INTERCITY EXPRESS

Welcome to the Railway Reservation System project! This software engineering initiative is dedicated to revolutionizing the way passengers book train tickets, making the process more efficient, accessible, and user-friendly. Utilizing the latest web technologies including HTML, CSS, and JavaScript, we have developed a dynamic and interactive platform that allows users to search for trains, check seat availability, and make reservations effortlessly.

The core of our system is powered by MySQL, a robust database management system that ensures secure and efficient handling of all user data, train schedules, and booking records. This integration not only guarantees data integrity and security but also enhances the overall performance and reliability of the system.

Our platform offers a wide range of features designed to improve the user experience. Customers can enjoy real-time updates on seat availability, detailed train schedules, and a streamlined booking process that minimizes hassle and maximizes convenience. For administrators, the system provides comprehensive tools to manage train schedules, monitor reservations, and generate detailed reports, thereby enhancing operational efficiency.

From a technical perspective, the project is highly feasible. HTML, CSS, and JavaScript ensure a responsive and user-friendly front-end experience, while MySQL provides a scalable and reliable backend for data management. Key functionalities include ticket reservation, booking, and cancellation, as well as user account management and payment processing. The system is designed to handle concurrent users efficiently, ensuring smooth operation even during peak times. Overall, our Railway Reservation System is not only technically robust but also user-centric, making it an ideal solution for modern railway ticketing needs.

FUNCTIONAL REQUIREMENTS:

•The customers should be able to register themselves on the portal. Agent registration should be provided through admin.

•Admin, Agent, Customer should be able to login.

•Users should be able to search trains according to the routes, dates, etc. with proper data filtering.

•Customers and agents should be able to book and cancel their tickets.

•Admin must be able to make all the administrative changes in the data like changing schedules of trains, updating cancelled trains, maintaining, and updating customers' and users' data, driver's data, driver's allocation, etc.

•Agent's bookings and commission according to the sales must be visible to the agent.

•Coach maintenance details should be kept separately and monitored for regular maintenance.

•Users should be able to view and manage their profile information. Users must have access to their booking history and be able to print or download their tickets.

• The system must provide real-time information on seat availability for the selected train and class.

NON-FUNCTIONAL REQUIREMENTS:

•Agents must be allowed to prebook tickets 1 month before. Customers should be allowed to book tickets only from 1 week before the trip date.

•The system must handle up to 10,000 concurrent users without performance degradation.

•Search results should be displayed within 3 seconds.

•The system should be scalable to accommodate future growth in the number of users and data volume.

•The system must ensure secure handling of user data, including encryption of sensitive information such as passwords and payment details.

•The user interface should be intuitive and easy to navigate for users of all technical skill levels.

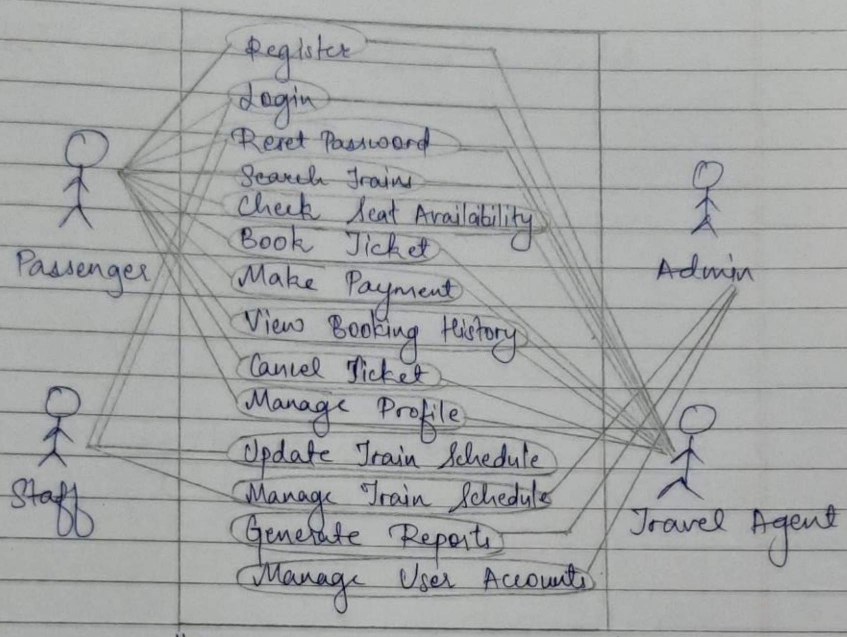
•It should be accessible from various devices, including desktops, tablets, and smartphones.

•The system must have an uptime of 99.9%, ensuring it is available to users at all times. It should include mechanisms for data backup and recovery to prevent data loss.

•Data in reports generated should be in proper format according to the filtering.

- Even if number of users increase suddenly and operate the system at the same time, the system should work efficiently. (Spike testing should be performed.)

* USE CASE DIAGRAM :-

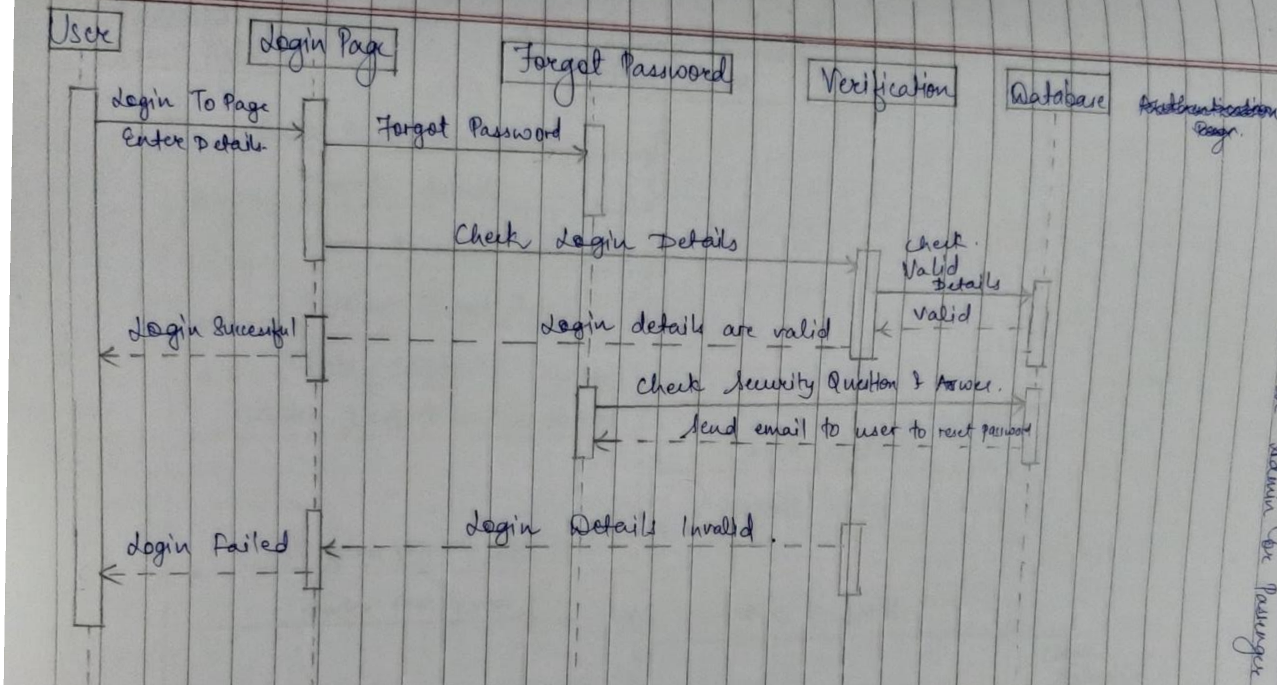


"RAILWAY RESERVATION SYSTEM
INTERCITY EXPRESS"

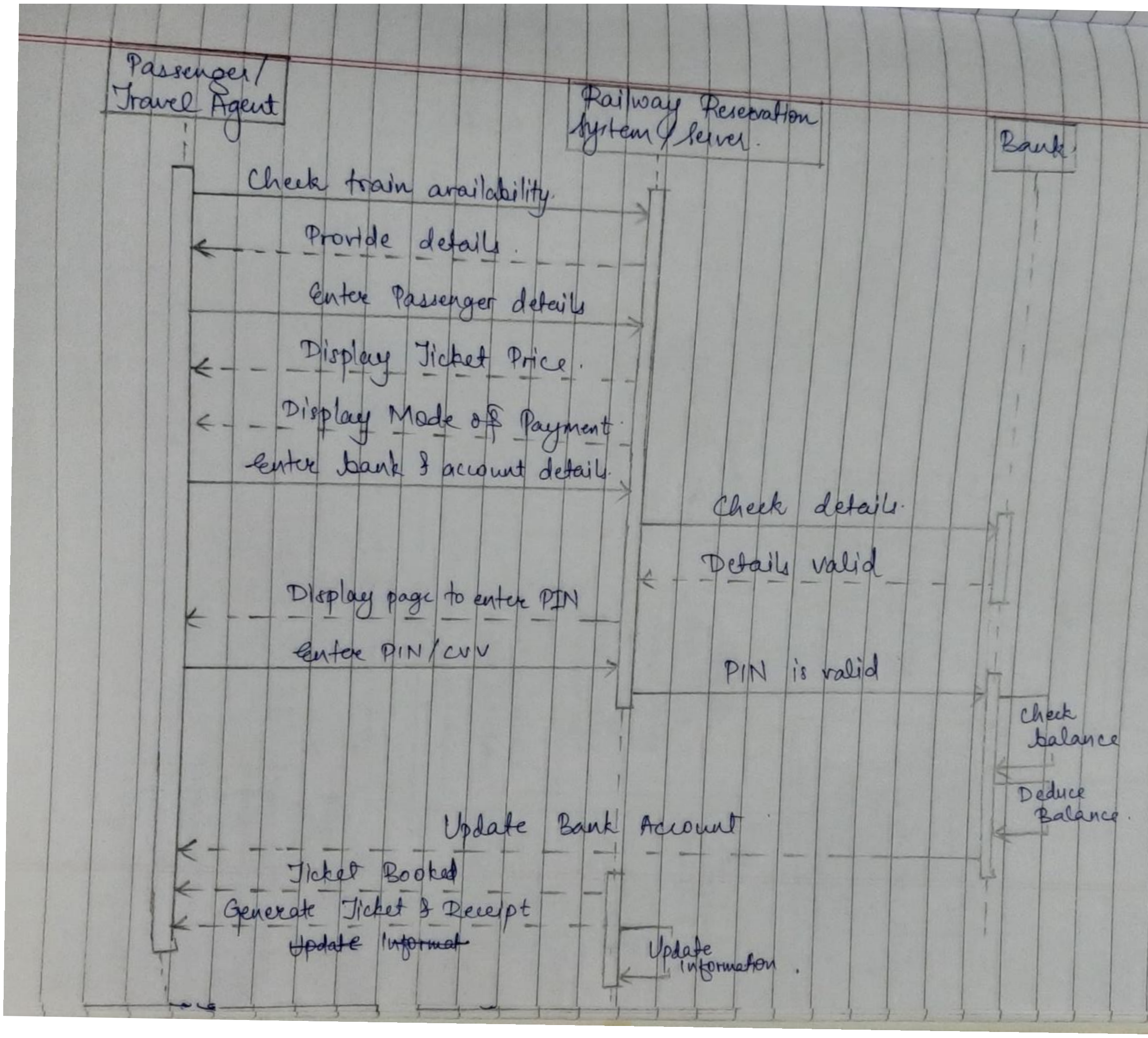
Actors : Admin, Passenger, Staff, Travel Agent.

Pre-conditions : The system should be operational & connected to Internet.

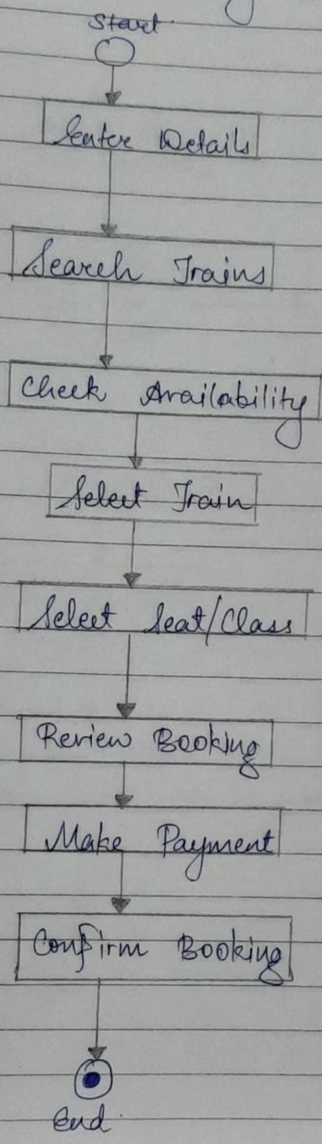
Post-conditions : 1) All changes made should be saved & database should be updated accordingly.
2) Payment receipt should be generated online after ticket is booked.



Admin or Passenger or Travel Agent/Staff.

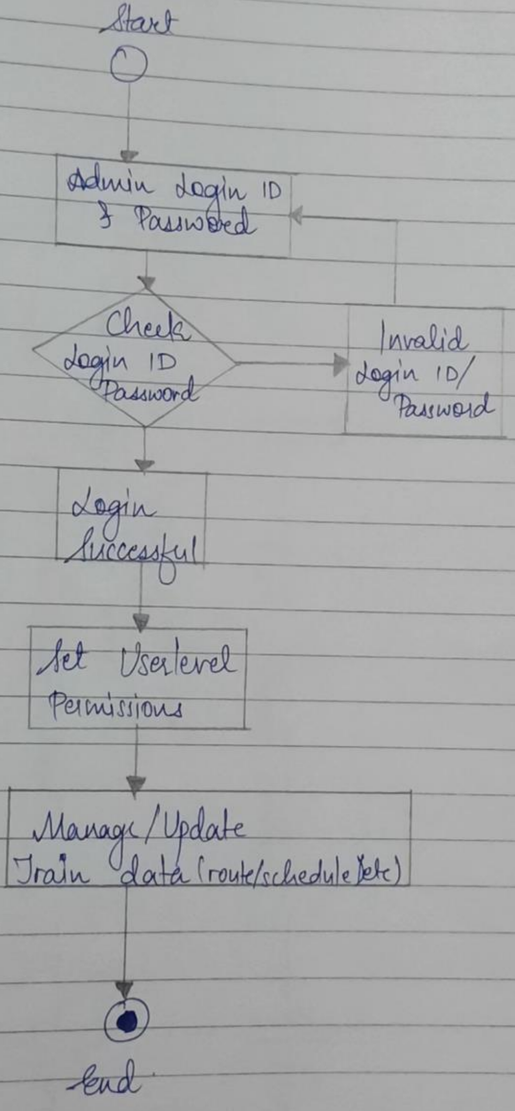

Passenger/
Travel Agent
Railway Reservation
System/Server.
Bank.
Check train availability.
Provide details.
Enter Passenger details
Display Ticket Price.
Display Mode of Payment.
enter bank & account details.
Check details.
Details valid
Display page to enter PIN
Enter PIN/CVV
PIN is valid
Check
balance
Deduce
Balance.
Update Bank Account
Ticket Booked
Generate Ticket & Receipt
~~Update Informat~~
Update
Information.

* ACTIVITY DIAGRAM :- Booking a Ticket.

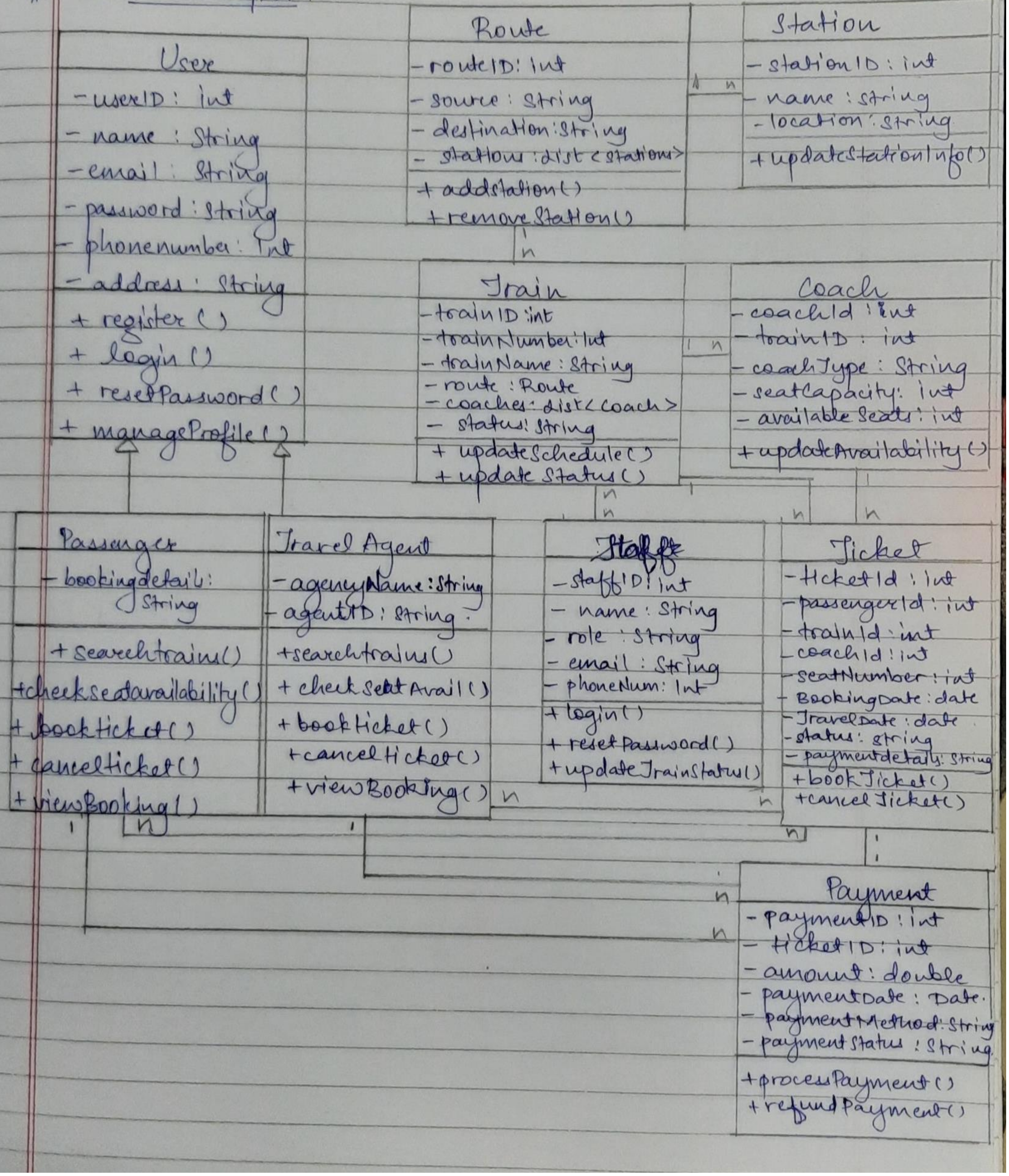


Assumption: The user has already registered/logged into the system.

* ACTIVITY DIAGRAM :- For Admin.


Start
Admin Login ID & Password
Check Login ID Password
Invalid Login ID/ Password
Login Successful
Set Userlevel Permissions
Manage/Update Train data (route/schedule)etc)
End

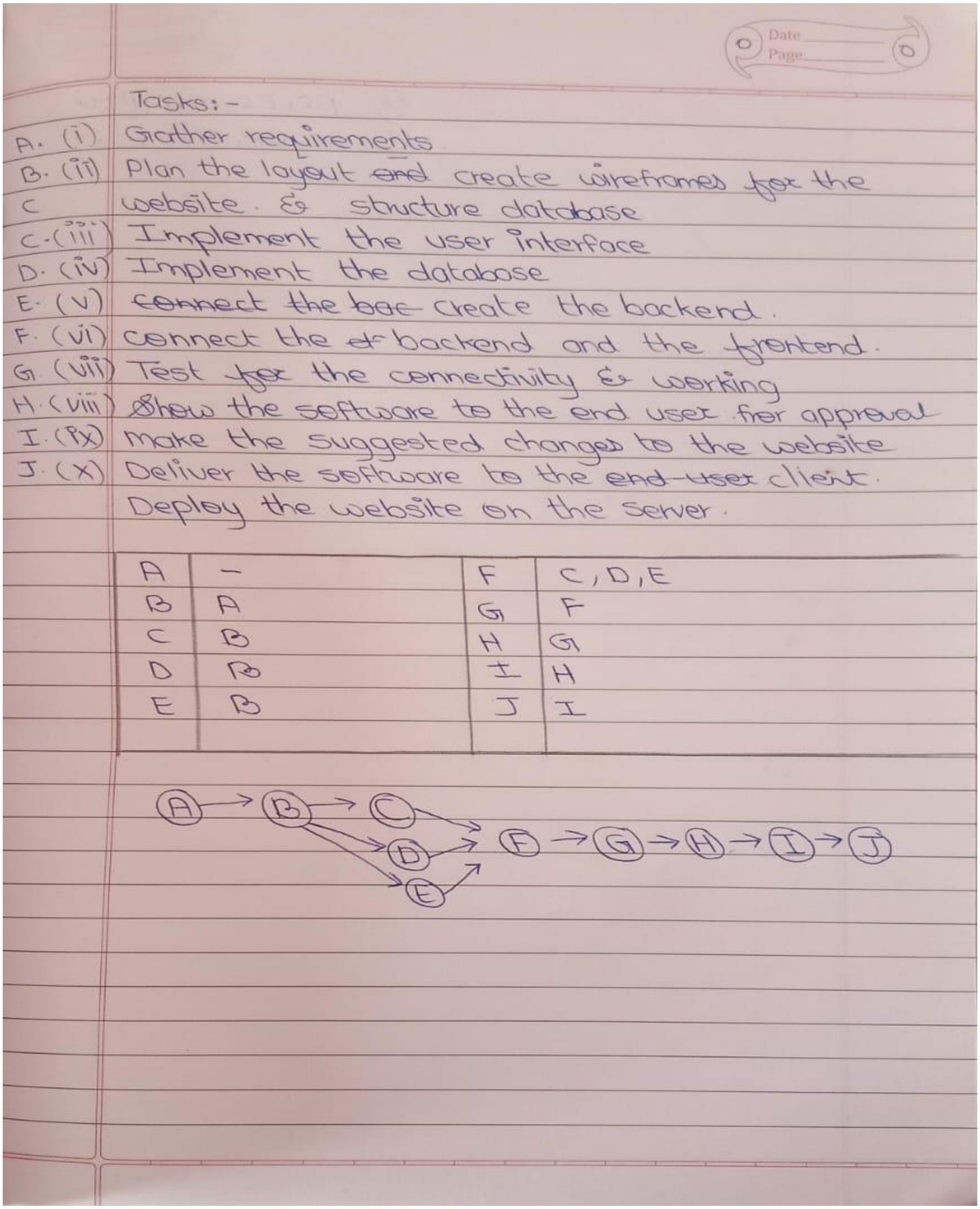
* CLASS DIAGRAM:



Tasks and Dependencies

1. **Requirement Analysis**
    - **Dependencies**: None
    - **Description**: Gather and document the requirements from stakeholders, including features, functionalities, and constraints.
2. **System Design**
    - **Dependencies**: Requirement Analysis
    - **Description**: Create the system architecture, including class diagrams, use case diagrams, and database schema design.
3. **Database Setup**
    - **Dependencies**: System Design
    - **Description**: Set up the MySQL database, including creating tables, relationships, and initial data population.
4. **Front-end Development**
    - **Dependencies**: System Design
    - **Description**: Develop the user interface using HTML, CSS, and JavaScript, ensuring it is responsive and user-friendly.
5. **Back-end Development**
    - **Dependencies**: System Design, Database Setup
    - **Description**: Develop the server-side logic using a suitable programming language (e.g., Node.js, PHP), including APIs for interacting with the database.
6. **Integration of Front-end and Back-end**
    - **Dependencies**: Front-end Development, Back-end Development
    - **Description**: Integrate the front-end interface with the back-end logic to ensure seamless data flow and interaction.
7. **User Authentication Module**
    - **Dependencies**: Database Setup, Back-end Development
    - **Description**: Implement user registration, login, and password reset functionalities, ensuring secure authentication and authorization mechanisms.
8. **Ticket Booking Module**
    - **Dependencies**: Database Setup, Back-end Development, User Authentication Module
    - **Description**: Implement functionalities for searching trains, checking seat availability, booking tickets, and generating booking confirmations.

9. **Payment Gateway Integration**
    - **Dependencies**: Ticket Booking Module
    - **Description**: Integrate a secure payment gateway to process payments for ticket bookings.
10. **Admin Panel Development**
    - **Dependencies**: Database Setup, Back-end Development
    - **Description**: Develop an admin panel for managing train schedules, routes, user accounts, and generating reports.
11. **Staff and Travel Agent Modules**
    - **Dependencies**: User Authentication Module, Admin Panel Development
    - **Description**: Implement functionalities specific to railway staff and travel agents, including assisting passengers, updating train statuses, and booking tickets on behalf of passengers.
12. **Testing**
    - **Dependencies**: Integration of Front-end and Back-end, User Authentication Module, Ticket Booking Module, Admin Panel Development, Staff and Travel Agent Modules
    - **Description**: Conduct thorough testing of the system, including unit testing, integration testing, system testing, and user acceptance testing to ensure the system meets the specified requirements and is free of bugs.
13. **Deployment**
    - **Dependencies**: Testing
    - **Description**: Deploy the system to a live server, ensuring all configurations are correctly set up and the system is accessible to users.
14. **User Training and Documentation**
    - **Dependencies**: Deployment
    - **Description**: Provide training sessions for end-users and create comprehensive documentation, including user manuals and technical documentation.
15. **Maintenance and Support**
    - **Dependencies**: Deployment
    - **Description**: Provide ongoing maintenance and support to ensure the system remains operational and up to date with any required changes or enhancements.



Tasks:-

A. (i) Gather requirements

B. (ii) Plan the layout ~~and~~ create wireframes for the

C website. & structure database

C.(iii) Implement the user interface

D. (iv) Implement the database

E. (v) ~~connect the bac~~ create the backend.

F. (vi) connect the ~~ef~~ backend and the frontend.

G (vii) Test for the connectivity & working

H.(viii) Show the software to the end user for approval

I.(ix) make the suggested changes to the website

J.(x) Deliver the software to the ~~end user~~ client.

Deploy the website on the server.

A	–	F	C, D, E
B	A	G	F
C	B	H	G
D	B	I	H
E	B	J	I

TEST CASES:

Test Case 1: User Registration

- **Test Case ID**: TC001
- **Description**: Verify that a new user can register successfully.
- **Preconditions**: User is on the registration page.
- **Steps**:
  -Enter valid details in the registration form (name, email, password, phone number, address).
  -Click on the "Register" button.
- **Expected Result**: User should be registered successfully and redirected to the login page with a success message.

Test Case 2: User Login

- **Test Case ID**: TC002
- **Description**: Verify that a registered user can log in successfully.
- **Preconditions**: User is registered and is on the login page.
- **Steps**:
  -Enter valid email and password.
  -Click on the "Login" button.
- **Expected Result**: User should be logged in successfully and redirected to the dashboard.

Test Case 3: Search Trains

- **Test Case ID**: TC003
- **Description**: Verify that a user can search for trains between two stations.
- **Preconditions**: User is logged in.
- **Steps**:
  -Enter source and destination stations.
  -Select travel date.
  -Click on the "Search" button.
- **Expected Result**: A list of trains available between the selected stations on the specified date should be displayed.

Test Case 4: Check Seat Availability

- **Test Case ID**: TC004

- **Description**: Verify that a user can check seat availability for a selected train.
- **Preconditions**: User is logged in and has searched for trains.
- **Steps**:
  -Select a train from the search results.
  -Click on the "Check Availability" button.
- **Expected Result**: Seat availability for the selected train should be displayed.

Test Case 5: Book Ticket

- **Test Case ID**: TC005
- **Description**: Verify that a user can book a ticket successfully.
- **Preconditions**: User is logged in and has checked seat availability.
- **Steps**:
  -Select the desired seats.
  -Enter passenger details.
  -Proceed to payment and complete the transaction.
- **Expected Result**: Ticket should be booked successfully and a booking confirmation should be displayed.

Test Case 6: Cancel Ticket

- **Test Case ID**: TC006
- **Description**: Verify that a user can cancel a booked ticket.
- **Preconditions**: User is logged in and has a booked ticket.
- **Steps**:
  -Navigate to the "My Bookings" section.
  -Select the ticket to be canceled.
  -Click on the "Cancel Ticket" button.
  -Confirm the cancellation.
- **Expected Result**: Ticket should be canceled successfully and a cancellation confirmation should be displayed.

Test Case 7: Payment Processing

- **Test Case ID**: TC007
- **Description**: Verify that payment processing works correctly during ticket booking.
- **Preconditions**: User is logged in and ready to book a ticket.
- **Steps**:

-Proceed to payment after selecting seats and entering passenger details.
-Enter valid payment details.
-Complete the payment process.

- **Expected Result**: Payment should be processed successfully and a booking confirmation should be displayed.

Test Case 8: Admin Login

- **Test Case ID**: TC008
- **Description**: Verify that an admin can log in successfully.
- **Preconditions**: Admin is registered and is on the admin login page.
- **Steps**:
  -Enter valid admin credentials.
  -Click on the "Login" button.
- **Expected Result**: Admin should be logged in successfully and redirected to the admin dashboard.

Test Case 9: Manage Train Schedule (Admin)

- **Test Case ID**: TC009
- **Description**: Verify that an admin can add, update, or delete train schedules.
- **Preconditions**: Admin is logged in.
- **Steps**:
  -Navigate to the "Manage Train Schedule" section.
  -Add a new train schedule or update/delete an existing schedule.
  -Save the changes.
- **Expected Result**: Train schedule should be added, updated, or deleted successfully.

Test Case 10: Generate Reports (Admin)

- **Test Case ID**: TC010
- **Description**: Verify that an admin can generate reports on bookings, cancellations, and payments.
- **Preconditions**: Admin is logged in.
- **Steps**:
  -Navigate to the "Reports" section.
  -Select the type of report (bookings, cancellations, payments).

-Select the date range.
-Generate the report.

- **Expected Result**: Report should be generated successfully and displayed/downloaded as specified.

CREDITS

APARIMITA RANPISE (23112031): UI, SQL Queries, Documentation(requirement gathering, UML diagrams, task & dependencies, test cases)

ARYA JAWARKAR (23112015): UI, SQL Queries, Database gathering , Database connectivity, backend

NAZRANA BAGWAN (23112047): UI, SQL Queries, Database gathering , Database connectivity, backend, Documentation(task network)