**Taxi Application for Groceries.**

Shashank Kasinadhuni

Abstract— With the humongous increment in number of vehicles on street, individuals around the nation particularly in metro urban areas have begun facing issue with groceries. People must catch up with schedule, as every individual face problems with groceries as they have no time. The basic idea is to design a web app for connecting a rider and a driver for doing groceries.

Keywords-**Driver, Rider, Trip Request and UserLogin.**

I. Introduction and Motivation.

Shopping for groceries in stores has remained almost the same for decades, everyone needs groceries from various stores. Though shopping for groceries has remained the same, the way we look at it is changing day by day. There are many Apps like Swiggy and Big basket have made their way to people, these apps deliver groceries to the people. Apps like Walmart and Costco when one tries to order groceries, it shows **in-store purchase only.** There are cases where people don’t have time and they always need to find someone who is doing groceries at the same time. I usually face this problem of groceries, where each time I need to call my friend who has a car and ask him to pick me up so that I get things done. The basic idea is to design a Web and Mobile app for grocery pooling with a very basic implementation for the rider and the car owner who is willing to go to any grocery store. The car Owner/driver can either help others or at a price.

II. Literature Survey

They journey of this application just started to take off with exploring various frameworks which are available for mobile and web apps.

**Ionic Framework:**

Ionic framework is based on Angular2 and JavaScript framework where it is logically separated following Model View Controller pattern. Ionic is based on typescript.

**React framework:**

In React, the Virtual DOM acts as a layer between the developer’s description of how things ought to look, and the work done to render your application onto the page. To render interactive user interfaces in a browser, developers must edit the browser’s DOM, or Document Object Model. This is an expensive step, and excessive writes to the DOM have a significant impact on performance. Rather than directly render changes on the page, react computes the necessary changes by using an in-memory version of the DOM, and re renders the minimal amount necessary. This is all possible because of the “bridge,” which provides React with an interface into the host platform’s native UI elements. React components return markup from their render function. When I was Looking for an end to end Solution, I found JSX getting supported with React. It is good to know React but for a beginner in JavaScript, it is slightly above my learning curve and I did not go for this application.

**Sockets**

Sockets have traditionally been the solution around which most real-time systems are architected, providing a bi-directional communication channel between a client and a server. This means that the server can push messages to clients. Whenever an event occurs, the idea is that the server will get it and push it to the concerned connected clients. When it comes to this taxi app, usage of sockets plays a crucial role where the rider tries to contact companion. The data which the rider sends hits the socket and reaches the endpoint to the companion. I still need to see how it is implemented.

**What does a Web Pack do?**

Webpack reads the entrypoint and looks for dependecies in package.json file.So webpack bundles the entire thing into a single file.These can be written as .jade files.

**Angular Js Framework:**

**Angular Js framework is a front-end framework based on MVC model where there is a controller which enables to connect to model.** It is a very good idea to decouple DOM manipulation from app logic. This dramatically improves the testability of the code. Mean Stack tries to give a web view for phones too.

**Embedded Java Script:**

Embedded JavaScript enables you to load the template file and render with data.The Ejs compiler just makes your job easier to send data from one Html File from one page to another.This made my life easier just for a Web app. I was mainly focused on bringing the solution out and resorted to in using EJS.

**III. Architecture**

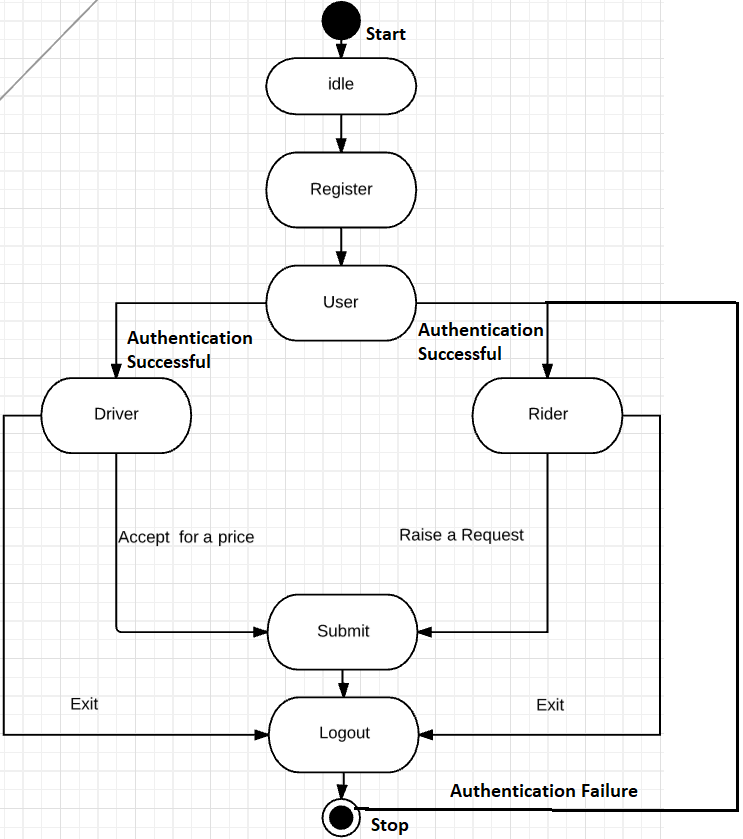
The architecture of this Web application goes as follows:

**Development:**

1. In the development stage, this web application is built in Node JS framework. This is comfortable framework as we can completely code in Java script on the client side and the server side.
2. This is a web app completely built using express framework particularly in Node Js as it is light weight and highly flexible with easy to read code.
3. This web application in Nodejs has enabled me consistently to route to modules, middleware interface.
4. Also, to display or send information from one page to another, the application uses an Embedded JavaScript templates often known as EJS module, provided by NPM libraries.
5. To connect to database, the application has used a single connection to SQL DB which has all the details regarding the host, host username and port number which are utilized.
6. Also, to give more details about this web application’s interface with Maps Embedded with the Api Key where there are two interfaces one is showrider Interface and other is showdriver interface, and it is all how they get connected.

**Working:**

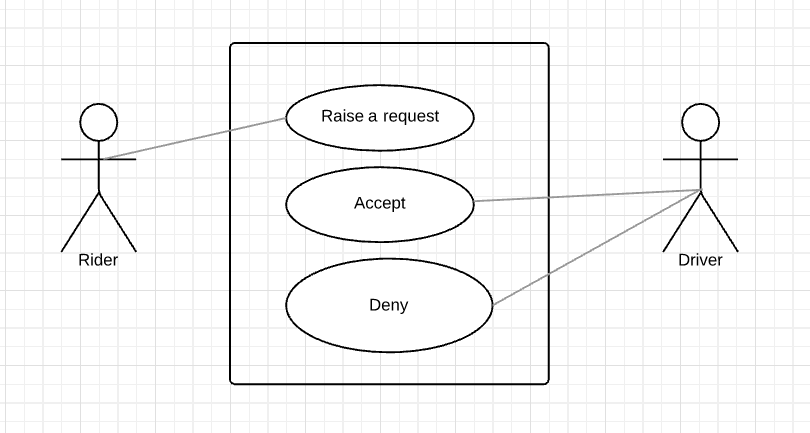
Initially in this application one must sign up and start as a afresh either as a rider and a driver. When a Rider logs in he gets the list of drivers visible on his map interface and can see all the cars which are registered from Database. Similiarly, the driver sees the list of riders on his map interface. In my application, the Rider tries to raise a request to all the drivers. So, the driver either rejects or accepts the request from a rider at a price. This message is visible in the console.



**Fig 1: State Diagram for Taxi Web application.**

The above is the state diagram of this complete taxi web application. It clearly replicates the working of this application, where any person just sign’s up and becomes either a rider/driver. The user tries to log in if the authentication is successful and either tries to become a rider or driver. His details are stored in the backend in mysql. once the he becomes a rider, the drivers are visible on the interface and if the rider raises a request the driver should accept the request for a price or reject it.

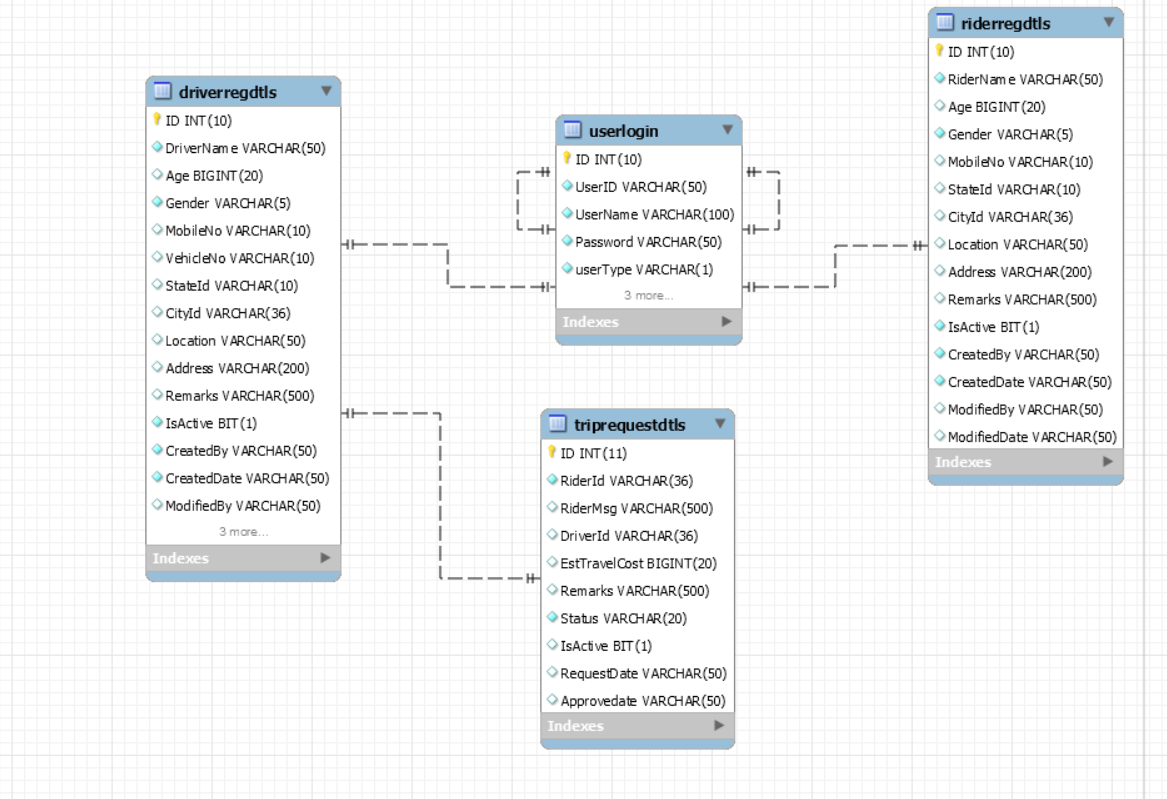
In case of any failure in authentication he does not log in and the application is just idle.



**Fig 2: Use case Diagram for Taxi Web Application**

The above describes a graphical depiction of interaction between elements of the taxi application. The actors here are Rider and the driver.

The rider tries to raise a request. The driver either accepts or denies. Thus, the three elements in the use case diagram are Raise a Request, Accept and deny.



**Fig 3: Schema Diagram for Taxi application.**

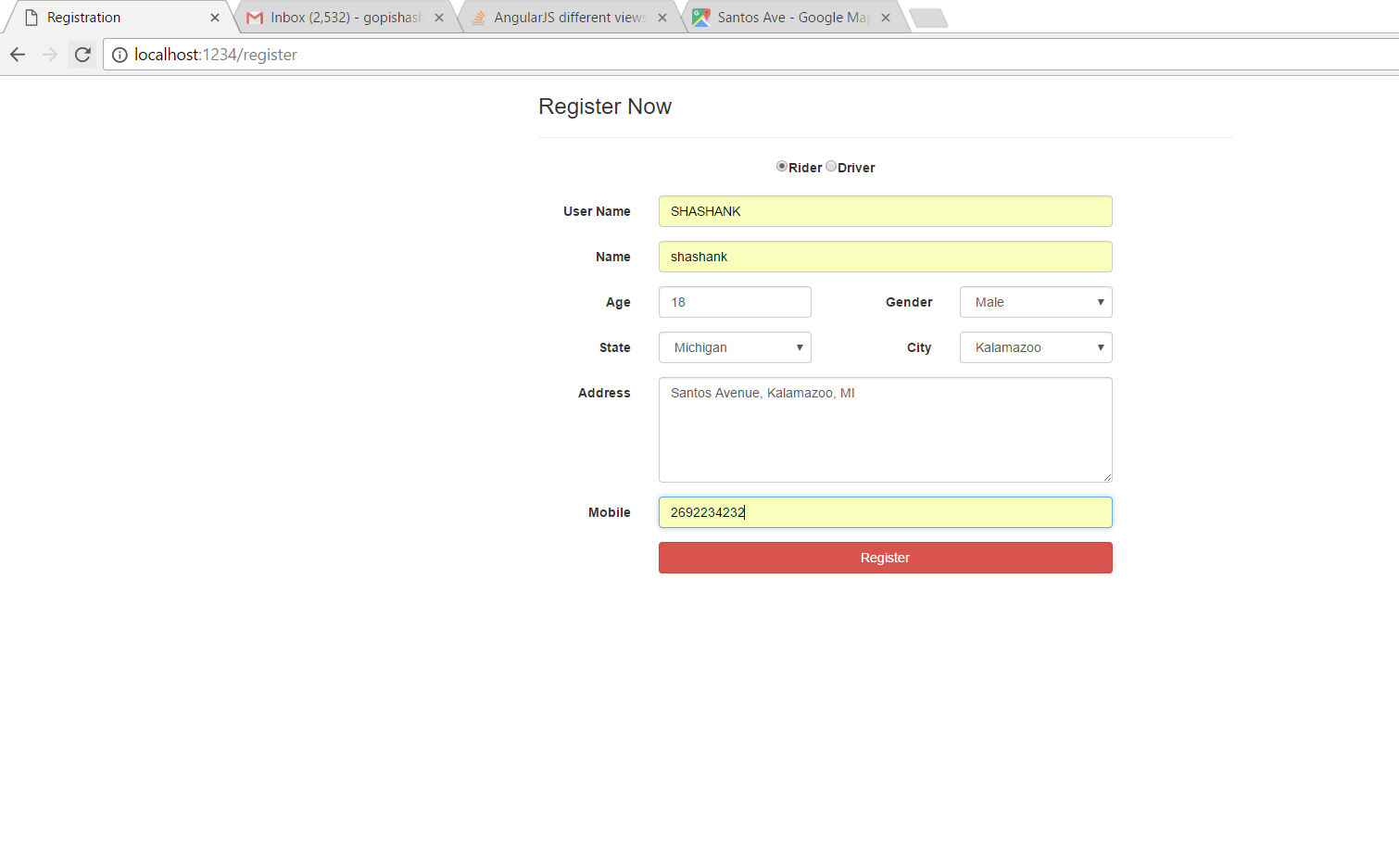
The above depicts the schema diagram for this Taxi application database which is utilized at the backend. It stores

all the data of the rider and driver. These details are stored and retrieved as soon as he submits the rider or the driver form. Once the rider submits the form, the rider’s data is retrieved, In, case the driver Submits the driver’s data is retrieved.

**IV. Screenshots**

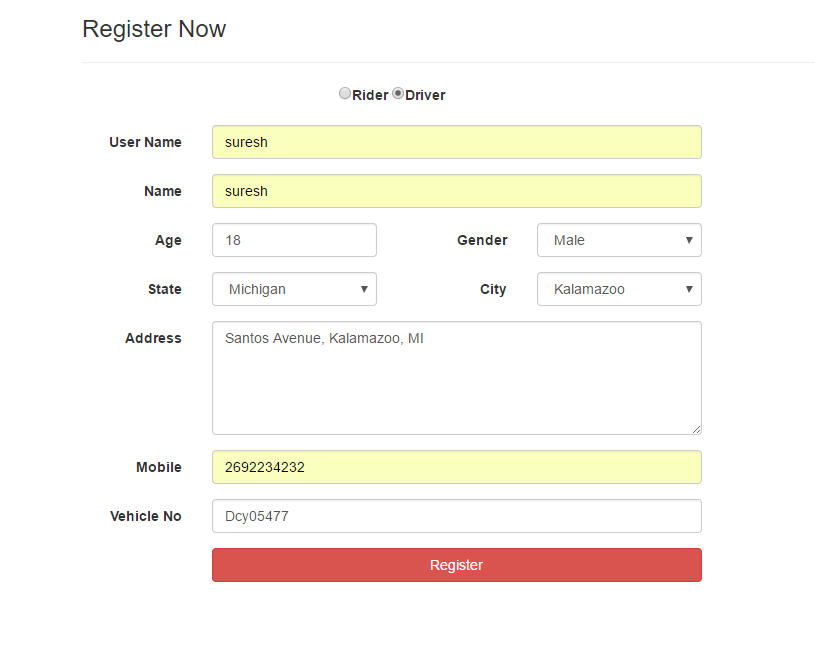
**Let’s start:**

**Rider Registration:**



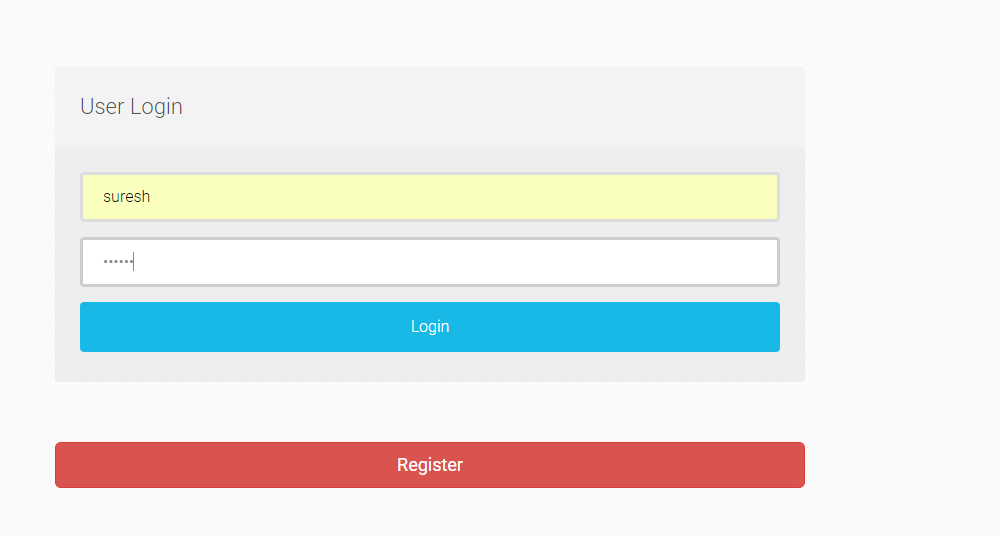
**Fig 1: Register as Rider.**

**Driver Registration:**



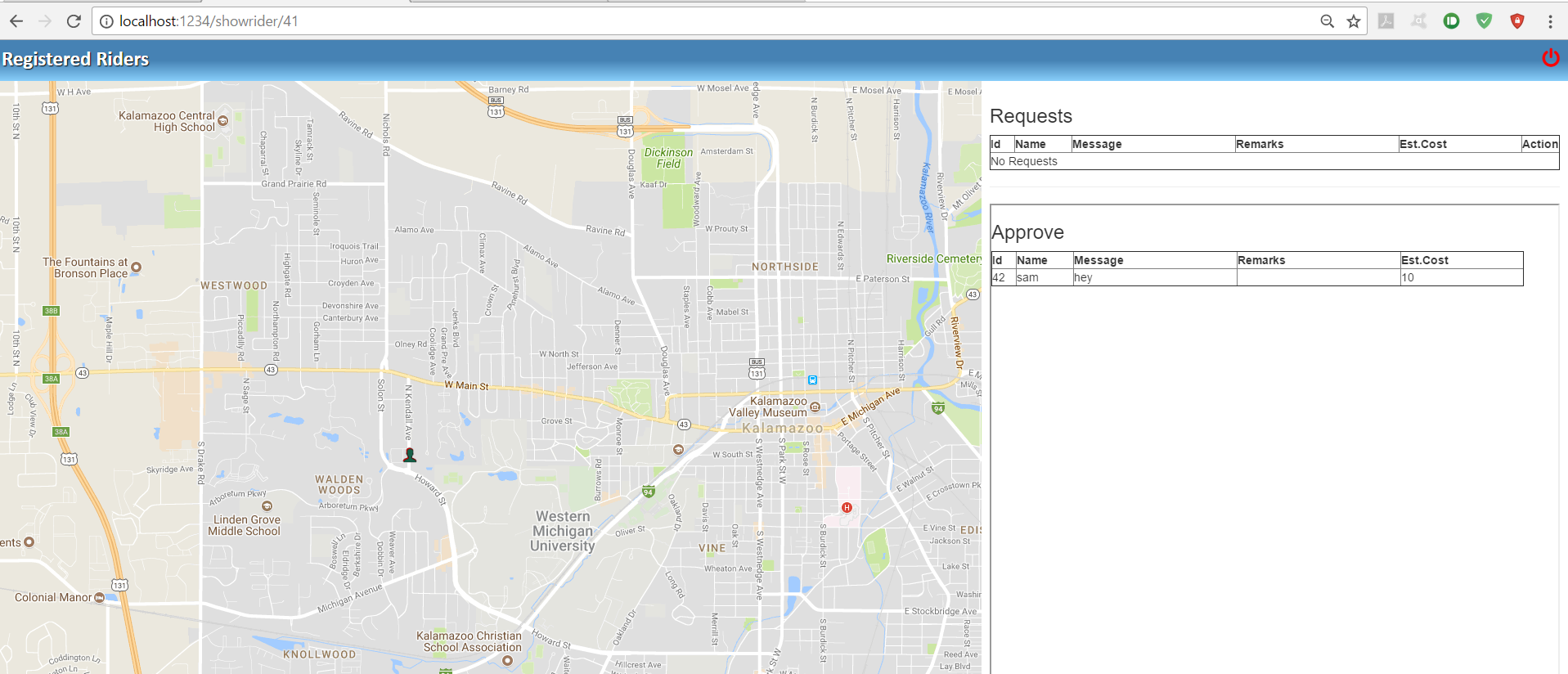
**Fig 2: Register as Driver.**

**Login Page:**

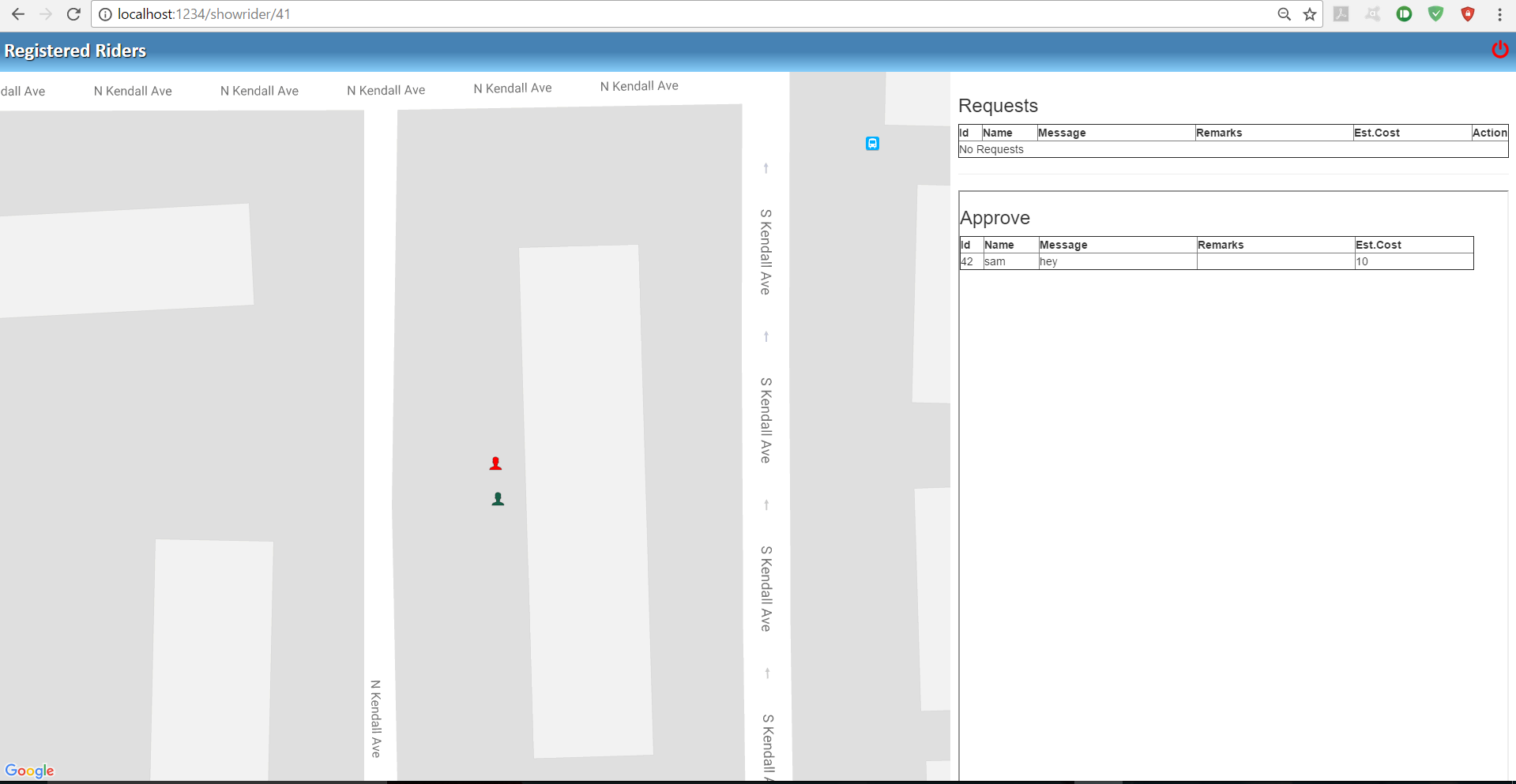


**Fig 3: Login Page**

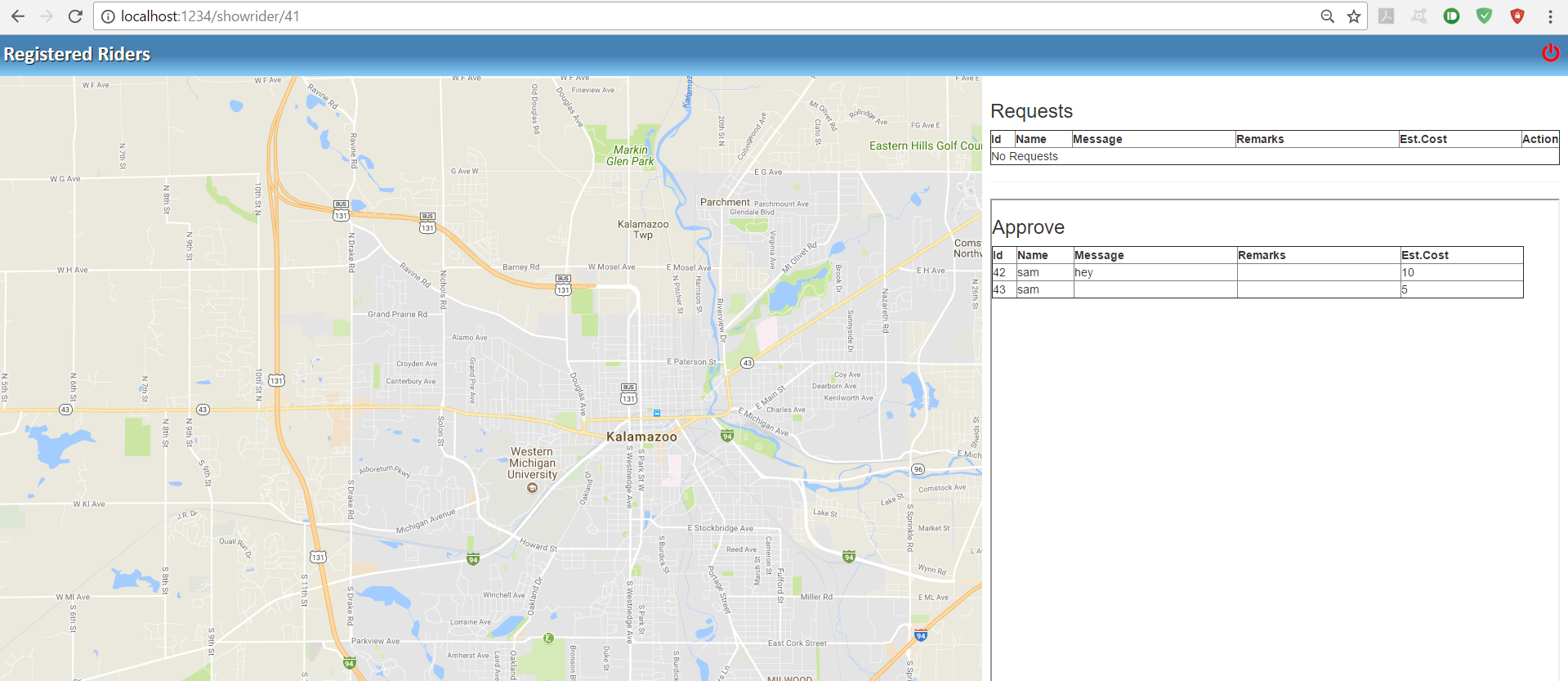
**Driver Interface:**

 **Fig 4: Driver Interface (All riders and requests are Visible)**

Riders:



**Fig 5: distinguished riders on a Driver interface**



**Fig 6: Rider’s Request**

**V. Challenges Faced.**

There are many challenges which I had to face in completing this application

First and Foremost, I tried to build a native app for phones using react-native and its libraries but later I found that I had some knowledge constraints using these libraries. So, moving on I started learning Front-end angular framework and tried to proceed with this application, so that I could just build a web view for the phones. This, further seemed to be time consuming as I am unaware of writing services and directives in this application. Now, this seemed to have a high learning curve.

For this reason, I tried to build a web application using Ejs and completed this application to get a working model. Since My main motive was to build a working model quickly and a sturdy approach. For this reason, I simplified and just made a web Application which I could finish within the time constraints.

**VI. Conclusion**

Thus, To conclude, this project is implemented in NodeJS, Ejs and an express application As such I can call this project successful as I have established a platform for the driver and rider to connect for the same cause and get groceries. I plan to work further and build an angular application with the same functionalities. Likewise, further this Web application can be improved to build a native application.

**VII. Future Enhancements**

There are always future enhancements for this application which I thought and would be working on in near future.

* Web chat application between Diver and Rider, so that they can communicate between each other.
* Complete the Project with Angular Front End and MongoDb as the Backend.
* Implement Haversine Formula and display the distance between the two.
* This project can be enhanced further with sockets to track the drivers and the riders end of the application.

**VIII. References.**

[1] W3schools,”BootstrapTemplates”,Internet: [https://www.w3schools.com/bootstrap/bootstrap\_templates.asp [Oct,3](https://www.w3schools.com/bootstrap/bootstrap_templates.asp%20%5bOct,3) 2017]

[2]. EmbeddedJS,” EJS - JavaScript Templates”,Internet: <http://www.embeddedjs.com/getting_started.html#include,[Oct> 5,2017]

[3]Google” Google Maps”,Internet: [https://www.google.com/maps[Oct](https://www.google.com/maps%5bOct) 7,2017]

[4] www.findlatitudeandlongitude.com,” Latitude and Longitude to Address”, Internet:<https://www.findlatitudeandlongitude.com/find-address-from-latitude-and-longitude/#.WilwoDeQxPY,[Oct> 11,2017].

[5] www.latlong.net,” Get Lat Long from Address Convert Address to Coordinates”, Internet: <https://www.latlong.net/convert-address-to-lat-long.html>

[6] Joshi. (2015). The Carpooling System. International Journal on Recent and Innovation Trends in Computing and Communication.

[7] youtube, “JavaScript Templating Tutorial” | Introduction To EJS – YouTube”,Internet: <https://www.youtube.com/watch?v=3Jz18kXCLwg>

[8]scotch.io,” Top Shelf Web Development Training ― Scotch”, Internet: https://scotch.io/

[9] www.npmjs.com,Internet: <https://www.npmjs.com/package/geocomplete>

[10] facebook.github.io,” Getting Started · React Native”,Internet:https://facebook.github.io/react-native/docs/getting-started.html

[11]medium.com,” Ionic vs React Native – Ankush Aggarwal – Medium”,Internet: <https://medium.com/@ankushaggarwal/ionic-vs-react-native-3eb62f8943f8>

[12]Angularjs “angular official Documentation”Internet: <https://docs.angularjs.org/guide/introduction>

[13] developers.google.com,” Getting Started | Google Maps Geocoding API | Google Developers”, Internet:<https://developers.google.com/maps/documentation/geocoding/start>

[14]Codepen,” codepen”,Internet: <https://codepen.io/>

[15] Lucidchart,” Blank UML: Lucidchart”, Internet: <https://www.lucidchart.com/documents/edit/08df5357-5a4f-499a-a4d1-02a11375ca3e/0>

[16] “Create a MEAN Stack Google Map App (Part I) ― Scotch” Internet: <https://scotch.io/tutorials/making-mean-apps-with-google-maps-part-i>.

[17] “Create a MEAN Stack Google Map App (Part II) ― Scotch” Internet: <https://scotch.io/tutorials/making-mean-apps-with-google-maps-part-ii>