

A Midterm Progress Report
on
INVENTORY MANAGEMENT SYSTEM
FOR FMCG INDUSTRY

**Submitted in partial fulfillment of the requirements for the award of
the degree of**

BACHELOR OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

DILRAJ SINGH

2104092

EKUSPREET SINGH

2104097

HARSIMRAN SINGH

2104117

UNDER THE GUIDANCE OF

PROF. SITA RANI

(MARCH-2024)



Department of Computer Science and Engineering
GURU NANAK DEV ENGINEERING COLLEGE,
LUDHIANA

INDEX

S.No.	Title	Page No.
1	Introduction	1-2
2	System Requirements	3
3	Requirement Specification	4-12
4	System Design	13-19
5	System Testing	20-22
6	Output Screens	23-25
7	Performance of the project developed	26-27
8	Reference	28

1. Introduction

1.1. Introduction to Fast Moving Consumer Goods Industry

In the landscape of Fast-Moving Consumer Goods (FMCG) Industry, high turnover and volume sales are significant for the success of the organization. The FMCG sector, currently positioned as the fourth-largest contributor to the Indian economy, is characterized by its relentless pace of production and distribution. FMCG includes an array of everyday products, ranging from consumables like foods and beverages to essential household and personal care items. To attain stability, the role of efficient inventory management is significant for any FMCG industry.

1.2. Challenges faced by FMCG Industry

FMCG companies face challenges in keeping up with the speedy demand, managing production, and handling inventory. If things go wrong, there can be too much or too little of a product, causing problems in how things run. Knowing exactly how much stuff to have in the warehouse is one of the biggest challenges.

Other challenges include management of Employees as FMCG companies often have large teams of people working together. If employees aren't managed well, it can lead to confusion and a decrease in overall productivity.

FMCG companies rely heavily on various raw materials to make their products. A consistent supply can only be ensured by monitoring their quantities.

FMCG products often require specific recipes or formulas to maintain consistent quality. Managing these recipes, ensuring that they are followed accurately, and incorporating any changes or innovations can be challenging. If there are errors in the production process, it can lead to variations in product quality.

1.3.Introduction to our project

This project aims to address the pressing challenges faced by Natural Animal Diets (a company in the FMCG Industry) through the implementation of a Digitized Inventory Management System. The project is sponsored by them and addresses a specific area of concern within the FMCG sector that is the production of Animal Fodder. The project titled, IMS, centralizes Raw Material data and Alerts teams when stock falls below predefined thresholds. Employee data, such as qualifications and roles, are stored securely in IMS. This information aids in workforce planning and task allocation. IMS monitors production cycles from raw material intake to finished goods. It schedules production runs, allocates resources(labor), and ensures definite completion. Deviations trigger alerts for corrective action. Animal fodder recipes are critical. IMS securely stores these recipes, ensuring consistency across batches. It also calculates material requirements based on recipe proportions. Furthermore, IMS can be adapted to other FMCG contexts such as **Food and Beverage Industry** by managing perishable ingredients, packaging materials and **Household Cleaning Products** by managing chemical ingredients, safety data sheets, and production cycles.

1.4.Objectives of the project

Upon completion, the project aims to achieve the following objectives:

- To ensure convenient management of inventory.
- To manage and store employee data.
- To mitigate inventory shortages through an alert system.

2. System Requirements

2.1. Hardware Requirements

- 2.1.1. Architecture:** A compatible system architecture to compile and run the project (e.g., x86, ARM, etc.).
- 2.1.2. RAM:** A RAM of at least 4 GB(s) is required for the smooth and uncompromised development of the project.
- 2.1.3. Storage:** At most 500 MB(s) of storage is required to safely develop without any storage issues.
- 2.1.4. Database:** The project utilizes MongoDB cloud database for securely storing and managing data. The premium subscription of the database will be required to make the project suitable for production.
- 2.1.5. Server:** A reliable server with low latency and higher response rate. A freemium server is being used for development stage.

2.2. Software Requirements

- 2.2.1. Browser:** Following browser versions are supported: Chrome ≥ 87 . Firefox ≥ 78 . Safari ≥ 14 .
- 2.2.2. Operating System:** Any operating system that can run a browser instance with a version above than aforementioned ones.
- 2.2.3. IDE:** An IDE such a Visual Studio Code is needed for the development, testing and deployment.
- 2.2.4. Virtual Environment:** A python virtual environment is required to manage python libraries.
- 2.2.5. Version Control System:** A version control system like Git is required to manage and control project versions. Currently the codebase is hosted on GitHub.

3. Software Requirement Analysis

3.1. Problem Definition: The problem at hand is to design a software that can enhance efficiency and resource allocation by centralizing raw material data, streamlining production cycles and allowing adaptability to other FMCG contexts.

3.2. Purpose and Scope: The project's purpose include automation to transform traditional operations, allowing employees to focus on value-added tasks rather than repetitive manual processes.

The initial scope will be within the domain of Animal Fodder production, with scalability considerations for potential future expansion into other FMCG sectors. The plan is to set things up so that if NAD wants to use this system for other FMCG (Fast-Moving Consumer Goods) areas in the future, it can easily be done. The project is kept simple and practical, looking at what's needed now and what might be needed later.

3.3. Functional Requirements: In order to achieve the goals of the IMS (Inventory Management System) project certain functional requirements have been identified

3.3.1. User Authentication: The user authentication module is a fundamental component of the IMS, designed to implement a robust authentication mechanism, such as password-based or multi-factor authentication and ensure a secure login.

This functionality aims to restrict access to the system, allowing only authorized personnel to use different features based on their roles within Natural Animal Diets (NAD).

In the context, Admin role can access multiple information sources meanwhile an employee would not get access to the business details. This is implemented while authentication.

3.3.2. User Role Management: Establish a role-based access control system where each user is assigned specific roles determining the features they can access. The Administrator, or Admin, holds the highest level of access within the system. They have the authority to manage user accounts, configure system settings, and have unrestricted access to all features.

The Manager role is an intermediary level of access, providing enhanced privileges compared to employees but with restrictions compared to administrators. Managers oversee specific aspects of the production process.

The Employee role is designed for general users involved in the production process. Their access is limited to features essential for their day-to-day responsibilities, promoting operational efficiency. Employees accounts can only be created by Administrator

3.3.3. Password Encryption: Employ strong hashing algorithms (currently SHA-256) to securely store and transmit passwords, protecting user credentials from unauthorized access. HTTP responses are sent using JWT (Json Web Token).

3.3.4. Dashboard: A user-friendly dashboard displaying key metrics, such as raw material inventory levels, production status, and alerts for inventory shortages to allow user to have a quick overview of essential information, facilitating informed decision-making.

3.3.5. Raw Material Management:

- Users, like admins or designated staff, can easily add new raw materials with details like name, quantity, and supplier info.
- Editing existing details ensuring real-time updates.
- The system provides instant updates on raw material levels, allowing

users to stay informed.

- For safety, a secure process for authorized deletion is in place, preventing accidental removal.

3.3.6. Employee Management:

- Functionality to add new employees and give them access to the application.
- Admin can view/modify employee's data.
- The system stores employee details with security protocols in place.
- Ability to set roles and allow employees to see the relevant jobs.

3.3.7. Production Cycle Monitoring:

- The system presents a simple and intuitive interface for users to easily visualize the different stages of the production cycle.
- Users receive instant updates on the progress of each production stage
- Users, including employees and managers, can effortlessly update the system with the current status of production cycles, promoting real-time data accuracy.
- Tight integration with the raw material management module ensures that users can correlate production progress with raw material availability, minimizing disruptions.

3.3.8. Storage of Recipes:

- The system establishes a centralized and secure repository dedicated to storing production recipes, ensuring a centralized location for authorized users to access and manage these essential documents.
- Only authorized users, such as administrators or designated personnel, have the permission to upload, edit, and access production recipes. This

control mechanism safeguards the integrity of the recipe database.

- As an additional layer of security, the system employs encryption protocols to safeguard the confidentiality of production recipes mitigating the risk of sensitive information exposure and maintaining the integrity of the production process.

3.3.9. Alert System:

- The alert system is equipped with predefined triggers that automatically generate alerts when raw material levels fall below a specified threshold. These triggers are configured to promptly identify potential shortages, ensuring timely notifications to relevant stakeholders.
- The alert system allows for customizable parameters ensuring that alerts align with the company's production dynamics.
- Alerts can be configured to be delivered through multiple channels, such as email notifications.

3.4. Non-Functional Requirements:

3.4.1. Performance: The system should provide responsive and efficient performance, ensuring quick response times for user interactions and minimal latency in data processing. Response time for common operations should be within 5 seconds, even under peak loads.

3.4.2. Scalability: The system should be designed to handle an increase in data volume, user base, and concurrent transactions without significant degradation in performance.

3.4.3. Reliability: The system should be highly reliable, minimizing the occurrence of system failures, crashes, or unexpected downtimes.

3.4.4. Maintainability: The system's codebase and architecture should be designed in

a modular and well-documented manner, facilitating ease of maintenance, updates, and bug fixes.

3.4.5. Data Backup and Recovery: Regular automated backups of system data should be performed, and a robust recovery plan should be in place to restore data in the event of data loss or system failure.

3.5. Definition of Modules and Functionality:

3.5.1. Front End Modules:

3.5.1.1. React: React is a JavaScript library for building user interfaces, allowing developers to create dynamic and interactive web applications with a modular and efficient component-based architecture. React uses a virtual DOM to optimize rendering performance. Changes to the virtual DOM are compared to the actual DOM, and only the necessary updates are applied, reducing the number of direct manipulations to the real DOM. JSX is a syntax extension for JavaScript that allows embedding HTML-like code within JavaScript.

3.5.1.2. React Router v6.0: React Router is a standard library for routing in React applications, allowing developers to create dynamic and navigable user interfaces. React Router uses components such as `BrowserRouter` to wrap the entire application, `Routes` to store all the routes, `Route` to create a route, `Link` to connect a route to a clickable button and `useNavigate` to create dynamic redirection.

3.5.1.3. Vite: Vite is a build tool for modern web development that aims to provide a fast and efficient development environment. It is designed to optimize the development experience by using native ES modules (ESM) and other modern JavaScript features. Vite supports a plugin system, enabling developers to extend and customize the build process. Vite includes built-in

support for creating Progressive Web Apps, simplifying the process of adding service workers and manifest files.

3.5.1.4. Tailwind: Tailwind CSS is a utility-first CSS framework that streamlines the process of styling web applications. Unlike traditional frameworks with pre-designed components, Tailwind provides low-level utility classes for building responsive designs directly in your markup. It emphasizes flexibility, rapid development, and easy customization.

3.5.1.5. Daisy UI: It is one of the most popular, free and open-source component libraries for Tailwind CSS. It is used to create standardized designs.

3.5.1.6. Axios: Axios is a popular JavaScript library used for making HTTP requests from web browsers and Node.js environments. It provides a simple and consistent API, making it easier to work with asynchronous data and interact with web servers. Axios is built on the Promise-based architecture, allowing developers to utilize the `async/await` syntax for handling asynchronous operations. Axios automatically parses JSON response data, simplifying the process of working with JSON APIs. It also allows sending JSON data in the request body without manual serialization.

3.5.2. Back End Modules:

3.5.2.1. Flask: Flask is a lightweight and versatile web framework for Python, designed to make web development straightforward and flexible. Known for its simplicity and ease of use. Its modular design allows developers to choose the components they need, making it scalable for both small projects and larger, more complex applications. Flask follows the WSGI (Web Server Gateway Interface) standard, making it compatible with various web servers. Flask's is one of the most popular among developers seeking an efficient and

customizable framework for web development in Python.

3.5.2.2. Flask-CORS: Flask-CORS is a Flask extension that simplifies Cross-Origin Resource Sharing (CORS) in Flask applications. CORS is a security feature implemented by web browsers to restrict web pages from making requests to a different domain than the one that served the original web page. When developing web applications, especially those with separate frontend and backend components hosted on different domains, CORS issues can arise. The configuration of Flask-CORS is flexible, allowing developers to define the specific origins, methods, headers, and other CORS-related settings tailored to their application's requirements.

3.5.2.3. JWT-Extended: Flask-JWT-Extended is a Flask extension that enhances the Flask ecosystem by providing robust support for JSON Web Tokens (JWT) in web applications. JWT is a compact, URL-safe means of representing claims between two parties, commonly used for secure and stateless authentication. Flask-JWT-Extended simplifies the integration and management of JWT-based authentication and authorization mechanisms in Flask applications.

3.5.2.4. Python Dotenv: The process of using environment variables from a .env file in a Python project typically involves using a library called python-dotenv. This library loads the environment variables from a .env file into the application's environment. The .env file is often used to store configuration settings for an application, such as database connection strings, API keys, and other environment-specific parameters. This allows developers to easily configure the application for different environments (development, testing, production) without modifying the source code.

3.5.2.5.Passlib: Passlib is a Python library for securely hashing passwords and managing password-related tasks. It provides utilities for generating secure password hashes, verifying passwords, and managing password policies, making it easy to implement secure password authentication in applications. Passlib supports multiple hashing algorithms, including bcrypt, SHA-256, and SHA-512, and provides features such as password hashing, salting, and stretching to enhance security. With Passlib, you can build web applications that protect user passwords and sensitive information from unauthorized access and data breaches.

3.5.3. Database Modules:

3.5.3.1. Mongo DB: It is an open-source, document-oriented database designed for flexibility, scalability, and ease of development. Unlike traditional relational databases, MongoDB does not rely on the tabular structure of rows and columns. Instead, it stores data in BSON (Binary JSON)-like documents, allowing for a more natural representation of complex, hierarchical structures. MongoDB is horizontally scalable, allowing seamless distribution of data across multiple servers. It supports sharding, enabling high-volume and high-throughput applications to scale easily. MongoDB ensures high availability through features like replica sets. Data is replicated across multiple servers, allowing automatic failover in case of hardware failure or other issues and provides document validation to enforce data integrity. Developers can define rules to validate documents, ensuring that data adheres to specific constraints.

3.5.3.2. PyMongo: PyMongo is the official Python driver for MongoDB. It provides tools and utilities for interacting with MongoDB databases, including querying data, inserting and updating documents, and performing

administrative tasks, making it easy to work with MongoDB in Python applications. PyMongo follows the MongoDB API and provides a high-level API for common database operations, as well as a low-level API for advanced database manipulation. With PyMongo, robust and scalable web applications can be built using MongoDB.

4. Software Design:

4.1. Abstract System Design:

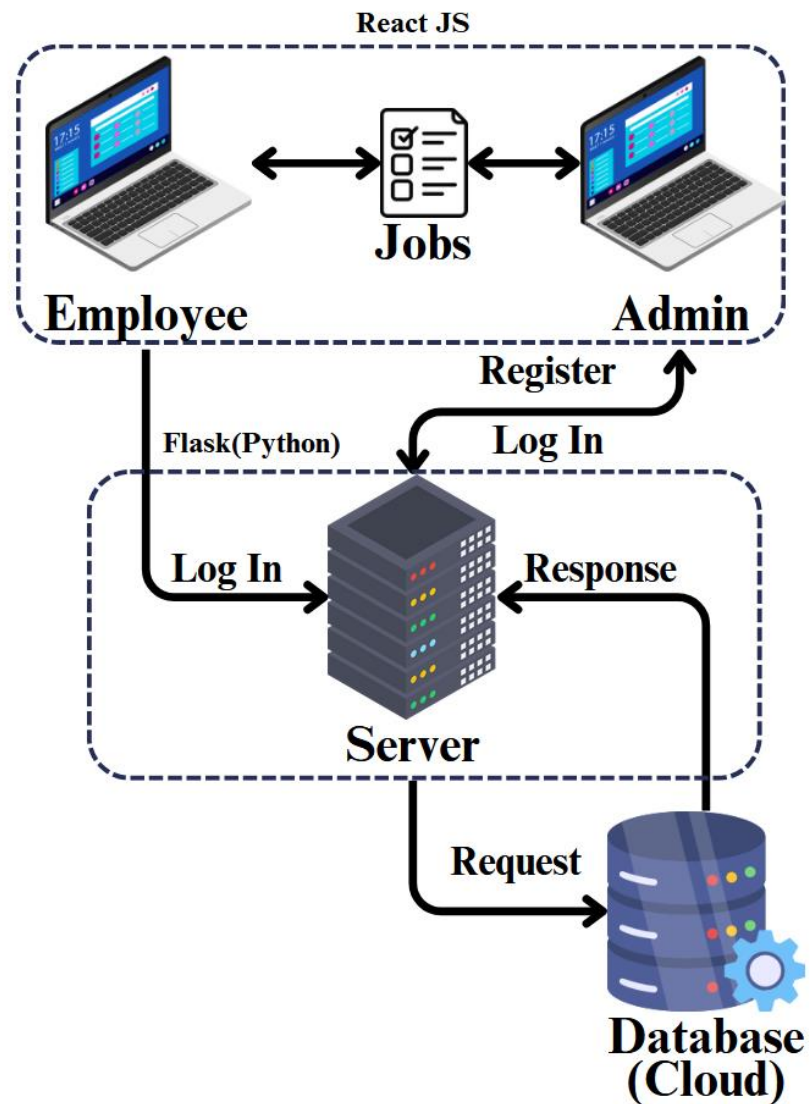


Figure 1 : A High-Level Abstraction of Application

4.2.Employee Flowchart:

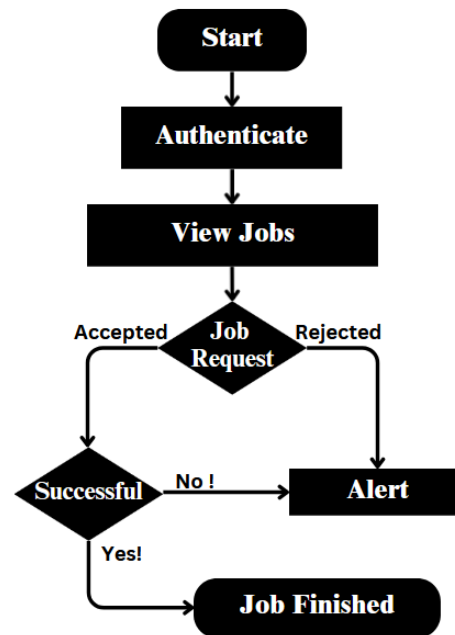


Figure 2: Employee View Flowchart

4.3. Administrator Flowchart

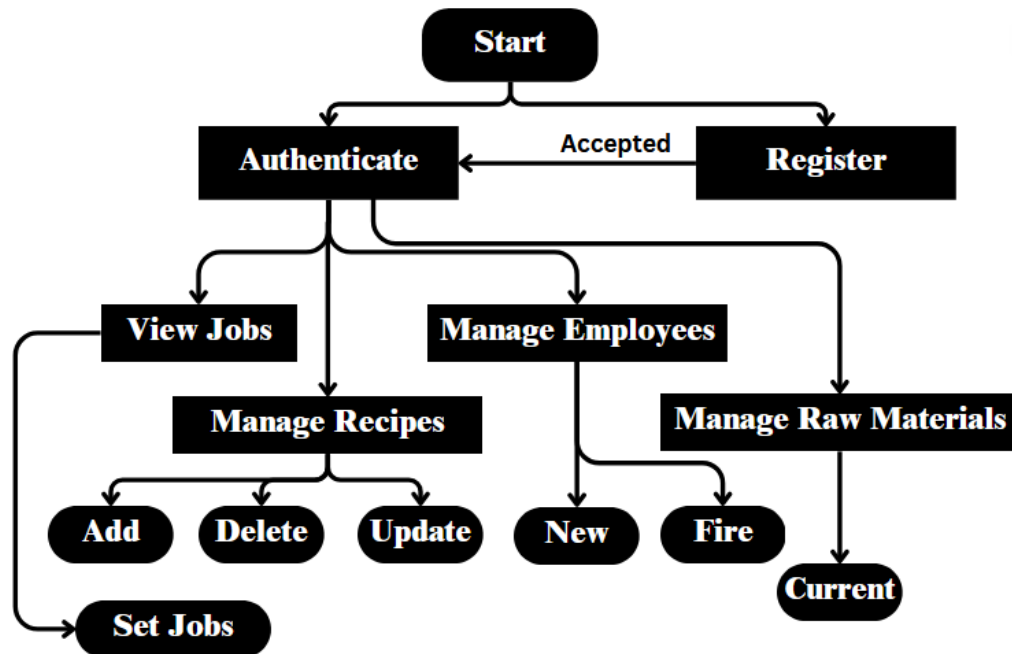


Figure 3: Admin View Flowchart

4.4. Level – 0 DFD:

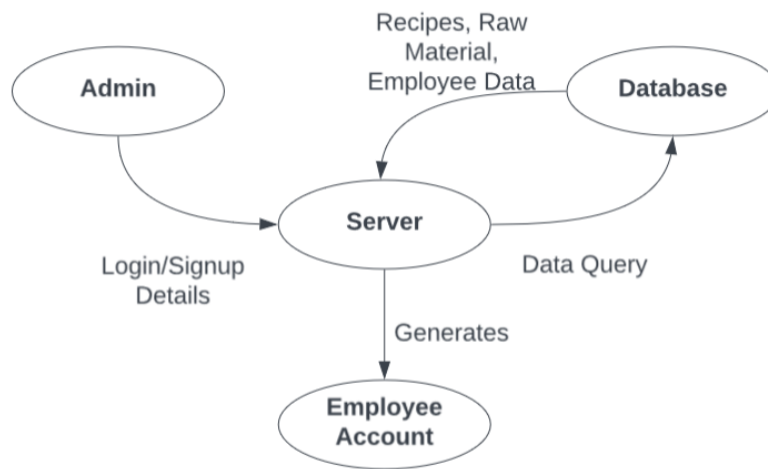


Figure 4: Level 0 DFD

4.5. Level – 1 DFD:

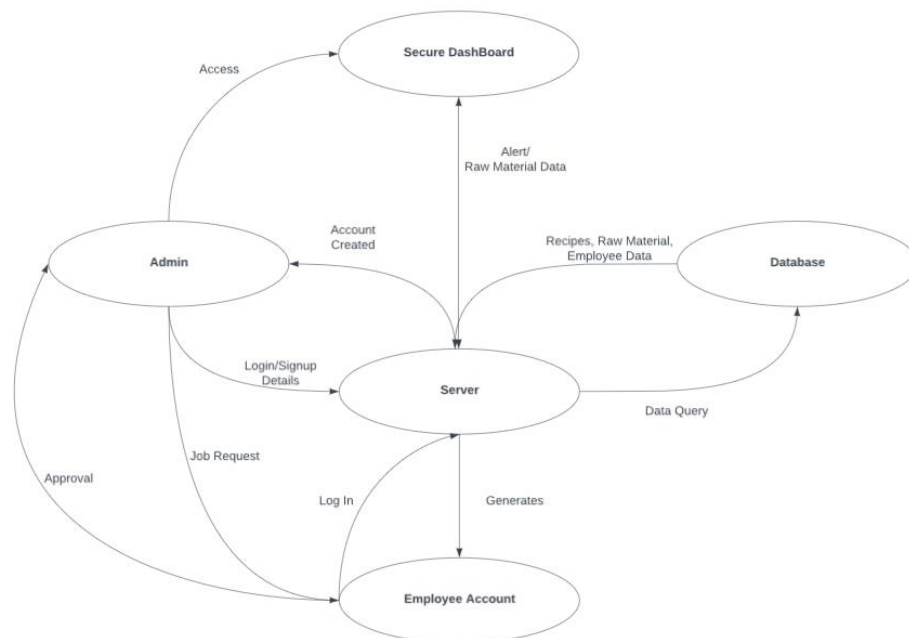


Figure 5: Level 1 DFD

4.6. Database Schema Design:

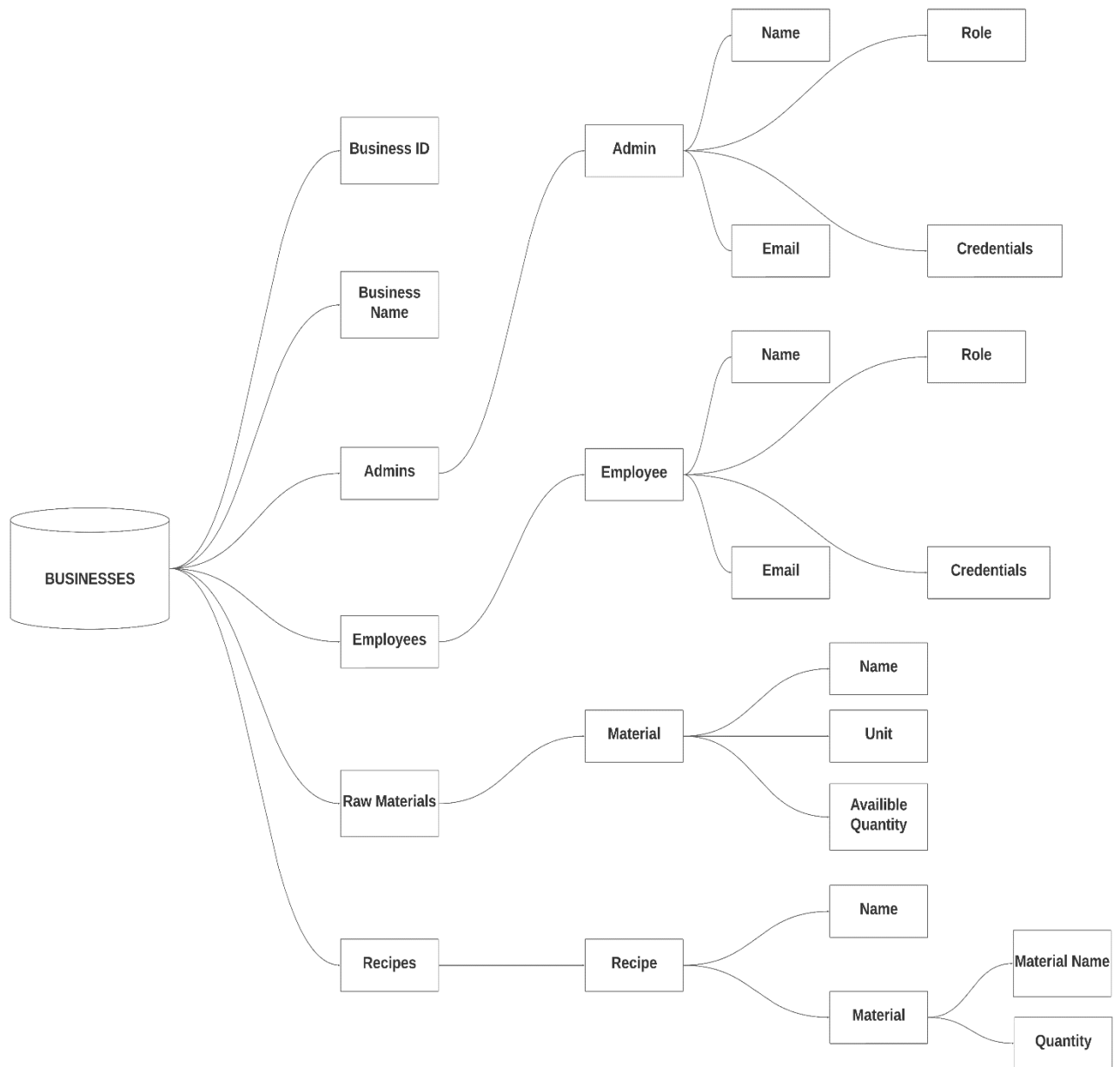


Figure 6: Database Design Schema

4.7. Visualization of the Routes:

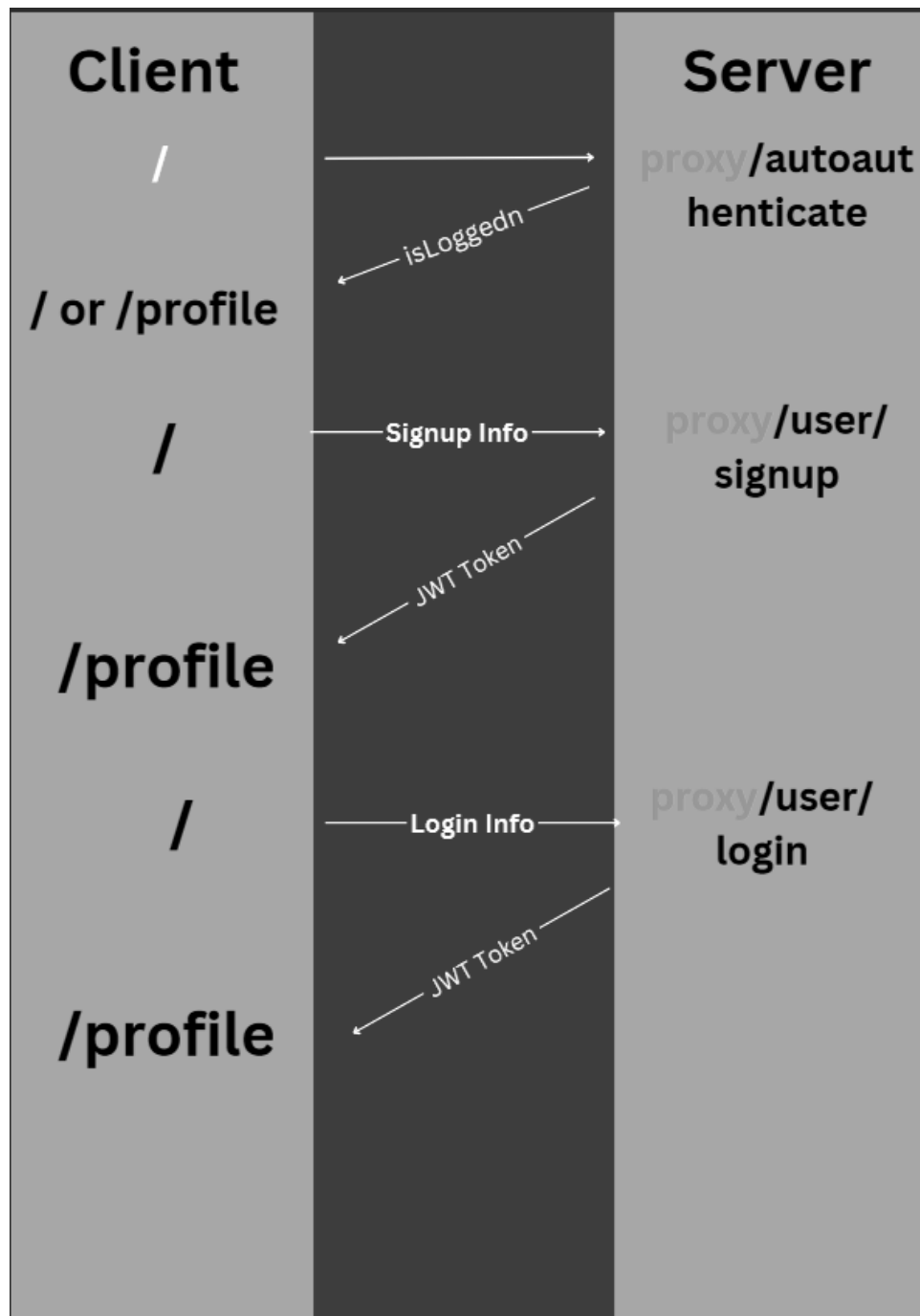


Figure 7: Visualization of HTTP Transaction

4.8. Project Directory Structure:

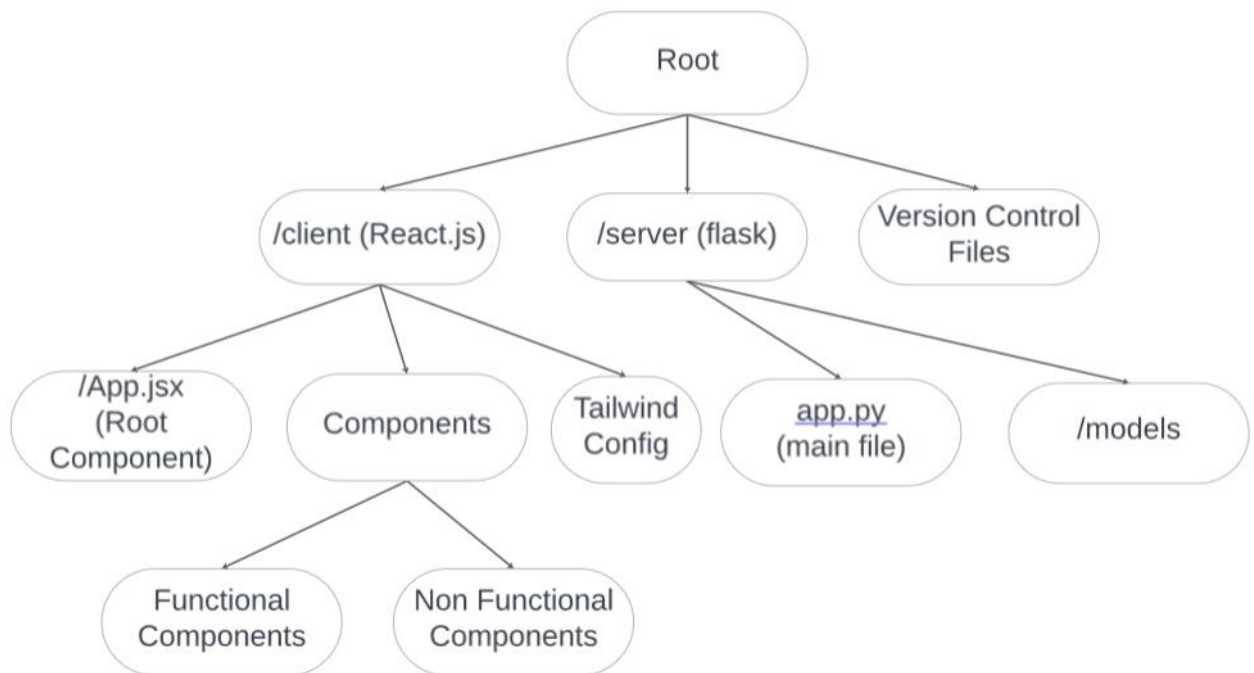


Figure 8: Directory Structure

4.9. Tech Stack

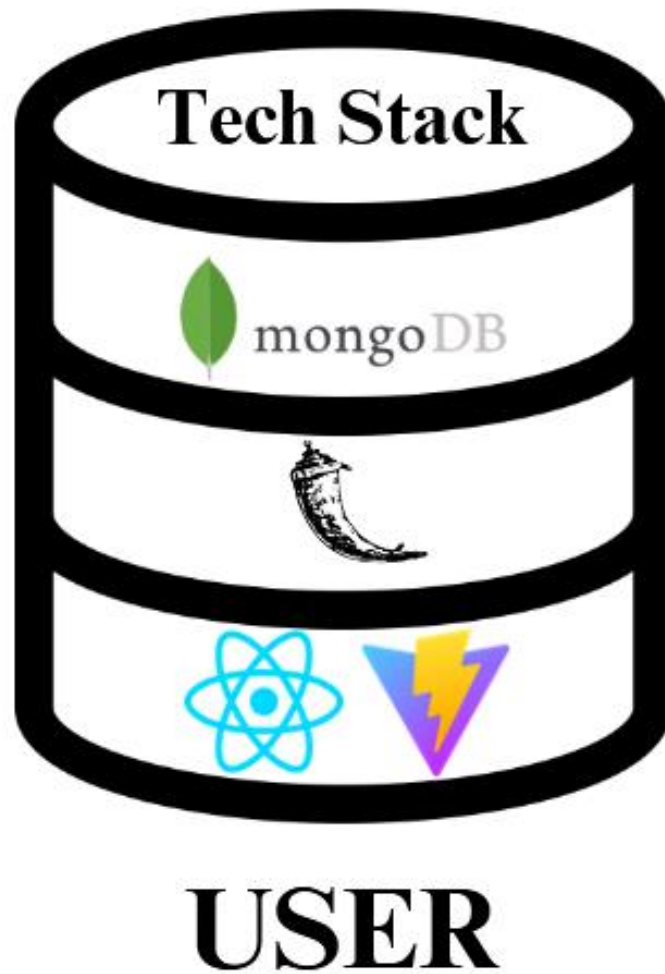


Figure 9: Tech Stack of the Project

5. Testing Modules: The testing of the project is done in three stages, Unit Testing, Module Testing, Integration Testing. All of the three stages are explained below.

5.1. Unit Testing: Unit testing is a fundamental practice in software development aimed at verifying the correctness of individual units or components of a software application. A unit is the smallest testable part of an application, such as a function, method, or procedure. Unit tests are automated checks that validate whether each unit of code performs as expected.

5.1.1. Isolation: Unit tests should be isolated, meaning they assess a specific unit of code in isolation from the rest of the application. This ensures that the test results accurately reflect the behavior of the individual unit.

5.1.2. Automated Execution: Unit tests are typically automated and run frequently throughout the development process. Automation ensures consistency and allows for quick identification of issues as the codebase evolves.

5.1.3. Repeatable and Deterministic: Unit tests are typically automated and run frequently throughout the development process. Automation ensures consistency and allows for quick identification of issues as the codebase evolves.

5.2. Front End Testing: Frontend testing is a crucial aspect of software development that focuses on verifying the functionality, usability, and performance of the user interface (UI) or frontend components of an application. The frontend, often referred to as the client-side, is the part of the application that users interact with directly. Effective frontend testing is essential to ensure a seamless and user-friendly experience across different devices.

Front End testing is done using host features of vite on a local network. Such that interface is consistent across devices.

5.3.Back End Testing: Backend testing is a type of testing that focuses on the server-side or

backend components of a software application. It involves verifying the functionality, performance, and reliability of the server-side logic, databases, APIs (Application Programming Interfaces), and other server components. Backend testing ensures that the application's backend processes data correctly, handles requests and responses appropriately, and interacts effectively with databases and external services.

Curl is a command-line tool and library for making HTTP requests, which makes it a handy tool for testing APIs and backend services. Here's a basic explanation of how backend testing can be done using curl commands:

API Endpoint Testing: Use curl to send HTTP requests to API endpoints and verify the responses.

To make a GET request to an API endpoint: `curl -X GET https://api.example.com/users`

To make a POST request with data: `curl -X POST -H "Content-Type: application/json" -d '{"username": "IMSMANAGER", "password": "secret"}' https://api.example.com/login`

Authentication Testing: Test authentication mechanisms by including appropriate headers in your curl commands.

Authentication with API Testing: `curl -H "Authorization: Bearer YOUR_API_KEY" https://api.example.com/data`

Database Interaction Testing: Check database interactions by sending requests that involve database operations. For example, if a backend has a CRUD (Create, Read, Update, Delete) API for a resource:

Create:

```
curl -X POST -H "Content-Type: application/json" -d '{"name": "New Item"}'  
https://api.example.com/items
```

Read

```
curl https://api.example.com/items/123
```

Update

```
curl -X PUT -H "Content-Type: application/json" -d '{"name": "Updated Item"}'  
https://api.example.com/items/123
```

Delete

```
curl -X DELETE https://api.example.com/items/123
```

5.4.Integration Testing: Integration testing is a crucial phase in the software testing lifecycle that focuses on verifying the seamless collaboration and interaction between different components or modules of a software application. The primary goal of integration testing is to detect defects in the interfaces and interactions between integrated components, ensuring that they function cohesively as intended. Unlike unit testing, which examines individual modules in isolation, integration testing evaluates the behavior of combined units to uncover any inconsistencies, data flow issues, or communication problems. This testing phase is essential for identifying potential integration-related issues early in the development process, promoting overall system reliability, and ensuring that the integrated components work together effectively to deliver the desired functionality. Integration testing can be performed using various strategies, such as top-down, bottom-up, or a combination of both, allowing developers to validate the collaboration between software modules and enhance the overall quality of the integrated system.

6. Output Screens:

6.1. Desktop View

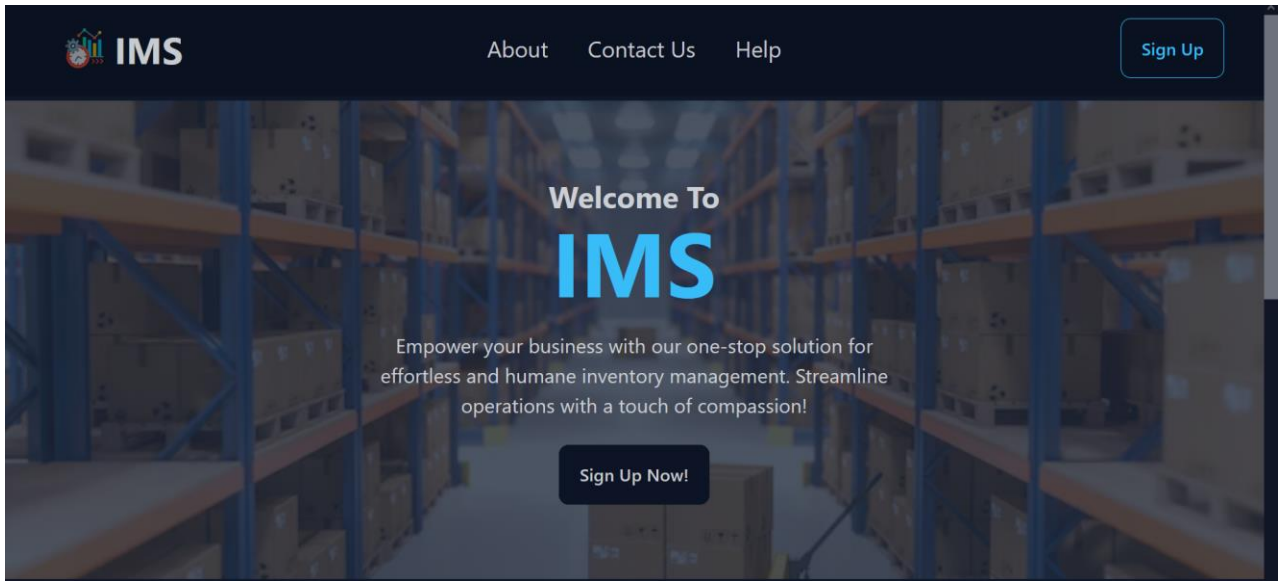


Figure 11: Landing User Interface

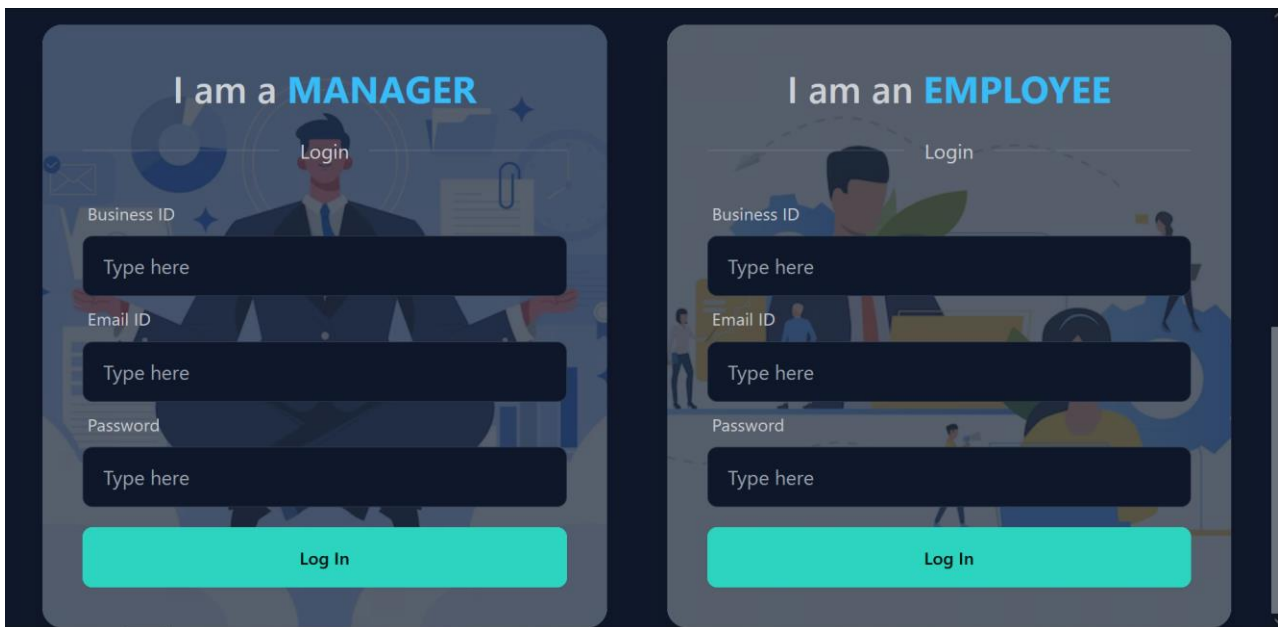




Figure 10: Login Portal

Welcome To IMS

Go Back



Business Name

Type here

Email ID

Type here

Admin Name

Type here

Password

Type here

Sign Up

Figure 12: Sign Up Module

6.2.Mobile View:

The landing page features a dark blue header with the IMS logo and a 'Sign Up' button. The main content area has a background image of a warehouse and displays the text 'Welcome To IMS' in large blue letters. Below this, a message states: 'Empower your business with our one-stop solution for effortless and humane inventory management. Streamline operations with a touch of compassion!'. A 'Sign Up Now!' button is centered. The page is divided into two sections for login: 'I am a MANAGER' and 'I am an EMPLOYEE'. Each section includes a 'Login' label, input fields for 'Business ID', 'Email ID', and 'Password', and a 'Log In' button.

Figure 13: Mobile View: Landing

The sign-up page features a dark blue header with the IMS logo and a 'Go Back' button. The main content area has a background image of a person standing next to a large smartphone displaying a login screen. Below this, the form includes input fields for 'Business Name', 'Email ID', and 'Admin Name', followed by a 'Password' field. A large 'Sign Up' button is at the bottom.

Figure 14: Mobile View: SignUp

7. Performance of the project developed:

7.1. Client-Side Development: Approximately 25% completion on the client side indicates a considerable focus on backend implementation, reflecting the project's strategic prioritization of backend development before frontend.

While the frontend's current state may be less advanced, this approach aligns with the project's demands, emphasizing the foundational backend infrastructure required for seamless functionality. The currently developed is standard to Web Interfaces, is Responsive to all devices and working as intended.

7.2. Server-Side Development: The project has witnessed significant progress in its development, with a primary focus on backend implementation. The performance of the backend is fast and smooth. It follows the structure of HTTP Requests and is successfully connected to the MongoDB Database.

Following are the backend code outputs:

```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"name":"John Doe","email":"john@example.com","password":"password123","role":"employee","business_id":"6bca9d8a6f77"}' http://127.0.0.1:5000/user/create
{
  "success": true
}

(venv) D:\React\MinorProject\ims-minorproject>
```

Figure 15: User Signed Up Successfully

```
(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"business_id":"6bca9d8a6f77","email":"john@example.com","password":"password123","role":"employee"}' http://127.0.0.1:5000/user/login
{
  "jwt_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsIm1hdCI6MTcwODQxMDU5NSwianRpIjoiMmE3OTZlODUtN2ZjYS00MzE2LWE2ZWMTMzIyMmU2YmY3ZTYwIiwidHlwZSI6ImFjY2VzcyIsInN1YiI6IkpvaG4gRG9lIiwibmJmIjoxNzA4NDExNTk1LCJpc3JmIjoiYTFjMGRiMzItNDZjYS00MzNiLWI4NGEtNTVmNGRjMTBjOTY0IiwiaXNjaXhwIjoxNzA4NDExNDk1LCJyYW11IjoiSm9obiBEb2UiLCJyb2x1IjoiZW1wbG95ZWUifQ.3MZBch4RftDSQTjy9mqXphA56JMbFgFJ6n6Ns5aR66A"
}
```

Figure 16: Successful Login

```
(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"item_name":"Ekus","quantity":500,"business_id":"6bca9d8a6f77"}' http://127.0.0.1:5000/item/add
{
  "message": "item added successfully"
}
```

Figure 17: Item Added Successfully

```
(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"item_name":"Ekus","quantity":500,"business_id":"6bca9d8a6f77"}' http://127.0.0.1:5000/item/add
{
  "message": "item already present"
}

(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"business_id":"6bca9d8a6f77","email":"john@example","password":"password123","role":"employee"}' http://127.0.0.1:5000/user/login
{
  "error": "Credentials not found"
}

(venv) D:\React\MinorProject\ims-minorproject>curl -X POST -H "Content-Type: application/json" -d '{"name":"John Doe","email":"john@example.com","password":"password123","role":"employee","business_id":"6bca9d8a6f77"}' http://127.0.0.1:5000/user/create
Email Already exists
(venv) D:\React\MinorProject\ims-minorproject>
```

Figure 18: Fail States

8. REFERENCES

- [1] React Documentation [Online]. Available: <https://react.dev/> .
- [2] Flask's Documentation [Online]. Available: <https://flask.palletsprojects.com/en/3.0.x/>
- [3] Flask's Tutorials by Tech With Tim [Online]. Available:
https://www.youtube.com/watch?v=mqhxxeeTbu0&list=PLzMtBGfZo4-n4vJJybUVV3Un_NFS5EOgX.
- [4] GeeksForGeeks. How to connect ReactJs with Flask API [Online]. Available:
<https://www.geeksforgeeks.org/how-to-connect-reactjs-with-flask-api/>
- [5] Towards Data Science. How To Build & Deploy a React + Flask App [Online].
Available: <https://towardsdatascience.com/build-deploy-a-react-flask-app-47a89a5d17d9>