**SOFTWARE DESIGN AND ARCHITECTURE**

**Submitted to: Sir Mukhtiar Zamin**



**ASSIGNMENT NO 2**

**Date: 26th June 2025**

DEPARTMENT OF SOFTWARE ENGINEERING

COMSATS UNIVERSITY ISLAMABAD

ABBOTTABAD CAMPUS

**Table of contents**

Contents

[CHAPTER NO 1: INTRODUCTION 5](#_Toc201837939)

[GROUP MEMBERS 5](#_Toc201837940)

[CHAPTER NO 2: USE CASE DIAGRAM 6](#_Toc201837941)

[FULL SYSTEM USECASE 6](#_Toc201837942)

[USE CASE NAME: REGISTRATION 7](#_Toc201837943)

[USE CASE NAME: LOGIN 8](#_Toc201837944)

[USE CASE NAME: MANAGE BOOKING 10](#_Toc201837945)

[USE CASE NAME: MANAGE BOOKING 11](#_Toc201837946)

[USE CASE NAME: TRACK VEHICLE LOCATION 12](#_Toc201837947)

[USE CASE NAME: FINANCE TRACKING 13](#_Toc201837948)

[USE CASE NAME: FINANCE TRACKING 14](#_Toc201837949)

[CHAPTER NO 3: FULLY DRESSED USE CASES 15](#_Toc201837950)

[FULLY DRESSED USE CASE: REGISTRATION 15](#_Toc201837951)

[FULLY DRESSED USE CASE : LOGIN 16](#_Toc201837952)

[Fully dress use case dashboard 17](#_Toc201837953)

[FULLY DRESSED USE CASE : MANAGE BOOKING 19](#_Toc201837954)

[View Available Routes 19](#_Toc201837955)

[Book journey 19](#_Toc201837956)

[Cancel Booking 20](#_Toc201837957)

[FULLY DRESSED USE CASE : MANAGE BOOKING 21](#_Toc201837958)

[Confirm Booking 21](#_Toc201837959)

[Payment 21](#_Toc201837960)

[Generate Receipt 22](#_Toc201837961)

[FULLY DRESSED USE CASE: TRACK VEHICLE LOCATION 23](#_Toc201837962)

[FULLY DRESSED USE CASE: FINANCE TRACKING 27](#_Toc201837963)

[FULLY DRESSED USE CASE : FINANCE TRACKING 31](#_Toc201837964)

[Track Defaulters 31](#_Toc201837965)

[Refund Handling 31](#_Toc201837966)

[Security and logs 31](#_Toc201837967)

[CHAPTER NO 4: SYSTEM SEQUENCE DIAGRAM 33](#_Toc201837968)

[SYSTEM SEQUENCE DIAGRAM : REGISTRATION 34](#_Toc201837969)

[SYSTEM SEQUENCE DIAGRAM: LOGIN 35](#_Toc201837970)

[SYSTEM SEQUENCE DIAGRAM: dashboard 36](#_Toc201837971)

[SYSTEM SEQUENCE DIAGRAM: MANAGE BOOKING 37](#_Toc201837972)

[SYSTEM SEQUENCE DIAGRAM: MANAGE BOOKING 38](#_Toc201837973)

[SYSTEM SEQUENCE DIAGRAM: TRACK VEHICLE LOCATION 39](#_Toc201837974)

[SYSTEM SEQUENCE DIAGRAM: FINANCE TRACKING 40](#_Toc201837975)

[SYSTEM SEQUENCE DIAGRAM: FINANCE TRACKING 41](#_Toc201837976)

[CHAPTER NO 5: PACKAGE DIAGRAM 42](#_Toc201837977)

[PACKAGE DIAGRAM: REGISTRATION 43](#_Toc201837978)

[PACKAGE DIAGRAM: LOGIN 43](#_Toc201837979)

[PACKAGE DIAGRAM: Dashboard 44](#_Toc201837980)

[PACKAGE DIAGRAM: MANAGE BOOKING 45](#_Toc201837981)

[PACKAGE DIAGRAM: MANAGE BOOKING 46](#_Toc201837982)

[PACKAGE DIAGRAM: TRACK VEHICLE LOCATION 47](#_Toc201837983)

[PACKAGE DIAGRAM: FINANCE TRACKING 48](#_Toc201837984)

[PACKAGE DIAGRAM: FINANCE TRACKING 49](#_Toc201837985)

[CHAPTER NO 6: COMMUNICATION DIAGRAM 50](#_Toc201837986)

[COMMUNICATION DIAGRAM : Registartion 51](#_Toc201837987)

[COMMUNICATION DIAGRAM : LOGIN 51](#_Toc201837988)

[COMMUNICATION DIAGRAM : dashboard 52](#_Toc201837989)

[COMMUNICATION DIAGRAM: MANAGE BOOKING 52](#_Toc201837990)

[COMMUNICATION DIAGRAM: MANAGE BOOKING 53](#_Toc201837991)

[COMMUNICATION DIAGRAM: TRACK VEHICLE LOCATION 54](#_Toc201837992)

[COMMUNICATION DIAGRAM: FINANCE TRACKING 55](#_Toc201837993)

[SEQUENCE DIAGRAM: FINANCE TRACKING 56](#_Toc201837994)

[CHAPTER NO 7: CLASS DIAGRAM 56](#_Toc201837995)

[CLASS DIAGRAM: REGISTRATION 57](#_Toc201837996)

[CLASS DIAGRAM: registration 57](#_Toc201837997)

[CLASS DIAGRAM: Login 57](#_Toc201837998)

[CLASS DIAGRAM: Dashboard 58](#_Toc201837999)

[CLASS DIAGRAM: MANAGE BOOKING 59](#_Toc201838000)

[CLASS DIAGRAM: MANAGE BOOKING 60](#_Toc201838001)

[CLASS DIAGRAM: TRACK VEHICLE LOCATION 61](#_Toc201838002)

[CLASS DIAGRAM: FINANCE TRACKING 62](#_Toc201838003)

[CLASS DIAGRAM: FINANCE TRACKING 63](#_Toc201838004)

[CHAPTER NO 8: STATE TRANSITION DIAGRAM 64](#_Toc201838005)

[STATE TRANSITION DIAGRAM: Registration ,Login, Dashboard 64](#_Toc201838006)

[STATE TRANSITION DIAGRAM: LOGIN & REGISTER 64](#_Toc201838007)

[STATE TRANSITION DIAGRAM: MANAGE BOOKING 64](#_Toc201838008)

[STATE TRANSITION DIAGRAM: MANAGE BOOKING 65](#_Toc201838009)

[STATE TRANSITION DIAGRAM: TRACK VEHICLE LOCATION 66](#_Toc201838010)

[STATE TRANSITION DIAGRAM: FINANCE TRACKING 67](#_Toc201838011)

[STATE TRANSITION DIAGRAM: FINANCE TRACKING 68](#_Toc201838012)

[STATE TRANSITION DIAGRAM: 69](#_Toc201838013)

[CHAPTER NO 9: ENTITY RELATIONSHIP DIAGRAM 71](#_Toc201838014)

[CHAPTER NO 10: CLASS DIAGRAM 72](#_Toc201838015)

[CHAPTER NO 11: PACKAGE DIAGRAM 73](#_Toc201838016)

[CHAPTER NO 12: CODING STANDARDS 74](#_Toc201838017)

# CHAPTER NO 1: INTRODUCTION

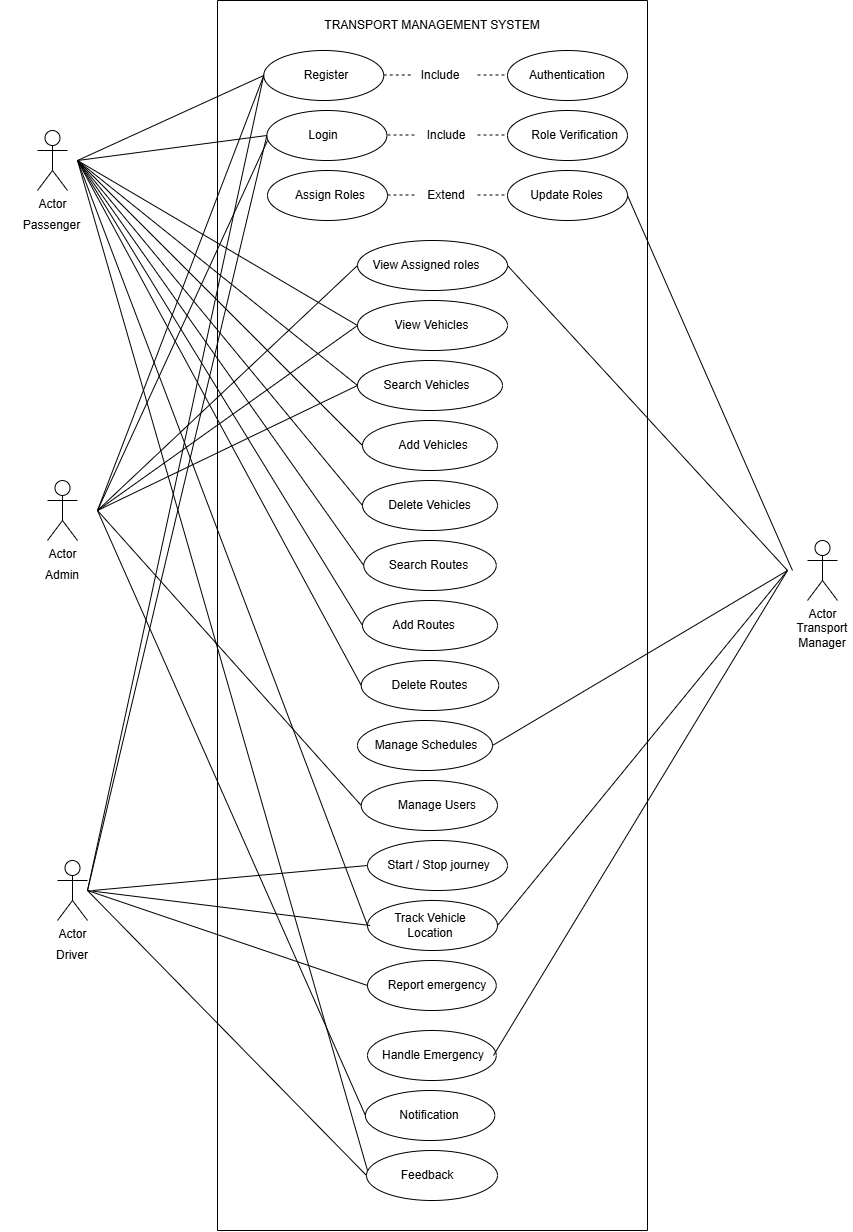
The Transport Management System is a comprehensive platform designed to streamline and optimize the planning, execution, and tracking of transportation operations. It facilitates efficient route planning, vehicle and driver management, real-time tracking, and delivery scheduling. The system aims to reduce operational costs, improve delivery timelines, and enhance overall logistics visibility, making transport operations more reliable and data-driven.

## GROUP MEMBERS

1. Maryam Khan (SP23-BSE-066) - Track Vehicle Location
2. Fatima Khan (SP23-BSE-102) - Manage Booking (Confirm Booking, Payment, Generate Receipt)
3. Alaina Khan (SP23-BSE-069) - Manage Booking (View Available Vehicles, view routes details, Book journey)
4. Warda Yousaf (SP23-BSE-042) - Login
5. Kashmala Zeb (SP23-BSE-048) - Finance (Manage fee structure, Generate Fee challan, Collect Payment, User Dashboard)
6. Tehreem Jilani (SP23-BSE-041) - Finance (Track Defaulters, Refund Handling, Security and logs)
7. Sarina Amjad (SP22-BSE-095) - Register

# CHAPTER NO 2: USE CASE DIAGRAM

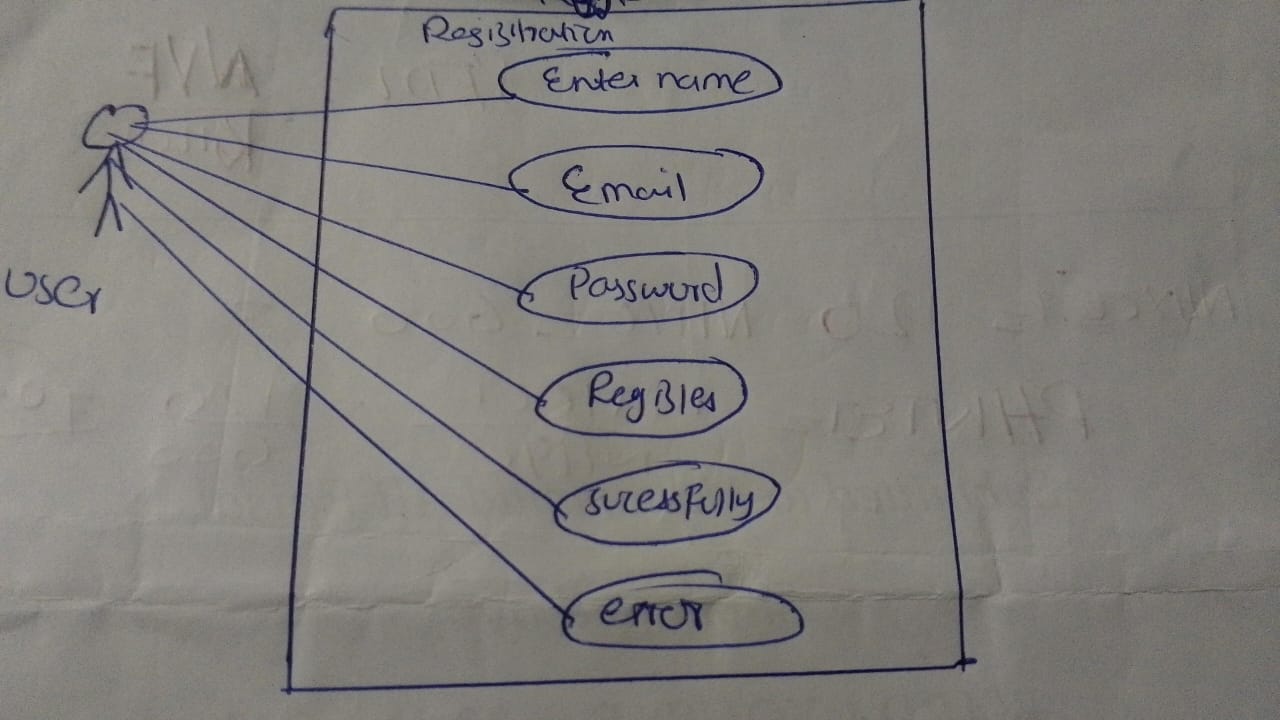
## FULL SYSTEM USECASE



Sarina Amjad SP22-BSE-095

Warda Yousaf SP23-BSE-042

### USE CASE NAME: REGISTRATION



### USE CASE NAME: LOGIN

A drawing of a login form

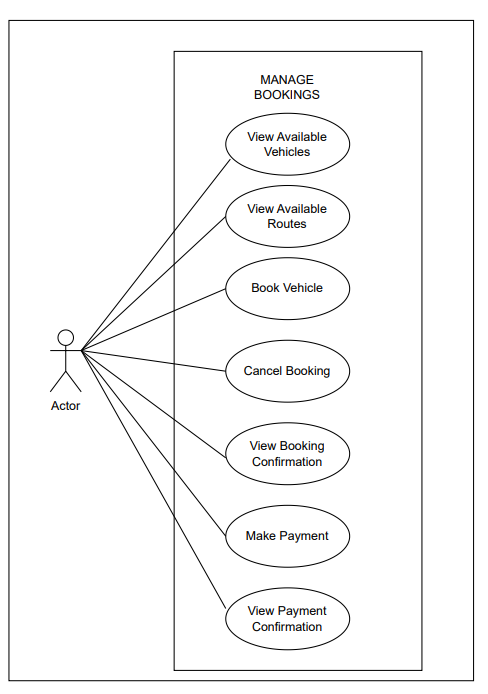
AI-generated content may be incorrect.

Usecase name :dashboardA diagram of a diagram

AI-generated content may be incorrect.

Alaina Khan SP23-BSE-069

### USE CASE NAME: MANAGE BOOKING



Fatima Khan SP23-BSE-102

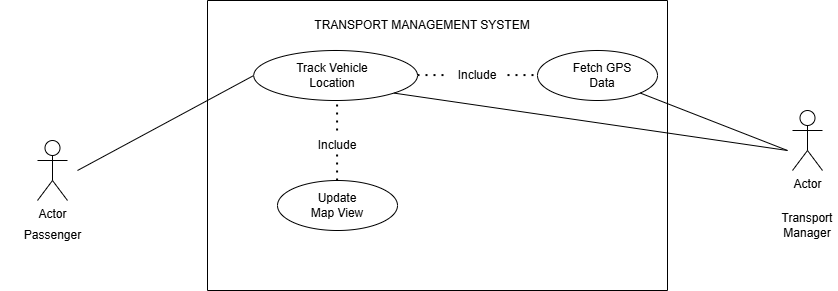
### USE CASE NAME: MANAGE BOOKING

A diagram of a person

AI-generated content may be incorrect.

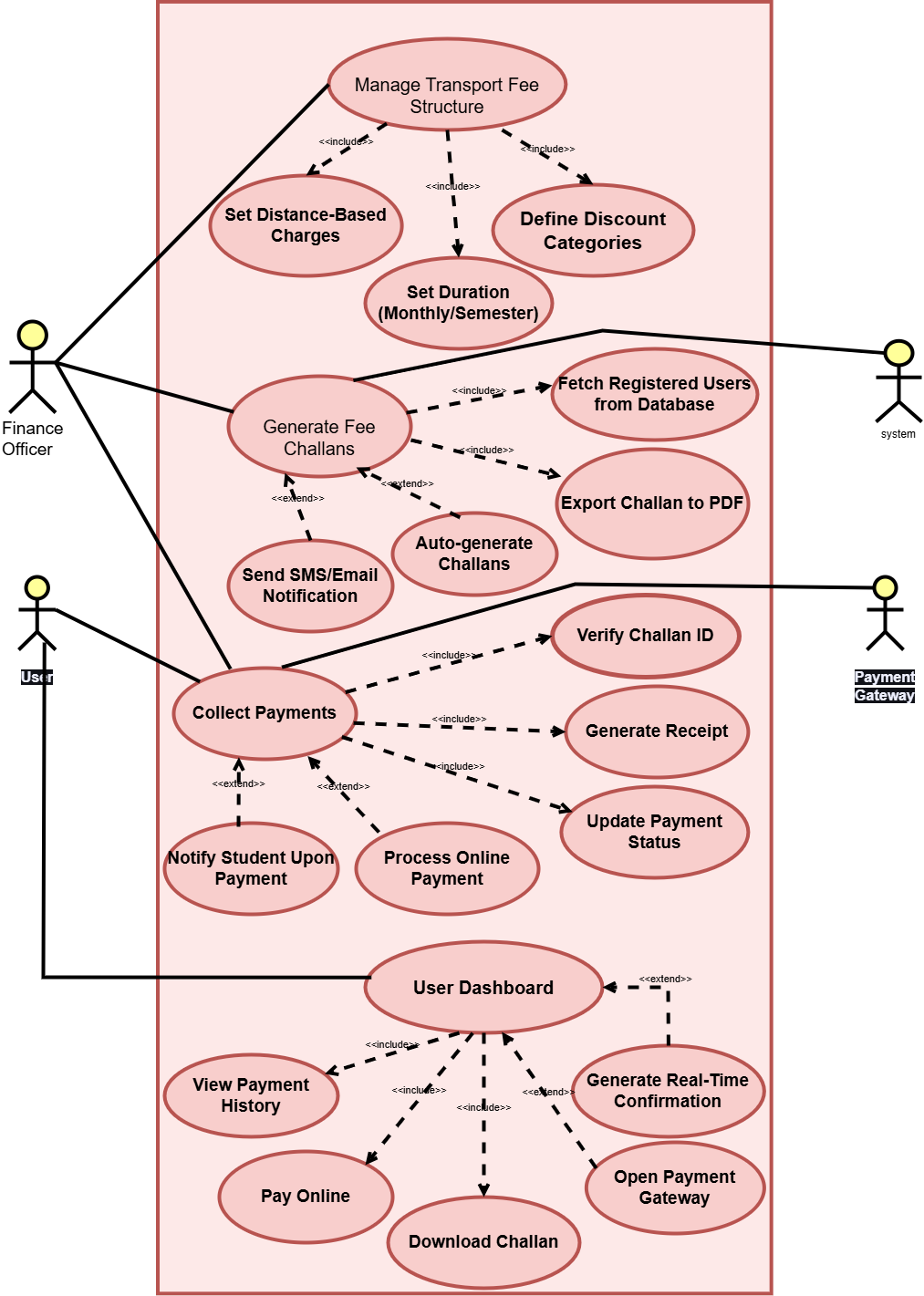
Maryam khan SP23-BSE-066

### USE CASE NAME: TRACK VEHICLE LOCATION



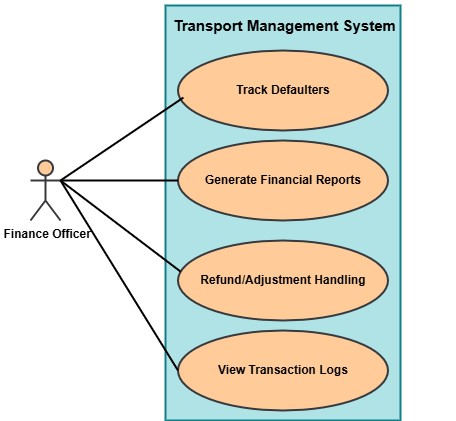
Kashmala Zeb SP23-BSE-048

### USE CASE NAME: FINANCE TRACKING



Tehreema Jilani SP23-BSE-041

### USE CASE NAME: FINANCE TRACKING



# CHAPTER NO 3: FULLY DRESSED USE CASES

Sarina Amjad SP22-BSE-095

Warda Yousaf SP23-BSE-042

### FULLY DRESSED USE CASE: REGISTRATION

|  |  |
| --- | --- |
| **Use Case Element** | **Details** |
| **Use Case ID** | UC-01 |
| **Use Case Name** | Register User |
| **Actor(s)** | User |
| **Description** | Allows a user to register by entering their name, email, and password. Upon successful validation, the data is stored in the database. |
| **Preconditions** | - Database must be connected- Email must not already exist |
| **Postconditions** | - User data is inserted into the database- Success message is shown- Input fields are cleared |
| **Main Flow (Basic Steps)** | 1. User opens registration form2. User enters name, email, and password3. User clicks "Register"4. System validates inputs5. System saves user data6. Success message shown and fields cleared |
| **Alternate Flows** | **AF1:** Any field empty → Show: *"Please fill all fields!"***AF2:** Duplicate email → Show: *"Email already registered!"***AF3:** SQL/Connection error → Show: *"Error: [Message]"* |
| **Business Rules** | - Email must be unique in the users table- All fields are mandatory- Password input must be masked (invisible while typing) |
| **Exceptions** | - SQLIntegrityConstraintViolationException if email exists- SQLException for DB/insert errors- NullPointerException if DB connection fails |
| **Assumptions** | - User uses official system interface- MySQL server is active at localhost:3306- Credentials are root/root- users(name, email, password) table already exists |

### FULLY DRESSED USE CASE : LOGIN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  |  |  |  | | --- | --- | | **User Login** | | |  | | UC-2 |
| |  |  | | --- | --- | | **Use Case Name** |  | | |  | | --- | |  |  |  | | --- | | User Login | |
| |  | | --- | | **Actor** |  |  | | --- | |  | | |  | | --- | |  |  |  | | --- | | End User (Customer) | |
| |  | | --- | | **Trigger** |  |  | | --- | |  | | |  | | --- | |  |  |  | | --- | | User navigates to the login page and clicks the login button after entering credentials | |
| **Description** | This use case allows a registered user to log into the Transport Management System using valid credentials. |
| **Preconditions** | * User must be registered in the system * User has valid login credentials (email/username and password) |
| **Postconditions** | * **Success**: User is logged in and redirected to their dashboard. * **Failure**: User remains on the login page and receives an error message |
| |  | | --- | | **Normal Flow** |  |  | | --- | |  | | 1. User navigates to the login page. 2. System displays the login form. 3. User enters email/username and password. 4. User clicks the “Login” button. 5. System validates the credentials. 6. If valid, system logs in the user and redirects to the dashboard. |
| **Alternative Flow** | **A1: Forgot Password** 1. User clicks on “Forgot Password”. 2. System prompts for the registered email. 3. User enters email. 4. System sends password reset link. 5. User resets password and logs in with the new one |
| |  | | --- | | **Exception Flow**  **Baseness Rules** |  |  | | --- | |  | | |  | | --- | |  |  |  | | --- | | **E1: Invalid Credentials** 1. User enters incorrect email or password. 2. System displays: "Invalid username or password."  **E2: Account Locked** 1. User fails login 3 times. 2. System locks account for 15 minutes and displays: Account temporarily locked due to multiple failed login attempts.  BR-01 Password must be at least 8 characters, contain letters and at least 1 digit.  BR-02 Aafter 5 failed login attempts, account is locked for 15 minutes  BR-03 Only registered users can access login functionality. | |
| **Assumptions** | |  | | --- | |  |  |  | | --- | | * User has stable internet connection. * User is using a supported browser. * User is already registered in the system. | |

### Fully dress use case dashboard

|  |
| --- |
| Use Case ID |
| UC-1 |
| Use Case Name |
| Dashboard Access and Navigation |
| Description |
| The dashboard provides an overview and centralized access point for different modules such as Manage Bookings, Vehicle Tracking, Finance Management, Customer Feedback, Logout, and Back to Login. |
| Primary Actor |
| Administrator/User |
| Secondary Actor |
| System UI |
| Trigger |
| The administrator/User logs into the system successfully and is redirected to the dashboard. |
| Preconditions |
| - Administrator/User must be authenticated through the login form. - The system must be running and the GUI components must be loaded. |
| Postconditions |
| - The dashboard window is displayed with header, footer, feature cards, and side menu (when toggled). - The admin can access other use cases from the dashboard. |
| Normal Flow |
| 1. Administrator logs in successfully. 2. System displays the dashboard window. 3. Administrator views welcome message and system status. 4. Administrator clicks the menu button to toggle the side menu. 5. Administrator clicks on a desired module (e.g., “Manage Bookings”). 6. System shows appropriate messages or opens sub-modules. |
| Alternative Flows |
| A1: Menu button not responding  1. The button is clicked but the side menu does not appear.  2. System shows an error message or no response.  A2: Window fails to load properly  1. GUI components not initialized properly.  2. System logs the exception and optionally shows a fallback screen. |
| Assumptions |
| - All Swing components are correctly initialized. - System resources are sufficient to load the GUI. - Admin is trained to use the system. |
| Business Rules |
| - The dashboard should not be accessible without login. - Navigation through buttons should be intuitive and responsive. - Only users with appropriate roles can see and use certain buttons (can be extended). |
| Exceptions |
| E1: System throws NullPointerException due to missing component E2: JButton actions not registered properly E3: The application crashes due to UI rendering failur |

Alaina Khan SP23-BSE-069

### FULLY DRESSED USE CASE : MANAGE BOOKING

#### View Available Routes

|  |  |
| --- | --- |
| **USE CASE ID** | **UC03** |
| **USE CASE NAME** | View Available Routes |
| **ACTOR** | User |
| **DESCRIPTION** | Allows users to view route stops, times, and vehicle mapping. |
| **TRIGGER** | User clicks “View Routes” in the UI. |
| **PRE-CONDITION** | Routes must exist in the database and be linked to vehicles. |
| **POST CONDITION** | Available routes are displayed in the UI. |
| **NORMAL FLOW** | 1. User opens ViewAvailableRoutesUI.2. System queries RouteRepository.3. Routes are fetched and shown. |
| **ALTERNATIVE FLOW** | No routes found → Show “No routes available” message. |
| **EXCEPTIONS** | - DB error → Show failure message.- Invalid vehicle\_id → Skip route loading. |
| **BUSINESS RULES** | Routes must be tied to vehicles via FK vehicle\_id. |
| **ASSUMPTIONS** | Route data exists in the system. |

#### Book journey

|  |  |
| --- | --- |
| **USE CASE ID** | **UC01** |
| **USE CASE NAME** | Book a Vehicle |
| **ACTOR** | User |
| **DESCRIPTION** | Allows a user to book a vehicle by selecting a route, vehicle, and number of seats. |
| **TRIGGER** | User clicks the "Book" button in the vehicle list UI. |
| **PRE-CONDITION** | - User must be logged in.- Vehicles and routes must be available. |
| **POST CONDITION** | Booking is stored in the database with status = "Pending" and payment\_status = "unpaid". |
| **NORMAL FLOW** | 1. User selects a vehicle.2. User enters booking details.3. System validates input.4. Controller creates booking.5. Booking saved in DB.6. Confirmation message shown. |
| **ALTERNATIVE FLOW** | If invalid details → Show validation error. |
| **EXCEPTIONS** | - DB connection error → Show system error.- Vehicle full → Show availability error. |
| **BUSINESS RULES** | - Seats booked ≤ available seats.- Initial booking state is “Pending” and “unpaid”. |
| **ASSUMPTIONS** | - System clock is reliable.- Vehicles/routes are already available in DB. |

#### Cancel Booking

|  |  |
| --- | --- |
| **USE CASE ID** | **UC02** |
| **USE CASE NAME** | Cancel a Booking |
| **ACTOR** | User |
| **DESCRIPTION** | Enables a user to cancel an existing booking and optionally see refund status. |
| **TRIGGER** | User clicks "Cancel Booking" in the UI. |
| **PRE-CONDITION** | Booking must exist and belong to the logged-in user. |
| **POST CONDITION** | Booking is moved to cancelled\_bookings table. |
| **NORMAL FLOW** | 1. User opens CancelBookingsUI.2. System loads bookings.3. User selects a booking.4. System checks payment status.5. Booking is cancelled.6. Confirmation displayed. |
| **ALTERNATIVE FLOW** | If booking already cancelled → Do not allow again. |
| **EXCEPTIONS** | - Booking not found → Error message.- DB error → Show system error. |
| **BUSINESS RULES** | Paid bookings cannot be cancelled automatically. |
| **ASSUMPTIONS** | - BookingController is integrated with BookingRepository. |

Fatima Khan SP23-BSE-102

### FULLY DRESSED USE CASE : MANAGE BOOKING

#### Confirm Booking

|  |  |
| --- | --- |
| **Use case ID** | UC- 6 |
| **Use case name** | Confirm booking |
| **Actors** | **Primary**: user, customer **secondary**: Transport Management System (TMS), Payment Gateway, Admin |
| **Description** | This use case describes the process by which a customer confirms a transport service booking (such as a seat on a bus, a cargo shipment, or a taxi), after selecting transport details and entering required information. |
| **Trigger** | The customer has completed selecting transport details and clicks the "Confirm Booking" button. |
| **Pre-conditions** | **Pre-1**: The user is logged in or has provided necessary personal details.  **Pre-2** The user has selected a valid transport route/service.  **Pre-3**: The system shows the availability of the selected service.  **Pre-4**: All required fields are completed. |

#### Payment

|  |  |
| --- | --- |
| **Use case ID** | UC-7 |
| **Use case name** | Make payment |
| **Actors** | **Primary**: user, customer **secondary**: Payment Gateway (e.g., Stripe, PayPal, etc.), Bank System (for verifying and processing transactions), admin. |
| **Description** | This use case describes the process by which a customer completes payment for a transportation-related service, such as booking a shipment, purchasing a ticket, or scheduling a delivery. The system verifies payment details, processes the transaction, and confirms the service booking. |
| **Trigger** | The customer initiates payment after selecting a transportation service (e.g., delivery, freight booking, ticket reservation). |
| **Pre-conditions** | **Pre-1**: Customer is registered and logged into the TMS platform  **Pre-2**: A valid service (shipment, route, vehicle, delivery schedule) has been selected  **Pre-3**: The system has calculated total charges including taxes and any additional fees.  **Pre-4**: A payment method is available and supported by the system. |

#### Generate Receipt

|  |  |
| --- | --- |
| **Use case ID** | UC-8 |
| **Use case name** | Generate receipt |
| **Actors** | **Primary**: system **secondary**: Customer (initiates or receives the receipt), Admin or Transport Operator (may request receipt generation), Payment Gateway (provides transaction reference) |
| **Description** | This use case allows the system to automatically or manually generate a receipt after a successful payment for transport services. The receipt includes transaction details, customer information, service summary, and a unique receipt number. |
| **Trigger** |  |
| **Pre-conditions** | **Pre-1**: Payment has been successfully completed.  **Pre-2**: A booking or service order exists in the system.  **Pre-3**: Customer and transaction data are stored. |
| **Post conditions:** | **Post-1:** A digital receipt (PDF or email format) is generated.  **Post-2:** The receipt is saved in the system for future reference.  **Post-3:**. A receipt and confirmation details are sent to the customer.  **Post-4:**. The customer receives a copy via email or downloads it from their dashboard. |

Maryam khan SP23-BSE-066

### FULLY DRESSED USE CASE: TRACK VEHICLE LOCATION

|  |  |
| --- | --- |
| USE CASE 1D | UC-9 |
|  |  |
| USE CASE NAME | Track Vehicle Location in a Transport Management System |
| Actor | Primary Actor: Passenger, Transport Manager Secondary Actor: System |
| Description | This use case allows Transport managers and users to view a vehicle's real-time location on a map using GPS data, helping with monitoring, route tracking, and arrival updates. |
| Trigger | The user (Transport Manager or passenger) selects the option to track a vehicle's location from the system dashboard or mobile app. |
| Pre-Condition | PRE-1: The vehicle has a GPS tracking device installed inside it.  PRE-2: The device has a GPS module, microprocessor, 4G SIM, and power source.  PRE-3: Server application is running on the correct protocol and port.  PRE-4: Frontend is built and is able to call backend. |
| Post Condition | POST-1: Location of vehicle is accurately measured by the tracker.  POST-2: The location is send to the server via 4G network.  POST-3: Server receives the data and store it in the database. POST-4: Real-time position is displayed to authorized users on a map UI. |
| Normal Flow | 13.0 Track Vehicle Location   1. When the user clicks on the GPS tracking button the satellites start sending signals to the GPS module inside the tracker. 2. Inside the module the antenna receives the signals and cleans and amplifies it. Then the GPS processor decodes the signals and gives output in the form of location. 3. 4G SIM installed in the tracker takes the signals to the server IP through the custom port using TCP or UDP protocol. 4. Server receives the data and stores it in database. 5. When accessed through the Web or mobile, server queries the database and gives the exact location. 6. The location is then displayed on the Google map where markers displays the location of vehicle. |

|  |  |
| --- | --- |
| Alternative Flow | **1a. No GPS Signals**  4a1. GPS is not able to get the signals from the satellite and displays on the screen “Location unavailable” **2a. GPS module not working**  2a1. There is a problem in the hardware of the module.  2a2. Displays a warning: Vehicle not connected. Last update at timestamp. **3a. No Internet**  3a1. Location is buffered in device memory  3a2. Shows message: Send later when the signal is back.  **4a. Server not available**  4a1. Server is not responding so the system displays message  “Resend message”.  4a2. If retries fail, device may switch to a backup server (if configured).  **5a. Privacy mode**  5a1. User has enabled the privacy mode.  5a2. Location can be viewed inside the car but not on other people devices. |
| Exceptions | * **Vehicle Not Connected:** * GPS device is offline or not transmitting data. * System displays the last known location with a warning.  **Map API Fails:** * If the map fails to load, show fallback coordinates or an error message.  **Unauthorized Access:** * A user tries to access tracking without proper permissions. System denies access. * **Vehicle Not Found:** * The selected vehicle ID doesn't exist or has been removed from the system. |
| Business Rules | * Vehicle location data must refresh at regular intervals (e.g., every 10 seconds). * Only authorized users (Transport managers or assigned users) can access tracking information. * Users can only track vehicles assigned to their specific trip or booking. * All tracking actions should be logged for audit and security purposes. * The system should store vehicle location history for reporting and analysis. |
|  | * Map display must be consistent and accurate based on the GPS coordinates received. * If a vehicle stops transmitting location, the system must indicate it with the last known location timestamp.  Users must not be able to spoof or modify location data manually. |
| Assumptions | 1. All vehicles are equipped with functioning GPS tracking devices. 2. There is a reliable internet connection for both GPS and user devices. 3. Passengers are informed of which vehicle they are linked to (e.g., via booking ID). 4. Map API integration is properly configured and available. 5. Users are familiar with basic map interaction (zoom, pan, select vehicle). |

Kashmala Zeb SP23-BSE-048

### FULLY DRESSED USE CASE: FINANCE TRACKING

**Manage Transport Fee Structure**

|  |  |
| --- | --- |
| **Use Case ID** | **UC-10** |
| **Use Case Name** | Manage Transport Fee  Structure |
| **Actors** | Primary: Finance Officer |
| **Trigger** | Admin/Officer wants to define or update transport fee structure |
| **Description** | Allows defining fee slabs based on user type, location, distance, and applicable discounts |
| **Preconditions** | Finance Officer must be logged in with permission |

|  |  |
| --- | --- |
| **Postconditions** | Fee slabs saved and reflected in challan generation |
| **Main Flow** | 1. Officer logs in 2. Navigates to fee config  3. Enters user-type, route, and slab details 4. Adds discounts if needed  5. Saves structure |
| **Alternative Flow** | Officer edits existing structure |
| **Includes** | * Admin authentication * User category validation |
| **Extends** | - N/A |
| **Exceptions** | Fields missing Invalid input |
| **Business Rules** | Only Admin/Finance roles can change fee structure |
| **Assumptions** | All fees are defined at  the start of each semester |

**Generate Fee Challans**

|  |  |
| --- | --- |
| **Use Case ID** | **UC-11** |
| **Use Case Name** | Generate Fee Challans |
| **Actors** | Primary: Finance  Officer, Students/Staff  (receive challan)  Secondary: system |

|  |  |
| --- | --- |
| **Trigger** | Start of semester/month triggers challan generation |
| **Description** | Generates transport challans  (manually/automatically  ), sends to students/staff |
| **Preconditions** | Fee structure must exist User must be registered |
| **Postcondition** | Challan PDF generated, sent via SMS/email, and saved in system |
| **Main Flow** | 1. Officer logs in 2. Selects user/batch 3. System fetches fees 4. Challan generated 5. Sent to users |
| **Alternative Flow** | Users download challan manually |
| **Includes** | * Generate PDF * Send Notification |
| **Extends** | - Manual download option |
| **Exceptions** | SMS/email failure Data not found |
| **Business Rules** | Must follow correct fee mapping rules |
| **Assumptions** | Internet/email system is functional |

**Collect Payments (Online & Offline)**

|  |  |
| --- | --- |
| **Use Case ID** | **UC-12** |
| **Use Case Name** | Collect Payments (Online & Offline) |
| **Actors** | Primary: Finance  Officer, Students/Staff  (FOR ONLINE), |

|  |  |
| --- | --- |
|  | Secondary: payment gateway |
| **Trigger** | User submits payment manually or via portal |
| **Description** | Allows both manual collection at office and online payment through gateway |
| **Preconditions** | Valid challan must exist System must be online |
| **Postconditions** | Payment status updated  Receipt generated |
| **Main Flow** | **Offline:**  1. User pays at office 2. Officer enters transaction 3. Receipt printed **Online:**  1. User logs in |
|  | 1. Clicks “Pay Now” 2. Redirects to gateway 3. Completes payment 4. Receipt saved |
| **Alternative Flow** | Online payment fails →  user retries or pays manually |
| **Includes** | * Payment Gateway API * Generate Receipt |
| **Extends** | - Online payment  (optional extension) |
| **Exceptions** | Payment timeout Bank server failure |
| **Business Rules** | System must sync with payment gateway securely |
| **Assumptions** | User has stable  connection for online payment |

**User Dashboard**

|  |  |
| --- | --- |
| **Use Case ID** | **UC-13** |
| **Use Case Name** | User Dashboard |
| **Actors** | Primary: user |
| **Trigger** | User logs into portal |
| **Description** | Enables users to view payment history, download challans, and make online payments |
| **Preconditions** | User must be registered and logged in |
| **Postconditions** | User actions recorded and system reflects updated views |
| **Main Flow** | 1. User logs in 2. Views fee status 3. Downloads challan 4. Pays online if needed 5. Views receipts/history |
| **Alternative Flow** | User has no previous payment history |
| **Includes** | * View Payment History * Download Challan |
| **Extends** | - Initiate Online  Payment |
| **Exceptions** | User not registered Invalid session |
| **Business Rules** | Only registered users can access dashboards |
| **Assumptions** | Assumes stable user login and account mapping |

Tehreema Jilani SP23-BSE-041

### FULLY DRESSED USE CASE : FINANCE TRACKING

#### Track Defaulters

#### Refund Handling

#### Security and logs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Track**  **Defaulters** | **Generate**  **Financial**  **Reports** | **Refund/Adjustment**  **Handling** | **View**  **Transaction**  **Logs** |
| **Use Case ID** | UC-FIN-14 | UC-FIN-15 | UC-FIN-16 | UC-FIN-17 |
| **Primary Actor** | Finance Officer | Finance  Officer | Finance Officer | Finance  Officer |
| **Trigger** | Officer clicks  "Track  Defaulters" | Officer requests report | Officer initiates refund | Officer selects  "View Logs" |
| **Preconditions** | Officer is logged in | Officer is logged in and authorized | Refund request exists | Log records exist |
| **Post conditions** | List of defaulters is shown | Report is generated and shown | Refund is handled | Logs are displayed |
| **Normal Flow** | 1. Officer selects2.  System queries  DB3. Show list | 1. Officer requests2. System queries data3. Show report | 1. Officer opens refund2. Fetch request3. Approve/Reject | 1. Officer clicks logs2. Fetch logs3.  Show logs |
| **Alternative**  **Flow** | 3a. No defaulters → Show message | 3a. No records → Show "No data" | 3a. Officer rejects request | 2a. No logs found |
| **Exceptions** | DB  unreachable →  Error | Report fail →  Error shown | Invalid request → Error | Log system down → Error |
| **Business**  **Rules** | Only finance officers access | Authorized users only | Refunds must be valid | Logs are viewonly |
| **Assumptions** | DB is updated regularly | Format is predefined | Requests are authentic | Logs are properly maintained |

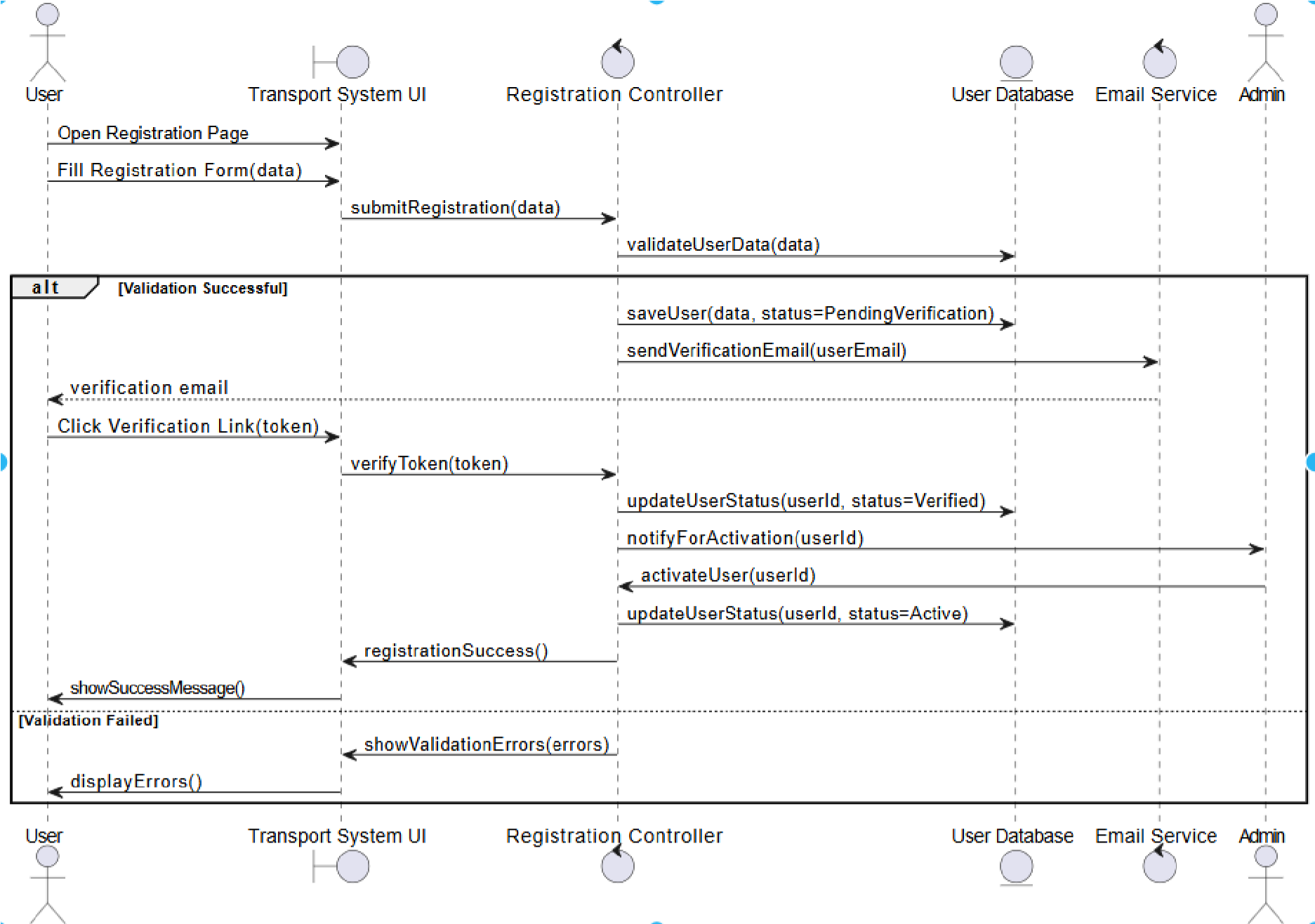
# 

# CHAPTER NO 4: SYSTEM SEQUENCE DIAGRAM

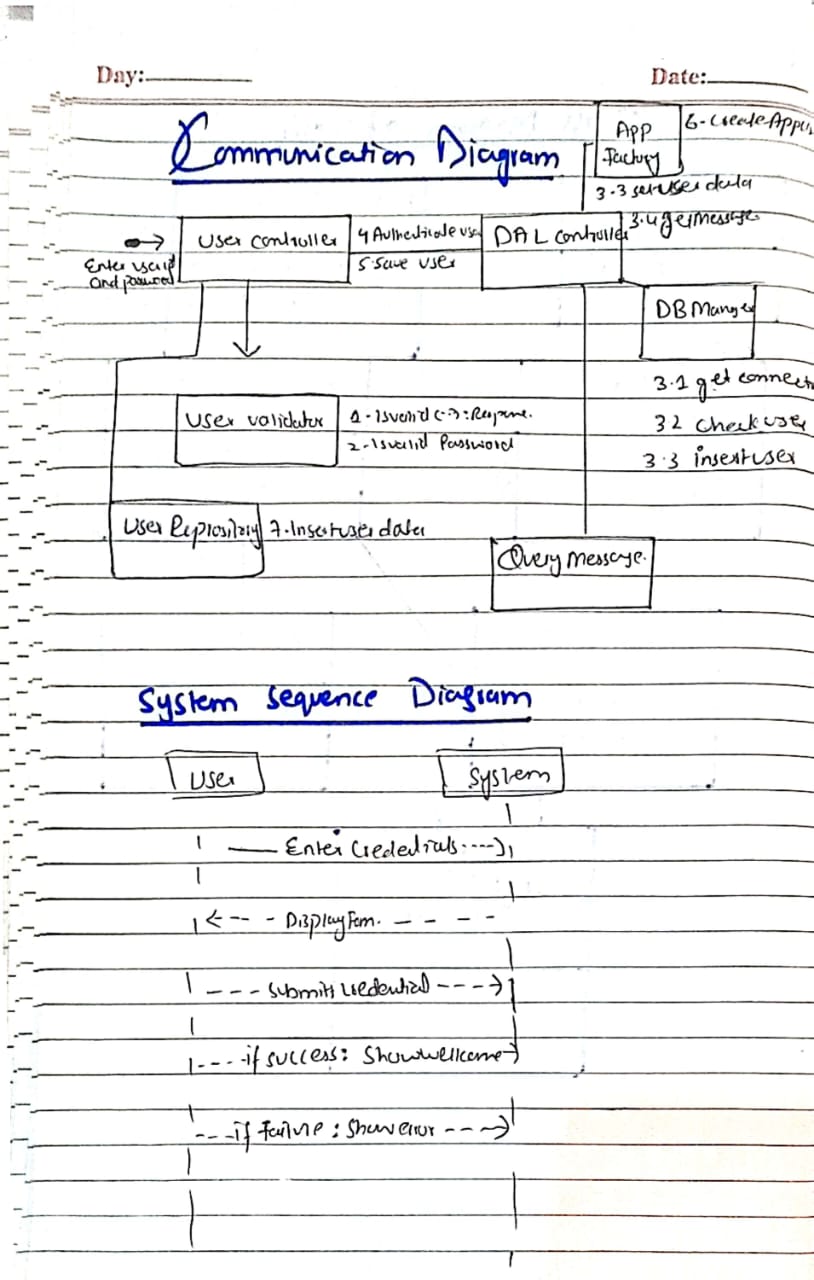
Sarina Amjad SP22-BSE-095

Warda Yousaf SP23-BSE-042

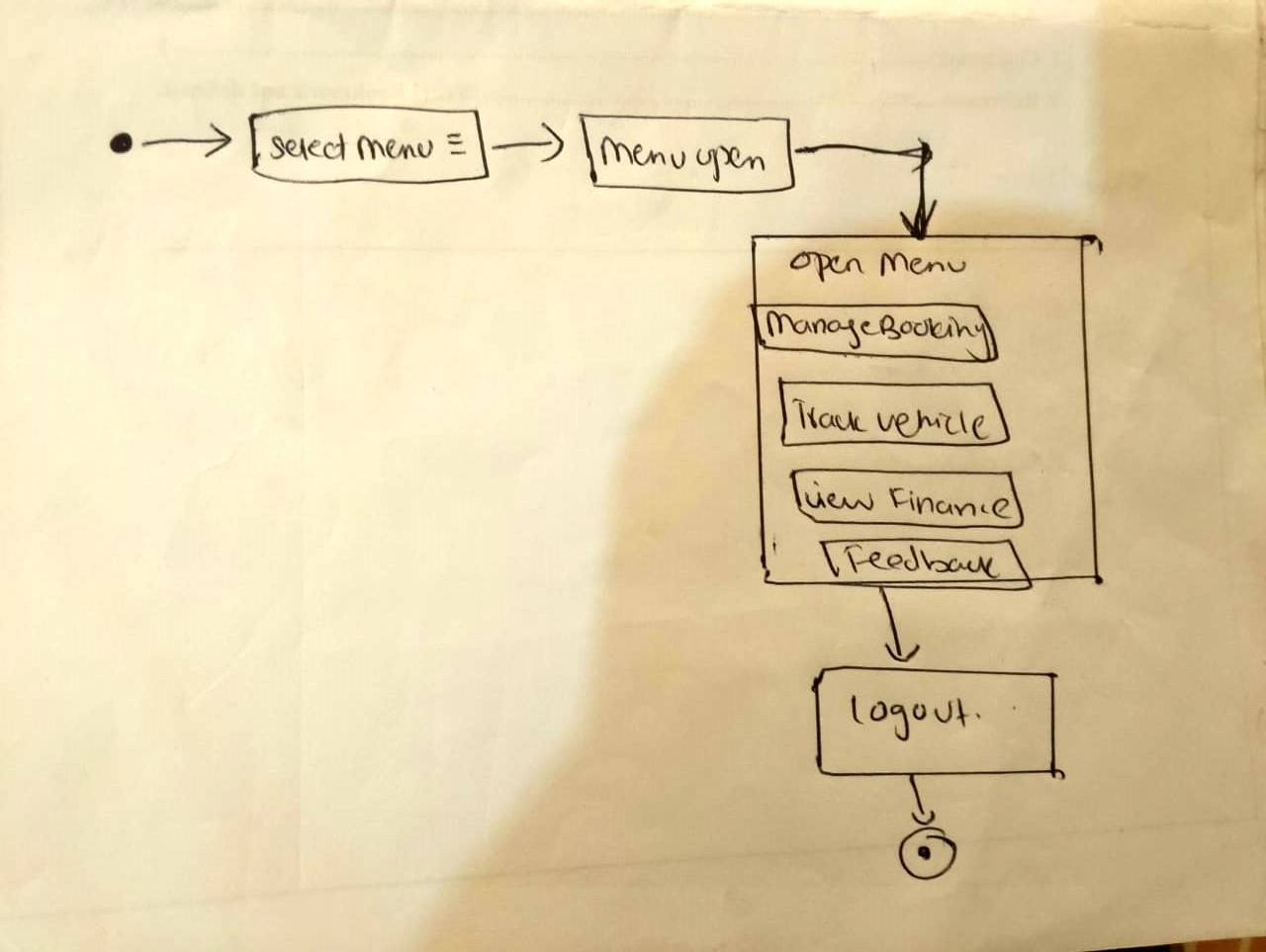
### SYSTEM SEQUENCE DIAGRAM : REGISTRATION



### SYSTEM SEQUENCE DIAGRAM: LOGIN

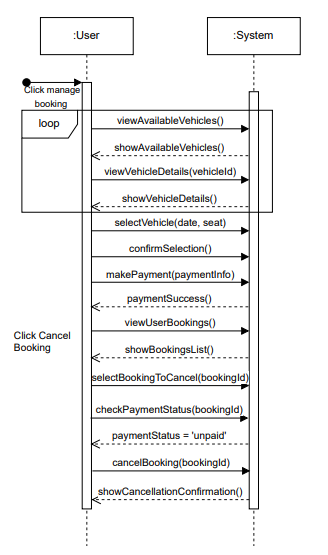


### SYSTEM SEQUENCE DIAGRAM: dashboard



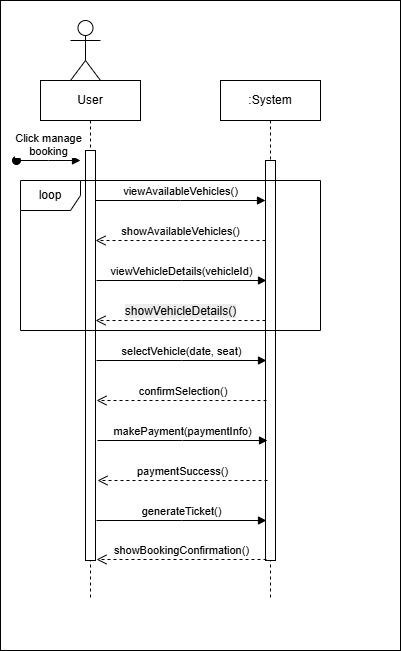
Alaina Khan SP23-BSE-069

### SYSTEM SEQUENCE DIAGRAM: MANAGE BOOKING



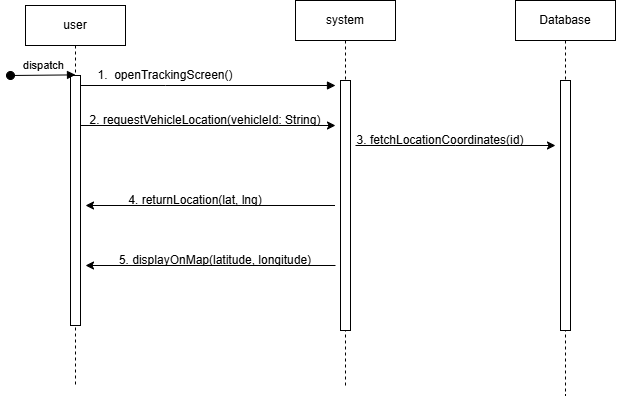
Fatima Khan SP23-BSE-102

### SYSTEM SEQUENCE DIAGRAM: MANAGE BOOKING



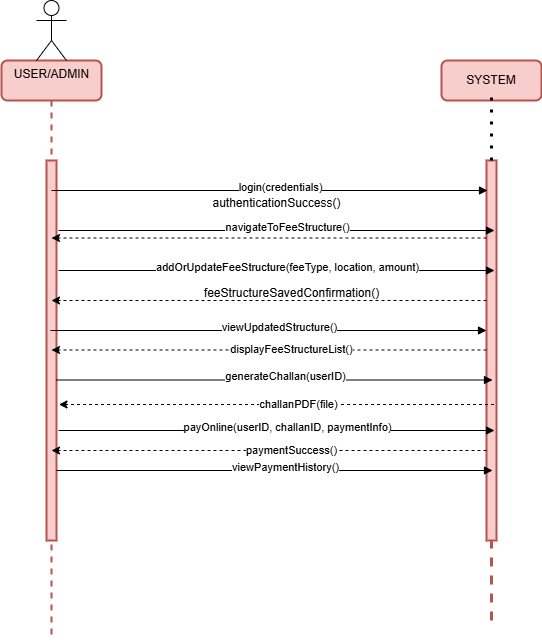
Maryam khan SP23-BSE-066

### SYSTEM SEQUENCE DIAGRAM: TRACK VEHICLE LOCATION



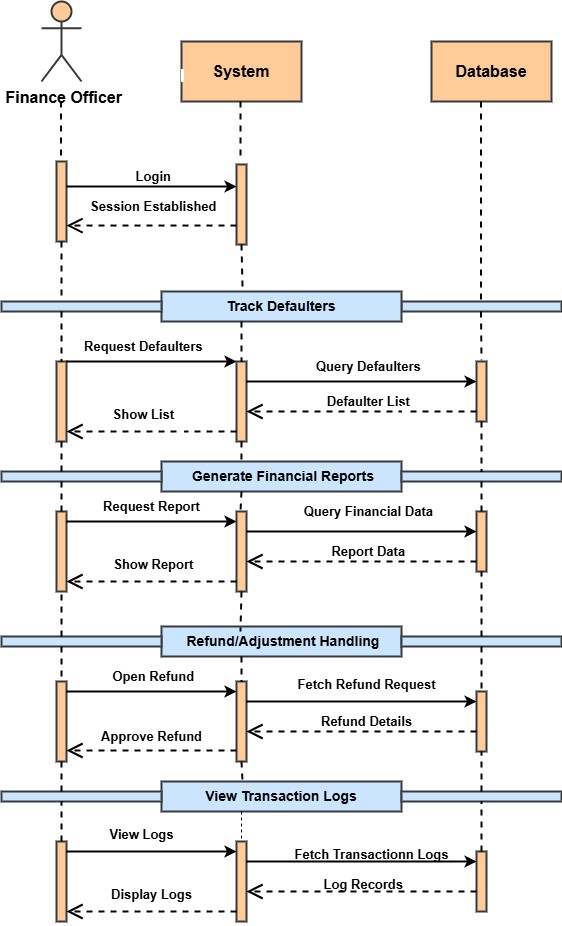
Kashmala Zeb SP23-BSE-048

### SYSTEM SEQUENCE DIAGRAM: FINANCE TRACKING



Tehreema Jilani SP23-BSE-041

### SYSTEM SEQUENCE DIAGRAM: FINANCE TRACKING

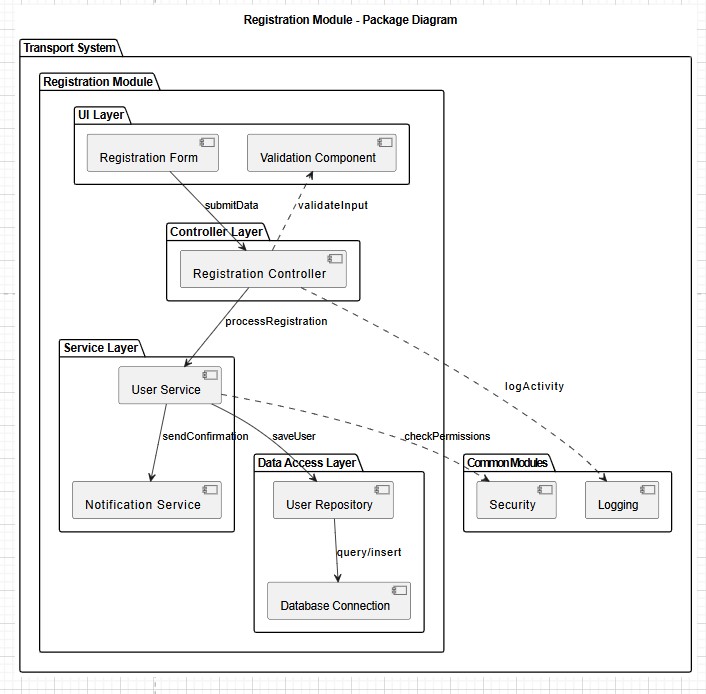


# CHAPTER NO 5: PACKAGE DIAGRAM

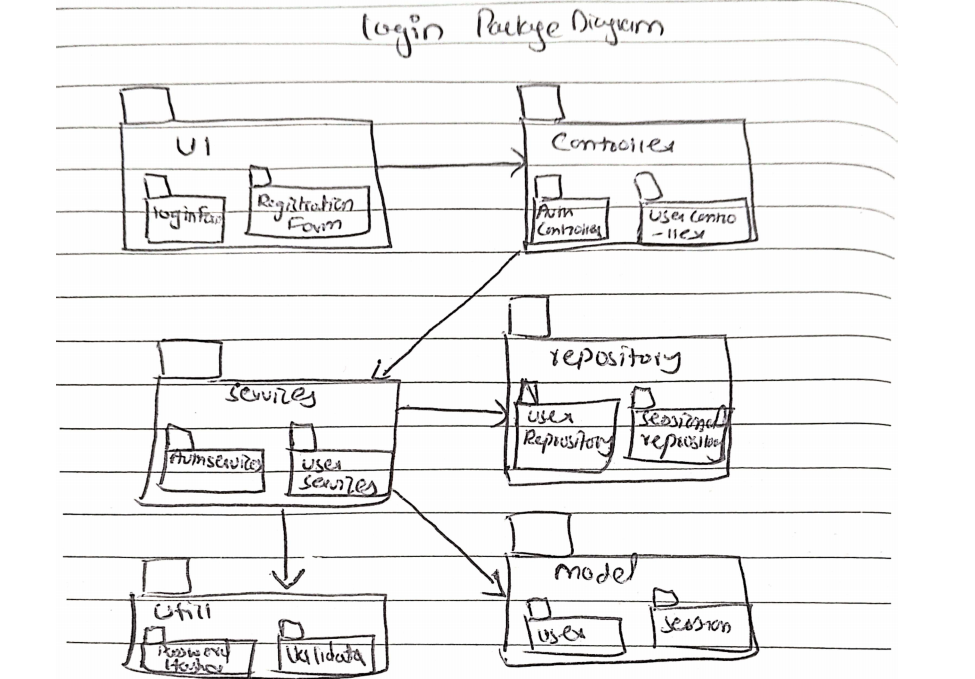
Sarina Amjad SP22-BSE-095

Warda Yousaf SP23-BSE-042

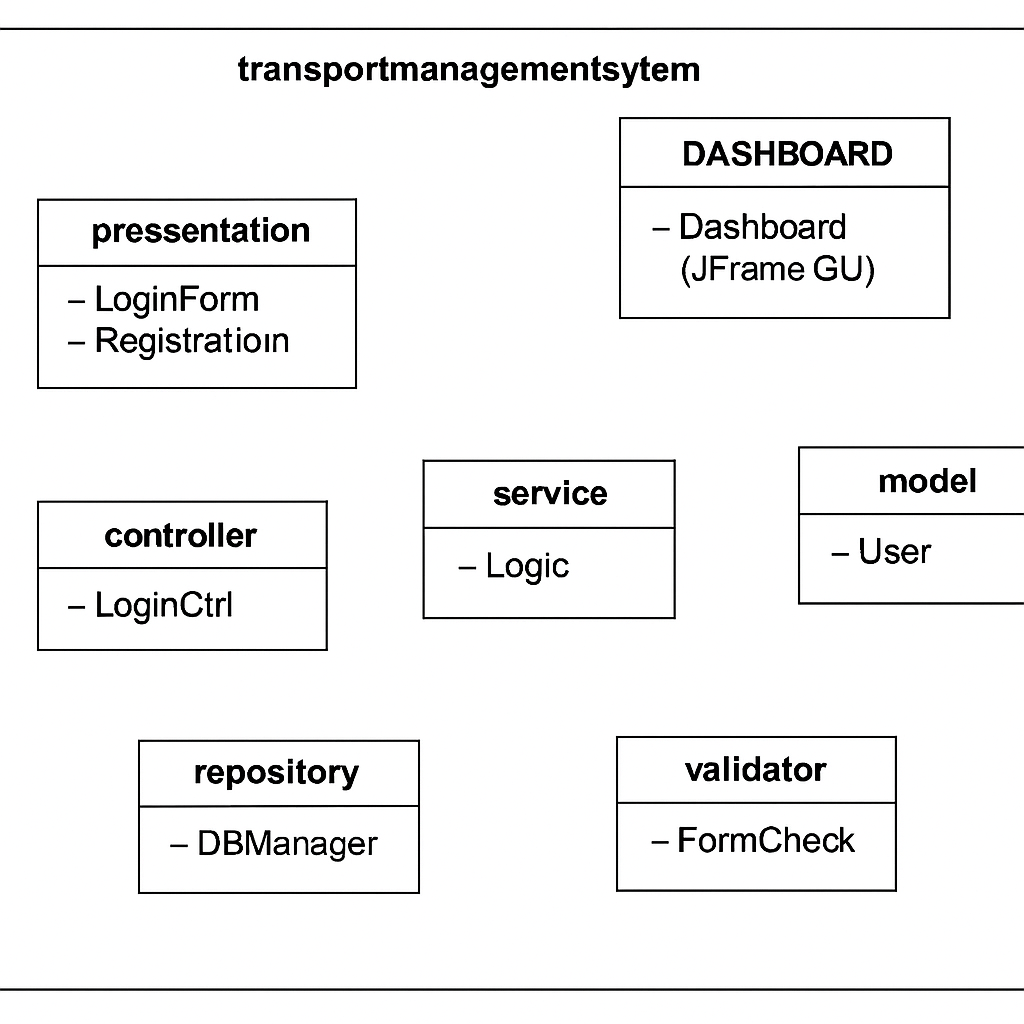
### PACKAGE DIAGRAM: REGISTRATION



### PACKAGE DIAGRAM: LOGIN

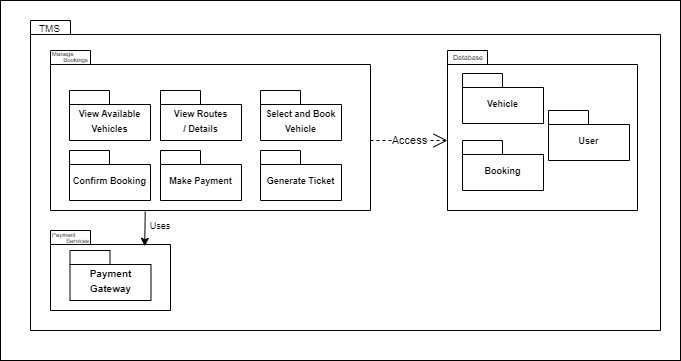


### PACKAGE DIAGRAM: Dashboard



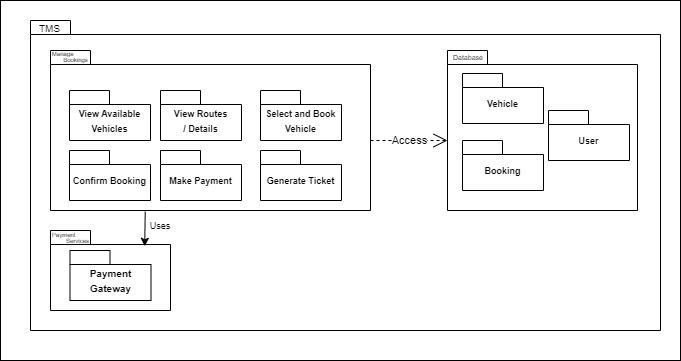
Alaina Khan SP23-BSE-069

### PACKAGE DIAGRAM: MANAGE BOOKING



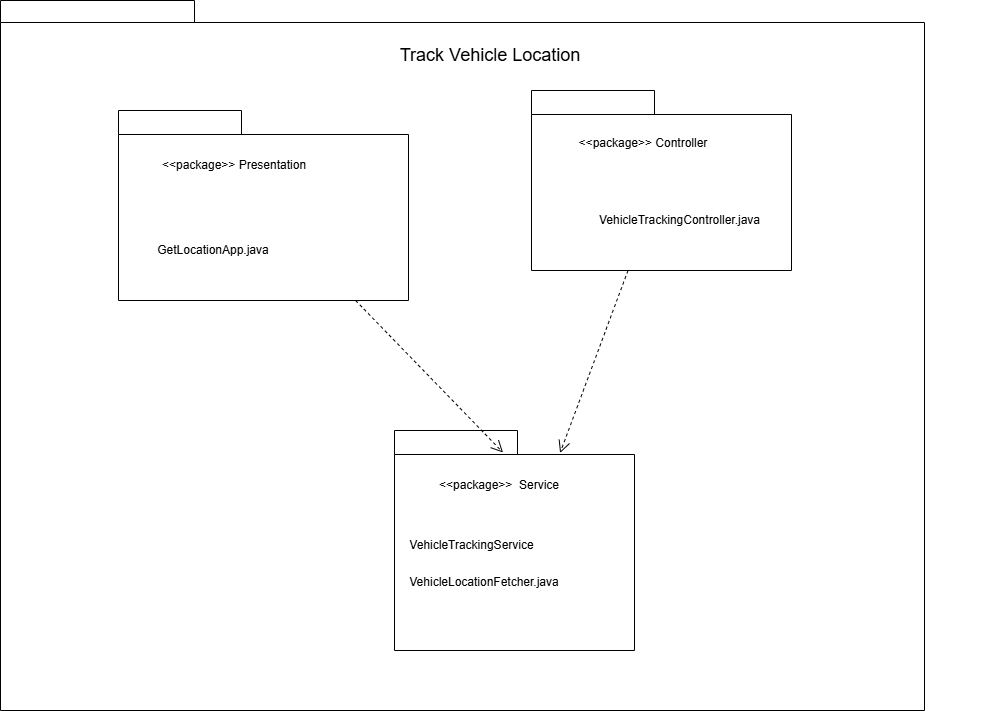
Fatima Khan SP23-BSE-102

### PACKAGE DIAGRAM: MANAGE BOOKING



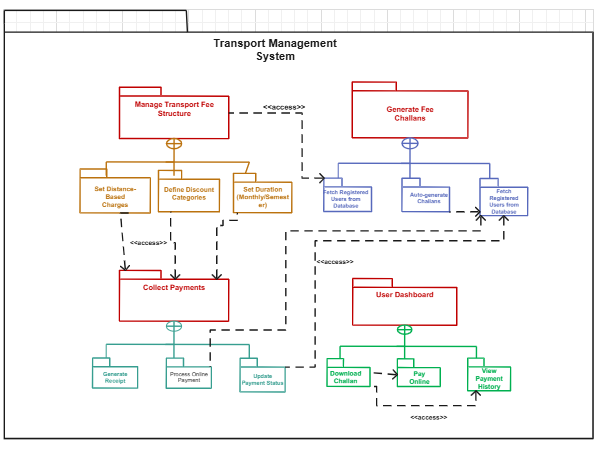
Maryam khan SP23-BSE-066

### PACKAGE DIAGRAM: TRACK VEHICLE LOCATION



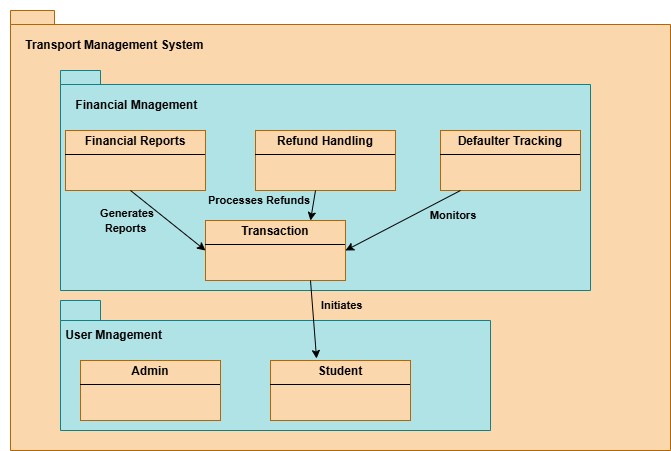
Kashmala Zeb SP23-BSE-048

### PACKAGE DIAGRAM: FINANCE TRACKING



Tehreema Jilani SP23-BSE-041

### PACKAGE DIAGRAM: FINANCE TRACKING

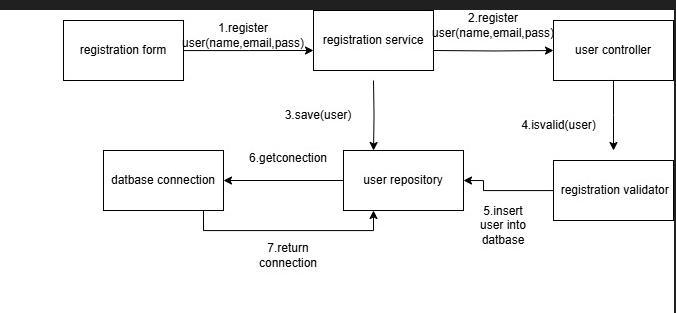


# CHAPTER NO 6: COMMUNICATION DIAGRAM

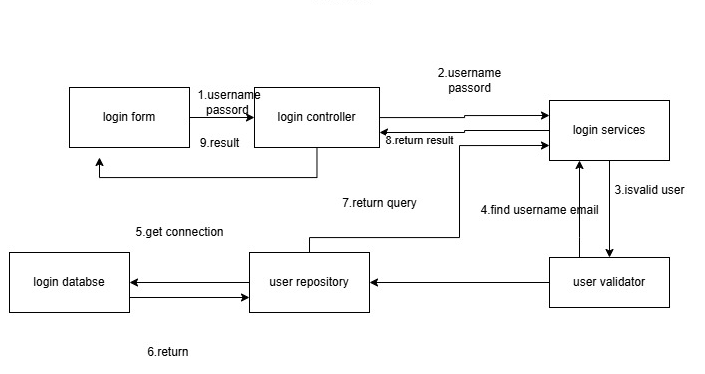
Sarina Amjad SP22-BSE-095

Warda Yousaf SP23-BSE-042

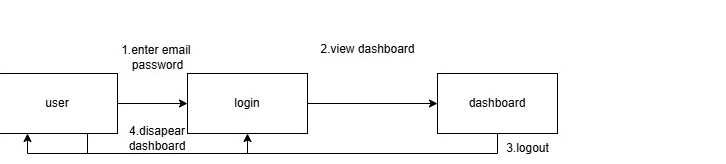
### COMMUNICATION DIAGRAM : Registartion



### COMMUNICATION DIAGRAM : LOGIN



### COMMUNICATION DIAGRAM : dashboard



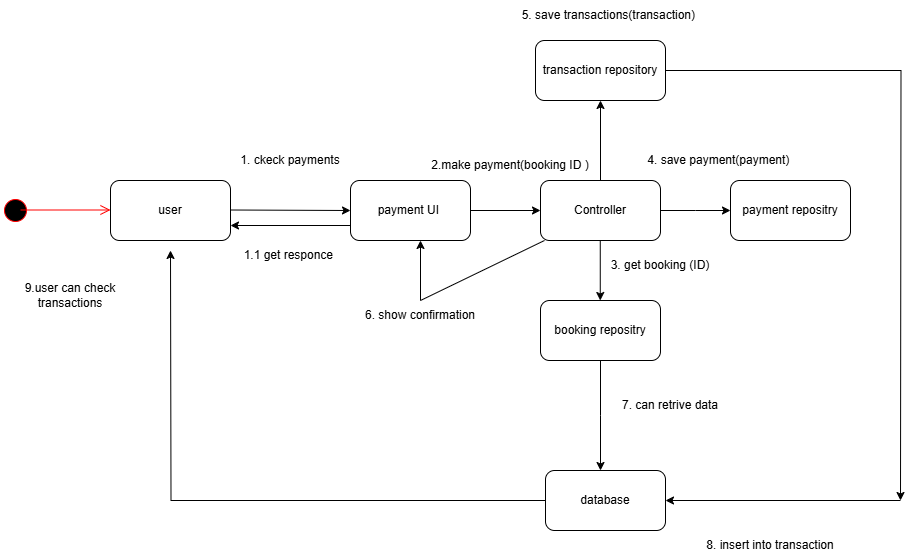
Alaina Khan SP23-BSE-069

### COMMUNICATION DIAGRAM: MANAGE BOOKING



Fatima Khan SP23-BSE-102

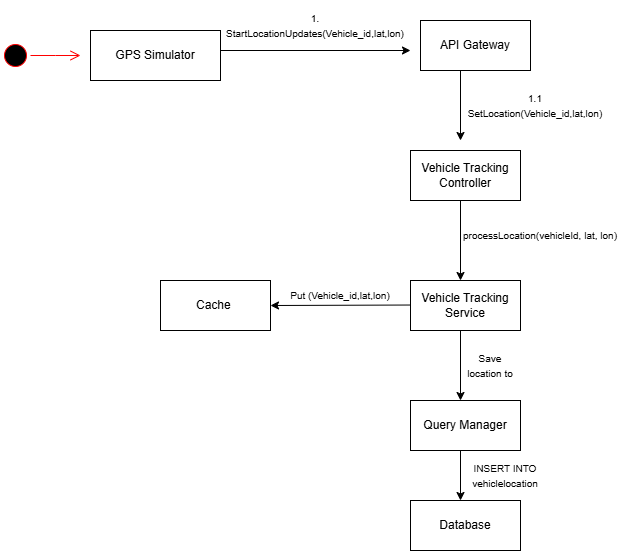
### COMMUNICATION DIAGRAM: MANAGE BOOKING



Maryam khan SP23-BSE-066

### COMMUNICATION DIAGRAM: TRACK VEHICLE LOCATION

**SET LOCATION**

****

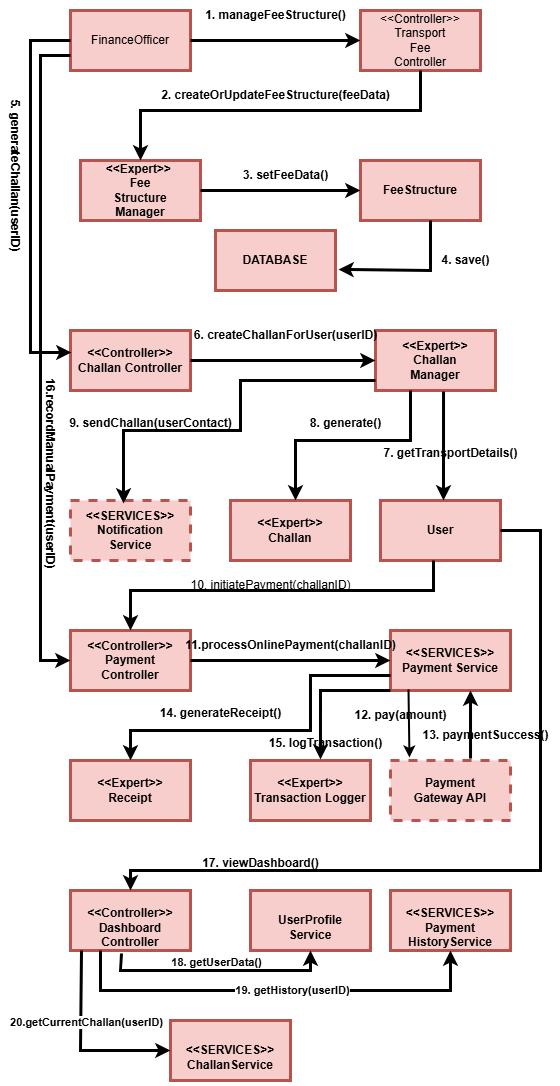
**GET LOCATION**

A screenshot of a computer

AI-generated content may be incorrect.

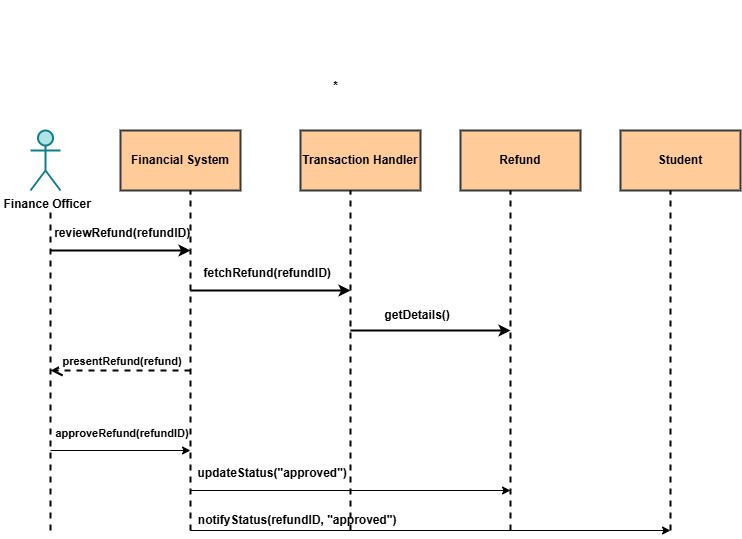
Kashmala Zeb SP23-BSE-048

### COMMUNICATION DIAGRAM: FINANCE TRACKING



Tehreema Jilani SP23-BSE-041

### SEQUENCE DIAGRAM: FINANCE TRACKING



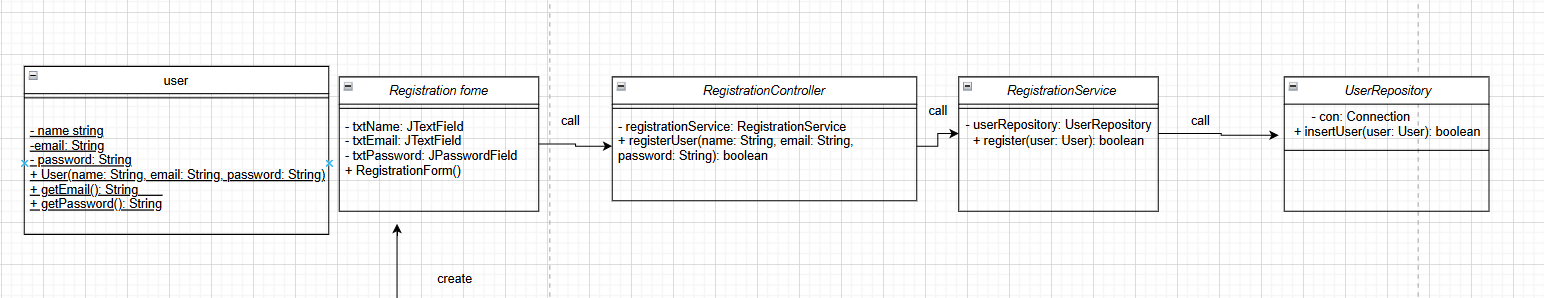
# CHAPTER NO 7: CLASS DIAGRAM

Sarina Amjad SP22-BSE-095

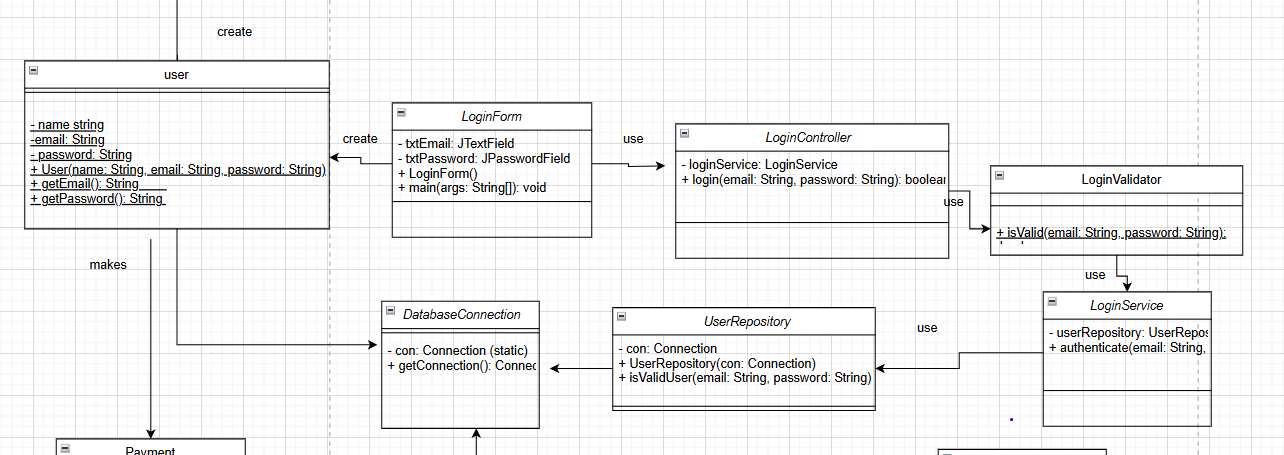
### CLASS DIAGRAM: REGISTRATION

Warda Yousaf SP23-BSE-042

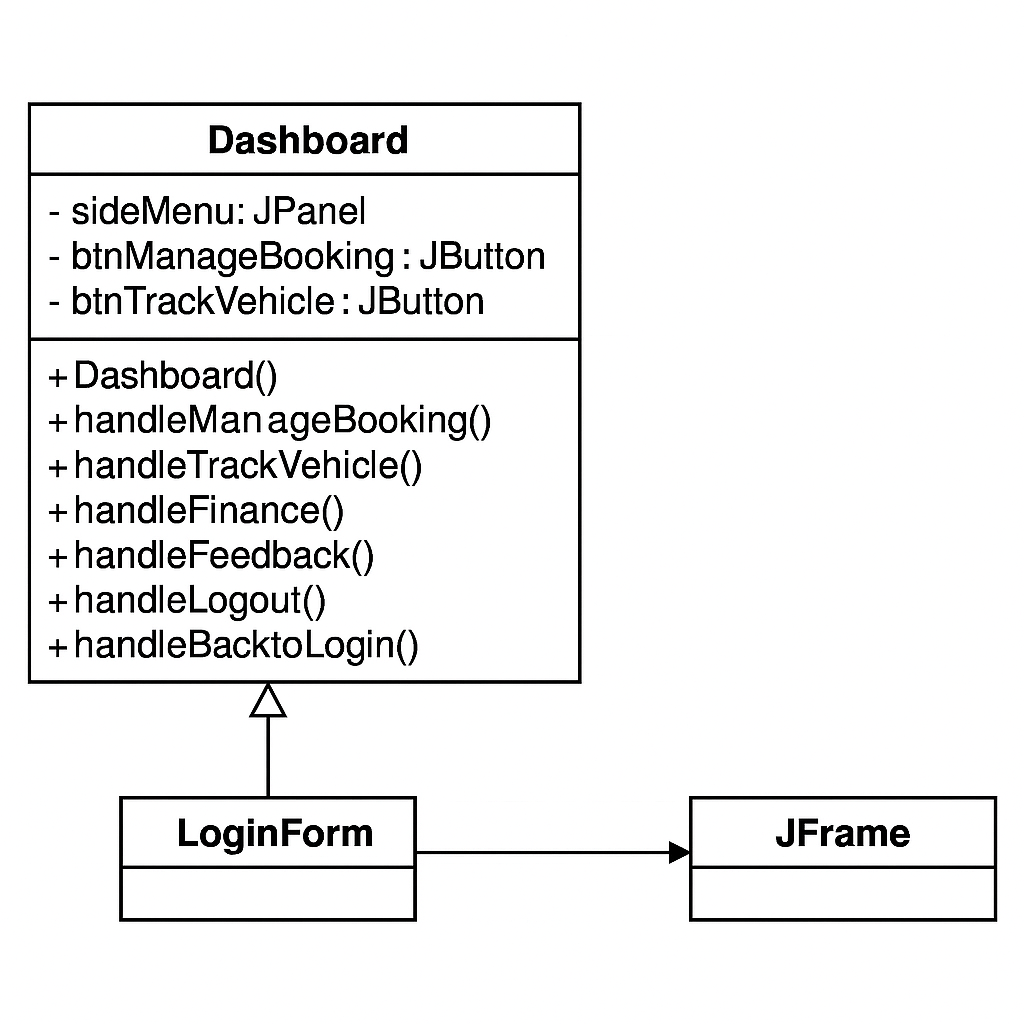
### CLASS DIAGRAM: registration



### CLASS DIAGRAM: Login

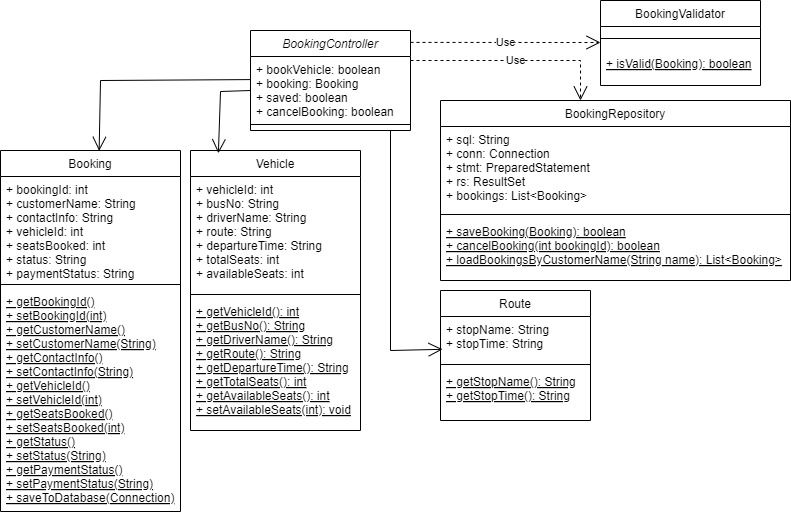


### CLASS DIAGRAM: Dashboard

****

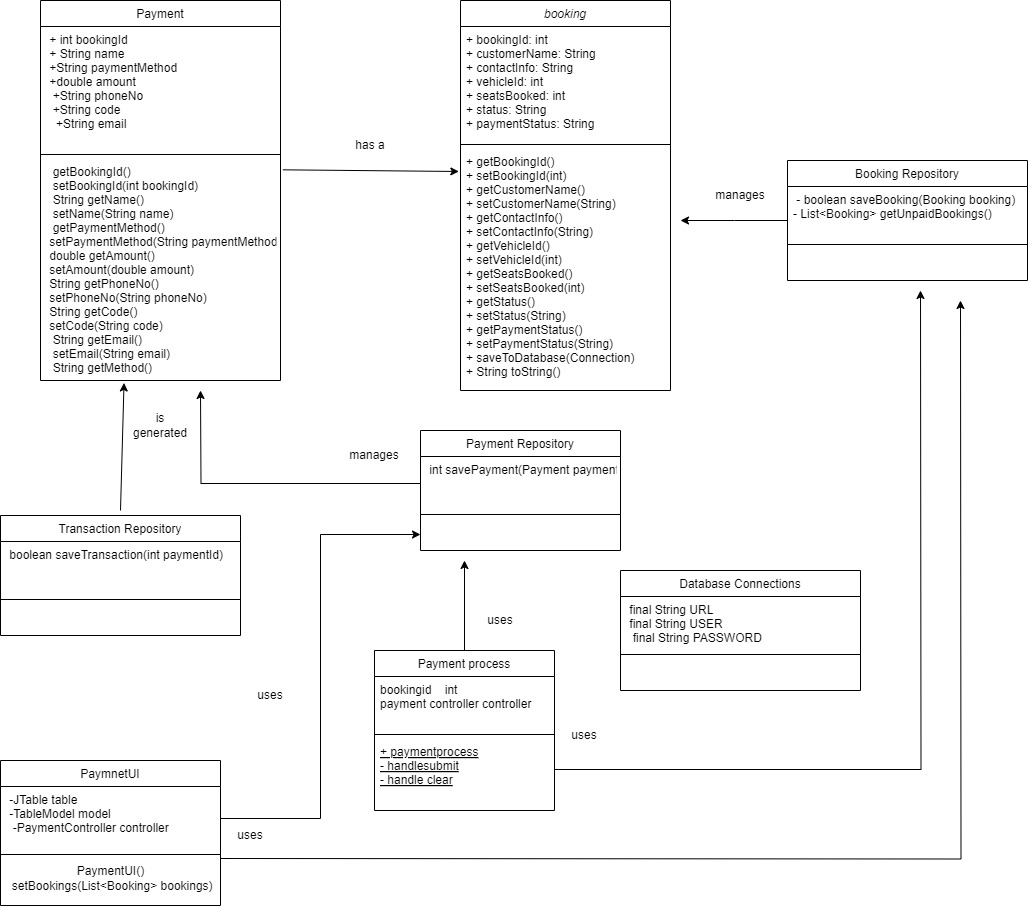
Alaina Khan SP23-BSE-069

### CLASS DIAGRAM: MANAGE BOOKING



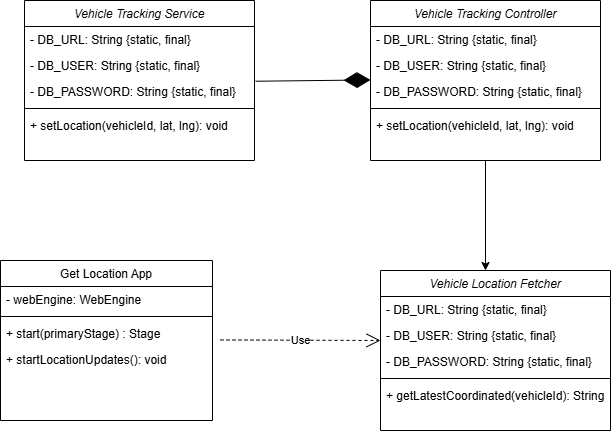
Fatima Khan SP23-BSE-102

### CLASS DIAGRAM: MANAGE BOOKING



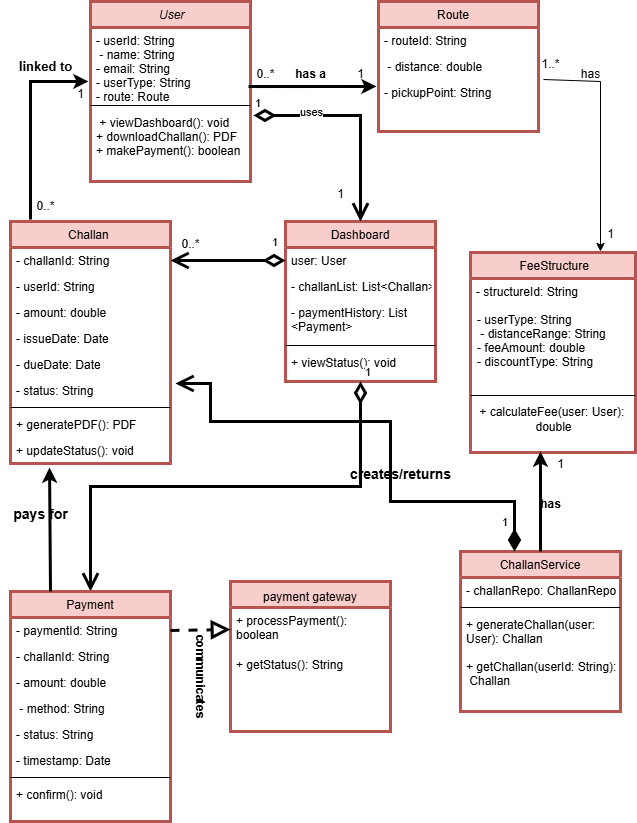
Maryam khan SP23-BSE-066

### CLASS DIAGRAM: TRACK VEHICLE LOCATION



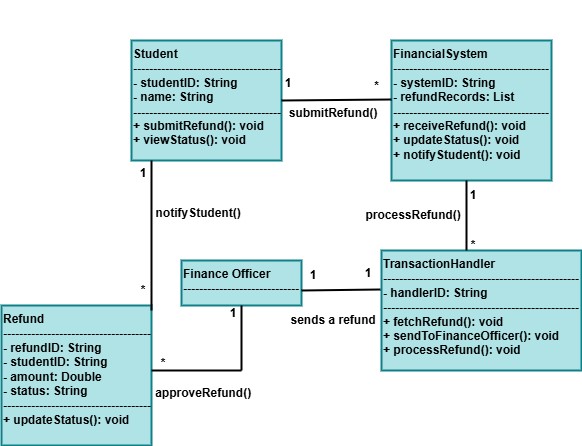
Kashmala Zeb SP23-BSE-048

### CLASS DIAGRAM: FINANCE TRACKING



Tehreema Jilani SP23-BSE-041

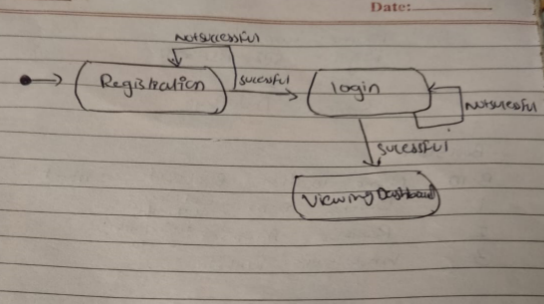
### CLASS DIAGRAM: FINANCE TRACKING



# CHAPTER NO 8: STATE TRANSITION DIAGRAM

Warda Yousaf SP23-BSE-042

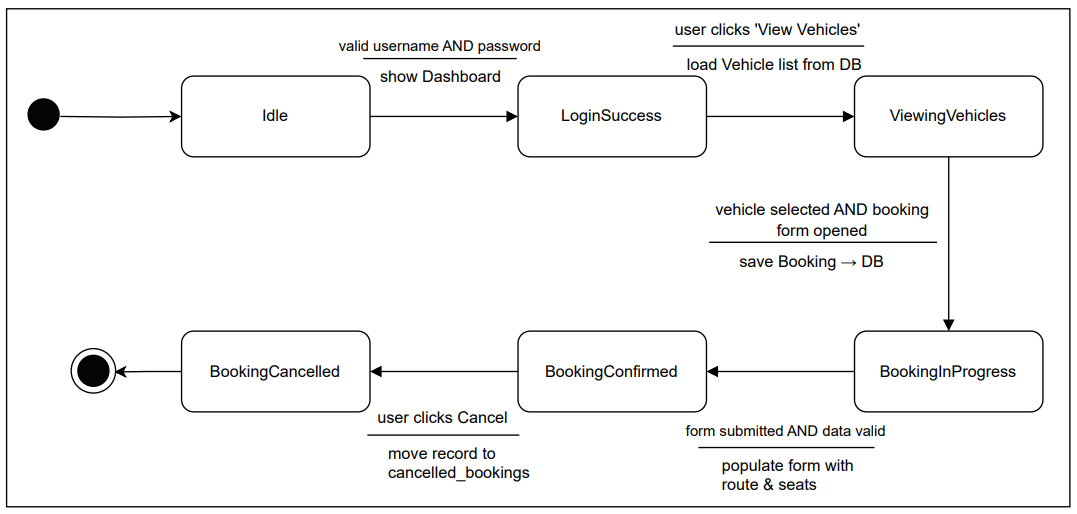
### STATE TRANSITION DIAGRAM: Registration ,Login, Dashboard



### STATE TRANSITION DIAGRAM: LOGIN & REGISTER

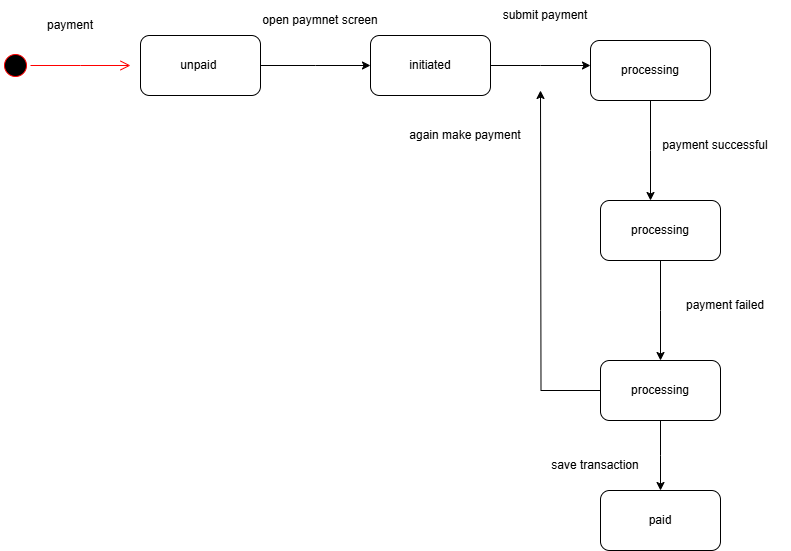
Alaina Khan SP23-BSE-069

### STATE TRANSITION DIAGRAM: MANAGE BOOKING



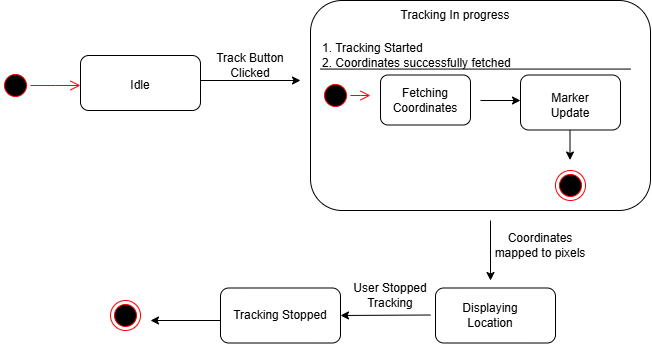
Fatima Khan SP23-BSE-102

### STATE TRANSITION DIAGRAM: MANAGE BOOKING



Maryam khan SP23-BSE-066

### STATE TRANSITION DIAGRAM: TRACK VEHICLE LOCATION



Kashmala Zeb SP23-BSE-048

### STATE TRANSITION DIAGRAM: FINANCE TRACKING

Tehreema Jilani SP23-BSE-041

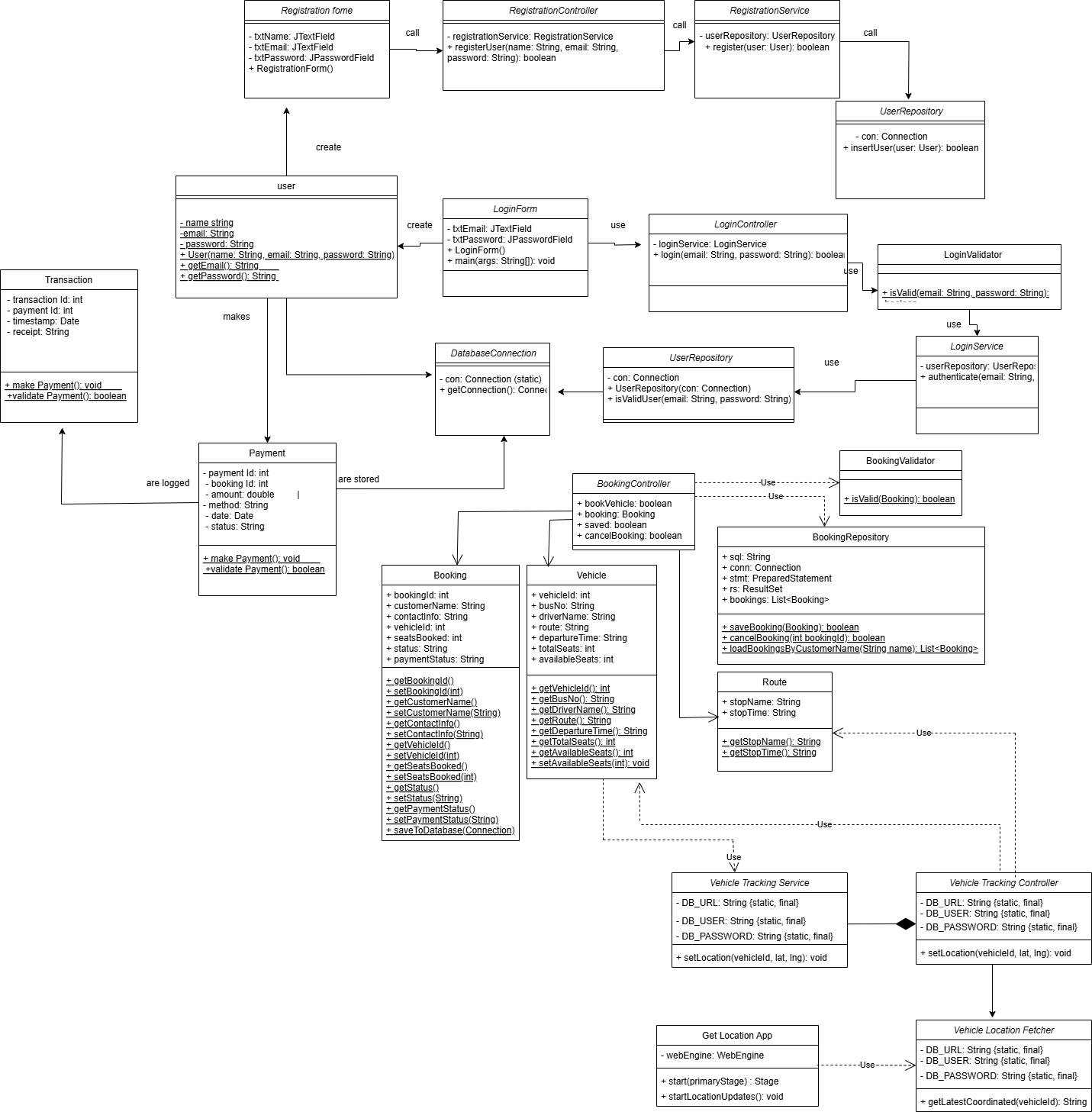
### STATE TRANSITION DIAGRAM: FINANCE TRACKING

Sarina Amjad SP22-BSE-095

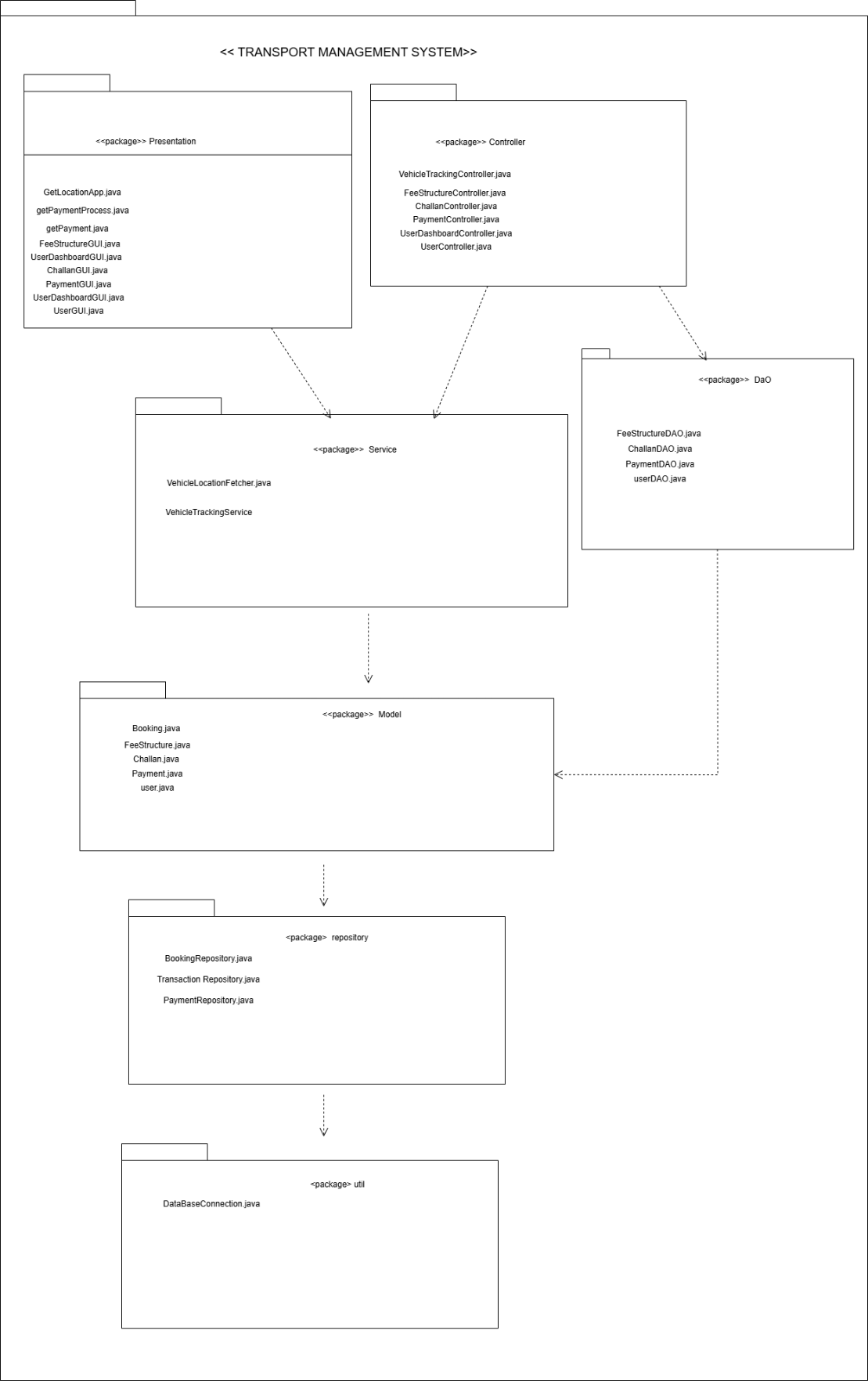
### STATE TRANSITION DIAGRAM:

# CHAPTER NO 9: ENTITY RELATIONSHIP DIAGRAM

# CHAPTER NO 10: CLASS DIAGRAM



# CHAPTER NO 11: PACKAGE DIAGRAM



# CHAPTER NO 12: CODING STANDARDS

**Coding Standards for TMS Manage Bookings Project**

We intend to follow a set of basic, consistent coding standards while developing the TMS (Transport Management System). These standards are aimed at making the code clear, readable, maintainable, and logically organized for both ourself and others who may review it.

Since this is a learning exercise, we will focus on fundamental conventions and practices suitable for beginner to intermediate-level projects.

1. **Naming Conventions** 
   * **Class Names:**

Class names will be written in **PascalCase** (each word starting with a capital letter).

Example: BookingController, VehicleService.

* + **Variables and Function Names:**

Variable and function names will be written in **camelCase** (the first word lowercase, each subsequent word capitalized).

Example: availableVehicles, validateBookingDetails().

* + **File Names:**

File names will be meaningful and consistent with the class or component they contain.

For React components, the file name will match the component name.

Example: ManageBookings.js, VehicleList.js.

1. **Indentation and Formatting** 
   * Code will be consistently indented using **4 spaces** per indentation level.
   * Each logical block of code (functions, conditionals, loops) will be separated by one blank line to enhance readability.
   * Opening braces { will be placed on the **same line** as the control structure or function declaration.

**Example:**

if (isAvailable) { confirmBooking();

}

1. **Commenting** 
   * Simple, clear comments will be included above functions and important code sections to describe their purpose.
   * Comments will also be added for any non-obvious logic or processes.
   * Comments will follow the single-line // format in JavaScript and React.

1. **Function Design** 
   * Functions will be **small and single-purpose**, focusing on doing one specific task.
   * Where applicable, reusable utility functions will be placed in a separate utility file.

1. **Error Handling** 
   * Basic error checking will be implemented where required, such as validating input values before processing them.
   * Error messages will be meaningful and assist in debugging.

1. **Code Structure and Organization** 
   * The codebase will follow a **modular structure**:
     + Components for UI elements.
     + Services for handling data and business logic.
     + Pages for complete views.
   * Related files and classes will be grouped logically in folders.

**Example Folder Structure:**

/src

/components

/services

/pages

/assets App.js index.js

1. **Consistency in Syntax and Practices** 
   * Consistent use of **const** and **let** for variable declarations in JavaScript.
   * Consistent arrow function syntax where appropriate.
   * Removal of any unused code or variables during cleanup.
   * Keeping code aligned with modern JavaScript (ES6+) and React conventions.

1. **Object-Oriented Practices** 
   * The project will implement **object-oriented programming (OOP) concepts** in JavaScript where suitable, particularly for services and controllers.
   * Classes will have clearly defined attributes (properties) and methods (functions) based on their responsibilities.