NAME : HAMZA ALI

REG_NO: SP23-BSE-065

LAB_ASSIGNMAENT:

SDA

WHICH DESIGN PATTERN AND GRASP PRINCIPLE USE IN CODE:

DESIGN PATTERN

1. MVC (Model-View-Controller) – Separates logic:

User, Booking, and BookingManager are part of the Model.

SmartTravelGUI is the View and also partly the Controller.

2. Singleton Pattern – Can be applied to BookingManager to ensure only one manager instance exists (not implemented, but commonly used).

3. Observer Pattern – Could be used for real-time UI updates or notifications (like sending email/SMS after booking).

GRASP PRINCIPLE

1. Controller:

A TravelManager or BookingController class handles user requests like booking, cancellation, or viewing schedules.

2. Creator:

BookingManager creates Booking objects, as it logically aggregates and initializes them.

3. Information Expert:

A Booking class knows about travel details (like passenger, time, ticket ID), so it handles checking availability or cost calculations.

4. Low Coupling:

Classes like User, Booking, Payment, and Route interact with minimal dependency, easing maintenance.

5. High Cohesion:

Each class has a focused responsibility: Route manages travel paths, Booking manages reservations, etc.

6. Polymorphism:

Different travel modes (Bus, Train, Flight) can implement a common interface (e.g., TravelMode) for booking.

7. Pure Fabrication:

A Logger or EmailNotifier class can be created for logging or sending confirmations, even though it doesn't map to a real-world concept.

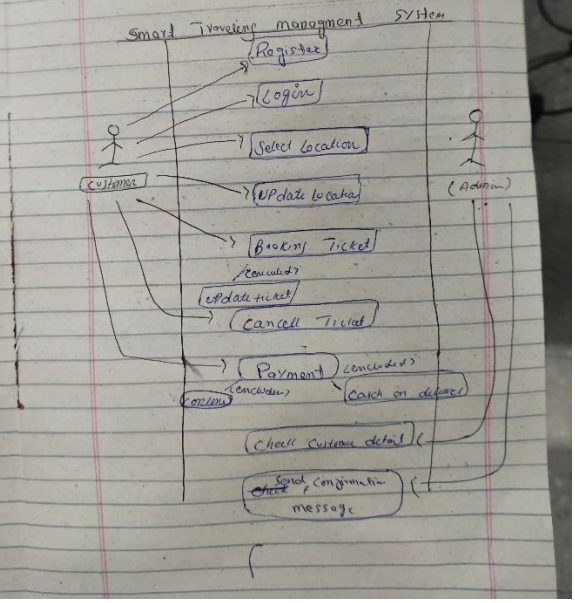
USE-CASE DIAGRAM

Name = Hamza Ali

Reg-No = SP23-BSE-065

Smart Traveling Management System

Use Case Diagram:



COMMUNICATION DIAGRAM: