Smartphone Interface for Incentivized Energy Optimization System - FMS Advisor

by

Narindra Peaks

B.S., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Author ....
Department of Electrical Engineering and Computer Science
May 18, 2018

Certified by..
Moshe Ben-Akiva
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by ....
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Smartphone Interface for Incentivized Energy Optimization System - FMS Advisor

by

Narindra Peaks

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The TRIPOD project aims to provide a system that optimizes energy savings during travel for a population of users, through personalized incentivization of efficient travel alternatives. The Future Mobility Sensing (FMS) Advisor provides the user interface for this system, allowing users to plan trips through a personalized menu of travel alternatives, receive rewards for selecting energy efficient options, and redeem those rewards for goods and services from partnered organizations. Packaged into a smartphone application, the Advisor provides an efficient user interface that allows users to plan trips, navigate those trips, view previously completed trips, and redeem rewards in the marketplace. However, it also requires a method to optimize usage of the sensors present in most smartphones to gather enough meaningful data on the users' travels in order to effectively personalize the trip planning process, as well as leverage that data to detect and validate the users' specified trips within the application so that rewards can properly be awarded. The validation algorithms are housed on a backend server that stores all the data for the system using a blockchain implementation in order to keep a record of transactions for all the trips and the marketplace. The FMS Advisor attempts to bring all four of these metrics, personalization, detection, validation, and incentivization, together into one mobile application framework, in a way that has not been done by any other applications currently on market.

Thesis Supervisor: Moshe Ben-Akiva
Title: Professor of Civil and Environmental Engineering

Acknowledgments

I would like to take this space to acknowledge all those who have helped me arrive at this point in my life: my wonderful coworkers in the ITS Lab, especially Ajinkya, Xiang, and Mazen; my supervisors, Bilge and Carlos; my advisor Moshe; and all of my friends and family that helped keep me energetic throughout my academic career. You are all wonderful people and I couldn't have made it this far without your support, encouragement, and advice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

None can deny the prevalence and the importance of the smartphone in modern society. As computational resources became more common, smaller, and more efficient, the science fiction concept of a pocket computer became reality, and with the advent of these mobile computers came a revolution in the way software and applications approach human centered problems. No longer is data confined to census results and desktop surveys. Studies can be carried out passively while yielding real-time results. Data can be collected on the fly without invasive devices dedicated to such collection. These new innovations had a profound effect on urban studies, introducing the mobile phone and the concept of smart mobility into urban planning and transportation analysis.

In the modern world, virtual all working people own a smartphone and carry it with them at all times. This essentially means that everyone commuting in a given area is carrying a box full of sensors that has more processing power than most home computers in the 1990s. People now use their phones to plan trips, to navigate to their destinations, and to explore unfamiliar areas, all at the tap of a screen. These pocket computers offer unique opportunities to leverage the data from day to day travel in order to propose solutions to problems plaguing cities across the globe.

The Intelligent Transportation Systems lab conducts research using smart-mobility and other modern concepts to improve the methods and modes of transportation used around the world. One of the projects under development in the lab is the TRIPOD project, also known as Sustainable Travel Incentives with Prediction, Optimization, and Personalization [1]. The main goal of the TRIPOD project is to collect data on users' travel habits and propose more energy efficient alternatives to optimize the energy usage of the overall traffic network, in real time. The user experience module of the TRIPOD project functions mainly as an extension of the Future Mobility Survey (FMS) [4] data collection system, utilizing the FMS trip detection algorithms as a means to verify the actions of users interacting with the interface of the TRIPOD project. The interface is a mobile application known as the FMS Advisor.

Built on top of the existing Future Mobility Sensing android application for ease of implementation, the FMS advisor allows users to initiate a trip request and then provides them with a set of personalized travel options. Each user is also associated with a set of preference parameters that are updated as the user makes choices in the system. These features allow the FMS Advisor to tailor the travel alternatives to each user, growing more and more personalized as the user makes more choices.

The options offered by the Advisor are calculated using real time traffic information compiled by a behavioral model of the traffic network. This gives the offered alternatives accurate travel times that are updated every 5 minutes. The travel options are also associated with some incentives, in the form of tokens, to encourage users to take more energy efficient modes of travel. These tokens can be redeemed for rewards in the marketplace section of the application, providing users with motivation to take the travel alternatives recommended by the system. The marketplace needs a set system to keep track of transactions that are made by users, in order to keep the value offered by the incentives.

In addition to all of this, the actions of the users must be validated in order to properly administer rewards. The advisor contains a framework designed to detect and analyze users' trips using the sensors present in most smartphones. The data gotten from analysis of the trips is then used to validate the users' trips based on the metrics used by the Advisor: route choice, mode of transit, departure time, occupancy, and driving style.

The route choice, mode, and departure time are based on the optimization that runs in the Advisor backend, but the driving style metric comes from work done on Trip Energy Consumption. Studies have shown that certain more aggressive driving behaviors are detrimental to fuel efficiency in vehicles, and in order to inform users of this, the TRIPOD application provides an incentive for environmentally friendly driving habits. All of these metrics together combine to flesh out the Advisor's incentive scheme and really give depth to the application's functionality.

Unfortunately, the application still lacks a viable method to achieve the features previously laid out. The current implementation of the core FMS android app does track trips and has algorithms for stop, mode, and activity detection, but it needs to be enhanced to provide the ability verify that the chosen alternative is actually performed (i.e. used the appropriate mode of transit, took the specified route, etc.). In addition, the infrastructure to personalize the system for each specific user must be put into place. The goal of this thesis is to provide a method to optimize usage of the sensors present in most smartphones to gather enough meaningful data on the users' travel preferences and behaviors to be used by the Tripod overall optimization. This will allow for a smooth user experience in terms of travel planning and token exchange, in addition to filling in all the rest of the gaps in functionality of the application. Overall, the work presented here aims to combine the aspects of personalization, detection, validation, and incentivization in a smartphone based mobility application.

Chapter 2

Background Research

The FMS Advisor is an application that aims to combine previously studied features in a novel way. This means that all the research done on the individual features can be leveraged to strengthen the Advisor as a whole. At its core, the Advisor brings together 3 different features:

1. A mobile interface centered around planning trips, viewing previous trips and other historical data, and exchanging currency for products and services
2. Algorithms necessary for obtaining and processing real time information in order to provide optimized routes on user given origins and destinations
3. Detecting these trips and validating the detected data against some ground truth

Development of mobile interfaces has been on going since the advent of the smartphone, as has optimization of mobile sensing, so a good amount has been done on these features already. In addition to independent studies, other projects in the ITS Lab have results that are relevant to the systems in the Advisor. The following paragraphs detail a few different methods for approaching development of the high level features of the Advisor.

2.1 Relevant User Interfaces

When designing interfaces for usability and overall user experience, there are multiple schools of thought as to what elements to use and where they should be placed. The Nielsen Norman Group is one of the most respected in this field has published a series of heuristics to use when designing interfaces [12]. These heuristics are the ones used when evaluating the interfaces used as inspiration for the Advisor.

Inspiration for user interfaces that plan trips is quite abundant in the modern application ecosystem. In today's era of mobility on demand services, many users in cities have foregone ownership of vehicles in favor of services like taxis and ride-sharing applications such as Uber and Lyft. These applications share the same sort of trip planning goal as the FMS Advisor, and can be used as sign-posts for design guidelines.

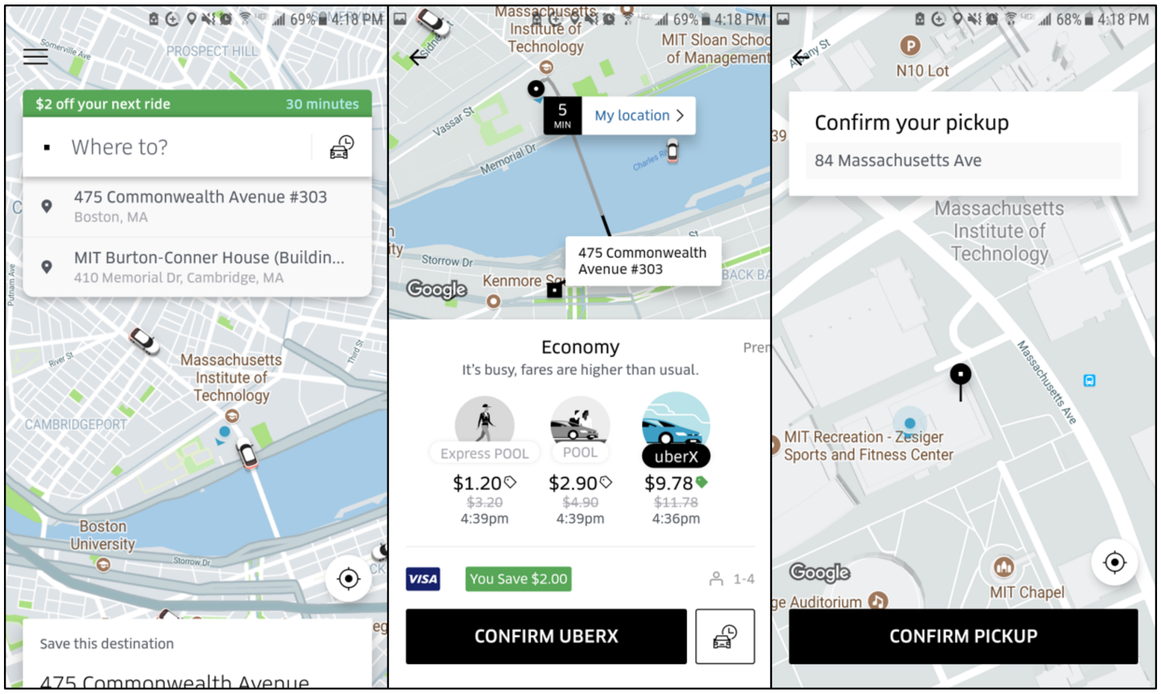


Figure 2-1: The interface for Uber's android application. Notice the maps and the visibility of all the different modes of transit.

The Uber mobile application has all the marks of a well designed mobile application: text large enough to read, intuitive swipable screens, and intelligent display

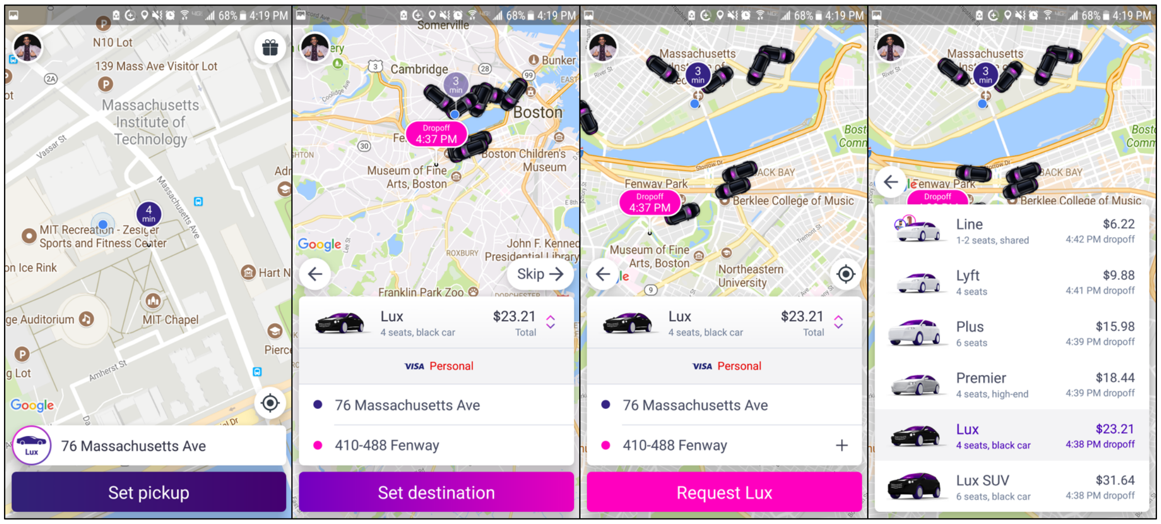
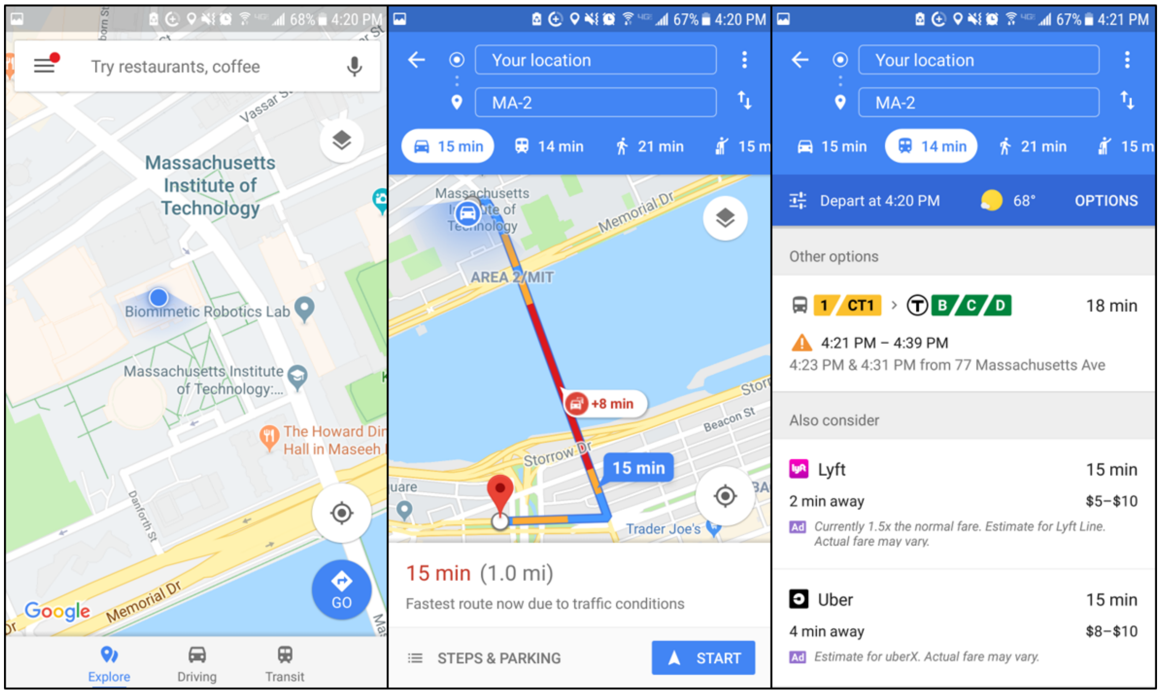


Figure 2-2: Lyft's user interface. Take note of the extra screen to reach the travel options.

of information, following the heuristics of recognition and flexibility according to Nielsen. The main screen is a map, which shows the user routes upon the selection of a destination, and after the user inputs one, a second overlay appears, offering different travel options for the user, while the map displays information relevant to each option. This is similar to Lyft's application, which uses the map to set origin and destination nodes directly on separate map screens, before choosing the mode of transit. In Uber's case, the user cannot interact with the map until after selecting all of the other parameters of the trip. One other difference is that Lyft's travel options are not directly presented to the user in a tabbed screen; the user must click on the selected option to reveal a hidden menu with the rest of the options; this is not as usable due to the extra screen.

A third interface to examine is the Google Maps interface. The main interface of this app is the titular map, which is interactive and rife with information through the entire trip planning process. However, in terms of trip planning flow, it is a medium between Uber and Lyft, since it infers the user's start location unless explicitly changed. In addition to this, the different travel options presented are more visible to the user than either Uber or Lyft, but the tabbed layout is more similar to Uber's. The one common factor across all of these trip planning applications is the



15 min (1.0 mi)

Fastest route now due to traffic conditions

1 / CT1 > B / C / D 18 min

4:21 PM – 4:39 PM

4:23 PM & 4:31 PM from 77 Massachusetts Ave

Lyft 15 min

2 min away $5–$10

Ad Currently 1.5x the normal fare. Estimate for Lyft Line. Actual fare may vary.

Uber 15 min

4 min away $8–$10

Ad Estimate for uberX. Actual fare may vary.

Figure 2-3: Google Maps has the least amount of screens of the three, requiring only an overlay to find the directions button, and a single screen to show all the options.

interactive map used to receive user input. This design feature makes the interfaces more engaging and aesthetically pleasing to users, and is worth implementing in other trip planning applications.

In addition to other trip planning applications, the Advisor can look to mobile marketplace applications such as the Apple App Store and the Google Play Store for inspiration and design concepts. In general, these applications are centered around presenting as much information to the user as possible. This follows the Nielsen heuristic of visibility, and helps uses discover new items they might be interested in. They also tend to have horizontal scrolls embedded within vertical scrolls, increasing the efficiency of user interactions with these components by reducing the number of screens a user needs to navigate in order find they want.

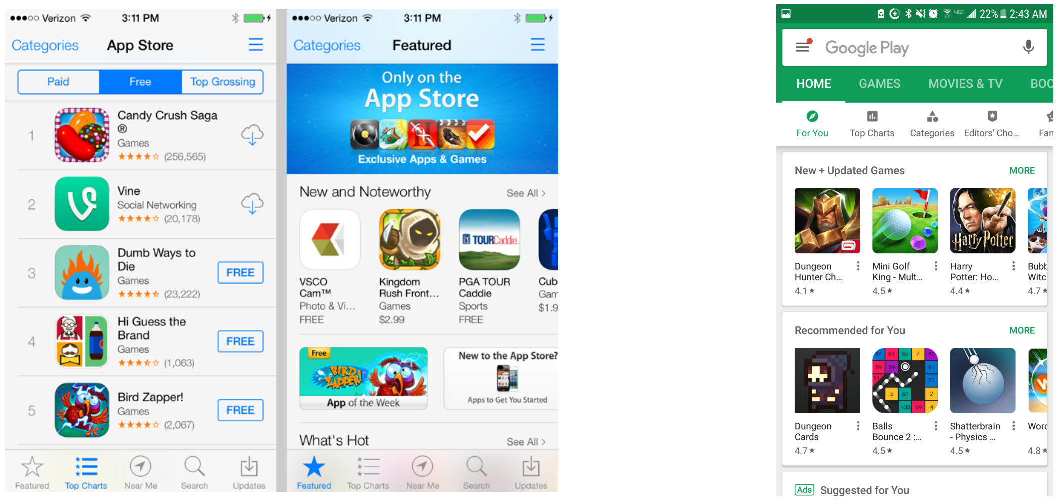


Figure 2-4: These are the Apple App Store and the Google Play Store. Take note of the tabbed layouts and all the horizontal scrolls to fit more information on a single screen.

2.2 Mobile Applications and Other Methods for Trip Planning

At the basest level, the FMS Advisor is the mobile application front end for the TRIPOD project, which is part of the TRANSNET program funded by the ARPA-E government agency. Started in 2015, it deals with optimizing energy consumed by personal transportation across a network of users, without major modifications to the network or the transportation options. 6 teams received funding to work on solutions to the challenges posed by this program, the most successful of which are the ITS Lab at MIT and the incenTrip team at the University of Maryland.

incenTrip is an application with functionality very similar to the FMS Advisor; it allows users to plan trips, while offering up some alternatives in an effort to reduce energy consumption. It also offers users incentives to take the alternatives, with these incentives being redeemable in a marketplace like interface. The team at incenTrip attempts to reduce energy consumption based on the same metrics as the Advisor, using route, departure time, driving style, and mode of transit to offer alternatives to the requested trip. All of these things together make incenTrip a very mature application, which has found success in the Baltimore-D.C. metropolitan area [13].

Separate from the work done by the University of Maryland on incenTrip, MIT's Intelligent Transportation Systems Lab has completed work on systems relevant to the trip planning problem as well. One of the most relevant is the DynaMIT 2.0 system developed by Seshadri et. al [10]. The DynaMIT system is a real time traffic prediction system that uses multiple data sources, as well as online simulation, to estimate and predict network traffic in real time. DynaMIT first uses all of the data at its disposal to estimate the state of the network; it then uses that state to predict the state of the network in the short term future, before repeating the process again. Given the speed at which this happens, DynaMIT is best suited to provide guidance to an online traffic regulation system, or in our case, a system providing optimal transportation policies.

In addition to the work done on predicting traffic, MIT has also completed work on estimating the energy consumed by a given trip. The Trancik Lab's Trip Energy system takes in the data provided by a series of GPS tracked trips to train a model that allows the energy consumption of new trips to be accurately estimated [9]. By utilizing a database of gps matched to real driving patterns, Trip Energy can account for variables in vehicles trips like driving style, traffic, average speed, length, weather, and even vehicle type when calculating the energy estimates. Trip Energy is best utilized as part of a larger system, and is the backbone of the energy estimates provided by the Advisor's trip planner.

2.3 Algorithms for Trip Validation

The Advisor leverages the algorithms employed by the FMS application for mode and trip detection. These are sufficient for validating the mode of transit. However, validating other aspects of the trip, like route and driving style, need new implementations. A few different methods for this have been proposed already. Due to the ubiquitous nature of smartphones in the current age, many people have conducted studies on the utility of data collected from smart phones. The general consensus is that the data is useful but noisy at times. In a study conducted on the accuracy of popular GPS tracking applications, the researchers found a deviation in distance of almost 2 kilometers over a marathon [2]. While the deviation is rather large, it only resulted in a time difference of about 12 minutes for the trip, which in the context of FMS seems to be a reasonable amount of error, especially for a marathon length trip. In a 2011 study, Johnson and Trivedi found that the data used collected from smartphones can be used to do driving style detection [8]. However, this system employed heavy usage of the phones cameras in addition to the GPS and accelerometer data. This provides an inconvenience to the users that prevents them from using the phone for navigation, a key portion of the FMS Advisor. In another study done by a team of researchers in Italy, the 1HZ data from phones was found to be appropriate to assigning road safety measures to prevent collisions [7].

Chapter 3

Overview of Methods and Contributions

During the time I worked on this project, most of the features of the application went through some sort of redesign or reimplementation, including a redesign of the flow of the entire application. From the users' perspective, the FMS Advisor itself is split into 3 main workflows: the Trip Planner, Travel Log, and Marketplace. This is because the Advisor's main functionality is centered around the idea of planning trips. This means the Trip Planner flow contains the bulk of the algorithms and back end structures for the application. Other than the Trip Planner, the rest of the functionality is housed under the umbrella of incentivization. For the user, this includes claiming incentives already earned and trying to alter habits in order to claim more incentives. In order to change habits, the user needs a method to view past trips and other summarizing statistics. This capability is housed under the Travel Log flow. The capability to claim incentives is housed under the Marketplace Flow.

These interfaces had versions already in place from prior work. Jamar Brooks, the previous research assistant, designed most of these interfaces previously [3]. His version of the FMS Advisor included all the interfaces present in the Trip Planner, Travel Log, and Marketplace flows, and all of his screens are present in this document for comparison. However, all of these were redesigned in order to maximize the efficiency, aesthetic appeal, and utility of the interfaces. In addition, an informative dashboard was added to the main screen of the application. This dashboard helps users keep track of previous information and energy optimization metrics, and allows the app to explore more gamification concepts.

In addition to the interfaces, each workflow has a backend component that helps fill out the functionality. As the most feature rich workflow, the Trip Planner allows users to input an origin and destination pair, receive a personalized menu of travel alternatives, choose one of them, and then receive token rewards at the conclusion of the trip. The advisor requires an extensive backend server to support all of these features, including trip validation, serving personalized routes, and updating the provided routes with real time traffic data. The trip validation is split up into route, mode, departure time, occupancy, and driving style, and most of the algorithms are housed in a backend server that collects all of the sensor data from the smartphone and processes it appropriately, while the personalized routes are handled with a database of user preferences and an algorithm that updates these preferences with every choice made by the user. The up-to-date route offerings are handled by a database of link travel times that gets updated with new estimates every 5 minutes.

The Travel Log flow deals mostly with historical data, so the backend component for it is simply focused on returning previous trips with all their metadata, including status of validation. This data is also used to supply the dashboard statistics. For the Marketplace's functionality, the advisor backend houses the infrastructure for all of the products listed in the Marketplace and a blockchain implementation to keep track of all the data in chronological order.

3.1 Overall Application Flow

A large part of the contributions in this thesis deal with the experience users have when using the FMS Advisor, and designing for that experience requires a understanding of the flow of data throughout the entire framework. These flows are summarized visually in Appendix A and will be further elaborated upon in the following paragraphs.

Upon opening the application, users are presented with the first screen of the Trip Planner workflow. The screen provides access to all the other main components of the application, as well as a straightforward, efficient way to continue through the Trip Planner flow. The Trip Planner flow is the most substantial out of the three, with the most transfers of data and logic between the phone interface and the backend server. These transfers go as follows, summarized visually in Appendices A-2 and A-3:

1. The user enters the origin, destination, and departure time into the map interface depicted in figure 4-6.
2. The app transitions to the route list interface (Figure 4-7), where the request for routes is made.
    (a) The request for routes goes to the backend server that houses the most recent traffic data updates and returns a set a routes with energy estimates based on real time data. This will be discussed in greater depth during Chapters 5.1 and 5.2.
    (b) These are currently only in place for driving options, but with further development will add other modes of transit as well.
3. Once the routes come back, the user's preferences are fetched from the backend server so that a few optimizations can be performed. We then filter the routes according to our personalization procedure, in which the user's preferences are used to select the 10 routes most relevant to the user's interests as described by

their previous choices. This process is detailed in the Personalization section in Chapter 5.3.

4. Once a user chooses a travel alternative, the app shifts to a preview of the trip route (Figure 4-8), where the relevant information is summarized and the turn by turn instructions for the trip are available to the user.

5. The user can either accept the alternative for the selected trip and start navigation, at which point the user's choice is recorded, or return to the list of travel alternatives. Navigation is currently handled by google maps, so when the user chooses to start navigation, the application switches over to google maps until the trip completes and also starts logging data from the phone's location and accelerometer sensors.

6. When the user accepts the selected alternative, the app records the choice made by the user in the context of all the travel alternatives offered and sends the data back to the server, where it is used to update the preferences stored for that specific user. These updated preferences are then used the next time the user plans a trip, allowing the application to further tailor the travel alternatives presented to each user.

7. Once the user reaches the destination of the trip, the trip is marked completed and all the collected sensor data and trip details are bundled and sent to the server so the trip can be validated. Once the trip goes through our trip validation procedure, the results are sent back to the user, and tokens are awarded as appropriate. This framework is discussed in Chapter 6.

The other two flows are much less involved and only involve singular interfaces without any substantial backend procedures. In the Travel Log flow, discussed in Chapter 4.3 and shown visually in Appendix A-4, trip data sent to the server once the trip is marked complete, and the route, tokens, energy, and times of the trip are recorded in the travel history of the user. The user can navigate to the travel history from anywhere in the application excluding the route listing procedure, and once there, the user can view all of the past trips in chronological order.

The Marketplace Flow can similarly be reached from any point in the application by using the app bar and provides the user with a means to redeem the token incentives for usable goods. The marketplace is where the true incentivization value of the FMS Advisor manifests to the user. It lists goods in a modern interface and allows the redemption of tokens received from the trip planner, completing the flow of the entire application, from incentivization to reward. The Marketplace interface is discussed in Chapter 4.4, and the flow can be seen in Appendix A-5.

In addition to these three workflows, the FMS Advisor also provides the users with an attractive summary interface page. Here, the user can see how their previous trips have put them in terms of the whole tripod community's stats, as well as against their own trips for the past month. This interface provides a skeleton for which future developers can implement a competitive form of incentivization, one the encourages users to compete with each other, not just against themselves. However, currently, it show the user's progress on the 5 different ways the advisor saves energy in the system: trip distance, trip energy, trip duration, tokens, and the overall number of trips. The Dashboard also contains links to start the other three flows. The design philosophy behind this interface is detailed in Chapter 4.1.

Understanding the flow of data and logic throughout the FMS Advisor is key to grasping how the features of the application have been implemented. This chapter was dedicated to delivering a cursory understanding of the flows in the FMS Advisor, which will also help clarify the ideas behind the design of the User Experience. The User Experience of an application dictates how users interact with a system and is the primary factor in determining user engagement with a product; a well designed User Experience, with proper logic workflows, enhances the impact of any application. The User Experience for the FMS Advisor is discussed in the next chapter.

Chapter 4

The User Experience

Now that the overarching ideas behind the user flow of the FMS Advisor have been explained, the design of the interfaces can be elaborated upon. The Advisor's main goal is to reduce the energy consumption of a population of users through incentivization, optimization, and personalization. In order for that goal to be achieved, the User Experience has to be tailored to emphasize those ideas to the user. As the main interfaces of the Advisor, the Dashboard, Trip Planner, Travel Log, and Marketplace each contribute to one or more aspects of this emphasis through their own individual designs. This chapter is dedicated to explaining the methodology behind each of these, to increase understanding of how the Advisor brings together all the features that it does.

4.1 Dashboard

In many account based user applications, there is a central screen where the user can go to view statistics about application usage, from the total number of hours invested into the app to the total amount of rewards gained during application uptime. These metrics are very important to the usability of any application, and add a number interesting ways for users to engage with the platform. In addition to that, the dashboard often serves as the hub for user activity, offering easy ways to access the rest of the functionality present within the application. In order for a dashboard to be effective, it needs to provide information to users that isn't immediately available anywhere else but is also relevant to the continued usage of the application, and overall, it must re-emphasize the features of the application to the user. This follows the Nielsen heuristics of visibility and recognition, since the display of the statistics here exposes the status of the users' historical data, as well as helping the user recognize previous as actions, as opposed to having to remember them [12]. Dashboards that follows these principles will add value to the interface and experience of any application in which they are present.

These design principles can be seen in many modern interface designs. Take for example, the homescreen for the Android Google Maps application, discussed in the Background section previously. The main element is a large interactive map, to encourage users to use the main feature of the app. However, underneath that is a draggable menu that contains entry points into a plethora a new information for users. It has 3 tabs, one that offers options to find new places of interest near the user, one for configuring driving options, and another for configuring transit options. As google maps is an application primarily used for finding directions and exploring surrounding areas, these features help emphasize and introduce users to the main draws of the application, engaging and encouraging them to make use of the functionality.

Another good example of these principles can be seen in the home screen for the Facebook Android application. Also known as the News Feed, it contains a seemingly infinite list of status updates and post from people relevant to the user. However, above the feed, Facebook placed a bar that illustrates the principles mentioned earlier. Facebook's main draw for many people is the ease of communication on the platform; the application emphasizes sharing updates with other users as well as spreading content and news around the network of users. In order to complement these goals, the dashboard of Facebooks app contains a bar that lists all the Stories of a user's friends, above the feed with other updates from the same group of people. It also contains a tab for news updates and other information the user might not receive from their friends. These two features, along with the list of notifications and a few others, serve to reemphasize Facebook's main draws and provide a very easy, efficient way for users to access all the main functionality from the same screen.

4.1.1 Metrics of Emphasis

The FMS Advisor, at heart, is a trip planning application. The main user feature centers around planning trips from an origin to a destination and helping users complete their transit. However, the purpose of the application is to optimize energy usage of the whole system of users using the FMS Advisor. To that end, the Advisor offers a menu of different travel alternatives to users when they plan a trip, incentivizing them to pick the most energy optimal. The alternatives offered to the user are incentivized through token rewards that can be redeemed through the marketplace interface found elsewhere in the app. These tokens are awarded to the user when they agree to compromise on one of the dimensions of the trip to make it more energy efficient. These dimensions include the energy consumed during the trip, the overall trip distance, the duration of the trip, and the number of overall trips. In order for the FMS Advisor dashboard to be effective, these metrics should be echoed back to the user in way that isn't found in other parts of the application but still has utility.

In addition to that, the FMS advisor has a strong community component to it. As the app aims to reduce the overall energy consumption of all of its users, it follows that these users would be interested in what others in the system are doing, and how they compare among the rest of the users. This sense of competition could drive engagement with the application higher. The app already encourages users to carpool as one of the ways to reduce energy consumption. The people carpooling would likely be acquaintances in real life, and could use the app more often if more rewards were given for saving more energy than their friends who also use the advisor.

4.1.2 Interface Design

Previous Iteration

The previous iteration of the FMS Advisor's dashboard attempted to follow a few of these principles, but overall did not add enough value to warrant inclusion in the application. The screen consisted of a mostly blank space with a large star, the user's name, a number of tokens, and a large button to start the trip planner. While the page was functional, it definitely left some things to be desired. First, the screen had a lot of empty space. This can be useful, if there are specific components that need more emphasis, but in this case, it only served to draw attention to the large star in the middle of the screen. The star was supposed to be the implementation of an incentive for users to reduce the total number of trips taken in a set time period, but was never implemented fully. In addition to that, it only provided access to the trip planner, neglecting the other two features of the Advisor, not to mention the fact that there isn't a way to access the dashboard after the user leaves it. Though this version of the dashboard lacked many things, it did attempt to follow the concepts mentioned previously, and paved the way for further iterations of the dashboard.

Version 1

The first iteration of the dashboard took some of the design considerations of the previous iteration and expanded upon them slightly. Here, I tried to use all of the space available on the screen in order to provide as much information to the user as possible, without crowding the screen unnecessarily. The tiled design would provide the ability to emphasize different pieces of information, or tie a few together, depending on the colors used in the boxes. This version of the interface provides the user with information about the overall number of all the trips taken, nd indicates a concrete for the user to receive another bonus. This was the goal of the star in the previous interface. Unfortunately, there are still a few issues with this one. While

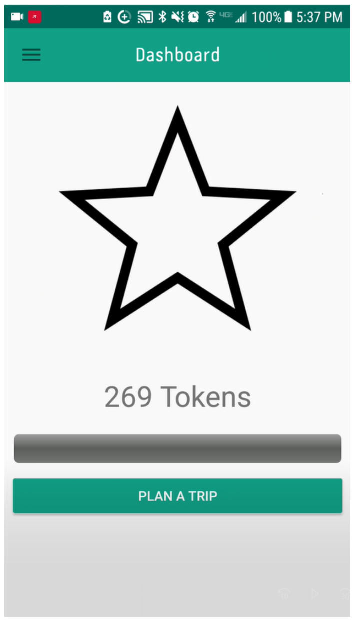


Figure 4-1: Existing dashboard interface

the interface is more aesthetically pleasing than the last, it doesn't actually offer the right kinds of information or interaction. Most of the information present on it is redundant, as it all references the number of different modes of trips taken, while the bonus is related to the same number of things. Also, it still only provides access to trip planner, and there still wasn't a way to return to the dashboard once the user left.

Version 2

The second version of the dashboard was mostly a redesign of the first, designed to be less cluttered and more in line with rest of the project. In this version, the fonts were updated to match the rest of the app. For this iteration, I focused on bringing the dashboard into parity with the rest of the FMS experience: the Advisor is one piece in an entire framework of Mobility applications, all of which have a dashboard. Most of these are quite different from one another, which is very confusing for users of more than one of the platforms. To help fix this problem, I rearranged the graphs to flow from left to right, and also eliminated the redundancy in the boxes. This

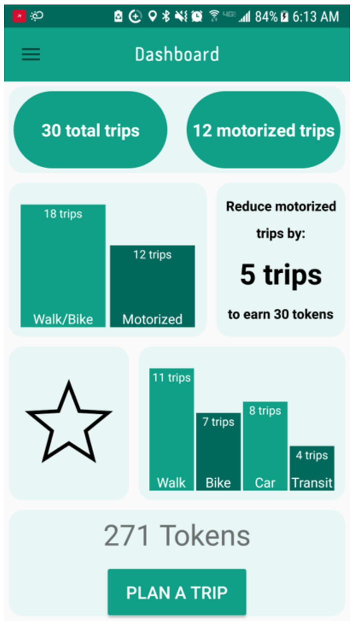


Figure 4-2: First iteration of the Dashboard interface. Notice the tiled design that allows for different blocks of information.

version of the dashboard uses the icons found on the web, and also in the Advisor, which provides even more consistency with the rest of the FMS suite. Despite these improvements, the interface could be stronger still. There is a wealth of information in the application that this dashboard ignores, while it still has not addressed the lack of access to the rest of the app's features.

Version 3

The third iteration of the dashboard finally takes the design principles mentioned earlier and acts on them, transforming the application for the better. This version of the dashboard is found as a pull up window present at the bottom of the first activity the user sees, the trip planner. Not only does this new home page emphasize the trip planning portion of the app, it reduces the number of steps taken for a user to actually plan a trip. In addition to this, dashboard includes icon based buttons to reach the other workflows in the app, allowing the user to find all the functionality very easily.

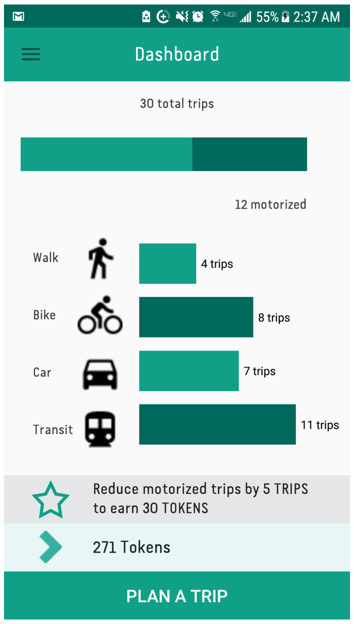


Figure 4-3: Second iteration of the Dashboard interface. The graph here is more in line with the other FMS dashboards.

As far as the data goes, this version of the Dashboard contains more data than any of the other iterations of the dashboard. This is because we decided that each metric of incentivization warranted its own visualization in the Dashboard, so that users could both see all the things the FMS Advisor incentivizes and track their own progress towards each of these goals. This increases the transparency of the application with the users. This version of the dashboard also finally specifies the time interval over which the advisor checks and calculates these numbers. I also added a section for community statistics underneath the user's personal statistics. This is an early foray into the competitive gamification of the Advisor and can easily be adapted to include more community aspects. With all of these things in consideration, the Advisor's dashboard is now an effective information gathering tool for the users of the app, providing relevant information and emphasizing the important features of the application.

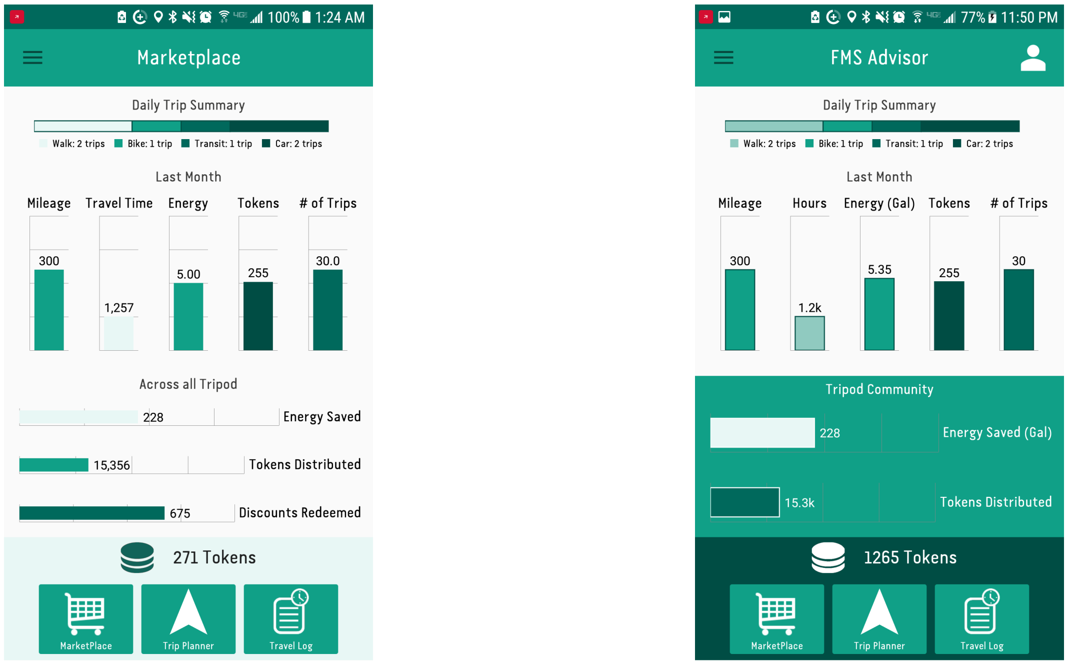


Figure 4-4: Third iteration of the Dashboard interface. This iteration was the most information rich of the 3 designs. These are two alternate color schemes for this version of the dashboard.

4.1.3 Implementation

The Final Dashboard interface was implemented using the MPAndroidChart library and some of the new material design concepts in Android. The screen is housed in a fragment that encapsulates all of the data grabbing logic and can be housed anywhere in the application. Currently, it is tied to the main Trip Planner Screen, where it resides as a bottom sheet that can be pulled up when the user needs to view the information. This interface was demoed to real users at the 2018 ARPA-E summit, where it was met with positive feedback.

4.2 Trip Planner

The main screen of the FMS advisor is the first stage of the Trip Planner workflow and contains the bulk of the functionality in the application. It contains an interface for the selection of departure time, origin, and destination, one for choosing the appropriate travel alternative, and one for the trip preview and navigation. In addition to the interfaces, the Trip Planner workflow also uses a series of algorithms and backend storage to provide full functionality to the user. Once the user requests a trip, the app queries a database updated with real time traffic data to get all the options for route that are available at the specified time, as well as an energy estimate for each one. Once these routes are retrieved, the 16 best are chosen using the User Optimization algorithms that consider a series of preferences specific to the user. These trips are then assigned a token value based on which ones the user is likely to choose, and displayed to the user. Once the user chooses a route, the choice, along with the entire menu of alternatives, is recorded and sent back to the server, which then updates the user's personalized preferences based on their choice of trips. While user is on the trip, the smartphone records their location as precisely as possible, and once the trip ends, the GPS traces are sent the backend for validation of all the metrics of the trip. Once the results of the validation are available, they are posted back to the user, and tokens are rewarded. The algorithms behind this workflow are discussed in Chapters 5 and 6, while the overall is visualized in Appendices A-2 and A-3. The rest of this section will focus on the interfaces of the workflow.

4.2.1 Interface Design

The interface of the trip planner has gone through a number of different revisions, all centered around emphasizing different elements to the user. The previous iteration of the trip planner looked very similar to a basic list view application. As described by Jamar Brooks in his 2017 thesis [3], the trip planner interface implemented during previous terms focused on minimalist user interface and clean aesthetic appeal. While the approach worked pretty well, as the project grew, more factors needed to be considered when designing the user interface for the planner. The previous iteration's biggest accomplishment was separating the interfaces for entering the origin, destination, and departure time for the trip, and viewing the actual alternatives for the trip. The original interface for the trip planner collapsed the entire work flow

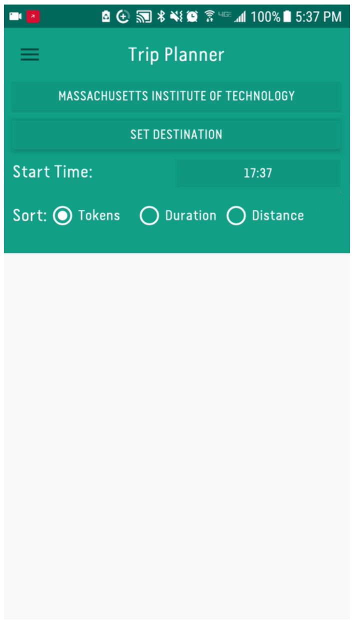


Figure 4-5: The existing interface for the Trip Planner. Notice how the only way to input arguments is through the text.

into one screen. Though it technically helped with the efficiency of the application, having everything on one screen was confusing and made the screen really cluttered and difficult to read. Splitting things up afforded more space to help the aesthetics and the usability of the interfaces, as well as allowing users to move through the flow of the planner more naturally. However, there was still room for improvement on top of the changes made before.

The main flow of the trip planner consists of the following steps: entering an origin and destination, selecting a travel alternative from the options presented by the planner, previewing the selected trip's route and turn by turn directions, and finally, navigating through the trip itself. The current version of the trip planner only offered one way to select the origin and destination for a trip, as it forced the user to click on the origin button, search for the start location, and select it from the options presented by the search. While it presented a number of attractive features, this approach to the interface failed to take advantage of a number characteristics inherent to problems of this nature. One of the largest of these is the fact that users generally want to plan a trip from their current location to somewhere else, and the

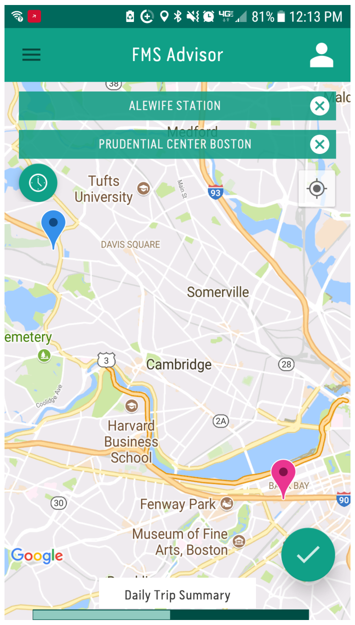


Figure 4-6: The new Trip Planner interface. There are now more ways to specify the origin and destination.

current interface lacks a method to specify current location. In addition to that, the search function used to find the places offered results from all over, not just the area immediately around the user. These two shortcomings meant that users were forced to take extra steps to click on this search button, enter a location that was close to their current one, and sift through potentially irrelevant results in order to start the Trip Planner flow, not to mention the whole process would be impossible if the user didn't know the name of their current location, or if the places search couldn't locate them properly.

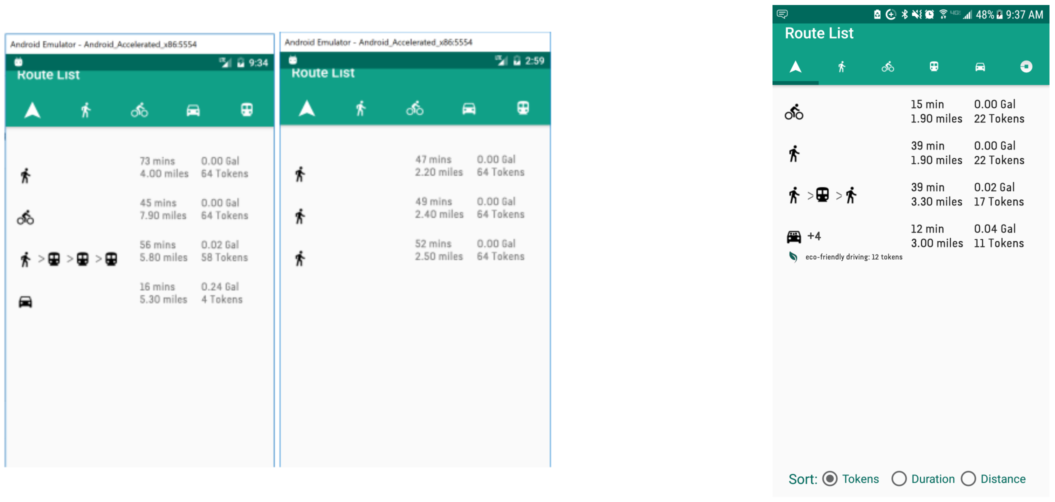
In order to fix these shortcomings, the main interface of the trip planner was transformed into an interactive map. Many contemporary trip planning interfaces, like Uber, Lyft, Google Maps, and others, all use maps as their primary interface, mainly because it reduces the amount of overheard users must go through to arrive at the main interface of the application. Now, upon entry into the trip planner flow, the users would find the origin automatically populated with their current location, reflected by a pin on the map. The color of this pin matches the color of a button that contains a text representation of the user's current location. This can be tapped to open a Google places search, in case the user wants to start the trip from somewhere other than their current location. Entering the destination can now be done in two ways as well. The old places search can still be performed, but the user can tap on the map to drop a pin anywhere they wish. These changes to the flow of the input for the trip planner benefit the user's efficiency in a few different ways. First and foremost, they reduce the amount of network usage by the application for this operation. Now, the user can find their origin and destination and start the user optimization process after only having loaded the map. Previously, the user would have had to go through two autocomplete transactions before being able to move on. In addition to this, moving the interface onto an interactive map increases the aesthetic appeal of the interface, offering the user more information about their surrounding area and allowing them to fine tune the exact destination of the trip.

4.2.2 Interface Implementation

The implementation of this interface relies on support libraries provided by google to allow the usage of its features other applications. The main map interface is a support map fragment from the Google Maps android library. This map is rendered in full view, with the controls and logo visible to the user. In addition, the view port given by the map is used to bias places search results to the location present on the screen. This means that the user is less likely to receive irrelevant places results from the searches performed by the buttons. The dashboard is loaded as a nested bottom sheet, a nice feature of the newer material design initiative started for android mobile devices.

4.2.3 Route List and Trip Preview Interfaces

The subsequent screens of the Trip Planner list the menu of travel alternatives for the user to select, and once a selection has been made, displays a preview of the


Android Emulator - Android_Accelerated_x86:5554

9:34

Route List

```
                     73 mins      0.00 Gal
                     4.00 miles   64 Tokens
                     45 mins      0.00 Gal
                     7.90 miles   64 Tokens
> > >                56 mins      0.02 Gal
                     5.80 miles   58 Tokens
                     16 mins      0.24 Gal
                     5.30 miles   4 Tokens
```

Android Emulator - Android_Accelerated_x86:5554

2:59

Route List

```
                     47 mins      0.00 Gal
                     2.20 miles   64 Tokens
                     49 mins      0.00 Gal
                     2.40 miles   64 Tokens
                     52 mins      0.00 Gal
                     2.50 miles   64 Tokens
```

48% 9:37 AM

Route List

```
                     15 min       0.00 Gal
                     1.90 miles   22 Tokens
                     39 min       0.00 Gal
                     1.90 miles   22 Tokens
> >                  39 min       0.02 Gal
                     3.30 miles   17 Tokens
+4                   12 min       0.04 Gal
                     3.00 miles   11 Tokens
eco-friendly driving: 12 tokens
```

Sort: [✓ Tokens] Duration Distance

Figure 4-7: The two left screens are the previous implementation. The new interface on the right adds only a few indicators to the driving options, in order to accommodate new features.

route. The user can either accept the alternative at this screen and begin navigation, or return to the route list and choose another alternative. These two interfaces have not changed much. The tabbed layout is a standard way to display routes in trip planning applications. The only real alterations are the eco driving and the occupancy indicators for car trips, shown in Figure 4-7. These are shown for trips that have the potential to save tokens with eco friendly driving practices (more on this in Chapter 6.4), or when the trip has an occupancy of greater than one. This indicates a rideshare trip and activates the occupancy validation detailed in Chapter 6.5.

4.3 Travel Log

The FMS Advisor's main functionality offering is the ability to plan trips and optimize energy usage, and in order to accomplish that goal, users have to plan and complete a large number of trips. The users can then look at some historical data on the Dashboard, in order gauge their most recent habits, but the dashboard ony provides an overall statistical view into that information. If users really want to know detailed information about their past trips, the travel log will provide it. The travel

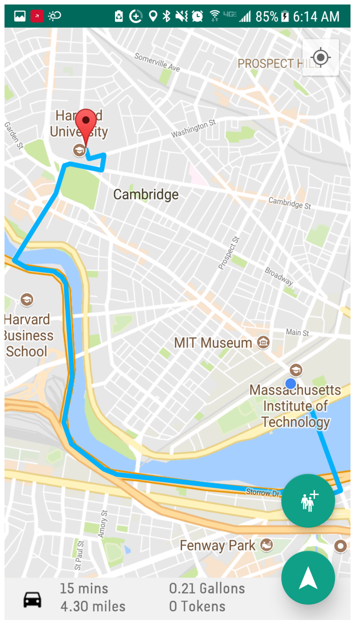


Figure 4-8: This is the screen for users to confirm their travel alternative selection. It gives a preview of the trip route, along with the other trip details across the bottom.

log lists all of the users' past trips in a friendly, readable interface that displays all of the relevant information about each trip. Trips are readable here once they have been completed and submitted to the validation process.

4.3.1 Interface Design

The Travel Log's previous design was very functional and displayed all the information users needed in a clear way, without any extra flair. All of the information was presented in a single list, with each trip represented in its own block. These blocks contained a map with the route specified by the trip, the mode of the trip, and the energy and token values as well. At the time, this was all the information needed for the log, so this interface was perfectly adequate. Unfortunately, this version of the interface is very pleasing to the eye, and also did not leave much room for adding new components. In the current interface, users had no way to know about the validaiton results from the block displayed.

The new version of the Travel Log addresses these concerns. The aesthetic

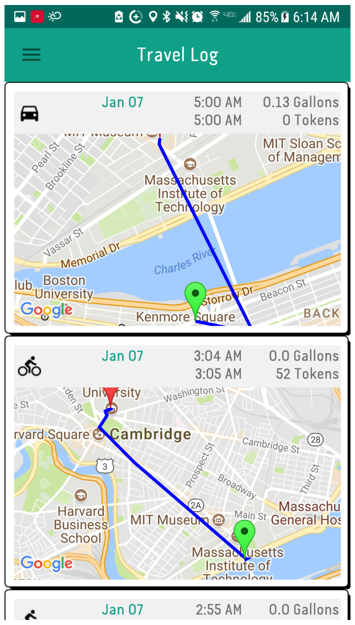


Figure 4-9: The existing Travel Log interface.

appeal is based off of trip histories in other applications similar to ours. In keeping with the design of the rest of the application, the shadows and rounded edges were removed in favor of block colors and sharp edges. In addition to this, the amount of data that needs to be stored in each log has increased quite a bit. Trips now include information about driving style, the tokens related to driving style savings, the validation results for each different metric used int eh validation process, and even new modes of transit.

All of these new metrics are rolled into the redesigned units of the travel log list. The mode icon and trip start and end times are now in the font of the application, while the main information bar is underneath the map. The eco-driving indicator from the route list can be found here as well, indicating the status of the eco friendly validation. The main validation can be found in a small check next to the number of tokens received from the trip. If the trip failed, then there is a red X instead.

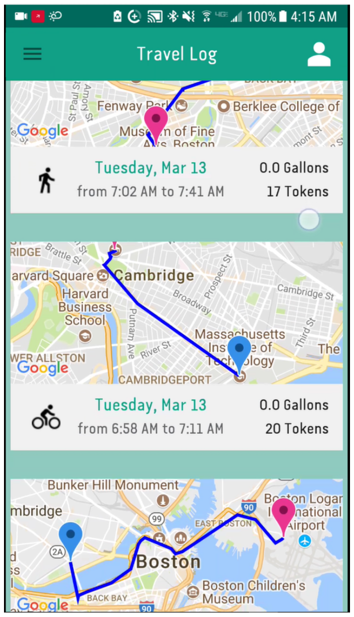


Figure 4-10: The implemented interface for the Travel Log. The increase in information required a redesign to effectively convey all the necessary components.

4.3.2 Implementation

The Implementation for the Travel Log relies largely on the RecyclerView class in android. The RecyclerView's ability too contain custom views makes holding and displaying the information about the trips very straightforward, and since the main list is a RecyclerView, we can use it to add new trips relatively easily. The map class required a few optimizations in order to make sure that the list was still performant. First, the full maps cannot be loaded, as they are too large and too memory intensive to each have all of the advertised full functionality. The 'lite-mode' option provided in the code for the map fragments converts the maps to bitmap images which greatly enhances the performance of the list. In addition to this, using a RecyclerView means that we can reuse the maps as we go down the scroll list. This saves memory for the application as well.

4.4 Marketplace

Once users of the FMS Advisor complete trips, they are awarded tokens upon successful validation of the trips. These tokens are at the heart of the optimization of the TRIPOD system, as they exist at the crux of the methodology aimed at reducing energy consumption: if users are effectively and optimally incentivized, then they will take the routes required by the system. The optimality of the token incentives is handled by the DynaMIT system discussed in later sections, but the efficacy of the incentives is handled by the Marketplace. The Marketplace offers a variety of goods and services in exchange for the tokens earned through completing trips. These goods and services offer tangible rewards for following the rules of the Advisor , and the effectiveness of the incentive policy hinges on the fact that these rewards are attractive enough that users will respond positively to the offers. Part of this is based on the actual goods and services offered in the marketplace, but some of it comes from how the rewards are presented and the usability of the interface through which users interact with them.

4.4.1 Interface Design

The existing interface for the FMS Advisor Marketplace was based on the findings of a small design study done on mobile app marketplaces across a variety of platforms. The results of this study indicated that most marketplaces split their goods into multiple sections, with the majority of the details of each item present on a separate screen after the user chooses an item. This makes sense for a few reasons. First, there isn't enough room in the list to effectively display all the information about an item; the interface would get too cluttered and unreadable. Second, having two screens to confirm the selection of a reward increases the safety of the interface from unwanted taps.

The actual layout of the interface consists of multiple tabs with large, blocky

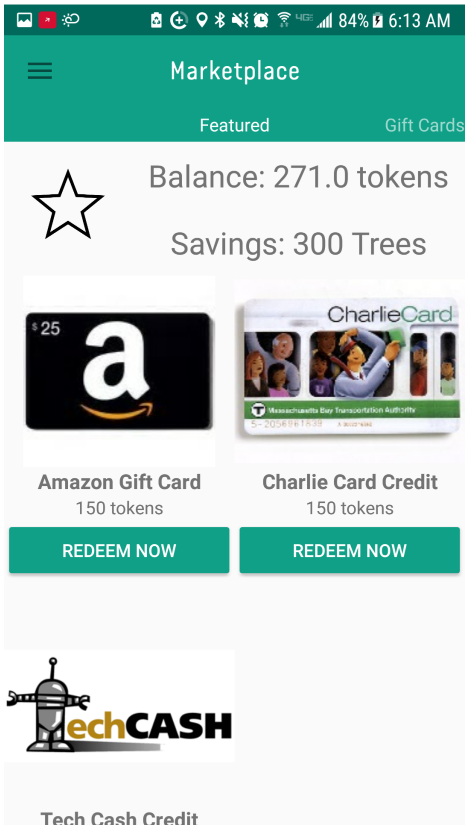


Figure 4-11: The existing interface for the Marketplace. Some of the UI elements are distracting and do not add any value to the interface.

cards representing each item available. The Tabs organize all the items by category, and can change their labels quickly and easily. This gives the old version of the interface adaptability and readability. Unfortunately, the extra tabs make it hard to view different types of items at once. In addition to this, the UI elements are rather clumsy and confusing, like the star, for example. This version of the interface also has no notion of transactions or any sort of wallet. These short comings are addressed in the new interface implementation.

The new version of the interface borrows a number of elements from modern marketplace applications, including the Google Play Store and the Apple App Store. Both of these marketplaces open with a page full of summaries. This principle is effective because it shows the user a large breadth of goods on first load, and allows more information gain upon further interaction with the interface. This principle is

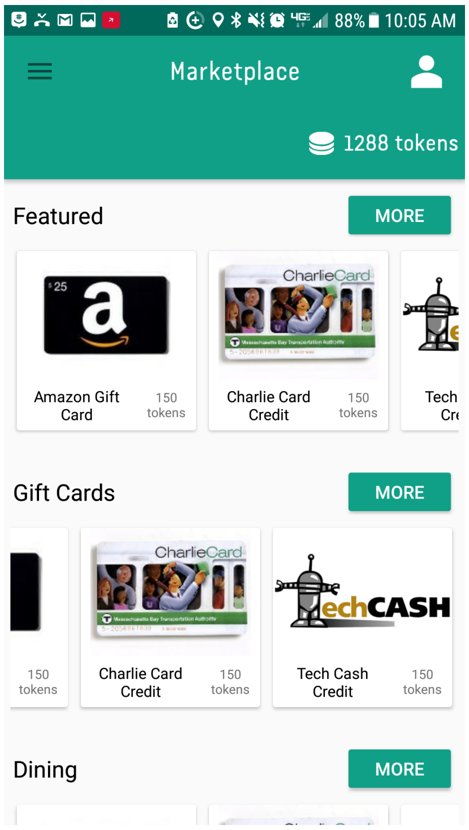


Figure 4-12: The redesigned interface for the Marketplace. The different sections on the first page allow more information to be absorbed by the user on first glance.

followed through in the new Marketplace interface, where the landing page is comprised of multiple sections with different content in each. Each of these takes the user to a separate tab with more goods of the same type. As the Marketplace grows in size, this will allow further customization of the methodology through which the products are displayed.

4.4.2 Implementation

The new interface was implemented using a series of RecyclerViews that contain RecyclerViews. I used a modular approach that allows one to specify a few different arguments to put in the ViewHolder. This allows the interface to render a second, horizontal list in the item slot, or another singular item, depending on what parameters are specified. The whole thing is wrapped in a View Pager that allows the interface to have separate tabs, and each tab has its own different set of rules. These can be customized depending on the layout desired for that type of market.

Chapter 5

Algorithms for Trip Planner

As the main source of functionality for the FMS advisor, the trip planner workflow also housing all of the substantial algorithms present in the application. These algorithms are what make the advisor special and differentiates it from other trip planning applications on the market. By utilizing the choices given to us by each user, as well as historical traffic data running through a behavioral model optimizer, the Advisor can effectively personalize all of the travel alternatives show to the user and incentivize each user with the optimal number of tokens to reduce the overall system wide energy consumption. These algorithms and backend components are split up into the System Optimization path set database, the effective trip energy calculation, the User Optimization procedure, and the Preference Updater. Each of these components has its own design and implementation details, as well as its own spot in the over framework of the Trip Planner.

5.1 System Optimization and Route Selection

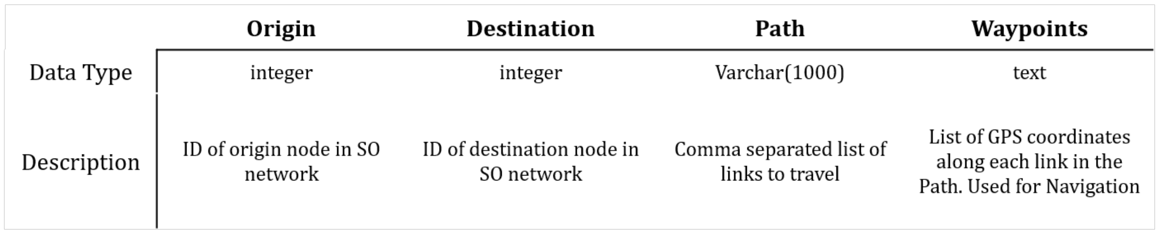
The FMS advisor boasts the ability to provide up-to-date routes for its users so they can help optimize energy usage across the whole system, and the way that it does this is through the utilization of the DynaMIT platform [10]. The DynaMIT

	Origin	Type	Distance	At_time	Travel_time
Data Type	integer	integer	decimal	integer	Decimal
Description	ID of link in SO network	The type of link	Distance in feet	The start of the 5 minute time period for which the travel time is valid	Time to travel across the link in seconds

Table 5.1: This is the format used in the MySQL table that stores all the link speeds from the updates by the SO framework.

system is a behavioral model that simulates traffic on a supply and demand basis in order to provide accurate travel times for all the links in a network over all times of day. Users from all over the system are simulated within this system, each with a demand for travel capacity, while the links across the whole system have a limited supply of this same capacity. By then simulating trips for each of these users, the DynaMIT platform can predict the travel speeds of each link in the network, which allows accurate estimation of overall trip travel times. What's really important about this, however, is that it also allows a more accurate assessment of the amount energy one might spend completing a trip that goes along certain links in a transit network. Using that estimation of energy, the optimal energy value for each token offered by the advisor can be set. This will allow us to incentivize our users such that the largest number of them will be likely to take the most energy efficient routes. These characteristics combined make DynaMIT an essential part of the optimization portion of the FMS Advisor.

In the current implementation, DynaMIT is used to simulate car trips only, but this will change in the future. The simulation is run so that the estimates for all the links are available every 5 minutes through out the day. These estimates are provided in the form of files that list all of the links in the entire network, each with the corresponding speed at the specified time. In addition to that, the optimal token value, based on the budget of tokens and the time of day is also provided. All of this information needs a way to be stored and accessed such that when a user requests a new trip, the system can effectively leverage all of the link data to provide a solution quickly. In order to do this, I made a framework to query all the data indexed on



	Origin	Destination	Path	Waypoints
Data Type	integer	integer	Varchar(1000)	text
Description	ID of origin node in SO network	ID of destination node in SO network	Comma separated list of links to travel	List of GPS coordinates along each link in the Path. Used for Navigation

Table 5.2: This is the format used in the MySQL table that stores all the paths that users can take in the System Optimization framework.

link and also time. The previous database stored the data by constructing a guidance table within a mysql instance running in the backend server. The new version takes that table and expands the dimension of the table by the time of day, allowing the user to get accurate measurements of link travel speeds for the entire 24 hour period. In addition to that, there needs to be a table that houses the paths used by the optimization to calculate the link speeds. These paths are the ones actually returned to the user when the trip planner is used and are stored in a MySQL table similar to the link speeds and is indexed on the origin column in order to speed up the searches, otherwise a simple select query took almost a minute.The formats for these tables are listed in Tables 5.1 and 5.2.

It is through this platform that the FMS advisor is able to claim that it optimizes energy savings. The optimization that happens through the usage of the DynaMIT platform provides the Advisor with real time data and link speeds, allowing it to make predictions and suggest routes that are least likely to involve energy intensive travel.

5.2 Trip Energy Calculation

The Trip Energy calculation is essential to the routes returned by the route selection algorithms and data structures. It is these estimates that are used when determining which routes are the most energy efficient, and without this calculation, the Advisor would have no way to claim that it optimizes energy for travel. Currently, the Trip Energy calculations are computed using the link distance, the link speed gotten from the system optimization, and the type of vehicle. These calculation is based on the research pursued by the TrancikLab at MIT, and more information can be found in [9].

5.3 Personalization

One of the main draws of the FMS Advisor is its ability to tailor results to a user's preferences, presenting each user with a menu of travel options suited to the choices made in the past. This is important because a user will be more likely to select a route that is more inline with their travel habits and preferences than one that is irrelevant to them. The functionality in this part of the Advisor is based on the work done by Xiang Song, Mazen Danaf, et al [14].

5.3.1 User Optimization

In order to compute and present this personalized menu of travel alternatives, the Advisor needs a measure of the user's preferences in relation to our application. These preferences are the metrics that user would use when deciding which route to take from a menu of choices and include things like the mode of transit, the number of tokens offered, the trip duration, the cost, and others. All of the specific metrics used in the Advisor are listed in Table 5.3. Each one of these metrics is represented in a utility coefficient that ranges from -1 to 1 and gives information on how useful a certain metric is to the user. For example, a coefficient of -1 means the user actively seeks to avoid that metric, while a positive coefficient indicates that the metric has an attractive quality to the user. All of these coefficients are stored in a MySQL table for easy access when the need arises.

The actual filtering itself is carried out by a matrix operation on the list of

Metric	Description
isCar	These indicate the mode of transit for the trip
isWalk	
isBike	
isTransit	
IVTTCar	These indicate the duration of the trip. (IVTT stands for In Vehicle Transit Time)
IVTTTransit	
walkTime	
bikeTime	
waitingTime	How long the user has to wait before the trip starts
numTransfers	How many transfers of mode the trip has
tokenValue	How many tokens the trip gives
carCost	These two deal with the total cost of the trip to the user.
transitFare	
earlySD	These deal with the logistics of departure time, when the system delays departure for energy optimization
lateSD	
energy	How much energy the trip consumes
occupancy	How many people are in the trip total
drivingStyle	The potential of the trip to save energy with eco driving techniques

Table 5.3: These are all of the metrics considered for each trip and choice when personalizing the FMS Advisor to each user.

routes returned by the System Optimization, detailed in the following paragraph. Once the user requests a trip, and the System Optimization framework returns a series of routes, the preliminary menu of routes must be converted into a form that explicitly considers the previously discussed metrics. Once these routes have been converted, the preferences of the user, in the form of utility coefficients for each of the metrics discussed previously, are grabbed from storage. These coefficents are then used to calculate the expected utility of each option on the menu, and then use that utility to calculate the probabilty of the option being chosen. Once these probabilities have been obtained, they are used to find the menu subset of all the routes that has the highest expected chance of a user choosing one of the routes(i.e. the expected hit rate).

One caveat to this process is that at least one option from each mode of transit must be offered to the user. This is to leave the user the option to change their preferences later on. If this condition were not in place, the user would eventually only find routes of the mode of transit most preferred, assuming there were enough routes to accommodate that.

In addition to filtering the routes returned by the System Optimization, the preferences are used to assign token values to each of the routes. The process is similar to route filtering procedure, but its applied to incentivizing the user instead. The utility of all the routes is again calculated, and the probability of each route computed. However, this time, these probabilities are used to compute a reference energy for the menu. This reference energy is the value used to compare all the other trips to when assigning tokens; essentially, it is the baseline energy consumption for the routes returned by System Optimization. As such, the choice of what calculation to use is important. For our purposes, the expected energy consumption is used: this is the energy of each route multiplied by the probability of each route being chosen, summed over all routes.

Once this baseline energy is calculated, it is used to assign token values to all of the trips. the token energy efficiency value from the System Optimization is used for this. Not only does the System Optimization framework provide optimized travel time estimates, it also computes the optimal token per energy value for the system at any given time. In essence, it computes the energy value that will maximize the number users of the Advisor that choose alternatives off the menu. The actual token value of the trips is taken as the difference between the energy of the trip and the reference energy, bounded at 0 and multiplied by the token energy efficiency.

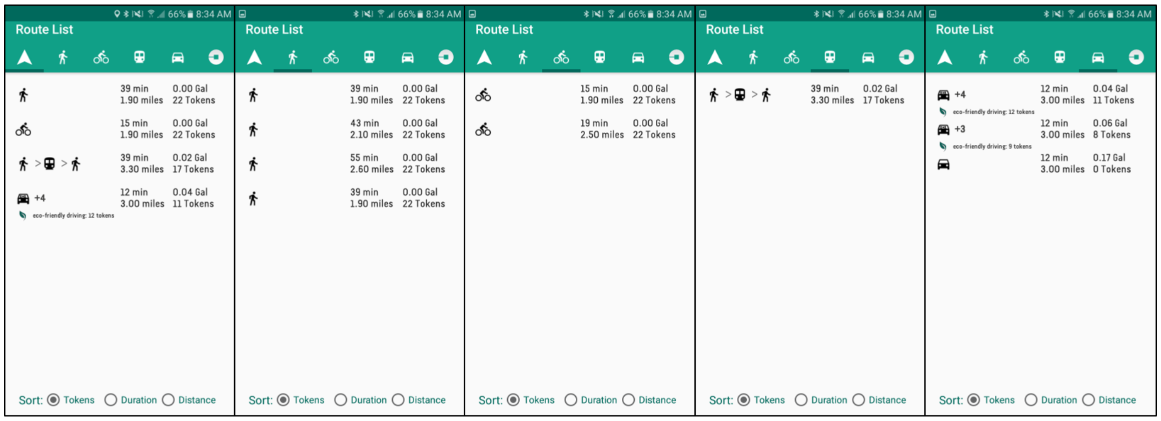
5.3.2 Preference Updater and Estimator

The User Optimization procedure takes the user's preferences and tailors the menu of options to fit their past choices. However, it does not do anything to alter or update those preferences. The actual personalization logic is handled separately by the Preference Updater and Estimator algorithms. These algorithms are housed in the backend and called every time the user makes a choice of route from the menu of options presented by the User Optimization.

The preference updater and estimator are based on the principle of Bayesian

Estimation, in which observations on an unknown model are used to infer qualities of the model. In this case, our model is the preferences of the user, and the observations are the choices of route they make from the menu presented by User Optimization. The Advisor employs the hierarchical Bayes estimator developed by Danaf et. al [5]. This estimator provides a solution to the issues of scalability in large Bayesian mixture models, but allowing each successive observation to update the model without recalculating the whole thing. In addition to this, the estimator employs the notion of individual specific parameters versus population level parameters. The approach to scalability involves using the new choices in addition to the population level parameters to provide updates to the individual parameters without needing to consider the entire history of choices. This makes the update viable to run in real time after every new choice. The population level parameters are estimated once a week using the entire population's history of choices. This estimation takes quite a long time and is not computationally efficient.

In order to properly update the preferences, the structure of the choice observations has to be chosen carefully. The goal is to capture the essence of the choice in a way that interfaces well with the Bayesian Estimator. In this case, every option must be represented as a binary choice, even if the choice is one among a range of values. The structure chosen uses all of the metrics listed in Table 5.3. Here, each characteristic of the alternative is represented in a separate attribute of the choice. First, we have whether the choice is available or not in this choice observation. Not every menu is the same size, so we have to make sure that the estimator knows how many options were in the choice. Then, we have a column for each parameter of the preferences from each alternative in the menu. Since there are 18 preference parameters, there will be $18n$ columns, where n is the number of alternatives in the menu. In addition to this, the algorithm needs to keep track of which trips are recommended by the User Optimization, so a separate column is kept for that as well.


Route List

39 min 1.90 miles 0.00 Gal 22 Tokens
15 min 1.90 miles 0.00 Gal 22 Tokens
> > 39 min 3.30 miles 0.02 Gal 17 Tokens
+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens

Sort: [✓ Tokens] Duration Distance

Route List

39 min 1.90 miles 0.00 Gal 22 Tokens
43 min 2.10 miles 0.00 Gal 22 Tokens
55 min 2.60 miles 0.00 Gal 22 Tokens
39 min 1.90 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

Route List

15 min 1.90 miles 0.00 Gal 22 Tokens
19 min 2.50 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

Route List

> > 39 min 3.30 miles 0.02 Gal 17 Tokens

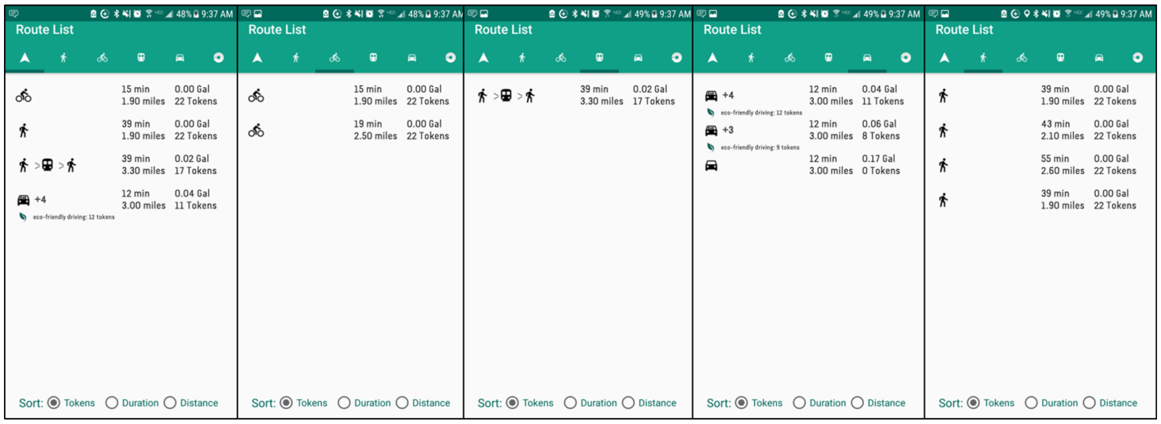
Sort: [✓ Tokens] Duration Distance

Route List

+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens
+3 12 min 3.00 miles 0.06 Gal 8 Tokens
eco-friendly driving: 9 tokens
12 min 3.00 miles 0.17 Gal 0 Tokens

Sort: [✓ Tokens] Duration Distance

Figure 5-1: The base case for the Personalization, before any choices have been made.


Route List

15 min 1.90 miles 0.00 Gal 22 Tokens
39 min 1.90 miles 0.00 Gal 22 Tokens
> > 39 min 3.30 miles 0.02 Gal 17 Tokens
+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens

Sort: [✓ Tokens] Duration Distance

Route List

15 min 1.90 miles 0.00 Gal 22 Tokens
19 min 2.50 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

Route List

> > 39 min 3.30 miles 0.02 Gal 17 Tokens

Sort: [✓ Tokens] Duration Distance

Route List

+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens
+3 12 min 3.00 miles 0.06 Gal 8 Tokens
eco-friendly driving: 9 tokens
12 min 3.00 miles 0.17 Gal 0 Tokens

Sort: [✓ Tokens] Duration Distance

Route List

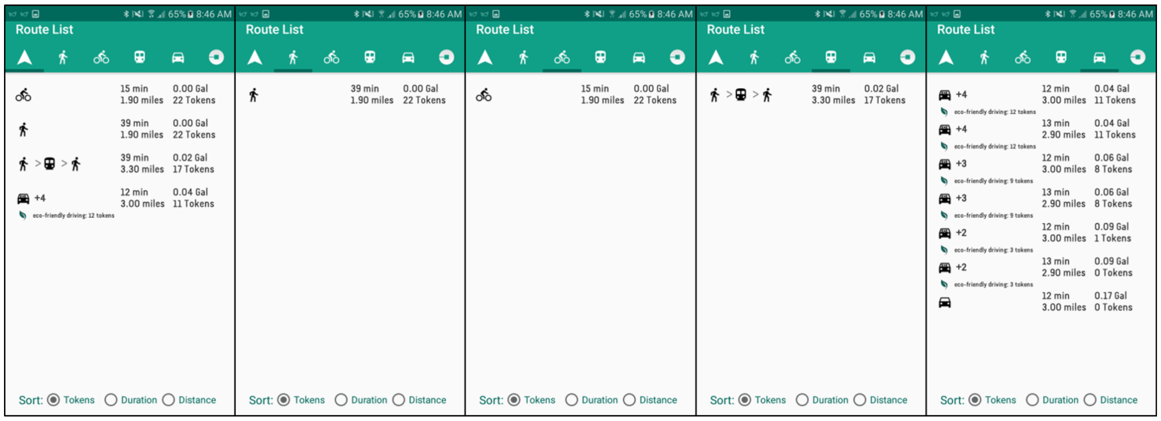
39 min 1.90 miles 0.00 Gal 22 Tokens
43 min 2.10 miles 0.00 Gal 22 Tokens
55 min 2.60 miles 0.00 Gal 22 Tokens
39 min 1.90 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

Figure 5-2: This is the menu seen by a user that favors walking trips. It's very similar to the base case because the base parameters already favor non-vehicular more than the other modes.

5.3.3 Results

The Personalization framework performs pretty well in practice. In order to test the framework, two users were initialized with the population level preferences. Then, each user went through 5 route choices to simulate a few days worth of travel. One user chose exclusively car routes, while the other chose walking routes. At the end of the simulation, the menus were quite different. The car loving user had nearly 4 times the driving options as the walking user. The menus are summarized in Figures 5-1 to 5-3.



65% 8:46 AM
Route List

15 min 1.90 miles 0.00 Gal 22 Tokens
39 min 1.90 miles 0.00 Gal 22 Tokens
> > 39 min 3.30 miles 0.02 Gal 17 Tokens
+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens

Sort: [✓ Tokens] Duration Distance

65% 8:46 AM
Route List

39 min 1.90 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

65% 8:46 AM
Route List

15 min 1.90 miles 0.00 Gal 22 Tokens

Sort: [✓ Tokens] Duration Distance

65% 8:46 AM
Route List

> > 39 min 3.30 miles 0.02 Gal 17 Tokens

Sort: [✓ Tokens] Duration Distance

65% 8:46 AM
Route List

+4 12 min 3.00 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens
+4 13 min 2.90 miles 0.04 Gal 11 Tokens
eco-friendly driving: 12 tokens
+3 12 min 3.00 miles 0.06 Gal 8 Tokens
eco-friendly driving: 9 tokens
+3 13 min 2.90 miles 0.06 Gal 8 Tokens
eco-friendly driving: 9 tokens
+2 12 min 3.00 miles 0.09 Gal 1 Tokens
eco-friendly driving: 3 tokens
+2 13 min 2.90 miles 0.09 Gal 0 Tokens
eco-friendly driving: 3 tokens
12 min 3.00 miles 0.17 Gal 0 Tokens

Sort: [✓ Tokens] Duration Distance

Figure 5-3: If a user choose car trips very often, the menu would start to look like this. Notice the abundance of car options and the fact that all the other modes still have at least one option.

Chapter 6

Algorithms for Trip Validation

In order to ensure that the incentivization of the FMS Advisor fulfills its role in the system, it is necessary to validate the actions of the user to verify that they meet the criteria for the incentives. To that end, the Advisor includes a validation framework that centers on 5 metrics provided for the users to reduce their energy consumption: route choice, mode choice, departure time, eco friendly driving, and occupancy. Each one of these metrics is independently verified, and tokens are not awarded until each trip completed by the user passes the required test. The route, mode, departure time, and driving style metrics are verified in the backend server, while occupancy is done in the mobile application. In order to get the tokens for trip, the user must pass all of the checks except for driving style, while passing the driving style check will merit the eco driving tokens.

6.1 Overall Architecture

The Trip Validation Architecture is encompassed within the Trip Planner workflow discussed earlier. It begins with the user selecting a trip from the menu of travel alternatives presented after User Optimization runs. Once the user chooses a trip and begins navigation, the first part of the Trip Validation framework activates and runs. This is the occupancy validation and is the only part of the framework that happens on the user's phone. Once the trip completes, all of the relevant metadata about the trip, as well as the sensor data collected while the trip was taking place, is bundled up and sent to the backend server. There, the rest of the trip validation framework is performed, validating the route, mode, departure time, and driving style. Once all of these have been checked and validated, the results are sent back to the user. This validation framework is what makes the incentivization feasible; if the users are held accountable, they will be more likely to perform the actions incentivized by the Advisor. As such, the validation framework is integral to the project's success.

6.2 Mode, and Departure Time Validation

In order to validate mode, route, and departure time, the framework relies on the FMS Core stop and mode detection algorithms. When a user approaches the destination specified in the trip, they enter a Geofence, which triggers the end of the trip on the mobile app. The app then bundles all of the GPS data and trip information (origin and destination pair (OD), specified route, and departure time), and sends it to the server. The server will then start a Mode and Stop detection job on the collected GPS data. This partitions the user's day into a series of structures known as intervals. Basically, each interval represents either a stop or a transit between two stops. The intervals of interest are those that correspond to the OD from the trip and the transit between them.

Once the results of the Mode and Stop detection are available, they are checked against the trip information sent in with all the data. The two stops outside the travel interval can tell us if the user started and stopped in the correct places.

First, the framework checks for stops that correspond to the origin and destination of the trip. If two stops cannot be found, in the proper order, that are within

100m of the origin and destination specified by the selected option, the trip fails.

For departure time, the origin stop must end within 5 minutes (the frequency of predictions from the System Optimization) of the specified departure time. This is because the user could have been at the stop for a long time before deciding to plan and complete the trip. This means that the algorithm needs to check the end of the stop to find out when the user actually started the trip.

The FMS Algorithms assign a transit mode to each travel interval. This assigned transit mode can be used to see if the user completed the trip using the mode required by the planner. The intervals that correspond to the are checked, and the assigned mode of each one is compared to the mode from the trip data sent in. If the mode is the same, then the check passes. This is complicated by transit trips, which are often multimodal and are comprised of multiple intervals in the FMS representation. To remedy this, the algorithm compares all the intervals in between the two stops found for the OD. This comparison is the same as detailed above.

6.3 Route Validation

The route validation is based on a partial route matching algorithm shown by Gjaldbaek [6]. The algorithm takes two routes in the form of a list of coordinates and rasterizes them to produce a series of intervals. It them computes the similarity between the two sets of intervals. The overall procedure is as follows:

1. The desired route from the trip metadata and the intervals associated with the user's actual route are taken and converted to a list of points. In our case, they are stored as encoded polyline strings, so they can simply be decoded to get the associated list of lat/long coordinates.

2. A grid is made from the bounding box of the two routes. This is to provide a set number of grid cells for the rasterization

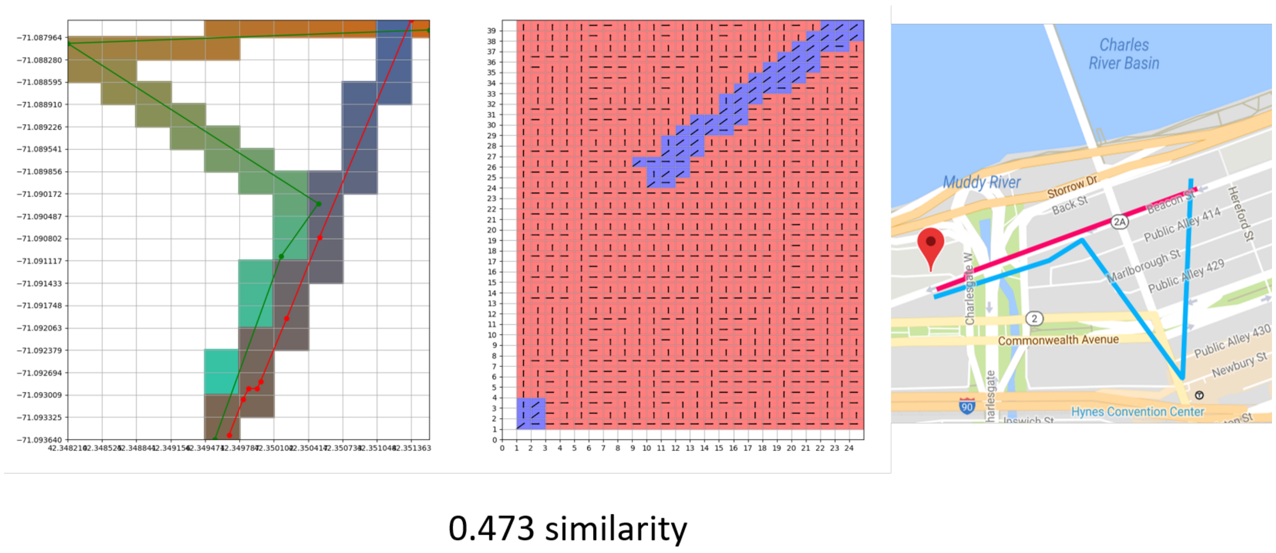
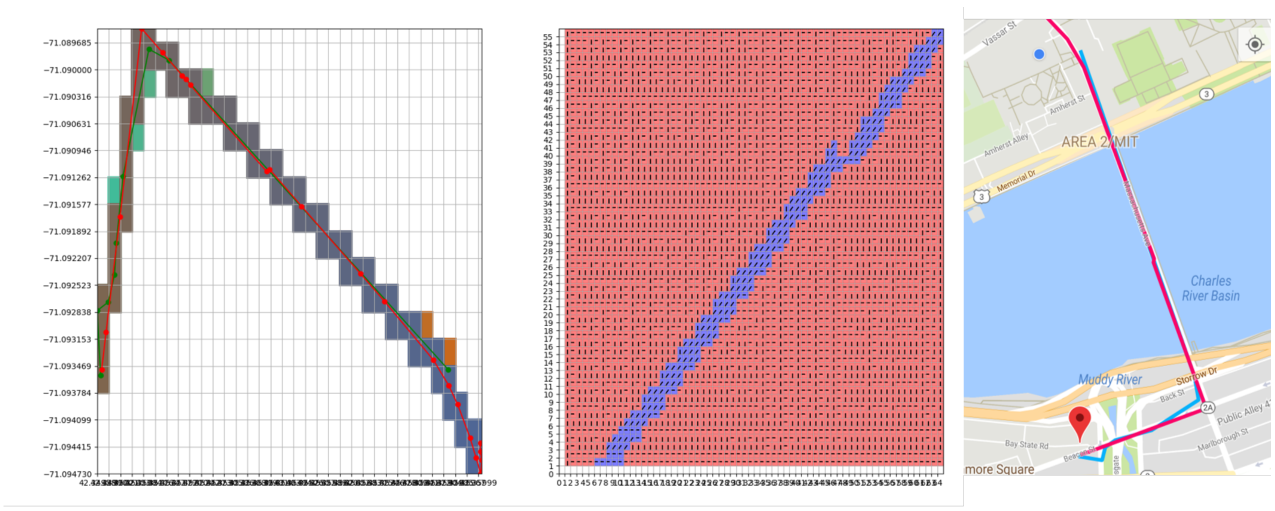


Figure 6-1: This is an example of a route in which the user deviated from the desired route. Notice the deviation shown in the rasters on the left and the lack of a complete path of matched cells in the dynamic programming matrix in the center.

3. Each route is then rasterized on this grid, producing two lists of cells through which each route passes, along with the overall distance passed by each route through each cell.

4. After obtaining a list of the rasters passed through by each route, we use a dynamic programming string matching algorithm with a tailored cost metric and proximity matching function to compute the similarity for the two routes.

This algorithm takes advantage of some of the concepts of partial string matching algorithms by converting each route into a representation similar to a string. Since both routes are rasterized on the same grid, each cell can be viewed as a letter, and the rasterized route as a word. The similarity matching algorithm basically looks at these two words and computes how similar the two words are. The main difference here is the cost function used to fin the optimal matching. For strings, the simple edit distance would suffice. In our case, the distance between the lat long coordinates, as calculated by the Haversine formula, is a more appropriate metric. In addition to this, since this is only an application of the string matching algorithm, not an actual string matching, the matching process happens requires some scrutiny. GPS data is



0.898 similarity

Figure 6-2: The routes here definitely match but not all the way due to a few artifacts from the matching process. Routes like these are an example of why the threshold was set to .85.

rather noisy, and discretizing the cells introduces the possibility for the two routes to be split along a boundary, rendering them different in the eyes of the grid. To fix this, a proximity constraint is introduced to the matching process. Practically, this is treating the cells in a 3x3 grid around the desired cell as matching, not just cells that match exactly.

The algorithm performed pretty well on the GPS traces found in the advisor. Since most of the testing is done in Boston where GPS tends to be messy, the consideration of proximity in the matching algorithm was essential in correctly matching the traces correctly. Some experimentation found that a threshold of about .85 was sufficient to avoid false negatives and false positives. Figures 6-1 to 6-3 show the results observed in a few of the trips from users of the Advisor. The leftmost box shows the rasterized routes, the middle cell shows the output of the dynamic programming matrix used to construct the similarity measure, and the right shows the two traces on the map.

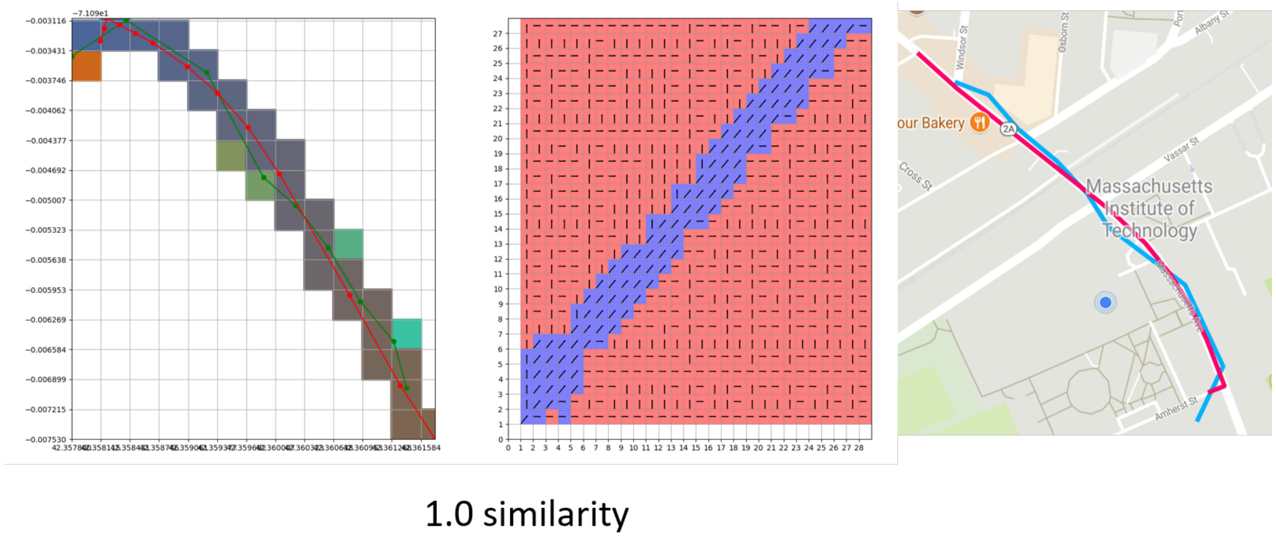


Figure 6-3: These routes matched perfectly. The dynamic programming matrix has a perfect path all the way down the diagonal and the rasters overlap extensively.

6.4 Driving Style Validation

The driving style validation aims to verify that the user has followed certain criteria during the completion of the trip and necessarily only runs on car trips. These criteria, developed by Miotti et al. in [11], include avoiding the following actions:

1. Driving at speeds over 65 mph
2. Sharp acceleration and hard brakes at higher speeds
3. Braking harshly and accelerating right afterward, like you might do in heavy traffic

These actions are proven to increase trip energy consumption, and the goal of the validation is to check whether the user avoided them while driving through the use of the GPS data provided by the user during use of the Advisor. Unfortunately, the GPS data collected by the Advisor tends to be very buggy and requires quite a bit of preprocessing before it is usable for validation purposes. The data is process according to the following procedure before moving on to assess the driving style of the user.

1. Interpolate the traces to 1 hz frequency. The data must be at 1 hz so that the profiles of interest have even datapoints and can be properly smoothed and averaged.

2. Smooth the route with a moving average. This helps remove inconsistencies in the data from jumps in the GPS.

3. If a gap in data is larger than 10 seconds, mark the data as lost. This is necessary to avoid unrealistic profiles by not interpolating data where there isn't any.

4. Convert the GPS coordinates to speed profiles, using the distance gotten from the Haversine formula, divided by the 1 hz frequency.

5. Convert the speed profile into an acceleration profile by taking the difference of values, and convert the acceleration profile into a jerk profile with the same method.

6. Remove an outlying, illogical, or impossible values in the speed and acceleration profiles, taking care to make sure the profiles stay the same length.

Once all of the data has been successfully processed, three indicators, one for each of the previously discussed metrics, can be calculated. The values gotten from these metrics are compared against distributions of numbers that are associated with energy savings. These distributions were calculated by analyzing many speed profiles gotten from the GPS data of FMS users.

The first metric deals with how long the user spent driving over 65 mph. In order to calculate this, the area between the parts of the speed profile curve that are above 65mph and the 65mph threshold is divided by the length of the whole trip. The second metric does the same thing, but calculates it for the acceleration profile and a threshold linearly dependent on the speed, again dividing by the length of the trip. The third metric uses the jerk profile. We smooth the jerk profile over 6 seconds, and then take the area of the curves where the jerk is above .15 m/s, where the speed is between 4 and 17 m/s.

6.5 Occupancy Validation

The Occupancy validation is the only part of the framework that happens on the phone itself, as well as during the trip. Occupancy is only considered for carpooling trips, since these are the only ones that require more than one person and also use up energy. There are a few different ways this check could be performed. The one most in line with the rest of the framework, albeit a bit indirect, is to take the traces of the two individuals on the trip together, and run the route matching algorithm on them. If the similarity is high enough, then the check is considered passed. The reason this isn't done is that it requires a measure of infrastructure not yet present in the FMS Advisor, and also that it is a post event check. This is currently handled by connecting the two riders' phones through bluetooth. This way, the proximity of the two users can be checked directly during the duration of the entire trip. Unfortunately, this does drain the battery of the phones more than the other method would.

Chapter 7

Next Steps

Compared to the state it was in one year ago, the FMS advisor has made significant strides. The interfaces have all been updated and missing functionality has been filled in. The preference updater and User Optimization algorithms are integrated and functioning. There is a trip validation framework working from end to end, validating trips on most of the metrics of the framework. The integration loop with the System Optimization framework has been closed, and the framework is runnable for simulations. All of these things have pushed the Advisor towards the final vision for the application.

Despite all the progress that has been made on the Advisor, there are still many things to be done. The entire framework does not run end to end without bugs, and some functionality is missing. For example, the System Optimization algorithms currently only considers car routes, while the Advisor incentivizes 4 different modes of travel. In addition to this, many of the algorithms are found in very disjointed places. The preference updater algorithms are not on the same server as the Trip Validation algorithms, while the latter algorithms aren't complete or optimized properly. Separate from all of these things, the Advisor also lacks a proper method to navigate the user from origin to destination. All of these things, and even more, remain to be done before the Advisor is ready for live user testing.

The FMS Advisor constitutes a unique look at the problem of personalizing a framework for its users. While the concept of personalization is nothing new, it has not been widely applied in the field of transportation. When one considers the added dimension of the incentives, the value of the Advisor becomes even more apparent. The advisor presents an optimized way to incentivize users to change their behavior. It is currently applied to travel, but the interfaces could easily be adapted to any other sort of problem. For example, if one wanted to track user's consumerism, to encourage more intelligent flow of capital, the advisor could be adapted here. The choice model and preferences would be centered around types of stores and other sellers, while the validation algorithms could be a linked bank account statement or something of that nature. One could even apply incentive alignment for the Marketplace, giving users discounts at stores according to their displayed preferences. This is only one example of how the Advisor framework could be applied to other projects.

In general, the work completed on the FMS Advisor has helped to bring to fruition the vision of an optimized application that grows with its userbase. Applied to the realm of transportation with a goal of reducing energy consumption during transit, the Advisor attempts to solve a problem in a way not done by any other applications on the market, and by virtue of the functionality achieved by the work discussed in this thesis, I believe it succeeds in many ways. Regardless, there is always room for improvement and further development, and I hope that this thesis will provide a strong foundation for all the future work to come.

Appendix A

Application Flow

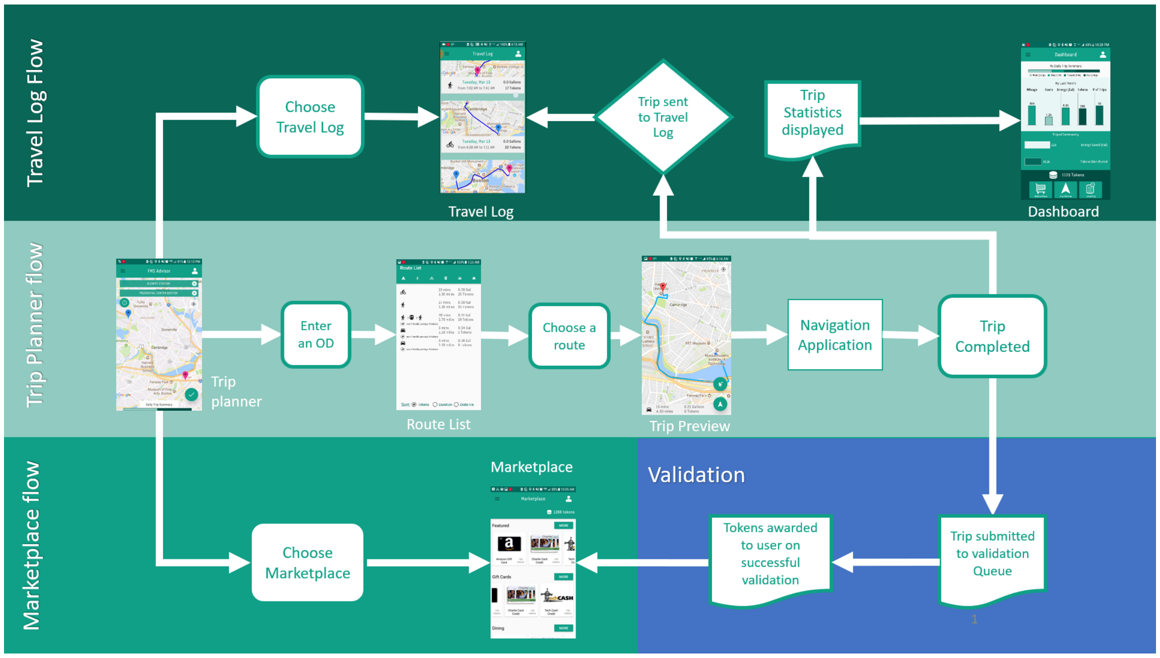


Figure A-1: This figure shows the overall user interface flows within the Advisor.

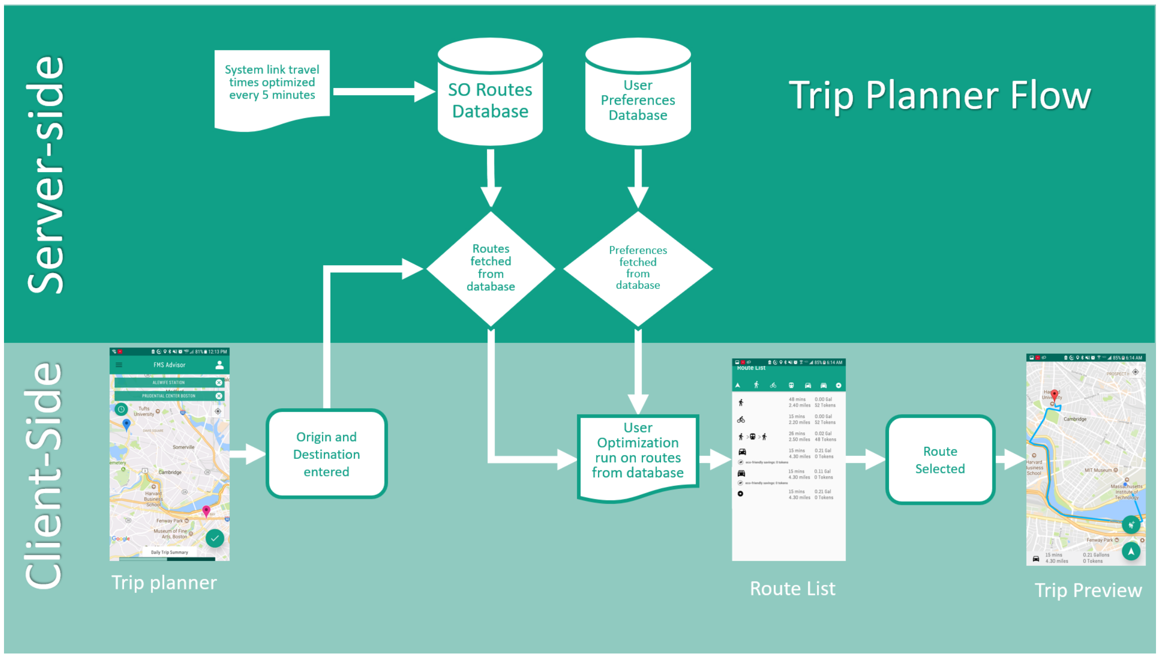

Server-side
Client-Side
Trip Planner Flow
System link travel times optimized every 5 minutes
SO Routes Database
User Preferences Database
Routes fetched from database
Preferences fetched from database
Origin and Destination entered
User Optimization run on routes from database
Route Selected
Trip planner
Route List
Trip Preview

Figure A-2: This diagram shows the logic and data flows in the Advisor for the Trip Planner until the trip starts.

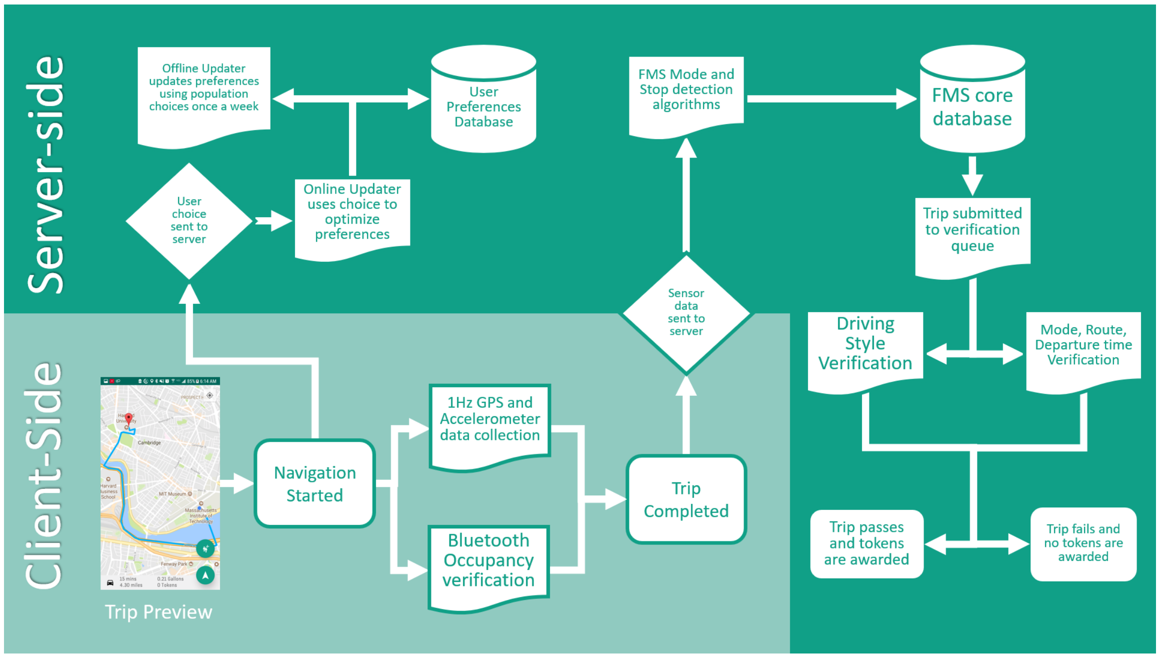

Server-side
Client-Side
Offline Updater updates preferences using population choices once a week
User Preferences Database
FMS Mode and Stop detection algorithms
FMS core database
User choice sent to server
Online Updater uses choice to optimize preferences
Trip submitted to verification queue
Sensor data sent to server
Driving Style Verification
Mode, Route, Departure time Verification
1Hz GPS and Accelerometer data collection
Navigation Started
Trip Completed
Bluetooth Occupancy verification
Trip passes and tokens are awarded
Trip fails and no tokens are awarded
Trip Preview

Figure A-3: The logic and data flows in the Advisor for the Trip Planner from when the trip starts until when the validation framework finishes.

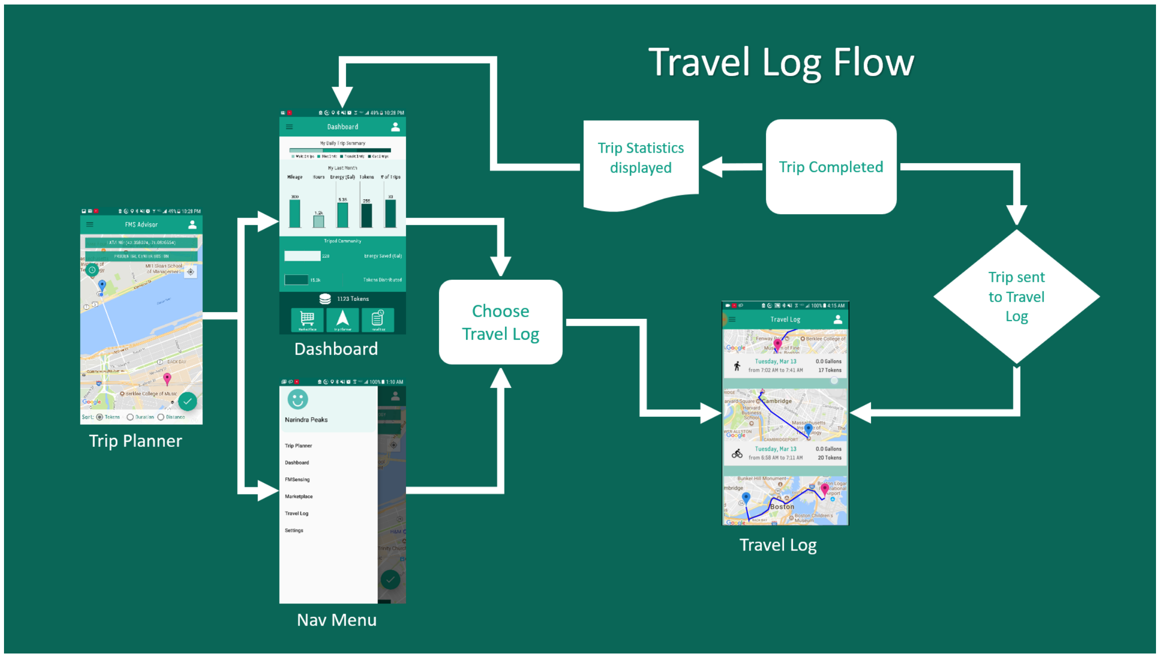


Figure A-4: The logic and data flows in the Advisor for the Travel Log and Dashboard.

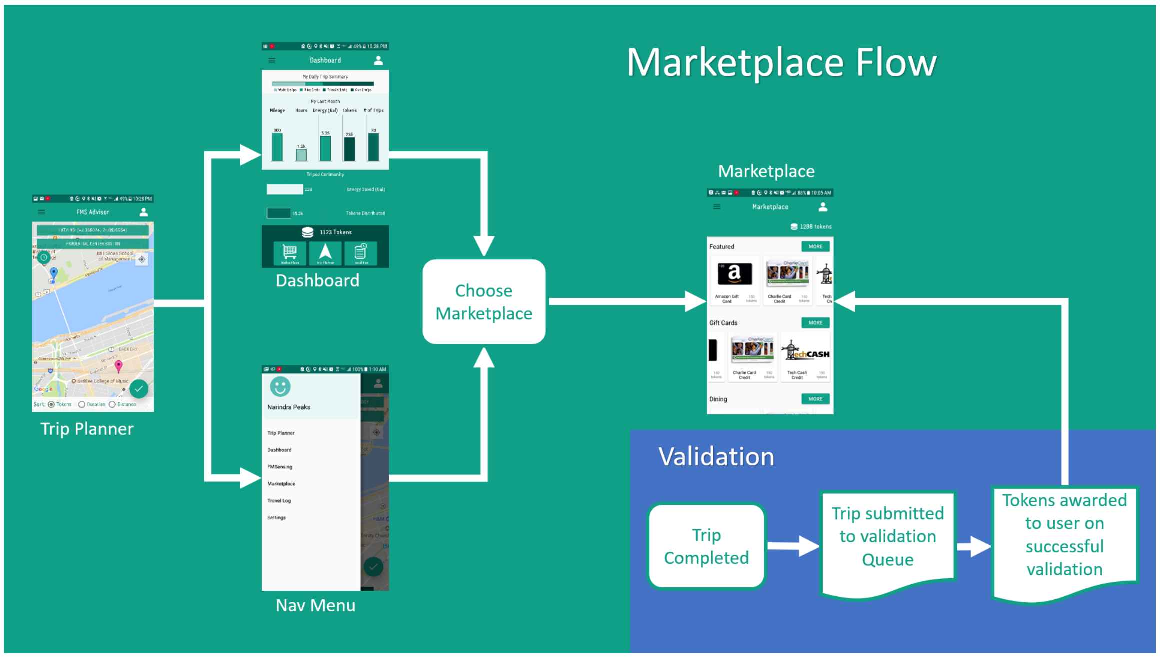


Figure A-5: The marketplace flows, including the distribution of tokens after trip validation.

Bibliography

[1] Carlos Lima Azevedo, Ravi Seshadri, Song Gao, Bilge Atasoy, Arun Prakash Akkinepally, Eleni Christofa, Fang Zhao, Jessika Trancik, and Moshe Ben-Akiva. Tripod: Sustainable travel incentives with prediction, optimization, and personalization. Technical report, 2018.

[2] Christine Bauer. On the (in-) accuracy of gps measures of smartphones: a study of running tracking applications. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, page 335. ACM, 2013.

[3] Jamar Brooks. *Mobile interface for mobility incentives schemes: FMS-Advisor*. PhD thesis, Massachusetts Institute of Technology, 2017.

[4] Caitlin Cottrill, Francisco Pereira, Fang Zhao, Iněs Dias, Hock Lim, Moshe Ben-Akiva, and P Zegras. Future mobility survey: Experience in developing a smartphone-based travel survey in singapore. *Transportation Research Record: Journal of the Transportation Research Board*, (2354):59–67, 2013.

[5] Mazen Danaf, F Becker, Xiang Song, Bilge Atasoy, and Moshe Ben-Akiva. Personalized recommendations using discrete choice models with inter-and intra-consumer heterogeneity. In *International Choice Modelling Conference 2017*, 2017.

[6] Martin Gjaldbaek. Practical polyline matching for gps data. Master's thesis, Technical University of Denmark, 2010.

[7] Giuseppe Guido, Alessandro Vitale, Vittorio Astarita, Frank Saccomanno, Vincenzo Pasquale Giofré, and Vincenzo Gallelli. Estimation of safety performance measures from smartphone sensors. *Procedia-Social and Behavioral Sciences*, 54:1095–1103, 2012.

[8] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1609–1615. IEEE, 2011.

[9] James McNerney, Zachary A Needell, Michael T Chang, Marco Miotti, and Jessika E Trancik. Tripenergy: Estimating personal vehicle energy consumption

given limited travel survey data. *Transportation Research Record: Journal of the Transportation Research Board*, (2628):58–66, 2017.

[10] Martin Milkovits, Eric Huang, Constantinos Antoniou, Moshe Ben-Akiva, and Jorge Alves Lopes. Dynamit 2.0: The next generation real-time dynamic traffic assignment system. In *Advances in System Simulation (SIMUL), 2010 Second International Conference on*, pages 45–51. IEEE, 2010.

[11] Marco Miotti, Zachary A Needell, and Jessika E Trancik. Quantifying reductions in personal vehicle energy consumption due to driving style changes. Technical report, 2018.

[12] J Nielsen. Usability heuristics for user interface design. nielsen norman group. 1995, 10.

[13] University of Maryland. incentrip, 2015. https://incentrip.org.

[14] Xiang Song, Mazen Danaf, Bilge Atasoy, and Moshe Ben-Akiva. Personalized menu optimization with preference updater: A boston case study. *Transportation Research Record*, page 0361198118758674, 2018.