Fast Track

Abdulghafar Al Tair (atair@princeton.edu) - project leader
Bilal Mukadam (bmukadam@princeton.edu)
Mahd Aamir Khan (mahdk@princeton.edu)

Table of Contents:

1. Overview
2. Requirements and Target Audiences
3. Functionality
4. Design
5. Timeline
6. Risks and Outcomes

1- Overview:

Fast Track seeks to provide an easy-to-use interface for users wanting to use the Princeton TigerTransit shuttle system. We will implement both a web application, which will scrape data from the Princeton "Routes and Schedules" page and use the real-time location of the buses provided by TransLoc to provide the most optimal i.e. fastest path from the source to the destination. The system will allow the user to view the bus or (series of buses) to take to get to their desired location.

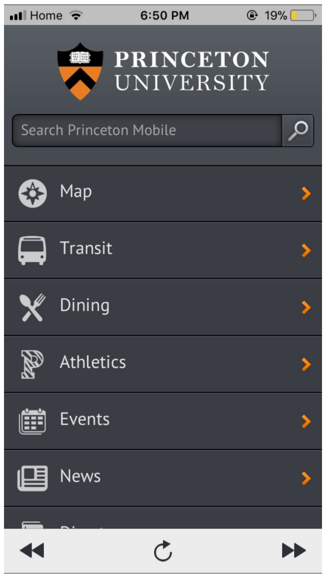
The primary focus of the system is optimization and user-friendliness. It utilizes an underused resource. A critical component of the system is the User Interface. The UI has to be implemented in a manner that allows usage from people of all backgrounds, regardless of their tech-savviness.

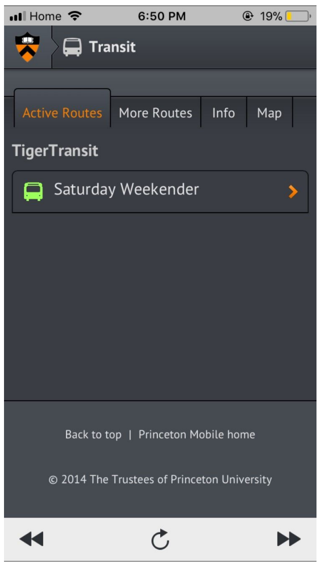
We hope that Fast Track will allow users to reach their destinations swiftly, so that they have more time to utilize on their research or studies. The system will be implemented during the semester and will see usage all-year round, including the summer.

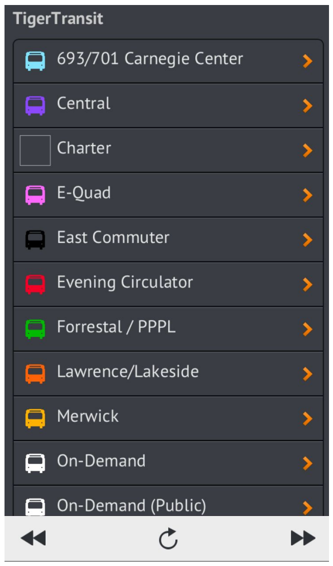
2- Requirements and Target Audiences:

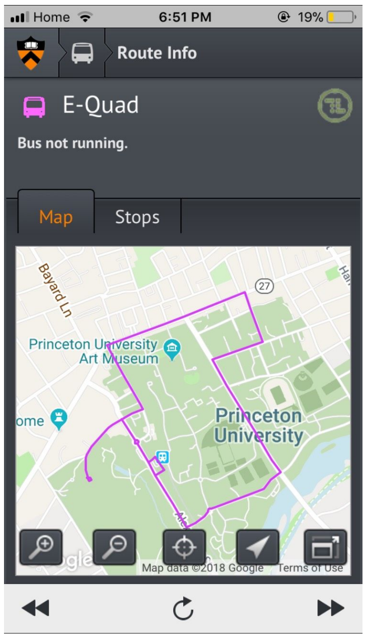
Problem:

The main problem we are trying to solve is transportation on campus and how it can be better used to make students' lives easier. Princeton has put out many solutions throughout the years to solve this issue like Zagster bike share and TigerTransit, but the matter of the fact is that most students still prefer to walk around campus rather than utilize these resources. We specifically believe that TigerTransit is a great resource but is unfortunately underutilized due to its complicated user interface and the clutter of information that it puts out to you. It is simply counter-intuitive and in an age where people (especially Princeton students) want information readily in their hands, the current implementation does not make the cut.









[1]

Intended Users:

Our intended users are mainly Princeton students both undergraduate and graduate students. Princeton graduate students currently utilize the bus system more since it takes them from Lakeside apartments to main campus but what seems to be happening is that they memorize the

[1] Images from the Princeton University iOS app

routes and the schedule so we want it to make it easier for them to use this system. We also believe that undergraduates can benefit a lot from Tiger Transit and we believe that producing a tool that simplifies the process of looking up the bus schedule will definitely increase the number of students that take the bus.

Existing solutions:

Transloc Rider app: TransLoc company currently has an application that provides a real time updated map of the bus locations. It can also be used to see how long it would take for a bus to come to a specific pick up point. We believe that our system will be better for the following reasons:

- Comparisons between different transportation systems. Walking v.s. taking the bus
- Pick up location comparisons (Is it faster for me to walk to the E-quad and take the bus from there or wait next to the Woodrow Wilson school)
- TransLoc only provides a map of the buses in real time and then the user has to figure out how long it will take for the bus to arrive, we will provide textual data, more information on what available choices there are, and a simpler UI.

3- Functionality:

Features:

- User authentication through the CAS system
- Source & destination input fields to specify user's desired trip
- A map UI (using google Maps API) that would display to the user the exact route to be taken. Would let the user know where to walk to catch the bus, where exactly the bus would drop the user off, and if the user would have to walk any further after being dropped off in order to reach their destination
- An estimated trip duration and an ETA (expected time of arrival) if the user were to take the route suggested by the system and importantly if the trip duration would be shorter than simply walking the entire length of the route
- Advanced features that could eventually be added:
    - Saved destinations. If a user will be frequently trying to reach a certain destination, they would be able to 'save' that destination
    - A game users can play while waiting for the TigerTransit shuttle

Sample Use Cases:

Scenario 1: Grad student Joshua has ten minutes to make to Peyton Hall from the Grad College:

Joshua has been busy toiling away at his doctoral research and has some interesting results to report to his adviser. Unfortunately, having not gotten much sleep last night, he has missed his alarm and has woken up late. He now has ten minutes to get to Peyton Hall to meet with his adviser. He quickly opens up this app, specifies that his current location as the grad college on the "source" prompt and his desired destination (on the "destination" prompt) as Peyton Hall. Within a few seconds, the system lets him know the following:

- The E-Quad bus line will be arriving within a minute directly outside the Grad College
- The bus would take an estimated 8.5 minutes to reach its Frist Campus Center bus-stop, the closest stop to Peyton Hall
- Joshua would then have to walk for roughly a minute from Frist to Peyton Hall
- The estimated total trip duration would be 10.5 minutes, roughly half the expected time it would take him if he were to walk the entire way

Joshua grabs his bag, dashes out his room and clambers onto the arriving bus. Within 10 minutes, he has arrived at his destination.

Scenario 2: It is raining outside and Emily wants to go from her dorm in Whitman to the Friend Center, she knows there are TigerTransit buses running but does not know when the next one will come:

Instead of having to go through multiple pages of the Princeton app, trying to figure out which bus she wants to take, then locating that bus on the map and calculating whether it is faster for her to walk to class or wait for the bus at a certain bus stop. Emily simply goes to our website and inputs her location and destination and then she gets the results back in a clear and formatted way so that she can decide easily how she will reach her destination.

Scenario 3: Having used the app on multiple occasions, Joshua notices he frequents certain places more than others:

Joshua has realized that most of his time on campus is spent either in the grad college residences or in Peyton hall. He's also realized that he frequently heads to these two destinations from a multiplicity of starting points that are spread all over campus. In order to minimize his commute time (and thereby maximize his research time), Joshua has decided to save these two destinations as 'favourites' in the application, thus making it easier to figure out if it would be faster/more convenient to take the bus from any of the various starting points he could be at to his two favored destinations (Grad College and Peyton Hall).

4- Design:

Front End Design:

- Will host on Heroku due to ease of usage and minimal cost requirements
- HTML/CSS/Javascript
- Twitter's Bootstrap for responsive design since we anticipate that people will be accessing the application from both their phones and laptops/desktops

Backend System:

- Postgres database (that's provided through Heroku). We're tentatively thinking of using two tables to store route information. One table would store bus stops, while the other would store info about the 'segments' that make up a route
- Table 1: Bus Stops structure:
    - Each row is a bus stop. Each row has the following attributes:
        - The route the bus stop is part of
        - Numerical bus stop ID (will be used to match with other DB table)
        - Latitude for bus stop
        - Longitude for bus stop

    Note: Bus stops can repeated since a bus stop can be part of multiple routes
- Table 2: Bus Routes Structure:
    - Each row is a unique bus route (ex: Equad Line). Each row has the following attributes:
        - The list of bus stops that fall on top of the route
        - The unique bus route ID associated with the respective bus route
        - The list of coordinates that make up the bus route i.e. each route segment will have a start point (consisting of the longitude and latitude) and an endpoint. This sequence of start points and endpoints will constitute a complete route.

5- Timeline:

- *February 20th*: Our group met with Professor Kernighan to discuss potential project ideas
- *March 17th*: Design Document Due
- *March 20th*: Setup Github Repository and make sure all team members are aware of how to utilize the system
- *March 24th:* Complete UI for main page (includes search fields, map navigation section, and route details/timing section). Research Google Maps API. Setup hosting on Heroku
- *March 25th*: Have project landing page (Project Status web page) setup and linked to main webpage.

- *March 26-28th:* How to access bus data feed and possibly organize it for our purposes
- *March: 31st*: Add CAS access to our platform
- *March 31-April 13th*: Expanding on what we have already, adding logic and decision making. Specifically, want to have the feature where it tells user which bus to take from a given point running. This only incorporates the "bus-taking" aspect of the route, not the "walking" aspect. In other words, it does not use the combination of the two to provide the most optimal route. Will format data give to us by TigerTransit into two sections. One consists of all the current bus routes, in segments. The other has the operating bus stops. We will store these forms of data in two seperate tables and parse user input to query the appropriate table to obtain the route segment. Use that segment in conjunction with Google Maps API to obtain the average travel time between those two points.
- *April 13th*: Project Prototype Due
- *April 13-27th*:
    - Add on walking to bus-stops, "is walking better than bus"
    - Incorporate logic to figure out optimal route. We anticipate that optimal routes will be calculated as a function of:

        1- Nearest Bus Stop in terms of walk-time

        2- Minimum bus arrival time
    - Create "test user" objects by using our own accounts and test how the app responds to different types of input and see if it gives the correct expected output i.e. the optimal route
    - Finish up loose ends, thoroughly test the application and debug corner/tricky cases
- *April 27-May 4th*: Alpha Test
    - Launch the application for general public use. Obtain feedback from users (either within the app or verbally) and use that feedback to adjust the response of the application
    - Opening the application to the public creates stress and boundary test cases automatically, which will be helpful in the overall testing of the application and smooth out the edges
- *May 4-9th*: Beta Test:
    - Buffer zone for if things did not meet deadlines and some parts of the implementation were delayed. In a perfect scenario, this time would be used to write the report
- *May 9-10th*: Demo Days in Class
- *May 13th*: Project Due

6- Risks and Outcomes:

- **Problem:** Difficulty in making app work on phone and what it would look like there
  Solution: Figuring out how to use BootStrap properly and how we can design the application so that it works well on all platforms, maybe try using React Native to develop it on the phone side
- **Problem**: Potential for GPS data feed, supplied by TransLoc, to go down.
  Solution: One possible workaround would be to use the static bus schedule to determine when buses would be arriving to certain stops. Of course, this has the downside of inaccuracy but would be better than having a “broken-system”
- **Problem:** Overlaying path on Google maps:
  Solution: Might have to provide results through text at first or use Google maps API (use Google Map’s Custom Overlay) in order to overlay which path user
- **Problem:** Dealing with corner cases like what if a bus goes down
  Solution: Have corner cases in code that deal with these specific corner cases