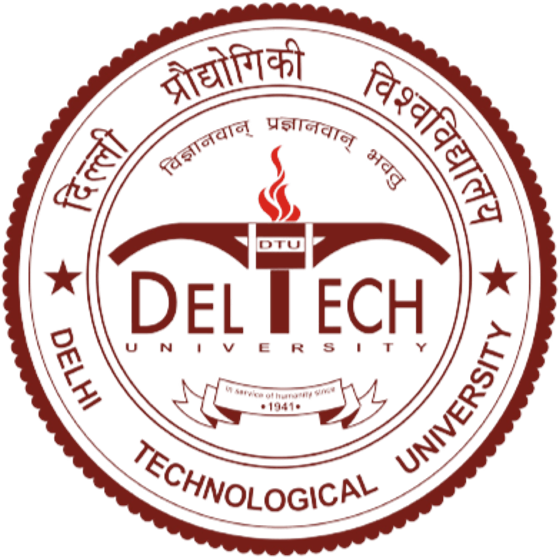
**DELHI TECHNOLOGICAL UNIVERSITY**

**Delhi – 110042**

**Department of Computer Science and Engineering**

****

**OBJECT ORIENTED SOFTWARE ENGINEERING LAB MANUAL (OOSE)**

**CO-362**

**SUBMITTED TO: SUBMITTED BY:**

**ARRAN P GONSALVES BINCY SEBASTIAN**

**2K21/CO/136**

**E5 Sec-1 (G3)**

**CASE STUDY ON RAILWAY RESERVATION SYSTEM.**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNO.** | **AIM** | **DATE** | **SIGN** | **PAGE NO.** |
| 1 | DRAW THE INITIAL REQUIREMENTS DOCUMENT (IRD) OF THE CASE STUDY | 07/02/2024 |  | 3 |
| 2 | WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT OF THE CASE STUDY | 07/02/2024 |  | 4 |
| 3 | DESIGN THE USE CASE DIAGRAM OF THE CASE STUDY | 07/02/2024 |  | 7 |
| 4 | WRITE USE CASE DESCRIPTION OF YOUR CASE STUDY | 08/04/2024 |  | 8 |
| 5 | DRAW CLASS DIAGRAM OF YOUR CASE STUDY | 19/02/2024 |  | 15 |
| 6 | DRAW SEQUENCE DIAGRAM OF CASE STUDY | 29/02/2024 |  | 16 |
| 7 | DRAW COLLABORATION DIAGRAM OF THE RAILWAY RESERVATION SYSTEM. | 04/04/2024 |  | 17 |
| 8 | DRAW ACTIVITY DIAGRAM OF RAILWAY RESERVATION SYSTEM. | 11/04/2024 |  | 18 |
| 9 | DESIGN TEST CASES FOR RAILWAY RESERVATION SYSTEM. | 11/04/2024 |  | 19 |

**EXPERIMENT 1**

**AIM:** DRAW THE INITIAL REQUIREMENTS DOCUMENT (IRD) OF THE RAILWAY RESERVATION SYSTEM.

**THEORY:** After the requirements are gathered, the initial requirement document (IRD) may be prepared. It is used to document and lists the initial set of requirements gathered through the various stakeholders.

|  |  |
| --- | --- |
| Title of the project | Railway Reservation System |
| Stakeholders involved in capturing requirement | Admin, User, Passenger |
| Technologies used in requirement capturing | Brainstorming, Interviews, FAST, User |
| Name of the person involved | Bincy Sebastian |
| Date | 07/02/2024 |
| Version | 1.0 |

|  |
| --- |
| Consolidated Initial Requirements   1. Performance requirements:   • User Satisfaction:- The system is such that it stands up to the user expectations.  • Response Time:- The response of all the operation is good. This has been made possible by careful programming.  • User friendliness: - The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.   1. Reliability: The reliability of the overall project depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated. 2. Availability: The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. A customer friendly system which is in access of people around the world should work 24 hours. 3. Maintainability: A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the project will be done. 4. Supportability: The code and supporting modules of the system will be well documented and easy to understand. Online User Documentation and Help System Requirements. 5. Safety and Robustness: The system is able to avoid or tackle disastrous action. In other words, it should be foul proof. The system safeguards against undesired events, without human intervention. 6. Portable: The software should not be architecture specific. It should be easily transferable to other platforms if needed. |

**EXPERIMENT 2**

**AIM:** WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT THE RAILWAY RESERVATION SYSTEM.

**THEORY:** Software Requirement Specification (SRS) Format as the name suggests, is a complete specification and description of requirements of the software that need to be fulfilled for the successful development of the software system. These requirements can be functional as well as non-functional depending upon the type of requirement. The interaction between different customers and contractors is done because it is necessary to fully understand the needs of customers.

**SRS DOCUMENT ON RAILWAY RESERVATION SYSTEM**

1. Introduction
   1. Purpose

Purpose of the Online Reservation System is to replace old practice of manual reservation. New system will let users register themselves and then make reservations online after logging into their account. It will make the ticket reservation easier, more accessible and transparent.

1.2. Scope

System will have following capabilities:

I. For users

A. Booking tickets online

B. Check train running status

C. Check train and berth availability

D. Check booking Status

II. For admin

A. Change user details

B. Change train details

C. Change station details

1.3. Definition, Acronyms and abbreviations

SRS: Software requirement

IRCTC: India Railways Catering And Tourism Corporation Ltd.

1.4. References

* IRCTC website
* Software Engineering (3rd ed.), By K.K. Aggarwal & Yogesh Singh

1.5. Overview

System broadly have following features:

* An interface to let new user register itself
* Login interface for users
* Login interface for admin
* Let user search and check availability of trains
* Let user check availability of seats
* Let user book seats
* Let user cancel reservation
* Let user modify it’s details
* Let admin modify user details
* Let admin modify train details
* Let admin modify station details

1. The Overall Description

2.1. Product Perspective

2.1.1. System Interfaces

System will have following major interfaces:

* User interface
* Internal admin interface
* Payment gateway integration
* Database interactions

2.1.2. Operations

System will Support following major operations

* Register
* Login
* Booking
* Cancellation
* Enquiries
* Payment processing

2.2. User Characteristics

* Public: Any person who wants to book or enquire about availability of train/seats. He must be comfortable in browsing and interacting with website or app.
* Admin: Representative of organisation who has authentication to make changes in the internal data.

2.3. Constraints

* System should support all major browsers like Chrome, Opera, IE.
* Website interface should adapt to handheld device screens.

2.4. Assumptions for dependencies

* System is dependent on internet connectivity
* It is assumed that the user has valid login credentials

1. Specific Requirements

3.1. External Interfaces

Users can access system through following interfaces:

* Using browser on a computer system
* Using browser on a handheld device
* Using Android application
* Using iPhone application

3.2. Functions

System will support following operations on all its interfaces:

* Register
* Login
* Booking
* Cancellation
* Enquiries

3.3. Other interfaces and operations

System will provide another interface internally accessed by admin to support following operations:

* Modify user details
* Modify train details
* Modify station details Performance requirements

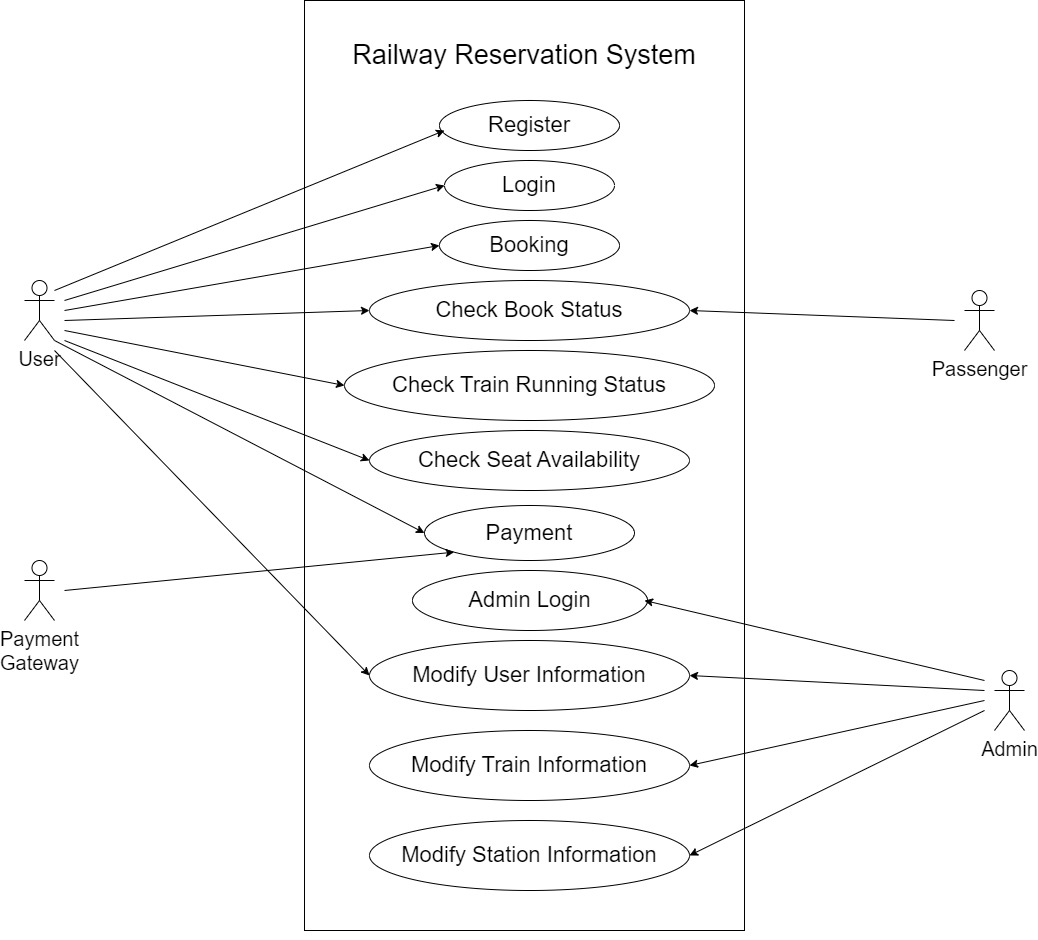
1. Supporting Information

* The project should be completed within [specified timeframe].
* The budget for the project is [specified budget].

**EXPERIMENT 3**

**AIM:** DESIGN THE USE CASE DIAGRAM OF RAILWAY RESERVATION SYSTEM.

**THEORY:** A use case diagram is a visual representation of the functional requirements of a system from the user's perspective. It illustrates the interactions between different actors and the system.



**EXPERIMENT 4**

**AIM:** WRITE USE CASE DESCRIPTION OF RAILWAY RESERVATION SYSTEM.

**THEORY:**

* Use cases should not be used to capture all the details of the system.
* Only significant aspects of the required functionality
* No design issues
* Use Cases are for "what" the system is , not "how" the system will designed
* Free of design characteristics.

**USE CASE DESCRIPTION**

1. Register

1.1. Introduction: This use case describes how a new user will registered in the Railway Reservation System.

1.2. Actors: End User

1.3. Pre-condition: None

1.4. Post-condition: If the use case is successful, the user will be registered in the system. If not system will remain unchanged.

1.5. Basic flow: This use case starts when a user wants to register itself on Railway Reservation System

(I) System shows user a Registration form to provide user details.

(II) The User enters it's required details, at least all mandatory data should be provided by user,

(III) System checks if user already exists or not, if not new user is registered.

1.6. Alternate flow:

(l) If any of user input is wrong user is show appropriate error.

(ll) If user details matches an already existing user, appropriate message is displayed to user.

1.7. Special Requirements: None

1.8. Use case relationships: None

1. Login

2.1. Introduction: This use case describes how a user will login to the Railway Reservation System

2.2. Actors: (1) End User (2) Admin

2.3. Pre-condition: User is already registered in system

2.4. Post-condition: If the use case is successful, the user will be logged in the system. If not system will remain unchanged.

2.5. Basic flow: This use case starts when a user wants to login to Railway System.

(I) System shows user a Login Form to provide user credentials

(II) The User enters valid user name and password.

(III) If user name and password are correct user is logged in the system.

2.6. Alternate flow:

(I) If user credentials are not correct, user is shown appropriate error.

2.7. Special Requirements: None

2.8. Use case relationships: None

1. Booking

3.1. Introduction: This use case describes how a user make a booking on Railway Reservation System.

3.2. Actors: End User

3.3. Pre-condition: User is already logged in the system

3.4. Post-condition: If the use Case is successful, the user will able to book tickets through the system

3.5. Basic flow: This use case starts when a user wants to book tickets on Railway Reservation System.

(I) User will search for the required train

(II) User will enquire seat status

(III) If seats available user will make bookings.

(IV) User will pay for the tickets.

(V) Seats will be for user.

3.6. Alternate flow:

(I) If trains not available between given status, appropriate message will be shown.

(II) If Seats are not available, user will be notified.

(III) If payment breaks in between, Seats will not be booked. User will be notified on the same.

3.7. Special Requirements: None

3.8. Use case relationships: None

1. Check booking status

4. I. Introduction: This use case describes how a user can check booking status

4.2. Actors: (l) End User

4.3. Pre-condition: User is already logged in the system and have already made bookings for upcoming journeys

4.4. Post-condition: If the use case is successful, the user will be able check status of bookings made by him/her for upcoming journeys.

4.5. Basic flow: This use case starts when a user wants check booking status of tickets for existing for upcoming journeys.

(I) User will enter the correct PNR value

(II) User will be shown booking status

4.6. Alternate flow:

(I) If PNR value is not correct and user will be notified of the same.

4.7. Special Requirements: None

4.8. Use case relationships: None

1. Check train running status

5.1. Introduction: This use case how a user can check train running

Status on Railway reservation system

5.2. Actors: End User

5.3. Pre-condition: User is already logged in the system

5.4. Post-condition: If the use case is successful, the user will be able to query status of trains running between given stations.

5.5. Basic flow: This use case starts when a user wants check Status Of trains running between stations selected by user

(I) User will enter station codes of start and end station

(II) User will be shown all trains running between those stations

5.6. Alternate flow:

(l) If Station code is incorrect, an appropriate error is shown to user.

5.7. Special Requirements: None

5.8. Use case relationships: None

1. Check seat availability

6.1. Introduction: This use case describes how a user can check availability of seats in selected train

6.2. Actors: (1) End User

6.3. Pre-condition: User is already logged in the system, and have executed case 5.

6.4. Post-condition: If the use case is successful, the user will be able to query seat availability for selected train.

6.5. Basic flow: This use case starts when a user wants to check status of trains running between Stations selected by user

(I) User will select the train and Coach type.

(II) User will be shown Scats availability for selected train and coach type.

6.6. Alternate flow: (I) None

6.7. Special Requirements: None

6.8. Use case relationships: None

1. Payment

7.1. Introduction: This use case describes how a user can make payment for his/her bookings.

7.2. Actors: (1) End User

7.3. Pre-condition: User is already logged in the system, and have executed use case 6.

7.4. Post-condition: If the use case is successful, the user will able make payment for the booking.

7.5. Basic flow: This use case starts when a user wants to make payment for the bookings he has initiated

(I) User has selected Seats and have filled passenger data.

(II) User will now proceed to payment page.

(III) User selected a payment method.

(IV) User is taken to payment gateway.

(V) User enter required details on gateway.

(VI) Payment is confirmed.

7.6. Alternate flow:

(I) If payment fails, user will be notified of the same.

7.7. Special Requirements: Internet connection should be stable during payment transaction else payment will fail.

7.8. Use case relationships: None

1. Admin login

8.1. Introduction: This use case describes how an admin will login to the System.

8.2. Actors: (1) Admin

8.3. Pre-condition: None

8.4. Post-condition: If the use case is successful, admin will be logged in the system.

8.5. Basic flow: This use case starts when admin wants to login in the Railway Reservation System

(I) Admin provide correct admin credentials.

(II) Admin will able to login to System

8.6. Alternate flow:

(I) If credentials are wrong, admin will not login. An appropriate error is shown to admin.

8.7. Special Requirements: Admin login will happen within organisation and the interface will remain internal to organisation.

8.8. Use case relationships: None

1. Modify Train information

9.1. Introduction: This use case describes how admin will make changes in train information.

9.2. Actors: (1) Admin

9.3. Pre-condition: Admin is logged in the system.

9.4. Post-condition: If the use case is successful, admin will be able to make changes to information of selected train.

9.5. Basic flow: This use case starts when admin wants to make changes in train information in Railway Reservation System

(I) Admin will select a train.

(II) Admin will make changes in train details.

(III) Admin will save changes made.

9.6. Alternate flow:

(I) If admin Ieaves any required field empty, appropriate error will shown by system.

(II) If admin doesn't save changes made, then train details will not be modified.

9.7. Special Requirements: None

9.8. Use case relationships: None

1. Modify Station information

10.1. Introduction: This use case describes how admin will make changes in station information

10.2. Actors: (1) Admin

10.3. Pre-condition: Admin is logged in the system

10.4. Post-condition: If the use case is successful, admin will be able to make changes to information of selected station.

10.5. Basic flow: This use case starts when admin wants to make changes station information in Railway Reservation System

(I) Admin will select a station.

(II) Admin will make changes in Station details.

(III) Admin will save changes made.

10.6. Alternate flow:

(I) If admin leaves any required field empty, appropriate error will be shown by system.

(II) If admin doesn't save changes made, then station details will not be modified.

10.7. Special Requirements: None

10.8. Use case relationships: None

1. Modify User Information

11.1. Introduction: This use case describes how admin will make changes in user information.

11.2. Actors: (1) Admin (2) End user

11.3. Pre-condition: Admin/End user is logged in the system

11.4. Post-condition: If the use case is successful, admin/end user will be able to make changes to information of user.

11.5. Basic flow: This use case starts when admin/end user wants to make changes in user information in Railway Reservation System.

(I) Admin/end user will make changes in user details.

(II) Admin/end user will save changes made.

11.6. Alternate flow:

(I) If admin/end user leaves any required field empty, appropriate error will be shown by system.

(II) If admin/end user doesn't save changes made, then user details will not be modified.

11.7. Special Requirements: None

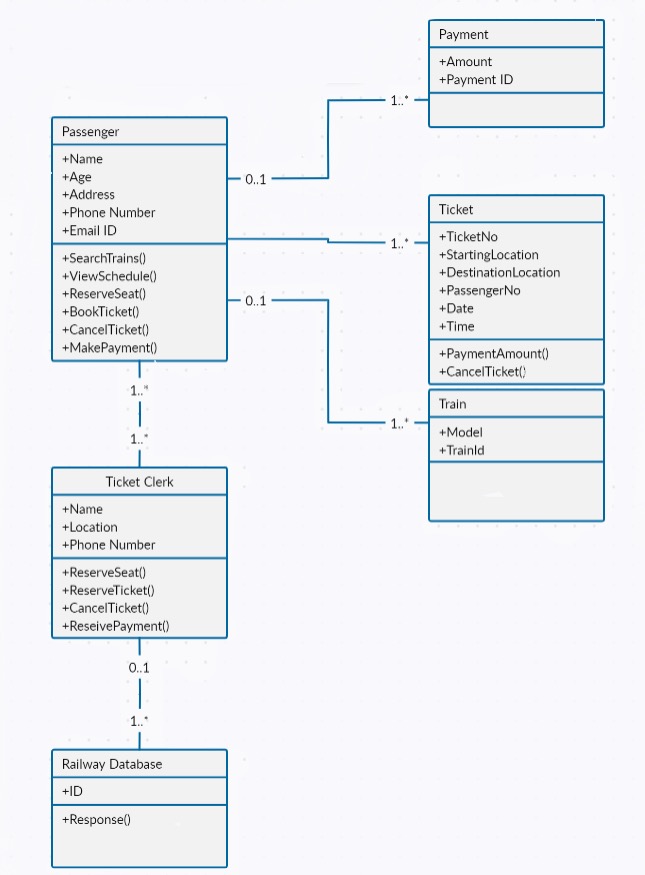
11.8. Use Case relationships: None

**EXPERIMENT 5**

**AIM:** DRAW CLASS DIAGRAM OF RAILWAY RESERVATION SYSTEM.

**THEORY:** Class diagrams are a type of UML (Unified Modelling Language) diagram used to visually represent the structure and relationships of classes within a system i.e. used to construct and visualize object-oriented systems. Class diagrams provide a high-level overview of a system’s design, helping to communicate and document the structure of the software. They are a fundamental tool in object-oriented design and play a crucial role in the software development lifecycle.

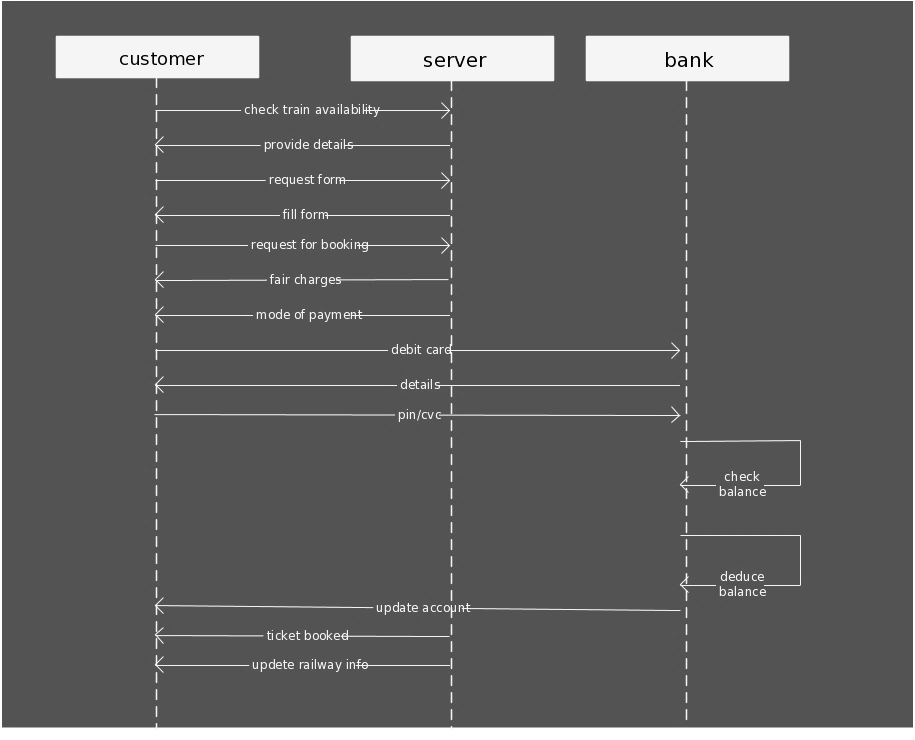
**CLASS DIAGRAM:**



**EXPERIMENT 6**

**AIM:** DRAW SEQUENCE DIAGRAM OF THE RAILWAY RESERVATION SYSTEM.

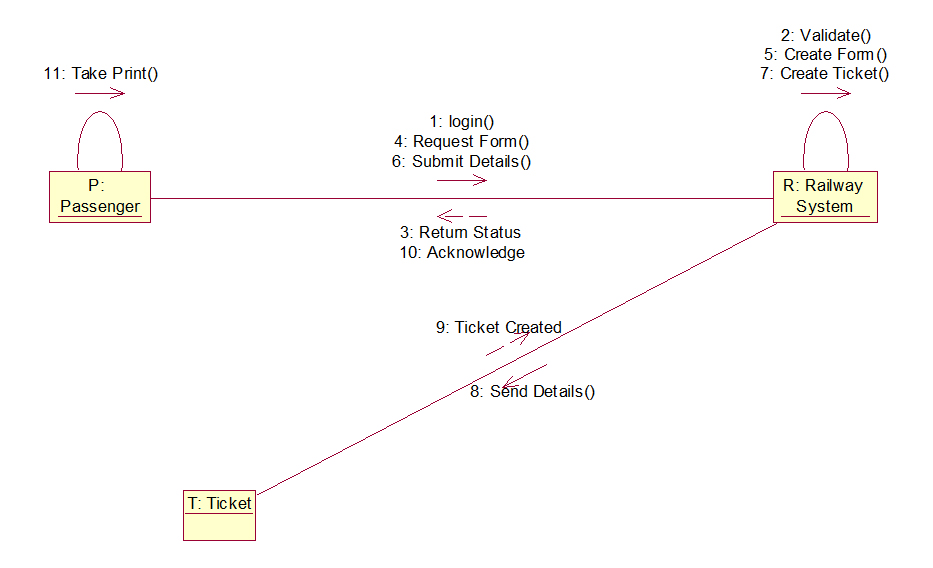
**THEORY:** A sequence diagram or system sequence diagram shows process interactions arranged in a time sequence. The diagram depicts the processes and objects involved and the sequence of messages exchanged as needed to carry out the functionality.



**EXPERIMENT 7**

**AIM:** DRAW COLLABORATION DIAGRAM OF THE RAILWAY RESERVATION SYSTEM.

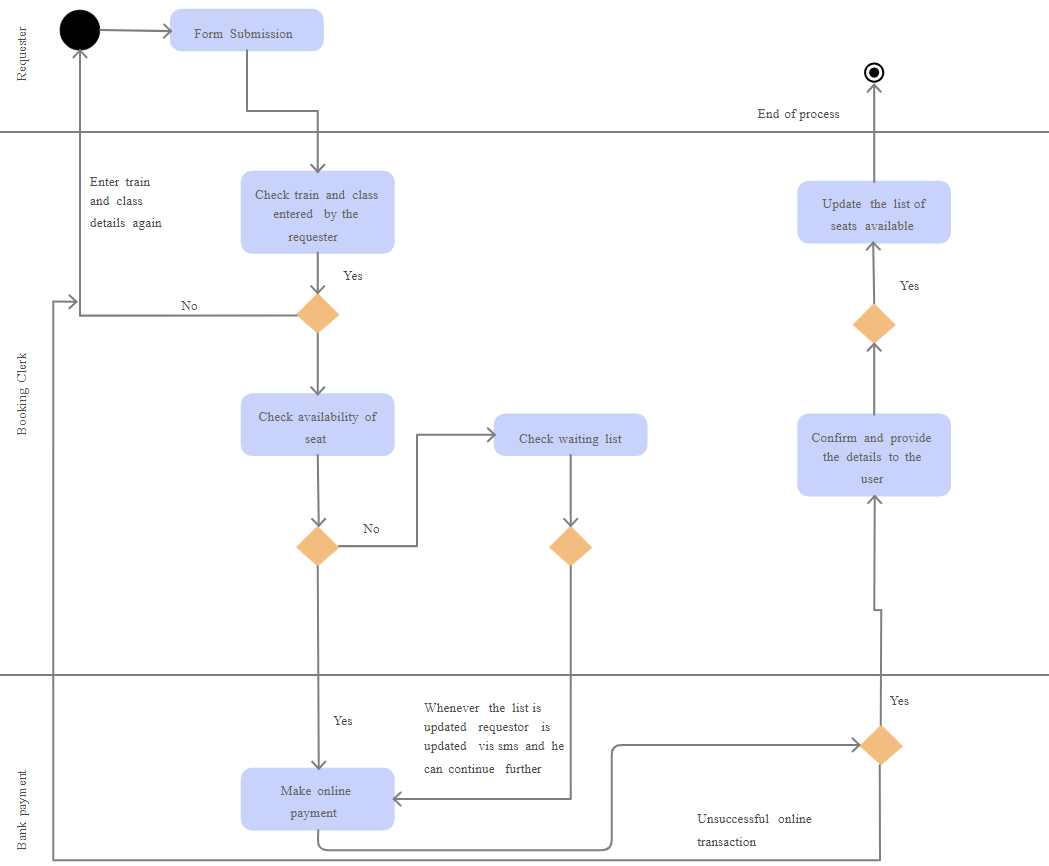
**THEORY:** The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



**EXPERIMENT 8**

**AIM:** DRAW ACTIVITY DIAGRAM OF RAILWAY RESERVATION SYSTEM.

**THEORY:** An activity diagram is a type of UML diagram used to represent the flow of control within a system or process. It consists of activities (tasks or actions), transitions (sequences between activities), decision points (conditions for branching), forks and joins (concurrency), and start and end nodes. It provides a visual representation of how activities are executed and decisions are made within the system or process.



**EXPERIMENT 9**

**AIM:** DESIGN TEST CASES FOR RAILWAY RESERVATION SYSTEM.

**THEORY:** Test cases for a railway reservation system cover registration, login, train search, booking, ticket cancellation, seat allocation, payment, error handling, performance, and cross-device compatibility. They ensure the system works smoothly, handles errors gracefully, performs well under load, and is compatible with various browsers and devices.

Here's an outline of test cases covering different aspects of the railway reservation system:

* **User Registration and Login:**

Test Case 1: Verify that a new user can register with valid details.

Test Case 2: Verify that an existing user can log in with correct credentials.

Test Case 3: Verify that an existing user cannot log in with incorrect credentials.

Test Case 4: Verify that the system prevents duplicate registrations with the same email or username.

* **Train Search and Booking:**

Test Case 5: Verify that users can search for trains by specifying the source and destination stations.

Test Case 6: Verify that the system displays a list of available trains for the specified route and date.

Test Case 7: Verify that users can select a train and view its details (e.g., timings, fare, seat availability).

Test Case 8: Verify that users can book tickets for a selected train with valid passenger details.

Test Case 9: Verify that the system prevents booking if the selected train is full or unavailable.

* **Ticket Cancellation:**

Test Case 10: Verify that users can cancel their booked tickets before the departure time.

Test Case 11: Verify that cancelled tickets are no longer valid and seats become available for booking.

* **Seat Availability and Allocation:**

Test Case 12: Verify that the system accurately displays seat availability for each class on a train.

Test Case 13: Verify that seats are allocated to passengers based on their preferences (e.g., window, aisle, upper berth).

* **Payment and Transaction:**

Test Case 14: Verify that users can make payments for their booked tickets using various payment methods (e.g., credit card, net banking, UPI).

Test Case 15: Verify that the payment gateway securely processes transactions and provides confirmation of successful payments.

* **Error Handling and Validation:**

Test Case 16: Verify that appropriate error messages are displayed for invalid inputs or actions (e.g., incorrect login credentials, missing fields).

Test Case 17: Verify that the system handles unexpected errors gracefully and provides meaningful error messages to users.

* **Performance and Scalability:**

Test Case 18: Verify that the system can handle a large number of concurrent users without significant performance degradation.

Test Case 19: Verify that the system remains responsive and stable even under peak load conditions.

* **Cross-browser and Cross-device Compatibility:**

Test Case 20: Verify that the railway reservation system functions correctly across different web browsers (e.g., Chrome, Firefox, Safari).

Test Case 21: Verify that the system is responsive and usable on various devices (e.g., desktops, laptops, tablets, smartphones).