University Room Booking Application

README – Additional Features included in the system:

1. All the names are trimmed for empty spaces in the end.

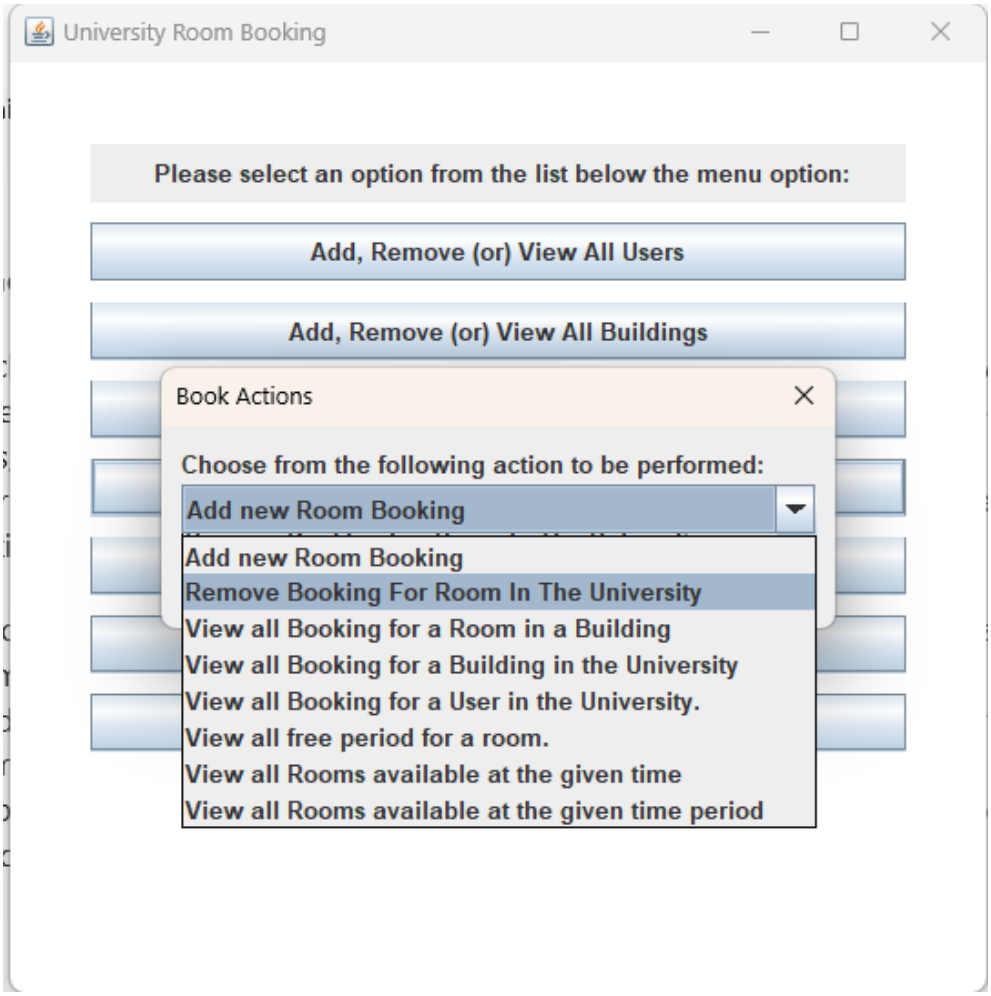
2. The check-In and Check-out date and time are taken into the system, even when the restriction given for this system is that no booking can be stretched across dates, and only a given date is to be allowed. This information of taking 2 separate inputs ,allows the system to be expanded for future application where booking can extend across days.

3. Service Layer usage - The benefits a Service Layer provides is that it defines a common set of application operations available to different clients and coordinates the response in each operation. Where you have an application that has more than one kind of client that consumes its business logic and has complex use cases involving multiple transactional resources - it makes sense to include a Service Layer with managed transactions.

4. Each action performed in the application, performs required validation, and allows the user to create/ view or remove missing parameters whenever needed (at any point in the application).
   Example: If the user tries to add a room booking, and if the building information is not already available in the system, then the user is given option to add a building and room, and then continue with the operation of booking that room. This allows for minimalized navigation around the application and providing ease of performance when using the application.

5. Booking can be viewed based on building, room, user, time and time-period.

6. The application allows to find booking based on additional parameter.
   As shown in the image below the application shows a concise list of inputs, with each section from the main menu allowing to navigate to additional functionality internally.



Each option shown on the main menu leads to its own additional set of commands, keeping the main menu concise and tagging all relevant option under one flow.

The same has been replicated by the command line interface.

MVC Pattern has been followed to build this application along with implementation of service layer to handle all business logic. Each operation has its own controller and service. The folder structure followed as per the standards MVC guidelines.

The application faces issue with respect to loading files, the segregation of controller and services caused the application layers to have its own attributes. On loading the data from an external file, the services create a new memory location for the retrieved data, thereby removing the reference to original university data that was passed across the application. Any operations performed after load, makes the services independent of each there, causing disruppition in the expected application flow