When building an app, it is common as a developer to be given a specification or set of requirements for how the app should work and what it should do. This page provides the required features for the Hotel Reservation App.

Remembering and applying these requirements will be easiest if you notice the reasons for them—for example, one requirement is that two people should not be able to book the same room on the same date. That requirement is a realistic one for any functional reservation app that isn't going to drive its users crazy!

Note: You don't need to memorize the information here—rather, you'll want to refer to these requirements as you build your application. We suggest that you either print this page or open it in a new tab.

Once you submit your project, we'll review your work and give you feedback if there's anything that you need to work on. If you'd like to see the exact points that your reviewer will check for when looking at your work, you can have a look over the project **rubric**.

User Scenarios

The application provides four user scenarios:

- **Creating a customer account.** The user needs to first create a customer account before they can create a reservation.
- **Searching for rooms.** The app should allow the user to search for available rooms based on provided checkin and checkout dates. If the application has available rooms for the specified date range, a list of the corresponding rooms will be displayed to the user for choosing.
- **Booking a room.** Once the user has chosen a room, the app will allow them to book the room and create a reservation.
- **Viewing reservations.** After booking a room, the app allows customers to view a list of all their reservations.

Admin Scenarios

The application provides four administrative scenarios:

- **Displaying all customers accounts.**
- **Viewing all of the rooms in the hotel.**
- **Viewing all of the hotel reservations.**
- **Adding a room to the hotel application.**

Reserving a Room – Requirements

The application allows customers to reserve a room. Here are the specifics:

- **Avoid conflicting reservations.** A single room may only be reserved by a single customer per check-in and check-out date range.
- **Search for recommended rooms.** If there are no available rooms for the customer's date range, a search will be performed that displays recommended rooms on alternative dates. The recommended room search will add seven days to the original check-in and check-out dates to see if the hotel has any availabilities and then display the recommended rooms/dates to the customer.

Example: If the customers date range search is 1/1/2020 – 1/5/2020 and all rooms are booked, the system will search again for recommended rooms using the date range 1/8/2020 - 1/12/2020. If there are no recommended rooms, the system will not return any rooms.

Rubric Tip: The reservation should use loops to process the data when finding an open room.

- Test the application to make sure rooms can be found based on different dates from the Main menu.
- Test the application to search for rooms that are already booked to make sure a recommended list of rooms is returned.

Room Requirements

- **Room cost.** Rooms will contain a price per night. When displaying rooms, paid rooms will display the price per night and free rooms will display "Free" or have a $0 price.
- **Unique room numbers.** Each room will have a unique room number, meaning that no two rooms can have the same room number.
- **Room type.** Rooms can be either single occupant or double occupant (Enumeration: SINGLE, DOUBLE).

Customer Requirements

The application will have customer accounts. Each account has:

- **A unique email for the customer.** RegEx is used to check that the email is in the correct format (i.e., **name@domain.com**).

- **A first name and last name.**

The email RegEx is simple for the purpose of this exercise and may not cover all real-world valid emails. For example "**name@domain.co.uk**" would not be accepted by the above RegEx because it does end with ".com". If you would like to try to make your RegEx more sophisticated, you may—but it is not required for this project.

Error Requirements

The hotel reservation application handles all exceptions gracefully (user inputs included), meaning:

- **No crashing.** The application does not crash based on user input.
- **No unhandled exceptions.** The app has `try` and `catch` blocks that are used to capture exceptions and provide useful information to the user. There are no unhandled exceptions.

Tip: There should exist at least one example in the model classes (Room, Customer, Reservation) that overrides both the hashcode and equals methods to utilize Collections functions like contains.