RESTAURANT MANAGEMENT SYSTEM

1. Introduction

1.1 Purpose

The purpose of the Restaurant Management System is to streamline the operations of a restaurant, providing functionalities for customers to view the menu, place orders, generate receipts, give reviews, and for administrators to manage transactions, ratings, and revenue.

1.2 Scope

The system will handle customer interactions, menu management, order processing, and administrative tasks related to transaction history, ratings, and revenue tracking.

2. User Types

2.1 Customer

Actions:

- Log in as a customer.
- Input name.
- View dashboard with options:
- See the menu.
- Generate receipt (after ordering).
- Give a review (after generating receipt).
- Exit the program.
- Choose menu type (starters, main course, dessert) or go back.
- View menu items.
- Order items.
- Repeat ordering.
- Generate receipt.
- Give a review.

2.2 Admin

Actions:

- Log in as an admin.
- View admin dashboard with options:
- See transactions for any date.
- See average ratings and feedback.
- See yearly and monthly revenue.
- Exit the program.

3. Functional Requirements

3.1 Customer Functionality

3.1.1 Login as a Customer:

- The system should prompt the user whether they are a customer or admin.
- If a customer, the system should ask for the customer's name.

3.1.2 Customer Dashboard:

- Display options for the customer: "See the menu," "Generate receipt," "Give a review," "Exit."

3.1.3 Menu Viewing:

- Display options to view starters, main course, and dessert menus.
- Allow the customer to go back to the previous dashboard.

3.1.4 Ordering:

- Allow the customer to choose menu items and order them.
- Allow the customer to repeat ordering multiple times.
- Validate the order against the available menu.

3.1.5 Generating Receipt:

- Allow the customer to generate a receipt after ordering.

3.1.6 Giving Review:

- Allow the customer to give a review after generating the receipt.

3.1.7 Exiting the Program:

- Allow the customer to exit the program.

3.2 Admin Functionality

3.2.1 Login as an Admin:

- The system should prompt the user for a username and password.
- Authenticate the admin based on the provided credentials.

3.2.2 Admin Dashboard:

- Display options for the admin: "See transactions," "See average ratings and feedback," "See yearly and monthly revenue," "Exit."

3.2.3 Transaction History:

- Allow the admin to see transactions for any date.

3.2.4 Average Ratings and Feedback:

- Allow the admin to see average ratings and feedback received by the restaurant.

3.2.5 Yearly and Monthly Revenue:

- Allow the admin to see the yearly and monthly revenue earned by the restaurant.

3.2.6 Exiting the Program:

- Allow the admin to exit the program.

4. Non-functional Requirements

4.1 Authentication:

- Secure authentication for admin login using a username and password.

4.2 Usability:

- The system should be user-friendly with clear prompts and menus.

4.3 Data Validation:

- Validate user input to ensure data consistency.

4.4 Data Security:

- Customer and admin data should be stored securely.

5. Assumptions and Constraints

5.1 Assumptions:

- Menu items and prices are predefined.
- The system will be run on a single machine.

5.2 Constraints:

- Limited to console-based interaction.
- No online payment features.

6. Testing Requirements

6.1 Unit Testing:

- Test each functionality separately, including customer and admin interactions.

6.2 Integration Testing:

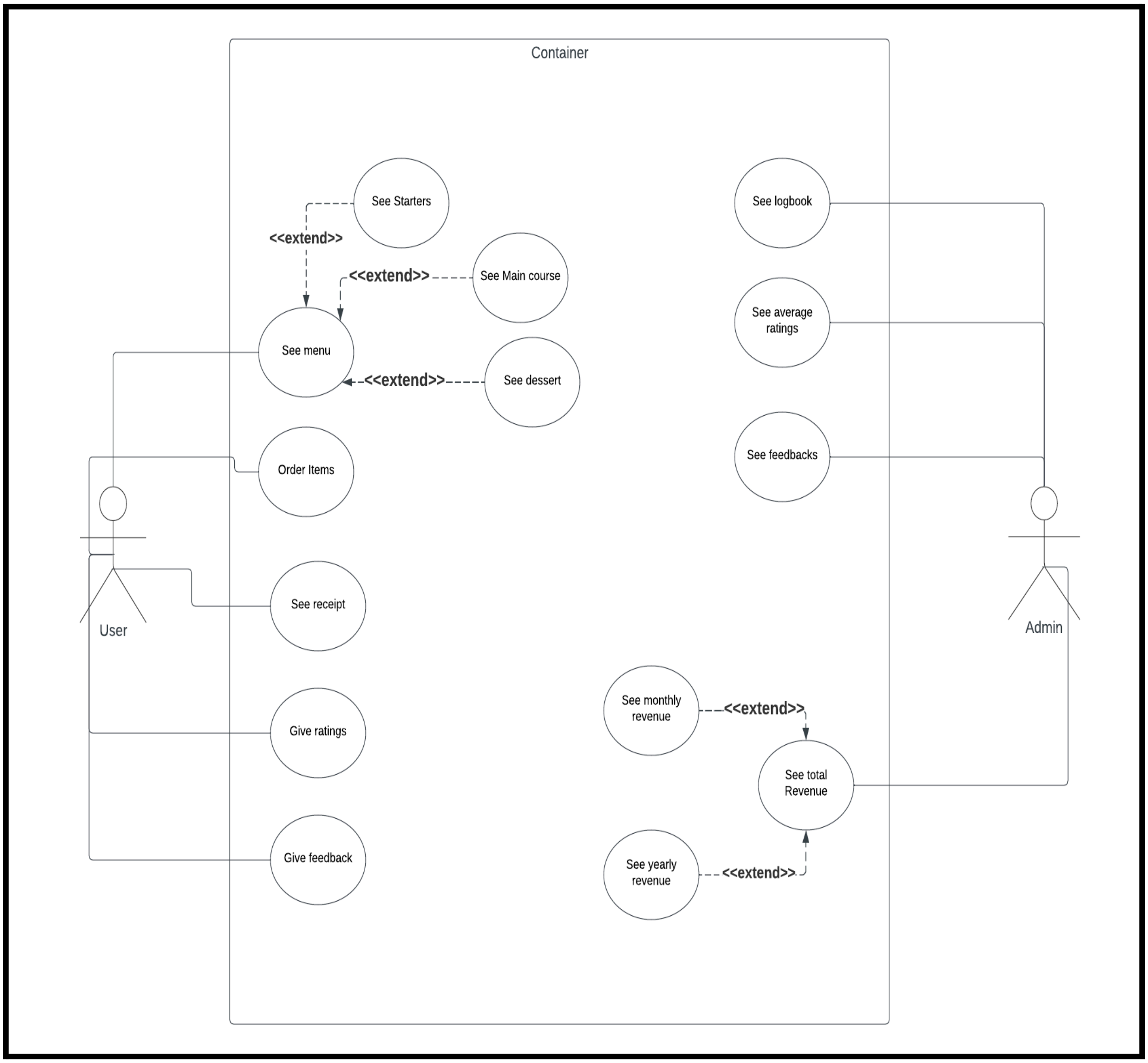
- Test the integration of customer and admin functionalities.

6.3 User Acceptance Testing:

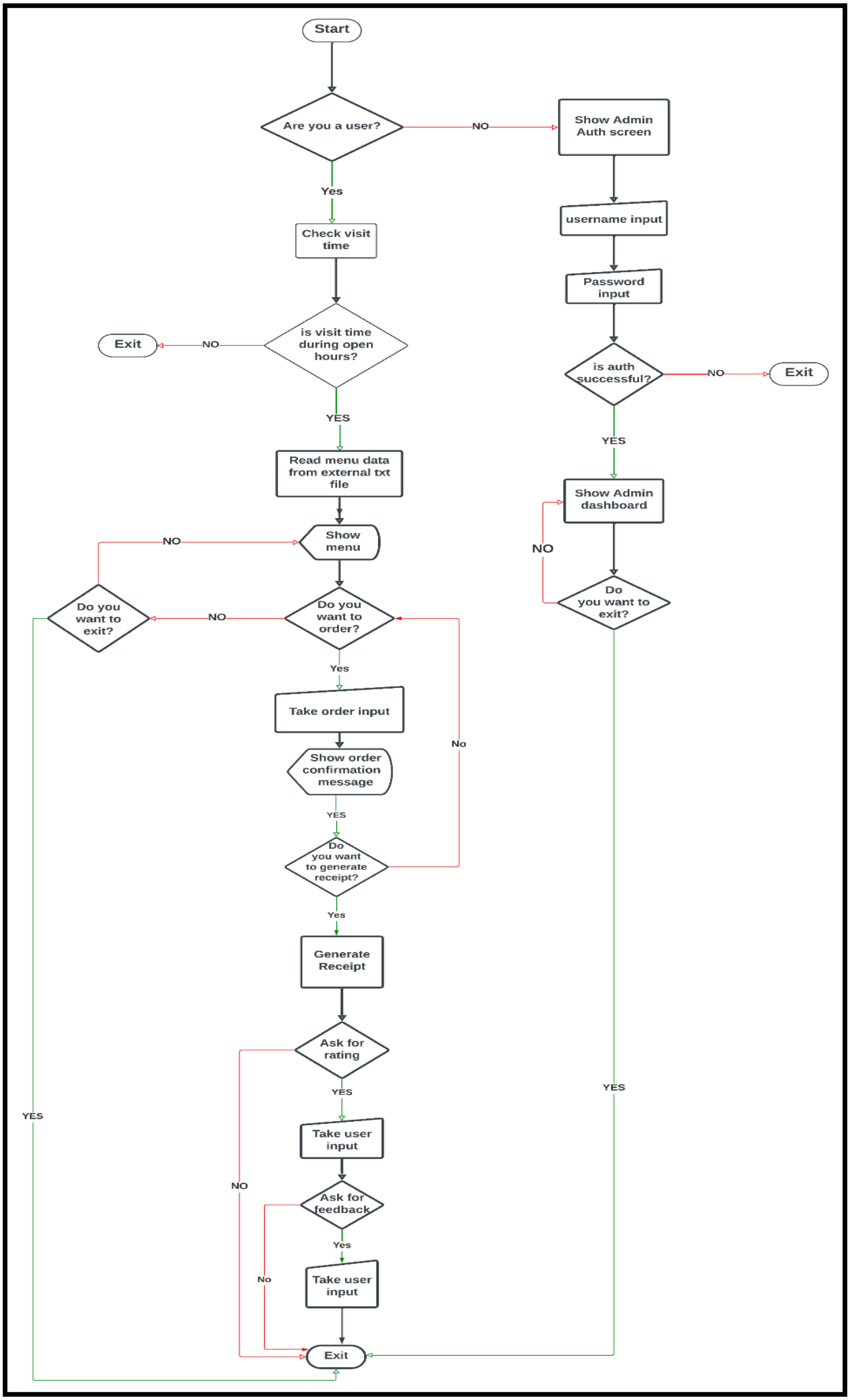
- Obtain feedback from users to validate that the system meets their expectations.

7. UML Diagrams

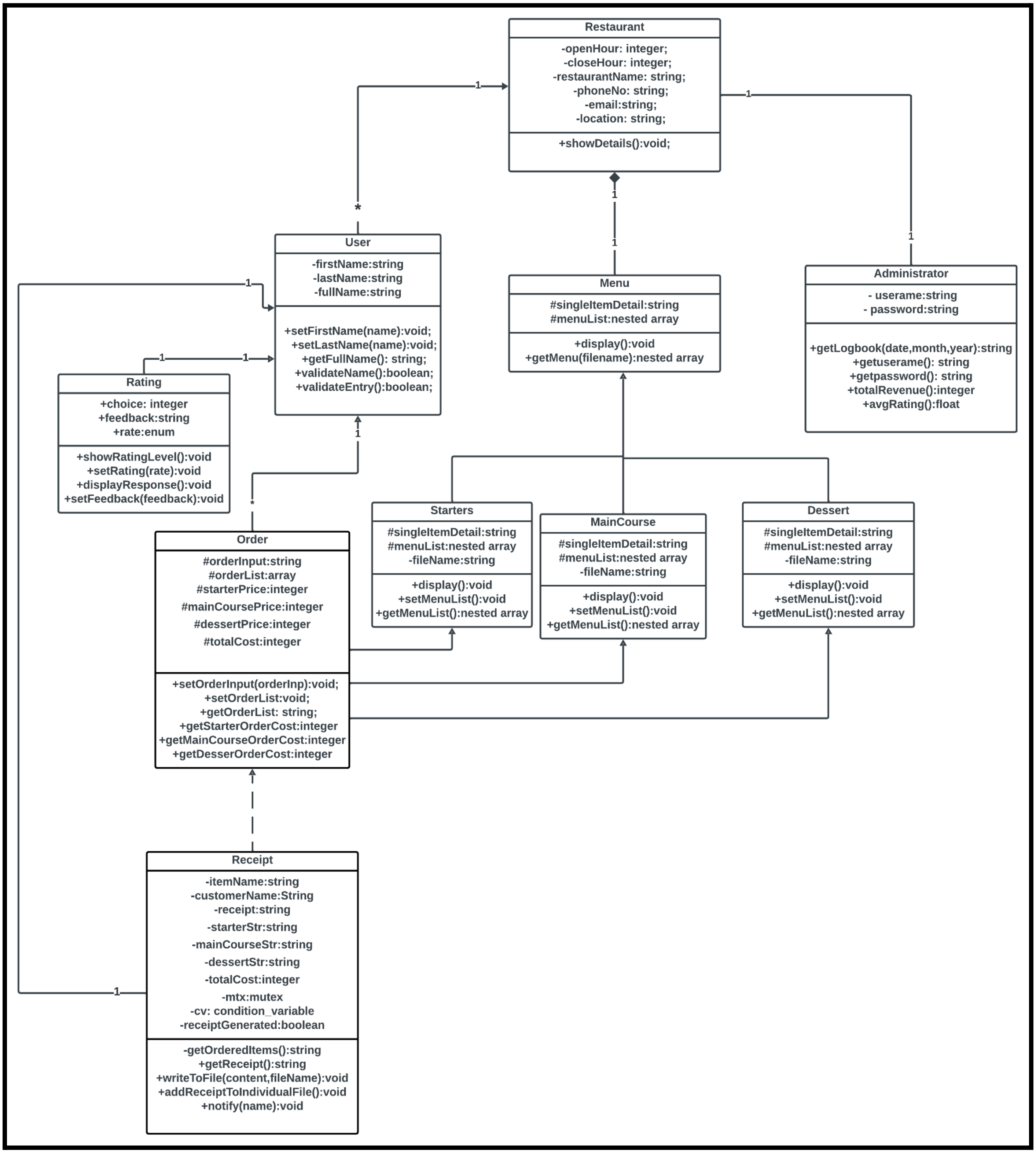
7.1 Usecase Diagram



7.2 Flowchart Diagram



7.3 Class Diagram


Restaurant
-openHour: integer;
-closeHour: integer;
-restaurantName: string;
-phoneNo: string;
-email:string;
-location: string;
+showDetails():void;
1
1
*
1
1
1
1
User
-firstName:string
-lastName:string
-fullName:string
+setFirstName(name):void;
+setLastName(name):void;
+getFullName(): string;
+validateName():boolean;
+validateEntry():boolean;
Menu
#singleItemDetail:string
#menuList:nested array
+display():void
+getMenu(filename):nested array
Administrator
- userame:string
- password:string
+getLogbook(date,month,year):string
+getuserame(): string
+getpassword(): string
+totalRevenue():integer
+avgRating():float
1
1
1
1
Rating
+choice: integer
+feedback:string
+rate:enum
+showRatingLevel():void
+setRating(rate):void
+displayResponse():void
+setFeedback(feedback):void
1
*
Order
#orderInput:string
#orderList:array
#starterPrice:integer
#mainCoursePrice:integer
#dessertPrice:integer
#totalCost:integer
+setOrderInput(orderInp):void;
+setOrderList:void;
+getOrderList: string;
+getStarterOrderCost:integer
+getMainCourseOrderCost:integer
+getDesserOrderCost:integer
Starters
#singleItemDetail:string
#menuList:nested array
-fileName:string
+display():void
+setMenuList():void
+getMenuList():nested array
MainCourse
#singleItemDetail:string
#menuList:nested array
-fileName:string
+display():void
+setMenuList():void
+getMenuList():nested array
Dessert
#singleItemDetail:string
#menuList:nested array
-fileName:string
+display():void
+setMenuList():void
+getMenuList():nested array
Receipt
-itemName:string
-customerName:String
-receipt:string
-starterStr:string
-mainCourseStr:string
-dessertStr:string
-totalCost:integer
-mtx:mutex
-cv: condition_variable
-receiptGenerated:boolean
-getOrderedItems():string
+getReceipt():string
+writeToFile(content,fileName):void
+addReceiptToIndividualFile():void
+notify(name):void
1