# GitGrub

## Software Requirement Analysis Document


**Submitted to**
Dr. Saifuddin Md. Tareeq, Professor
and
Redwan Ahmed Rizvee, Lecturer

**Submitted by**
Meherun Farzana, Roll 05,
Mehrajul Abadin Miraj, Roll 20
and
Aniket Joarder, Roll 48

Group A, Third Year
Department of Computer Science & Engineering
University of Dhaka

**Submitted On**
18 February 2024

# Table of Contents

## 4. Supporting Information

# Description of Contents

## 1. Introduction

### 1.1 Purpose of the system

The purpose of the GitGrub canteen management system is to streamline and enhance the operations of cafeteria or canteen facilities within educational or corporate settings. The primary objectives are as follows:

1. **Efficiency Enhancement**: By automating various tasks such as order processing, inventory tracking, GitGrub aims to significantly improve the overall efficiency of canteen operations. This efficiency enhancement translates to faster service and reduced waiting times for customers.

2. **Order Accuracy**: With features designed to handle customer orders effectively, GitGrub ensures that orders are processed accurately and promptly. This minimizes errors in order preparation and helps maintain customer satisfaction.

3. **Inventory Management**: Effective inventory tracking is crucial for maintaining optimal stock levels and minimizing wastage. GitGrub provides comprehensive inventory management functionalities, enabling canteen managers to monitor stock levels, track food items, and efficiently manage supplies.

4. **Employee Supervision**: GitGrub facilitates the supervision of canteen employees by providing tools for managing schedules, roles, and tasks. This ensures that staffing levels are adequate during peak hours and that employees are assigned tasks efficiently, contributing to smoother operations.

5. **Customer Interaction and Feedback**: GitGrub includes features for engaging with customers and gathering feedback, allowing canteen managers to understand customer preferences, address concerns, and continuously improve the quality of service.

6. **Data Integrity and Optimization**: The system's optimized database architecture ensures the integrity and security of canteen-related data. This reliability is essential for making informed decisions based on accurate information and maintaining smooth operations.

Overall, the purpose of GitGrub is to provide a comprehensive solution for managing all aspects of canteen operations, ultimately aiming to

maximize accuracy, efficiency, and customer satisfaction in educational or corporate settings.

## 1.2 Scope of the system

The scope of the GitGrub canteen management system encompasses all essential features and functionalities necessary for optimizing cafeteria or canteen operations within educational or corporate settings. Specifically, the software will cover the following key areas:

1. **Order Processing:** The system will facilitate efficient processing of customer orders, including order placement, modification, and cancellation.

2. **Inventory Management:** GitGrub will include tools for managing inventory levels, tracking food items, and optimizing stock replenishment to minimize wastage.

3. **Employee Management:** The software will provide functionality for managing employee schedules, roles, and tasks to ensure smooth operation and optimal staffing levels.

4. **Customer Interaction:** GitGrub will enable interaction with customers, including order status updates, feedback collection, and notifications about promotions or specials.

5. **Reporting and Analytics:** The system will offer reporting and analytics capabilities to provide insights into sales trends, customer preferences, and operational efficiency.

6. **Security and Access Control:** GitGrub will incorporate robust security measures to protect sensitive data and ensure that access to system functionalities is appropriately restricted based on user roles.

It's important to note that while this scope outlines the initial functionalities of the system, GitGrub may undergo updates or revisions in the future to address new requirements or enhance existing features. Any changes to the system will be documented and communicated to stakeholders accordingly.

## 1.3 Objective and success criteria of the project

The objective of the GitGrub canteen management system is to streamline and optimize cafeteria or canteen operations within educational or corporate settings. The primary goals of the project include:

1. **Efficiency Enhancement**: Improve the efficiency of canteen operations by automating tasks such as order processing, inventory management, and employee scheduling.

2. **Accuracy Improvement:** Ensure order accuracy and minimize errors in order preparation to enhance customer satisfaction.

3. **Resource Optimization:** Optimize inventory levels and staffing resources to reduce wastage and operational costs.

4. **Customer Satisfaction:** Enhance customer satisfaction by providing a seamless ordering experience, gathering feedback, and responding to customer preferences.

### Success Criteria:

The success of the GitGrub project will be measured against the following criteria:

1. **Improved Efficiency**: A measurable increase in the speed and accuracy of order processing and other canteen operations.

2. **Reduced Wastage**: Demonstrable reductions in food and inventory wastage, leading to cost savings for the organization.

3. **Enhanced Customer Satisfaction**: Positive feedback from customers regarding the ease of ordering, order accuracy, and overall dining experience.

4. **Operational Insights**: The availability of actionable insights through reporting and analytics tools, enabling informed decision-making and continuous improvement.

5. **Adherence to Budget and Timeline**: Completion of the project within the allocated budget and timeframe, meeting all specified requirements and milestones.

By achieving these objectives and meeting the established success criteria, the GitGrub project will contribute to the efficient and effective management of canteen facilities, ultimately enhancing organizational performance and customer satisfaction.

## 1.4 Definitions and Acronyms and abbreviations

- **UI:** User Interface
- **API:** Application Programming Interface
- **UX:** User Experience
- **JS:** JavaScript
- **CRUD:** Create, Read, Update, Delete
- **HTTPS:** Hypertext Transfer Protocol Secure
- **SQL:** Structured Query Language
- **JSON:** JavaScript Object Notation

## 1.5 References


- FastAPI (tiangolo.com)
- The web framework for perfectionists with deadlines | Django (djangoproject.com)
- PostgreSQL: The world's most advanced open source database
- Intelligent Diagramming | Lucidchart
- (3) Lucid Software - YouTube


## 1.6 Overview

GitGrub is a comprehensive canteen management system designed to revolutionize operations within educational or corporate settings. Focused on optimizing efficiency and customer satisfaction, the system automates key tasks such as order processing, inventory management, and employee scheduling. GitGrub aims to enhance accuracy in order fulfillment, reduce operational costs through resource optimization, and foster positive customer experiences through seamless interactions and feedback mechanisms. With robust security measures and future adaptability in mind, GitGrub is positioned to elevate the daily operations of canteen

facilities, aligning with the organization's goals of improved service quality and operational excellence.

# 2. Overall Description

## 2.1 Product perspective

GitGrub is a robust web application aimed at revolutionizing canteen management within educational or corporate environments. It serves as a comprehensive solution to streamline operations, including order processing, inventory tracking, employee management, and customer interactions. GitGrub is designed to be a standalone system, providing a centralized platform for all canteen-related activities. However, it also offers the flexibility to integrate with other systems or databases as needed. Built with modern technologies such as FastApi, django, PostgreSQL. GitGrub ensures scalability, security, and adaptability for future enhancements. The primary objective of GitGrub is to optimize efficiency, accuracy, and customer satisfaction within canteen facilities, empowering staff to focus on delivering high-quality service while providing a seamless experience for customers.

## 2.2 Product functions

GitGrub has the following major functions:

- **Create account:** Customers and shop owners can create accounts from the web application. For this their email will be verified.
- **Forget password:** If a user forgets their password they can change it with their email address.
- **Online order** : From this application users will be allowed to order food online and pay for it. After the order is processed they can take their food from the counter by their order ID.
- **History:** A user can see their history of ordering or serving.
- **Menu management:** Shop owners and managers can update their menu according to their needs.
- **Sell food online**: In this system, traditional wait staff within the restaurant are rendered unnecessary. Instead, restaurant owners

will have the capability to list their menus online, allowing customers to place orders directly through the platform. Additionally, the system facilitates the notification of customers once their orders are ready for pickup.

- **Inventory control:** Shop owners and managers will be able to see their stocks.
- **Employee management:** Administrators can check the employee status and schedule shifts among them.
- **Review system:** In this system users will be able to review food and all the users using GitGrub will be able to see the review before ordering so that they can decide whether this food is really up to the mark as the owner claims.
- **Sale history** : Shop owners will be able to keep track of their sales via this app.
- **Inventory management** : From this app managers will be able to let the user know which of the items are currently available and users will be able to order accordingly.

## 2.3 User Profiles

- **Administrator**: Will oversee the whole canteen, access analytics, and manage user accounts.
- **Customers**: Will use the app to browse menus, place orders online and offline, and provide feedback. Customers will include students, teachers and staff of a department/university

## 2.4 Constraints

There will be no general constraint within our target.

- Technology: The software is only available for PC operating systems, specifically Windows. Therefore, the development team should keep in mind the compatibility requirements of the software.
- Order Delivery: Due to time constraints we are not implementing the home delivery services which will be included later.

## 2.5 Assumptions and dependencies

The GadgetGrove project relies on the following assumptions and dependencies:

**Assumptions:**

- **User familiarity with online Ordering**: We assume users possess a basic understanding of online ordering processes, including adding foods to a cart, proceeding to checkout, and making secure online payments.
- **Technical feasibility**: We assume the chosen technology stack (Django, Fast Api, chosen database) can effectively support the functionalities envisioned for the initial release.
- **Payment processing integration**: We assume seamless integration with a secure payment processing service like Stripe for handling financial transactions.
- **User adoption**: We assume there will be a positive user perception towards GitGrub's functionalities and user interface.

**Dependencies:**

- **External Payment Processing Service (Stripe)**: The functionality and security of the platform depend on successful integration with a secure payment processing service like Stripe.
- **Database Management System (RDBMS)**: The project relies on a chosen relational database management system to store product information, user accounts, and order details.

# 3. Specific Requirement

## 3.1 Overview

GitGrub is a comprehensive canteen management system designed to revolutionize operations within educational or corporate settings. Focused on optimizing efficiency and customer satisfaction, the system automates

key tasks such as order processing, inventory management, and employee scheduling. GitGrub aims to enhance accuracy in order fulfillment, reduce operational costs through resource optimization, and foster positive customer experiences through seamless interactions and feedback mechanisms. With robust security measures and future adaptability in mind, GitGrub is positioned to elevate the daily operations of canteen facilities, aligning with the organization's goals of improved service quality and operational excellence.

## 3.2 Functional Requirements

| | |
|---|---|
| **ID** | 3.2.1 |
| **Name** | User Management |
| **Description** | 1. The system will allow the users to create an account using their mobile number.<br>2. The system shall allow the users to update their profile and change password using mobile number. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.2 |
| **Name** | Employee Management |
| **Description** | 1. The system shall allow the admin to create and manage staff accounts.<br>2. The system shall allow staff to update their account information.<br>3. The system shall allow the admin to reset passwords for staff accounts. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.3 |
| **Name** | Placing Order |
| **Description** | 1. The system will allow the the users to order their food items and number of items from seeing the menu.<br>2. The system allows them to select dine in or take away or home delivery. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.4 |
| **Name** | Pereapring Order management |
| **Description** | 1. The system will allow the managing staff to accept the order.<br>2. The system will allow the managing staff to update all the details of any particular order.<br>3. The system shall allow the managing staff to set the order completed |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.5 |
| **Name** | Delivering Order Management |
| **Description** | 1. The system shall allow to see the managing staff and user to see how long it is required to delever a food for delivery person.<br>2. It allows the delivery person to update the order delivery status. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.6 |
| **Name** | Menu Management |
| **Description** | 1. The system shall allow the inventory manager to set and update the menu according to the food items.<br>2. The system shall allow the inventory manager to remove or add a item from the list |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.7 |
| **Name** | Invetroy Management |
| **Description** | 1. The system shall allow the inventory manager and chef to see how many ingredients are remaining in the stock.<br>2. The system shall allow the inventory manager and the chef to update the inventory after taking somethin or addin into it. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.8 |
| **Name** | Transaction Management |
| **Description** | 1. The system show a transaction option for making paymen.<br>2. The system shall allow the managing staff to update the payment transition if it is selected to cash. |
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.9 |
| **Name** | Feedback system |
| **Description** | 1. The system shall allow the users to give feedback of their experiences.<br>2. The system shall allow the admin to see the feedback of the customers. |

| | |
|---|---|
| **Priority** | High |
| **Reference** | |

| | |
|---|---|
| **ID** | 3.2.10 |
| **Name** | Reports and Analysis |
| **Description** | 1. The system will allow the admin to generate reports of the orders over a time period<br>2. The system will allow both the admin and user to see analytics of orders for different time |
| **Priority** | Medium |
| **Reference** | |

## 3.3 Non Functional requirement

### 3.3.1 Usability

This application is very user-friendly due to its intuitive interface, streamlined ordering process, and helpful features. Users can easily search in the menu and select their desired item and quantity and other short things. Again the managing staff can easily manage in a single place. The application also minimizes the need for extensive training or technical knowledge, making it highly accessible and user-friendly.

### 3.3.2 Reliability

The gitgrub application is reliable due to various factors such as redundancy in the system, error handling capabilities, and fault tolerance. The application is designed to handle errors gracefully, without crashing or losing user data. It also has multiple backup systems to ensure uninterrupted service for users, making it a reliable platform for the CSEDU canteen.

### 3.3.3 Performance

Performance requirements for the gitgrub are critical to ensure a smooth and efficient user experience. Specific performance requirements include response times for food

search and ordering, with the goal of ensuring that results are returned in under one second and that ordering transactions are completed in under five seconds. The application must also be able to handle a high volume of concurrent users, with a target of supporting up to 1000 users at any given time. These requirements are necessary to ensure that the application meets user expectations for speed and efficiency, while also enabling the business to handle a large number of users and transactions.

### 3.3.4 Supportability

As a web application, the gitgrub is inherently supportable due to its accessibility from any device with an internet connection. It is compatible with a wide range of web browsers, making it accessible to a broad audience. The application's containerization and orchestration also ensure that it can be easily deployed and managed across multiple environments. These factors make the application highly supportable as a web app

### 3.3.5 Implementation

The software should be developed using a high-level programming language, Python, Django and JS.The software should use a database management system (DBMS) to store and retrieve data efficiently. The preferred DBMS for the project is MySQL.The software should be developed using version control software, such as Git, to enable collaboration and manage code changes effectively.

### 3.3.6 Scalability

The application is highly scalable due to its microservice architecture, which allows for the easy addition of new services as the need arises. The use of containerization and orchestration through Kubernetes enables efficient resource utilization and autoscaling, making it easy to accommodate high-traffic loads. Additionally, the use of messaging 13 queues through Kafka enables asynchronous communication between services, improving scalability and fault tolerance. These features make the application highly scalable and easily adaptable to changing demands.

### 3.3.7 Security

The application places a high priority on security, implementing various measures to safeguard sensitive information. These include establishing secure communication channels between services via HTTPS, employing JWT tokens for authentication and authorization, and encrypting sensitive data stored within the system. Furthermore, the application adheres to industry best practices for secure coding and undergoes routine security audits to proactively identify and address potential vulnerabilities, ensuring continuous protection against security threats.

### 3.3.8 Testability

Ensuring the reliability and maintainability of our application is paramount, making testability a critical aspect of our development process. We will adopt a test-driven development methodology, incorporating comprehensive unit testing, integration testing, and end-to-end testing to validate adherence to all requirements. Utilizing testing frameworks such as Jest and Supertest will streamline the testing process, allowing for early detection and resolution of any bugs or performance concerns. Our overarching objective is to cultivate a highly testable application that fulfills both functional and non-functional specifications with precision.

### 3.3.9 Maintainability

The product's maintainability requirements emphasize the importance of easy extensibility, modularity, and comprehensive documentation. Code readability, organization, and flexibility for modifications are key considerations. Encouraging the use of common design patterns and principles fosters good coding practices. Additionally, the availability of testing and debugging tools simplifies troubleshooting processes. Lastly, the product design should facilitate effortless upgrading or replacement of components as needed, ensuring long-term maintainability and adaptability.

## 3.4 System Models

### 3.4.1 Scenarios

#### 3.4.1.1 Scenario 1: Customer Places An Order

**Flow of Events:**

- Customer registers for an account (or logs in if already registered).
- Customer browses the menu and searches for desired food items.
- Customer selects items, chooses quantities, and adds them to the cart.
- Customer reviews the cart contents and total price.
- Customer submits the order.
- System verifies order details and deducts the amount from the customer's balance (if applicable).
- (Success) System confirms the order, displays an estimated delivery time.
- Customer receives the order within the estimated timeframe and enjoys their meal.

#### 3.4.1.2 Scenario 2: Employee Adds New Menu Item

**Flow of Events:**

- Employee logs in and accesses the menu management section.
- System provides a form for adding new items.
- Employee enters details for the new item (name, description, price, category, image).
- Employee may also specify additional options.
- Employee submits the new item information.
- System validates the information and adds the new item to the menu database, making it available for customer orders.

#### 3.4.1.3 Scenario 3: Employee Adds New Menu Item

**Flow of Events:**

- Employee logs in and accesses the menu management section.
- System provides a form for adding new items.
- Employee enters details for the new item (name, description, price, category, image).
- Employee may also specify additional options.
- Employee submits the new item information.
- System validates the information and adds the new item to the menu database, making it available for customer orders.

#### 3.4.1.4 Scenario 4: Customer Cancels Order Before Preparation

**Flow of Events:**

- Customer places an order.
- Before order preparation begins, the customer decides to cancel.
- Customer locates the order and initiates cancellation.
- System displays a confirmation prompt.
- Customer confirms cancellation.
- System cancels the order, refunds the customer (if applicable), and notifies the canteen staff (optional).

#### 3.4.1.5 Scenario 5: Customer Orders Out-of-Stock Item

**Flow of Events:**

- Customer places an order, including an item that is out of stock (unknown to the customer).
- System processes the order but cannot fulfill the out-of-stock item.
- (Failure) System informs the customer about the unavailable item and offers options:
- Substitute the item with a similar option.
- Remove the item from the order and adjust the total price.
- Cancel the entire order and receive a full refund.

**3.4.1.6 Scenario 6: Invoice Generation and Email:**

**Flow of Events:**

- Upon successful order confirmation, the system generates an invoice.
- The invoice details include order items, quantities, prices, total amount, and any applicable taxes.
- The system retrieves the customer's email address from their user profile.
- The system composes an email containing the invoice details as an attachment (PDF or similar format) and a brief message (e.g., "Thank you for your order!").
- The system sends the email to the customer's email address.

**3.4.1.7**

**Scenario 7: Forgot Password**

**Flow of Events:**

- Customer attempts to log in but forgets their password.
- The login page provides a "Forgot Password" link or button.
- Customer clicks the "Forgot Password" option.
- The system prompts the customer to enter their registered email address.
- The system verifies the email address exists in the user database.
- (Success) The system generates a unique password reset link and sends it to the customer's email address.
- (Failure) The system displays an error message if the email address is not found.
- Customer checks their email for the password reset link.
- Customer clicks the password reset link in the email. The link might redirect them to a password reset page on the web app.
- The password reset page prompts the customer to enter a new password (meeting complexity requirements) and confirm it by re-entering it.
- The system validates the new password and updates the customer's account with the new password.

- The system displays a confirmation message or redirects the customer to the login page with the prompt to use their new password.

### 3.4.2 Use cases

#### 3.4.2.1 Use case 1

| | |
|---|---|
| **Name** | Register Customer/Employee |
| **Actor** | New User |
| **Flow of events** | • Actor chooses "Register" option.<br>• System prompts for user type (Customer/Employee).<br>• Actor selects user type.<br>• System prompts for registration details (name, email, password etc.).<br>• Actor enters details.<br>• System validates details and checks for existing email.<br>• (Success) System creates a new account and displays success message.<br>• (Failure) System displays error message and prompts for correction. |
| **Entry condition** | User is a new visitor to the web app. |
| **Exit condition** | • Successful registration: User account is created and a success message is displayed.<br>• Unsuccessful registration: Error message is displayed, and the user remains unregistered. |

#### 3.4.2.2 Use case 2

| | |
|---|---|
| **Name** | Login |
| **Actor** | Registered Customer/Employee |

| | |
|---|---|
| **Flow of events** | • Actor enters email and password.<br>• System validates credentials against the user database.<br>• (Success) System logs the user in and redirects them to the appropriate dashboard (Customer/Employee).<br>• (Failure) System displays an error message (e.g., invalid credentials). |
| **Entry condition** | User has a registered account. |
| **Exit condition** | • Successful login: User is redirected to their dashboard.<br>• Unsuccessful login: User remains on the login page with an error message. |

#### 3.4.2.3 Use case 3

| | |
|---|---|
| **Name** | Profile Update |
| **Actor** | Logged-in Customer/Employee |
| **Flow of events** | • Actor navigates to the "Profile" section.<br>• System displays current profile information.<br>• Actor edits desired information (e.g., name, phone number).<br>• Actor submits changes.<br>• System validates changes and updates the user record.<br>• (Success) System displays a confirmation message.<br>• (Failure) System displays an error message (e.g., invalid data format). |
| **Entry condition** | User is logged in. |

| | |
|---|---|
| **Exit condition** | • Successful update: User profile is updated, and a confirmation message is displayed.<br>• Unsuccessful update: User profile remains unchanged, and an error message is displayed. |

#### 3.4.2.4 Use case 4

| | |
|---|---|
| **Name** | Search Food |
| **Actor** | Logged-in Customer |
| **Flow of events** | • Actor enters search criteria (e.g., food name, category) in the search bar.<br>• System searches the database based on criteria.<br>• System displays a list of matching food items with details (name, description, price). |
| **Entry condition** | User is logged in as a customer. |
| **Exit condition** | System displays search results or a message indicating no matches found. |

#### 3.4.2.5 Use case 5

| | |
|---|---|
| **Name** | Place an Order |
| **Actor** | Logged-in Customer |
| **Flow of events** | • Actor selects a food item from the search results or menu.<br>• System displays item details (including price, options).<br>• Actor selects quantity and any additional options.<br>• Actor adds the item to the cart.<br>• (Repeat steps 1-4 for additional items) |

| | |
|---|---|
| | • Actor reviews the cart contents and total price.<br>• Actor submits the order.<br>• System verifies order details and customer balance (if applicable).<br>• (Success) System confirms the order, displays an estimated delivery time, and deducts the amount from the user's balance (if applicable). |
| **Entry condition** | User is logged in as a customer and has browsed the menu. |
| **Exit condition** | Successful order: Order is confirmed, and the user receives an estimated delivery time.<br>Unsuccessful order: Order is not placed, and an error message is displayed. |

#### 3.4.2.6 Use case 6

| | |
|---|---|
| **Name** | Update Order Status (continued) |
| **Actor** | Involved Employee |
| **Flow of events** | • Employee selects the new order status (e.g., "preparing," "ready for pickup," "delivered").<br>• System updates the order status and sends a notification to the customer (optional). |
| **Entry condition** | Employee is logged in. |
| **Exit condition** | Order status is updated, and the employee can view other pending orders. |

#### 3.4.2.7 Use case 7

| | |
|---|---|
| **Name** | Give Feedback |

| | |
|---|---|
| **Actor** | Logged-in Customer |
| **Flow of events** | • Customer navigates to the "Feedback" section.<br>• System provides options for submitting feedback (e.g., rating system, text box).<br>• Customer selects a rating and/or enters comments about their experience (food quality, service, etc.).<br>• Customer submits the feedback.<br>• System stores the feedback and may display a confirmation message. |
| **Entry condition** | User is logged in as a customer. |
| **Exit condition** | Customer submits their feedback, and the system acknowledges receipt. |

#### 3.4.2.8 Use case 8

| | |
|---|---|
| **Name** | Update Canteen Inventory |
| **Actor** | Employee |
| **Flow of events** | • Employee logs in to the canteen system and accesses the inventory management section.<br>• System displays a list of current inventory items (food items, ingredients) with stock levels.<br>• Employee selects an item.<br>• System displays details of the selected item (including stock level, supplier information).<br>• Employee edits the stock level (e.g., add new stock, mark items as out of stock).<br>• System validates the update and adjusts the inventory record. |
| **Entry condition** | Employee is logged in and has access to inventory management. |

| | |
|---|---|
| **Exit condition** | Inventory level for the selected item is updated. |

#### 3.4.2.9 Use case 9

| | |
|---|---|
| **Name** | Add New Items |
| **Actor** | Employee |
| **Flow of events** | • Employee logs in to the canteen system and accesses the menu management section.<br>• System provides a form for adding new items.<br>• Employee enters details for the new item (name, description, price, category, image).<br>• Employee may also specify additional options (e.g., size, ingredients).<br>• Employee submits the new item information.<br>• System validates the information and adds the new item to the menu database. |
| **Entry condition** | Employee is logged in and has access to menu management. |
| **Exit condition** | New item is added to the menu with all specified details. |

#### 3.4.2.10 Use case 10

| | |
|---|---|
| **Name** | Cancel An Order |
| **Actor** | Customer (before order preparation) or Employee (after preparation) |
| **Flow of events** | • Customer (before preparation): Locates the order they wish to cancel. |

| | |
|---|---|
| | • System displays options to cancel the order (confirmation prompt).<br>• Customer confirms cancellation.<br>• System cancels the order, refunds the customer (if applicable), and sends a notification to the canteen staff (optional). OR<br>• Employee (after preparation): Locates the order requiring cancellation.<br>• System displays justification options for cancellation (e.g., out-of-stock item).<br>• Employee selects a reason and confirms cancellation.<br>• System cancels the order, updates inventory (if applicable), and notifies the customer. |
| **Entry condition** | • Customer: Logged in customer with an unprocessed order.<br>• Employee: Logged in employee with access to order management. |
| **Exit condition** | Order is cancelled, and the customer is notified (and refunded if applicable). |

#### 3.4.2.11 Use case 11

| | |
|---|---|
| **Name** | Manage Employees |
| **Actor** | Administrator |
| **Flow of events** | • Administrator logs in to the canteen system and accesses the employee management section.<br>• System displays a list of current employees with their roles and information.<br>• Administrator can perform various actions: Add new employee, edit employee details, deactivate/activate employee accounts. |
| **Entry condition** | Administrator is logged in. |
| **Exit condition** | Administrator completes the desired employee management action (add, edit, deactivate/activate). |

### 3.4.3 Use case model


Figure : Use Case Diagram

## 3.4.4 Dynamic Model

### 3.4.4.1 Activity Diagram

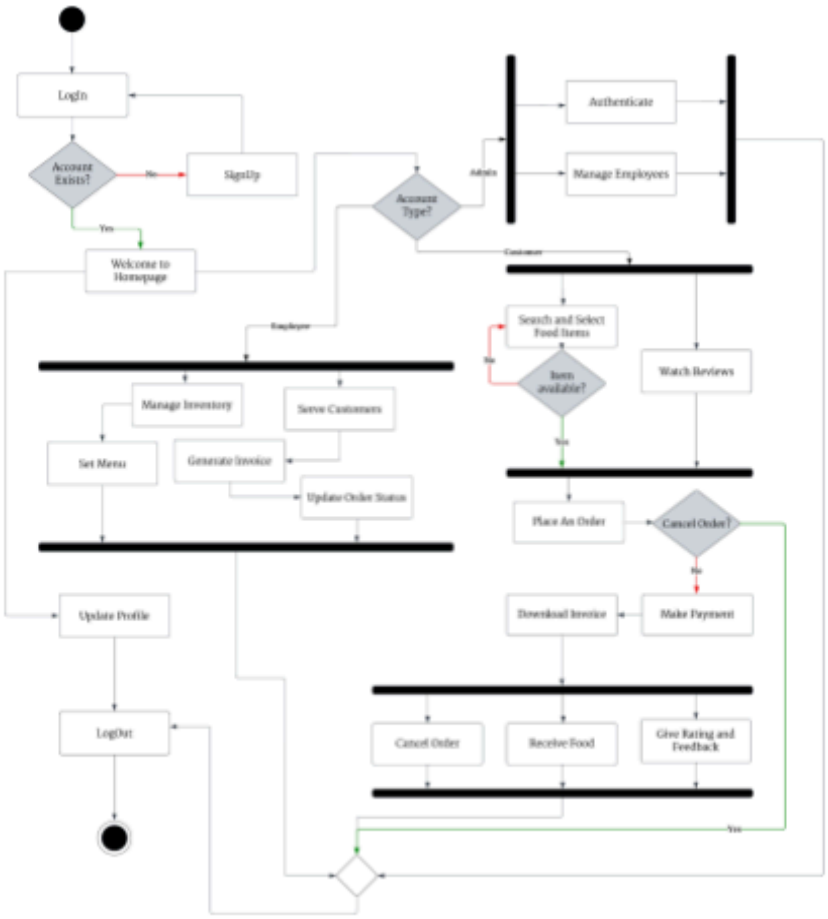


Figure: Activity Diagram

### 3.4.4.2 Statechart Diagrams


Figure: Statechart (Customer)

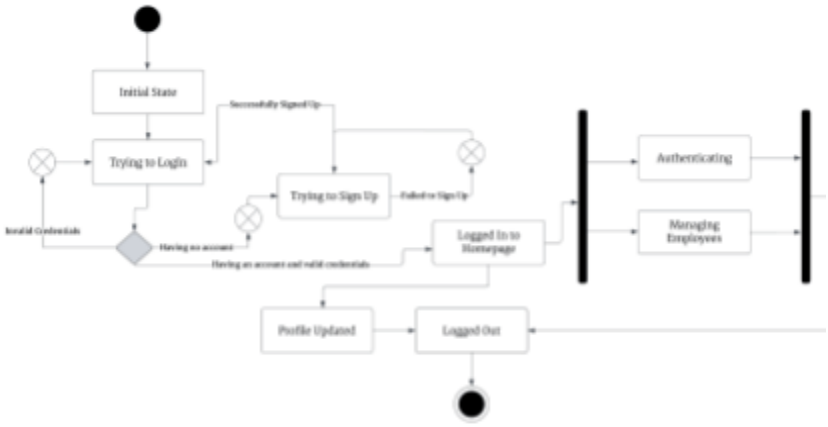


Figure: Statechart (Admin)

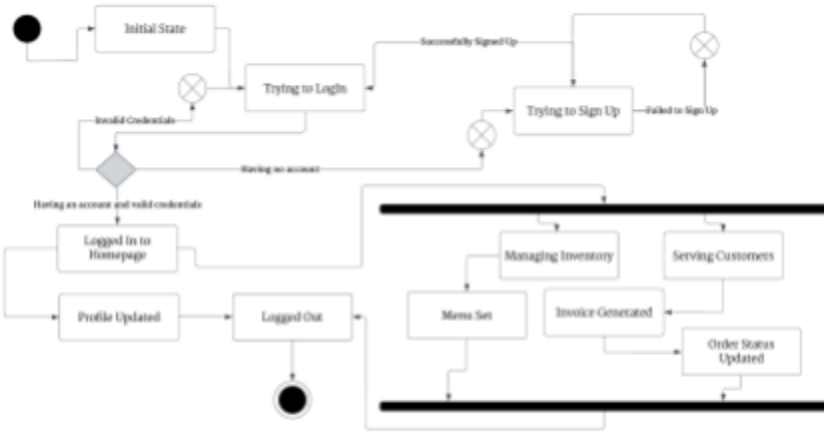


Figure: Statechart (Employee)

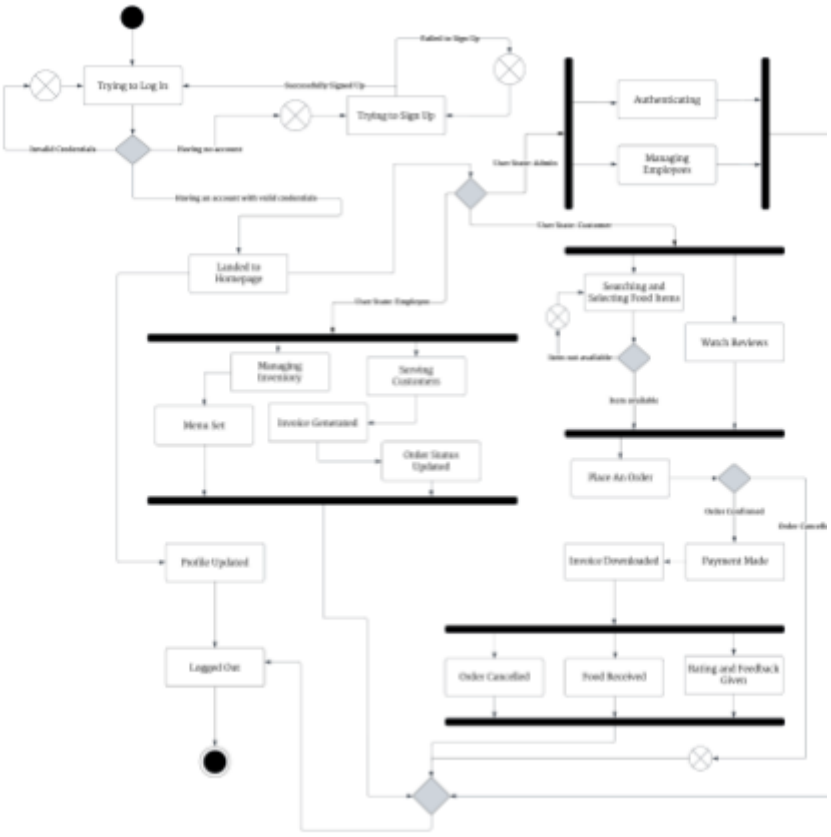


Figure: Statechart (Whole System)

#### 3.4.4.3 Sequence Diagram

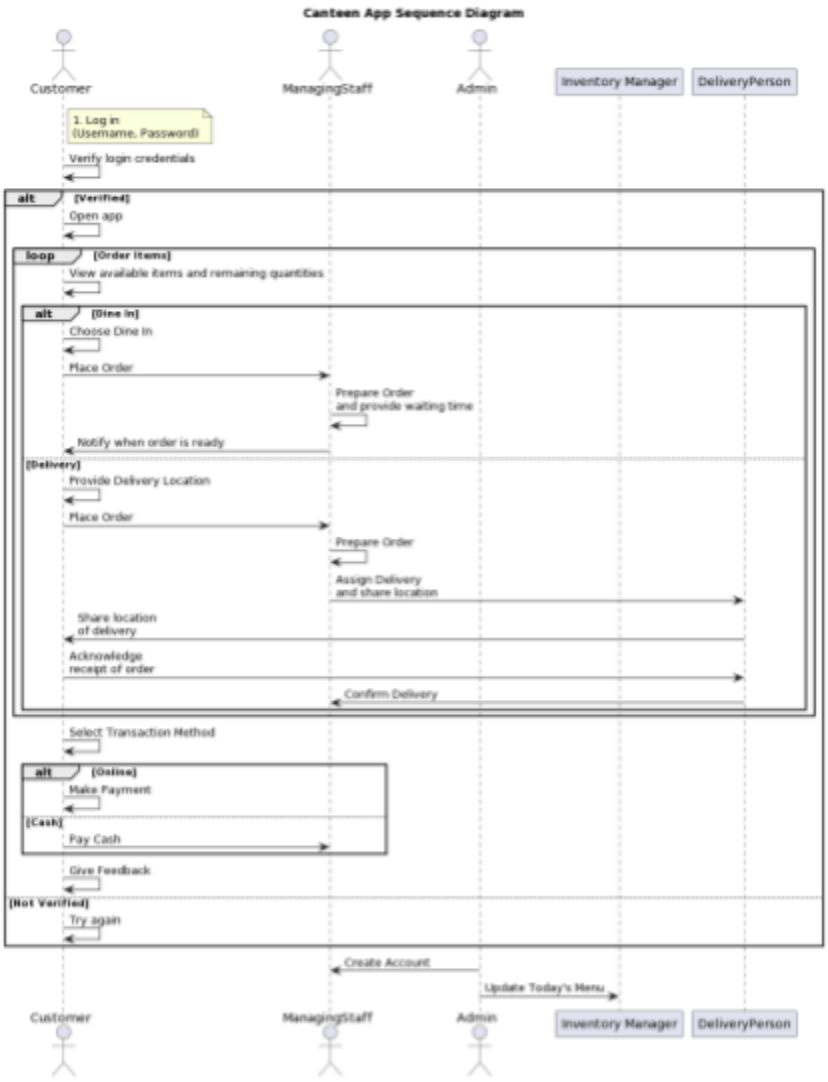


Figure: Sequence Diagram

### 3.4.5 Interfaces

#### 3.4.5.1 User Interface

The prototype for this application is done using Figma. The color scheme and keyboard
typography has been maintained throughout the application. The UI/UX has been
designed by researching and taking insight from various well-known and successful
travel agency businesses. The application is a web application so its responsiveness has
been ensured by testing the application on various devices like Desktop, Laptop, Ipad,

and Mobile phones. The overall User Interface workflow is described below:

1. At first anyone can see a login page whenever they visit our platform There will be two buttons login and sign up
2. If the user clicks the sign-up button he will re-directed to a sign up page where here he will see a form required for his sign up.
3. Clicking sing up a pop up will appear for otp after successfully giving the otp he will go to the home page.
4. If the user signs in he will redirected to the home page.
5. In the home page he will see options for ordering, current order and statistics of his orderings.
6. If he goes to statistics then he will see his past order statistics.
7. If he goes for order he will redirected to order page.
8. In the order page he will see the items available and their quantity and price. There he could select which items he would like to buy.
9. Then after selecting place order he will have two options one is cash other is online payment.
10. In the case of cash he will redirected to home page and his order will show pending because of payment not completed.
11. For selecting the online there will be the online platforms transaction page. After completing he will redirected to the home page ane in the current order page he will see order is preparing. And estimated time.
12. Then selecting his name or photo he can update his details there.
13. For the manager he will see update payment. Update order, statistics option
14. Selecting the payment option he will be able to update and take the cash payment orders
15. Selecting the update order he will able to add estimated time, order delivered or not.
16. In the statistics portion he can see the data of sales and orders.
17. The admin will able to update employees' information and their details.

18. The users will be able to give feedback on the feedback page.

#### 3.4.5.2 Software Interface

**Front-End (Presentation Layer):**

**HTML:** Provides the basic structure and content of web pages.
**CSS:** Styles the visual presentation of the web pages (layout, fonts, colors).

**Back-End (Logic Layer):**

**Programming Language (Python):** Python code handles server-side logic, user interactions, and data processing.
**Web Framework (Django/Flask):** Chosen framework (Django/ Flask) provides a foundation for building the web application, including functionalities like routing, templating, and database access.
**Database (PostgreSQL/MongoDB):** Stores all application data (user information, orders, menus, inventory).
**Data Flow:**

**Incoming Data:**

**User Input:** Through web forms, users provide data like registration details, order selections, and feedback.
**API Requests (Optional):** If using external APIs for payments, delivery services, or ingredient ordering, data will be received from those APIs.

**Outgoing Data:**

**Web Page Responses:** The system sends HTML, CSS, and JavaScript code to the user's browser to display web pages dynamically.
**Database Interactions:** The system interacts with the database to store user data, order information, and manage inventory levels.
**Emails:** The system sends emails for order confirmations, password resets, or notifications (optional).
**API Interactions (Optional):** Data might be sent to external APIs for payment processing, delivery requests, or ingredient resupply orders.

**Communication Services:**

**Database Management System (PostgreSQL or MongoDB):** The chosen database management system provides functionalities for storing, retrieving, and manipulating data within the database.

**Implementation Constraints:**

**Data Security:** Sensitive user data (passwords) should be stored securely using hashing algorithms. Payment information might require additional security measures depending on the chosen payment gateway.

**API Authentication:** If external APIs are used, proper authentication methods will be established to securely exchange data.

#### 3.4.5.3 Hardware Interface

As per the plan for the gitgrub microservices application, the hardware parts include the servers for hosting the app and the devices clients use to access it online.

The way the software connects with the hardware involves supporting different web browsers like Chrome, Firefox, and Safari, and making sure the devices have internet access to reach the app.

The physical aspects of this connection include the network linking client devices and servers, the specifications of server hardware, and how much data the server can store.

To communicate between client devices and servers, the system will use HTTP over the internet, with SSL encryption for safety. The interactions between the software and hardware will happen through API endpoints on the servers, with data shared in JSON format.

The hardware needs to meet certain requirements to run the app, outlined in the installation and deployment guides. However, since this is a web app, it can work on any device with an internet connection.

## 4. Supporting Information

Here are some of the supporting information about the gitgrub microservices application:

- The application is a webapp where users can easily order food seeing the items available .
- It's made using microservices, which means it's easy to grow and update.
- We've used modern web techs.
- The app's interface is simple and easy to use, focusing on ordering hassle-free and managing details easy.
- We take security seriously to keep user data safe.
- We've made sure the app is easy to test and maintain, so both customers and developers have a good experience.
- Overall, it's a modern, reliable platform that makes ordering easy in the canteen. It's built to fit the growing canteen and restaurant business in Bangladesh. This app will help the both user and company to order and maintain their customers and their resources. It will help to keep the company data at a single place and maintain all resources very easily.