San José State
UNIVERSITY

CMPE 273 : ENTERPRISE DISTRIBUTED SYSTEMS

Project Report

Amazon Fresh Replica

Submitted To

Prof. Simon Shim

Date of Submission

1st May, 2016

TEAM MEMBERS (GROUP 5)
Aashay Shah (010725078)
Vansh Shah (010823761)
Neel Patel (010823878)
Vivek Mehta (010807693)
Darshil Saraiya (010825321)

INTRODUCTION

❖ Aim:

➢ This is a web application which simulates the working of Amazon Fresh market. In this project the application has been developed using MEAN stack. We used Angular.js, Node.js, Express framework for routing application and server handling. We also used Bootstrap and JQuery components to get a better and similar look like amazon fresh. RabbitMQ handles the AMQP and works as a middleware between the client and server. The server is implemented such a way that it handles both MySQL and MongoDB connections and it returns valid records.

❖ Goals and Purpose of System:

➢ The main goal of this system is to see how the introduction of queues give higher throughput. To understand and learn how to make our system scalable with RabbitMQ. To know about various encryption techniques, about how they work and to decide which algorithm is better to encrypt password.

➢ The application also uses REDIS caching which enhanced the performance of our application.

➢ Other purposes of the system are to get knowledge of database distribution and when to use SQL database and when to use NoSQL database, to learn new technologies like testing harness – mocha.js and performance tuning techniques.

❖ System Design:

➢ The purpose of this system was to implement the replica of Amazon Fresh web application. According to the requirement, performance tuning of the whole application was main and important point of the problem statement.

➢ Technologies used for the development

- Scripting Languages: JavaScript
- Front end: Angular.Js
- Back-end: Node.Js
- Messaging Protocol: AMQP.
- Middleware: RabbitMQ
- Markup Language: HTML
- UI/UX: CSS, Bootstrap, Jquery.
- Database: MongoDB, MySQL.
- Session: Express Session (Here the data gets stored in MongoDB when a user logs in)
- Encryption Technique: Bcrypt
- Framework: Express.

- SQL Caching: Redis
- Testing Harness: Mocha.Js
- Image Handling: bodyparser of Express. Ng-upload in Angular.Js.

❖ What our application can do.

➢ Admin:
- Admin is the main controller of the whole application.
- Admin can Add, Remove and Edit: Farmer, Product, Truck, Driver and Customer.
- Admin can view all the data within the system and change and edit as per required.

➢ Customer:
- Customer can create account using signup page.
- Customer will use the service after login using the credentials.
- Customer will provide all the details like address, credit card details.
- Customer can search, and buy products as per requirement.
- Customer will have options to review the products and farmers.

➢ Product:
- Product can be added by admin and farmers.
- Product will have quantity and the user will be notified when the products available is less than 5.
- Product will be shown when searched through parameters.
- Product page will have all the product details and all the reviews given to that product. There will be a bar which shows the average rating which will be calculated from all users.

➢ Farmer:
- Farmer can be added by admin and they can also register themselves providing details.
- Farmer can add products to sell. Farmer can define the tax.
- Farmer can show the video which gives a small introduction about the farmer.
- Farmer will only able to login when the farmer is approved.

➢ Cart:
- Cart will be disabled when the user is not logged in.
- Cart will be enabled when the user logs in.
- The data in the cart will always be saved whenever a user goes to cart.
- From cart the user can proceed to checkout only if the order is >= 50$. Where the user can.

➢ Order:
- User can place an order and view the orders which are placed by orders.
- User can only add the order if the address and credit card details are present.
- User cannot place an order if the products quantity available is less than needed.

Member's Contributions to Project

Members:

Aashay Shah - Aashay designed the entire schema before starting the implementation part. He then created a private repository on github for working together efficiently. Everyone decided entire project details and Aashay along with Neel distributed the work among members. Aashay completed home page with search button with autocomplete functionality. Aashay created cart with validation of minimum shopping amount of $50. He created order page with maximum four days delivery and user can also select preferable time for delivery. He did login and sign up implementation with user session and Bcrypt for encoding user password. He implemented order logic when user places his/her order. He completed product search based on user input and various category.

Vansh Shah – Vansh implemented the user side of the project. He implemented the 'Manage Your Account' under user where user can look on his entire activity. User can edit and update address details whenever needed, User can edit and update credit card details whenever needed, User can check all the reviews given on products as well as farmers, User can check all the order details where User can check current orders, Canceled orders and successful past orders. Apart from this, Vansh implemented few dummy pages like help, carrer, condition of user, Privacy Notice, About etc. I also worked with the product search page.c

Neel Patel – Neel implemented Products and Farmer management functionality with Add, Update and Delete functionality. He added accept and reject farmer feature. So only approved farmers can login and add products. Neel also implemented login and signup for farmer with session using Passport middleware. Neel added farmer profile feature where farmer can update his/her information. He also added Add, Update and Delete function for products, where farmer can manage the products that he/she is selling.

Vivek Mehta – Vivek implemented the product side of the application. He did the entire validation and UI of the product's description. He also completed the review side of the product where the customer can place a review. The farmer page was also implemented by Vivek where the user can view the farmers video, info and create a review about a farmer. He also did the part of the Product Search page which lists all the products according to the search by the farmer. He worked on the category search, where the user can search for the products according to the category defined.

Darshil Saraiya – Darshil implemented the management of Trucks, Drivers, Customers and Orders of the admin use-case. He implemented all the Add, Update and Delete functionalities of Drivers as well as Customers and Trucks. He also implemented the Driver Assign functionality of Order List page admin side. He also did the session management in admin use case. Darshil did the UI in admin side. The password encryption during admin-login is also done by Darshil.

He also implemented the listing of information with editable and savable data of Trucks, Drivers, Customers and Orders. He also implemented the MySQL queries using Sequelize module.

Project Description

1. **Your Object Management Policy.**

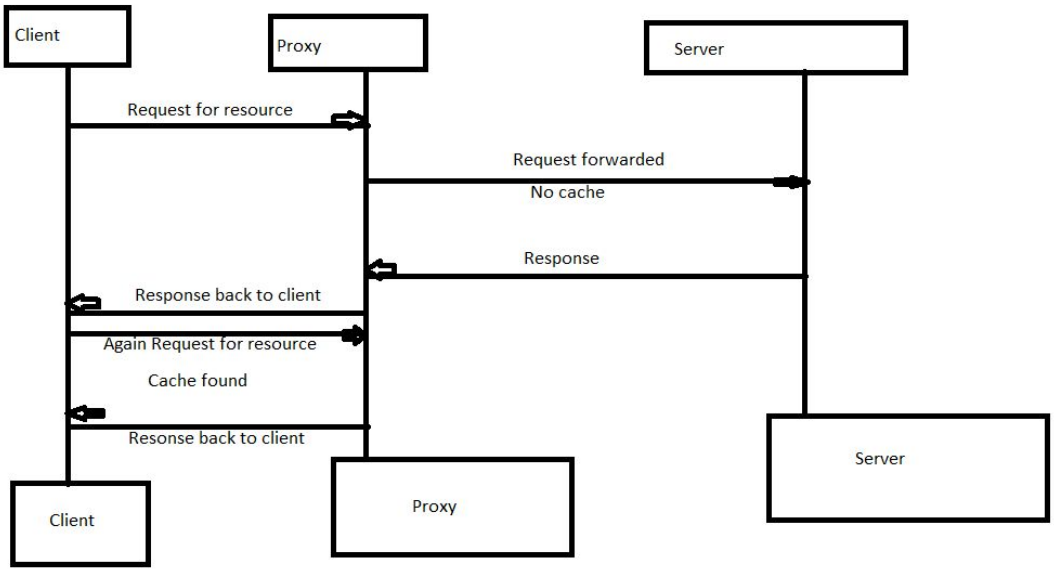
➢ **Sequalize** – ORM – Object relational mapping.

➢ **Mongoose** – ODM – Object Document Modelling.

2. **How you handled heavy weight resources?**

➢ **RabbitMQ :** Implements the Advanced Message Queuing Protocol (AMQP). To handle very large number of parallel user requests from client to server side to get data very fast from different queues. Messages are routed through exchanges before arriving at queues. Several RabbitMQ servers on a local network can be clustered together, forming a single logical broker.

➢ **Redis Caching:** Caching GET requests with particular timeout for particular GET API's.



- **Connection Pooling:** To create multiple connections so that we can use same connections to respond number of database requests. This way we can save time by not creating new connections on every call and also the connections will be closed.
- **SQL Query Optimization:** Here we optimized the query by reading only needed data and send only the data which are needed. This way the performance increases as the data is sent from server to client side.

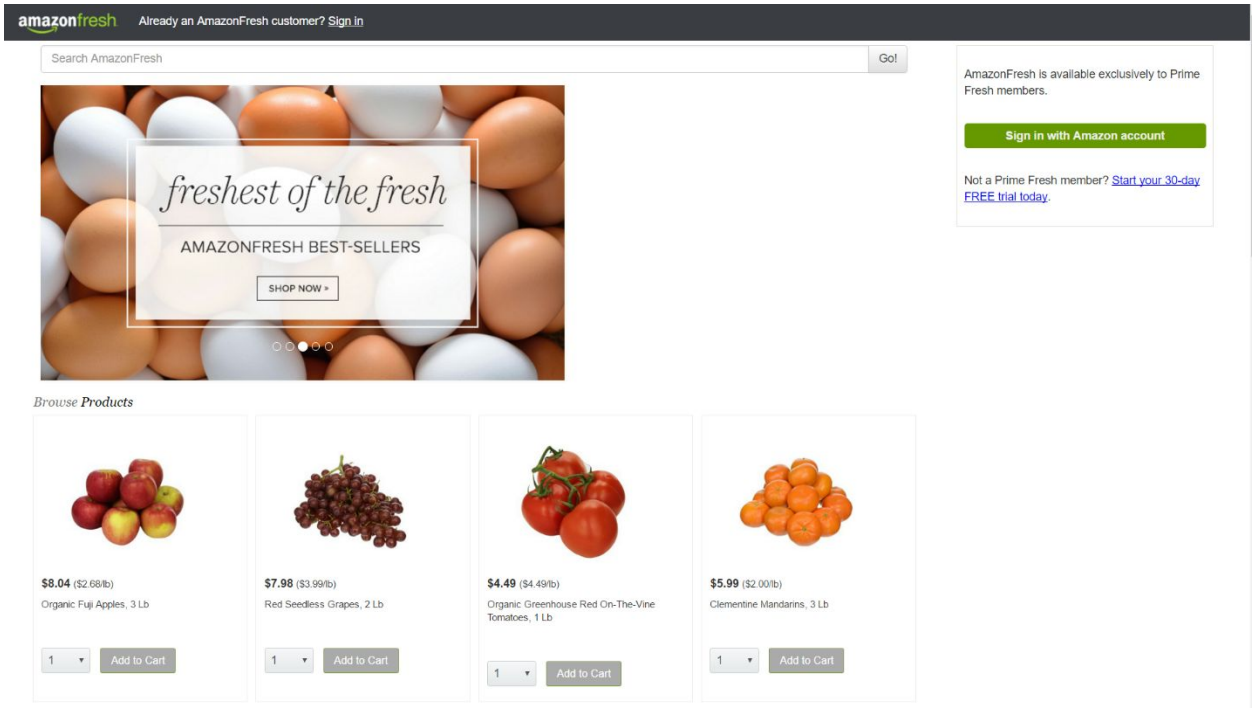
3. **The policy you used to decide when to write data into the database**

- When new data write or data update is happening at that time we are writing respected data into database. Also the data is written in Redis for caching purpose.

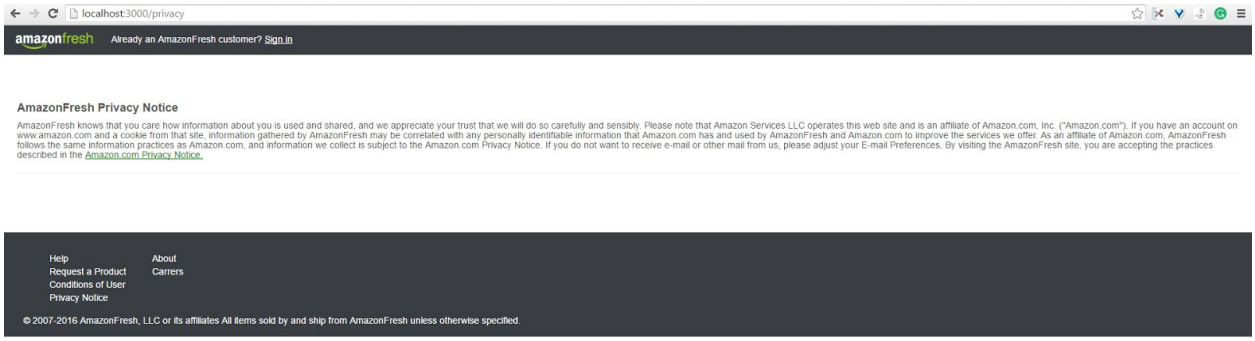
Client Application

Browse Without Login.

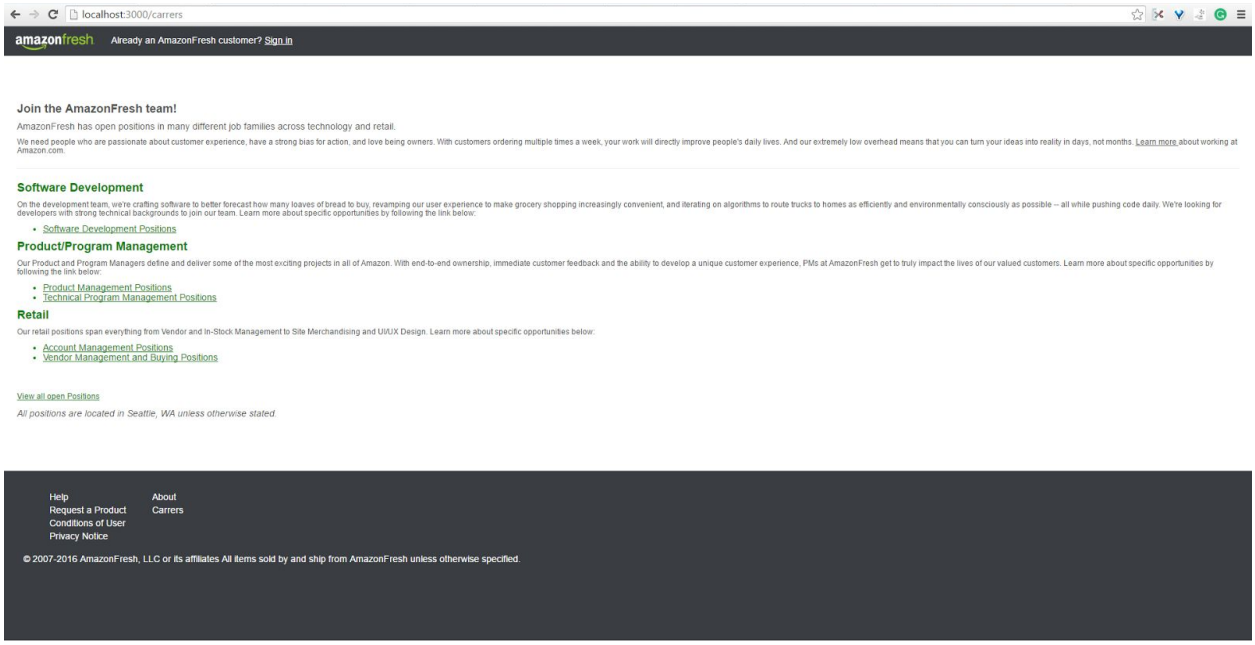
Home page: Please Signin menu at top



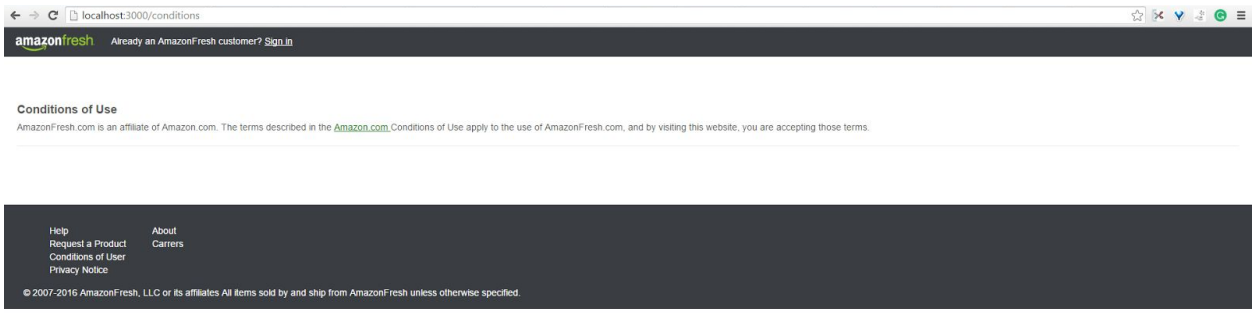
Dummy Page 1: Privacy Notice



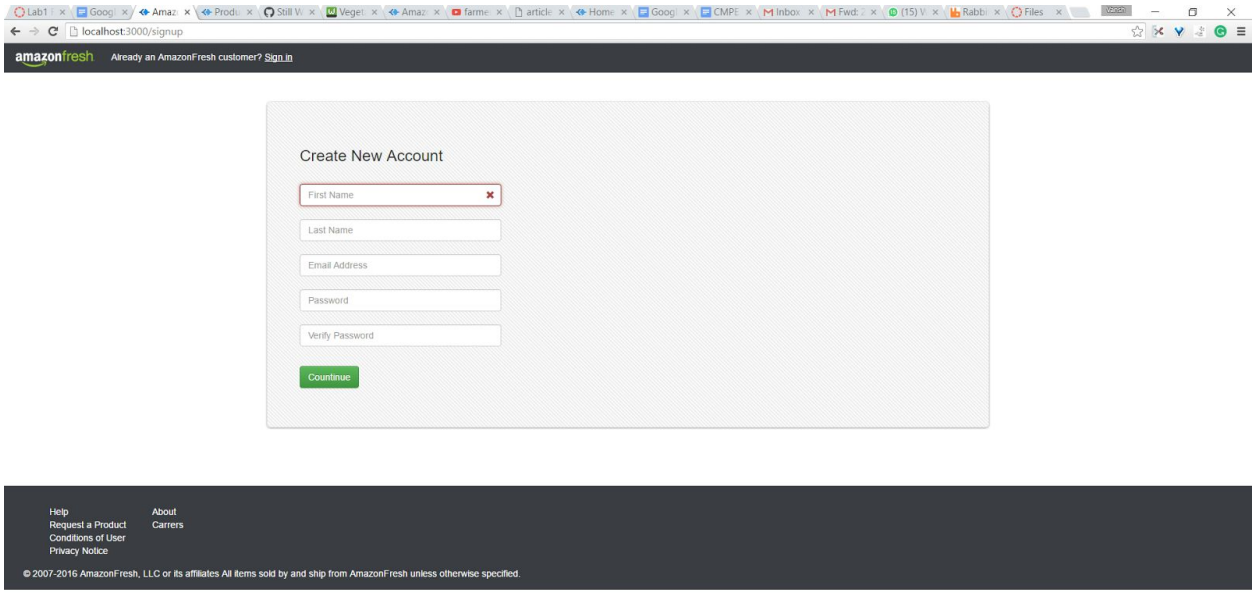
Dummy Page 2: Carrers

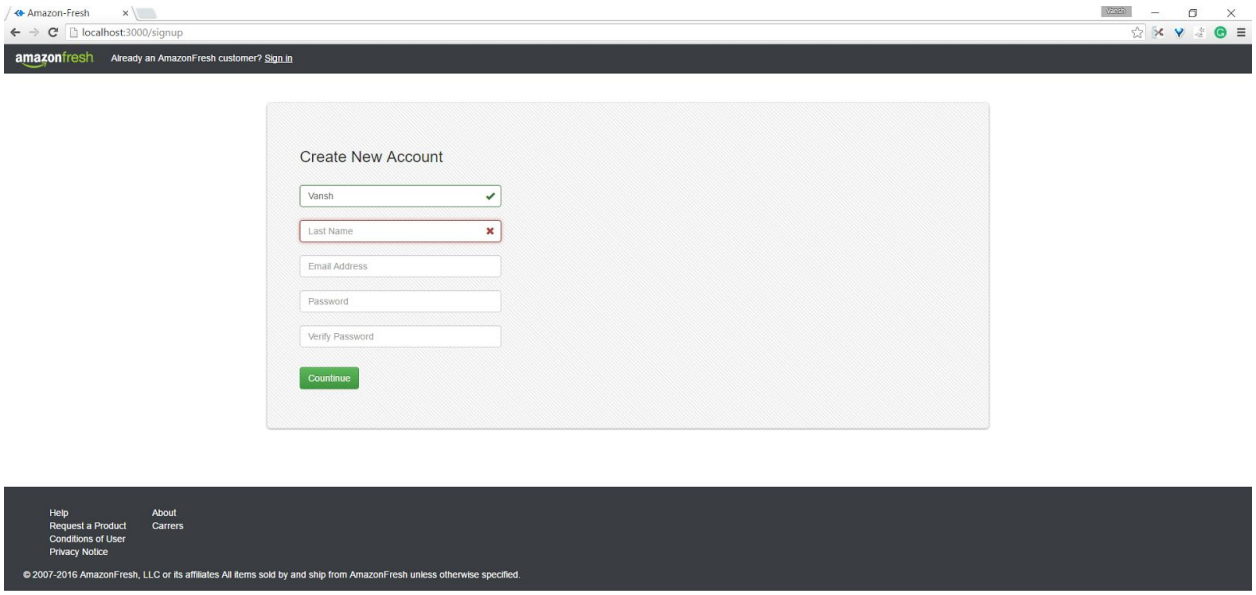


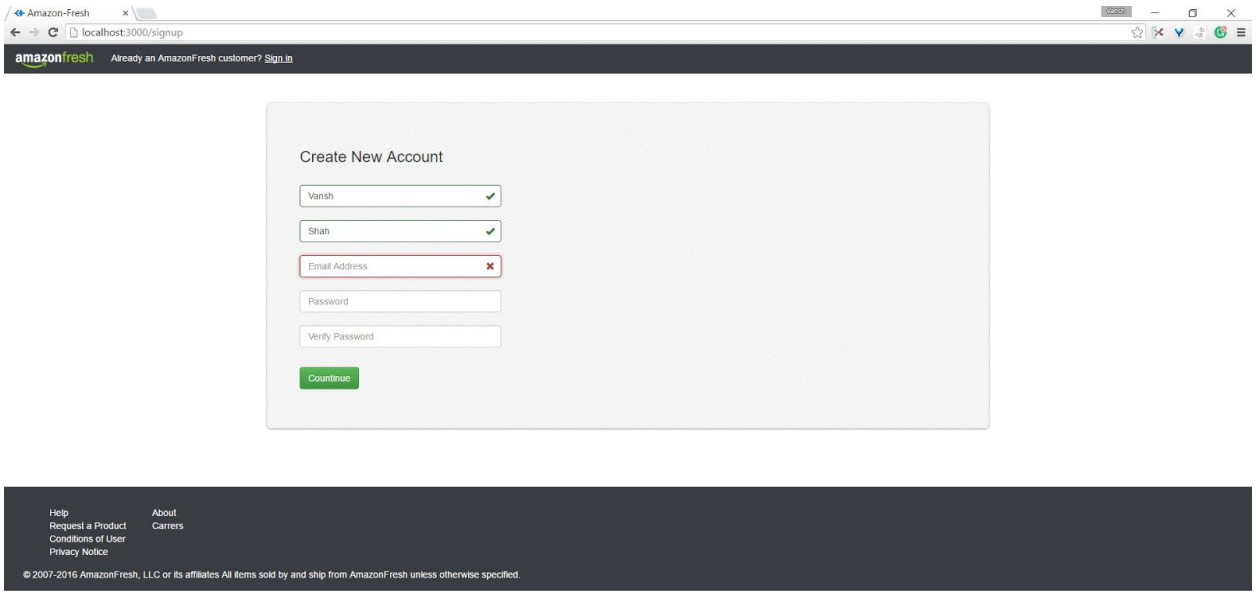
Dummy Page 3: Conditions of User

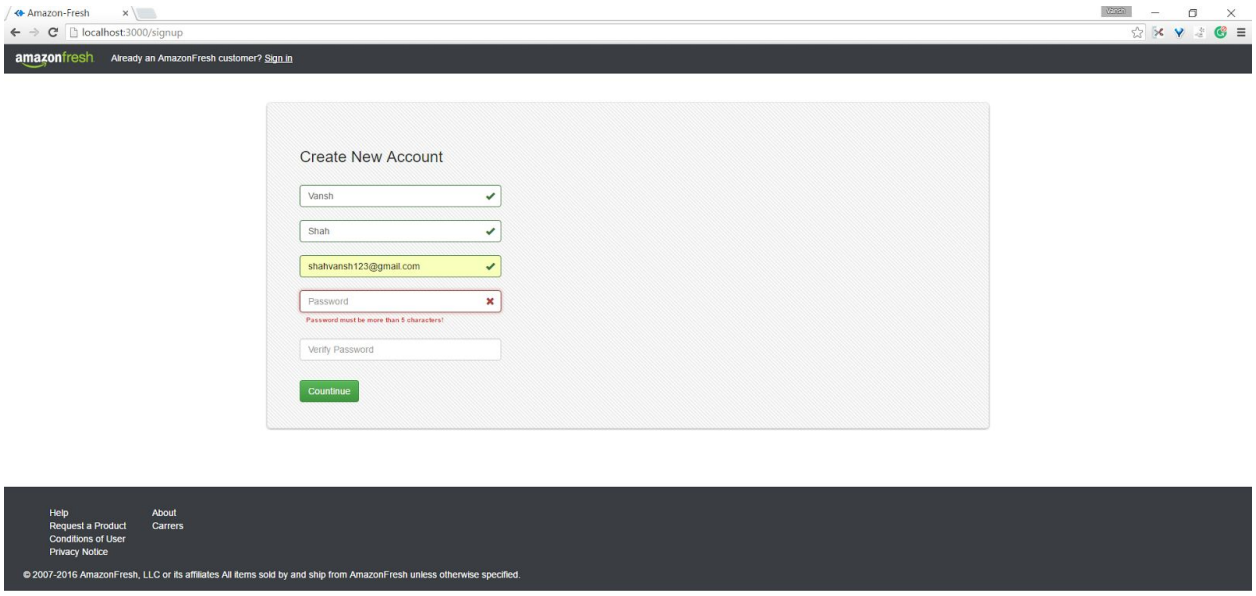


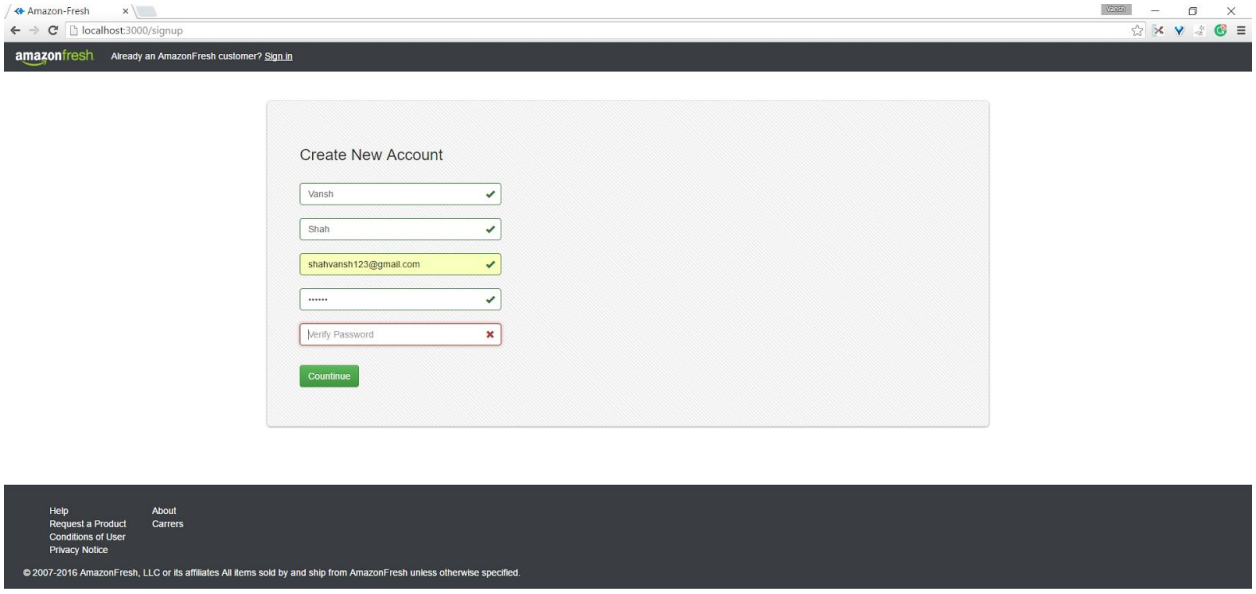
Client needs to Signup

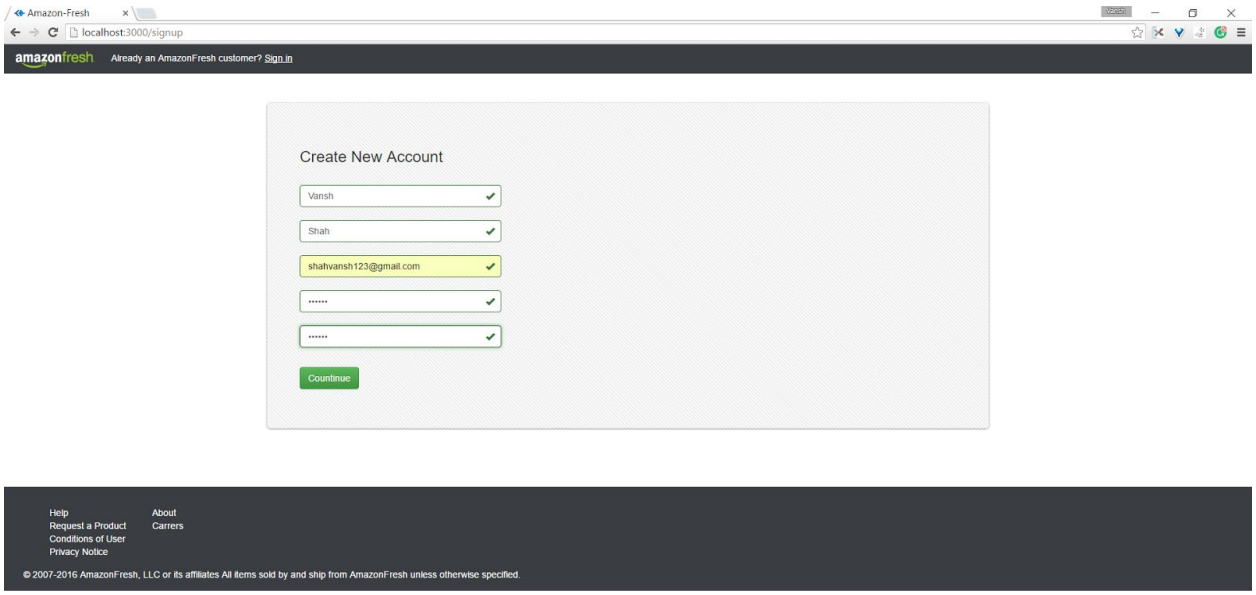




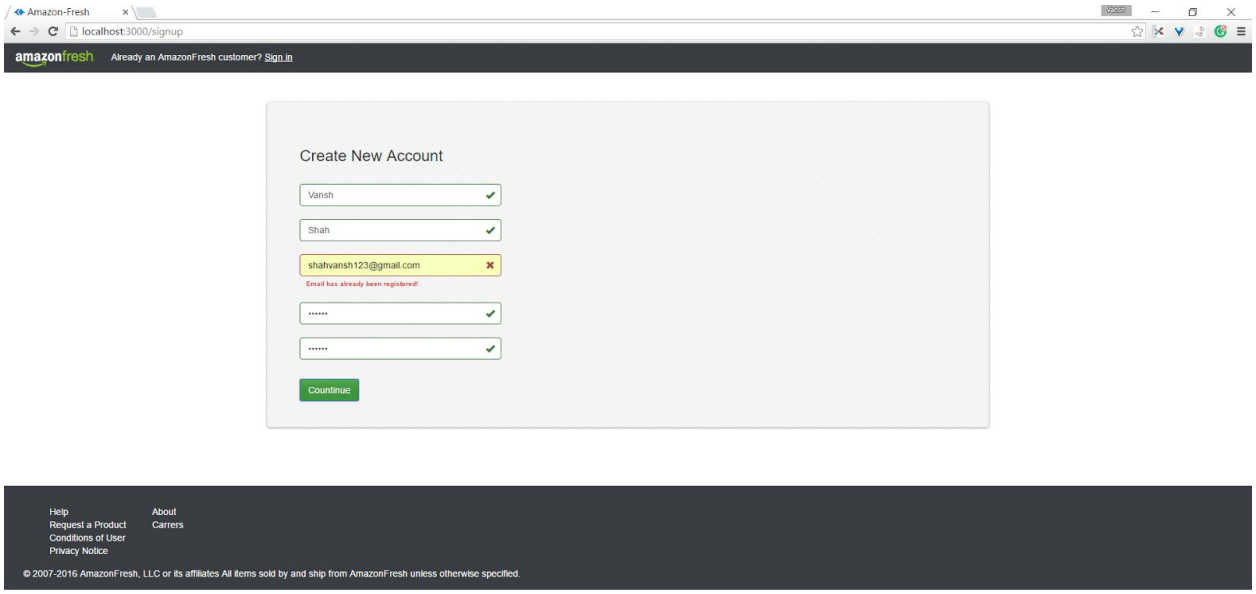

Amazon-Fresh
localhost:3000/signup
amazonfresh
Already an AmazonFresh customer? Sign in
Create New Account
Vansh
Shah
Email Address
Password
Verify Password
Countinue
Help
Request a Product
Conditions of User
Privacy Notice
About
Carrers
© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

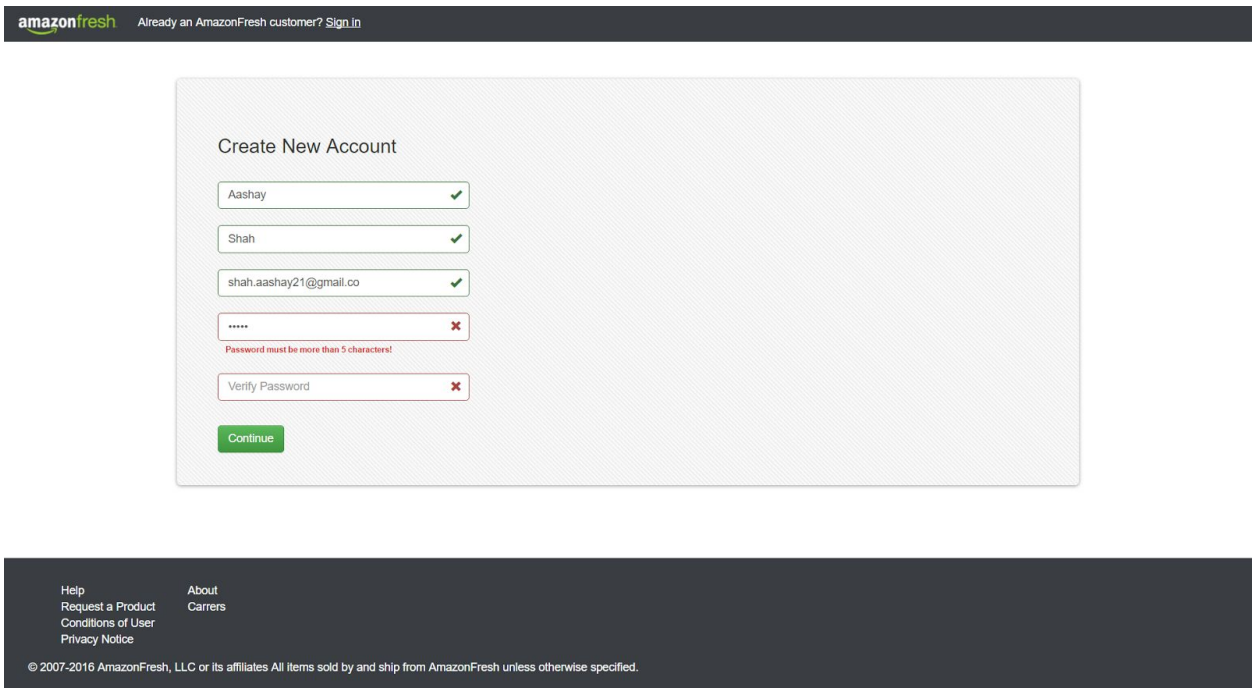

Amazon-Fresh
localhost:3000/signup
amazonfresh
Already an AmazonFresh customer? Sign in
Create New Account
Vansh
Shah
shahvansh123@gmail.com
Password
Password must be more than 5 characters!
Verify Password
Countinue
Help
Request a Product
Conditions of User
Privacy Notice
About
Carrers
© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.


Amazon-Fresh
localhost:3000/signup
amazonfresh
Already an AmazonFresh customer? Sign in
Create New Account
Vansh
Shah
shahvansh123@gmail.com
Verify Password
Countinue
Help
Request a Product
Conditions of User
Privacy Notice
About
Carrers
© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

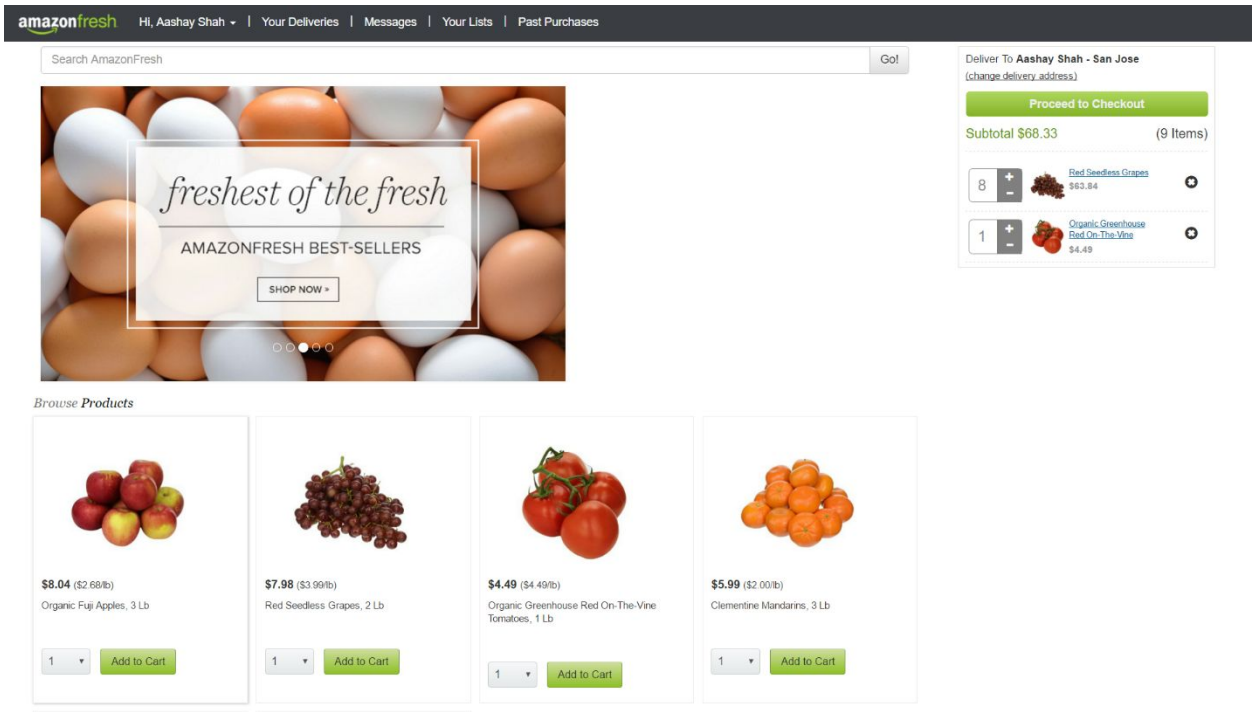

Amazon-Fresh
localhost:3000/signup
amazonfresh
Already an AmazonFresh customer? Sign in
Create New Account
Vansh
Shah
shahvansh123@gmail.com
Countinue
Help
Request a Product
Conditions of User
Privacy Notice
About
Carrers
© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

Email Id already Taken:

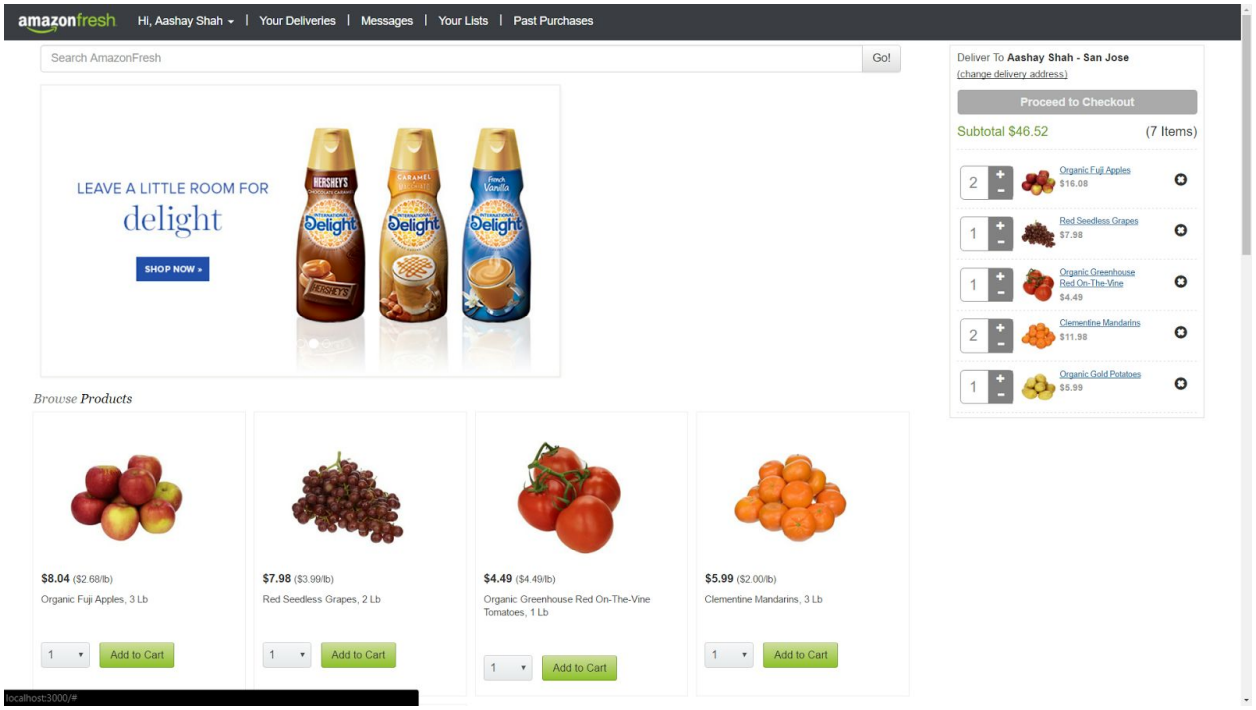




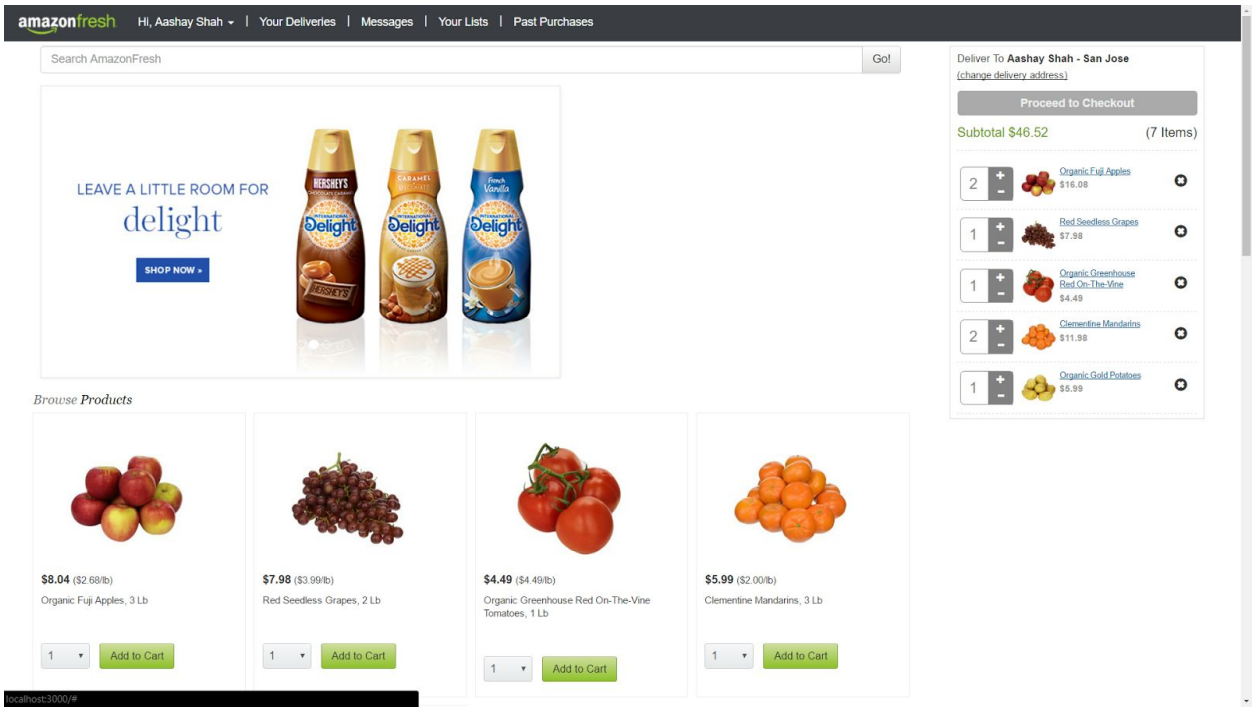
The menu bar changes on Login



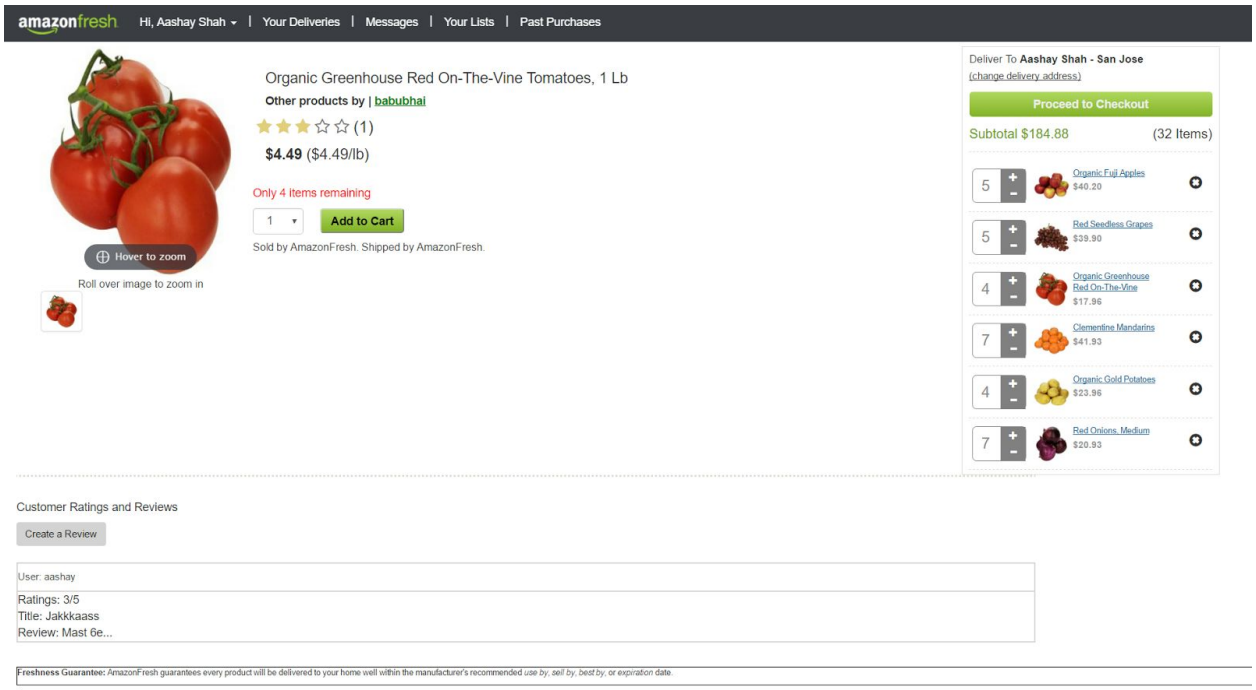
The cart appears on the right side. Procced to checkout is disabled as order is less than 50$



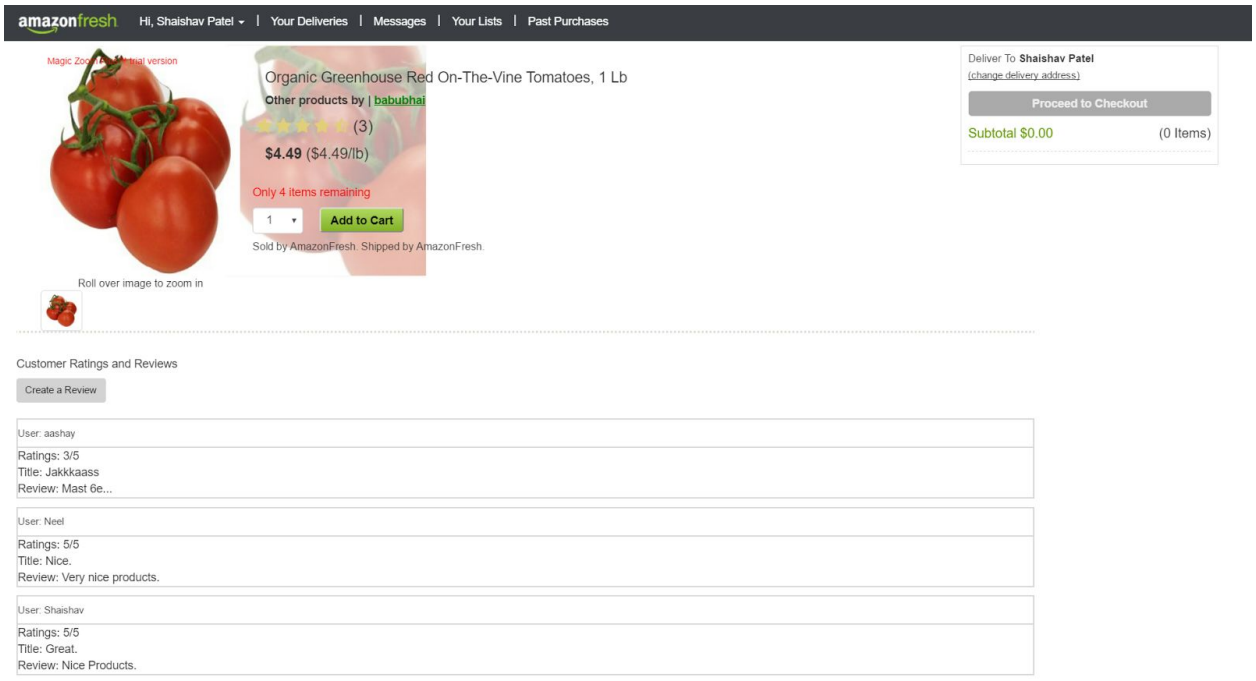
All the products ‘Add to Cart’ button Enabled



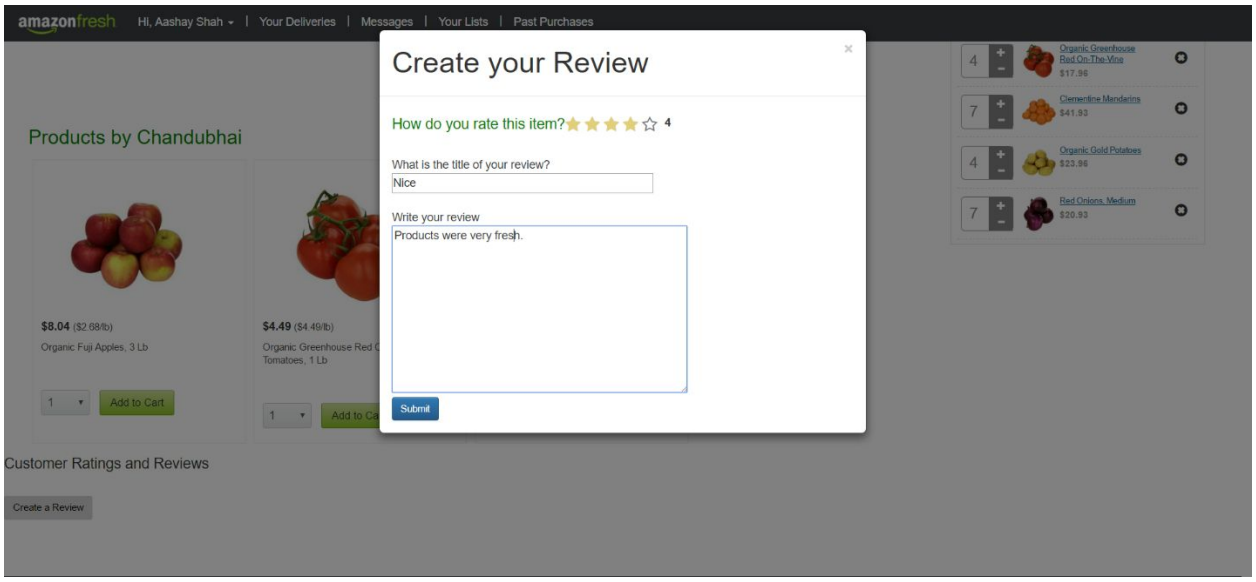
Product Page after Login



Zoom Effect:

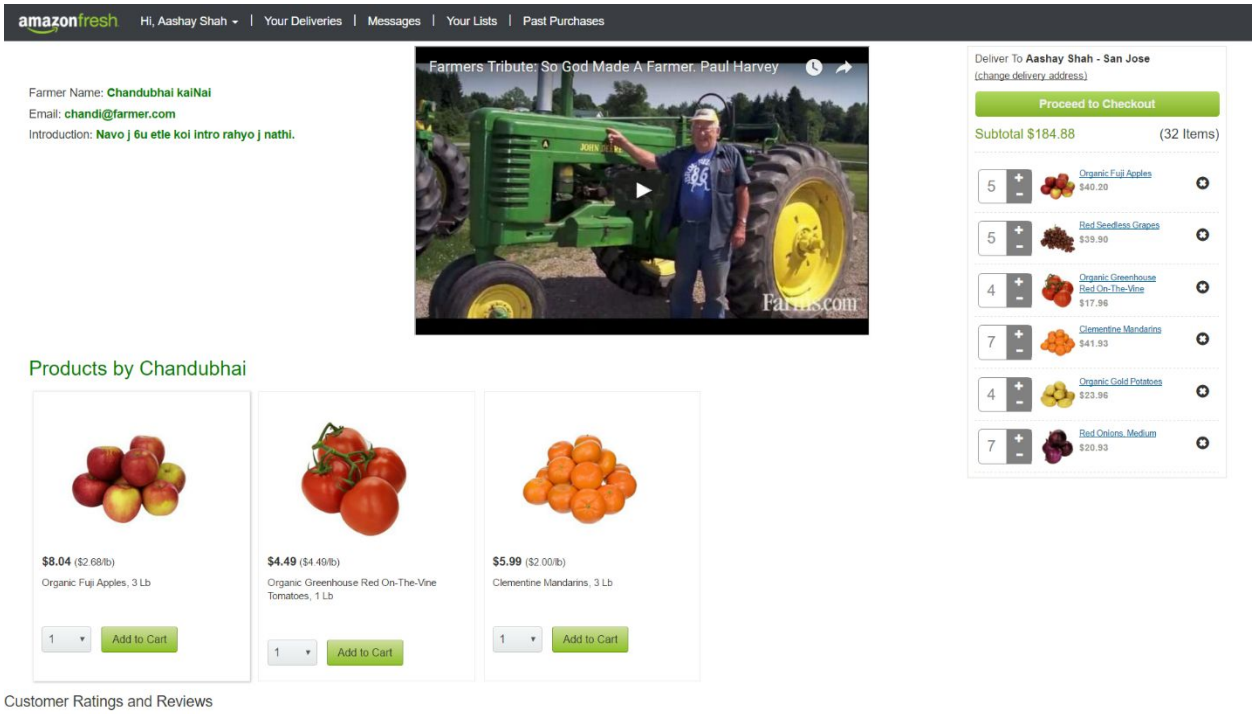


Create Review

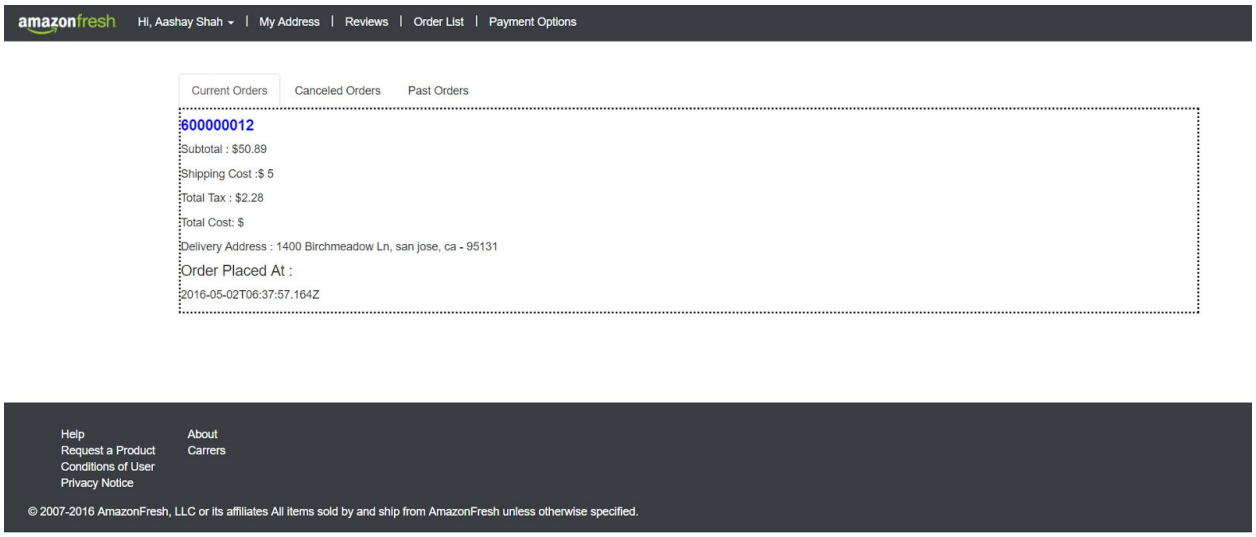


Product Reviews

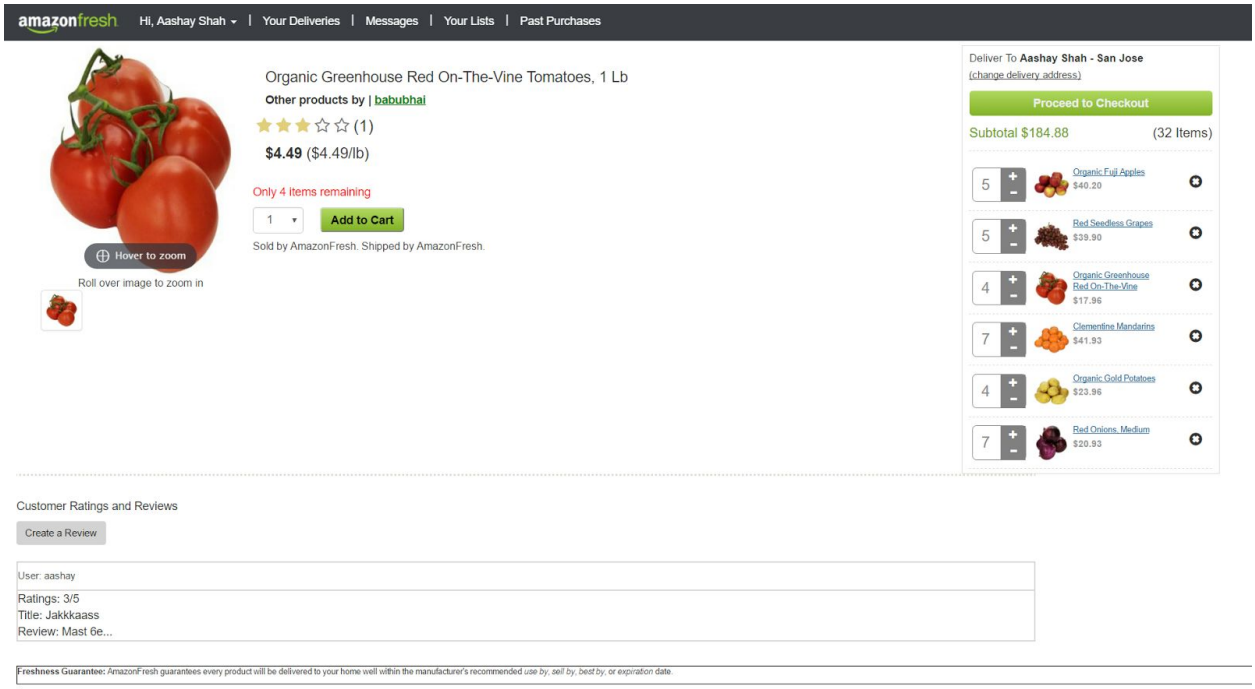
Farmer page after User login

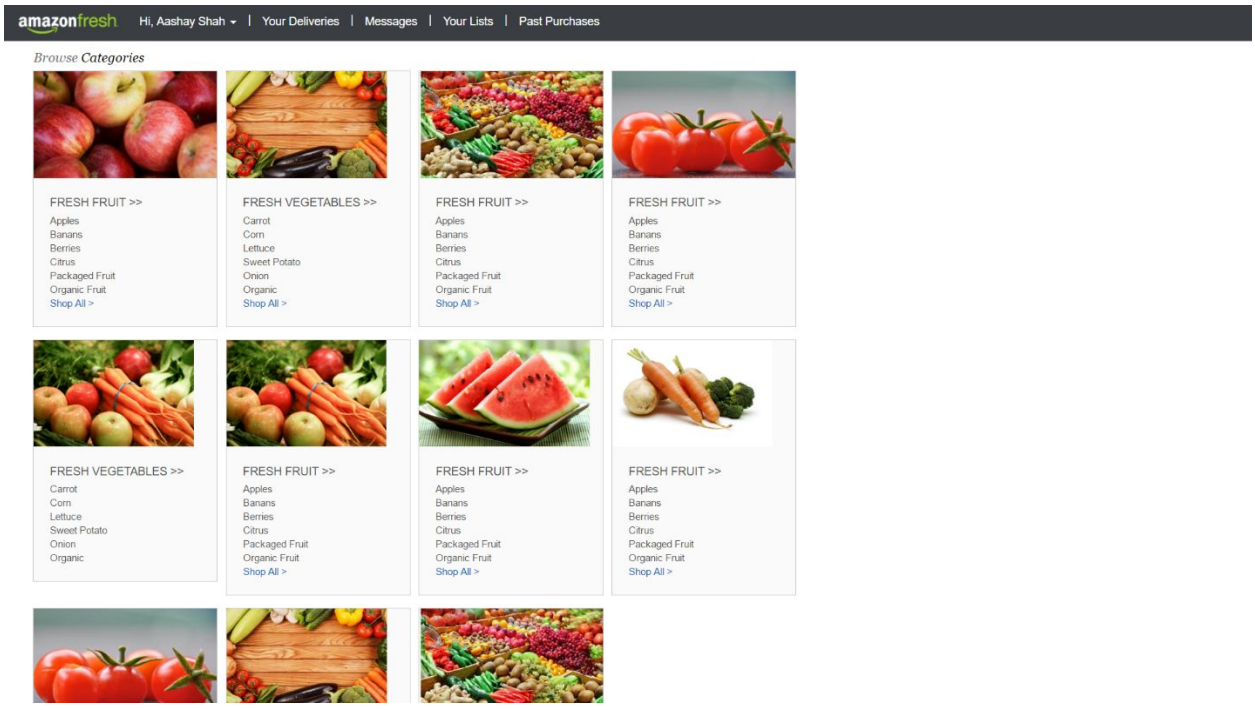


Order Details

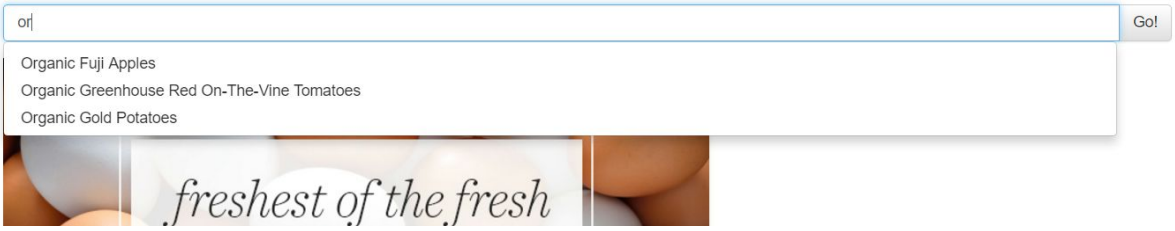


Checking the Product Quantity and displaying when quantity isless than 5

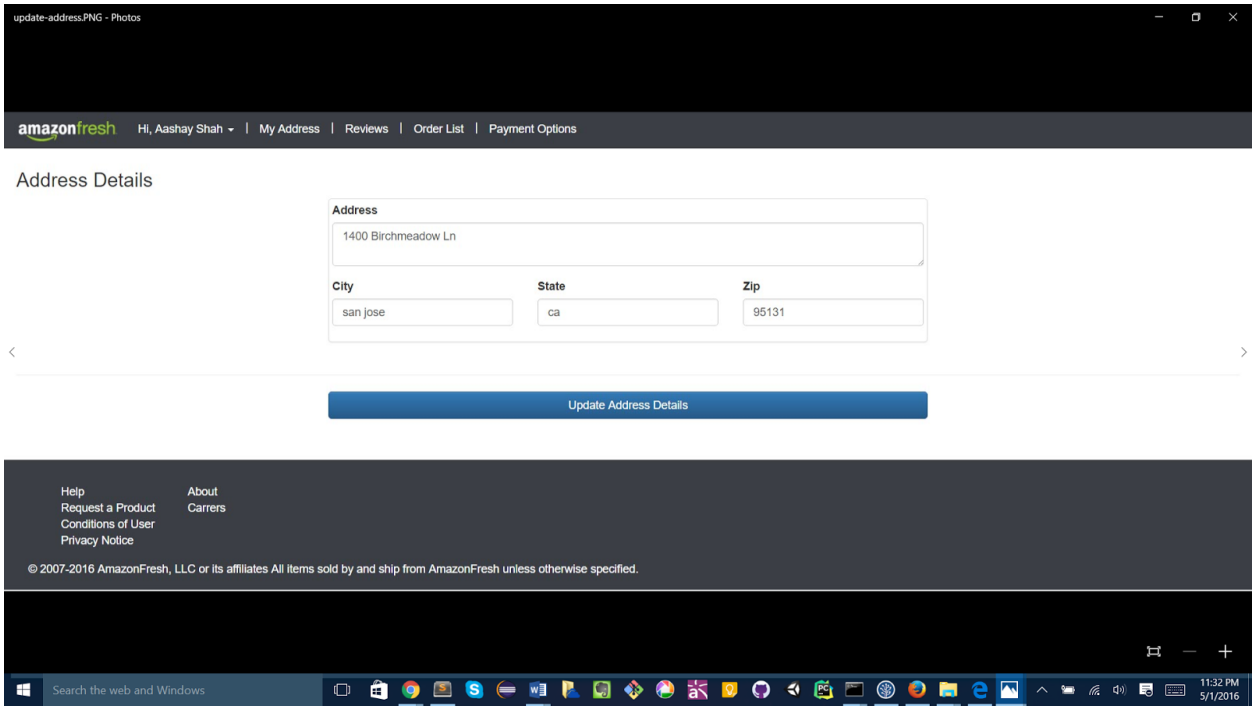




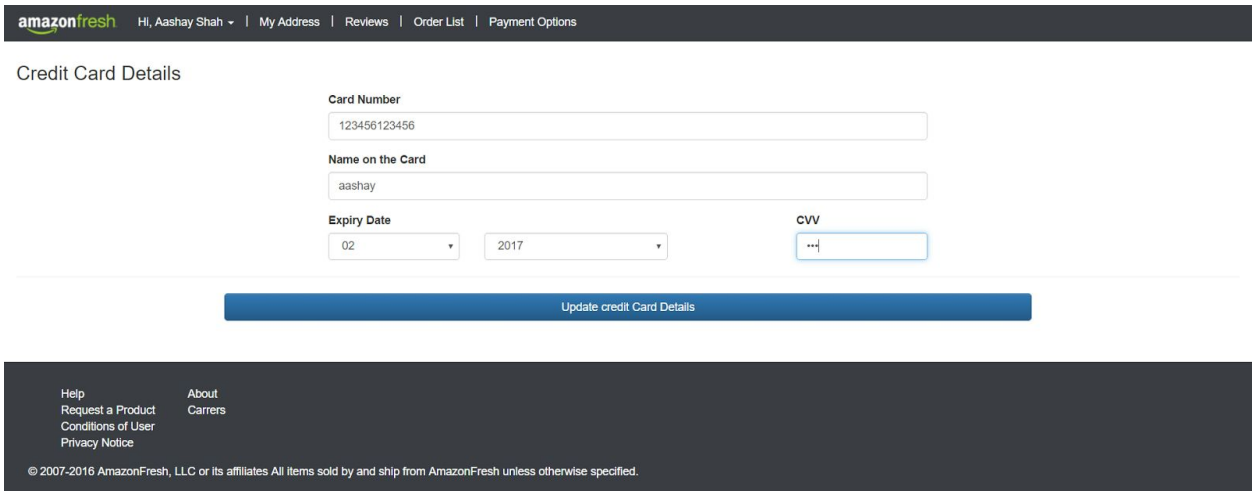
Search the product in search Tab



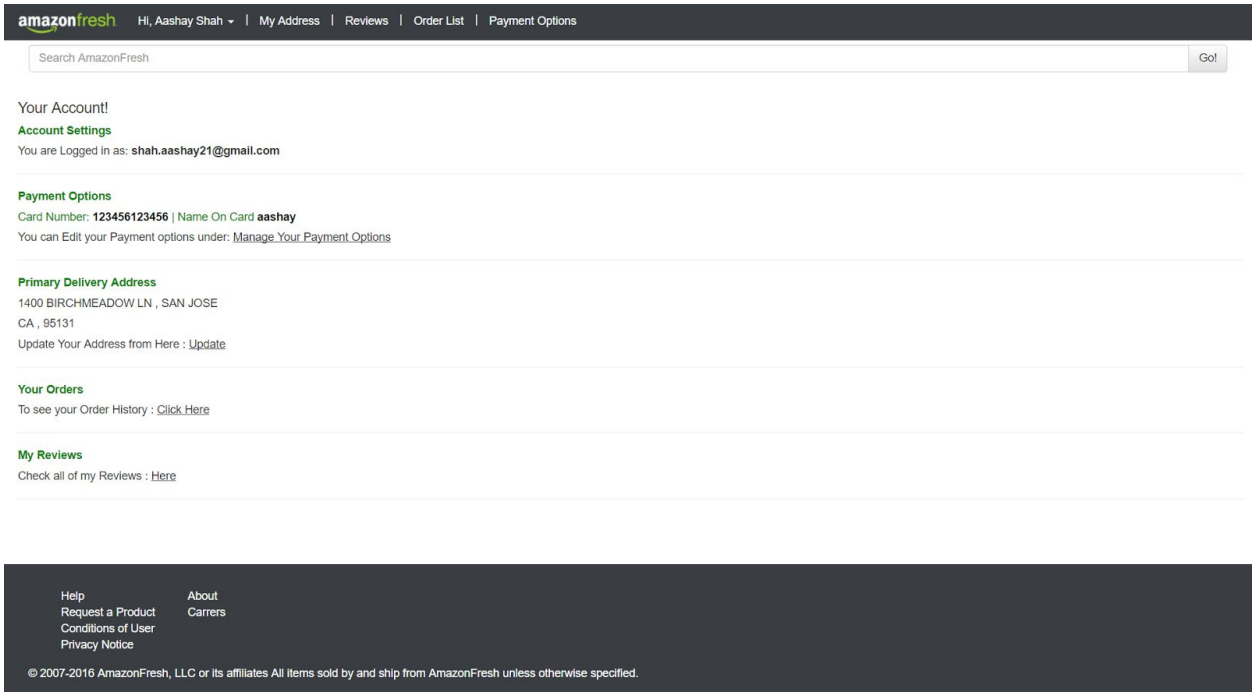
Edit Address



Edit Card Details



The User Account Details



Cart Details

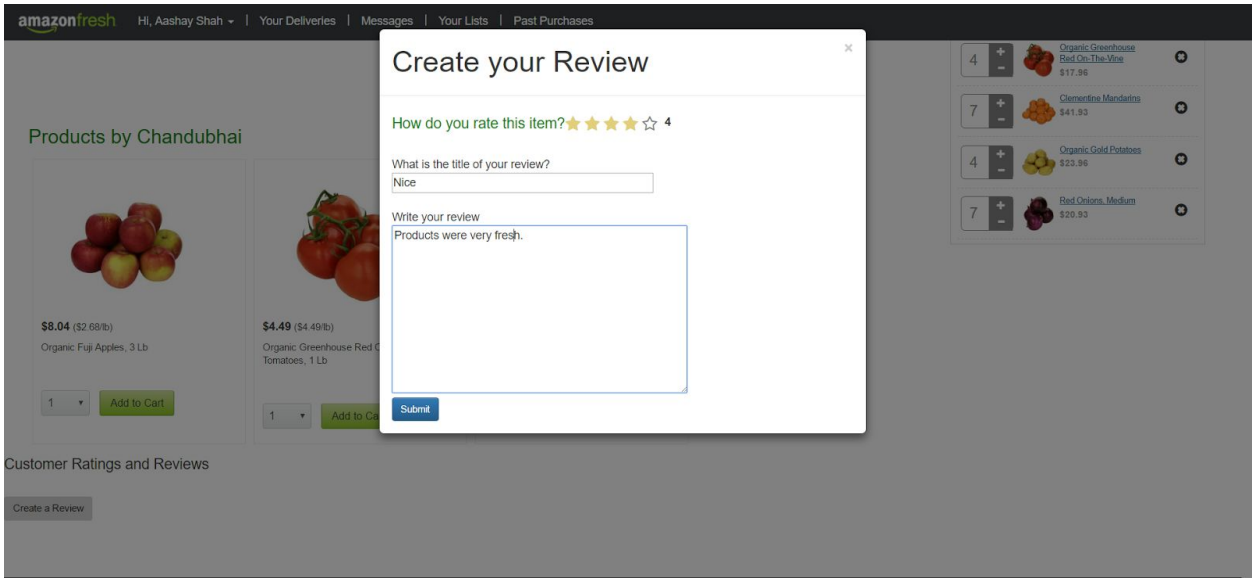
Deliver To **Aashay Shah - San Jose**
(change delivery address)

Proceed to Checkout

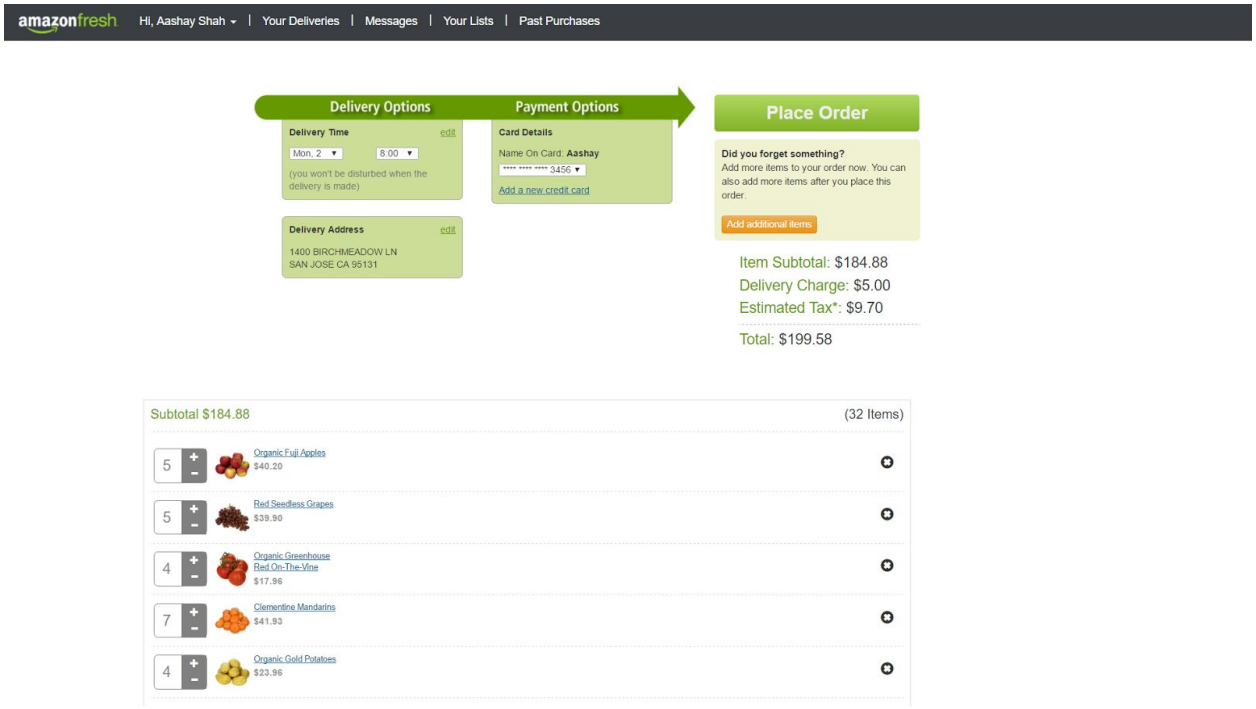
Subtotal $184.88 (32 Items)

5	 Organic Fuji Apples $40.20	
5	 Red Seedless Grapes $39.90	
4	 Organic Greenhouse Red On-The-Vine $17.96	
7	 Clementine Mandarins $41.93	
4	 Organic Gold Potatoes $23.96	
7	 Red Onions, Medium $20.93	

Write a Review with providing the stars

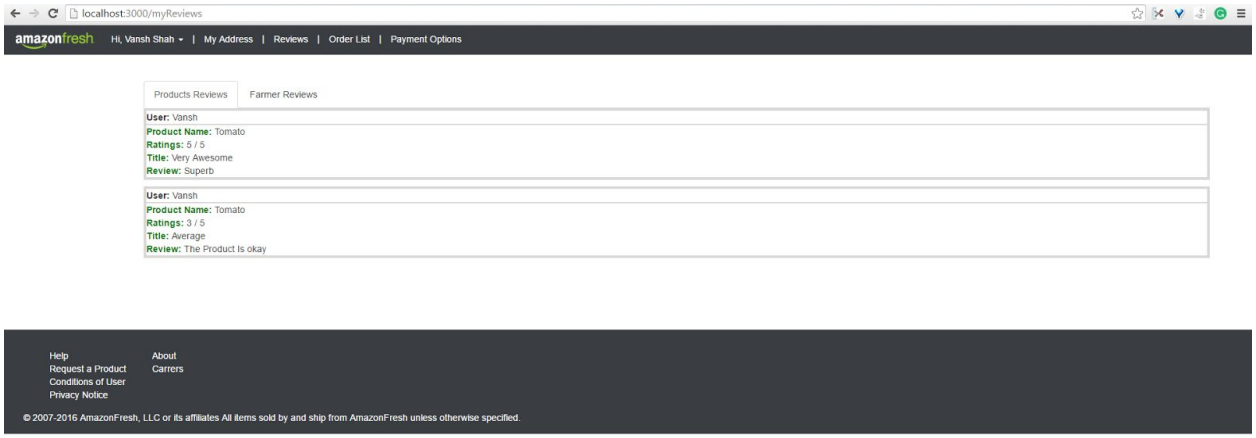


Proceed to Checkout

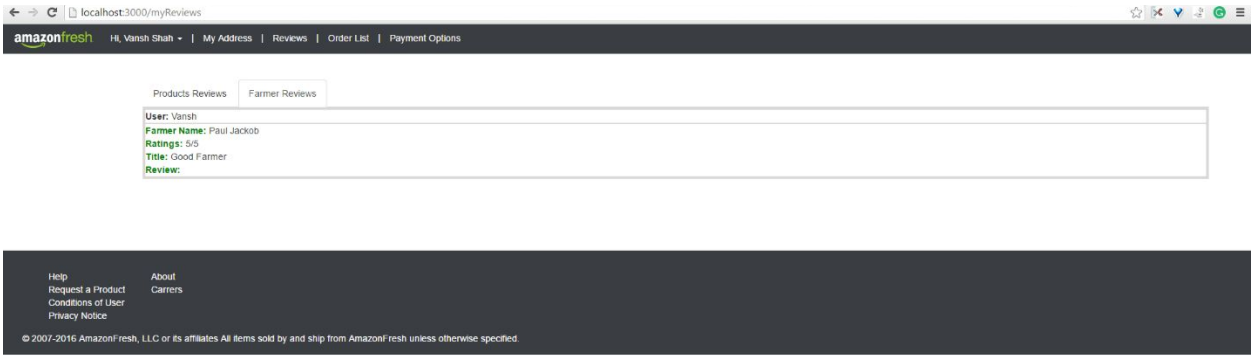


User Reveiws

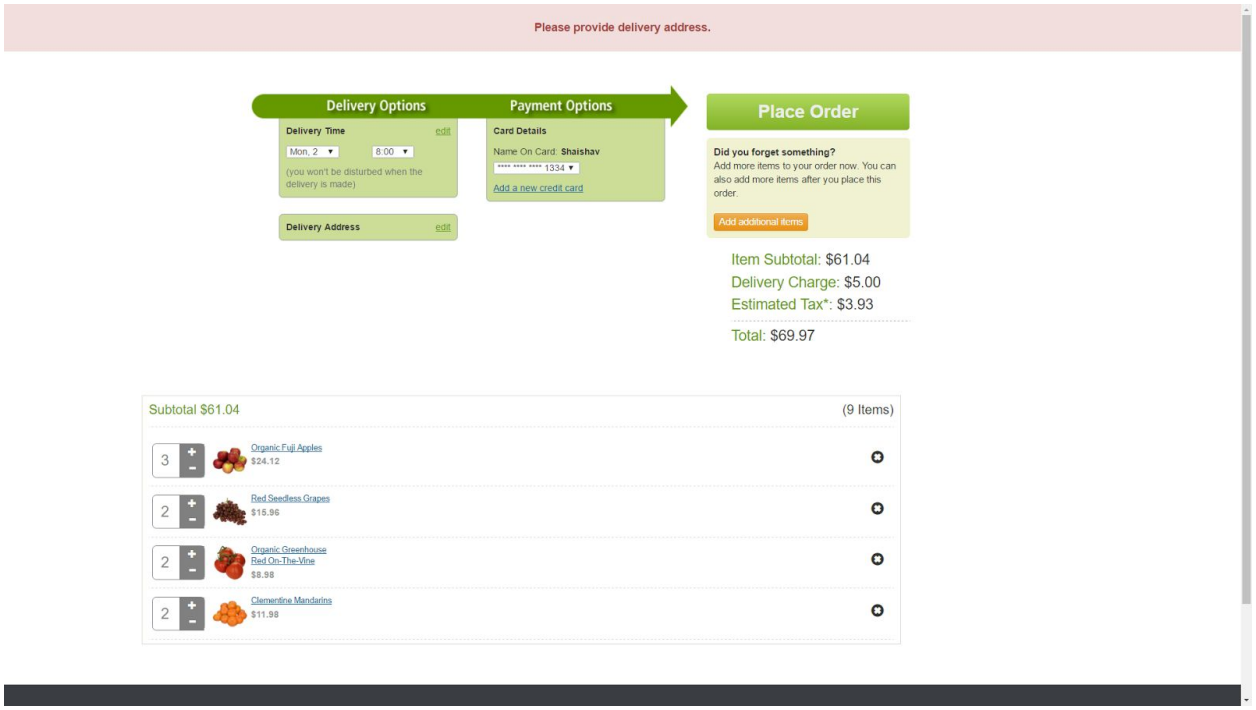
Product Review



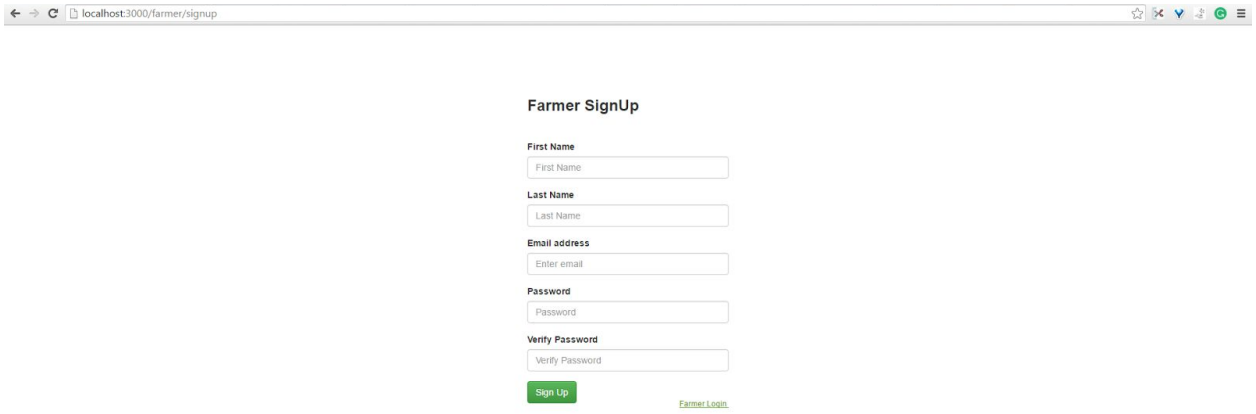
Farmer Review



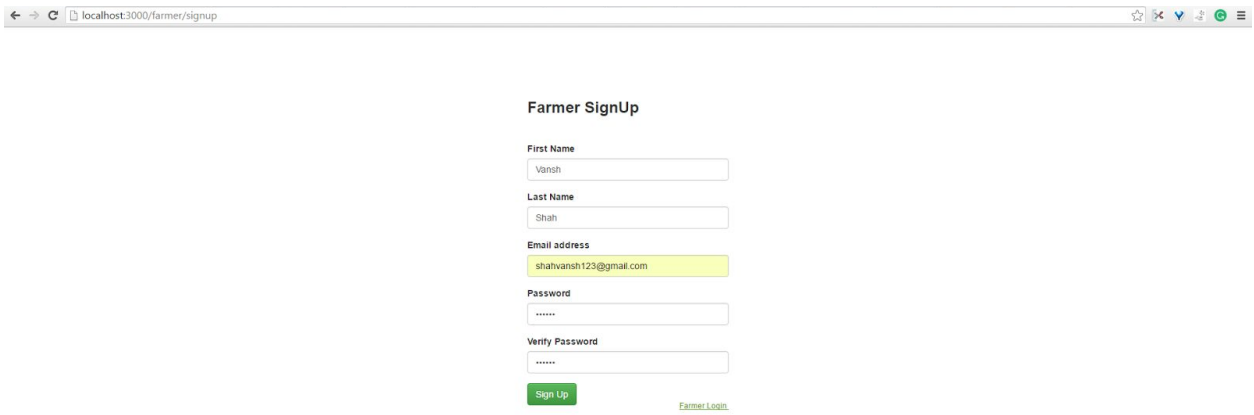
Please Provide Address Details



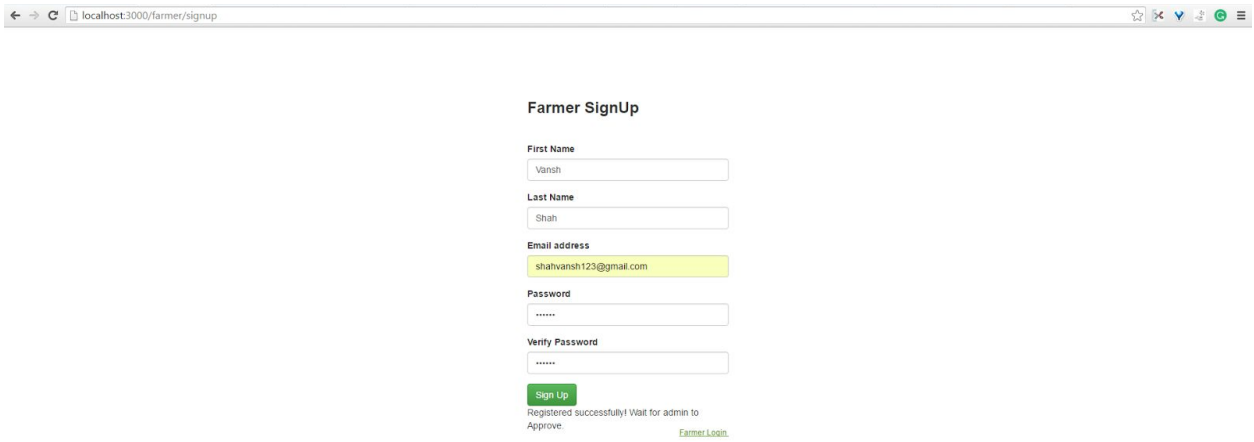
Farmer Signup Page



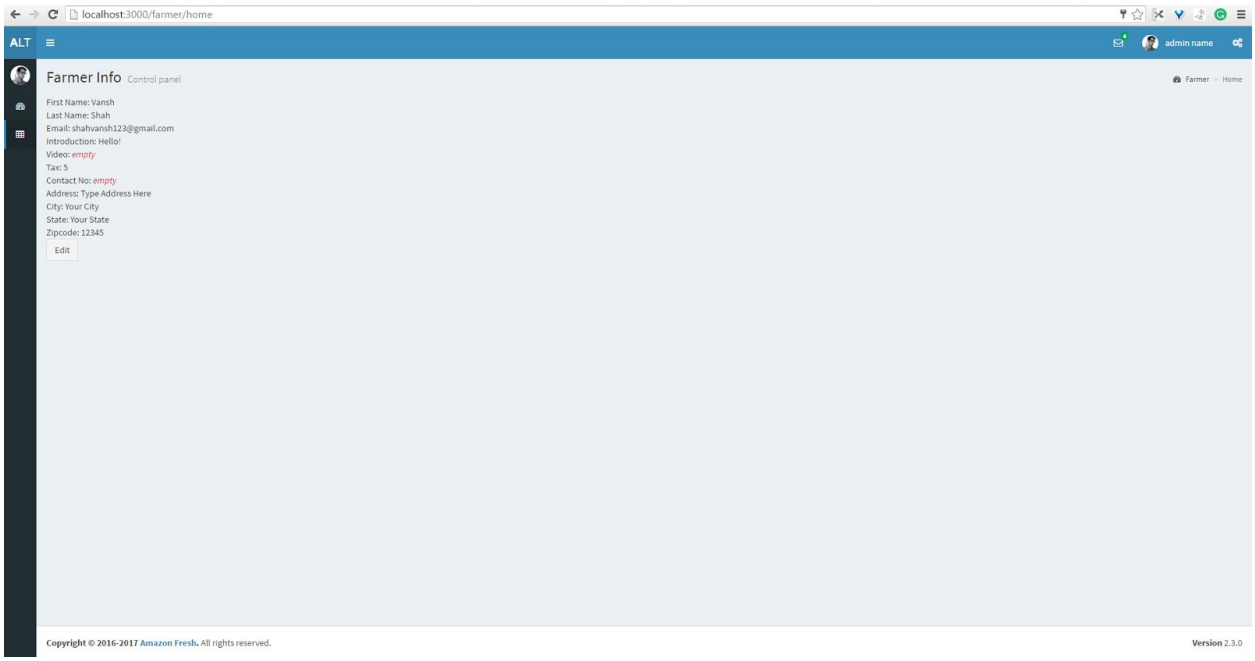
Farmer Signing Up



Farmer signed up successfully Now admin has to accept the farmer in order to login.

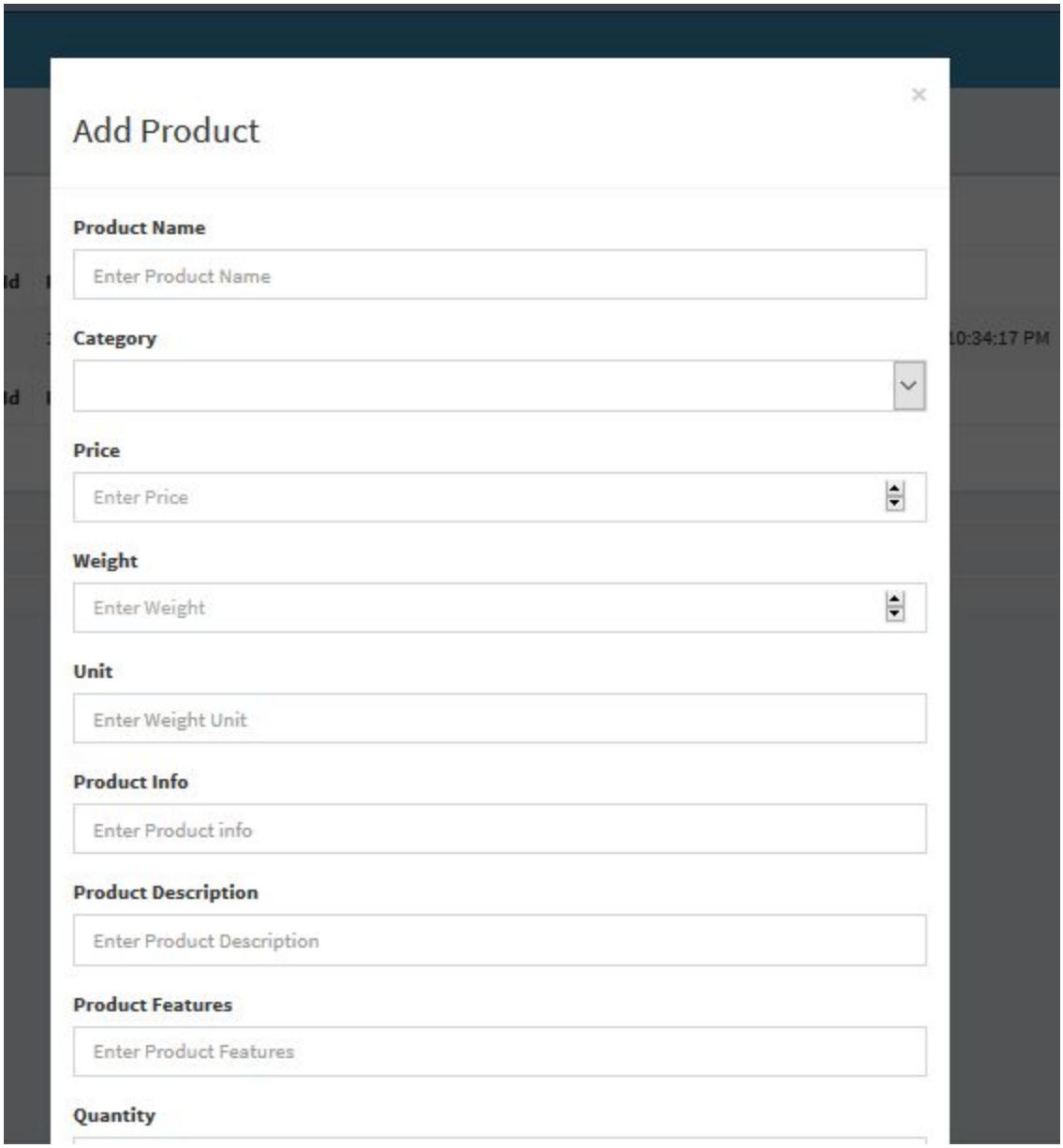


Farmer page after Login



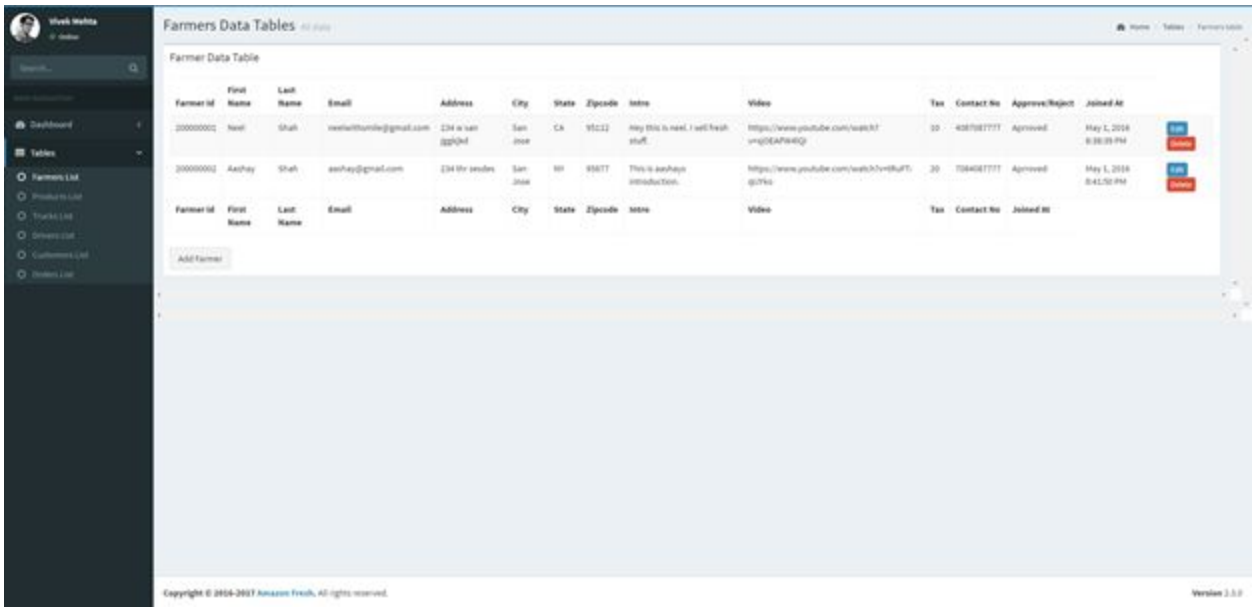
Farmer can edit the Farmer details


localhost:3000/farmer/home
ALT
admin name
Farmer Info
Control panel
Farmer
Home
First Name: Vansh
Last Name: Shah
Email: shahvansh123@gmail.
Introduction: Hello!
Video:
Tax: 5
Contact No:
Address: Type Address Here
City: Your City
State: Your State
Zipcode: 12345
Save
Cancel

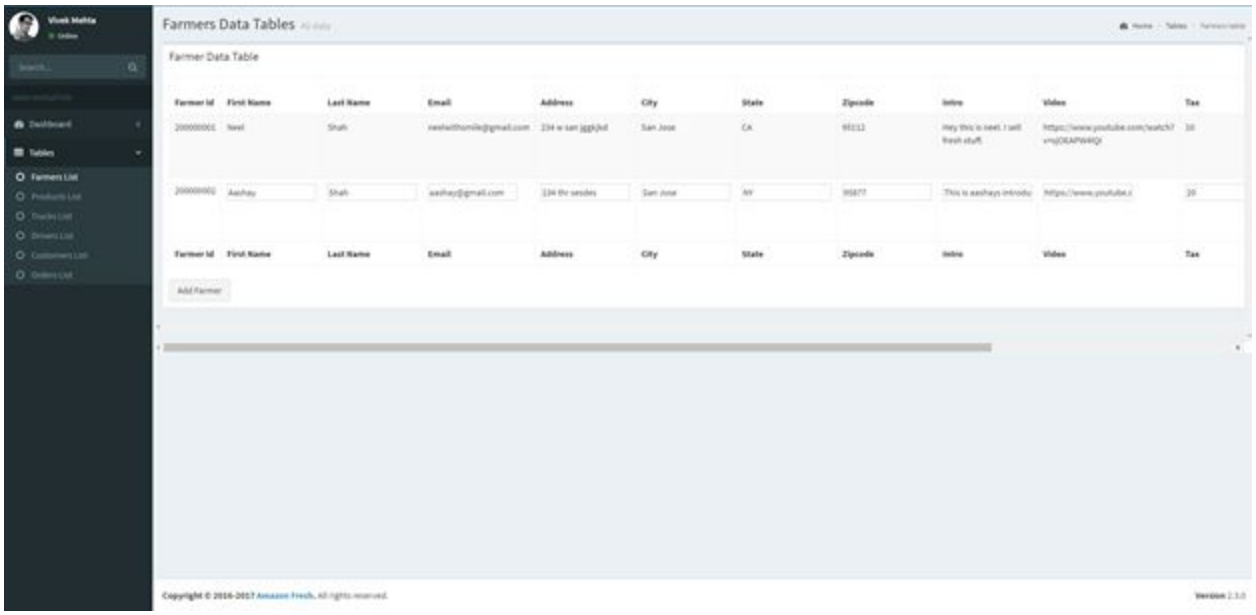


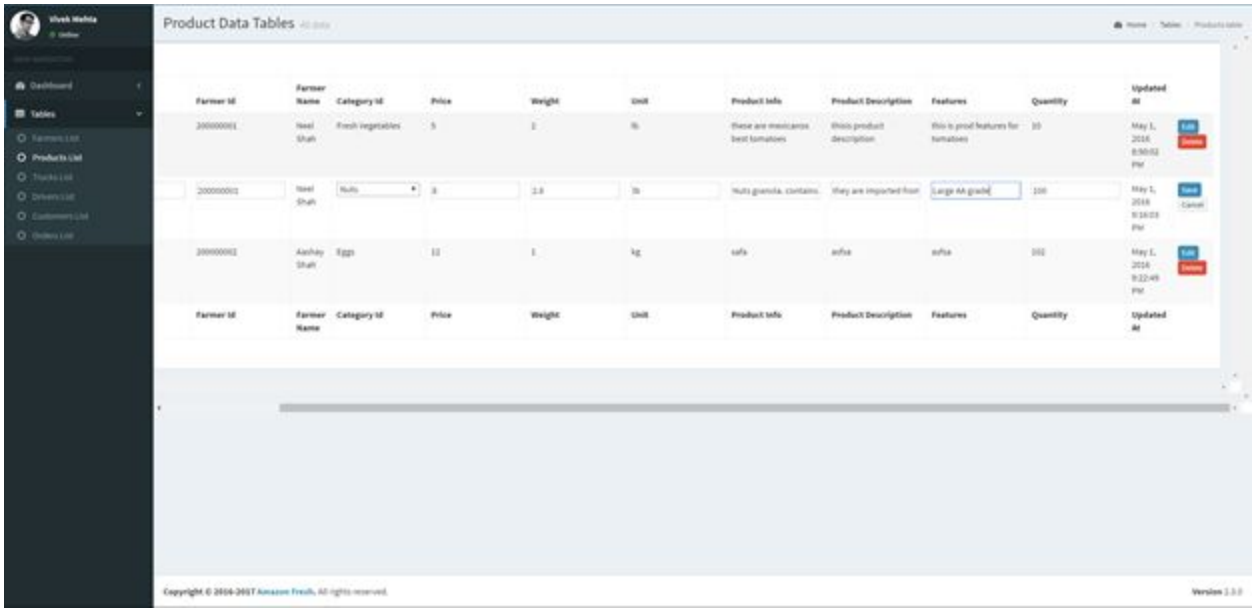
ategory Id	Price	Weight	Unit	Product Info	Product Description	Features	Quantity	Updated At	
Nuts	12	3	oz	sad	sda	AS	12	May 1, 2016 10:34:17 PM	Save Cancel
ategory Id	Price	Weight	Unit	Product Info	Product Description	Features	Quantity	Updated At	

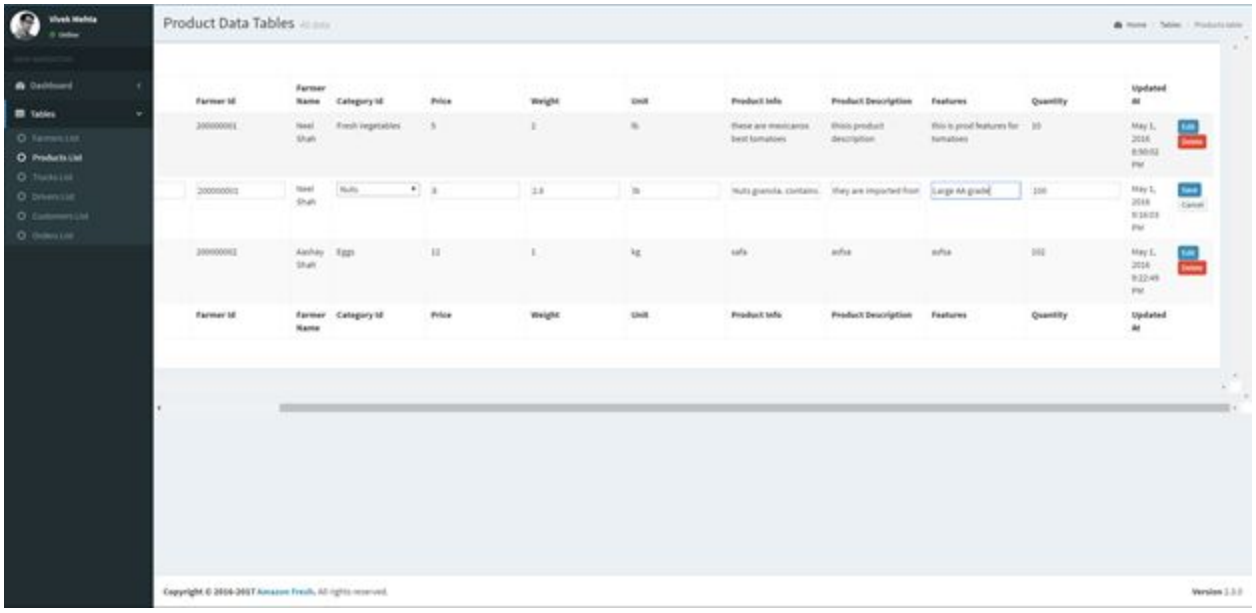
Admin Tab

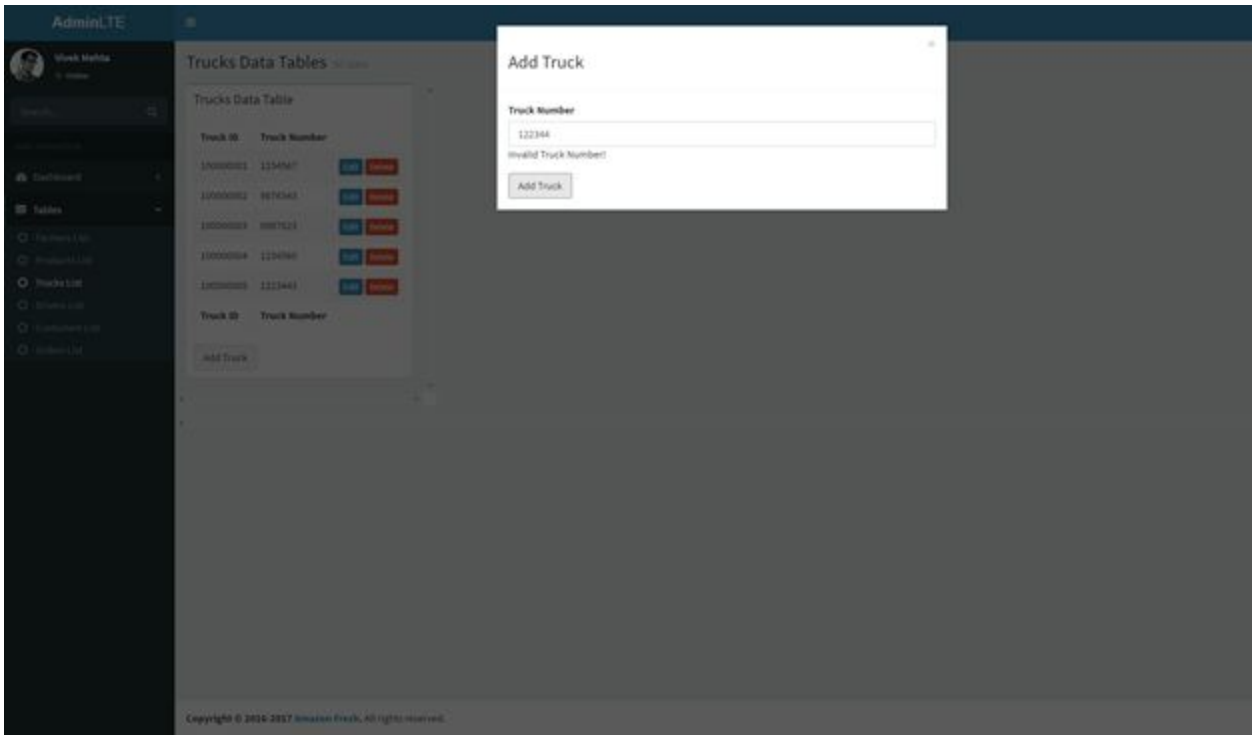

Farmers Data Tables
Farmer Data Table
Add Farmer

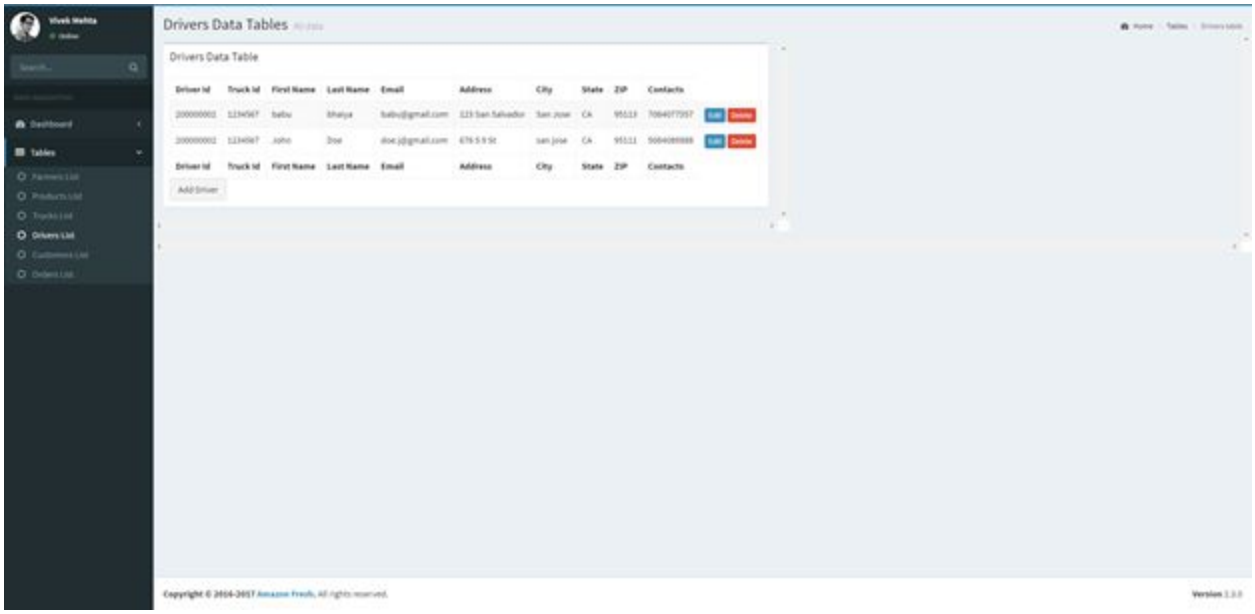

AdminLTE
Farmers Data Tables
Farmer Data Table
Add Farmer
First Name
Last Name
Email
Password
Introduction
Contact No
Youtube Video Link
Tax (Percentage)
Address
City
State
Zipcode

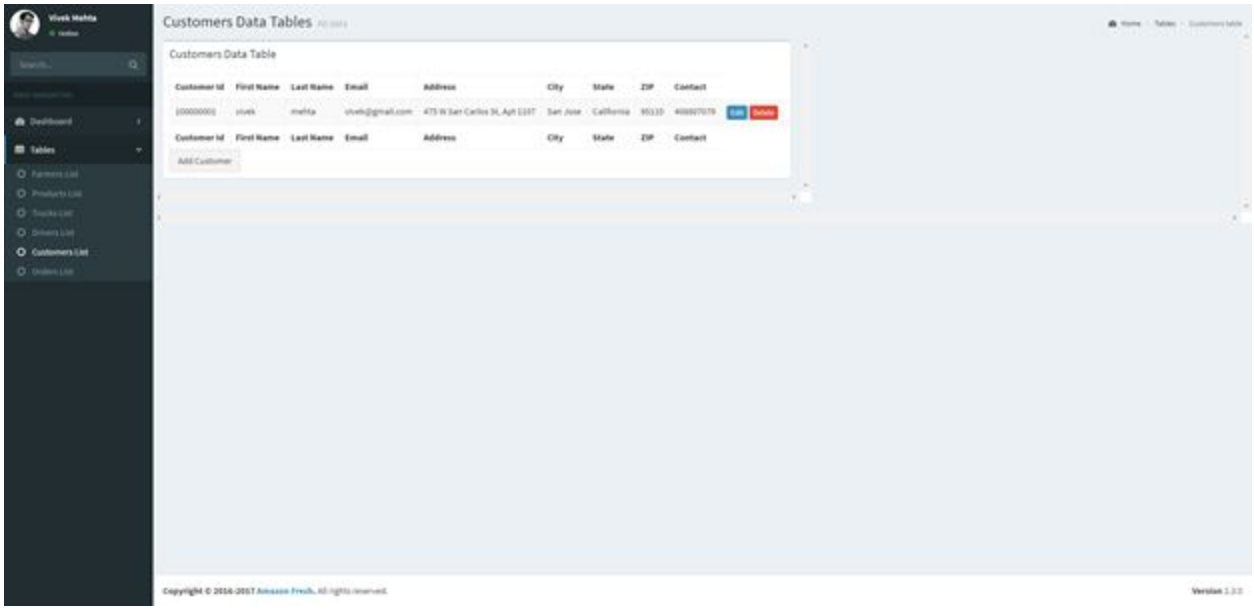

Vivek Mehta
Online
Dashboard
Tables
Farmers List
Products List
Trucks List
Drivers List
Customers List
Orders List
Farmers Data Tables
Farmer Data Table
Farmer Id
First Name
Last Name
Email
Address
City
State
Zipcode
Intro
Video
Tax
Add Farmer
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

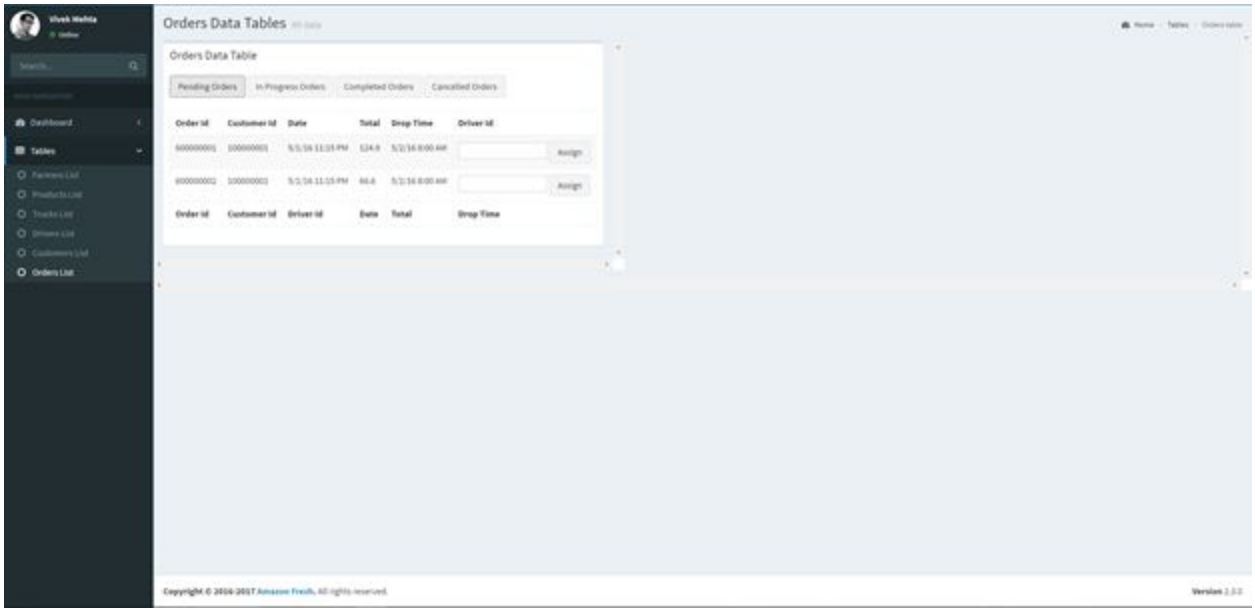

Vivek Mehta
Online
Dashboard
Tables
Farmers List
Products List
Trucks List
Drivers List
Customers List
Orders List
Product Data Tables
Farmer Id
Farmer Name
Category Id
Price
Weight
Unit
Product Info
Product Description
Features
Quantity
Updated At
Edit
Delete
Save
Cancel
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

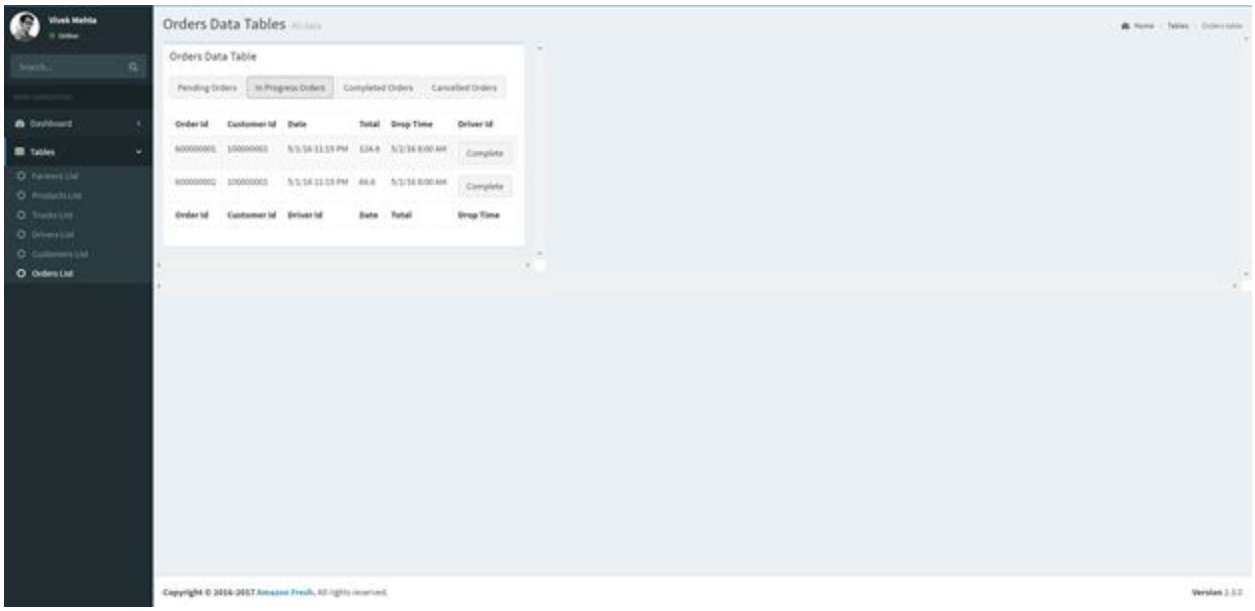

Vivek Mehta
Product Data Tables
Dashboard
Tables
Farmers List
Products List
Trucks List
Drivers List
Customers List
Orders List
Farmer Id
Farmer Name
Category Id
Price
Weight
Unit
Product Info
Product Description
Features
Quantity
Updated At
Edit
Delete
Save
Cancel
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

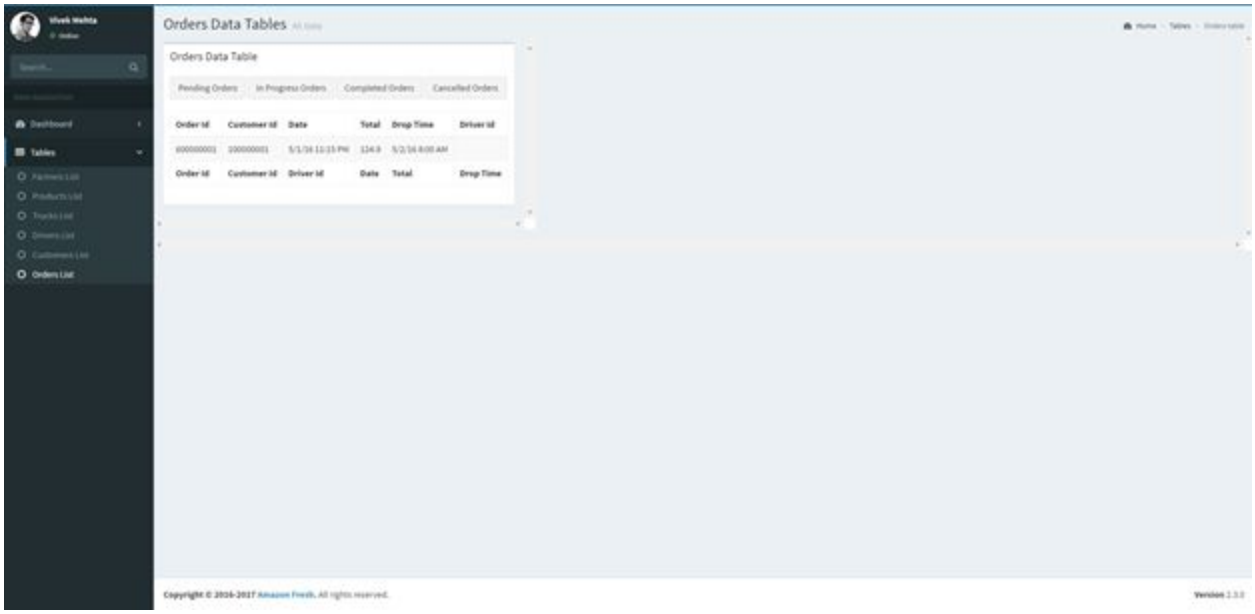

AdminLTE
Vivek Mehta
Trucks Data Tables
Trucks Data Table
Truck ID
Truck Number
Edit
Delete
Add Truck
Add Truck
Truck Number
122344
Invalid Truck Number!
Add Truck
Copyright © 2016-2017 Amazon Fresh, All rights reserved.

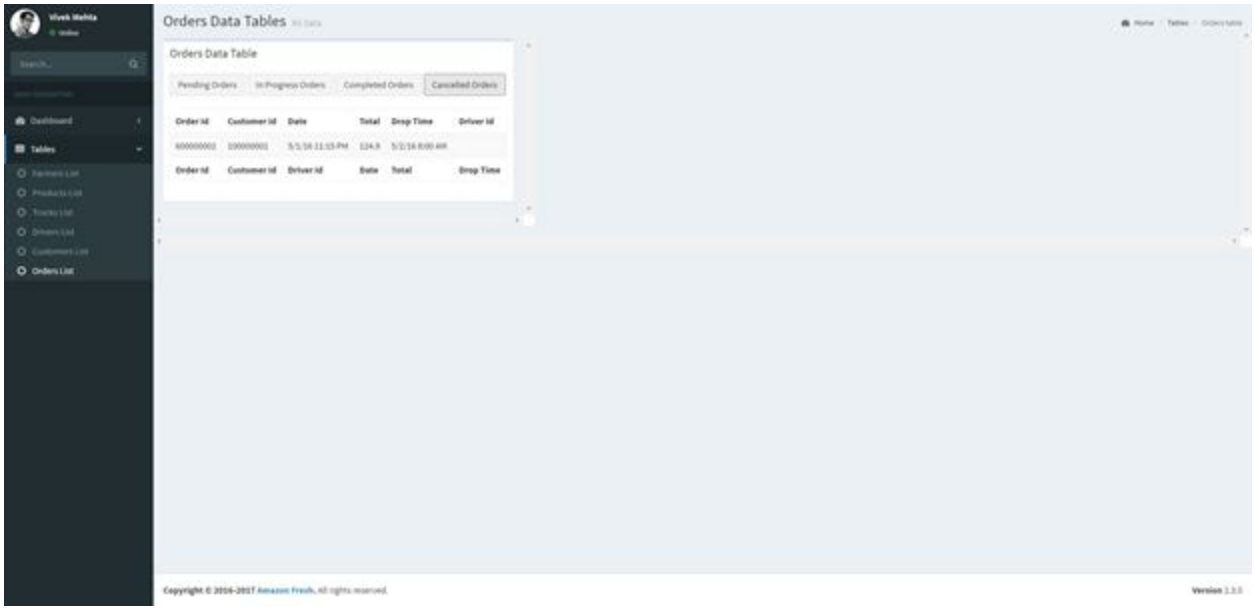

Vivek Mehta
Drivers Data Tables
Drivers Data Table
Driver Id
Truck Id
First Name
Last Name
Email
Address
City
State
ZIP
Contacts
Add Driver
Dashboard
Tables
Drivers List
Copyright © 2016-2017 Amazon Fresh, All rights reserved.

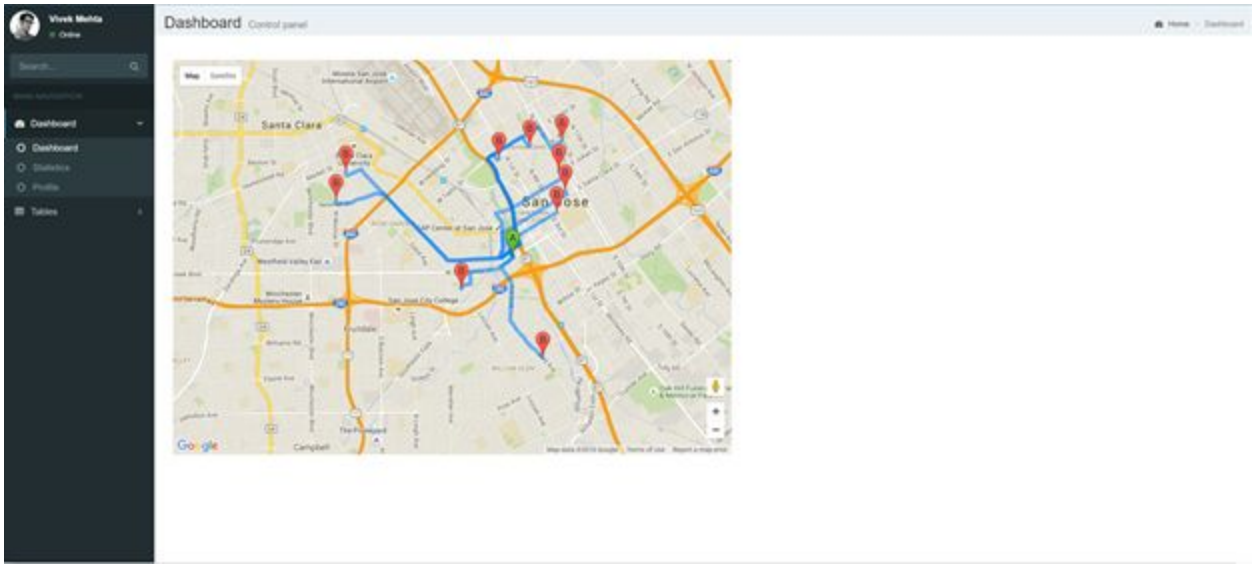

Vivek Mehta
Customers Data Tables
Customers Data Table
Customer Id
First Name
Last Name
Email
Address
City
State
ZIP
Contact
Add Customer
Dashboard
Tables
Customers List
Copyright © 2016-2017 Amazon Fresh, All rights reserved.


Vivek Mehta
Online
Search...
Dashboard
Tables
Farmers List
Products List
Trucks List
Drivers List
Customers List
Orders List
Orders Data Tables
Orders Data Table
Pending Orders
In Progress Orders
Completed Orders
Cancelled Orders
Order Id
Customer Id
Date
Total
Drop Time
Driver Id
Assign
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

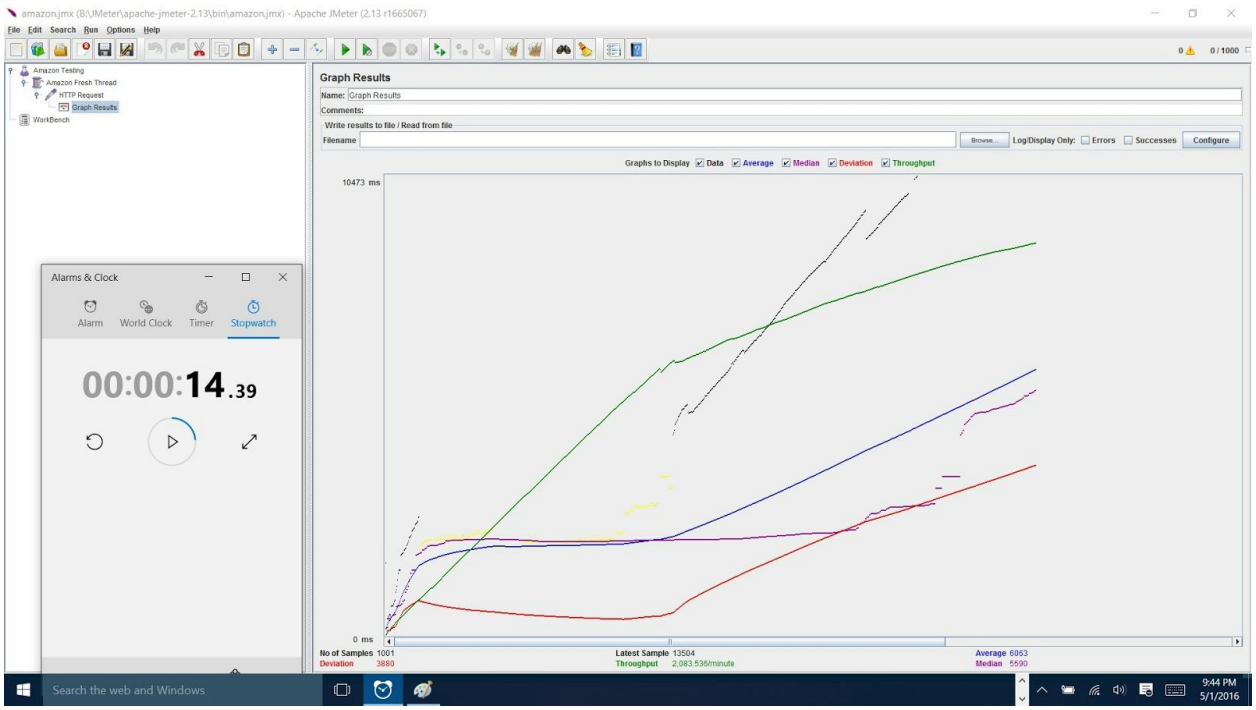

Vivek Mehta
Online
Search...
Dashboard
Tables
Farmers List
Products List
Trucks List
Drivers List
Customers List
Orders List
Orders Data Tables
Orders Data Table
Pending Orders
In Progress Orders
Completed Orders
Cancelled Orders
Order Id
Customer Id
Date
Total
Drop Time
Driver Id
Complete
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

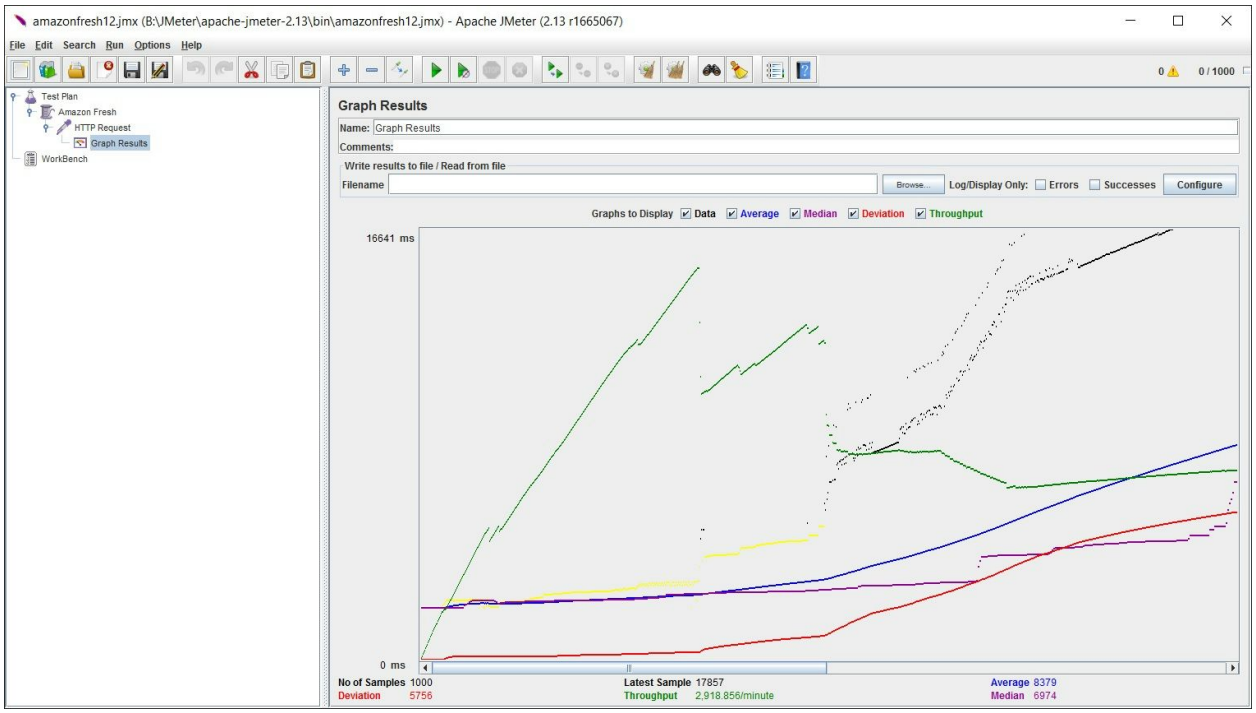

Vivek Mehta
Orders Data Tables
Orders Data Table
Pending Orders
In Progress Orders
Completed Orders
Cancelled Orders
Order Id
Customer Id
Date
Total
Drop Time
Driver Id
Dashboard
Tables
Orders List
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

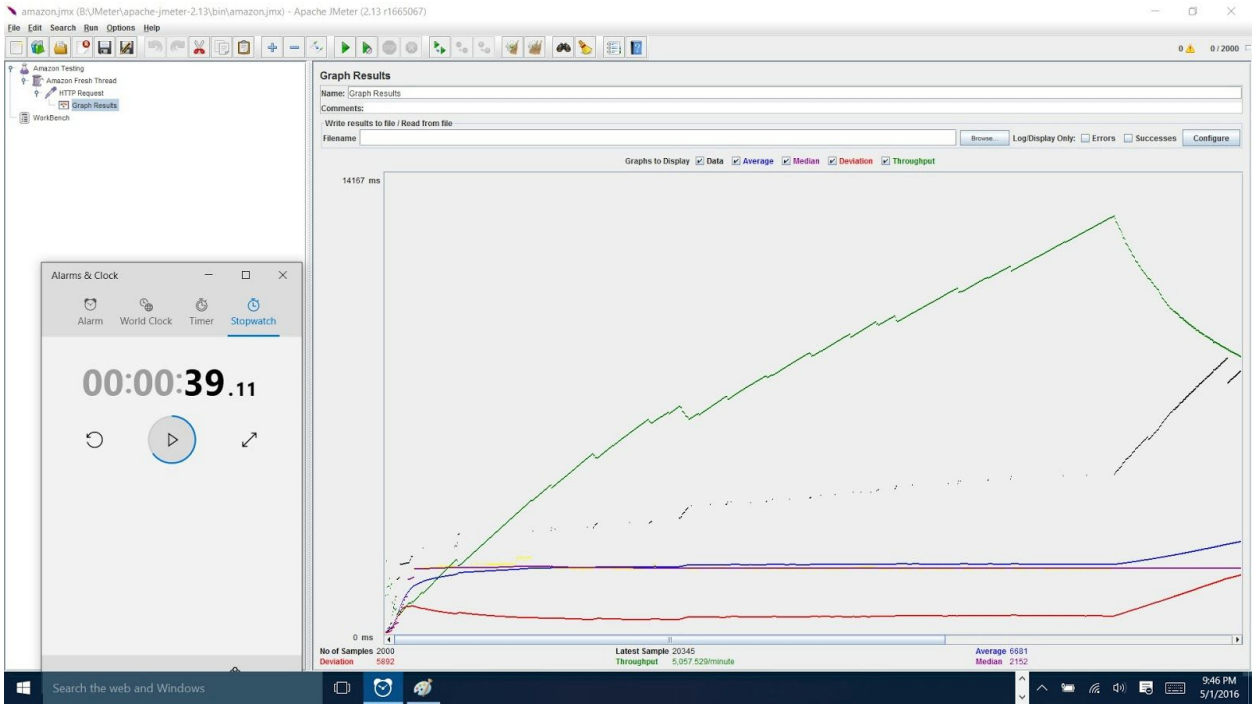

Vivek Mehta
Orders Data Tables
Orders Data Table
Pending Orders
In Progress Orders
Completed Orders
Cancelled Orders
Order Id
Customer Id
Date
Total
Drop Time
Driver Id
Dashboard
Tables
Orders List
Copyright © 2016-2017 Amazon Fresh, All rights reserved.
Version 2.3.0

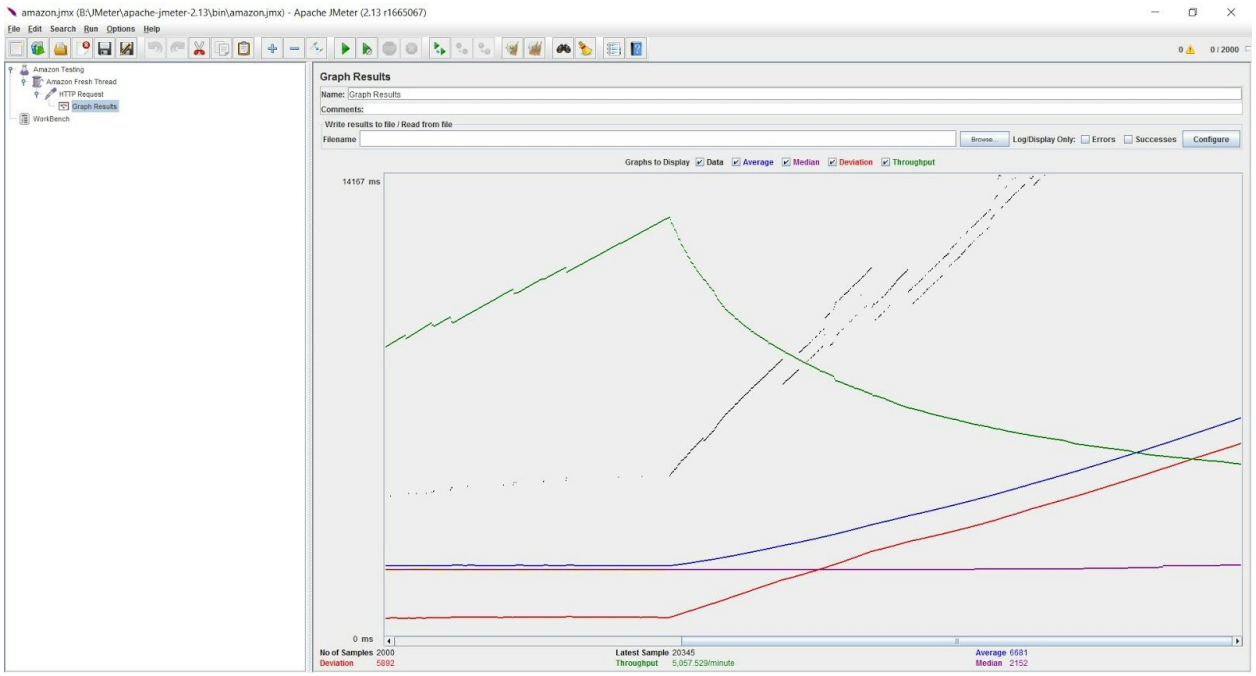

Vivek Mehta
Online
Dashboard
Control panel
Dashboard
Dashboard
Statistics
Profile
Tables
Santa Clara
San Jose
Google

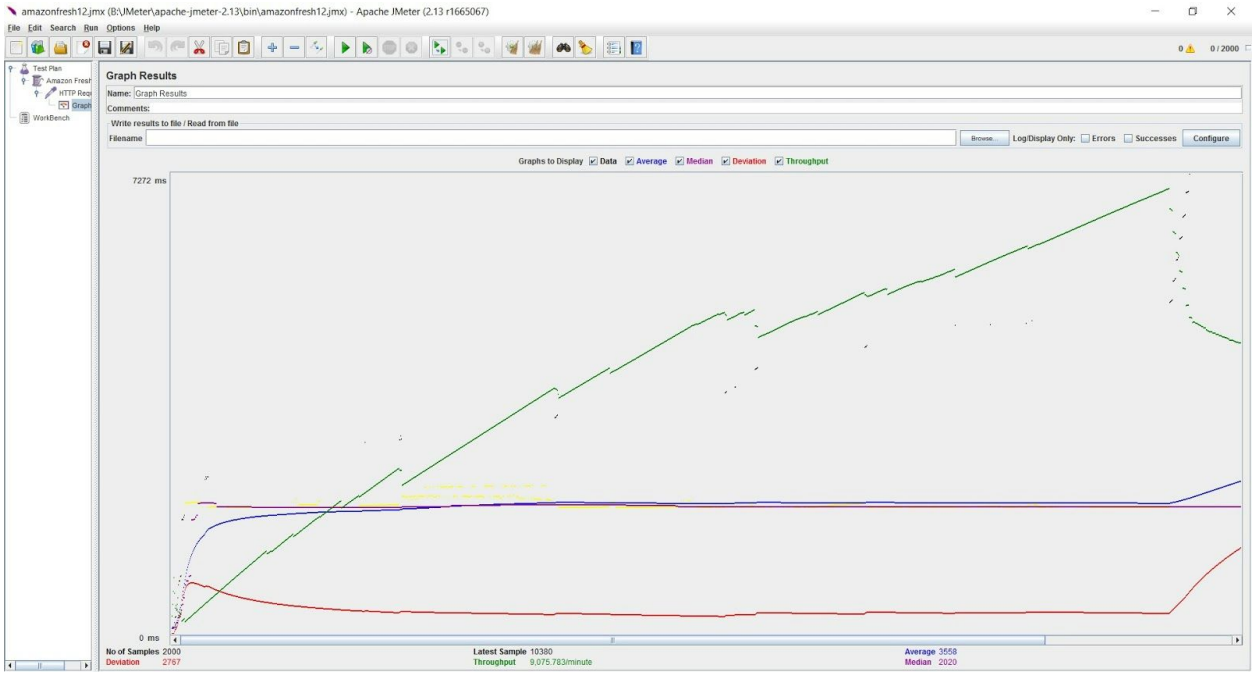
JMETER

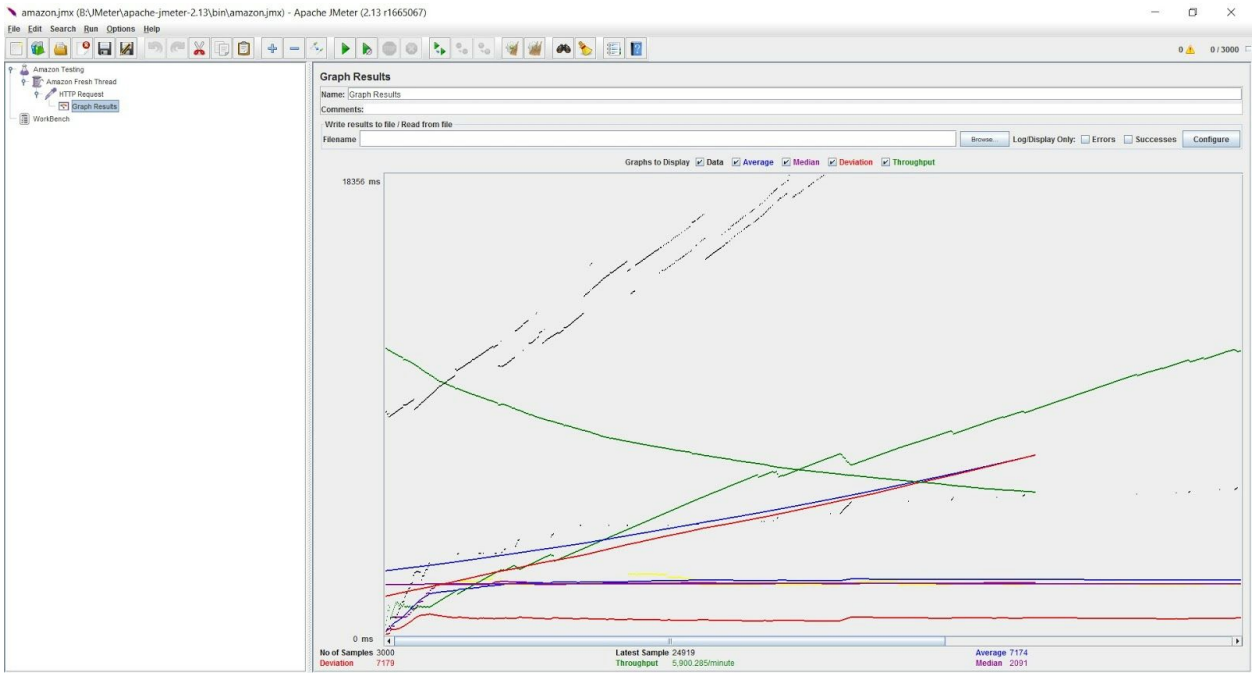

amazon.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazon.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
Amazon Testing
Amazon Fresh Thread
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
10473 ms
0 ms
No of Samples 1001
Latest Sample 13504
Average 6053
Deviation 3880
Throughput 2,083.536/minute
Median 5590
Alarms & Clock
Alarm
World Clock
Timer
Stopwatch
00:00:14.39
Search the web and Windows
9:44 PM
5/1/2016

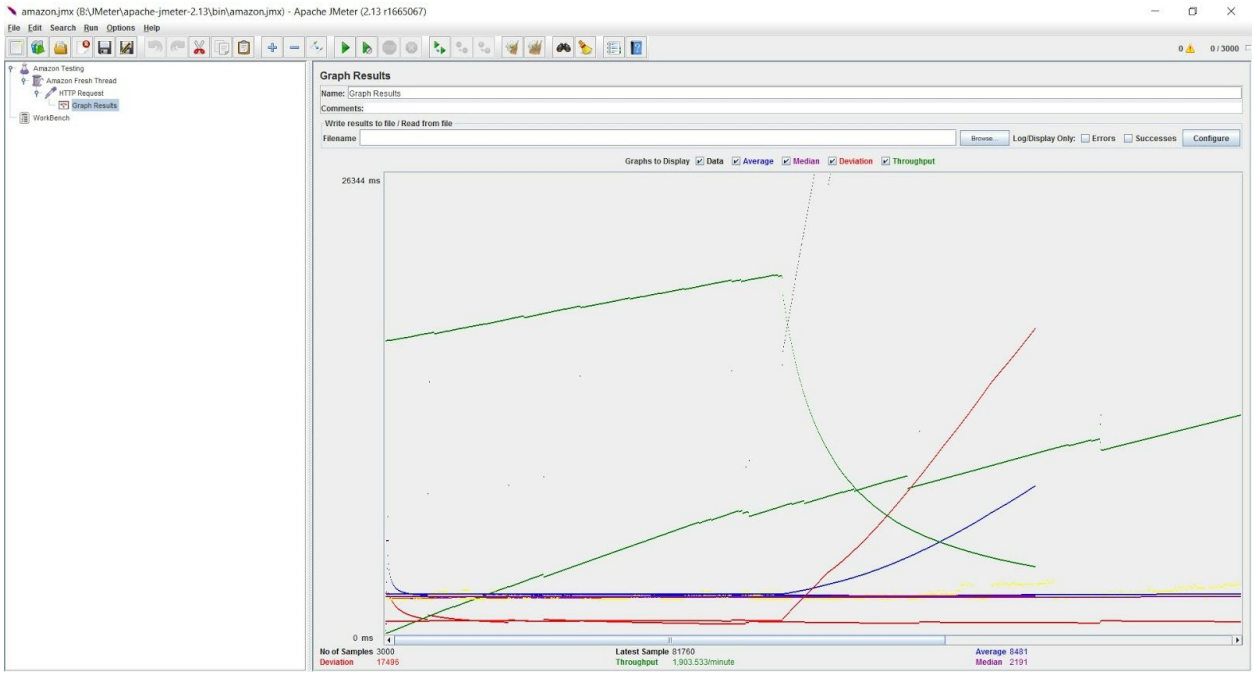

amazonfresh12.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazonfresh12.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 / 1000
Test Plan
Amazon Fresh
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
16641 ms
0 ms
No of Samples 1000
Latest Sample 17857
Average 8379
Deviation 5756
Throughput 2,918.856/minute
Median 6974

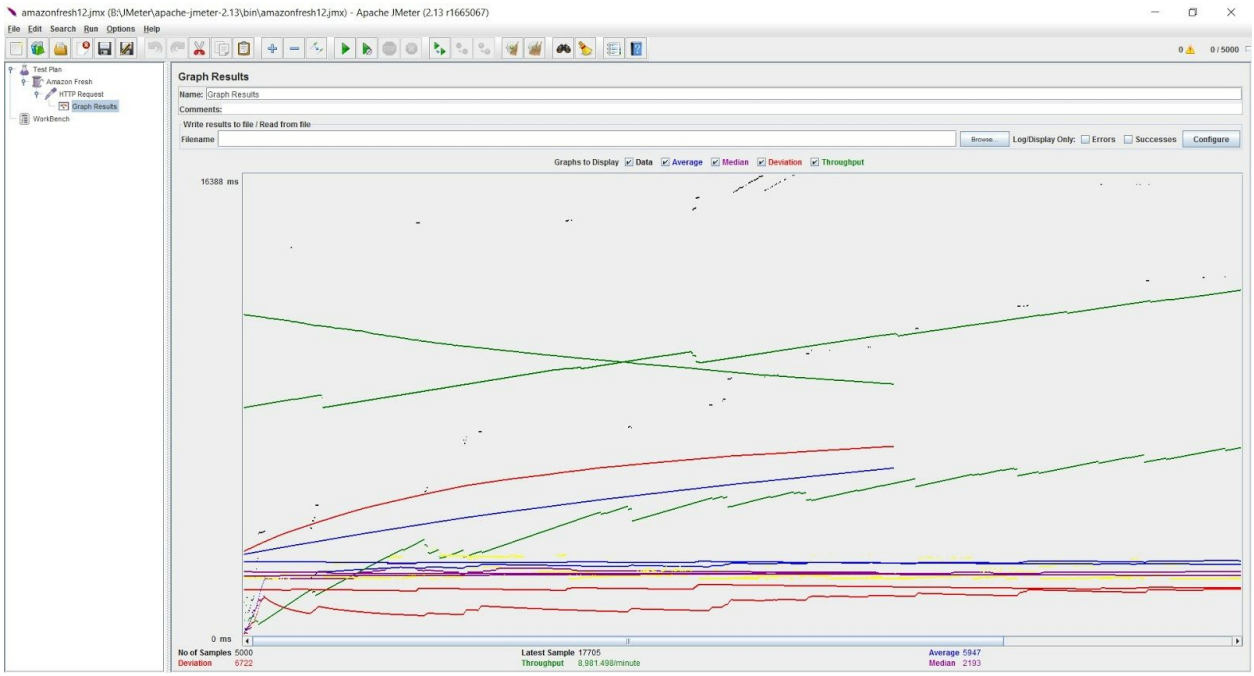

amazon.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazon.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
Amazon Testing
Amazon Fresh Thread
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
14167 ms
0 ms
No of Samples 2000
Deviation 5892
Latest Sample 20345
Throughput 5,057.529/minute
Average 6681
Median 2152
Alarms & Clock
Alarm
World Clock
Timer
Stopwatch
00:00:39.11
Search the web and Windows
9:46 PM
5/1/2016


amazon.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazon.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
Amazon Testing
Amazon Fresh Thread
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
14167 ms
0 ms
No of Samples 2000
Deviation 5892
Latest Sample 20345
Throughput 5,057.529/minute
Average 6681
Median 2152


amazonfresh12.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazonfresh12.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 / 2000
Test Plan
Amazon Fresh
HTTP Req
Graph
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
7272 ms
0 ms
No of Samples 2000
Deviation 2767
Latest Sample 10380
Throughput 9,075.783/minute
Average 3558
Median 2020


amazon.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazon.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 / 3000
Amazon Testing
Amazon Fresh Thread
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
18356 ms
0 ms
No of Samples 3000
Deviation 7179
Latest Sample 24919
Throughput 5,900.285/minute
Average 7174
Median 2091


amazon.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazon.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 / 3000
Amazon Testing
Amazon Fresh Thread
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
26344 ms
0 ms
No of Samples 3000
Latest Sample 81760
Average 8481
Deviation 17496
Throughput 1,903.533/minute
Median 2191


amazonfresh12.jmx (B:\JMeter\apache-jmeter-2.13\bin\amazonfresh12.jmx) - Apache JMeter (2.13 r1665067)
File Edit Search Run Options Help
0 / 5000
Test Plan
Amazon Fresh
HTTP Request
Graph Results
WorkBench
Graph Results
Name: Graph Results
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Configure
Graphs to Display Data Average Median Deviation Throughput
16388 ms
0 ms
No of Samples 5000
Latest Sample 17705
Average 5947
Deviation 6722
Throughput 8,981.498/minute
Median 2193

Server Listing

```
/**
 * Module dependencies.
 */

var express = require('express')
  , routes = require('./routes')
  , user = require('./routes/user')
  , http = require('http')
  , path = require('path')
  , farmer = require('./routes/farmer')
  , product = require('./routes/product')
  , login = require('./routes/login')
  , cart = require('./routes/cart')
  , order = require('./routes/order')
  , truck = require('./routes/truck')
  , driver =  require('./routes/driver')
  ,farmerLogin = require('./routes/farmerLogin')
  //ADMIN
  , admin = require('./routes/admin');

//JUST FOR PASSPORT LOGIN
var passport = require('passport');
require('./routes/passport')(passport);


var mongoose = require('mongoose');
mongoose.connect("mongodb://localhost/amazon");

var mongoURL = "mongodb://localhost:27017/amazon";
var expressSession = require("express-session");
var mongoStore = require("connect-mongo")(expressSession);

var app = express();

// all environments
```

```
// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.favicon());
app.use(express.logger('dev'));
app.use(express.bodyParser({ keepExtensions: true, uploadDir: __dirname + '/public/img', limit: '3mb'}));
app.use(express.methodOverride());
//app.use('/uploads', express.static(path.join(__dirname,'uploads')));
//app.use(multer({dest: './uploads/'}))

//EXPRESS SESSION CONFIG
app.use(expressSession({
  key: 'amazon_cookie',
    secret: 'amazon',
    resave: false,
    saveUninitialized: false,
    cookie: {},
    store: new mongoStore({
    url: mongoURL
  })
}));
app.use(passport.initialize());
// app.use(passport.session());


app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

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

```
//GET REQUEST
app.get('/', routes.index);
app.get('/users', user.list);
app.get('/PreviewOrder', isAuthenticated, order.home);


//ADMIN API
app.get('/admin/home',admin.home);
app.get('/admin/login',admin.login);
app.get('/admin/logout', admin.logout);
app.post('/admin/checkLogin', admin.checkLogin);
app.post('/admin/profile', admin.profile);
app.get('/admin/farmers/list',admin.farmersList);
app.get('/admin/products/list',admin.productsList);
app.get('/admin/trucks/list',admin.trucksList);
app.get('/admin/drivers/list',admin.driversList);
app.get('/admin/customers/list',admin.customersList);
app.get('/admin/orders/list',admin.ordersList);
//app.post('/admin/addFarmer', admin.addFarmer);;

//TRUCK API
app.post('/truck/create', truck.createTruck);
app.get('/truck/all', truck.getTrucks);
app.post('/truck/edit', truck.editTruck);
app.delete('/truck/delete',truck.deleteTruck);

//DRIVER API
app.post('/driver/create', driver.createDriver);
app.get('/driver/all', driver.getDrivers);
app.post('/driver/edit', driver.editDriver);
app.delete('/driver/delete',driver.deleteDriver);

//CUSTOMER API
app.post('/customer/create', user.createCustomer);
```

```
//CUSTOMER API
app.post('/customer/create', user.createCustomer);
app.get('/customer/all', user.getCustomers);
app.post('/customer/edit', user.editCustomer);
app.delete('/customer/delete',user.deleteCustomer);
app.get('/farmer/products/list', function(req,res){
  if(req.session.farmer) {
    res.render('./farmer/productlist', {
      email : req.session.farmer.email,
      fname : req.session.farmer.fname,
      lname : req.session.farmer.lname,
      createdAt : req.session.farmer.createdAt
    });
  }
  else{
    res.redirect('/farmer/login');
  }
});
app.get('/farmer/product/all',farmerLogin.productlist);


//ORDER API
app.post('/order/pending', order.getPending);
app.post('/order/inprogress', order.getInProgress);
app.post('/order/complete', order.getComplete);
app.post('/order/cancel', order.getCancel);
app.post('/order/assignDriverId', order.assignDriverId);
app.post('/order/assignComplete', order.assignComplete);

app.post('/farmer/login', function(req, res, next) {
  passport.authenticate('farmerLogin', function(err, farmer, info) {
    if(err) {
      console.log(err);
      return next(err);
```

```
app.post('/farmer/login', function(req, res, next) {
  passport.authenticate('farmerLogin', function(err, farmer, info) {
    if(err) {
      console.log(err);
      return next(err);
    }

    if(!farmer) {
      req.session.wrongSignIn = true;
      console.log("login failed");
      return res.redirect('/farmer/login');
    }
    else{
      req.logIn(farmer, {session: false}, function(err) {
        if(err) {
          return next(err);
        }
        console.log("login success");
        req.session.farmer = farmer;
        res.send({"status": 200, "data": true});
      })
    }
  })(req, res, next);
});

app.post('/farmer/signup',farmerLogin.signup);
app.get('/farmer/signup',farmerLogin.userSignUp);
app.get('/farmer/login',farmerLogin.userSignIn);
app.get('/farmer/checkEmail',farmerLogin.checkEmail);
app.get('/farmer/home',farmerLogin.home);
app.get('/farmer/product/all', function(req,res){ res.render('/farmer/productlist'); });
app.get('/farmer/order/pending', function(req,res){ res.render('/farmer/pendinglist'); });
app.get('/farmer/order/complete', function(req,res){ res.render('/farmer/completelist'); });
app.get('/farmer/all',farmer.getFarmers);
```

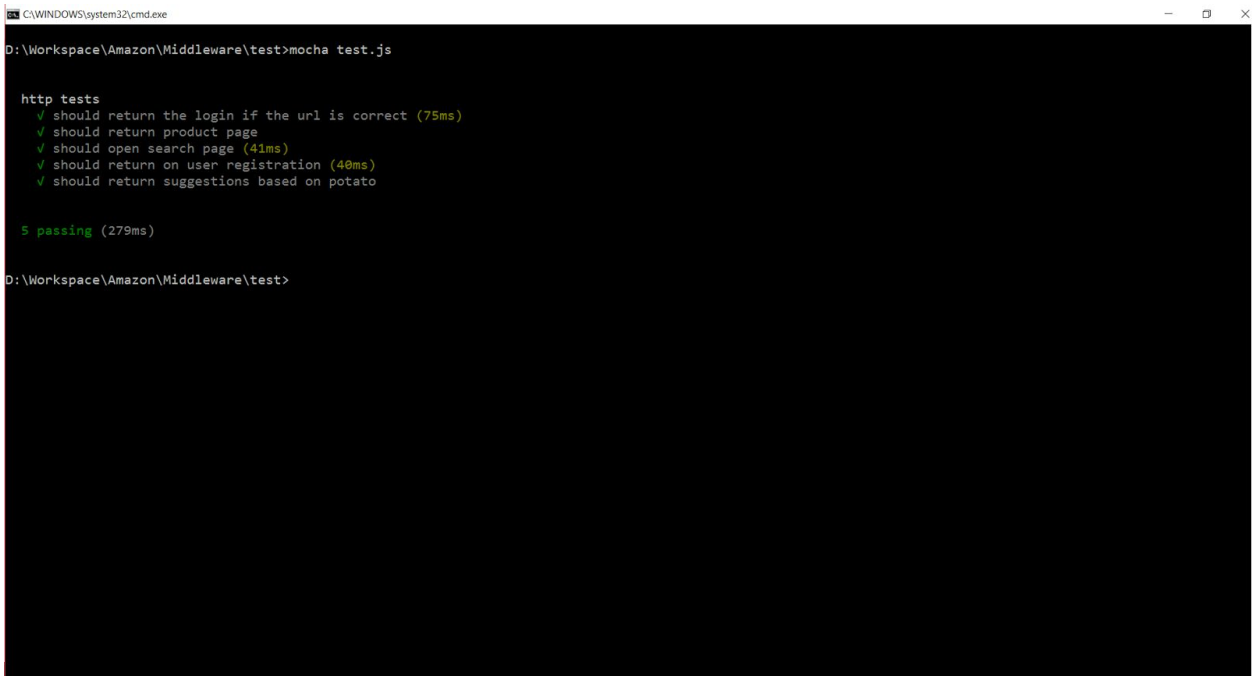
```
app.get('/farmer/product/all', function(req,res){ res.render('/farmer/productlist'); });
app.get('/farmer/order/pending', function(req,res){ res.render('/farmer/pendinglist'); });
app.get('/farmer/order/complete', function(req,res){ res.render('/farmer/completelist'); });
app.get('/farmer/all',farmer.getFarmers);
app.post('/farmer/create',farmer.createFarmer);
app.delete('/farmer/delete',farmer.deleteFarmer);
app.post('/farmer/edit',farmer.editFarmer);
app.get('/farmer/get', function(req,res){ console.log(req.session.farmer); res.send({"status":200,"data":req.session.farmer});
app.post('/farmer/product/create',farmerLogin.createProduct);
app.post('/farmer/product/delete',farmerLogin.deleteProduct);
app.post('/farmer/product/edit',farmerLogin.editProduct);

app.post('/user/address/update',user.editAddress);
app.post('/user/card/update',user.editCard);
app.get('/user/address',user.getAddress);
app.get('/user/orders',order.orderDetails);
//app.get('/user/orders',user.getOrders);


app.get('/product/all',product.getProducts);
app.post('/product/create',product.createProduct);

app.post('/fileUpload', product.fileUpload);

app.delete('/product/delete',product.deleteProduct);
app.post('/product/edit',product.editProduct);


app.get('/category/get', product.getCategory);


//app.get('/prod_details', user.prod_details);
app.get('/myReviews', product.myReviews);
app.get('/search', product.prod_search);
app.get('/product', product.prod_details);
```

```
//app.get('/prod_details', user.prod_details);
app.get('/myReviews', product.myReviews);
app.get('/search', product.prod_search);
app.get('/product', product.prod_details);
app.post('/create_review',product.create_review);
app.post('/f_create_review',product.f_create_review);
app.get('/farmer_page',product.farmer_page);

app.get('/frame', function(req,res){
  res.render('frame');
})
app.get('/login', login.signIn);
app.get('/signup', login.signUp);
app.get('/logout', function(req,res) {
  req.session.destroy(function(err){
    res.redirect('/');
  })
});



app.get('/myOrders', function(req, res){

  if(typeof req.session.user != 'undefined'){
    console.log(req.session.user);
    res.render('myOrders', { user: req.session.user });
  }else{
    res.render('index');
  }
});

// app.get('/orderDetails', function(req, res){

//   if(typeof req.session.user != 'undefined'){
```

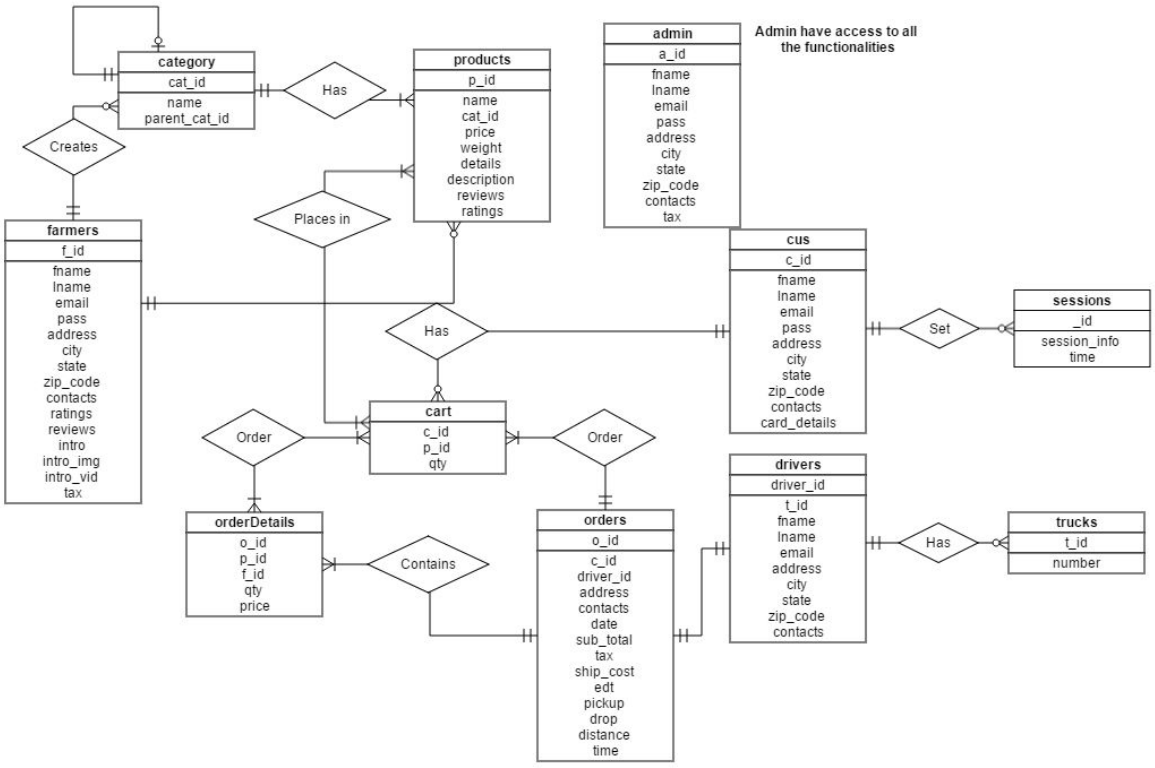
```
app.get('/customerAccount', function(req, res){

  if(typeof req.session.user != 'undefined'){
    console.log(req.session.user);
    res.render('customerAccount', { user: req.session.user });
  }else{
    res.render('index');
  }
});

app.get('/help', function(req, res){

  if(typeof req.session.user != 'undefined'){
    console.log(req.session.user);
    res.render('help', { user: req.session.user });
  }else{
    res.render('index');
  }
});

app.get('/addressDetails', function(req, res){
  if(typeof req.session.user !== 'undefined'){
    console.log(req.session.user);
    res.render('addressDetails', { user: req.session.user });
  }else{
    res.render('index');
  }
  });

app.get('/conditions', function(req, res){
  if(typeof req.session.user !== 'undefined'){
    console.log(req.session.user);
    res.render('conditions', { user: req.session.user });
  }else{
```

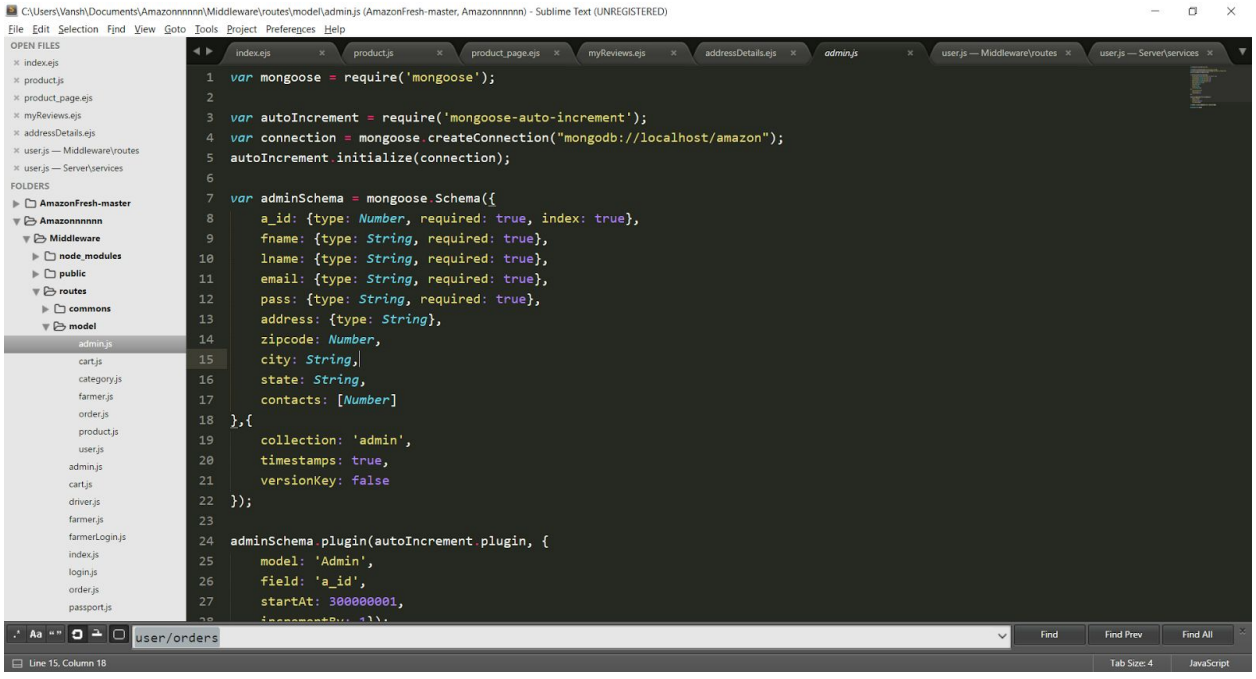
```
//POST REQUEST
app.post('/login', function(req, res, next) {
  passport.authenticate('login', function(err, user, info) {
    if(err) {
      return next(err);
    }

    if(!user) {
      req.session.wrongSignIn = true;
      res.redirect('/login');
    }
    else{
      req.logIn(user, {session: false}, function(err) {
        if(err) {
          return next(err);
        }
        req.session.user = user;
        return res.redirect('/');
      })
    }
  })(req, res, next);
});


app.post('/reg', login.regUser);
app.post('/additem', cart.addItem);
app.post('/cart', cart.cartItems);
app.post('/suggest', product.suggest);
app.post('/order', order.createOrder);

http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});
```

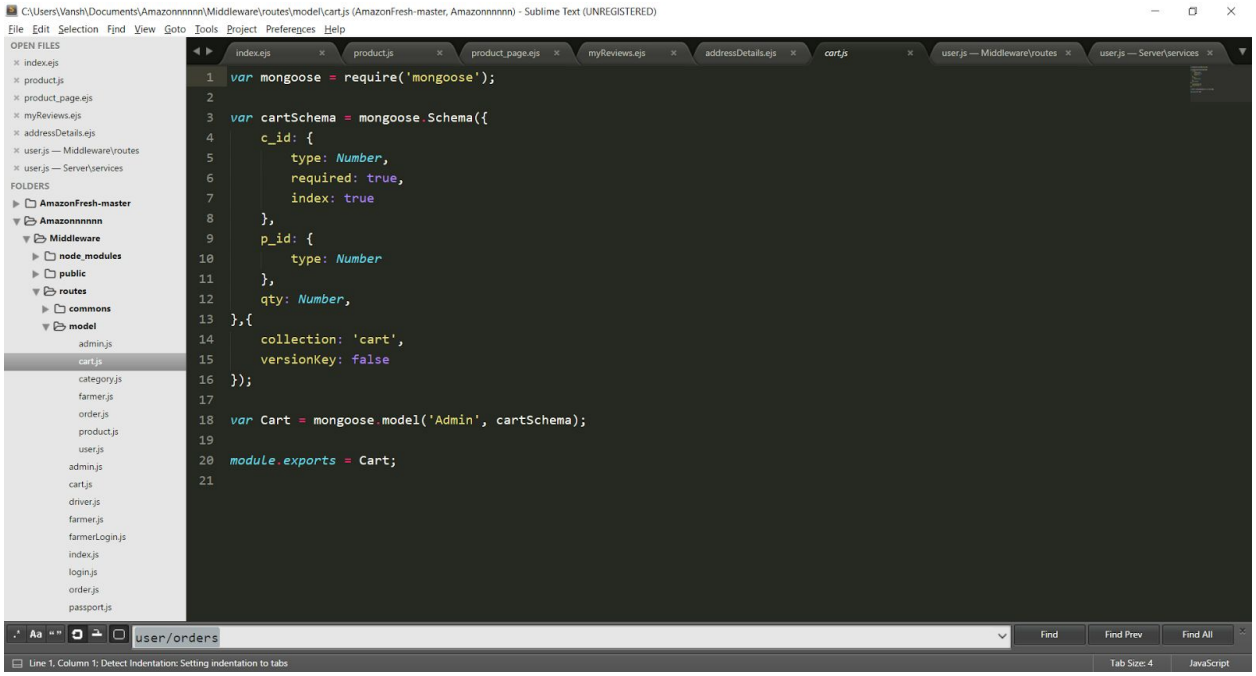
Mocha Test

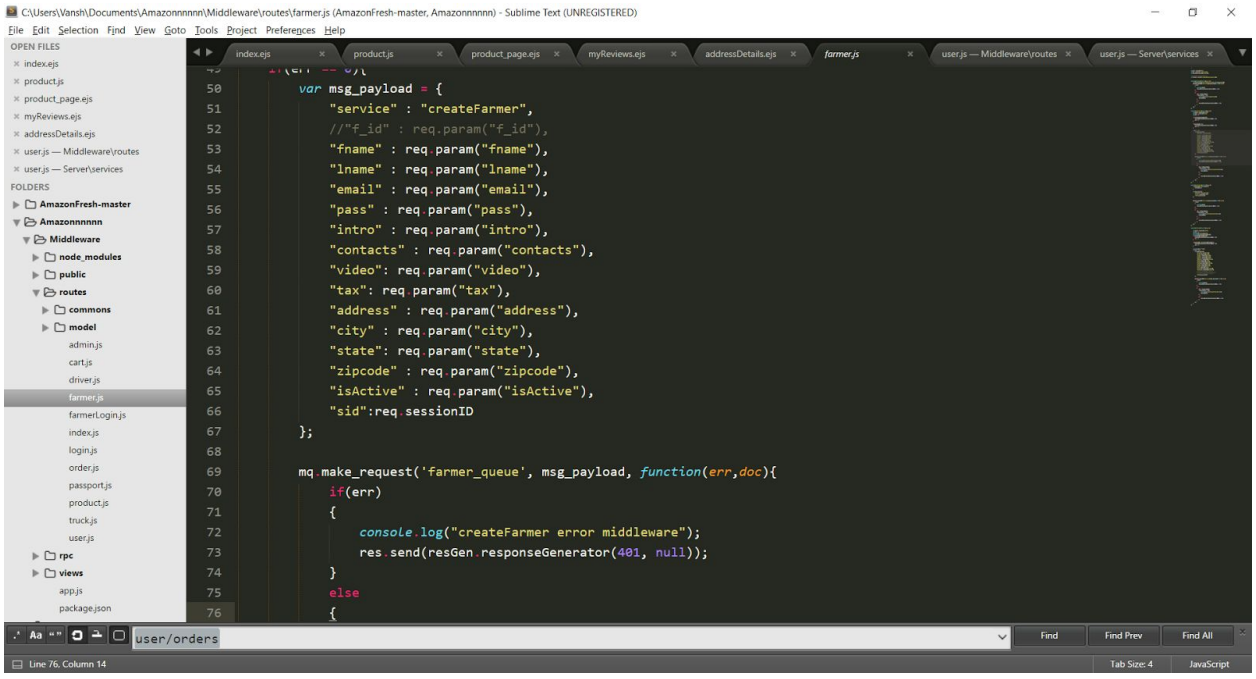


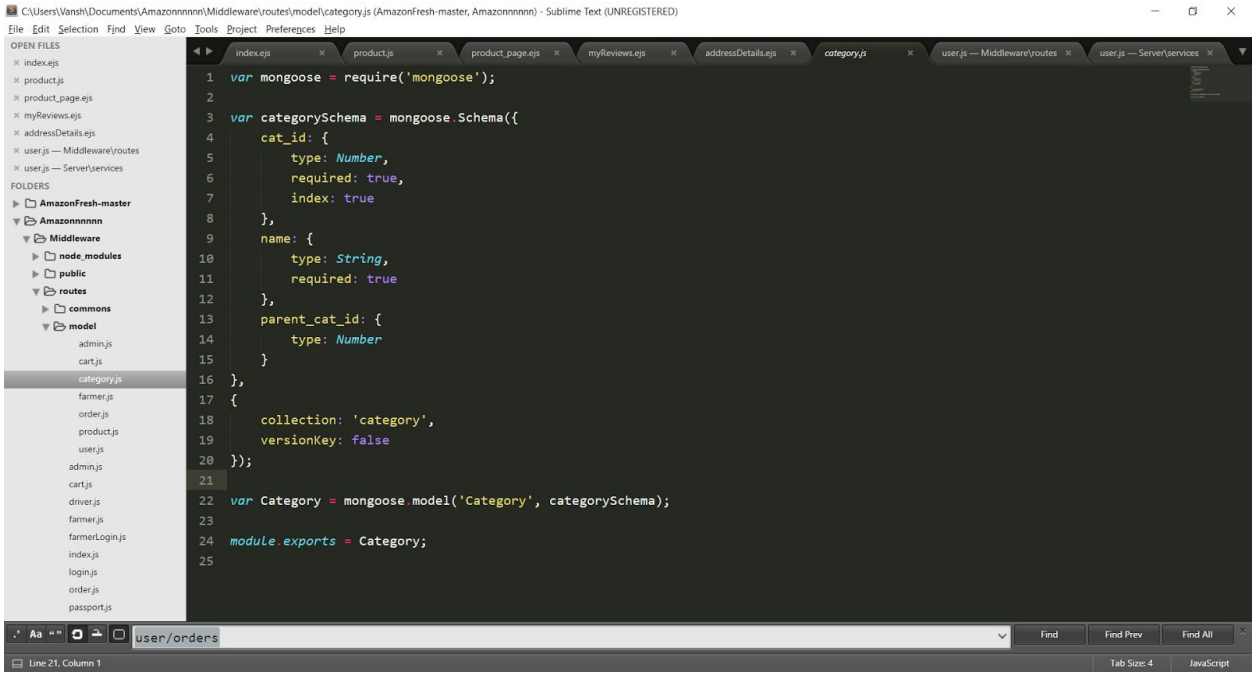
Database Schema

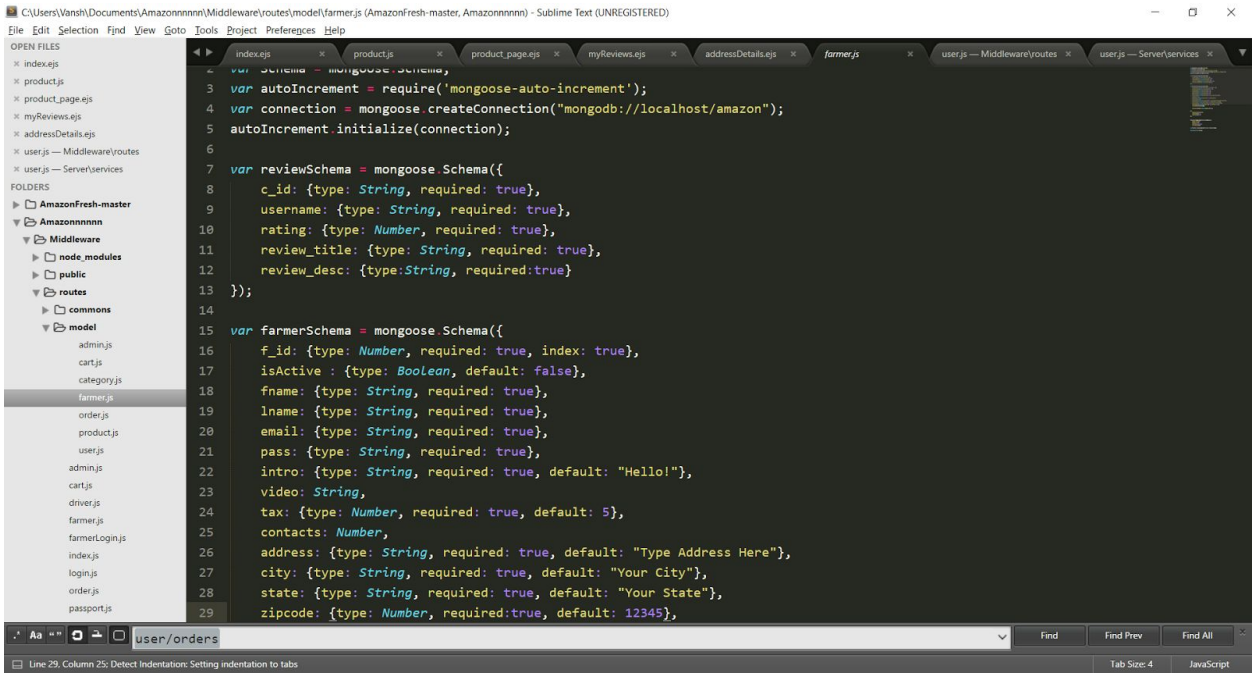

category
cat_id
name
parent_cat_id
Has
products
p_id
name
cat_id
price
weight
details
description
reviews
ratings
admin
a_id
fname
lname
email
pass
address
city
state
zip_code
contacts
tax
Admin have access to all the functionalities
Creates
Places in
farmers
f_id
fname
lname
email
pass
address
city
state
zip_code
contacts
ratings
reviews
intro
intro_img
intro_vid
tax
cus
c_id
fname
lname
email
pass
address
city
state
zip_code
contacts
card_details
Has
Set
sessions
_id
session_info
time
cart
c_id
p_id
qty
Order
Order
orderDetails
o_id
p_id
f_id
qty
price
Contains
orders
o_id
c_id
driver_id
address
contacts
date
sub_total
tax
ship_cost
edt
pickup
drop
distance
time
drivers
driver_id
t_id
fname
lname
email
address
city
state
zip_code
contacts
Has
trucks
t_id
number

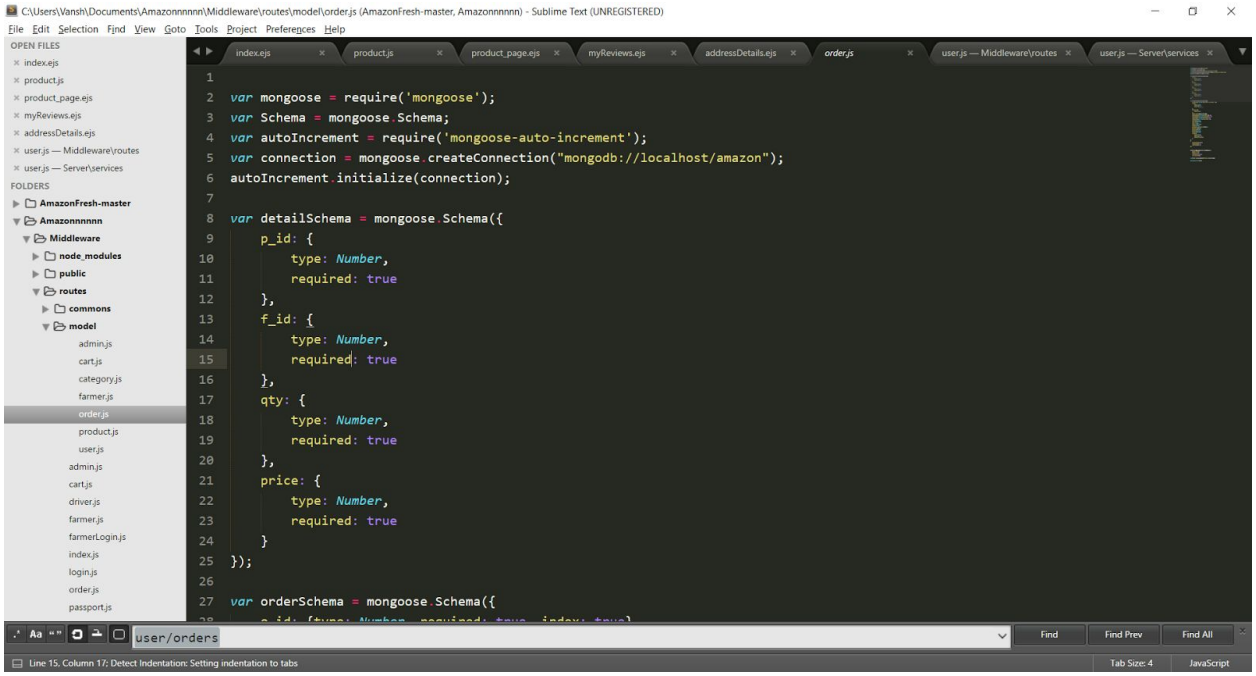

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\admin.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
admin.js
user.js — Middleware\routes
user.js — Server\services
var mongoose = require('mongoose');
var autoIncrement = require('mongoose-auto-increment');
var connection = mongoose.createConnection("mongodb://localhost/amazon");
autoIncrement.initialize(connection);
var adminSchema = mongoose.Schema({
a_id: {type: Number, required: true, index: true},
fname: {type: String, required: true},
lname: {type: String, required: true},
email: {type: String, required: true},
pass: {type: String, required: true},
address: {type: String},
zipcode: Number,
city: String,
state: String,
contacts: [Number]
},{
collection: 'admin',
timestamps: true,
versionKey: false
});
adminSchema.plugin(autoIncrement.plugin, {
model: 'Admin',
field: 'a_id',
startAt: 300000001,
user/orders
Find
Find Prev
Find All
Line 15, Column 18
Tab Size: 4
JavaScript

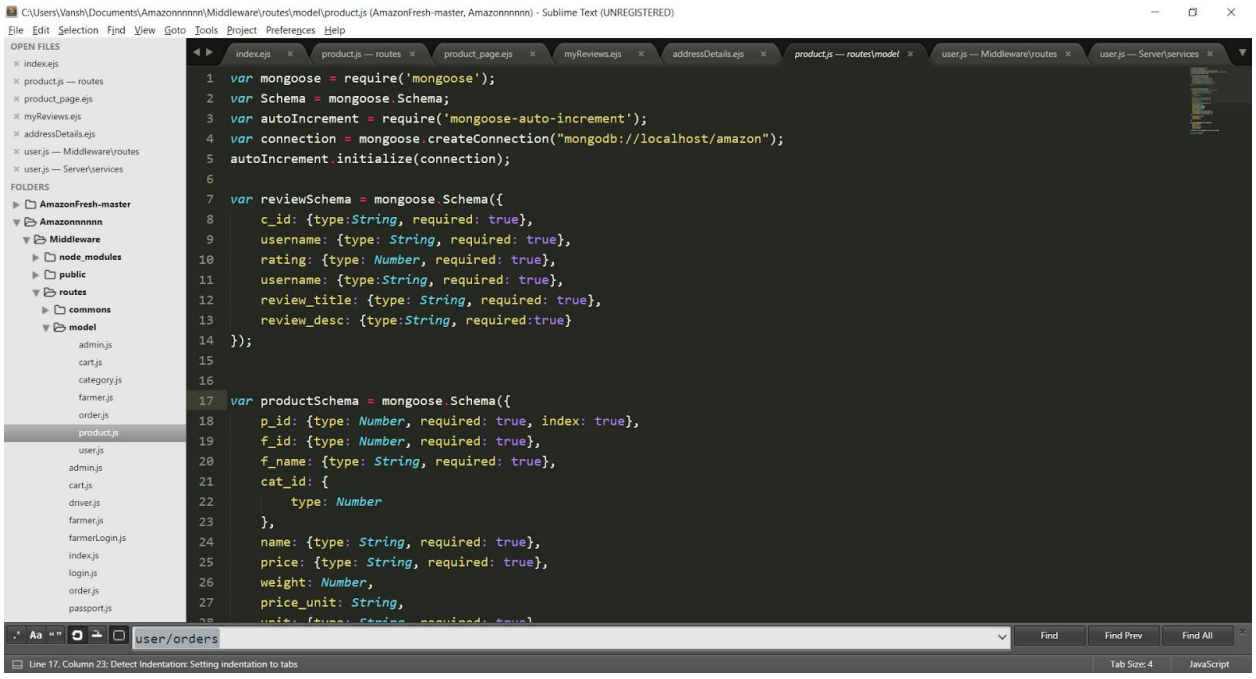

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\cart.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
cart.js
user.js — Middleware\routes
user.js — Server\services
1 var mongoose = require('mongoose');
2
3 var cartSchema = mongoose.Schema({
4 c_id: {
5 type: Number,
6 required: true,
7 index: true
8 },
9 p_id: {
10 type: Number
11 },
12 qty: Number,
13 },{
14 collection: 'cart',
15 versionKey: false
16 });
17
18 var Cart = mongoose.model('Admin', cartSchema);
19
20 module.exports = Cart;
21
user/orders
Find
Find Prev
Find All
Line 1, Column 1; Detect Indentation: Setting indentation to tabs
Tab Size: 4
JavaScript

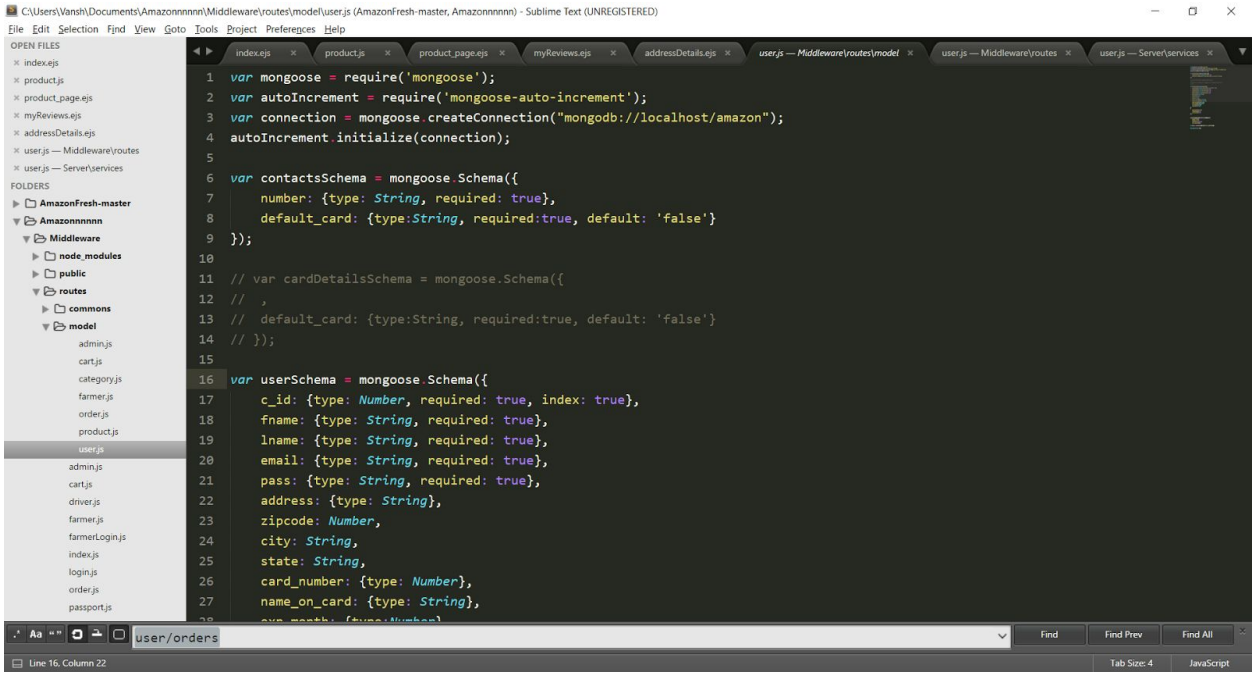

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\farmer.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
product.js
truck.js
user.js
rpc
views
app.js
package.json
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
farmer.js
user.js — Middleware\routes
user.js — Server\services
50 var msg_payload = {
51 "service" : "createFarmer",
52 //"f_id" : req.param("f_id"),
53 "fname" : req.param("fname"),
54 "lname" : req.param("lname"),
55 "email" : req.param("email"),
56 "pass" : req.param("pass"),
57 "intro" : req.param("intro"),
58 "contacts" : req.param("contacts"),
59 "video": req.param("video"),
60 "tax": req.param("tax"),
61 "address" : req.param("address"),
62 "city" : req.param("city"),
63 "state": req.param("state"),
64 "zipcode" : req.param("zipcode"),
65 "isActive" : req.param("isActive"),
66 "sid":req.sessionID
67 };
68
69 mq.make_request('farmer_queue', msg_payload, function(err,doc){
70 if(err)
71 {
72 console.log("createFarmer error middleware");
73 res.send(resGen.responseGenerator(401, null));
74 }
75 else
76 {
user/orders
Find
Find Prev
Find All
Line 76, Column 14
Tab Size: 4
JavaScript

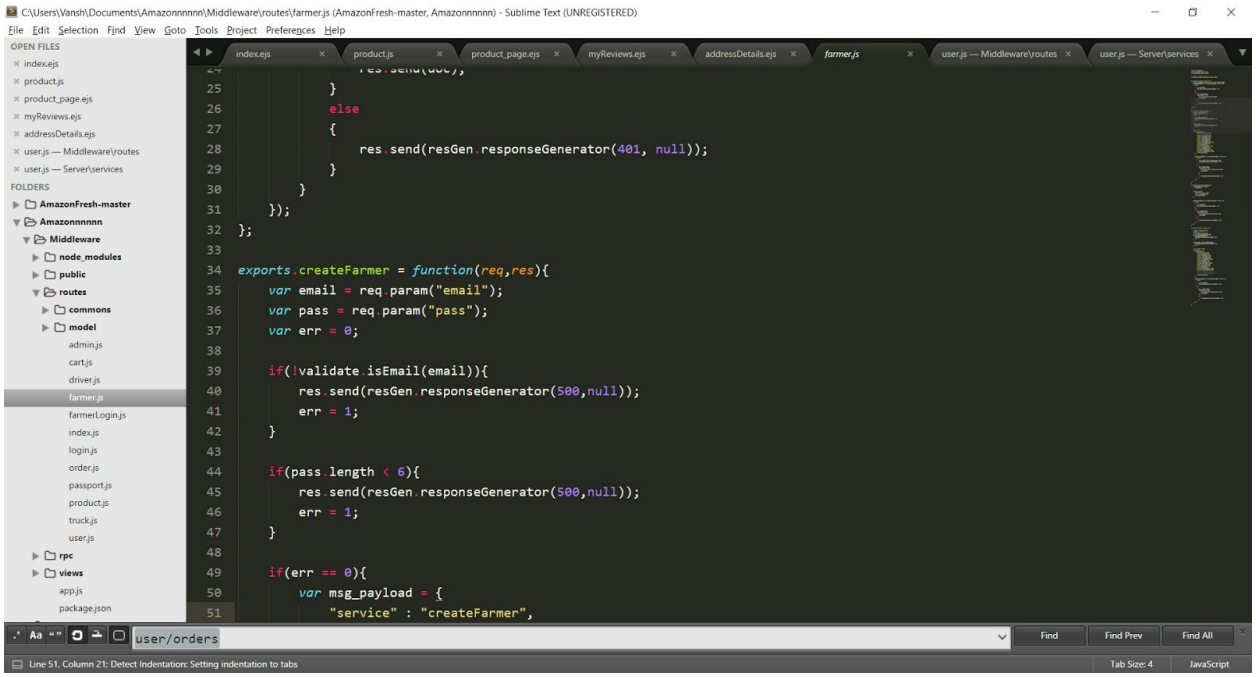

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\category.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
category.js
user.js — Middleware\routes
user.js — Server\services
var mongoose = require('mongoose');
var categorySchema = mongoose.Schema({
cat_id: {
type: Number,
required: true,
index: true
},
name: {
type: String,
required: true
},
parent_cat_id: {
type: Number
}
},
{
collection: 'category',
versionKey: false
});
var Category = mongoose.model('Category', categorySchema);
module.exports = Category;
user/orders
Find
Find Prev
Find All
Line 21, Column 1
Tab Size: 4
JavaScript


C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\farmer.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
farmer.js
user.js — Middleware\routes
user.js — Server\services
var autoIncrement = require('mongoose-auto-increment');
var connection = mongoose.createConnection("mongodb://localhost/amazon");
autoIncrement.initialize(connection);
var reviewSchema = mongoose.Schema({
c_id: {type: String, required: true},
username: {type: String, required: true},
rating: {type: Number, required: true},
review_title: {type: String, required: true},
review_desc: {type:String, required:true}
});
var farmerSchema = mongoose.Schema({
f_id: {type: Number, required: true, index: true},
isActive : {type: Boolean, default: false},
fname: {type: String, required: true},
lname: {type: String, required: true},
email: {type: String, required: true},
pass: {type: String, required: true},
intro: {type: String, required: true, default: "Hello!"},
video: String,
tax: {type: Number, required: true, default: 5},
contacts: Number,
address: {type: String, required: true, default: "Type Address Here"},
city: {type: String, required: true, default: "Your City"},
state: {type: String, required: true, default: "Your State"},
zipcode: {type: Number, required:true, default: 12345},
user/orders
Find
Find Prev
Find All
Line 29, Column 25; Detect Indentation: Setting indentation to tabs
Tab Size: 4
JavaScript

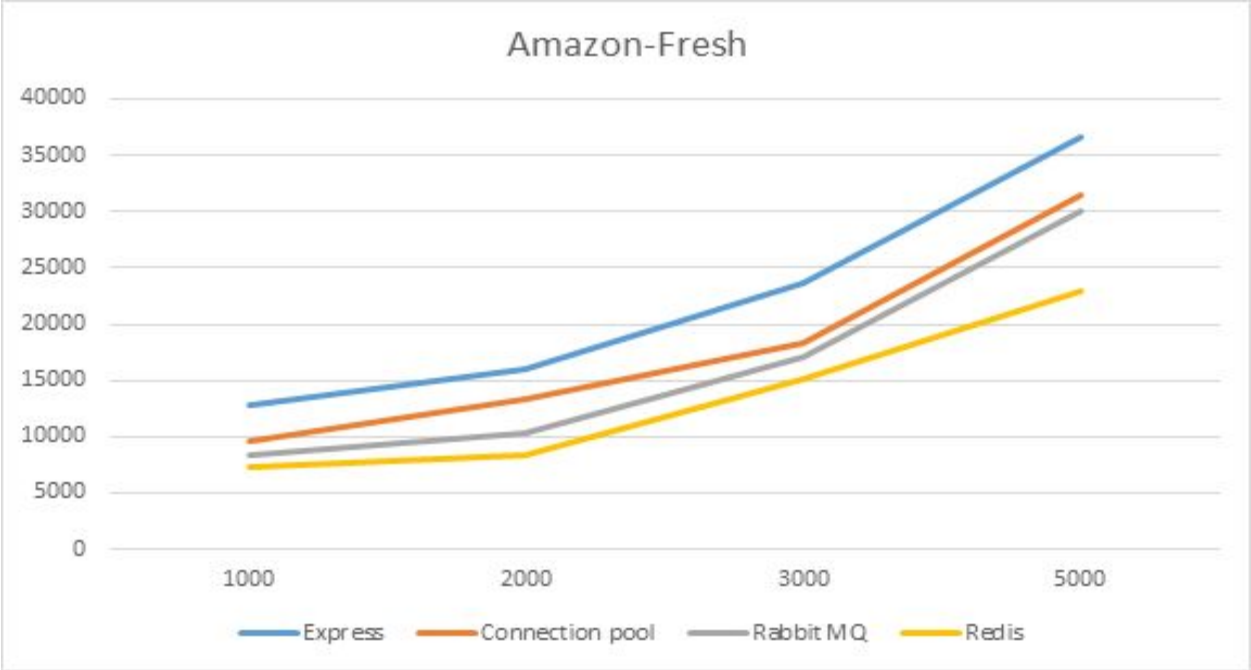

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\order.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
order.js
user.js — Middleware\routes
user.js — Server\services
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var autoIncrement = require('mongoose-auto-increment');
var connection = mongoose.createConnection("mongodb://localhost/amazon");
autoIncrement.initialize(connection);
var detailSchema = mongoose.Schema({
p_id: {
type: Number,
required: true
},
f_id: {
type: Number,
required: true
},
qty: {
type: Number,
required: true
},
price: {
type: Number,
required: true
}
});
var orderSchema = mongoose.Schema({
user/orders
Find
Find Prev
Find All
Line 15, Column 17; Detect Indentation: Setting indentation to tabs
Tab Size: 4
JavaScript


C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\product.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js — routes
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js — routes
product_page.ejs
myReviews.ejs
addressDetails.ejs
product.js — routes\model
user.js — Middleware\routes
user.js — Server\services
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var autoIncrement = require('mongoose-auto-increment');
var connection = mongoose.createConnection("mongodb://localhost/amazon");
autoIncrement.initialize(connection);
var reviewSchema = mongoose.Schema({
c_id: {type:String, required: true},
username: {type: String, required: true},
rating: {type: Number, required: true},
username: {type:String, required: true},
review_title: {type: String, required: true},
review_desc: {type:String, required:true}
});
var productSchema = mongoose.Schema({
p_id: {type: Number, required: true, index: true},
f_id: {type: Number, required: true},
f_name: {type: String, required: true},
cat_id: {
type: Number
},
name: {type: String, required: true},
price: {type: String, required: true},
weight: Number,
price_unit: String,
user/orders
Find
Find Prev
Find All
Line 17, Column 23; Detect Indentation: Setting indentation to tabs
Tab Size: 4
JavaScript


C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\user.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
category.js
farmer.js
order.js
product.js
user.js
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
user.js — Middleware\routes\model
var mongoose = require('mongoose');
var autoIncrement = require('mongoose-auto-increment');
var connection = mongoose.createConnection("mongodb://localhost/amazon");
autoIncrement.initialize(connection);
var contactsSchema = mongoose.Schema({
number: {type: String, required: true},
default_card: {type:String, required:true, default: 'false'}
});
// var cardDetailsSchema = mongoose.Schema({
// ,
// default_card: {type:String, required:true, default: 'false'}
// });
var userSchema = mongoose.Schema({
c_id: {type: Number, required: true, index: true},
fname: {type: String, required: true},
lname: {type: String, required: true},
email: {type: String, required: true},
pass: {type: String, required: true},
address: {type: String},
zipcode: Number,
city: String,
state: String,
card_number: {type: Number},
name_on_card: {type: String},
user/orders
Find
Find Prev
Find All
Line 16, Column 22
Tab Size: 4
JavaScript


C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\farmer.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
user.js — Middleware\routes
user.js — Server\services
FOLDERS
AmazonFresh-master
Amazonnnnnn
Middleware
node_modules
public
routes
commons
model
admin.js
cart.js
driver.js
farmer.js
farmerLogin.js
index.js
login.js
order.js
passport.js
product.js
truck.js
user.js
rpc
views
app.js
package.json
index.ejs
product.js
product_page.ejs
myReviews.ejs
addressDetails.ejs
farmer.js
user.js — Middleware\routes
user.js — Server\services
}
else
{
res.send(resGen.responseGenerator(401, null));
}
}
});
};
exports.createFarmer = function(req,res){
var email = req.param("email");
var pass = req.param("pass");
var err = 0;
if(!validate.isEmail(email)){
res.send(resGen.responseGenerator(500,null));
err = 1;
}
if(pass.length < 6){
res.send(resGen.responseGenerator(500,null));
err = 1;
}
if(err == 0){
var msg_payload = {
"service" : "createFarmer",
user/orders
Find
Find Prev
Find All
Line 51, Column 21; Detect Indentation: Setting indentation to tabs
Tab Size: 4
JavaScript

Perfomance Graph :



Observations and lesson learnt

We learnt how to effectively work together in a team. In this era, team-work and communication between all the team members is the most important module for the project to work well. We observed that how a single minor problem can affect whole project and thus learnt to communicate solutions of all the problems to each and every member of the team. We observed that how practical implementation is carried out in an industry and learnt to adopt a particular model while traversing through the project development, implementation and testing phase. We also learnt how to adopt new technologies and implement them concurrently. Moreover, our project mainly targeted on performance of the system. We learnt how we can divide data into different databases as per their use, how to cache the required data and how to maintain session. We used Node.js which implements single threaded event loop handling, and so to increase the performance and handle multiple requests at the same time we learnt to implement queue viz. RabbitMQ. Above this we

also learnt time management, work distribution, module integration and different kinds of testing such as MOCHA testing