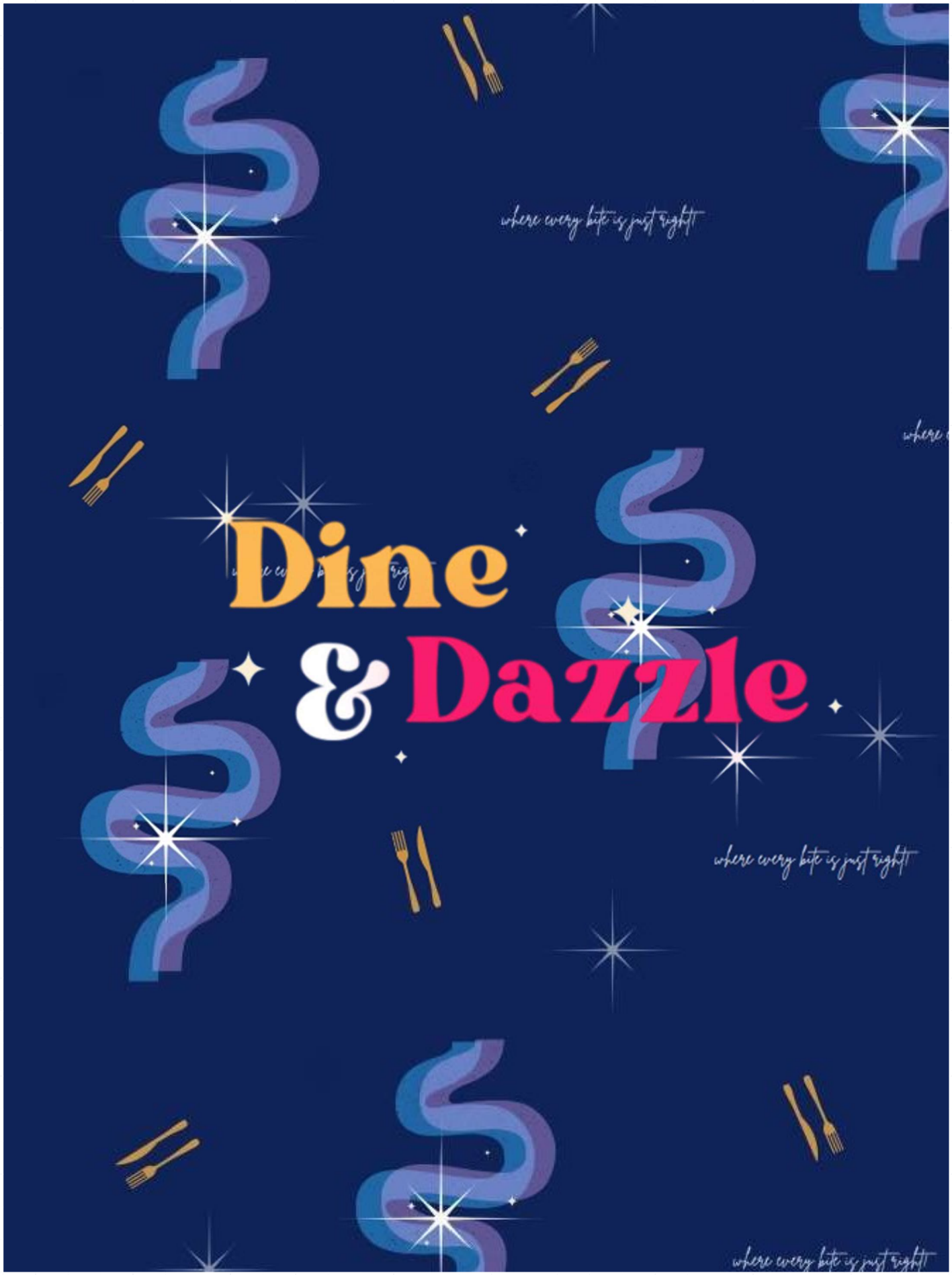

where every bite is just right!
Dine
& Dazzle
where every bite is just right!
where every bite is just right!

Abstract: -

Our project, *Dine and Dazzle*, is a user-friendly website designed to make dining experiences more convenient and enjoyable. With *Dine and Dazzle*, customers can easily browse the restaurant's menu, place online orders, and reserve tables—all from the comfort of their own devices. The website also allows users to create accounts, log in, and view their past orders, making repeat visits quick and seamless. Designed with simplicity and functionality in mind, *Dine and Dazzle* helps the restaurant connect with its customers while streamlining operations and enhancing the overall dining experience.

Table of Contents

1. Introduction: -

1.1 Purpose: -

This is a detailed documentation for our "CSE 232: Advanced Software Engineering" semester project, it includes all the steps from the beginning of the design till the phase of the implementation of what we chose to be a web-based application named "Dine & Dazzle" that provides different services to customers and staff of the dine and dazzle restaurant where the user can create an account, log in to enjoy the website. Users can also view a menu full of different categories, and from these categories, the user can pick whichever item they desire. They can also view their carts and clear them if need be. Furthermore, if the customer decides to reserve a table, they can do that through our website. Also, staff members can check these reservations and add new items to the menu. Consequently, this documented file is mainly handed to our instructors "Dr. Gamal Abdel-Shafy" and "Eng. Sally Edward" to be reviewed and checked to evaluate our work and show them our design and implementation.

1.2 List of definitions: -

Definition	Description
UML	Unified Modeling Language
CRC	Class Responsibility Collaboration
GUI	Graphical User Interface
CRM	Customer relation management

1.3 Scope: -

Dine & Dazzle is a desktop food ordering app designed to provide users with a seamless and efficient way to browse, order, and pay for Dine & Dazzle restaurant meals. The system supports customers and staff, ensuring smooth operations for all stakeholders. Users of the system are required to create an account, which allows them to log in and access personalized features. A menu, organized into various categories such as chicken sandwiches, beef sandwiches, beverages, and add-ons, is available for browsing. Each menu item includes details such as its name and price.

The system enables users to select items from the menu and add them to their virtual cart. Users can modify their cart by adjusting quantities or removing items before proceeding to place their order. Once an order is finalized, it is recorded in the system and linked to the user's account. Each order comprises multiple order items, where each order item represents a specific menu item and its quantity. The system tracks orders of each user and records.

To complete an order, the system offers multiple payment options, including cash on delivery and credit card payments. All payment information is securely processed and associated with the corresponding order.

Staff can manage the menu by adding, updating, or removing menu items and organizing them into appropriate categories. The system also allows administrators to review and update orders, ensuring timely processing and accurate delivery of meals.

Dine & Dazzle's primary focus is to deliver a robust, user-friendly platform for managing food orders, payments, and menu organization, enhancing the dining experience for customers while simplifying operations for administrators.

1.4 List of References: -

1. Roger S. Pressman and Bruce R. Maxim, Software Engineering: A Practitioner's Approach, 9th Edition, McGraw-Hill, ISBN: 978-1260548006, 2019.

2. Ian Sommerville, Software Engineering, 10th Edition, Pearson Education, ISBN: 978-1- 292-09613-1, 2016.

1.5 Report overview: -

This document describes a revolutionary software project, an online website for a food restaurant "Dine & Dazzle, that aims to transform online ordering for Users. The document contains detailed information about the project's design process, including the general description, specific requirements, use-case diagrams, narratives, component diagrams, time plan, research report, OOAD methodologies, software architecture styles, and more will be covered in the following sections.

2. General Description: -

2.1 Product perspective: -

"Dine & Dazzle" is a software that helps its users with any service needed when they order delivery. The software can concurrently be integrated with other systems to better enhance the user experience, for example:

- A social media platform that will give the business a chance to promote its products and services, as well as engage with potential and existing customers by sharing posts, videos, photos, etc.
- A loyalty or reward system that gives the business more options for sales by offering discounts to customers who make frequent or large purchases or refer other customers to the business.
- A chatting system where users can track the state of their order, ask about offers or the history of the restaurant, and leave feedback for their orders and a rating for the restaurant.
- A customer relationship management (CRM) system that allows the business to manage and track the interactions with the customers, such as their contact details, purchase history, preferences, Reviews, etc.

2.2 General Capabilities: -

- User Account creation.
- Customer and staff login.
- View menu
- View different menu categories
- Add items.
- Remove items
- View items in the shopping cart.
- Clear the cart
- Customers can reserve a table.
- Staff can check reservations.
- Staff can add new categories and items.
- Customers can view past orders
- Customers can Pay in cash or credit.
- When paying with credit, customers can enter information.
- View hotline.

2.3 General Constraints: -

This is a web-based software that uses Java and Scene Builder. It has a user-friendly interface that does not require any prior training to use. It is also compatible and scalable for future integration with other systems or expansion to more users. Any updates or improvements can be done easily after the software is launched. The software also ensures the maximum security of the users' data and transactions, preventing any data loss or privacy violation.

2.4 User characteristics: -

There are two expected user roles of such a system:

- The "customer" who is the main user who wants to order food and use other services
- The "staff" who is also a user hired to help the customers and is capable of viewing a reservation list and adding a new menu item or category.

2.5 Environment Description: -

The operational environment for the "Dine & Dazzle" Website encompasses the various aspects that ensure the system's functionality. Users include customers and staff each with distinct roles. The system relies on web and database servers, requiring a compatible operating system, web server software, and database management system. Users access the website from a range of client devices, necessitating compatibility with common web browsers. Network infrastructure, a robust internet connection, and security protocols are vital. Compliance with healthcare data regulations may be required. While the operational environment assumes a secure setting and internet access, external factors such as device capabilities and internet speed may influence performance.

2.6 Assumptions and dependencies: -

Since "Dine & Dazzle" is designed to develop a user-friendly web application that enables hungry customers to create accounts and order food or reserve tables in a restaurant, this makes the CRM system depend on the health department system. Additionally, a loyalty system depends mainly on the CRM management system to enhance the interactions between customers and the system to provide the desired services.

3. Specific Requirements: -

3.1 Capability Requirement: -

1. **Login / Register**: Users should be able to log in with existing accounts or register new ones securely.

2. **View menu**: Users should have the capability to view and browse the menu easily.

3. **View menu categories**: Users should be able to change between different menu categories and view them.

4. **Add items**: While viewing the menu the user should add any item they desire.

5. **Remove items**: If the user second guesses their decision they can remove or decrease the quantity of items they have selected.

6. **View items in the shopping cart**: After the user is done picking their order, they can view their cart and take a look at the items that they want to purchase.

7. **Clear cart**: If the user suddenly wants to change their entire order instead of removing items one by one, they can clear the entire cart.

8. **Reserve a table**: The customer can choose to visit the restaurant instead of ordering takeout; thus, he can reserve a table.

9. **Check reservations**: Staff members can check reservation lists that contain some information about the user.

10. **Add new items or categories**: when the restaurant comes up with a new item or decides to expand to new categories staff members can add these new products to the menu.

11. **View past orders**: The website allows the customers to view any past order they have purchased and reorder it.

12. **Payment choice**: After the customer views their non-empty cart and goes to the payment page, they can choose to pay cash or credit.

13. **Entering card details**: In case the user decides to pay with a credit card, they will enter the card's information and proceed to pay.

14. **View Hotline**: The user can view the restaurant's hotline.

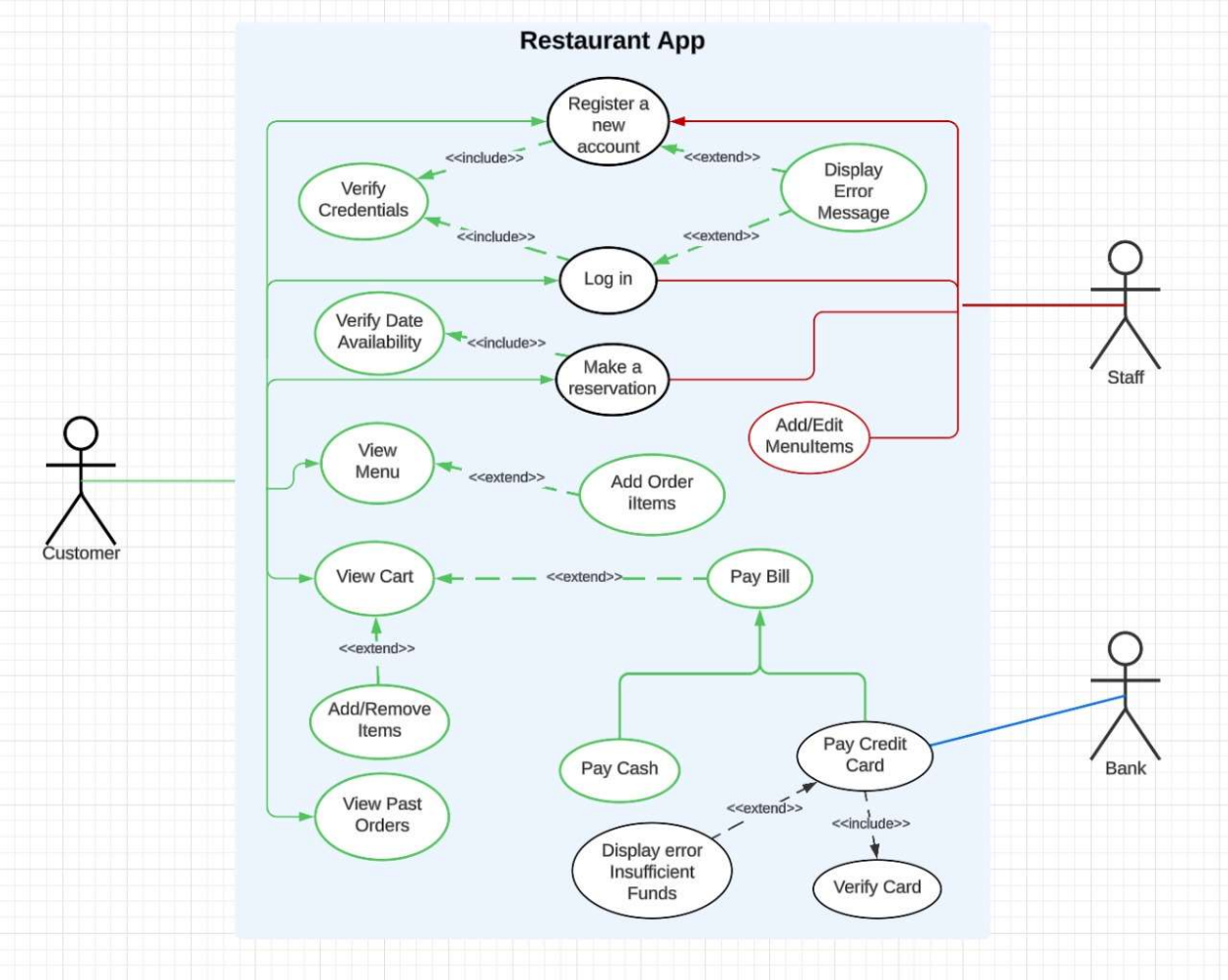
3.2 Constraint Requirements: -

Constraint requirements encompass limitations and restrictions on user requirements that need to be considered during the development of the "Dine & Dazzle" Website:

1. **Regulatory Compliance**: The system must adhere to local and international regulations, especially regarding user personal information and e-commerce regulations for the restaurant.
2. **Browser Compatibility**: The website must be compatible with commonly used web browsers such as Chrome, Firefox, Safari, and Edge.
3. **Internet Connectivity**: Users are expected to have a stable and high-speed internet connection for optimal website performance.
4. **Security Measures**: The website must implement robust security measures, including encryption (HTTPS), firewalls, and access controls, to protect sensitive data and user information.
5. **Availability and Uptime**: The website should aim for high availability, with minimal downtime, especially during peak usage times.

4. Use Case Diagram and the Narrative Description of all use cases: -

4.1 Use Case Diagram: -



4.2 Narrative Description of all use cases: -

4.2.1Registering:

Use Case Name	Register a New Account
Actors	Customer, Staff
Goal	Enable the customer to create a new account in the app.
Preconditions	The customer has downloaded the app and opened it for the first time.
Postconditions	The customer account is successfully registered.
Main Flow	
1.	Customer selects 'Register a New Account.'
2.	System prompts for credentials.
3 include:: Verify Credentials	System verifies credentials.
4.	System creates an account and confirms registration.
Extensions:	
3.1	the customer entered a name that already exists or left a required field empty
3.2	Display an error message

4.2.2 Logging in:

Use Case Name	Log In
Actors	Customer, Staff
Goal	Allow customers to access their accounts.
Preconditions	The customer has a registered account.
Main Flow	
1.	Customer selects 'Log In.'
2.	System prompts for credentials.
3 include:: Verify Credentials	System verifies credentials.
4.	System logs the customer in and displays the main menu.
Extensions:	
3.1	the customer entered a name that already exists or left a required field empty
3.2	Display an error message
Postconditions	The customer has successfully logged in.

4.2.3 Making a reservation:

Use Case Name	Make a Reservation
Actors	Customer, Staff
Goal	Allow customers to reserve a table.
Preconditions	The customer is logged in.
Main Flow	
1.	Customer selects 'Reserve.'
2.	System prompts for date and time and other user details
3 include:: Verify Date availability	System checks availability.
4.	System confirms the reservation.
5.	Staff receives reservation details.
Extensions:	
3.1	The date the customer chose turned out to be fully booked
3.2	A error message displayed prompts the customer to try a different date
Postconditions	A reservation is made and stored in the system.

4.2.4 Viewing menu:

Use Case Name	View Menu
Actors	Customer
Goal	Allow customers to browse the restaurant's menu.
Preconditions	The customer is logged in.
Postconditions	The customer views the menu and can proceed to add items.
Main Flow	
1.	Customer selects 'View Menu.'
2.	System displays the menu with categories.
3.	Customer switches between categories to explore items.

4.2.5 Adding items to the cart:

Use Case Name	Add Items
Actors	Customer
Goal	Allow customers to add menu items to their cart.
Preconditions	The customer is logged in and viewing the menu.
Postconditions	The item is added to the customer's cart.
Main Flow	
1.	Customer chooses a menu item
2.	System increases its quantity
3.	Customer confirms selection, and the item is added to the cart with its quantity

4.2.6 Viewing Cart:

Use Case Name	View Cart
Actors	Customer
Goal	Allow customers to view their cart.
Preconditions	The customer is viewing the menu.
Postconditions	Cart is shown with its full price
Main Flow	
1.	Customer presses the "cart" button in the menu page
2.	System returns and renders the cart list of the customer.
3.	System shows total price of order and the price of each menu Item chosen

4.2.7 Paying bill:

Use Case Name	Pay Bill
Actors	Customer, Bank
Goal	Allow customers to pay for their order.
Preconditions	The customer has items in the cart and is ready to checkout.
Postconditions	The payment is processed, and the order is confirmed.
Main Flow	
1.	Customer presses 'Pay' button
2.	System prompts for payment method. (Cash or credit)
3 include::Verify card	System verifies card through the bank.
4.	System processes the payment.
5.	System confirms payment.
Extensions:	
3.1	the card credentials is false or the card has insufficient funds
3.2	Display error message

4.2.8 Viewing past orders:

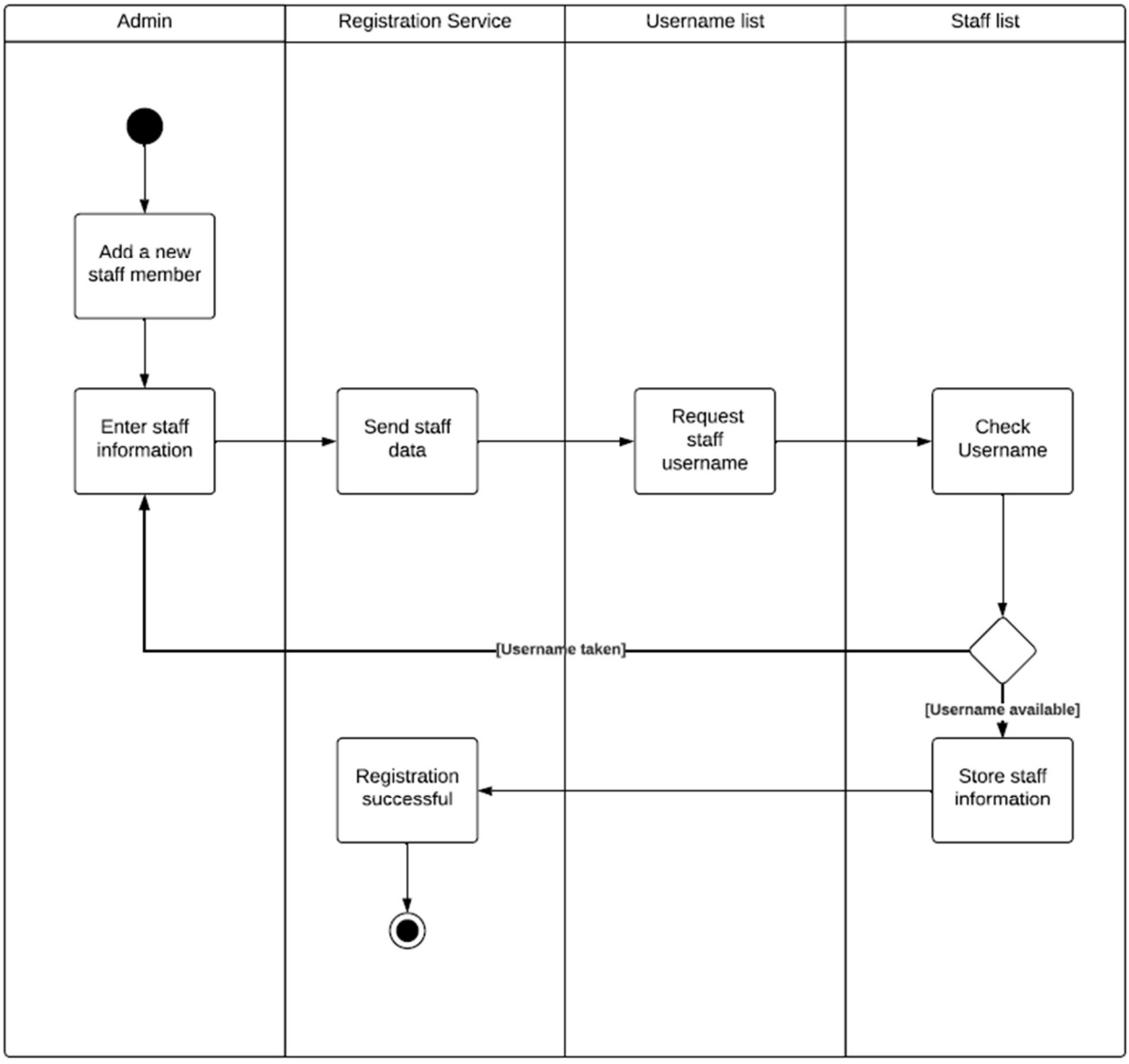
Use Case Name	View Past Orders
Actors	Customer
Goal	Allow customers to view their past orders.
Preconditions	The customer is logged in.
Postconditions	The customer reviews their past orders.
Main Flow	
1.	Customer selects 'View Past Orders.'
2.	System displays past orders with details.

4.2.9 Adding/Editing menu items:

Use Case Name	Add/Edit Menu Items
Actors	Staff
Goal	Allow staff to add or edit menu items.
Preconditions	The staff is logged in with admin privileges.
Postconditions	The menu is updated in the system.
Main Flow	
1.	Staff selects 'Add/Edit Menu Items.'
2.	System prompts to add or edit items.
3.	Staff saves changes.

5. Swimlane and activity diagrams: -

5.1 Staff Register: -



Staff register explanation:

Staff registration

Goal

The Admin Registration Service provides a workflow for admins to register new staff members. It ensures that usernames are unique and stores staff information securely upon successful registration.

Actors

1.Admin

Responsible for initiating and completing the staff registration process.

Swimlane flow:

1. Admin Actions

1.1 Add a New Staff Member

The admin begins the process by selecting the option to add a new staff member.

1.2 Staff Information

The admin provides necessary details for the new staff member, such as:

1.3 Send Staff Data

Once all required details are entered, the admin submits the information to the Registration service.

2. System Actions

2.1: Request Staff Username

The Registration service extracts the submitted username and prepares it to verify its credentials.

2.2: Check Username

The system checks if the requested username is already in use:

If the username is taken: The system notifies the admin to choose a different username.

If the username is available: The process continues to the next step.

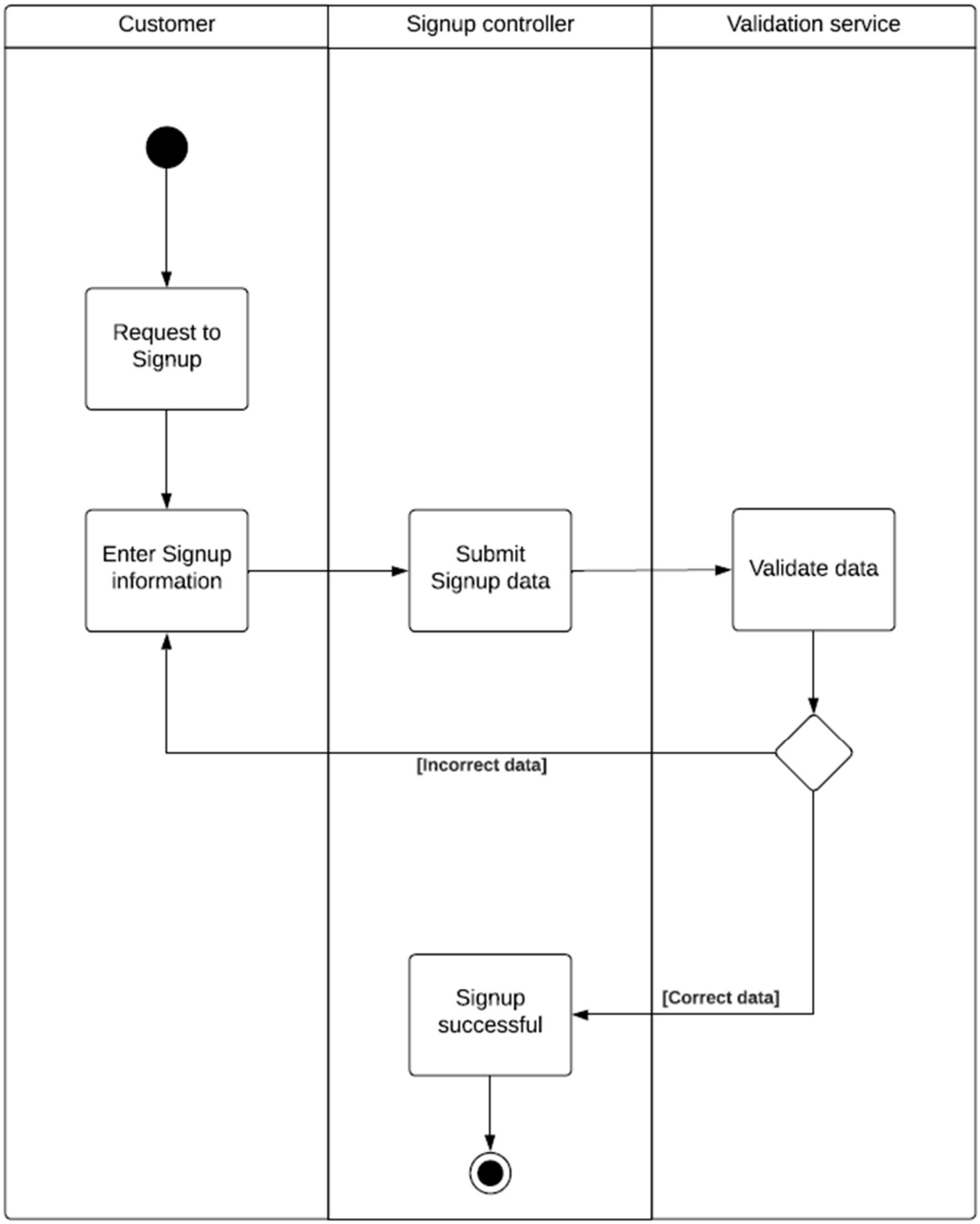
2.3: Store Staff Information

The system securely stores the staff's information into the database.

2.4: Confirm Registration

A success message is sent back to the admin, confirming that the staff member has been registered.

5.2 Customer Registration



Customer Registration Explanation

Overview

This swimlane diagram depicts the process of customer signup involving three primary entities:

1. Customer
2. Signup Controller
3. Validation Service

Each entity is represented as a lane, and the actions or decisions taken by these entities are shown in sequential steps.

Entities and Responsibilities

1. **Customer**

- Represents the end user who initiates the signup process and interacts with the system.
- **Responsibilities**:
    - Requesting to sign up.
    - Entering signup information.

2. Signup Controller

- Serves as the intermediary between the customer and the validation service.
- **Responsibilities**:

    Receiving signup data from the customer.

    Submitting data to the validation service.

    Handling responses from the validation service (valid or invalid data).

    Confirming successful signup.

3. Validation Service

Responsible for validating the correctness and completeness of the data submitted by the customer.

Responsibilities:

Processing the received signup data.

Validating whether the data is correct or incorrect.

Steps in the Process

1. Customer Initiates Signup

The process starts with the customer requesting to sign up.

2. Customer Enters Signup Information

The customer fills out the required signup form with their information.

The information entered is passed to the Signup Controller.

3. Submit Signup Data

The Signup Controller submits the received data to the Validation Service for verification.

4. Validate Data

The Validation Service checks the correctness of the submitted information.

A decision point determines whether the data is correct or incorrect.

5. Incorrect Data

If the data is incorrect, the process loops back to the "Enter Signup Information" step, where the customer is prompted to re-enter the correct details.

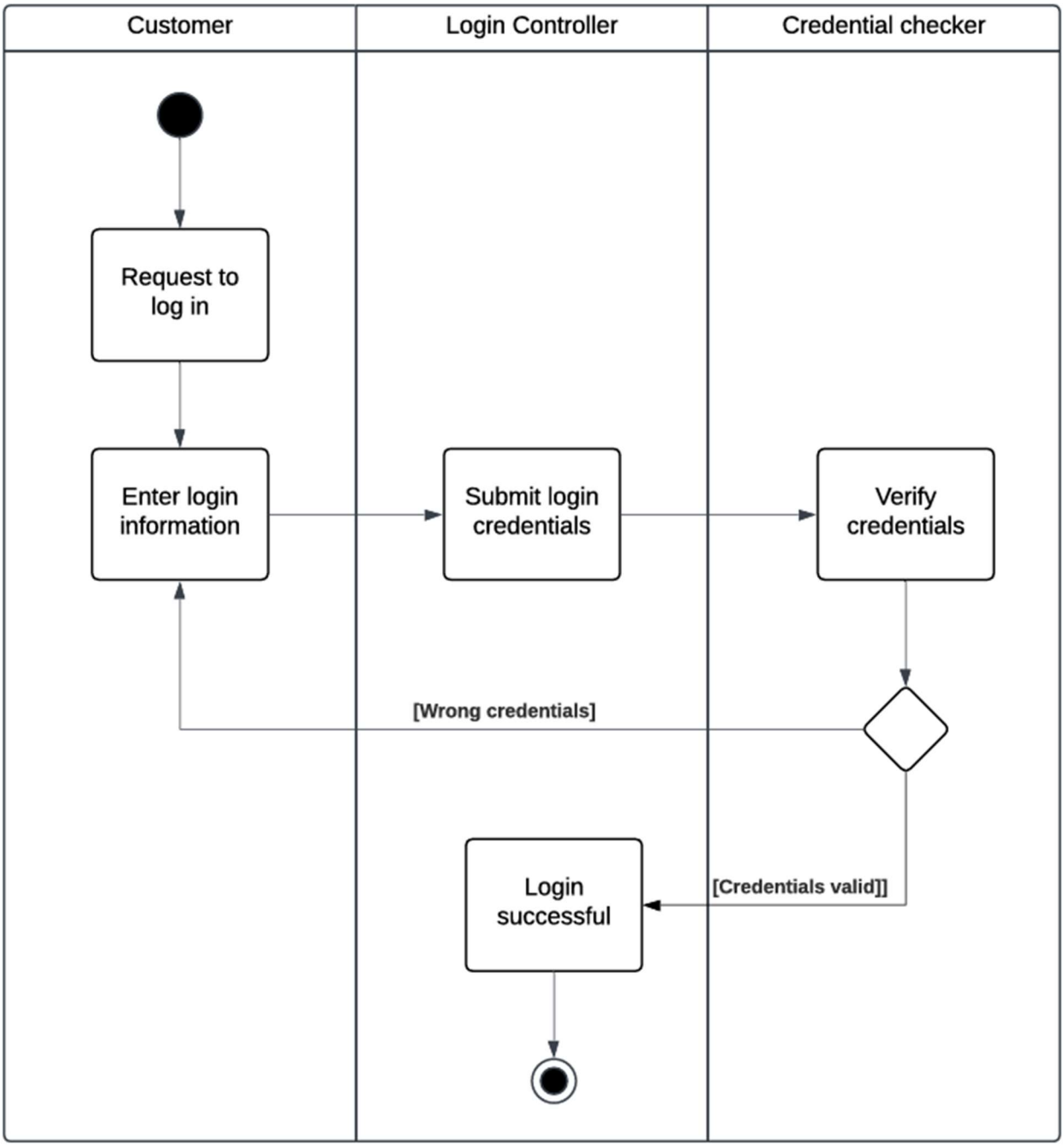
6. Correct Data

- If the data is correct, the Validation Service confirms this and sends a positive response to the Signup Controller.

7. Signup Successful

- The Signup Controller confirms the signup process is successful.
- The process ends successfully for the customer.

5.3 Login swim lane diagram: -



Login Explanation:

Goal

The swim lane diagram for the login flow, is responsible for logging in into the system whether it's a staff or a customer. It checks the credentials entered and then display an output depending on the user input.

Actors & objects

Customer/Staff

The user that will login.

Login Controller

Responsible for the login process.

Credential Checker

Verifies the validity of the submitted credentials against stored data.

Swimlane flow

1. Customer/Staff Actions

1.1: Request to Log In

The user starts by pressing the login button in the page.

1.2: Enter Login Information

The user enters the credentials required.

1.3: Submit Login Credentials

The user submits the entered information to proceed with the authentication process.

2. Login Controller Actions

2.1: Receive Login Request

The Login controller receives the user's login request and credentials.

2.2: Forward Credentials for Verification

The Login controller forwards the submitted credentials to the Credential Checker for validation.

3. Credential Checker Actions

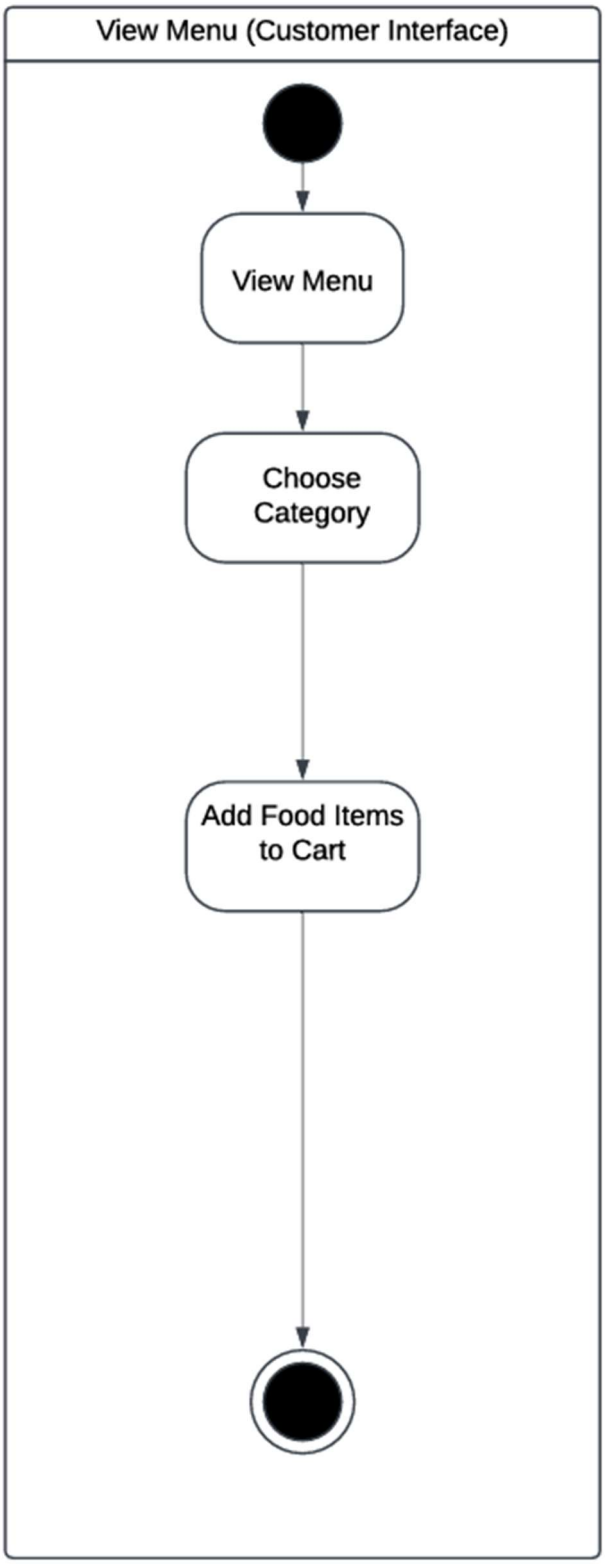
Step 1: Verify Credentials

The Credential Checker compares the submitted credentials against stored accounts.

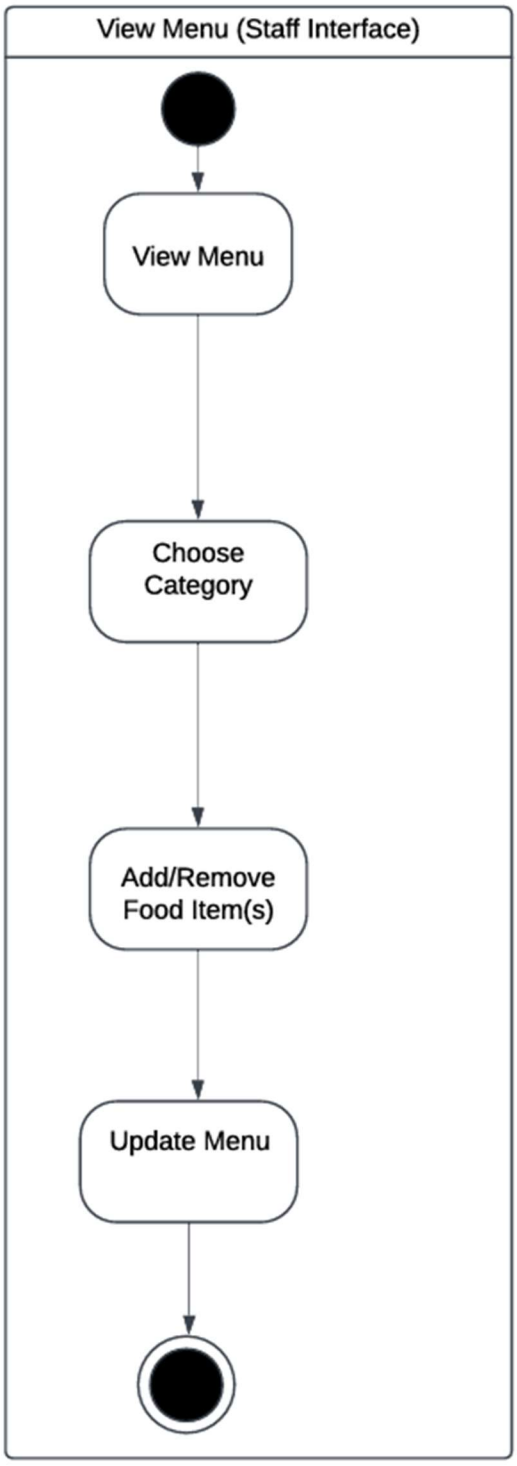
If credentials are invalid: It notifies the Login controller to inform the user of the error.

If credentials are valid: It notifies the Login controller to allow the login process to proceed.

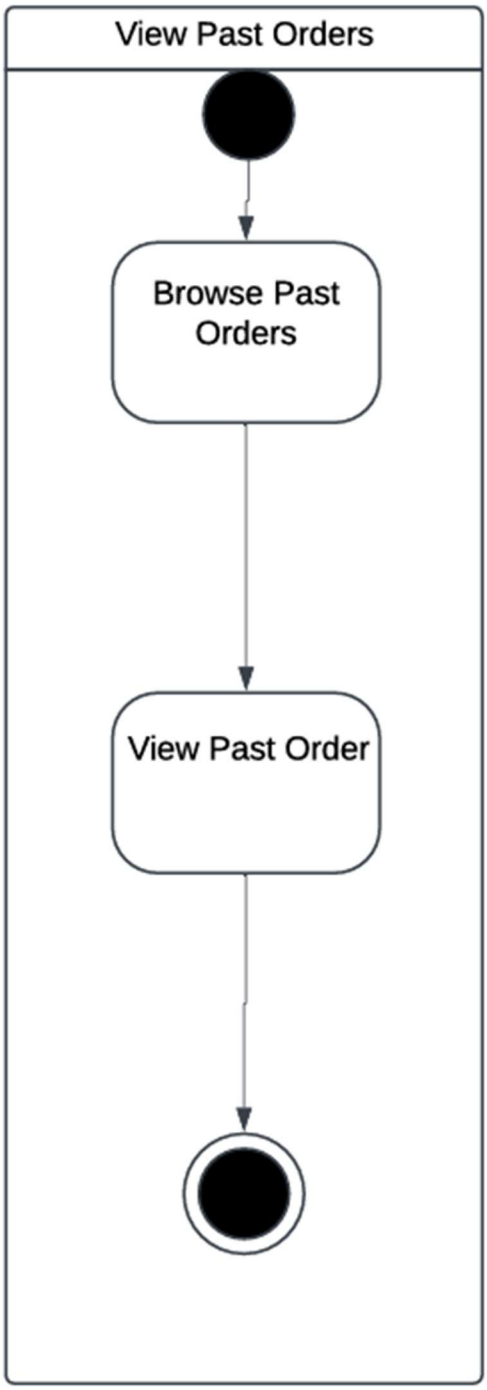
5.4 View Menu (Customer): -



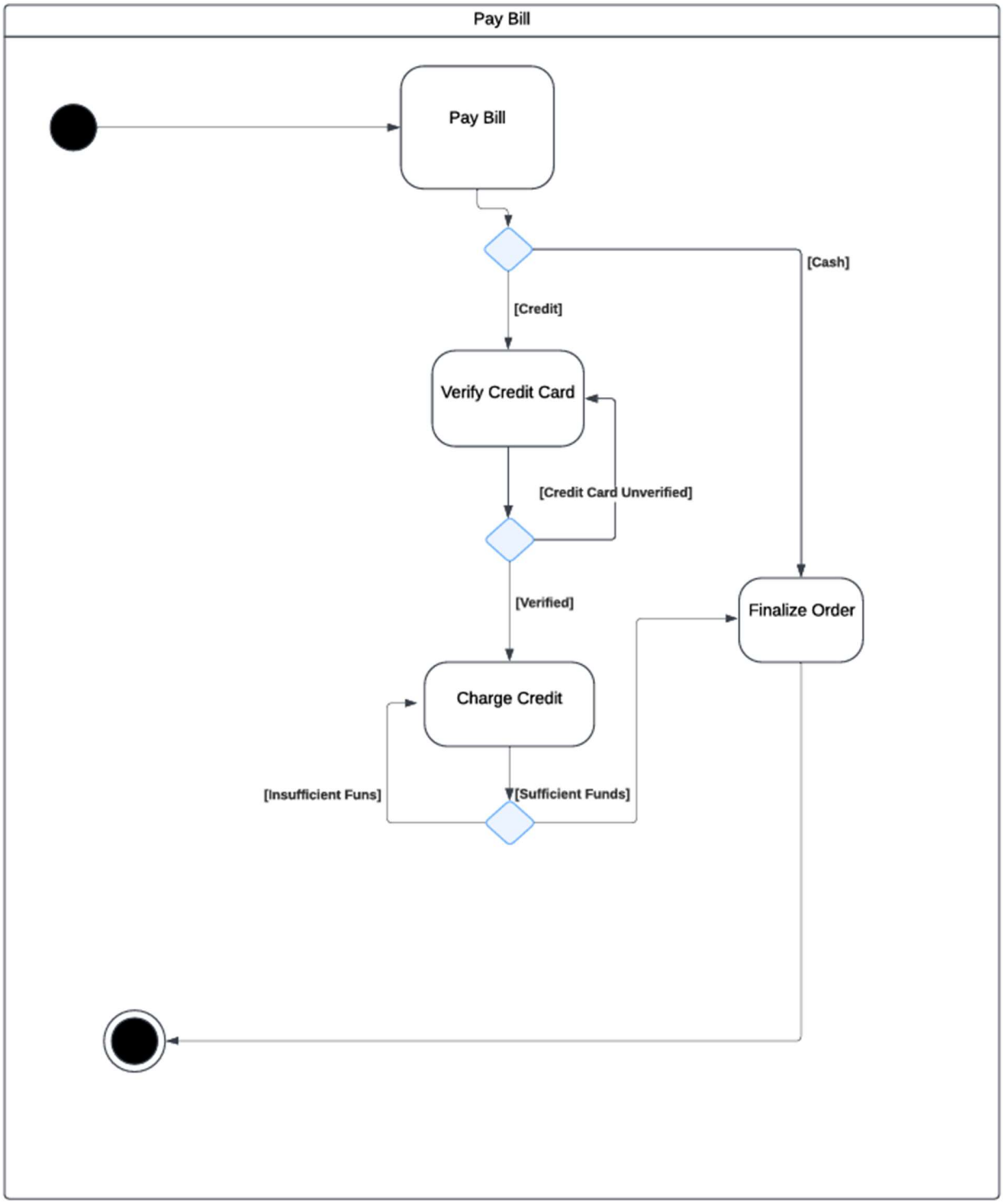
5.5 View menu (Staff): -



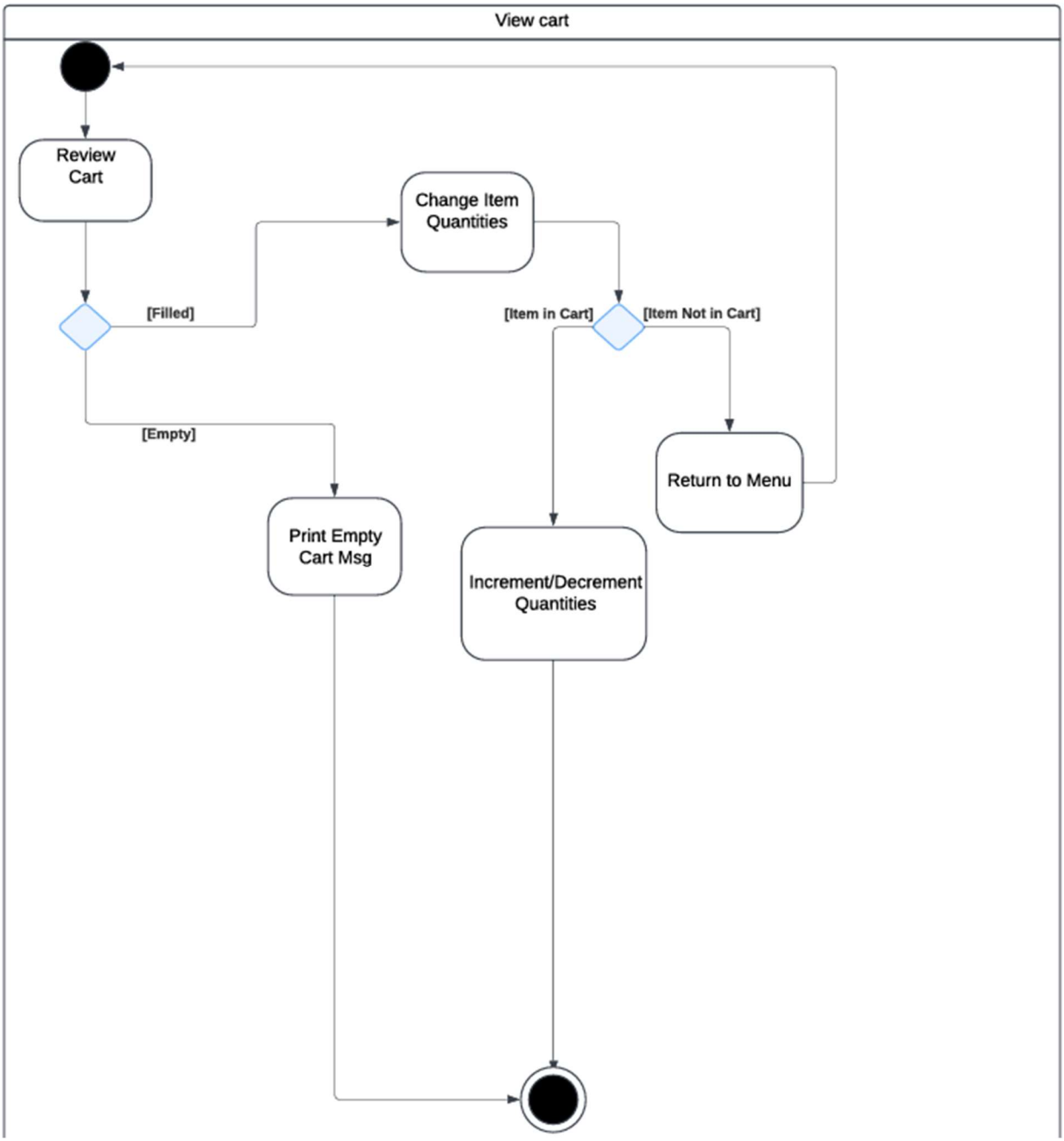
5.6 View past orders: -



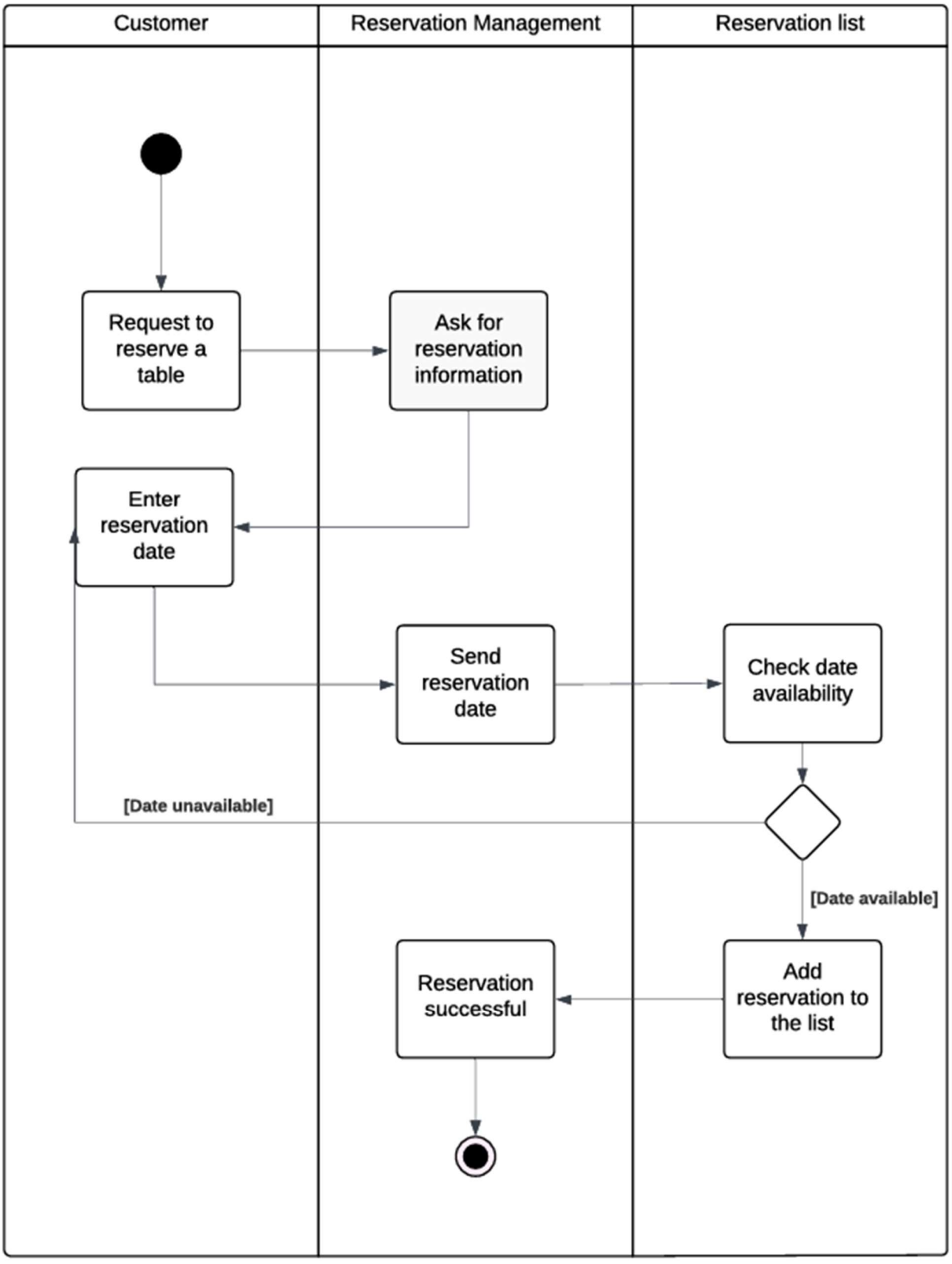
5.7 Pay bill: -


Pay Bill
Pay Bill
[Cash]
[Credit]
Verify Credit Card
[Credit Card Unverified]
[Verified]
Finalize Order
Charge Credit
[Insufficient Funs]
[Sufficient Funds]

5.8 View cart: -


View cart
Review Cart
Change Item Quantities
[Filled]
[Item in Cart]
[Item Not in Cart]
[Empty]
Return to Menu
Print Empty Cart Msg
Increment/Decrement Quantities

5.9 Reservation: -



Reservation explanation: -

Goal

This process enables customers to reserve a table efficiently, ensuring availability is checked before confirmation. It shows the interaction between customers and the reservation system.

Actors

1. **Customer**

   The user requesting a table reservation.

2. **Reservation Management System**

   Handles reservation requests, checks date availability, and manages the reservation list.

Swimlane Breakdown

1. Customer Actions

Step 1: Request to Reserve a Table

The customer initiates the process by requesting to reserve a table.

Step 2: Enter Reservation Date

The customer provides the desired reservation date and other necessary details (e.g., time, number of people).

Step 3: Send Reservation Date

The customer submits the entered information to the system for processing.

2. Reservation Management System Actions

Step 1: Ask for Reservation Information

The system prompts the customer to provide details about the reservation.

Step 2: Check Date Availability

- The system checks whether the requested date is available:

  If the date is unavailable: The system notifies the customer to choose another date.

  If the date is available: The process continues to add the reservation.

Step 3: Add Reservation to the List

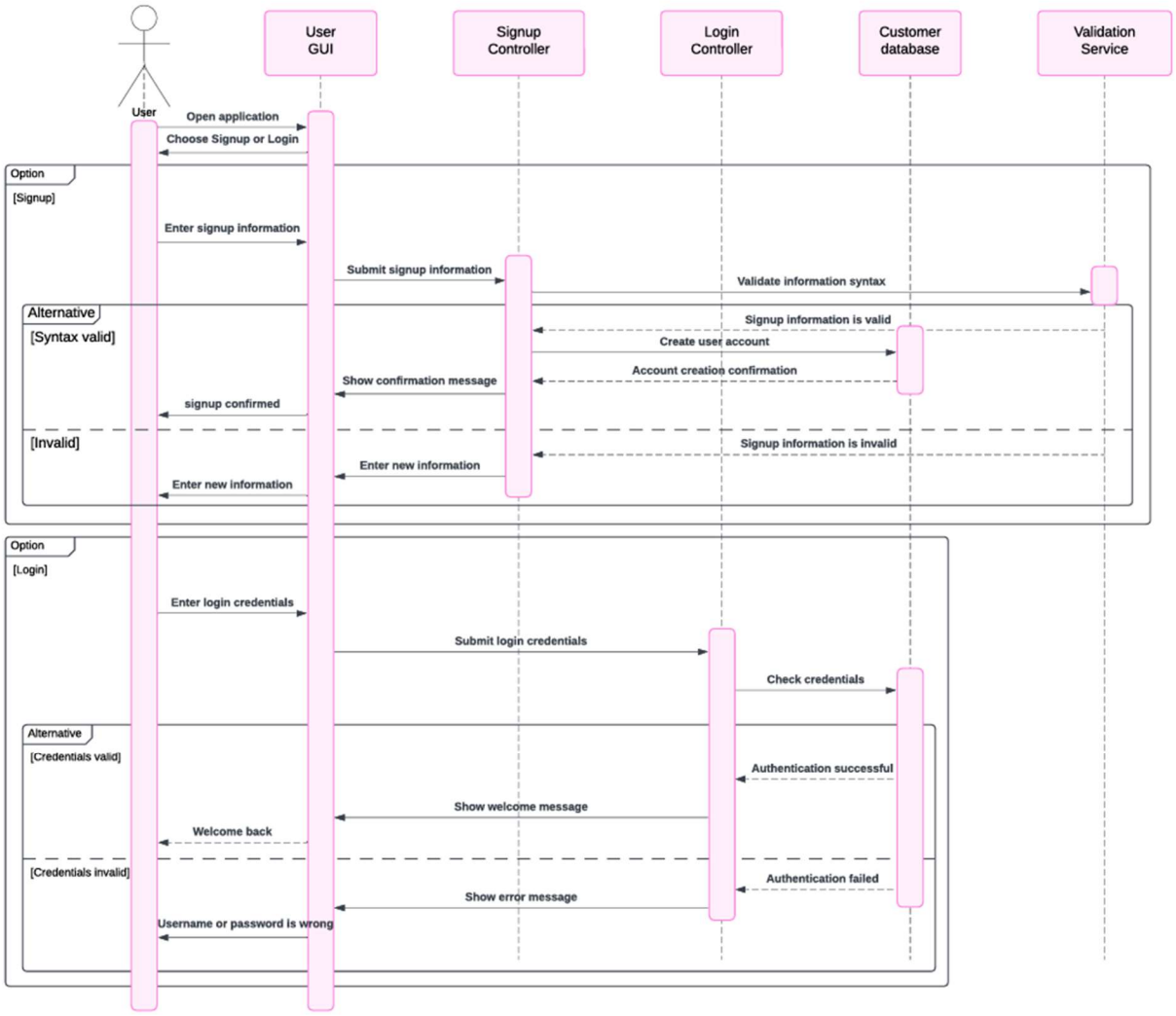
The system records the reservation details in the reservation list.

Step 4: Confirm Reservation

The system sends a confirmation to the customer, indicating that the reservation was successful.

6. Sequence diagrams: -

6.1 Customer & staff login / Sign up: -



Customer & staff login / Sign up explanation: -

Goal:

This sequence diagram illustrates the sequence of the user login and signup. It highlights the interactions between the User GUI, Signup Controller, Login Controller, Validation Service, and the Customer Database. The goal is to register and login user and handling input errors and check data if login.

Actors and Objects:

Actors

1. User

   The actor that chooses between signup and login.

Objects

- User GUI

The interface through which the user interacts with the system to choose between login or signup and input the required information.

- 2. Signup Controller

Handles the signup process, ensuring that new user accounts are created only when the data entered is valid.

- 3. Login Controller

Manages the login process, authenticating users based on their credentials.

- 4. Validation Service

Validates the syntax and correctness of the signup information or login credentials.

- Customer Database

Stores user account details and is queried during authentication and account creation.

Sequence flow

1. Steps:

1. The User opens the application.
2. The user chooses between Signup or Login.

2. Signup Process

1. If the user selects Signup, they enter the required information in the User GUI.
2. The Signup Controller receives the submitted information and sends it to the Validation Service for syntax validation.
3. Alternatives:
   - [Valid Syntax]: The Signup Controller interacts with the Customer Database to create a new user account. Once successful, a confirmation message is displayed on the User GUI.
   - [Invalid Syntax]: An error message is displayed on the User GUI, prompting the user to re-enter valid signup information.

3. Login Process

1. If the user selects Login, they enter their login credentials in the User GUI.
2. The Login Controller forwards the submitted credentials to the Validation Service for authentication.
3. The Validation Service checks the credentials against the Customer Database.

4. Alternatives:

- [Credentials Valid]: The Login Controller displays a welcome message on the User GUI, granting access to the application.
- [Credentials Invalid]: An error message is displayed on the User GUI, prompting the user to re-enter their credentials.

Error Handling

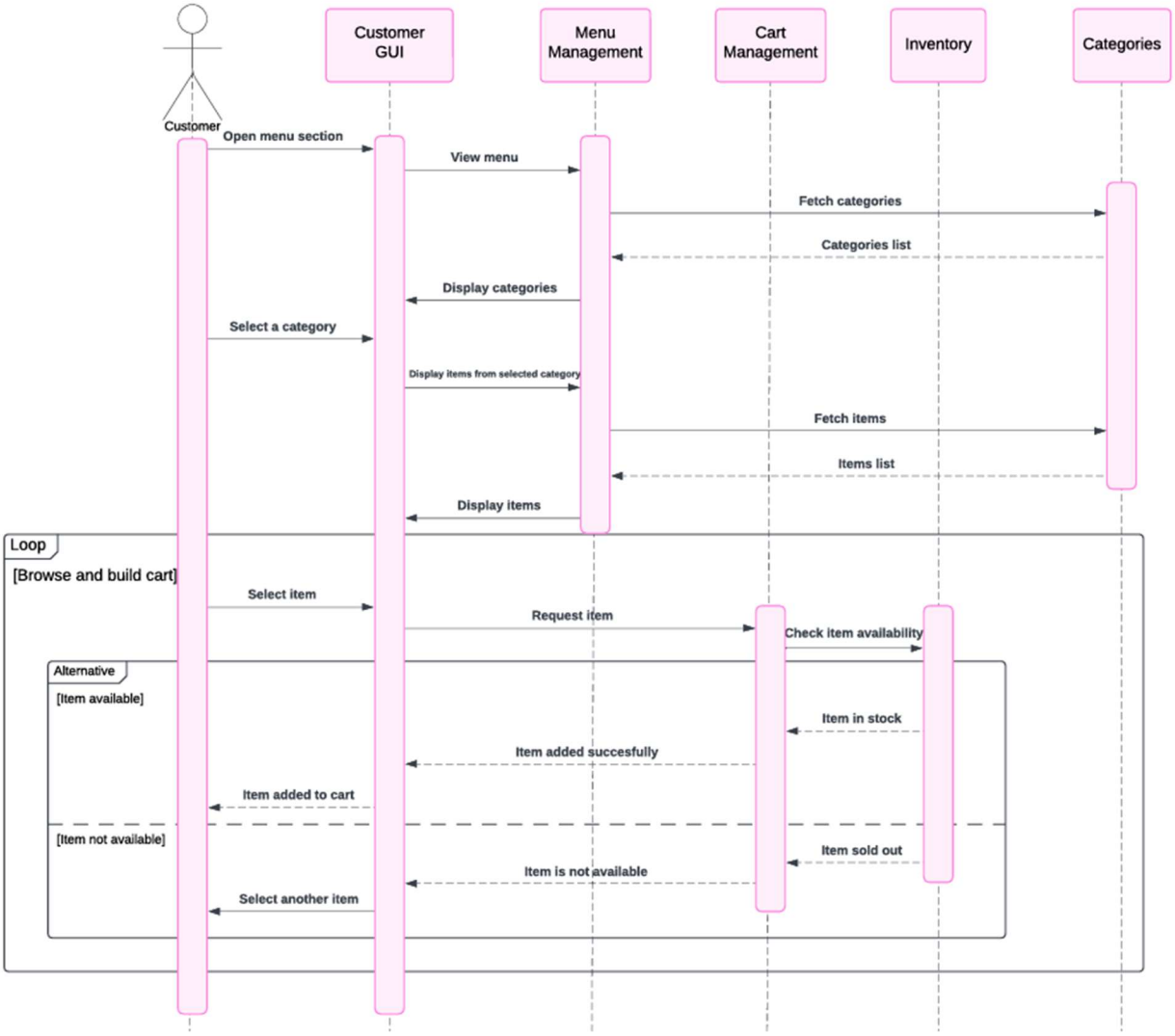
1. Signup Syntax Invalid:

The User GUI prompts the user to re-enter valid information.

2. Login Credentials Invalid:

The User GUI prompts the user to re-enter correct credentials with the message: "Username or password is wrong."

6.2 Viewing menu and ordering items: -



Viewing menu and ordering items explanation: -

Goal:

This sequence diagram illustrates the process of browsing a menu, selecting items, and managing the cart. It details the interactions between the Customer GUI, Menu Management, Cart Management, Inventory and Categories. The goal is to ensure customers can view menu categories, browse items, and add available items to their cart.

Actors and Objects

Actors

1. Customer

The customer browsing the menu and adding items to their cart.

Objects

- Customer GUI

The interface used by the customer to interact with the system for browsing the menu and managing their cart.

- Menu Management

Responsible for requesting fetch items, and categories chosen by the customer and display it.

- Cart Management

Handles the addition of items to the customer's cart, and check if item is available or sold out through inventory

- Categories

Stores and manages the list of menu categories.

- Inventory

Maintains information about item availability and stock status.

Sequence flow:

1. Accessing the Menu

- The Customer opens the menu section in Customer GUI.
- The Customer GUI sends a request to Menu Management to fetch the list of categories.
- Menu Management interacts with the Categories system to retrieve the category list.
- The Customer GUI displays the retrieved categories to the customer.

2. Browsing Items

- The customer selects a category from the displayed list.
- Customer GUI requests Menu Management to fetch the items under the selected category.
- Menu Management retrieves the item list from the Inventory system.
- The Customer GUI displays the list of items to the customer.

3. Adding Items to the Cart

- The customer selects an item to add to the cart.
- Customer GUI requests Cart Management to check the item's availability.
- Cart Management queries the Inventory system to verify stock status.

4. Alternatives:

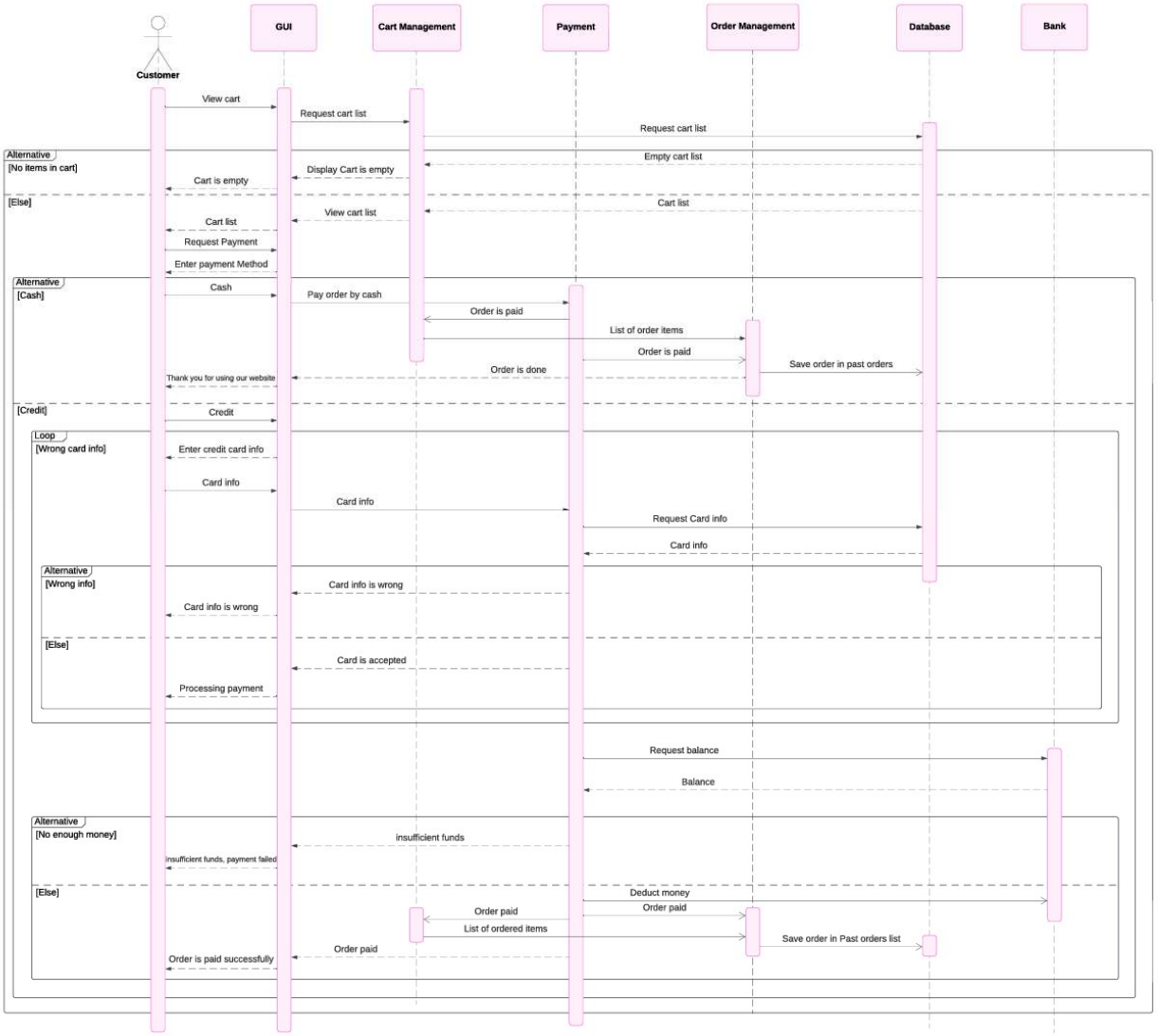
[Item Available]:

The item is added to the cart, and the Customer GUI displays a confirmation message: "Item added successfully."

[Item Not Available]:

The Customer GUI displays an error message: "Item is not available," prompting the customer to select another item.

6.3 View cart: -

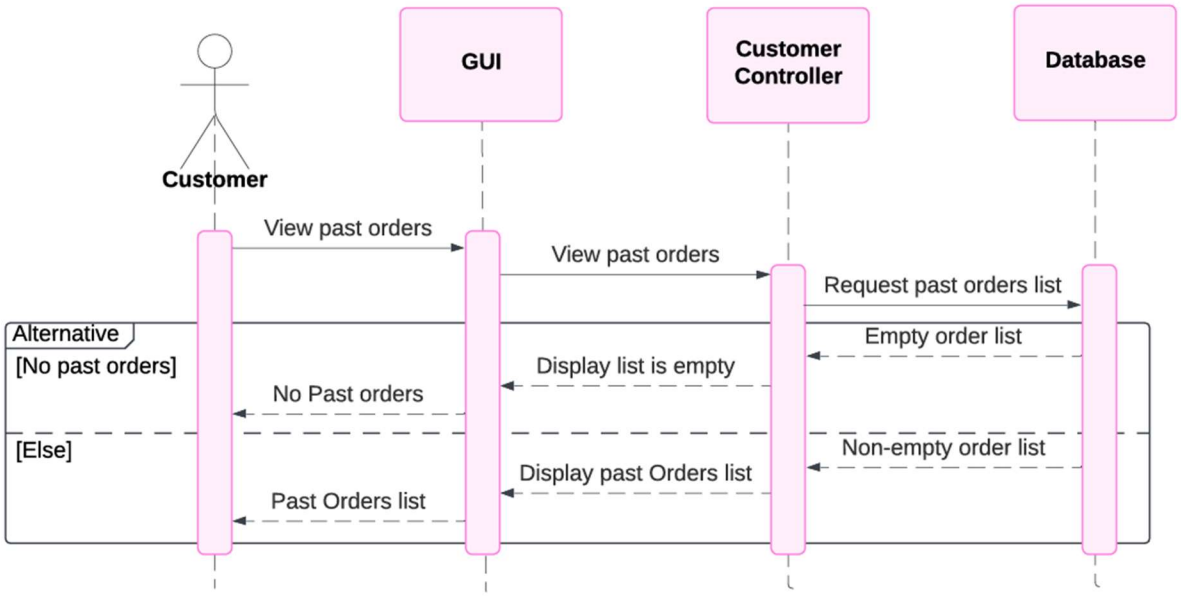

Customer
GUI
Cart Management
Payment
Order Management
Database
Bank
View cart
Request cart list
Request cart list
Alternative
[No items in cart]
Empty cart list
Display Cart is empty
Cart is empty
[Else]
Cart list
View cart list
Cart list
Request Payment
Enter payment Method
Alternative
[Cash]
Cash
Pay order by cash
Order is paid
List of order items
Order is paid
Save order in past orders
Order is done
Thank you for using our website
[Credit]
Credit
Loop
[Wrong card info]
Enter credit card info
Card info
Card info
Request Card info
Card info
Alternative
[Wrong info]
Card info is wrong
Card info is wrong
[Else]
Card is accepted
Processing payment
Request balance
Balance
Alternative
[No enough money]
insufficient funds
insufficient funds, payment failed
[Else]
Deduct money
Order paid
Order paid
List of ordered items
Save order in Past orders list
Order paid
Order is paid successfully

View cart explanation: -

This sequence diagram illustrates the interactions involved in the **restaurant application's View Cart and Payment use cases**. Here's a brief explanation of the key elements:

1. **Customer Actions**:
   The customer can view the cart to check the selected items, their quantities, and the total cost. If the cart is empty, they are notified and redirected to the menu.
2. **Cart Management**:
   Handles requests to display the cart's content, ensuring the list of ordered items is presented to the customer.
3. **Payment Process**:
   The customer chooses a payment method—cash or credit card. If paying by credit card, the system validates the entered card information:
   - **Loop**: If the card info is incorrect, the customer is prompted to re-enter it.
   - **Alternative**: If the card has insufficient funds, the payment fails.
   - Once the card is valid and has sufficient balance, the payment is processed successfully.
4. **Order Management**:
   After successful payment, the order is saved in the system as part of the customer's past orders for future reference.
5. **Bank Integration**:
   For credit card payments, the system interacts with the bank to validate card details and process the transaction.

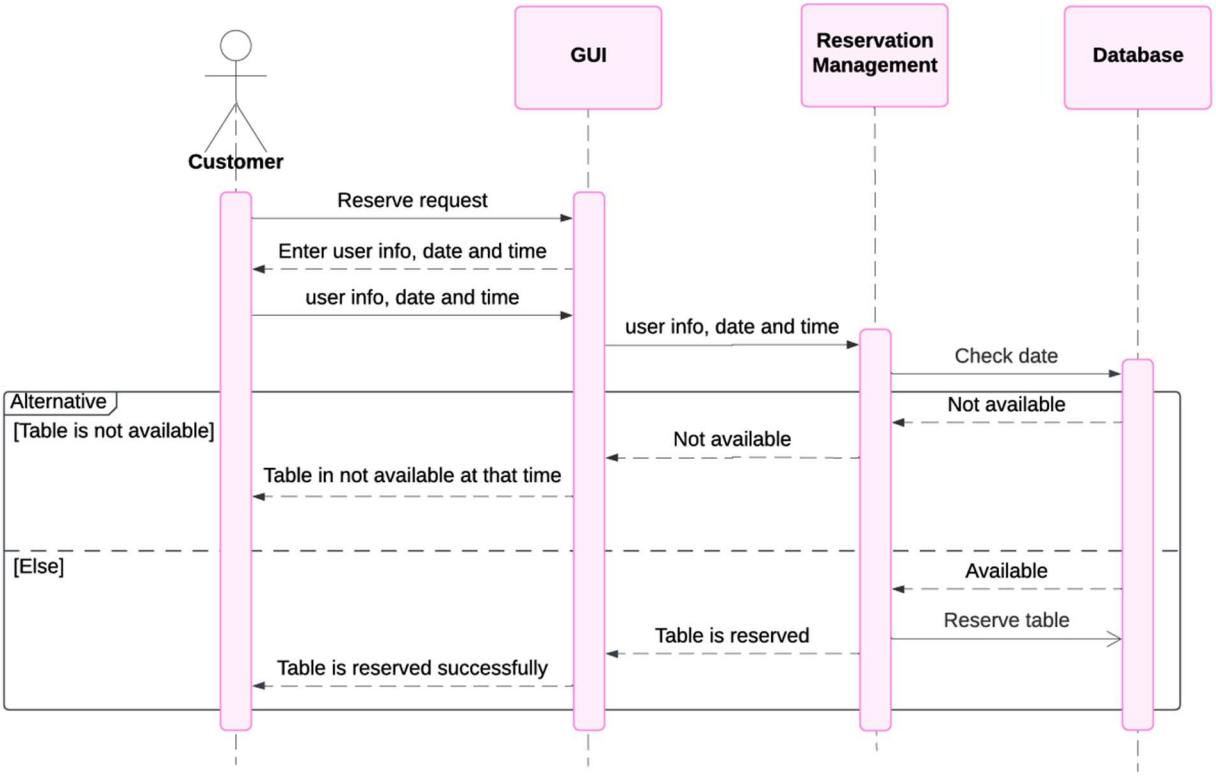
6.4 View past orders: -



View past orders Explanation: -

1. **Customer Actions**:
   The customer initiates the process by requesting to view past orders through the GUI.
2. **System Processing**:
   - The **Customer Controller** retrieves the list of past orders from the database.
   - **Alternative Scenarios**:
     - If no past orders are found, a message is displayed informing the customer.
     - Otherwise, the list of past orders, including details like date, quantity, and type, is displayed to the customer.

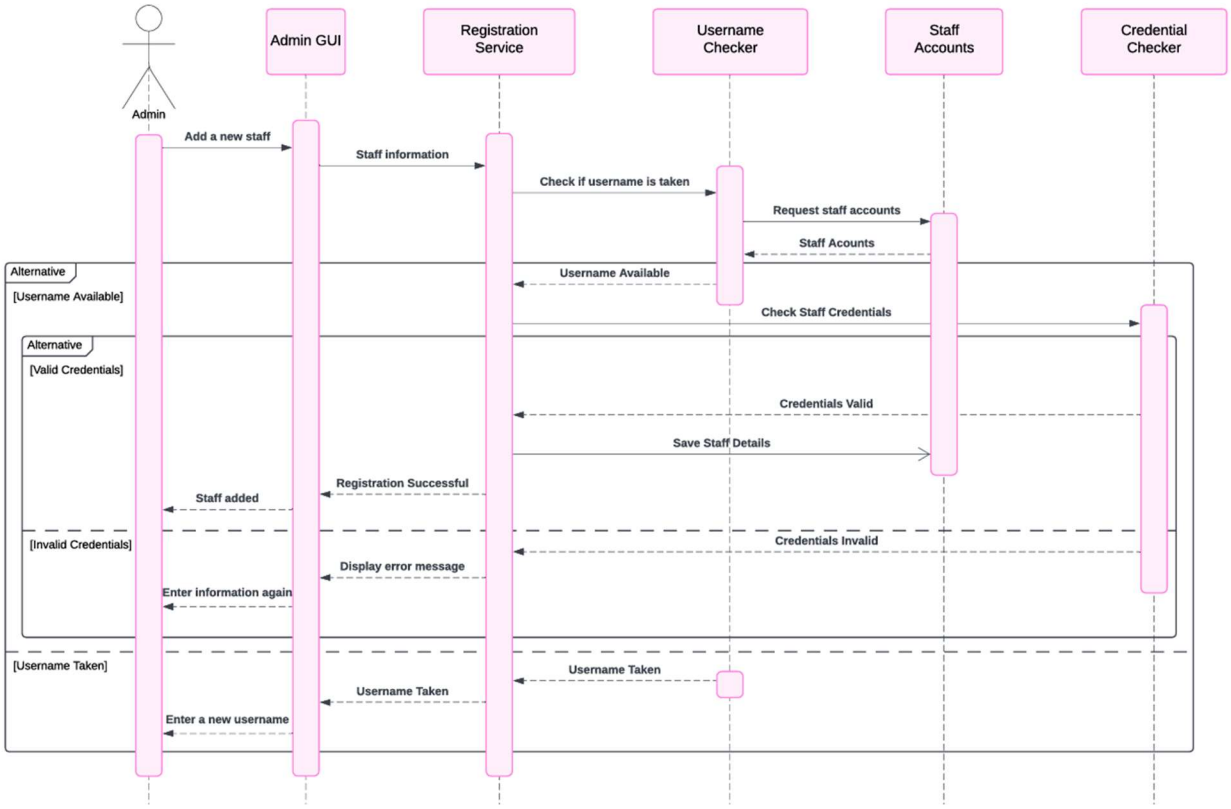
6.5 Reservation: -



Reservation explanation: -

1. **Customer Actions**:
   The customer requests a table reservation by entering user information, date, and time.
2. **System Processing**:
   - The **Reservation Management** system checks the availability of a table for the specified time using the database.
   - **Alternative Scenarios**:
     - If a table is unavailable, a message is displayed to the customer.
     - Otherwise, the table is reserved successfully, and the reservation is stored in the database.

6.6 Admin registering staff: -



Goal:

This sequence diagram describes the process of adding new staff members to the system. It highlights the interactions between the Admin GUI, Registration Service, Username Checker, Credential Checker, and Staff Accounts systems. The primary goal is to ensure that new staff accounts are registered only if the chosen username is available and valid credentials are provided, and only the admin is available to add a new staff member.

Actors and Objects

1. **Admin**

The actor responsible for requesting and entering the new staff member's information

2. **Admin GUI**

The interface is used by the Admin to input staff details, including usernames and credentials.

3. **Registration Service**

Acts as the core registration process, coordinating the entire registration process.

4. **Username Checker**

A service responsible for verifying if the entered username is available or already taken by another staff member.

5. **Credential Checker**

A system that validates the staff credentials provided by the Admin.

6. **Staff Accounts**

The database that stores and manages staff account details.

Sequence flow:

1. **Username Validation**

1. The admin initiates the process by entering staff information, including the username.

2. The admin GUI forwards the entered details to the Registration Service.

3. The registration Service invokes the Username Checker to verify if the entered username is available.

4. **Alternatives:**

- [Username Available]: The process proceeds to credential validation.
- [Username Taken]: An error message is displayed on the Admin GUI, prompting the admin to enter a new username.

2. **Credential Validation**

1. If the username is available, the Registration Service checks the provided credentials by invoking the Credential Checker.

2. **Alternatives:**

- [Valid Credentials]: The process proceeds to save the staff details.
- [Invalid Credentials]: An error message is displayed on the Admin GUI, prompting the admin to re-enter the credentials.

3. **Staff Account Creation**

1. Once the credentials are validated successfully, the Registration Service interacts with the Staff Accounts system to save the new staff details.

2. A confirmation message, "Registration Successful," is displayed on the Admin GUI.

Error Handling

1. Username Taken:

The Admin GUI prompts the admin to enter a new username.

2. Invalid Credentials:

The Admin GUI prompts the admin to re-enter valid credentials.

7. Noun Extraction and CRC Cards: -

7.1 Noun Extraction:

Dine & Dazzle is a desktop food ordering app designed to provide users with a seamless and efficient way to browse, order, and pay for meals from Dine & Dazzle restaurant. The system supports both customers and staff, ensuring smooth operations for all stakeholders. Users of the system are required to create an account, which allows them to log in and access personalized features. A menu, organized into various categories such as chicken sandwiches, beef sandwiches, beverages, and add-ons, is available for browsing. Each menu item includes details such as its name and price. The system enables users to select items from the menu and add them to their virtual cart. Users can modify their cart by adjusting quantities or removing items before placing their order. Once an order is finalized, it is recorded in the system and linked to the user's account. Each order comprises multiple order items, where each order item represents a specific menu item and its quantity. The system tracks the orders of each user and records them. The system offers multiple payment options to complete an order, including cash on delivery and credit card payments. All payment information is securely processed and associated with the corresponding order. Staff can manage the menu by adding, updating, or removing menu items and organizing them into appropriate categories. The system also allows administrators to register other staff members who have been newly hired to the system. Dine & Dazzle's primary focus is to deliver a robust, user-friendly platform for managing food orders, payments, and menu organization, enhancing the dining experience for customers while simplifying operations for administrators.

Excluded nouns: -

- **Dine& Dazzle**

  Reason: Out of scope (name of the system, not a class).

- **Desktop**

  Reason: No physical existence (part of the platform, not part of the model).

- **App**

  Reason: Out of scope (represents the system, not a class).

- **Food**

  Reason: Redundant (captured as menu items).

- **Ordering**

  Reason: Redundant (action captured under order).

- **Restaurant**

  Reason: Out of scope (represents the context, not part of the system).

- **Way**

  Reason: No physical existence (an abstract concept).

- **Browsing**

  Reason: No physical existence (action, not a class).

- **Operations**

  Reason: No physical existence (an abstract concept).

- **Stakeholders**

  Reason: Out of scope (general participants, not system-specific).

- **Account**

  Reason: Equivalent to User (part of user attributes).
- **Features**

  Reason: No physical existence (an abstract concept).
- **Beef, Chicken, Side Items, Add-ons**

  Reason: Redundant (captured under categories).
- **Details**

  Reason: Redundant (part of menu item attributes).
- **Name, Price**

  Reason: Redundant (attributes of the menu item).
- **Cart**

  Reason: Equivalent to Order (part of the ordering process).
- **Quantities**

  Reason: Redundant (attribute of order item).
- **Cash, Delivery, Credit Card Payments**

  Reason: Equivalent to Payment (specific instances).
- **Focus, Platform, Menu Organization, Experience**

  Reason: No physical existence (abstract concepts).
- **Customers**

  Reason: Equivalent to User (another term for end-users).

Classes identified: -

- **User**

From: users, user

Reason: Represents the end-user interacting with the system.

- **Menu**

From: menu

Reason: Represents the collection of menu items available for browsing.

- **Categories**

From: categories

Reason: Groups items into logical types (e.g., beef, chicken, side items).

- **Menu Item**

From: menu item

Reason: Represents individual dishes or items within the menu.

- **Order Item**

From: order item

Reason: Represents a specific menu item within an order, including quantity.

- **Order**

From: order, orders

Reason: Represents the collection of order items placed by a user.

- **Payment**

From: payment options, payment information

Reason: Represents payment details and method (e.g., cash, credit card).

- **Customer**

From: Customer

Reason: Represents the customer who will perform all customer services (e.g. add items to the cart and pay)

- **Staff**

From: Staff

Reason: represents staff members responsible for updating the menu, checking reservations, and registering other staff members.

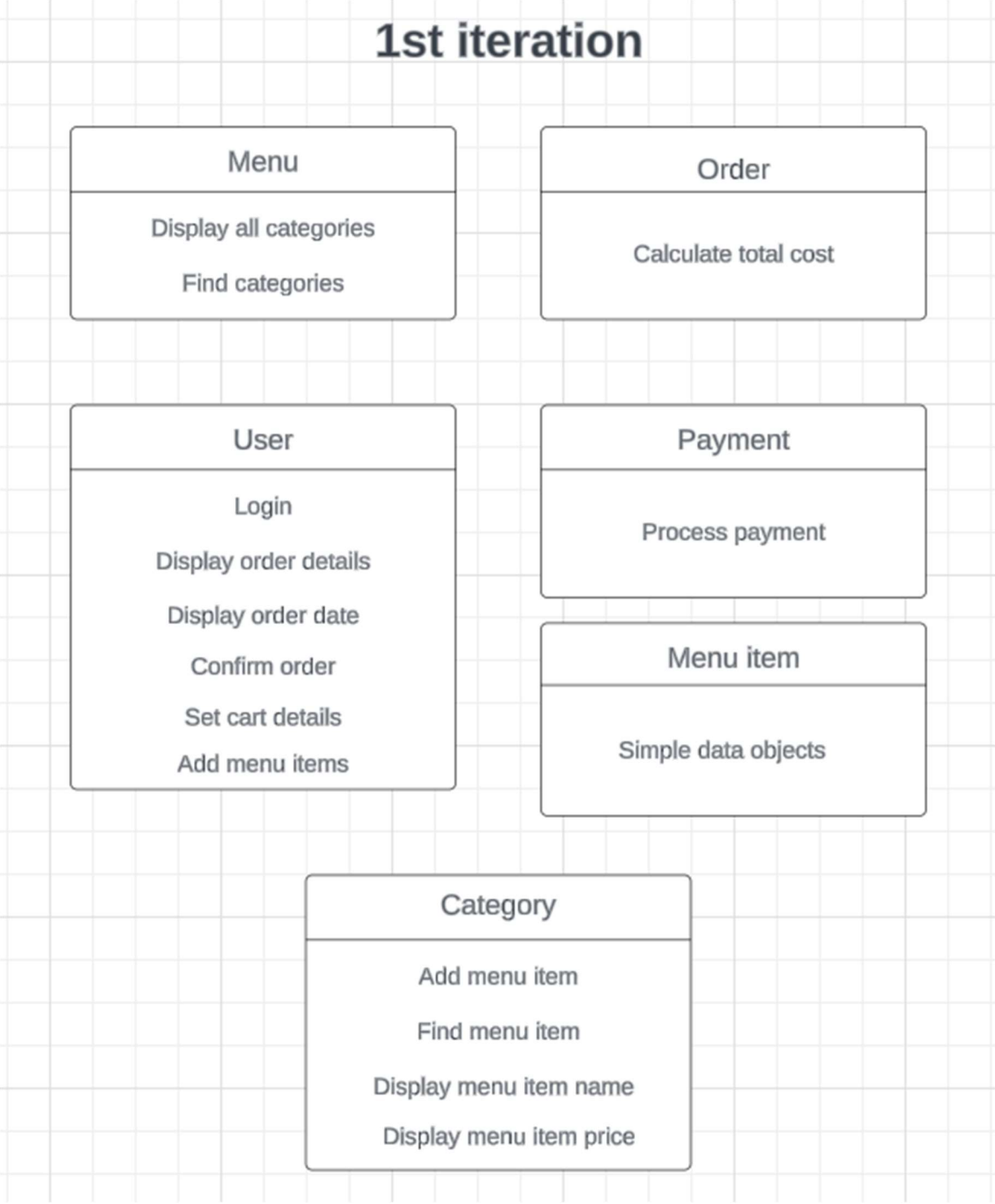
- **Cart**

From: Cart

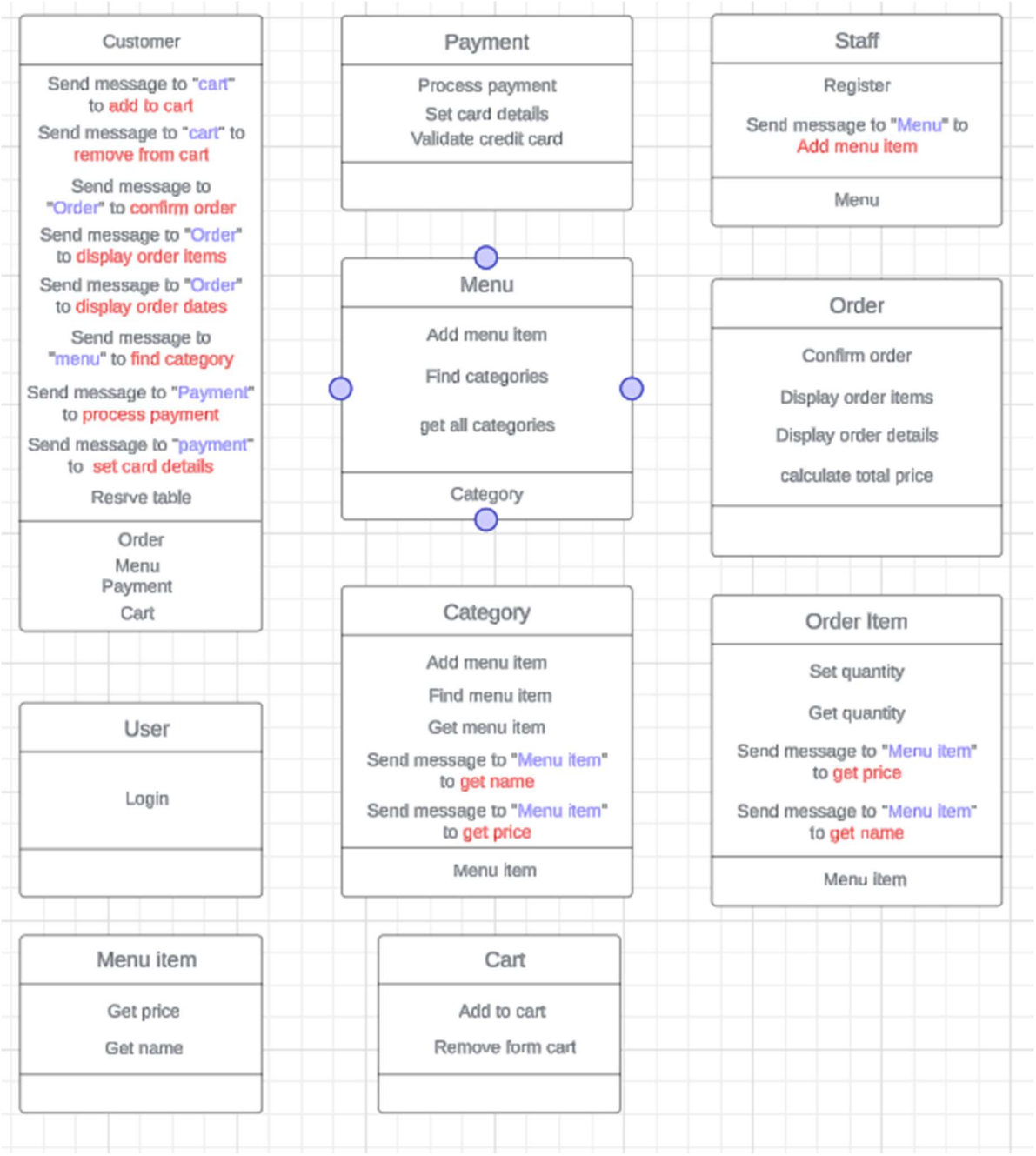
Reasons: where ordered items of users will be stored before paying.

7.2 CRC Cards: -

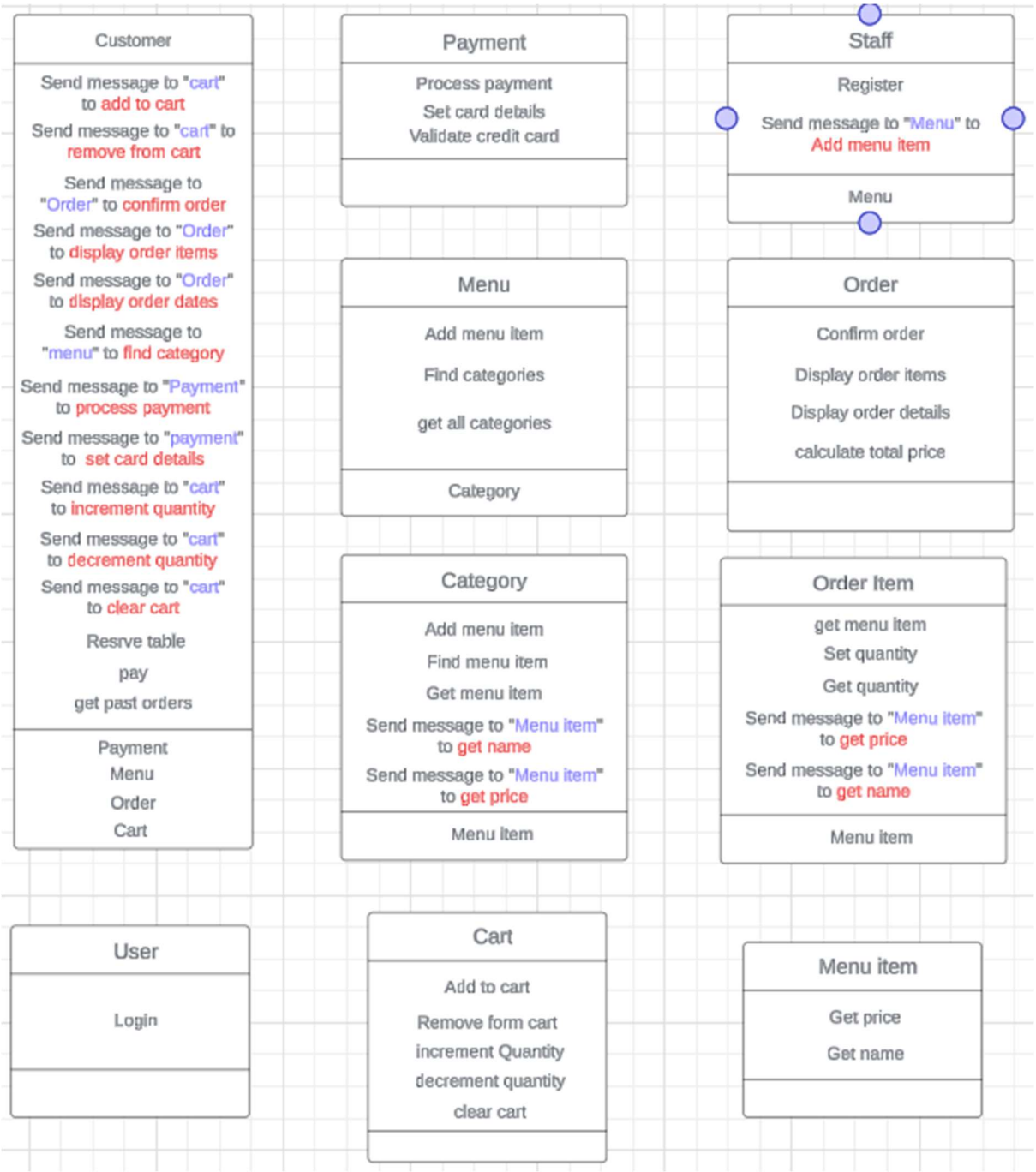
1st Iteration:



2nd iteration:



3rd iteration:



8. OOAD Methodologies: -

8.1 Responsibility Driven Design: -

RDD focuses on identifying objects in the system and assigning clear responsibilities to each one. This ensures that each object has a well-defined role, promoting cohesion and minimizing unnecessary dependencies.

1. Customer:

- Responsibilities:

  - Login/Sign-Up.
  - Browse the menu.
  - Manage orders (add/remove items, view cart, view past orders).
  - Reserve tables.
  - Make payments.

2. Staff:

- Responsibilities:

  - Login.
  - Check reservations.
  - Add new menu items.
  - Register other staff users.

3. Menu:

- Responsibilities:

  - Store information about menu items.
  - Allow updates (adding/removing items) by staff.
  - Serve menu details to customers.

4. Order

- Responsibilities:
  - Track items added/removed by customers.
  - Calculate the total cost of the order.

5. Reservation

- Responsibilities:
  - Record the table reservation requests.
  - Check the availability of tables.

6. Payment

- Responsibilities:
  - Handle payment transactions (cash or credit).
  - Validate credit card information.
  - Confirm payment status.

8.2 Behavior Driven Development Scenarios: -

Scenario 1: New Customer Signing Up

Given the customer is on the sign-up page

When the customer enters their name, email, phone number, and password

And the customer clicks the "Sign Up" button

Then the customer account should be created

And the customer should be logged in automatically

And the system should send a confirmation email

Scenario 2: Returning Customer Logging In

Given the customer is on the login page

When the customer enters their email and password

And the customer clicks the "Log In" button

Then the customer should be logged in

And the customer should be redirected to their account dashboard

2. View Menu

Scenario: Customer Viewing the Menu

Given the customer is logged in

When the customer navigates to the "Menu" section

Then the menu should be displayed with categories such as appetizers, mains, and desserts

And each item should display the name, description, and price

3. Add/Remove Items to Cart

Scenario: Customer Adding Items to Cart

Given the customer is logged in

And the customer is viewing an item on the menu

When the customer clicks "+"

Then the item should be added to the customer's cart

And the cart total should be updated to reflect the new item

Scenario: Customer Removing Items from Cart

Given the customer has items in their cart

When the customer clicks "-" next to an item

Then the item should be decremented from the cart

And the cart total should be updated to reflect the decrement

4. Make a Payment (Cash or Credit)

Scenario: Customer Paying with Cash

Given the customer has items in their cart

When the customer proceeds to checkout

And selects "Cash" as the payment option

Then the system should confirm the total amount

And the order should be placed as "Paid by Cash"

And the customer should receive an order confirmation

Scenario: Customer Paying with Credit Card

Given the customer has items in their cart

When the customer proceeds to checkout

And selects "Credit Card" as the payment option

And the customer enters valid credit card information

Then the system should process the payment securely

And the order should be placed as "Paid by Credit Card"

And the customer should receive an order confirmation

5. Reserve a Table

Scenario: Customer Making a Reservation

Given the customer is on the "Reserve a Table" page

When the customer selects the date, time, and enters email address and phone number

And clicks "Reserve"

Then the system should display available time slots

And if a time slot is available, it should confirm the reservation

And the customer should receive a confirmation email

Scenario: Staff Member Logging In

Given the staff member is on the login page

When the staff enters their username and password

And clicks the "Log In" button

Then the staff member should be logged into the system

And the staff member should be redirected to the staff homepage

2. Check Reservation List

Scenario: Staff Checking Reservations

Given the staff member is logged in

When the staff navigates to the "Reservations" section

Then the system should display all upcoming reservations with details like date, time, and number of guests

And the reservations should be sorted by date and time

3. Add Menu Items

Scenario: Staff Adding a Menu Item

Given the staff member is logged in

And the staff has access to the "Add Menu Item" section

When the staff enters the menu item details (name, description, price)

And clicks "Add Item"

Then the item should be added to the menu

And the menu should be immediately updated and visible to customers

4. Register Other Staff Users

Scenario: Staff Registering New Staff

Given the staff member is logged in with administrative privileges

When the staff enters the new staff member's details (name, role, contact info)

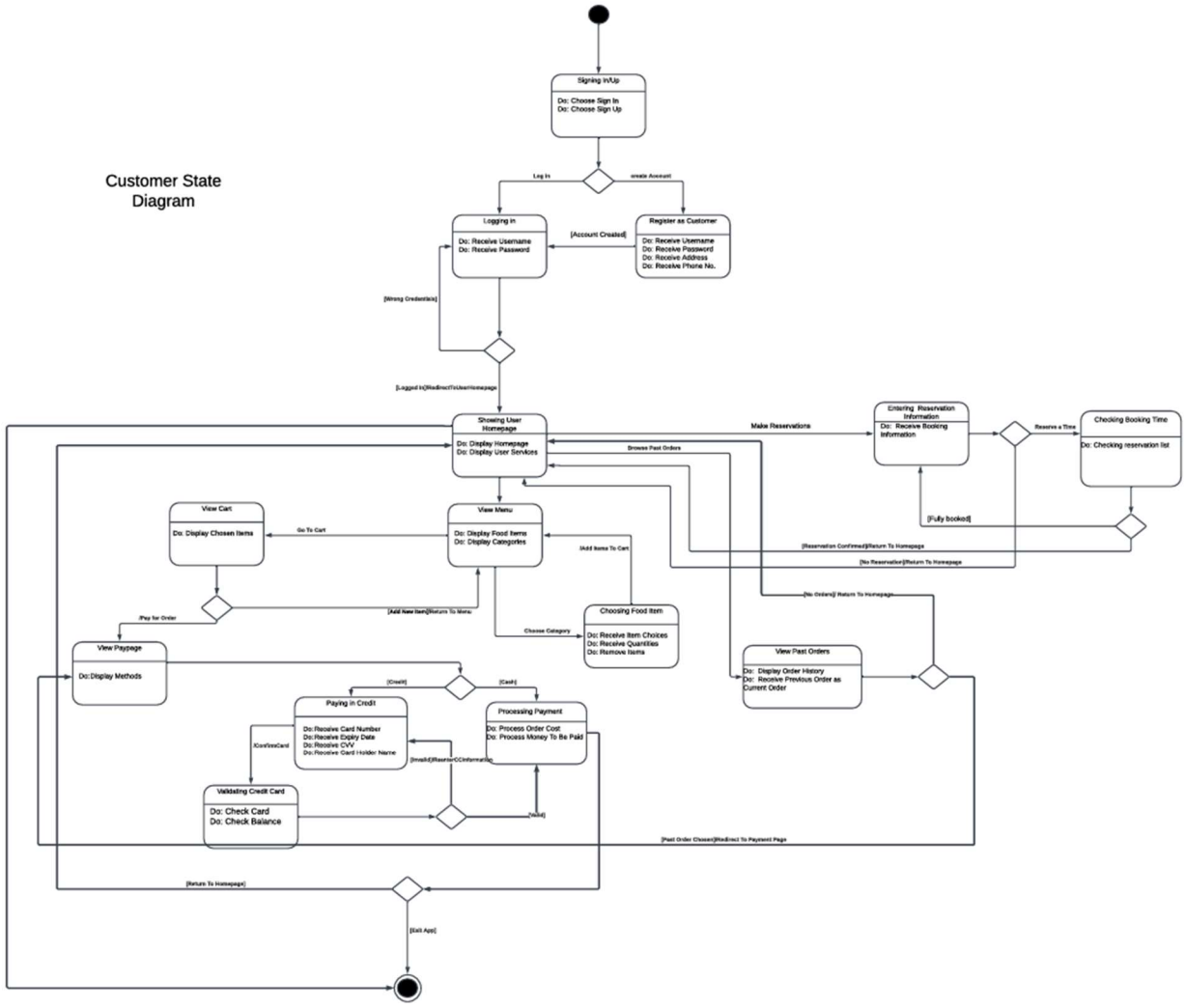
And clicks "Register New Staff"

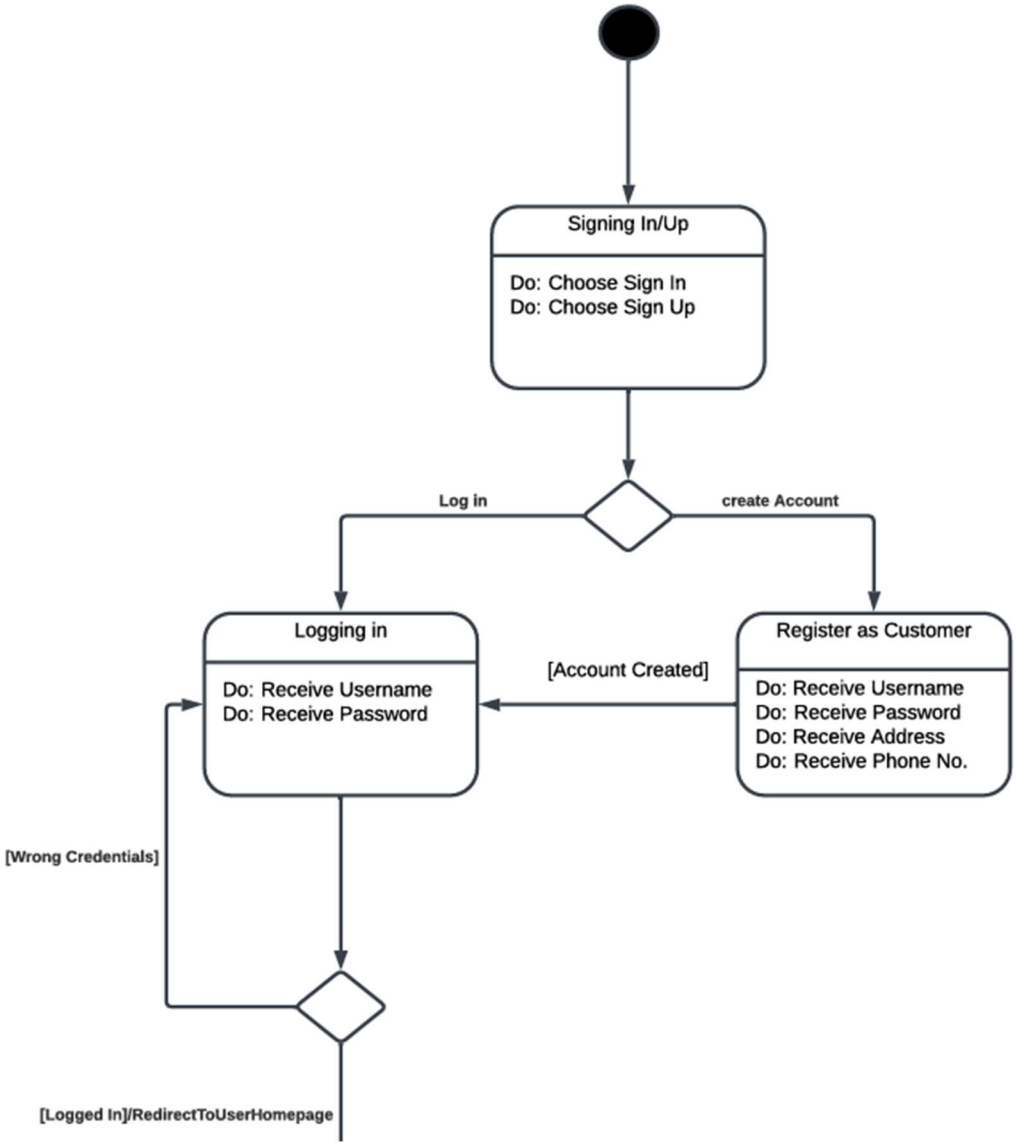
Then the new staff member should be created in the system

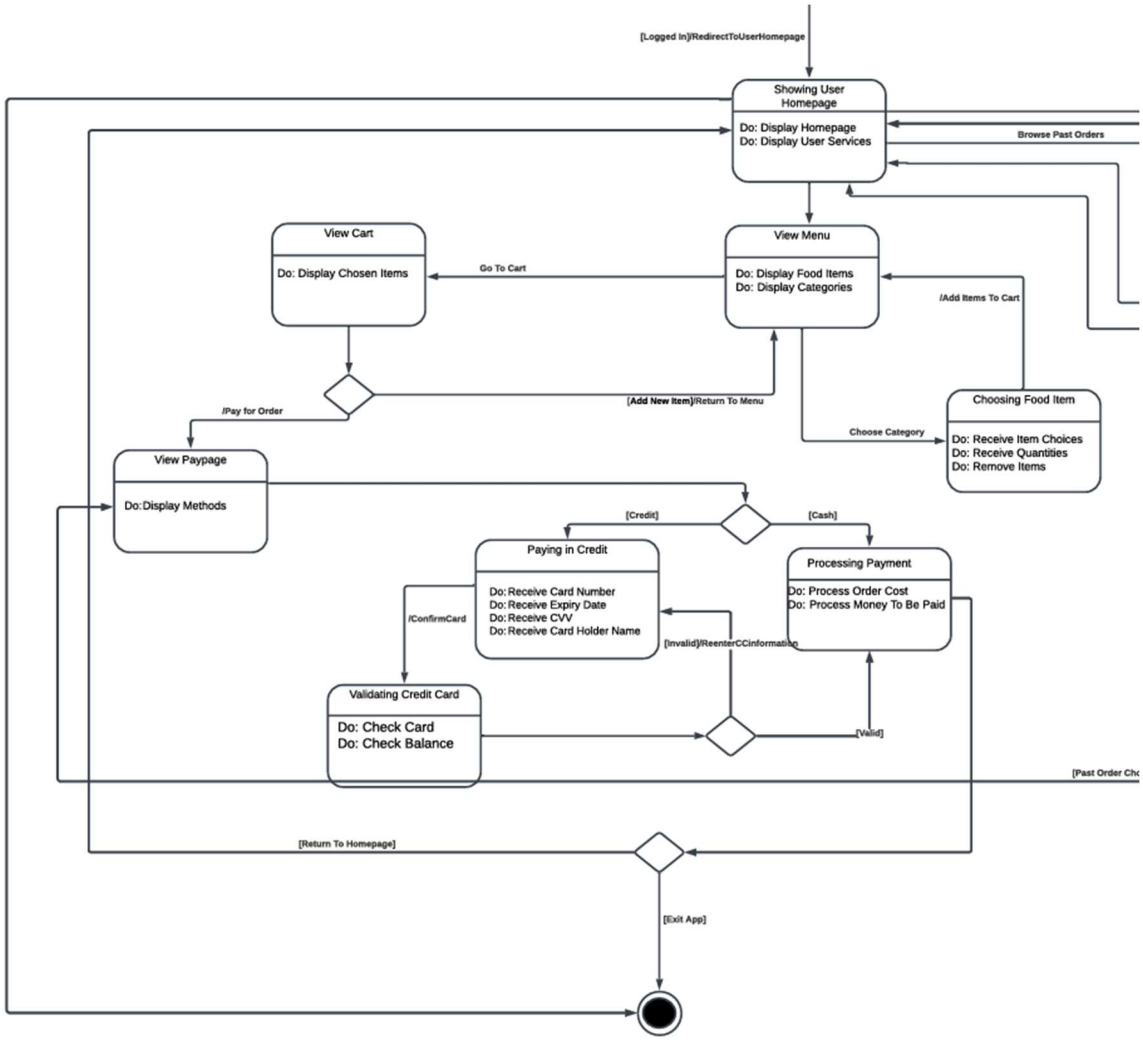
And the new staff member should receive an email with login credentials

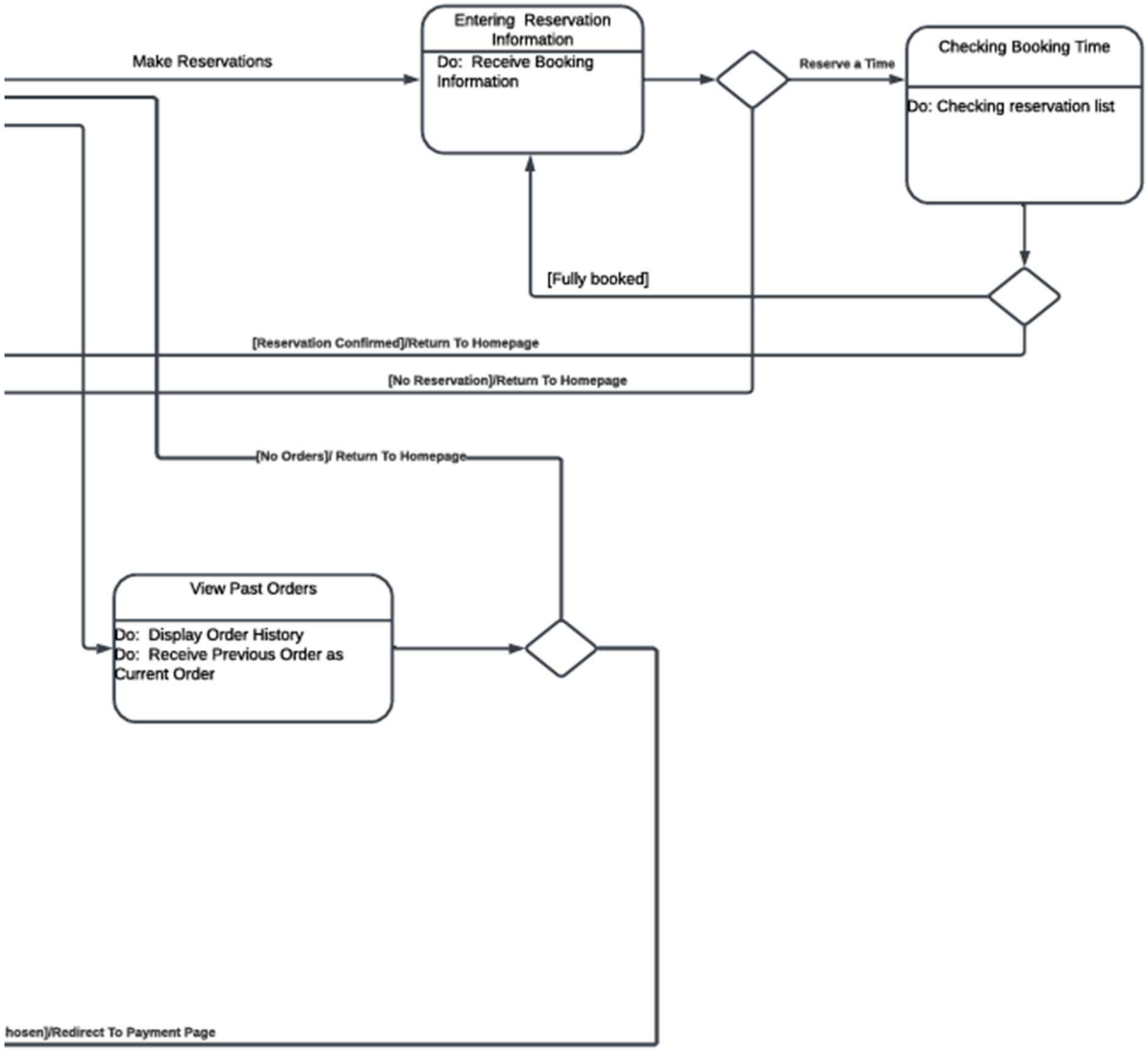
9. State Diagram: -

9.1 Customer State Diagram: -




Signing In/Up
Do: Choose Sign In
Do: Choose Sign Up
Log in
create Account
Logging in
Do: Receive Username
Do: Receive Password
[Account Created]
Register as Customer
Do: Receive Username
Do: Receive Password
Do: Receive Address
Do: Receive Phone No.
[Wrong Credentials]
[Logged In]/RedirectToUserHomepage


[Logged In]/RedirectToUserHomepage
Showing User Homepage
Do: Display Homepage
Do: Display User Services
Browse Past Orders
View Cart
Do: Display Chosen Items
Go To Cart
View Menu
Do: Display Food Items
Do: Display Categories
/Add Items To Cart
[Add New Item]/Return To Menu
/Pay for Order
Choosing Food Item
Choose Category
Do: Receive Item Choices
Do: Receive Quantities
Do: Remove Items
View Paypage
Do: Display Methods
[Credit]
[Cash]
Paying in Credit
Do: Receive Card Number
Do: Receive Expiry Date
Do: Receive CVV
Do: Receive Card Holder Name
Processing Payment
Do: Process Order Cost
Do: Process Money To Be Paid
/ConfirmCard
[Invalid]/ReenterCCinformation
Validating Credit Card
Do: Check Card
Do: Check Balance
[Valid]
[Past Order Ch
[Return To Homepage]
[Exit App]


Make Reservations
Entering Reservation Information
Do: Receive Booking Information
Reserve a Time
Checking Booking Time
Do: Checking reservation list
[Fully booked]
[Reservation Confirmed]/Return To Homepage
[No Reservation]/Return To Homepage
[No Orders]/ Return To Homepage
View Past Orders
Do: Display Order History
Do: Receive Previous Order as Current Order
hosen)/Redirect To Payment Page

Customer state diagram description: -

This state diagram represents the customer's journey interacting with our food ordering and reservation system, the documentation explaining the state transitions is as follows:

1. Signing Up/Logging In

- **Start Point:** As soon as the web app is initialized, the user is prompted to either sign in or sign up.
    - **Sign In:** Log in to an existing account.
    - **Sign Up:** Register a new account.
- **Sign In Path:**
    - **Log In:**
        - Actions:
            - Receive Username.
            - Receive Password.
        - Decision:
            - **Wrong Credentials:** Redirect to retry login.
            - **Successful Login:** Proceed to the **Main Dashboard**.
- **Sign Up Path:**
    - **Register as Customer:**
        - Actions:
            - Receive Username.
            - Receive Password.
            - Receive Address.
            - Receive Phone Number.
        - Completion: Redirect to the **Sign In/Sign Up page (the start point)**, where the user will be asked to sign in.

2. User Homepage

- After logging in, the user is redirected to the user homepage with the four options:
    - View Menu.
    - Make Reservations.
    - View Past Orders
    - Exit the application entirely

3. Viewing Menu

- **Browse Food Items:**
    - Display the available categorized food items.
    - User chooses categories to explore items.
- **Add to Cart:**
    - Items selected by the user are added to the cart.
    - Option to proceed to view the cart or continue browsing.
- **Viewing Cart**
    - **Display Chosen Items:** Show the items the user added to the cart.
    - Options:
        - **Pay for Order:** Proceed to payment.
        - **Return to Menu:** Redirect back to the menu for modifications.
- **Payment Process**
    - **View Payment Methods:**
        - Display payment options (e.g., credit card and cash).
    - **Pay Using Cash:**
        - Actions:
            - Head to Processing Payment state
    - **Pay Using Credit:**
        - Actions:
            - Receive Credit Card Number.

        - Receive Expiry Date.
        - Receive CVV.
        - Receive Cardholder Name.

- **Validate Credit Card:**
    - Verify:
        - Card authenticity.
        - Sufficient balance.
    - Decision:
        - **Invalid Card:** Return to payment methods.
        - **Valid Card:** Proceed to process payment.
- **Processing Payment:**
    - Actions:
        - Process the order.
        - Display readiness for payment.

7. Exiting Application

- After completing a reservation or payment, the user either exits the application or returns to the homepage

End Point: The state diagram concludes here in the case of exiting the application

4. Making Reservations

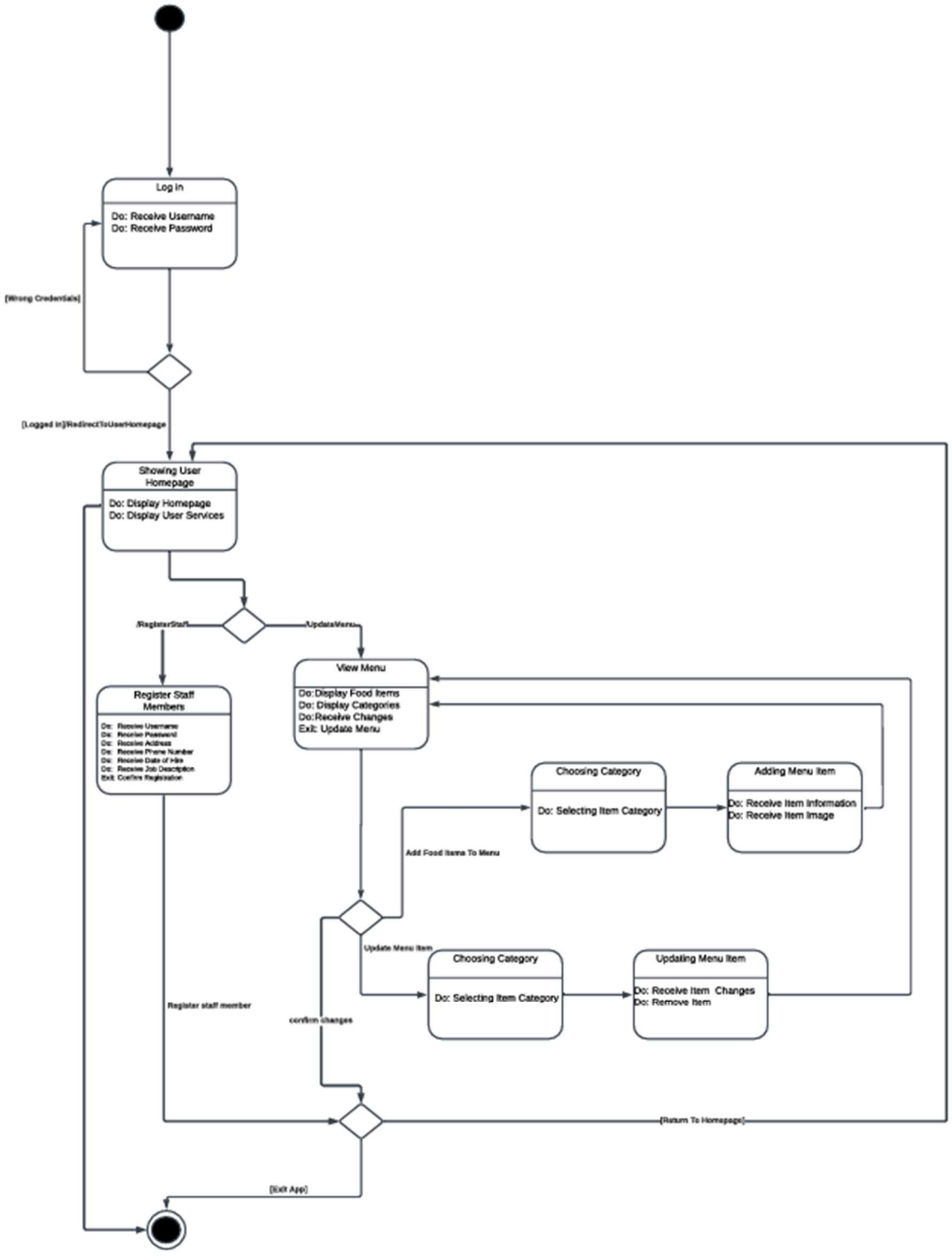
- **Choosing Reservation:**
    - User receives reservation options and instructions.
- **Check Booking Time:**
    - Review available slots.
    - Decision:
        - **Fully Booked:** Display a message and redirect to the choosing reservation page.

- **Reserve a Time:** Confirm and move to confirmation.
- **Exit:** Return to homepage, if the user doesn't want to make reservations.

5. View Past Orders:

- **Browsing Past Orders:**
    - User Chooses to view his past orders.
- **Checks if user ordered a past order:**
    - Decision:
        - Order a past order
        - Return to homepage
    - Payment:
        - Head to payment page if user ordered a past order.

9.2 Staff State Diagram: -



Staff State diagram description: -

1. This state diagram represents the customer's journey interacting with our food ordering and reservation system, the documentation explaining the state transitions is as follows:

   Start State:

   - Initial state where the system begins.

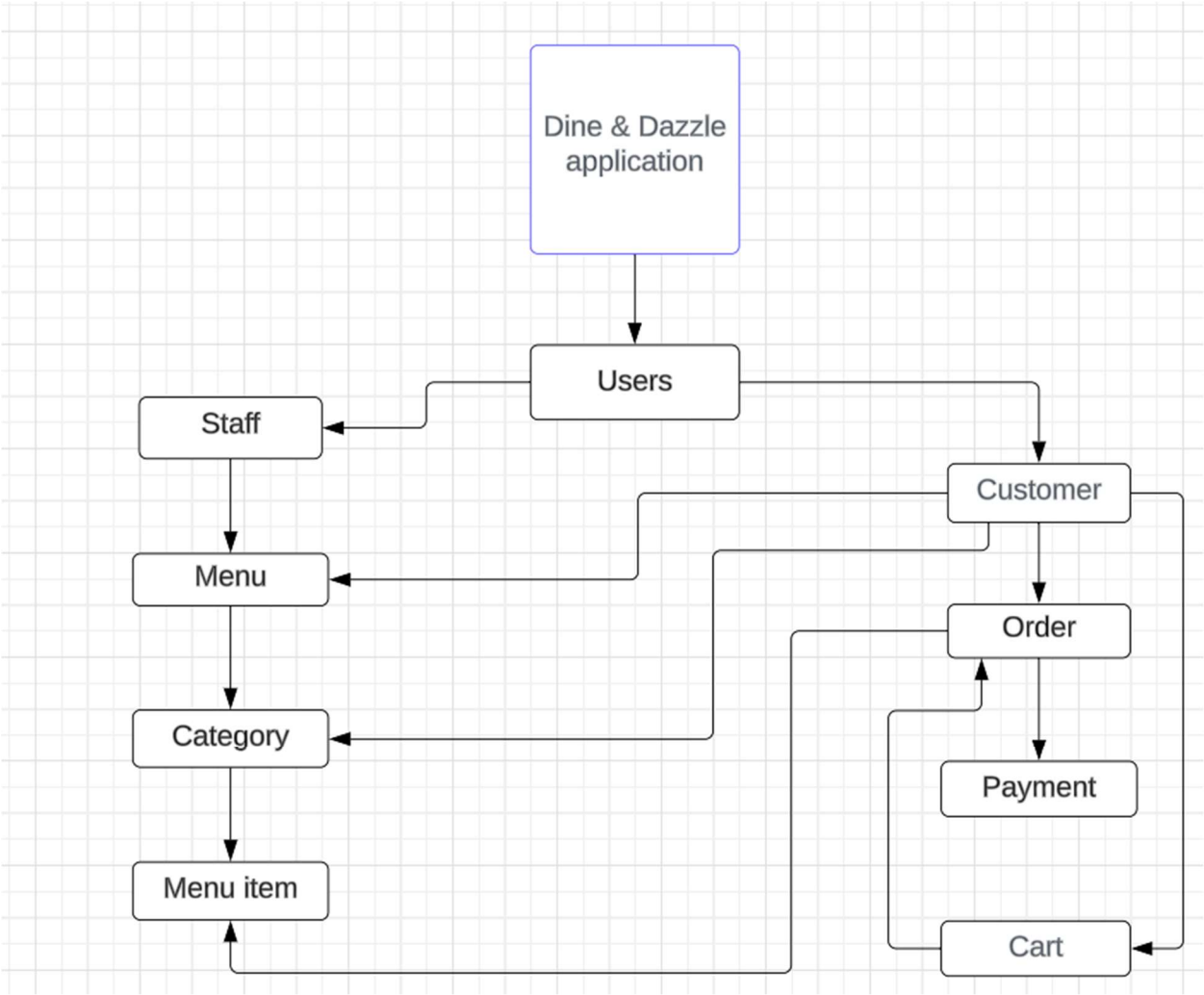
2. **Log In**:
   - **Inputs**: Receive username and password.
   - **Outcome**:
     - Valid credentials: Transition to "Showing User Homepage."
     - Invalid credentials: Stay in the "Log In" state.
3. **Showing User Homepage**:
   - **Actions**:
     - Display the homepage.
     - Show available user services.
   - **Transitions**:
     - "Register Staff Members" for managing staff.
     - "View Menu" for handling menu items.
     - End if the session concludes.
4. **Register Staff Members**:
   - **Inputs**:
     - Staff details such as name, role, phone number, email, and password.
   - **Outcome**:

        - Register a new staff member.
    - **Transitions**:
        - Back to "Showing User Homepage."
        - Exit the App
5. **View Menu**:
    - **Actions**:
        - Display menu categories and items.
        - Allow modifications to the menu.
    - **Transitions**:
        - "Choosing Category."
        - Back to "Showing User Homepage."
6. **Choosing Category**:
    - **Actions**:
        - Select a category for menu items.
    - **Transitions**:
        - "Adding Menu Item."
        - "Updating Menu Item."
7. **Adding Menu Item**:
    - **Inputs**:
        - Receive item information (name, description, price).
        - Upload item image.
    - **Outcome**:
        - Add the item to the menu.

- **Transitions**:
  - Back to "Choosing Category."

8. **Updating Menu Item**:
   - Inputs:
     - Receive item changes (e.g., price, description updates).
     - Remove item if necessary.
   - Outcome:
     - Update the menu.
   - Transitions:
     - Back to "Choosing Category."
9. **End State**:
   - Final state indicating the termination of the process.

10. Client-Object Relation Diagram: -



Client-Object Diagram Description: -

Objects:

1. **Dine & Dazzle Application:** The core system connecting users, menu, orders, and payments.
2. **User:** Generalized class representing all users of the system. Divided into:
    - **Customer:** Can browse the menu, place orders, and make payments.
    - **Staff:** Can manage menu items, categories, and oversee orders.
3. **Menu:** Displays categories and items for customers to view and order.
4. **Category:** Groups menu items logically (e.g., drinks, appetizers, desserts).
5. **MenuItem:** Represents individual items with attributes like name and price.
6. **Order:** Represents a customer's order, containing cart items.
7. **CartItem:** Tracks specific menu items added by a customer, including quantity.
8. **Payment:** Handles payment processes for a customer's order.

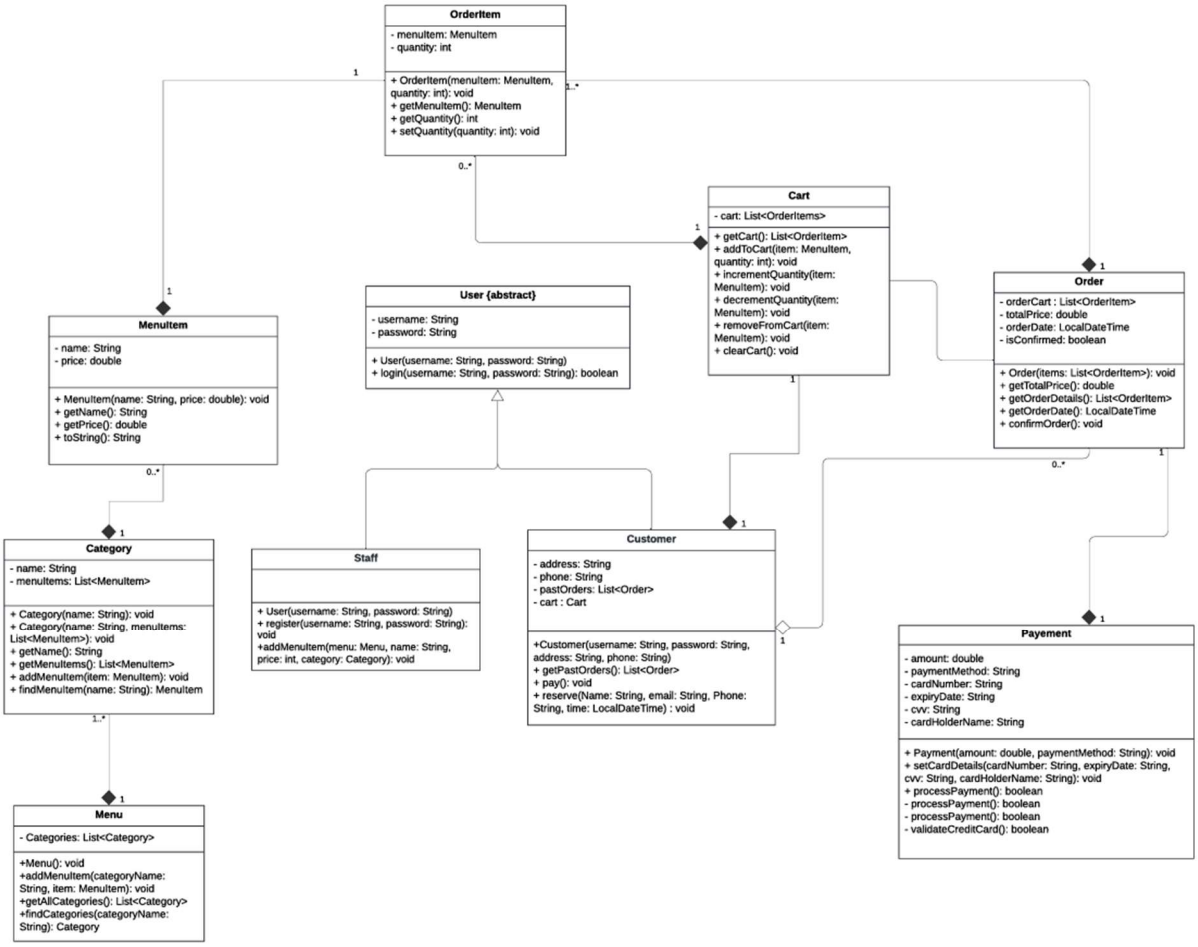
Relationships:

1. **Dine & Dazzle Application → User**
    - *1:M*: The system supports multiple users (both Customers and Staff).
2. **User → Customer & Staff**
    - *Bidirectional (Generalization)*: Both inherit shared attributes from the User class.
3. **Customer → Menu**
    - *1:M*: A customer can view the menu, which contains multiple categories.
4. **Menu → Category**
    - *1:M*: The menu organizes items into multiple categories.
5. **Category → MenuItem**
    - *1:M*: Each category contains multiple menu items.
6. **Customer → CartItem**
    - *1:M*: A customer can add multiple cart items, each representing a menu item and its quantity.
7. **CartItem → Order**
    - *1:1*: Each cart item is associated with one order.
8. **CartItem → MenuItem**
    - *1:1*: Each cart item corresponds to one specific menu item.
9. **Order → Customer**
    - *1:1*: An order belongs to one customer.
10. **Order → Payment**
    - *1:1*: A payment is tied to one specific order.

11. **Staff → Menu/Category/MenuItem**

- *1:M*: Staff can manage menu items, categories, and the menu.

11. Class Diagram: -



Class Diagram Description: -

Classes and Their Roles

1. **MenuItem**
   Represents individual food or drink items available on the menu, like "Cheeseburger" or "Fries." It provides basic details about each item, such as its name and price.

2. **Category**
   Groups similar menu items together into categories, such as "Burgers," "Sides," or "Drinks," making it easier for customers to browse the menu.

3. **Menu**
   Acts as the complete menu for the burger place, containing all the categories and their respective items. It's the central point for accessing the available options.

4. **OrderItem**
   Represents a specific menu item a customer wants to order, along with the quantity. It ensures that orders are tracked properly.

5. **Cart**
   Functions as a temporary holding place for the items a customer selects before finalizing their order. Customers can add, remove, or modify items in the cart.

6. **Order**
   Represents a finalized order placed by the customer. It tracks details like what items were ordered, the total cost, and whether the order has been confirmed.

7. **Payment**
   Handles payment processing for an order. It ensures payments are securely validated and processed using the chosen payment method.

8. **User (abstract)**
   A base class for all users of the app. It defines common functionality like logging in, which applies to both staff and customers.
9. **Staff (inherits from User)**
   Represents employees of the burger place. Staff manage the menu, such as adding or updating items, and can register other staff accounts.
10. **Customer (inherits from User)**
    Represents customers of the burger place. Customers can browse the menu, add items to their cart, place orders, make payments, and reserve tables.

Relationships and Their Types

1. **MenuItem ↔ Category**
   Type: Composition
   - A Category is made up of 0 or multiple MenuItem objects.
2. **Category ↔ Menu**
   Type: Composition
   - The Menu is composed of 0 or multiple Category objects.
3. **MenuItem ↔ OrderItem**
   Type: Composition
   - An OrderItem contains a MenuItem and quantity.
4. **OrderItem ↔ Cart**
   Type: Composition
   - A Cart is a collection of 0 or multiple OrderItem objects, which can not exist independently of the cart.
5. **OrderItem ↔ Order**
   Type: Aggregation
   - An order has a collection of 1 or multiple OrderItem objects, which can exist independently of the order.
6. **Cart ↔ Order**
   Type: Association
   - OrderItems in Cart are copied to the Order.
7. **Order ↔ Payment**
   Type: Composition
   - A Payment is linked to an Order to handle transactions.An order has one payement.
8. **User ↔ Staff and Customer**
   Type: Inheritance
   - Both Staff and Customer inherit from the abstract User class, reusing its login functionality and extending it with their own behavior.
9. **Customer ↔ Cart**
   Type: Composition
   - A Cart belongs to a specific Customer and cannot exist without the customer. A customer only has one cart.
10. **Customer ↔ Order**
    Type: Aggregation
    - A Customer can have 0 or multiple orders associated with them, but the orders do not depend on the customer for their existence.

12. Comparative Analysis of The Output of The Adopted Methodologies: -

1. CRC Cards (Class-Responsibility-Collaborator): -

Best Use Cases:

- Systems with complex object-oriented structures and clear class interactions.
- Early stages of design when brainstorming object interactions and responsibilities.
- Collaborative teams that prefer lightweight, informal tools for design discussions.
- Educational contexts to teach object-oriented thinking.

Strengths:

- Encourages teamwork and collaborative brainstorming.
- Simple and low-cost: no specialized tools required.
- Focuses on identifying responsibilities and relationships between classes.
- Easy to adapt and iterate during initial design phases.

Weaknesses:

- Limited scalability for large systems with many classes.
- Lack of precision in defining detailed system behavior.
- Informal nature can lead to ambiguity.
- Not well-suited for continuous integration or automation.

2. RDD (Responsibility-Driven Design): -

Best Use Cases:

- Object-oriented systems where defining clear responsibilities for each class is critical.
- Systems that require modular and maintainable architecture.
- Teams following traditional object-oriented practices.
- Medium-sized projects where class responsibilities and interactions are the primary focus.

Strengths:

- Helps ensure good object-oriented design principles (encapsulation, modularity).
- Focuses on high cohesion within classes and loose coupling between them.
- Guides the distribution of responsibilities across the system.
- Produces designs that are easier to maintain and extend.

Weaknesses:

- May result in overemphasis on class structure, neglecting overall system behavior.
- Requires skilled designers with experience in object-oriented principles.
- Less emphasis on the integration of the system with real-world user stories or scenarios.

3. BDD (Behavior-Driven Development): -

Best Use Cases:

- Systems with complex user interactions or business logic.
- Agile projects emphasizing iterative development and continuous delivery.
- Cross-functional teams where collaboration between developers, testers, and non-technical stakeholders is vital.
- Systems requiring clear, testable requirements and behavior.

Strengths:

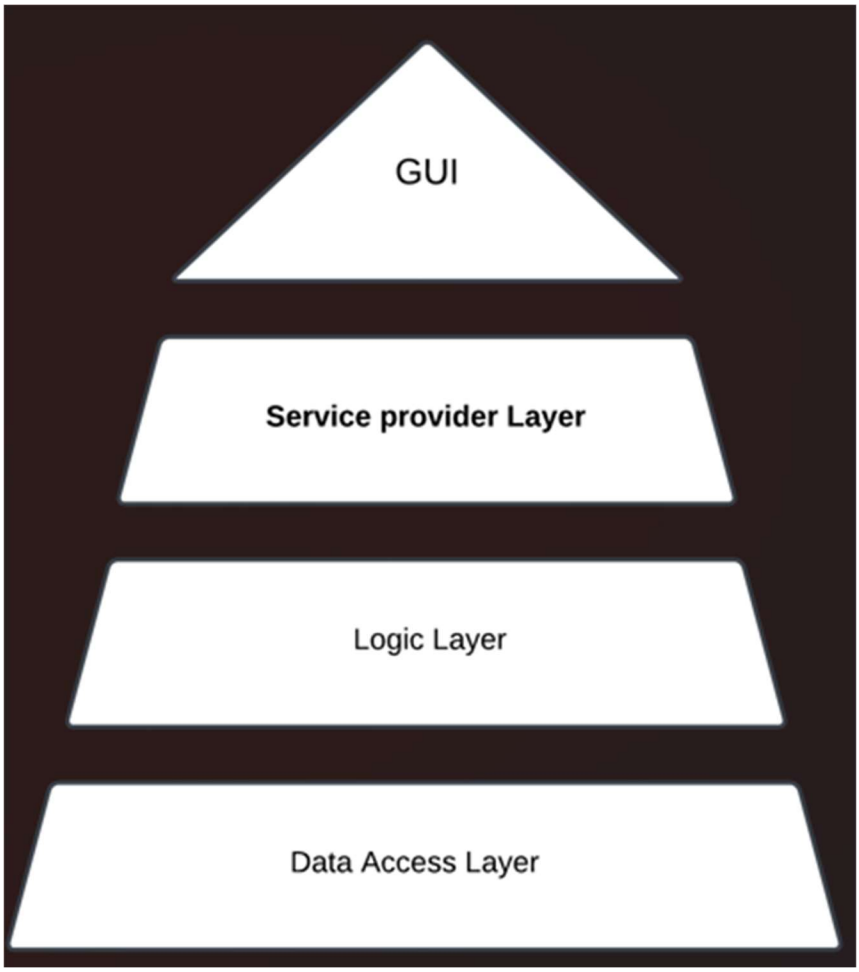
- Promotes clear communication between technical and non-technical stakeholders.
- Bridges the gap between business requirements and technical implementation.
- Produces living documentation (test cases double as specs).
- Encourages writing testable, user-focused code from the outset.
- Supports automated testing and continuous integration workflows.

Weaknesses:

- Steeper learning curve for teams unfamiliar with BDD tools or practices.
- Overhead in maintaining test cases, especially in systems with rapidly changing requirements.
- May lead to over-focus on test scenarios, neglecting broader architectural concerns.
- Dependence on tools and frameworks (e.g., Cucumber, Spec Flow) for full benefit.

13. Architectural Model: -

13.1 Layered Model: -



Layered Model Description: -

GUI:

Purpose: Manages the application's user interface and user interactions.

Applied with:

Using CSS for JavaFX

Separate views for:

Menu display (for customers).

Cart management.

Reservation system.

User/staff login and management.

logical layer:

Purpose: Encapsulates core business rules and domain-specific logic.

Components:

Java classes representing:

Menu items.

Cart.

Reservations.

Users (with roles: Customer/Staff).

Business logic methods:

Validate reservation times.

Check staff permissions.

Service provider:

Purpose: Controls business logic and application flow. Acts as a mediator between the GUI and the data providing gui with all needed data.

Components:

Service classes for:

Menu handling (fetch items, filter categories).

Cart operations (add, remove, calculate totals).

Reservation management.

Payment processing.

User and staff operations (login, roles).

Includes validation and additional logic.

Data Access Layer

Purpose: Manages interaction with the database or any persistent storage.

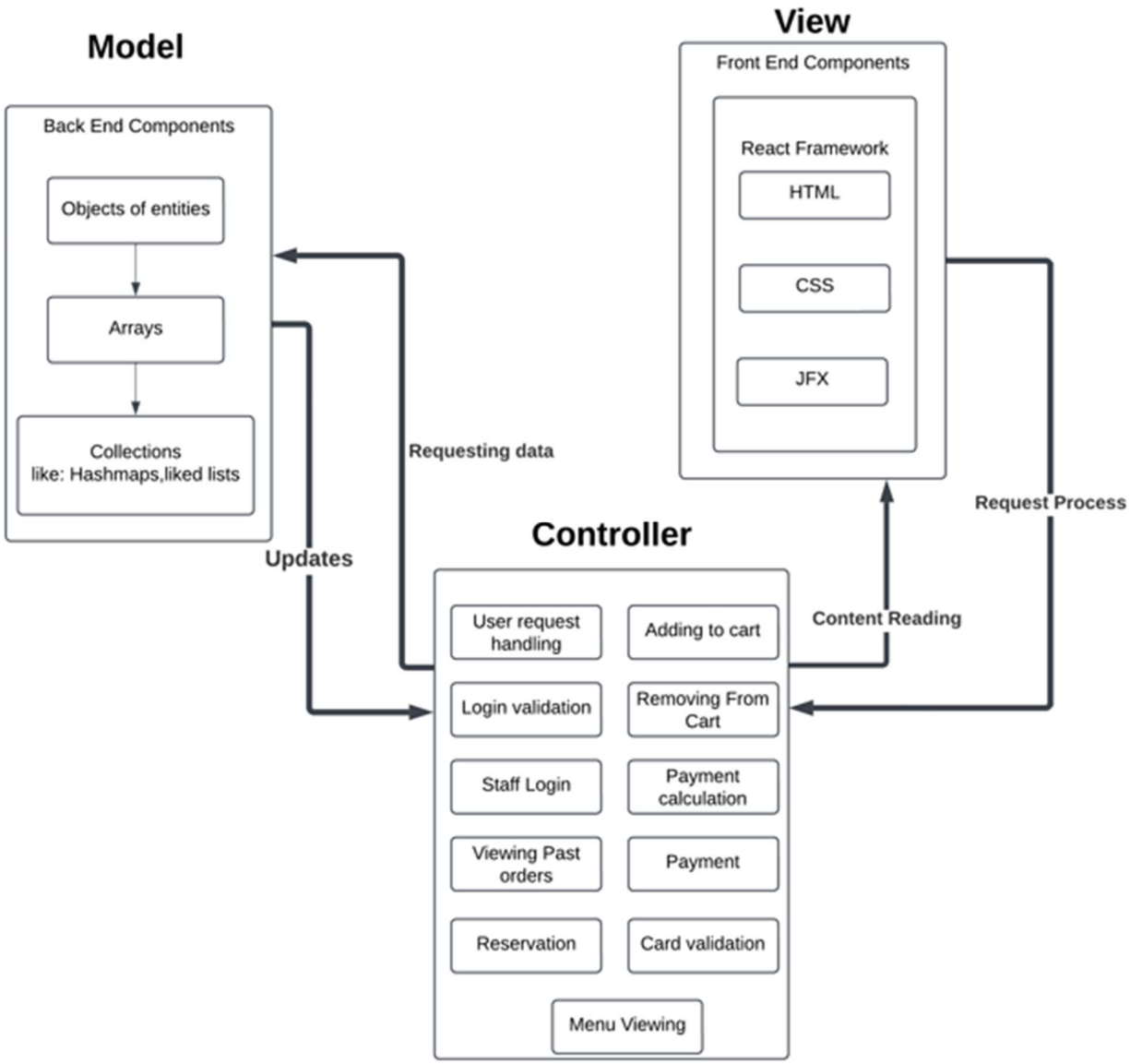
Components:

DAO (Data Access Object) classes for:

operations for menu items, reservations, users, and cart details.

Database connection and query execution.

13.2 Model View Controller: -



Model View Controller Description: -

The Model-View-Controller (MVC) architectural pattern is used to design and implement the Java-based restaurant menu application. This pattern allows the application to be modular, scalable, and easy to maintain by separating its logic into three distinct components: Model, View, and Controller. Each component handles a specific aspect of the application, ensuring a clean separation of concerns.

Overview of the Components

1. **Model:**

The Model is responsible for managing the application's data, logic, and rules. The restaurant menu application, it includes the backend components and data structures necessary to represent and manipulate the application's core entities. Key elements of the Model include:

- Objects of Entities: Represent data objects such as menu items, cart contents, reservations, and user accounts.
- Data Structures: Arrays and collections like Hash Maps and linked lists are used to store and retrieve data efficiently.
- Backend Logic: Encapsulates business rules such as calculating cart totals, validating reservations, and managing user roles.
- The Model interacts with the Controller, updating it whenever data changes, ensuring that the Controller can reflect these changes in the View.

2. **View**:

The View represents the application's user interface, displaying data to the user and capturing their interactions. In this application, the View is implemented using a combination of technologies:

JavaFX: Provides a desktop UI for users.

HTML, CSS, and React Framework: Enhances the web-based interface, ensuring a responsive and visually appealing design.

The View is responsible for requesting data from the Controller, rendering it for the user, and forwarding user inputs back to the Controller for processing.

3. **Controller:**

The Controller acts as the intermediary between the Model and the View. It processes user requests, updates the Model, and communicates the updated data to the View. Key responsibilities of the Controller include:

- Handling user inputs such as adding items to the cart, making reservations, and processing payments.
- Validating user login credentials for both customers and staff.
- Managing core functionalities like menu viewing, order history retrieval, and card validation.
- Ensuring data consistency by coordinating between the Model and View layers.
- Interaction Workflow
- User Interaction: A customer interacts with the View.
- Controller Processing: The Controller processes the input, validates it, and updates the Model accordingly.

- Model Update: The Model performs the necessary logic and notifies the Controller of the changes.

- View Update: The Controller retrieves the updated data from the Model and updates the View to reflect the changes, ensuring a seamless user experience.

Benefits of Using MVC:

The adoption of the MVC architecture for the restaurant menu application provides several advantages:

- Separation of Concerns:

The application logic is divided into three distinct layers, making it easier to develop, debug, and maintain.

Changes in one layer do not directly affect the others.

- Scalability:

The modular design allows for scaling individual components. For example, the backend Model can accommodate more entities without altering the View or Controller significantly.

- Reusability:

Models and Controllers can be reused across different Views.

- Flexibility:

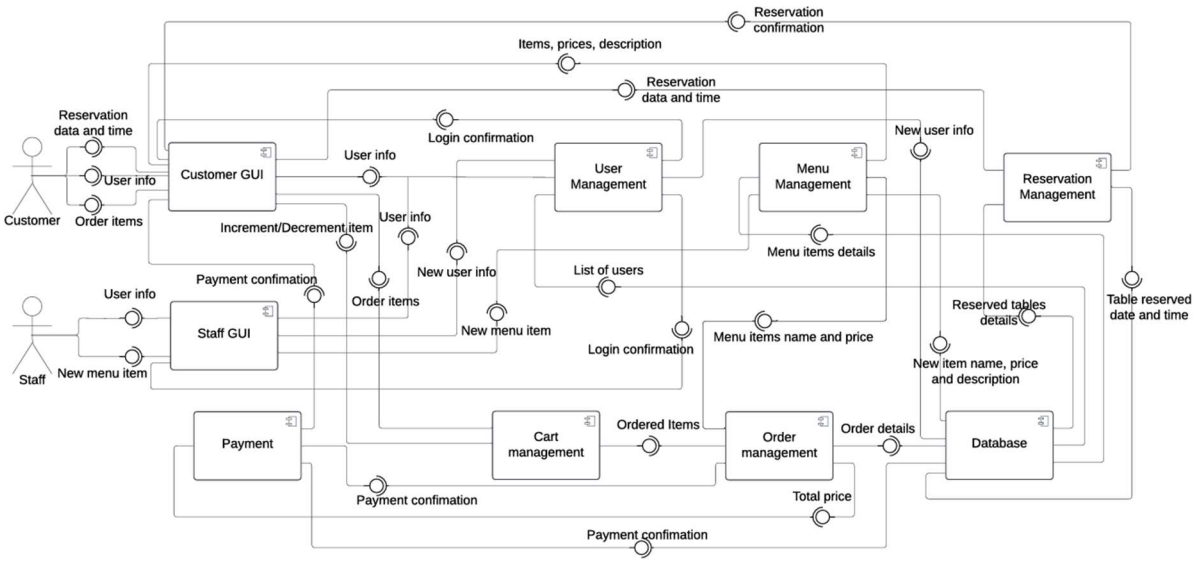
The Controller serves as a centralized point for managing complex business logic, making it easier to adapt to new requirements.

- Improved User Experience:

Dynamic updates to the View ensure a responsive and interactive application.

Finally, The MVC architecture effectively supports the Java restaurant menu application by dividing its functionality into modular components. This design pattern ensures a clean separation of concerns, enhances scalability, and provides a solid foundation for future growth and feature enhancements. Its implementation in the restaurant menu application has proven to be an efficient and robust approach for managing complex user interactions and business logic.

14. Component Diagram: -



Component Diagram Description: -

1. Key Components and Their Responsibilities

1.1 Customer GUI

- **Purpose**: Provides an interface for customers to interact with the system.
- **Responsibilities**:
  - Allow customers to view menu items.
  - Enable customers to place orders and manage their cart.
  - Facilitate table reservation by accepting reservation details such as date and time.
  - Forward login information for authentication.

1.2 Staff GUI

- **Purpose**: Provides an interface for staff to manage the system.
- **Responsibilities**:
  - Enable staff to add new menu items.
  - Register new staff accounts.
  - View order details for monitoring and management.

1.3 User Management

- **Purpose**: Handles user-related operations such as authentication and user registration.
- **Responsibilities**:
  - Verify login credentials and provide confirmation to the GUIs.
  - Maintain a list of users (customers and staff).
  - Register new users upon staff input.

1.4 Menu Management

- **Purpose**: Manages the menu items available in the restaurant.
- **Responsibilities**:
    - Store menu item details such as name, price, and description.
    - Allow staff to add or update menu items.
    - Provide menu details to both the Customer GUI and Staff GUI.

1.5 Reservation Management

- **Purpose**: Handles table reservations in the restaurant.
- **Responsibilities**:
    - Accept reservation details such as date and time from the Customer GUI.
    - Confirm reservations and store reserved table details.
    - Provide reservation details when needed.

1.6 Cart Management

- **Purpose**: Manages the customer's cart.
- **Responsibilities**:
    - Store items added to the cart along with their quantities.
    - Provide an interface to increment or decrement item quantities.
    - Send ordered items to the Order Management component for finalization.

1.7 Order Management

- **Purpose**: Handles the creation and confirmation of orders.
- **Responsibilities**:
  - Finalize orders based on items in the cart.
  - Calculate the total price of the order.
  - Send order details to the database for storage.

1.8 Payment

- **Purpose**: Handles payment operations.
- **Responsibilities**:
  - Process payments using cash or credit.
  - Confirm successful payments and communicate with Order Management for order confirmation.

1.9 Database

- **Purpose**: Serves as the central repository for system data.
- **Responsibilities**:
  - Store user information, menu items, orders, reservations, and payment confirmations.
  - Provide data retrieval for components like Menu Management, User Management, and Reservation Management.

2. Connections Between Components

- **Customer GUI ↔ User Management**:
  - The Customer GUI sends user login or signup information to User Management.
  - The User Management component confirms successful login or registration.

- **Customer GUI ↔ Menu Management**:
    - The Customer GUI retrieves menu item details (name, price, description) from Menu Management.
- **Customer GUI ↔ Cart Management**:
    - The Customer GUI sends items to the cart and retrieves cart details for display.
- **Customer GUI ↔ Reservation Management**:
    - The Customer GUI sends reservation details (date, time) for table booking.
    - The Reservation Management component confirms the reservation and provides reservation details.
- **Cart Management ↔ Order Management**:
    - The Cart Management component sends ordered items to the Order Management component for order creation.
- **Order Management ↔ Payment**:
    - The Order Management sends the total order price to the Payment component.
    - The Payment component confirms the payment and notifies Order Management.
- **Staff GUI ↔ User Management**:
    - The Staff GUI sends new user information to User Management for staff registration.
- **Staff GUI ↔ Menu Management**:
    - The Staff GUI sends new menu item details to Menu Management.

- **Database ↔ All Management Components**:
    - The Database interacts with all management components (User Management, Menu Management, Reservation Management, and Order Management) to store and retrieve relevant data.

15. Time Plan: -

Phase 1: Project Initialization (Oct 12 - Oct 15)

- **Objective**: Set up the development environment and prepare the project structure.
- **Tasks**:
    1. Install necessary tools (Java IDE, Scene Builder, etc.).
    2. Define the base project structure.
    3. Write a detailed requirements document.

Phase 2: Backend Design and Implementation (Oct 16 - Nov 10)

- **Objective**: Design and implement the core backend logic.
- **Tasks**:

Oct 16 - Oct 22

1. **Class Design**:
    - Finalize class structures (e.g., User, Order, OrderItem, Menu, Category, MenuItem, Payment).
    - Create UML diagrams for class relationships.
2. **Implementation (Part 1)**:
    - Implement User, OrderItem and Order classes with necessary methods.
    - Add functionality for managing past orders and cart-related operations.

Oct 23 - Oct 29

3. **Implementation (Part 2)**:
    - Implement MenuItem, Category and Menu classes.
    - Add menu-related operations (e.g., add/remove items or categories).
    - Write unit tests for individual classes.

Oct 30 - Nov 5

4. **Implementation (Part 3)**:
    - Implement Payment class.
    - Add logic for cash and credit card payments.
    - Integrate order confirmation.

Phase 3: Frontend Design and Implementation (Nov 14 - Dec 5)

- **Objective**: Build the user interface and connect it to the backend.
- **Tasks**:

Nov 14 - Nov 20

1. **UI Design**:
    - Create wireframes for the homepage, menu page, cart page, checkout, and past orders.
    - Finalize design elements and themes.
2. **Scene Controllers**:
    - Implement HomepageSceneController and integrate login/signup functionality.
    - Develop MenuSceneController and display categories and items.

Nov 21 - Nov 30

3. **Integrate Backend with UI**:
    - Link CartSceneController to cart operations.
    - Implement CheckoutSceneController and payment options.
    - Add functionality to display past orders in PastOrdersSceneController.

Dec 1 - Dec 5

4. **Final Touches**:
    - Add validations (e.g., input fields, payment info).
    - Ensure smooth navigation between scenes.

Phase 4: Testing and Deployment (Dec 6 - Dec 20)

- **Objective**: Test the application thoroughly and prepare for submission or deployment.
- **Tasks**:

Dec 6 - Dec 10

1. **Unit Testing**:
    - Test all backend methods and ensure data consistency.
    - Check for edge cases (e.g., empty cart, invalid login).
2. **UI Testing**:
    - Verify scene transitions and UI responsiveness.
    - Test for usability and user experience.

Dec 11 - Dec 15

3. **Integration Testing**:
    - Ensure smooth interaction between frontend and backend.
    - Validate full user workflows (e.g., login → menu → cart → checkout → past orders).

Dec 16 - Dec 20

5. **Bug Fixing and Final Deployment**:
    - Fix any identified bugs.
    - Prepare documentation (user manual, technical guide).

Milestones

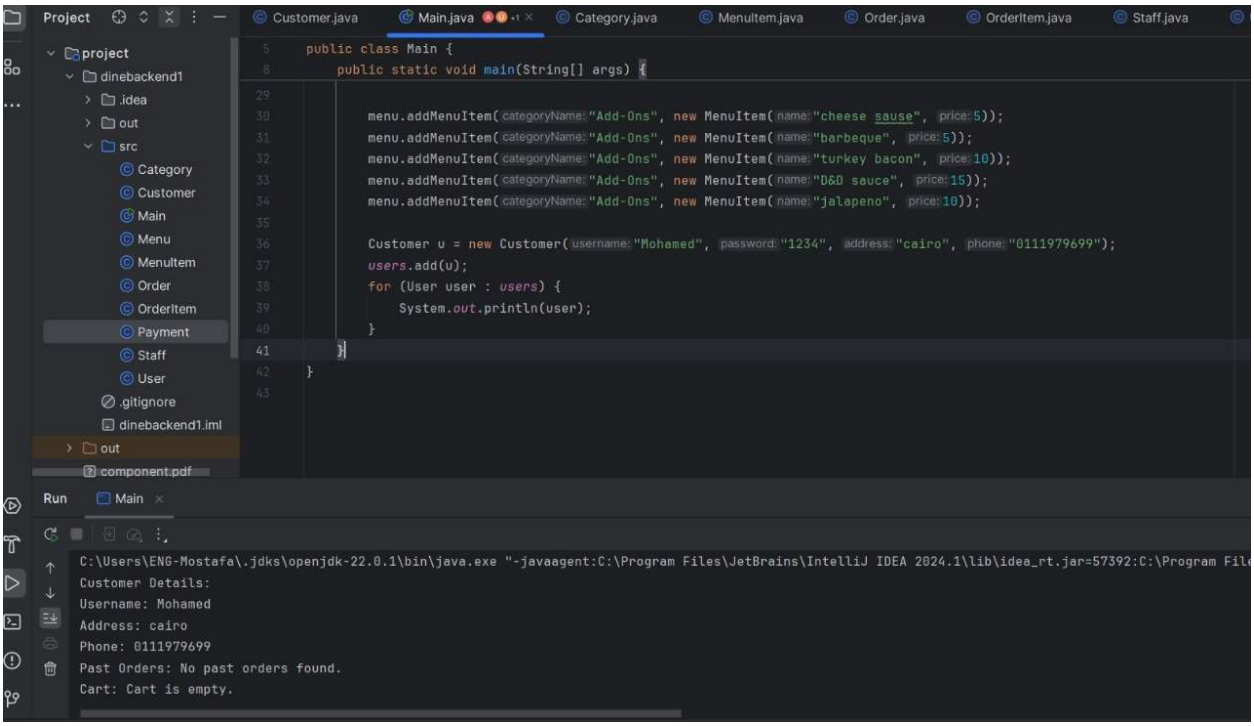
1. **Oct 15**: Project structure and requirements finalized.
2. **Nov 8**: Core backend implemented and tested.
3. **Dec 5**: Full application functionality completed.
4. **Dec 20**: Application fully tested, documented, and ready for submission or deployment.

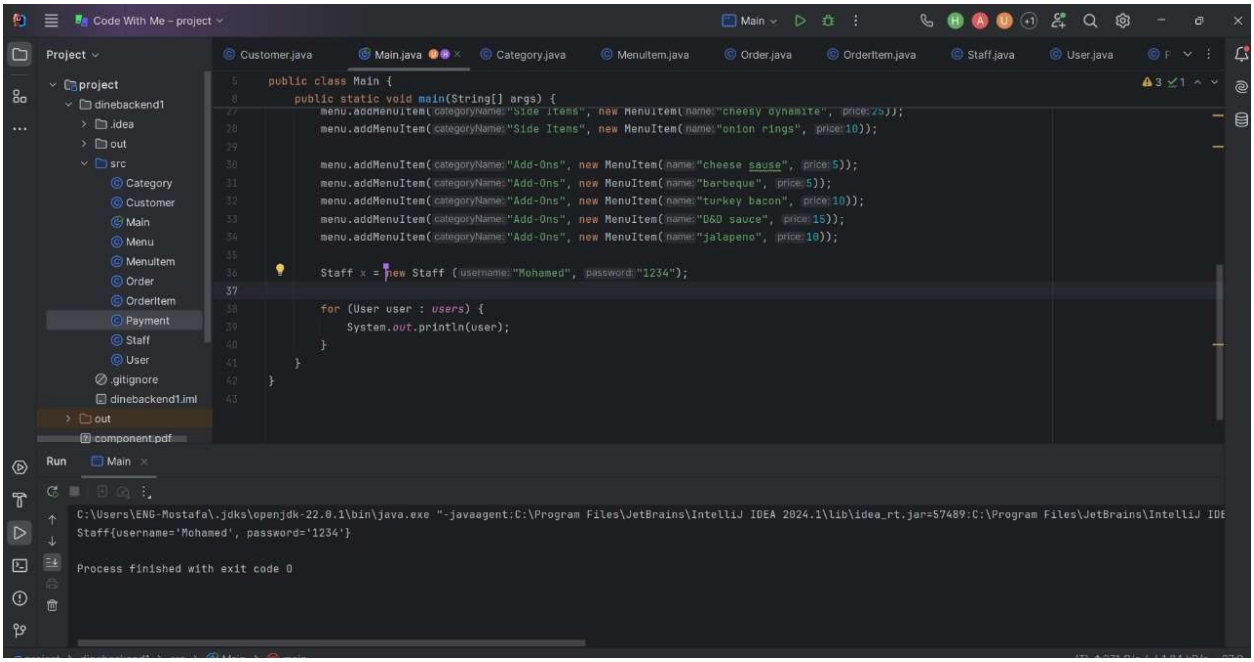
16. Prototyping: -

16.1 Unit Testing

1) Signup Functionality

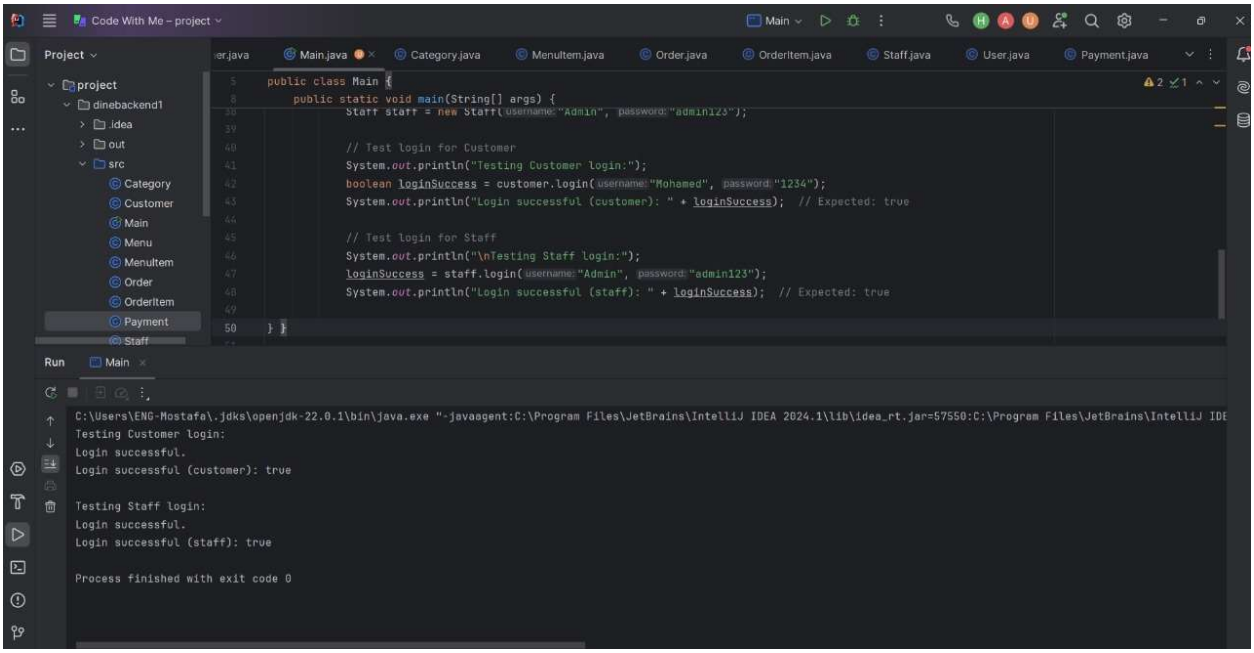
The signup function was tested to verify that new users can successfully register with a username and password

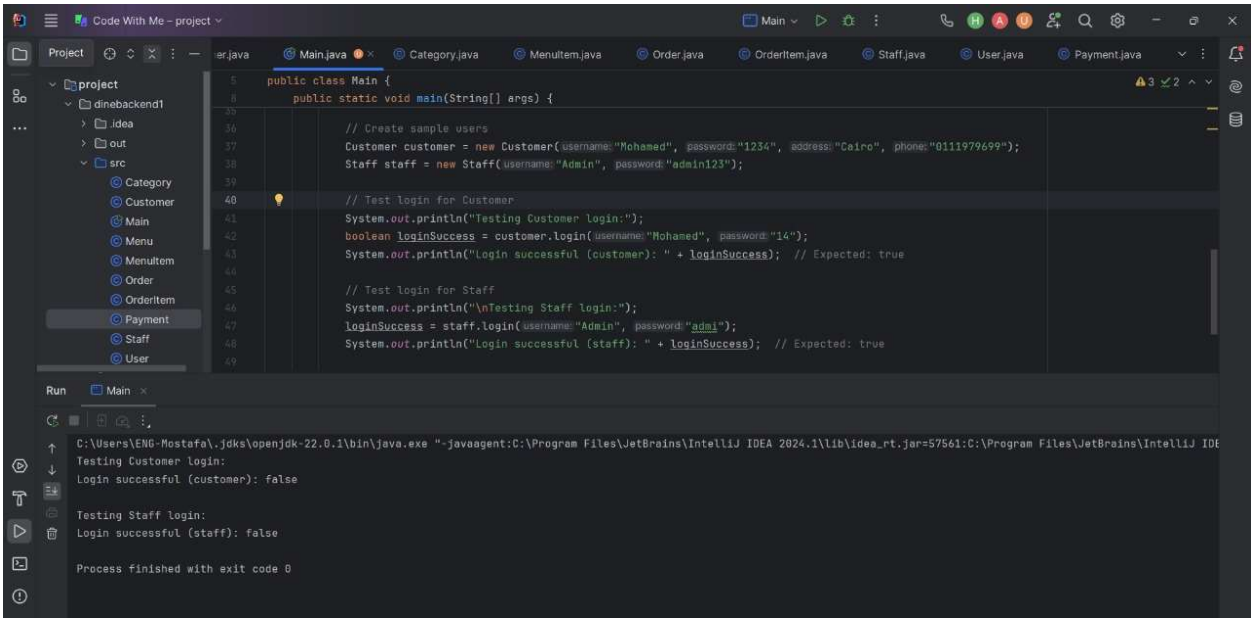




2) Login Functionality

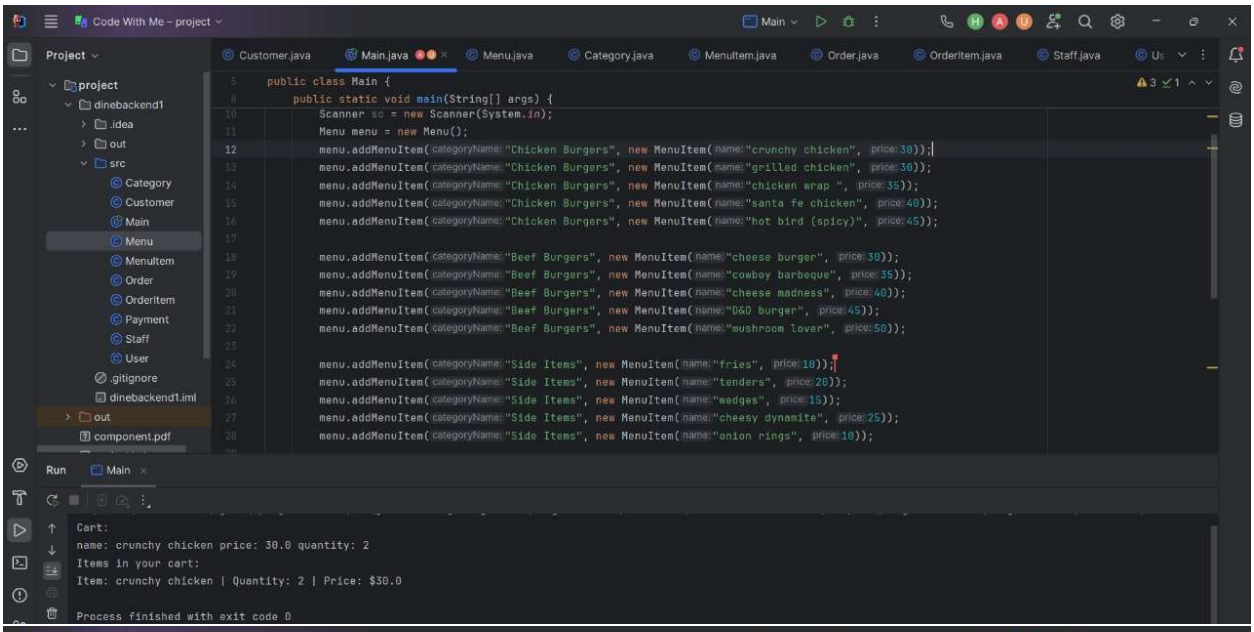
The login function was tested to ensure that users can authenticate successfully with the correct credentials. The test cases checked for both valid and invalid usernames and passwords.

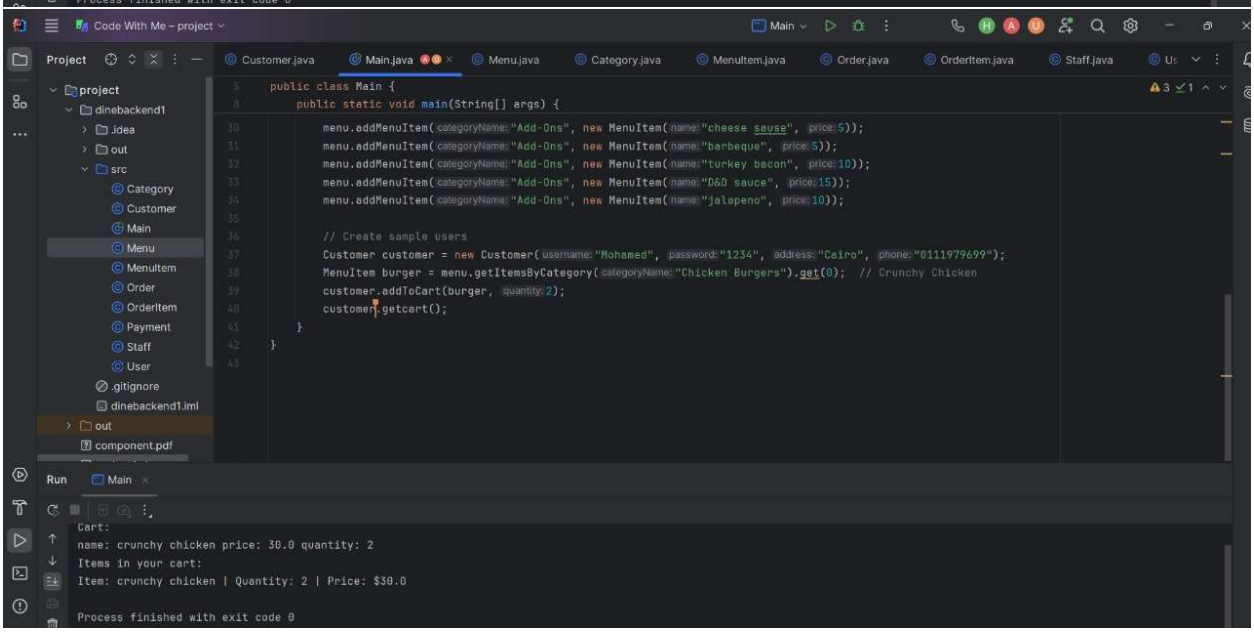


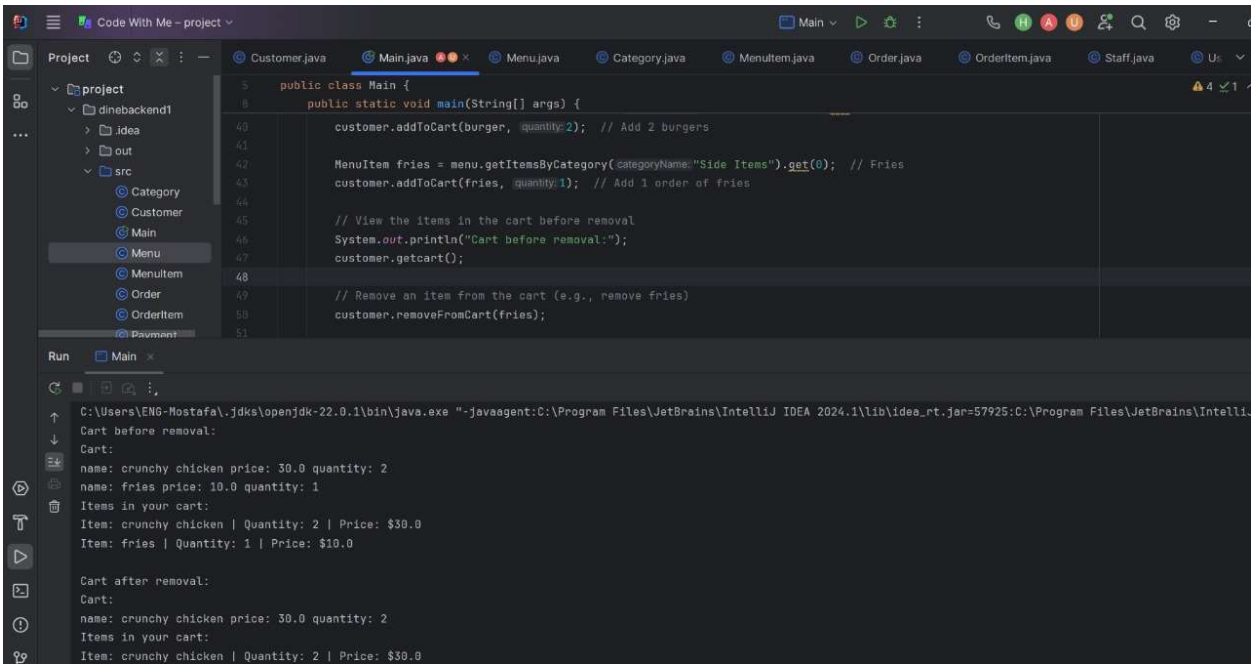


3) Add to /Remove from Cart:

The add to / remove from cart functions were tested to ensure users can successfully add and remove items from their shopping cart. Test cases included verifying that the correct item is added to the cart with the correct quantity and price, and that items can be removed without leaving inconsistencies in the cart.

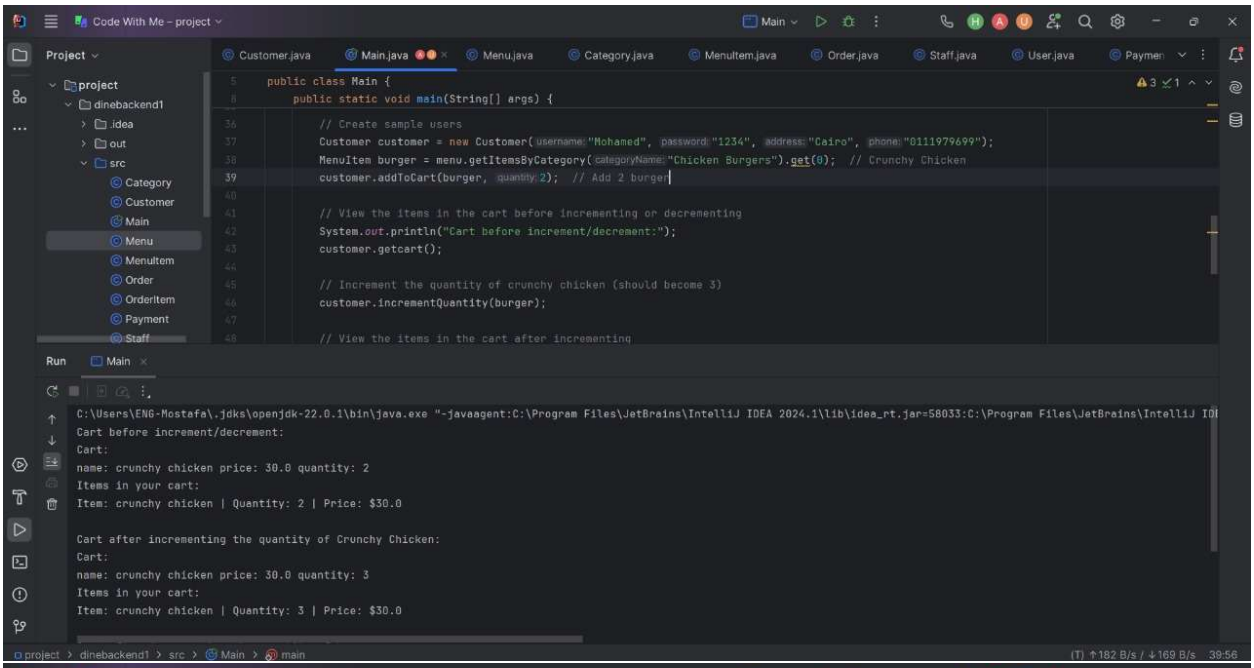

Code With Me – project
Main
Project
Customer.java
Main.java
Menu.java
Category.java
MenuItem.java
Order.java
OrderItem.java
Staff.java
project
dinebackend1
.idea
out
src
Category
Customer
Main
Menu
MenuItem
Order
OrderItem
Payment
Staff
User
.gitignore
dinebackend1.iml
out
component.pdf
public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
Menu menu = new Menu();
menu.addMenuItem("Chicken Burgers", new MenuItem("crunchy chicken", 30));
menu.addMenuItem("Chicken Burgers", new MenuItem("grilled chicken", 30));
menu.addMenuItem("Chicken Burgers", new MenuItem("chicken wrap ", 35));
menu.addMenuItem("Chicken Burgers", new MenuItem("santa fe chicken", 40));
menu.addMenuItem("Chicken Burgers", new MenuItem("hot bird (spicy)", 45));
menu.addMenuItem("Beef Burgers", new MenuItem("cheese burger", 30));
menu.addMenuItem("Beef Burgers", new MenuItem("cowboy barbeque", 35));
menu.addMenuItem("Beef Burgers", new MenuItem("cheese madness", 40));
menu.addMenuItem("Beef Burgers", new MenuItem("D&D burger", 45));
menu.addMenuItem("Beef Burgers", new MenuItem("mushroom lover", 50));
menu.addMenuItem("Side Items", new MenuItem("fries", 10));
menu.addMenuItem("Side Items", new MenuItem("tenders", 20));
menu.addMenuItem("Side Items", new MenuItem("wedges", 15));
menu.addMenuItem("Side Items", new MenuItem("cheesy dynamite", 25));
menu.addMenuItem("Side Items", new MenuItem("onion rings", 10));
Run
Main
Cart:
name: crunchy chicken price: 30.0 quantity: 2
Items in your cart:
Item: crunchy chicken | Quantity: 2 | Price: $30.0
Process finished with exit code 0

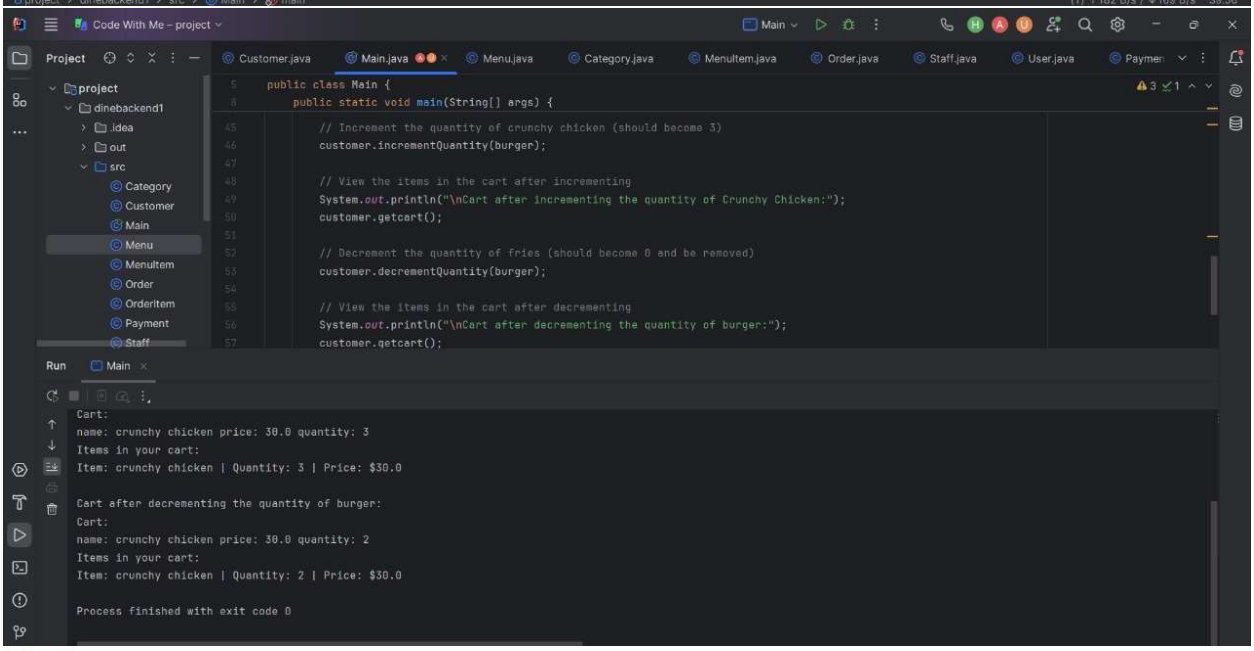

Code With Me – project
Main
Project
Customer.java
Main.java
Menu.java
Category.java
MenuItem.java
Order.java
OrderItem.java
Staff.java
project
dinebackend1
.idea
out
src
Category
Customer
Main
Menu
MenuItem
Order
OrderItem
Payment
Staff
User
.gitignore
dinebackend1.iml
out
component.pdf
public class Main {
public static void main(String[] args) {
menu.addMenuItem("Add-Ons", new MenuItem("cheese sauce", 5));
menu.addMenuItem("Add-Ons", new MenuItem("barbeque", 5));
menu.addMenuItem("Add-Ons", new MenuItem("turkey bacon", 10));
menu.addMenuItem("Add-Ons", new MenuItem("D&D sauce", 15));
menu.addMenuItem("Add-Ons", new MenuItem("jalapeno", 10));
// Create sample users
Customer customer = new Customer("Mohamed", "1234", "Cairo", "0111979699");
MenuItem burger = menu.getItemsByCategory("Chicken Burgers").get(0); // Crunchy Chicken
customer.addToCart(burger, 2);
customer.getcart();
}
}
Run
Main
Cart:
name: crunchy chicken price: 30.0 quantity: 2
Items in your cart:
Item: crunchy chicken | Quantity: 2 | Price: $30.0
Process finished with exit code 0



4) Increment/ Decrement quantity:

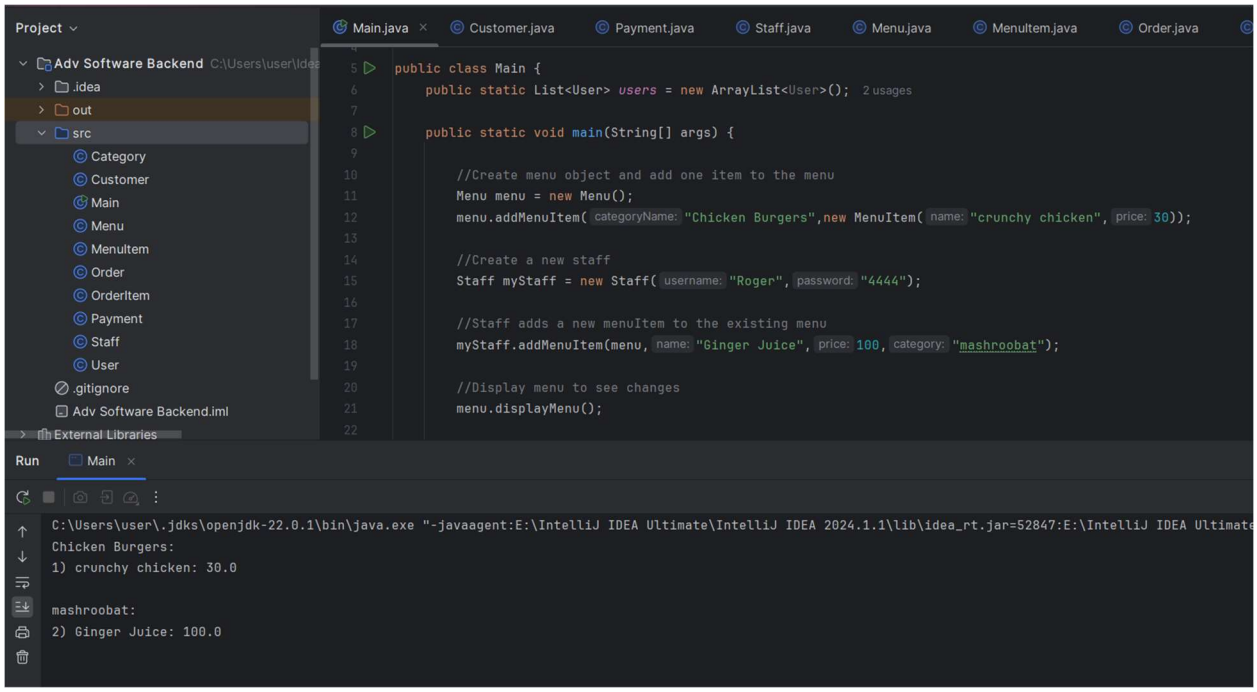
The Increment/ Decrement quantity functions were tested to ensure that users can modify the quantity of items in their cart. Test cases included verifying that the quantity updates correctly when the user increases or decreases it, and that the item's price adjusts accordingly.


Code With Me - project
Main
Project
Customer.java
Main.java
Menu.java
Category.java
MenuItem.java
Order.java
Staff.java
User.java
project
dinebackend1
.idea
out
src
Category
Customer
Main
Menu
MenuItem
Order
OrderItem
Payment
Staff
public class Main {
public static void main(String[] args) {
// Create sample users
Customer customer = new Customer(username: "Mohamed", password: "1234", address: "Cairo", phone: "0111979699");
MenuItem burger = menu.getItemsByCategory(categoryName: "Chicken Burgers").get(0); // Crunchy Chicken
customer.addToCart(burger, quantity: 2); // Add 2 burger
// View the items in the cart before incrementing or decrementing
System.out.println("Cart before increment/decrement:");
customer.getcart();
// Increment the quantity of crunchy chicken (should become 3)
customer.incrementQuantity(burger);
// View the items in the cart after incrementing
Run
Main
C:\Users\ENG-Mostafa\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.1\lib\idea_rt.jar=58033:C:\Program Files\JetBrains\IntelliJ IDE
Cart before increment/decrement:
Cart:
name: crunchy chicken price: 30.0 quantity: 2
Items in your cart:
Item: crunchy chicken | Quantity: 2 | Price: $30.0
Cart after incrementing the quantity of Crunchy Chicken:
Cart:
name: crunchy chicken price: 30.0 quantity: 3
Items in your cart:
Item: crunchy chicken | Quantity: 3 | Price: $30.0
project
dinebackend1
src
Main
main
(T) ↑182 B/s / ↓169 B/s
39:56


Code With Me - project
Main
Project
Customer.java
Main.java
Menu.java
Category.java
MenuItem.java
Order.java
Staff.java
User.java
project
dinebackend1
.idea
out
src
Category
Customer
Main
Menu
MenuItem
Order
OrderItem
Payment
Staff
public class Main {
public static void main(String[] args) {
// Increment the quantity of crunchy chicken (should become 3)
customer.incrementQuantity(burger);
// View the items in the cart after incrementing
System.out.println("\nCart after incrementing the quantity of Crunchy Chicken:");
customer.getcart();
// Decrement the quantity of fries (should become 0 and be removed)
customer.decrementQuantity(burger);
// View the items in the cart after decrementing
System.out.println("\nCart after decrementing the quantity of burger:");
customer.getcart();
Run
Main
Cart:
name: crunchy chicken price: 30.0 quantity: 3
Items in your cart:
Item: crunchy chicken | Quantity: 3 | Price: $30.0
Cart after decrementing the quantity of burger:
Cart:
name: crunchy chicken price: 30.0 quantity: 2
Items in your cart:
Item: crunchy chicken | Quantity: 2 | Price: $30.0
Process finished with exit code 0

5) Adding new item to menu:

The functionality of adding new items to the menu was tested by having an object of class staff call the addMenuItem function in Menu. We start by initializing our menu to initially have only one item, and testing that after adding an item, the count goes up to two items. We then create a staff member that calls the addMenuItem function with the needed parameters, and since the "Mashroobat" category didn't initially exist in the menu, a new category with that name is created and our menu item is added.



To verify our item has been added, we created a new testing function "displayMenu" in our Menu class to list our whole menu in the terminal. We can now confirm that our new item has been successfully added.

```java
public class Menu {  3 usages
    public void displayMenu() {  1 usage
        int i=1;
        for (Category category : categories) {
            System.out.println(category.getName()+":");
            for (MenuItem menuItem : category.getMenuItems()) {
                System.out.println(i+") "+menuItem.getName()+": "+menuItem.getPrice());
                i++;
            }
            System.out.println("");
        }
    }

```

6) Process payment

The process payment is meant to start the payment sequence by confirming the total price of the cart and prompting the user to choose either cash or credit payment methods. Two different cases were tested, the first case where the cart is empty, so we do not proceed with payment. And the second case is where there are items in the cart, where we then display the total price and prompt the user to choose payment method.

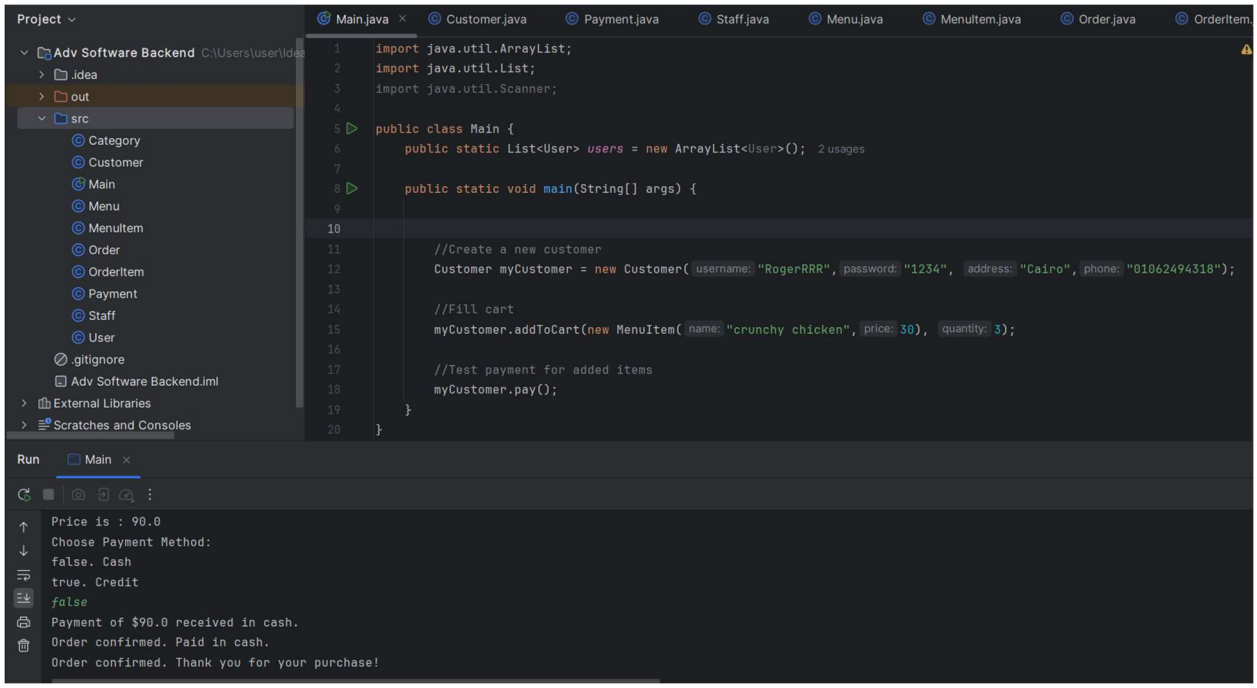
```java
public class Main {
    public static void main(String[] args) {



        //Create a new customer
        Customer myCustomer = new Customer( username: "RogerRRR", password: "1234",  address: "Cairo", phone: "01062494318");

        //Try to pay while cart is empty
        System.out.println("Testing with empty cart");
        myCustomer.pay();

        //Fill cart
        myCustomer.addToCart(new MenuItem( name: "crunchy chicken", price: 30),  quantity: 3);

        //Test payment for added items
        System.out.println("Testing with full cart");
        myCustomer.pay();
    }
}
```

```
Testing with empty cart
Cart is empty. Cannot proceed with payment.
Testing with full cart
Price is : 90.0
Choose Payment Method:
false. Cash
true. Credit
```

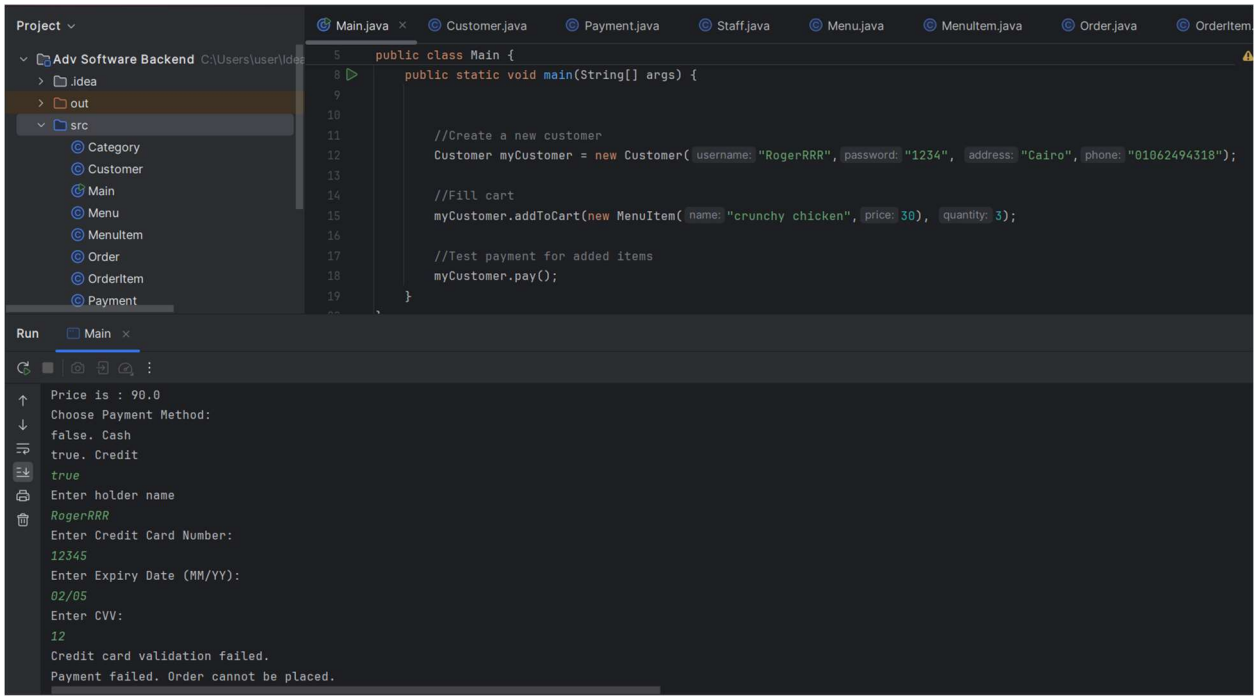
7) Process cash/credit payment & card validation

To test the cash/credit payment and card validation, we consider the following 3 cases:

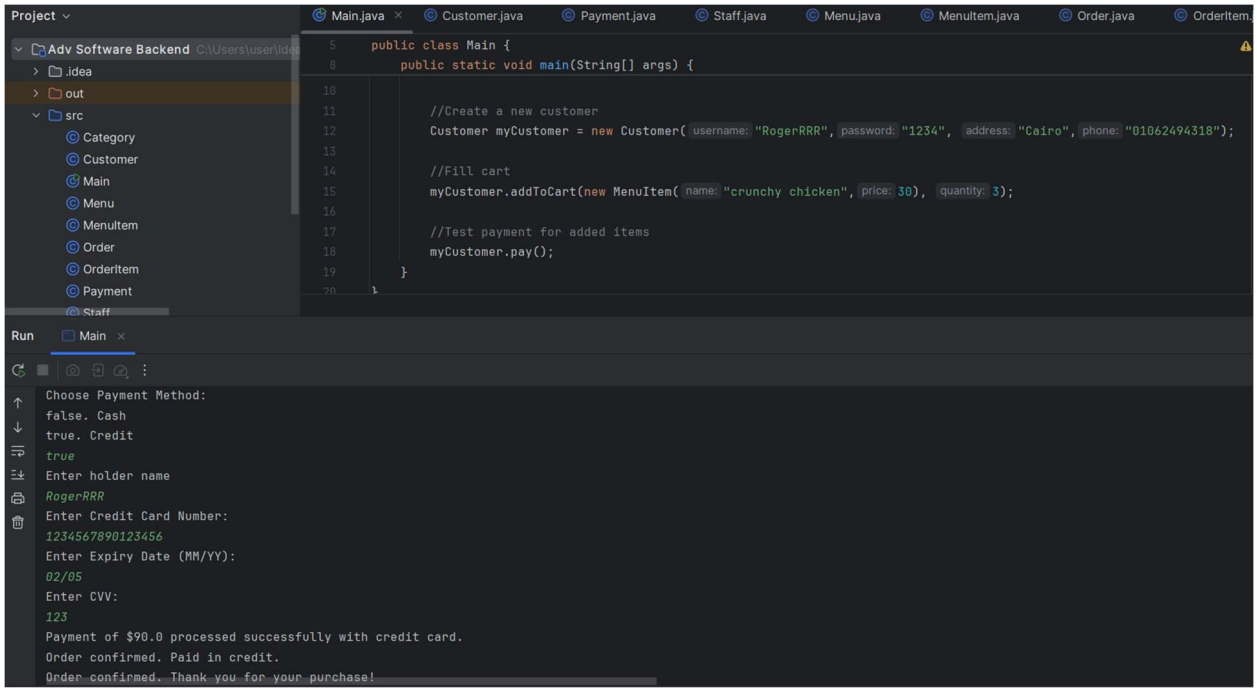
Case 1: Where the payment method is cash.



Case 2: In this case, the user selects "credit card" as the payment method but enters invalid credentials. Specifically, the card number entered does not meet the required 16-digit format, and the CVV does not match the required 3-digit format. As a result, the payment process is interrupted, and the order is not placed.



Case 3: In this case, the user selects "credit card" as the payment method and enters valid credentials. As a result, the payment is successfully processed, and the order is placed.


Project
Adv Software Backend C:\Users\user\Idea
.idea
out
src
Category
Customer
Main
Menu
MenuItem
Order
OrderItem
Payment
Staff
Main.java
Customer.java
Payment.java
Staff.java
Menu.java
MenuItem.java
Order.java
OrderItem.
public class Main {
public static void main(String[] args) {
//Create a new customer
Customer myCustomer = new Customer(username: "RogerRRR", password: "1234", address: "Cairo", phone: "01062494318");
//Fill cart
myCustomer.addToCart(new MenuItem(name: "crunchy chicken", price: 30), quantity: 3);
//Test payment for added items
myCustomer.pay();
}
Run
Main
Choose Payment Method:
false. Cash
true. Credit
true
Enter holder name
RogerRRR
Enter Credit Card Number:
1234567890123456
Enter Expiry Date (MM/YY):
02/05
Enter CVV:
123
Payment of $90.0 processed successfully with credit card.
Order confirmed. Paid in credit.
Order confirmed. Thank you for your purchase!

16.2 Integration Testing: -

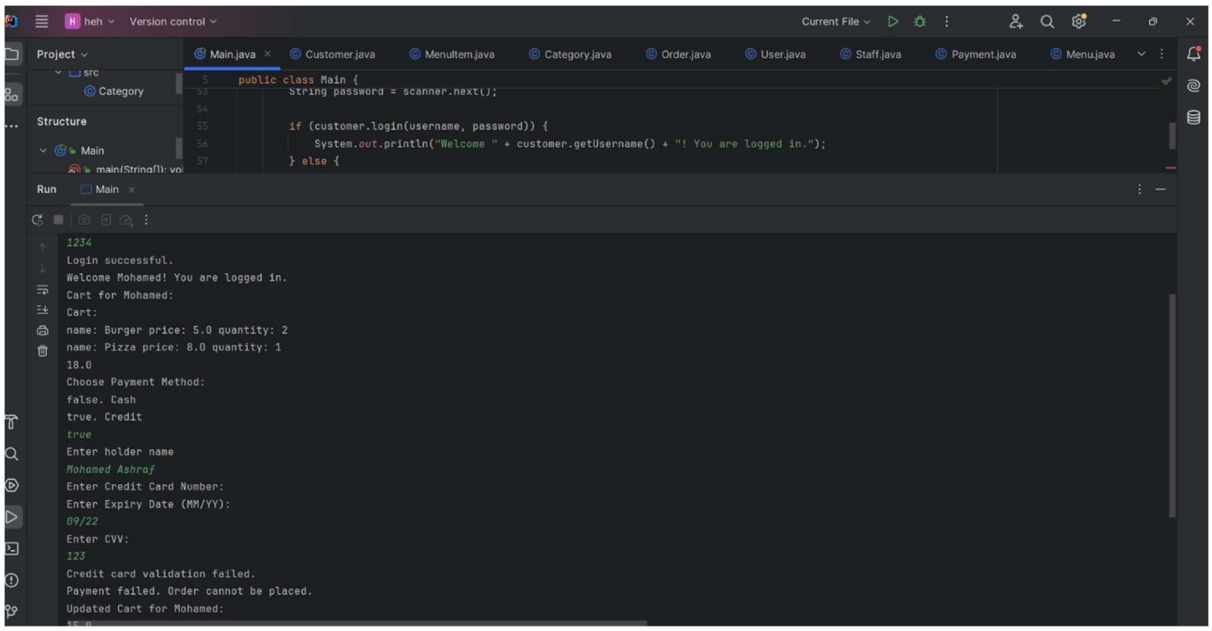
16.2.1 Integration between classes in the backend

In this scenario, a previously signed-up user, Mohamed logs in successfully, and his cart is displayed and it's empty. He then adds stuff to his cart which is displayed and proceeds to pay. He chooses to pay with credit but his payment is unsuccessful as he enters the wrong credit card info so he opts to pay with cash instead.

```
45          // Create a customer and add to the system
46          Customer customer = new Customer( username: "Mohamed", password: "1234", address: "123 Main St", phone: "123-4
47
48          // Customer logs in (dummy login for the sake of this test)
49          Scanner scanner = new Scanner(System.in);
50          System.out.println("Enter your username:");
51          String username = scanner.next();
52          System.out.println("Enter your password:");
53          String password = scanner.next();
54
55          if (customer.login(username, password)) {
56              System.out.println("Welcome " + customer.getUsername() + "! You are logged in.");
57          } else {
58              System.out.println("Login failed.");
59              return;
60          }
61
62          // Customer adds items to their cart
63          MenuItem burger = menu.findCategories( categoryName: "Fast Food").findMenuItem( name: "Burger");
64          MenuItem pizza = menu.findCategories( categoryName: "Fast Food").findMenuItem( name: "Pizza");
65
66          customer.addToCart(burger,  quantity: 2);  // Add 2 burgers
67          customer.addToCart(pizza,  quantity: 1);   // Add 1 pizza
68
69          // Display cart contents
70          System.out.println("Cart for " + customer.getUsername() + ":");
71          customer.getCart();  // Display cart
72
```

```
        // Display cart contents
        System.out.println("Cart for " + customer.getUsername() + ":");
        customer.getCart();  // Display cart

        // Customer proceeds to pay
        customer.pay();  // Proceed to payment (choose cash or credit)

        // Customer updates cart and proceeds with another order
        customer.addToCart(burger, quantity: 1);  // Add 1 more burger
        customer.decrementQuantity(pizza);  // Decrease the quantity of pizza

        // Display updated cart
        System.out.println("Updated Cart for " + customer.getUsername() + ":");
        customer.displayCart();

        // Customer makes a new payment (Cash or Credit)
        customer.pay();  // Payment process
    }
}
```

Figure 1 Backend Testing integration between classes main code



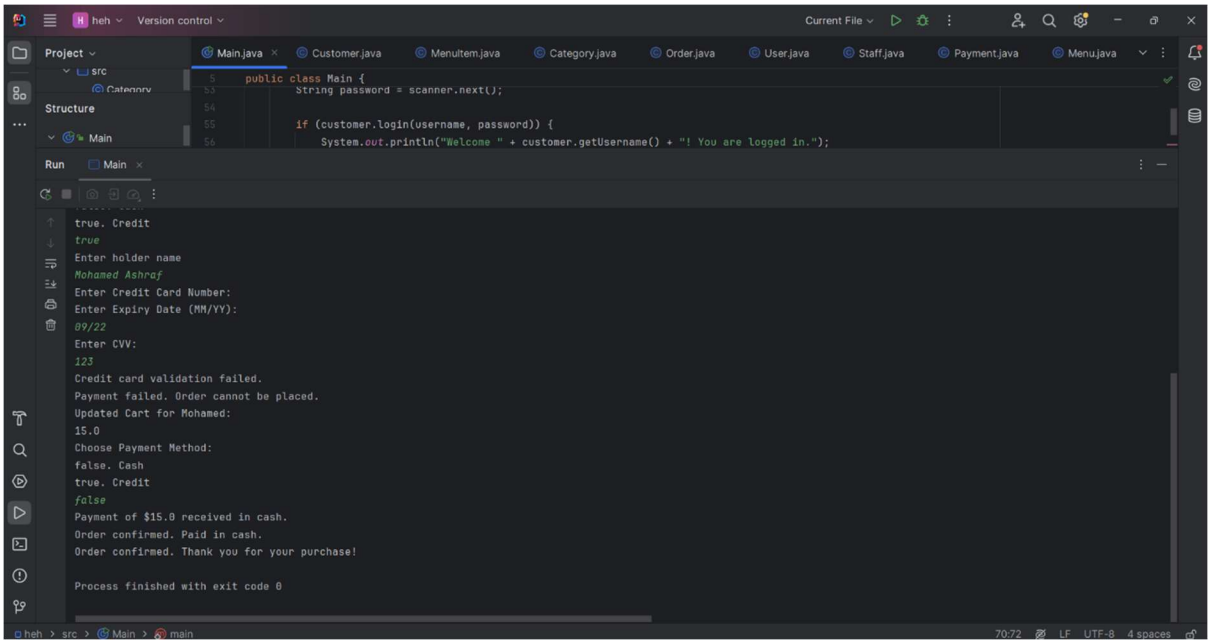


Figure 2 Output showing Integration between classes is working well

16.2.2 GUI with Backend Integration Testing

1) Test 1

In this test scenario we try to login in with this user , then we presses the menu, adds items to the cart then views the carts , presses pay, checks credit first then changes his mind and picks cash then presses pay again to confirm his order

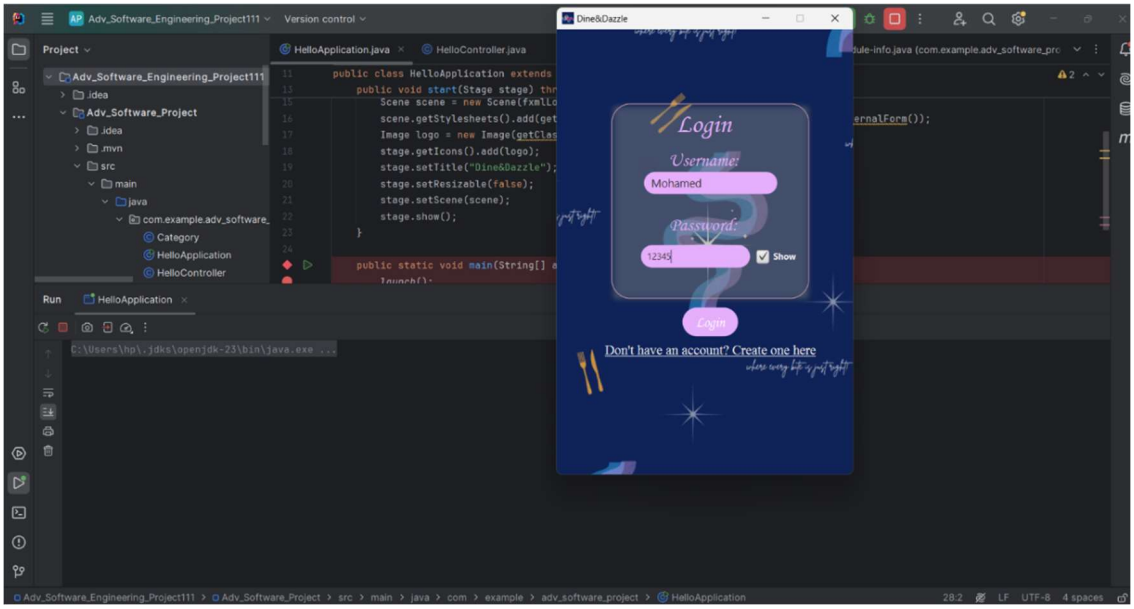


Figure 3 login page before logging in

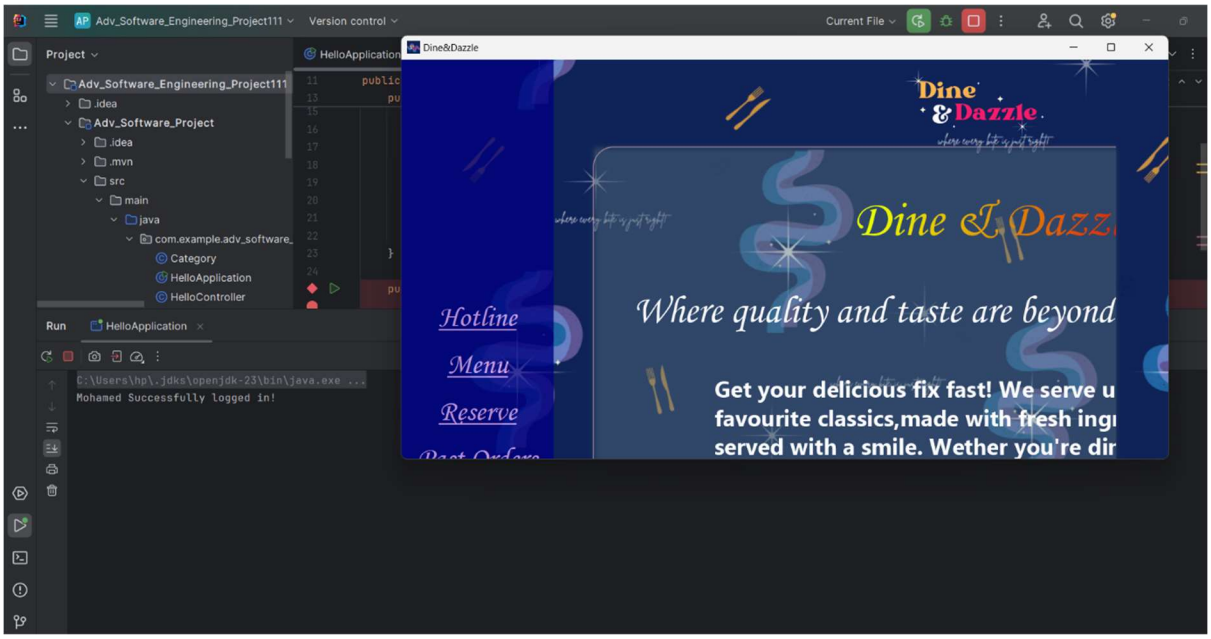


Figure 4 terminal showing login was successful

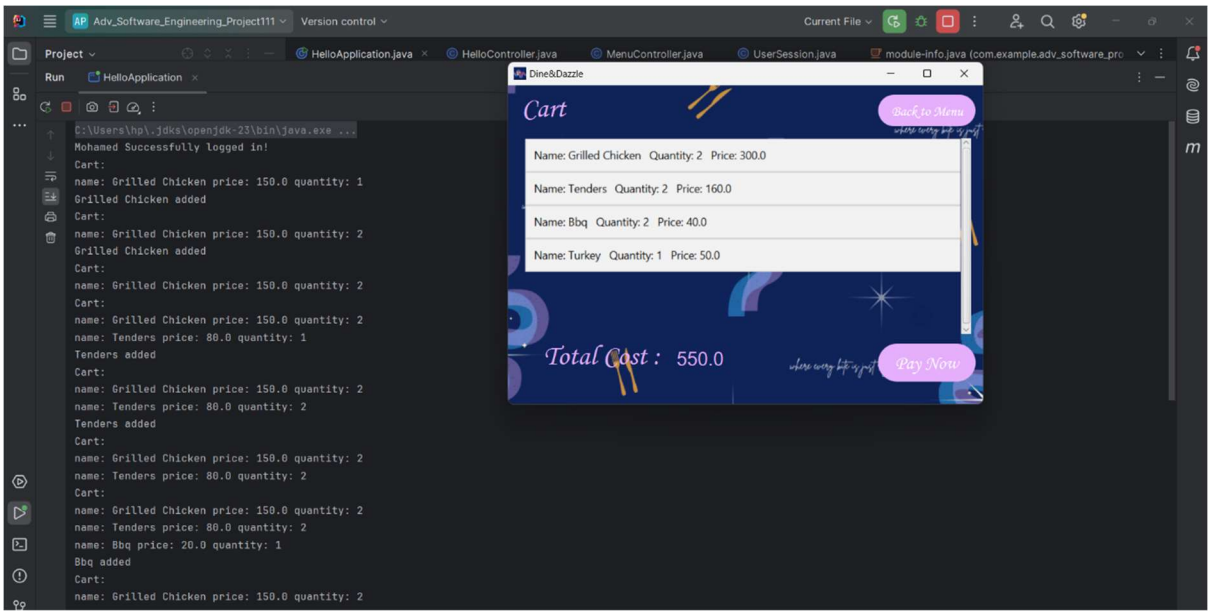


Figure 5 Cart view and whats in it is also displayed in terminal

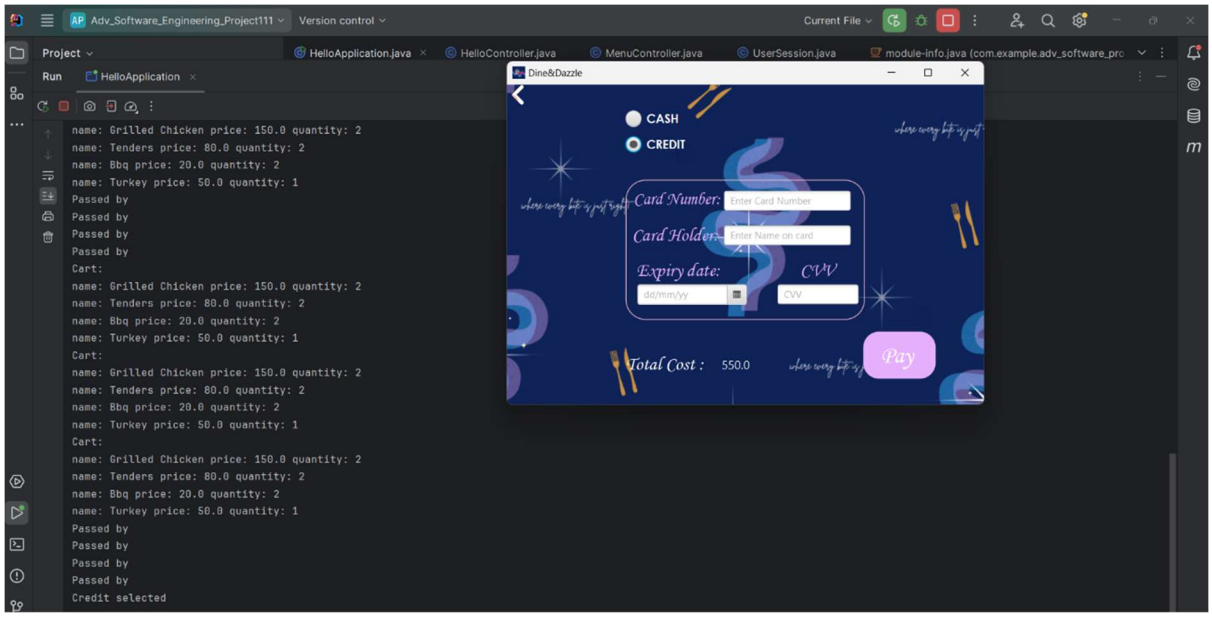


Figure 6 Terminal showing Credit payment is selected

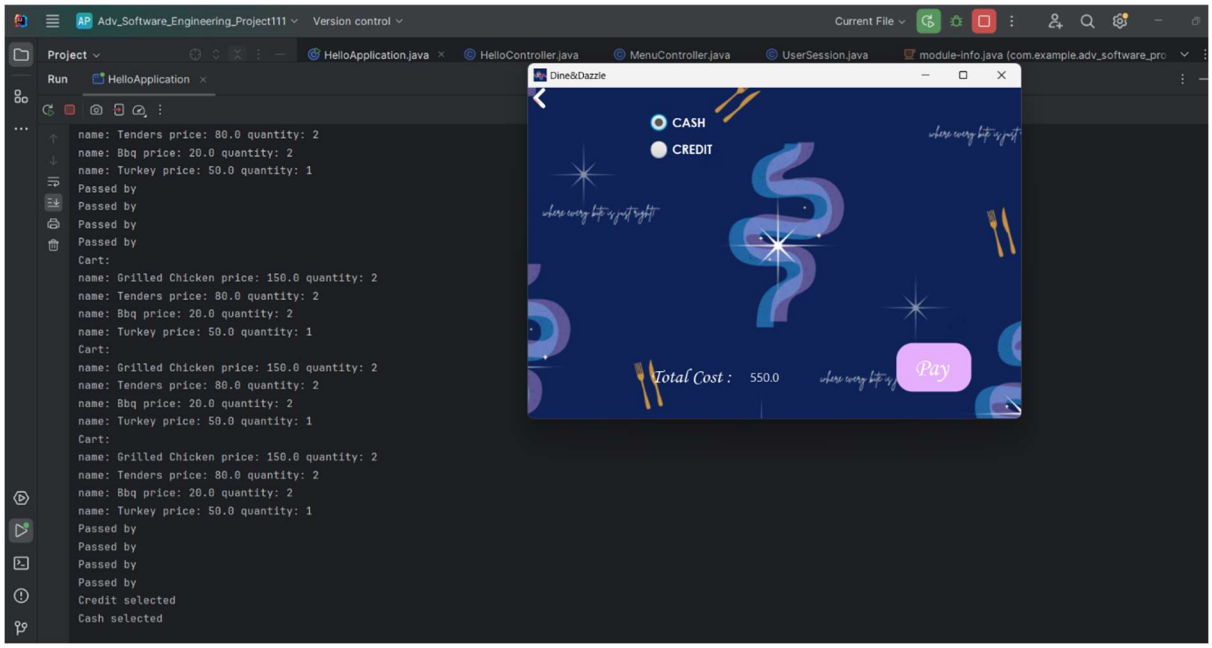


Figure 7 Terminal showing Cash Payment is selected

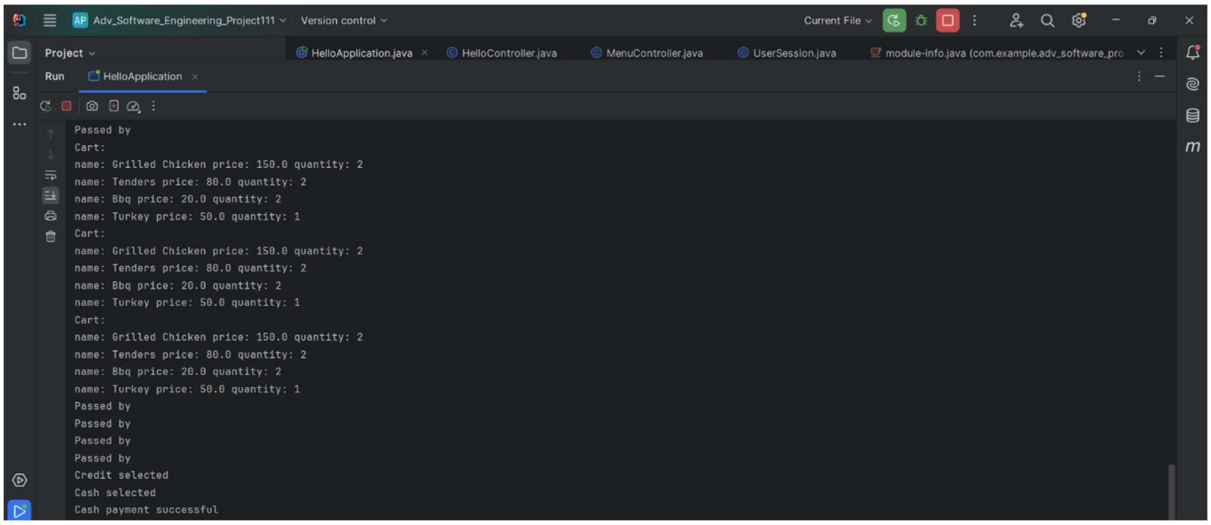


Figure 8 Termial showing Cash payment was succesfull

2) Test case 2

In the following testing scenario, the user doesn't initially have an account, so he goes to sign up and fills the fields and then proceeds to log in to his created account. The user then follows a normal scenario of adding items to his card and proceeds to pay by his credit card.

Figure 9 User sign-up

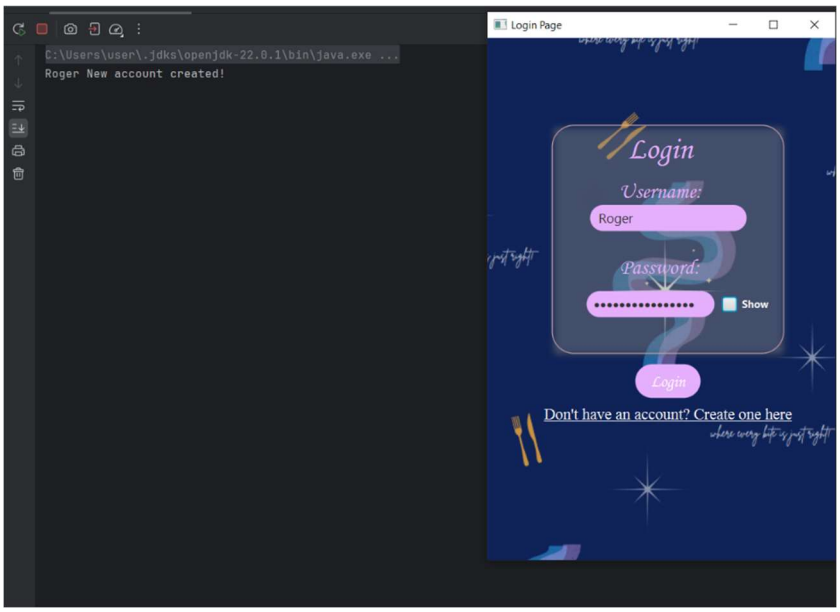


Figure 11 User Login

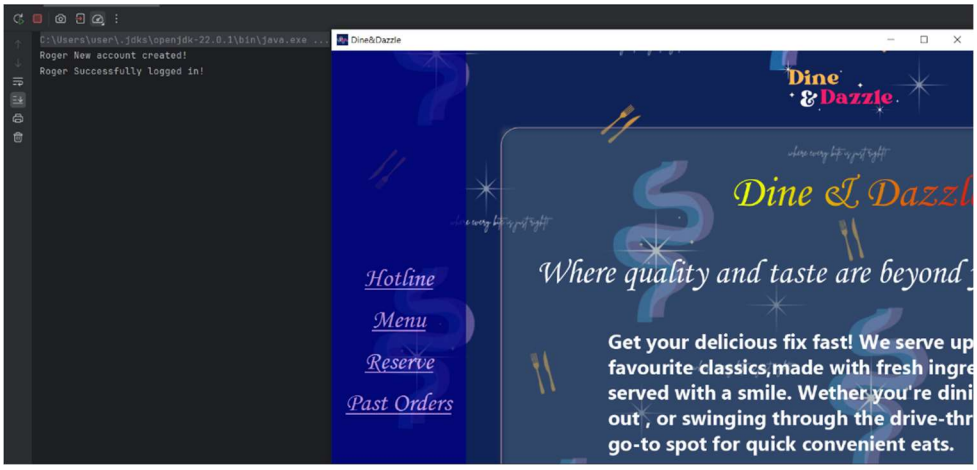


Figure 10 User Logged in and in Home page

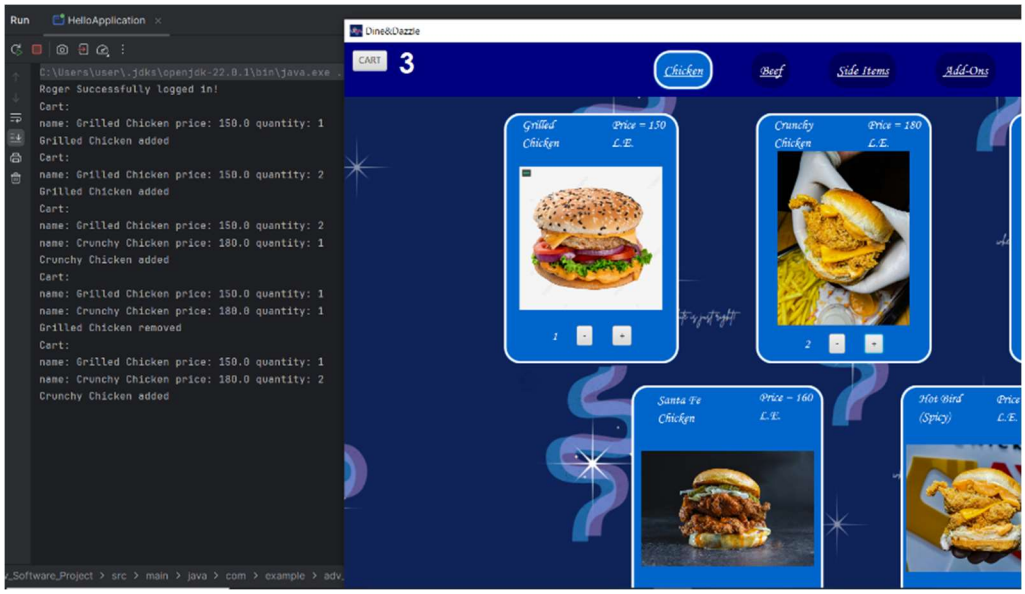


Figure 13 User added and removed multiple items from cart

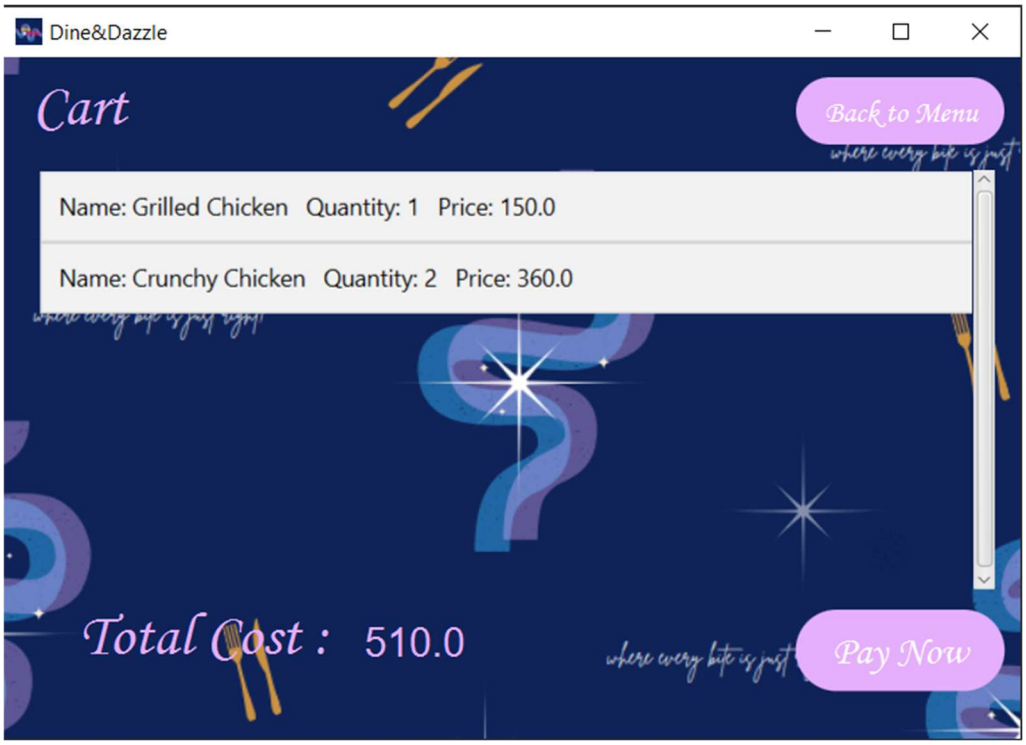


Figure 12 Viewing cart before payment

Figure 14 Credit Card selected for payment

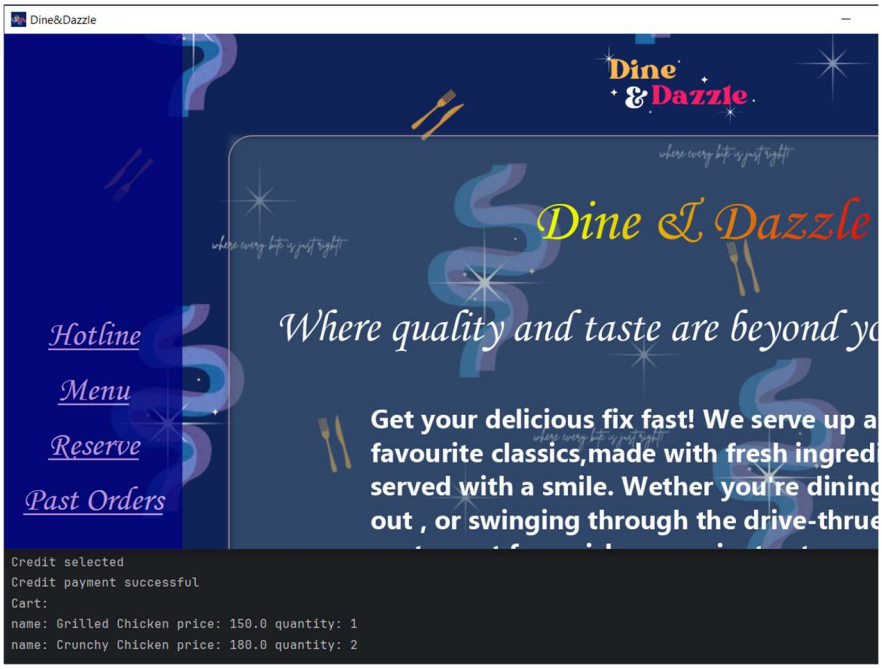
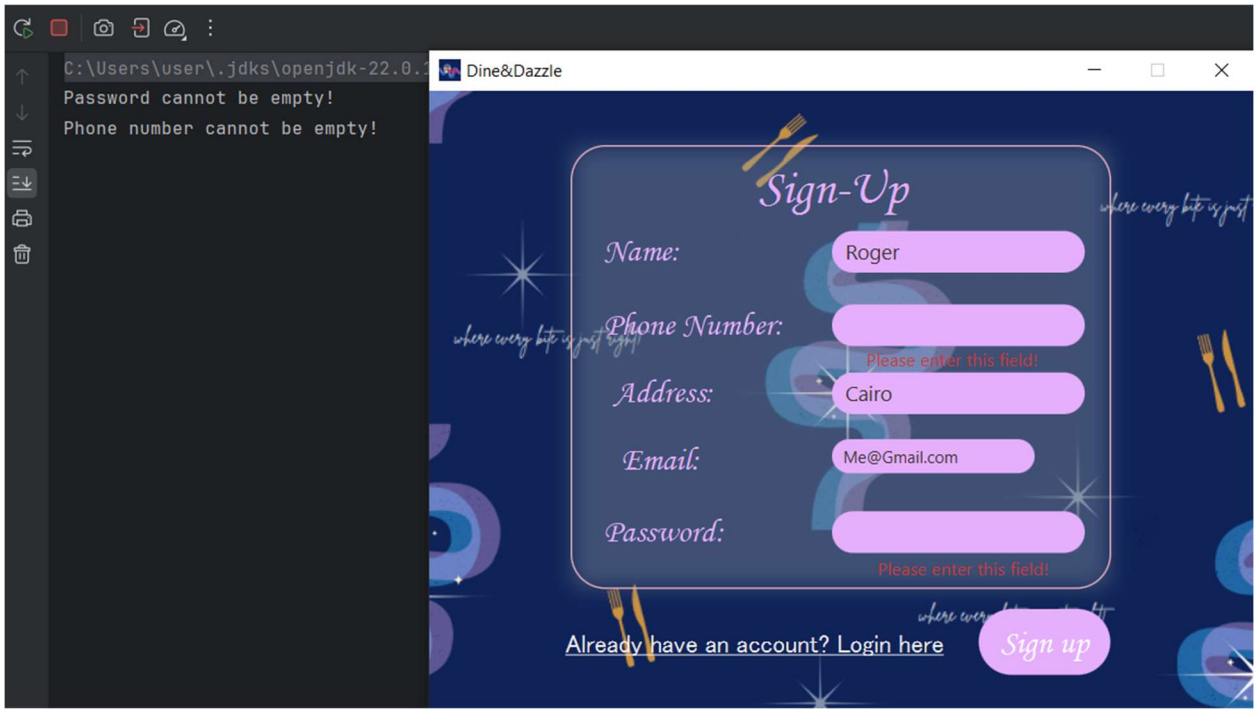


Figure 15 Payment successful, back to Home

3) Test case 3

In this scenario, the user attempts to sign-up with missing information, so he is prompted with an error message.



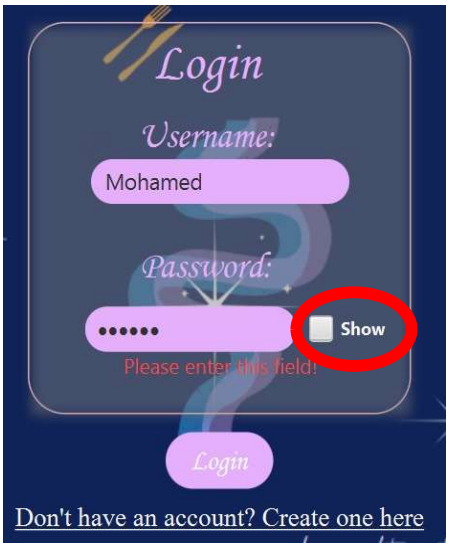
16.3 UI testing: -

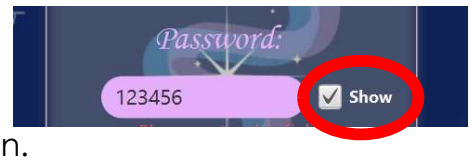
UI TESTING: The following is a guide to Dine & Dazzle, a food servicing web application.

LOGIN PAGE:

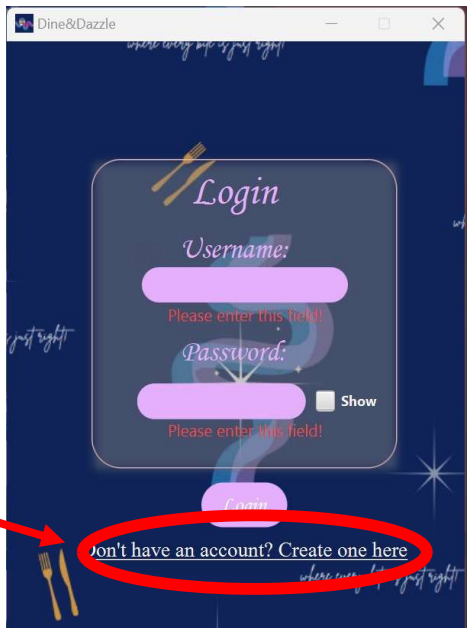
As soon as the web app is launched, you, the user, will be faced with the login terminal shown in the photos, where you will be required to type in your username and password to access the user homepage. However, you won't be able to access the homepage for the following reasons: you entered wrong user credentials, you don't have an account, or you didn't enter any information.

As you type your credentials in the "Username" and the "Password" textboxes, you will notice that any inputted characters in the "Password" textbox is hidden, to display your input, check the "show" checkbox on the right side of the "Password" textbox and uncheck it if you want to hide it again.



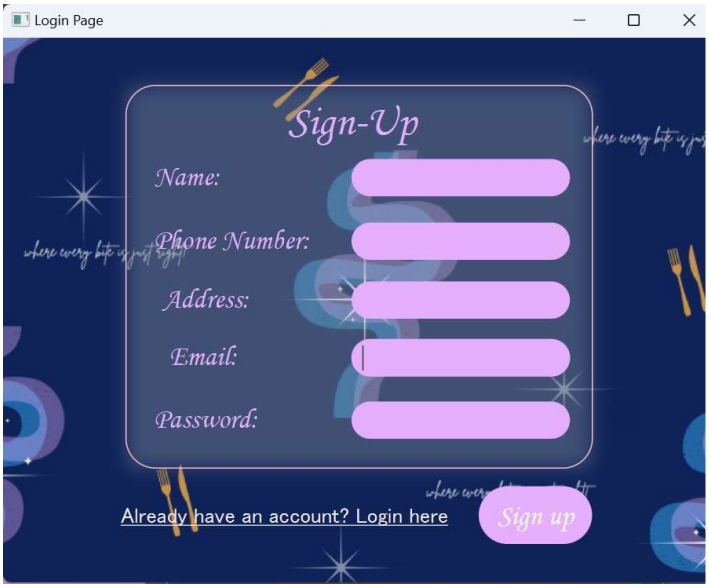


If you don't have an account, and you want to access the web app, then you will have to create an account by pressing "Don't have an account? Create one here" prompt, you will be directed to the Signup Page.



SIGNUP PAGE:

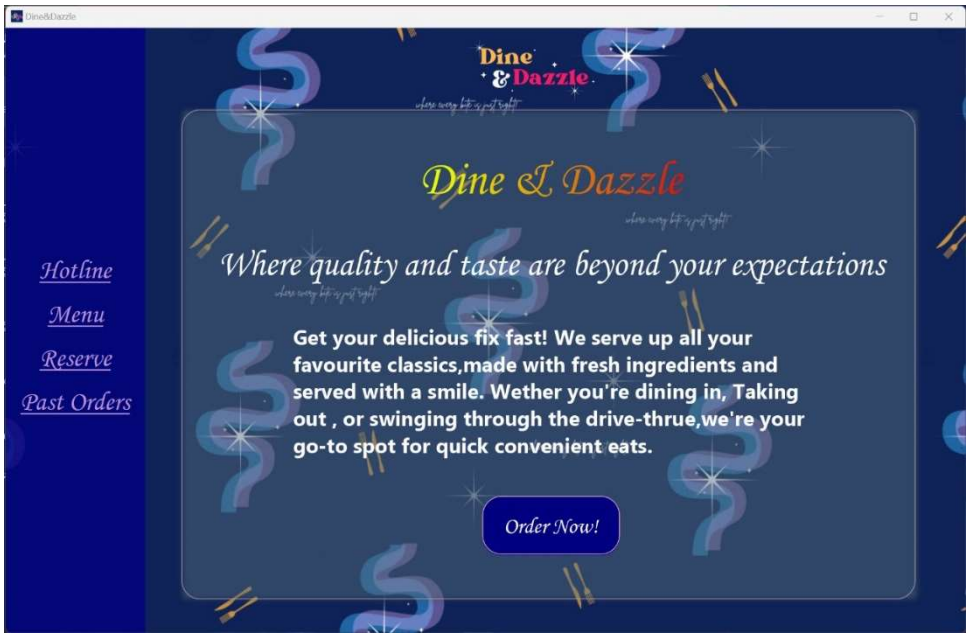
When the User chooses to create an Account, he/she will be redirected to the page displayed in the photos shown, where he/she will be required to provide their name, phone number, address, email address, and password



If the user doesn't fill in any of the user credentials' textboxes and presses the "Sign up" button, a message in red will be displayed under each field, "Please enter this field!". However, if the user signs up correctly, by filling in each field and pressing the 'Sign up" button he will be redirected to the Login page, where he/she will be required to enter their username and password and press the "Login" button. If the user already has an account, then he can simply press "Already have an account? Login here" prompt, where he/she will be redirected to the "Login" page.

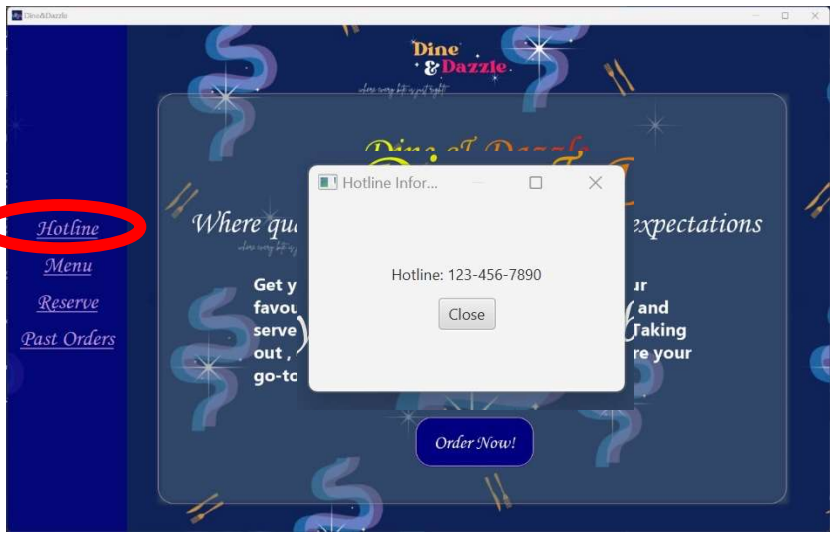
HOMEPAGE:

When The user successfully logs in to his/her account, he will be redirected to the user homepage, where the user can view the hotline, view the menu, head to the reservations' window, and view his/her past orders.



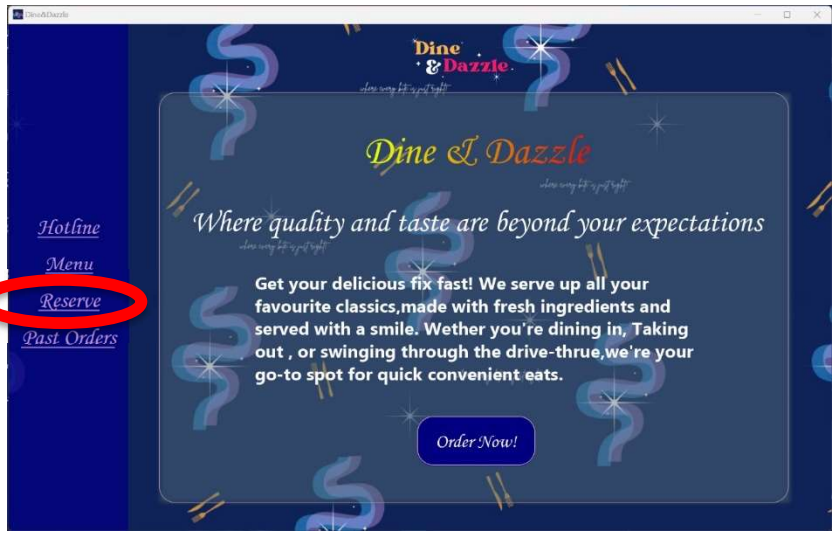
HOTLINE PAGE:

If the user presses the "Hotline" button, a popup with the name "Hotline Information" will appear, displaying the "Dine & Dazzle" hotline.

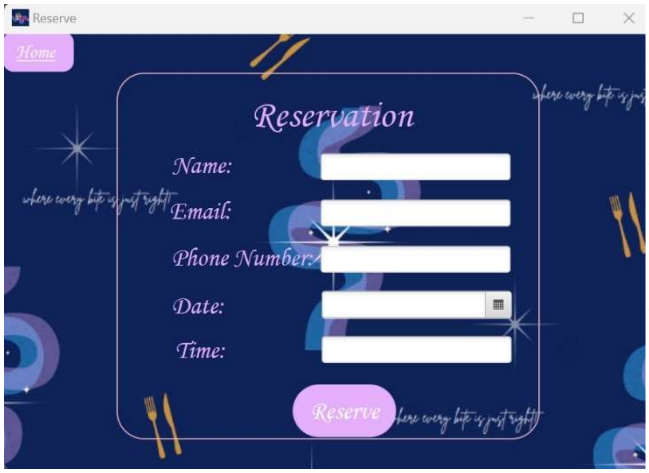


RESERVATION PAGE:

If the user presses the "Reserve" button, he will be redirected to the Reservation Page

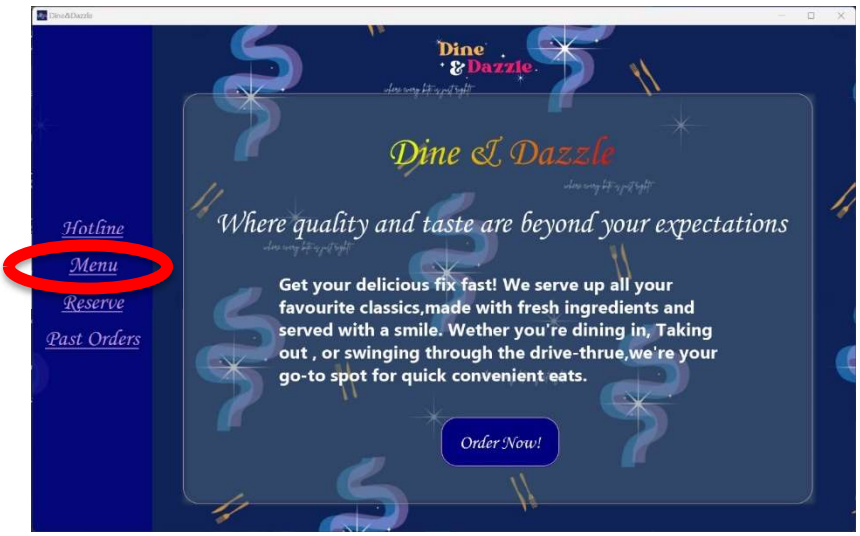


In the Reservation Page, the user will be asked to enter his name, email, phone number, and the date and time of the reservation. If the user entered any invalid data, he will be asked to reenter them, and if the user entered any unavailable date or time, he will be informed and asked to choose another.

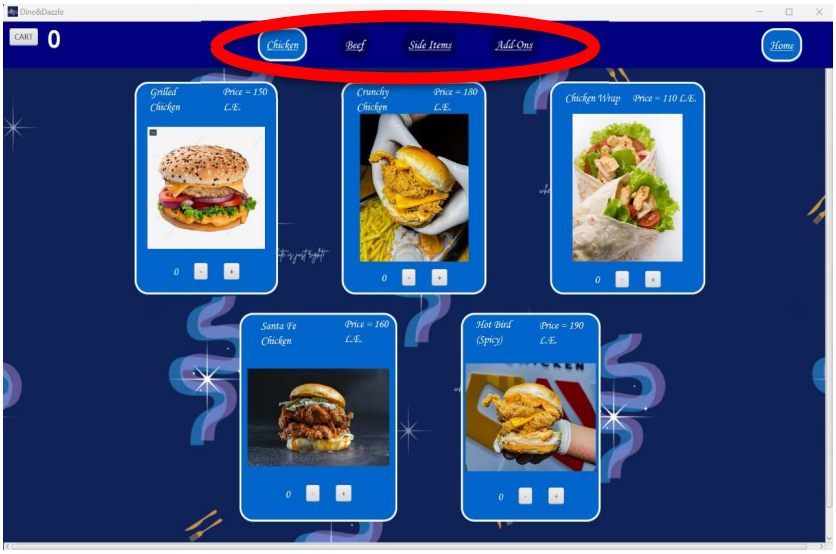


MENUPAGE:

If the user chooses to press the “Menu” button, he will be redirected to the “Dine & Dazzle” menu.

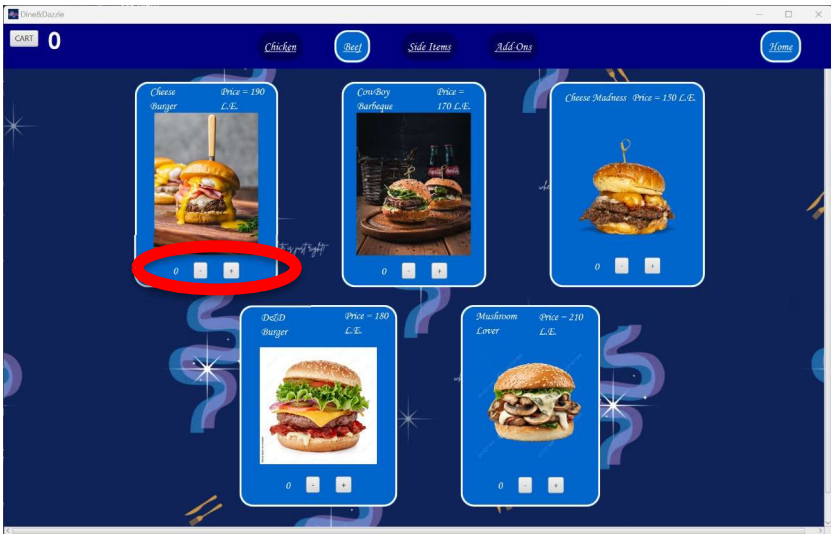


In the menu, several food categories will be shown, and the default menu category that initially loads is the chicken category. The user can choose different categories from the top area circled in the following screenshot.



In each category the user can add and remove any food item he wants and control the quantity by pressing the “+” and “-“ buttons under each food item in the menu.

Whenever the user adds a food item by pressing the “+”, it will be added in the cart, and whenever he/she press the “-“ it will be removed from the user’s cart. Whenever the user adds something to the cart the



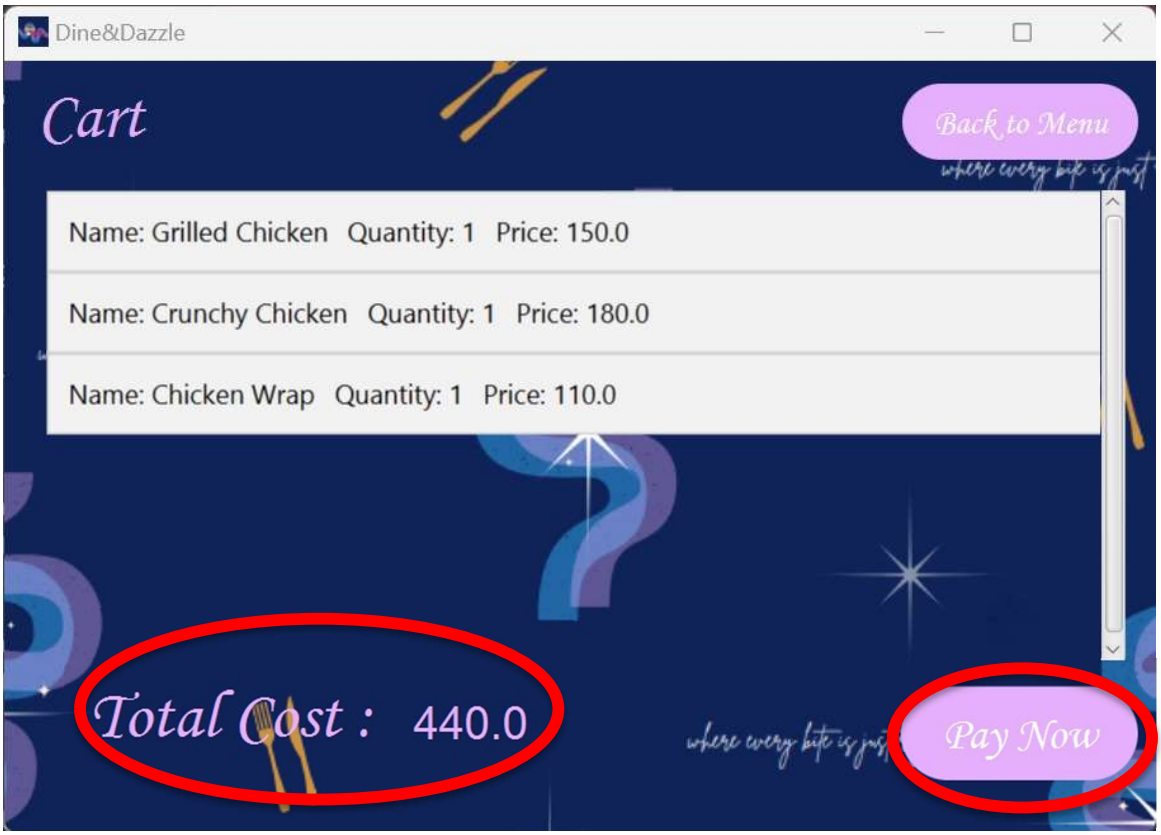
number beside the cart button is incremented.

CART PAGE:

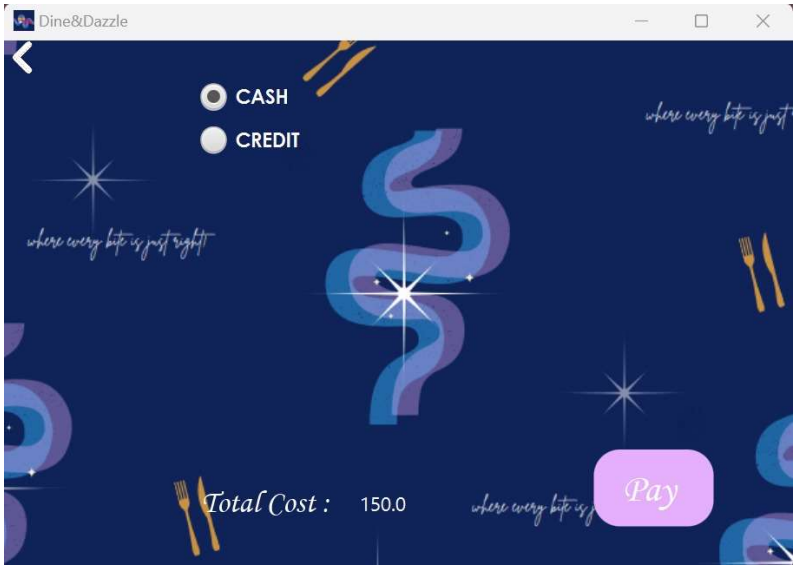
When the user presses the “Cart” button, circled in the screenshot below, he/she is redirected to their cart’s page.

If the user didn’t add any item from the menu page, the text message “No items in your cart :(“ will be shown, the user can then choose to return to the menu, to either add items or return to the homepage.

However, as seen in the following photo, whenever the user adds any item from the menu to the cart, it will be displayed in the cart page, and the total cost of the user's chosen items will be displayed near the bottom left corner, as well as the user will now be allowed to go to the payment page, by pressing the "Pay Now" button, which he/she weren't able to press when they had no items in cart.

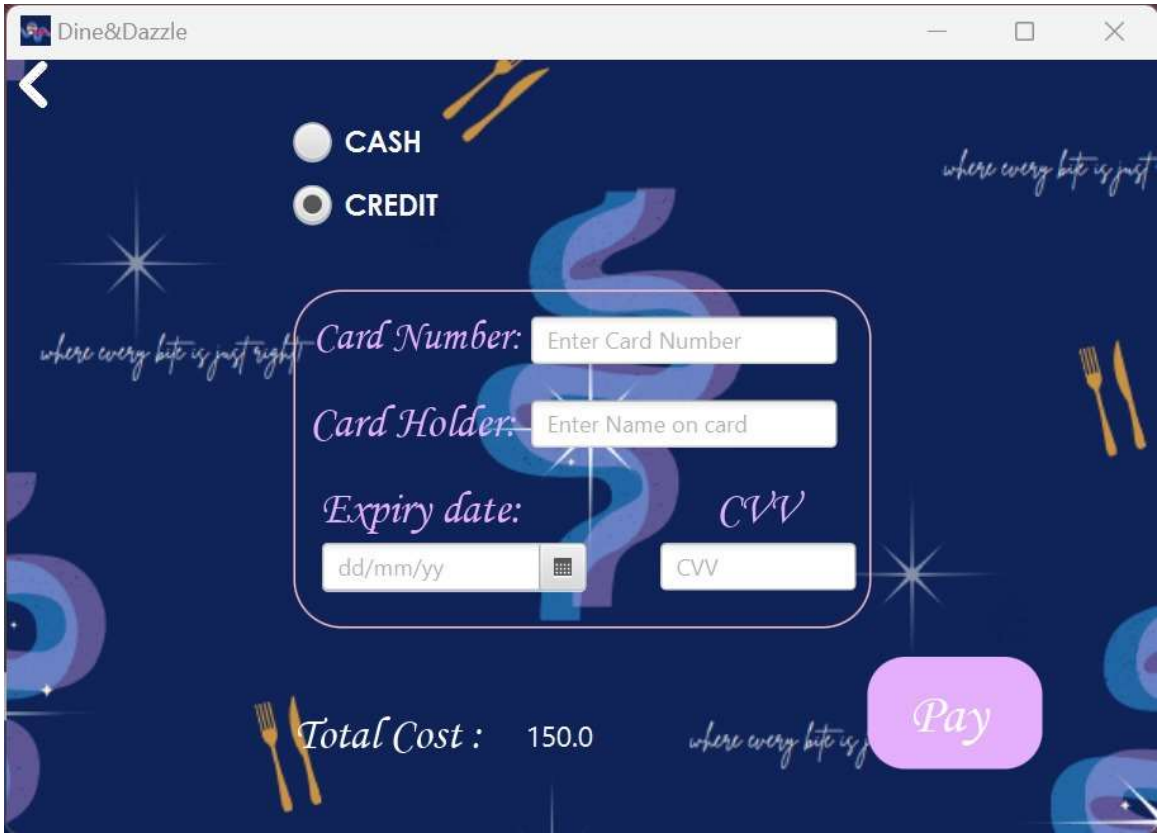


PAYMENT PAGE:



Whenever the user presses the "Pay Now" button in the Cart page, he/she is automatically redirected to the payment page, where the user will be asked to choose a payment method; cash or credit, to confirm payment the user has to press the "Pay" button. If the user chose cash, then he will have to go to any branch to pay for his order.

However, if the user chose credit, text boxes prompting the user to enter his credit card number, card holder name, expiry date and the cvv will appear and these credentials will then be validated, and if valid, successful payment will take place, and the user will be returned to the homepage.



17. References: -

1. **References for Diagrams and Modeling Techniques**

- Dr Gamal A. Ebrahim Slides
- UML Diagrams: *UML Specification by OMG (Object Management Group)*
  https://www.omg.org/spec/UML
  Why to cite: Official documentation for UML, covering all standard diagram types.
- Visual Paradigm: *UML Diagram Tutorials*
  https://www.visual-paradigm.com/guide/uml-unified-modeling-language
  Why to cite: Guides and examples for creating various UML diagrams.
- Lucidchart: *What is Object-Oriented Analysis and Design (OOAD)?*
  https://www.lucidchart.com/pages/what-is-object-oriented-analysis-and-design
  Why to cite: Overview of OOAD concepts and techniques.

2. References for Frontend and Backend Development

- Spring Framework: *Spring Documentation*
  https://spring.io/projects/spring-framework
  Why to cite: Official guide for developing Java-based backend applications.
- JUnit: *JUnit 5 User Guide*
  https://junit.org/junit5/docs/current/user-guide/
  Why to cite: Reference for implementing unit tests in Java.

3. References for Architecture

- IBM: *Software Architecture Overview*
  https://www.ibm.com/docs/en/
  Why to cite: Industry-standard practices for designing software architectures.
- Microsoft: *Designing Distributed Systems*
  https://learn.microsoft.com/en-us/azure/architecture/guide/
  Why to cite: Comprehensive resource for modern architectural patterns.

4. References for OOAD Methodologies

- GeeksforGeeks: *Object-Oriented Analysis and Design (OOAD)*
  https://www.geeksforgeeks.org/object-oriented-analysis-and-design/
  Why to cite: Covers OOAD basics, use case design, and class diagrams.
- IBM Developer: *OOAD Techniques and Applications*
  https://developer.ibm.com/articles/ooad-intro/
  Why to cite: Explains OOAD principles in an easy-to-follow manner.

5. References for Documentation Standards

- IEEE Standards Association: *Software Engineering Standards*
  https://standards.ieee.org/
  Why to cite: Guidelines for documenting software processes and requirements.
- ISO: *Requirements Engineering Standards*
  https://www.iso.org/standard/72046.html
  Why to cite: Standards for documenting use cases and system requirements.

6. Tools Used

- IntelliJ IDEA: *Official Documentation*
  https://www.jetbrains.com/idea/documentation/
  Why to cite: Documentation for the IDE used for implementation.
- Lucidchart: *Diagramming Software*
  https://www.lucidchart.com/
  Why to cite: Tool used for creating diagrams like use case, sequence, and class diagrams.
- Scene Builder: *Official Gluon Scene Builder Documentation*
  https://gluonhq.com/products/scene-builder/
  Why to cite: Tool for designing JavaFX user interfaces using a drag-and-drop WYSIWYG approach.