Midterm Exam

Part 2:2 - 4

User Stories

Guest:

- As a guest, I want to search for available rooms so that I can view different types of rooms that are available in the Hotel.
- As a guest, I want to book room so that the room can be available for me when I arrive in the Hotel.
- As a guest, I want to cancel reservations so that in the chance of me not making to the hotel, I can have my refund back as well as the Hotel can know about my change of plan.
- As a guest, I want to view my reservation history so that I can know about the previous rooms where I have stayed.

Receptionist:

- As a receptionist, I want to add new bookings so that we can reserve particular rooms for the intended guests.
- As a receptionist, I want to update or cancel bookings so that if there are any changees to the intended plan of the guest, I can update that information in the system, and in some cases I can even cancel it if the guest intends to do so.
- As a receptionist, I want to check guest reservation history so that I can know if they have previously visited our hotel or cancelled their bookings many times and based on that I can proceed to whether to give them loyalty discount or do some more enquiry.

Manager:

- As a manager, I want to add or remove receptionist so that when there is a new receptionist in our hotel or if someone has left it then based on that I can either make a new receptionist account or delete the one who is leaving.
- As a manager, I want to view occupancy reports so that I can know which of the rooms are available and which are occupied.
- As a manager, I want to generate revenue reports so that I can keep track of the income and expenses of our hotel.

Use Case Description

Guest: Book Room

User Action	System Response
1.User input their login credentials or login as a guest.	1.System gets the login credentials of the user and based on the user type, which is either verified user or guest, leads the user to the main screen UI.
2.Users enter the requirements for the room that they want to book.	2.System gets the requirements and displays various types of rooms in different hotels.
3.User selects the room that they want to book.	3.System displays the room details and the booking form to the user.
4.User input their details such as duration of stay, number of people, etc. And submit the form.	4.System receives the form and displays the booking successful message.

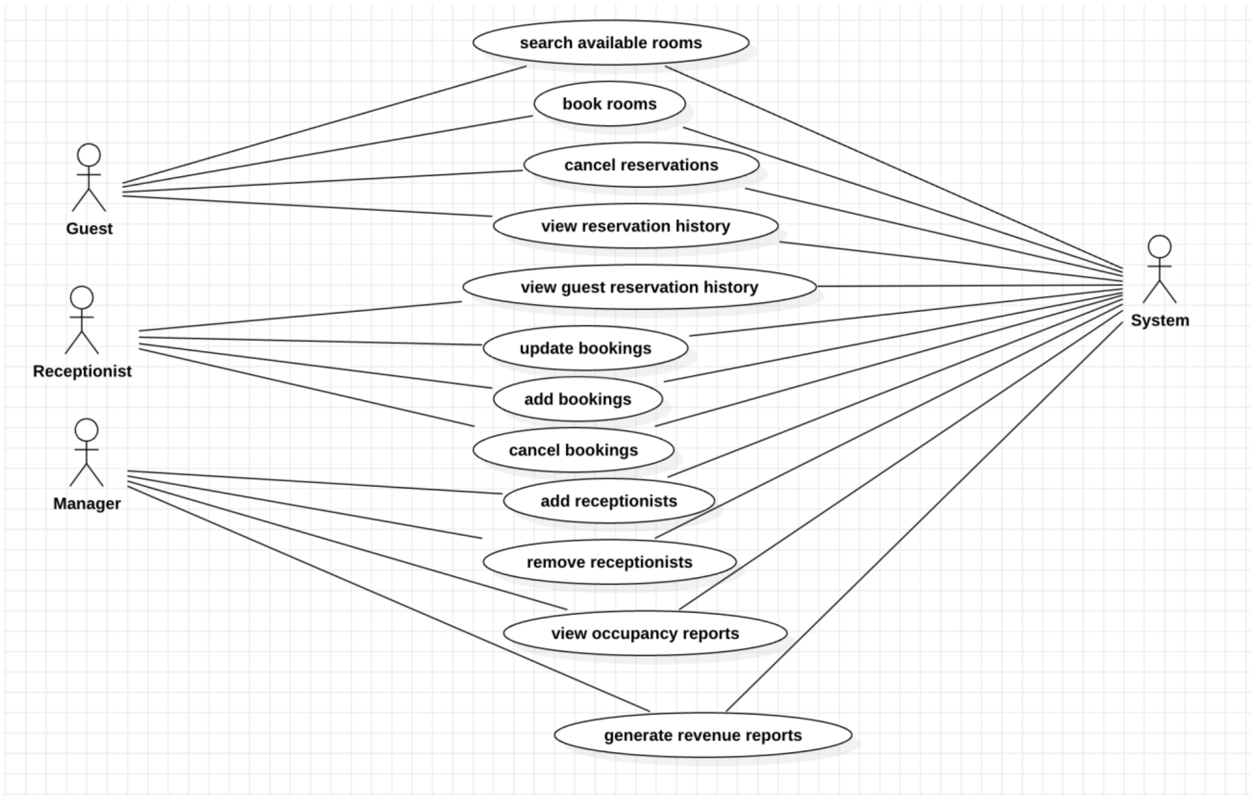
Receptionist: Add Booking

User Action	System Response
1.User input their login credentials.	1.System verifies the login credentials of the receptionist and displays the main screen UI.
2.User selects the notification panel to check for new bookings.	2.System displays the notification panel of any new bookings from the guests.
3.User selects any booking request.	3.System guides the user into the booking confirmation screen.
4.User verifies the booking details whether it is valid or not and confirms it.	4.System shows the successful booking confirmation message.

Manager: Add receptionist

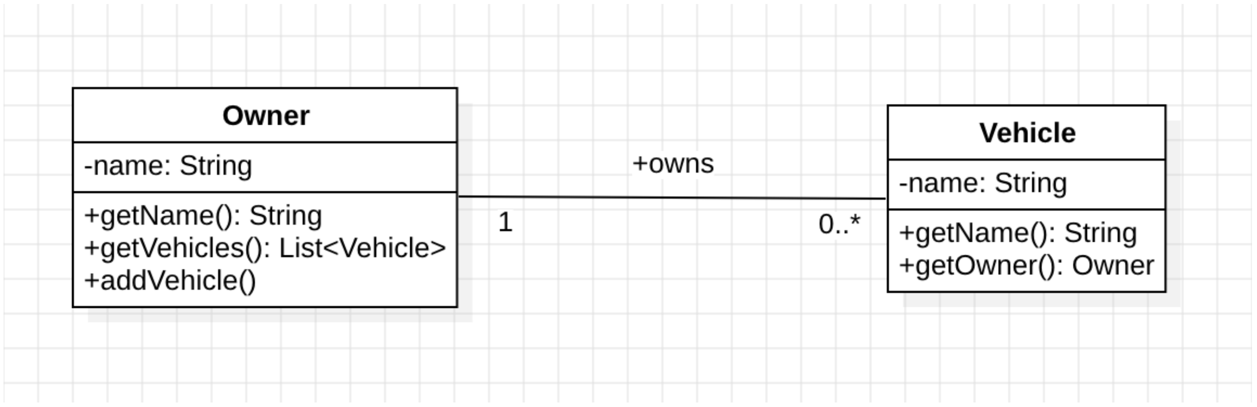
User Action	System Response
1.User input their login credentials	1.System verifies the login credentials of the manager and guides them to the main screen.
2.User selects the “Add new receptionist” button.	2.System displays the form for adding a new receptionist.
3.User input the details of the receptionist and submit them.	3.System verifies the details of the receptionist and displays the successful message.

Use Case Diagram



Part 2:2 – 5

- Classes involved:
  - Owner
  - Vehicle
- Class Diagram:



Part 2:2 – 6

Here, in this implementation of BookShelf, The OO Design principle Inheritance is violated. In Inheritance, we must follow two principles which are IS-A principle and Liskov Substitution Principle. And in this particular implementation, Liskov Substitution Principle is being violated. This principle says that if A is a subclass of B then A can use the methods of B. Similarly, here BookShelf is a subclass of ArrayList which means that it inherits the properties of ArrayList and it can use its properties.

But according to the requirement, we can only remove the book from the end of the shelf but using it as an ArrayList will allow us to remove it from any position from the shelf which violates our requirements.

If someone call remove(0) on BookShelf then it will remove the book which is at the starting position of our BookShelf. But according to our requirement, we are only allowed to remove the book from the end of BookShelf which in turn will make us lose track of the books on the bookshelf.
For better design, we can use Composition instead of Inheritance. I have put the design in code.