Team 13

- Aparna Kulkarni
- Tanvi Shukla
- Mounika Muga
- Preeti Patil
- Hema Begur
- Jasdeepsingh Oberoi

Contribution by team members

- **Aparna Kulkarni**
  - Designed driver signup page
  - Completed driver functionality along with Preeti
  - Integration and testing of customer and driver module with Preeti.

- **Tanvi Shukla**
  - Completed maps functionality.
  - Completed booking a ride functionality along with Hema.
  - Integration and testing of customer modules with Hema.

- **Mounika Muga**
  - Designed customer signup page
  - Designed and coded admin functionality.
  - Integration and testing of Admin module with Jasdeep.

- **Preeti Patil**
  - Designed and implemented driver signup and verification pages.
  - Designed home page of Uber.
  - Completed driver functionality along with Aparna.
  - Integration and testing of customer and driver module with Aparna.

- **Hema Begur**
  - Designed customer and driver home pages.
  - Completed booking a ride functionality along with Tanvi.
  - Integration and testing of customer modules with Tanvi.
  - Written report along with Preeti.

- **Jasdeepsingh Oberoi**
  - Designed admin pages along with Mounika.
  - Integration and testing of Admin module with Mounika.

All the team members contributed towards complete integration and testing of the entire project.

Prototype of Uber:

(Enterprise Distributed Systems, Fall 2015)

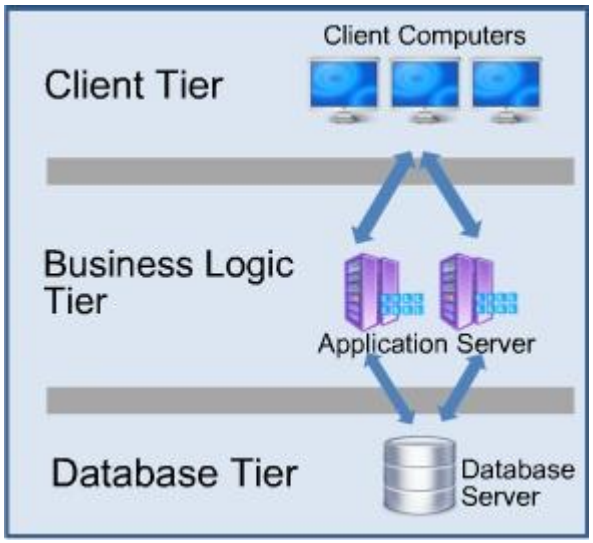
Abstract

Nowadays we always all the time use taxi services. With the advent of smart phones, using cab services has been redefined. There are numerous mobile phone apps which we can use to call a cab at any time during the day and use its services very efficiently. One of those services is Uber. During its initial development company worked on predicting demand for private hire car drivers and where demand is highest. Later it partnered with local taxi drivers facilitating customers to book a ride when requested using his or her smart phone and has grown rapidly.

Keeping in mind the idea we have created prototype of Uber. We have replicated functionalities and created a web application. It has functionalities like booking a ride, customer and driver signup and ask for approval to become a driver, customer signup, searching for past bills, rides etc. Also customer can add his credit card details and after the ride the money will be deducted from his account.

System Architecture

Our architecture consists of three tiers, shown below:



Technologies Implemented		
Client Tier	**Business Logic Tier**	**Database Tier**
• HTML • jQuery • Ajax • CSS • Bootstrap • Angular.js	• Node.js Express Framework • Rabbit MQ	• MySQL • Mongodb

Database

Database Scripts

Administrator

```
CREATE TABLE `administrator` (
  `idadministrator` int(11) NOT NULL,
  `firstName` varchar(45) DEFAULT NULL,
  `lastName` varchar(45) DEFAULT NULL,
  `address` varchar(45) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `state` varchar(45) DEFAULT NULL,
  `zipcode` int(11) DEFAULT NULL,
  `phoneNumber` int(11) DEFAULT NULL,
  `emailID` varchar(45) NOT NULL,
  `password` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`emailID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Billing

```
CREATE TABLE `billinginfo` (
  `idbillingInfo` int(11) NOT NULL,
  `date` date DEFAULT NULL,
  `pickupTime` datetime DEFAULT NULL,
  `dropOffTime` datetime DEFAULT NULL,
  `distanceCovered` int(11) DEFAULT NULL,
  `rideCost` varchar(45) DEFAULT NULL,
  `sourceAddress` varchar(45) DEFAULT NULL,
  `destinationAddress` varchar(45) DEFAULT NULL,
  `driverID` varchar(45) DEFAULT NULL,
  `customerID` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Customer

```
CREATE TABLE `customers` (
  `idCustomer` int(11) NOT NULL,
  `firstName` varchar(45) DEFAULT NULL,
  `lastName` varchar(45) DEFAULT NULL,
  `address` varchar(45) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `state` varchar(45) DEFAULT NULL,
  `zipCode` varchar(45) DEFAULT NULL,
  `phoneNumber` varchar(45) DEFAULT NULL,
```

```
  `email` varchar(45) NOT NULL DEFAULT '',
  `creditCardDetails` varchar(45) DEFAULT NULL,
  `rideHistory` varchar(45) DEFAULT NULL,
  `rating` varchar(45) DEFAULT NULL,
  `reviews` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Driver

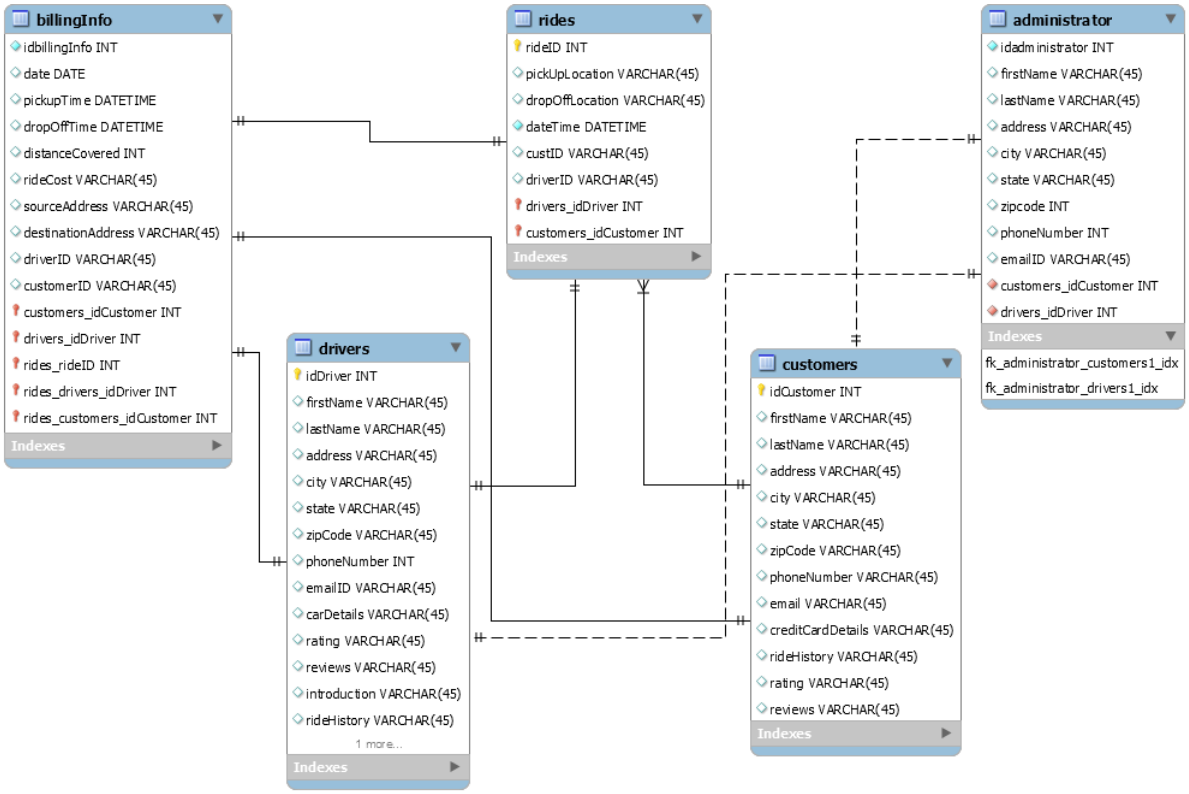
```
CREATE TABLE `drivers` (
  `idDriver` int(11) NOT NULL,
  `firstName` varchar(45) DEFAULT NULL,
  `lastName` varchar(45) DEFAULT NULL,
  `address` varchar(45) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `state` varchar(45) DEFAULT NULL,
  `zipCode` varchar(45) DEFAULT NULL,
  `phoneNumber` int(11) DEFAULT NULL,
  `emailID` varchar(45) NOT NULL DEFAULT '',
  `carDetails` varchar(45) DEFAULT NULL,
  `rating` varchar(45) DEFAULT NULL,
  `reviews` varchar(45) DEFAULT NULL,
  `introduction` varchar(45) DEFAULT NULL,
  `rideHistory` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  `approveStatus` int(11) DEFAULT NULL,
  PRIMARY KEY (`emailID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Rides

```
CREATE TABLE `rides` (
  `rideID` int(11) NOT NULL,
  `pickUpLocation` varchar(45) DEFAULT NULL,
  `dropOffLocation` varchar(45) DEFAULT NULL,
  `dateTime` datetime DEFAULT NULL,
  `custID` varchar(45) DEFAULT NULL,
  `driverID` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`rideID`),
  KEY `emailID_idx` (`driverID`),
  KEY `email_idx` (`custID`),
  CONSTRAINT `email` FOREIGN KEY (`custID`) REFERENCES `customers` (`email`) ON DELETE NO ACTION
ON UPDATE NO ACTION,
```

CONSTRAINT `emailID` FOREIGN KEY (`driverID`) REFERENCES `drivers` (`emailID`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Database Schema



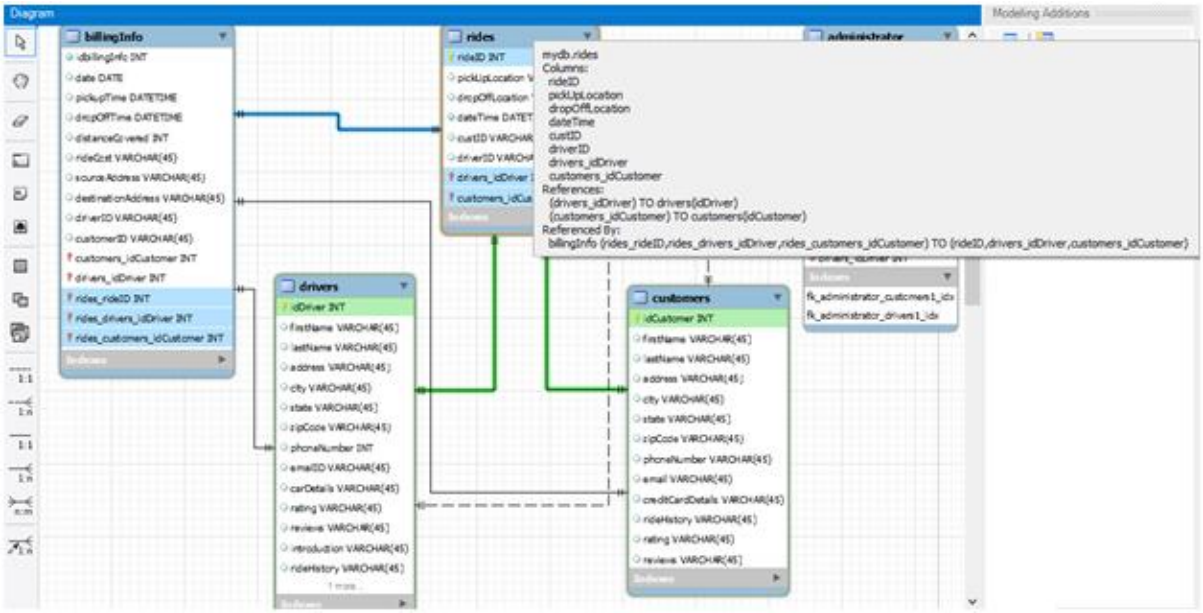


Diagram
Modeling Additions
billingInfo
idbillingInfo INT
date DATE
pickupTime DATETIME
dropOffTime DATETIME
distanceCovered INT
rideCost VARCHAR(45)
sourceAddress VARCHAR(45)
destinationAddress VARCHAR(45)
driverID VARCHAR(45)
customerID VARCHAR(45)
customers_idCustomer INT
drivers_idDriver INT
rides_rideID INT
rides_drivers_idDriver INT
rides_customers_idCustomer INT
rides
rideID INT
pickUpLocation
dropOffLocation
dateTime
custID VARCHAR
driverID VARCHAR
drivers_idDriver
customers_idCu
mydb.rides
Columns:
rideID
pickUpLocation
dropOffLocation
dateTime
custID
driverID
drivers_idDriver
customers_idCustomer
References:
(drivers_idDriver) TO drivers(idDriver)
(customers_idCustomer) TO customers(idCustomer)
Referenced By:
billingInfo (rides_rideID,rides_drivers_idDriver,rides_customers_idCustomer) TO (rideID,drivers_idDriver,customers_idCustomer)
administrator
fk_administrator_customers1_idx
fk_administrator_drivers1_idx
drivers
idDriver INT
firstName VARCHAR(45)
lastName VARCHAR(45)
address VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipCode VARCHAR(45)
phoneNumber INT
emailID VARCHAR(45)
carDetails VARCHAR(45)
rating VARCHAR(45)
reviews VARCHAR(45)
introduction VARCHAR(45)
rideHistory VARCHAR(45)
customers
idCustomer INT
firstName VARCHAR(45)
lastName VARCHAR(45)
address VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipCode VARCHAR(45)
phoneNumber VARCHAR(45)
email VARCHAR(45)
creditCardDetails VARCHAR(45)
rideHistory VARCHAR(45)
rating VARCHAR(45)
reviews VARCHAR(45)

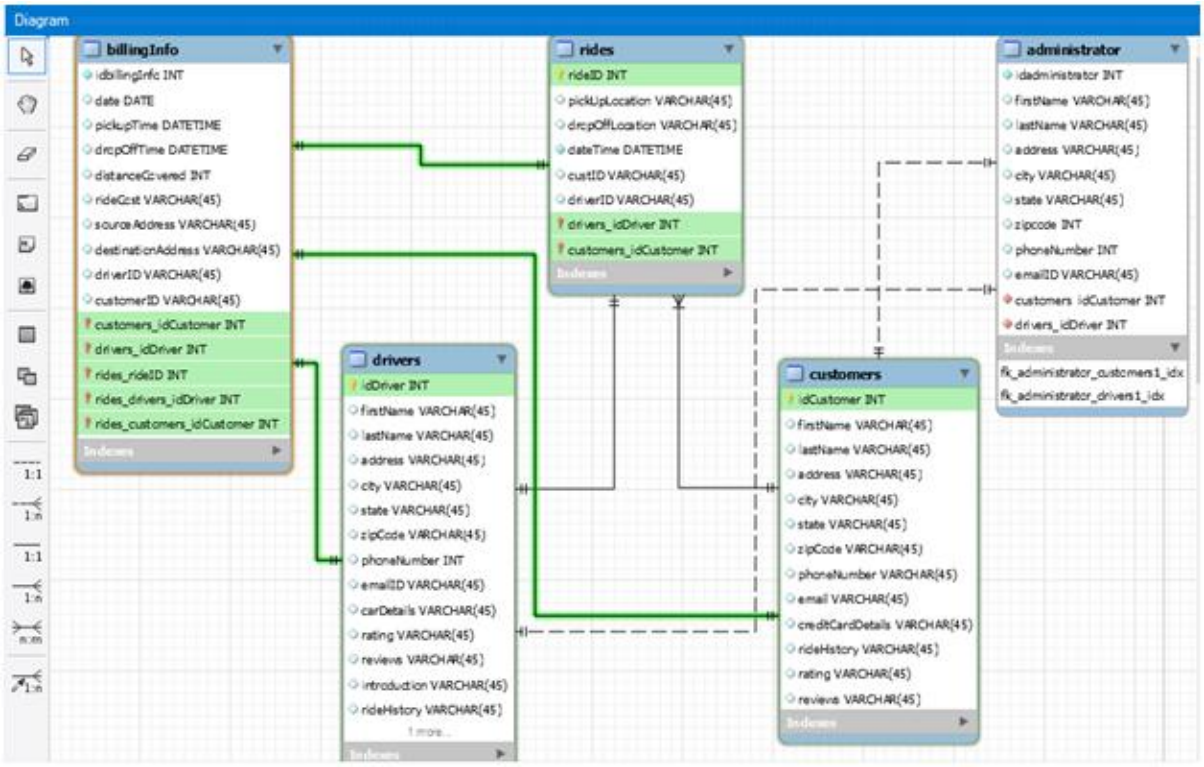


Diagram
billingInfo
idbillingInfo INT
date DATE
pickupTime DATETIME
dropOffTime DATETIME
distanceCovered INT
rideCost VARCHAR(45)
sourceAddress VARCHAR(45)
destinationAddress VARCHAR(45)
driverID VARCHAR(45)
customerID VARCHAR(45)
customers_idCustomer INT
drivers_idDriver INT
rides_rideID INT
rides_drivers_idDriver INT
rides_customers_idCustomer INT
rides
rideID INT
pickUpLocation VARCHAR(45)
dropOffLocation VARCHAR(45)
dateTime DATETIME
custID VARCHAR(45)
driverID VARCHAR(45)
drivers_idDriver INT
customers_idCustomer INT
administrator
idadministrator INT
firstName VARCHAR(45)
lastName VARCHAR(45)
address VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipcode INT
phoneNumber INT
emailID VARCHAR(45)
customers_idCustomer INT
drivers_idDriver INT
fk_administrator_customers1_idx
fk_administrator_drivers1_idx
drivers
idDriver INT
firstName VARCHAR(45)
lastName VARCHAR(45)
address VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipCode VARCHAR(45)
phoneNumber INT
emailID VARCHAR(45)
carDetails VARCHAR(45)
rating VARCHAR(45)
reviews VARCHAR(45)
introduction VARCHAR(45)
rideHistory VARCHAR(45)
customers
idCustomer INT
firstName VARCHAR(45)
lastName VARCHAR(45)
address VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
zipCode VARCHAR(45)
phoneNumber VARCHAR(45)
email VARCHAR(45)
creditCardDetails VARCHAR(45)
rideHistory VARCHAR(45)
rating VARCHAR(45)
reviews VARCHAR(45)

Resource management

To improve the efficiency and performance of the system, we used a Radis Server. Our aim was to use data structures provided by the radis server to hold data values for listing and search functionalities. We know that the key-value pair in Radis is not limited to just string values and are capable of holding more complex data structures like JPEG file or empty string.

In our system, we have implemented the following features:

Radis Lists: We have used the 'Lists' data structure for all listing operations. The idea is to store all values in the Radis lists, such that when a user requests the list of say categories, we fetch the data from the Radis Lists instead of hitting database with 'Select' statements, which is one of the most expensive operations. Also, this avoids using 'select * from' statements, which degrades the performance of the system.

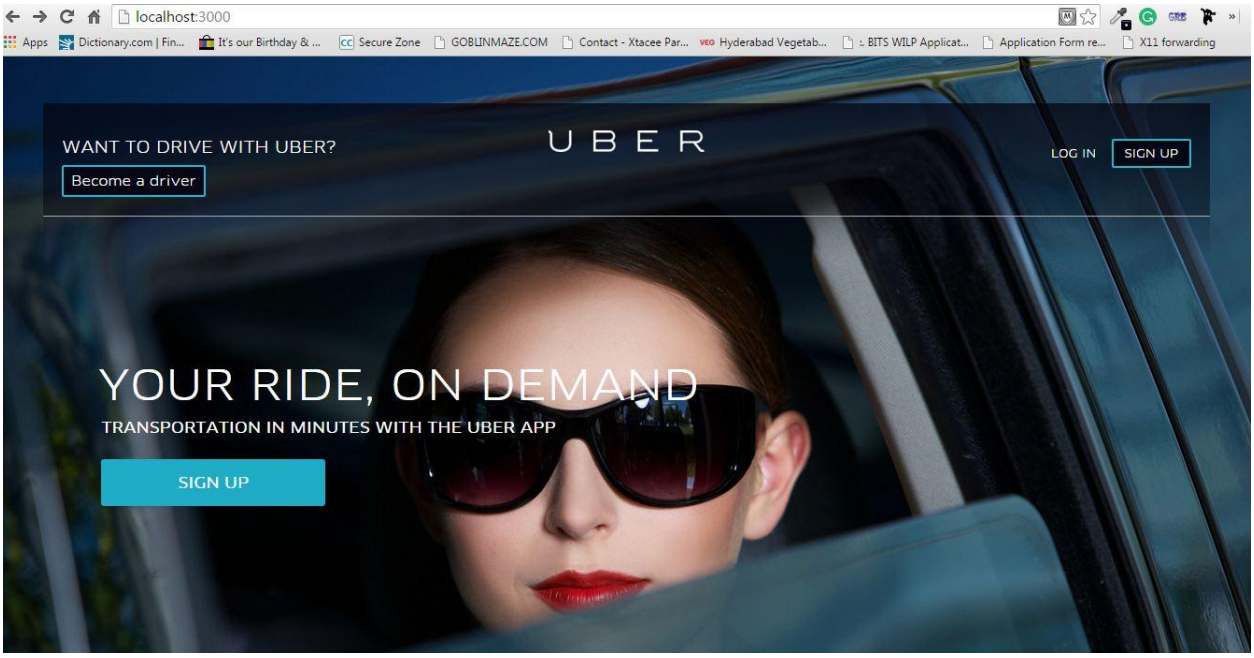
Hashes: This is used for search operations in the system; each key is a query which has an associated value as a json string of result. Every time an update is made, the key-value pair is also update. Thus, in this way the database select hits are reduced, thereby improving the performance of the system.

Connection Pooling

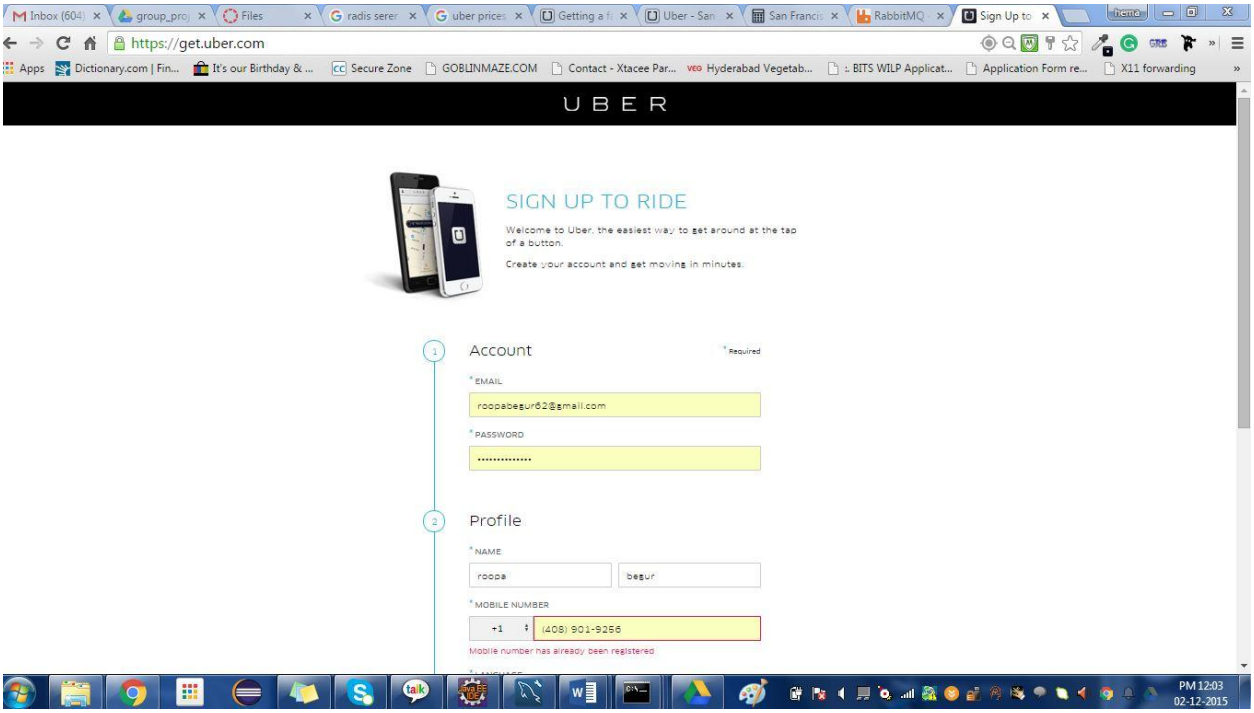
Nodejs provide connection pooling in multiple ways, either u can use library like generic pool or create it directly without using any library. We have implemented non generic pool, wherein we are creating a poo; pf 10 databse connections.

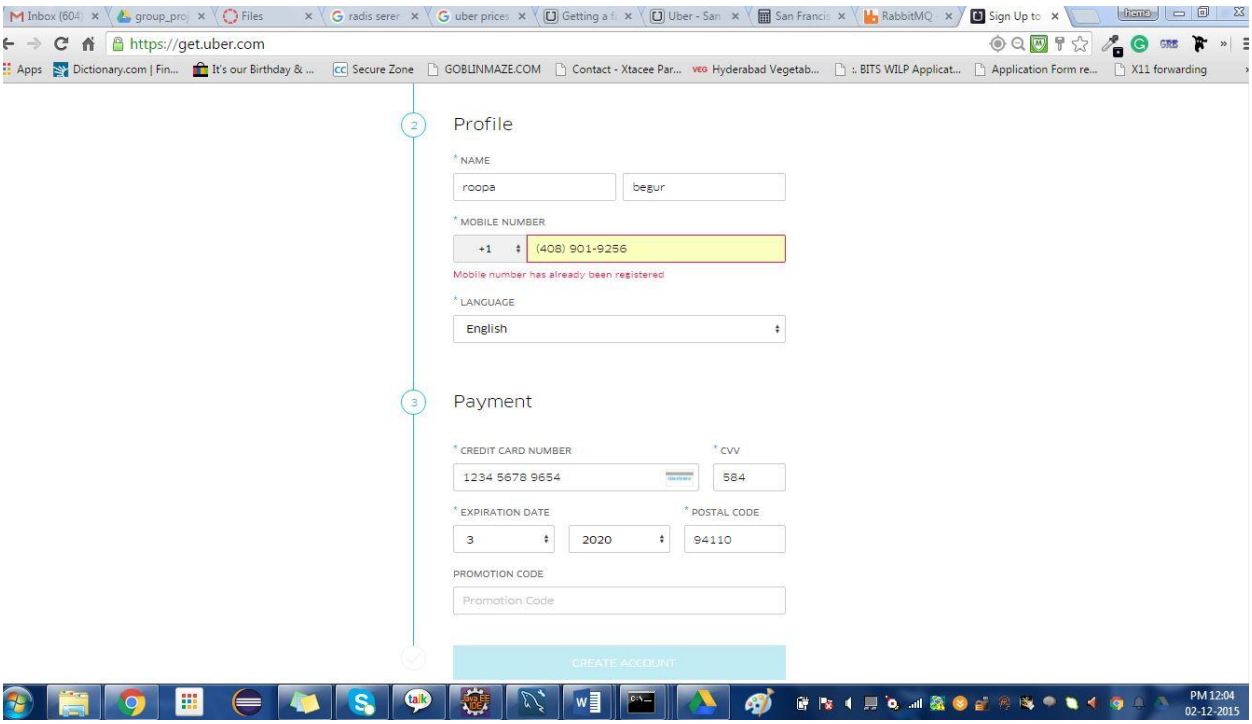
Functionalities

Home page

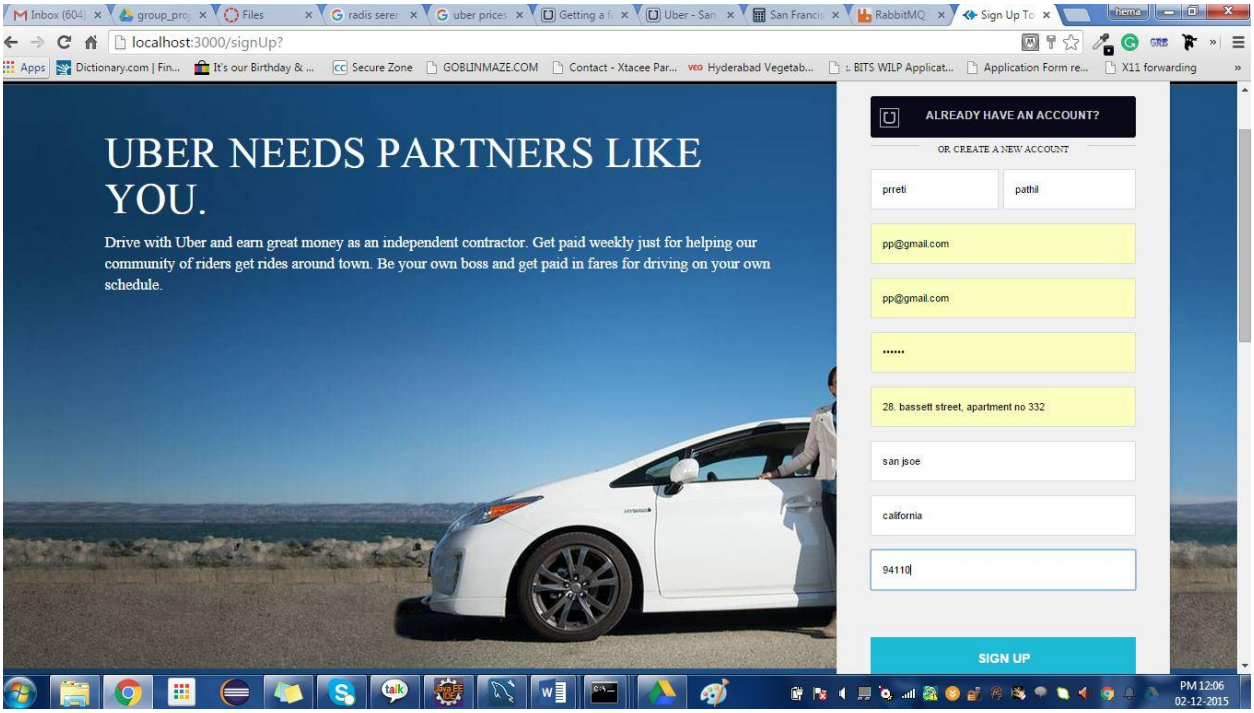


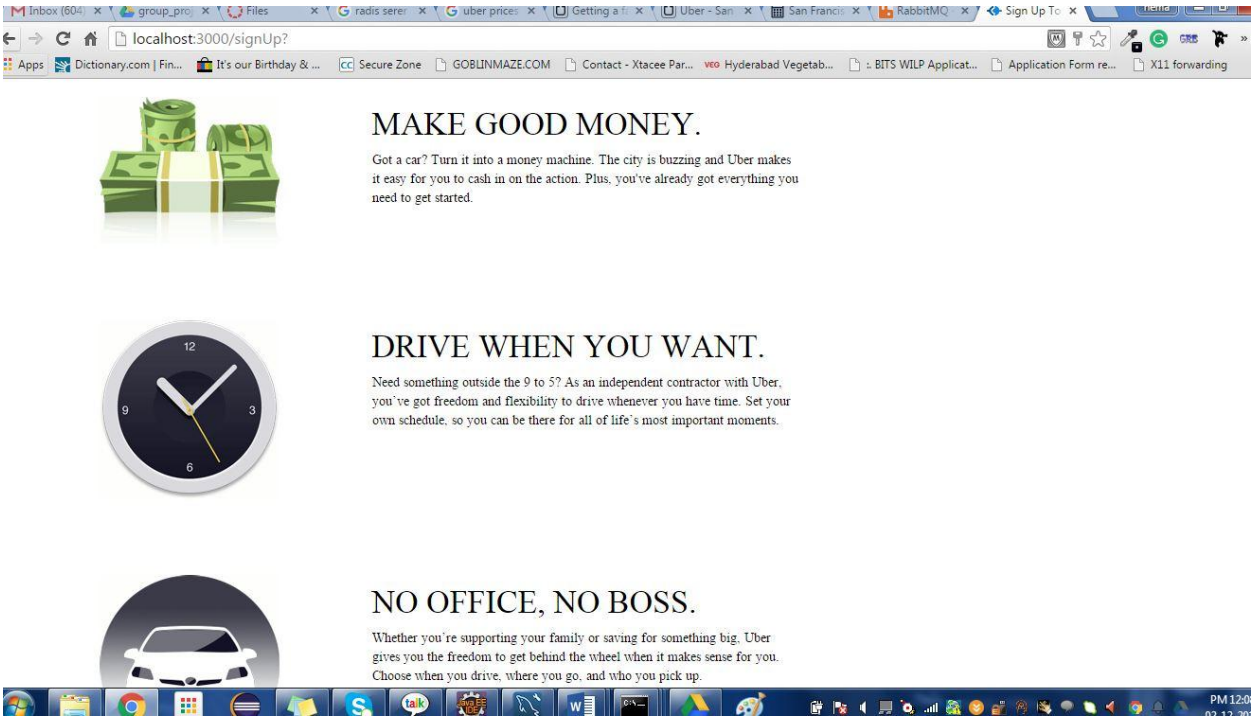
Customer signup



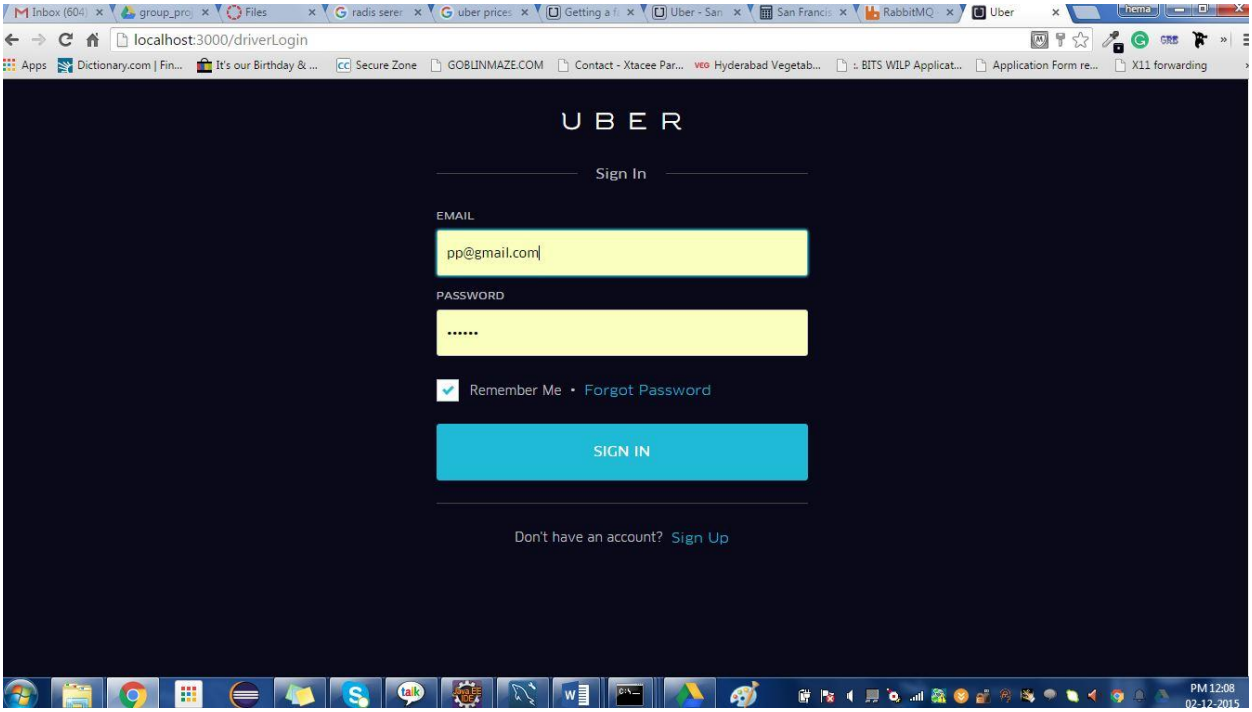


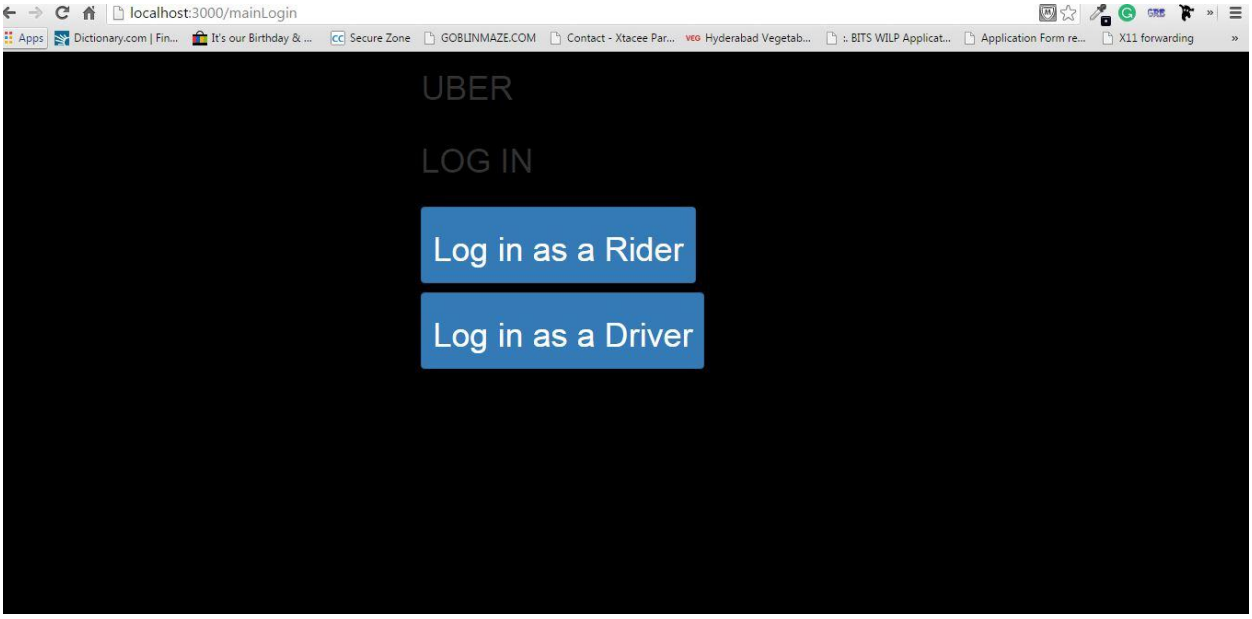
Driver signup



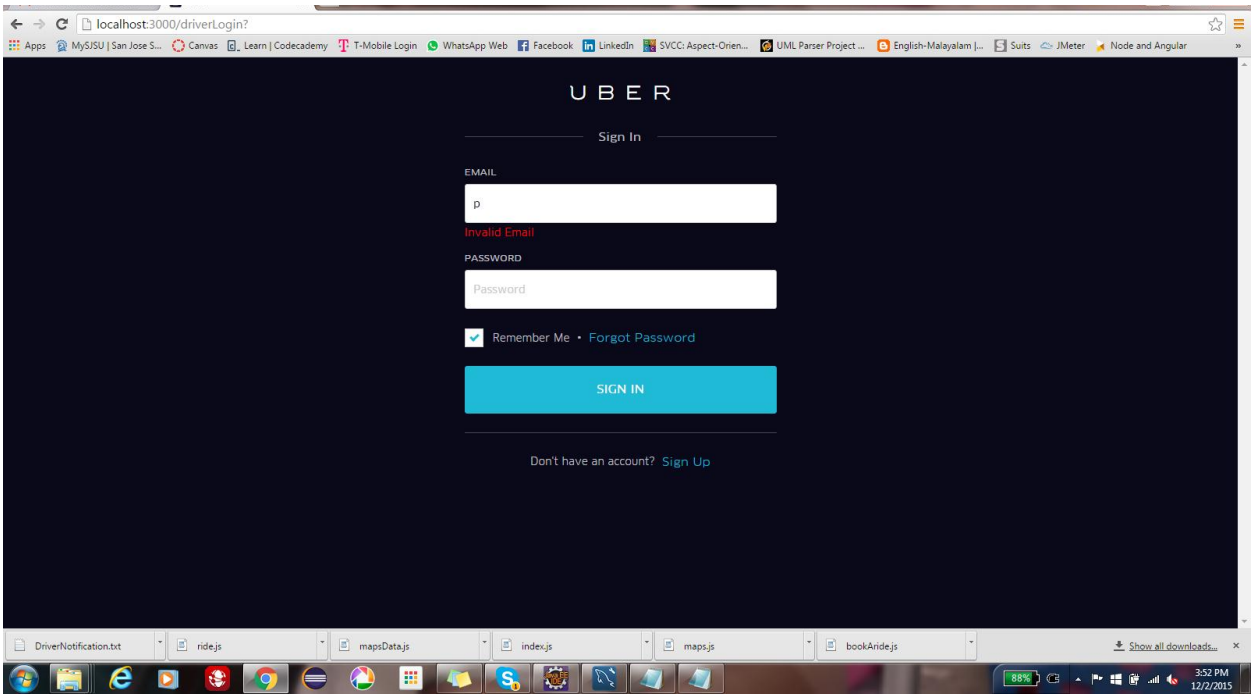


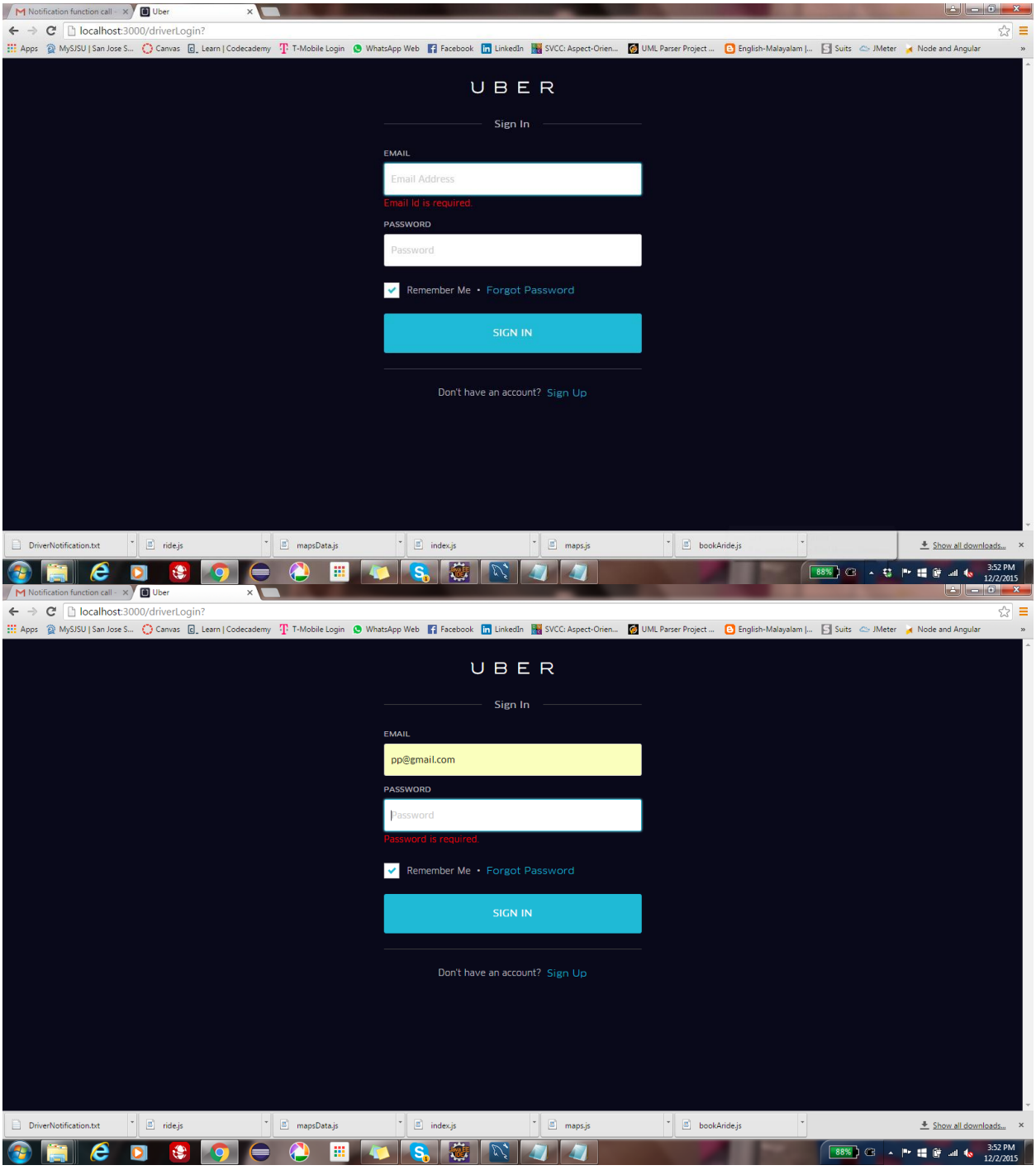
Login page





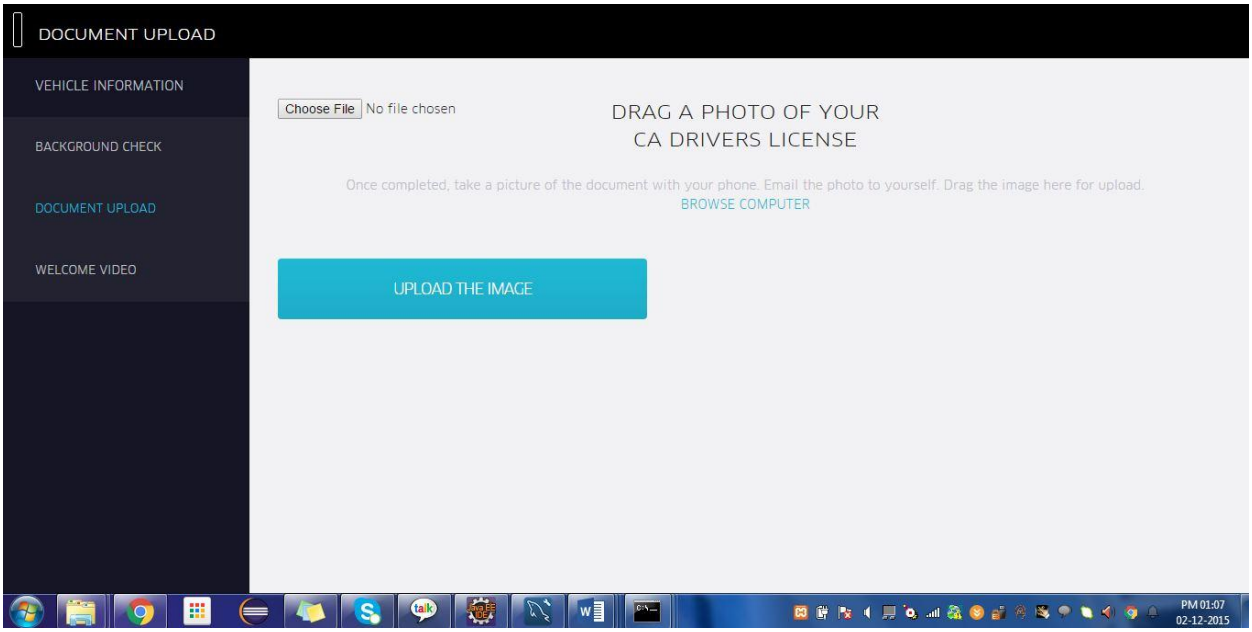
Verification of user input


localhost:3000/driverLogin?
UBER
Sign In
EMAIL
p
Invalid Email
PASSWORD
Password
Remember Me • Forgot Password
SIGN IN
Don't have an account? Sign Up
DriverNotification.txt
ride.js
mapsData.js
index.js
maps.js
bookAride.js
Show all downloads...
3:52 PM
12/2/2015

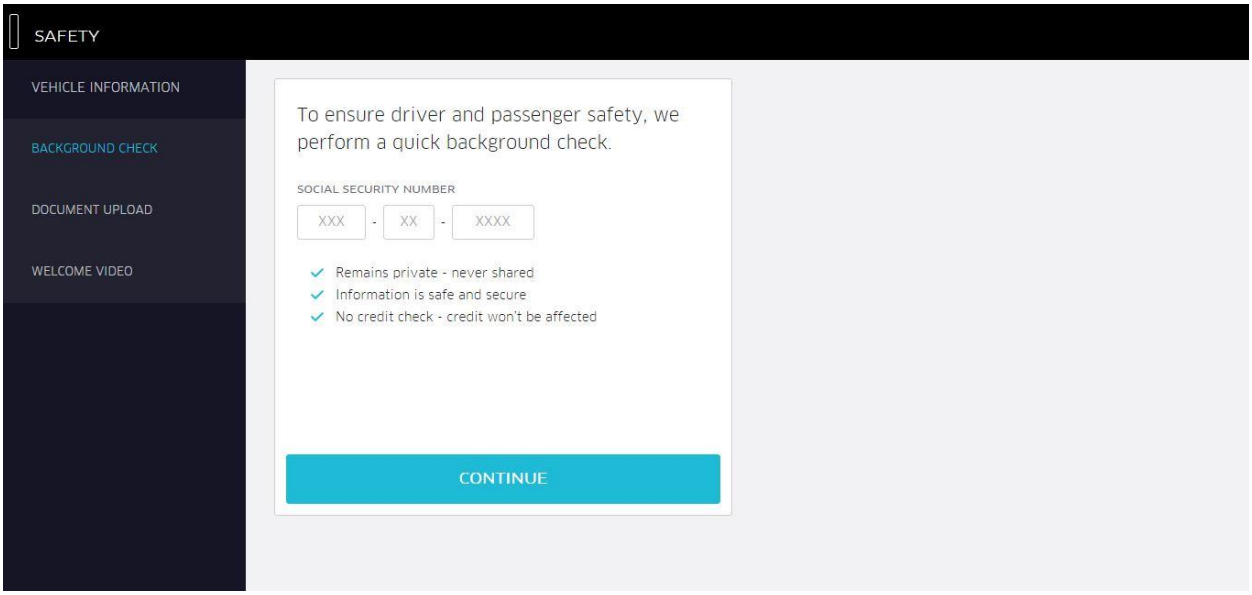

Notification function call -
Uber
localhost:3000/driverLogin?
Apps
MySJSU | San Jose S...
Canvas
Learn | Codecademy
T-Mobile Login
WhatsApp Web
Facebook
LinkedIn
SVCC: Aspect-Orien...
UML Parser Project ...
English-Malayalam |...
Suits
JMeter
Node and Angular
UBER
Sign In
EMAIL
Email Address
Email Id is required.
PASSWORD
Password
Remember Me • Forgot Password
SIGN IN
Don't have an account? Sign Up
DriverNotification.txt
ride.js
mapsData.js
index.js
maps.js
bookAride.js
Show all downloads...
88%
3:52 PM
12/2/2015
Notification function call -
Uber
localhost:3000/driverLogin?
Apps
MySJSU | San Jose S...
Canvas
Learn | Codecademy
T-Mobile Login
WhatsApp Web
Facebook
LinkedIn
SVCC: Aspect-Orien...
UML Parser Project ...
English-Malayalam |...
Suits
JMeter
Node and Angular
UBER
Sign In
EMAIL
pp@gmail.com
PASSWORD
Password
Password is required.
Remember Me • Forgot Password
SIGN IN
Don't have an account? Sign Up
DriverNotification.txt
ride.js
mapsData.js
index.js
maps.js
bookAride.js
Show all downloads...
88%
3:52 PM
12/2/2015

Driver pages

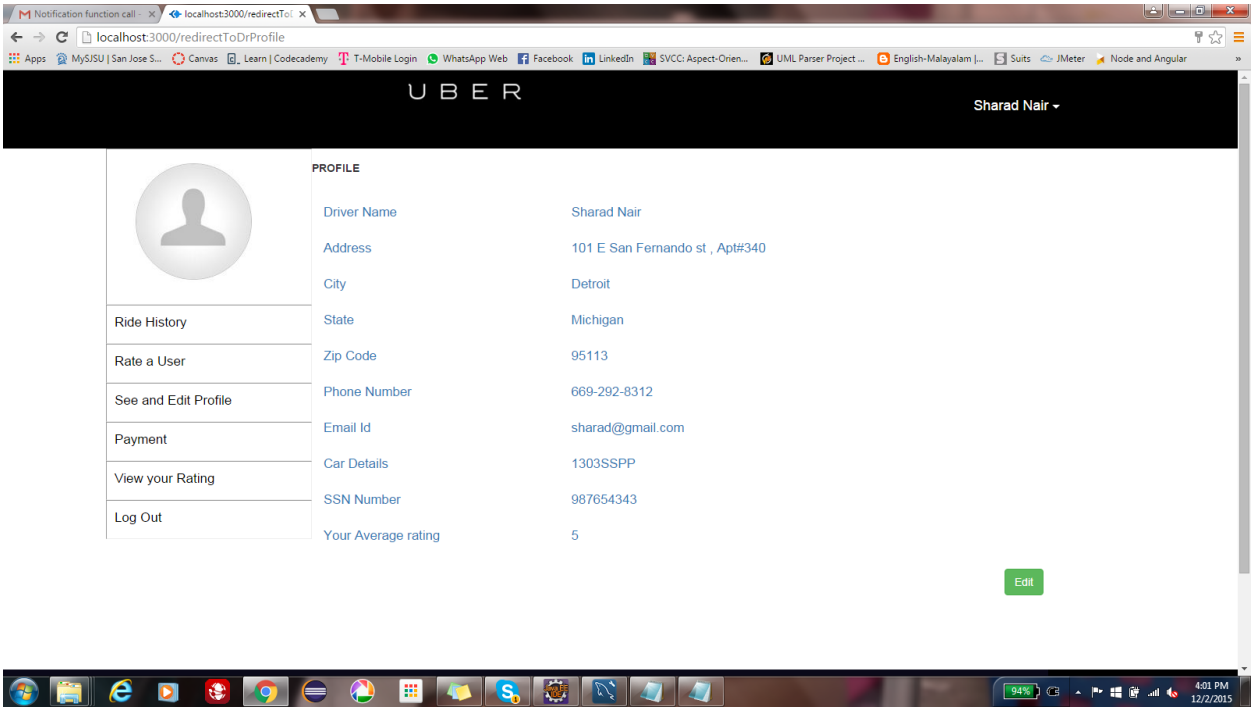
Document uploading



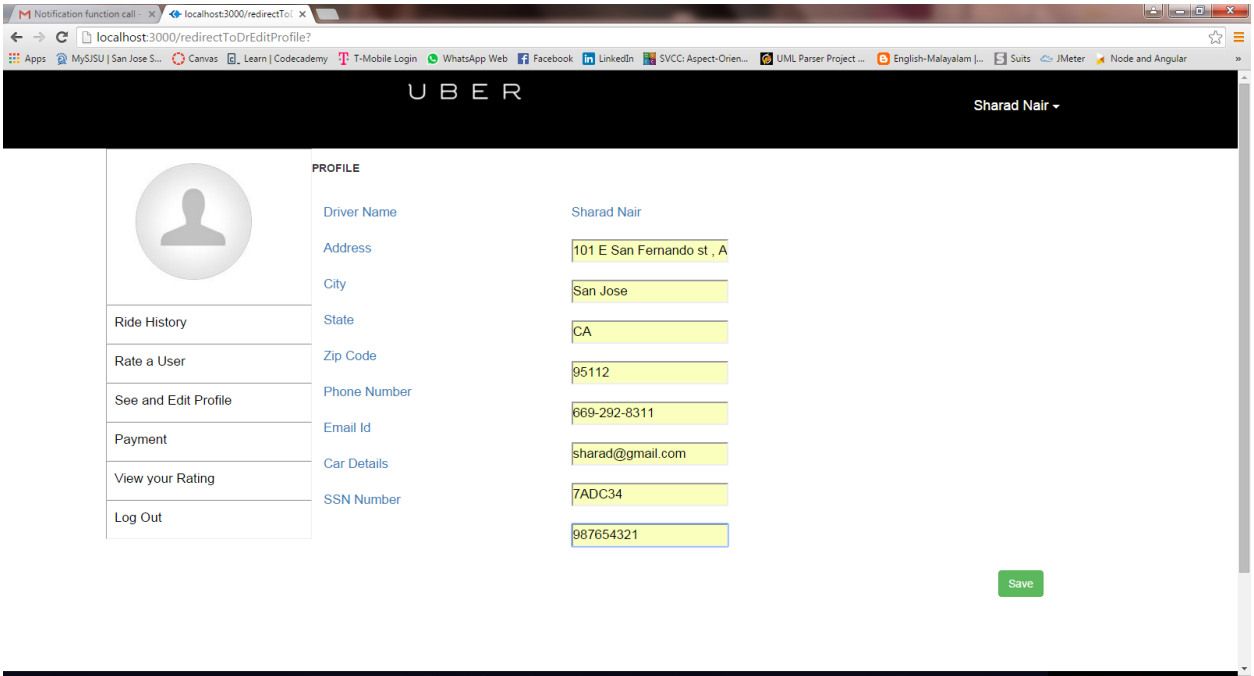
Background check on driver

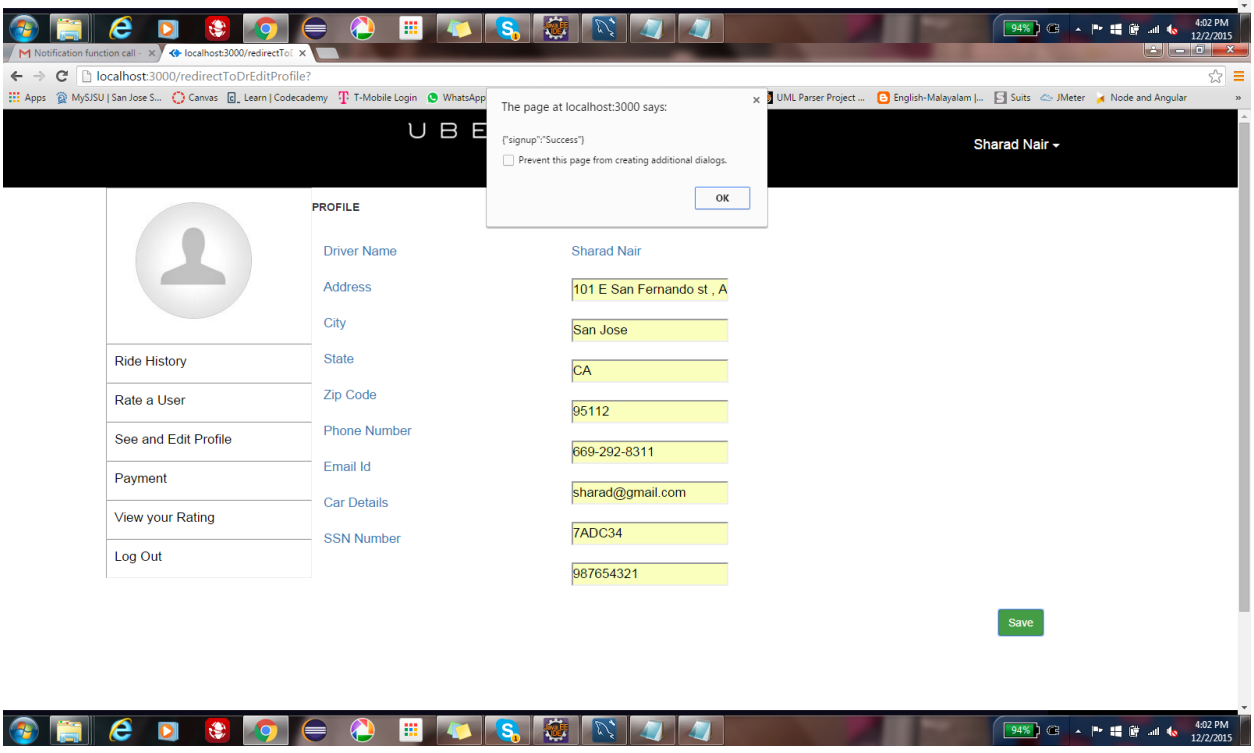


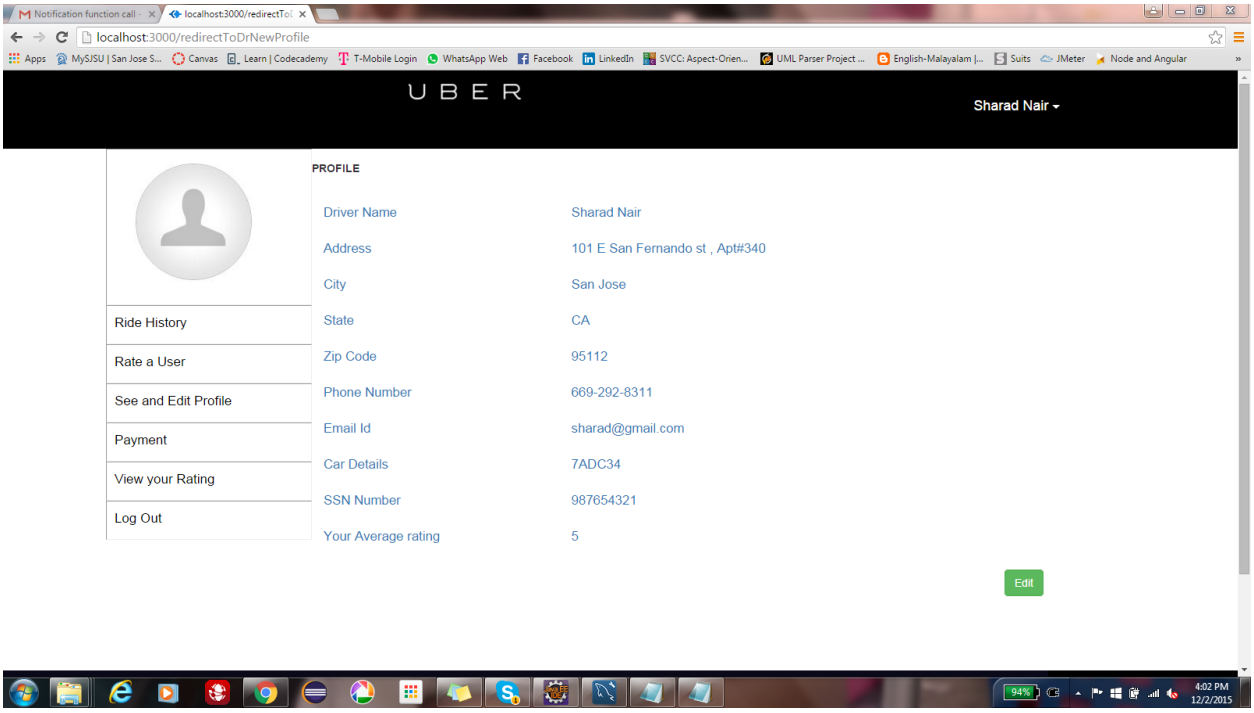
View information



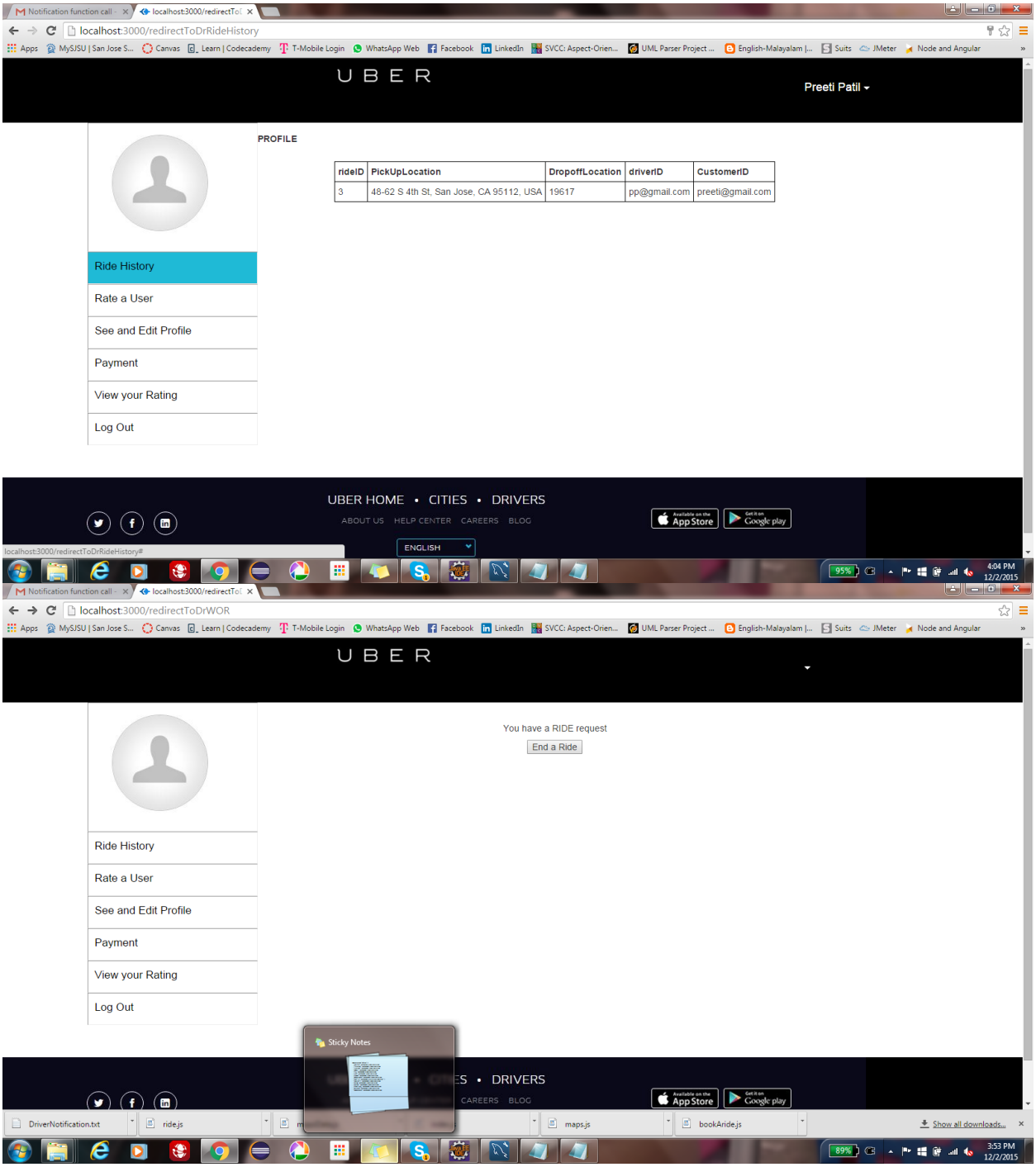
Edit information

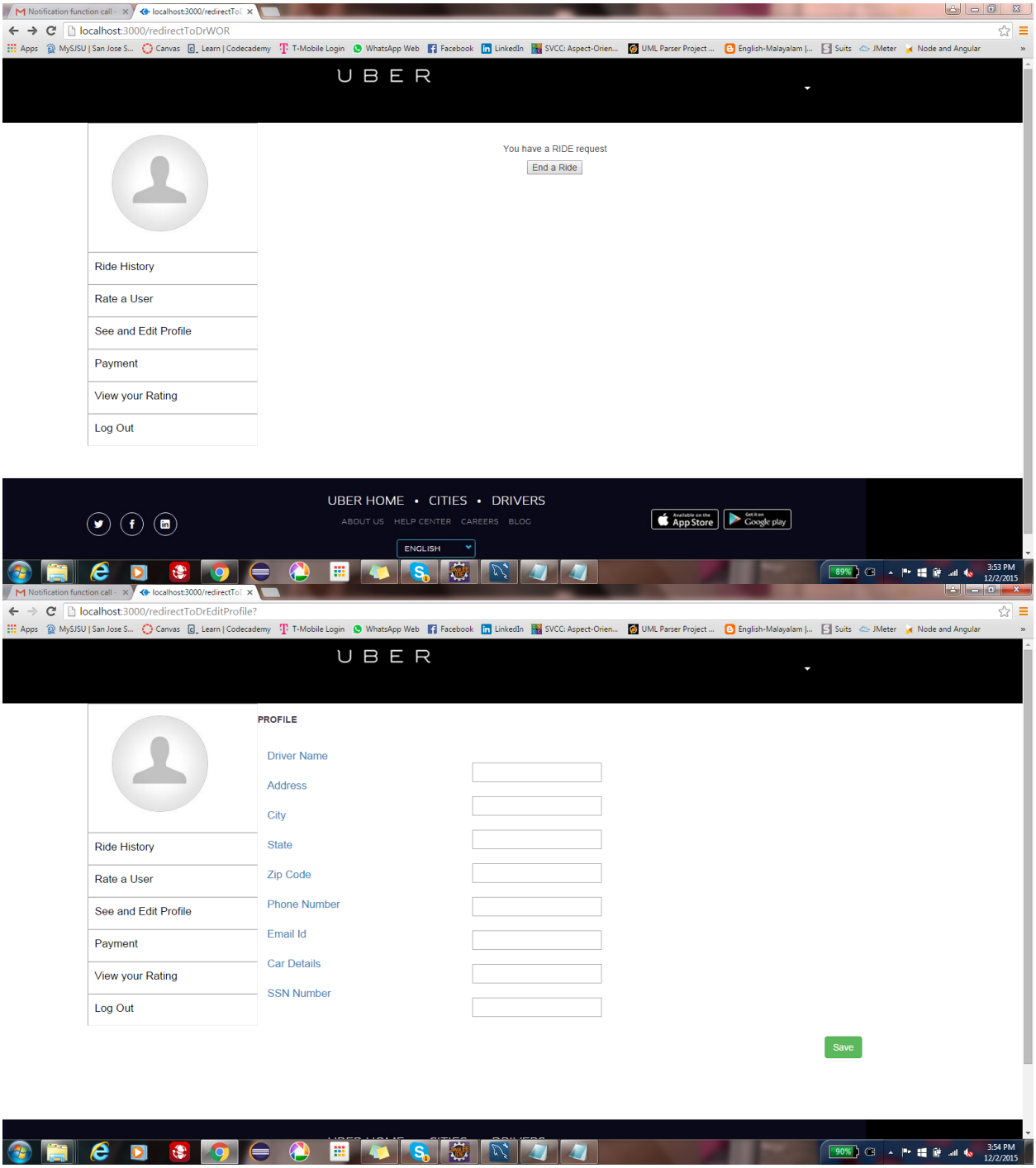


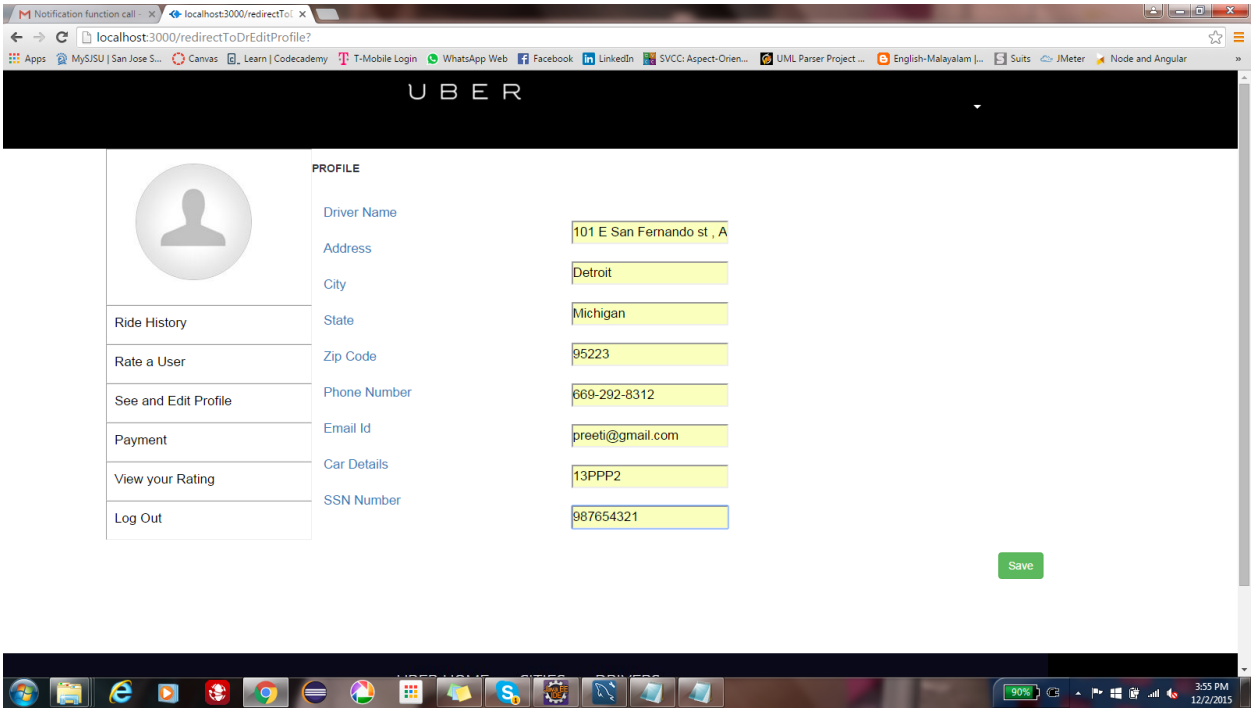



localhost:3000/redirectToDrNewProfile
UBER
Sharad Nair
PROFILE
Ride History
Rate a User
See and Edit Profile
Payment
View your Rating
Log Out
Driver Name
Sharad Nair
Address
101 E San Fernando st , Apt#340
City
San Jose
State
CA
Zip Code
95112
Phone Number
669-292-8311
Email Id
sharad@gmail.com
Car Details
7ADC34
SSN Number
987654321
Your Average rating
5
Edit

Ride History

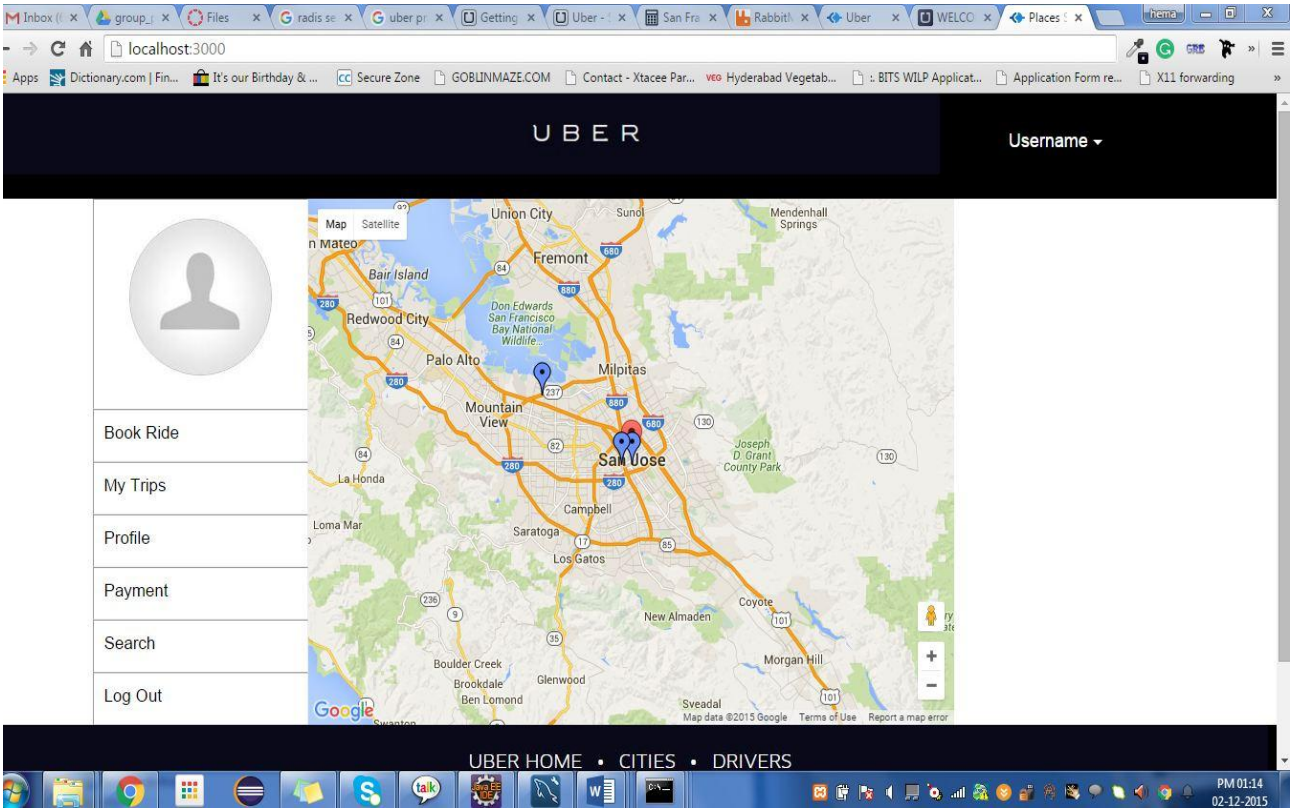



localhost:3000/redirectToDrWOR
UBER
You have a RIDE request
End a Ride
Ride History
Rate a User
See and Edit Profile
Payment
View your Rating
Log Out
UBER HOME • CITIES • DRIVERS
ABOUT US HELP CENTER CAREERS BLOG
ENGLISH
App Store
Google play
localhost:3000/redirectToDrEditProfile?
UBER
PROFILE
Driver Name
Address
City
State
Zip Code
Phone Number
Email Id
Car Details
SSN Number
Save
Ride History
Rate a User
See and Edit Profile
Payment
View your Rating
Log Out

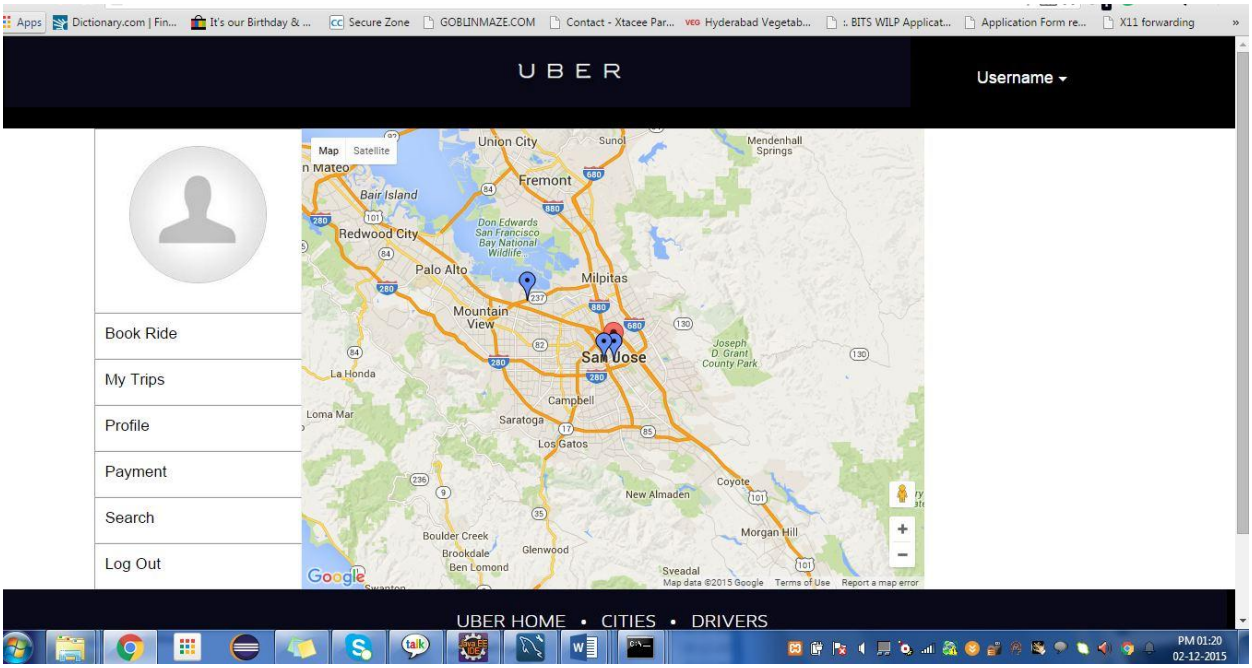


Customer features

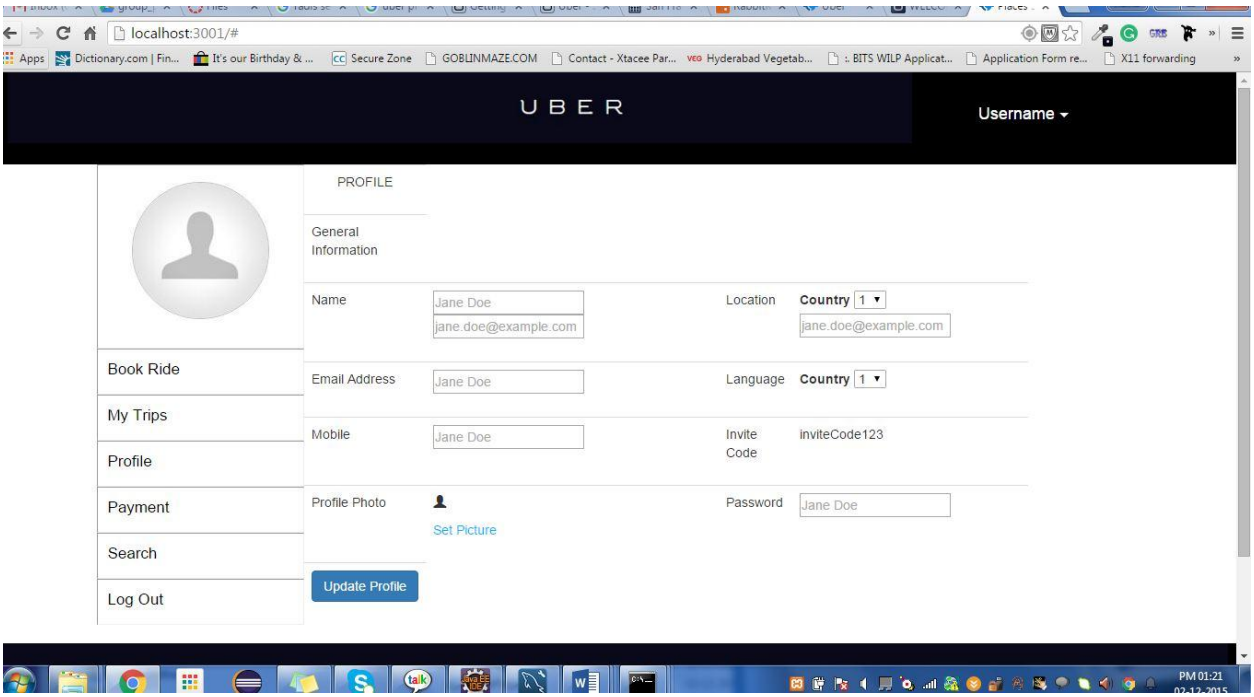
Customer home page



Search for drivers

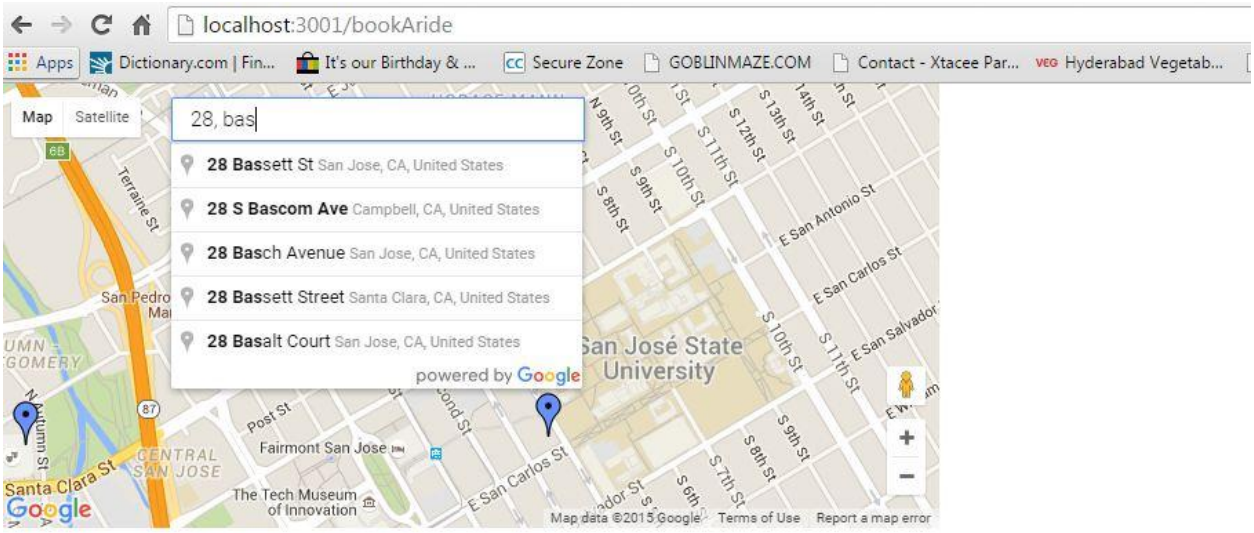


Update profile

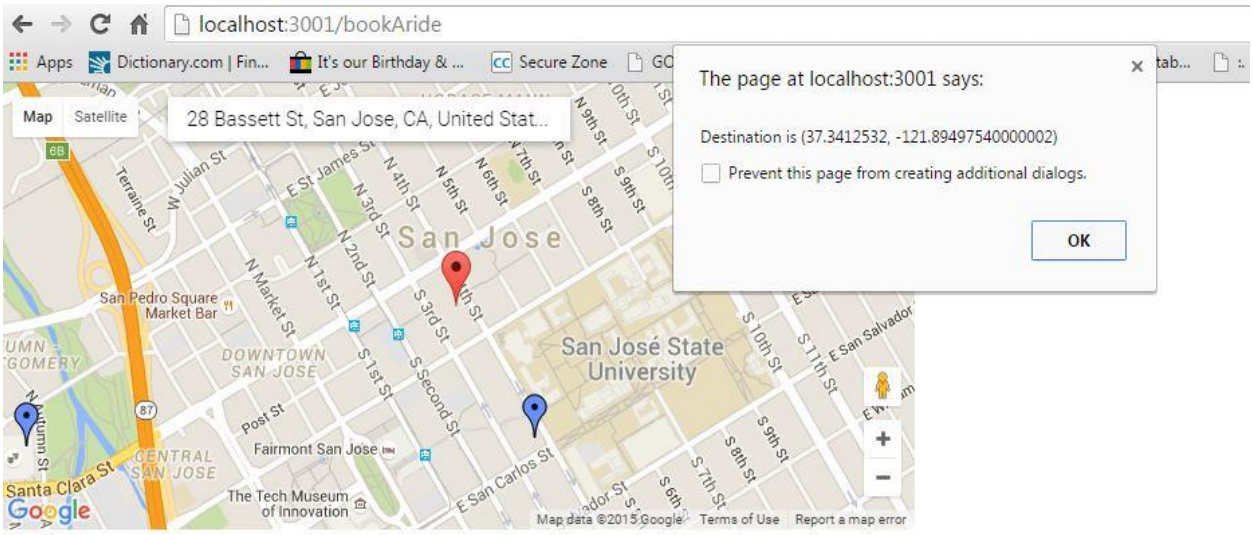


Book a ride

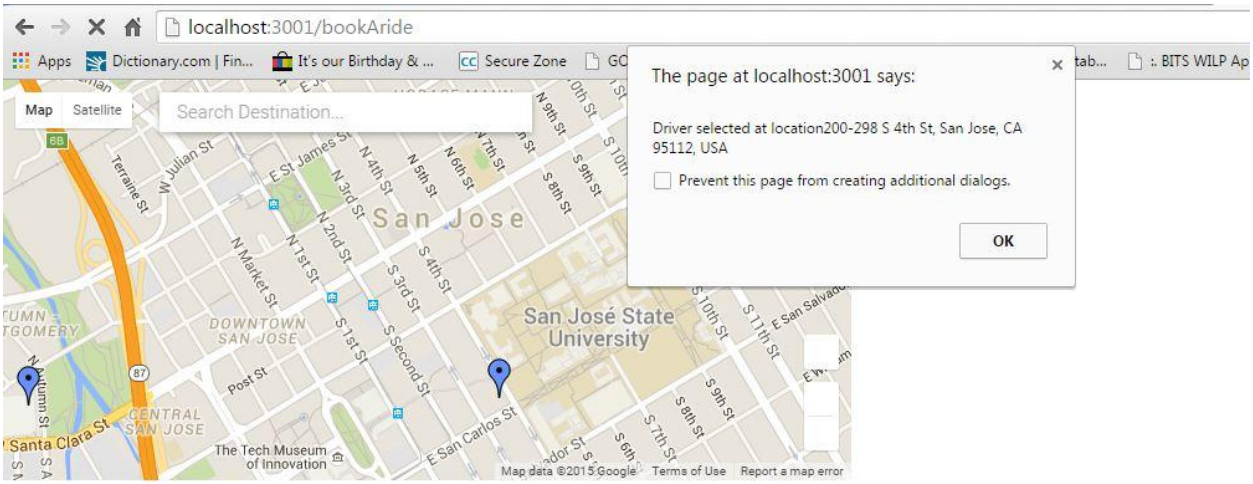
Enter destination



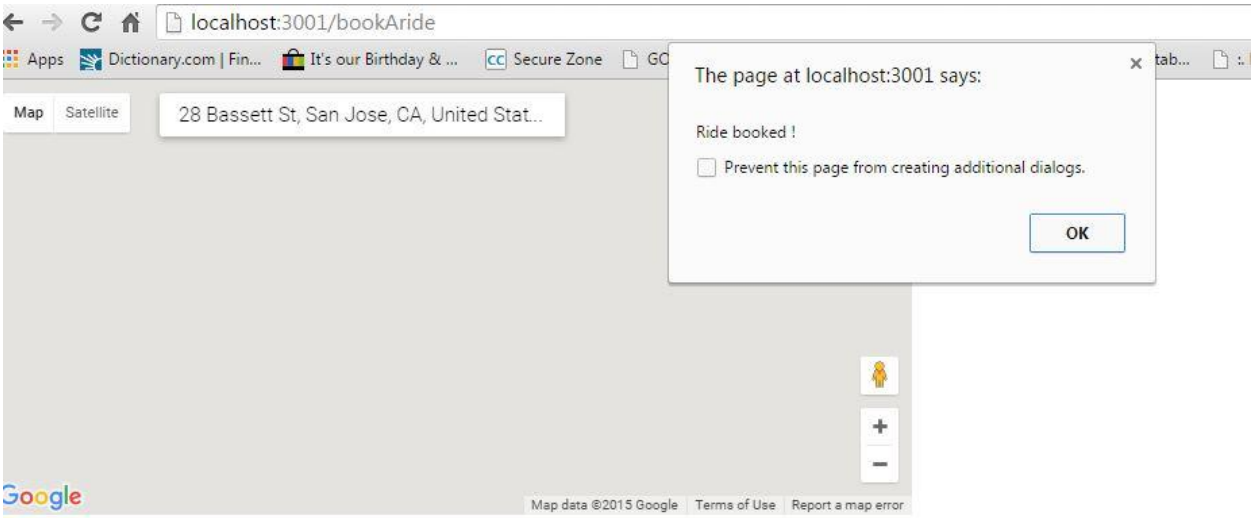
Search for drivers



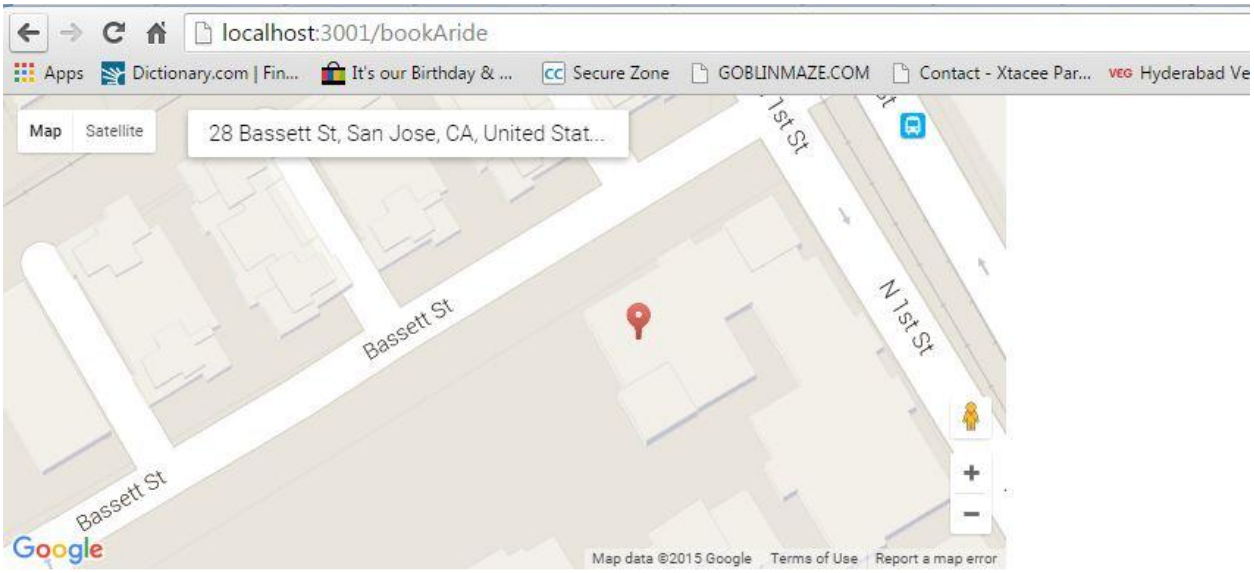
Select a driver



Ride booked

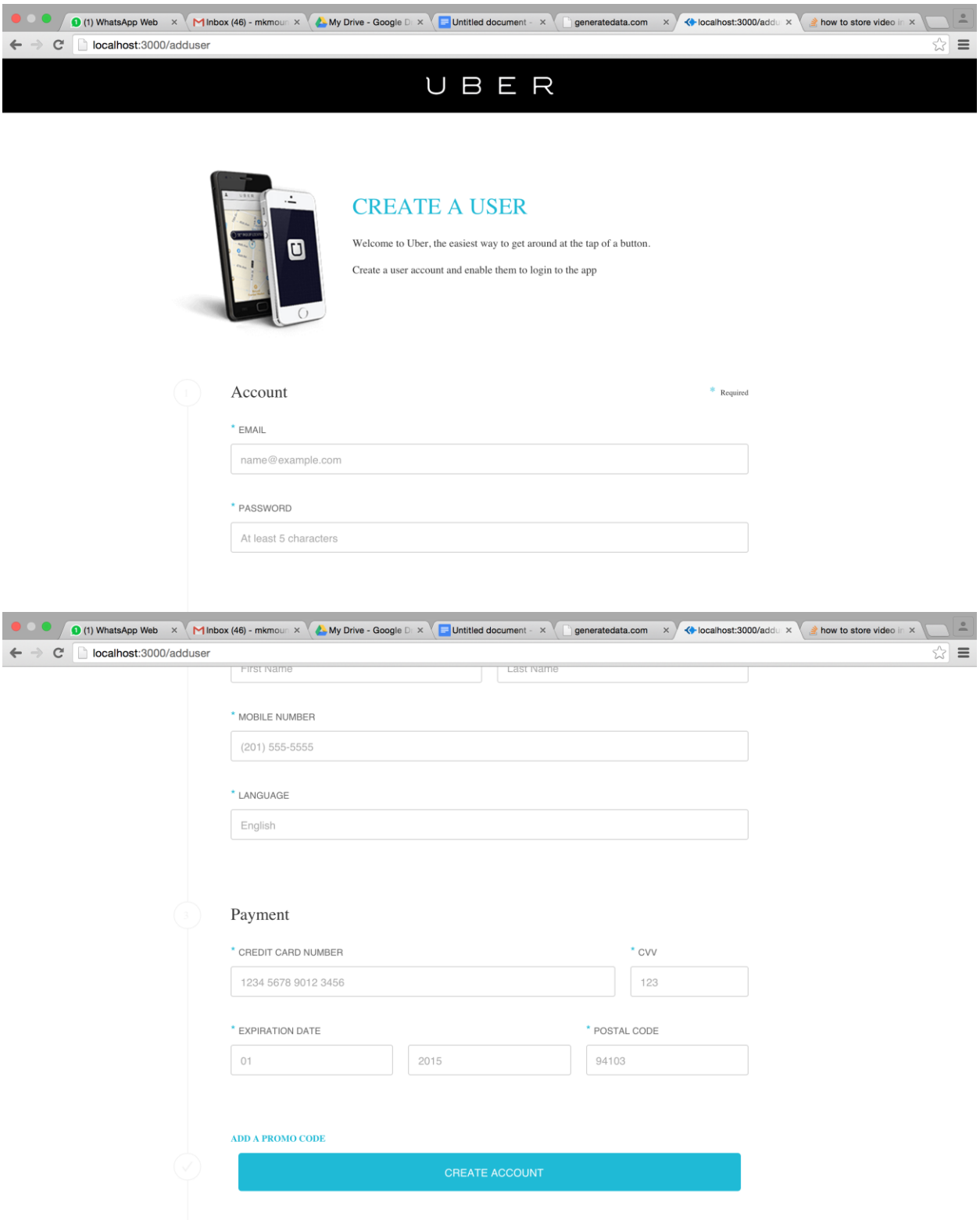


Show driver location

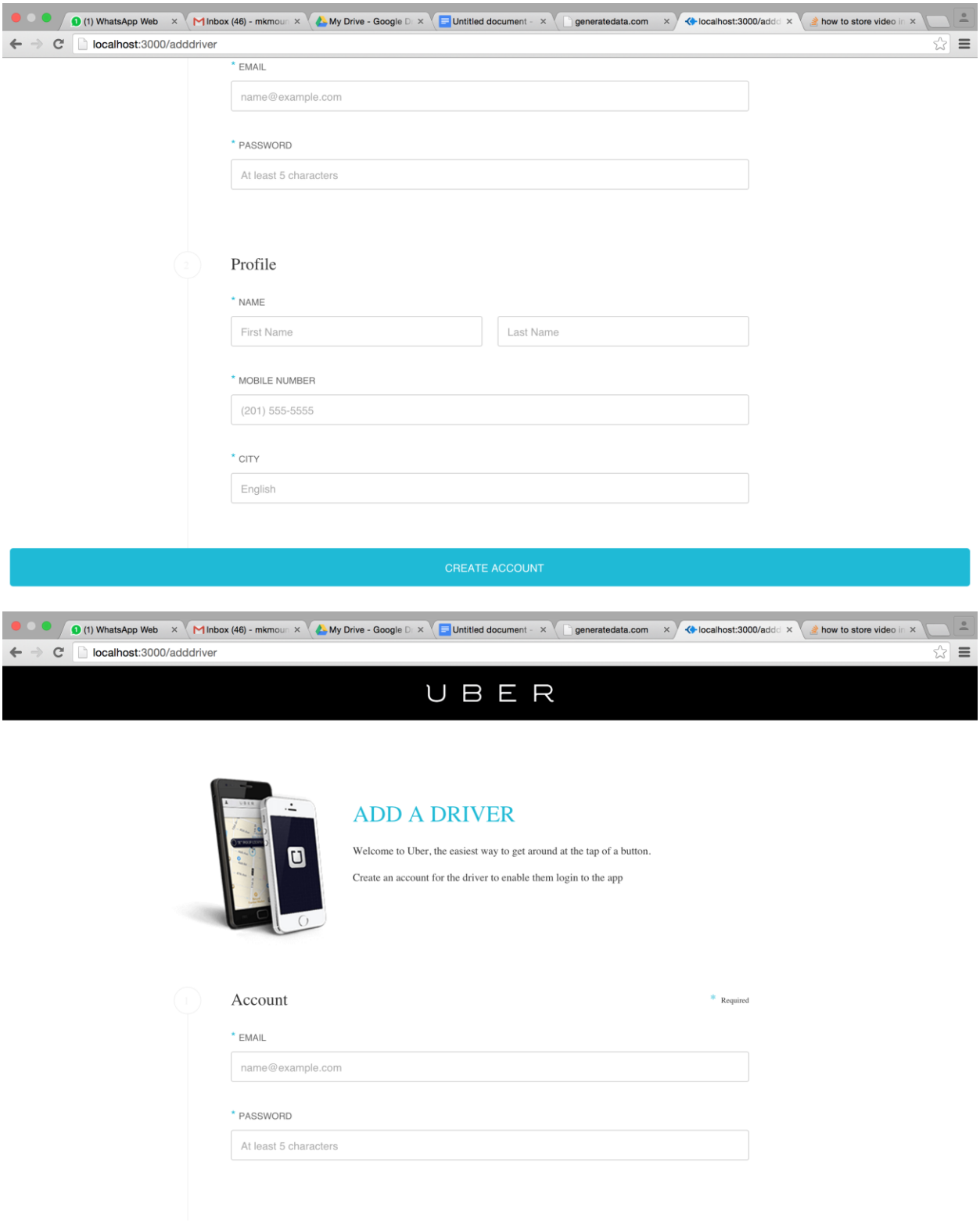


Admin features

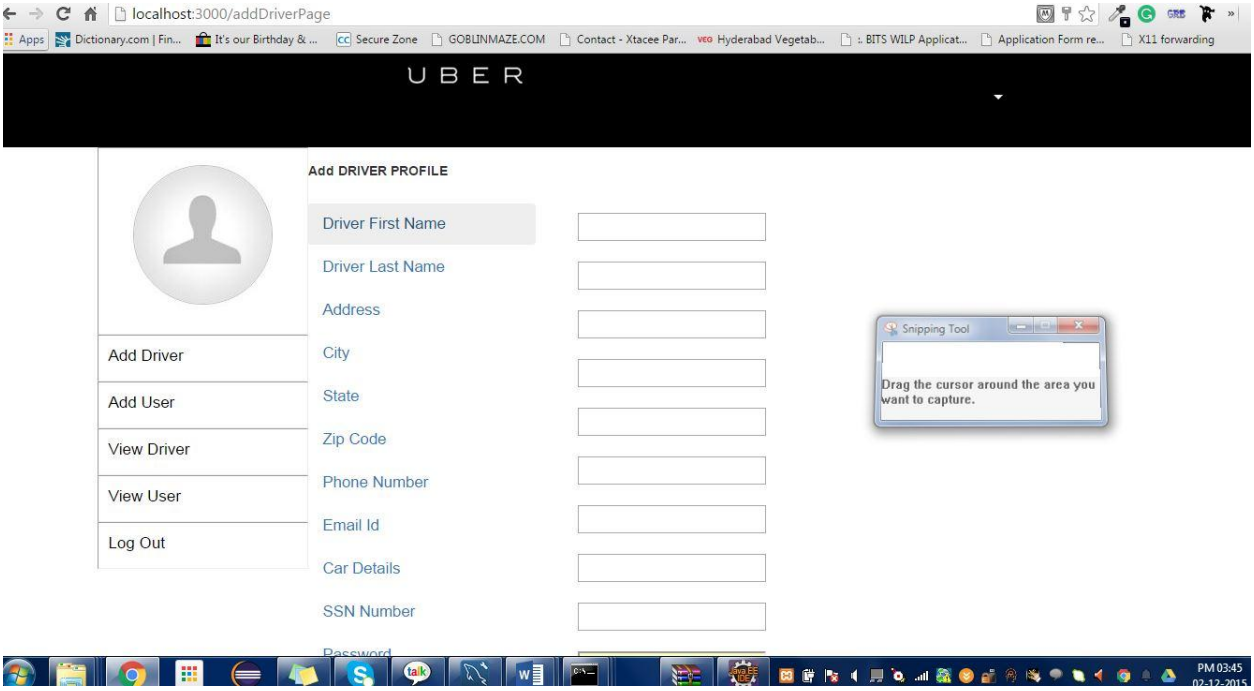
Add user



Add driver


localhost:3000/adddriver
* EMAIL
name@example.com
* PASSWORD
At least 5 characters
Profile
* NAME
First Name
Last Name
* MOBILE NUMBER
(201) 555-5555
* CITY
English
CREATE ACCOUNT
localhost:3000/adddriver
UBER
ADD A DRIVER
Welcome to Uber, the easiest way to get around at the tap of a button.
Create an account for the driver to enable them login to the app
Account
* Required
* EMAIL
name@example.com
* PASSWORD
At least 5 characters

Admin Home page



Scalability, Performance and Reliability:

While developing a web application, many points are to be kept in mind – The data consistency, whether the user is accessing the updated data, which in turn signifies the reliability of a web site. Similarly, while most of the web applications are distributed, they should be capable of handling load of multiple users, keeping in consideration the speed of execution. The success of a web application relies majorly on these three paradigms. We have implemented the following techniques to make our more robust and adaptive:

SQL Caching

Database 'Select' is one of the most expensive operations and thus special consideration should be taken to optimize the searches. We have implemented the following techniques in our project:

Pagination: Radis Lists

As discussed before in Radis Server, the listing is being done by saving the data in a result set, typically Radis Lists and shown in pages. In our system, we have implemented 'pagination' where each page displays 10 values. Thus, this implementation technique reduces the database hits and reducing usage of 'Select' statement. Each hit stored is update when the database is updated. Moreover, since the retrieval is being done from a list instead of database, the retrieval time reduces thereby increasing the efficiency of the system.

Radis Hash-Map Searches

Search module requires 'Select' statements which is the most expensive database operation. As discussed before in Radis Server, to reduce cost of system and increase its efficiency we have implemented our searching module using hash maps, where we the keys are simple search queries and its corresponding result value is a json string. This technique reduces the number of database hits, thereby increasing the performance and reducing the cost drastically. When an update in database is encountered, the previous json string also gets updated.

Given below is an example wherein each time a user requests for a ride history, it would first check the radis list, if it is present then it will fetch the data from there, otherwise it will call the a function (displayed below) which fetches the data from the data base. The list is flushed each time there is an update, thereby ensuring that the data is consistent.

```
exports.handle_request = function(msg,callback){
	var res={};
	var trip=[];


	console.log("in mytrips handle req"+ msg.userid);
	var loginUser = "select * from rides where CustId ='" + msg.userid + "'";
	console.log(loginUser);

	if (true) {
		mysql.fetchData(function(err, user) {
			if (err) {
				throw err;

			} else {
				if (user.length > 0) {
					console.log("Trips exists");
					res.code="200";
					res.value = "Success validation";
					res.car = user[0].car;
					res.fare = user[0].fare;
					res.driver = user[0].driver;
					res.city = user[0].city;
					res.card_number = user[0].card_number;
					res.pickup = user[0].pickup;

					console.log(res.car);
					console.log(res.fare);
					console.log(res.driver);
					console.log(res.city);
					console.log(res.card_number);
					console.log(res.pickup);
					callback(null,res);
				}
				else {
					res.code="401";
					res.value="no trips";
					callback(null,res);
				}
			}

		},loginUser);
	}

}
```

Denormalization

The person table is a generic table for each user registered with the system. However, keeping the performance in mind, we have denormalized the person table into buyer and seller table. These tables are still linked to the person table with member id.

Views

In database, a query with joins shows poorer performance as compared to a direct query. Thus, as per our system's data model, auction module is designed as a view to avoid unnecessary number of joins between seller, product and auction table. Similarly, we have implemented product view. This not only saves time and increases throughput but is also less complicated.

Connection Pooling

We have implemented efficient way of managing database connection objects used for multiple connections required at different instance of time. This principle is helping our system in limiting the number of connection objects and reusing the pool of objects. We know that connecting to database is an expensive and slow process, thereby reducing the cost of our system and increasing the speed. Also, we know that it is easier if the database connection is taken care by a single module to maintain it at later stage and debug. Thus, this increases the maintainability and diagnostic efficiency of our system.

```
var ejs= require('ejs');
var mysql = require('mysql');
var pool=mysql.createPool(
{
connectionLimit:100,
host     : 'localhost',
   user     : 'root',
   password : 'Pmapsn@1900',
   database : 'UBER',
   port  : 3308
});
function fetchData(callback,sqlQuery){
console.log("\nSQL Query::"+sqlQuery);
var connection=pool.getConnection(function(err,connection){
if(err)
{
console.log("error");
}
else{
connection.query(sqlQuery, function(err, rows, fields) {
connection.release();
if(err){
console.log("ERROR: " + err.message);
}
else
{ // return err or result
console.log("DB Results:"+rows);
callback(err, rows);
}
});
```

```
console.log("\nConnection closed..");

}
});
}

exports.fetchData=fetchData;



function insertData(callback,sqlQuery){
console.log("\nSQL Query::"+sqlQuery);
var connection=pool.getConnection(function(err,connection){
if(err)
{
console.log("error");
}
else{
connection.query(sqlQuery, function(err, rows, fields) {
connection.release();
if(err){
console.log("ERROR: " + err.message);
}
else
{ // return err or result
console.log("DB Results:"+rows);
callback(err, rows);
}
});
console.log("\nConnection closed..");

}
});
}
```

Dependency Injection

Dependency injection is achieved in the system using prepared statement. Our prepared statements are nothing but parameterized queries which in turn do not rely on the client side sanitation and validation. Thus, we are avoiding building SQL code strings. So in totality we are doing the validation code on the server side and making it thin client and a fat server as per the set standards.

```
exports.userLogin - function (req,res){

var user - reo.paroWinput(mail");

vor padreq.parom("inputpwd');

vor cols - [];

if(usee !-- null){

globols.queryString -

'SELECT • FROM ?? WHERE ?? ?';

console.log('Query is :• globals.queryString);

// Connection Poling Coll

mysolpool.execQuery(globols.queryString.Uperson'.'emoiloddr'.userLfunction(err,results){

if(err){throw err;

) else {

if(results.length o0){ ear

resultStr - JSON.porse(J500.stringify(results));

ifCresultStr[0].posswordpwd){ globols.Auth - true;

globols.usrFRomeresultStr[0].firstnome;

globols.usrlMome - resultStr[0].lostnome;

globols.oddress -resultStr[0].oddress;

globals.city -resultStr[0].city;

globols.stote -resultStr[0].state;
```

globols.country -resultStr[0].country;

globols.xipeode -resultStr[0],ipcode;

globols.emoil - resultStr[0].emailaddr;

globols.lostlogin - resultStr[0].lostlogin;

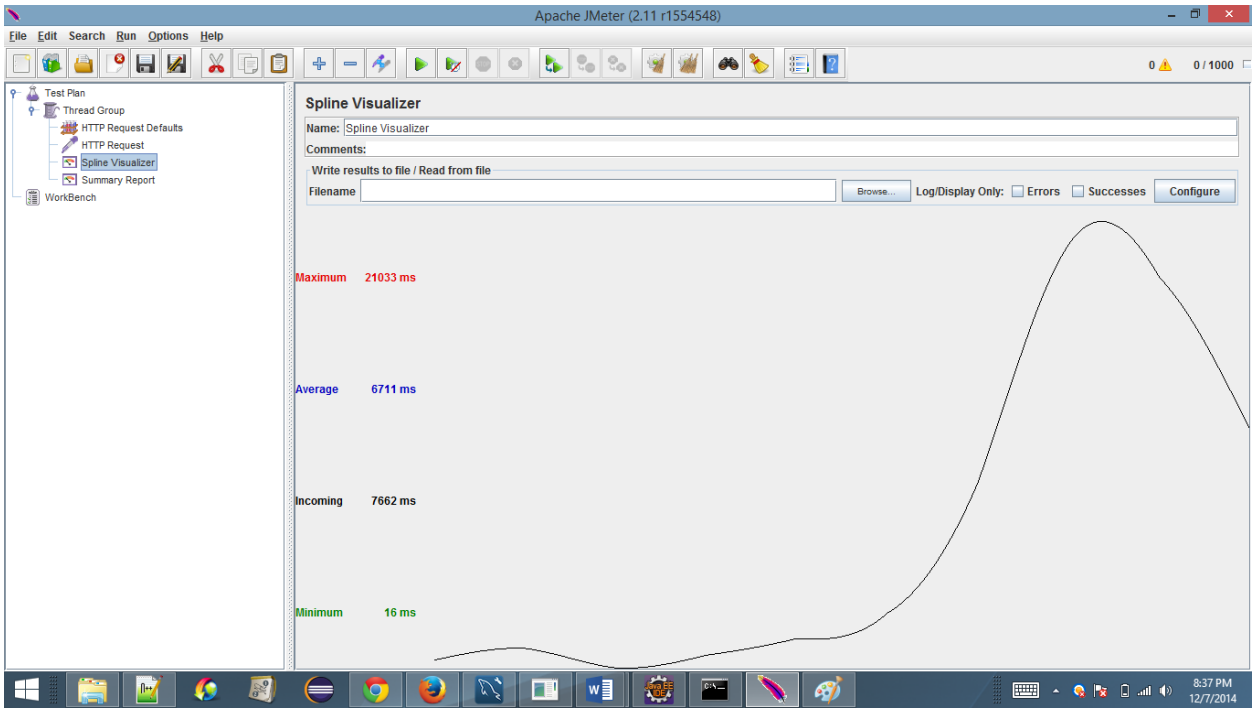
globols.user_ld - resultStr[0].memberid;

nor odminresultStr[0].admin;

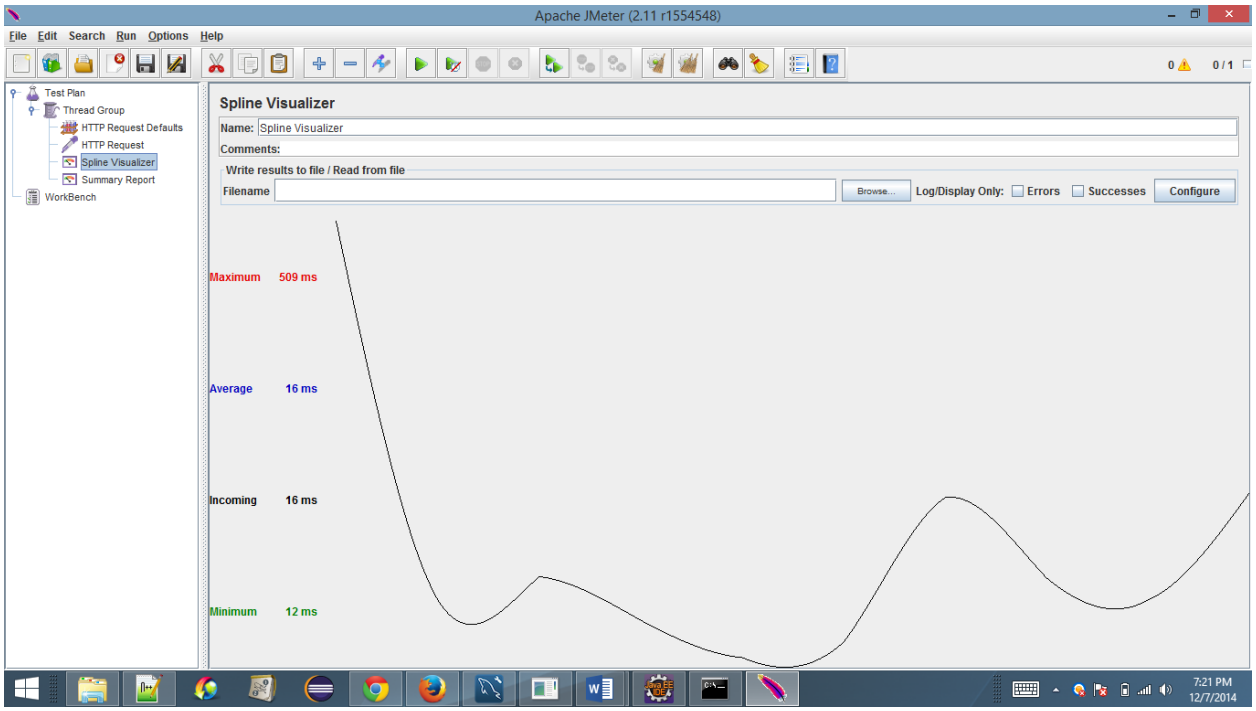
Testing

JMeter: Performance Testing

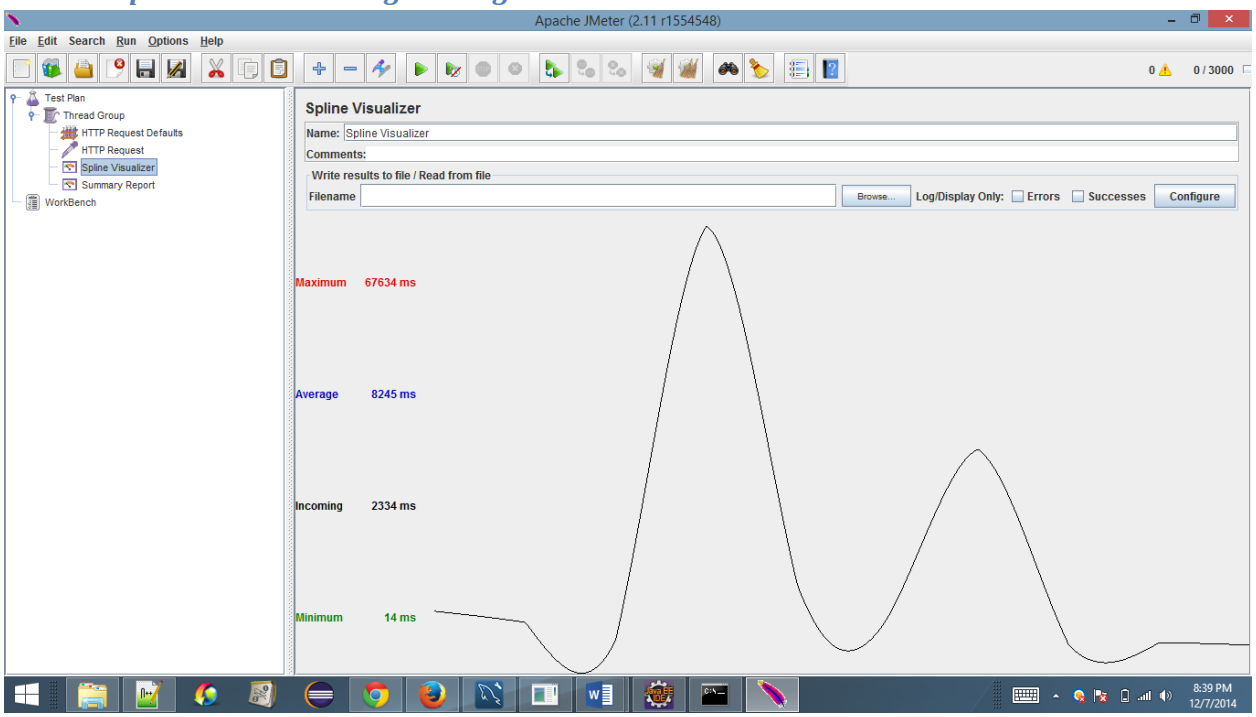
3000 Samples without caching: Average is 6711ms



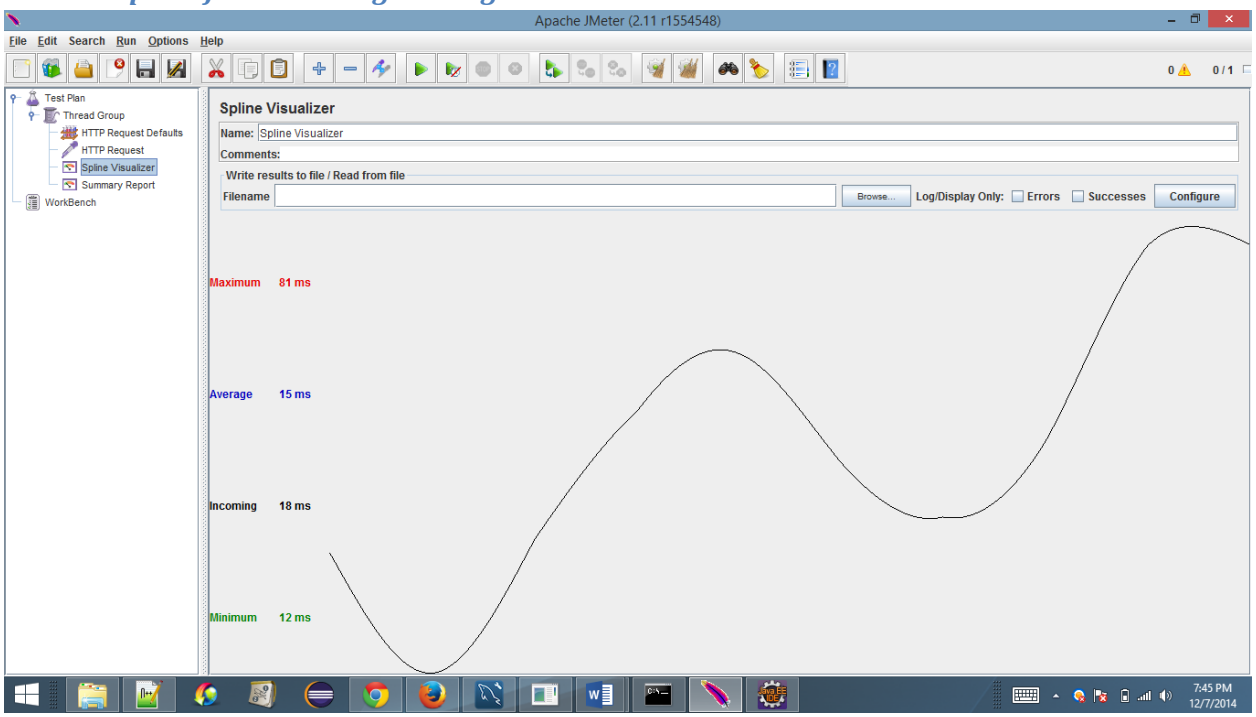
3000 Samples with caching: Average is 16ms



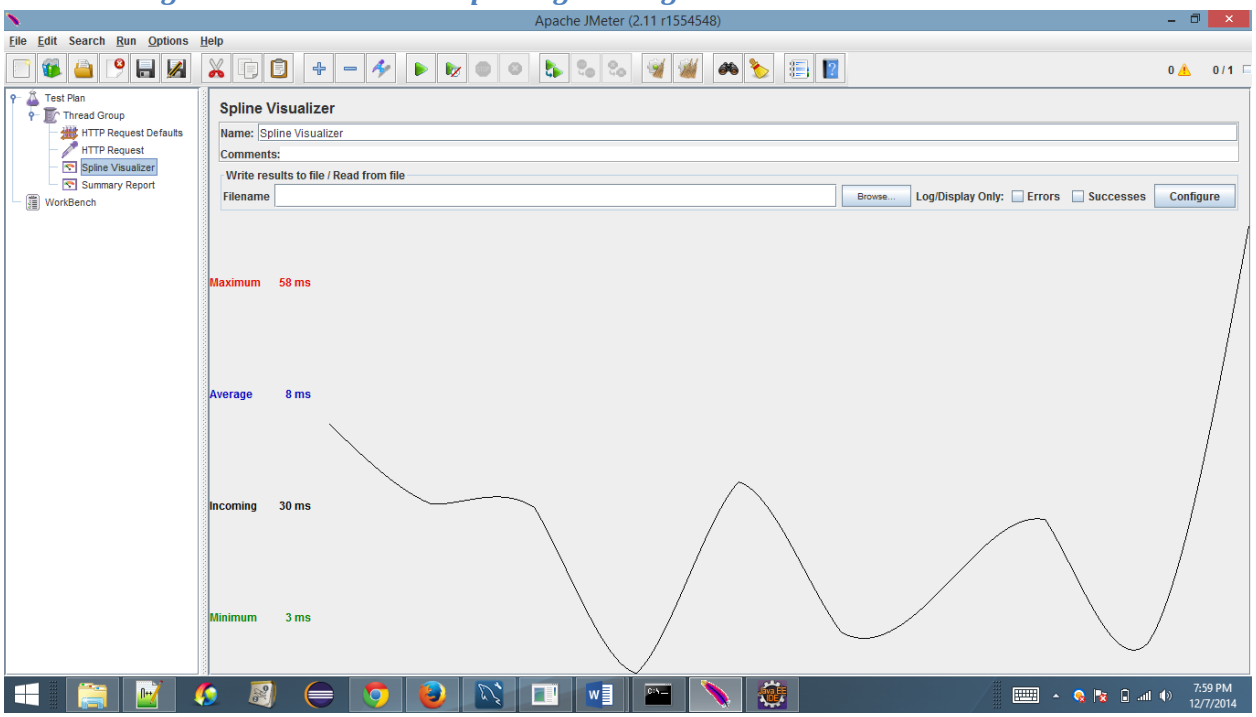
5000 Samples without caching: Average is 8245ms



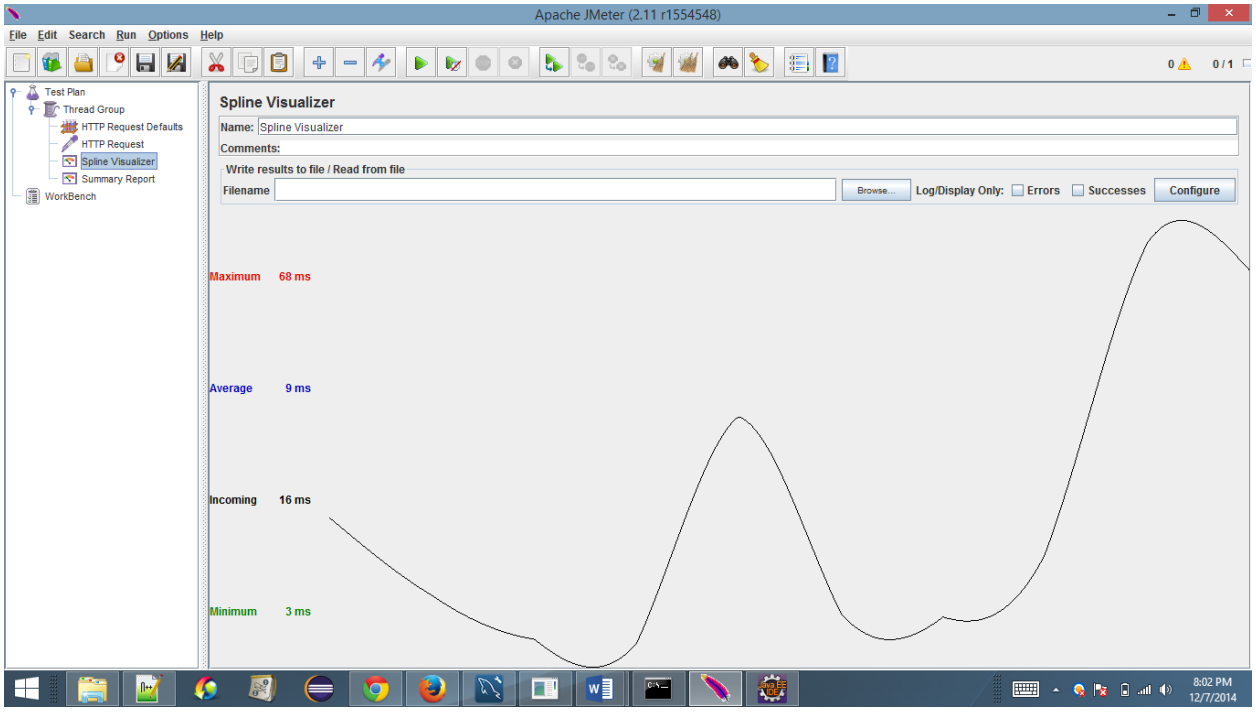
5000 Samples of with caching: Average is 15ms



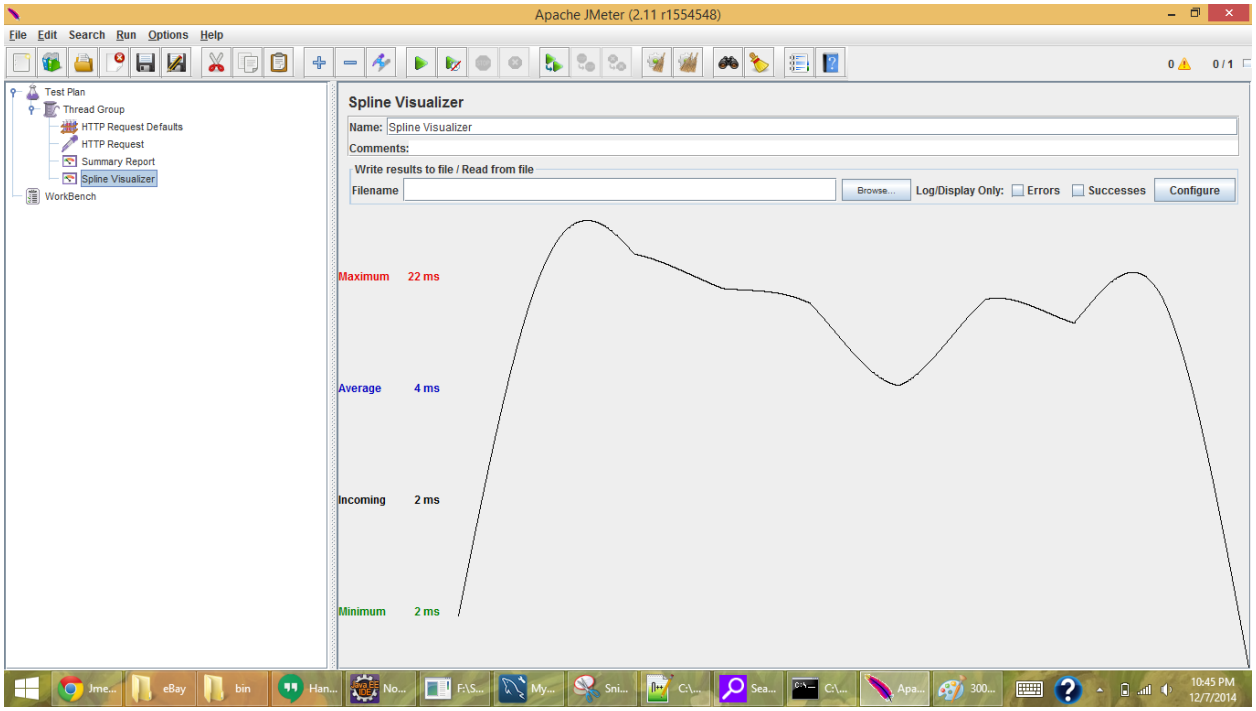
3000 user login without connection pooling: Average is 8ms



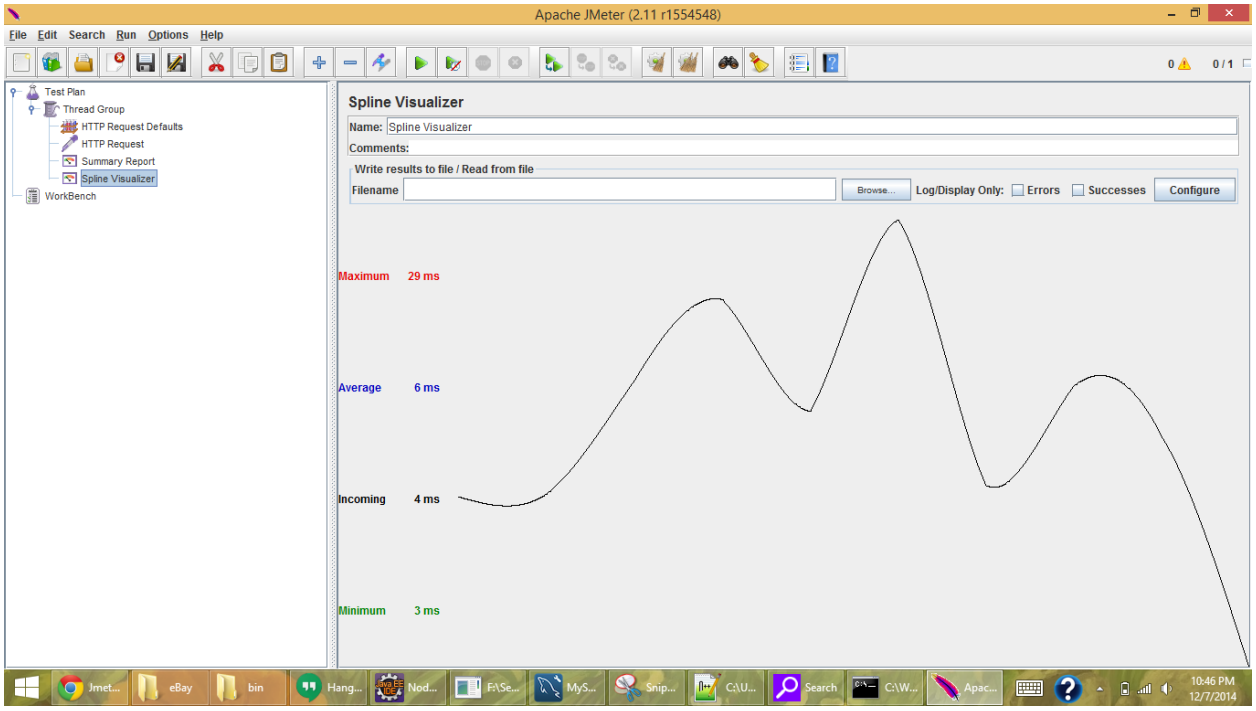
5000 user login without connection pooling: Average is 9ms



3000 user login with connection pooling: Average is 4ms



5000 user login with connection pooling: Average is 6ms



Test cases

Customer Module:

Starting Point	Steps	Expected Behaviour
Signup page	Click on sign up	Redirect to signup page.
	Enter the details as per requirements.	
	Click on Submit.	Redirect to login page.

Starting Point	Steps	Expected Behaviour
Login page	Click on login	Redirect to login page.
	Enter the details as per requirements.	
	Click on Submit.	Redirect to customer home page.

Starting Point	Steps	Expected Behaviour
Homepage	Click on Book a ride after setting location.	
	Enter the details as per requirements.	
	Click on Submit.	Ride should be booked and driver is assigned.

Starting Point	Steps	Expected Behaviour
Homepage	Click on profile	Redirect to profile page and displays information..
	View details.	

Driver Module:

Starting Point	Steps	Expected Behaviour
Signup page	Click on signup	Redirect to signup page.
	Enter the details as per requirements.	
	Click on Submit.	Redirect to login page.

Starting Point	Steps	Expected Behaviour
Login page	Click on login	Redirect to login page.
	Enter the details as per requirements.	
	Click on Submit.	Redirect to driver home page.

Starting Point	Steps	Expected Behaviour
Homepage	Click on submission links.	
	Submit required details.	
	Click on Submit.	Driver is approved.

Starting Point	Steps	Expected Behaviour
Homepage	View ride request.	
	Accept ride request.	Ride should be accepted and driver is assigned to a customer.

Admin Module:

Starting Point	Steps	Expected Behaviour
Signup page	Click on signup	Redirect to signup page.

	Enter the details as per requirements.	
	Click on Submit.	Redirect to login page.

Starting Point	Steps	Expected Behaviour
Login page	Click on login	Redirect to login page.
	Enter the details as per requirements.	
	Click on Submit.	Redirect to admin home page.

Starting Point	Steps	Expected Behaviour
Homepage	Click on Add driver	Redirect to add driver page.
	Submit required details.	
	Click on Submit.	Driver is added.

Starting Point	Steps	Expected Behaviour
Homepage	Click on Add user	Redirect to add user page.
	Submit required details.	
	Click on Submit.	User is added.

Starting Point	Steps	Expected Behaviour
Homepage	Click on Delete driver	Redirect to delete driver page.
	Submit required details.	
	Click on Submit.	Driver is deleted.

Starting Point	Steps	Expected Behaviour
Homepage	Click on Delete user	Redirect to delete user page.
	Submit required details.	
	Click on Submit.	User is deleted.

Starting Point	Steps	Expected Behaviour
Homepage	Click on View driver	Redirect to view driver page.
	Submit required details.	
	Click on Submit.	Driver details are displayed.

Starting Point	Steps	Expected Behaviour
Homepage	Click on view user.	Redirect to view user page.
	Submit required details.	
	Click on Submit.	User details are displayed.

Starting Point	Steps	Expected Behaviour
Homepage	Click on view bill.	Redirect to search bill page.
	Submit required details.	
	Click on Submit.	Bill details are displayed if exists.

Code Listing

App.js

```
var express = require('express');
var routes = require('./routes');
var user = require('./routes/user');
var http = require('http');
var path = require('path');
var index = require('./routes/index');
var app = express();
var session = require('client-sessions');
var mysql = require('mysql');
var mongoSessionConnectURL = "mongodb://localhost:27017/sessions";
var expressSession = require("express-session");
var mongoStore = require("connect-mongo")(expressSession);
var mongo = require("./routes/mongo");

// all environments
app.use(session({

	cookieName: 'session',
	secret: 'uber_test_string',
	duration: 30 * 60 * 1000,
	activeDuration: 5 * 60 * 1000,  }));
app.set('port', process.env.PORT || 3000);
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.favicon());
//app.use();
app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.json());
app.use(express.urlencoded());
app.use(express.methodOverride());
app.use(expressSession({
	secret: 'cmpe273_teststring',
	resave: false,  //don't save session if unmodified
	saveUninitialized: false, // don't create session until something stored
	duration: 30 * 60 * 1000,
	activeDuration: 5 * 60 * 1000,
	store: new mongoStore({
		url: mongoSessionConnectURL
	})
}));
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));


app.get('/', routes.index);

// development only
if ('development' == app.get('env')) {
  app.use(express.errorHandler());
```

```
}

app.get('/', routes.index);
app.get('/index', index.index)
app.get('/user', index.user);
app.get('/signUp', index.signUp);
app.get('/mainLogin', index.mainLogin);
app.get('/signUpCust', index.signUpCust);
app.post('/driverSignIn', index.driverSignIn);
app.get('/redirectToDrLogin',index.redirectToDrLogin);
app.get('/redirectToDrHome',index.redirectToDrHome);
app.get('/redirectToBackground',index.redirectToBackground);
app.get('/redirectToDrProfile',index.redirectToDrProfile);
app.get('/redirectToDrEditProfile',index.redirectToDrEditProfile);
app.get('/redirectToDrNewProfile',index.redirectToDrNewProfile);
app.get('/redirectToDrRideHistory',index.redirectToDrRideHistory);
app.post('/saveDriverProfile',index.saveDriverProfile);
app.get('/driverLogin', index.driverLogin);
app.post('/driverLog', index.driverLog);
app.post('/backgroundcheck', index.backgroundcheck);
app.get('/background-check', index.redirectToBackground);
app.post('/liscUpload',index.liscUpload);
app.get('/redirectToDocumentUpload',index.redirectToDocumentUpload);
app.post('/vehiclecheck', index.vehiclecheck);
app.get('/documents', index.documentUpload);
app.get('/video', index.welcomeVideo);
app.get('/logout', index.logout);
app.post('/goToRideHistory',index.goToRideHistory);


mongo.connect(mongoSessionConnectURL, function(){
	console.log('Connected to mongo at: ' + mongoSessionConnectURL);
	http.createServer(app).listen(app.get('port'), function(){
		console.log('Express server listening on port ' + app.get('port'));
	});
});
```

Driver

Create a driver

```
function driverSignIn(req, res) {
	var fName = req.param("first_name");
	var lName = req.param("last_name");
	var emailid = req.param("email");
	var contact = req.param("contactinfo");
	var password = req.param("password");
	var address = req.param("address");
	var cityName = req.param("city_name");
	var state = req.param("state");
	var zip_code = req.param("zip_code");
	console.log(fName + lName + emailid + contact + password + address + cityName
+ state+ zip_code);
	if(fName!=undefined && fName!=undefined && lName!=undefined &&
emailid!=undefined && contact!=undefined && password!=undefined &&
zip_code!=undefined && cityName!=undefined && state!=undefined)
	{
		console.log("here");
		var msg_payload = { "fName": fName, "lName":lName, "email":emailid,
"contact":contact,"password": password,"address":address,"cityName":cityName,
"state":state,"zip_code":zip_code };

		mq_client.make_request('uber_queue',msg_payload, function(err,results){

			console.log("Hi"+JSON.stringify(results.code));
			if(err){
				throw err;
			}
			else
			{
				console.log("value of status code at client is :
"+results.code);
				if(results.code == 200){
					req.session.username=results.user;
					console.log("valid Login");
					res.send({"signup":"Success"});

				}
				else if(results.code == 401) {

					console.log("Invalid Login");
					res.send({"signup":"Fail"});
				}
			}
		});
	}
```

Update driver information

```
exports.saveDriverProfile=function (req, res) {
	var ssn = req.param("ssn");
	var emailid = req.param("email");
	var contact = req.param("contact");
	var carDetails = req.param("carDetails");
	var address = req.param("address");
	var cityName = req.param("city_name");
	var state = req.param("state");
	var zip_code = req.param("zip_code");
	var pastEmail = req.session.username[0].emailID;
	console.log(contact + carDetails + address + cityName + state+
zip_code+emailid);
	if(ssn!=undefined && contact!=undefined && carDetails!=undefined &&
zip_code!=undefined && cityName!=undefined && state!=undefined)
	{
		console.log("here");
		var msg_payload = { "ssn":ssn, "email":emailid,
"contact":contact,"carDetails": carDetails,"address":address,"cityName":cityName,
"state":state,"zip_code":zip_code , "pastEmail":pastEmail };

		mq_client.make_request('editDriver_queue',msg_payload,
function(err,results){

			console.log(JSON.stringify(results.code));
			console.log(JSON.stringify(results.user));
			if(err){
				throw err;
			}
			else
			{
				console.log("value of status code at client is :
"+results.code);
				if(results.code == 200){
					console.log("New user set to session");
					req.session.username1=results.user;
					console.log(JSON.stringify(req.session.username1));
					console.log("valid Login");
					res.send({"signup":"Success"});

				}
				else if(results.code == 401) {

					console.log("Invalid Login");
					res.send({"signup":"Fail"});
				}
			}
		});
	}
};
```

Background check on driver

```
function backgroundcheck(req,res) {

	var SSN = req.param("ssn1")+req.param("ssn2")+req.param("ssn3");
	var data = req.session.username;
	var email =data[0].emailID;
	console.log(email);
	if(req.param("ssn1")!=undefined && req.param("ssn2")!=undefined &&
req.param("ssn3")!=undefined)
	{

		var msg_payload = { "SSN": SSN , "email":email};
		console.log("ye ki  "+JSON.stringify(msg_payload));
		mq_client.make_request('SSN_queue',msg_payload, function(err,results){

			console.log("Hi"+JSON.stringify(results.code));
			if(err){
				throw err;
			}
			else
			{
				console.log("value of status code at client is :
"+results.code);
				if(results.code == 200){
					console.log("SSN added");
					res.send({"ssn":"Success"});

				}
				else if(results.code == 401) {

					console.log("Invalid SSN");
					res.send({"ssn":"Fail"});
				}
			}
		});
	}

}
```

Update vehicle information

```
function vehiclecheck(req,res) {
	var vehicleNum = req.param("vehicle");
	var data = req.session.username;
	var email =data[0].emailID;
	console.log(email);
	if(vehicleNum!=undefined){
		var msg_payload = { "vehicle": vehicleNum , "email":email};
		console.log("itheeeeeee "+JSON.stringify(msg_payload));
		mq_client.make_request('vehicle_queue',msg_payload,
function(err,results){
```

```
                        console.log("Hi"+JSON.stringify(results.code));
                        if(err){
                                throw err;
                        }
                        else
                        {
                                console.log("value of status code at client is :
"+results.code);
                                if(results.code == 200){
                                        console.log("Vehicle added");
                                        res.send({"vehicle":"Success"});

                                }
                                else if(results.code == 401) {

                                        console.log("Invalid Login");
                                        res.send({"vehicle":"Fail"});
                                }
                        }
                });
        }
};
```

Database interaction

Mysql.js

```
function getConnection(){
        var connection = mysql.createConnection({
            host     : 'localhost',
            user     : 'root',
            password : 'Pmapsn@1900',
            database : 'UBER',
            port      : 3308
        });
        return connection;
}


function fetchData(callback,sqlQuery){

        console.log("\nSQL Query::"+sqlQuery);

        var connection=getConnection();

        connection.query(sqlQuery, function(err, rows, fields) {
                if(err){
                        console.log("ERROR: " + err.message);
                }
                else
                {       // return err or result
                        console.log("DB Results:"+rows);
                        callback(err, rows);
```

```
		}
	});
	console.log("\nConnection closed..");
	connection.end();
}

exports.fetchData=fetchData;



function insertData(callback,sqlQuery){

	console.log("\nSQL Query::"+sqlQuery);

	var connection=getConnection();

	connection.query(sqlQuery, function(err, rows, fields) {
		if(err){
			console.log("ERROR: " + err.message);
		}
		else
		{	// return err or result
			console.log("Inserted:"+rows);
			callback(err, rows);
		}
	});
	console.log("\nConnection closed..");
	connection.end();
}
```

Rabbit MQ messaging service interaction

server.js

```
//super simple rpc server example
var amqp = require('amqp')
, util = require('util');

var adminDriver = require('./services/adminDriver')
var signup = require('./services/signup_serverside')
var uber = require('./services/uberMysql')
var bookRide = require('./services/BookARide')

var cnn = amqp.createConnection({host:'127.0.0.1'});

cnn.on('ready', function(){
console.log("listening on uber_queue");
cnn.queue('uber_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_request(message, function(err,res){
```

```
console.log("In server.js*********");
		console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on uber_queue");
cnn.queue('notify_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handlenotify_request(message, function(err,res){
console.log("In server.js*********");
		console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on bookride_queue");
cnn.queue('bookride_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
bookRide.handle_bookride(message, function(err,res){
console.log("In server.js*********");
		console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on vehicle_queue");
cnn.queue('vehicle_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_vehicleValidation(message, function(err,res){
console.log("In vehicle server.js*********");
		console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
```

```
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on SSN_queue");
cnn.queue('SSN_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_SSNValidation(message, function(err,res){
console.log("In SSN server.js*********");
        console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on editDriver_queue");
cnn.queue('DrRideHistory_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_DrRideHistory_queue(message, function(err,res){
console.log("In SSN server.js*********");
        console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on editDriver_queue");
cnn.queue('editDriver_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_editDriver(message, function(err,res){
console.log("In SSN server.js*********");
        console.log(res.code);
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
```

```
});
console.log("listening on driverLog_queue");
cnn.queue('driverLog_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
uber.handle_driverLogin(message, function(err,res){

//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on customerLog_queue");
cnn.queue('customerLog_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
signup.handle_custLogin(message, function(err,res){

//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on adminDriver_queue");
cnn.queue('adminDriver_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
adminDriver.handle_addDriverRequest(message, function(err,res){

//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
console.log("listening on signup_queue");
cnn.queue('signup_queue', function(q){
q.subscribe(function(message, headers, deliveryInfo, m){
util.log(util.format( deliveryInfo.routingKey, message));
util.log("Message: "+JSON.stringify(message));
util.log("DeliveryInfo: "+JSON.stringify(deliveryInfo));
signup.handle_request(message, function(err,res){
```

```
//return index sent
cnn.publish(m.replyTo, res, {
contentType:'application/json',
contentEncoding:'utf-8',
correlationId:m.correlationId
});
});
});
});
});
```

services.js

uberMysql.js

```
var mysql = require('./mysql');
var ejs = require('ejs');



//handlenotify_request

function handlenotify_request(msg, callback){
var res={};

loginUser = "select * from rides where driverID ='" + msg.email + "' and rideStatus='started'";
console.log(loginUser);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
} else {
if (user.length > 0) {
console.log("Account already exists");
res.code="200";
res.user=user;
res.value = "Success validation";
}
else {
console.log("here it is");
}
}
},loginUser);
}

}

function handle_request(msg, callback){
var res={};

loginUser = "select * from drivers where emailID ='" + msg.email + "'";
console.log(loginUser);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
```

```
throw err;
} else {
if (user.length > 0) {
console.log("Account already exists");
res.code="200";
res.user=user;
res.value = "Success validation";
}
else {
console.log("here it is");
var inserted = "insert into drivers (firstName, lastName, emailID ,phoneNumber,password,city, address, state,
zipCode) values('"+msg.fName + "'" + ",'" + msg.lName + "'" + ",'" + msg.email + "'" +",'"+msg.contact + "'"+",'" +
msg.password +  "'"+",'" + msg.cityName + "'" + ",'" + msg.address + "'" + ",'" + msg.state + "'"
+",'"+msg.zip_code + "')";

mysql.insertData(function(err, user1) {
if (err) {
throw err;
} else {
if (user1 != undefined) {
console.log(user1);
console.log("valid Login");
res.user1 = user1;
res.code="200";
console.log("**********This is the res.code**********");
console.log(res.code);
res.value = "Success SignUp";
} else {

res.code="401";
}
}
callback(null, res);
}, inserted);
}
}
},loginUser);
}

}

function handle_driverLogin(msg, callback){
var res={};
console.log("heloooo");
loginDriver = "select * from drivers where emailID ='" + msg.email + "' and password = '"+msg.password+"'";
console.log(loginDriver);

if (msg.email != undefined) {

mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("there");
} else {
if (user.length > 0) {
console.log("Account exists");
var rideStatus="select * from rides where driverID='"+msg.email+"' and rideStatus  = 'started'";
mysql.fetchData(function(err, user1) {
if (err) {
```

```
throw err;
console.log("there");
} else {
if (user1.length > 0) {
console.log("Account exists");
res.code="200";
res.user=user1;
res.value = "Success validation";
}
else {
res.code="300";
res.user=user;
res.value = "Success validation";
}
} callback(null, res);
},rideStatus);
}
else {
res.code="401";
res.value="Failed Login";
}
}
},loginDriver);
}
}

function handle_vehicleValidation(msg, callback){
var res={};
console.log("heloooo");
vehicleNumber = "update drivers set carDetails = '"+ msg.vehicle +"' where emailID ='" + msg.email+"'";
console.log(vehicleNumber);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("vehicle is here");
} else {
if(user!=undefined){
console.log("Data updated successfully");
res.code="200";
res.user=user;
res.value = "Successfully updated";
}
else {
res.code="401";
res.value="Failed Login";
}
}

callback(null, res);
},vehicleNumber);
}
}

function handle_SSNValidation(msg, callback){
var res={};
console.log("heloooo");
SSN = "update drivers set idDriver = '"+ msg.SSN +"' where emailID ='" + msg.email+"'";
console.log(SSN);
```

```
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("vehicle is here");
} else {
if(user!=undefined){
console.log("Data updated successfully");
res.code="200";
res.user=user;
res.value = "Successfully updated";
}
else {
res.code="401";
res.value="Failed Login";
}
}

callback(null, res);
},SSN);
}
}



function handle_editDriver(msg, callback){
var res={};
console.log("heloooo");
SSN = "update drivers set idDriver = '"+ msg.ssn  +"', phoneNumber='"+ msg.contact +"',
address='"+msg.address +"', city='"+msg.cityName+"',state = '"+msg.state +"',zipCode = '"+msg.zip_code
+"',carDetails = '"+msg.carDetails +"' ,emailID = '"+msg.email +"' where emailID ='" + msg.pastEmail+"'";
console.log(SSN);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("vehicle is here");
} else {
if(user!=undefined){
console.log("Data updated successfully");
//res.code="200";
res.value = "Successfully updated";
//res.user=user;
///
loginDriver = "select * from drivers where emailID ='" + msg.email +"'";
console.log(loginDriver);
mysql.fetchData(function(err, user1) {
if (err) {
throw err;
console.log("there");
} else {
if (user1.length > 0) {
console.log("Account exists");
res.code="200";
res.user=user1;
//res.value = "Success validation";
console.log(user1);
}
else {
```

```
res.code="401";
res.value="Failed Login";
}
}

callback(null, res);
},loginDriver);
}
else {
res.code="401";
res.value="Failed Login";
}
}
},SSN);
}
}

//ride history for driver
function handle_DrRideHistory_queue(msg, callback){
var res={};
console.log("heloooo");
rides = "select * from rides where driverID ='" + msg.pastEmail+"' and rideStatus='finished'";
console.log(rides);
if (msg.pastEmail != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("rides is here");
} else {
if(user.length>0){
//console.log("Data updated successfully");
res.code="200";
res.value = "Successfully fetched rides";
res.user=user;
}
else {
res.code="401";
res.value="Failed Login";
}
}
callback(null, res);
},rides);
}
}
exports.handle_SSNValidation=handle_SSNValidation;
exports.handle_request = handle_request;
exports.handlenotify_request = handlenotify_request;
exports.handle_driverLogin = handle_driverLogin;
exports.handle_vehicleValidation = handle_vehicleValidation;
exports.handle_editDriver=handle_editDriver;
exports.handle_DrRideHistory_queue=handle_DrRideHistory_queue;
```

signup.js

```
var util = require("util");
var mysql = require('./mysql');
var ejs = require('ejs');
```

```
function handle_request(msg, callback){
var res={};
console.log("in signup handle req"+ msg);
var loginUser = "select * from customers where email ='" + msg.email + "'";
console.log(loginUser);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
if (err) {
throw err;
console.log("there");
} else {
if (user.length > 0) {
console.log("Account already exists");
res.code="200";
res.value = "Success validation";
}
else {
console.log("here it is");
console.log(msg.first_name);
var inserted ="insert into
customers(firstName,lastName,zipCode,phoneNumber,email,creditCardDetails,password) VALUES('" +
msg.first_name + "'" + ",'" + msg.last_name + "'" + ",'" + msg.billing_zip +"'" + ",'" + msg.contact + "'" + ",'" +
msg.email +"'" + ",'" + msg.card_number + "'" + ",'" + msg.password +"')";
mysql.insertData(function(err, user1) {
if (err) {
throw err;
} else {
if (user1 != undefined) {
console.log(user1);
console.log("valid signup");
res.code="200";
res.user=user1;
console.log(res.code);
console.log(res.user);
res.value = "Success SignUp";

} else {
res.code="401";
}
}callback(null,res);

}, inserted);
}
}

},loginUser);
}

}


exports.handle_custLogin= function (msg, callback){
var res={};
console.log("heloooo");
loginCust = "select * from customers where email ='" + msg.email + "' and password = '"+msg.password+"'";
console.log(loginCust);
if (msg.email != undefined) {
mysql.fetchData(function(err, user) {
```

```
if (err) {
throw err;
console.log("there");
} else {
if (user.length > 0) {
console.log("Account exists");
res.code="200";
res.user=user;
res.value = "Success validation";
}
else {
res.code="401";
res.value="Failed Login";
}
}

callback(null, res);
},loginCust);
}
}



exports.handle_request = handle_request;
```

bookARide.js

```
var mysql = require('./mysql');
var ejs = require('ejs');


function handle_bookride(msg, callback){
var res={};
console.log(msg.riderID+msg.driverID);
var inserted = "insert into billinginfo
(date,pickupTime,distanceCovered,sourceAddress,destinationAddress,driverID,customerID,rideCost)
values(CURDATE(),CURRENT_TIMESTAMP,'"+msg.dist+"','"+msg.src+"','"+msg.dest+"','"+msg.driverID+"','"+
msg.riderID+"','"+msg.rideCost+")";

mysql.insertData(function(err, user1) {
if (err) {
throw err;
} else {
if (user1 != undefined) {
console.log(user1);
console.log("valid Login");
res.user1 = user1;
res.code="200";
console.log("**********This is the res.code**********");
console.log(res.code);
res.value = "Success SignUp";
///
var inserted1 = "insert into rides (pickUpLocation,dropOffLocation,DateTime,driverID,customerID,rideStatus)
values('"+msg.src+"'," + msg.dist+"',CURRENT_TIMESTAMP,'"+msg.driverID+"','"+msg.riderID+"','started')";
console.log(inserted1);
mysql.insertData(function(err, user2) {
if (err) {
throw err;
```

```
} else {
if (user2 != undefined) {
console.log(user2);
console.log("valid Login");
//res.user1 = user1;
//res.code="200";
console.log("****************This is the res.code******************");
//console.log(res.code);
// res.value = "Success SignUp";
} else {

//res.code="401";
}
}
}, inserted1);

///
} else {

res.code="401";
}
}
callback(null, res);
}, inserted);
}



exports.handle_bookride=handle_bookride;
```

Implementation of session

```
var mongoSessionConnectURL = "mongodb://localhost:27017/sessions";
var mongoStore = require("connect-mongo")(expressSession);
var mongo = require("./routes/mongo");

app.use(expressSession({
        secret: 'cmpe273_teststring',
        resave: false,  //don't save session if unmodified
        saveUninitialized: false, // don't create session until something stored
        duration: 30 * 60 * 1000,
        activeDuration: 5 * 60 * 1000,
        store: new mongoStore({
                url: mongoSessionConnectURL
        })
}));

mongo.connect(mongoSessionConnectURL, function(){
        console.log('Connected to mongo at: ' + mongoSessionConnectURL);
        http.createServer(app).listen(app.get('port'), function(){
```

```
		console.log('Express server listening on port ' + app.get('port'));
	});
});
```

Admin functionalities

Add an user

```
exports.adduser = function(req,res)
{
	var email= req.param("email");
	var password = req.param("password");
	var first_name = req.param("first_name");
	var last_name=req.param("last_name");
	var mobile = req.param("mobile");
	var lang = req.param("lang") ;
	var card_number = req.param("card_number") ;
	var card_code = req.param("card_code") ;
	var card_expiration_month = req.param("card_expiration_month") ;
	var card_expiration_year = req.param("card_expiration_year") ;
	var billing_zip = req.param("billing_zip") ;
	console.log("signup client");
	var msg_payload = { "email": email, "password": password,"first_name"
:first_name ,"last_name" : last_name, "mobile" : mobile, "lang":lang, "card_number" :
card_number, "card_code" : card_code, "card_expiration_month" :
card_expiration_month, "card_expiration_year" : card_expiration_year, "billing_zip" :
billing_zip};
    mq_client.make_request('signup_queue', msg_payload, function(err,result){
		console.log("result in client %j",result);
      if(err){
			throw err;
		}
		else
		{
			if(result.code == 200){
				console.log("valid signup");
				res.send({"adduser":"Success"});
			}
			else {

				console.log("Invalid signup");
				res.send({"adduser":"Fail"});
			}
		}
    });
};
```

Add a driver

```
exports.adddriver = function(req,res)
{
	var email= req.param("email");
	var password = req.param("password");
	var first_name = req.param("first_name");
	var last_name=req.param("last_name");
	var mobile = req.param("mobile");
	var city = req.param("city") ;

	console.log("signup client");
	var msg_payload = { "email": email, "password": password,"first_name"
:first_name ,"last_name" : last_name, "mobile" : mobile, "city":city};
    mq_client.make_request('driver_queue', msg_payload, function(err,result){
		console.log("result in client %j",result);
	  if(err){
			throw err;
		}
		else
		{
			if(result.code == 200){
				console.log("valid signup");
				res.send({"adddriver":"Success"});
			}
			else {

				console.log("Invalid signup");
				res.send({"adddriver":"Fail"});
			}
		}
    });
};
```

Delete driver

```
exports.deldriver = function(req,res)
{
	var email= req.param("email");


	console.log("del driver");
	var msg_payload = { "email": email};
    mq_client.make_request('deldriver_queue', msg_payload, function(err,result){
		console.log("result in client %j",result);
	  if(err){
			throw err;
		}
		else
		{
			if(result.code == 200){
				console.log("driver deleted");
				result.deldriver="Success";
				console.log(result.deldriver);
				res.send(result);
			}
			else {
```

```
                        console.log("Invalid email");
                        result.deldriver="Fail";
                        res.send(result);
                }
        }
    });
};
```

Delete a user

```
exports.deluser = function(req,res)
{
       var email= req.param("email");


       console.log("del user");
       var msg_payload = { "email": email};
    mq_client.make_request('deluser_queue', msg_payload, function(err,result){
            console.log("result in client %j",result);
       if(err){
                throw err;
            }
            else
            {
                if(result.code == 200){
                        console.log("user deleted");
                        result.deluser="Success";
                        console.log(result.deluser);
                        res.send(result);
                }
                else {

                        console.log("Invalid email");
                        result.deluser="Fail";
                        res.send(result);
                }
            }
    });
};
```

Show statistics

```
exports.showstatistics =  function (req,res) {

       ejs.renderFile('./views/showstatistics.ejs',function(err, result) {
          // render on success
          if (!err) {
                   res.end(result);
          }
          // render or error
```

```
        else {
                res.end('An error occurred');
                console.log(err);
        }
    });
}
```

Billing

Billing algorithm

```
exports.locationdest=function(req,res) {

var dest = req.param("dest");
var src = req.param("src");
var dist=req.param("dist");
console.log("Destination is :");
console.log(dest)
console.log("Source is:");
console.log(src);
console.log("Distance is:");
console.log(dist);
/// billing algo
var fixedrate=2.35;
var distrecieved=dist/1000*0.62;
var total =distrecieved*2.35;
console.log(total);
// if(CURDATE)
///
console.log("driver id is....."+driverID);
console.log("IN LOCATION DEST*********"+req.session.RidedriverID);
console.log("customer session"+JSON.stringify(req.session.username));
var msg_payload = { "dest": dest, "src": src, "dist" :dist,"driverID" : driverID, "riderID"
:req.session.username[0].email,"rideCost":total };
  mq_client.make_request('bookride_queue',msg_payload, function(err,results){
console.log(results);
if(err){
throw err;
}
else
{

if(results.code == 200){
console.log("valid Login");
res.send({"login":"Success"});
}
else {
console.log("Invalid Login");
res.send({"login":"Fail"});
}
}
});

}
```

Search for a bill

```
exports.searchbill = function(req,res)
{
	var idbillingInfo= req.param("idbillingInfo");


	console.log("search bill");
	var msg_payload = { "idbillingInfo": idbillingInfo};
    mq_client.make_request('bill_queue', msg_payload, function(err,result){
		console.log("result in client %j",result);
	if(err){
			throw err;
		}
		else
		{
			if(result.code == 200){
				console.log("bill found");
				result.searchbill="Success";
				console.log(result.searchbill);
				res.send(result);
			}
			else {

				console.log("Invalid billid");
				res.send({"viewuser":"Fail"});
			}
		}
    });
};
```

Customer features

Create a new customer

```
function handle_request(msg, callback){
	var res={};
	console.log("in signup handle req"+ msg);
	var loginUser = "select * from customer where email ='" + msg.email + "'";
	console.log(loginUser);
	if (msg.email != undefined) {
		mysql.fetchData(function(err, user) {
			if (err) {
				throw err;
				console.log("there");
			} else {
				if (user.length > 0) {
					console.log("Account already exists");
					res.code="201";
					res.value = "Success validation";
					callback(null,res);
```

```
                        }
                        else {
                              console.log("here it is");
                              console.log(msg.first_name);
                              var inserted = "insert into customer (first_name,
last_name, email, cpassword, mobile, lang,
card_number,card_code,card_expiration_month, card_expiration_year, billing_zip ) " +
                                    "values('"+msg.first_name + "'" + ",'"
+ msg.last_name + "'" + ",'" + msg.email +"'" + ",'" + msg.password + "'" + ",'" +
msg.mobile +"'" + ",'" + msg.lang + "'" + ",'" + msg.card_number +"'" + ",'" +
msg.card_code + "'" + ",'" + msg.card_expiration_month +"'" + ",'" +
msg.card_expiration_year + "'" + ",'" + msg.billing_zip + "')";

                              mysql.insertData(function(err, user1) {
                                    if (err) {
                                          throw err;
                                    } else {
                                          if (user1 != undefined) {
                                                console.log(user1);
                                                console.log("valid signup");
                                                res.code="200";
                                                console.log(res.code);
                                                res.value = "Success SignUp";
                                                callback(null,res);
                                          } else {
                                                res.code="401";
                                                callback(null,res);
                                          }
                                    }

                              }, inserted);
                        }
                  }

            },loginUser);
      }

}
```

Search for a driver within 10 miles from customer location

```
//For current location
  if (navigator.geolocation) {
  flag=1;
    navigator.geolocation.getCurrentPosition(function(position) {
      var pos = {
        lat: position.coords.latitude,
        lng: position.coords.longitude
      };

       map.setCenter(pos);
       //alert("Latitude for pos is"+pos.lat);
       //alert("Longitude for pos is"+pos.lng);
```

```
       /*start = pos;

        for(var i = 0;i<locations.length;i++){

             var loc={
        lat: locations[i][1],
        lng: locations[i][2]
       };
       //alert("location is ----------->"+loc.lat+ "and "+loc.lng);
       calculateDistance(start,loc);
       //alert("after calling calculate distance"+driverCount);
    }
     */



    var marker = new google.maps.Marker({
    position: pos,
    map: map,
    title: 'You are here !'
  });

}, function() {


      handleLocationError(true, infoWindow, map.getCenter());

    });
  } else {
    // Browser doesn't support Geolocation
    handleLocationError(false, infoWindow, map.getCenter());
  }


//


  // Create the search box and link it to the UI element.
  var input = document.getElementById('pac-input');
  var searchBox = new google.maps.places.SearchBox(input);
  map.controls[google.maps.ControlPosition.TOP_LEFT].push(input);

  // Bias the SearchBox results towards current map's viewport.
  map.addListener('bounds_changed', function() {
    searchBox.setBounds(map.getBounds());
  });

  //var markers = [];
  // [START region_getplaces]
  // Listen for the event fired when the user selects a prediction and retrieve
  // more details for that place.
  searchBox.addListener('places_changed', function() {
    var places = searchBox.getPlaces();
```

```
    if (places.length == 0) {
      return;
    }


    // Clear out the old markers.
    markers.forEach(function(marker) {
      marker.setMap(null);
    });
    markers = [];

    // For each place, get the icon, name and location.
    var bounds = new google.maps.LatLngBounds();
    places.forEach(function(place) {
      var icon = {
        url: place.icon,
        size: new google.maps.Size(71, 71),
        origin: new google.maps.Point(0, 0),
        anchor: new google.maps.Point(17, 34),
        scaledSize: new google.maps.Size(25, 25)
      };


   end = place.geometry.location;
   //alert("Destination is "+end);

      // Create a marker for each place.
           markers.push(new google.maps.Marker({
        map: map,
        icon: icon,
        title: place.name,
        position: place.geometry.location
      }));

      if (place.geometry.viewport) {
        // Only geocodes have viewport.
        bounds.union(place.geometry.viewport);
      } else {
        bounds.extend(place.geometry.location);
      }
    });
    map.fitBounds(bounds);
  });
  // [END region_getplaces]



}
```

Rides Module

Book a ride

```
//Start location of the distance matrix
var start;

//End location of the distance matrix
var end;

 var functioncount=0;

var info = new Array();

//Locations of drivers
var locations = [
   ['<h4>SAP Centre</h4>', 37.333031,-121.900797],
   ['<h4>Sunnyvale Hindu Temple</h4>', 37.405052,-122.013994],
   ['<h4>Colonnade </h4>', 37.333261,-121.884503],
 ];


function notifyDriver(info){

       //alert("The driver at shortest distance from the current user is at distance-
->"+info[0].distance+" at location "+info[0].location);
       alert("Notify driver at location "+info[0].location+" to pick up rider from
location "+info[0].rider);


}


//Function which gets the response from Distance Matrix
function callback(response,status){

//if the returned status from distance matrix is  OK
if(status==google.maps.DistanceMatrixStatus.OK){

  //alert("Driver found ! At distance -----
>"+response.rows[0].elements[0].distance.value+" from the current user at
location!"+response.destinationAddresses);
  //alert("response is "+JSON.stringify(response));

info.push({rider:response.originAddresses,location:response.destinationAddresses,dist
ance:response.rows[0].elements[0].distance.value});
  functioncount++;
  //Add distance to the drivers array

  //drivers[driverCount]=response.rows[0].elements[0].distance.value;
  //driverCount++;

  //*drivers.sort(function(a, b){return a-b});
  info.sort(function(a,b){return a.distance - b.distance;});
```

```
  //getDriver(drivers,response);

}
else
alert("Distance not found !");

//alert("function count is "+functioncount);
if(functioncount>=3)
       notifyDriver(info);
else
       alert("function count not 3");

}

//Function which ccalculateDistancealculates distance between two points
function calculateDistance(start,end){

//alert("In calculate distance function !");
var service = new google.maps.DistanceMatrixService();
service.getDistanceMatrix(
{
  origins : [start],
  destinations : [end],
  travelMode: google.maps.TravelMode.DRIVING,
            unitSystem: google.maps.UnitSystem.IMPERIAL,
            avoidHighways: false,
            avoidTolls: false

},callback);


}


//Function which handles current position location error
function handleLocationError(browserHasGeolocation, infoWindow, pos) {
  infoWindow.setPosition(pos);
  infoWindow.setContent(browserHasGeolocation ?
                        'Error: The Geolocation service failed.' :
                        'Error: Your browser doesn\'t support geolocation.');
}

//The main function which gets loaded
function initAutocomplete() {

 var flag=0;

 //Map variable
 var map= new google.maps.Map(document.getElementById('map'), {
    //center: {lat: -34.397, lng: 150.644},
    zoom: 15
  });
```

```
 // Setup the different icons and shadows
 var iconURLPrefix = 'http://maps.google.com/mapfiles/ms/icons/';


 //icons for Uber cars
 var icons = [
            iconURLPrefix + 'blue-dot.png'
            ]


 var iconsLength = icons.length;

 var infowindow = new google.maps.InfoWindow({
 maxWidth: 160
 });

var markers = new Array();

var iconCounter = 0;

// Add the markers of Uber cars to the map
for (var i = 0; i < locations.length; i++) {
var marker = new google.maps.Marker({
position: new google.maps.LatLng(locations[i][1], locations[i][2]),
map: map,
 icon: icons[iconCounter]
      });

 markers.push(marker);

 google.maps.event.addListener(marker, 'click', (function(marker, i) {
  return function() {
  infowindow.setContent(locations[i][0]);
  infowindow.open(map, marker);
 }
      })(marker, i));

  iconCounter++;
      // We only have a limited number of possible icon colors, so we may have to
restart the counter
      if(iconCounter >= iconsLength) {
       iconCounter = 0;
      }
  }
```

Search for a bill/ ride

```
exports.handle_request = function(msg,callback){
       var res={};
       var trip=[];
```

```
	console.log("in mytrips handle req"+ msg.userid);
	var loginUser = "select * from rides where CustId ='" + msg.userid + "'";
	console.log(loginUser);

	if (true) {
		mysql.fetchData(function(err, user) {
			if (err) {
				throw err;

			} else {
				if (user.length > 0) {
					console.log("Trips exists");
					res.code="200";
					res.value = "Success validation";
					res.car = user[0].car;
					res.fare = user[0].fare;
					res.driver = user[0].driver;
					res.city = user[0].city;
					res.card_number = user[0].card_number;
					res.pickup = user[0].pickup;

					console.log(res.car);
					console.log(res.fare);
					console.log(res.driver);
					console.log(res.city);
					console.log(res.card_number);
					console.log(res.pickup);
					callback(null,res);
				}
				else {
					res.code="401";
					res.value="no trips";
					callback(null,res);
				}
			}

		},loginUser);
	}

}

exports.searchbill =	function (req,res) {

	ejs.renderFile('./views/searchbill.ejs',function(err, result) {
	   // render on success
	   if (!err) {
			res.end(result);
	   }
	   // render or error
	   else {
			res.end('An error occurred');
			console.log(err);
	   }
   });
```

```
}
```