THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Metro Trip Planner

Final Report

CSCI 6221 – Spring 2015 Nicholas Capurso nickcapurso@gwmail.gwu.edu

Table of Contents

Introduction

Abstract

In big cities such as Washington D.C., New York, or Chicago, the use of the metro (or subway) for transportation is very common. In some cases, users may find it a cheaper or more convenient alternative to other types of mass transportation.

However, to tourists or those who have never used the subway, determining what trains to catch and what stations to switch lines at can be an intimidating task. Consider the New York Subway, which has over 400 stations and service lines that consist of a letter or number and a color (not to mention the direction that of a specific train, given by its final station). Stepping on the wrong train may potentially result in a large waste of time – a user would need to get off at the next stop and wait for a train heading back in the opposite direction. Thus, a welcome mechanism would simply tell the user what trains to take to get to their desired location.

The use of smartphones is now widespread in today's society. Perhaps the most widely known app to get directions is Google Maps. A user can see the route they need to take on a street map and can follow written (or spoken) directions to reach their destination.

Google Maps also supports mass transportation options, including the subway, but the initial set of directions displayed may not be given strictly in terms of subway lines (for example, "take the blue line to X and then take the bus to Y" instead of "take the blue line to X, transfer to the red line and take it to Y"). At this point, many users can't be bothered (or perhaps don't know how) to browse through alternate sets of directions or to return and refine their search parameters. Because a rider pays the metro fare when they ultimately exit the metro (regardless of how many trains they take) it is typically cheaper or more convenient to take the metro all the way to your destination, if possible.

Additionally, metro-specific apps exist, but they vary in functionality and may not provide a user's route on a map (a la Google Maps) which helps them determine where they are going. The goal of my project is to develop a simple, but intuitive app to aid visitors or newcomers in using the D.C. metro to get where they need to go. Users will be able to input a street address or location and the app will attempt to get the user as close as possible to that destination by solely using the metro – with the idea that they user can either walk or take another mode of transportation to their destination after stepping off the metro. Their route will be displayed on a map using the Google Maps (and Places) API and they will also receive textual directions on what trains to take, where to transfer, etc. The WMATA API (and JSON / Apache HTTP communication) will also be used to get information on the metro lines and stations. I also intend to implement the ability to display any service warnings / disruptions (and perhaps factor this into the trip planning) and keep a history of previous trips.

Problem Statement

Newcomers to the metro need a simple, but intuitive mobile application that tells them what trains to take and where to transfer at to ultimately arrive at a desired destination.

Objectives

The main objective of this project is the development of an Android application that allows users to input a desired destination and the specific Metro trains to take to get to that destination will be displayed. Secondary objectives are to allow the user to view service disruptions and keep a history of past planned trips.

Intended Demographic

The application will target those who may new to D.C.'s Metro, such as tourists or out-of-state college students. More experienced riders who have memorized the Metro map or their common routes may have no need for this kind of application.

Motivation

The motivation for this project stems from my own experiences being new to the subway in D.C. and New York. Although the subway system becomes easy to use once you understand it, it can be initially confusing or intimidating. I have also tried out various mobile metro apps, but haven't found one that does everything I'd like – for example, being able to input addresses instead of just choosing and starting/ending metro stations.

Requirements

Requirements will be preceded by a code in the form “FR X” (functional requirement) or “NFR X” (non-functional requirement), where X is the number of the requirement. This list will be expanded upon if time allows.

Functional Requirements

FR 1

The application will take as input a desired destination address or location.

FR 2

The application will determine what trains to take and/or stations to switch trains at to ultimately arrive at/near the inputted destination.

FR 3

The application will display a user’s route (given by FR 2) on Google Maps and also give these directions in text.

FR 4

The application will display any service warnings or disruptions (for example, train delays).

FR 5

The application will keep a history of past trips.

FR 6

The application will allow the user to view an image of the metro map.

FR 7

The application will alert the user to any network problems, in the form of time-outs.

Non-Functional Requirements

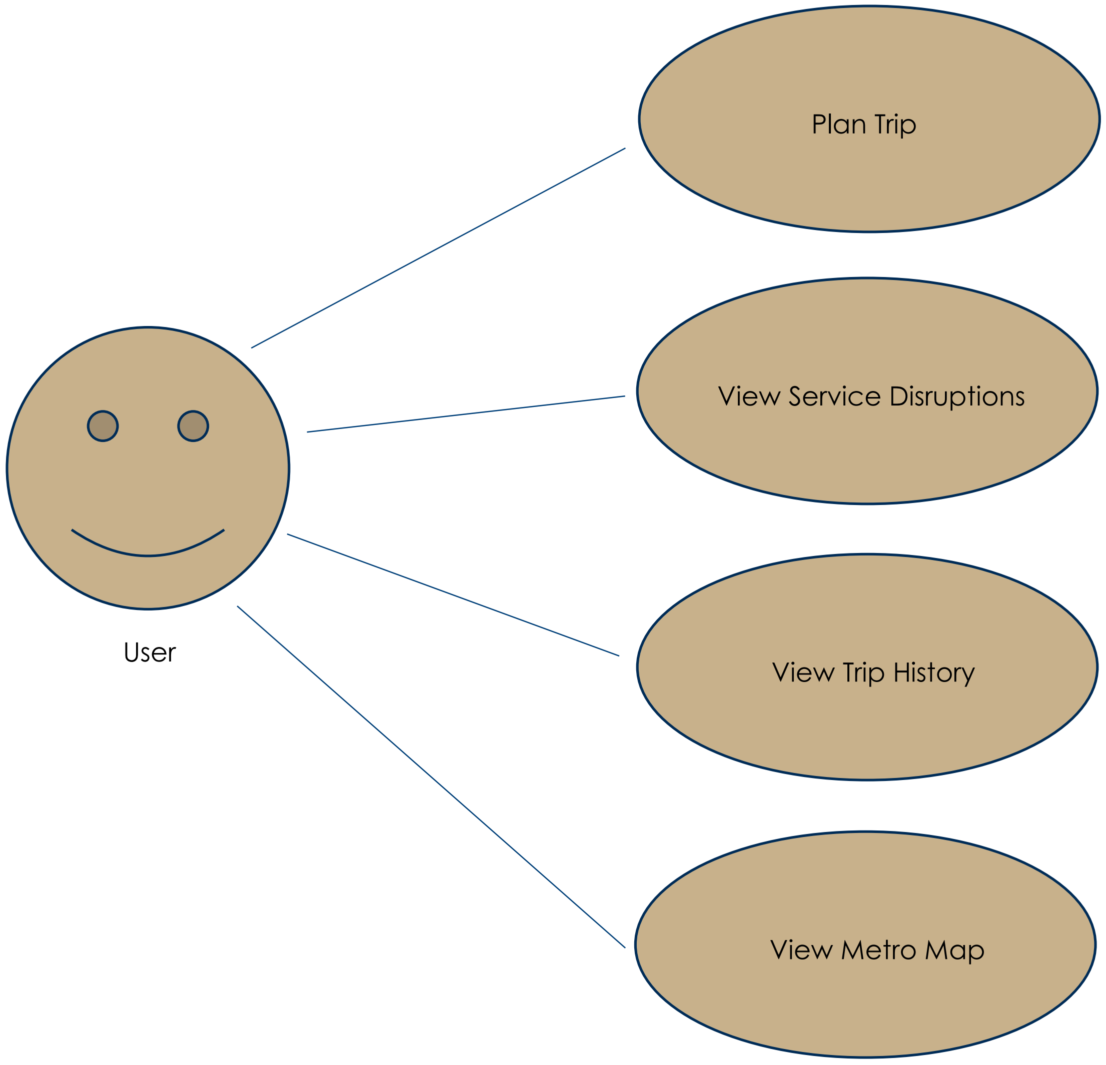
NFR 1

The time it takes to calculate a user’s Metro route to their destination (i.e. the waiting time) will be kept reasonable (under 10 seconds).

NFR 2

The external storage space occupied by the trip history (FR 5) or Metro map (FR 6) image will be kept at a reasonable amount (total is likely under 5 MB).

Use-Case Diagram



Implementation Details

Hardware Details

Development and testing was done using two different Android smartphones: an HTC One M8 running Android version 4.4.4 (KitKat) and a LG Nexus 5, which initially ran 4.4 and was upgraded to 4.4.4. To make network requests, the One M8 had access to Wi-Fi and cellular data (4G), while the Nexus 5 only had Wi-Fi (no cellular data plan). Both devices had a similar 4-core CPU model (Snapdragon 800 vs. Snapdragon 801).

Also relevant, both devices had different resolution screens so it could be seen how the mobile application was displayed at different resolutions.

Software Details

The application was targeted at Android devices running a minimum version of 4.1 (Jelly Bean) and was developed using Android Studio 1.0.

External API Details

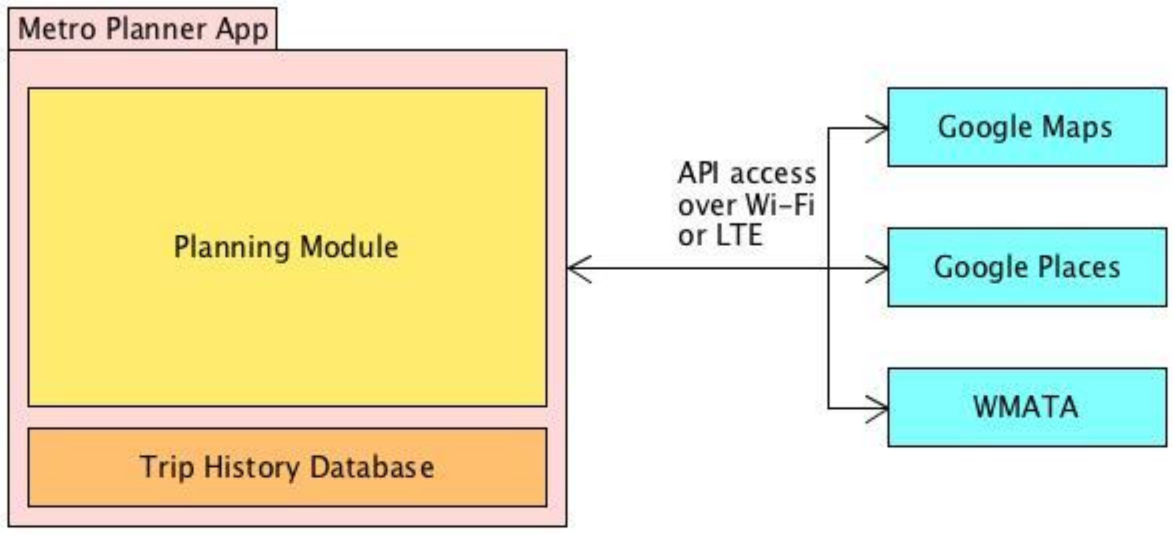
The following external APIs / libraries were used to create the application:

- Apache HTTP Components for Android – used to create, execute, and receive HTTP messages. In this case, these messages were used to request information from the other APIs (Google Places and WMATA).
- Google Maps API – used to receive and draw information on a scrollable, zoomable map.
- Google Places API – used for address resolution. Translated user-inputted addresses or locations (ex. "White House") into street addresses and longitude/latitude coordinates. This information was used to feed information to the WMATA API.
- WMATA API – used to get information about metro stations and lines – information ultimately used to determine a path from start to end according to the user's input.

It should be noted that the Google Places and the WMATA APIs have a quota on the number of requests made per day as well as during short periods of time. The Google Places' quota is not much of a concern, but the WMATA's quota of requests-per-second had to be controlled due to the number of WMATA API calls that are made during the trip-planning state (refer to the later state diagrams).

System Diagrams

System Architecture Diagram



System Architecture Components

Metro Planner App

The actual application running on an Android phone. Aside from the UI components (shown in a later state diagram), the two major components are the Planning Module and the Trip History Database. The Planning Module is responsible for taking the user's intended destination and determining what Metro trains they should take. This requires multiple API requests to Google Places (for determining the correct addresses and longitude/latitude coordinates) and WMATA (to determine nearby Metro stations, what lines service what stations, etc.). Once the route is determined, it will be displayed on Google Maps. The Trip History Database simply contains a record of past Metro trips that can be recalled and the directions displayed.

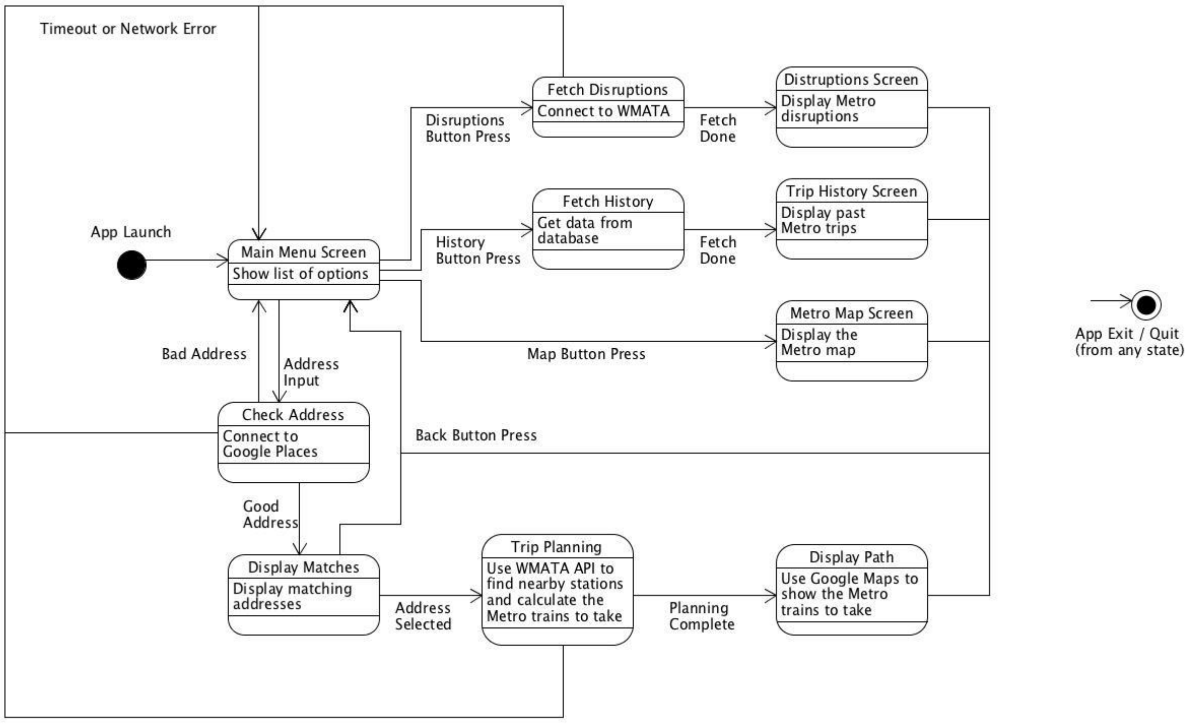
Trip History Database Table Entries

An entry in the trip history database has the following 8 fields:

- A unique ID (Android-generated)
- The starting location (longitude, latitude, name)
- The ending location (longitude, latitude, name)
- The date the trip was made on

<u>ID</u>	Start Lng	Start Lat	Start Name	End Lng	End Lat	End Name	Date

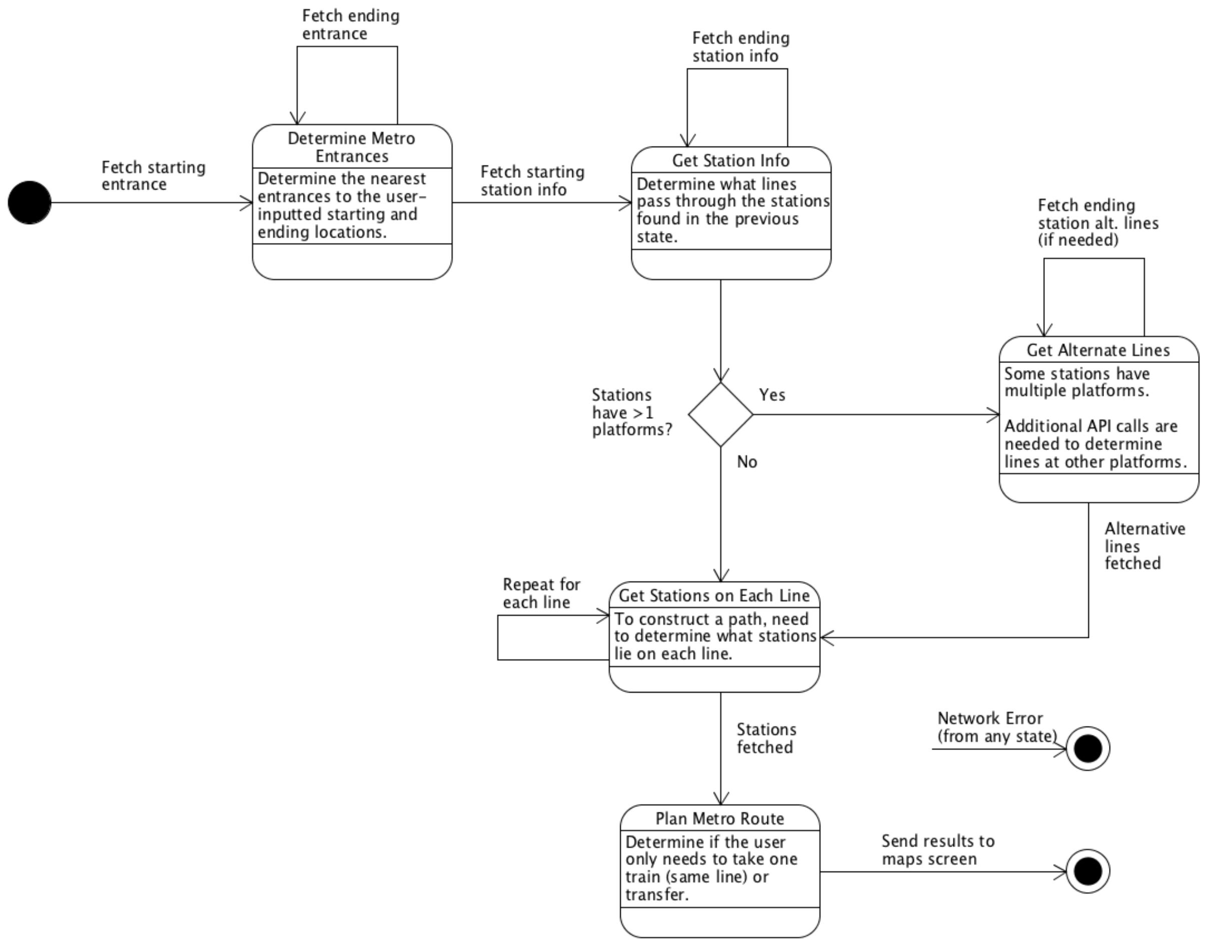
Overall State Diagram



This diagram shows the interactions between the various states of the application (including the interactions between the various "screens" that comprise it). In general, when on a specific screen, the user can press the "Back" button to return to the main menu screen. Additionally, any sort of network error when performing a networked API call will return the user to the main menu screen (with an error message displayed). For simplicity, arrows aren't drawn from every state to the final state (user closes the app).

The "Trip Planning" state is quite involved and has its own state diagram on the next page.

Trip Planning State Diagram


Fetch starting entrance
Determine Metro Entrances
Determine the nearest entrances to the user-inputted starting and ending locations.
Fetch ending entrance
Fetch starting station info
Get Station Info
Determine what lines pass through the stations found in the previous state.
Fetch ending station info
Stations have >1 platforms?
Yes
No
Fetch ending station alt. lines (if needed)
Get Alternate Lines
Some stations have multiple platforms.
Additional API calls are needed to determine lines at other platforms.
Alternative lines fetched
Repeat for each line
Get Stations on Each Line
To construct a path, need to determine what stations lie on each line.
Stations fetched
Network Error (from any state)
Plan Metro Route
Determine if the user only needs to take one train (same line) or transfer.
Send results to maps screen

This is the diagram for the Trip Planning state shown on the previous page. Its functionality is implemented in the Planning Module (refer to the architecture diagram). Working with the WMATA API causes operations to be done in a specific sequence before the path between the starting and ending locations can be determined. The states are described in a table on the next page.

Trip Planning State Descriptions

State	Description
Determine Metro Entrances	After entering in a desired destination address/location and an (optional) starting address/location, the Google Places API is called to perform address lookup and return GPS coordinates. WMATA provides an API that returns nearby stations given GPS coordinates and a search radius. This is done for both the starting and ending addresses.
Get Station Info	Upon completion of the previous state, the closest stations to both ends are chosen. However, it is not known yet what lines pass through these stations. This requires two more API calls to get this information (one for the starting station and one for the ending station).
Get Alternate Lines	The WMATA API separates different platforms of the same station as separate stations (same name, but different WMATA station codes). Thus, stations that have multiple platforms (ex. L'Enfant Plaza) require additional API calls to determine which lines pass through these platforms.
Get Stations on Each Line	By this point, we know all the lines that pass through the starting and ending stations. However, it is not known which stations lie on these lines – this information is needed determine a path between stations that don't share lines (ex. by finding the intersection). It is also needed to draw a more concise path on Google Maps.
Plan Metro Route	With all the required information, a route between the starting and ending stations can now be made. If the two stations share a line, the process is done. Otherwise, the intersection between the two's lines is used as the "transfer station". The resulting metro path is sent back to the Google Maps screen for displaying to the user.

Test Cases

The following table presents various tests performed on the application, in a general ordering of descending importance. Note: not every test conducted is listed in the table, but rather, the more noteworthy ones are.

Test Type	Description	Result
Path Planning	Starting and ending locations are on the same line.	Map displays the single path and also allows the user to view any shared lines (ex. "Take orange, blue, or silver to X")
Path Planning	Starting and ending locations are on different lines.	Map displays two "legs" of the trip, with the transfer station shown and any shared lines on either leg.
Path Planning	Starting and ending locations are on the same line, but can be reached by transferring intermediately and taking a different line.	App prioritizes direct paths and doesn't consider "shorter" paths by transferring early (perhaps a consideration for the future). For a newcomer, it is less stressful for them to stay on the same line.
Path Planning	Either location is unreachable by the D.C. metro (ex. user inputs "Empire State Building")	After returning from the "Determine Metro Entrances" state, the app alerts the user that a metro entrance cannot be found.
Path Planning	Determine that the calculated path is accurate/reasonable.	Compared the path with a map of the metro to determine that the paths are correct.
Reliability	Performance of the app in places with unreliable network connections (ex. down in the metro itself)	App "waits" on a network request for a specified amount of time before alerting the user to a network timeout and cancels the current action.
Reliability	User attempts to use the "Quick Plan" option (uses GPS for the starting location) in an area where GPS is not available.	Like the network test, the app waits for a location for a specified amount of time, after which the app informs the user that a location fix could not be determined.
Reliability	User leaves the app while it is planning the trip.	App continues to operate "in the background" and does not restart when the user returns to the app.
Input Validation	User enters nothing or nonsense into the address/location fields.	The app will either tell the user that nothing has been inputted, or that the input cannot be matched to a street address (results returned from Google Places are null).
Reliability	User attempts to clear the trip history when it is empty.	The database is simply deleted and recreated again – no change.

Schedule and Distribution of Tasks

Proposed Timeline

Week of (Monday)	Task
March 2nd	**Learn APIs (Google, WMATA)**
March 9th	**Build app UI**
March 16th	**Get and convert user input into addresses and GPS coordinates (geocoding) with Google Places API**
March 23rd	**Determine route from starting address to ending address using WMATA API**
March 30th	**Continued route planning and displaying results on Google Maps**
April 6th	**Implement trip history database and recalling past trips / routes**
April 13th	**Provide additional details about a trip (next train arrivals, delays, possible alternatives)**
April 20th	**Any extra features, testing, and finalizing reports**
April 27th	**Presentation and submission**

Timeline Changes

Due to other classes, duties, and certain areas of the project taking longer than expected, one of the planned tasks had to be cut. The least vital task was the one planned for the week of April 13th, since it was only providing extra details about the trip. Thus, it was partly cut (alternative paths are still given).

Tasks Distribution

As this is an individual term project, I completed all of the work myself.

References

Apache HTTP Components for Android:
https://hc.apache.org/httpcomponents-client-4.3.x/android-port.html

Google Maps API:
https://developers.google.com/maps/

Google Places API:
https://developers.google.com/places/documentation/

Design Document References:
http://www.atilim.edu.tr/~dmishra/se112/sdd_template.pdf
https://www.oasis-open.org/committees/download.php/24846/Example-SoftwareDesignDocument-LegalXMLUtility.pdf
http://en.wikipedia.org/wiki/Software_design_document

Microsoft Word (template):
https://products.office.com/en-us/word

Requirements Document References:
http://www.cdl.edu/cdl_resources/writing-requirements
http://csis.pace.edu/~marchese/CS775/Requirements%20Specification%20Template.doc
http://en.wikipedia.org/wiki/Product_requirements_document

UMLet (for diagrams):
http://www.umlet.com/

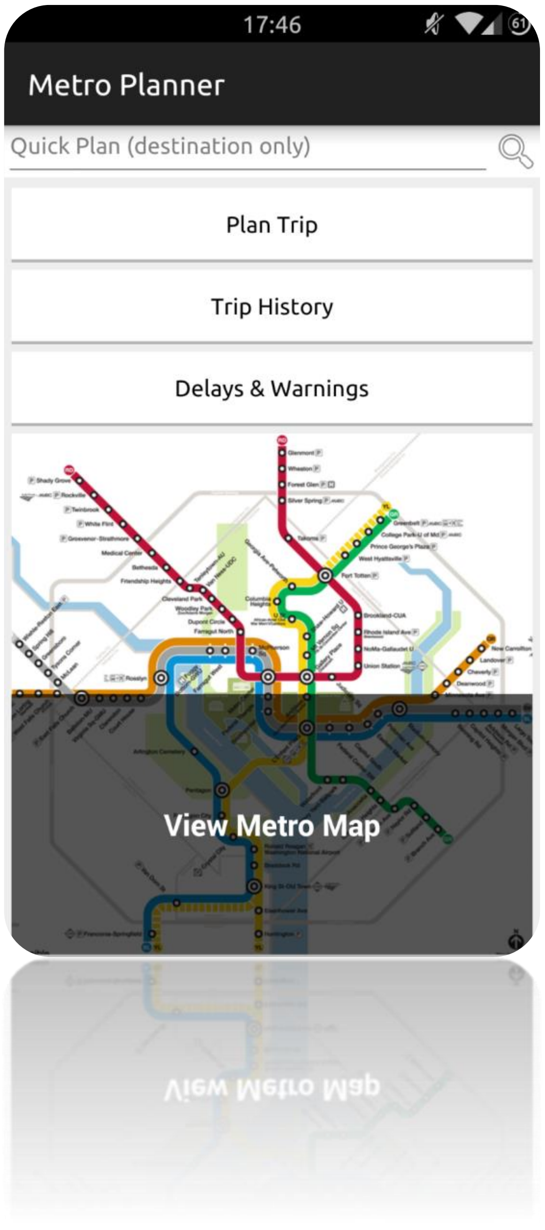
WMATA API:
https://developer.wmata.com/

Appendix A – User Manual

The following is a description of each screen of the Metro Planner application and its functionality. Note: for all screens, the phone's Back button is used to navigate backwards to previous screens (or to cancel certain dialog boxes).

Home Screen

This is the first screen the user sees when they open the Metro Planner applications.



There are five buttons on the screen:

Quick Plan (magnifying glass icon) – allows the user to enter a destination address or location into the adjacent text field. A metro path will be planned between the user's current location (obtained via GPS) and the inputted destination.

Plan Trip – similar to the previous, except the user is prompted for both a starting and ending location.

Trip History – the user is taken to the Trip History screen where they can view and recall past trips.

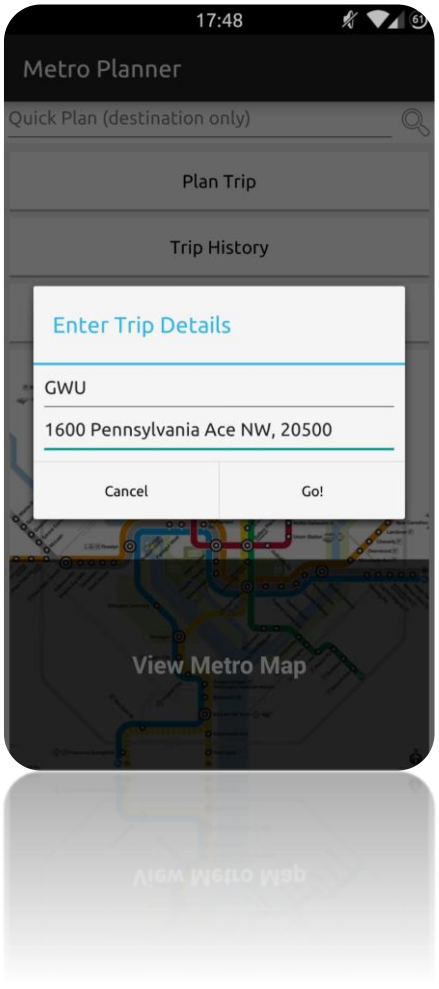
Delays & Warnings – the user is taken to the Delays & Warnings screen where they can view metro service disruptions

View Metro Map – opens the users browser to an image of the metro map which can be panned and zoomed.

Using the Quick Plan or the Plan Trip Option

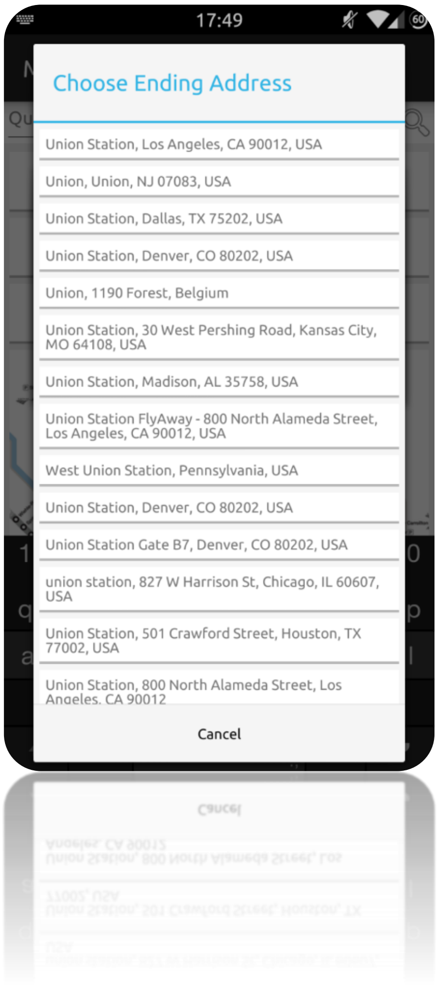
Upon entering an address or location into either the Quick Plan text box or the Plan Trip dialog (shown below), the user will be shown a list of possible matches to choose from (obtained from Google Places). The Quick Plan option will only prompt for the destination, while the Plan Trip option will prompt first for the starting location and then second for the destination.

If they do not see their desired location in the list, their original input may have to be refined (for example, by including the city or zip).



GWU

1600 Pennsylvania Ace NW, 20500



Union Station, Los Angeles, CA 90012, USA

Union, Union, NJ 07083, USA

Union Station, Dallas, TX 75202, USA

Union Station, Denver, CO 80202, USA

Union, 1190 Forest, Belgium

Union Station, 30 West Pershing Road, Kansas City, MO 64108, USA

Union Station, Madison, AL 35758, USA

Union Station FlyAway - 800 North Alameda Street, Los Angeles, CA 90012, USA

West Union Station, Pennsylvania, USA

Union Station, Denver, CO 80202, USA

Union Station Gate B7, Denver, CO 80202, USA

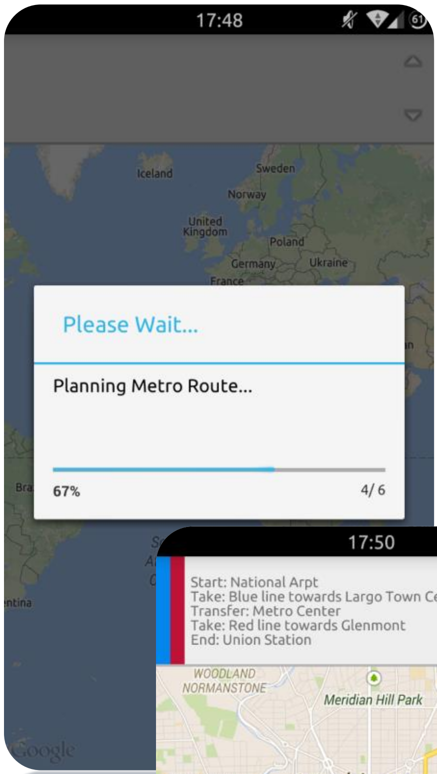
union station, 827 W Harrison St, Chicago, IL 60607, USA

Union Station, 501 Crawford Street, Houston, TX 77002, USA

Union Station, 800 North Alameda Street, Los Angeles, CA 90012

Google Maps Screen

The user is brought to this screen while their metro trip is being planned (during which a loading dialog is shown). After their trip has been planned, the metro path is drawn on the map and textual directions are shown at the top.



17:48

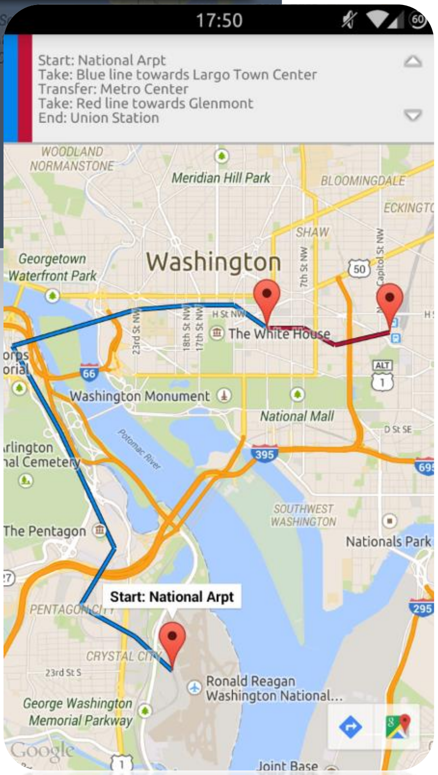
Please Wait...

Planning Metro Route...

67% 4/ 6

As shown below, the textual directions are laid out as follows:

• Two colored bars represent the lines the user must take to reach their destination. In the case that the starting and ending locations can be reached by only one line (i.e. the stations are on the same line), only one bar is shown.
• The user is given in text what station to start at, which direction to take the train in (given by the final station), and the ending station. In the case that the user has to transfer, the transfer station and second line is shown.
• Finally, the user can use the arrow buttons to browse the alternative lines they can take to reach their destination.



17:50

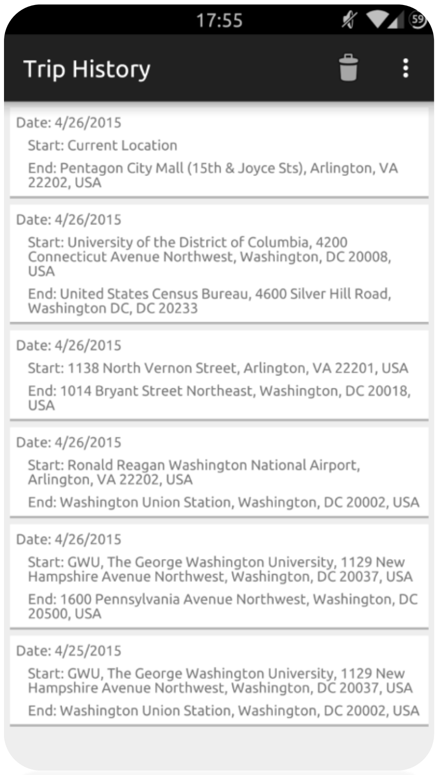
Start: National Arpt
Take: Blue line towards Largo Town Center
Transfer: Metro Center
Take: Red line towards Glenmont
End: Union Station

Start: National Arpt

Additionally, map markers are placed on the starting station, the ending station, and the transfer station (if one is needed). The user can click on these to display the name of the station at that point (as shown to the left with the National Airport station).

Trip History Screen

Each time the user plans a trip, it is saved to a SQL database. The user can later view the previous trips and select one to re-plan (i.e. it is displayed again on the Google Maps Screen).



Date: 4/26/2015
Start: Current Location
End: Pentagon City Mall (15th & Joyce Sts), Arlington, VA 22202, USA

Date: 4/26/2015
Start: University of the District of Columbia, 4200 Connecticut Avenue Northwest, Washington, DC 20008, USA
End: United States Census Bureau, 4600 Silver Hill Road, Washington DC, DC 20233

Date: 4/26/2015
Start: 1138 North Vernon Street, Arlington, VA 22201, USA
End: 1014 Bryant Street Northeast, Washington, DC 20018, USA

Date: 4/26/2015
Start: Ronald Reagan Washington National Airport, Arlington, VA 22202, USA
End: Washington Union Station, Washington, DC 20002, USA

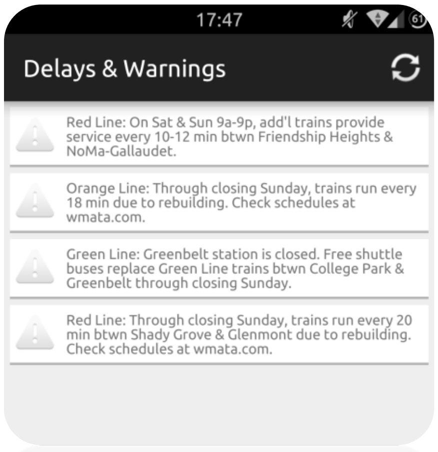
Date: 4/26/2015
Start: GWU, The George Washington University, 1129 New Hampshire Avenue Northwest, Washington, DC 20037, USA
End: 1600 Pennsylvania Avenue Northwest, Washington, DC 20500, USA

Date: 4/25/2015
Start: GWU, The George Washington University, 1129 New Hampshire Avenue Northwest, Washington, DC 20037, USA
End: Washington Union Station, Washington, DC 20002, USA

The results are shown in the format: date, starting location, ending location. For trips that were planned using the Quick Plan option, the starting location is listed as "Current Location" and will use the user's current location (invoking the GPS) to re-plan the trip if the user selects one.

There are two buttons on the top action bar. The trash can button simply clears the user's trip history. The other button opens a small menu with a checkbox labeled "Save Recalled History Items." If this option is checked, then anytime a user selects a history item to re-plan, it will also be copied to the top of the history.

Delays & Warnings Screen



17:47

Delays & Warnings

Red Line: On Sat & Sun 9a-9p, add'l trains provide service every 10-12 min btwn Friendship Heights & NoMa-Gallaudet.

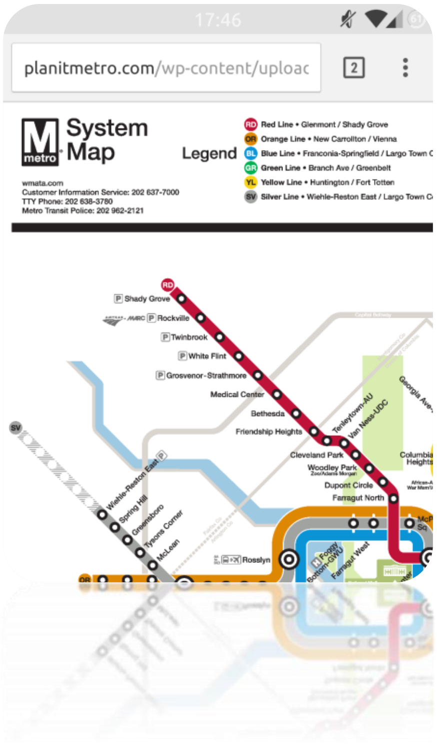
Orange Line: Through closing Sunday, trains run every 18 min due to rebuilding. Check schedules at wmata.com.

Green Line: Greenbelt station is closed. Free shuttle buses replace Green Line trains btwn College Park & Greenbelt through closing Sunday.

Red Line: Through closing Sunday, trains run every 20 min btwn Shady Grove & Glenmont due to rebuilding. Check schedules at wmata.com.

This screen displays the current list of metro services disruptions (also known as “Alerts” on the WMATA website). There is a single button on the top action bar to refresh the list.

Metro Map Screen



Allows the user to view a map of the metro in their browser app. The map is large enough that it can be panned and zoomed-in.