

Phase 2

Restaurant system Documentation report

MMA restaurant BY

Name	ID
Mazen Ahmed	6999
Mohamed Ahmed	7004
Amr Ahmed	6798

Restaurant Menu System Requirements:

1. Function Requirements:

1.1. Menu Display

- The system should display the restaurant menu, including food and drink items.
- The menu should be categorized into sections such as appetizers, main courses, desserts, etc.
- Each menu item should include a name, description, price, and any applicable dietary information.

1.2. Ordering

- The system should enable customers to select menu items and add them to their order.
- Customers should be able to specify quantity for each item.
- The system should calculate the total cost of the order based on the selected items and provide a summary later in the cart summary.

1.3. Order Confirmation and Feedback

- The system should provide customers with a confirmation of their order, and choosing payment method as cash or paypal.

1.4. Menu Management

- The system should allow restaurant staff to add, remove, or update menu items as needed.
- Staff should be able to specify prices, descriptions, and other relevant information for each menu item.

1.5. Order Management

- The system should allow restaurant staff to view and manage incoming orders.
- Staff should be able to mark orders as "in progress" or "completed" and update the order status.

1.6. Payment Processing

- The system should support various payment methods, such as credit cards, cash, or mobile payments.
- Customers should be able to pay for their orders securely and easily through the system.

2. Non-Functional Requirements:

2.1. User Interface

- The system should have an intuitive and user-friendly interface for customers to browse the menu and place orders.
- The interface should be responsive and compatible with the device in use.

2.2. Performance

- The system should have fast response times to ensure a smooth and efficient user experience.
- Menu items should load quickly provide results promptly.

2.3. Reliability and Availability

- The system should be highly reliable, ensuring that menu items and order information are consistently accurate and up-to-date.
- The system should have minimal downtime and be available to customers during restaurant operating hours.

2.4. Security

- The system should implement appropriate security measures to protect admin data, including order history and payment information.
- Admin authentication and authorization should be in place to ensure only authorized individuals can access the system.

2.5. Scalability

- The system should be scalable to handle a growing number of menu items, customers, and orders without performance degradation.

User Requirements for Restaurant Menu:

1. Functionality Requirements:

1.1. Add to Order

- Users should be able to add desired items to their order, specifying quantities or customization options if available.

1.2 Dietary Information

- The menu should provide clear indications of dietary information, such as vegetarian, vegan, gluten-free, or allergen-free options.

1.3 Pricing and Currency

- The system should display accurate prices in the appropriate currency, including any applicable taxes or additional charges.

2. Non-Functional Requirements:

1.1. User-Friendly Interface

- The menu system should have an intuitive and easy-to-use interface, allowing users to navigate through categories, items, and options effortlessly.

1.2. Compatibility

- The menu system should be compatible with the device.

1.3. Usability

- The system should be designed with an intuitive and easy-to-use interface, allowing users to navigate through categories, items, and options effortlessly. The system should provide clear instructions to guide users through the menu selection and ordering process.

Agile methodology:

In this project, the Agile methodology will be employed to develop the restaurant menu system. Agile is an iterative and collaborative approach that promotes flexibility, adaptability, and continuous improvement throughout the development process. It allows for frequent feedback, quick decision-making, and the ability to respond to changing requirements.

1. **Daily Stand-up Meetings:** Daily stand-up meetings will be conducted, where the development team members will provide brief updates on their progress, discuss any challenges or impediments, and coordinate their work. These meetings foster communication, collaboration, and transparency among team members.
2. **Incremental Development:** The menu system will be developed incrementally, with each sprint delivering a potentially shippable product increment. The development team will work on implementing the selected user stories, focusing on delivering value to the users at the end of each sprint.

Incremental features (Division of phases)

- Create Database and Database tables
- User login and logout
- Admin login and logout
- User welcome page
- Admin dashboard
- Admin manage tables
- Admin manage categories
- Admin manage dishes
- Admin manage orders
- Admin update order status
- User welcome page
- User choosing categories and dishes
- User manage cart
- User checkout
- User seeing orders and managing orders

3. **Continuous Integration and Testing:** Throughout the development process, continuous integration and testing practices will be followed. This means that the code changes made by different team members will be integrated regularly, ensuring that any conflicts or issues are identified and resolved early. Automated tests will be written and executed to verify the functionality and performance of the menu system.

4. **Frequent User Feedback:** User feedback and involvement are essential in Agile development. Regular feedback sessions will be conducted at the end of each sprint to gather feedback on the implemented features. This feedback will be used to make necessary adjustments, refine requirements, and prioritize future development efforts.

5. **Adaptability and Prioritization:** Agile allows for flexibility in responding to changing requirements or priorities. If new user needs or market trends emerge, the development team can adapt the project scope and adjust the backlog accordingly. By constantly reevaluating and reprioritizing user stories, the team can ensure that the most valuable features are developed and delivered first.

By utilizing the Agile methodology, the development team can embrace change, deliver value incrementally, and continuously improve the restaurant menu system based on user feedback and evolving needs. This iterative approach allows for greater flexibility, adaptability, and collaboration, ultimately resulting in a high-quality and user-centric product.

Software Design and Implementation

Software design and implementation are interleaved activities the level of detail in the design depends on the type of system and whether you are using a plan-driven or agile approach

Software Design

to design a database schema of the restaurant ordering system to manage the users, table bookings, menus, inventory, orders, and payments. It provides the food order database design to manage the food orders for restaurants. It can be further used to develop on-premises restaurant order system applications.

Such ordering systems are implemented to automate order processing and efficiently deal with peak ordering times, thus improving customer satisfaction with less efforts

- The very first step is to create the Restaurant Database

Database: `mma_restraunt`

- We will need a table for admin information

```
CREATE TABLE `admin` (`adm_id` int(11) NOT NULL, `username` varchar(222) NOT NULL, `password` varchar(222) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- we will design the **Tables** table to store table information(number and status).

```
CREATE TABLE `tables` (`table_id` int(11) NOT NULL, `status` varchar(50) NOT NULL DEFAULT '1') ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- we will implement tables for categories and dishes (where each category has its own dishes/menu)

```
CREATE TABLE `dishes` (`d_id` int(11) NOT NULL, `category_id` int(11) DEFAULT NULL, `title` varchar(222) NOT NULL, `slogan` varchar(222) NOT NULL, `price` decimal(10,2) NOT NULL, `img` varchar(222) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `category` (`category_id` int(11) NOT NULL, `title` varchar(222) NOT NULL, `image` text NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- Finally, we will implement a table for user orders

```
CREATE TABLE `users_orders` (`o_id` int(11) NOT NULL, `table_id` int(11) DEFAULT NULL, `title` varchar(222) NOT NULL, `quantity` int(11) NOT NULL, `price` decimal(10,2) NOT NULL, `status` varchar(222) DEFAULT NULL, `date` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()) ENGINE=InnoDB DEFAULT
```

So finally we will have that database with these tables

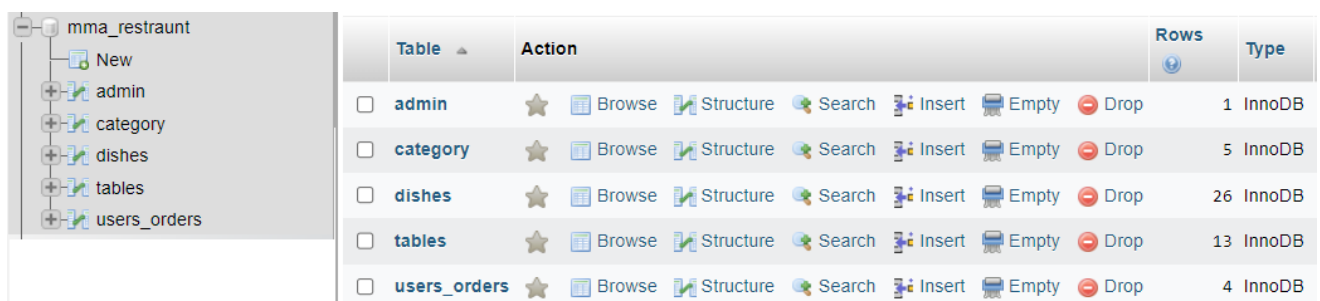


Table	Action	Rows	Type
<input type="checkbox"/> admin	★ Browse Structure Search Insert Empty Drop	1	InnoDB
<input type="checkbox"/> category	★ Browse Structure Search Insert Empty Drop	5	InnoDB
<input type="checkbox"/> dishes	★ Browse Structure Search Insert Empty Drop	26	InnoDB
<input type="checkbox"/> tables	★ Browse Structure Search Insert Empty Drop	13	InnoDB
<input type="checkbox"/> users_orders	★ Browse Structure Search Insert Empty Drop	4	InnoDB

Implementation (Development Environment and coding)

A development environment in software and web development is a workspace for developers to make changes without breaking anything in a live environment.

In some cases, the term "development environment" is used to refer to an Integrated Development Environment (IDE).

Microsoft's Visual Studio . NET. The term computer-aided software engineering referring to a set of tools and practices that facilitates the management of a software development project.

While in coding we used html/php and javascript for the websites. And we used phpMyAdmin SQL for the database management

Architectural Design:

a) Architectural Pattern:

The purpose of this part is to analyze the implementation of **client-server architecture** for a food ordering website in a restaurant. Client-server architecture is a widely adopted model for designing web applications. This part will explore the benefits, components, and considerations of utilizing client-server architecture in the context of a food ordering website.

1. Overview of Client-Server Architecture:

Client-server architecture is a network model where client devices (such as web browsers) communicate with a centralized server to request and receive data. In the case of a food ordering website, clients (restaurant patrons) interact with the server to browse menus, place orders, and receive order status updates.

2. Benefits of Client-Server Architecture:

- a. **Scalability:** Client-server architecture allows for scaling the server resources independently of the client devices. This enables the website to handle increased traffic and accommodate a growing number of concurrent users.
- b. **Centralized Data Management:** With a centralized server, data related to menus, orders, inventory, and customer information can be efficiently stored, managed, and updated.
- c. **Enhanced Security:** By centralizing sensitive data and implementing secure authentication protocols, client-server architecture helps protect user information, reduce vulnerabilities, and enforce access control measures.
- d. **Consistent User Experience:** Clients interact with the server to fetch and update data, ensuring a consistent experience across different devices and platforms.

3. Components of the Client-Server Architecture:

a. Client-Side Components:

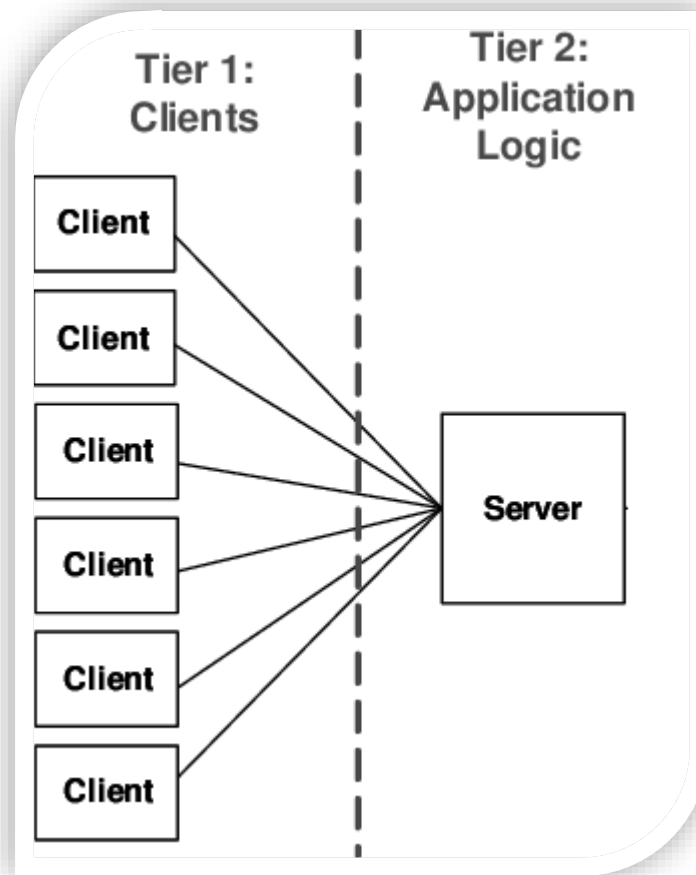
- i. **Web Browser:** Provides the user interface for browsing menus, selecting items, and placing orders.
- ii. **User login:** Allows user to log in with table number, and manage personal Orders.
- iii. **User Interface (UI):** Presents the menu, order history, and other relevant information to the user.

b. Server-Side Components:

- i. **Web Server:** Handles HTTP requests from clients and routes them to appropriate modules or services.
- ii. **Application Server:** Processes client requests, and interacts with databases.
- iii. **Database Management System (DBMS):** Stores and retrieves data related to menus, orders, customers, and inventory.

v. Order Management System: Tracks order statuses, communicates with the kitchen, and updates customers about their orders.

Utilizing client-server architecture for a food ordering website in a restaurant offers numerous benefits, including scalability, centralized data management, enhanced security, and consistent user experience. By carefully considering the various components and implementation considerations outlined in this report, restaurant owners can build a robust and efficient system that provides a seamless online ordering experience for their customers.



b) Application Architecture:

The food industry has witnessed a significant shift towards online platforms, and food ordering websites have become increasingly popular among consumers. To efficiently manage the processes involved in these platforms, transaction processing applications and event processing systems play a crucial role. These systems work in tandem to handle customer orders, process payments, manage inventory, and provide real-time updates and personalized experiences.

Transaction Processing Applications:

✧ Transaction processing applications

- Data-centered applications that process user requests and update information in a system database.

1. **Order Management System:** This application handles the processing of customer orders placed on the food ordering website. It manages the workflow of orders from placement to delivery, including order confirmation, payment processing, and tracking.
2. **Inventory Management System:** This application tracks the availability of food items and ingredients in real-time. It ensures that the website only displays items that are currently in stock and updates the inventory as orders are placed and fulfilled.
3. **Customer Relationship Management (CRM) System:** A CRM system is used to manage customer information, including order history, preferences, and contact details. It helps improve customer service by providing a centralized database of customer interactions and enabling personalized communication.

Event Processing Systems:

✧ Event processing systems

- Applications where system actions depend on interpreting events from the system's environment.

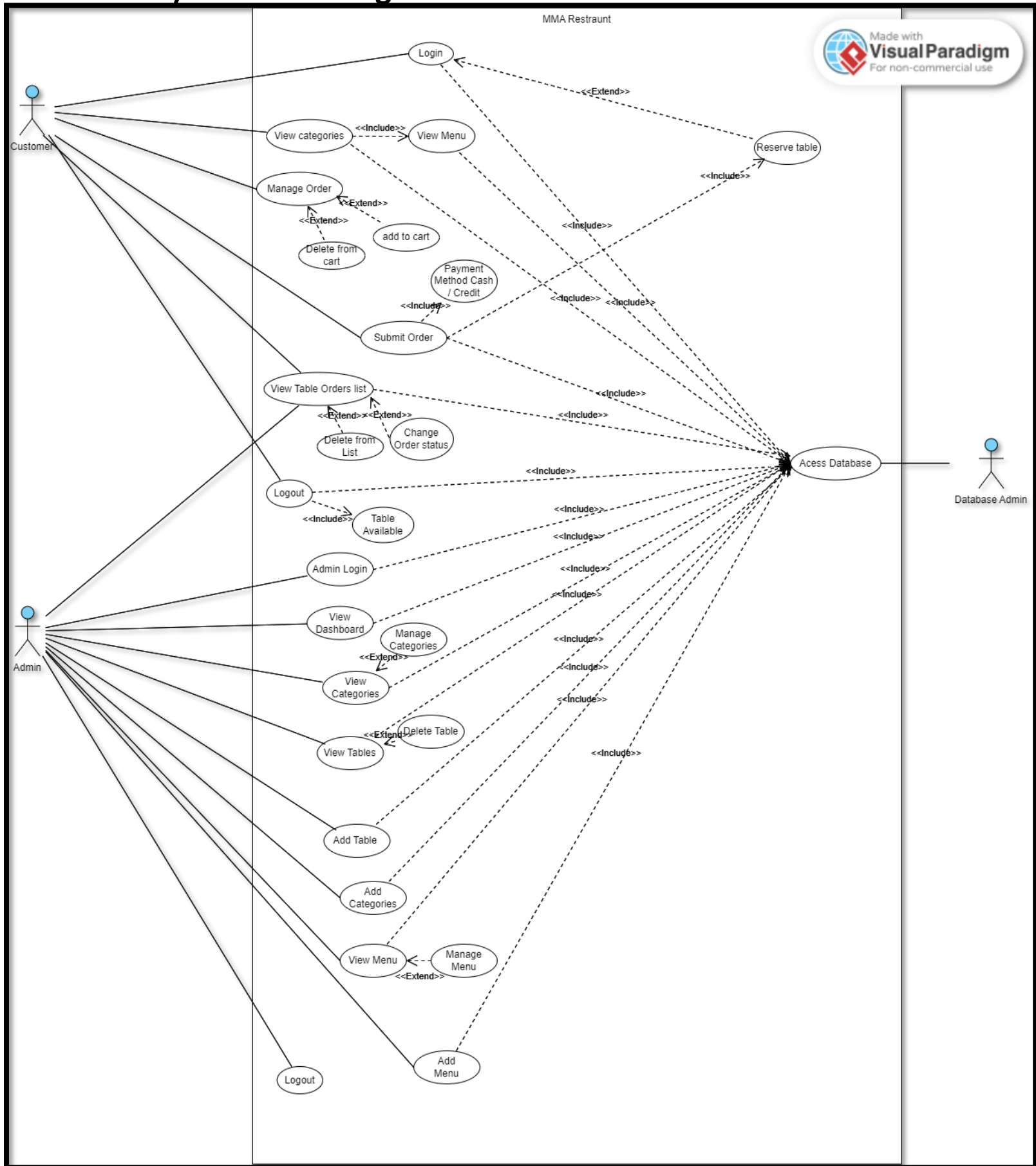
1. **Real-Time Order Tracking:** An event processing system can track the progress of orders in real-time. It captures events such as order placement, preparation, dispatch, and delivery updates. Customers can track their orders and receive notifications at each stage, enhancing transparency and improving the overall customer experience.
2. **Personalized Recommendations:** Event processing can analyze customer behavior, such as past orders and browsing history, to provide recommendations.

.

These transaction processing applications and event processing systems work together to ensure smooth and efficient operations for a food ordering website, handling customer orders, managing inventory, processing payments, and providing real-time updates and personalized experiences.

System Modeling:

1) Use Case Diagram:



The "Customer" actor

Login: The "Customer" can log in to the restaurant website by providing the table number which he is being served at

View Categories: Once logged in, the "Customer" can view different categories available in the restaurant's menu, such as appetizers, main courses, desserts, etc. This use case enables the "Customer" to browse through the menu.

View Menu: Within each category, the "Customer" can select a specific category to view the items available. This use case allows the "Customer" to see the details of each menu item, such as description, price

Manage Orders: The "Customer" can manage their orders, including adding other items to the order, or removing items from the order. This use case enables the "Customer" to customize their order according to their preferences.

Submit Order: Once the "Customer" has selected the desired items and made necessary modifications, they can submit their order. This use case sends the order details to the restaurant system for processing. the "Customer" can choose to pay either by cash or credit. This use case allows the "Customer" to indicate their preferred payment method.

View Order Lists: After placing an order, the "Customer" can view their list of previous ordered food. This use case provides access to order details, including order status,

Log Out: The "Customer" can log out from the restaurant website. This use case terminates the session.

The "Admin" actor

Login: The "Admin" can log in to the restaurant website by providing their credentials, such as username and password. This allows them to access administrative features and privileges.

View Categories: Once logged in, the "Admin" can view the existing categories in the restaurant's menu. This use case allows the "Admin" to see the list of available categories.

Modify Categories: The "Admin" can modify the categories in the restaurant's menu, such as adding new categories, updating existing categories, or deleting categories. This use case enables the "Admin" to manage and customize the categories according to the restaurant's needs.

View Menus: Within each category, the "Admin" can view the menus available. This use case allows the "Admin" to see the details of each menu item, such as description, price.

View Tables: By which the Admin can view the number of tables which is available in the restaurant and can also modify the table by adding or deleting a table from the restaurant

Modify Menus: The "Admin" can modify the menus in the restaurant's menu, such as adding new menu items, updating existing menu items, or deleting menu items. This use case enables the "Admin" to manage and customize the menus according to the restaurant's offerings.

View Orders: The "Admin" can view all the orders made by customers. This use case provides access to order details, including customer information, items ordered, order status, and delivery information.

Log Out: The "Admin" can log out from the restaurant website.

The "Database" actor

Login: Both the "User" and "Customer" can log in to the system by providing their credentials. This use case allows them to access personalized features and information.

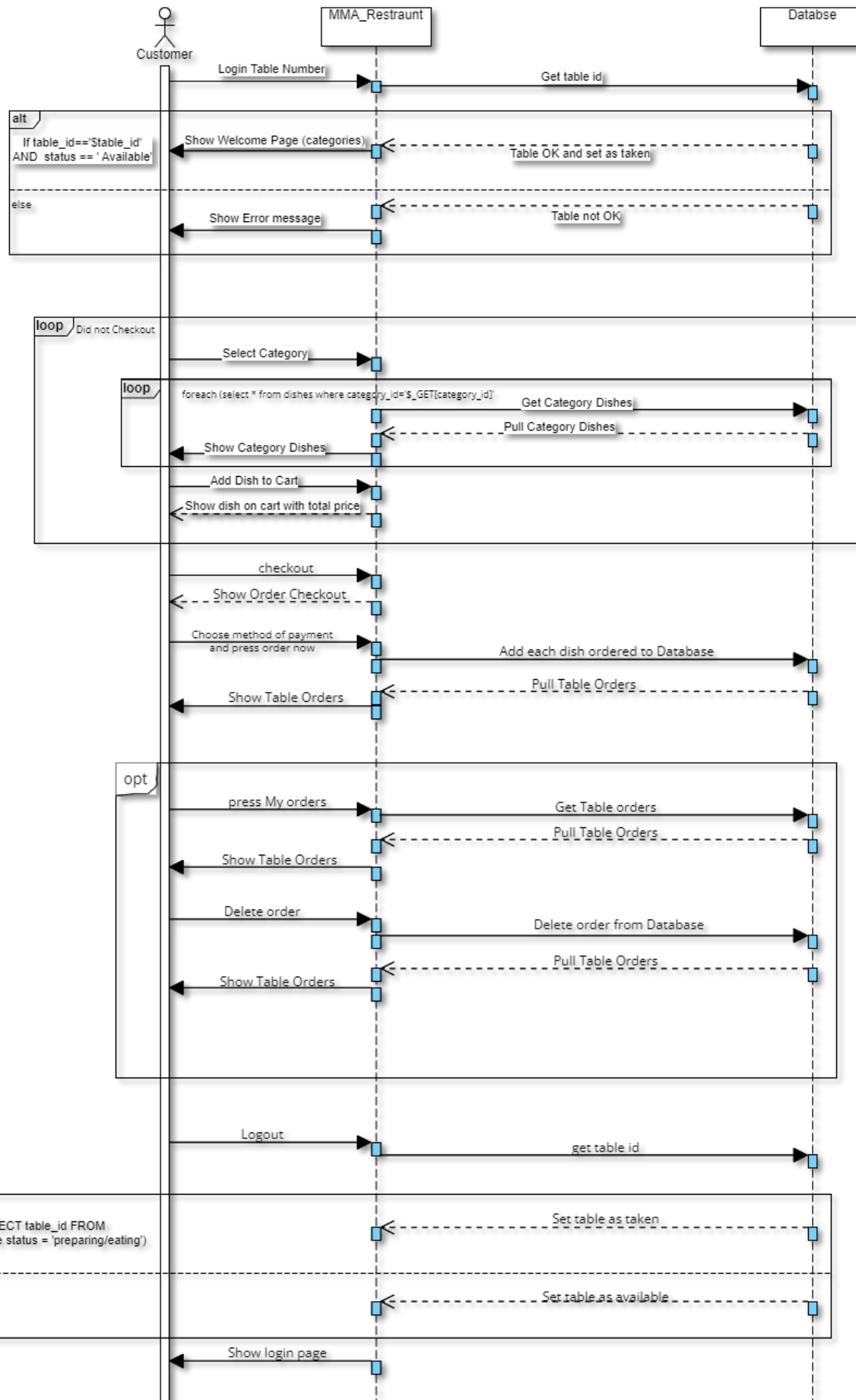
View Data: The "User" and "Customer" can view data from the database. This use case enables them to retrieve information, such as user profiles, product details, or order history, from the database.

Update Data: The "User" and "Customer" can update data in the database. This use case allows them to modify existing information, such as updating their profile details, adding or removing items from an order, or changing personal preferences.

Submit Data: The "User" and "Customer" can submit new data to the database. This use case includes actions like creating, placing a new order. Adding new category by user etc .

Sequence Diagram:

1) User Sequence Diagram



The **sequence diagram** represents the interactions and sequence of messages between the customer, the restaurant system, and the database in a restaurant website. It illustrates the flow of events starting from the customer logging in until they log out and set the table as available.

The sequence diagram includes the following interactions:

Customer Login: The customer initiates the login process by providing their credentials. The system verifies the credentials and checks if the table is registered or not. If the table is not registered, an error message is shown to the customer.

Selection Loop: Once the customer is logged in and the table is registered, a loop begins where the customer selects different categories and adds dishes from the menu of the selected category. After each selection, the system updates the customer's choices in the database to reflect the changes made.

Checkout and Payment: When the customer decides to checkout, they choose the payment option they prefer. The system manages the order and the payment process accordingly. The order details are saved in the database.

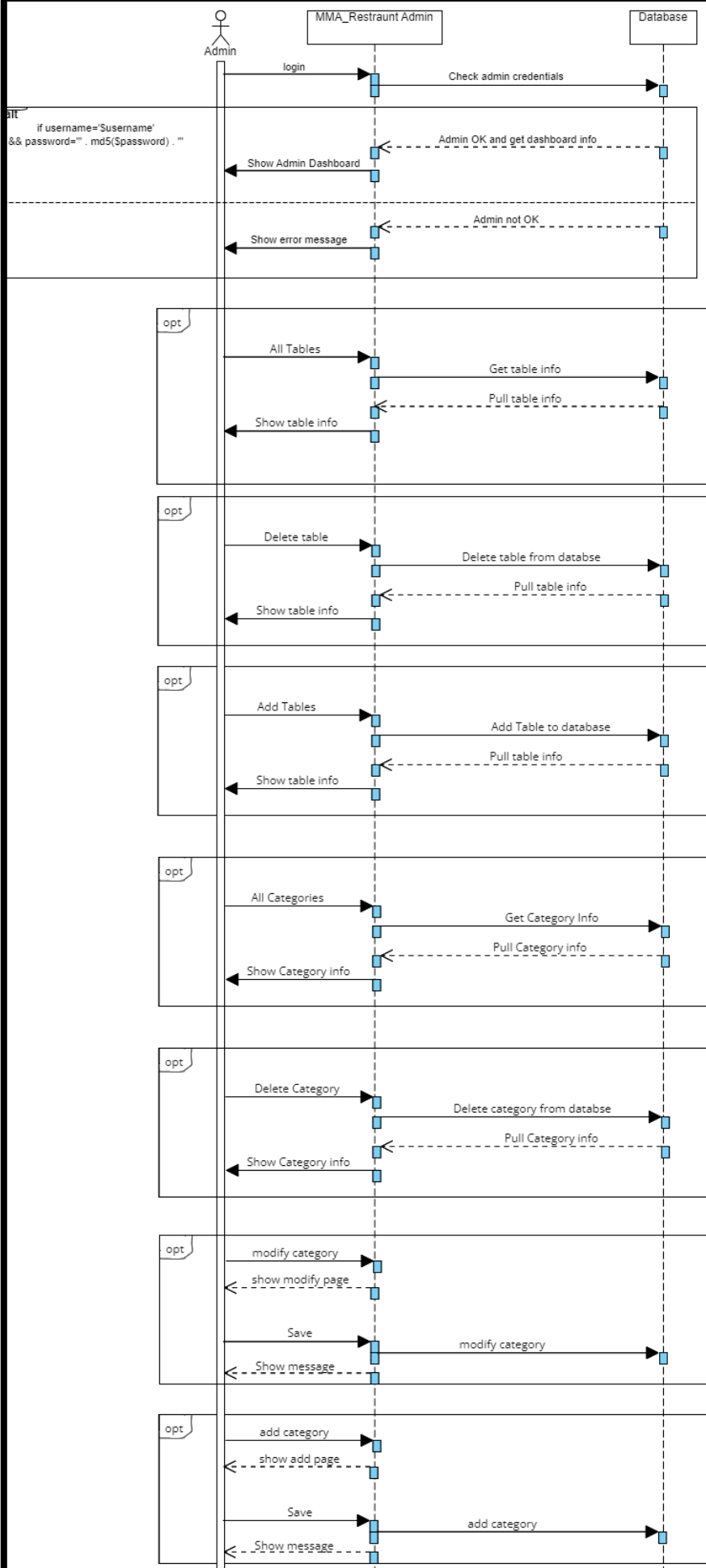
Order Submission: The customer submits the order to the system. The system processes the order and saves the necessary information in the database

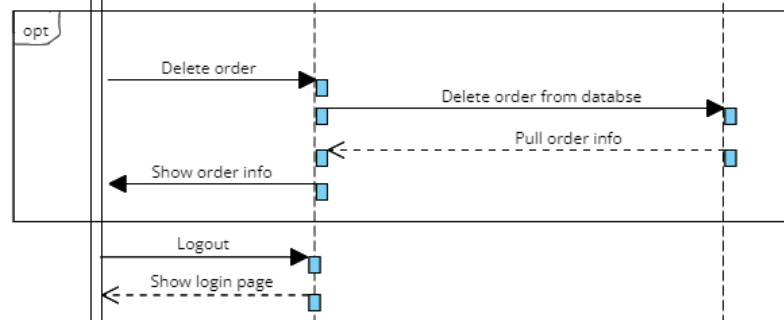
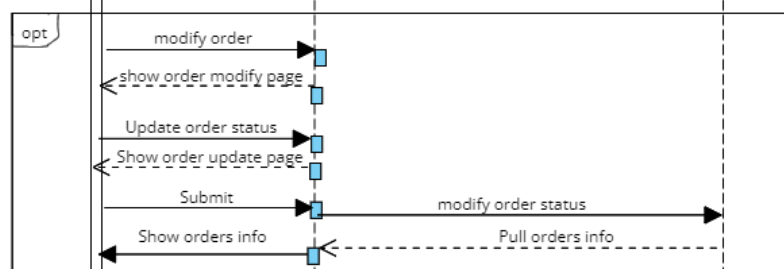
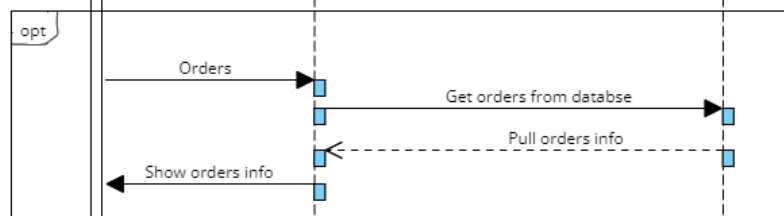
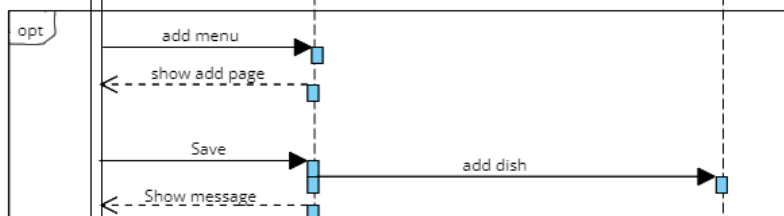
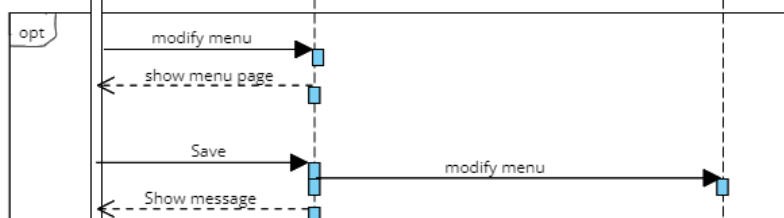
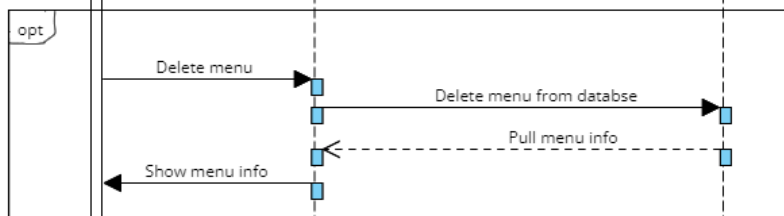
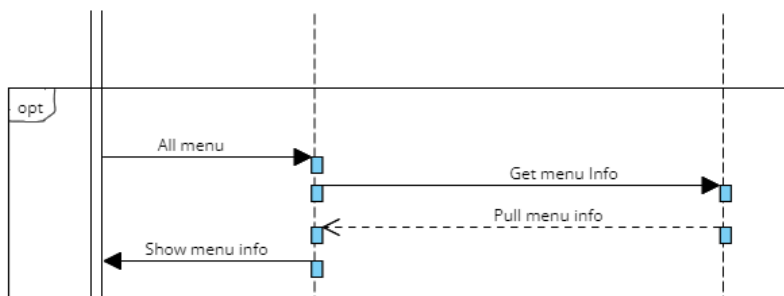
Order Management: After the customer has submitted the order, they have the option to manage their orders. This includes viewing order lists, deleting an order (if it is still possible), or performing other actions related to the orders. The system retrieves the order information from the database to display to the customer and updates the database if any changes are made.

View Order Lists: After submitting the order, the customer can view their order lists to review the details and status of their orders. The system retrieves the order information from the database and presents it to the customer.

Log Out: Finally, when the customer decides to leave the restaurant, they log out from the system. This action sets the table as available for new customers. The system updates the database to reflect the change in table status.

Admin Sequence Diagram





The **Admin sequence** diagram includes the following interactions:

Admin Login: The Admin initiates the login process by providing their credentials. The database system verifies the credentials, and if incorrect credentials are entered, an error message is shown to the Admin.

Admin Options: Once the Admin is successfully logged in, they have multiple optional actions they can choose from. These options may include viewing categories, adding new categories, deleting existing categories, viewing dishes, adding new dishes, deleting existing dishes, viewing tables, or deleting tables. The Admin can choose any combination of these actions.

View Categories/Dishes/Tables: If the Admin chooses to view categories, dishes, or tables, the system retrieves the relevant information from the database and presents it to the Admin.

Add Category/Dish/Table: If the Admin chooses to add a new category, dish, or table, they provide the necessary information, and the system processes the request, updating the relevant data in the database.

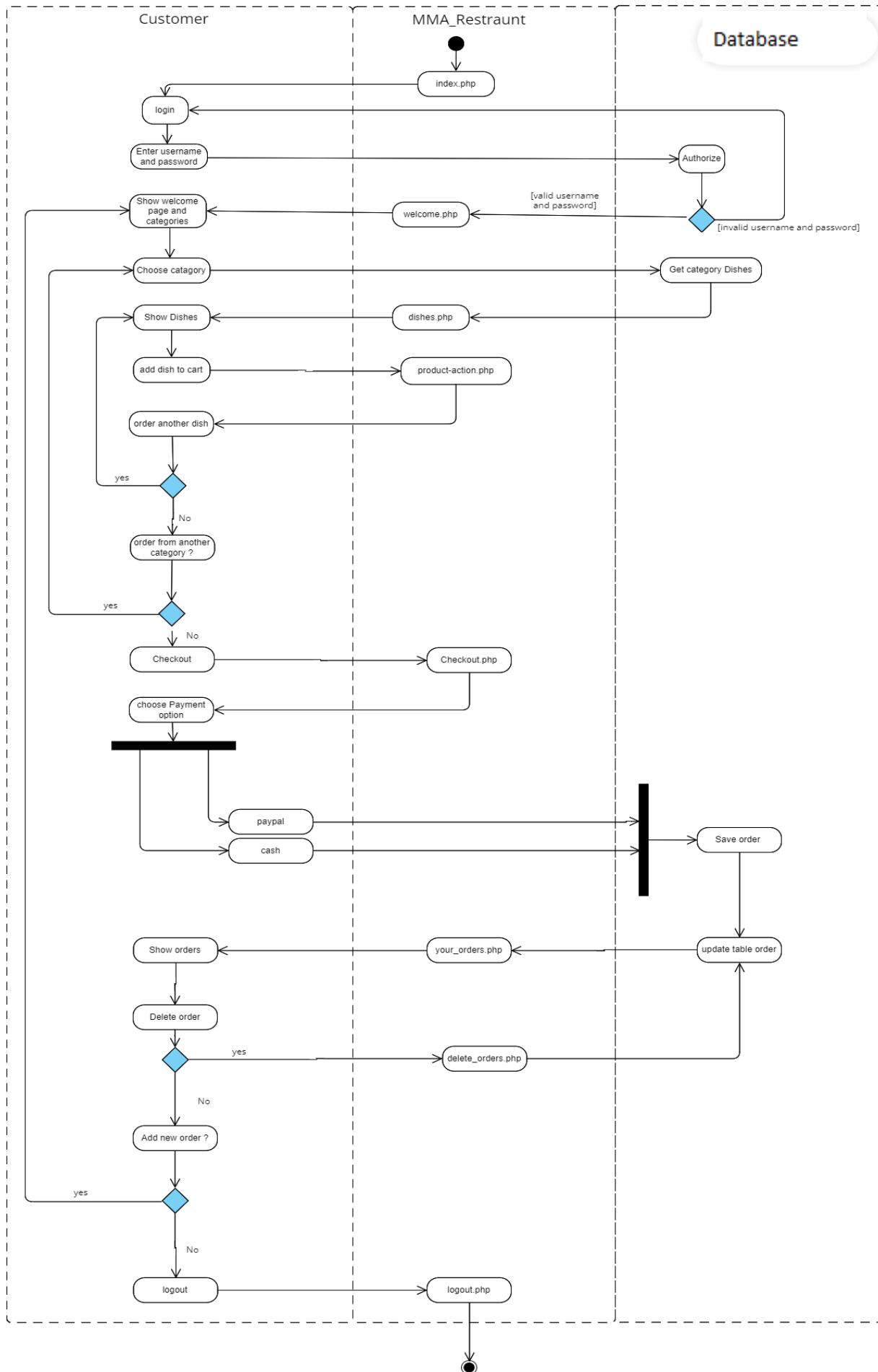
Delete Category/Dish/Table: If the Admin chooses to delete an existing category, dish, or table, they select the item to be deleted, and the system removes the corresponding data from the database.

View Order Lists: The Admin can view the order lists to check the details and status of the orders placed by customers. The system retrieves the order information from the database and presents it to the Admin.

Modify Order Status: If needed, the Admin can modify the status of orders. This may involve updating the status to "in progress," "completed," or any other relevant status. The system updates the order status in the database accordingly.

Log Out: Finally, when the Admin is done managing the restaurant, they log out from the system. This action terminates the session and ensures the security of the Admin's account.

Activity Diagram: 1) Customer Activity Diagram



The activity diagram represents the flow of activities performed by the customer in a restaurant website. It illustrates the steps starting from the customer logging in until they log out and set the table as available for new customers.

The activity diagram includes the following activities:

Customer Login: The customer initiates the login process by providing their table number. The system verifies the credentials in the database if available, and if the table is registered, an error message is shown to the customer.

Category Selection Loop: Once the customer is logged in and the table is registered, they enter a loop to select different categories and add dishes from the menu of the selected category. This loop allows the customer to continue selecting and adding dishes until they want to proceed to checkout.

Checkout and Payment: When the customer decides to checkout, they choose the payment option they prefer. This activity involves managing the order, selecting the payment method..

Order Management: After the customer has submitted the order, they have the option to manage their orders. This activity includes viewing order lists, reviewing the details and status of orders, and potentially deleting an order if it is still possible.

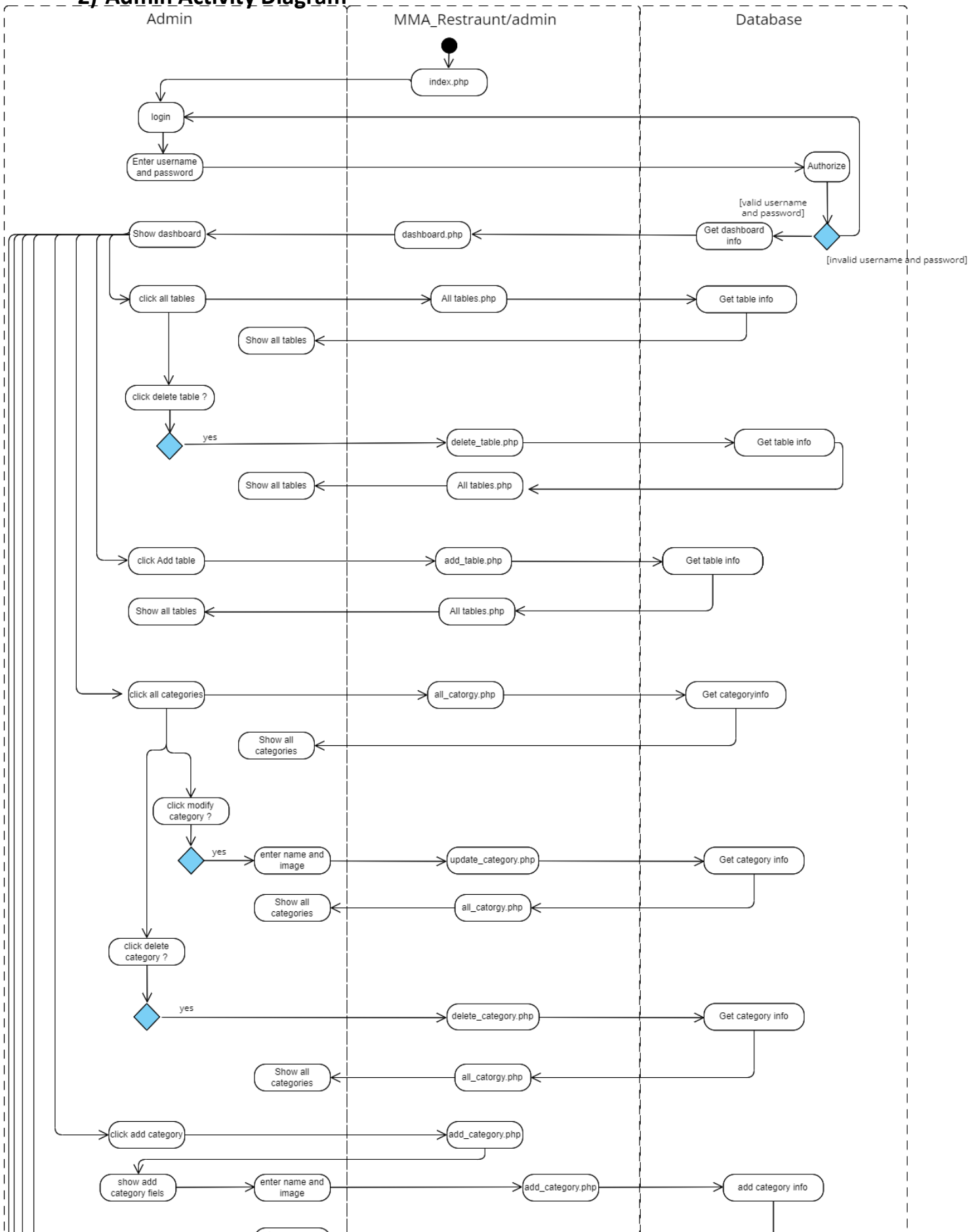
Submit Order: The customer submits the order to the system. This activity involves finalizing the order details and sending the order for processing.

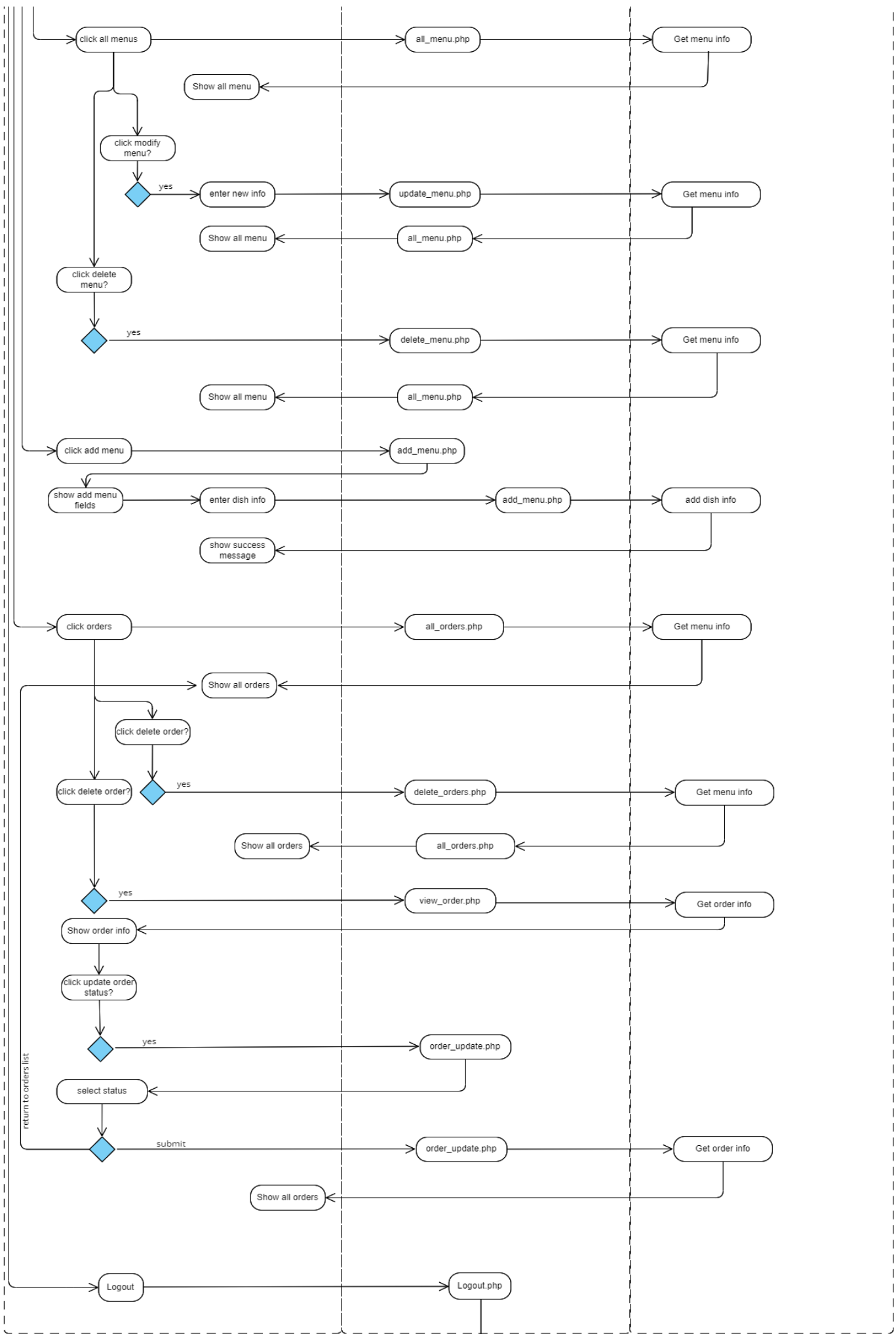
View Order Lists: After submitting the order, the customer can view their order lists to review the details and status of their orders.

Optional Order Deletion: The customer may choose to delete an order after it has been placed. This activity allows them to remove an order if it is still cancellable.

Log Out: Finally, when the customer is ready to leave the restaurant, they log out from the system. This action sets the table as available for new customers.

2) Admin Activity Diagram





Admin Login: The Admin initiates the login process by providing their credentials. The system verifies the credentials from the database, and if incorrect credentials are entered, an error message is shown to the Admin.

Admin Options: Once the Admin is successfully logged in, they have multiple optional actions they can choose from. These options may include viewing categories, adding new categories, deleting existing categories, viewing dishes, adding new dishes, deleting existing dishes, viewing tables, or deleting tables. The Admin can choose any combination of these actions.

View Categories/Dishes/Tables: If the Admin chooses to view categories, dishes, or tables, the system retrieves the relevant information from the database and presents it to the Admin.

Add Category/Dish/Table: If the Admin chooses to add a new category, dish, or table, they provide the necessary information, and the system processes the request, updating the relevant data in the database.

Database Update: Throughout the activities performed by the Admin, such as viewing, adding, or deleting categories, dishes, or tables, the system updates the relevant data in the database to reflect the changes made by the Admin.

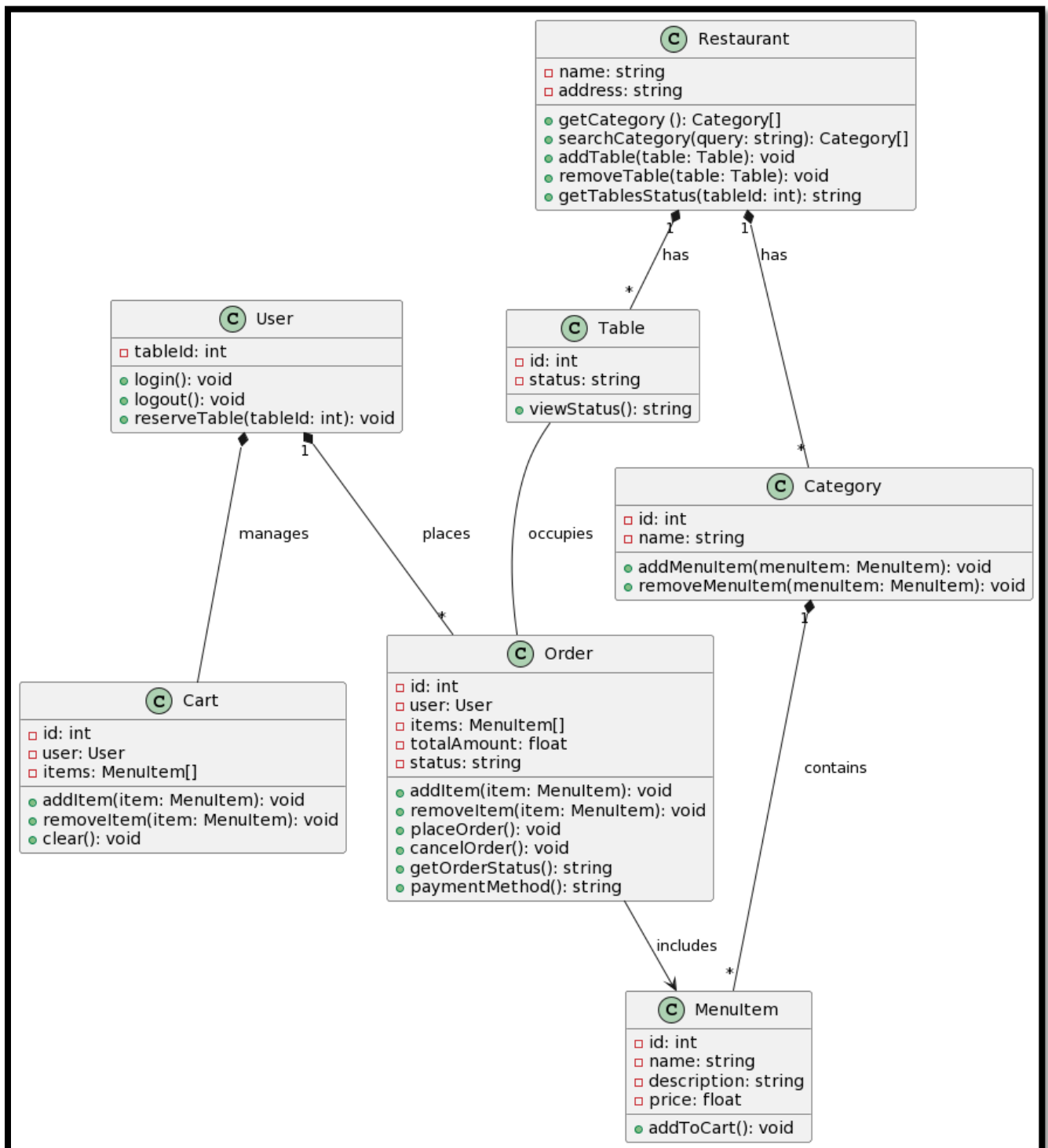
Delete Category/Dish/Table: If the Admin chooses to delete an existing category, dish, or table, they select the item to be deleted, and the system removes the corresponding data from the database.

View Order Lists: The Admin can view the order lists to check the details and status of the orders placed by customers. The system retrieves the order information from the database and presents it to the Admin.

Modify Order Status: If needed, the Admin can modify the status of orders. This may involve updating the status to "in progress," "completed," or any other relevant status. The system updates the order status in the database accordingly.

Log Out: Finally, when the Admin is done managing the restaurant, they log out from the system.

Class diagram:

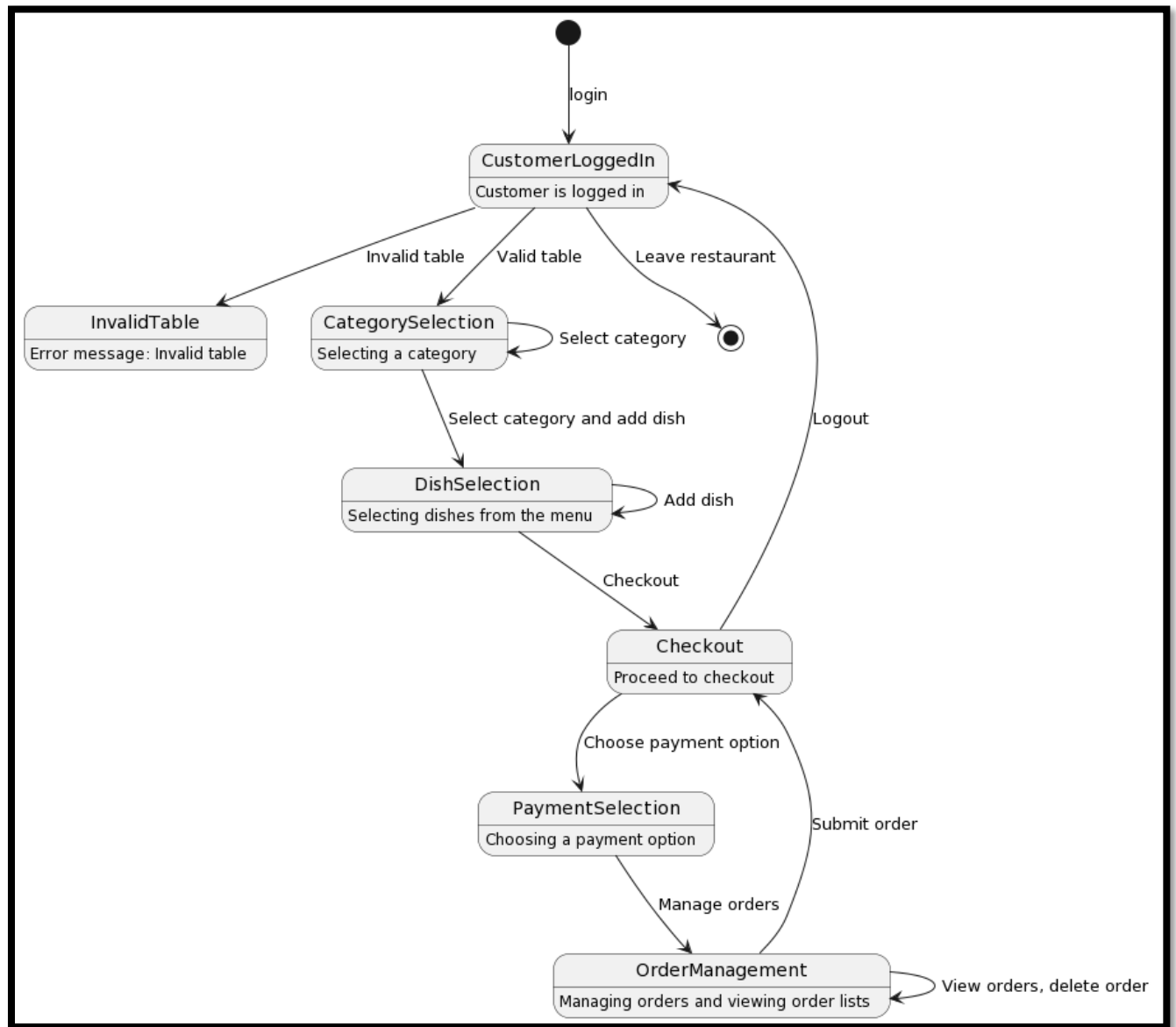


Class diagram

- **User:** Represents a user of the system who can log in, log out, and reserve tables. The "tableId" attribute indicates that a user can reserve a table by specifying its ID. The "login" and "logout" methods allow a user to log in or log out of the system. The "reserveTable" method allows a user to reserve a table by specifying its ID.
- **Restaurant:** Represents a restaurant with a name, address, and various functionalities like retrieving categories, searching categories, adding/removing tables, and getting table status. The "getCategory" method retrieves a list of all the categories in the restaurant's menu. The "searchCategory" method allows a user to search for categories by specifying a query string. The "addTable" and "removeTable" methods allow the restaurant to add or remove tables. The "getTablesStatus" method returns the status of a particular table specified by its ID.
- **MenuItem:** Represents an item on the restaurant's menu with an ID, name, description, and price. The "addToCart" method allows a user to add the item to their cart.
- **Order:** Represents an order placed by a user, containing items, total amount, and status. The "addItem" and "removeItem" methods allow a user to modify the items in their order. The "placeOrder" method allows a user to place the order. The "cancelOrder" method allows a user to cancel the order. The "getOrderStatus" method returns the status of the order (e.g. "pending", "in progress", "delivered"). The "paymentMethod" method returns the payment method used for the order (e.g. "credit card", "cash").
- **Cart:** Represents a user's cart, which contains items that they have added. The "addItem" and "removeItem" methods allow a user to add or remove items from their cart. The "clear" method removes all items from the cart.
- **Category:** Represents a category of items in the restaurant's menu. The "addItem" and "removeMenuItem" methods allow the restaurant to add or remove items from the category.

- **Table**: Represents a table in the restaurant with an ID and status. The "viewStatus" method returns the current status of the table (e.g. "available", "occupied", "reserved").
- **User** manages **Cart**: Indicates that a user can manage their cart, adding and removing items.
- **User** places **Order**: Indicates that a user can place an order by creating an order object.
- **Restaurant** has **Category**: Indicates that a restaurant can have multiple categories.
- **Order** includes **MenuItem**: Indicates that an order can include multiple menu items.
- **Category** contains **MenuItem**: Indicates that a category can contain multiple menu items.
- **Restaurant** has **Table**: Indicates that a restaurant can have multiple tables.
- **Table** occupies **Order**: Indicates that a table can be associated with an order (occupied).

State diagram for User



The provided state diagram represents the state transitions and interactions of a restaurant ordering system. It outlines the various states and actions that occur when a customer interacts with the system, from logging in to placing an order and managing it. This report aims to explain the different states and transitions in the state diagram.

State Diagram Analysis:

The state diagram consists of several states and transitions that depict the flow of actions within the restaurant ordering system. Let's examine each state and transition in detail:

1. **CustomerLoggedIn:**

This state represents the initial state when a customer successfully logs into the system. From this state, two possible transitions can occur:

- **InvalidTable:** If the customer provides an invalid table number, an error message is displayed.
- **CategorySelection:** If the customer provides a valid table number, they proceed to select a category from the menu.

2. **InvalidTable:**

This state signifies an error condition where the customer has entered an invalid table number. The system displays an error message, and the customer is prompted to re-enter the table number.

3. **CategorySelection:**

Upon reaching this state, the customer has successfully entered a valid table number and can now select a category from the menu. The customer has the option to choose a category or continue browsing the menu.

4. **DishSelection:**

Once a category is selected, the customer enters the DishSelection state. Here, they can add dishes to their order. The customer has the ability to select multiple dishes from the menu and add them to their order.

5. **Checkout:**

After selecting the desired dishes, the customer proceeds to the Checkout state. In this state, the customer can review their order and proceed to the payment process. Additionally, the customer has the option to remove or modify the selected dishes before finalizing the order.

6. PaymentSelection:

Once the customer is ready to proceed with payment, they enter the PaymentSelection state. Here, they can choose their preferred payment option, such as cash, credit card, or mobile payment.

7. OrderManagement:

After completing the payment, the customer enters the OrderManagement state. This state allows the customer to manage their orders. They can view their order history, delete an order if needed, or perform other actions related to order management.

8. Checkout (Loop):

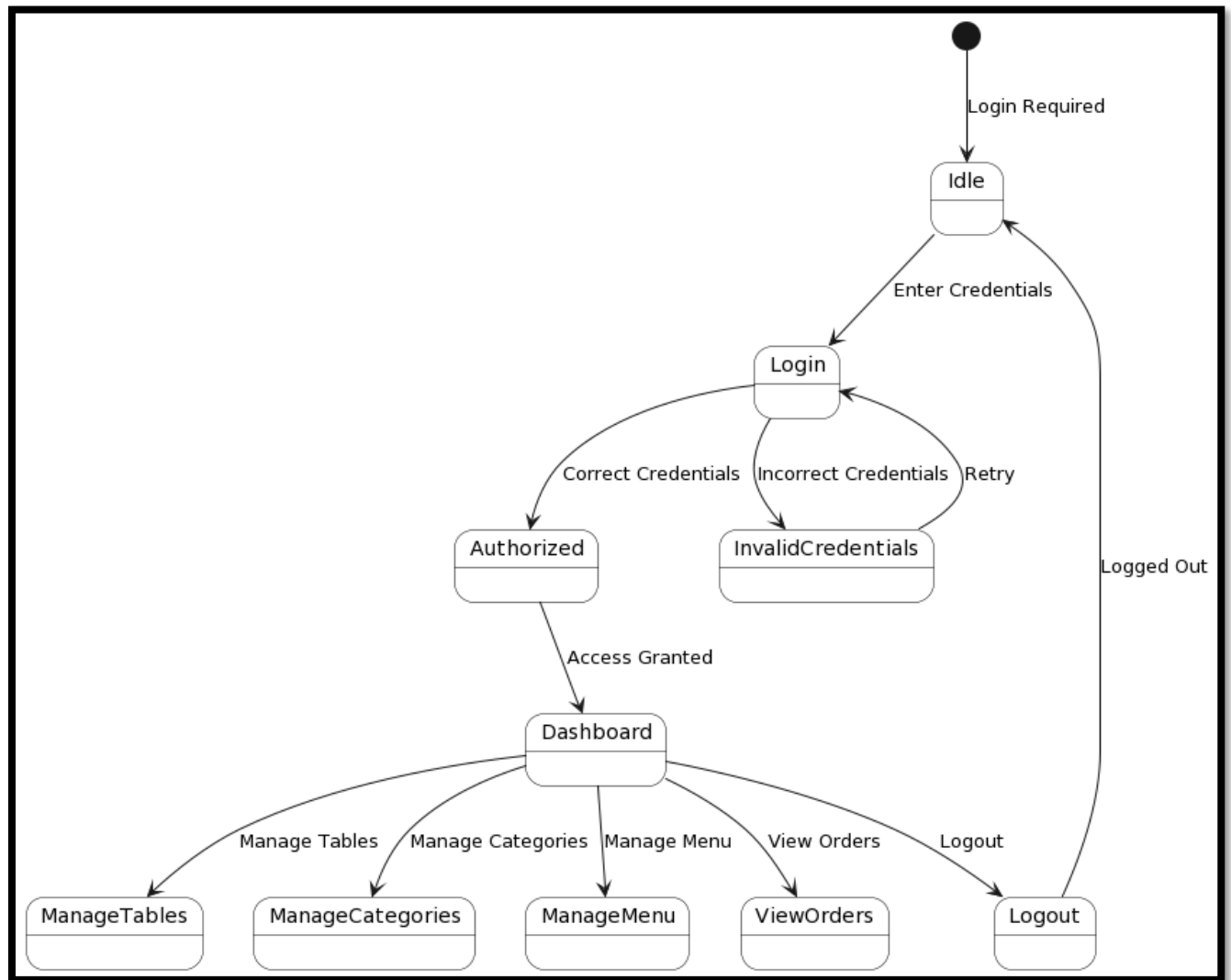
From the OrderManagement state, the customer can either return to the Checkout state to place another order or proceed to the CustomerLoggedIn state to log out.

9. CustomerLoggedIn (Loop):

Upon logging out, the customer returns to the initial state, where they have the option to log in again or leave the restaurant.

The provided state diagram outlines the flow of actions and states within a restaurant ordering system. It begins with a customer logging into the system, followed by selecting a table, choosing a category, selecting dishes, proceeding to checkout, choosing a payment option, managing orders, and eventually logging out. The diagram provides a clear visualization of the system's behavior and the various states and transitions involved in the ordering process.

State diagram for Admin



The provided code represents a state diagram depicting the various states and transitions involved in a login and dashboard system. The diagram visualizes the flow of actions and conditions that occur during the interaction between a user and the system.

1. Initial State:

The diagram begins with the initial state represented by [*]. This state indicates that the user is not logged in and a login is required to access the system.

2. Idle State:

From the initial state, the system transitions to the "Idle" state. In this state, the user is prompted to enter their credentials to log in.

3. Login State:

Upon entering the credentials, the system moves to the "Login" state. Here, the entered credentials are verified against some stored data, such as a username and password combination.

4. Authorized State:

If the entered credentials are correct, the system transitions to the "Authorized" state. This state represents that the user has successfully logged in, and their access is granted.

5. InvalidCredentials State:

In case the entered credentials are incorrect, the system moves to the "InvalidCredentials" state. In this state, the user is informed about the incorrect credentials and prompted to retry entering the correct ones.

6. Retry State:

From the "InvalidCredentials" state, the system transitions back to the "Login" state, allowing the user to retry entering the credentials.

7. Dashboard State:

Once the user's credentials are verified and they are granted access, the system moves to the "Dashboard" state. This state represents the main interface of the system, where the user can perform various actions.

8. ManageTables State:

From the "Dashboard" state, the user can transition to the "ManageTables" state. This state allows the user to manage tables, which may include adding, modifying, or removing tables from the system.

9. ManageCategories State:

Similarly, from the "Dashboard" state, the user can move to the "ManageCategories" state. Here, the user can manage categories, which could involve tasks like adding, editing, or deleting categories associated with the system.

10. ManageMenu State:

The "ManageMenu" state is another option from the "Dashboard" state. In this state, the user has the ability to manage the menu, such as adding, modifying, or removing items from the menu list.

11. ViewOrders State:

From the "Dashboard" state, the user can transition to the "ViewOrders" state. Here, the user can view the current orders placed by customers or perform actions related to managing orders.

12. Logout State:

Finally, from any state within the system, the user can choose to log out. The system moves to the "Logout" state, and the user is successfully logged out.

13. Transition to Idle:

From the "Logout" state, the system transitions back to the "Idle" state. This completes the cycle and brings the system back to the initial state, where a login is required for further access.

In conclusion, the state diagram provides a clear visualization of the login and dashboard system. It illustrates the progression from the initial state to login, authorization, and subsequent actions that can be performed within the dashboard. The diagram helps to understand the different states and transitions involved, facilitating the development and understanding of the system's functionality.