Hitchhikers

Complete Documentation

by
Aneel Yelamanchili, William Wang, David Sealand & Adam Espinoza

Table of Contents

High-Level Requirements Document

Team Contacts:
Adam Espinoza - adamespi@usc.edu
David Sealand - sealand@usc.edu
William Wang - wang975@usc.edu
Aneel Yelamanchili - ayelaman@usc.edu

Hitchhikers Concept:
We intend to develop an application to allow ride sharing between users across longer distances than is feasible for common ride-sharing applications in today's market.

Hitchhikers Features:
First off, the app needs to have a login system where users can register the first time they are using the app or login if they already have an account. A database would store each user and their details (picture, name, email, etc.) while their password would be hashed into a database for security purposes. Another feature we would like to include would be an in-app payment system -- possible payment options include Venmo & Stripe that we could import over from an API to allow users to interact with their clients.

As for the functionality of the app, we would like to include additional features to optimize user experience. There will be some sort of separate interfaces for drivers and riders. With destination searching using apple maps as a platform, drivers can place their destination on widgets so riders can assess where they can end up. Drivers should be able post rides specifying the day and time they leave, their destination, route, origin, and cost. A suggested cost will be given based on the mileage of the route but ultimately, the cost is up to the driver. Riders should be able to search for rides by the date and time of the ride, where the driver is going, their route (for possible pick-ups/drop-offs along the way), their starting location, and the cost of the ride.

In order to make finding a ride easier, the app will show an upcoming trips feed so that users can find open spots for quick weekend getaways or look within specific time periods for rides when they know they are free. Additionally, riders will be able to interact and chat with their driver in order to plan specific details regarding pick-up and drop-off times and locations along with other crucial details.

Finally, to improve the user's experience, we will provide a settings page that specifies the terms and conditions of using Hitchhikers and allows for profile changes, the user's ride history in case they need to get into contact with the person that previously drove them, and a help page with frequently asked questions and contact information to the Hitchhikers organization.

Technical Specifications Document

1. INTRODUCTION

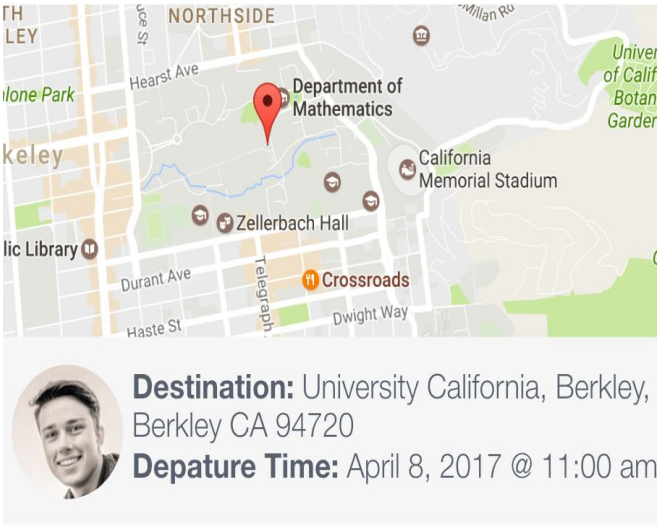
1.1 Objectives

We intend to develop a mobile application to support ride sharing between users across longer distances than is feasible for common ride-sharing applications in today's market.

1.2 Scope

HitchHikers will be marketed towards individuals who are looking for a long-distance ridesharing solution such as young adults in college and the general adult population. The application will be finished by April 2017.

2 FUNCTIONALITY



Our system intends to increase ease of travel, while allowing users to gain an experience traveling cross country. We offer a hub for users to post their destination ahead of their departure, allowing others to join in through the application. We want to create an interactive social media platform that allows ride-sharing to take on a whole new level that it has yet to see with larger competitors such as Uber and Lyft.

2.1 Use Case Diagram

Users	Use
Riders	They will search for rides based on destination location.
Drivers	They will post rides detailing their origin, destination, date and time, cost, and route.

2.2 Site Map

Our application, while in progress, will allow users to seamlessly go back and forth between various pages.

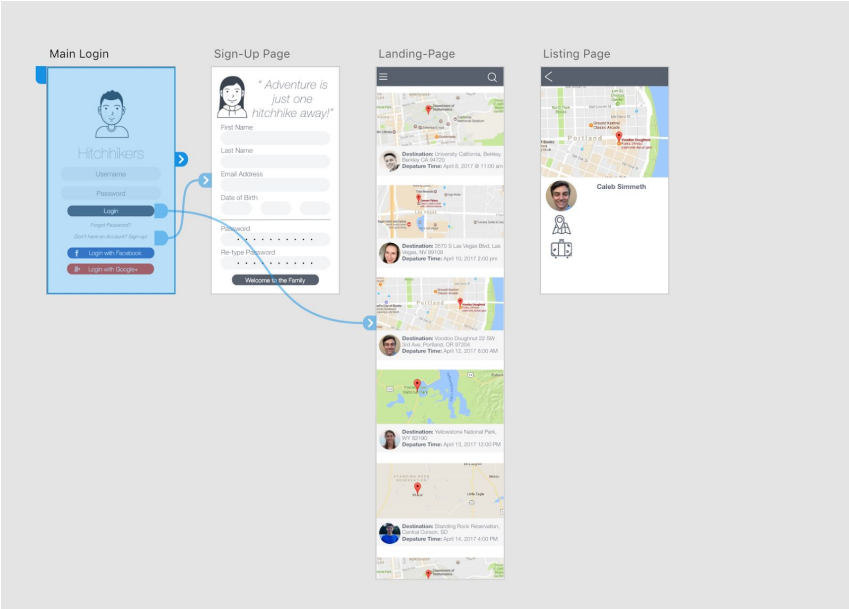
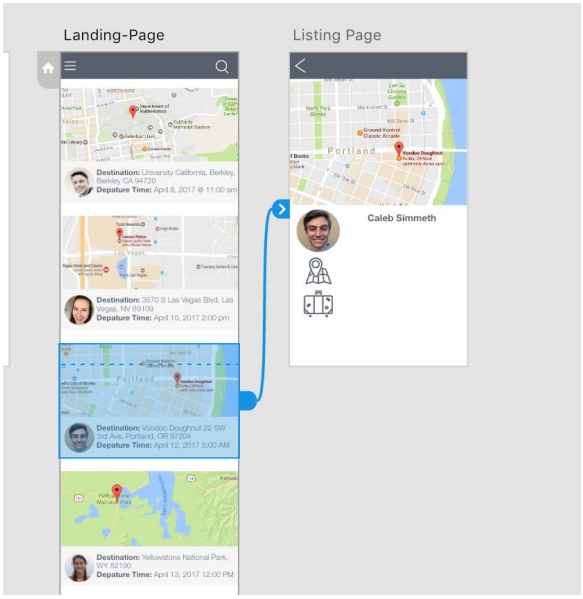


Figure 1 - The Main Login page will let users make a new account or login to a current account if they already have an account.



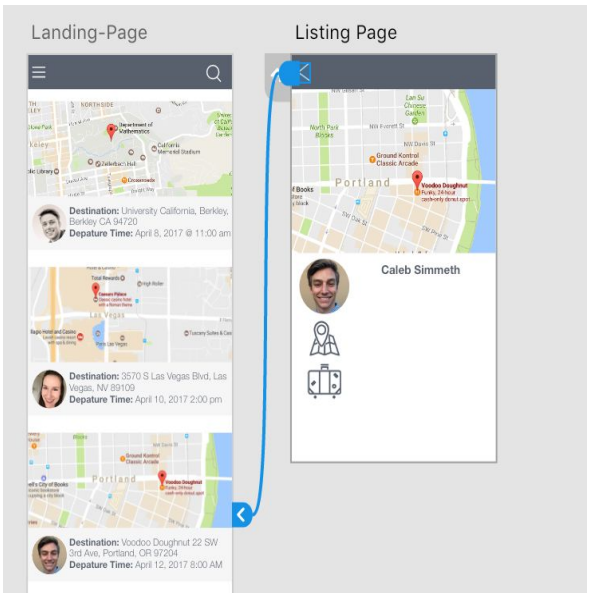


Figure 2 & 3 - Users will be able to inspect which locations they would like to go to, and see what each person has listed for their trip

3 ARCHITECTURE

3.1 Architecture Overview

The application will consist of both client side and server side code. All client side code will be written in Swift, and all server side code will be written in Java. We will use Starscream Websocket package in Swift to send messages back and forth to the Java backend. The frontend Swift code will control the View of the application.

3.2 Component Structure

We are using an MVC-based design pattern for our application. Each Model will reside in the models directory (server side), Controllers will reside in the controllers directory (client side), and the View will reside in a xib file under Xcode's Interface Builder.

3.3 Application Interfaces

We will be using iOS interface builder to create the View of the application. All assets will be Xcode design elements, with some supplementary Adobe Illustrator images.

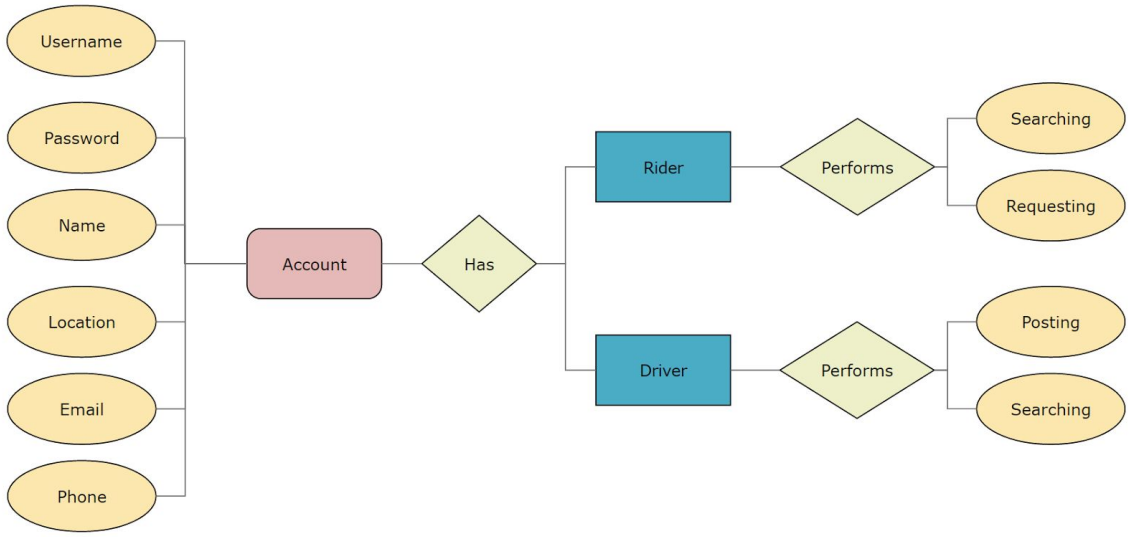
4 DESIGN

4.1 Object Diagram

Classes/Files	Sub-Functionalities	Description
SideMenuController	● Profile ● Settings ● Past Trips ○ Payments ● Contact Us	The Slide Menu is a hamburger menu on the user's feed page that will slide out to reveal a set of helpful options for the user.
ProfileController	● Upper ○ The user's picture will be displayed on the upper part of the viewController. ● Lower ○ The user's information will	The User Profile will allow the user to see his or her information.

	be displayed in a tableView under the upper part of the viewController. First Name, Last Name, username, password, contact info, will all be present.	
FeedController	● Settings slide out menu to show ***SideMenuController*** ● Clickable TableViewCells that show ***RideController*** ○ Map of user's destination ○ Time that he or she is tentatively planning to leave ○ Driver profile picture ○ Posted cost of trip ● Rely on locations services to display trips that are leaving within your location	The feed will display trips that are leaving from your location.. A search option at the top will allow the user to search a destination. Options will dynamically pop up based on current string.
RideController	● Time that he or she is tentatively planning to leave ● Route ● Origin/destination ● Cost	The Ride Controller will display the specifics of the ride, updating details when riders are added to the ride along with being editable by the driver.

4.2 Entity Relationship Diagram



Detailed Design Document

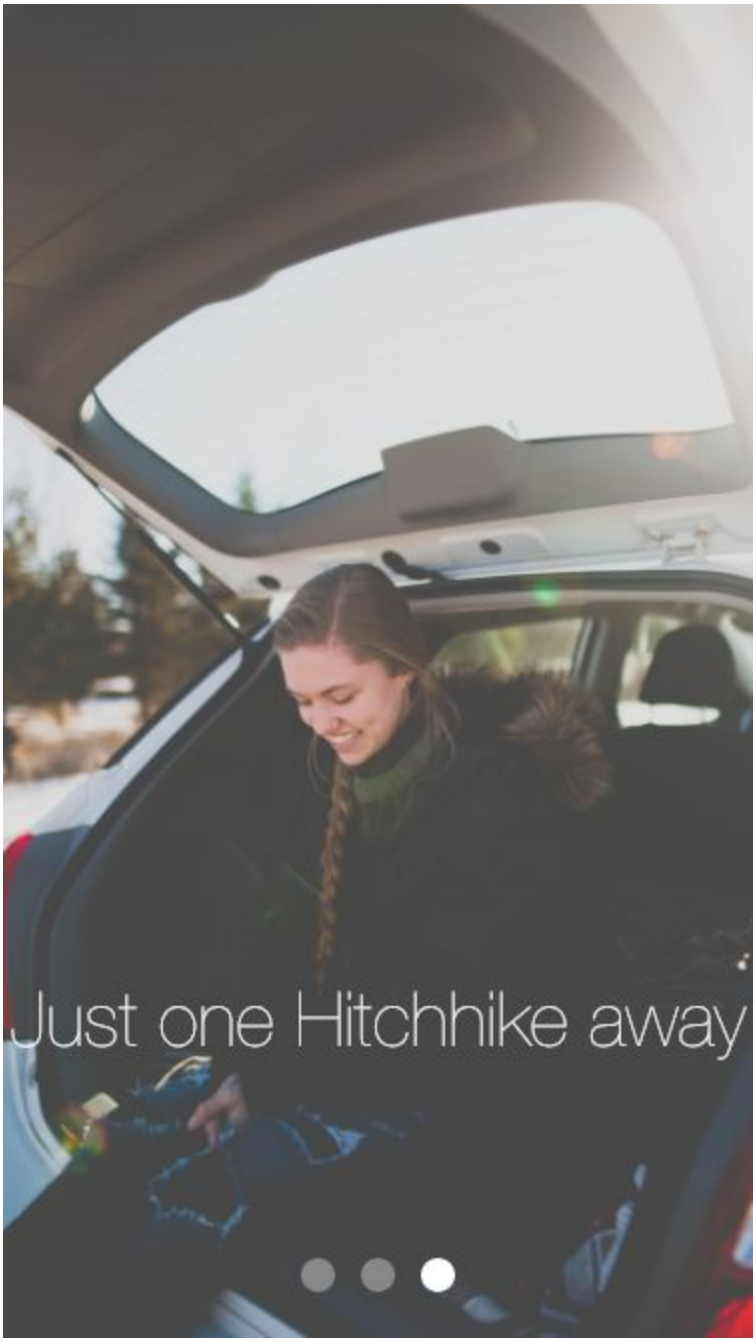
Hardware/Software Requirements

- iOS 10.0
- Xcode 8 -- Swift 3
- MySQL
- Apache Tomcat Server
- JVM/Java

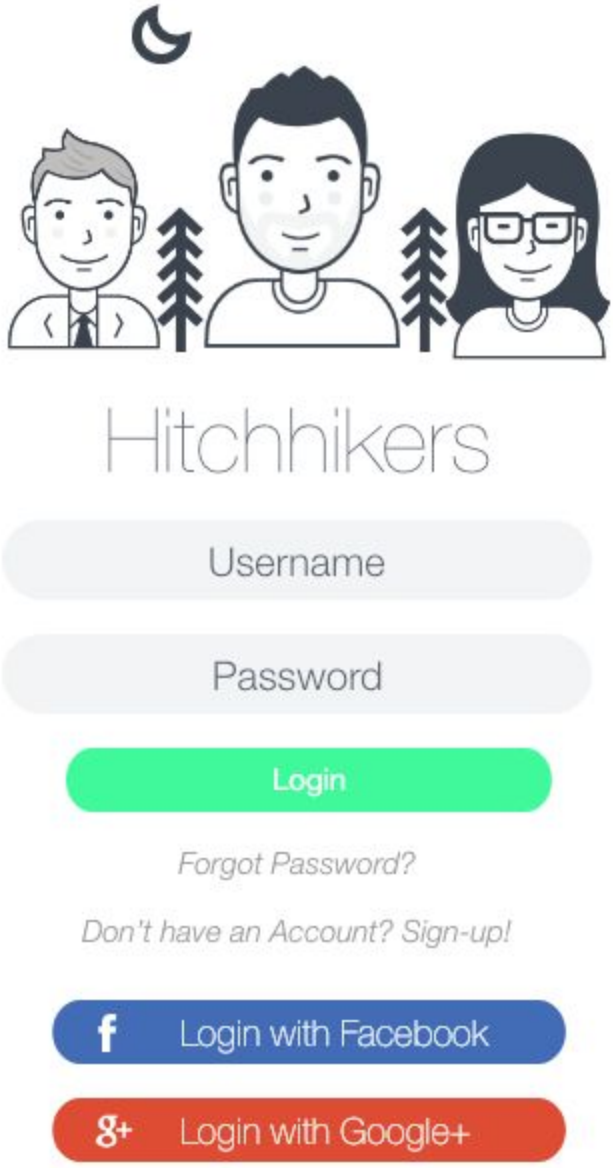
GUI Mockup (on following pages)

Installation Screen 1

Installation Screen 2



Installation Screen 3



Login Main Page

“ Adventure is just one hitchhike away!”

First Name

Last Name

Email Address

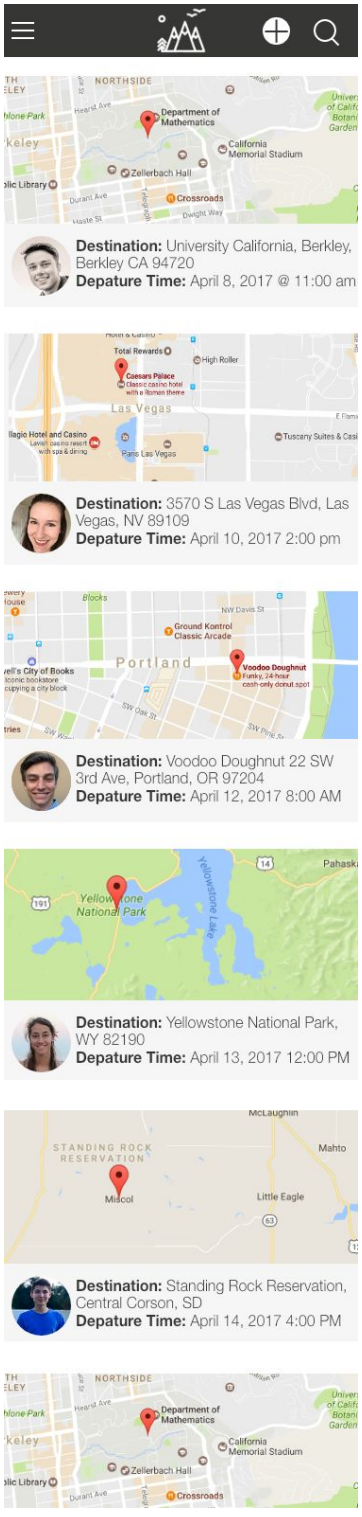
Date of Birth

Password

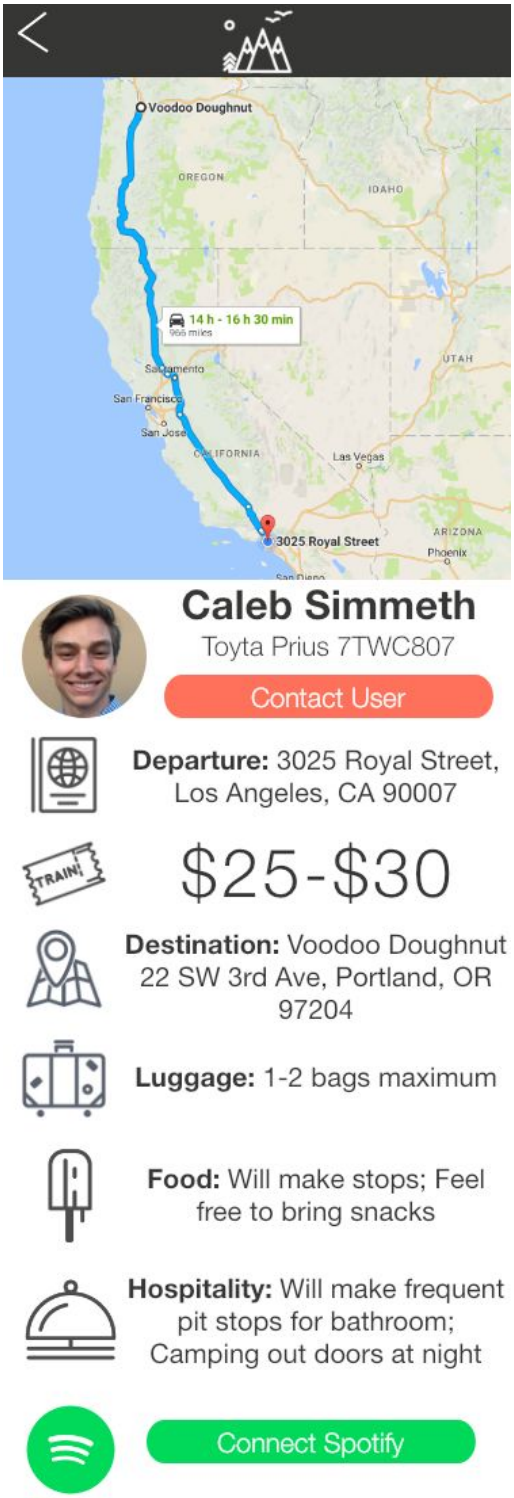
Re-type Password

Welcome to the Family

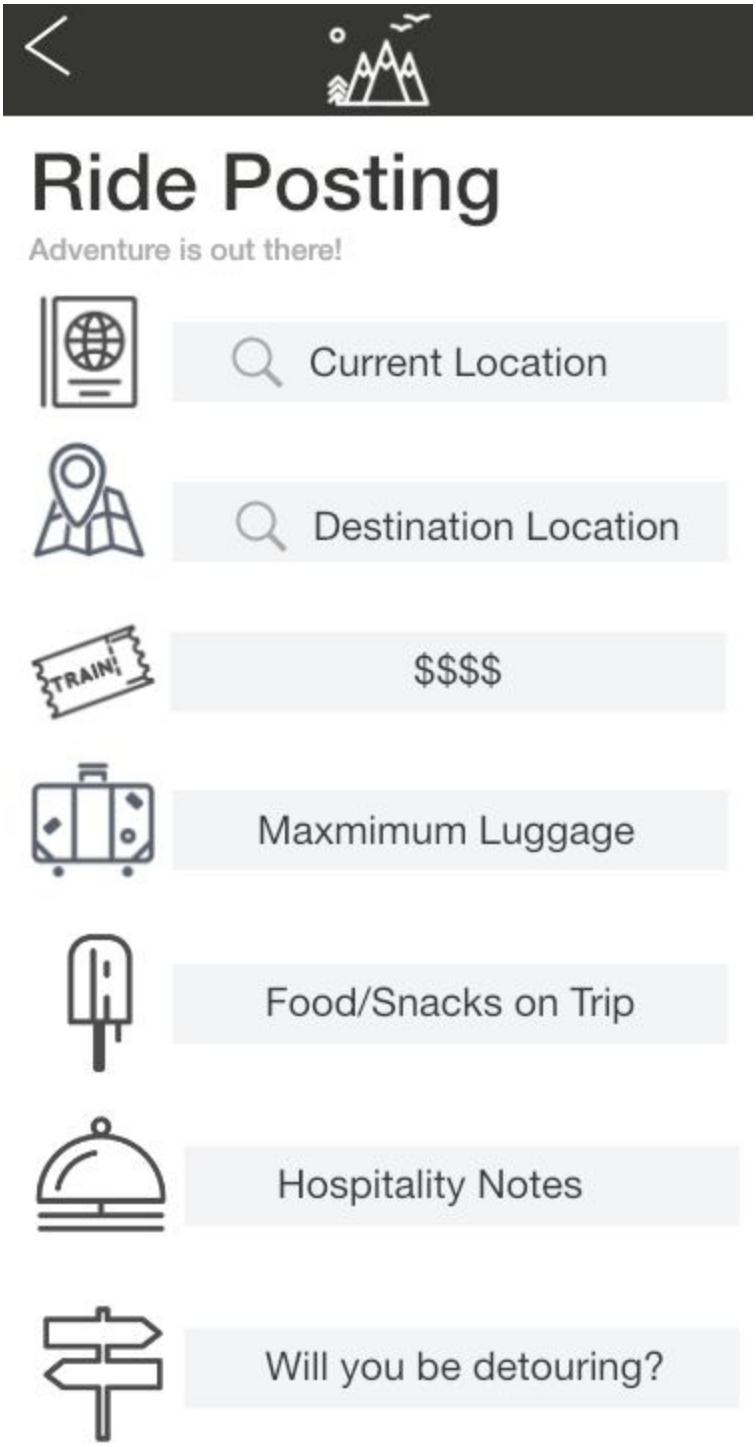
Sign-Up Page



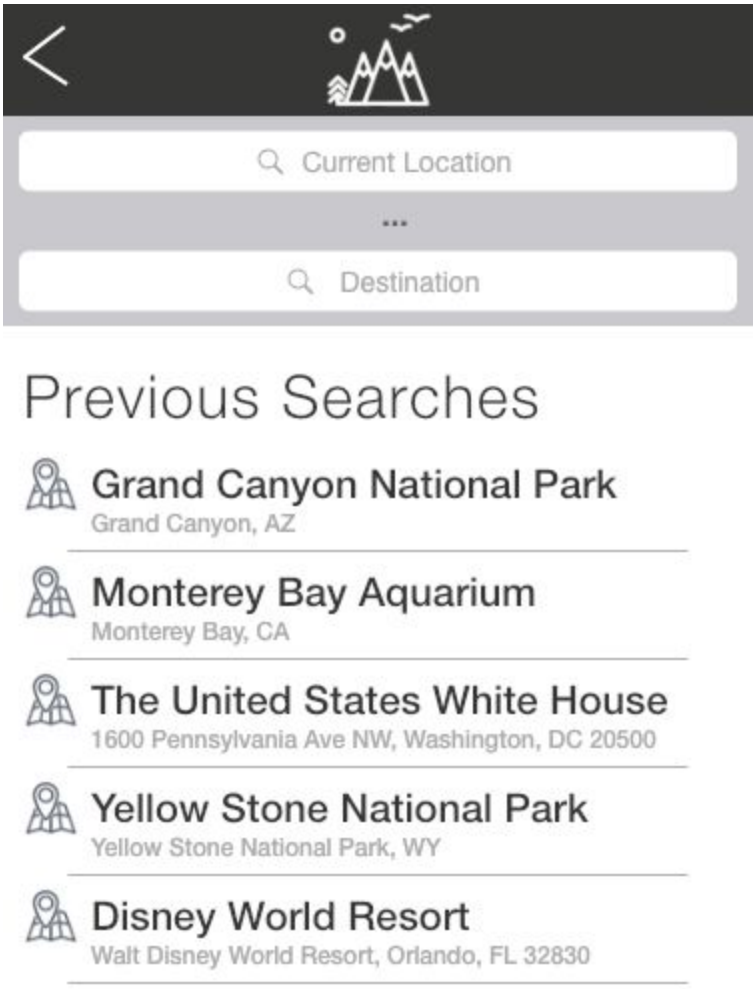
Landing Page (main after login)



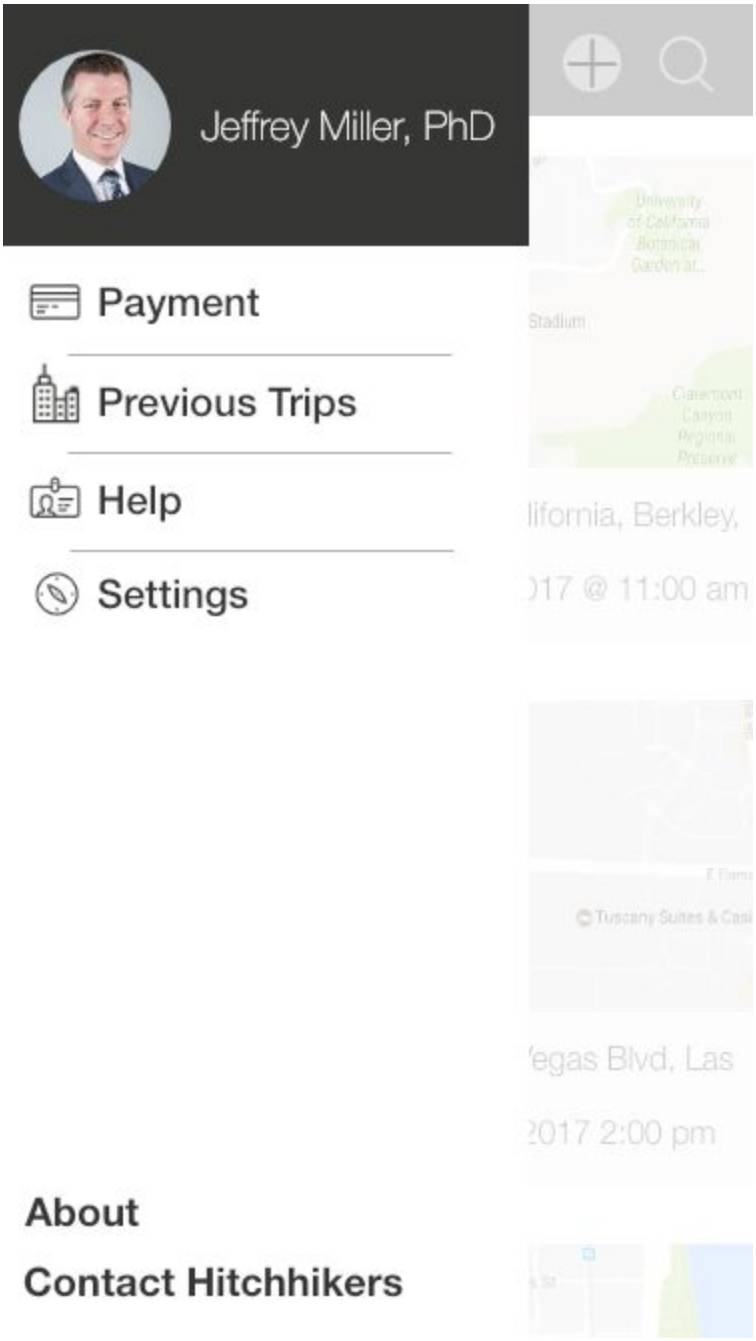
Listing Page



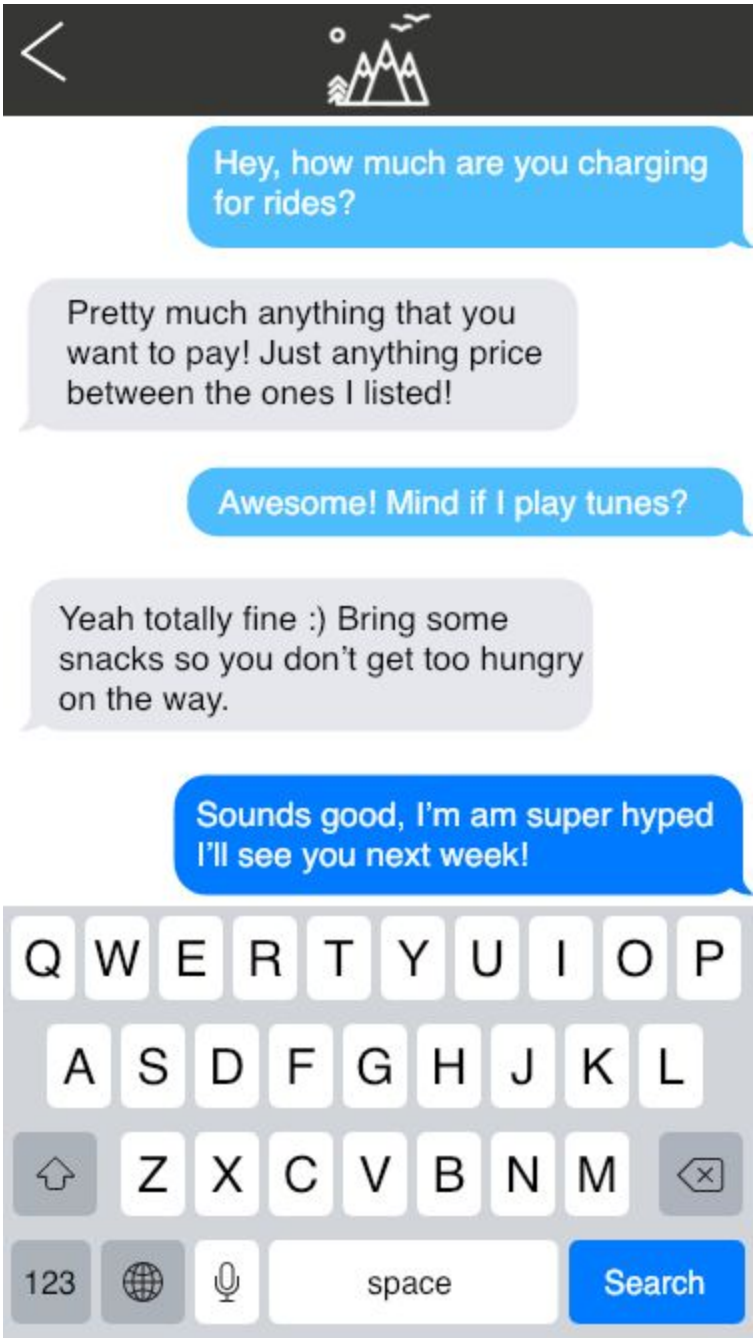
Ride Posting Page



Ride Searching Page

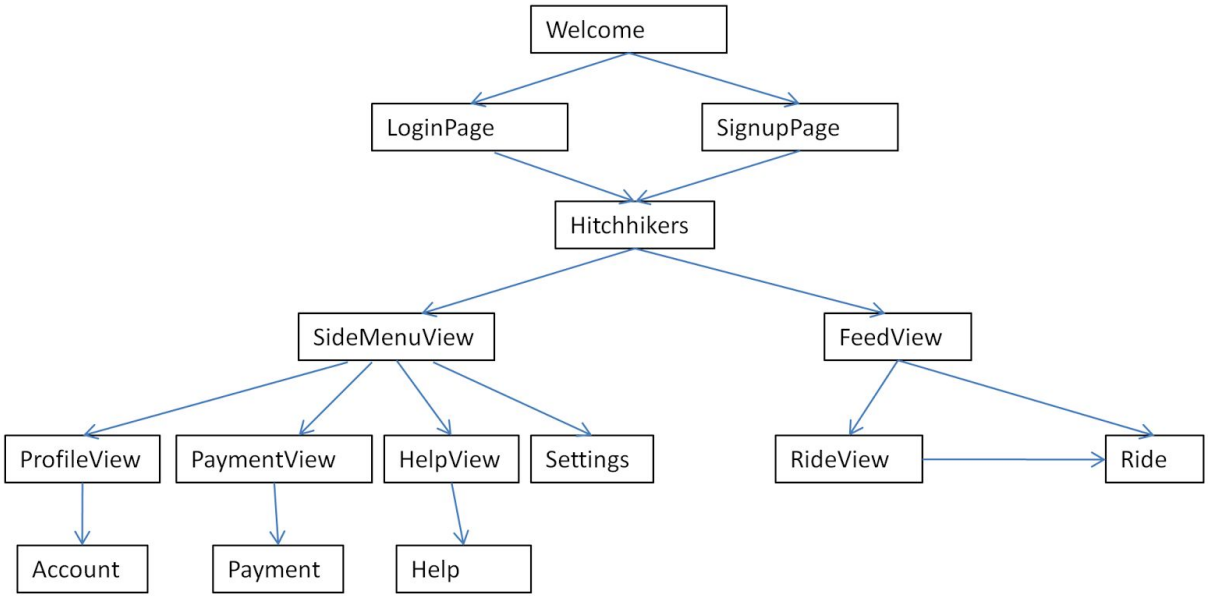


Side-Bar Toggle



Contacting a Ride

Inheritance Hierarchy



The inheritance hierarchy demonstrates how each class will use other classes. Users will login or sign up using the separate classes and then are brought to the main Hitchhikers page which will display the SideMenu and Feed. Each of those have their separate classes which will display the next views which all have their own back-end Java code.

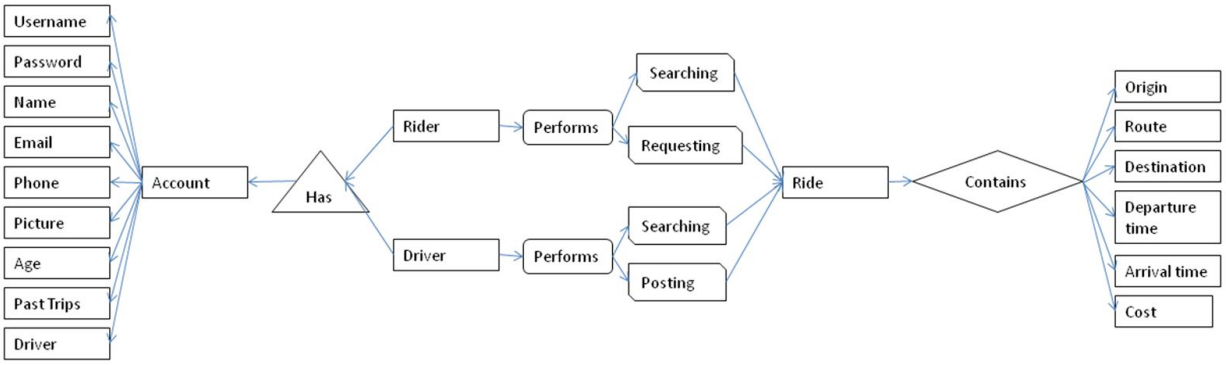
Class Diagram

Classes/Files	Sub-Functionalities	Description
SideMenuController	● Profile ● Settings ● Past Trips ○ Payments ● Contact Us	The Slide Menu is a hamburger menu on the user's feed page that will slide out to reveal a set of helpful options for the user.
ProfileController	● Upper ○ The user's picture will be displayed on the upper part of the viewController.	The User Profile will allow the user to see his or her information.

	● Lower ○ The user's information will be displayed in a tableView under the upper part of the viewController. First Name, Last Name, username, password, contact info, will all be present.	
FeedController	● Settings slide out menu to show ***SideMenuController*** ● Clickable TableViewCells that show ***RideController*** ○ Map of user's destination ○ Time that he or she is tentatively planning to leave ○ Driver profile picture ○ Posted cost of trip ● Rely on locations services to display trips that are leaving within your location	The feed will display trips that are leaving from your location.. A search option at the top will allow the user to search a destination. Options will dynamically pop up based on current string.
RideController	● Time that he or she is tentatively planning to leave ● Route ● Origin/destination ● Cost	The Ride Controller will display the specifics of the ride, updating details when riders are added to the ride along with being editable by the driver.

This class diagram details how each class will work within itself and then will be put together so users can interact with each seamlessly.

Entity Relationship Diagram



In this diagram, we show how the riders and drivers interact with the app. Each of them have accounts with their personal details and then they can perform different actions. Riders search for rides and request rides from drivers while the drivers post rides and also search for rides. Each ride contains its specific details.

Database Schema

User

userID	First Name	Last Name	Age	Phone Number	Email	Password	Picture
1	Jeffrey	Miller	35	(421) 636-6067	phd@jeffreymiller.io	jmiller	(Insert Picture Here)
2	David	Kempe	45	(123) 456-7890	dkempe@usc.edu	dkempe	(Insert Picture Here)

Current and Past Trips (Each database is linked to a specific user):

Date/Time	Car Model	License Plate	Cost	Starting Point	Destination	Driver Name	Driver ID
3/9/17, 12:05PM	Toyota Prius	7SAL0101	$30	San Francisco, California	Eureka, California	Jeffrey Miller, PhD	1

Added to trips table

Detours	Hospitality	Food	Luggage	Total Seats	Seats Available
none	Bring tunes	Bring some	none	4	3

Note: Driver ID is the same as the User ID. We will implement the database schema with MySQL through the SQL Workbench interface.

Testing Document

Test Case	Method	Expected Result
1: Create user	Click the “Sign Up” button on the frontpage	Takes all the the user’s inputted values for First Name, Last Name, Date of Birth, Password
2: Create ride	Click the “+” button on the top right of the feed	Takes the user to the “Ride Creation” page, which will take the user’s inputs and post a ride to the feed.
3: Search destination	Click the magnifying glass to the right of the “+” button on the top right of the feed controller	Take the user to search page, similar to that of many iOS applications, that will allow the user to input a location that the app will search around. After successful completion of searching and clicking of a ride, will take user to an “updated feed” with rides based on the search result.
4: Validate login	Click “Log In” button on the initial page.	Will check against our MySQL Database to make sure the specific username exists, and if so, if the password is correct. If all information is correct, correct user information will populate feed, profile, and all respective controllers

5: Validate contact info	Cross-reference contact info in the Model with data in the database	Will return error: “This _____ is already in use” if a field already exists in the database. If all fields are new, then create a new user in the database with the provided contact information.
6: First Time Sign Up	Check if user is signing up for the first time. If so, show the initial login views	Check if all credentials provided already exist in the database. If yes, return error. If no, populate database with new user.
7: Ride History	Cross-reference ride data with User’s id number to ensure it is the correct user	Displays correct ride history of the user from the database in ride history controller.
8: Sign Up Button Links to main page	Check navigation segues are properly set up.	Will display sign up page with location based feed.

9: Sidebar Profile	Sidebar properly slides out into view (modally presented horizontally)	Displays sidebar.
10: Login Button links to main page	Check navigation segues are properly set up.	Will display location based ride feed
11: Previous Trips links to trips menu	Check navigation segues are properly set up.	Will display controller that shows past rides
12: Payment links to payment menu	Check navigation segues are properly set up between views.	Will display controller that will allow choice of payment: Venmo/Paypal.

13: Help links to help page	Check navigation segues are properly set up between views.	Will display help controller and options.
14: Search Icon links to search page	Check navigation segues are properly set up between views.	Will display the search controller with search functionalities highlighted above.
15: Plus button links to ride posting page	Check navigation segues are properly set up.	Will display the post a ride controller with functionalities highlighted above
16: Contact User Links to Contact Page	Check navigation segues are properly set up.	Will display the user's profile controller that allows current user to see selected user's information.

17: Listing page grabs correct information	Data from database is parsed accurately and displayed appropriately	Will populate the current user's feed with location based rides from users in database who qualify.
18: Ride Posting saves information for user	Data is written to the database properly. Checks that data is written and available within the database.	Will add the ride post into the user's id section of the database. Will exit the ride posting controller on completion. If not, return error and display to user.
19: Maps show up on main menu page	All maps for rides that are posted are visible above the ride and poster information	Small maps are supposed to show up above each user widget.
20: Installation screens show up	Pop up when app opens	Slide from right to left giving information.

Deployment Document

Deployment Steps

Backend:

1. Run the Java Project 'HitchhikersBackend' on the TomCat server with the port number that you choose.
2. Run the SQL file 'Hitchhikers.sql' in MySQL Workbench.
3. On line 43 of the class Application.java, set your username and password for MySQL (currently set to root).
4. Download and install the program ngrok from ngrok.com.
5. Run ngrok.exe and type 'ngrok http _____' where _____ is the port that your TomCat server runs on (default is 8080).

Frontend:

6. Download and install XCode.
7. Open Hitchhikers.xcworkspace.
8. Go to the Google Developer Console and follow the steps to get a Google Maps/Google Places API Key (should be one API key).
9. Replace the Google API Key in AppDelegate.swift with the one you created.
10. Get the correct ngrok URL from the backend/ngrok server. This will change every time you restart ngrok.
11. If need be, go to the left side bar and click the topmost blue "Hitchhikers" row to access the settings. You may have to change the "Team" under the "Signing" section to your own personal team.
12. In the same view, change the bundle identifier to the something different than what was already there. This is so that no two projects have the same identifier.
13. Make sure the server is on, and build and run the program.