Charter

Project Manager: Matthew Rosen (mcr3);
Waqarul Islam (wislam), Usama Bin Shafqat (ushafqat),
Manisha Sivaiah (mtumkur), Kelly Zhou (kzhou)

Professor Brian Kernighan & Dr. Christopher Moretti

March 2016

1 Overview

Travel, in particular around recesses in the academic calendar, amplifies issues of logistics. Flights seem to depart at convenient times and from convenient locations only rarely; train timetables, as if by design, seldom align to minimize time en route. Princeton students experience this especially acutely. Though the Dinky runs on a regular schedule during the business week, its early-morning and weekend hours are restrictive. Those seeking transportation into or out of Princeton Station—most University travelers—have few options outside taxis and their surrogates, ride-hailing services like Uber or Lyft. These frequently are not cost-effective.

In a reflection of this phenomenon, University email listservs see a marked uptick in ride-sharing requests around school breaks. Sharing rides with fellow students is an intuitive solution, and it confers two distinct benefits: first, sharing a ride facilitates the defraying of travel expense; second, sharing with another Princeton student (or group of students) in particular, regardless of previous acquaintance, allows for this cost amortization in a trust-affirmed way.

Our project aims to connect groups of traveling students and to facilitate shared rides through Uber. Past approaches have sought almost uniformly to connect passengers and drivers — a model predicated both on a preponderance of users with cars and on their willingness to transport others. At least one of these assumptions does not hold when considering the Princeton student body; recent changes to parking policy have curtailed students' ability to own cars on campus. To enable maximum adoption within the student body, we will create a cross-platform application using Facebook's React Native framework. This application will perform the following core functions:

1. Allow users to join extant rides (e.g. those previously created by other users)
2. Allow users to create rides

2 Requirements & Target Audiences

College students are constantly looking for efficient and cost-effective travel options. Particularly around break times, students leaving campus want to optimize their travel experience in terms of cost, time, and comfort. Students often default to public transportation as the cheapest and most convenient means of transport, but public transit has clear disadvantages. Its predetermined schedule allows for less flexibility in travel times, the several stops may make the trip less efficient, and the ride is generally less comfortable and private than taking a car. As such, many students turn to other on-demand travel services, like Uber or Lyft. While taking an Uber is often more expensive than taking the train, splitting an Uber among several people can often make the individual fare more cost-effective than a train ticket. Splitting an Uber

or Lyft, then, allows students more flexibility in travel times, as they can call their car to leave at any time they choose, offers a more comfortable travel experience, and is often more efficient. Recognizing this, students often email listservs or post in Facebook groups in search of other students to share Ubers. The challenge here lies in finding enough students who share a similar time and destination requirement such that sharing an Uber is mutually beneficial and better than alternative travel options. Given limited personal networks, it is often difficult to find enough people who share the same travel needs, and simply sending mass emails or writing mass Facebook posts decentralizes the requests and makes them easily overlooked. Charter aims to address this campus problem by offering a centralized platform for students to connect with other students who share the same travel needs (similar time and destination) so that they are able to share Ubers and thus optimize their travel experience for comfort, time, and cost.

While Uber does offer UberPool as a pooling option for different users traveling in similar directions, the functionality does not address the campus problem we have identified. With UberPool, individual users must call their Uber with their destination and are randomly matched with other users on the route. While pooling is cheaper than taking an entire uber alone, the markdown in price is marginal and the ride is still much more expensive than splitting an Uber with another rider directly, especially for longer distance rides. Moreover, the user has no control over who will join their ride, if anyone. With Charter, however, students can pre-plan trips with other students on campus and split fares directly to be much more cost-efficient.

Charter will begin by targeting Princeton students. The application will require Princeton netID for login to ensure riders will be matched only with other students for safety and comfort. The initial application will cater primarily Princeton students travelling to the airport or New York City, mostly around break times. Ultimately, Charter can expand to cater to other universities, and eventually cater to a more general service that users can utilize anywhere.

3 Functionality

Charter will be used by college students on different campuses. Charter will first be used on Princeton's campus for students traveling during breaks.

3.1 Scenario 1: Waqa goes home for break

Waqa is a Computer Science student who is looking to go NYC for spring break. The most convenient way for him to the city would be to take an Uber, because he is carrying luggage and knows that a car is the fastest way into the city. Normally, Waqa texts his closest friends to see if anyone else is traveling into the city that day to try and split the fair. If that doesn't work, he sends out emails to listservs (such as the residential colleges, eating clubs, student groups) asking to split a ride. Often, he'll see that other students are also sending out similar emails but sorting through all those emails is frustrating and annoying. Finally, Waqa's last attempt to find people who want to split his ride will be through the free and for sale group on Facebook.

Waqa is frustrated that he needs to go through all these channels, spending social capital and time to find a ride. Instead, he downloads Chater onto his phone and posts a ride on it. He uses his netid to login easily and posts a ride there. Soonafter, he gets a notification saying three other people have joined his ride to NYC. Each person will now pay less than they would have had they taken the train, making this Charter ride both time efficient and cost efficient. An Uber to the city costs $80 and 75 minutes without surge, whereas the NJ Transit and MTA for four people would take twice the time and cost the same as Uber (assuming each person pays $20 in train tickets). Charter would save users a lot of time.

3.2 Scenario 2: Manisha needs a ride to the airport

Manisha is another computer science student who needs to get to the airport for a flight she will take to Europe. Her flight is later in the day at EWR so she can get to the airport anytime earlier that day.

She downloads Charter and posts a ride, and within a few hours she receives three requests for people to split her ride. Now she can ride to the airport without having to carry her checked bags on the train.

Charter's prototype core features include: login, home screen, and add/join a ride flow, and a trip summary.

`Login`: Upon opening the app, a user will be shown a login screen (which accepts netids) to make sure they belong to the university before they can access rides.

`Home Screen`: The home screen will display a list of rides that will be shown immediately to the user upon login. It will also:

- Allow users to add their ride to the app through a "New Ride" button.
- Allow users to tap on a ride to access it and see who is on the ride.
- Each ride will contain information about date/time/location immediately viewable on the home screen.
- Allow users to access "My Rides" which is a list of rides that they are a part of.

`New Ride Flow`: The new ride interface will be initiated through a button that will allow the user to add a new ride. The user will then be able to enter information like the date/time/location of the ride. Furthermore, the user will be made the ride leader, which means they would get all the information about riders who want to join the trip. The screen that a user will see after pressing the "new ride" button will allow them to fill in these criteria. Items like destination and pickup will be from a dropdown of popular locations for students in order to allow for a simpler matching system.

`Trip Summary`: The trip summary will be accessible from any page that has a list of rides (which will be the home page and the "My Rides" tabs). This trip summary will include the list of people who will be on the trip, the estimated cost, and information about pickup.

3.3 Additional Features

`Payments system`: We plan to integrate Apple Pay to allow riders to pay each other their portion of the ride payment. This will also allow the app to have a security deposit for rides ($5-10) so that people don't abandon rides.

`Splash screen`: We plan to have a splash screen that will have our logo and explain briefly how to use the app

`Search rides`: Once we have enough volume on the app, we would want to create a system for people to be able to search through rides instead of repeat posting several rides.

4 Design

We will be using the React Native development environment for the user-facing frontend. Although we will focus on iOS during this project in order to roll out as quickly as possible, using React enables us to port to Android when needed with minimal effort. The only iOS-specific elements we plan to use are Apple Pay and Apple's Push Notification Service.

The home screen features a search bar that allows searching by a combination of destination and departure time range. The database is queried and matching "trips" are displayed in a list. Tapping on a trip pulls its details from the database and displays them along with an option to join the trip. This diagram summarizes how the various screens interface with the database:

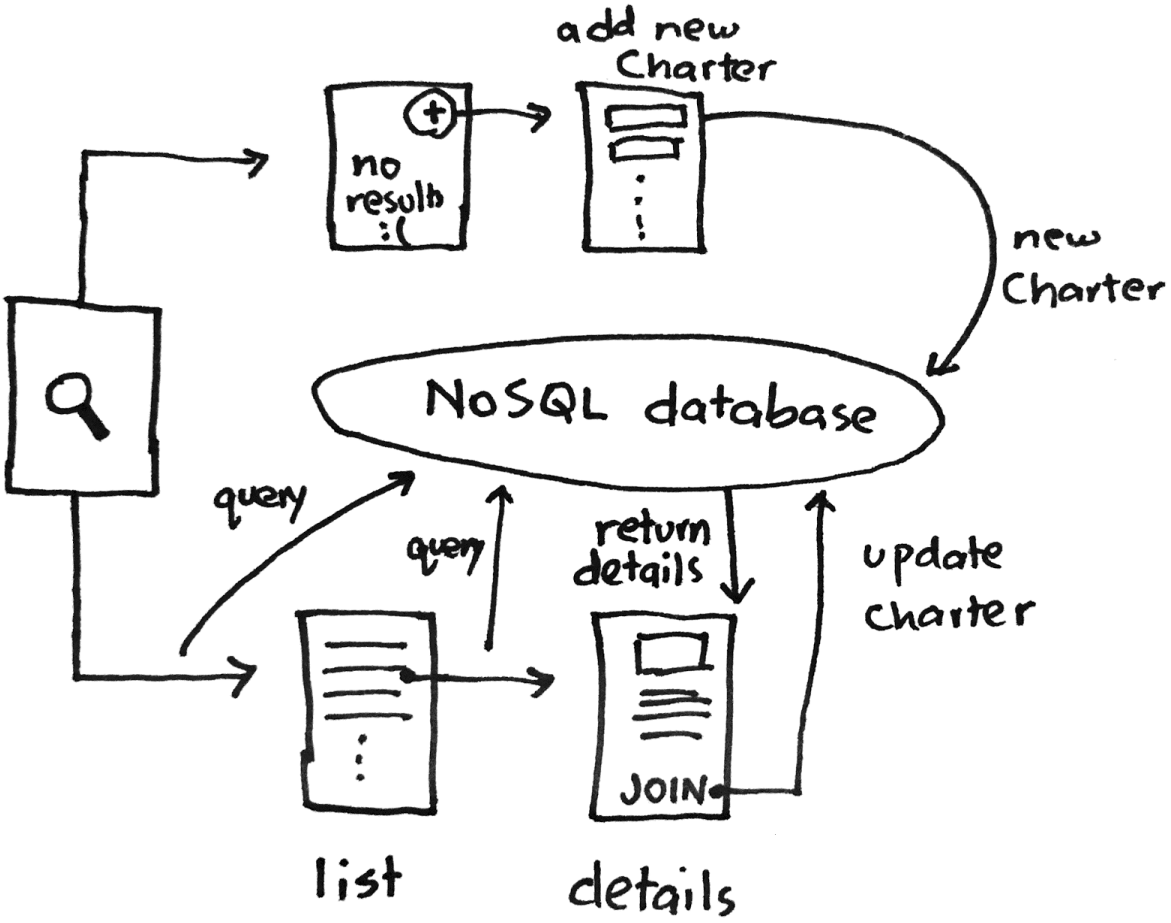

Figure 1: Database interface summary.

We will be hosting the NoSQL database on Firebase. This allows for very easy interfacing with React and also provides several integrated features including push notifications and user authentication that will scale easily both in terms of number of users and to other platforms including Android and Web. We will use the Stripe API for payments and the Uber/Lyft APIs for ride requests. We currently plan for two databases roughly structured as follows:

USER:

```
id
name
email
photo
trips-owned
trips-joined
```

CHARTER:

```
id
destination
time
```

```
owner (id)
riders (max 6 ids)
timeline
```

5 Timeline

Timeline	
Date	Milestone
March 24	Set up React Native development environment Set up Firebase backend Set up Github repo
March 31	Mockups, set up login and email verification Database structure set up
April 7	Build results page Build ride data page
April 14 (complete prototype)	Ability to join rides Ability to create rides Complete ride data page to account for other riders
April 21	Set up search for ride criterion Stripe/Apple Pay (for deposit)
April 28 (alpha test)[1]	Push notifications Complete deposit system and test with multiple users
May 5 (beta test)[2]	Build a one-page web promo site Test app before classes end
May 8-10 (demo week)	Work out last minute bugs & polish output for students traveling home

Figure 2: Timeline.

.

6 Risks & Outcomes

6.1 Possible Issues With React Native

Our plan is to utilize React Native and build out features in one language, Javascript, so our final product with be compatible across operating systems. This will allow us to more efficiently create our cross-platform app. Our team will have to get used to React Native's markup style. This should not take too much time, as the markup style is similar to CSS - which most of us are familiar with CSS - however

[2]Take rides to local destinations using the app
[2]Open up for public by special invitation.

this is still a consideration for the flow of development. We plan to implement third-party libraries and might run into issues here because there is not a guarantee of the functionality or further support of these libraries. In this case we might have to write our own components/modules, which might throw us off time-wise. If it becomes a better trade-off to develop on two platforms rather than recreating modules, we'd essentially develop Android and iOS versions. This would also throw us off time-wise as members of our team would have to learn Swift.

6.2 Searching the Database with Complex Queries

Currently, we are using Firebase to hold our database. This could pose an issue when users attempt to update the database while other users attempts to access it. Most simple queries, which will ebb the type we need in our app, should work fine. However, in case we run into issues during something complex like searching the database, we can try to restructure our Firebase database to find a way to add flags or efficiently parse the data. In case we see a larger structural issue with Firebase, we could mitigate this through redoing the backend through AWS incase we run into huge problems with queries, but this is unlikely to be the case.

6.3 Issues Accessing the Uber API

We don't need any authentication to obtain pricing estimates, but if we want to request a ride on Uber on the user's behalf, we need to register an application to make requests on the API. If we are denied access, we can just implement a button that takes the user directly to Uber's app so they can book the ride themself.

6.4 Accessing the Stripe API through React Native

There is potential for the Stripe API or Apple Pay interfaces to not be accessible through React Native. However, we found libraries that should allow these to work through React. There is a small potential for errors here though, which would be mitigated through use of Swift or Xcode for that part because they are interoperable.