



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 $\geq 50\$$. 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, career, 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.

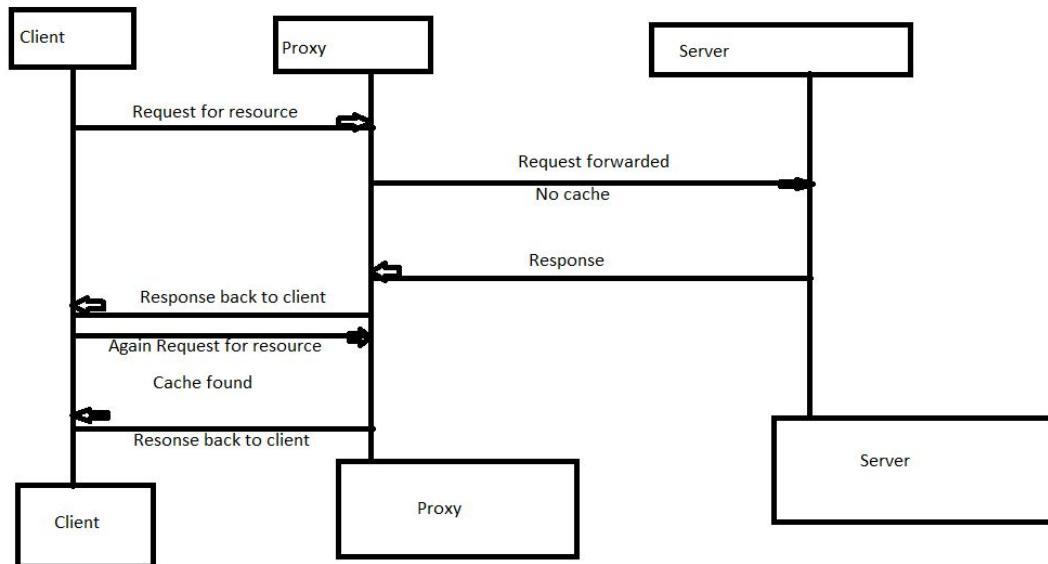
Project Description

1. Your Object Management Policy.

- **Sequelize** – 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

[amazonfresh](#) Already an AmazonFresh customer? [Sign In](#)

Search AmazonFresh Go!

freshest of the fresh

AMAZONFRESH BEST-SELLERS

[SHOP NOW >](#)

○ ○ ● ○ ○

AmazonFresh is available exclusively to Prime Fresh members.

[Sign in with Amazon account](#)

Not a Prime Fresh member? [Start your 30-day FREE trial today.](#)

[Browse Products](#)

\$8.04 (\$2.68/lb)
Organic Fuji Apples, 3 Lb

1

\$7.98 (\$3.99/lb)
Red Seedless Grapes, 2 Lb

1

\$4.49 (\$4.49/lb)
Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb

1

\$5.99 (\$2.00/lb)
Clementine Mandarins, 3 Lb

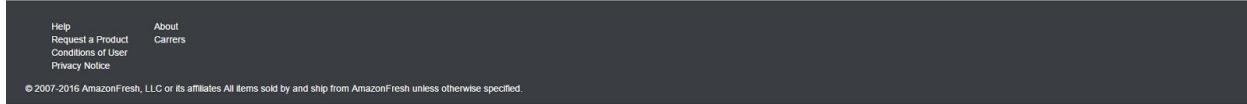
1

Dummy Page 1: Privacy Notice

[amazonfresh](#) Already an AmazonFresh customer? [Sign in](#)

[AmazonFresh Privacy Notice](#)

AmazonFresh® knows that you care how information about you is used and shared, and we appreciate your trust that we will do so carefully and sensibly. Please note that Amazon Services LLC operates this web site and is an affiliate of Amazon.com, Inc. ("Amazon.com"). If you have an account on www.amazon.com and a cookie from that site, information gathered by AmazonFresh may be correlated with any personally identifiable information that Amazon.com has and used with AmazonFresh and Amazon.com to improve the services we offer. As an affiliate of Amazon.com, AmazonFresh follows the same information practices as Amazon.com, and information we collect is subject to the Amazon.com Privacy Notice. If you do not want to receive e-mail or other mail from us, please adjust your E-mail Preferences. By visiting the AmazonFresh site, you are accepting the practices described in the [Amazon.com Privacy Notice](#).



Dummy Page 2: Carrers

The screenshot shows a web browser window with the URL 'localhost:3000/careers'. The page header includes the AmazonFresh logo and a link to 'Sign in'. Below the header, there's a section titled 'Join the AmazonFresh team!' with a brief description. A list of categories follows: 'Software Development', 'Product/Program Management', and 'Retail'. Each category has a brief description and a list of specific positions. At the bottom, there's a link to 'View all open Positions' and a note about location.

Join the AmazonFresh team!
AmazonFresh has open positions in many different job families across technology and retail.
We need people who are passionate about customer experience, have a strong bias for action, and love being owners. With customers ordering multiple times a week, your work will directly improve people's daily lives. And our extremely low overhead means that you can turn your ideas into reality in days, not months. [Learn more](#) about working at Amazon.com.

Software Development
On the development team, we're crafting software to better forecast how many loaves of bread to buy, revamping our user experience to make grocery shopping increasingly convenient, and iterating on algorithms to route trucks to homes as efficiently and environmentally consciously as possible – all while pushing code daily. We're looking for developers with strong technical backgrounds to join our team. Learn more about specific opportunities by following the links below:

- [Software Development Positions](#)

Product/Program Management
Our Product and Program Managers define and deliver some of the most exciting projects in all of Amazon. With end-to-end ownership, immediate customer feedback and the ability to develop a unique customer experience, PMs at AmazonFresh get to truly impact the lives of our valued customers. Learn more about specific opportunities by following the links below:

- [Product Management Positions](#)
- [Technical Program Management Positions](#)

Retail
Our retail positions span everything from Vendor and In-Stock Management to Site Merchandising and UI/UX Design. Learn more about specific opportunities below:

- [Account Management Positions](#)
- [Vendor Management and Buying Positions](#)

[View all open Positions](#)
All positions are located in Seattle, WA unless otherwise stated.

Help About
Request a Product Carrers
Conditions of User
Privacy Notice

© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

Dummy Page 3: Conditions of User

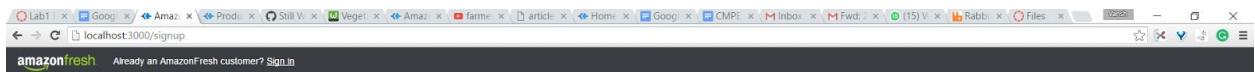
The screenshot shows a web browser window with the URL 'localhost:3000/conditions'. The page header includes the AmazonFresh logo and a link to 'Sign in'. Below the header, there's a section titled 'Conditions of Use' with a brief description. At the bottom, there's a note about the website being an affiliate of Amazon.com.

Conditions of Use
AmazonFresh.com is an affiliate of Amazon.com. The terms described in the [Amazon.com](#) Conditions of Use apply to the use of AmazonFresh.com, and by visiting this website, you are accepting those terms.

Help About
Request a Product Carrers
Conditions of User
Privacy Notice

© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

Client needs to Signup



Create New Account

First Name x

Last Name x

Email Address

Password

Verify Password

Continue



Create New Account

First Name ✓

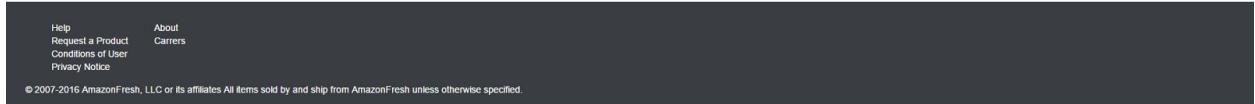
Last Name x

Email Address

Password

Verify Password

Continue



Amazon-Fresh localhost:3000/signup

amazonfresh Already an AmazonFresh customer? Sign in

Create New Account

Vanish ✓
Shah ✓
Email Address ✗
Password
Verify Password

Continue

Help Request a Product About Carrers Conditions of User Privacy Notice

© 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

Vanish ✓
Shah ✓
shahvansh123@gmail.com ✓
Password ✗
Password must be more than 8 characters!
Verify Password

Continue

Help Request a Product About Carrers Conditions of User Privacy Notice

© 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

Vanish ✓
Shah ✓
shahvansh123@gmail.com ✓
..... ✓
Verify Password ✗

Continue

Help Request a Product About Carrers
Conditions of User Privacy Notice

© 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

Vanish ✓
Shah ✓
shahvansh123@gmail.com ✓
..... ✓
..... ✓

Continue

Help Request a Product About Carrers
Conditions of User Privacy Notice

© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

Email Id already Taken:

The screenshot shows a web browser window for 'Amazon-Fresh' at 'localhost:3000/signup'. The title bar says 'Amazon-Fresh'. The main content is a 'Create New Account' form. The first two fields ('First Name' and 'Last Name') have green checkmarks. The third field, 'Email Address', contains 'shahvansh123@gmail.com' and has a red 'X' icon, with the validation message 'Email has already been registered!' below it. The next two fields are password inputs, both with green checkmarks. A 'Continue' button is at the bottom.

Create New Account

Vansh ✓

Shah ✓

shahvansh123@gmail.com ✗
Email has already been registered!

..... ✓

..... ✓

Continue



The screenshot shows a 'Create New Account' form. The first three fields ('First Name', 'Last Name', and 'Email Address') have green checkmarks. The fourth field, 'Password', has a red 'X' icon and the validation message 'Password must be more than 5 characters!'. The fifth field, 'Verify Password', has a red 'X' icon. A 'Continue' button is at the bottom.

Create New Account

Aashay ✓

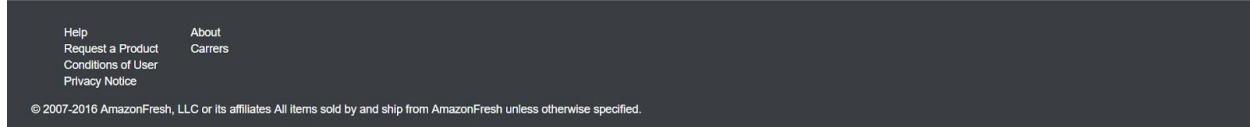
Shah ✓

shah.aashay21@gmail.co ✓

..... ✗
Password must be more than 5 characters!

Verify Password ✗

Continue



The menu bar changes on Login

This screenshot shows the AmazonFresh website with a user logged in as "Aashay Shah". The top navigation bar includes links for "Your Deliveries", "Messages", "Your Lists", and "Past Purchases". A search bar and a "Go!" button are also present. On the right side, there is a shopping cart summary for "Aashay Shah - San Jose" with a subtotal of \$68.33 for 9 items. The cart contains two items: "Red Seedless Grapes" (8 units) and "Organic Greenhouse Red On-The-Vine Tomatoes" (1 unit). A "Proceed to Checkout" button is visible. Below the cart, there is a promotional banner for "AMAZONFRESH BEST-SELLERS" featuring eggs, followed by a grid of four product cards: Organic Fuji Apples, Red Seedless Grapes, Organic Greenhouse Red On-The-Vine Tomatoes, and Clementine Mandarins, each with an "Add to Cart" button.

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

This screenshot shows the same AmazonFresh website, but the user is now a guest. The top navigation bar remains the same. The shopping cart summary on the right shows a subtotal of \$46.52 for 7 items. The cart contains five items: "Organic Fuji Apples" (2 units), "Red Seedless Grapes" (1 unit), "Organic Greenhouse Red On-The-Vine Tomatoes" (1 unit), "Clementine Mandarins" (2 units), and "Organic Gold Potatoes" (1 unit). The "Proceed to Checkout" button is now disabled. The rest of the page layout is identical to the logged-in version, including the promotional banner and product grid.

All the products 'Add to Cart' button Enabled

Screenshot of the AmazonFresh homepage showing the shopping cart interface.

The top navigation bar includes links for "Hi, Aashay Shah", "Your Deliveries", "Messages", "Your Lists", and "Past Purchases". The search bar shows "Search AmazonFresh" and a "Go!" button. To the right, it says "Deliver To Aashay Shah - San Jose" and "(change delivery address)". A "Proceed to Checkout" button is visible.

The main content area features a promotional banner for "HERSHEY'S Delight" syrups. Below the banner, there's a section titled "LEAVE A LITTLE ROOM FOR delight" with a "SHOP NOW" button. On the right, the shopping cart summary shows a subtotal of \$46.52 for 7 items:

- 2 Organic Fuji Apples (\$16.08)
- 1 Red Seedless Grapes (\$7.98)
- 1 Organic Greenhouse Red On-The-Vine Tomatoes (\$4.49)
- 2 Clementine Mandarins (\$11.98)
- 1 Organic Gold Potatoes (\$5.99)

Below the cart summary, there are four product cards for "Organic Fuji Apples", "Red Seedless Grapes", "Organic Greenhouse Red On-The-Vine Tomatoes", and "Clementine Mandarins", each with a price, quantity dropdown, and an "Add to Cart" button.

Product Page after Login

Screenshot of a product page for "Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb".

The top navigation bar is identical to the homepage. The product image shows several red tomatoes on a vine. A "Hover to zoom" button is present. A "Roll over image to zoom in" link is below the image. The product details include:

- Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb
- Other products by [babubhai](#)
- ★★★☆☆ (1) reviews
- \$4.49 (\$4.49/lb)

A note indicates "Only 4 items remaining". The "Add to Cart" button is located below the price. The product is sold by "AmazonFresh" and shipped by "AmazonFresh".

To the right, the shopping cart summary shows a subtotal of \$184.88 for 32 items:

- 5 Organic Fuji Apples (\$40.20)
- 5 Red Seedless Grapes (\$39.90)
- 4 Organic Greenhouse Red On-The-Vine Tomatoes (\$17.96)
- 7 Clementine Mandarins (\$41.53)
- 4 Organic Gold Potatoes (\$23.96)
- 7 Red Onions, Medium (\$20.93)

Below the product page, there's a "Customer Ratings and Reviews" section with a "Create a Review" button. A review entry from "User: aashay" is shown:

User: aashay
Ratings: 3/5
Title: Jakkaass
Review: Mast 6e...

At the bottom, a "Freshness Guarantee" note states: "AmazonFresh guarantees every product will be delivered to your home well within the manufacturer's recommended use by, sell by, best by, or expiration date."

Zoom Effect:

Hi, Shaishav Patel | Your Deliveries | Messages | Your Lists | Past Purchases

Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb
Other products by babubhai

(3)
\$4.49 (\$4.49/lb)

Only 4 items remaining
1 Add to Cart

Sold by AmazonFresh. Shipped by AmazonFresh.

Customer Ratings and Reviews
Create a Review

User: aashay
Ratings: 3/5
Title: Jakkaass
Review: Mast 6e...

User: Neel
Ratings: 5/5
Title: Nice.
Review: Very nice products.

User: Shaishav
Ratings: 5/5
Title: Great.
Review: Nice Products.

Create Review

Hi, Aashay Shah | Your Deliveries | Messages | Your Lists | Past Purchases

Products by Chandubhai

Organic Fuji Apples, 3 Lb
\$8.04 (\$2.68/lb)

Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb
\$4.49 (\$4.49/lb)

Organic Gold Potatoes
Red Onion, Medium

Create your Review

How do you rate this item? 4

What is the title of your review?
Nice

Write your review
Products were very fresh.

Submit

Product Reviews

Customer Ratings and Reviews

Create a Review

User: aashay

Ratings: 3/5

Title: Jakkkaass

Review: Mast Ge...

Farmer page after User login

amazonfresh Hi, Aashay Shah - | Your Deliveries | Messages | Your Lists | Past Purchases

Farmer Name: Chandubhai kaiNai
Email: chandi@farmer.com
Introduction: Navo j 6u etle koi intro rahyo j nathi.



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

Proceed to Checkout

Subtotal \$184.88 (32 Items)

Quantity	Item	Price
5	Organic Fuji Apples	\$40.20
5	Red Seedless Grapes	\$39.99
4	Organic Greenhouse Red On-The-Vine Tomatoes	\$17.98
7	Clementine Mandarins	\$41.93
4	Organic Gold Potatoes	\$23.96
7	Red Onions, Medium	\$20.53

Products by Chandubhai



\$8.04 (\$2.68/lb)
Organic Fuji Apples, 3 Lb

1 Add to Cart



\$4.49 (\$4.49/lb)
Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb

1 Add to Cart



\$5.99 (\$2.00/lb)
Clementine Mandarins, 3 Lb

1 Add to Cart

Customer Ratings and Reviews

Order Details

amazonfresh Hi, Aashay Shah | My Address | Reviews | Order List | Payment Options

Current Orders Canceled Orders Past Orders

600000012

Subtotal : \$50.89
Shipping Cost : \$ 5
Total Tax : \$2.28
Total Cost: \$
Delivery Address : 1400 Birchmeadow Ln, san jose, ca - 95131
Order Placed At :
2016-05-02T06:37:57.164Z

Help About
Request a Product Careers
Conditions of User Privacy Notice

© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

Checking the Product Quantity and displaying when quantity is less than 5

amazonfresh Hi, Aashay Shah | Your Deliveries | Messages | Your Lists | Past Purchases



Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb
Other products by | [babubhai](#)

★★★☆☆ (1)
\$4.49 (\$4.49/lb)

Only 4 Items remaining
1 Add to Cart

Sold by AmazonFresh. Shipped by AmazonFresh.

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

Customer Ratings and Reviews
[Create a Review](#)

User: aashay
 Ratings: 3/5
 Title: Jakkkaass
 Review: Mast 6e...

Freshness Guarantee: AmazonFresh guarantees every product will be delivered to your home well within the manufacturer's recommended use by, sell by, best by, or expiration date.

amazonfresh Hi, Aashay Shah | Your Deliveries | Messages | Your Lists | Past Purchases

Browse Categories



FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)



FRESH VEGETABLES >>
 Carrot
 Corn
 Lettuce
 Sweet Potato
 Onion
 Organic
[Shop All >](#)



FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)



FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)



FRESH VEGETABLES >>
 Carrot
 Corn
 Lettuce
 Sweet Potato
 Onion
 Organic



FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)



FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)



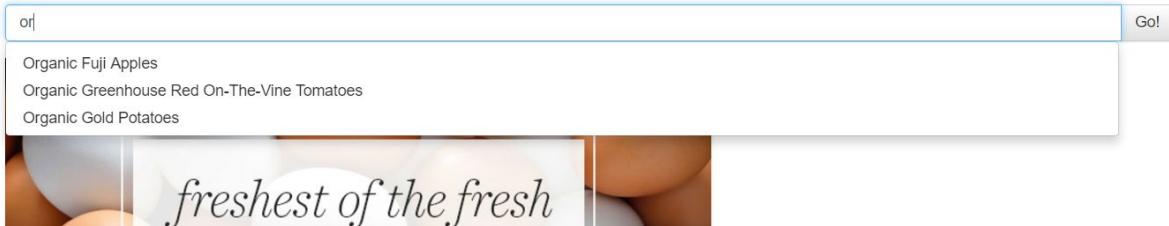
FRESH FRUIT >>
 Apples
 Bananas
 Berries
 Citrus
 Packaged Fruit
 Organic Fruit
[Shop All >](#)





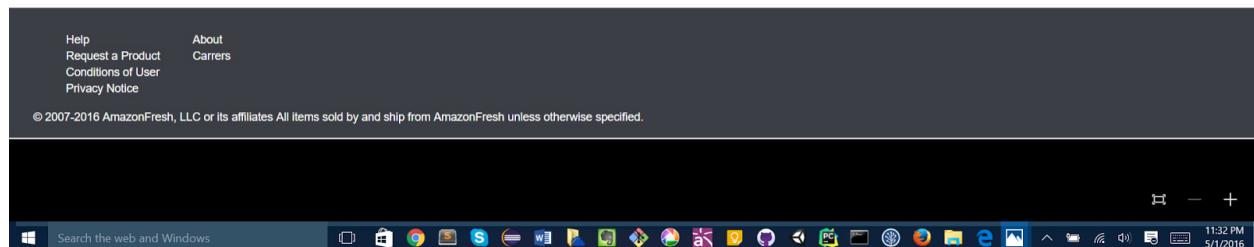


Search the product in search Tab



Edit Address

The screenshot shows the 'Edit Address' page. At the top, there is a navigation bar with links: 'Hi, Aashay Shah ▾ | My Address | Reviews | Order List | Payment Options'. Below the navigation bar is a section titled 'Address Details' containing fields for 'Address' (1400 Birchmeadow Ln), 'City' (san jose), 'State' (ca), and 'Zip' (95131). At the bottom of this section is a blue 'Update Address Details' button.



Edit Card Details

The screenshot shows the 'Edit Card Details' page. At the top, there is a navigation bar with links: 'Hi, Aashay Shah ▾ | My Address | Reviews | Order List | Payment Options'. Below the navigation bar is a section titled 'Credit Card Details' containing fields for 'Card Number' (123456123456), 'Name on the Card' (aashay), 'Expiry Date' (02 2017), and 'CVV' (represented by three dots). At the bottom of this section is a blue 'Update credit Card Details' button.

The footer area includes links for 'Help', 'Request a Product', 'Conditions of User', 'Privacy Notice', 'About', and 'Careers'. Below these links is a copyright notice: '© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.'

The User Account Details

amazonfresh Hi, Aashay Shah - | My Address | Reviews | Order List | Payment Options

Search AmazonFresh Go!

Your Account!

[Account Settings](#)

You are Logged in as: shah.aashay21@gmail.com

Payment Options

Card Number: 123456123456 | Name On Card aashay
You can Edit your Payment options under: [Manage Your Payment Options](#)

Primary Delivery Address

1400 BIRCHMEADOW LN , SAN JOSE
CA , 95131
Update Your Address from Here : [Update](#)

Your Orders

To see your Order History : [Click Here](#)

My Reviews

Check all of my Reviews : [Here](#)

Help About
Request a Product Careers
Conditions of User
Privacy Notice

© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified.

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

amazonfresh Hi, Aashay Shah | Your Deliveries | Messages | Your Lists | Past Purchases

Create your Review

How do you rate this item? ★★★★☆ 4

What is the title of your review?
Nice

Write your review
Products were very fresh.

Submit

Products by Chandubhai

	\$8.04 (\$2.68/lb) Organic Fuji Apples, 3 Lb
	\$4.49 (\$4.49/lb) Organic Greenhouse Red On-The-Vine Tomatoes, 1 Lb
	7 \$41.93 Clementine Mandarins
	4 \$23.96 Organic Gold Potatoes
	7 \$20.93 Red Onions, Medium

Customer Ratings and Reviews

Create a Review

Proceed to Checkout

amazonfresh Hi, Aashay Shah | Your Deliveries | Messages | Your Lists | Past Purchases

Delivery Options

Delivery Time edit
Mon, 2 ▾ 8:00 ▾
(you won't be disturbed when the delivery is made)

Card Details
Name On Card: Aashay
***** 3456 ▾
[Add a new credit card](#)

Payment Options

Delivery Address edit
1400 BIRCHMEADOW LN
SAN JOSE CA 95131

Place Order

Did you forget something?
Add more items to your order now. You can also add more items after you place this order.

[Add additional items](#)

Item Subtotal: \$184.88
Delivery Charge: \$5.00
Estimated Tax*: \$9.70
Total: \$199.58

Subtotal \$184.88		(32 Items)
	5 \$40.20	
	5 \$39.90	
	4 \$17.96	
	7 \$41.93	
	4 \$23.96	

User Reviews

Product Review

The screenshot shows a web browser window for localhost:3000/myReviews. The header includes the AmazonFresh logo, a user greeting "Hi, Vansh Shah", and navigation links for "My Address", "Reviews", "Order List", and "Payment Options". The main content area displays two reviews for a product named "Tomato".

Review 1:
User: Vansh
Product Name: Tomato
Rating: 5 / 5
Title: Very Awesome
Review: Superb

Review 2:
User: Vansh
Product Name: Tomato
Rating: 3 / 5
Title: Average
Review: The Product Is okay

The footer contains links for "Help", "Request a Product", "Conditions of User", "Privacy Notice", and "About Carrers". A copyright notice at the bottom states: "© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified."

Farmer Review

The screenshot shows a web browser window for localhost:3000/myReviews. The header includes the AmazonFresh logo, a user greeting "Hi, Vansh Shah", and navigation links for "My Address", "Reviews", "Order List", and "Payment Options". The main content area displays one farmer review.

Review:
User: Vansh
Farmer Name: Paul Jackob
Rating: 5/5
Title: Good Farmer
Review:

The footer contains links for "Help", "Request a Product", "Conditions of User", "Privacy Notice", and "About Carrers". A copyright notice at the bottom states: "© 2007-2016 AmazonFresh, LLC or its affiliates All items sold by and ship from AmazonFresh unless otherwise specified."

Please Provide Address Details

Please provide delivery address.

Delivery Options	Payment Options	Place Order
Delivery Time edit Mon, 2 ▾ 8:00 ▾ <small>(you won't be disturbed when the delivery is made)</small>	Card Details Name On Card: Shaishav <small>***** 1334 ▾</small> Add a new credit card	Did you forget something? <small>Add more items to your order now. You can also add more items after you place this order.</small> Add additional items
Delivery Address edit		Item Subtotal: \$61.04 Delivery Charge: \$5.00 Estimated Tax*: \$3.93 Total: \$69.97
Subtotal \$61.04 (9 Items)		
 Organic Fuji Apples <small>\$24.12</small>  Red Seedless Grapes <small>\$15.96</small>  Organic Greenhouse Red On-The-Vine <small>\$8.98</small>  Clementine Mandarins <small>\$11.98</small>		

Farmer Signup Page

localhost:3000/farmer/signup

Farmer SignUp

First Name	<input type="text" value="First Name"/>
Last Name	<input type="text" value="Last Name"/>
Email address	<input type="text" value="Enter email"/>
Password	<input type="password" value="Password"/>
Verify Password	<input type="password" value="Verify Password"/>
Sign Up Farmer Login	

Farmer Signing Up

localhost:3000/farmer/signup

Farmer SignUp

First Name
Vansh

Last Name
Shah

Email address
shahvansh123@gmail.com

Password
.....

Verify Password
.....

[Sign Up](#)

[Farmer Login](#)

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

Farmer SignUp

First Name

Last Name

Email address

Password

Verify Password

Registered successfully! Wait for admin to Approve.

[Farmer Login](#)

Farmer page after Login

A screenshot of a web browser window showing the "Farmer Info" control panel. The URL is "localhost:3000/farmer/home". The page displays the following information:

- First Name: Vansh
- Last Name: Shah
- Email: shahvansh123@gmail.com
- Introduction: Hello!
- Video: empty
- Tax: 5
- Contact No: empty
- Address: Type Address Here
- City: empty
- State: Your State
- Zipcode: 12345

A "Edit" button is located at the bottom left. The top right corner shows the user "admin name" and a "Logout" link. The footer includes copyright information "Copyright © 2016-2017 Amazon Fresh. All rights reserved." and a "Version 2.3.0" link.

Farmer can edit the Farmer details

localhost:3000/farmer/home

ALT Farmer Info Control panel

First Name: Vansh
Last Name: Shah
Email: shahvansh123@gmail.com
Introduction: Hello!
Video:
Taxi: 5
Contact No:
Address: Type Address Here
City: Your City
State: Your State
Zipcode: 12345

Save Cancel

Add Product

Product Name
Enter Product Name

Category
Select Category

Price
Enter Price

Weight
Enter Weight

Unit
Enter Weight Unit

Product Info
Enter Product info

Product Description
Enter Product Description

Product Features
Enter Product Features

Quantity
Enter Quantity

Category 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

Category ID	Price	Weight	Unit	Product Info	Product Description	Features	Quantity	Updated At
-------------	-------	--------	------	--------------	---------------------	----------	----------	------------

Admin Tab

Farmers Data Tables

Farmer ID	First Name	Last Name	Email	Address	City	State	Zipcode	Intro	Video	Tax	Contact No	Approve/Reject	Joined At	
2000000001	Neel	Shah	neelwittome@gmail.com	234 W San Jose	San Jose	CA	95121	hey this is neel, I sell fresh stuff.	https://www.youtube.com/watch?v=9qjDAPWnQ	10	4087887777	Approved	May 1, 2018 8:30:39 PM	Approve Reject
2000000002	Aashay	Shah	aashay@gmail.com	123 Wr seides	San Jose	CA	95121	This is aashay's introduction.	https://www.youtube.com/watch?v=9qjDAPWnQ	10	7040887777	Approved	May 1, 2018 8:41:50 PM	Approve Reject

Add Farmer

Copyright © 2018-2019 Amazon Fresh. All rights reserved.

Version 1.3.0

Add Farmer

First Name	<input type="text" value="Neel"/>
Last Name	<input type="text" value="Shah"/>
Email	<input type="text" value="neelwittome@gmail.com"/>
Address	<input type="text" value="479 W San Carlos"/>
City	<input type="text" value="San Jose"/>
State	<input type="text" value="CA"/>
Zipcode	<input type="text" value="95121"/>
Introduction	<input type="text" value="Get all my products with freshness guarantee."/>
Contact No	<input type="text" value="4087887777"/>
Youtube Video Link	<input type="text" value="https://www.youtube.com/watch?v=9qjDAPWnQ"/>
Tax (Percentage)	<input type="text" value="10"/>

Copyright © 2018-2019 Amazon Fresh. All rights reserved.

Version 1.3.0

Farmers Data Tables All data

Farmer Data Table

Farmer Id	First Name	Last Name	Email	Address	City	State	Zipcode	Info	Video	Tax
2000000001	Neel	Shah	neelshah99@gmail.com	234 w san jgk jd	San Jose	CA	95112	hey this is neel. I sell fresh stuff	https://www.youtube.com/watch?v=vnjQEAfPwHQ	10
2000000002	Aashay	Shah	aashay@gmail.com	114 th street	San Jose	CA	95117	This is aashays introduct	https://www.youtube.com	20
Add Farmer										

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 2.1.0

Product Data Tables All data

Product Data Table

Farmer Id	Farmer Name	Category Id	Price	Weight	Unit	Product Info	Product Description	Features	Quantity	Updated At
2000000001	Neel Shah	Fresh vegetables	5	1	kg	these are mexican best tomatoes	>this product description	this product features for tomatoes	10	May 1, 2018 8:00:02 PM
2000000002	Neel Shah	Nuts	8	2.0	kg	Nuts guiana, contains	they are imported from	large A+ grade	100	May 1, 2018 8:10:02 PM
2000000003	Aashay Shah	Eggs	12	1	kg	soft	soft	soft	100	May 1, 2018 8:22:49 PM
Add Product										

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 2.1.0

Product Data Tables All data

Home Tables Product table

Logout

Dashboard Tables Farmers List Products List Trucks List Drivers List Customers List Orders List

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

200000001 Neal Utah Fresh vegetables \$ 1 lb. These are mexican best tomatoes. This product description this is good features for tomatoes.

200000001 Neal Utah Nutri Multi 2 1.8 lb. Nutri granola. contains They are imported from Large A+ grade 100 May 1, 2018 8:30:02 PM

200000001 Archey Eggs Shaff 12 1 kg. safe safe safe 100 May 1, 2018 9:18:28 PM

Farmer ID Farmer Name Category ID Price Weight Unit Product Info Product Description Features Quantity Updated At

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.0

AdminLTE

Logout

Dashboard Tables Farmers List Products List Trucks List Drivers List Customers List Orders List

Tables

Farmers List Products List Trucks List Drivers List Customers List Orders List

Trucks Data Table

Truck ID Truck Number

100000001 123456789 Edit Delete

100000002 987654321 Edit Delete

100000003 0987654321 Edit Delete

100000004 1234567890 Edit Delete

100000005 1234567890 Edit Delete

Truck ID Truck Number

Add Truck

Truck Number

123456

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	Contact
2000000001	1234567	bhu	khanya	bhu@gmail.com	123 San Salvador	San Jose	CA	95111	708477997
2000000002	1234567	john	doe	doe12@gmail.com	678 55 St	San Jose	CA	95111	5684889999

Add Driver

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.0

Vivek Mehta

Customers Data Tables

Customers Data Table

Customer Id	First Name	Last Name	Email	Address	City	State	ZIP	Contact
1000000001	vivek	mehta	vivek@gmail.com	475 W San Carlos St, Apt 1237	San Jose	California	95110	408887079

Add Customer

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.0

Vivek Mehta

Online

Search...

View Dashboard

Tables

- Farmers List
- Products List
- Trucks List
- Drivers List
- Customers List
- Orders List

Orders Data Tables All Data

Pending Orders In Progress Orders Completed Orders Cancelled Orders

Order ID	Customer ID	Date	Total	Drop Time	Driver ID
6000000001	1000000001	5/5/18 11:59 PM	\$14.8	5/1/18 8:00 AM	<input type="text"/> Assign
6000000002	1000000002	5/5/18 11:59 PM	\$6.8	5/1/18 8:00 AM	<input type="text"/> Assign

Order ID	Customer ID	Driver ID	Date	Total	Drop Time
----------	-------------	-----------	------	-------	-----------

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.2

Vivek Mehta

Online

Search...

View Dashboard

Tables

- Farmers List
- Products List
- Trucks List
- Drivers List
- Customers List
- Orders List

Orders Data Tables All Data

Pending Orders In Progress Orders Completed Orders Cancelled Orders

Order ID	Customer ID	Date	Total	Drop Time	Driver ID
6000000001	1000000001	5/5/18 11:59 PM	\$14.8	5/1/18 8:00 AM	<input type="text"/> Complete
6000000002	1000000002	5/5/18 11:59 PM	\$6.8	5/1/18 8:00 AM	<input type="text"/> Complete

Order ID	Customer ID	Driver ID	Date	Total	Drop Time
----------	-------------	-----------	------	-------	-----------

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.2

Vivek Mehta
AI India

Search...

New Assessment

Dashboard

Tables

- Farmers List
- Products List
- Trucks List
- Drivers List
- Customers List
- Orders List

Orders Data Tables All 0 rows

Orders Data Table

Pending Orders In Progress Orders Completed Orders Cancelled Orders

Order ID	Customer ID	Date	Total	Drop Time	Driver ID
0000000001	0000000001	5/1/18 10:13 PM	\$14.9	5/1/18 8:00 AM	

Order ID	Customer ID	Driver ID	Date	Total	Drop Time

Copyright © 2016-2017 Amazon Fresh. All rights reserved.

Version 1.3.0

This screenshot shows the 'Orders Data Table' page. The top navigation bar includes a user profile for 'Vivek Mehta AI India', a search bar, and a dashboard link. A sidebar on the left lists various data tables: Farmers List, Products List, Trucks List, Drivers List, Customers List, and Orders List. The main content area is titled 'Orders Data Tables' with a note '(All 0 rows)'. It contains two tables: one for 'Pending Orders' and one for 'Cancelled Orders'. Both tables have columns for Order ID, Customer ID, Date, Total, Drop Time, and Driver ID. The first row in the Pending Orders table is highlighted with a light orange background. The second table is currently empty. At the bottom, there is a copyright notice and a version number 'Version 1.3.0'.

Vivek Mehta
AI India

Search...

New Assessment

Dashboard

Tables

- Farmers List
- Products List
- Trucks List
- Drivers List
- Customers List
- Orders List

Orders Data Tables All 0 rows

Orders Data Table

Pending Orders In Progress Orders Completed Orders Cancelled Orders

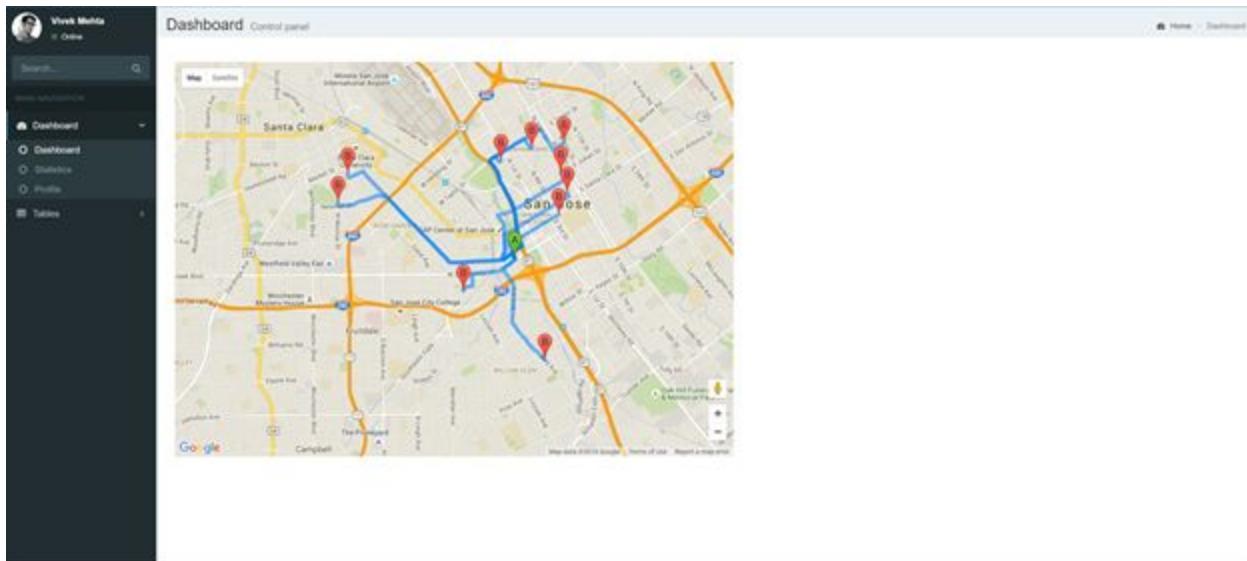
Order ID	Customer ID	Date	Total	Drop Time	Driver ID
0000000001	0000000001	5/1/18 11:53 PM	\$14.9	5/1/18 8:00 AM	

Order ID	Customer ID	Driver ID	Date	Total	Drop Time

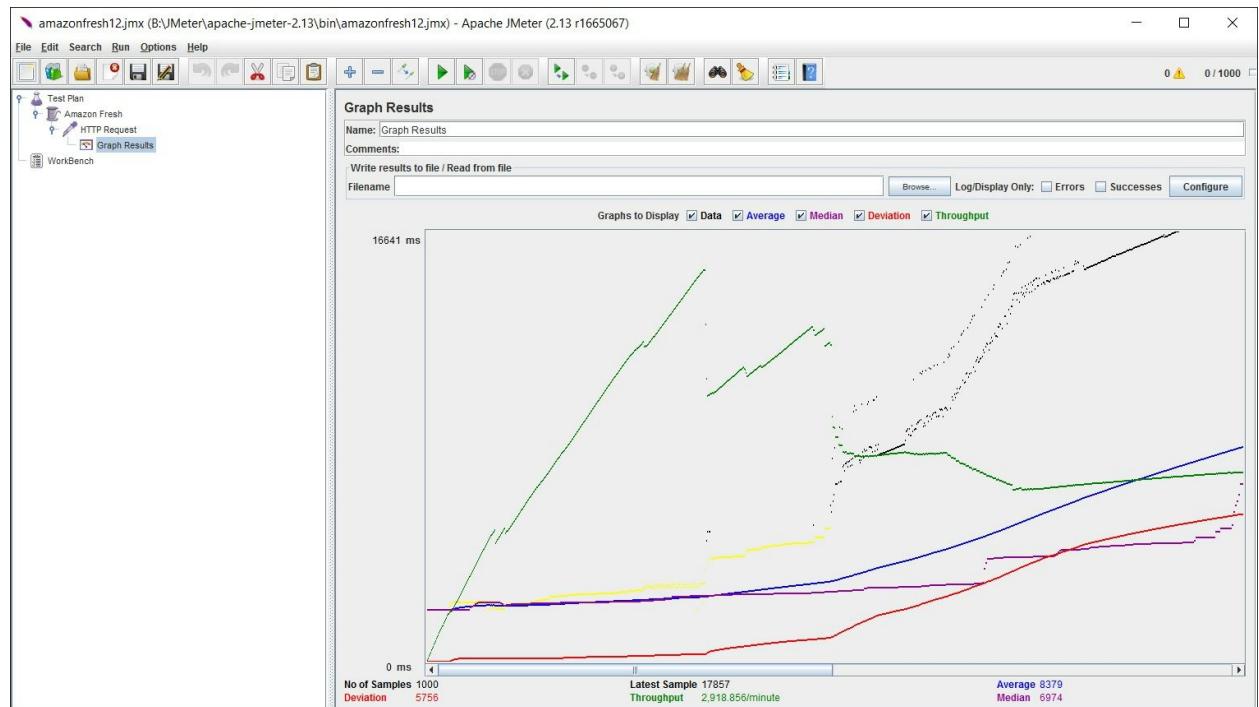
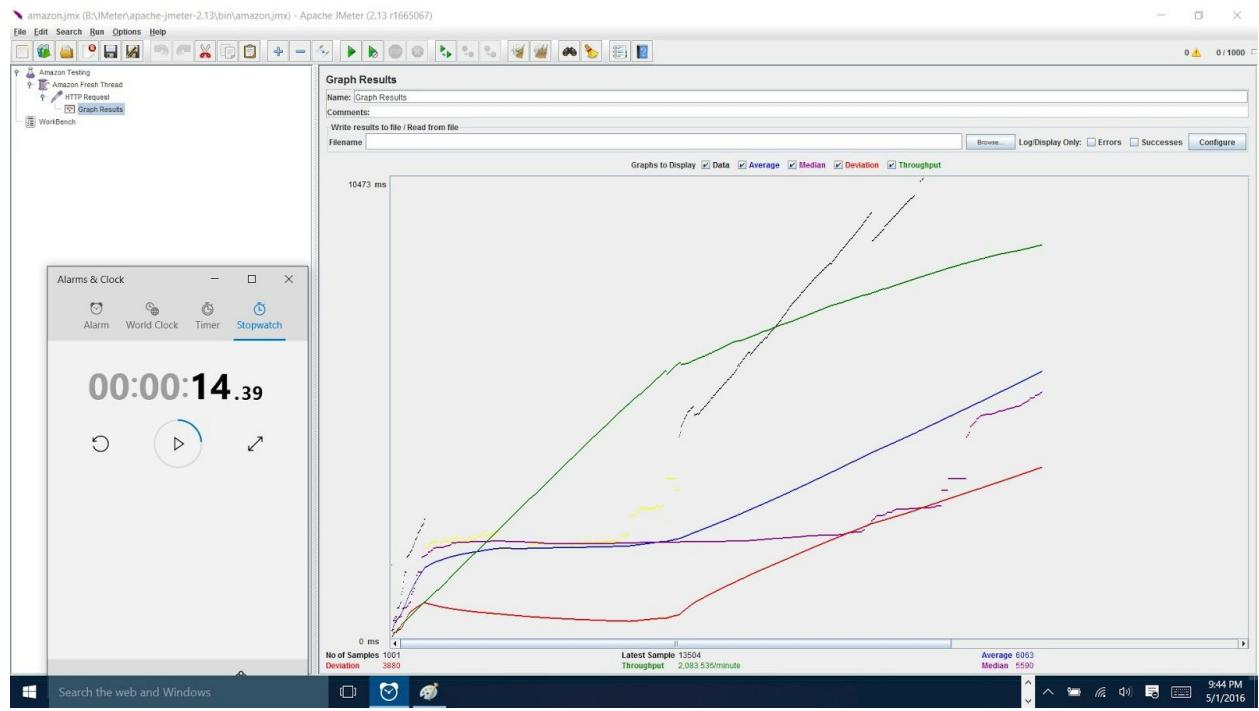
Copyright © 2016-2017 Amazon Fresh. All rights reserved.

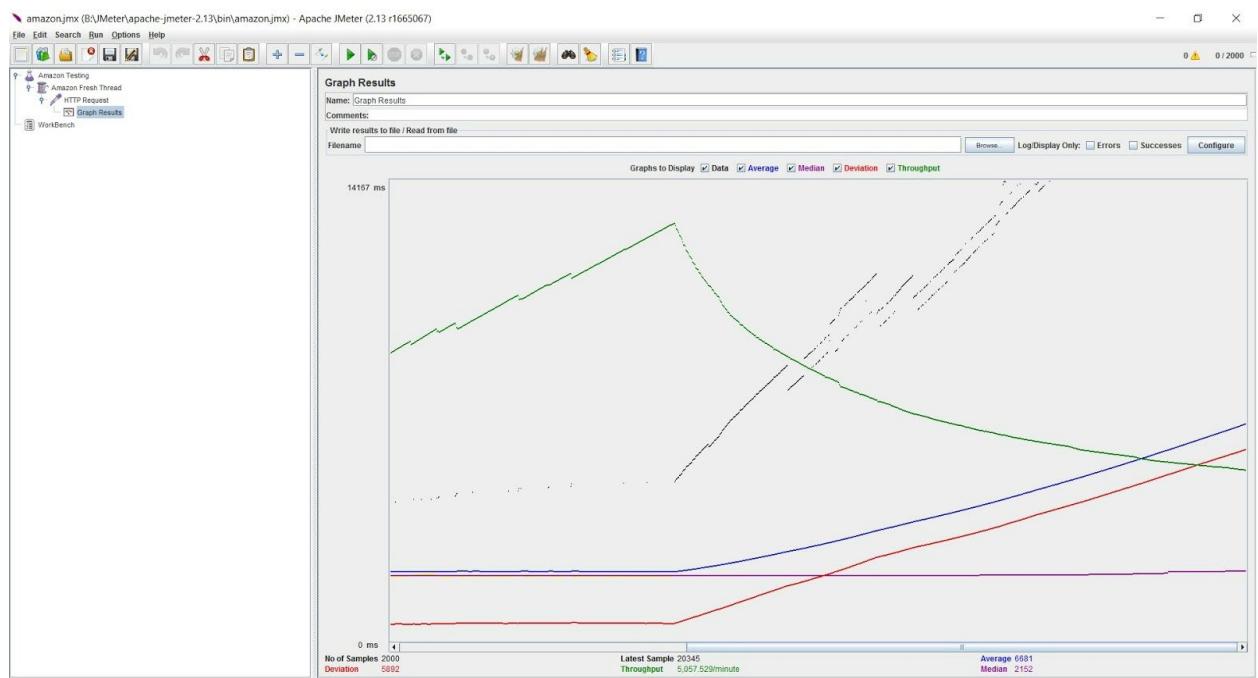
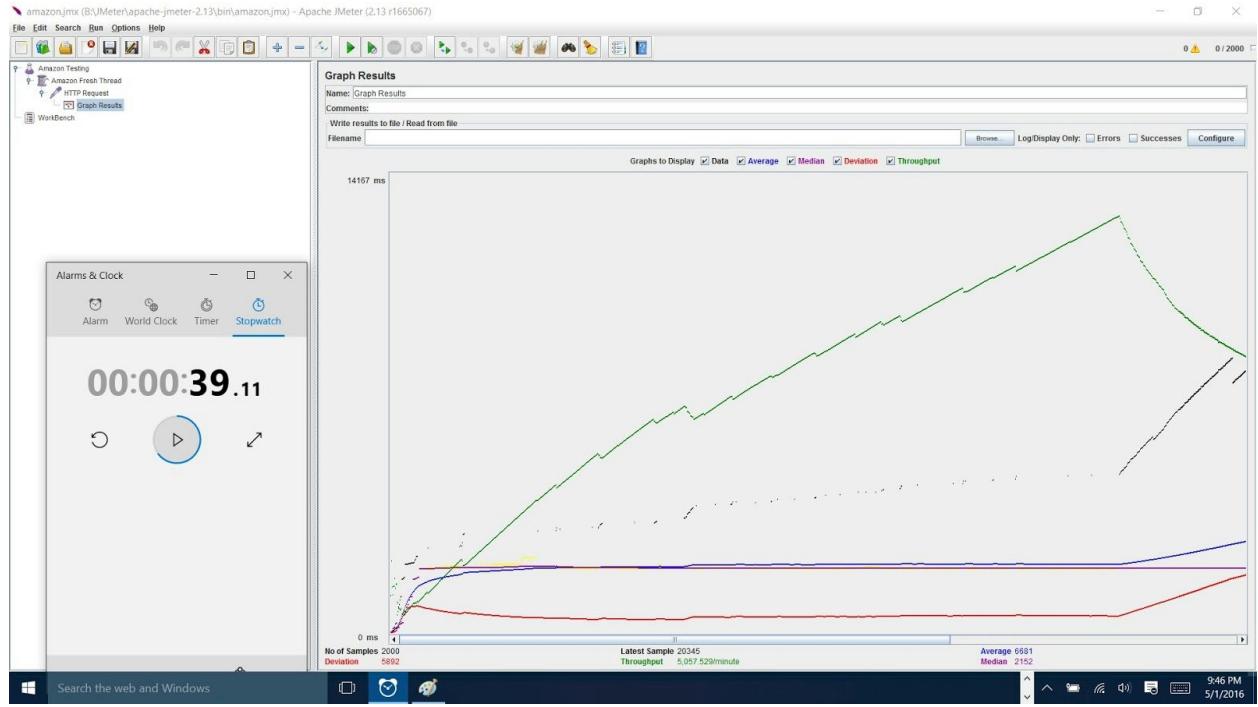
Version 1.3.0

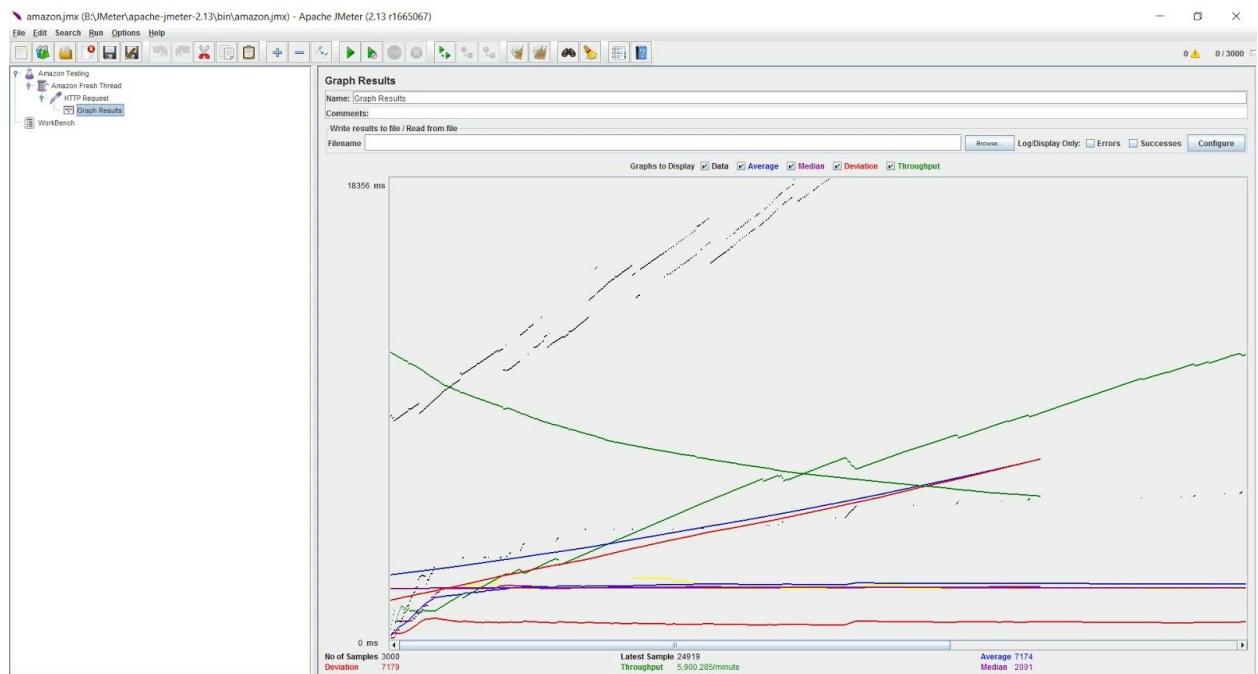
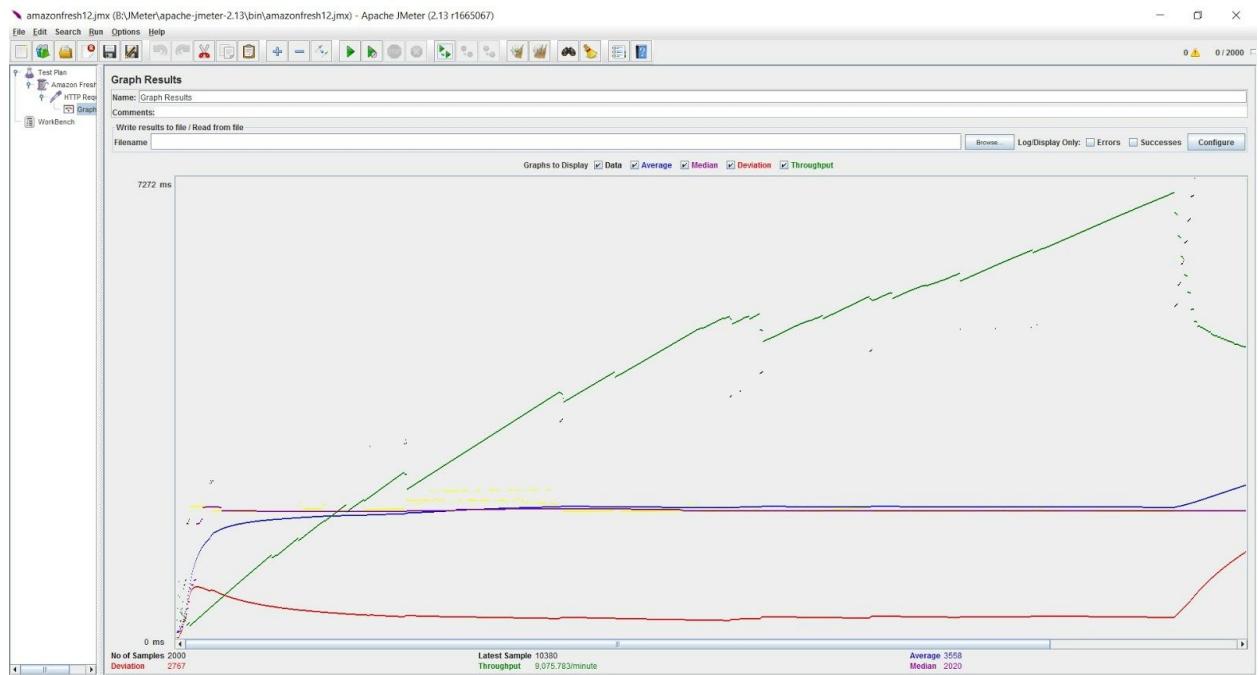
This screenshot is identical to the one above, showing the 'Orders Data Table' page. The layout, data tables, and overall interface are the same, including the pending order entry and the empty cancelled orders table.

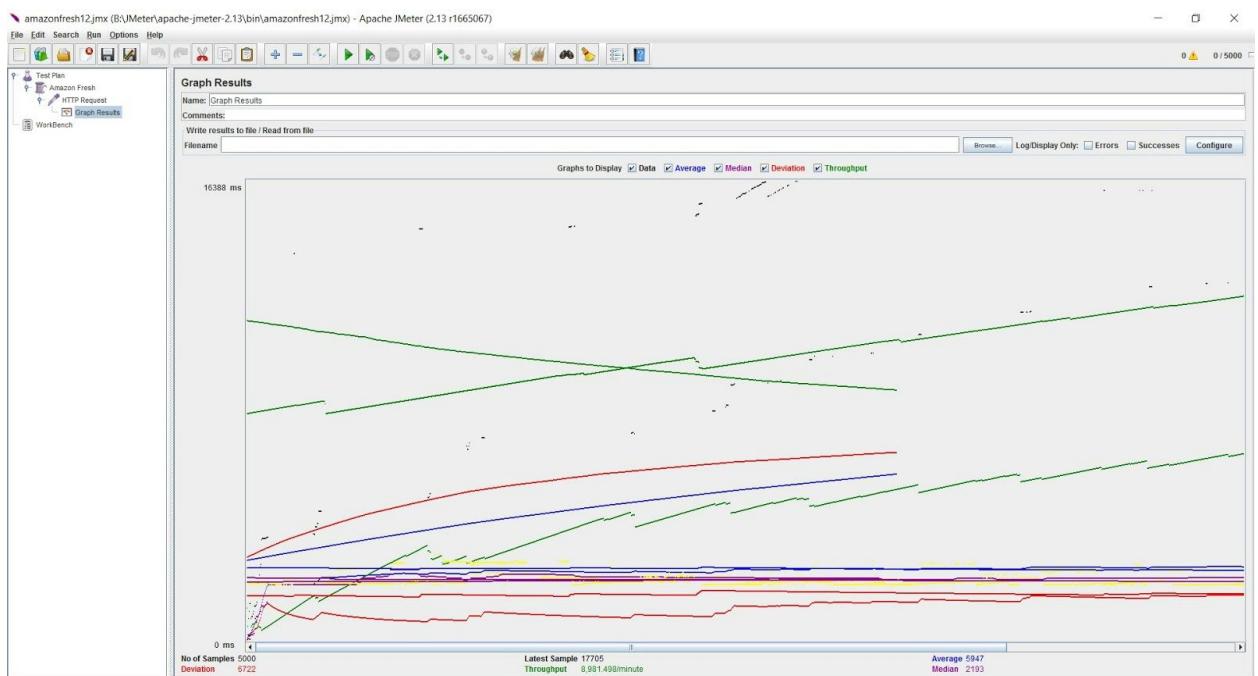
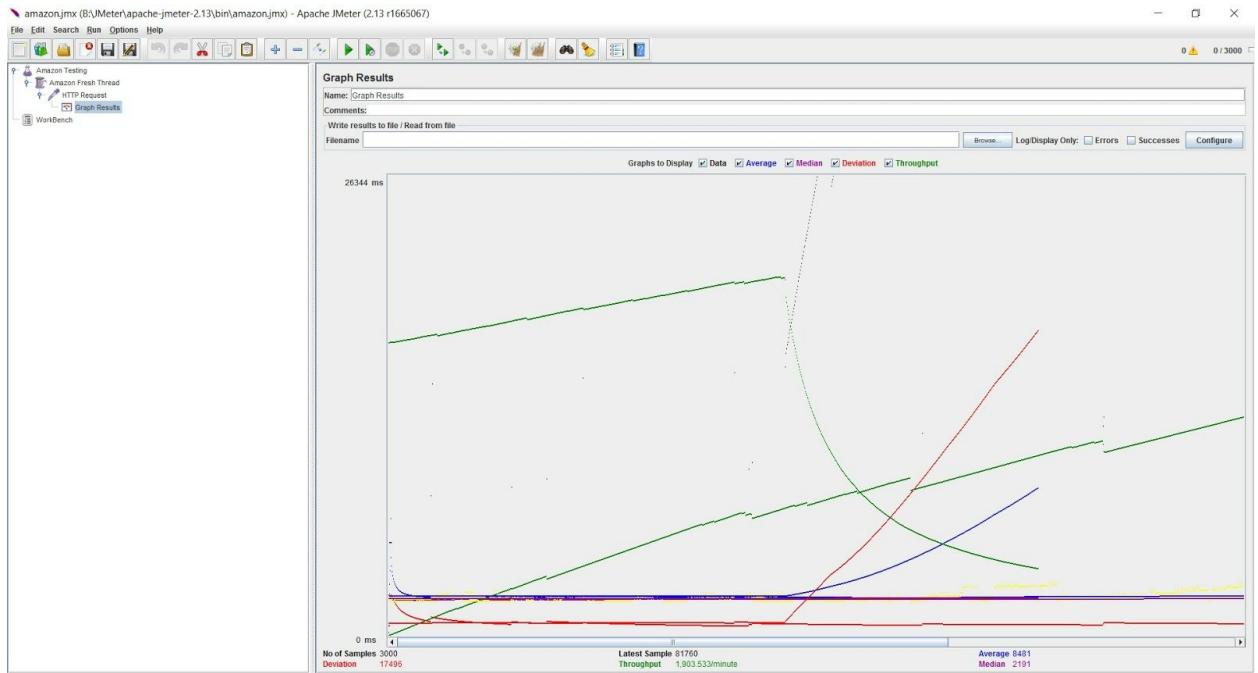


JMETER









Server Listing

```
1  /**
2   * Module dependencies.
3   */
4
5  var express = require('express')
6  , routes = require('./routes')
7  , user = require('./routes/user')
8  , http = require('http')
9  , path = require('path')
10 , farmer = require('./routes/farmer')
11 , product = require('./routes/product')
12 , login = require('./routes/login')
13 , cart = require('./routes/cart')
14 , order = require('./routes/order')
15 , truck = require('./routes/truck')
16 , driver = require('./routes/driver')
17 , farmerLogin = require('./routes/farmerLogin')
18 //ADMIN
19 , admin = require('./routes/admin');
20
21 //JUST FOR PASSPORT LOGIN
22 var passport = require('passport');
23 require('./routes/passport')(passport);
24
25
26 var mongoose = require('mongoose');
27 mongoose.connect("mongodb://localhost/amazon");
28
29 var mongoURL = "mongodb://localhost:27017/amazon";
30 var expressSession = require("express-session");
31 var mongoStore = require("connect-mongo")(expressSession);
32
33 var app = express();
34
35 // 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
```

```
//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



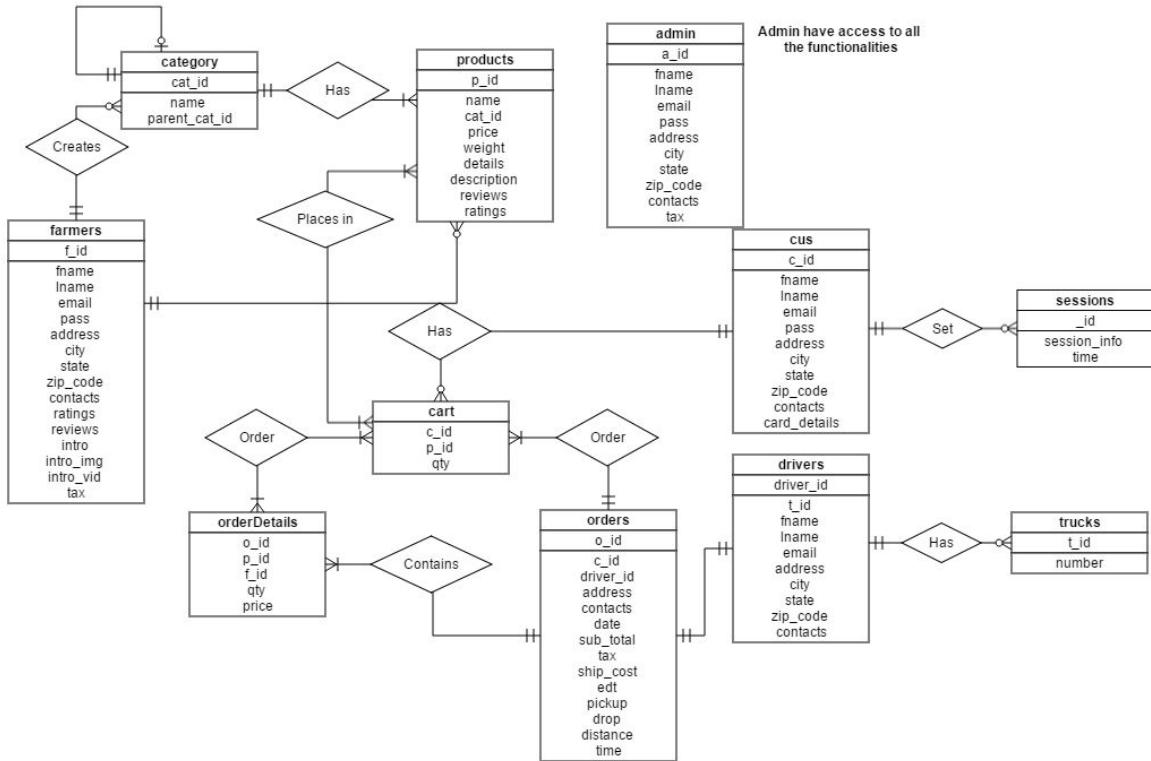
```
C:\WINDOWS\system32\cmd.exe
D:\Workspace\Amazon\Middleware\test>mocha test.js

http tests
  ✓ should return the login if the url is correct (75ms)
  ✓ should return product page
  ✓ should open search page (41ms)
  ✓ should return on user registration (40ms)
  ✓ should return suggestions based on potato

  5 passing (279ms)

D:\Workspace\Amazon\Middleware\test>
```

Database Schema



C:\Users\Vansh\Documents\Amazonnnnn\Middleware\routes\model\admin.js (AmazonFresh-master, Amazonnnnn) - 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
Amazonnnnn
Middleware
node_modules
public
routes
common
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
user/orders
Line 15, Column 18
user/orders
Find Find Prev Find All JavaScript
1 var mongoose = require('mongoose');
2
3 var autoIncrement = require('mongoose-auto-increment');
4 var connection = mongoose.createConnection("mongodb://localhost/amazon");
5 autoIncrement.initialize(connection);
6
7 var adminSchema = mongoose.Schema({
8     a_id: {type: Number, required: true, index: true},
9     fname: {type: String, required: true},
10    lname: {type: String, required: true},
11    email: {type: String, required: true},
12    pass: {type: String, required: true},
13    address: {type: String},
14    zipcode: Number,
15    city: String,
16    state: String,
17    contacts: [Number]
18 },
19     collection: 'admin',
20     timestamps: true,
21     versionKey: false
22 });
23
24 adminSchema.plugin(autoIncrement.plugin, {
25     model: 'Admin',
26     field: 'a_id',
27     startAt: 300000001,
28     incrementBy: 1
29 });

```

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\cart.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

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

cart.js

```
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

Line 1, Column 1: Detect Indentation: Setting indentation to tabs

Find Find Prev Find All Tab Size: 4 JavaScript

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\farmer.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

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
 - product.js
 - passport.js
 - truck.js
 - user.js
 - rpc
 - views
 - app.js
 - package.json

farmer.js

```
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

Line 76, Column 14

Find Find Prev Find All Tab Size: 4 JavaScript

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\category.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

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

```
1 var mongoose = require('mongoose');
2
3 var categorySchema = mongoose.Schema({
4     cat_id: {
5         type: Number,
6         required: true,
7         index: true
8     },
9     name: {
10        type: String,
11        required: true
12    },
13     parent_cat_id: {
14        type: Number
15    }
16 },
17 {
18     collection: 'category',
19     versionKey: false
20 });
21
22 var Category = mongoose.model('Category', categorySchema);
23
24 module.exports = Category;
25
```

user/orders

Line 21, Column 1

Find Find Prev Find All Tab Size: 4 JavaScript

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\farmer.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

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

```
1 var Schema = mongoose.Schema;
2 var autoIncrement = require('mongoose-auto-increment');
3 var connection = mongoose.createConnection("mongodb://localhost/amazon");
4 autoIncrement.initialize(connection);
5
6 var reviewSchema = mongoose.Schema({
7     c_id: {type: String, required: true},
8     username: {type: String, required: true},
9     rating: {type: Number, required: true},
10    review_title: {type: String, required: true},
11    review_desc: {type: String, required: true}
12 });
13
14 var farmerSchema = mongoose.Schema({
15     f_id: {type: Number, required: true, index: true},
16     isActive : {type: Boolean, default: false},
17     fname: {type: String, required: true},
18     lname: {type: String, required: true},
19     email: {type: String, required: true},
20     pass: {type: String, required: true},
21     intro: {type: String, required: true, default: "Hello!"},
22     video: String,
23     tax: {type: Number, required: true, default: 5},
24     contacts: Number,
25     address: {type: String, required: true, default: "Type Address Here"},
26     city: {type: String, required: true, default: "Your City"},
27     state: {type: String, required: true, default: "Your State"},
28     zipcode: {type: Number, required: true, default: 12345},
29 });
30
```

user/orders

Line 29, Column 25: Detect Indentation: Setting indentation to tabs

Find Find Prev Find All Tab Size: 4 JavaScript

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\order.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

```
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
```

1
2 var mongoose = require('mongoose');
3 var Schema = mongoose.Schema;
4 var autoIncrement = require('mongoose-auto-increment');
5 var connection = mongoose.createConnection("mongodb://localhost/amazon");
6 autoIncrement.initialize(connection);
7
8 var detailSchema = mongoose.Schema({
9 p_id: {
10 type: Number,
11 required: true
12 },
13 f_id: {
14 type: Number,
15 required: true
16 },
17 qty: {
18 type: Number,
19 required: true
20 },
21 price: {
22 type: Number,
23 required: true
24 }
25 });
26
27 var orderSchema = mongoose.Schema({
28 p_id: {
29 type: Number,
30 required: true
31 },
32 f_id: {
33 type: Number,
34 required: true
35 },
36 quantity: {
37 type: Number,
38 required: true
39 },
40 total_price: {
41 type: Number,
42 required: true
43 },
44 status: {
45 type: String,
46 required: true
47 },
48 created_at: {
49 type: Date,
50 default: Date.now
51 },
52 updated_at: {
53 type: Date,
54 default: Date.now
55 }
56 });
57
58 module.exports = {
59 detailSchema,
60 orderSchema
61 };

user/orders

Line 15, Column 17: Detect Indentation: Setting indentation to tabs

Find Find Prev Find All Tab Size: 4 JavaScript

C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\model\product.js (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)

```
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
```

1 var mongoose = require('mongoose');
2 var Schema = mongoose.Schema;
3 var autoIncrement = require('mongoose-auto-increment');
4 var connection = mongoose.createConnection("mongodb://localhost/amazon");
5 autoIncrement.initialize(connection);
6
7 var reviewSchema = mongoose.Schema({
8 c_id: {type:String, required: true},
9 username: {type: String, required: true},
10 rating: {type: Number, required: true},
11 username: {type:String, required: true},
12 review_title: {type: String, required: true},
13 review_desc: {type:String, required:true}
14});
15
16
17 var productSchema = mongoose.Schema({
18 p_id: {type: Number, required: true, index: true},
19 f_id: {type: Number, required: true},
20 f_name: {type: String, required: true},
21 cat_id: {
22 type: Number
23 },
24 name: {type: String, required: true},
25 price: {type: String, required: true},
26 weight: Number,
27 price_unit: String,
28 unit: String, required: true
29 });
30
31 module.exports = {
32 reviewSchema,
33 productSchema
34 };

user/orders

Line 17, Column 23: Detect Indentation: Setting indentation to tabs

Find Find Prev Find All Tab Size: 4 JavaScript

```
OPEN FILES
index.ejs x product.js x product_page.ejs x myReviews.ejs x addressDetails.ejs x user.js - Middleware\routes\model x user.js - Middleware\routes x user.js - Server\services

index.ejs x product.js x product_page.ejs x myReviews.ejs x addressDetails.ejs x user.js - Middleware\routes\model x user.js - Middleware\routes x user.js - Server\services

OPEN FILES
index.ejs x product.js x product_page.ejs x myReviews.ejs x addressDetails.ejs x user.js - Middleware\routes\model x user.js - Middleware\routes x user.js - Server\services

1 var mongoose = require('mongoose');
2 var autoIncrement = require('mongoose-auto-increment');
3 var connection = mongoose.createConnection("mongodb://localhost/amazon");
4 autoIncrement.initialize(connection);
5
6 var contactsSchema = mongoose.Schema({
7   number: {type: String, required: true},
8   default_card: {type:String, required:true, default: 'false'}
9 });
10
11 // var cardDetailsSchema = mongoose.Schema({
12 //   ,
13 //   default_card: {type:String, required:true, default: 'false'}
14 // });
15
16 var userSchema = mongoose.Schema({
17   c_id: {type: Number, required: true, index: true},
18   fname: {type: String, required: true},
19   lname: {type: String, required: true},
20   email: {type: String, required: true},
21   pass: {type: String, required: true},
22   address: {type: String},
23   zipcode: Number,
24   city: String,
25   state: String,
26   card_number: {type: Number},
27   name_on_card: {type: String},
28   exp_month: {type: Number}
29 });

Line 16, Column 22
Find Find Prev Find All
Tab Size: 4 JavaScript
```

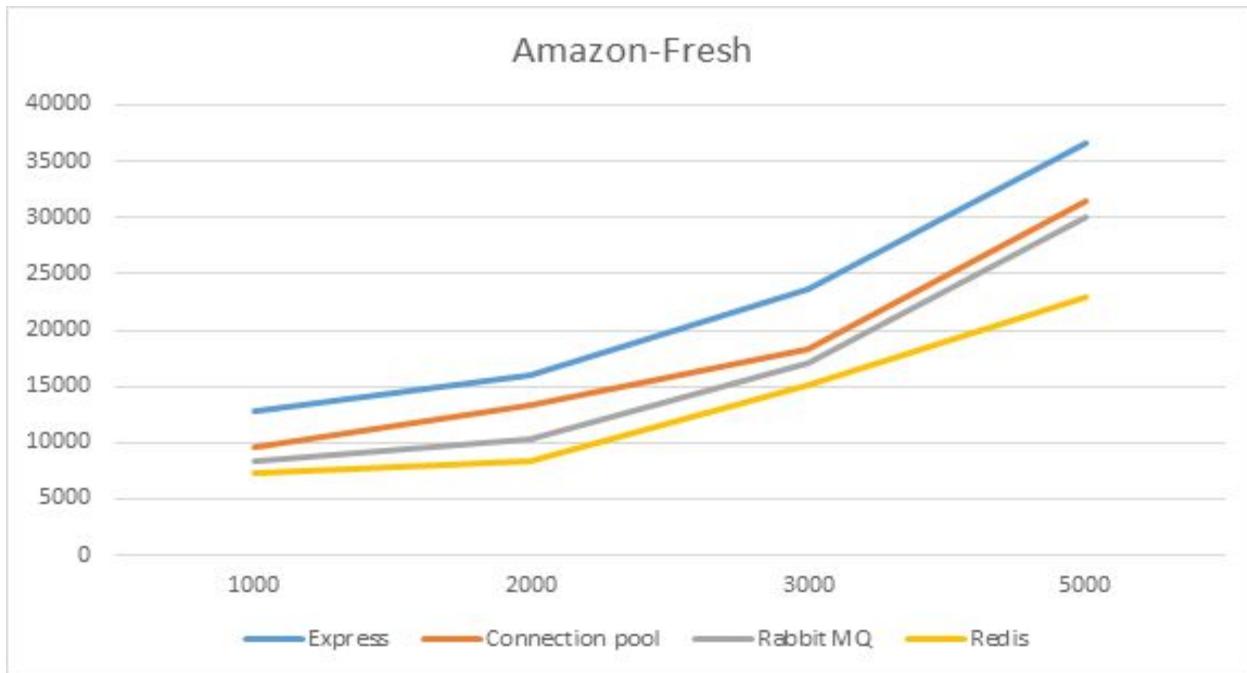
The screenshot shows the Sublime Text interface with the following details:

- File Path:** C:\Users\Vansh\Documents\Amazonnnnnn\Middleware\routes\farmer.js
- File Type:** JavaScript (AmazonFresh-master, Amazonnnnnn) - Sublime Text (UNREGISTERED)
- Open Files:** index.ejs, product.js, product_page.ejs, myReviews.ejs, addressDetails.ejs, farmer.js, 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 (highlighted), farmerLogin.js, index.js, login.js, order.js, passport.js, product.js, truck.js, user.js
- Code Content (farmer.js):**

```
index.ejs * product.js * product_page.ejs * myReviews.ejs * addressDetails.ejs * farmer.js * user.js — Middleware/routes * user.js — Server/services *
OPEN FILES
x index.ejs
x product.js
x product_page.ejs
x myReviews.ejs
x addressDetails.ejs
x user.js — Middleware/routes
x 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
Line 51, Column 21: Detect Indentation: Setting indentation to tabs
Tab Size: 4    JavaScript
```

The code in the 'farmer.js' file handles the creation of a new farmer. It starts by validating the email and password length. If either is invalid, it sends a 500 error and sets an error flag. Finally, if no errors occur, it creates a payload object with the service name.

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