

Experiment No. - 8

Name- Bhavesh Kewalramani

Roll No.- A-25

Section- A

Semester- 6th

Shift- 1st

Aim:

To study the IEEE SRS standard and prepare SRS for the identified systems conceptualization.

Theory:

What Is a Software Requirements Specification (SRS) Document?

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

An SRS can be simply summarized into four Ds:

- Define your product's purpose.
- Describe what you're building.
- Detail the requirements.
- Deliver it for approval.

We want to DEFINE the purpose of our product, DESCRIBE what we are building, DETAIL the individual requirements, and DELIVER it for approval. A good SRS document will define everything from how software will interact when embedded in hardware to the expectations when connected to other software. An even better SRS documents also account for real-life users and human interaction.

The best SRS documents define how the software will interact when embedded in hardware — or when connected to other software. Good SRS documents also account for real-life users.

Why Use an SRS Document?

An SRS gives you a complete picture of your entire project. It provides a single source of truth that every team involved in development will follow. It is your plan of action and keeps all your teams — from development to maintenance — on the same page (no pun intended).

This layout not only keeps your teams in sync but it also ensures that each requirement is hit. It can ultimately help you make vital decisions on your product's lifecycle, such as when to retire an obsolete feature.

The time it takes to write an SRS is given back in the development phase. It allows for better understanding of your product, team, and the time it will take to complete.

Software Requirements Specification vs. System Requirements Specification

A **software requirements specification (SRS)** includes in-depth descriptions of the software that will be developed.

A **system requirements specification (SyRS)** collects information on the requirements for a system.

“Software” and “system” are sometimes used interchangeably as SRS. But, a software requirement specification provides greater detail than a system requirements specification.

How to Write an SRS Document

Writing an SRS document is important. But it isn't always easy to do.

Here are five steps you can follow to write an effective SRS document.

1. Define the Purpose With an Outline (Or Use an SRS Template)

Your first step is to create an outline for your software requirements specification. This may be something you create yourself. Or you may use an existing SRS template.

If you're creating this yourself, here's what your outline might look like:

1. Introduction

- 1.1 Purpose

- 1.2 Intended Audience

- 1.3 Intended Use

- 1.4 Scope

- 1.5 Definitions and Acronyms

2. Overall Description

- 2.1 User Needs

- 2.2 Assumptions and Dependencies

3. System Features and Requirements

3.1 Functional Requirements

3.2 External Interface Requirements

3.3 System Features

3.4 Nonfunctional Requirements

This is a basic outline and yours may contain more (or fewer) items. Now that you have an outline, let's fill in the blanks.

2. Define your Product's Purpose

This introduction is very important as it sets expectations that we will hit throughout the SRS.

Some items to keep in mind when defining this purpose include:

Intended Audience and Intended Use

Define who in your organization will have access to the SRS and how they should use it. This may include developers, testers, and project managers. It could also include stakeholders in other departments, including leadership teams, sales, and marketing. Defining this now will lead to less work in the future.

Product Scope

What are the benefits, objectives, and goals we intend to have for this product? This should relate to overall business goals, especially if teams outside of development will have access to the SRS.

Definitions and Acronyms

It's important to define the risks in the project. What could go wrong? How do we mitigate these risks? Who is in charge of these risk items?

For example, if the failure of a medical device would cause slight injury, that is one level of risk. Taking into account the occurrence level and the severity, we can come up with a strategy to mitigate this risk.

3. Describe What You Will Build

Your next step is to give a description of what you're going to build. Is it a new product? Is it an add-on to a product you've already created? Is this going to integrate with another product? Why is this needed? Who is it for?

Understanding these questions on the front end makes creating the product much easier for all involved.

User Needs

Describe who will use the product and how. Understanding the user of the product and their needs is a critical part of the process.

Who will be using the product? Are they a primary or secondary user? Do you need to know about the purchaser of the product as well as the end user? In medical devices, you will also need to know the needs of the patient.

Assumptions and Dependencies

What are we assuming will be true? Understating and laying out these assumptions ahead of time will help with headaches later. Are we assuming current technology? Are we basing this on a Windows framework? We need to take stock of these assumptions to better understand when our product would fail or not operate perfectly.

Finally, you should note if your project is dependent on any external factors. Are we reusing a bit of software from a previous project? This new project would then depend on that operating correctly and should be included.

4. Detail Your Specific Requirements

In order for your development team to meet the requirements properly, we **MUST** include as much detail as possible. This can feel overwhelming but becomes easier as you break down your requirements into categories. Some common categories are:

Functional Requirements

Functional requirements are essential to your product because, as they state, they provide some sort of functionality.

Asking yourself the question “does this add to my tool’s functionality?” Or “What function does this provide?” can help with this process. Within Medical devices especially, these functional requirements may have a subset of risks and requirements.

You may also have requirements that outline how your software will interact with other tools, which brings us to external interface requirements.

External Interface Requirements

External interface requirements are specific types of functional requirements. These are especially important when working with embedded systems. They outline how your product will interface with other components.

There are several types of interfaces you may have requirements for, including:

- User
- Hardware
- Software
- Communications

System Features

System features are types of functional requirements. These are features that are required in order for a system to function.

Other Nonfunctional Requirements

Nonfunctional requirements can be just as important as functional ones.

These include:

- Performance
- Safety
- Security
- Quality

The importance of this type of requirement may vary depending on your industry. In the medical device industry, there are often regulations that require the tracking and accounting of safety.

IEEE also provides guidance for writing software requirements specifications, if you're a member.

5. Deliver for Approval

We made it! After completing the SRS, you'll need to get it approved by key stakeholders. This will require everyone to review the latest version of the document.

Writing an SRS in Microsoft Word vs. Requirement Software

You can write your software requirement specification in Microsoft Word. A smart way to do this is to create an SRS template that you can use as a starting point for every project.

However, even with a template, writing an SRS this way can be a painstaking process. And if a requirement changes, your SRS can fall easily out-of-date. Plus, there can be versioning issues with requirements documents in Word.

Conclusion:

SRS helps the customers to define their needs with accuracy, while it helps the development team understand what the customers need in terms of development. Investing time in writing the SRS document will lead to the successful development of the software the customers need.

SRS For E-IRCTC Reservation System



Table of Contents

1. INTRODUCTION

- 1.1. Objective
- 1.2. Scope
- 1.3. Glossary
- 1.4. Overview

2. OVERALL DESCRIPTION

- 2.1. Product Perspective
- 2.2. Product Functions
- 2.3. User Characteristics
- 2.4. Constrains
- 2.5. Assumptions and Dependencies
- 2.6. Apportioning of requirements

3. REQUIRMENT SPECIFICATION

- 3.1. Function Requirements
 - 3.1.1. Performance Requirements
 - 3.1.2. Design Constraints
 - 3.1.3. Hardware Requirements
 - 3.1.4. Software Requirements
 - 3.1.5. Other Requirements
 - 3.2. Non-Function Requirement
-

- 3.2.1. Security
- 3.2.2. Reliability
- 3.2.3. Availability
- 3.2.4. Maintainability
- 3.2.5. Supportability

4. DIAGRAM

- 4.1. Use Case Diagram
- 4.2. Class Diagram
- 4.3. Sequence Diagram
- 4.4. Collaboration Diagram
- 4.5. Activity Diagram
- 4.6. Statechart Diagram
- 4.7. Component Diagram
- 4.8. Deployment Diagram

5. GUI

- 5.1. Screen Shots

6. REFERENCES



Introduction



1.Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS purpose ,scope, definitions, acronyms, abbreviations, references and overview of SRS.A **Software Requirements Specification (SRS)** - a [requirements specification](#) for a [software system](#) - is a complete description of the behaviour of a system to be developed. It includes a set of [use cases](#) that describe all the interactions the users will have with the software. Use cases are also known as [functional requirements](#). In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. [Non-functional requirements](#) are requirements which impose constraints on the design or implementation (such as [performance engineering](#) requirements, [quality](#) standards, or design constraints). The aim of this document is to gather and analyse and give an in-depth insight of the complete Marvel Electronics and Home Entertainment software system by defining the problem statement in detail. This is a documentation of the project **E-IRCTC Reservation System** done sincerely and satisfactorily by me. A Software has to be developed for automating the Online Railway Reservation System.

1.1 Objective:

The purpose of this source is to describe the railway reservation system which provides the train timing details, reservation, billing and cancellation on various types of reservation namely,

- Book Tickets – Booking form has to be filled by passenger. If seats are available entries like train name, number, destination are made. The amount will be paid by the user using valid credit/debit card.
- Cancel Tickets – Cancellation form has to be filled by passenger. The tickets will get cancelled and the amount will be refunded.
- Update Tickets - The user deletes the entry in the System and changes in the Reservation Status.
- VIEW STATUS - The user needs to enter the PNR number printed on ticket.
- Search Trains – The user can search the train and find the info and route. He/She can also check the seat availability in that particular train.

The origin of most software systems is in the need of a client, who either wants to automate the existing manual system or desires a new software system. The software system is itself created by the developer. Finally, the end user will use the completed system. Thus, there are three major parties interested in a new system: the client, the user, and the developer. Somehow the requirements for the system that will satisfy the needs of the clients and the

concerns of the users have to be communicated to the developer. The problem is that the client doesn't usually design the software or the software development process and the developer does not understand the client's problem and the application area. This causes a communication gap between the parties involved in the development of the project.

The basic purpose of Software Requirement Specification (SRS) is to bridge this communication gap. SRS is the medium through which the client's and the user's needs are accurately specified; indeed SRS forms the basis of software development.

Another important purpose of developing an SRS is helping the clients understanding their own needs. An SRS establishes the basis for agreement between the client and the supplier on what the software product will do.

An SRS provides a reference for validation of the final product. A high quality SRS is a prerequisite to high quality software and it also reduces the development cost.

A few factors that direct us to develop a new system are given below -:

1. Faster System
2. Accuracy
3. Reliability
4. Informative
5. Reservations and cancellations from anywhere to any place

1.2 Scope:

"E-IRCTC Reservation System" is an attempt to simulate the basic concepts of an online Reservation system. The system enables to perform the following functions:

- Allow the user to create the account and validate it
- Search the train
- Check train availability and seat availability
- Fill the form, book the ticket and pay the ticket
- Fill the form and cancel the tickets
- Update the details and then the ticket will be updated
- Check train status using PNR number

1.3 Abbreviations, Definitions and Acronyms:

This should define all technical terms and abbreviations used in the document

NTES	National Train Enquiry System
IVRS	Interactive Voice Response system
PRS	passenger reservation system
SRS	Software Requirements Specification
STD	State Transition Diagram
IRCTC	Indian Railway Catering and Tourism Corporation
E-IRCTC	Online Indian Railway Catering and Tourism Corporation

1.4 Overview:

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. General description of the project is discussed in section 2 of this document. Section 3 gives the functional requirements, data requirements and constraints and assumptions made while designing the website. It also gives the user viewpoint of product. Section 3 also gives the specific requirements of the product. Section 3 also discusses the external interface requirements and gives detailed description of functional requirements. Section 4 is for supporting information.





Overall Description

2. Overall Description

This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company. It further contains a list of the stakeholders and users of the proposed solution. It also illustrates the needs and wants of the stakeholders that were identified in the brainstorming exercise as part of the requirements workshop. It further lists and briefly describes the major features and a brief description of each of the proposed system.

2.1 Product Perspective:

Before the automation, the system suffered from the following DRAWBACKS:

- The existing system is highly manual involving a lot of paper work and calculation and therefore may be erroneous. This has lead to inconsistency and inaccuracy in the maintenance of data.
- The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural calamity like fire and water.
- The existing system is sluggish and consumes a lot of time causing inconvenience to customers and the railway staff.
- Due to manual nature, it is difficult to update, delete, add or view the data.
- Since the number of passengers have drastically increased therefore maintaining and retrieving detailed record of passenger is extremely difficult.
- Railways has many offices around the Country, an absence of a link between these offices can lead to lack of coordination and communication.

Hence the railways reservation system is proposed with the following

- The computerization of the reservation system will reduce a lot of paperwork and hence the load on the railway administrative staff.
 - The machine performs all calculations. Hence chances of error are nil.
 - The passenger, reservation, cancellation list can easily be retrieved and any required addition, deletion or updation can be performed.
 - The system provides for user-ID validation, hence unauthorized access is prevented.
-

2.2 Project Functions:

Booking agents with varying levels of familiarity with computers will mostly use this system. With this in mind, an important feature of this software is that it be relatively simple to use. The scope of this project encompasses: -

Search: This function allows the booking agent to search for train that are available between the two travel cities, namely the "Departure city" and "Arrival city" as desired by the traveller. The system initially prompts the agent for the departure and arrival city, the date of departure, preferred time slot and the number of passengers. It then displays a list of train available with different trains between the designated cities on the specified date and time.

Selection: This function allows a particular train to be selected from the displayed list. All the details of the train are shown:

1. Train Number
2. Date, time and place of departure
3. Date, time and place of arrival
4. Train Duration
5. Fare per head
6. Number of stoppages – 0, 1, 2...

Review: If the seats are available, then the software prompts for the booking of train. The train information is shown. The total fare including taxes is shown and train details are reviewed.

Traveller Information: It asks for the details of all the passengers supposed to travel including name, address, mobile number and e-mail id.

Payment: It asks the user to enter the valid credit card details of the person making the reservation.

1. Credit card type
2. Credit card number
3. CVC number of the card
4. Expiration date of the card
5. The name of the card holder

Cancellation: The system also allows the passenger to cancel an existing reservation. This function registers the information regarding a passenger who has requested for a cancellation of his/her ticket. It includes entries pertaining to the train No., Confirmation No., Name, Date of Journey, Fare deducted.

View Ticket and Train Details: From admin's side, he/she can the booked tickets, cancelled tickets and updated tickets. He/she can also check the train status and schedule with date,time and routes.

Generate Report: Provision for generation of different reports should be given in the system. The system should be able to generate reservation chart, monthly train report etc.

Verify login: For security reasons all the users of the system are given a user id and a password. Only if the id and password are correct is the user allowed entry to the system and select from the options available in the system.

2.3 User Characteristics:

- **EDUCATIONAL LEVEL:-** At least user of the system should be comfortable with English language._
- **TECHNICAL EXPERTISE:** - User should be comfortable using general purpose applications on the computer system.

Users of Project:

Normal Educated Person/Clerk: This person uses this system to create reservation, cancel reservation, view reservation status, update reservation details, view train schedule.

Manager/Admin: This person uses this system to update train information and to generate reports.

2.4 Constraints:

Software constraints:

- The system will run under windows98 or higher platforms of operating system.
- SQL used will be Oracle 11g.

User constraints:

- User must have a good knowledge of computer and net banking
- User must be have a good internet connection to access website

Hardware constraints:

- No specific hardware requirements

2.5 Assumptions and Dependencies:

- User will be having a valid user name and password to access the software
- The user needs to have complete knowledge of railways reservation system, computer and browsing websites.
- Software is dependent on access to internet.





*Requirement
Specification*

3.1 Function Requirements

3.1.1 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.
- **Error Handling:** - Response to user errors and undesired situations has been taken care of to ensure that the system operates without halting.
- **Safety and Robustness:** - The system is able to avoid or tackle disastrous action. In other words, it should be fool proof. The system safeguards against undesired events, without human intervention.
- **Portable:** - The software should not be architecture specific. It should be easily transferable to other platforms if needed.
- **User friendliness:** - The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.

3.1.2 Design constraint:

There are a number of factors in the client's environment that may restrict the choices of a designer. Such factors include standards that must be followed, resource limits, operating environment, reliability and security requirements and policies that may have an impact on the design of the system. An SRS (Software Requirements Analysis and Specification) should identify and specify all such constraints.

Standard Compliance: - This specifies the requirements for the standards the system must follow. The standards may include the report format and accounting properties.

Hardware Limitations :- The software may have to operate on some existing or predetermined hardware, thus imposing restrictions on the design. Hardware limitations can include the types of machines to be used, operating system available on the system, languages supported and limits on primary and secondary storage.

Reliability and Fault Tolerance: - Fault tolerance requirements can place a major constraint on how the system is to be designed. Fault tolerance requirements often make the system more complex and expensive. Requirements about system behavior in the face of certain kinds of faults are specified. Recovery requirements are often an integral part here, detailing what the system should do if some failure occurs to ensure certain properties. Reliability requirements are very important for critical applications.

Security: - Security requirements are particularly significant in defence systems and database systems. They place restrictions on the use of certain commands, control access to data,

provide different kinds of access requirements for different people, require the use of passwords and cryptography techniques and maintain a log of activities in the system.

3.1.3 Hardware requirements:

For the hardware requirements the SRS specifies the logical characteristics of each interface b/w the software product and the hardware components. It specifies the hardware requirements like memory restrictions, cache size, the processor, RAM size etc... those are required for the software to run.

Minimum Hardware Requirements

Processor Pentium III

Hard disk drive 40 GB

RAM 128 MB

Cache 512 kb

Preferred Hardware Requirements

Processor Pentium IV

Hard disk drive 80 GB

RAM 256 MB

Cache 512 kb

3.1.4 Software requirements:

- Any window based operating system with DOS support are primary requirements for software development. Windows XP, FrontPage and dumps are required. The systems must be connected via LAN and connection to internet is mandatory.

3.1.5 other requirements:

Software should satisfy following requirements as well:-

- User Interface Requirements:
-

1. A login screen is provided in the beginning for entering the required username/pin no. and account number.
2. An unsuccessful login leads to a reattempt (maximum three) screen for again entering the same information. The successful login leads to a screen displaying a list of supported languages from which a user can select anyone.
3. In case of administrator, a screen will be shown having options to reboot system, shut down system, block system, disable any service.
4. In case of reboot/ shut down, a screen is displayed to confirm the user's will to reboot and also allow
5. After the login, a screen with a number of options is then shown to the user. It contains all the options along with their brief description to enable the user to understand their functioning and select the proper option.
6. A screen will be provided for user to check his account balance.
7. A screen will be provided that displays the location of all other ATMs of same bank elsewhere in the city.
8. A screen will be provided for the user to perform various transactions in his account.

Other various user interface requirements that need to be fulfilled are as follows:-

The display screen shall be of 10" VGA color type.

The display screen shall have 256 color resolution.

The display screen shall also support touch screen facility. The speakers shall support Yamaha codecs.

The keypad shall consist of 16 tactile keys. There shall be 8 tactile function keys. The keyboard will be weather resistant. The transaction receipt shall be 3.1" × 6". The statement receipt shall be 4.2" × 12". The deposit envelopes shall be 9" long and 4" wide the user to take any backup if needed.

- Communication Interface Requirements

The machine needs to communicate with the main branch for each session for various functions such as login verification, account access etc. so the following are the various communication interface requirements that are needed to be fulfilled in order to run the software successfully:-

The system will employ dial-up POS with the central server for low-cost communication. The communication protocol used shall be TCP/IP. Protocol used for data transfer shall be File Transfer Protocol.(FTP)

- SECURITY
 - PORTABILITY
 - CORRECTNESS
-

- EFFICIENCY
- FLEXIBILITY
- TESTABILITY
- REUSABILITY

3.2 Non-Function Requirements

3.2.1 Security:

The system use SSL (secured socket layer) in all transactions that include any confidential customer information. The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's password. The system's back-end servers shall only be accessible to authenticated management.

3.2.2 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 to reflect the most recent changes. Also the system will be functioning inside a container. Thus the overall stability of the system depends on the stability of container and its underlying operating system.

3.2.3 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. In case of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the Organizer. Then the service will be restarted. It means 24 x 7 availability.

3.2.4 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. Also the software design is being done with modularity in mind so that maintainability can be done efficiently.

3.2.5 Supportability:

The code and supporting modules of the system will be well documented and easy to understand. Online User Documentation and Help System Requirements.



Diagram



4.1 Use-case Diagram

The Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

When to apply use case diagrams

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case

Use case diagram components

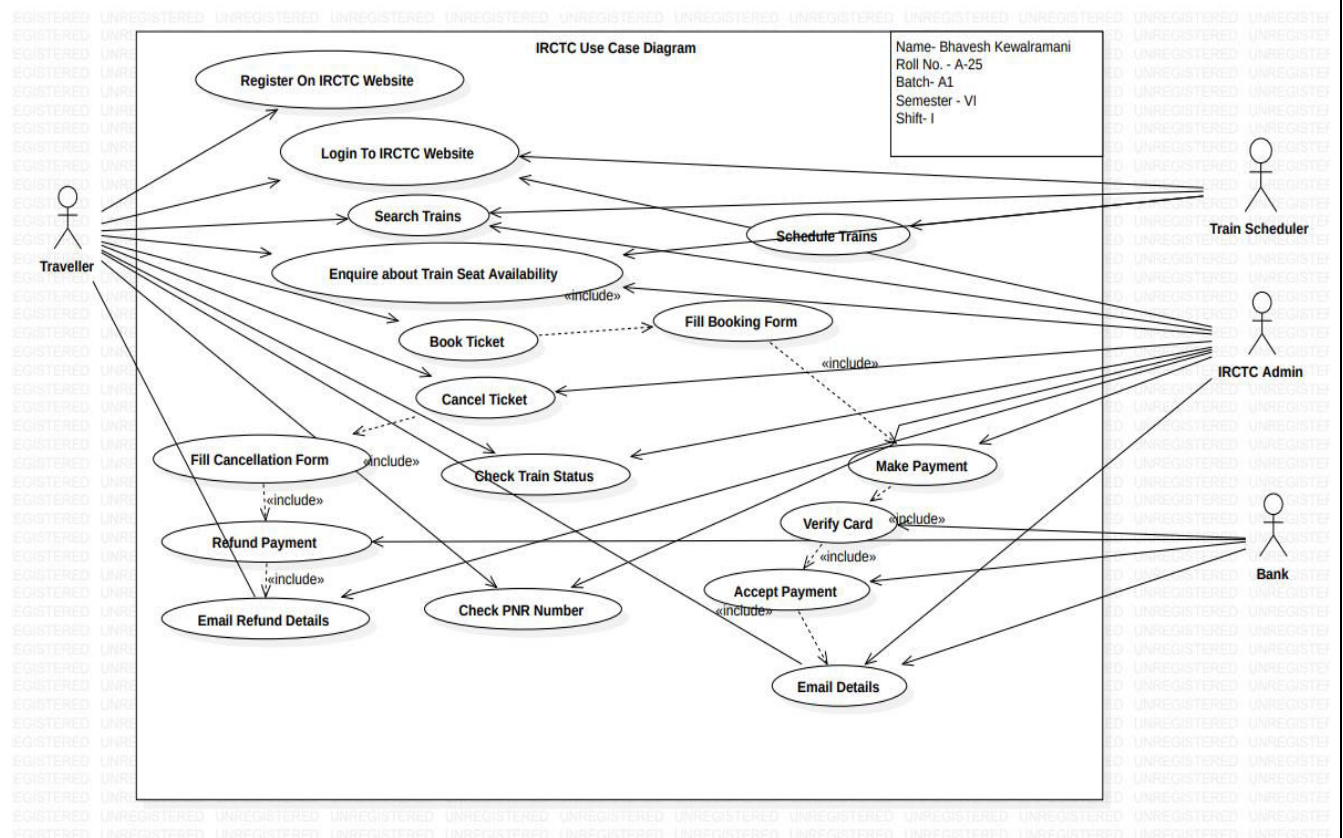
To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

- **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
-

- **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

Use case diagram symbols and notation

- **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** Stick figures that represent the people actually employing the use cases.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.



4.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

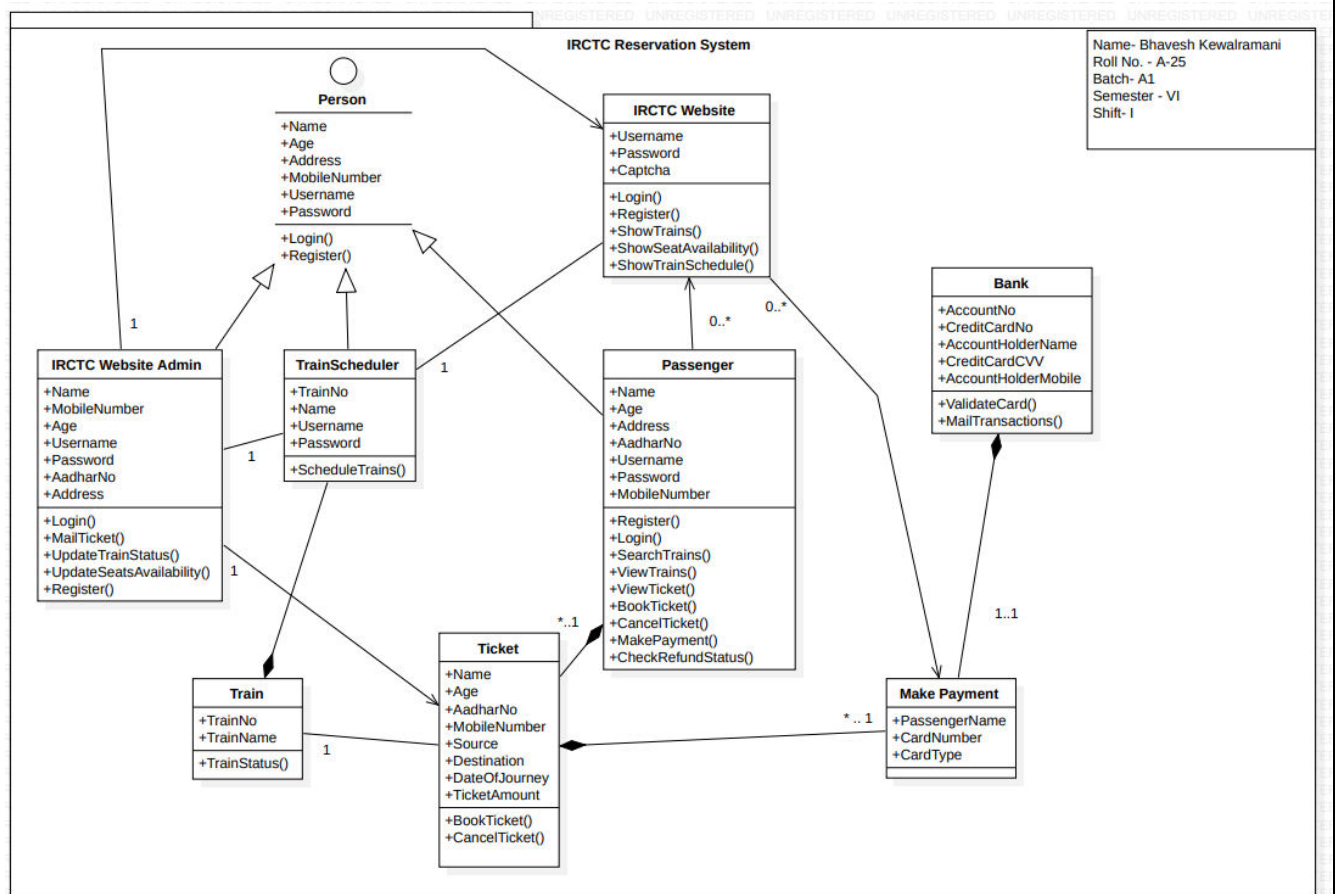
The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.





Name- Bhavesh Kewalramani
Roll No. - A-25
Batch- A1
Semester - VI
Shift- I

4.3 Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

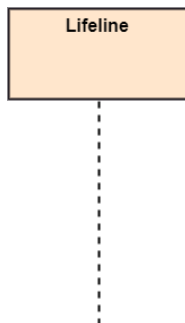
Purpose of a Sequence Diagram

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either models generic interactions or some certain instances of interaction.

Notations of a Sequence Diagram

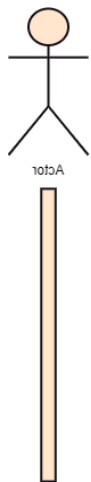
Lifeline

An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram.



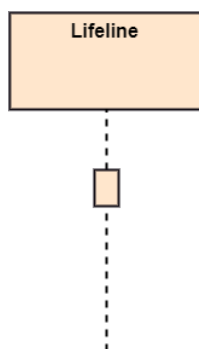
Actor

A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.



Activation

It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.

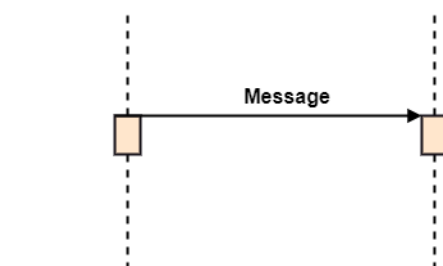


Messages

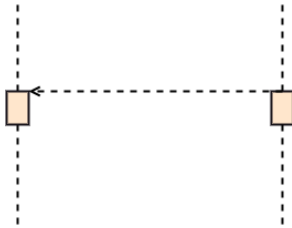
The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

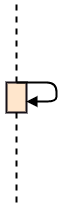
- **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.



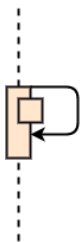
- **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.



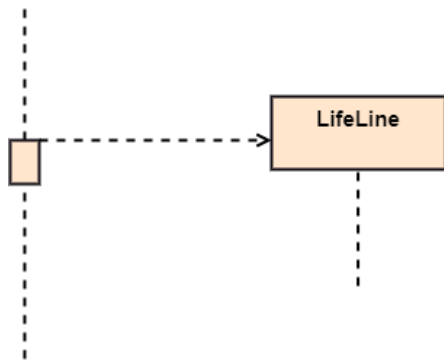
- **Self Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.



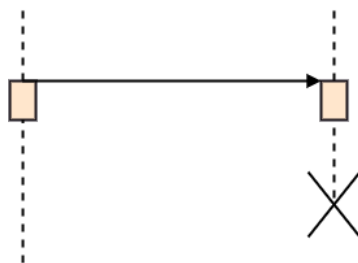
- **Recursive Message:** A self message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self message as it represents the recursive calls.



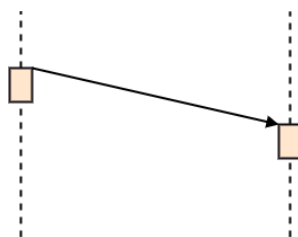
- **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.



- **Destroy Message:** It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.



- **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modeling a system.



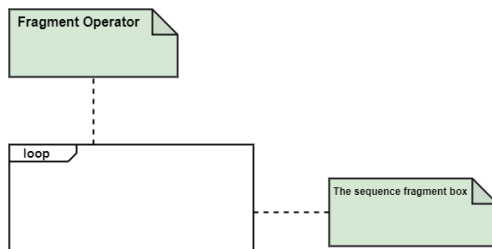
Note

A note is the capability of attaching several remarks to the element. It basically carries useful information for the modelers.



Sequence Fragments

1. Sequence fragments have been introduced by UML 2.0, which makes it quite easy for the creation and maintenance of an accurate sequence diagram.
2. It is represented by a box called a combined fragment, encloses a part of interaction inside a sequence diagram.
3. The type of fragment is shown by a fragment operator.



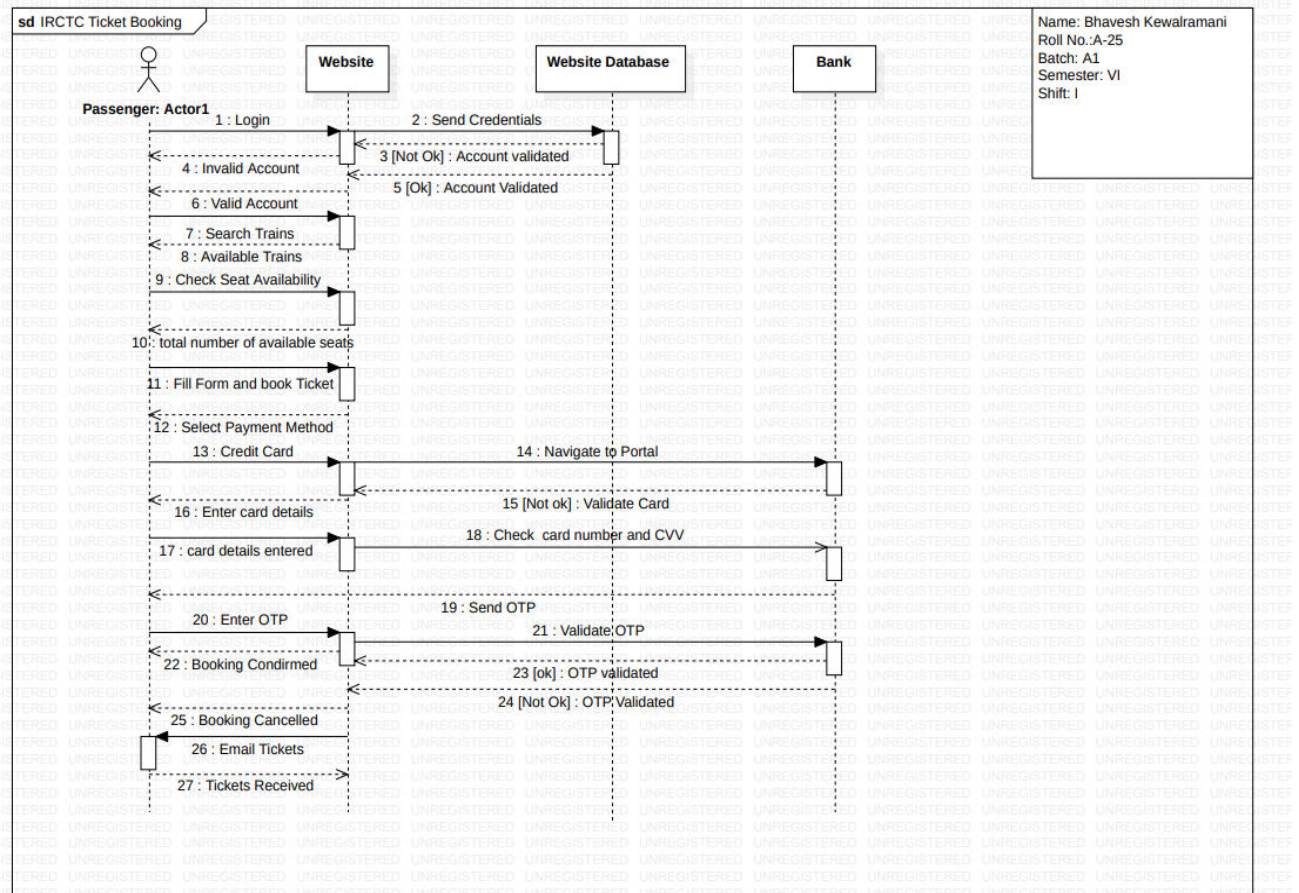
Types of fragments

Following are the types of fragments enlisted below;

Operator	Fragment Type
Alt	Alternative multiple fragments: The only fragment for which the condition is true, will execute.
Opt	Optional: If the supplied condition is true, only then the fragments will execute. It is similar to alt with only one trace.
Par	Parallel: Parallel executes fragments.
Loop	Loop: Fragments are run multiple times, and the basis of interaction is shown by the guard.
Region	Critical region: Only one thread can execute a fragment at once.
Neg	Negative: A worthless communication is shown by the fragment.
Ref	Reference: An interaction portrayed in another diagram. In this, a frame is drawn so as to cover the lifelines involved in the communication. The parameter and return value can be explained.

Sd

Sequence Diagram: It is used to surround the whole sequence diagram.



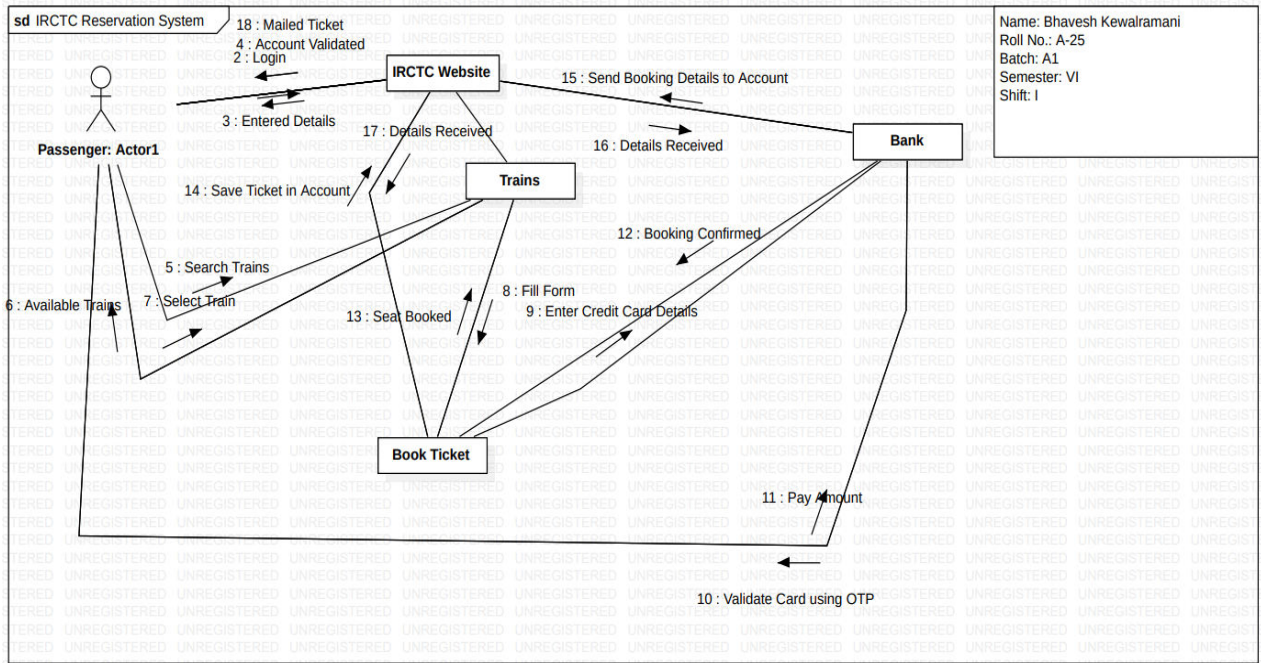
4.4 Collaboration Diagram

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.

Notations of a Collaboration Diagram

Following are the components of a component diagram that are enlisted below:

1. **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.
In the collaboration diagram, objects are utilized in the following ways:
 - The object is represented by specifying their name and class.
 - It is not mandatory for every class to appear.
 - A class may constitute more than one object.
 - In the collaboration diagram, firstly, the object is created, and then its class is specified.
 - To differentiate one object from another object, it is necessary to name them.
 2. **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
 3. **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.
 4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.
-



4.5 Activity Diagram

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

Components of an Activity Diagram

Following are the component of an activity diagram:

Activities

The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.

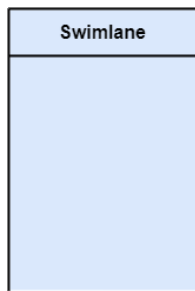
The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity. The activities are initiated at the initial node and are terminated at the final node.



Activity partition /swimlane

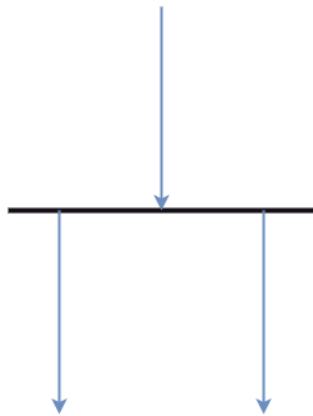
The swimlane is used to cluster all the related activities in one column or one row. It can be either vertical or horizontal. It used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.





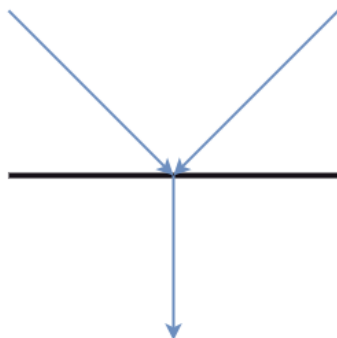
Forks

Forks and join nodes generate the concurrent flow inside the activity. A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.



Join Nodes

Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.



Pins

It is a small rectangle, which is attached to the action rectangle. It clears out all the messy and complicated thing to manage the execution flow of activities. It is an object node that precisely represents one input to or output from the action.

Notation of an Activity diagram

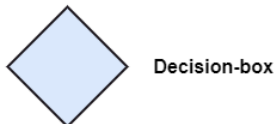
Activity diagram constitutes following notations:

Initial State: It depicts the initial stage or beginning of the set of actions.

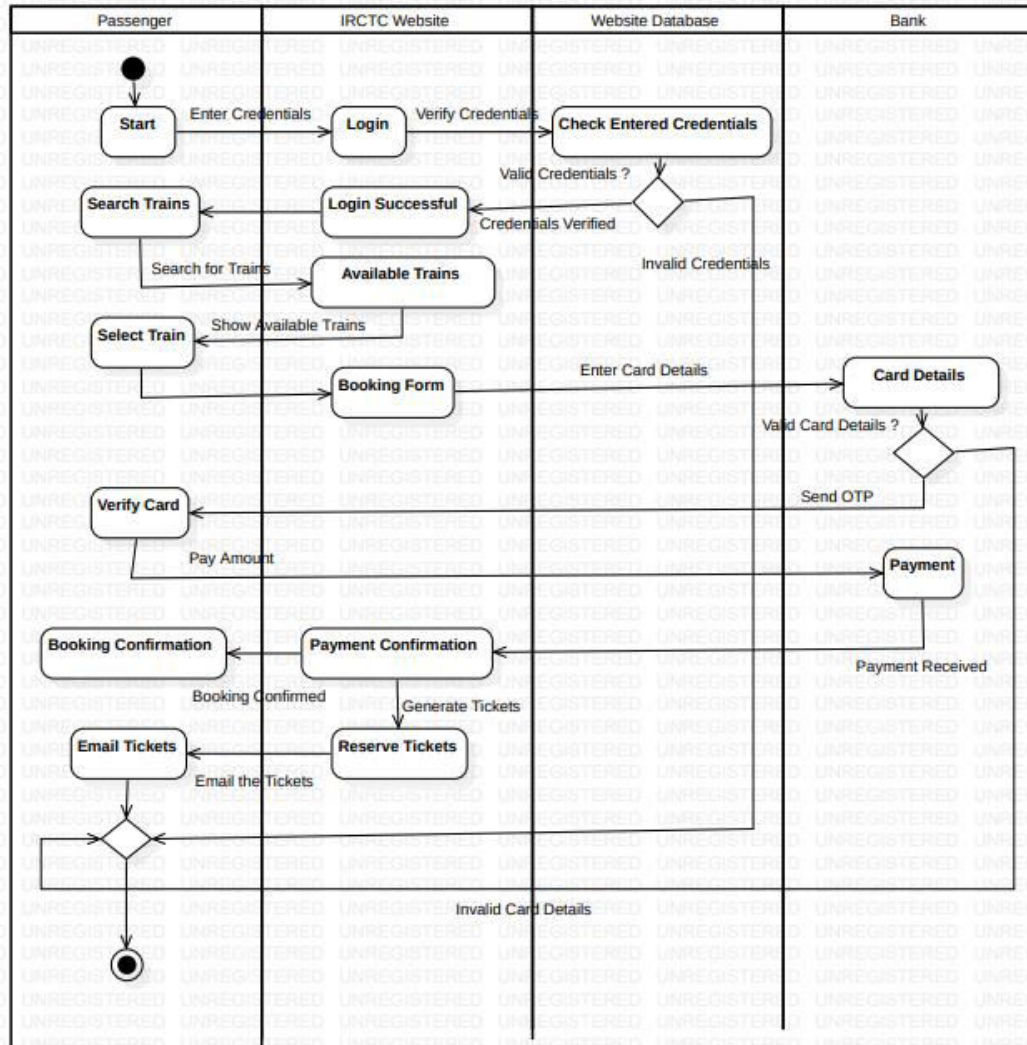
Final State: It is the stage where all the control flows and object flows end.

Decision Box: It makes sure that the control flow or object flow will follow only one path.

Action Box: It represents the set of actions that are to be performed.



Name: Bhavesh Kewalramani
Roll No.: A-25
Batch: A1
Semester: VI
Shift: I



4.6 Statechart Diagram

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in the next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines the states, it is used to model the lifetime of an object.

Purpose of Statechart Diagrams

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

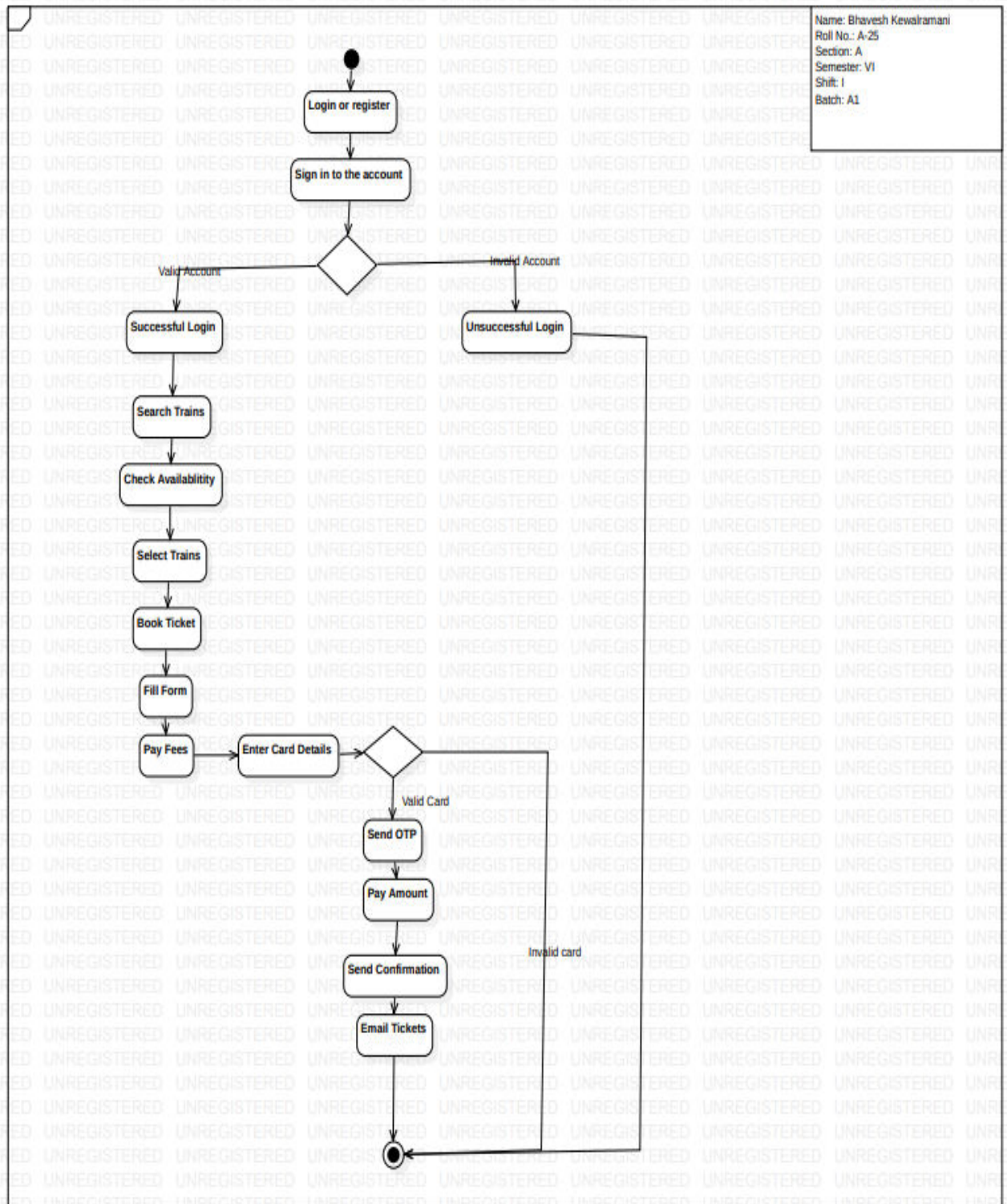
Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.



Name: Bhavesh Kewalramani
Roll No.: A-25
Section: A
Semester: VI
Shift: I
Batch: A1



4.7 Component Diagram

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose of Component Diagrams

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as –

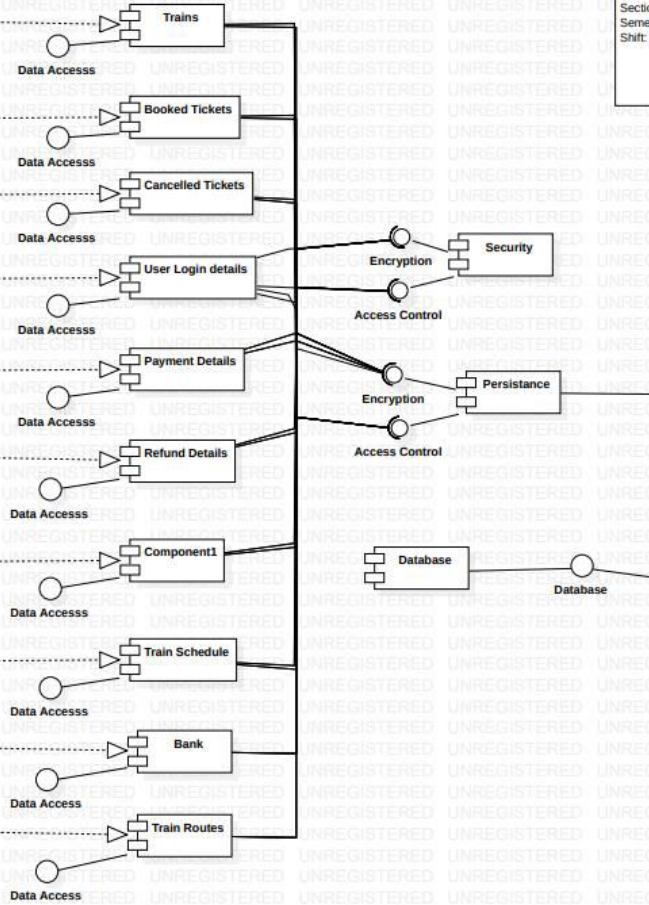
- Visualize the components of a system.
 - Construct executables by using forward and reverse engineering.
 - Describe the organization and relationships of the components.
-

IRCTC Reservation System

Name: Bhavesh Kewlamani
Roll No.: A-25
Section: A
Semester: VI
Shift: I

E-IRCTC Reservation System

Website Admin



4.8 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose of Deployment Diagrams

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related.

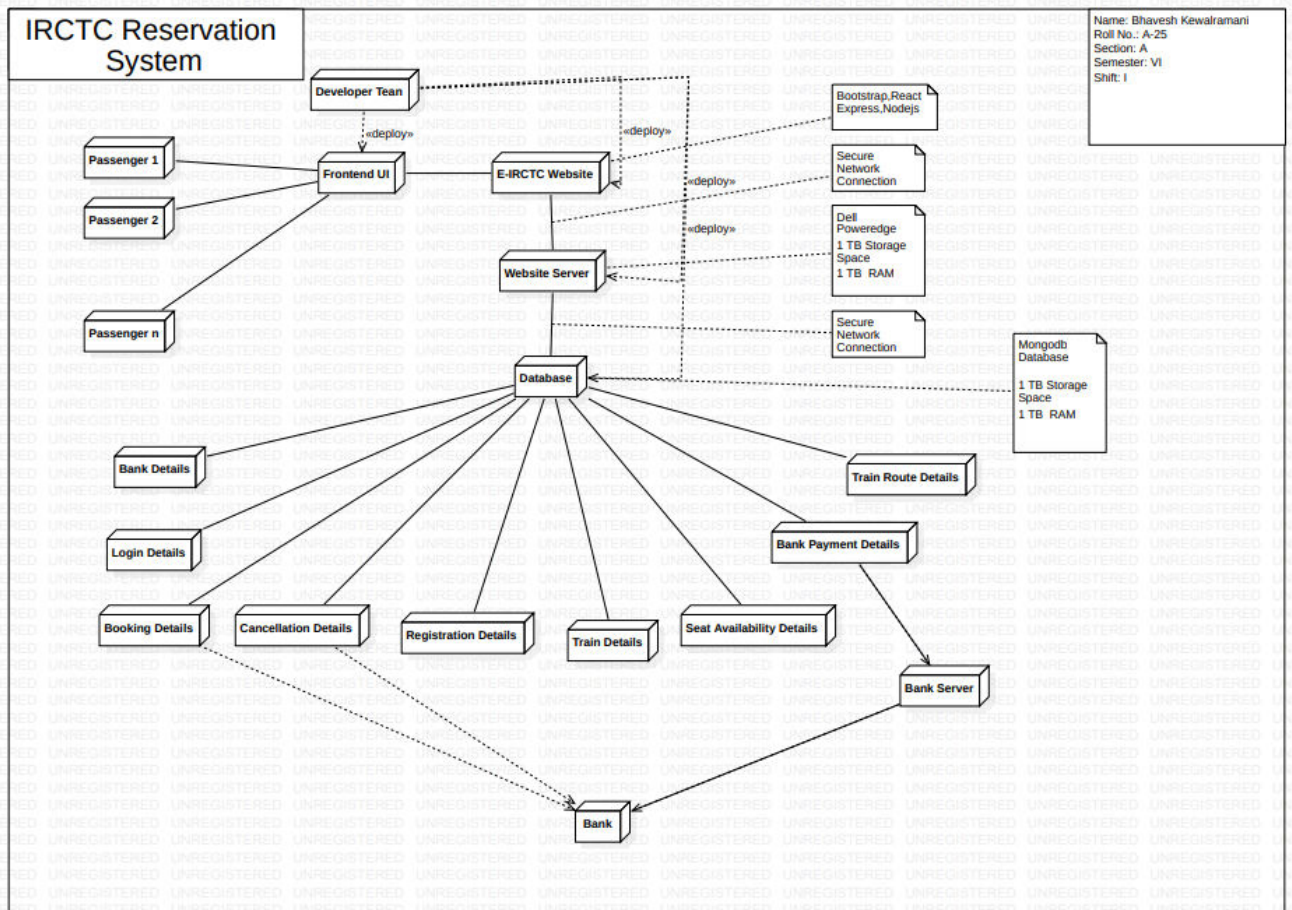
Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as –

- Visualize the hardware topology of a system.
 - Describe the hardware components used to deploy software components.
 - Describe the runtime processing nodes.
-





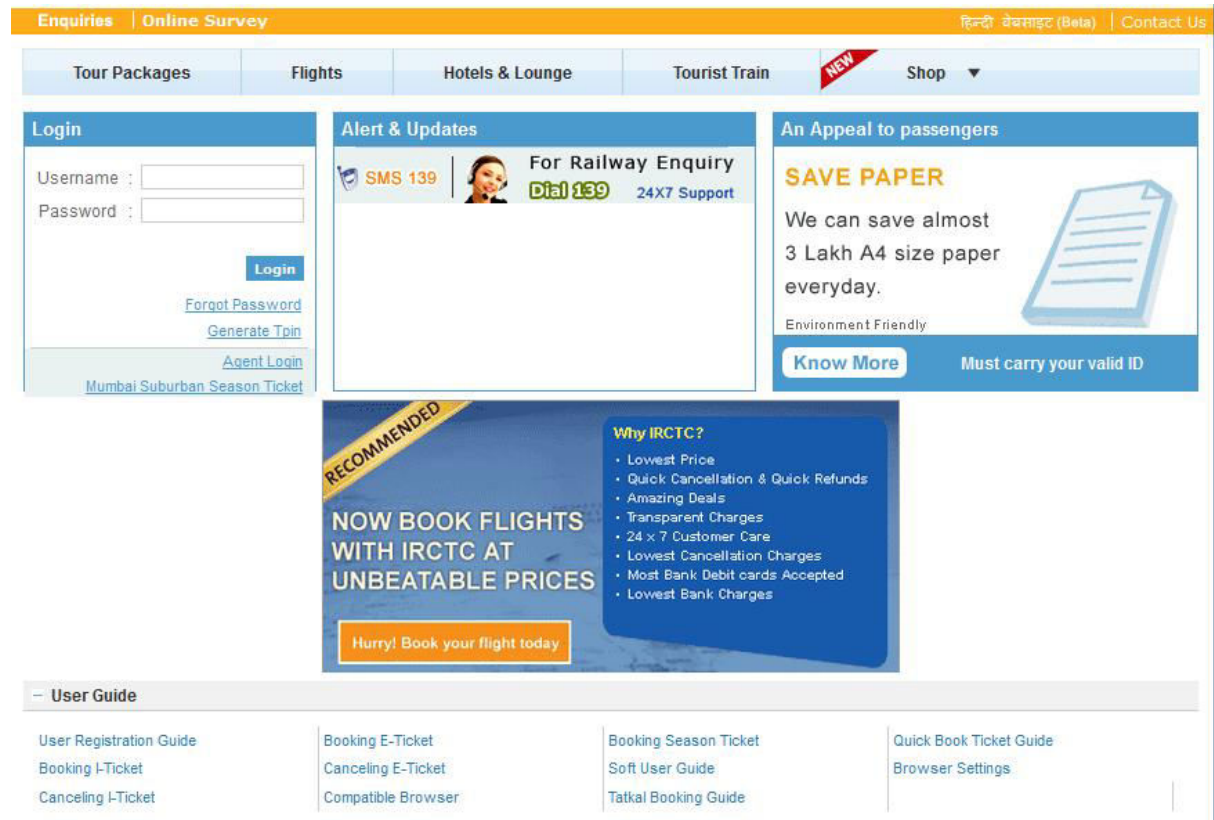
Graphical User Interface



5.1 Screen Short

The system shall provide a uniform look and feel between all the web pages.

Home Page:



The screenshot displays the IRCTC Home Page with a clean, uniform design. At the top, a yellow navigation bar contains links for 'Enquiries', 'Online Survey', and 'हिन्दी वेबसाइट (Beta) | Contact Us'. Below this is a blue header with tabs for 'Tour Packages', 'Flights', 'Hotels & Lounge', 'Tourist Train', and 'Shop'. The main content area is divided into three columns. The left column features a 'Login' section with fields for 'Username' and 'Password', a 'Login' button, and links for 'Forgot Password', 'Generate Tpin', 'Agent Login', and 'Mumbai Suburban Season Ticket'. The middle column has an 'Alert & Updates' section with a 'SMS 139' icon, a 'For Railway Enquiry' section with a 'Dial 139' icon and '24x7 Support' text, and a large blue banner for 'NOW BOOK FLIGHTS WITH IRCTC AT UNBEATABLE PRICES' with a 'Hurry! Book your flight today' button. The right column contains an 'An Appeal to passengers' section with the text 'SAVE PAPER' and 'We can save almost 3 Lakh A4 size paper everyday.', a 'Know More' button, and a 'Must carry your valid ID' button. At the bottom, a 'User Guide' section lists various guides: 'User Registration Guide', 'Booking E-Ticket', 'Booking Season Ticket', 'Quick Book Ticket Guide', 'Booking I-Ticket', 'Canceling E-Ticket', 'Soft User Guide', 'Browser Settings', 'Canceling I-Ticket', 'Compatible Browser', 'Tatkal Booking Guide', and 'Browser Settings'.

Login:

Username

This is the email address you registered with

Password

[\(I forgot my password\)](#)

☒ Remember me on this computer

Sign in



Registration:

First Name

Last Name

Mobile Number

Email

Password

Confirm Password

☒ I would like to be kept informed of special promotions and offers by Yatra.

Register

Search Train:

Search trains

Indian Railways IRCTC Train Tickets Reservation



From

To

Class

Select class

Date

dd/mm/yyyy



Adults

1

(12-60 yrs)

Children

0

(5-11 yrs)

Senior men

0

(60+ yrs)

Senior women

0

(58+ yrs)

Search trains

Other Screen:



Example:

Indian Railways Official Website:



INDIAN RAILWAYS PASSENGER RESERVATION ENQUIRY

[Hindi Version](#)

CRIS

[Accident Train Chart](#)
[for PNR Status update launched by Indian Railways](#)

[PNR Status](#)

[Reserved Train Between Imp Stations](#)

[Seat Availability](#)

[Fare Enquiry](#)

[Internet Reservation](#)

Services

- Availability at Major Stations
- Reserved Train Schedule
- National Train Enquiry System
- SMS Service
- Current Booking Availability
- Train Berth Availability

Information

- Train Type Information
- View Codes
- Trains at a Glance
- Rules
- International Tourists
- Tatkal Scheme
- Other Railway Websites



Experience Our Service...

[india.gov.in](#)
The national portal of India

India has some of the most spectacular and unforgettable rail journeys in the world. Here you experience a simple way to find out everything you need to know in one easy place. There's no better way to enjoy India's outback, cities, coastal towns and regional areas in comfort.

[Home](#) | [Ministry of Railways](#) | [Trains between Stations](#) | [Booking Locations](#) | [CRIS](#) | [CONCERT](#) | [FAQ](#) | [Stemap](#) | [Feedback](#)

Copyright © 2010, Centre For Railway Information Systems, Designed and Hosted by CRIS | Disclaimer
Best viewed at 1024 x 768 resolution with Internet Explorer 5.0 or Mozilla Firefox 3.5 and higher



References

1. IEEE SRS Format
2. Yatra.com
3. Irctc.co.in
4. Indianrail.gov.in
5. www.google.com

