**MentoringPOT** (Node Version)

**Functional Specification**

Jaycen Milling

Last Updated: March 18, 2019

**How to download/clone and run application**

1. Download
   1. Navigate to <https://github.com/jaycen9887/MentoringPOT>
   2. Click on the green button to the right “Clone or download”
   3. From the drop down select “Download ZIP”
2. If you have GIT cli tool installed in your cmd/terminal:
   1. Navigate to the directory you wish to clone this project into.
   2. Type the following: git clone <https://github.com/jaycen9887/MentoringPOT>
3. Run application
   1. Open a cmd/terminal window in the project directory and type command: npm install
      1. Note: keep this terminal open as this will be used later on.
   2. Inside the “Config” directory create a new file called “database.js” and add a connection string to your MongoDB database. For example:

module.exports = {

'url': 'mongodb://{username}:{password}@ds277965.mlab.com:77965/{database}'

}

* 1. Open a new cmd/terminal window and start Zookeeper Server
  2. Open a new cmd/terminal window and start Kafka Server
  3. Inside the cmd/terminal window from step **3a** type: node server.js
     1. Inside your browser, navigate to: localhost:3000 to start the “Customer” process (no login required) (no login required)
     2. Navigate to localhost:3000/chefs to start the “Chefs” process (no login required)
     3. Navigate to localhost:3000/admin/updateMenu to start the admin process (no login required)

**Overview**

**MentoringPOT** is a service that allows “customers” to select items from the menu (hypothetical Restaurant), review their order to make changes (if necessary), and then submit their order.

Orders are reviewed by the “Chefs” they see each order which has how many of each item each order requires.

It also allows “administrators” to update the menu, adding items, removing items, updating current items (changing the name, price, or quantities).

**These specs are not all complete or functional and will need to be updated and completed.** The screenshots shown are merely to illustrate the underlying functionality. The actual functionality, look, and feel will be developed over time.

**Scenarios**

**Scenario 1**: Customer

Customer navigates to the home page and the menu is populated. The customer looks at all the items and realizes they want to order some “Cheese Fries” as an Appetizer, “Grilled Chicken Alfredo” as the main entree, and “Tea” to drink. They click the “Review Order” button and are directed to the Review order page where they realize they accidently added two “Cheese Fries” and changed to amount from two to one. They then click the “Submit Order” button and are directed to the Submit Order page and it shows them the quantity of each item they ordered, the subtotal (the overall price of all he items added together), and the total (subtotal + 7% taxes).

**Scenario 2**: Chef

A “customer” submits an order (1 x Cheese Fries, 1 x Chicken Alfredo, and 1 x Tea) and it is pushed through to the “Chef”. The Chef sees the order which has an item name and a quantity next to each item to tell the chef how many of each item they need to make.

**Scenario 3**: Administrator

Administrator gets a shipment in to restock all the items on the menu and gets a Chicken Quesadilla (new Appetizer), Fish and Chips (new Dinner entrée), BLT Sandwich (Lunch entrée), Green Beans (a new side), and Coke (a new drink). He navigates to the Admin console where he changes the quantity of the current items to the correct amount. He then clicks the “+” button next to the Appetizers section and adds the “Chicken Quesadilla” under the item, “8.99” as the price of the item, and “6” as the quantity. Next he clicks the “+” button next to the Lunch section, he adds the “BLT Sandwich” under the item, “7.99” as the price of the item, and “10” as the quantity. The admin then clicks the “+” button next to the Dinner section and enters “Fish and Chips” under the item section, “10.99” as the price of the item, and “10” as the quantity. When the admin is finished, he clicks the “Save Menu” button and it updates the menu for when a Customer views the menu again.

**Screen by Screen Specification**

**Home Page**

The home page is the Producer and Consumer side of the Kafka Menu topic.

It Consumes the Menu topic to create the menu in the UI for the User to select certain items.

When the user has selected the items it wishes to “order” they can press the “Review Order” button that will take them to the Review Page.



**Review**

The user is taken to this screen to review their order, they can add, remove, and change the amount of the items they want. Then the hit the “Submit Order” button that will direct the user to the “Order Sbmitted” page (essentially a receipt), but also push the order to the Order topic in kafka.



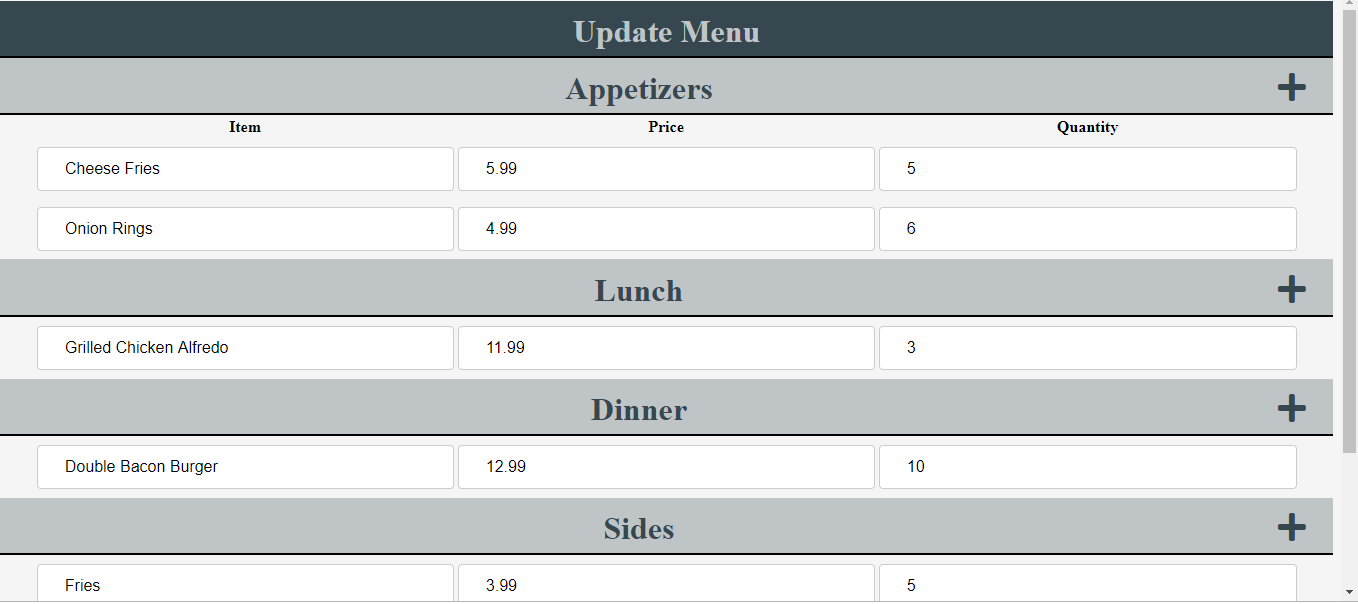
**Order (Submitted)**

This screen is more like a receipt for the Customer, it tells the user how many of each of the items they ordered, subtotal, and Overall total of their order.



**Admin (Update Menu)**

Allows the admin to update the “Item” (name), “Price”, and “Quantity” as well as adding items under each category or deleting items.



**Technical Specs**

**Technologies Utilized**

1. **NodeJS**
   1. Dependencies Utilized
      1. Express – Web application framework
         1. Create an express app variable
            1. Set up app to use body-parsers ‘urlencoded’ function
            2. Set up app to use body-parsers ‘json’ function
            3. Set up apps static folder
            4. Set up apps view engine to ‘ejs’
            5. Set app to listen on a specific port
      2. MongoClient
         1. Require the database config file that contains the connection string
         2. Use the mongoclient connect function passing the database connect string, { useNewUrlParser: true} to use the latest version of the connection string, and a call back function.
      3. Body-Parser – Express Middleware that parses HTTP request body.
      4. EJS – **E**mbedded **J**ava**S**cript templating language
      5. Kafka-Node – Is a Node client for Apache Kafka.
         1. Used to communicate with Kafka server (running on local machine for testing purposes)
2. **Kafka**
3. **MongoDB**

**External Interface Requirements**

1. **User Interfaces**
2. Front-end software: NodeJS version
3. Back-end software: ExpressJS, Kafka, and MongoDB
4. **Hardware Interfaces**
5. A browser which supports HTML and Javascript
6. **Software Interfaces**
7. Developed in Windows utilizing Visual Studio Code.
8. NodeJS for the Javascript runtime engine.
9. MongoDB as a backup of Kafka.
10. Kafka real-time topic monitoring.
11. **Communications Interfaces**
12. This project supports all types of web browsers.