Mobile Interface For Mobility Incentives Schemes: FMS-Advisor

by

Jamar Brooks

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Masters of Engineering in Computer Science and Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.

Signature redacted

Author ..... \( \vdash \) \( \vdash \) \( \vdash \) \( \vdash \), \( \vdash \), \( \vdash \) \( \vdash \)
Department of Electrical Engineering and Computer Science
May 26, 2017

Signature redacted

Certified by \( \vdash \) \( \vdash \), \( \vdash \), \( \vdash \)
Moshe Ben-Akiva
Professor
Thesis Supervisor

Signature redacted

Accepted by \( \vdash \) \( \vdash \), \( \vdash \), \( \vdash \)
Christopher J. Terman
Chairman, Department Committee on Graduate Theses
DISCLAIMER NOTICE

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available.

Thank you.

The images contained in this document are of the best quality available.
Mobile Interface For Mobility Incentives Schemes:

FMS-Advisor

by

Jamar Brooks

Submitted to the Department of Electrical Engineering and Computer Science on May 26, 2017, in partial fulfillment of the requirements for the degree of Masters of Engineering in Computer Science and Engineering

Abstract

Reducing energy consumption is a common research topic in the United States because of its importance in environmental, health, and industrial fields. In this project, we aim to reduce global energy consumption for the Greater Boston Area by encouraging users to make more energy-efficient travel plans. We believe that if users can be provided with energy-efficient routes, alongside differing mode options and departure time, and provided the proper incentive in the form of rewards, they will be willing to modify their plans in order to obtain them. Therefore, we have created an interface on Android where users can plans trips, view past trips, and earn rewards.

Thesis Supervisor: Moshe Ben-Akiva
Title: Professor
Acknowledgments

I would like to thank my friends and family for their support along the way. I would like to thank Carlos Lima Azevedo, Bilge Atasoy, Xiang Song, Fang Zhao, and Ajinkya Ghorpade for their support in defining my project as well as technical help, and of course, Professor Moshe Ben-Akiva.
1 Introduction 13

2 Related Work 17

3 Thesis Work 19
   3.1 Trip Planner 19
      3.1.1 Design 19
      3.1.2 Implementation 23
   3.2 Trip Preview and Navigation 26
      3.2.1 Design 26
      3.2.2 Implementation 27
      3.2.3 Issues 30
   3.3 Trip History 31
      3.3.1 Design 31
      3.3.2 Implementation 33
   3.4 Rewards Marketplace 35
      3.4.1 Design Evaluations 35
      3.4.2 Implementation 37
      3.4.3 Database 38

4 Contribution Overview 41

5 Next Steps 43
6 Conclusion

A Figures
List of Figures

3-1 Original Trip Planner .............................................. 23
3-2 Trip Planner Input ................................................. 24
3-3 Screenshot Bunched Trip Planner ................................. 24
3-4 Trip Planner Modes ................................................. 25
3-5 Screenshot Bunched Trip Planner ................................. 28
3-6 Screenshot Bunched Trip Planner ................................. 28
3-7 Screenshot Bunched Trip Planner ................................. 29
3-8 Screenshot Bunched Trip Planner ................................. 29
3-9 Trip Navigation ..................................................... 30
3-10 First Trip History .................................................. 33
3-11 Fancy Trip History .................................................. 34
3-12 Marketplace Menus .................................................. 37
4-1 The expected flows through the app that we expect users to take. ... 42
A-1 Amazon Android search results list. ............................. 48
A-2 Amazon Android individual results page (top half) .......... 49
A-3 Amazon Android individual results page (bottom half) ....... 50
A-4 Google Play Store Android home page. ......................... 51
A-5 Microsoft Rewards Marketplace web page home. .............. 52
A-6 Microsoft Rewards Marketplace individual result page. ....... 53
A-7 WHAFF Rewards Android app home page. ....................... 54
A-8 WHAFF Rewards Android app home page with item selected. ... 55
A-9 Windows Store Windows App home page. ....................... 56
List of Tables

3.1 Preference Parameters ........................................ 39
3.2 Alternative Attributes ....................................... 40
Chapter 1

Introduction

Transportation is one of the largest consumers of energy in the United States. All different forms of transportation consumed approximately 28 quadrillion BTUs of energy in 2016, more than the residential or commercial. It was surpassed only by the industrial [1]. Therefore, by focusing on reducing overall energy consumption with the Boston traffic network, we hope to make a significant reduction in the energy consumption in the city and prove a concept that can be extended to others. By studying past traffic data, we have seen the amount of energy consumed can be reduced by adjusting a variety of parameters, including mode of transportation, route, speed, traffic, time of day, etc.

Our project has been named Tripod, more formally known as Sustainable Travel Incentives with Prediction, Optimization and Personalization, for its three legs: 1) Prediction 2) Personalization 3) Optimization. We feel that leveraging these three pillars we can encourage globally optimal energy consumption through transportation. In order to get data to determine the current prediction, we build upon a system, Future Mobility Sensing, which gathers data about an individual’s transportation patterns and uses machine learning to study that data [2][8].

Building upon the FMS system, FMS Advisor will not only gather data, but will also recommend more energy-efficient transportation alternatives, potentially by recommending a different route, mode of transportation, or departure time. Currently, these routing options are retrieved from two sources, DynaMIT from the ITS
Lab and the Google Directions API. The ITS Lab’s DynaMIT "is a next generation multi-modal traffic state prediction and network control platform. The system also includes a strategy optimization module that optimizes network control strategies in real time for congestion mitigation and other objectives." [6]. Currently our project uses Dynamit to provide routes, network link details like speed, distance, and time, and calculate token rewards. In the future all routes will come from the DynaMIT, but to allow for simultaneous development, we rely on the Google API for modes of transit that have not yet been implemented by DynaMIT. We provide these options in a list format allowing the user to see details about the route such as the estimated distance, duration, and energy consumption. Each route can then be previewed fully and selected for step-by-step navigation.

In order to encourage user adoption, the FMS Advisor app will use extrinsic motivation in the form of goods and services as incentives based on estimated travel energy savings to convince users to consistently use the app. Reward tokens are given to users for successfully completing their chosen trips, and the number of tokens for a trip will be determined on a global scale so that energy savings are globally optimal. These tokens can be redeemed in exchange for gift cards, discounts, coupons, etc. from partners we will recruit during the later stages of the TRIPOD project.

For this project, I have (1) built a user-preferences database, (2) revised the Android-based FMS Dashboard to integrate a personalized trip, energy and rewards summary (3) implemented trip display and navigation and (4) designed and implemented a first version of the personalized Android-based marketplace, where rewards can be exchanged for goods and services. The new Dashboard will contain different daily, weekly, and monthly summaries of user travel, energy and rewards performance. The FMS-Advisor rewards are in the form of tokens that can be accumulated and exchanged for goods and services in a new marketplace to be developed. The main goal of this project is to develop the first versions of this extension using User Preference Data (UPD) and User History Data (UHD) to decide which and how information should be provided to users without overwhelming the user and to increase the FMS-Advisor usage, therefore indirectly contributing for energy-efficiency (on the premise
of app effectiveness). Therefore, new design elements, with options informed by user specific data, would seek to get users to use the app more frequently. For example, we want to draw attention to our most exciting features, make sure all the elements are easy to use, show users a marketplace with many desirable rewards, and show them how following these routes will earn them these rewards. We also aimed for all elements to have as much utility as possible while simultaneously reducing clutter.
Chapter 2

Related Work

A lot of research has been done over the past two decades related to gamification as the use of digital, game-like interfaces became more commonplace. Deterding et. al. expands on the concept of "gaming" by highlighting that gaming also includes a sense of "playing structured by rules and competitive strife toward goals." Gamification instead is more about adding "artifactual elements conceived more in terms of affording gameful interpretations and enactments, rather than being gameful." [3]

This project seeks to keep to these guidelines by adding game design elements to keep users engaged. There are no opponents, no definition of winning or losing, minimal social interaction, and little to no dimensions of freedom for the user to play or explore. However, the additions to the current interface merely add personalized design aspects of information provision in terms of rewards, marketplace, and pointers to them to encourage continued user engagement of the system as a tool to optimize travel energy savings. This user experience will hopefully be a catalyst for users to revisit the app or website more consistently without requiring those notifications.

This project builds upon previous work described in Zhao et. al. [8] and Cottrill et. al. [2]. Both of these works describe FMS systems three major components: a smartphone app, a back-end server, and a web interface. As in [3], a primary goal of this project will be to provide adequate information without overwhelming the user. In addition, this project aims to add design elements that entice the user to spend more time considering optimal travel alternatives than they otherwise would.
Also, we aim to use a rewards-based marketplace to motivate users to continue earning rewards by using the app. Motivation is demonstrated by an individual's choice to engage in an activity and the intensity of effort or persistence in that activity. [5] In our marketplace, we focus on extrinsic motivation, encouraging users to earn more tokens in order to redeem for more rewards. Though the literature is currently split on the effectiveness of this kind of motivation, we believe the tokens will be successful not only because of their redeemable value, but also as a tangible conversion for the environmental benefits of using TRIPOD.

In this way, the app will seek to influence user travel decision. In contrast to previous iterations focused on rich interactions with zoomable maps and editable lists showing historical data, this iteration of FMS Adviser will focus more on design elements that are aesthetically attractive and minimally interactive. This forward-looking, goal-oriented interface will combat the need for explicit reminders to conduct further interaction while also remaining non-intrusive. [2]
Chapter 3

Thesis Work

When deciding on the budget widgets, and layouts of the updated interface, the key focuses were on utility, usability, and clarity. In order to have a full user experience with this kind of system, the app must support many uses and functionalities. However, if the design doesn’t ensure that each of these feature is clearly visible and easy to use, the utility loses value. The designs must keep all of these ideals in mind while producing a consistent experience across the different features of the app that are being designed.

My project consisted of four major features where I contributed to the design or redesign of their experiences. More specifically, I worked on the recreation and breakdown of the trip planner/trip selection screen, created an activity to allow users to preview a trip and have subsequent step-by-step navigation, travel log/trip history page, and finally developed the framework of a rewards marketplace where the tokens earned during the use of the app can be redeemed for goods and services.

3.1 Trip Planner

3.1.1 Design

With the trip planner, we began with a design very similar to earlier versions of the Google Maps interface. The planner form at the top of the interface in Figure 3-4
contains input fields for start and end locations, preferred mode of transportation, preferred start time, and options to avoid tolls and avoid highways. This hope was that the symmetry between the interface would communicate the similar purpose and make it intuitive to use. Below the form is a menu of possible trips that appear after you click Directions to Location. Each item in the menu shows the primary mode of transportation along with metadata such as the distance, duration, and the number of tokens to be earn by completing that specific trip.

Even though this design is simple and understandable, it had a few problems we needed to resolve. First of all, our current backend does not have parameters for avoiding highways and tolls, so including those fields would only confuse and frustrate users. We also added the option to sort the search results by distance, duration, and number of tokens received. Because of the large number of results returned (often three or more for each mode), looking through all of them trying to find the result which will give you the most rewards, have the shortest distance, or take the shortest amount of time would be tedious. We felt add the option to sort the results would counteract this issue. Finally, we wanted to update the design to use newer Android support library elements, so I added a floating action button with a check mark to replace the Directions to Location button. This also adds the flexibility to use other features of the floating action button in future such as expanding to display multiple actions. At this point, this iteration of the interface didn’t connect to any trip preview. It instead immediately connected to Google Maps.

However, the primary change between these two iteration was the shift from a one-screen trip selection process to a two-screen solution. This switch was primarily because we wanted to make the design seem brighter and more open and also wanted to allow more space to display trip metadata. We made some other changes to achieve those goals also. We reduced the height of each of the input fields, lightened the theme color and removed the gray background, and as mentioned earlier, replace the Directions to Location button to a floating action button.

On the second screen, I put a tabbed layout where a user can see the results for each mode of transportation individually. We also decided to keep the menu
which shows our recommendations based on our User Optimization (UO) algorithm. The UO algorithm takes the set of all available alternatives between an origin and destination as an input and returns an optimized subset of those alternatives. The generation of this subset is based on an objective function such as consumer surplus in order to maximize the utility provided to the traveler. As the optimization runs over user specific preference parameters, different travelers will receive different menus of travel options. Each subset of the results shown takes up the entire page outside of the header and title. The user is able to navigate through them by swiping or tapping the section icon in the header.

Separating the input section from the results display has many benefits. First, it provides a separation of concerns and clarity of the stages. The input section functions as the place where the user outlines the specifics of their trip (as can be seen in the form) and the results page is separately used for viewing the results. If both items were on the same page, a user will attempt to scroll down through the results while consciously or subconsciously looking back at the input to confirm details about the trip, wasting time. Further, when a user updates the form after an initial search, the results don't automatically reload in order to avoid wasted requests, Therefore, placing the inputs and the results on the same page could be confusing during the moments when they aren't consistent. This consistency issue disappears when the two sections are separate. After all the inputs are valid, a submission action button appears and the user can submit the form, and the results appear on the next page.

Secondly, there is now more space available to display the results. In versions where the input form and the results were on the same page, the form submission button causes issues. It either takes up too valuable screen as a "Direction to Location" button or appears awkward as a floating button overlapping the bottom of the form. Though placing the floating button on the bottom of the screen as in the final design fits better, when combined with the display, other issues pop up. The button overlaps the results list and blocks the view of the last result being displayed. Though it goes away after the form is submitted, it must return when the form is interacted with so that the form can be resubmitted. However, if someone accidentally taps on
any of the input elements, the results are blocked and this feature becomes a nuisance.

More important even than the reduction of overlapping elements, separation into two screen provides a ton of flexibility and expands the options on how we could display the results. Previously, because of space constraints, the results displays was basically limited to a very basic list format. A grid wouldn’t make much sense and the extra margins and padding needed between elements for a grid to be readable simply wouldn’t fit. Any sort of complex, segmented, or separated list format was also out of the question, again due to the lack of space. Another idea would be to segment the results and display them by segment. Though it would provide clear distinction when scrolling through the list, it would only make the scrolling longer and wouldn’t allow for easy switching between the different modes of transit. Furthermore, when the interface want to feature certain trips at the top of the menu, these results would become harder to access once a user reaches the bottom of the list, reducing the value of highlighting them. On the other hand, separating the results completely by category would help the scrolling and transition between different modes of transit problem. This would also give the "featured" routes their own distinct list. The featured routes is composed of the best route for each mode of transit based upon the user’s preferences. Since we predict the user will want to pick one of these alternatives more than any other, giving them their own space is ideal. Unfortunately, though this option seems nice to put into that section of the interface, it generally requires some sort of extra header at the top of the lists, taking up even more space.

Finally, the bigger area on the second page increases visibility which allows for efficient information assimilation and minimizes the memory load on the user [7]. Looking at the half screen results display in earlier designs, the text is hard to vertically segment because it runs together, despite horizontal lines in between. This is especially evident in Figure 3-4, where the column containing the value for gallons saved and tokens earned devolves into one contiguous, unclear rectangular block, further exasperated by the fact that Tokens and gallons are approximately the same length in this font. Comparing the two versions of the lists, the second has more spacing between each row, removing the need for lines as separators and allowing each
to read as its own paragraph. The decision to change the background to white came from a general desire from the team to brighten the interface, but it also allowed us to choose a thinner font to display the trip metadata, further increasing spacing and readability. Also, changing the labels for the units of each measurement, i.e. gallons to Gal and mi to miles, meant the results would be not be tightly packed rectangle, making them easier to read at a glance.

3.1.2 Implementation

The application opens to a DrawerLayout containing a Fragment for the dashboard. The majority of the app's screens are achieved by replacing the internal fragment of the DrawerLayout and updating the title view. The Trip Planner input is composed of a LinearLayout of the input elements. The ListView which was initially on Trip Planner input was replaced with a RecyclerView. RecyclerView was added to the Android library in version 22 for this purpose. Though it lacks some built-in features...
Figure 3-2: *Input Form* Trip Planner screen before route details are submitted.

Figure 3-3: *Bunched* Early iteration of menu with poor spacing between rows.
Figure 3-4: *Mode Separated* Trip Planner route display, separated by mode of transit.

like selection modes and the OnItemClickListener class, it is natively leaner, more flexible with its listeners, and supports different layouts via layout managers such as the LinearLayoutManager, the GridLayoutManager (used in the current marketplace design), and the StaggeredGridLayoutManager. In the later iterations of the interface, this RecyclerView was moved to the new route list display with one for each tab. The form submission button to search for directions was re moved and replaced with an Android FloatingActionButton at the bottom of the page. These buttons with a built-in function to open a menu of multiple actions on top of the basic onClick event. Updates like these will benefit the app in the future as Android continues to develop them, optimizing their functions and adding more features to them. The results list page is TabbedActivity where each modes list of results is a Fragment containing an instance of the aforementioned RecyclerView.
3.2 Trip Preview and Navigation

3.2.1 Design

Originally, the FMS Advisor app didn’t have any sort of trip preview screen. Once a trip was selected from the list, the app would save the trip internally as RUNNING and send the user to Google Maps with all the waypoints selected. In order to use the trip planner again for another trip, the user would have to complete the previous trip or completely close the app (including all background process). Otherwise, if a trip was not completed, upon reopening the app, the user would see a screen noting that they have a trip currently running and that the user should complete it. Then, if the user doesn’t want to follow the path they chose or decides to not go on that trip and doesn’t know how to completely close the app, the app becomes unusable until the operating system closes it in order to reuse memory.

In order to combat this issue, I decided to increase the safety of the app by adding a trip preview page that is shown before the user actually starts the trip. Safety, in this case, can be defined as [protecting] the user from making mistakes [4]. This page functions as a confirmation that the trip the user chooses is actually the one they want to take. The goal, then, is to display as much information as necessary in a way that is aesthetically pleasing. The first design (see Figure 3-5) focused completely on the map display of the path of the trip. The map view, whose view is bounded around the trips path, fills the entire screen, outside of the trip metadata row. The metadata row is placed at the top of the screen and serves as a visual confirmation that this trip is the one the user meant to select. Since this interface relies on Google Maps navigation, a floating button is placed in the bottom right corner.

We liked this design; however, it lacked the level of detail we wanted to show and placed some elements slightly out of place. Though the map is an intuitive way to see route of the trip, it doesn’t show exact details of which streets a driver should turn on or what corner a bus stop would be located unless the user zooms in. Furthermore, doing this for every turn would take as much time as the rest of the trip planning process combined and would render the app virtually unusable if the user wanted to
do this for every trip they take. Therefore, we decided to add step-by-step instructions to the preview page, but it was important that they not take up too much space so the map could be understood. Looking at the current designs for Google and Bing Maps, both of those use hidden panels, so as not to cover the map. The panel is revealed with a swipe gesture from the bottom of the screen. In order to avoid confusion and increase safety, I decided to use the same gesture. Not only because of the symmetry, but also because the other directions already have other common uses for swiping. Tap gestures on a map have the same issue because tapping on a map already has a different purpose. Therefore, to show the affordance for this and use Androids bottom sheet behavior, the metadata row was moved to the bottom of the screen.

For the step-by-step display, I implemented a simple right detail list view (same as in Figures 3-7 and 3-8), where the icon represents the mode of transit for that step and the text of the form continue straight, turn left/right, exit, etc.. For directions that are received from the Google Maps API, the text is received from that API call. For the routes returned from DynaMIT, we generate text using the latitude and longitude and taking dot products between different adjacent legs. Currently, this text includes actual latitude and longitude numbers instead of street names, but this does negatively impact the usability of the app when displaying DynaMIT trips. Hopefully, future iterations will display those street names, whether by querying an external API or directly returned from DynaMIT, preferably the latter.

### 3.2.2 Implementation

On this page, the map that fills the majority of the screen is an Android SupportMapFragment. This map provides all the interaction and viewing capabilities as you would see in the Android Maps application. The path displayed on the map uses Google Maps APIs PolyUtil and PolyLine classes to encode, decode, and display the paths returned from the servers. This allows us to send shorter less data in the Intent from the TripPlanner to the TripPreview and to use Googles more optimized encoding and decoding algorithms. The footer view is composed of a NestedScrollView with a FloatingActionButton pinned to the top left of it. This footer exhibits the
Figure 3-5: *Preview Initial* First iteration of the trip preview.

Figure 3-6: *Preview Final* Current iteration of the trip preview design.
Figure 3-7: **Google Steps** Displays the step-by-step instructions received from the Google Maps API.

Figure 3-8: **DynaMIT Steps** Displays the step-by-step instructions produced from the routes we receive from DynaMIT.
Figure 3-9: Navigation Trip shown was selected from the Trip Planner and the waypoints were provided to Google Maps for navigation.

BottomSheetBehavior, a feature of the Android library that can used in the combination with a CoordinatorLayout to give the effects described in the design section. In the NestedScrollView, the piece that is peaking out (visible even when rest is not) is implemented using a TableRow, similar to the one in the routes list and the list of steps revealed by swiping from the bottom is another RecyclerView.

### 3.2.3 Issues

Currently, the DynaMIT backend does not return street names for the locations it sends us. This means that in order to give street label turn-by-turn directions, we must use the Google Maps APIs Geocoder class which queries Google's servers. This poses two issues. First, the paths returns to us often have hundreds of points, which is great for displaying a smooth path, but that also means multiple points may appear on the same road, even with the same address. This makes it difficult for the app
to determine the different before a remaining on a curving road or fading onto a different street. Second, with multiple paths returned from our servers and hundreds of Geocoder calls to be made for each path (one for each point), performing thousands of calls in total for all the paths causes the app to crash because of too much delay. Therefore, at the current moment, the steps preview for driving is far less useful than that of Android maps since we can't display street names. We have also attempted to use an open-source navigation library for turn-by-turn navigation in order to allow the navigation to remain inside our application. However, waypoints are not yet supported in this library, so the app must continue to rely on navigation from Google Maps (Figure 3-9) in the meantime.

3.3 Trip History

3.3.1 Design

After completing a trip or a collection of trips, users may want to review their past trips. This can serve as a way of verifying that they received the number of tokens they believed they would, understanding which trips gave them the most tokens, or even for the generic case of seeing what they left work or school. Either way, a page to look back at a user's past trips is useful.

The first design I created was focused on being simple and usable. The map at the top displays the start and end of the selected trip on map display. The previous trips are shown in a list view at the bottom half of the layout in a table row showing the selected mode of transit, date, start time, end time, energy usage, and tokens earned. Trips are selected by tapping on the row that metadata was displayed on and the start and end location would appear on the map. The primary benefit of this layout is that it allows a large number of trips to be accessible on the screen at one time. The user's trip history for the past month (100 trips) can all be loaded into the list view and then accessed by scrolling and selecting.

Unfortunately, this design, while simple to implement, leaves much to be desired.
aesthetically and functionally, mainly because only one trip can be displayed on the map at a time. The list view in this interface visually faced the same issues as the original trip planner. Not only that, but there is no visual feedback as to which trip is being displayed. Even with the map display showing the origin and destination, the user doesn't have a unique identifier. The same trip can be taken on multiple days or multiple times on the same day, potentially with differing numbers of tokens. These trips would appear exactly the same on the map, and in the case where the user has only used TRIPOD for one trip (one origin and destination), no visible action would be taken by app when a trip is selected and the feature would be hidden from the user. Furthermore, since only one trip is shown at a time, it can be hard to compare different trips. If a user wanted to check which path earned them more tokens, took less time, or used less energy, it would be difficult to do visually, reducing one of the benefits we hope to achieve. By allowing the user to see the benefits of taking a different route for the same origin and destination, we hope that users will recognize which routes are generally better and continue to take them even when they're not using the app or when the app doesn't provide any token incentive for them to do so.

Therefore, the final version of the trip history page is displayed a format more similar to an RSS feed (see Figure 3-11). The trips are each given their block in the UI and displayed individually in the feed. Each block consists of the same row from the list view in the previous iteration along with a map displaying that trip with the path added between the origin and destination. The blocks are also padded with a small amount of whitespace on all sides and a black border. As drivers may potentially deviate from path provide, we decided to show the actual path the user drove, walked, or rode in the history. This will assure that time, tokens, and energy match up with the route shown and avoid confusion.

This design is limited, however, in how many trips can be loaded at one time. Because each trip has its own map display, the amount of memory and screen space needed for each trip is drastically increased. Even with the ability to use progressive rendering where new trips are rendered when a user reaches the bottom of the page and with all the interactions with the map disabled, the page will likely slow down as
the user goes further back in history and returning to your most recent trips would require lots of scrolling. This speed issue could also affect other pages since all the fragments of the drawerlayout remain loaded once they are used inside that activity. These issues could likely be solved by implementing Back to top and History reset functions which can automatically scroll the page back to the top of feed whenever a user wants to do so and reset the feed to the default size when a user switches to another page such as the trip planner or marketplace. Unfortunately, we have not yet had the time to implement these or design when and where they should be used.

### 3.3.2 Implementation

The first version of the interface was implemented using a SupportMapFragment in the top half of the interface and a RecyclerView in the bottom half, like the one used in the trip planner. Then, upon switching to the new design, I attempted to use a RecyclerView for the entire interface where each block would be inflated and attached to its ViewHolder. However, because of the way the views inside the RecyclerView
Figure 3-11: *History Feed* Trip History displayed in the style of a RSS Feed.
are inflated and the way SupportMapFragment interacts with the view, rendering it threw many errors. These arose because of conflicts with the map layout ids and ids being reset during the rendering. Therefore, all the blocks were inflated in a loop and added to a LinearLayout manually. This allowed me to set the ids in the loop without any worry of a layoutmanager resetting them. When retrieving the actual route the user took, the routes are queried from the database on our server and matched based on time, approximate origin, and approximate destination. The routes are logged based on GPS data using algorithms developed in the previous work on the FMS project.

3.4 Rewards Marketplace

3.4.1 Design Evaluations

When evaluating current interfaces, we looked at many types of marketplaces. We evaluated a wide variety of store and marketplace interfaces ranging Amazon to Microsoft Rewards to the Tetris Friends shop in order to get ideas for the things we wanted to see in our interface along with things we felt would be inappropriate or ineffective our interface. The interfaces in the figures below (and those in the Appendix) are marked with the following three colors: green, yellow, and red. A green box is placed around aspects we would like to include in our design, either based on appearance, concept, or function. A yellow box is placed around items that represent a useful concept with poor execution or a useful concept no application for our system currently, but potentially an application for future iterations. Items in red either look very poor aesthetically or show a concept that would take away from our application. Working off this analysis, we recognized these features as the most important for our marketplace: aesthetically pleasing, easy to understand, quick to use.

Looking at marketplaces or stores in apps or on the internet, we noticed that a super-majority have their process for buying items or redeeming rewards broken into two stages, searching and inspecting. In the searching phase, the user looks the
catalog of items displayed in some sort of list or grid view. Then, after selecting one of those items, the user is taken to a screen where they can see further details about them items alongside a button that allows them to purchase or redeem it. Since this style is fairly universal, no comments in this analysis will be made about whether or not TRIPOD should use the style, but instead I will analyze how well the different aspects of each interface would contribute to the TRIPOD marketplace. Therefore, only interfaces that fall into that category will be analyzed here.

In the interfaces we analyzed, we liked when there was clear labeling and good spacing. Images showing the reward item were large and clearly aligned with the corresponding text. on the pages for both stages. Titles were generally clearly identified in bold and descriptive subtexts were generally placed immediately underneath in smaller fonts. The elements marked in green also provide other necessary information such as the value and the company providing the reward. Also, not only is there a button used to purchase a selected reward, but also a final confirmation that the user actually meant to the purchase that item.

Though these interfaces had many components we liked, there were plenty of components that made interfaces unattractive, harder to use, and more difficult to understand. Both Amazon interfaces had lots of small, uniformly sized, black or gray text bunched together, making all it much more difficult to read. Also, lots of extra information was provided in the list view, making it harder to discern the more important details as a user scans the page. Furthermore, a couple of interfaces had detail pages where different options could be selected a dropdown or radio button list. Since our initial marketplace will likely not have an excessive number of rewards, the detriment of hiding different values of rewards from the searching screen will outweigh the benefit of shortening the list by grouping rewards. We also decided a rating system would be irrelevant for our system since we are the only "seller" or "distributor".

Analyzing the other interfaces also gave us some ideas we hadn’t thought of before that may be useful for the marketplace in its future iterations. When the number of rewards becomes large, it could be useful to separate different categories of items into
their own tab alongside a featured tab. This should be familiar to our users since the app does the same thing with the mode in the Trip Planner. Also, depending on the item, we will likely have to provide a section of terms and conditions as is done at the bottom of most of the pages where the user buys an item.

### 3.4.2 Implementation

Currently, there are two possible alpha versions of the interface implemented (see Figures 3-12 and XX). The interface implementations are mostly the same, outside of the fact that one is wrapped inside of tab layout. A LinearLayout is used for the top row, and a RecyclerView occupies the remainder of the screen. Separating this RecyclerView from the previous ones is that its LayoutManager is a GridLayoutManager instead of LinearLayoutManager. Other than that, most of the RecyclerView.Adapter code is similar to the others, making the code more understandable for the development team in the future.
3.4.3 Database

Though the majority of my work focused on the Android user interface, I also contributed to the database to facilitate the models for personalization and optimization. The "Alternative Attributes" table stores the menu results that are displayed to a user, each marked by a unique ID given to that menu of results and a boolean noting if an alternative was chosen by the user or not. Each of the alternatives will have columns in their which are not applicable. For example, trips by car do not have any transfers and parking costs are not applicable trips made by biking. This redundancy is unavoidable, however, when storing the alternatives in a single SQL table. Splitting the different modes of transportation into different tables could solve this issue. However, it would likely slow queries for all the alternatives presented in a menu since a menu may contain alternatives with different modes. Four SQL queries would be required to retrieve a menu instead of one, and the queries would lose the benefits of SQL indexing when grouping data. Therefore, we chose to store the attribute data into a single SQL table. This data can then be studied by optimization and machine learning algorithms to determine which alternatives are more likely to be chosen when compared against others. This technique can be applied on both a global scale for a generic menu and an individual scale for personalized menus.

The results from this analysis are used in determining user preferences, which are stored in the "Preference Parameters" table. These parameters signify how important factors like mode of transit, cost, time, and tokens are to that specific user so that the menu may be optimized for them. As an example, one user may have a propensity to choose transit alternatives with no transfers or car alternatives regardless of time. Another user may choose bike alternatives most often or choose transit alternatives which have low in vehicle travel time. These two users would receive different menus as a fast, energy-efficient transit alternative with multiple transfers would be a terrible option for first user, but a fitting one for the second, as could also happen with alternatives in other forms of transit. Therefore, placing these preferences in a feedback loop with the menu selections should provide increased personalization as the
user continues to use FMS Advisor.

The full list of column identifiers and data types for each table are shown in tables 3.1 and 3.2. Note that IVTT stands for "in vehicle travel time" and TC stands for "total cost" in the Preference Parameters table.
Table 3.2: Alternative Attributes

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu_id</td>
<td>string</td>
</tr>
<tr>
<td>total_travel_time</td>
<td>float</td>
</tr>
<tr>
<td>vehicle_travel_time</td>
<td>float</td>
</tr>
<tr>
<td>walking_time</td>
<td>float</td>
</tr>
<tr>
<td>early_schedule_delay</td>
<td>float</td>
</tr>
<tr>
<td>late_schedule_delay</td>
<td>float</td>
</tr>
<tr>
<td>total_cost</td>
<td>float</td>
</tr>
<tr>
<td>toll_cost</td>
<td>float</td>
</tr>
<tr>
<td>fare</td>
<td>float</td>
</tr>
<tr>
<td>parking_cost</td>
<td>float</td>
</tr>
<tr>
<td>number_of_tokens</td>
<td>int</td>
</tr>
<tr>
<td>number_of_transfers</td>
<td>int</td>
</tr>
<tr>
<td>relative_energy_consumption</td>
<td>float</td>
</tr>
<tr>
<td>driving_score</td>
<td>float</td>
</tr>
</tbody>
</table>
Chapter 4

Contribution Overview

With these four stages designed and implemented, we have created simple and complete user flows that can be used and refined as we do large-scale testing this summer and fall. The diagram below describes the potential flows that we expect the users to go through when using our app. When a user opens the app, there are three main points of entry: the trip planner, the trip history, or the marketplace. From the trip planner, a user can now complete the entire action of searching for, selecting, previewing, and navigating through a trip with selected waypoints using Google Maps. This allows for a consistent user experience and greater customization to the interface in the future. Not only that, but the newly added preview step increases the safety of this scenario since the user is less likely to start navigation on a trip they didn’t plan to.

Additionally, the user can review their trip history. Upon completing trips, this will provide in-app confirmation to the user that the trip was recorded along with the exact number of tokens including bonuses and detractions due to driving style and carpooling. When deciding upon a new route, the trip history can provide a reminder of which routes have worked out well in the past. Though the route rankings will be updated each time based on traffic conditions, users may want to travel on more familiar routes in the case of tie.

Finally, we now have a marketplace serves as an avenue for us to incentivize users to actually take the routes we recommend. With our previous and current
trip planners, tokens can be collected by users upon completing routes; however, the marketplace and the rewards inside give these tokens tangible value. If the tokens had no value outside of themselves, they would become the approximate equivalent of an energy savings counter, motivating only those users who already have a desire to travel in a more energy efficient manner. By creating the marketplace, we now have a way to incentivize people would react more strongly to monetary value than environmental protection.
Chapter 5

Next Steps

The most immediate needs for the app interface are the completion of the marketplace and full integration with DynaMIT. Currently, the marketplace design only has the bare necessities and functions more as a proof of concept than a production marketplace. The design of the interface needs to be refined and final details on things like terms and conditions need to be defined. Also, the app still to add real rewards to the marketplace. This also includes settings the prices on these items so that tokens end up having a reasonable value. Further, the app should query the DynaMIT server to get transit, biking, and walking directions instead of the Google Maps API. This will require both server and app updates, but should possible in the near future on the application side since the format will be similar to that of the driving routes.
Chapter 6

Conclusion

The work done here creates a more complete trip advising system for people's day-to-day travel. While it builds upon work done by previous researchers on gathering travel data and providing recommendations, this work adds new features in the form of an improved trip planner, as well as a new trip history and marketplace. These features should make the app more usable and should enable us to give users tangible rewards for their energy savings. As user testing is performed, the interface design and analysis here will be combined with real user data to serve as a platform for final iterations for when the complete product is taken to market.
Appendix A

Figures

These figures are of the marketplace interfaces that were evaluated from all the different sources.
Figure A-1: Amazon Android search results list.
Figure A-2: Amazon Android individual results page (top half).
Size: 72 CT

$22.49

FREE Shipping on orders over $25.

In Stock.

Want it Wednesday, April 12? Order within 6 hrs 4 mins and choose Two-Day Shipping at checkout.

Ships from and sold by Amazon.com. Gift-wrap available.

Figure A-3: Amazon Android individual results page (bottom half)
Figure A-4: Google Play Store Android home page.
Figure A-5: Microsoft Rewards Marketplace web page home.
Figure A-6: Microsoft Rewards Marketplace individual result page.
Figure A-7: WHAFF Rewards Android app home page.
Figure A-8: WHAFF Rewards Android app home page with item selected.
Description
Dropbox lets you take your photos, docs, and videos anywhere and share them easily. Access any file you save to your Dropbox from all of your computers, phones, tablets, and on the web. With Dropbox, you’ll always have your important documents and work

Figure A-9: Windows Store Windows App home page.
Bibliography


