KASHMALA ZEB

SP23-BSE-048

MID-LAB

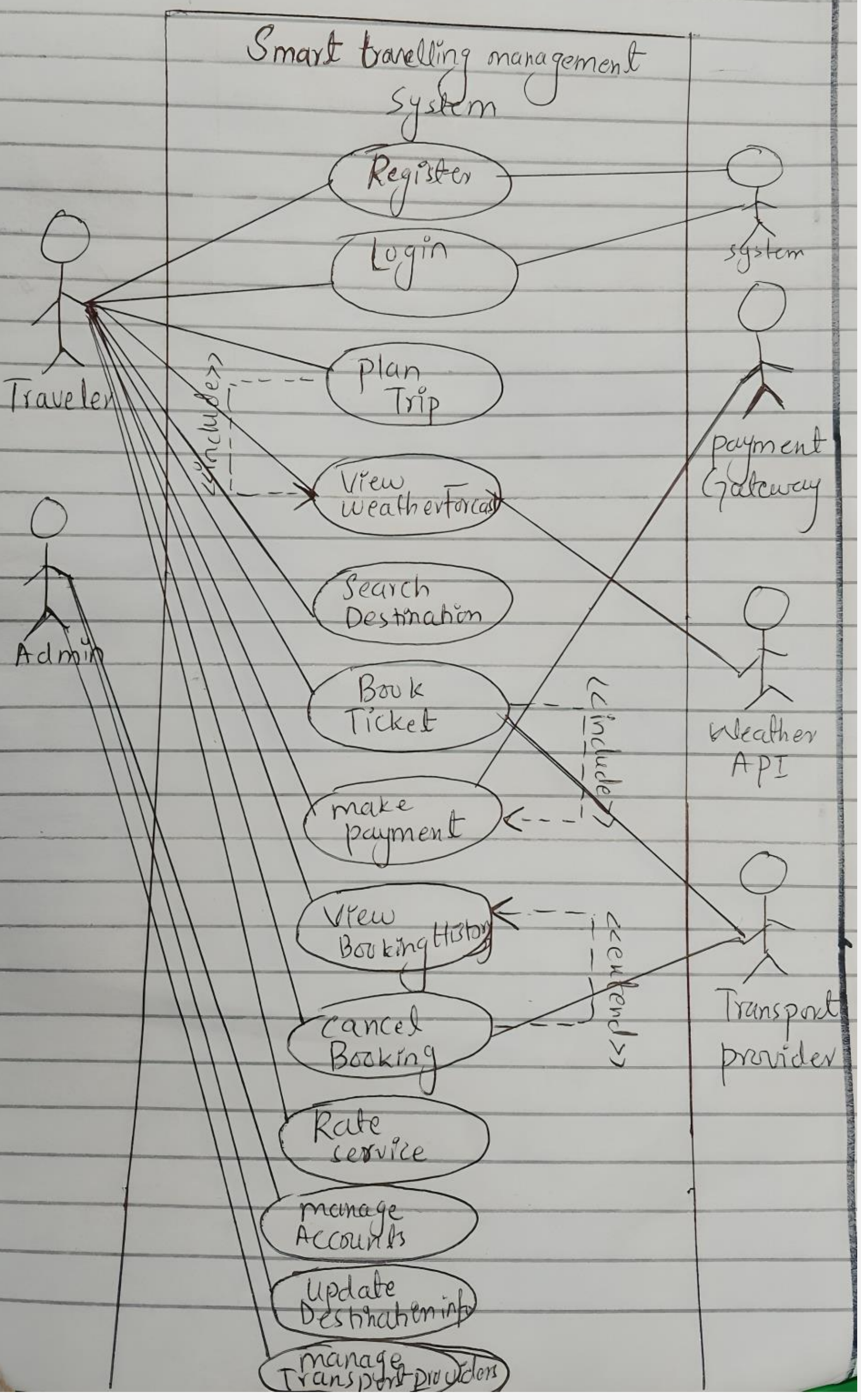
SUBMITTED TO: MUKHTIAR ZAMIN

DATE: 22/05/2025

USE CASE DIAGRAM:

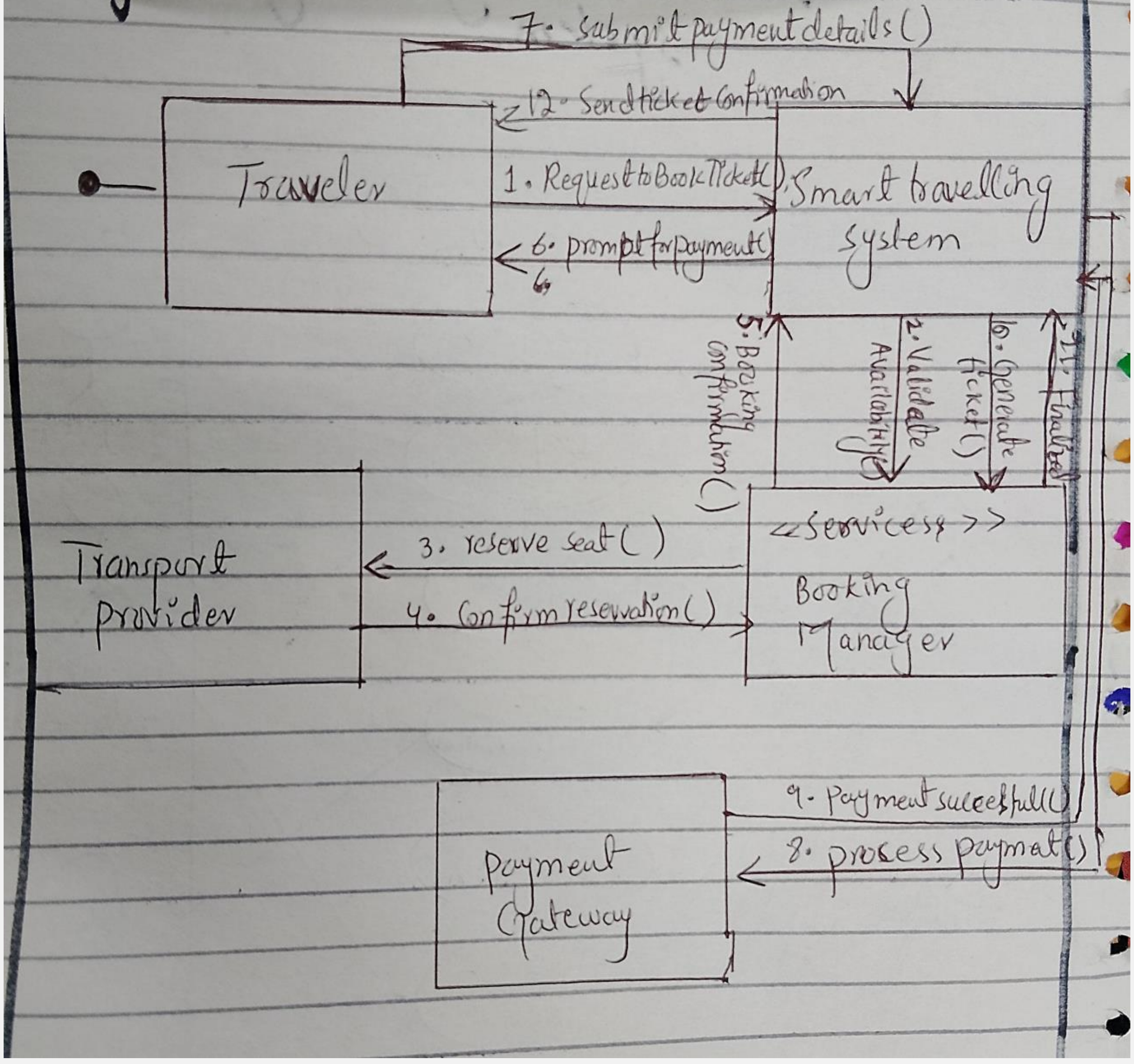
Kashmala Zeb Sp23-BSE-048

Smart travelling management system



Communication Diagram

System Event: User Books a Ticket



Selected Use Case:

User Books a Ticket

Applied GRASP Principles and Design Patterns

1. GRASP Principles

GRASP Principle	Application in Use Case	Justification
Controller	Assign the BookingManager class as the controller.	It handles user requests related to booking and delegates tasks to appropriate components (e.g., Payment Gateway, Transport Provider).
Information Expert	The BookingManager has all booking-related data (user info, trip data, seat availability).	This promotes cohesion and helps manage the logic for booking effectively.
Low Coupling	Each class interacts only as necessary (e.g., BookingManager → PaymentGateway).	Reduces system fragility; makes components easier to test or replace.
High Cohesion	BookingManager only handles booking logic, not unrelated responsibilities like payments.	Improves maintainability and clarity.
Polymorphism	Different types of TransportProvider (Bus, Train, Airline) can implement a common interface.	Helps the system handle multiple transport types uniformly.

2. Design Patterns

Design Pattern	Application in Use Case	Justification
Factory Pattern	Used to create specific transport provider objects dynamically (e.g., create a BusTicketProvider or AirlineProvider).	Allows easy extension for new transport types without changing booking logic.
Strategy Pattern	Used to implement different payment methods (credit card, wallet, bank transfer).	Enables flexible switching between payment strategies at runtime.
Observer Pattern	When booking is confirmed, notify the user (via email/SMS) and admin/log.	Ensures decoupled notification system for multiple subscribers.

Why These Principles/Patterns Are Suitable

- They **reduce complexity** by assigning responsibilities to the right classes (GRASP).
- They **enhance flexibility**, allowing new providers or payment methods without rewriting the core logic.
- They **promote scalability**, making the system ready for future features like discounts, booking history analytics, or cancellation policies.