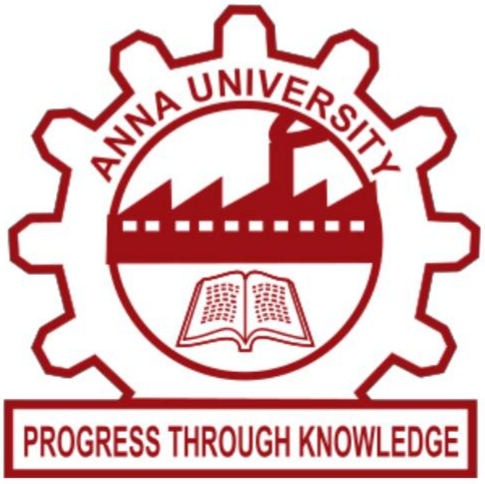
UNIVERSITY COLLEGE OF ENGINEERING NAGERCOIL

(ANNA UNIVERSITY CONSTITUENT COLLEGE)

KONAM, NAGERCOIL – 629 004



RECORD NOTE BOOK

CCS356-OBJECT ORIENTED SOFTWARE ENGINEERING

REGISTER NO :

NAME :

YEAR/SEMESTER :

DEPARTMENT :

UNIVERSITY COLLEGE OF ENGINEERINGNAGERCOIL

(ANNA UNIVERSITY CONSTITUENT COLLEGE)

KONAM, NAGERCOIL – 629 004

Register No:

Certified that, this is the bonafide record of work done by ***Mr./Ms****. …… of VI Semester in Computer Science and Engineering of this college, in the* ***CCS356 – OBJECT ORIENTED SOFTWARE ENGINEERING*** *during academic year 2023-2024 in partial fulfillment of the requirements of the B.E Degree course of the Anna University Chennai.*

Staff-in-charge **Head of the Department**

This record is submitted for the University Practical Examination held on .....

Internal Examiner **External Examiner**

INDEX

Exp No	Date	Title	Page	sign
1.		Identify a software system that needs to be developed		
2.		Document the Software Requirements Specification (SRS) for the Railway Reservation system		
3.		Identify the use cases and the develop the use Case model		
4.		Identify the conceptual classes and develop a Domain model and also derive a Class diagram for the Railway Reservation system		
5.		Using the identified scenarios, find the interaction between objects and represent them using UML Sequence and Collaboration diagram		
6.		Draw relevant State Chart and Activity Diagram for the Railway Reservation System		
7.		Implement the system as per the detailed design		
8.		Test the software system for all the scenarious identified as per the case diagram		
9.		Improve the reusability and maintainability of the software system by applying appropriate design patterns		

Ex No: 1	IDENTIFY A SOFTWARE SYSTEM THAT NEEDS TO BE DEVELOPED-RAILWAY RESERVATION SYSTEM

Aim:

To identify a software system that needs to be developed - Railway Reservation system.

Introduction:

The Railway Reservation System (RRS) is a comprehensive software application designed to streamline the booking, cancellation, and management of train tickets. Accessible via web and mobile platforms, the RRS facilitates user registration, train search, seat selection, payment processing, and PNR status inquiry, ensuring a seamless and efficient user experience. The system integrates with the existing railway database for real-time data synchronization, enhancing accuracy and reliability. Additionally, it provides robust administrative tools for managing train schedules, seat allocations, and generating reports. This document outlines the objectives, scope, and architecture of the RRS, serving as a guideline for its development, implementation, and maintenance, and ensuring all stakeholders have a clear understanding of its capabilities and constraints.

User:

Represents users of the system who interact with the Railway Reservation System. Users can register, login, search for train, book tickets , cancel reservation, and check booking status. They can also interact with the administrator for the registration purposes.

Administrator:

Represents the administrator of the system who manages the system by interacting with the database and the users. An administrator can login to the system with their own credentials, reports to track performance, security, and various other functions that manages the database and resolves issues within the system.

Database:

Represents the storage mechanism for storing and managing reservation table record and user data. Responsible for operations such as adding, removing, updating, tracks payment transactions, and managing user accounts.

Interface for downloads:

The download interface allow user to download their e-tickets and administrators to download system reports and easy accessible to both passenger and admin dashboards.

Ratings and Reviews:

The users can and rate and review a train service. This feature helps other users determine the quality of the passenger review and recommend it to others or warn other users of its short comings.

Cross Platform Support:

Develop the website to be responsive and accessible across various devices.

Architecture of Railway Reservation System:

Client-Server Architecture:

- Implement a client-server architecture where the web browser acts as the client, and the Server hosts the application logic, data, and services.
- Use RESTful APIs for communication between the client and server to ensure scalability and maintainability.

Database Design:

NoSQL Database Schema:

Design a noSql database schema to store train collection, user collection (username, password, email), user preferences, reading history, and relationships (e.g. user-collection).

Normalization:

Normalize the database schema to reduce redundancy and maintain data integrity, ensuring each piece of data is stored in only one place.

Indexes and Query Optimization:

- Create indexes on frequently queried fields to improve database performance.
- Optimize database queries by avoiding unnecessary joins, using appropriate data types, and caching frequently accessed data.

Testing:

Unit Testing:

Write unit tests for individual components, such as controllers, services, and models, to ensure they behave as expected.

Integration Testing:

Perform integration tests to verify interactions between different modules and components of the system.

End-to-End Testing:

Conduct end-to-end tests to simulate user interactions and validate the system's functionality from the user's perspective.

System Testing:

Validate railway reservation system functionalities comprehensively to ensure reliability, usability, and performance in real-world scenarios.

Result:

Thus a software system that needs to be developed – the railway reservation system was identified successfully.

Ex No: 2	DOCUMENT THE SOFTWARE REQUIREMENT SPECIFICATION (SRS) FOR THE RAILWAY RESERVATION SYSTEM

Aim:

To document the Software Requirements Specification (SRS) for the Railway Reservation system.

Introduction:

The Railway Reservation System is an advanced digital platform designed to simplify and enhance the process of booking train tickets. This system provides users with the ability to effortlessly search for trains, view schedules, select seats, and make secure payments from the comfort of their home or on-the-go via web and mobile applications.

Purpose:

The purpose of this SRS document is to define the requirements of the Railway Reservation System. It serves as a guideline for the development team to understand the scope of the project, implement the desired functionalities, and ensure the system meets the needs of its users.

Problem Statement:

Design and implement a simple railway reservation system that allows users to online bookings and provide real time updates. The system also has an administrator role who manages the database that contains the booking and the system itself.The system should utilize design patterns to enhance modularity, flexibility, and maintainability.

Scope:

The Railway Reservation System will allow users to:

- Ensure efficient management of seat allocation.
- User friendly booking.
- Download the tickets from the platform into their system.
- Provide real-time updates.

Product Functions:

- User Account Management: Users can create accounts to manage their profiles, authenticate ticket management process, and to keep track of their interactions with the booking.
- Search and Discovery: Users can easily search for booking based on seat availability, registration, with advanced filtering options for efficient browsing.
- Content Distribution: The system facilitates the distribution of booking to users through secure downloads.
- Notification System: Users receive notifications via email or SMS regarding new seat availability releases, or account activities.
- Reporting and Analytics: Administrators can generate reports on seat occupancy rates, revenue analysis, and booking trends to inform marketing strategies and content curation.
- Content Security: Robust encryption and Digital Rights Management (DRM) mechanisms are implemented to protect transactions from unauthorized access and piracy.

User Classes and Characteristics:

User:

Represents users of the system who interact with the Railway Reservation System. Users can register, login, search for train, book tickets, cancel reservation, and check booking status. They can also interact with the administrator for the registration purposes.

Administrator:

Represents the administrator of the system who manages the system by interacting with the database and the users. An administrator can login to the system with their own credentials, reports to track performance, security, and various other functions that manages the database and resolves issues within the system.

Database:

Represents the storage mechanism for storing and managing reservation table record and user data. Responsible for operations such as adding, removing, updating, tracks payment transactions, and managing user accounts.

Functional Requirements:

Assumptions and Dependencies:

- Users and administrators are assumed to possess basic computer literacy and proficiency in the English language.
- Applicants may be required to scan and upload documents for verification purposes.
- Each user must have a unique user ID and password for accessing the platform.
- An administrator role is essential for overseeing system operations and managing user accounts.
- Continuous internet connectivity is necessary for seamless access to the railway reservation system.
- Users must have compatible web browsers installed on their devices to access the platform effectively.

Operating Environment:

Particulars: Client System, Server System

Operating System: Windows/Linux/Android/iOS, Linux

Processor: Intel or AMD

Hard Disk: 16 GB client system, 1 TB for server system

RAM: 8 GB for client system, 16 GB for server system

User Interfaces

• Reader/User: Users provide an easy-to-use search interface for users. This interface is designed to be intuitive and user- friendly, accessible via web browsers on desktops, laptops, tablets, and smartphones.

• Administrator: System administrators have access to a comprehensive dashboard for managing user accounts, moderating content, and generating reports. This interface is designed for efficient data management and decision-making.

Hardware Interfaces

The client systems (devices used by users, and administrators) interactwith the server hardware to access and manage booking. These client systems require standard hardware components such as processors, memory, and storage devices to support web browsing and application usage.

Software Interfaces

• Frontend Client: The user interface for users and admin is developed using HTML, CSS, and JavaScript, with dynamic elements powered by JSP (JavaServer Pages). Additionally, the administrator interface utilizes Java for local application development.

• Web Server: The railway reservation system is hosted on apache tomcat server, ensuring robust performance and scalability for handling user requests and data processing.

• Backend Database: The system utilizes an Mongo db database for storing booking tickets, user information, transaction records, and other relevant data. This backend infrastructure provides reliability, security, and efficient data management capabilities.

Communications Interfaces

The railway reservation system relies on internet connectivity for user interactions and data transmission. Communication between client devices and the server occurs over the HTTP (Hypertext Transfer Protocol), enabling seamless exchange of information and interactions between users and the system. Additionally, email notifications and SMS

alerts are sent to users and administrators using appropriate communication protocols for timely updates and notifications.

Non-functional Requirements

Performance Requirements

Response Time: The system should respond to user actions, such as search queries or page loading, within 2 seconds under normal load conditions to ensure a seamless user experience.

Scalability: The system should be able to handle a growing number of users and booking without significant degradation in performance. It should be able to support at least 10,000 concurrent users without experiencing slowdowns or crashes.

Data Retrieval Speed: Booked id's should be retrieved from the database and delivered tousers for viewing or download within 5 seconds on average, even during peak usage periods.

Safety Requirements

Data Security: All user data, including personal information and payment details, should be encrypted and stored securely to prevent unauthorized access or data breaches.

Backup and Recovery: Regular backups of the system data should be performed to ensure data integrity and facilitate quick recovery in the event of system failures or disasters.

Security Requirements

Authentication: Each user must be provided with a unique ID and password for accessing their account. Authentication should be required before accessing any account information to prevent unauthorized usage.

Data Encryption: Communication between client devices and the server should be encrypted using SSL/TLS protocols to protect data during transmission.

Access Control: The system should implement role-based access control mechanisms to restrict access to sensitive features or data based on user roles and permissions.

Software Quality Attributes

Usability: The system should be intuitive and easy to use, with clear navigation and minimal learning curve for users.

Reliability: The system should be reliable and available 24/7, with minimal downtime for maintenance or upgrades.

Maintainability: The system should be designed with modular architecture and well-commented code to facilitate future updates and modifications.

Portability: The system should be compatible with a wide range of devices and operating systems, allowing users to access website from desktops, laptops, tablets, and smartphones.

Interoperability: The system should be able to integrate with external systems and services, such as payment gateways and content delivery networks, to provide seamless functionality.

Result:

Thus the Software Requirement Specifications (SRS) for the railway reservation system is documented successfully.

Ex.No:3	IDENTIFY THE USE CASES AND DEVELOP THE USE CASE DIAGRAM FOR RAILWAY RESERVATION SYSTEM

Aim:

To identify the Use Cases and develop the Use Case diagram for the Railway Reservation System using the StarUML tool.

Introduction of Use Case Diagram:

Use case diagrams are graphical representations that depict the interactions network users (actors) and a system to achieve specific goals or tasks. They are an essential tool in software development for capturing and communicating system requirements in a clear and Visual manner.

Purpose:

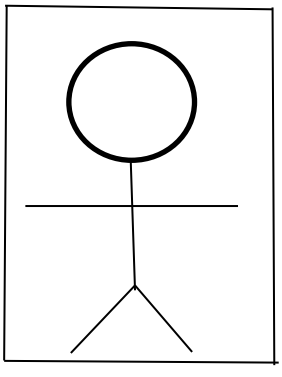
. Requirement Analysis: Use case diagrams to help stakeholders, including developers, designers, and clients, understand the system's functionality and behaviour from a user's perspective. They provide a high-level overview of the system's features and interactions.

. Communication: Use case diagrams to facilitate communication between stakeholders by providing a common visual language to discuss system requirements and functionalities. They help ensure that everyone involved in the project has a shared understanding of the system's scope and objectives.

. Design Validation: Use case diagram to assist in validating the system design by identifying potential gaps, inconsistencies, or missing functionalities. They allow stakeholders to review and refine the system requirements before proceeding with the implementation phase.

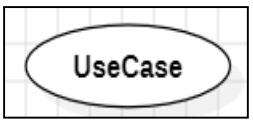
Notation:

1. Actor:



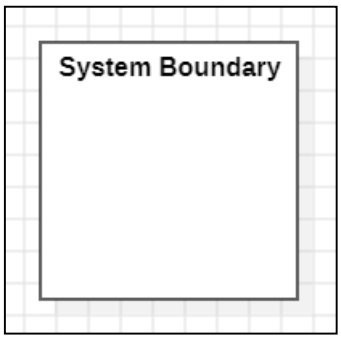
Actors are external entities that interact with the system. These can include users, other systems, or hardware devices. In a Use Case Diagram context, actors initiate use cases and receive the outcomes.

2. Use Case:

Use cases are like 

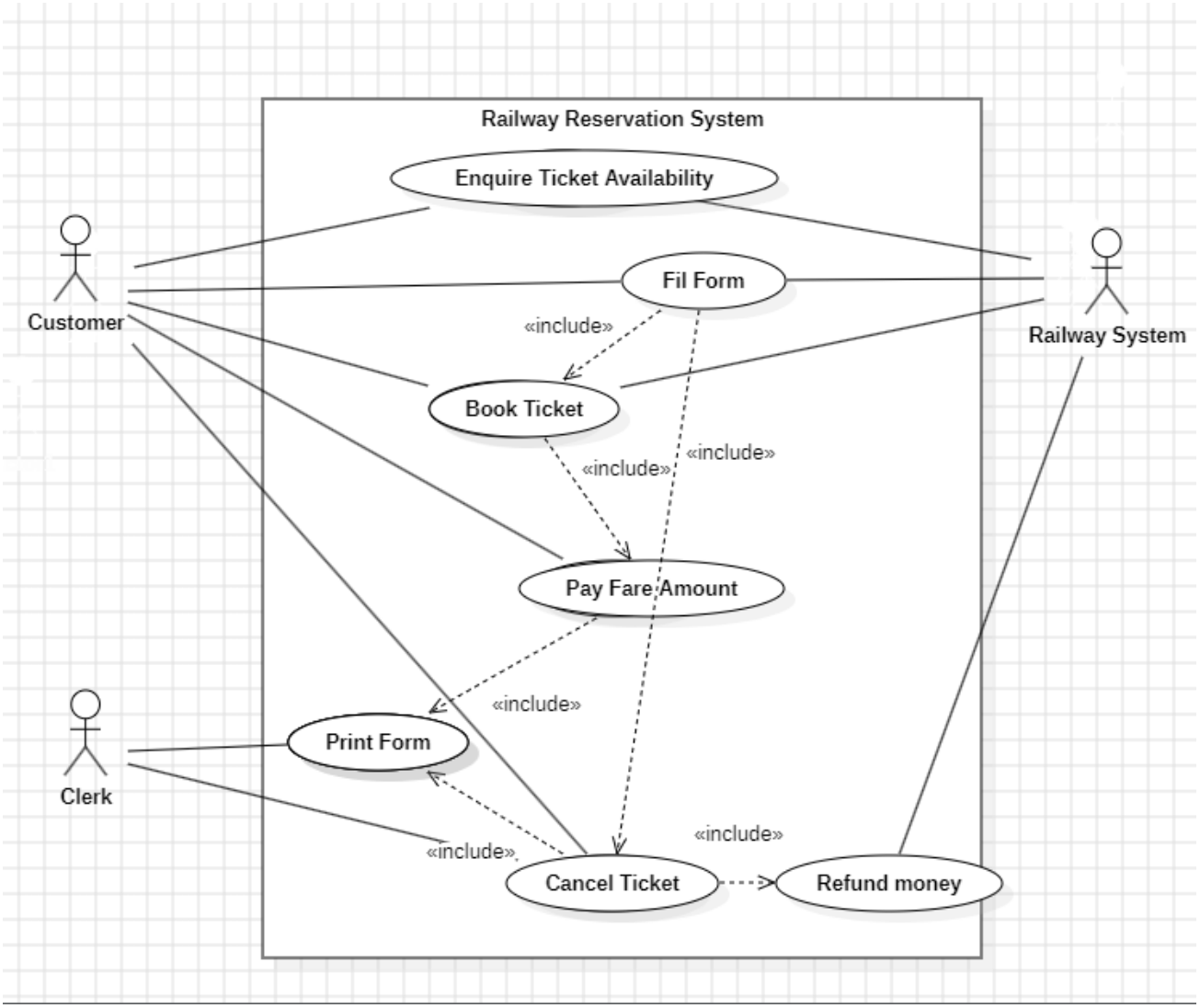
scenes in the play. They represent specific things your system can do. In the online shopping system, examples of use cases could be “Place Order,” “Track Delivery,” or “Update Product Information”. Ovals represent use cases.

3. System Boundary:



The system boundary is a visual representation of the scope or limits of the system you are modelling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it.

Use Case of Railway Reservation System:



1. Enquire Ticket Availability:

- **Description:** Customer checks the availability of tickets.
- **Actors:** Customer
- **Primary Actor:** Customer
- **Preconditions:** The customer must have access to the railway reservation system.

Scenario:

- The customer logs into the railway reservation system.
- The customer selects the route and date for travel.
- The system check the availability of tickets for the selected route and date.
- The system displays the availability status to the customer.

2. Fill Form

- **Description:** Allows customers filling out necessary details in a form.
- **Actors:** Customer
- **Primary Actor:** Customer
- **Preconditions:** The customer must have selected the route and date.

Scenario:

- The customer selects the form for ticket booking or cancellation.
- The customer fills in the required details such as personal information, journey details.
- The system validates the entered details.
- The system saves the form data.

3. Book Ticket

- **Description:** Booking a ticket, including filling out the form.
- **Actors:** Customer
- **Primary Actor:** Customer
- **Preconditions:** The customer must have filled out the form with valid details.

Scenario:

- The customer fills out the booking form.
- The system validates the form details.
- The system proceeds to make the payment.
- The system confirms the payment.

4. Pay Amount

- **Description:** Making a payment for the ticket fare.
- **Actors:** Customer
- **Primary Actor:** Customer
- **Precondition:** The customer must have filled out the booking form with valid details.

Scenario:

- The customer selects the payment method.
- The customer enters payment details.
- The system processes the payment.
- The system confirms the payment and updates the booking status.

5. Print Form:

- **Description:** Printing the form contain ticket details.
- **Actors:** Clerk
- **Primary Actor: Clerk**
- **Precondition:** The ticket must be booked or canceled.

Scenario:

- The clerk accesses the railway reservation system.
- The clerk selects the form to print.
- The system retrieves the form data.
- The system prints the form.

Result:

Thus to identify the use cases and develop the use case diagram for railway reservation system was executed successfully.

Ex No: 4	IDENTIFY THE CONCEPTUAL CLASSES AND DEVELOP A DOMAIN MODEL AND ALSO DERIVE A CLASS DIAGRAM FOR RAILWAR RESERVATION SYSTEM

Aim:

To identify the conceptual classes and develop a Domain Model and derive a Class Diagram from that Railway Reservation system.

Conceptual Classes:

In the Railway Reservation System, several conceptual classes can be identified to represent the various entities and functionalities of the system. Here are some key conceptual classes for the Railway Reservation System.

User:

Represents users of the system who interact with the Railway Reservation System. Users can register, login, search for train, book tickets , cancel reservation, and check booking status. They can also interact with the administrator for the registration purposes.

Administrator:

Represents the administrator of the system who manages the system by interacting with the database and the users. An administrator can login to the system with their own credentials, reports to track performance, security, and various other functions that manages the database and resolves issues within the system.

Database:

Represents the storage mechanism for storing and managing reservation table record and user data. Responsible for operations such as adding, removing, updating, tracks payment transactions, and managing user accounts.

Domain Model for Railway Reservation System:

User:

- **Attributes:**
  - CustomerID
  - name
  - email
- **Operations:**
  - searchTrains (route,ticket)
  - bookTicket(ticket)
  - cancelTicket(ticketID)
  - viewBookingTicket(ticket)

Clerk:

- **Attributes:**
  - clerkID
  - name
  - email
- **Operations:**
  - printForm(ticket)
  - assistBooking(customer,ticket)
  - assistCancellaton(customer,ticket)
  - calculateFare()
  - generateTicketDetails()
  - managesUsers()
  - resolvesIssues()

Payment:

- **Attributes:**
  - paymentID
  - paymentDate
  - amount
  - paymentMethod
  - status
  - ticketID
  - description
  - rating
  - reviews

- **Operations:**
  - open()
  - close()
  - processPayment()
  - rate(rating)
  - addReview(review)
  - getAverageRating()
  - getReviews()
  - refundAmount()
  - generateReceipt()
  - getTicketId()

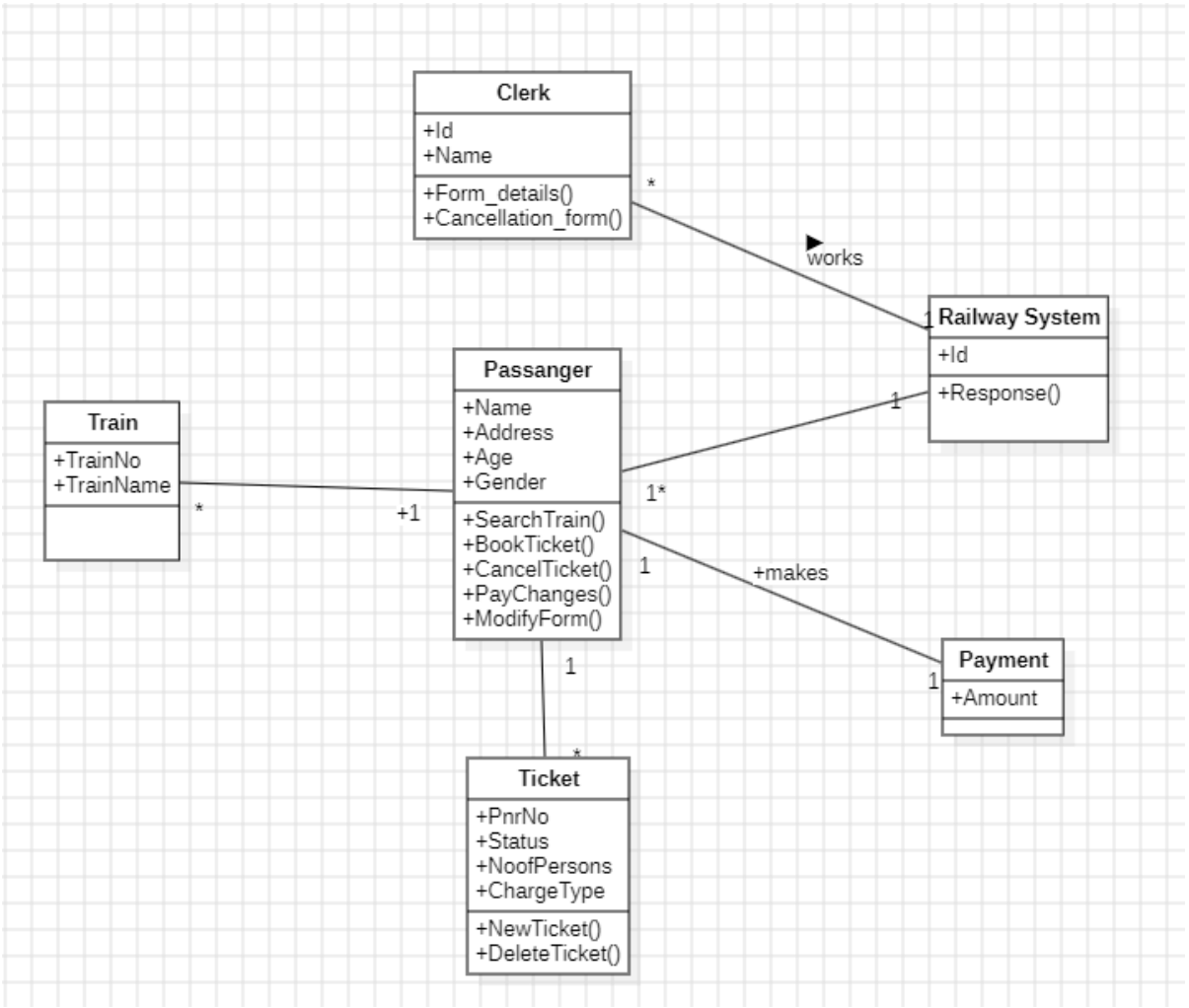
Train:

- **Attributes:**
  - trainID
  - trainNumber
  - trainName
  - source

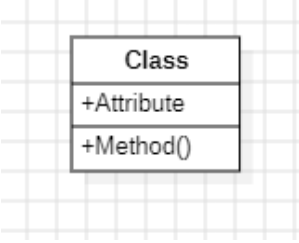
- desstination
- depaturetime

- **Operations:**
  - checkAvailability(data,time)
  - getschedule()
  - getRouteDetails()
  - confirmReservation()
  - cancelReservation()
  - generateReservationDetails()

Class Diagram for Railway Reservation System:



Notation:



Result:

Thus the conceptual classes, Domain Model and Class Diagram for railway reservation System was identified and developed successfully.

Ex No:5	USING THE IDENTIFIED SCENARIOS, FIND THE INTERACTION BETWEEN OBJECTS AND REPRESENT THEM USING UML SEQUENCE AND COLLABORATION DIAGRAMS

Aim:

Using the identified scenarios, find the interaction between objects and represent them using UML Sequence and Collaboration Diagrams for Railway Reservation Systems.

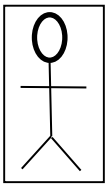
Sequence Diagram Definition:

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

A sequence diagram shows how different parts of a system work in a 'sequence' to get something done.

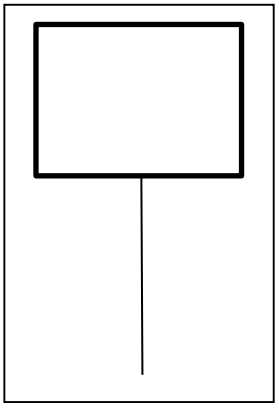
Sequence Diagram Notation:

1. Actors:



An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.

2. Lifelines:



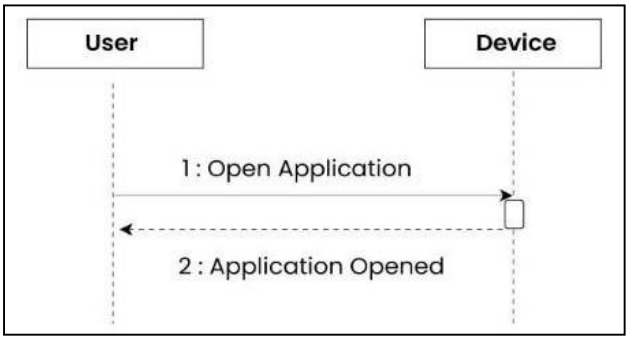
x:Class 1

A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

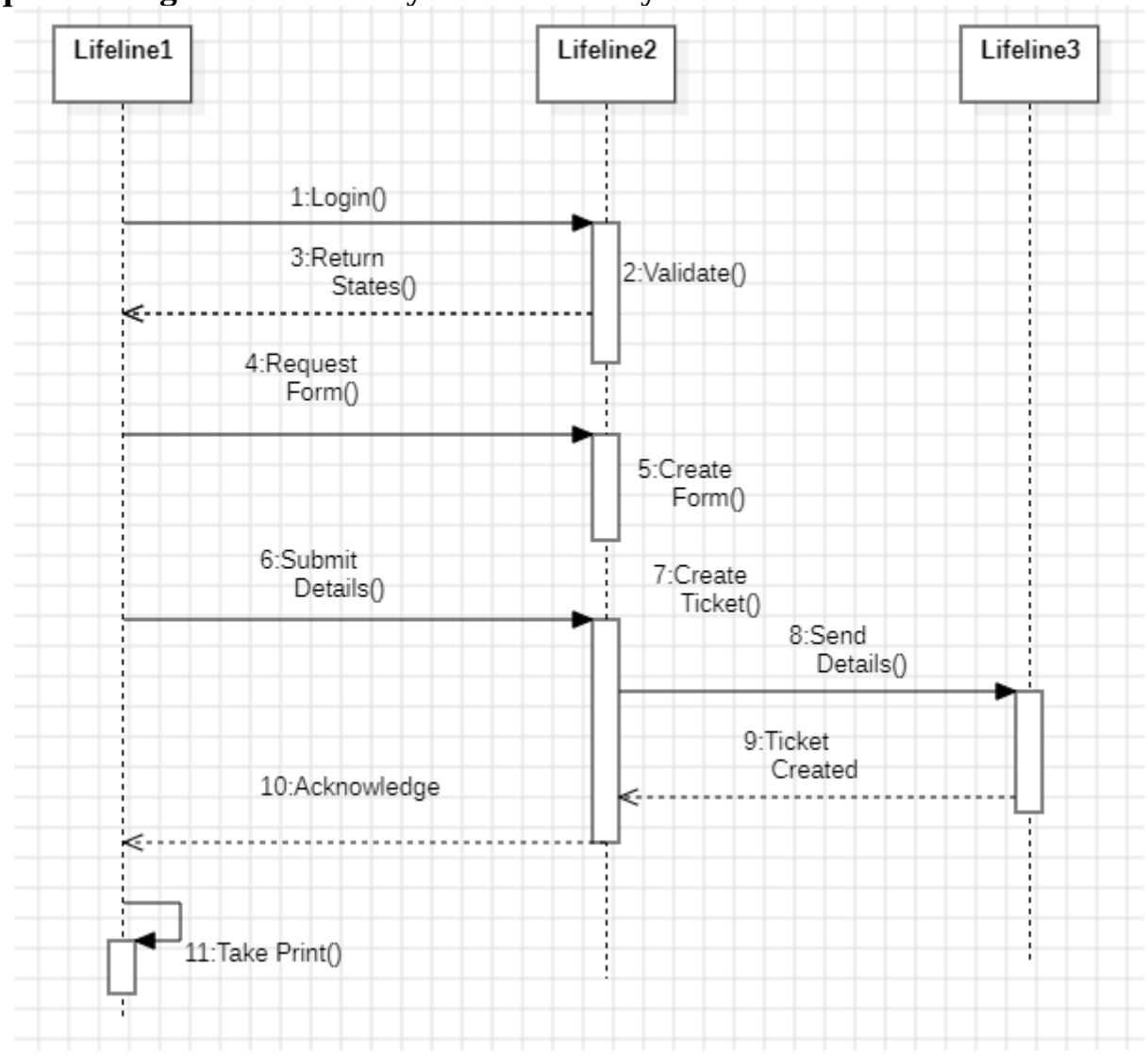
3. Messages:

Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline.

- We represent messages using arrows.
- Lifelines and messages form the core of a sequence diagram.



Sequence Diagram for Railway Reservation System:

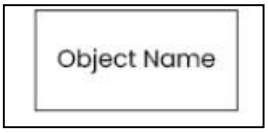


Collaboration Diagram Definition:

A Collaboration Diagram is a type of Interaction Diagram that visualises the interactions and relationships between objects in a system. It shows how objects collaborate to achieve a specific task or behaviour. Collaboration diagrams are used to model the dynamic behaviour of a system and illustrate the flow of messages between objects during a particular scenario or use case.

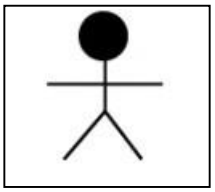
Collaboration Diagram Notation:

1. Objects/Participants:



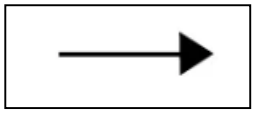
Objects are represented by rectangles with the object's name at the top. The diagram shows each object participating in the interaction as a separate rectangle. Objects are connected by lines to indicate messages being passed between them.

2. Actors:



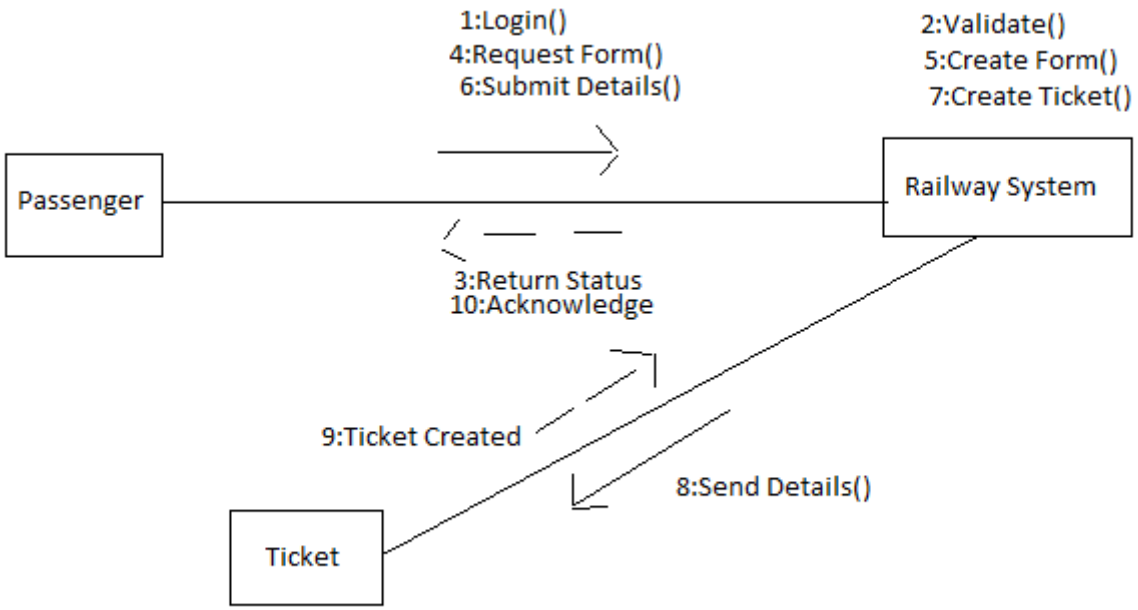
They are usually depicted at the top or side of the diagram, indicating their involvement in the interactions with the system's objects or components. They are connected to objects through messages, showing the communication with the system.

3. Messages:



Messages represent communication between objects. Messages are shown as arrows between objects, indicating the flow of communication. Each message may include a label indicating the type of message (e.g., method call, signal). Messages can be asynchronous (indicated by a dashed arrow) or synchronous (solid arrow).

Collaboration Diagram for Railway Reservation System:



Result:

Thus, using the identified scenarios, it was successfully executed to find the interaction between objects and represent them using UML Sequence and Collaboration Diagrams for Railway Reservation System.

Ex. No:6	DRAW RELEVANT STATE CHART AND ACTIVITY DIAGRAMS FOR THE SAME SYSTEM

Aim:

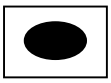
To draw relevant state charts and activity diagrams for the Railway Reservation system.

State Chart Diagram Definition:

A State Machine Diagram represents the system's condition or part of the system at finite instances of time. It's a behavioural diagram and it represents the behaviour using finite state transitions.

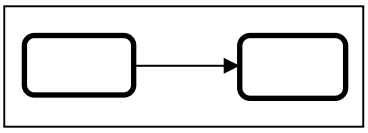
State Chart Diagram Notation:

1. Initial State:



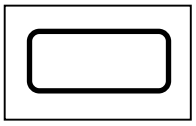
We use a black-filled circle to represent the initial state of a System or a Class.

2. Transition:



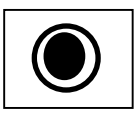
We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

3. State:

 **state**

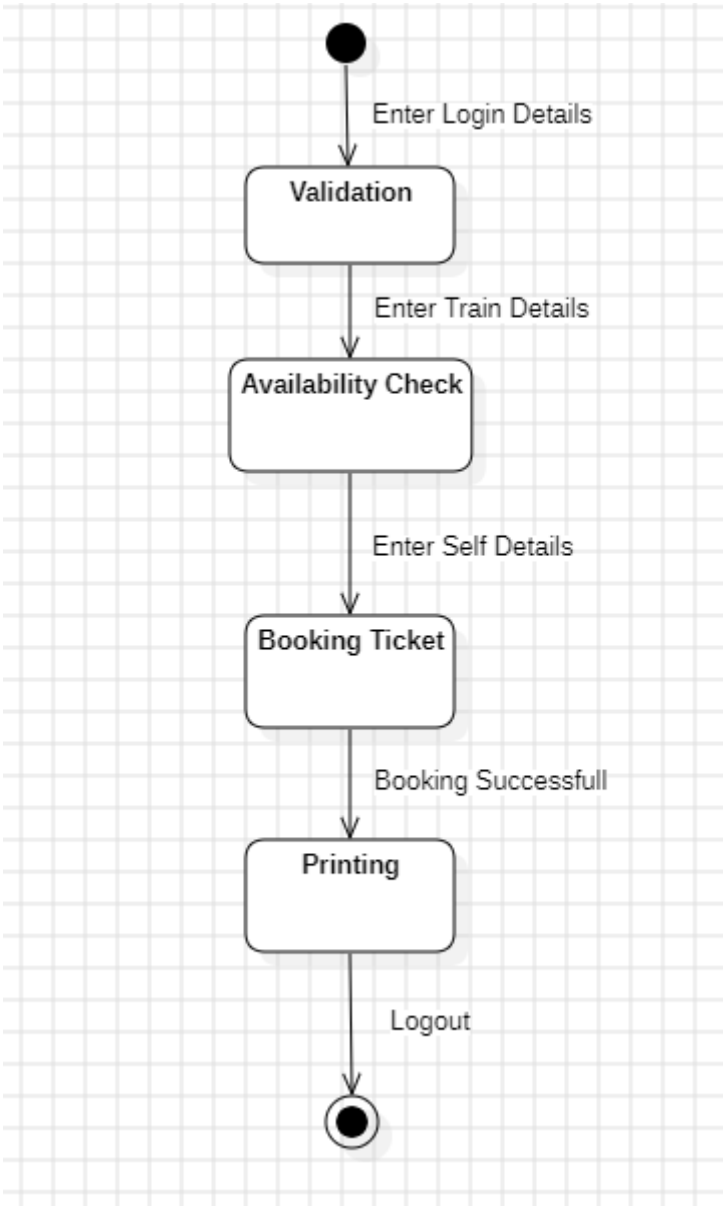
We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

4. Final State:



We use a filled circle within a circle notation to represent the final state in a state machine diagram.

State Chart Diagram for Railway Reservation System:



Activity Diagram Definition:

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We can depict both sequential processing and concurrent processing of activities using an activity diagram i.e., an activity diagram focuses on the condition of flow and the sequence in which it happens.

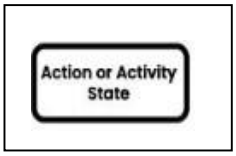
- We describe what causes a particular event using an activity diagram.
- An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

Activity Diagram Notation:

1. Initial State:

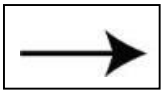
A process can have only one initial state unless we are depicting activities.We use a black-filled circle to depict the initial state of a system. For objects, this is the state when are instantiated. The Initial State from the UML Activity Diagram marks the entry point and initial Activity State.

2. Action or Activity State:



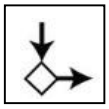
An activity represents the execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically, any action or event that takes place is represented using an activity.

3. Action Flow or Control Flow:



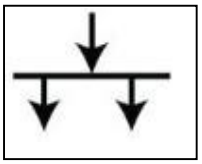
Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another activity state.

4. Decision Node or Branching:



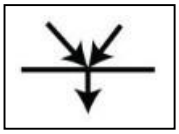
When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.

5. Fork:



Fork nodes are used to support concurrent activities. When we use a fork node both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts.

6.**Join:**

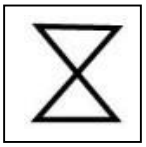


Join nodes are used to support concurrent activities converging into one. We have two or more incoming edges and one outgoing edge for join notations.

7.Final State or End State:

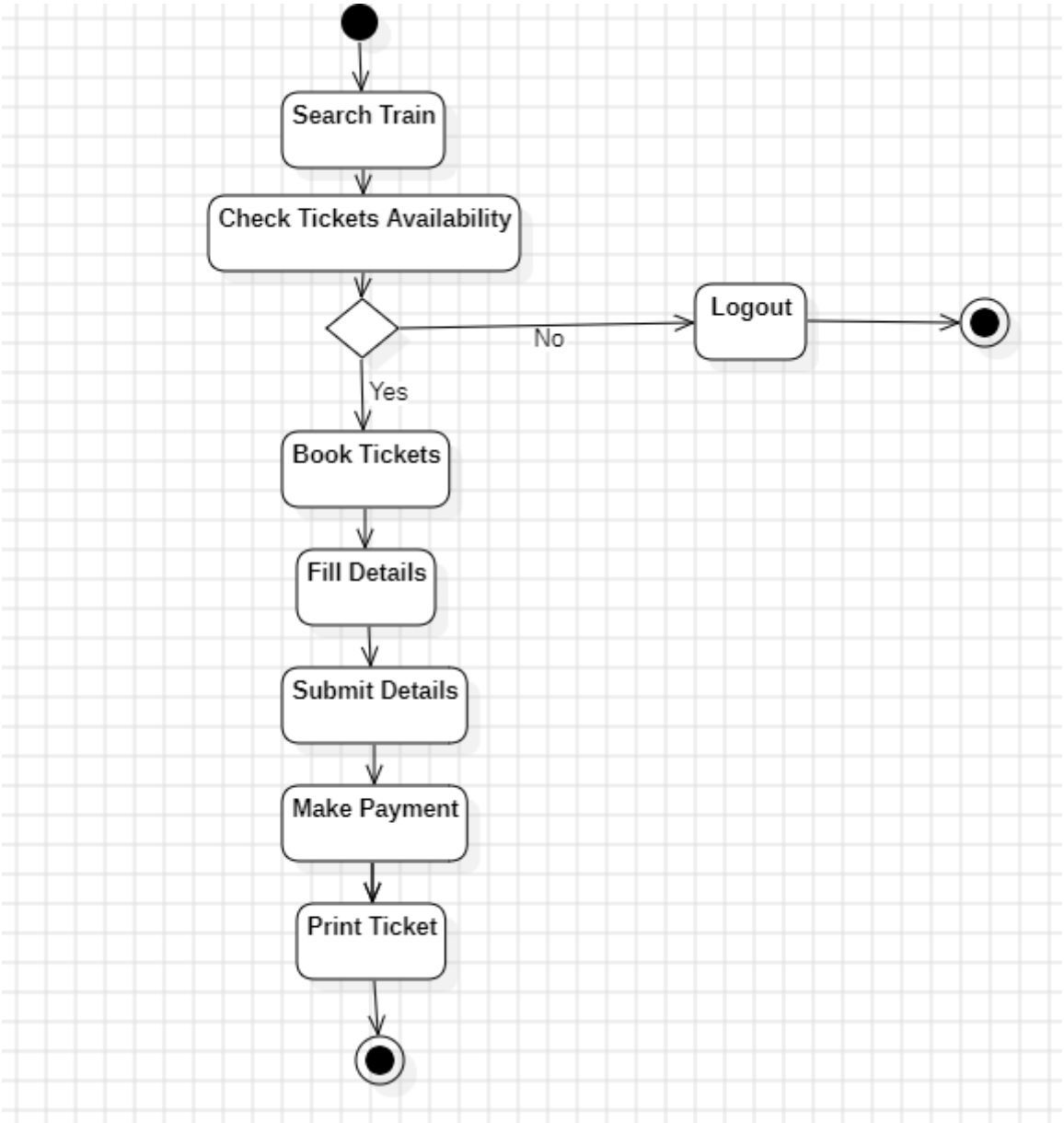
The state that the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

8.Time Event:



This refers to an event that stops the flow for a time; an hourglass depicts it. We can have a scenario where an event takes some time to complete.

Activity Diagram for Railway Reservation System:



Result:

Thus the relevant state chart and activity diagrams for the Railway Reservation system was executed successfully.

Ex No:7	IMPLEMENT THE SYSTEM AS PER THE DETAILED DESIGN

Aim:

To implement the Railway Reservation System as per the detailed design.

Component Diagram Definition:

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system.

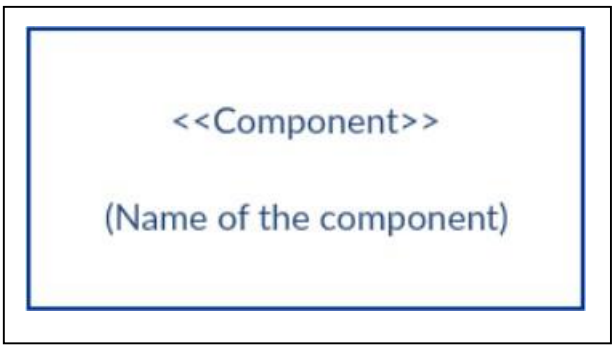
The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.

Component Diagram Notation:

1. Component:

There are three ways the component symbol can be used.

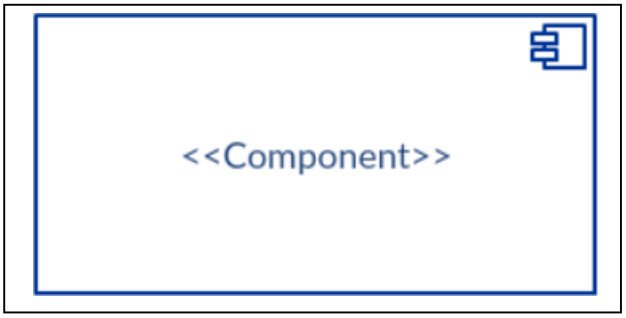
a.



Rectangle with the component stereotype (the text <<component>>). The component stereotype is usually used above component name to avoid confusing the shape with a class icon.

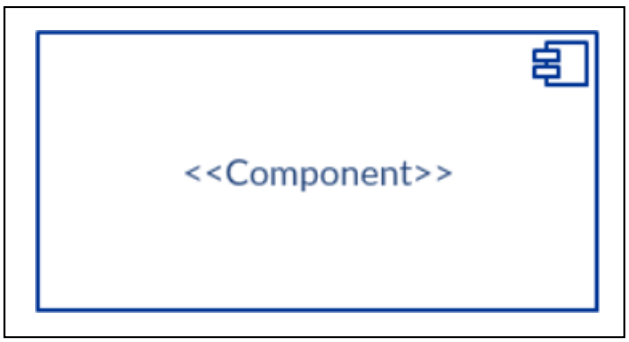
b.

Rectangle with the component icon in the top right corner and the name of the component.

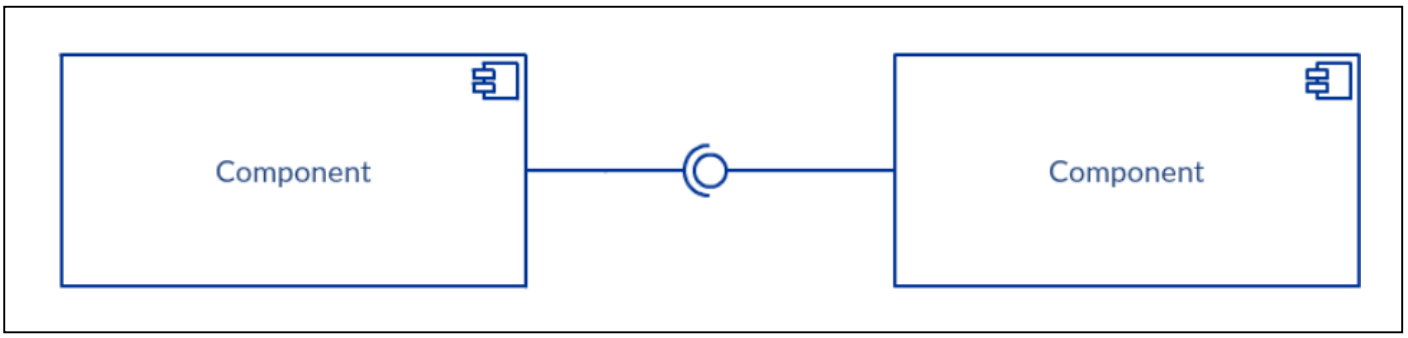


c.

Rectangle with the component icon and the component stereotype.

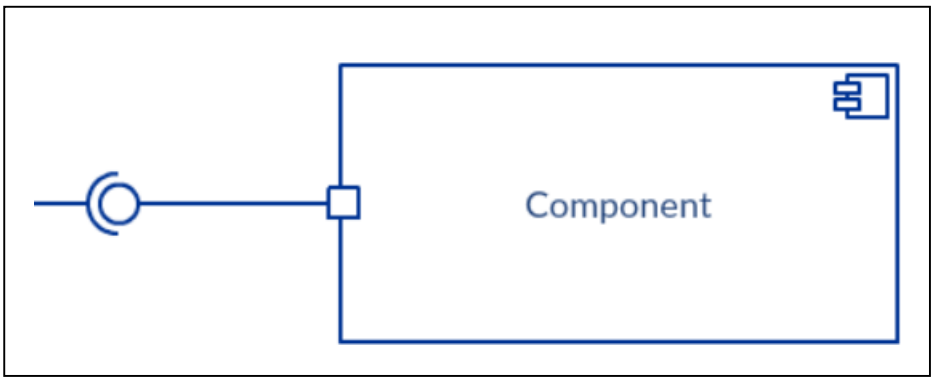


2. Provided Interface and Required Interface:



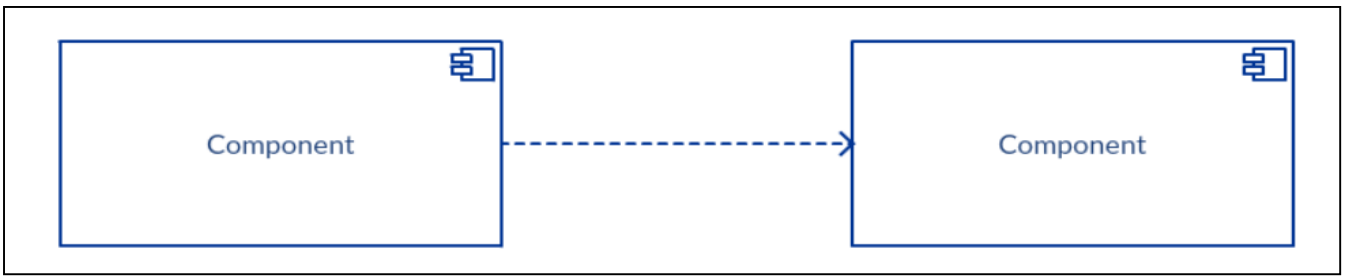
Interfaces in component diagrams show how components are wired together and interact with each other. The assembly connector allows linking the component's required interface (represented by a semi-circle and a solid line) with the provided interface (represented by a circle and solid line) of another component. This shows that one component is providing the service that the other is requiring.

3. Port:



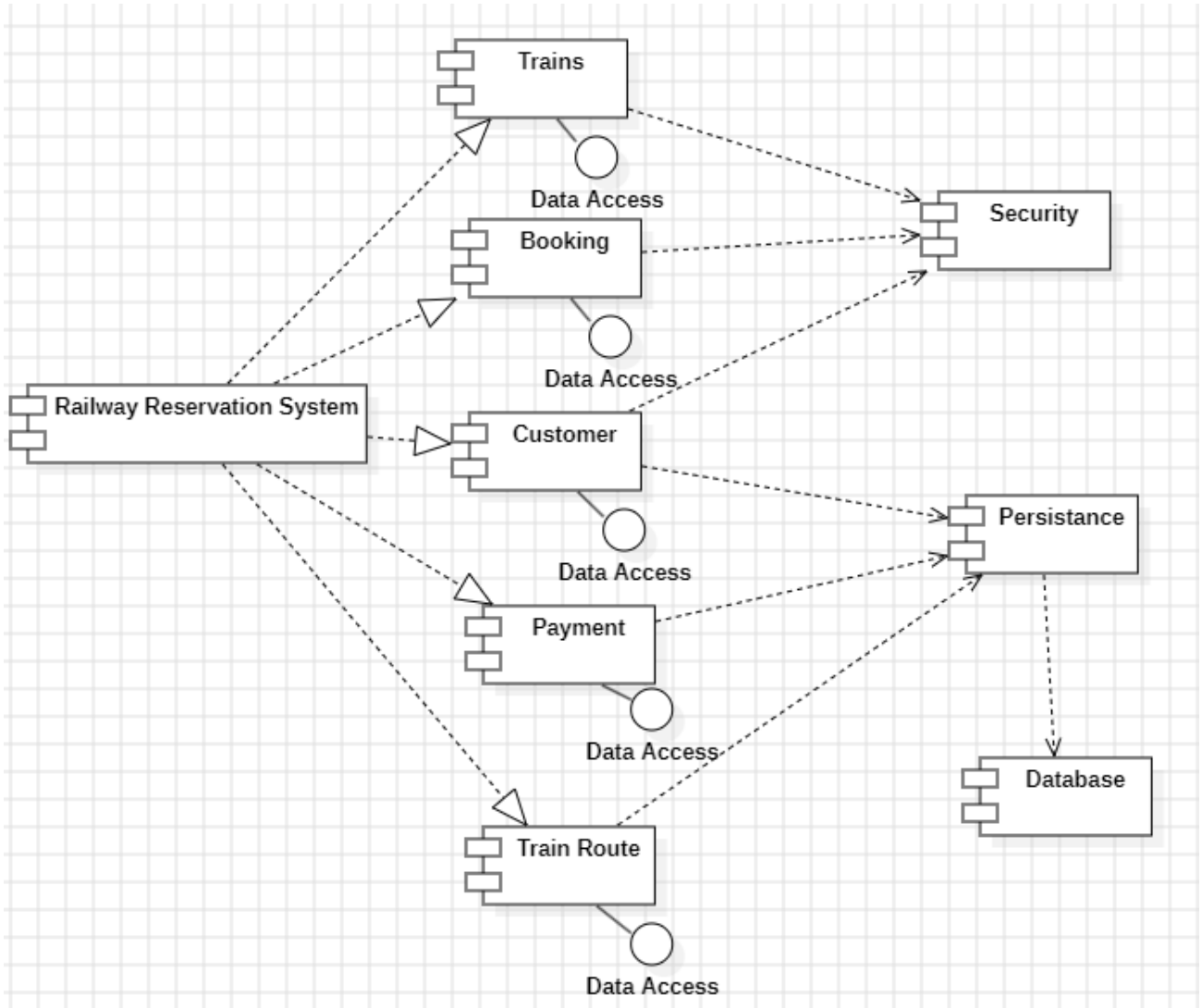
Port (represented by the small square at the end of a required interface or provided interface) is used when the component delegates the interfaces to an internal class.

4. Dependencies:



Although you can show more detail about the relationship between two components using the ball-and-socket notation (provided interface and required interface), you can just as well use a dependency arrow to show the relationship between two components.

Component Diagram for Railway Reservation System:



Deployment Diagram Definition:

A Deployment Diagram in software engineering is a type of Structural UML Diagram that shows the physical deployment of software components on hardware nodes. It illustrates the mapping of software components onto the physical resources of a system, such as servers, processors, storage devices, and network infrastructure.

Deployment Diagram Notation:

1. Component:

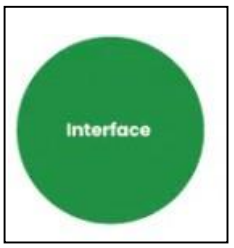
A component represents a modular and reusable part of a system, typically implemented as a software module, class, or package. It encapsulates its behaviour and data and can be deployed independently.

2. Artifact:

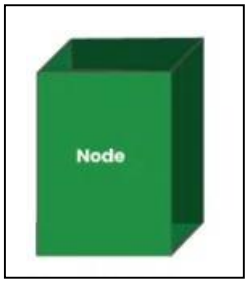
An artifact represents a physical piece of information or data that is used or produced in the software development process. It can include source code files, executables, documents, libraries, configuration files, or any other tangible item.

3. Interface:

An interface defines a contract specifying the methods or operations that a component must implement. It represents a point of interaction between different components or subsystems.

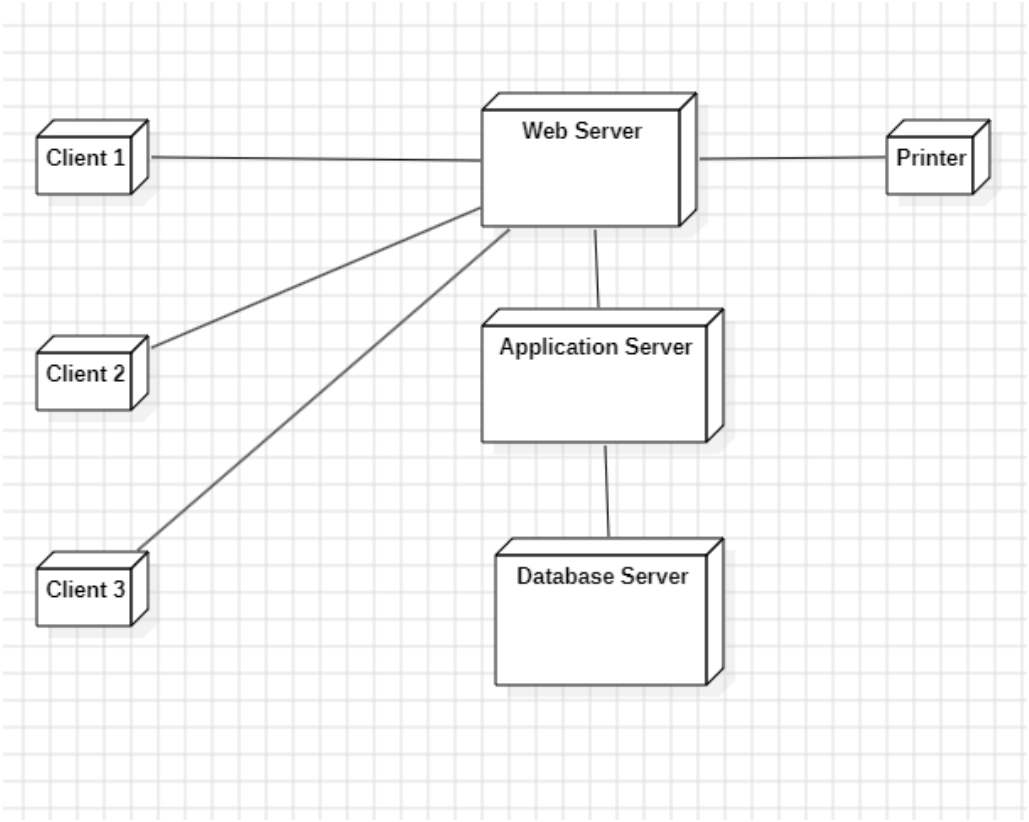


4. Node:



A node represents a physical or computational resource, as a hardware device, server, workstation, or computing resource, on which software components can be deployed or executed.

Deployment Diagram for Railway Reservation System:



Implementation For Railway Reservation System:

Client.java:

```
import java.util.ArrayList;

import java.util.List;


public class Customer {

    private String name;

    private List<Ticket> tickets;


    public Customer(String name) {

        this.name = name;

        this.tickets = new ArrayList<>();

    }
```

```
    public Ticket bookTicket(Train train) {
        Ticket ticket = new Ticket(train);
        tickets.add(ticket);
        return ticket;
    }

    public void cancelTicket(Ticket ticket) {
        ticket.cancel();
    }

    public void printTickets() {
        for (Ticket ticket : tickets) {
            System.out.println(ticket.getDetails());
        }
    }
}
```

Train.java:

```
public class Train {
    private String number;
    private String name;
    private String source;
    private String destination;

    public Train(String number, String name,
String source, String destination) {
```

```
        this.number = number;
        this.name = name;
        this.source = source;
        this.destination = destination;
    }

    public String getDetails() {
        return "Train " + number + " - " + name + "
from " + source + " to " + destination;
    }
}    }
```

Ticket.java:

```
public class Ticket {
    private Train train;
    private String status;

    public Ticket(Train train) {
        this.train = train;
        this.status = "Booked";
    }

    public void cancel() {
        this.status = "Cancelled";
    }
```

```
    public String getDetails() {
        return "Ticket for " + train.getDetails() + "
is " + status;
    }
}
```

Clerk.java:

```
public class Clerk {
    private String name;

    public Clerk(String name) {
        this.name = name;
    }

    public void printTicket(Ticket ticket) {
        System.out.println("Clerk " + name + "
printing: " + ticket.getDetails());
    }

    public void assistBooking(Customer customer,
Train train) {
        Ticket ticket = customer.bookTicket(train);
        printTicket(ticket);
}
```

Result:

Thus, the Railway Reservation System implementation as per the detailed design was executed successfully.

EX NO: 8 DATE:	**Test the Software System for All Scenarios Identified as Per the Use Case Diagram**

Aim:

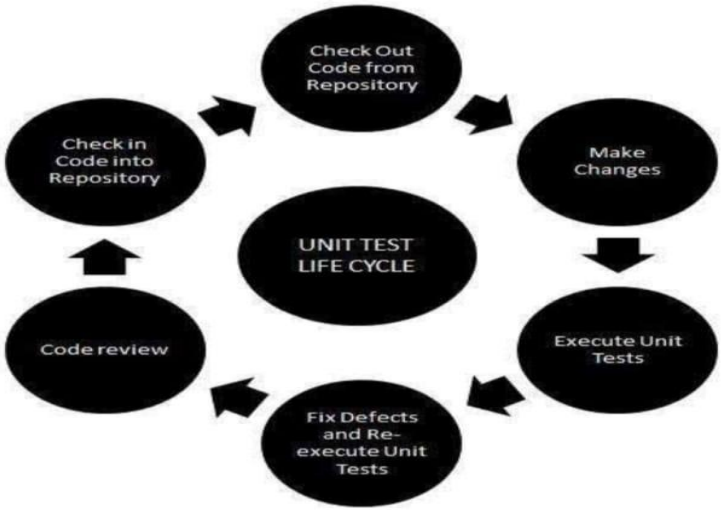
To test the software system for all scenarios identified in the use case diagram.

Railway Reservation System:

In the domain of railway reservation management, efficient handling of human resources is vital for organizational productivity and employee satisfaction. A Railway Resevation System facilitates the process of managing employee data, assigning tasks, evaluating performance, and streamlining communication within the organization.

i) Unit Testing:

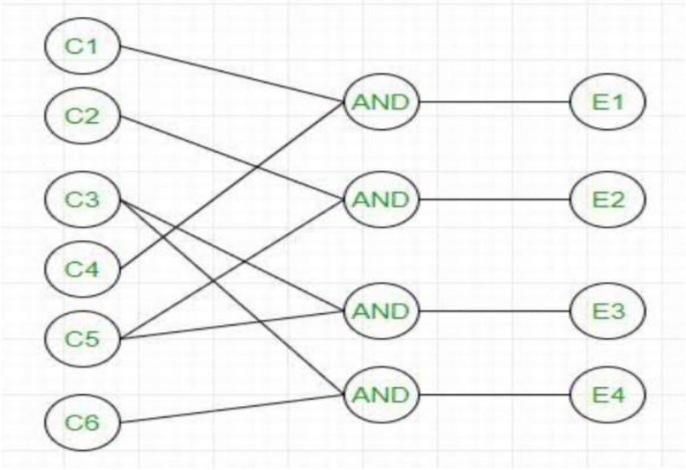
- Unit testing involves testing individual components or modules of the software to ensure they function correctly in isolation.
- Each module, such as employee database management, task allocation, performance evaluation, etc., would undergo unit testing.
- Example: Testing the "addClient()" function to ensure that a new employee is successfully added to the system.



ii) Black Box Testing:

- Black box testing is a technique where the internal workings of the system are not known to the tester. The tester only tests the system's functionality based on its specifications.

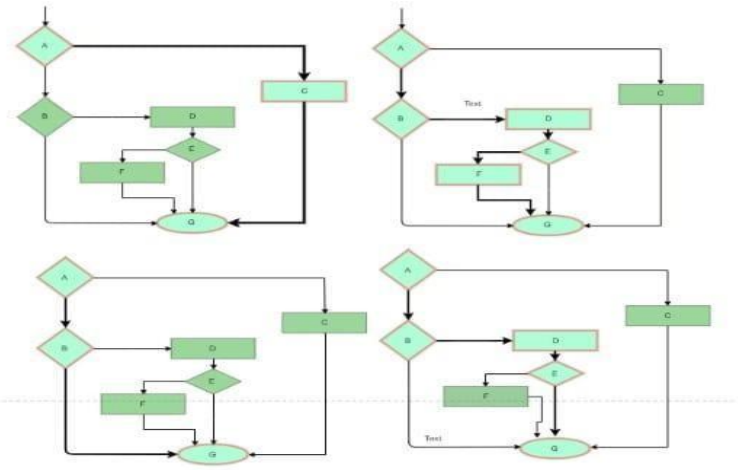
- Testers input various sets of data into the system and verify that the expected output is produced.
- Example: Testing the "searchForTrain()" functionality to ensure that it returns the expected results based on different search criteria.



		1	2	3	4
CAUSES	C1	1	0	0	0
	C2	0	1	0	0
	C3	0	0	1	1
	C4	1	0	0	0
	C5	0	1	1	0
	C6	0	0	0	1
EFFECTS	E1	x	-	-	-
	E2	-	x	-	-
	E3	-	-	x	-
	E4	-	-	-	x

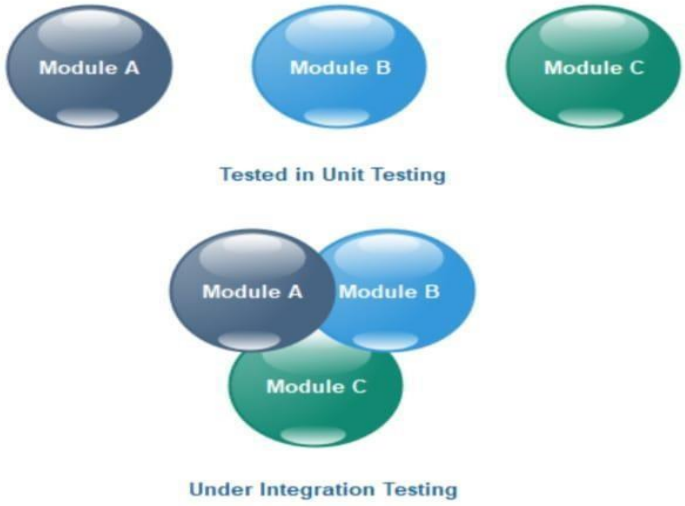
iii) White Box Testing:

- White box testing involves testing the internal logic and structure of the software code.
- Testers examine the code of individual modules to ensure that all code paths are tested.
- Example: Testing the "evaluatePerformance()" function to ensure that it adequately calculates employee performance metrics.



iv) Integration Testing:

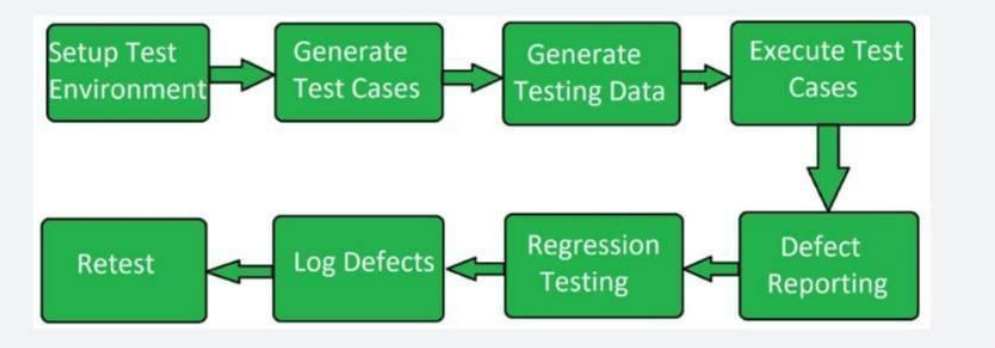
- Integration testing verifies that different modules of the software work together as expected.
- Testers test the interaction between different modules, such as employee database management, task allocation, and performance evaluation.
- Example: Testing the interaction between the "addClient()" and "assignBooking()" functions to ensure that tasks can be assigned to employees successfully.



v) System Testing:

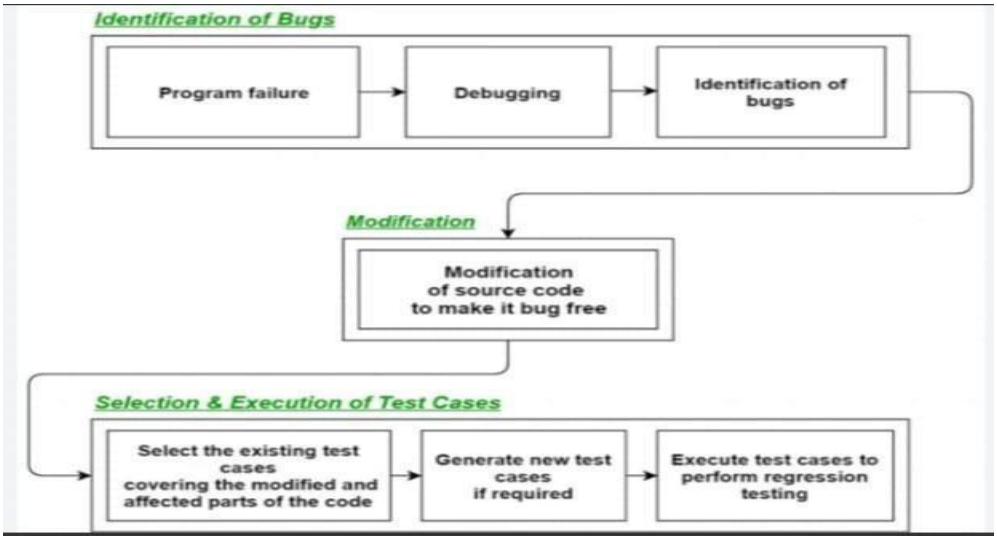
- System testing involves testing the entire system as a whole to verify that it meets the specified requirements.
- Testers perform end-to-end testing of the entire system, including all modules and their interactions.

- Example: Testing the entire process from client onboarding to booking completion and performance review to ensure that it functions as expected.



vi) Regression Testing:

- Regression testing ensures that new changes or additions to the system do not adversely affect existing functionalities.
- Testers re-run previously conducted tests after new changes or additions are made to the system to ensure that existing functionalities are not affected.
- Example: After adding a new feature to the system, testers would re-run all existing tests to ensure that the new feature did not introduce any bugs or errors.



Result:

Thus the software system for all scenarios identified in the use case diagram was tested successfully.

EX NO: 9 DATE:	**Improve the reusability and maintainability of the software by applying appropriate design patterns**

Aim:

To improve the reusability and maintainability of the software system by applying appropriate design patterns in Railway Reservation System.

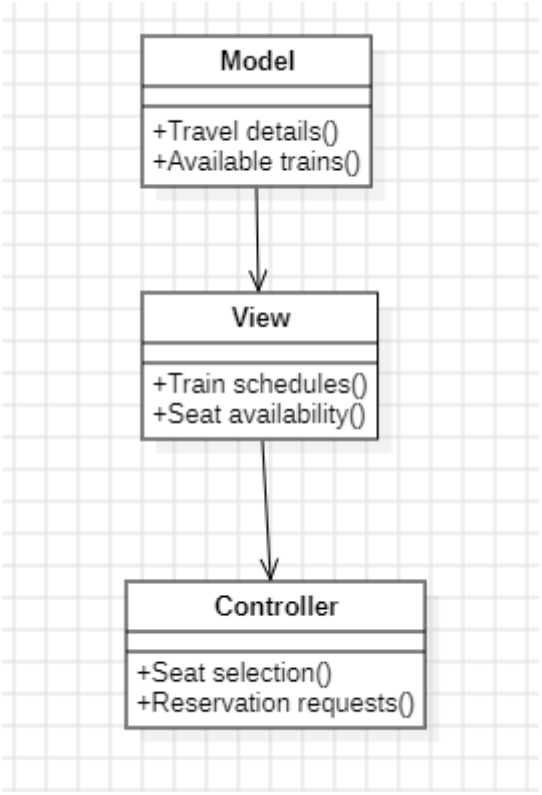
Railway Reservation System:

A Railway Reservation System facilitates the efficient management of human resources within an organization. It includes functionalities such as client data management, task allocation, performance evaluation, and communication.

To enhance the reusability and maintainability of the RRS management system, we can incorporate several design patterns. Here are some design patterns that can be applied:

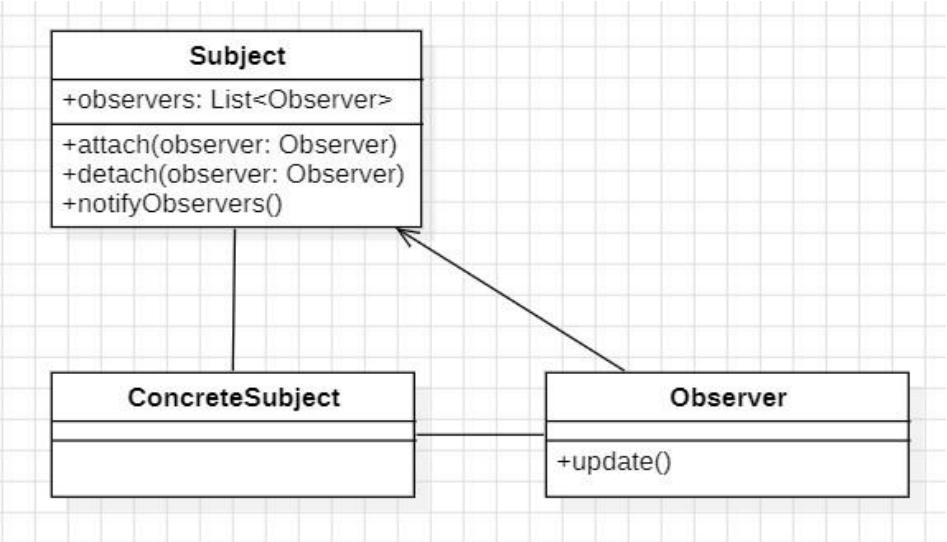
i) Model-View-Controller (MVC) Pattern:

The MVC pattern separates the representation of information from the user's interaction with it. In the Railway Reservation System, it divides the system into three interconnected components: Model (data and business logic), View (user interface), and Controller section.



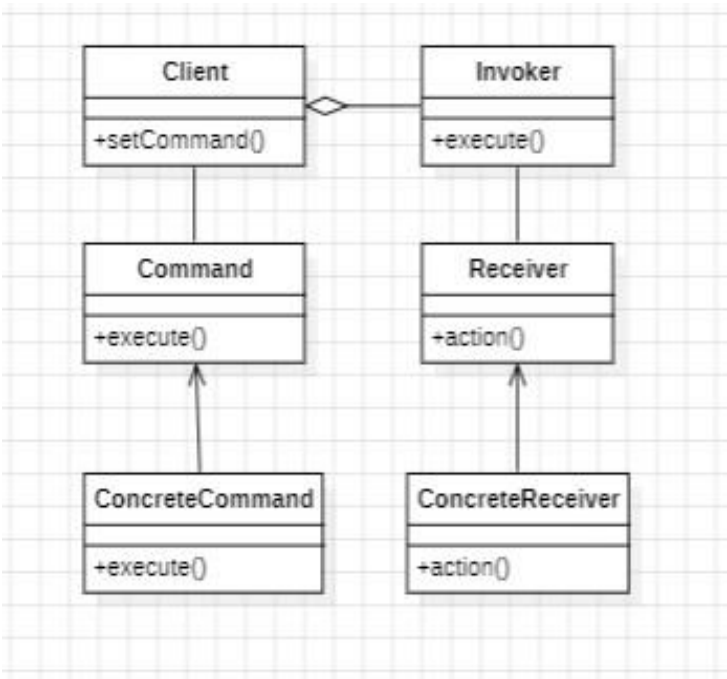
ii) Observer Pattern:

The Observer pattern establishes a one-to-many dependency between objects, where multiple observers are notified of changes in a subject. In the Railway Reservation System, it can be used to notify stakeholders about important events such as client onboarding, booking assignments, or performance reviews.



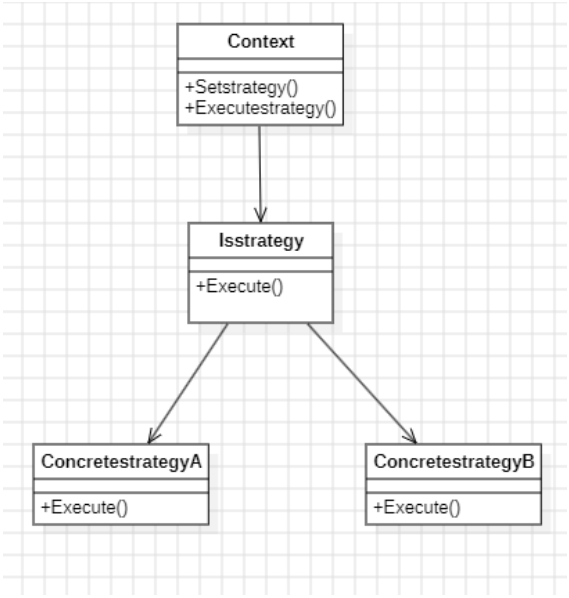
iii) Command Pattern:

The Command pattern encapsulates requests as objects, allowing for parameterization of clients with queued or logged requests. In the railway Reservation System, it can be used to implement features like undo/redo functionality for booking assignments, batch processing of client data updates, or logging of administrative actions.



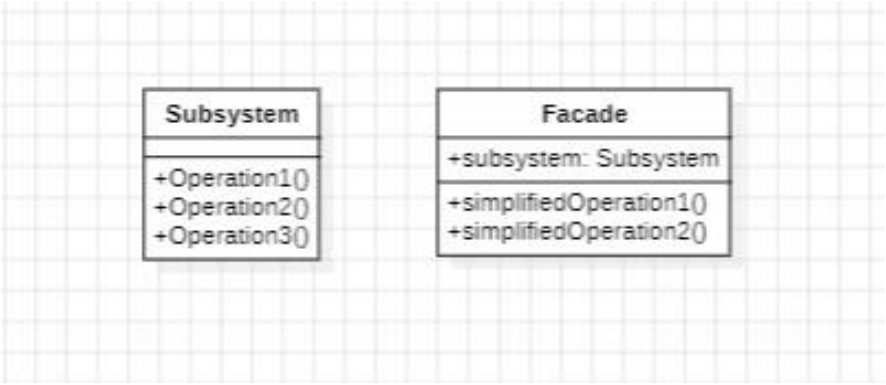
iv) Strategy Pattern:

The Strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. In the Railway Reservation System, it can be used to implement different strategies for booking prioritization, performance evaluation, or communication protocols based on organizational needs.



v) Facade Pattern:

The Facade pattern provides a simplified interface to a complex subsystem, hiding its complexities from clients. In the Railway Reservation System, it can be used to abstract away the complexities of client data management, booking allocation, or performance evaluation, providing a unified interface for interacting with these subsystems.



Result:

Thus the reusability and maintainability of the software system by applying appropriate design patterns was executed successfully.