of the bug you are willing to provide. Thank you for your time!

Appendix B: Questionnaire

*Before leaving today, **please fill out this questionnaire**. If you don't have time to stay and complete it, please take it with you, complete it, and return it to Corrine Olson's box in the CS Office (SS 401).*

*I would like to **thank you** for generously volunteering your time to participate in this usability testing. Your input will be **invaluable** in the development of the mobile UM application. I hope that you found it to be an interesting and enjoyable experience!*

Question 1 Do you feel that you successfully completed all the tasks on the task sheet?

Yes No

Question 2 In relation to other software I have used, I found the prototype to be:

Very difficult to use 1 … 2 … 3 … 4 … 5 … 6 Very easy to use

Question 3 I found the prototype to be:

Very difficult to use 1 … 2 … 3 … 4 … 5 … 6 Very easy to use N/A

Question 4 The menu items were well organized and functions were easy to find.

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 5 I immediately understood the function of each menu item.

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 6 All of the functions I expected to find in the menus were present.

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 7 The buttons were well organized and easy to find.

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 8 I immediately understood the function of each button.

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 9 All of the functions worked without error:

Strongly disagree 1 … 2 … 3 … 4 … 5 … 6 Strongly agree

Question 10 I found navigating around the application to be:

Very difficult 1 … 2 … 3 … 4 … 5 … 6 Very easy

Question 11 My overall impression of the prototype is:

Very negative 1 … 2 … 3 … 4 … 5 … 6 Very positive

Comments

Appendix C: IRB Approval

The University of Montana

INSTITUTIONAL REVIEW BOARD
for the Protection of Human Subjects
FWA 00000078

Research & Development
University Hall 116
The University of Montana
Missoula MT 59812
Phone 406-243-6670 | Fax 406-243-6330

Date: April 19, 2012

To: Corrine Olson, Computer Science
Min Chen, Computer Science

From: [X] Paula Baker, IRB Coordinator
[ ] Dan Corti, IRB Chair

RE: IRB 79-12: "Simplification of the Student Experience"

Your IRB proposal cited above is **exempt** from the requirement of review by the Institutional Review Board in accordance with the Code of Federal Regulations, Part 46, section 101. The specific paragraph which applies to your research is:

___(b)(1) Research conducted in established or commonly accepted educational settings, involving normal educational practices, such as (i) research on regular and special education instructional strategies, or (ii) research on the effectiveness of or the comparison among instructional techniques, curricula, or classroom management methods.

_X_(b)(2) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless: (i) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and (ii) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

___(b)(3) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior that is not exempt under paragraph (b)(2) of this section, if: (i) The human subjects are elected or appointed public officials or candidates for public office; or (ii) federal statute(s) require(s) without exception that the confidentiality of the personally identifiable information will be maintained throughout the research and thereafter.

___(b)(4) Research, involving the collection or study of existing data, documents, records, pathological specimens, or diagnostic specimens, if these sources are publicly available or if the information is recorded by the investigator in such a manner that subjects cannot be identified, directly or through identifiers linked to the subjects.

___(b)(5) Research and demonstration projects which are conducted by or subject to the approval of department or agency heads, and which are designed to study, evaluate, or otherwise examine: (i) Public benefit or service programs; (ii) procedures for obtaining benefits or services under those programs; (iii) possible changes in or alternatives to those programs or procedures; or (iv) possible changes in methods or levels of payment for benefits or services under those programs.

___(b)(6)

Taste and food quality evaluation and consumer acceptance studies, (i) if wholesome foods without additives are consumed or (ii) if a food is consumed that contains a food ingredient at or below the level and for a use found to be safe, or agricultural chemical or environmental contaminant at or below the level found to be safe, by the Food and Drug Administration or approved by the Environmental Protection Agency or the Food Safety and Inspection Service of the U.S. Department of Agriculture.

University of Montana IRB policy does not require you to file an annual Continuation Report (Form RA 109) for exempt studies. However, you are required to timely notify the IRB if there are any significant changes or if unanticipated or adverse events occur during the study, if you experience an increased risk to the participants, or if you have participants withdraw from the study or register complaints about the study.

Form RA-108
(Rev. 12/09)

IRB Protocol No. 79-12

THE UNIVERSITY OF MONTANA-MISSOULA

Institutional Review Board (IRB)
for the Protection of Human Subjects in Research
CHECKLIST / APPLICATION

At The University of Montana (UM), the Institutional Review Board (IRB) is the institutional review body responsible for oversight of all research activities involving human subjects outlined in the U.S. Department of Health and Human Services' Office of Human Research Protection and the National Institutes of Health, Inclusion of Children Policy Implementation.

Instructions: A separate application form must be submitted for each project. IRB proposals are approved for no longer than one year and must be continued annually. Faculty and students may email the completed form as a Word document to *IRB@umontana.edu*, or submit a hardcopy to the Office of the Vice President for Research & Development, University Hall 116. Student applications must be accompanied by email authorization by the supervising faculty member or a signed hard copy. *All fields must be completed. If an item does not apply to this project, write in n/a.*

1. Administrative Information

Project Title: Simplification of the Student Experience	
Principal Investigator: Corrine Olson	UM Position: Masters Student
Department: Computer Science	Office location: SS 423
Work Phone: 406-207-7619	Cell Phone: 406-207-7619

2. Human Subjects Protection Training

(All researchers including faculty supervisors for student projects, must have completed a self-study course on protection of human research subjects ***within the last three years*** *(http://www.umt.edu/research/complianceinfo/IRB/) and be able to supply the "Certificate(s) of Completion" upon request. Add rows in table if needed.)*

Research Team Members	PI	CO-PI	Faculty Supervisor	Research Assistant	DATE COMPLETED Human Subjects Protection Course
Name: Min Chen Email: min.chen@mso.umt.edu	☐	☐	☒	☐	4/11/2012
Name: Corrine Olsen Email: corrine.olson@umontana.edu	☒	☐	☐	☐	4/11/2012
Name: Email:	☐	☐	☐	☐	
Name: Email:	☐	☐	☐	☐	

3. Project Funding

(If federally funded, you must submit a copy of the abstract.)

Is grant application currently under review at a grant funding agency? ☐ Yes *(If yes, cite sponsor on ICF if applicable)* ☒ No

Has grant proposal received approval and funding? ☐ Yes *(If yes, cite sponsor on ICF if applicable)* ☒ No

Agency	Grant No.	Start Date	End Date	PI

For UM-IRB Use Only

IRB Determination:

____ Not Human Subjects Research
X Approved Exempt from Review, Exemption # 2 *(see memo)*
____ Approved by Expedited Review, Category # ____ *(see* ***Note to PI)**
Full IRB Determination
____ Approved *(see* ***Note to PI)**
____ Conditional Approval *(see memo)* - IRB Chair Signature/Date:
____ Conditions Met *(see* ***Note to PI)**
Resubmit Proposal *(see memo)*
Disapproved *(see memo)*

*** Note to PI:** Study is approved for one year. Use any attached IRB-approved forms (signed/dated) as "masters" when preparing copies. If continuing beyond the expiration date, a continuation report must be submitted. Notify the IRB if any significant changes or unanticipated events occur. Notify the IRB in writing when the study is terminated.

Risk Level: Minimal

Final Approval by IRB ~~Chair~~/Coordinator: __________ Date: 4/19/12 Expires: N/A

1

Approval Expires On N/A
Date Approved By UM-IRB 4-17-12
IRB-Chair

SUBJECT INFORMATION AND INFORMED CONSENT

Study Title: Simplification of the Student Experience

Investigator(s):

Name	*Email*	*Office*	*Phone*	*Role*
Corrine Olson	*corrine.olson@umconnect.edu*	*SS 423*	*406-207-7619*	*Primary Investigator*
Min Chen	*min.chen@mso.umt.edu*	*SS 413*	*406-243-2886*	*Faculty Advisor*

Special instructions:

This consent form may contain words that are new to you. If you read any words that are not clear to you, please ask the person who gave you this form to explain them to you. If you decide at any point that you would like to end your participation in this test, please simply notify the investigator.

Purpose:

- You are being asked to take part in this testing in order to facilitate research on mobile applications enhancing the student experience on the University campus.
- You have been chosen because you are a student, and as such regularly interact with the content of this application.
- The purpose of this testing is to ensure the usability of the prototype application and its usefulness to the student body.

Procedures:

- If you agree to take part in this test, you will be given an android device with the application prototype installed.
- You will be given a list of tasks to perform, which you will use the application to complete while being recorded (audio and device screen only).
- While completing the tasks, it would be helpful if you talked your way through them, but if this makes you uncomfortable, then you do not need to do so.
- Afterwards, you will be asked to complete a questionnaire about the tasks you completed and your overall experience with the prototype.

The study will take place in SS 402.

The session will last for as long as the tasks take to complete, but not more than 30 minutes. It will take about four minutes to complete the survey.

Risks/Discomforts:

Minimal risks or discomforts are anticipated in this study, but if you experience any inform the investigator immediately.

Benefits:

- Your testing may improve this application which, if released for public distribution as is the goal, will benefit you as a student

Approval Expires On N/A
Date Approved By UM-IRB 4-19-12
IRB-Chair

Confidentiality:

- The recording of the session will be reviewed and then destroyed, with no information identifying you associated with it.
- The questionnaire you fill out is anonymous, and once results are pulled from all questionnaires, they will be destroyed.
- This signed consent form will be stored securely, and not a part of the published work.

Voluntary Participation/Withdrawal:

- You participation is entirely voluntary.
- You may refuse to take part in the test or withdraw at any time.
- You may leave for any reason.

Questions:

- If you have any questions about the research now or during the test contact Corrine Olson (contact info above).
- If you have any questions regarding your rights as a research subject, you may contact the Chair of the IRB through The University of Montana Research Office at 243-6670.

Statement of Consent:

I have read the above description of this research study, I have been informed of the risks and benefits involved, and all my questions have been answered to my satisfaction. Furthermore, I have been assured that any future questions I may have will also be answered by a member of the research team. I voluntarily agree to take part in this study. I understand I will receive a copy of this consent form.

_____
Printed (Typed) Name of Subject

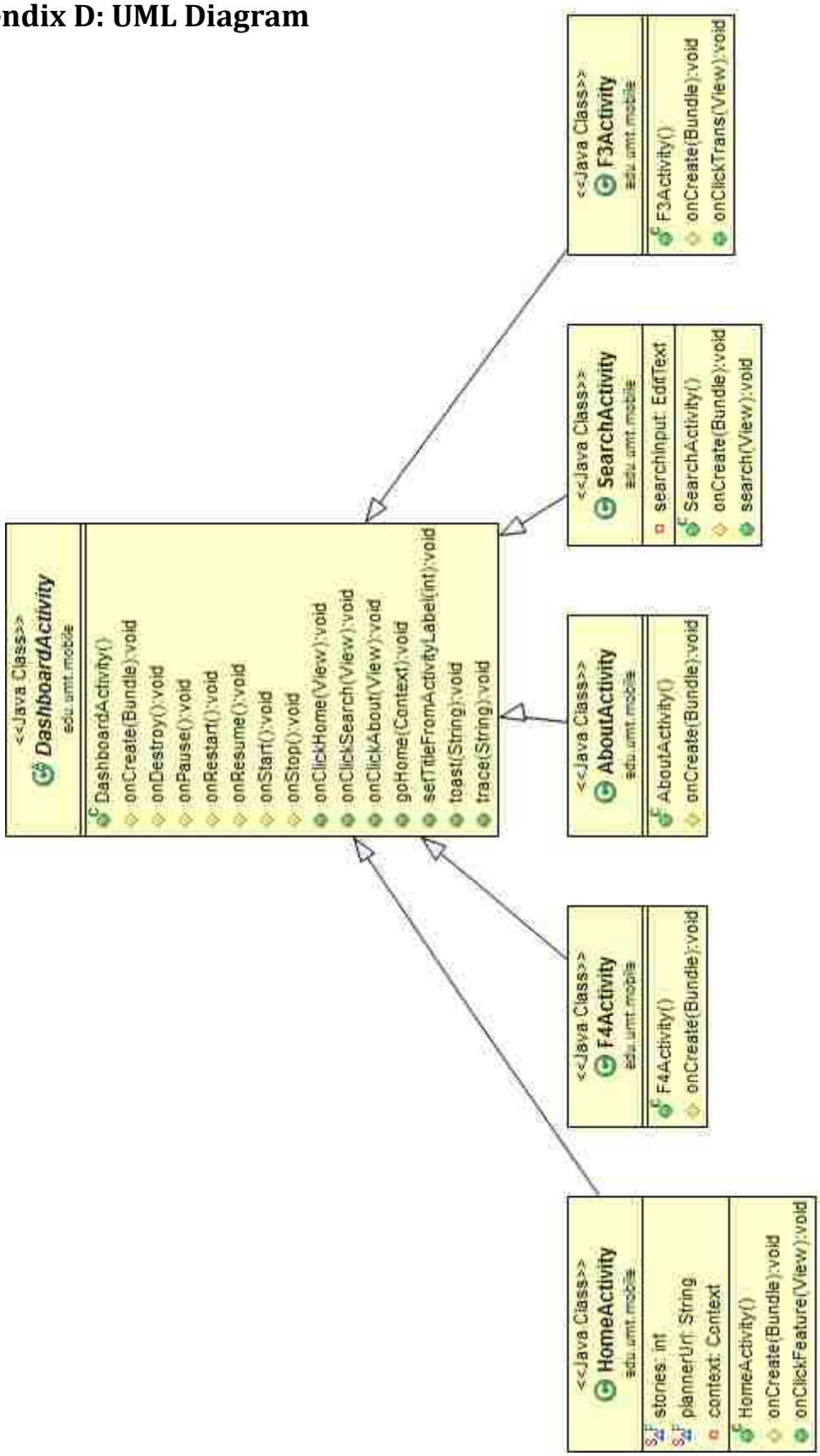
_____ _____
Subject's Signature Date

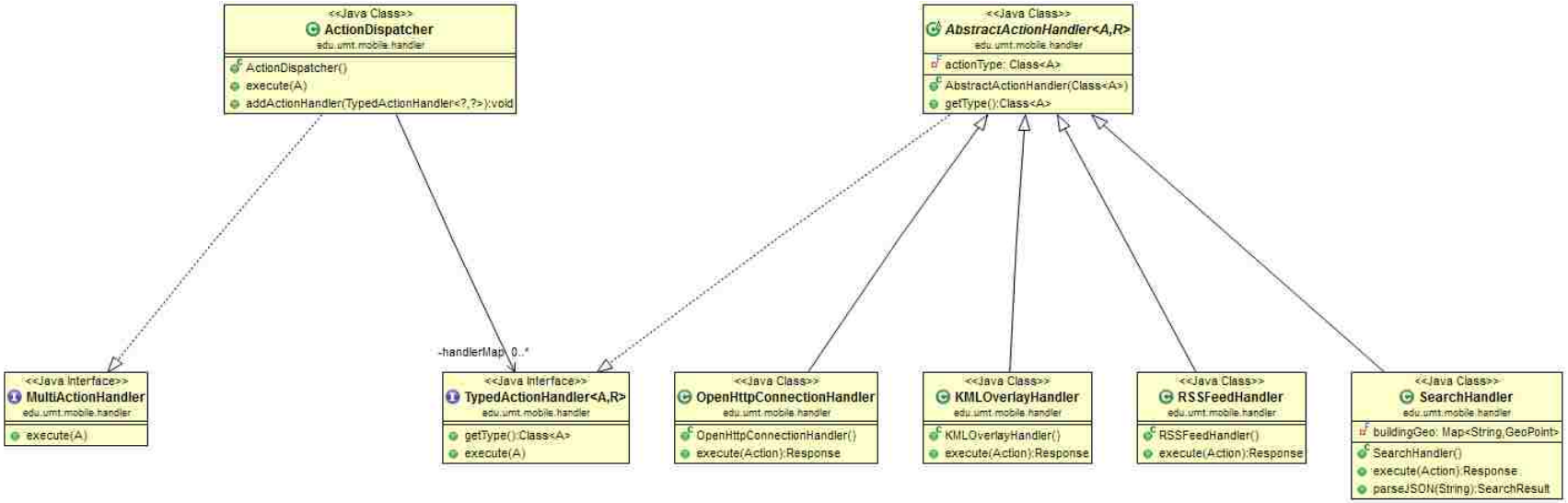
Statement of Consent to be Recorded:

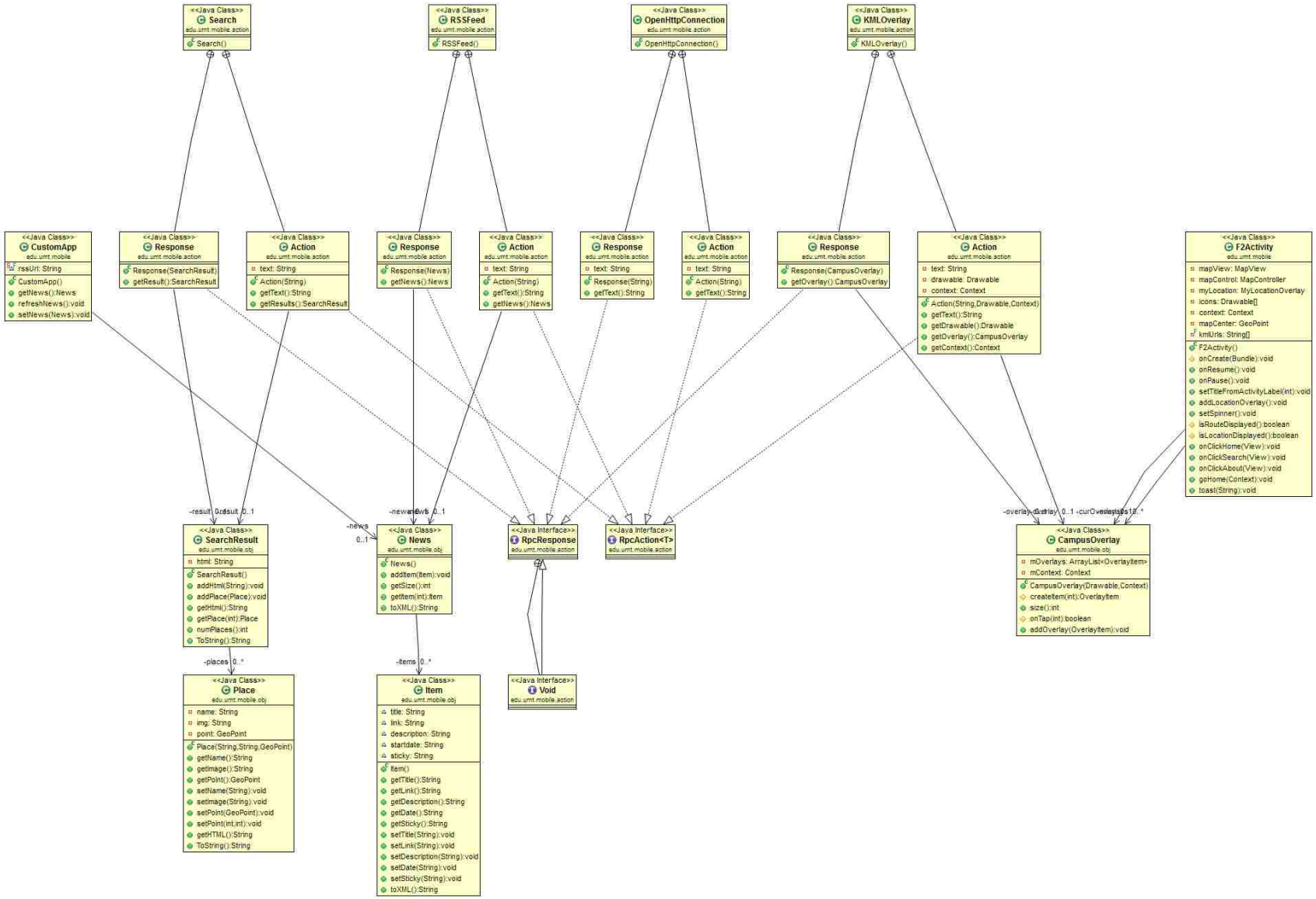
- I understand that video recordings may be taken during the test.
- I consent to be being recorded (screen actions and audio).
- I understand that the recordings will be destroyed following transcription, and no identifying information will be included in the transcription.

_____ _____
Subject's Signature Date

Appendix D: UML Diagram




<<Java Class>>
ActionDispatcher
edu.umt.mobile.handler
ActionDispatcher()
execute(A)
addActionHandler(TypedActionHandler<?,?>):void
<<Java Class>>
AbstractActionHandler<A,R>
edu.umt.mobile.handler
actionType: Class<A>
AbstractActionHandler(Class<A>)
getType():Class<A>
-handlerMap 0..*
<<Java Interface>>
MultiActionHandler
edu.umt.mobile.handler
execute(A)
<<Java Interface>>
TypedActionHandler<A,R>
edu.umt.mobile.handler
getType():Class<A>
execute(A)
<<Java Class>>
OpenHttpConnectionHandler
edu.umt.mobile.handler
OpenHttpConnectionHandler()
execute(Action):Response
<<Java Class>>
KMLOverlayHandler
edu.umt.mobile.handler
KMLOverlayHandler()
execute(Action):Response
<<Java Class>>
RSSFeedHandler
edu.umt.mobile.handler
RSSFeedHandler()
execute(Action):Response
<<Java Class>>
SearchHandler
edu.umt.mobile.handler
buildingGeo: Map<String,GeoPoint>
SearchHandler()
execute(Action):Response
parseJSON(String):SearchResult


<<Java Class>>
Search
edu.umt.mobile.action
Search()
<<Java Class>>
RSSFeed
edu.umt.mobile.action
RSSFeed()
<<Java Class>>
OpenHttpConnection
edu.umt.mobile.action
OpenHttpConnection()
<<Java Class>>
KMLOverlay
edu.umt.mobile.action
KMLOverlay()
<<Java Class>>
CustomApp
edu.umt.mobile
rssUrl: String
CustomApp()
getNews():News
refreshNews():void
setNews(News):void
<<Java Class>>
Response
edu.umt.mobile.action
Response(SearchResult)
getResult():SearchResult
<<Java Class>>
Action
edu.umt.mobile.action
text: String
Action(String)
getText():String
getResults():SearchResult
<<Java Class>>
Response
edu.umt.mobile.action
Response(News)
getNews():News
<<Java Class>>
Action
edu.umt.mobile.action
text: String
Action(String)
getText():String
getNews():News
<<Java Class>>
Response
edu.umt.mobile.action
text: String
Response(String)
getText():String
<<Java Class>>
Action
edu.umt.mobile.action
text: String
Action(String)
getText():String
<<Java Class>>
Response
edu.umt.mobile.action
Response(CampusOverlay)
getOverlay():CampusOverlay
<<Java Class>>
Action
edu.umt.mobile.action
text: String
drawable: Drawable
context: Context
Action(String,Drawable,Context)
getText():String
getDrawable():Drawable
getOverlay():CampusOverlay
getContext():Context
<<Java Class>>
F2Activity
edu.umt.mobile
mapView: MapView
mapControl: MapController
myLocation: MyLocationOverlay
icons: Drawable[]
context: Context
mapCenter: GeoPoint
kmlUris: String[]
F2Activity()
onCreate(Bundle):void
onResume():void
onPause():void
setTitleFromActivityLabel(int):void
addLocationOverlay():void
setSpinner():void
isRouteDisplayed():boolean
isLocationDisplayed():boolean
onClickHome(View):void
onClickSearch(View):void
onClickAbout(View):void
goHome(Context):void
toast(String):void
-result 0..1
-result 0..1
-news 0..1
-news 0..1
-overlays 0..*
-overlay 0..1
-curOverlay 0..1
-overlays 0..*
<<Java Class>>
SearchResult
edu.umt.mobile.obj
html: String
SearchResult()
addHtml(String):void
addPlace(Place):void
getHtml():String
getPlace(int):Place
numPlaces():int
ToString():String
<<Java Class>>
News
edu.umt.mobile.obj
News()
addItem(Item):void
getSize():int
getItem(int):Item
toXML():String
<<Java Interface>>
RpcResponse
edu.umt.mobile.action
<<Java Interface>>
RpcAction<T>
edu.umt.mobile.action
<<Java Class>>
CampusOverlay
edu.umt.mobile.obj
mOverlays: ArrayList<OverlayItem>
mContext: Context
CampusOverlay(Drawable,Context)
createItem(int):OverlayItem
size():int
onTap(int):boolean
addOverlay(OverlayItem):void
-places 0..*
-items 0..*
<<Java Class>>
Place
edu.umt.mobile.obj
name: String
img: String
point: GeoPoint
Place(String,String,GeoPoint)
getName():String
getImage():String
getPoint():GeoPoint
setName(String):void
setImage(String):void
setPoint(GeoPoint):void
setPoint(int,int):void
getHTML():String
ToString():String
<<Java Class>>
Item
edu.umt.mobile.obj
title: String
link: String
description: String
startdate: String
sticky: String
Item()
getTitle():String
getLink():String
getDescription():String
getDate():String
getSticky():String
setTitle(String):void
setLink(String):void
setDescription(String):void
setDate(String):void
setSticky(String):void
toXML():String
<<Java Interface>>
Void
edu.umt.mobile.action