FARGO MOORHEAD BUS ROUTES ANDROID APPLICATION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Avijeet Tomer

In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

January 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

FARGO MOORHEAD BUS ROUTES ANDROID APPLICATION

By

AVIJEET TOMER

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia
Chair

Dr. Kendall E. Nygard

Dr. Simone Ludwig

Dr. Joseph G. Szmerekovsky

Approved by Department Chair:

1/28/2013
Date

Dr. Kenneth Magel
Signature

ABSTRACT

Fargo and Moorhead have good public transportation provided by Metro Area Transit bus service. Google Maps Navigation, which provides public transportation routes and directions in several major cities, is not available in Fargo or Moorhead, so bus users have to either carry and use printed maps provided by MATBUS to determine the bus routes they need to take to reach their destination, or if they have a smartphone with internet connectivity, they can visit MATBUS website to view the maps on their device. This paper examines the inconveniences of both procedures like using a big printed map outside in Fargo winter, or panning across an approximately A0 sized PDF (actual dimensions are 30.67 inches width and 46.67 inches height) on the small screen of a smartphone and presents a tool for Android-based smartphone users to alleviate some of those inconveniences.

ACKNOWLEDGEMENTS

I'm grateful to my adviser Dr. Gursimran Walia for his continuous help, support, patience and guidance in the development and completion of this project and paper. He has been a great adviser who was always available to answer my questions with detailed guidance instead of just replying with facts and helped me finalize on a very practical and useful topic for my project and paper by patiently taking out significant amount of time over my semesters in the Master's program to have rational discussions about my choices and decisions.

I'm grateful to Dr. Kendall Nygard for always being helpful with various stages in the progress of my Master's program and providing useful guidance in program related issues, and for taking out the time to be a part of my supervisory committee.

I'm grateful to Dr. Simone Ludwig for taking out the time to be a part of my supervisory committee despite my not having had any course or seminar with her in my Master's or Bachelor's program in Computer Science department.

I'm grateful to Dr. Joseph Szmerekovsky for taking out the time to be a part of my supervisory committee despite not being a faculty of Computer Science department and having to travel from a two mile far building for my final defense!

Finally, I'm grateful to the Computer Science department faculty and staff in all the ways I could use their help in the progress and completion of my Master's program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

1. INTRODUCTION

The MATBUS service in Fargo-Moorhead area was my de facto mode of transportation for the first three years here before I bought a car. In a bigger city like Minneapolis, MN, one could get public transit directions via Google Maps Navigation on their Android smartphones when they are outside. Fargo-Moorhead area is not covered by that facility [11]. MATBUS provides an approximately A1 sized map showing all bus schedules and routes. By going to the MATBUS website [23], you can download not only a PDF of that map, but also smaller (more conveniently sized) PDFs of individual bus schedules. NDSU makes this a step easier for the students by providing a leaflet of the 5 NDSU bus routes, route 31 to 35 and two more that pass through NDSU, route 13 and route 13U.

Finding my own location in the big map, following routes of buses in that map with all the routes jammed in together, scanning through the time schedules of buses with all of them again on that one big map, and having to do that outside in the winter of Fargo was not a convenient experience. While it was more convenient to view the small sized individual time schedules of the buses, they did not show the routes of those buses on map. The NDSU transit guide leaflet provided schedules as well as routes, but besides being useful only for NDSU, it presented the difficulty of determining intersecting routes to know where to change buses, because the routes were displayed individually. In order to not have to carry those maps, after I got a smartphone, the Android OS based HTC Eris, I downloaded the PDFs onto it and used them while outside. Since I had internet connectivity, I could go to MATBUS and NDSU websites to download changed routes if required. However, while the viewing the individual schedules and NDSU transit guide was doable, the same problems of using their printed versions were still present. On the other hand, viewing the PDF version (approximately of A0 size) did

overcome those problems, but it had its own problem: its dimensions were too big for the phone. It took some seconds for a part of the PDF to be rendered on the 3.4 inch sized screen's visible area, and then I had to pan it across to the area in order to view, zoom in and out, and wait for it to render after those actions.

The inconvenience and difficulties of using both the printed maps and the PDFs on that smartphone motivated me to develop and publish the first version of Fargo Moorhead Bus Routes Android application in Google Play store (then Android Market) in December 2010. It had the schedules and maps of only five NDSU routes, 31 to 35. It got a good response from users on Google Play store (4.1 out of 5 star rating) and I still receive comments and email requests to add more routes to it. Though I don't live in Fargo any more to use the MATBUS service, those requests and the lack of an alternative solution of that type in Fargo/Moorhead area were a motivation for me to undertake this project.

In Android SDK version 1.0, developers could generate a route between two points and show it on an embedded map view in their application. Since version 1.1, Google removed that functionality from the SDK [24] to promote the use of Google Maps application (available for free on Android based devices) for routing purposes, allowing developers to only show points of data on the map views in their application (places of interest, user's location, location of other users, etc. depending on application). However, there was a workaround for this: one could create a desired map on Google Maps website [25], then point to its KML (Keyhole Markup Language) URL in the application (or download the KML file and package it with the application to parse it) and get the coordinates of the route, then pass them to Google Maps service to get the route path information in KML format (so it is exactly the same) and draw it on the map view in the application. I used the same approach in my old application and it worked

well till July this year, after which Google stopped giving out that information in KML format, only JSON and XML formats are provided now. While that workaround would still work for simply providing "a route" between two coordinates, or the shortest one as per Google Maps' algorithms, because we can simply provide the coordinates we get in the KML to the Maps service, and use the returned JSON or XML to draw a path on map view, and even if the returned path information differs from your original creation on Maps website, it would still be a valid path. However, my problem was different: I was not looking for the shortest path between two points, but a closed loop to exactly denote the route of a bus. So the difference in the route information between what I drew on Google Maps website and the information in the JSON or XML provided by the service when requested with the coordinates from my created map would alter the route of the bus.

So for my new application, I embraced Google Maps application and use particular features of it to my advantage with my application and for the benefit for the users of my application: the ability to layer two routes to view them together to determine where to change buses and walking directions to a bus stop. The data to create those routes in Google Maps application is available on my Google Maps account and is accessible via a public URL. In the old application if I had to update a route or add new one, I had to add or modify the code and update the application package on Google Play, making the users having to update the entire application on their devices. Since the coordinates were hard coded, changing a route or adding a new one took a significant amount of time.

In the new application, I adopted a MVC (Model-View-Controller) approach and cleanly separated the model (data and the methods to manage it). The model retrieves the data on the application's first run from a server that I have set up, and after that, only updates the data that

has been changed or added on the server, instead of downloading the entire application. So if I have to update a route or add a new one, I make the changes to the route saved on my Google Maps account or add a new route and make it public and add the update information (the changed or new URL and description of the update) to a file on the server. Besides the advantage of minimizing the bandwidth consumed by not having to re-download all the data for a small update, another advantage to this approach is the more accurate definition of what the term "update" defines: the application version does not change in Google Play when I update a bus route or add a new one, which is the way it should be, because I am not changing the application itself, only its data.

The new application presently has the schedules of all MATBUS buses and the routes of the 5 NDSU routes. No information is hard coded into the app. An internet connection is required on the first run of the application, which in a normal use case, would be available because the application would have been downloaded from Google Play, to download the schedules and maps information. After that, an internet connection is required to update the information or view maps, but not to just view the downloaded schedules.

1.1. Explanation of terms

Terms used in the paper:

1. Android: An open source operating system primarily meant for touchscreen mobile devices like smartphones and tablets owned by Google since 2005 [1].
2. OS: Short for operating system, it's a software that manages the hardware of a computer system and provides common services for applications used on that system like managing application's access to hardware [2].
3. A0: ISO 261 page size 46.8 inches height and 33.1 inches width [3].

4. A1: ISO 261 page size 33.1 inches height and 23.4 inches width [3].
5. PDF: Short for Portable Document Format, it's an open standard for electronic exchange of documents which are mirror images of the way they appear in print [4].
6. Google Maps: Google's online mapping service, accessible at URL https://maps.google.com/ [25] and through application called Google Maps for Android and iOS mobile operating systems. While a feature of the online service is providing directions between two points on map for driving a car, using public transportation, riding a bicycle or walking, the mobile applications go a step ahead and use the device's GPS to provide turn-by-turn navigation.
7. Google Play: Google's applications and media distribution service for Android OS based devices, accessible at URL https://play.google.com/store [31].
8. URL: Short for Uniform Resource Locator, it's a unique string that points to an address on the Internet.
9. String: In terms of computer programming, it's a sequence of characters which may have a constant or variable value [5].
10. SDK: Short for Software Development Kit, it's a set of software development tools put together for development of applications for a certain software platform [6].
11. KML: Short for Keyhole Markup Language, it's a file format used to display geographic data [7].
12. XML: Short for Extensible Markup Language, it's a simple data exchange format [8].
13. JSON: Short for JavaScript Object Notation, it's another data exchange format which is simpler than XML [9].

14. MVC: Short for Model-View-Controller, an architecture or pattern followed in development of software to modularize the user interface and interaction part of software, the data and its management, and the control and management of the application itself [10].
15. Crowdsourcing: The act of outsourcing a task to a large group of people with common goals or interests [12].
16. Use case: An action or a sequence of actions that provides a measurable value to a user [13].
17. Client-server: A model of organizing computer systems in a network such that one system, called a server, selectively shares its resources and one or more systems, call clients, connects to the server to use those resources [14].
18. Java: A popular computer programming language.
19. IDE: Short for Integrated Development Environment, it's a software that provides a comprehensive set of tools like code editor, build automation, code debugger for software development [15].
20. Eclipse: A popular IDE.
21. API: Short for Application Programming Interface, it's a set of functions, data structures, classes and variables exposed as an interface for other software components to interact with it [16].
22. Class: In programming, a class is a construct that defines a specific object, usually a noun, and has its properties and functions that allow interaction with that object [17]. A class diagram shows the classes of a system, their properties and functions, and the relationships between the classes [13].

23. HTML: Short for Hypertext Markup Language, it's the standard language to define the markup of a web page.
24. Sequence diagram: Shows the flow of logic in a system. The instance of a class calling functions of other classes using their instances are displayed in order of occurrence from top to bottom, with solid headed arrows representing synchronous calls and empty headed arrows representing asynchronous calls [13].
25. Black box testing: A method to test the functionality of a software without knowledge of its internal structures or workings [18].
26. State: In programming, the state of an application is the term for information available to that application and its output determined by that information [19]. A state transition is the set of the initial state of an application, an event and the application's new state due to the event.
27. SQLite: A server-less, self-contained database engine [20].

1.2. Organization of paper

The remainder of this paper is organized as follows: Section 2 discourses the research into existing methods of determining directions for public transit, not limited to Fargo/Moorhead area; section 3 discourses the design phase of the application, determining the use cases and planning an architecture; section 4 discourses the developed application, the classes and sequence of flow of logic, and the flow of a user's interaction with the application exemplified by screenshots; section 5 discourses the testing and its results; and section 6 discusses some shortcomings of application with suggested improvements.

2. BACKGROUND RESEARCH

Google Maps application on Android smartphones with its navigation feature is popularly for public transit directions. In some major cities, developers have made applications specific to their areas with additional features like, for example, the arrival time of an intra-city train. In Fargo/Moorhead area, neither does Google Maps application have public transit navigation, nor is there any other application for it.

2.1. Google Maps Navigation

Google Maps Navigation provides point to point public transit directions in about 500 cities around the world [11]. You choose the starting point (usually your current location, which can be detected by Google Maps, if you choose to allow it), your destination and the time you want to start (usually the current time).

In **Figure 1**, the search bar on the top of the application screen is activated by pressing the search key on the bottom left of the application. Search returns matches that are closest to the area displayed on the screen (around user location by default). This is the map view of search results, Mall of America as the first result. Tapping on the arrow on the left of balloon opens navigation options to that destination.

Choosing to navigate using public transit, routes are suggested with the start and end time of travel, as shown in **Figure 2**. Tapping on one of the suggested routes opens up a map view that shows the paths you need to walk, the paths you would ride a bus or/and the paths you would ride an intra-city train to reach the destination, as shown in **Figure 3**.

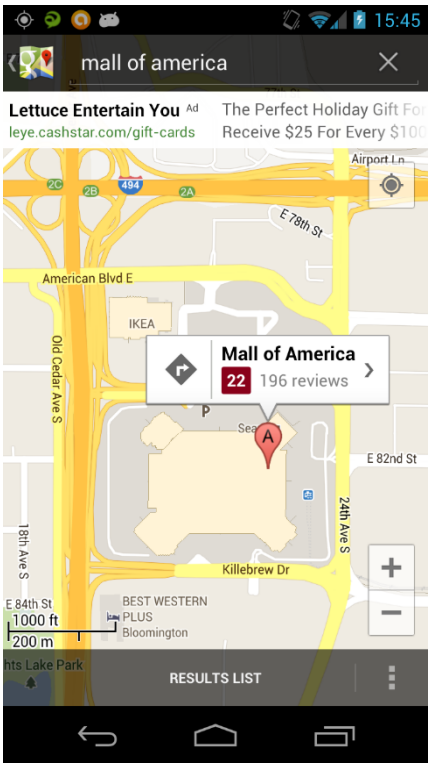

Figure 1. Searching for destination on Google Maps application

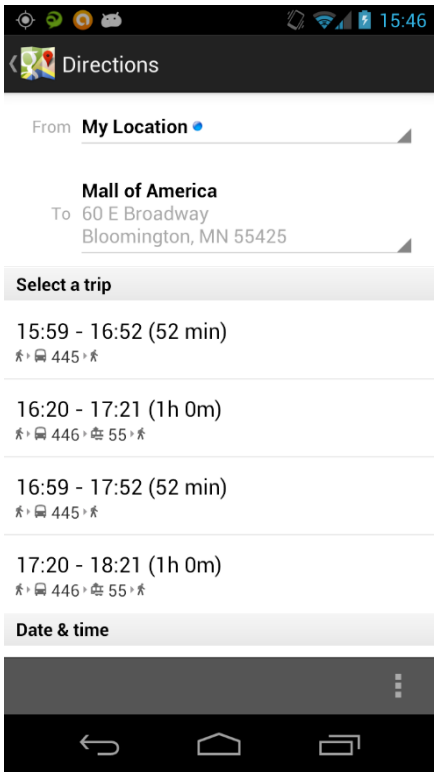

Figure 2. List of suggested public transportation routes from source to destination

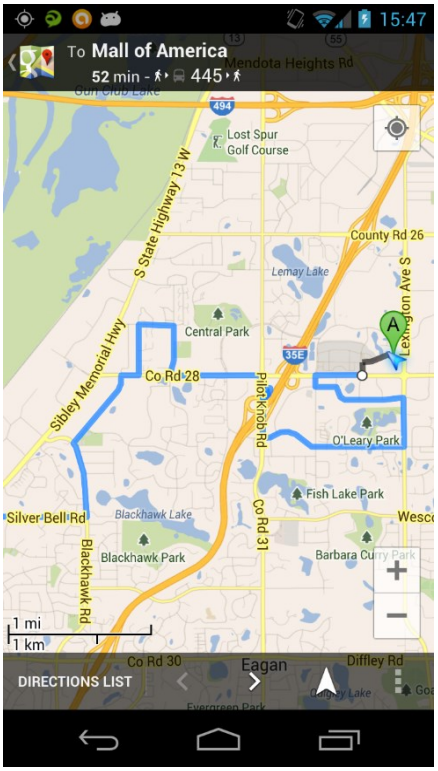

Figure 3. Map view in Google Maps Navigation

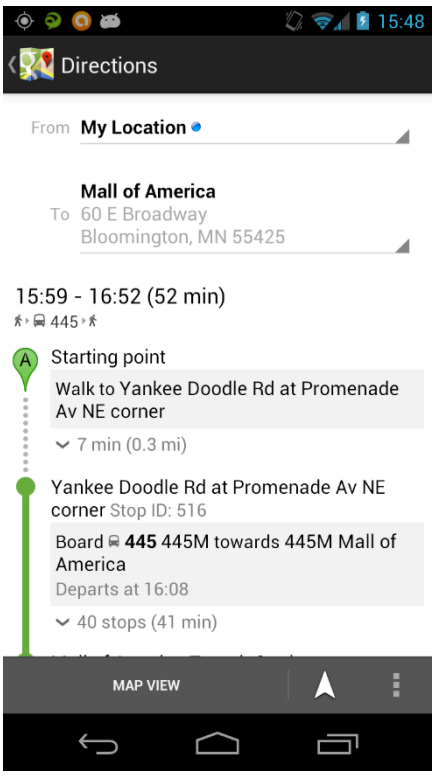

Figure 4. List view of the directions of the route

Tapping Directions List lists the steps to take to follow that public transit route, as shown in **Figure 4**. Tapping on Map View takes to back to viewing the route on map, as in **Figure 3**. Public transit in Fargo/Moorhead area is not covered by Google Maps data [11].

2.2. Alternatives to Google Maps Navigation in other cities

NYCMate is a public transit guide Android application for New York City [26], which also shows the real time train arrival timings by using crowdsourcing information. **Figure 5** shows a screenshot of the application.

Chicago Transit Tracker, on the same OS, is for Chicago, which provides official train tracking information [27]. **Figure 6** shows its screenshot.

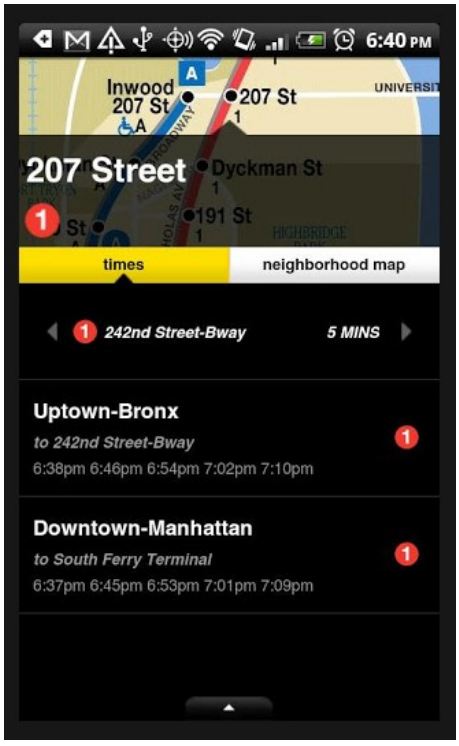

Figure 5. NYCMate application screenshot

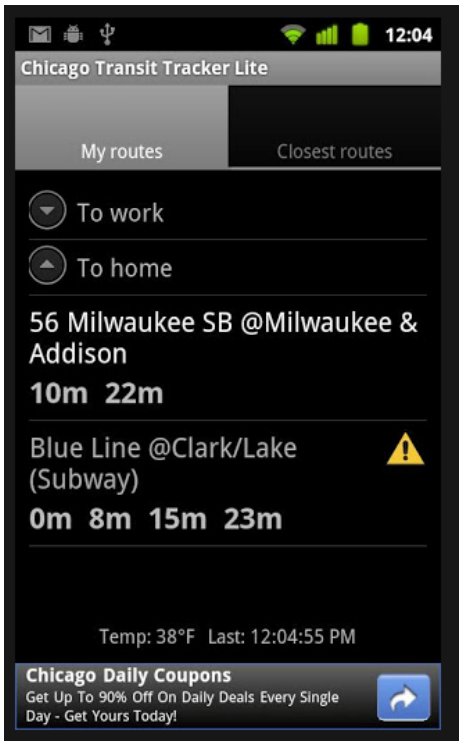

Figure 6. Chicago Transit Tracker application screenshot

2.3. Determining bus routes in Fargo/Moorhead area

Determining which bus or buses to take to reach a particular destination in Fargo and Moorhead area can be done by using the map provided by MATBUS service or using a smartphone to view that map and/or bus schedules, avoiding the need to carry a big map.

2.3.1. Using MATBUS provided map

The map provided by MATBUS is approximately of A1 size, and shows all bus routes on the map of both Fargo and Moorhead. It is shown in **Figure 7**.

The typical process of determining the route to take is:

1. If unaware of it, learn your present location, by looking at your surroundings and comparing to the map.
2. For bus routes closest to your location, trace their paths to see how they can connect you to your destination, whether they go there directly or if you have to change buses.
3. Check the schedule of the bus, or schedules of the buses if a change is required. If the bus or buses are available, determine the time required to walk to the nearest stop and start accordingly. Otherwise, repeat the steps if another route is available nearby.

The advantages of this method are:

1. It is provided free by MATBUS at their bus station in Fargo downtown as well as on all the buses.

The disadvantages are:

1. You are required to know your own location.
2. It is not convenient to use a big map outside if the weather is not good.
3. It could get confusing to understand where more than one routes are shown on same roads in a small area on the map.

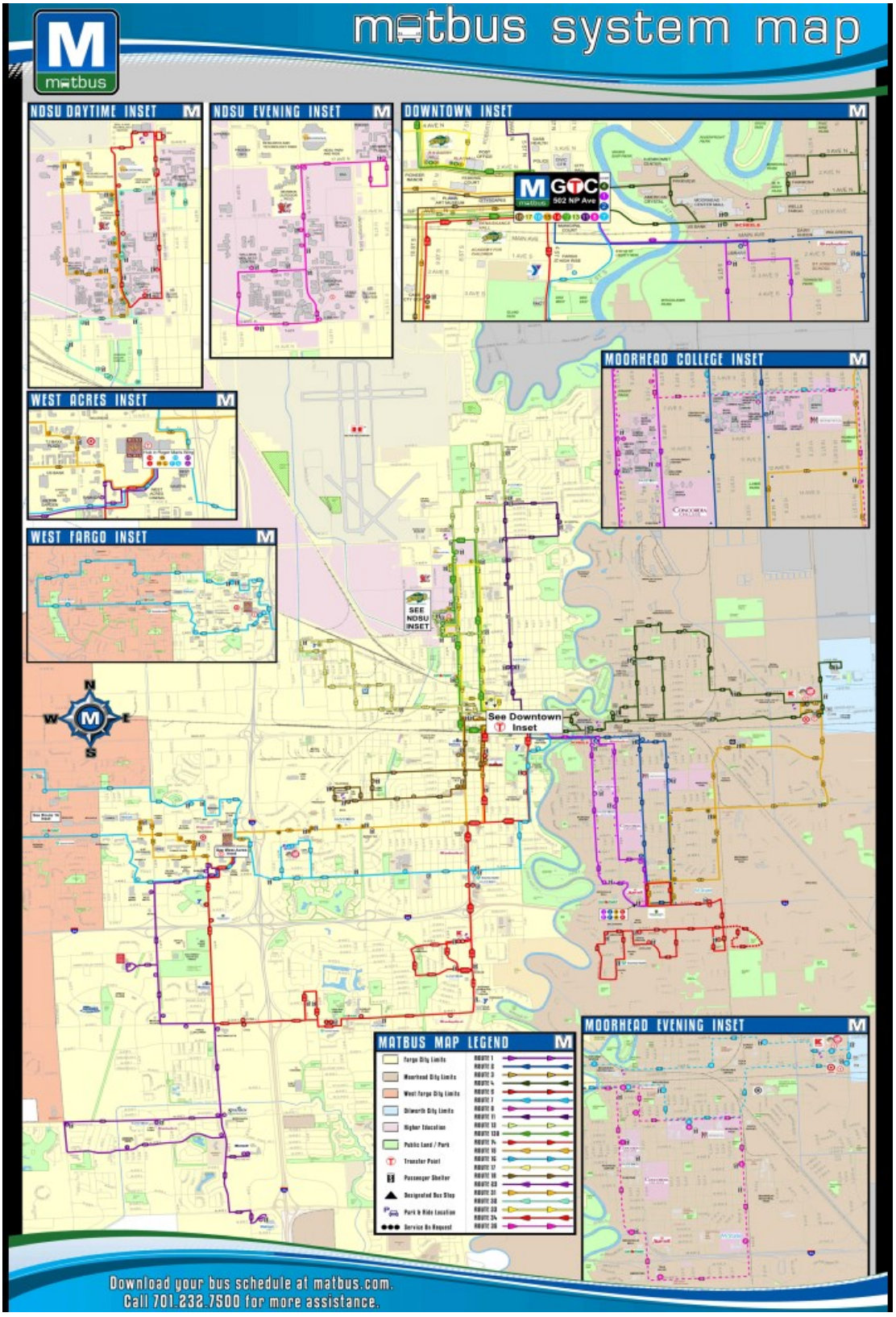

Figure 7. MATBUS provided map

Project Objectives:

Objective 1: The user should be able to find his own location on a map or Fargo/Moorhead area.

Objective 2: The project should make it more convenient for user to view a map of Fargo/Moorhead area than having to use a big map outside in winter.

Objective 3: The project should make it more convenient for user to view the schedules of bus routes of Fargo/Moorhead area than having to use a big map outside in winter.

Objective 4: The project should make it more convenient for user to view the routes of buses on a map of Fargo/Moorhead area than having to use a big map outside in winter.

Objective 5: The user should be able to view one bus route at a time on the map.

2.3.2. Using smartphone while outside

MATBUS website provides this map in PDF format [22] and JPEG format [28]. It also provides schedules of individual bus routes [29] in PDF format.

A smartphone with access to internet can be used to visit the MATBUS website to access the map. Alternatively, the PDFs can be downloaded to the phone when access to internet is available to be viewed later when outside.

The advantages of this method are:

1. Not having to carry the big map and facing the discomfort of using it in in a bad weather, compared to smaller size of a phone.

2. If viewing routes is not required but only the schedules, then finding schedules on a big map can be avoided by viewing individual bus schedules.

The disadvantages are:

1. A smartphone capable of viewing PDF file format and/or JPEG image format is required.
2. Internet access is required at least one time, either to download the files using a computer and transfer them to the phone, or if the phone is capable of browsing a website and downloading files from it, then doing it directly on it.
3. If the task is to be done outside where no wireless internet access point is available or accessible, then mobile internet connection is required on the phone. The advantage is always getting the latest updated map and/or schedules.
4. Viewing an approximately A0 sized PDF file or JPEG image on the small screen of a smartphone involves a lot of panning and zooming in/out. A smartphone does not have the processing power of a personal computer and it takes significant time for it to render the area of the map the user pans or zooms to.
5. The website provides only the bus routes on the map in one file, whether in PDF format or JPEG, and schedules of the buses separately.

Project Objectives:

Objective 6: The user should not have to wait for a significant amount of time for the map to be rendered by the smartphone.

Objective 7: The project should make it convenient for user to view the schedule of a bus route and its map more conveniently than the MATBUS website.

3. DESIGN

The design phase including determining the use cases for the application and planning out an architecture for development.

3.1. Use cases

The high-level use case diagram is shown in **Figure 8** and is discussed below in terms of the functionalities and the relationships between different use cases.

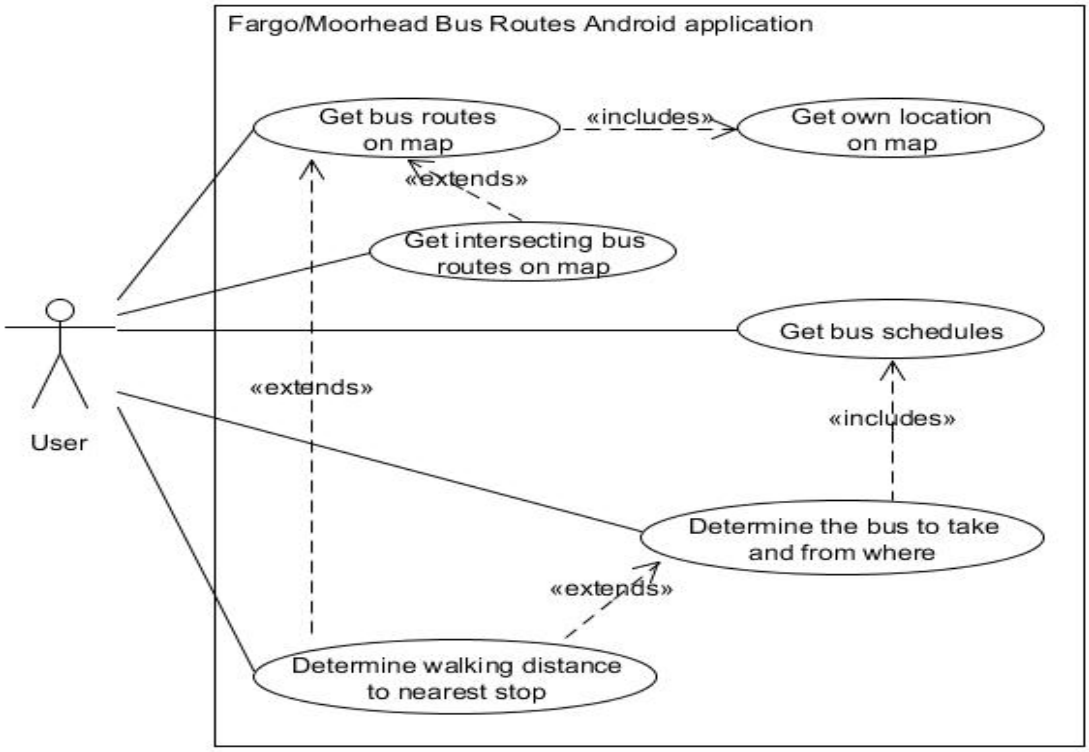

Figure 8. Use case diagram

1. The user can view a bus schedule in the application.

2. The user can view a bus route on Google Maps application. If the user is near the route, he would also see his own location. Viewing own location is a feature of Google Maps application.
3. The user can view two routes at the same time on Google Maps application to see their intersection and determine where to change the bus.
4. The user can determine the walking distance to the nearest stop of his chosen route using Google Maps application.

3.2. Architecture

A MVC (Model-View-Controller) architecture is used in the development of the application as shown in **Figure 9**.

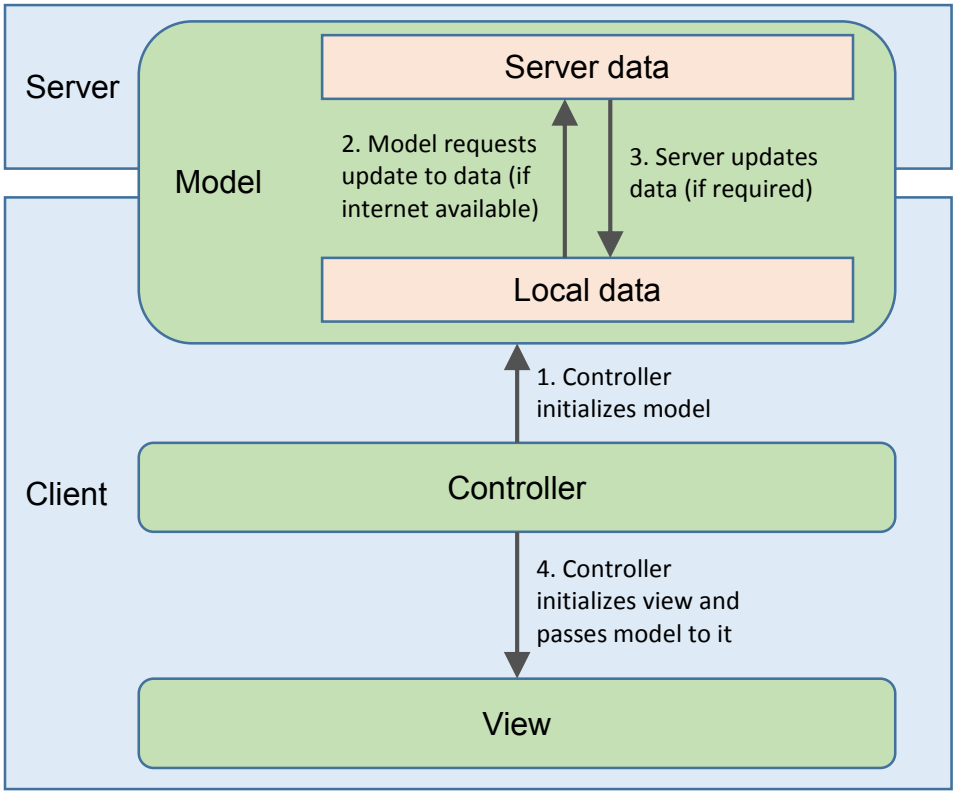

Figure 9. Architecture diagram

The model layer contains the data and includes the functions to retrieve, use and change it. The view layer presents the model to the user in a meaningful way and includes the functions to respond to user's interaction with the application. The controller layer contains the functionality which manages the application, invoking the appropriate view on application startup and passing the appropriate model or models to a view according to user interaction with the application.

The application also follows a client-server architecture. On starting the application, a connection to the server is attempted (if the device has access to internet) to check for an update to local data (schedules of the buses and URLs to their routes on Google Maps).

4. DEVELOPMENT

The development phase followed the planned architecture accurately in terms of implementing classes and their behavior. The developed application benefited from the MVC architecture by separating out the updates to only the data from updates to the whole application.

4.1. Tools

Fargo/Moorhead bus routes Android application was developed using Java SE 1.7 in Eclipse 3.7 IDE, with Android 2.2 SDK (API Level 8).

Android applications developed for a chosen platform are only forward compatible, so a balance should be made between choosing newer platforms with more efficient development and older ones with greater device compatibility coverage, depending on the requirements of the application and the time available to develop it.

As of January 3, 2013, using API Level 8 makes this application compatible with 97.4% Android devices [30].

4.2. Classes

Following MVC architecture, the application is divided into three packages, com.fargomoorheadbusroutes.controller to contain the controller classes, com.fargomoorheadbusroutes.model to contain the model classes and com.fargomoorheadbusroutes.view to contain the view classes. Only one class of each type ended up being required for this application. **Figure 10** shows the classes with their properties and methods.

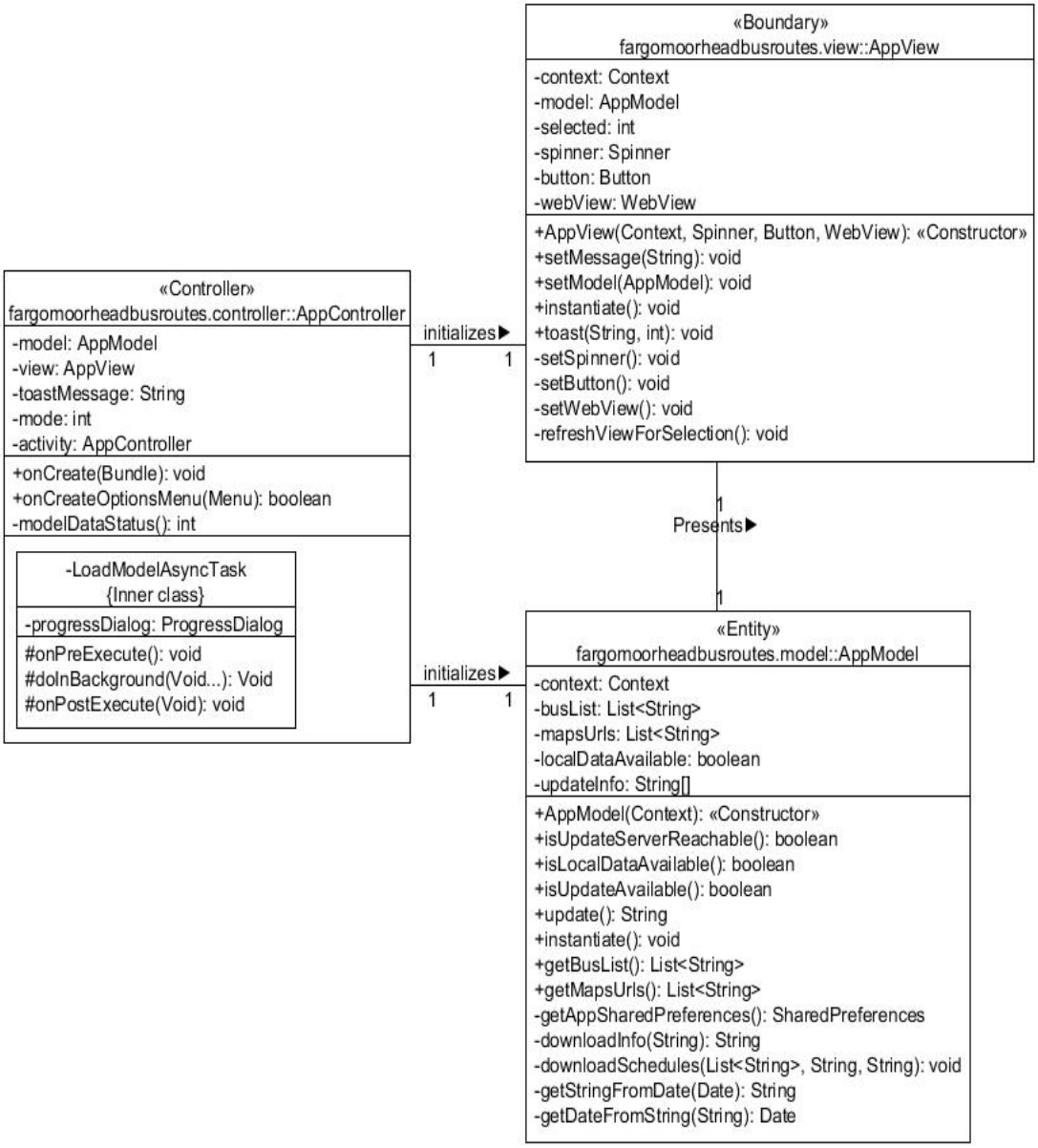

Figure 10. Class diagram

Android OS follows a stack based architecture [21], where each "view" that the user sees on the screen is defined as an activity. An application is comprised of at least one activity, which the user sees when he runs it on the device. The application may start new activities, which may or may not belong to it. For example, a restaurant finder application may have the first activity

show you a list of types of restaurants, after you choose one, a second activity opens up and shows you Google search results of that type in a WebView, a user interface element in Android to render and show HTML. The same application can be developed in another way. When you click on a type of restaurant in the first activity, it starts a browser application activity to show you the Google search results. Both approaches have their advantages, using the first one is better when more work is to be done in the application upon user interaction in the second activity, because the context is still within the same application; while the second one is better when the application doesn't need to listen to user interaction in the second activity and features of the application launched (a browser application in this case) could be useful, like bookmarking the webpage opened up, or downloading a file from a link in it. The user can navigate backwards in the activity stack using the standard Android back key, or switch applications using the multitasking key on the newer Android devices or by long pressing the home key on older ones, which brings up the list of last used applications on the device, with the last viewed activity on each of them.

The AppController class in this application is the main and only activity class, which instantiates the AppModel class and passes it to the AppView class to present it to the user. AppView (in the same activity) launches Google Maps application activity when user requests the route of a bus, allowing taking advantage of the features of Google Maps application, like layering two routes and determining walking distance and time to a stop from user's location.

4.3. Sequence

Figure 11 shows a high-level sequence diagram of the application's flow of logic. LoadModelAsyncTask inner class is of type AsyncTask, which is used in Android SDK to run asynchronous background operations, like in this case, downloading updates.

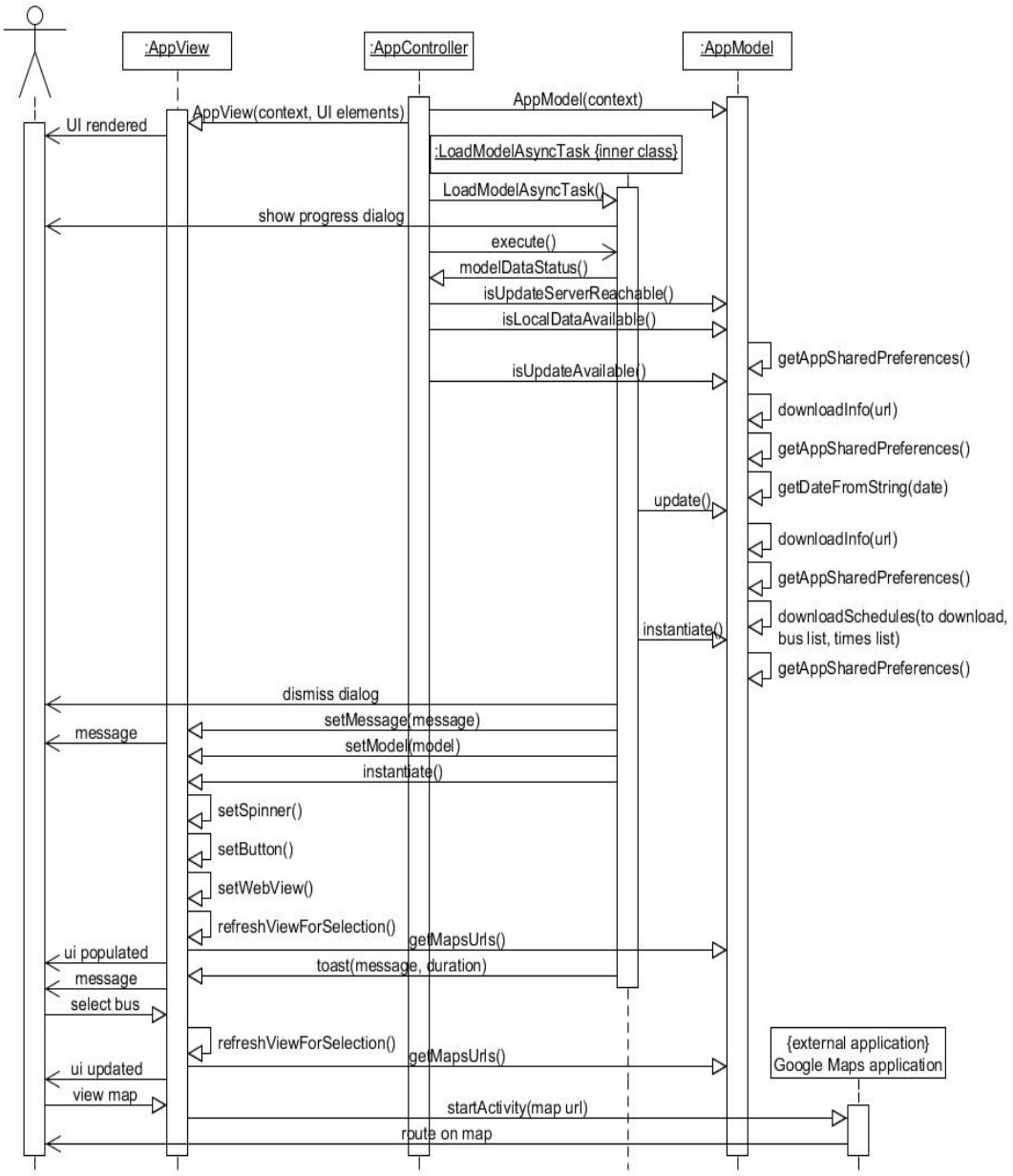

Figure 11. Sequence diagram

4.4. User interaction flow

The application can be installed on an Android device with access to internet by opening Google Play Store application, searching for "Fargo Moorhead Bus Routes", opening up application description in the search results, tapping Install button followed by tapping Accept & download button. On first run, the application downloads the initial data for the application from the server. Screenshots of the procedure are provided in alphabetical order in **Figure 12**.

To view the schedule of a particular route, select it from the dropdown on top left. At present, the application has schedules of all the routes, but maps of only the five NDSU routes (31 to 35). The button on top right becomes enabled if a route's map is available. Tap on the Map button to view the bus route on Google Maps application. Tapping on a stop shows its name and letter (to match it with schedule). To find the time the bus arrives on that stop, tap the multitasking key if available (available on newer Android devices), or long press the home key (on older Android devices) to switch back to Fargo Moorhead Bus Routes application. To view two routes on the map to see where they intersect, which is helpful to determine where to change buses if the one you can catch near your location doesn't take you to your destination, select the other one and tap the Map button. Google Maps application will layer this and the previous one together. Screenshots of the procedure are provided in alphabetical order in **Figure 13**.

To view only one of them, you can unselect the other layer by opening up Google Maps application menu (the 3 vertical dots key on bottom right), tap on Layers option and unselect one of the routes. To determine the walking distance to a stop, tap on it and tap the navigation arrow on the left of the balloon that pops up on the stop. Tap on the Walking directions option on the right (default selection is Car directions). Since I was in Minneapolis when I worked on the application, I chose a location near that stop as source instead of my location. Tap on Get

Directions button to get the directions on walking to that stop. Screenshots of the procedure are provided in alphabetical order in **Figure 14**.

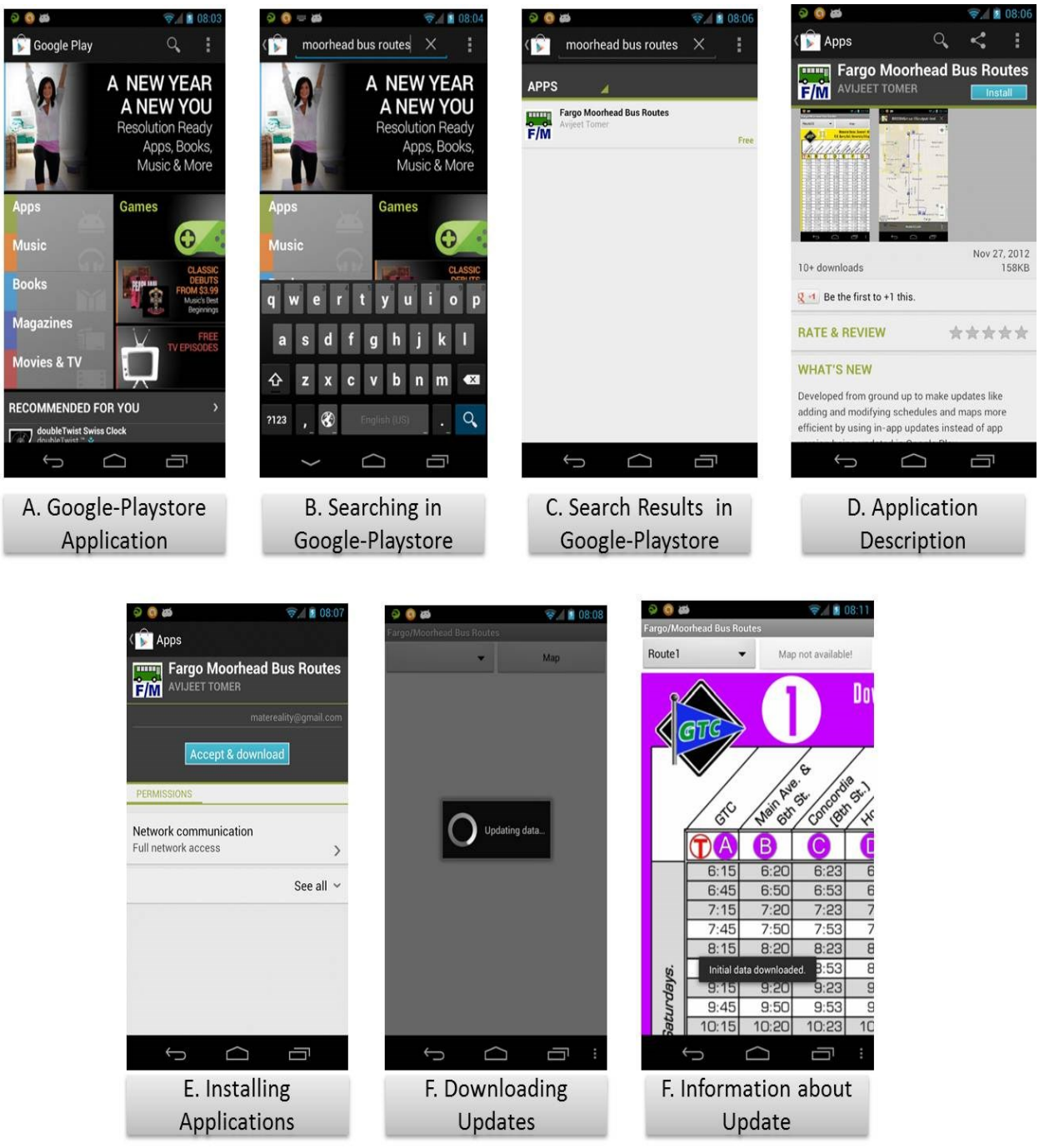

Figure 12. Installing and running the application

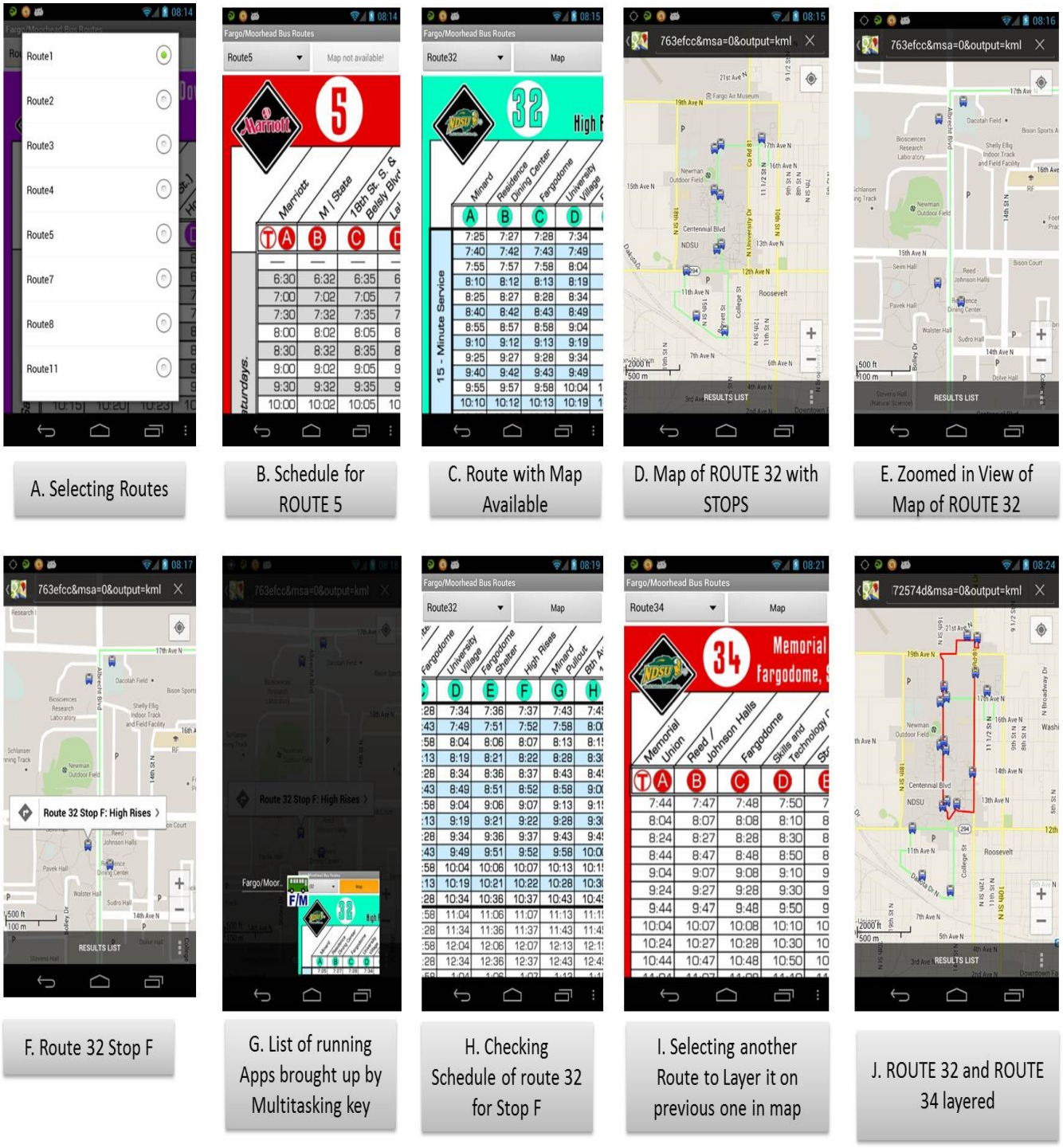

Figure 13. Viewing schedule of another route, its map and layering maps of two routes

The next time the user opens the application, if an internet access is available, the server would be contacted for any available updates to the data. If internet access is not available or if no updates are available, the application would continue using the initially downloaded data. Screenshots of the procedure are provided in alphabetical order in **Figure 14**.

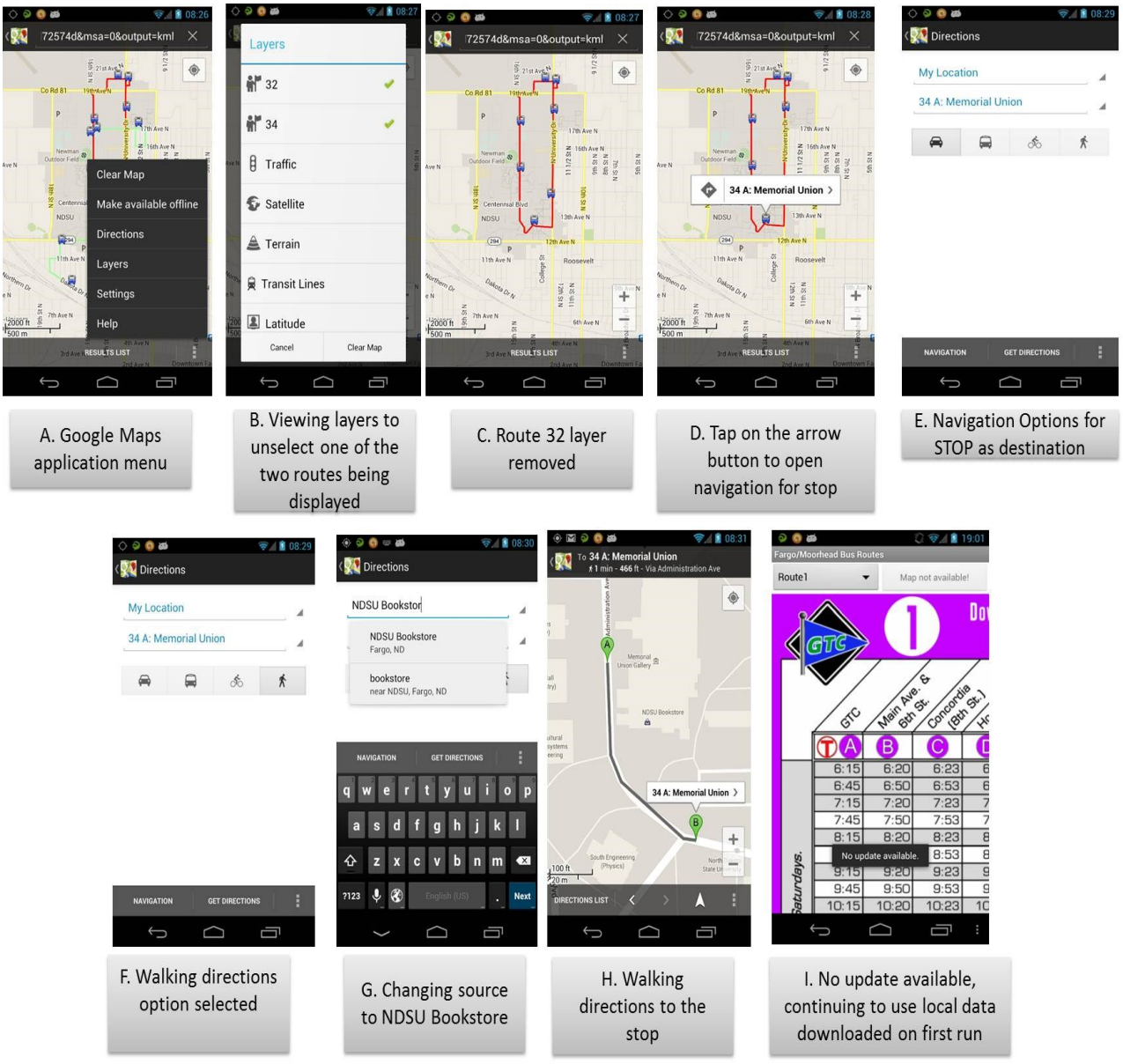

Figure 14. Switching layers of routes on or off and getting walking directions to a stop

5. TESTING

Google Maps application features useful to this application's utility like layering two bus routes and determining walking directions to a bus stop were not tested because it is a separate application that not developed in this project. Also, Android OS features like sending application in background on Android activity stack on pressing Home key were not tested, but testing if Fargo Moorhead Bus Routes application comes back into foreground correctly when user expects was part of the tests.

Table 1 lists the possible states of the application for black box state transition testing.

Table 1. States of the application

State	Description
S_1	Application installed, never run, internet access not available
S_2	Application installed, never run, internet access available, server down
S_3	Application installed, never run, internet access available, server available
S_4	Application displays dialog with message: "Updating data…", dialog closes, empty application screen shows message: "Server not reachable for initial data download, exit app and try later!"
S_5	Application displays dialog with message: "Updating data…", application downloads data from server, dialog closes, application displays schedule of bus route 1, shows message to user: "Initial data downloaded."
S_6	Application displays dialog with message: "Updating data…", dialog closes, application displays schedule of bus route 1 from local data, shows message to user: "No update available."

(continued)

Table 1. States of the application (continued)

State	Description
S_7	Application displays dialog with message: "Updating data…", application downloads update to data from server, dialog closes, application displays schedule of bus route 1, shows message to user with information about update downloaded (example: "Route 32 schedule updated, route 33 map added").
S_8	Application not running, internet access not available
S_9	Application not running, internet access available, server down
S_{10}	Application not running, internet access available, server available
S_{11}	Application displays dialog with message: "Updating data…", dialog closes, application displays schedule of bus route 1 from local data, shows message to user: "Server not reachable for update."
S_{12}	Application running, schedule of bus route 1 displayed
S_{13}	Application displaying schedule of bus route selected, map button disabled
S_{14}	Application displaying schedule of bus route selected, map button enabled, internet access not available
S_{15}	Application displaying schedule of bus route selected, map button enabled, internet access available
S_{16}	Application in background, Google Maps application attempts to search for map data but fails due to lack of internet access.

(continued)

Table 1. States of the application (continued)

State	Description
S_{17}	Application in background, Google Maps application displaying route of selected bus on map
S_{18}	Application in foreground visible to user displaying schedule of last selected bus route, Google Maps application in background

The application was functionally tested by subjecting it to events which changed its state from one to another. If the state expected matched the state it actually ended up in, the test was considered to have been passed. **Table 2** lists the tests and their results.

Table 2. State transition tests

Test Number	Initial State	Event	Expected State	Test Result
1	S_1	Run application.	S_4	Passed
2	S_2	Run application.	S_4	Passed
3	S_3	Run application.	S_5	Passed
4	S_8	Run application.	S_{11}	Passed
5	S_9	Run application.	S_{11}	Passed
6	S_{10}	Run application. Conditions: No update available on server.	S_6	Passed

(continued)

Table 2. State transition tests (continued)

Test Number	Initial State	Event	Expected State	Test Result
7	S_{10}	Run application. Conditions: Update available on server.	S_7	Passed
8	S_{12}	Another bus route selected. Conditions: Selected bus route does not have map data.	S_{13}	Passed
9	S_{13}	Another bus route selected. Conditions: Selected bus route has map data, internet access is not available.	S_{14}	Passed
10	S_{13}	Another bus route selected. Conditions: Selected bus route has map data, internet access is available.	S_{15}	Passed
11	S_{14}	Map button tapped.	S_{16}	Passed
12	S_{15}	Map button tapped.	S_{17}	Passed
13	S_{17}	Switch back to Fargo Moorhead Bus Routes application using multitasking key on new Android devices or long pressing Home key on old ones.	S_{18}	Passed

6. CONCLUSION

Objective 1: The user should be able to find his own location on a map or Fargo/Moorhead area.

Status: Completed.

Objective 2: The project should make it more convenient for user to view a map of Fargo/Moorhead area than having to use a big map outside in winter.

Status: Completed.

Objective 3: The project should make it more convenient for user to view the schedules of bus routes of Fargo/Moorhead area than having to use a big map outside in winter.

Status: Completed.

Objective 4: The project should make it more convenient for user to view the routes of buses on a map of Fargo/Moorhead area than having to use a big map outside in winter.

Status: Completed.

Objective 5: The user should be able to view one bus route at a time on the map.

Status: Completed.

Objective 6: The user should not have to wait for a significant amount of time for the map to be rendered by the smartphone.

Status: Completed.

Objective 7: The project should make it convenient for user to view the schedule of a bus route and its map more conveniently than the MATBUS website.

Status: Completed.

Fargo Moorhead Bus Routes application is a significant improvement over its previous version in terms of how it manages data and updates to that data, how much easier and faster it is for developer to update the data and the benefits provided by using Google Maps application to display maps of bus routes instead of embedded map view in the application.

Also, it does have scope for further improvement, like adding the times a bus stops on the description of stops on the map view.

6.1. Improvements over previous version

A project built on the default Android SDK project template has an XML file strings.xml, used for global string values, which by default includes values for the application name, title of main activity and labels of options available in the application when the Menu key is pressed. It is meant for storing string values like labels of controls in an application, or a value that is accessed at multiple places in the code. An advantage of this approach is that while testing with different string values during the development of the application, the code does not need to be recompiled and application redeployed (to either a test Android device connected to the development machine or a software emulator of an Android device). Only the application resources (like this strings.xml file) need to be reloaded, saving development time. Also, it helps clean separation of such data from the code. The SDK provides a wrapper class to directly access values in not only that file, but also other similar files like styles.xml for storing application skinning templates and XML files for user interface layouts of activities in an application.

In the previous version of Fargo Moorhead Bus Routes application, the strings.xml file was used to store schedules of buses. It was certainly a better coding practice than storing those values in variables in the code, but whether those values were in strings.xml or in code, there was one problem: updating that data (modifying an existing schedule or adding a new one) meant the

application package was changed from what was available on Google Play store (because that file is a part of the installation package of the application). So the application on Google Play store had to be updated, thereby changing its version. However, application should only change version when there are changes the application itself, not its data.

Along with storing schedules on the strings.xml file, the coordinates for drawing a map of a bus route were stored in a SQLite database file in the package. Again, updating the information in that database would mean having to update application package on Google Play store, changing the application version.

The new version solves this problem by not packaging the data with the application installation package on Google Play store, but just URLs to locations of the elements of data on a server. The data on server includes URLs to publicly shared maps of bus routes on my Google account. So if the data is updated (changes in schedules or maps, addition of schedules or maps) on the server, the application checks for the change when it starts and downloads the update to its local data (which it downloads on first run). So the application version does not get changed.

The procedure for modifying a bus schedule in the previous version of the application was not too time consuming, it just required changing appropriate text in the strings.xml file. Even if a small change was to be made, the process that consumed significant time was packaging the application after the change, signing it with my Android developer key, uploading it on Google Play store and updating necessary information about the application on Google Play publisher account. In the new version, this is even simpler. The schedules are displayed as images of the PDF files of schedules available on MATBUS website, so changing a schedule is as simple as replacing the old image with the new one on the server and adding the update information in Update.txt, a text file on the server that the application checks against when it

checks for data updates on startup. The application will download the new schedule and return a message to the user with information about the update to its data.

Changing the map of a route in the previous application was a very time consuming process. A new custom map was drawn out on Google Maps website, its KML file downloaded, the set of coordinates from the KML file added to the SQLite database file in the application package, the number of coordinate pairs (latitude and longitude) to expect for that bus route modified in code to match the new set, the code recompiled and application redeployed for testing the new map of route to ensure it is rendered in the map view exactly as it was drawn out. Followed by that was the process of updating the application package on Google Play store, as described above. While the process of drawing a custom map on Google Maps website remains unchanged in the new version, there is no other significant time consuming procedure following that. The URL for public sharing of that map is generated, modified in MapsUrls.txt, a text file containing URLs to maps on the server and again, update information added to Update.txt. Google Maps Android application renders it exactly as it is drawn on Google Maps website.

Using Google Maps application to display the routes of buses to the user instead of a map view embedded in the application provides the advantage of the features of Google Maps application. Two advantages that could find popular usage are: first, the user can layer two bus routes to see their points of intersection to determine where to change buses if required on his journey. Second, he can use walking navigation feature to get the directions and time required to walk down to the nearest bus stop, or use car navigation if he has a friend with a car to drop him off to a particular stop. In the future, as Google improves its Maps application, it might add more features like the weather situation in an area on the map, to help the user make decisions based on that information near a bus stop of his interest.

6.2. Future work

Areas of improvement in the application that require minor changes:

1. The bus stops shown on maps of the routes can show the times that bus stops on them. This would help the user avoid the need to switch back and forth between Google Maps and Fargo Moorhead Bus Routes applications.
2. At present, the color of the routes in maps of buses are based on the color used by MATBUS on the schedules of those buses, to help user differentiate two routes when he layers them. However, some routes have a bad visibility on the map because of their light color. One way around this problem is to not follow MATBUS color scheme and only use darker, more visible colors.

Areas of improvement that require major changes:

1. The application could inform user of the route or routes closest to his location and only if those routes are available at that time. This would require information like the coordinates of the bus stops on those routes and the times the buses arrive on those stops to be available in the data that application uses. Code would have to be updated to add this feature and the application permissions from the OS would include the use of device GPS and system time information.

Areas of improvement not in the application but for the benefit of users:

1. Since I do not live in Fargo, I may not put enough effort to keep data up to date. Also, the information I use for the data is from MATBUS website, which if not updated by them when they change schedules or routes of buses, cannot be available to me. A suggestion for the benefit of users would be that a group of NDSU students who use MATBUS server frequently take up the responsibility of maintaining the data.

7. REFERENCES

[1] Elgin, Ben. "Google Buys Android for Its Mobile Arsenal." Bloomberg Businessweek. Bloomberg, 16 Aug. 2005. Web. 21 Jan. 2013.

[2] Stallings, William. *Operating Systems: Internals and Design Principles*. Upper Saddle River, NJ: Prentice Hall, 2005. Print.

[3] Kuhn, Markus. "International Standard Paper Sizes." N.p., 29 Oct. 1996. Web. 21 Jan. 2013. <http://www.cl.cam.ac.uk/~mgk25/iso-paper.html>. Explains the ISO 216 paper size system.

[4] *PDF Reference: Adobe® Portable Document Format*. Tech. no. 1.7. Sixth ed. N.p.: Adobe Systems Incorporated, 2006. Print.

[5] Bryant, Randal, and David Richard. O'Hallaron. *Computer Systems: A Programmer's Perspective*. Upper Saddle River, NJ: Prentice Hall, 2003. Print.

[6] "Software Development Kit." *Wikipedia*. Wikimedia Foundation, 1 Jan. 2013. Web. 21 Jan. 2013. <http://en.wikipedia.org/wiki/Software_development_kit>.

[7] Wilson, Tim, ed. *OGC® KML*. Tech. no. OGC 07-147r2. 2.2.0 ed. N.p.: Open Geospatial Consortium, 2008. Print.

[8] Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau, eds. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Tech. W3C, 26 Nov. 2008. Web. 21 Jan. 2013. <http://www.w3.org/TR/REC-xml/>.

[9] Crockford, D. *The Application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627. Network Working Group, July 2006. Web. 21 Jan. 2013. <http://tools.ietf.org/html/rfc4627>.

[10] Deacon, John. *Model-View-Controller (MVC) Architecture*. Publication. N.p.: John Deacon Computer Systems Development, Consulting & Training, 1995. Print.

[11] "Google Maps Transit." *Google*. N.p., n.d. Web. 21 Feb. 2013. <http://www.google.com/intl/en/landing/transit/text.html>.

[12] Brabham, Daren C. "Crowdsourcing as a Model for Problem Solving." *Convergence: The International Journal of Research into New Media Technologies* 14 (2008): 75-90. Print.

[13] *Unified Modeling Language: Superstructure*. Tech. no. Formal/05-07-04. 2.0nd ed. N.p.: Object Management Group, 2005. Print.

[14] "Client–server Model." *Wikipedia*. Wikimedia Foundation, 20 Jan. 2013. Web. 21 Jan. 2013. <http://en.wikipedia.org/wiki/Client_server>.

[15] "Integrated Development Environment." *Wikipedia*. Wikimedia Foundation, 18 Jan. 2013. Web. 21 Jan. 2013. <http://en.wikipedia.org/wiki/Integrated_development_environment>.

[16] "Application Programming Interface." *Wikipedia*. Wikimedia Foundation, 16 Jan. 2013. Web. 21 Feb. 2013. <http://en.wikipedia.org/wiki/Application_programming_interface>.

[17] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Reading, MA: Addison-Wesley, 1995. Print.

[18] Patton, Ron. *Software Testing*. Indianapolis, IN: Sams Pub., 2006. Print.

[19] Harris, David Money., and Sarah L. Harris. *Digital Design and Computer Architecture*. Amsterdam: Morgan Kaufmann, 2007. Print.

[20] "Distinctive Features Of SQLite." *SQLite*. N.p., n.d. Web. 21 Jan. 2013. <http://www.sqlite.org/different.html>.

[21] "Tasks and Back Stack." *Android Developers*. N.p., n.d. Web. 21 Jan. 2013. <http://developer.android.com/guide/components/tasks-and-back-stack.html>.

[22] "MATBUS All Routes Map." *MATBUS*. N.p., n.d. Web. 21 Jan. 2013. <http://www.matbus.com/images/be%20moved.%20%20matbus.%20%20%281-2-2013%29/MATBUS-for-Web.pdf>.

[23] "MATBUS--Fargo, Moorhead and West Fargo." *MATBUS*. N.p., n.d. Web. 21 Jan. 2013. <http://www.matbus.com/>.

[24] Lin, Mathias. "Android Google Maps Import Are Not Recognized." *Stackoverflow*. N.p., 24 Sept. 2011. Web. 21 Jan. 2013. <http://stackoverflow.com/a/7540458>.

[25] "Google Maps." *Google*. N.p., n.d. Web. 21 Jan. 2013. <http://maps.google.com/>.

[26] "NYCMate (NYC Bus & Subway)." *Android Apps on Google Play*. N.p., n.d. Web. 21 Jan. 2013. <https://play.google.com/store/apps/details?id=com.densebrain.android.nycsubwaymap>.

[27] "Chicago Transit Tracker Lite." *Android Apps on Google Play*. N.p., n.d. Web. 21 Jan. 2013. <https://play.google.com/store/apps/details?id=com.jasonshah.ctatracker>.

[28] "MATBUS All Routes Map." *MATBUS*. N.p., n.d. Web. 21 Jan. 2013. <http://www.matbus.com/images/be%20moved.%20%20matbus.%20%20%281-2-2013%29/MATBUS-for-Web.jpg>.

[29] "Door-to-Door Transportation Services." *MATBUS*. N.p., n.d. Web. 21 Jan. 2013. <http://www.matbus.com/the new matbus.htm>.

[30] "Dashboards | Android Developers." *Android Developers*. N.p., n.d. Web. 21 Jan. 2013. <http://developer.android.com/about/dashboards/>.

[31] *Google Play*. N.p., n.d. Web. 21 Jan. 2013. <https://play.google.com/store>.