DALHOUSIE
UNIVERSITY

**Mobile Computing**

**Project Milestone - 1**

**B00991820**

**ns910276@dal.ca**

**App Name**: Take Me There

## Problem Statement

Could you imagine switching between 4 different apps to book a cab? Sadly this is the situation in India. A trip is not confirmed immediately when booked because of reasons like a driver expecting a high pay, a driver not willing to go to the drop location, or a driver waiting to get a trip that satisfies their expectation better. When cabs are on demand it even takes more than 10 minutes to book a cab.

The problem is currently handled by booking a cab for the same destination from multiple cab booking applications and cancelling all the other ride requests, once a ride is confirmed. Sometimes you may get more than a ride confirmed and you are forced to pay a cancellation fee for cancelling a confirmed ride. This is a huge inconvenience and currently there are no other ways to handle this easily.

## App Concept

How easier will it be if all the application switching and ride cancellation which are manually done currently is automated by a mobile application. The app I am proposing to develop will let the user select the pickup and drop location, and the type of vehicle. That's all the user has to do, the app will communicate with different cab service providers and notifies the user when a ride is confirmed. The major purpose of this app is to enhance the cab booking experience in India.

## Target Audience

Many people in India move to developed cities like Chennai, Bangalore, etc. for work. These people usually don't take their personal vehicle to the city where they work and will be dependent a lot on cabs for commuting. Most of them book cabs at least twice a day for different reasons like going to and coming back from work, shopping, party, visiting a friend and many such reasons. They might get anxious if they are trying to reach somewhere on time and their ride is taking a lot of time to get confirmed. Also they might get frustrated switching apps trying to book a ride. These people will be the primary users of my application which will help them book a cab without getting frustrated or anxious by automating the manual steps.

## Features and Functionalities

The user will first see an **animated preloader** while opening the app. Every user will have a profile and is not allowed to use the app without signing in. A new user is asked to sign up before using the app. The **authentication** part will be handled using **Firebase SDK**.

The app will require **backend** support to handle ride booking, also it will act like a bridge between the mobile app's frontend and the database. The api to support ride bookings will be written using **Node.js** and **Express**. 4 apis mocking the behaviour of 4 cab booking service providers (OLA, Uber, Rapido and Namma Yatri) will be written. These apis will return a response at a random time using setTimeout. Based on which response is first received a ride will be confirmed with that service provider. Then the current ride's OTP will be updated in **Firestore** so that the mobile app will be updated with **real time data**. Also a **notification** will be sent using **Firebase Cloud Messaging**. To mock the completion of the trip, if the trip is not cancelled within 5 minutes of confirmation, it will be automatically completed from the backend. These **backend apis** will be hosted in **Heroku** and will be integrated with the frontend using **axios**. Also **Google maps API** will be used for fetching the route.

MongoDb will be used as the database to store, retrieve, update and delete user data and ride details. **MongoDb Atlas** will be used as it allows access to the database remotely. Even though all the data is stored remotely in a database, it is an efficient approach to fetch and store the constant data like user data locally fetch again whenever there is a change. This will be achieved using **AsyncStorage**.

The app will have a feature that allows the user to update their profile picture. They can either upload a picture from the gallery or use the **native camera feature** to click a picture and upload it. This will not be the only native feature the app will have, it will also use the **native map** to let the user select the pick up and drop location. The user's current location will be the initial pick up location but the user will be allowed to change it by dragging through the native map.

The **final mobile app** will allow the user to create an account before using it. Once the user is logged in to the app, they can select the pickup and drop location, and the vehicle type and request a ride. And the app's backend will communicate with different cab booking service providers and confirm a ride. Once the ride is confirmed the user will be given a one time password to start the app and the option to cancel the ride. From the home screen where the user requests a ride they can navigate to the profile screen and the trip history screen. From the profile screen the user can edit profile details like profile picture, name, mobile number, etc. And in the trip history screen the user can see all the trip details they have requested.

## App Architecture

The  login screen is the root screen when not logged in else the ride booking screen acts as the main screen. Sign up screen can be navigated from the login screen. Start/cancel trip, user profile, trip history screens can be navigated from the ride booking screen. Trip details screen can be navigated from the trip history screen.

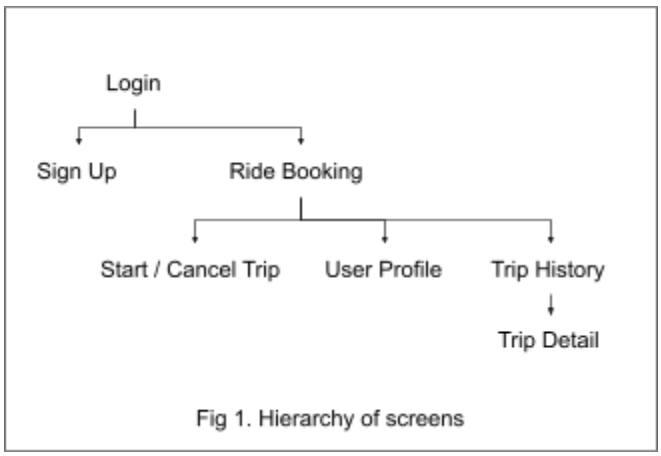

Fig 1. Hierarchy of screens

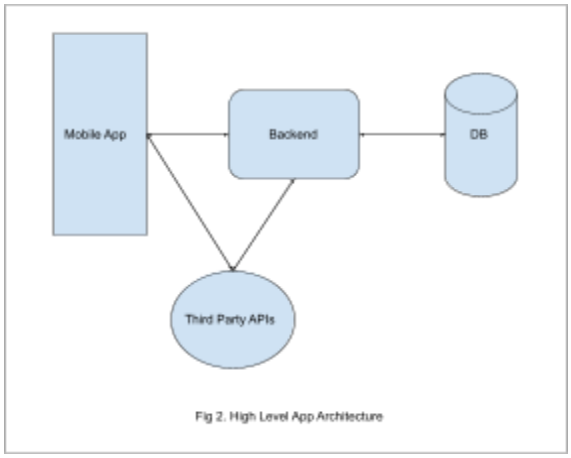

Fig 2. High Level App Architecture

On a high level the frontend of the app will communicate with third party apis and the Node.js backend which will perform the business logics and act as a bridge between the app's frontend and database. Also apis like Google places api will be integrated through Node.js backend for safety purposes and to keep track of the api call hits so the free limit is not exceeded.

## Development Timeline

- Feb 4 - Feb 10: Login and signup screen, ride booking screen
- Feb 11 - Feb 17: Profile screen, trip details screen
- Feb 18 - Feb 24: Backend apis, establish DB connectivity with MongoDB
- Feb 25 - Mar 3: Ride booking screen map integration
- Mar 4 - Mar 10: Camera and gallery access in profile screen with backend support
- Mar 11 - Mar 17: Trip details screen
- Mar 18 - Mar 24: Host backend and DB and integrate it with the app
- Mar 25 - Mar 31: Add an animated preloader
- Mar 1 - Mar 7: Finish up the pending tasks if any

Login, sign up, sign out, and ride booking screen will be developed first as these are the pages required to build the MVP.