**Final Project Submission**

COSC612/AIT624: Software Engineering Fundamentals

2023 Fall Semester

Formula 1 LLC

* Oladiran Apara
* Cara Galingana
* Devere Weaver
* Swadha Bhatt
* Victoria George

December 4, 2023

[**Section 1. Final Problem Statement** 3](#_Toc152600982)

[I. Background 3](#_Toc152600983)

[II. Business Need 3](#_Toc152600984)

[III. Changes 4](#_Toc152600985)

[**Section 2. Front End** 4](#_Toc152600986)

[I. Front End Implementation 4](#_Toc152600987)

[II. Website Storyboard and Interface Structure Diagram 5](#_Toc152600988)

[**Section 3. Back End** 13](#_Toc152600989)

[I. Database Specification and Analysis 13](#_Toc152600990)

[II. Database Implementation 14](#_Toc152600991)

[**Section 4. API** 15](#_Toc152600992)

[**Section 5. Teamwork** 17](#_Toc152600993)

[**Section 6. Future Work** 17](#_Toc152600994)

**Table of Contents**

# **Section 1. Final Problem Statement**

## I. Background

Formula 1 LLC aims to alleviate the burden of micro to small-scale business in managing ordering workflows without a system that can track inventory status alongside customer demands. The manual processes currently in practice by such businesses lack efficiency, wastes resources, and are not responsive to dynamic customer demands.

These businesses require appropriate software development, system administration and business stakeholders’ collaboration to evaluate solutions that can help accomplish its business goals and grow market share.

## II. Business Need

F1 Coffee Roasters is a company that sells coffee beans in packages to commercial clients. This small-scale business needs an inventory management system to efficiently manage stock levels, streamline ordering processes, improve customer service, maintain business-supplier connections, and generate reports for informed business decision-making.

If the inefficiencies are not resolved, F1 Coffee Roaster may be unable to retain customer and supplier relationships. Moreover, its market value can be at risk and may have a difficult existence in a very competitive business environment.

*Top-level objectives:*

F1 LLC aims to create a system that automatically sends an order to a supplier whenever the minimum threshold of the inventory stock is reached. By integrating into a point-of-sale system, it enables accurate tracking of stock consumption. Achieving this objective will improve supply chain management and reduce errors from manual stocking processes.

*Who is it for:* The product is for micro to small-scale business, such as the F1 Coffee Roasters.

*What problem does it solve:*This system will improve inventory accuracy, reduce operational costs, and enhance overall business productivity. Inefficient inventory management practices plague businesses, leading to overstocking, understocking, operational inefficiencies, and increased costs. Manual record-keeping and disjointed inventory processes hinder the ability to make data-driven decisions. The absence of a centralized, interactive inventory system limits businesses from optimizing stock levels, responding to market demands, and reducing operational complexities. The need for an innovative solution that seamlessly tracks, manages, and reports on inventory data in real-time is paramount for businesses striving to enhance accuracy, reduce operational expenses, and elevate overall productivity.

*Differentiators:*The biggest differentiator of our inventory management systemis it automates supplier ordering workflows through efficient tracking of stock inventory status and client order consumption.

*Scope of product*: The scope of the inventory management system includes streamlining the inventory managing workflows of micro to small-scale businesses.

## III. Changes

During the project development, some changes to the problem statement were made to reduce the scope of the final project. From the first sprint, we visualized a system that can demand forecasting of customer demands. However, moving forward, we decided to focus on the stock optimization feature, wherein an employee is able to indicate a product’s minimum and maximum stock threshold for the system to assess whether an order to the supplier needs to be placed immediately. Real-time notifications and alerts were also removed in the project, but logic of restocking remains.

To further focus on stock optimization and inventory management, we removed login authentication, user privileges, and other methods involving access management. Reports were reduced to client orders, stock orders, and product information. The project no longer includes the capability of generating reports for users, nor does it maintain a list of approved suppliers.

Lastly, we have decided to include the point-of-sale system in the development of the project, instead of integrating with a third-party system like Oracle Micros Simphony. This way, we are able to fully test the life cycle of inventory management, instead of worrying about compatibility and other limitations of integration.

On our latest update, we decided to implement a chatbot that could answer relevant questions regarding coffee processing, roasting, origins, and many others. This is helpful for the user to quickly get information regarding business processes during working hours.

# **Section 2. Front End**

## I. Front End Implementation

The front-end of the application was implemented on Visual Studio Code using standard HTML, CSS, and JavaScript (Express JS). Express JS framework was very useful in implementing functions that needed to be dynamic, such as updating the stock quantity when an order is fulfilled or when a restock is prompted. By using Express JS’ view engine feature, it renders dynamic content on the server side before sending it to the client.

In this code example, <div><%= quantity %></div>, the tags allow JavaScript code to be directly embedded into the HTML templates. The value of the quantity variable will be dynamically inserted into the HTML.

Express JS also provides an easier way to route different URL paths and render views by combining the template with the data.

## II. Website Storyboard and Interface Structure Diagram

1. Home/ Products Page

A screenshot of a computer

Description automatically generated

Navigation bar on the Home/Products page

The Home/Products Page displays a navigation bar (Products, Client Orders, Stock Orders) for the user to easily navigate through the main tables of the inventory management system. This page displays the products on the system, as well as its attributes. The action buttons on the right side of rows (Edit and Order) correspond to each product and would redirect to the Edit Product page or the Fulfill Client Order page.

A screenshot of a computer

Description automatically generated

Action buttons corresponding to each product on the Home/Products page

This page also displays the chatbot icon where the user can ask relevant questions regarding coffee processing, roasting, origins, and many others related to the coffee roasting workflow.

A screenshot of a computer

Description automatically generated

Chatbot icon on the Home/Product page

A screenshot of a computer

Description automatically generated

Chatbot responding to user’s question

2. Add Product page

The add product page displays a blank Product Specification form where the user enters the information of a product that will be added to the system. These attributes include Product Name, Barcode, Roast Category, Retail and Wholesale Price, Initial Quantity, and the Maximum and Minimum threshold values. Placeholders are visible on the form for the user to know the expected values accepted by the system.

A screenshot of a computer

Description automatically generated

The submit button is clicked to add a product on the system

On this page, the user is able to set the product threshold values using the quantity sliders ranging from 0 to 100. This limits the user from entering values that are out-of-range (negative values, values greater than 100).

A screenshot of a computer

Description automatically generated

Maximum and Minimum Threshold Range Sliders allowing values from 0 to 100

An error message is displayed above the product specifications form when the entered minimum threshold is greater than the maximum threshold.

A screenshot of a computer

Description automatically generated

Error Message displayed when entered minimum threshold value is greater than the maximum

3. Edit Product page

The Edit Product page displays the information retrieved from an added product. The user is directed to this page when the Edit button corresponding to the product row is clicked from the Home/Products page. The user may also set the threshold values on this page, as seen on the Add Product section. Aside from the threshold values, the user is also able to edit all the other attributes of the product. The same conditions of accepted values apply.

A screenshot of a computer

Description automatically generated

The Edit Product button is clicked to add a product on the system

4. Fulfill Client Order page

The Fulfill Client Order displays a form which allows the user to fulfill an order of a specific product. The user is directed to this page when the Order button corresponding to the product row is clicked from the Home/Products page. The Product ID and current stock quantity is retrieved. The user enters an order quantity on the form, which will be reduced from the stock quantity when the Fulfill Order button is clicked. The current stock quantity on the Products page will then be updated.

A screenshot of a computer

Description automatically generated

The Fulfill Order button will reduce the stock quantity according to the order quantity

In this example, we will order 15 units from the stock quantity of 15. When the user clicks Fulfill Order, the website redirects to the Products page, where the current quantity will be updated to 0.

A screenshot of a computer

Description automatically generated

The current quantity on the Products page has been updated

An error message also appears when the user enters an invalid order quantity value.

A screenshot of a computer

Description automatically generated

Error message displayed prompting the user to enter a valid order quantity value

5. Client Orders Page

The Client Orders Page displays a table of orders fulfilled by the user. Attributes of each Client Order are Client Order ID, Product ID, Client Order quantity, and the date it was ordered. From the order fulfilled in the above example, we can see that the client order quantity reflects the 15 units reduced from the current stock quantity.

A screenshot of a computer

Description automatically generated

Client Orders Page displaying the order fulfilled and its attributes

6. Stock Orders Page

The Stock Orders Page displays a table of stock orders automatically placed by the system once the current stock quantity reaches the minimum threshold. The value displayed on the stock order is the amount of units that will fill the product’s maximum threshold capacity. In the example above, the minimum threshold capacity was reached when the current stock quantity was updated to 0. This prompted the system to automatically place a stock order of 100, which is the product’s maximum threshold capacity.

A screenshot of a computer

Description automatically generated

Stock Order page displaying the stock order quantity needed to reach the maximum threshold

Once the user “receives” the ordered quantity, the user can click on the Restock button to add the ordered quantity to the current stock quantity.

A screenshot of a computer

Description automatically generatedThe Restock button will update the current stock quantity with the added stock order quantity

A screenshot of a computer

Description automatically generated

The current stock quantity of the product has been replenished to its maximum threshold capacity of 100

# **Section 3. Back End**

## I. Database Specification and Analysis

The tables for the following entities were implemented:

|  |  |  |
| --- | --- | --- |
| Entity No.1 | | Product |
| Description | Table that contains information on the products | |
| Attributes | ProductID (Primary Key, String) - a unique identifier for each individual product in the products table, automatically generated | |
|  | Product Name (String) – product’s name, must be at least 3 characters | |
|  | Barcode (Number) - the barcode used for the item | |
|  | Category (String) - the category of products the item belongs to, valid values are: Light Roast, Medium Roast, Dark Roast | |
|  | Retail (Number) - how much the product sells for | |
|  | Wholesale (Number) - how much the product was purchased for | |
|  | Quantity (Number) – the initial amount of the product when the product is added to the database | |
|  | Projected Stock Quantity (Number) - the amount of the item currently in stock | |
|  | Max Stock – The maximum capacity the inventory can hold for this product | |
|  | Min Stock – The minimum threshold set for this product; When value is reached, an automatic restock order will be placed | |

|  |  |  |
| --- | --- | --- |
| Entity No.2 | | Client Order |
| Description | Table that contains information on fulfilled orders | |
| Attributes | Client Order ID (primary key, String) - unique order identifier to be used as primary key, automatically generated | |
|  | Client Order Quantity (Number) - the total number of products fulfilled | |
|  | ProductID (foreign key, String) - the unique product identifier for each product in the order | |
|  | Date (timestamp) – Date and time when order was fulfilled | |

|  |  |  |
| --- | --- | --- |
| Entity No.3 | | Stock Order |
| Description | Table that contains information on fulfilled orders | |
| Attributes | Stock Order ID (primary key, String) - unique order identifier to be used as primary key, automatically generated | |
|  | Stock Order Quantity (Number) - the total number of products ordered for restock | |
|  | ProductID (foreign key, String) - the unique product identifier for each product in the order | |
|  | Date (timestamp) – Date and time when order was fulfilled | |

The entity-relationship diagram was also refined accordingly.

A diagram of a product

Description automatically generated

ER Diagram of F1 Inventory Management System

## II. Database Implementation

Table were implemented on MongoDB. Using MongoDB Compass and MongoDB Atlas, a proper database was integrated into the web application.

Since MongoDB is a NoSQL database, the relationships of classes were established in the construction of each model, where the foreign key is cited as a model reference. For example, in the code of constructing a stock order, the product ID was used as a foreign key, referencing the ‘product’ class (see photo below).

A screen shot of a computer program

Description automatically generated

The local server acts as the primary server for testing functionality that interacts with the database. The MongoDB Compass functions as the application’s live database, which will be updated when each database route is implemented and tested for correctness.

A screenshot of a computer

Description automatically generated

Screenshot of MongoDB Compass showing the classes and its values

We changed our previous database from PostgreSQL to MongoDB, due to difficulties in installation on developers’ workstations. MongoDB will also be useful in this project due to the frequent schema changes that may be required. This allows insertion of data into the database without needing to alter the fixed schema.

# **Section 4. API**

Express is the web application framework that is used to connect the front end and the back end. It is a very common and lightweight framework often used for web applications that use Node.js as the backend technology. It has a simple API that exposes Middleware functions. These are functions that help facilitate the response-request cycle that is typical of web applications to host dynamic web pages.

The templates engine in Express is also used to transform static template files to dynamic HTML files that are then forwarded to the client after being updated and served from the backend.

To ensure that our development server was able to properly communicate with both the database and the frontend, we implemented two of our use cases. Both use cases require manipulating the databases via form information from the frontend. To test that these use case implementations are working correctly, an Express.js route was created. This route reads the HTML form data, parses it, and then uses the data to the connected database to update the required tables.

A screenshot of a computer

Description automatically generated

MongoDB live deployment

A screen shot of a computer program

Description automatically generated

Code for route to update the Product table

The front and back ends of the project communicate any time an update is made to the product database. One example is when editing an order. A user completes the HTML form with the desired data for a given product. The HTML form data is parsed, and the requisite information is formatted then sent to the database.

Another example of communication between the two sides is when an order is placed. Again, the user completed the HTML form which is parsed then sent to the database. The necessary changes are made in the database and then this updated information is served back to the front end to update the displayed values.

As for the chatbot, it was created using Chatbase's chatbot API, https://docs.chatbase.co/docs. The large language model used was built on top of OpenAI's ChatGPT. It was then added to the index page of the web application as a floating element to facilitate chat functionality for the user.

## **Section 5. Teamwork**

The first example where we had a difficult situation regarding terms of coordination, was trying to figure out times and dates for team meetings. As graduate students most of us are not only full-time students, but also have other jobs and other responsibilities. As a team we had to respect that everyone had different priorities that had to be fulfilled, however we also understood that we had to make this class a priority. Prioritizing this class meant that some of us couldn’t go out on a Friday night in order to attend weekly team meetings, so that the sprints deliverables could be successfully completed on schedule. When we faced difficult situations regarding coordination of times and dates for weekly meetings, we as a team tried to be understanding of each other’s schedules but at the same time also change our schedules so that we can all attend the decided time and date for the weekly meetings.

An example of the situation where we had differences in perspective was in the beginning sprints where we were deciding on how the future product would function, and what kind of classes and attributes would be in the system. We initially had different perspectives on where to take the project. We came to a consensus by voting on each of our perspectives, and we also tried to evaluate if the idea that a team member had proposed was technically feasible for us to complete in one semester. With the help of a democratic process and a little bit of logistical flexibility, the team came to a decision on the final potential design for our inventory system.

# **Section 6. Future Work**

There are several considerations for future work considering what was delivered is the minimum viable product. One consideration for future work is to make improvements to the chatbot that is available in the current version. As of now the chatbot utilizes the ChatGPT language model which is very useful for general tasks; however, the specific implementation of the chatbot was given a few PDFs containing information related to our product to train it. The performance is adequate but can be improved by training it more to create a good knowledge base related to the product. It can also potentially be replaced by OpenAI's new Assistants API that is used to create virtual assistants. This can help narrow down the focus on the chatbot to tasks that are specific to the application instead of being a more generalized chatbot.

Originally, the application was envisioned to record data and provide analytics that were useful for optimizing stock inventory and providing time series analysis and prediction; however, due to time and technological constraints, this feature had to be cut from the project. Future development of this product should include these features as those could give the inventory system an edge in the market.

Even though we as a team came up with a great inventory system, given the timeframe constraint and our own technical capabilities constraints. There are a lot of changes in the future that we can implement to make sure that our inventory system is not only efficient, the best but also economical and market leading. One of the first things we could implement is a forecasting function for the sales. A lot of inventory systems in the market tend to have some sort of forecasting function for their clients and our inventory system lacks that function. If we add a sales forecasting function by using machine learning algorithms, then our application would be more marketable, and it also would be more appreciated by our clients as they would be able to get a deeper insight into their business processes.