LAB MANUAL OF

SOFTWARE ENGINEERING LAB

CIC-357

Maharaja Agrasen Institute of Technology, PSP area,
Sector – 22, Rohini, New Delhi – 110085
(Affiliated to Guru Gobind Singh Indraprastha University,New Delhi)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

"To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values."

MISSION

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal needs.

TABLE OF CONTENTS

INTRODUCTION TO THE LAB

Course Objective:

1. To Introduce the basic concepts of the software development process, Software requirements and specification.

2. To impart knowledge of Software Project Planning and various Software design techniques for developing large software systems.

3. To understand Software Metrics, Software Reliability, and Quality assurance using ISO 9001 and SEI-CMM.

4. To impart the knowledge and use of software engineering processes and tools in analysis, design, implementation, software testing, documentation, and maintenance for software systems.

Course Outcomes

At the end of the course, a student will be able to:

CIC-357.1 Ability to have an understanding of SDLC Models, Techniques for Requirement Elicitation, and SRS Document.

CIC-357.2 To be able to explain Software Project Planning and various methods for software design.

CIC-357.3 To understand Software Metrics, Software Reliability, and Quality assurance.

CIC-357.4 Ability to have an understanding of software testing, documentation and maintenance.

LAB DETAILS

H/W Detail

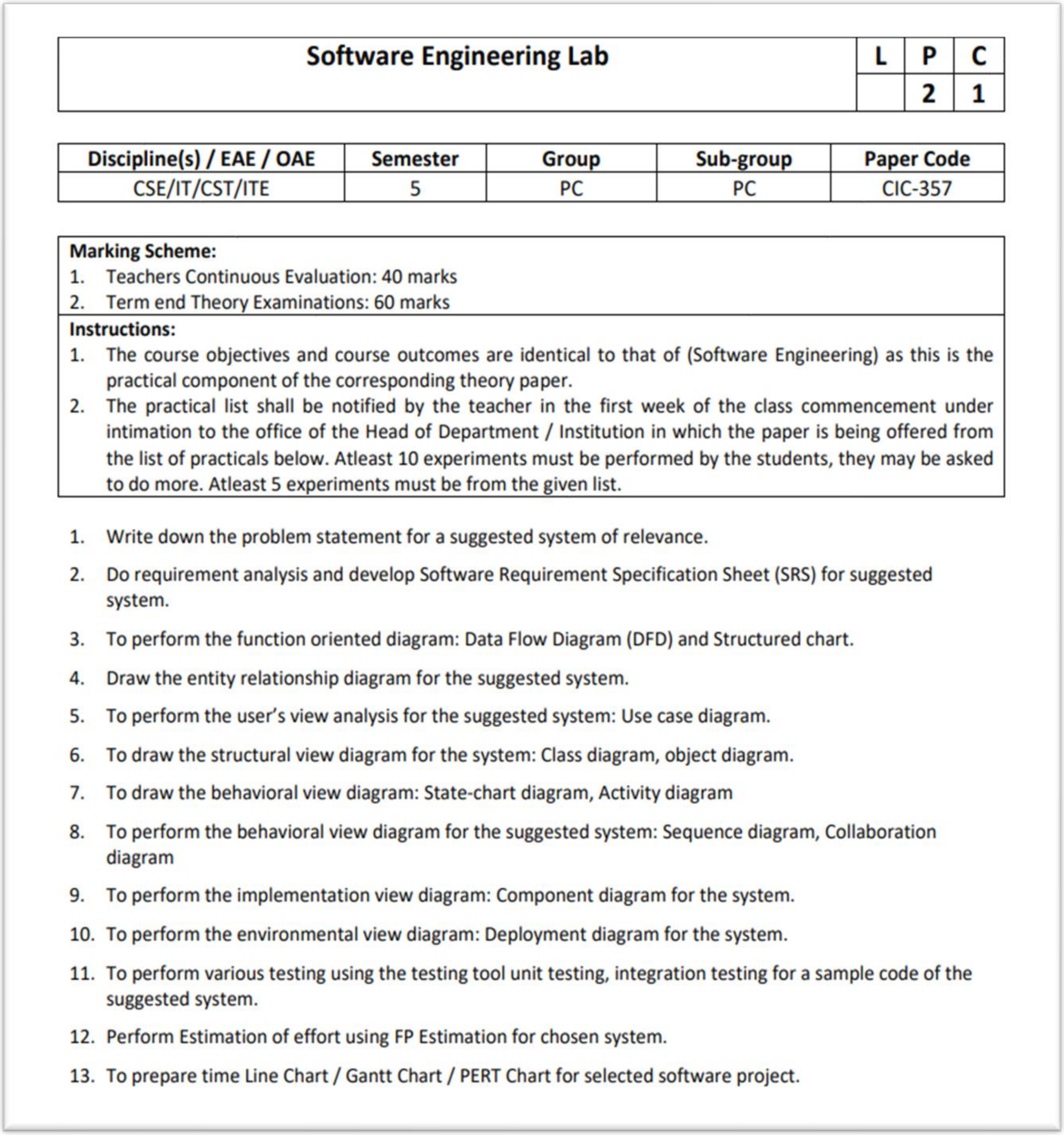
Intel i3/C2D Processor/2 GB RAM/500GB HDD/MB/Lan Card/
Key Board/ Mouse/CD Drive/15” Color Monitor/ UPS 24 Nos

LaserJet Printer 1 No

S/W Detail

Fedora Linux, starUML

LIST OF EXPERIMENTS
(As prescribed by G.G.S.I.P.U)



Software Engineering Lab	L	P	C
		2	1

Discipline(s) / EAE / OAE	Semester	Group	Sub-group	Paper Code
CSE/IT/CST/ITE	5	PC	PC	CIC-357

Marking Scheme:

1. Teachers Continuous Evaluation: 40 marks
2. Term end Theory Examinations: 60 marks

Instructions:

1. The course objectives and course outcomes are identical to that of (Software Engineering) as this is the practical component of the corresponding theory paper.
2. The practical list shall be notified by the teacher in the first week of the class commencement under intimation to the office of the Head of Department / Institution in which the paper is being offered from the list of practicals below. Atleast 10 experiments must be performed by the students, they may be asked to do more. Atleast 5 experiments must be from the given list.

1. Write down the problem statement for a suggested system of relevance.
2. Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.
3. To perform the function oriented diagram: Data Flow Diagram (DFD) and Structured chart.
4. Draw the entity relationship diagram for the suggested system.
5. To perform the user's view analysis for the suggested system: Use case diagram.
6. To draw the structural view diagram for the system: Class diagram, object diagram.
7. To draw the behavioral view diagram: State-chart diagram, Activity diagram
8. To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram
9. To perform the implementation view diagram: Component diagram for the system.
10. To perform the environmental view diagram: Deployment diagram for the system.
11. To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.
12. Perform Estimation of effort using FP Estimation for chosen system.
13. To prepare time Line Chart / Gantt Chart / PERT Chart for selected software project.

(Handbook of B.Tech Programmes offered by USCIT)

Choose any one project and do the above exercises for that project

a. Student Result Management System

b. Library management system

c. Inventory control system

d. Accounting system

e. Fast food billing system
f. Bank loan system

g. Blood bank system

h. Railway reservation system

i. Automatic teller machine

j. Video library management system

k. Hotel management system

l. Hostel management system

m. E-ticking

n. Share online trading

o. Hostel management system

p. Resource management system

q. Court case management system

REVISED LAB EXPERIMENT LIST

1. Write down the problem statement for a suggested system of relevance.
2. Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.
3. To perform the function-oriented diagram: Data Flow Diagram (DFD) and Structuredchart.
4. Draw the entity relationship diagram for the suggested system.
5. To perform the user's view analysis for the suggested system: Use case diagram.
6. To draw the structural view diagram for the system: Class diagram, object diagram.
7. To draw the behavioral view diagram: State-chart diagram, Activity diagram
8. To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram
9. To perform the implementation view diagram: Component diagram for the system.
10. To perform the environmental view diagram: Deployment diagram for the system.
11. To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.
12. To Perform Estimation of effort using FP Estimation for chosen system.
13. To Prepare time line chart/Gantt Chart/PERT Chart for selected software project.

NOTE: - At least 8 Experiments out of the list must be done in the semester.

LIST OF EXPERIMENTS

(Beyond the syllabus)

1. To give the comparison between the starUML and the Rational rose.
2. Why are entity relationship diagrams are used. Show the connectivity of entity relationship diagrams with database. Explain with the help of an example.
   (This will be useful when you will study the DBMS Subject in upcoming semester)

FORMAT OF LAB RECORD TO BE PREPARED BY STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Paper Code

Font should be (Size 20”, italics bold, Times New Roman)

Faculty name

Student name

Roll No.:

Semester:

Font should be (12”, Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,Sector
– 22, Rohini, New Delhi – 110085

Font should be (18”, Times Roman)

Software Engineering Lab (CIC-357)
Lab Assessment Sheet/Index

Student Enrollment No: Student Name:

Experiment No	Marks					Total marks	Remarks if any	Signature
	R1	R2	R3	R4	R5			
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								

Overall Comments:

Faculty Name

Signature

MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

i. Internal Practical Exam

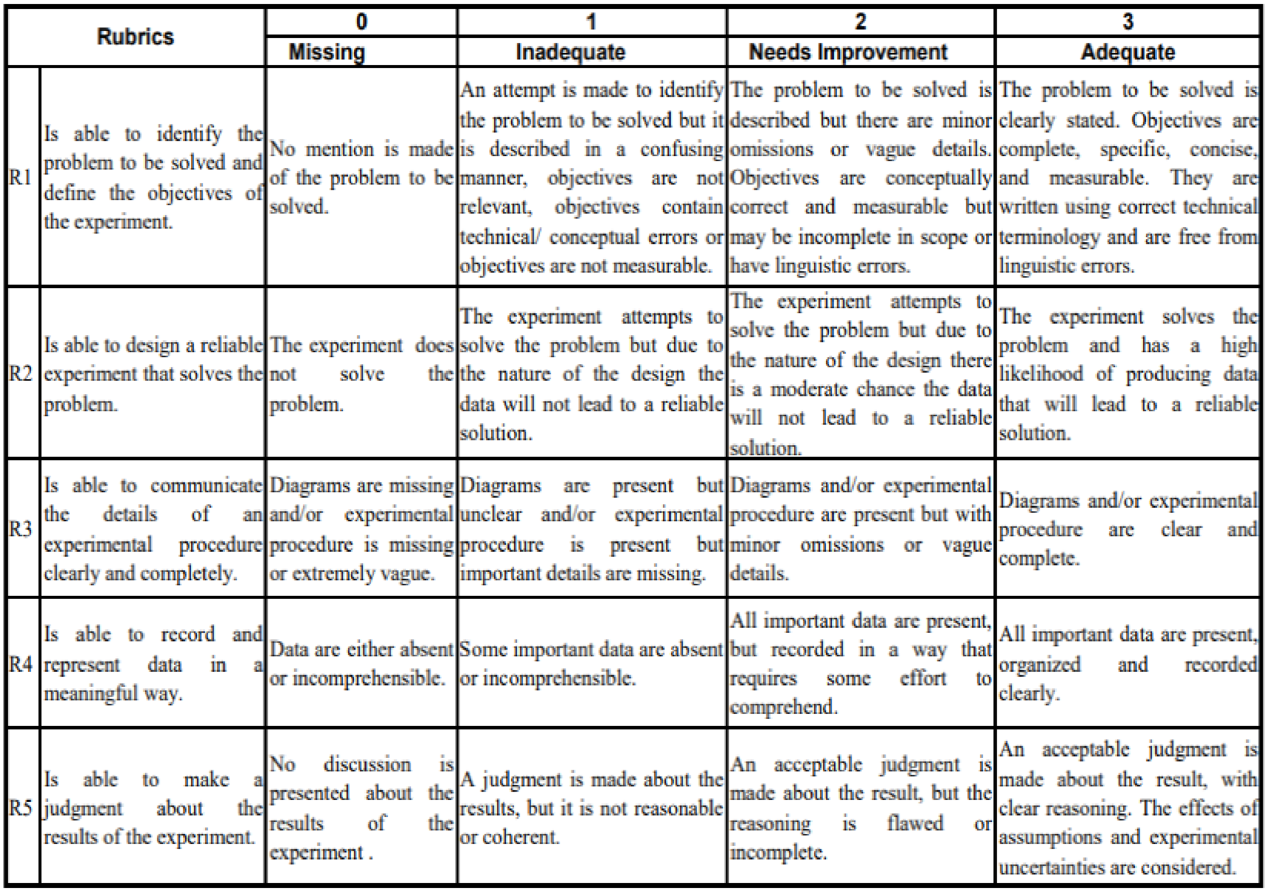
ii. External Practical Exam

INTERNAL PRACTICAL EXAM

Total Marks: 40

Marking of internal practical exam depends upon the below rubrics: -

Rubrics for Lab Assessment



Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment .	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME:

Total Marks: 60

Note: -

- Experiments given to perform can be from any section of the lab.

INTRODUCTION TO STAR UML

StarUML is an open-source software modeling tool that supports UML (Unified Modeling Language). It is based on UML version 1.4, provides eleven different types of diagrams and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approachby supporting the UML profile concept and allowing to generate code for multiple languages.

StarUML is an open-source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. The goal of the StarUML project is to build a software modeling tool and also platform that is a compelling replacement of commercial UML tools such as Rational Rose, Together and so on.

- **UML 2.0**: UML is continuously expanding standard managed by OMG (Object Management Group). Recently, UML 2.0 is released and StarUML support UML 2.0 and will support latest UML standard.
- **MDA (Model Driven Architecture)**: MDA is a new technology introduced by OMG. To get advantages of MDA, software modeling tool should support many customization variables. StarUML is designed to support MDA and provides many customization variables like as UML profile, Approach, Model Framework, NX(notation extension), MDA code and document template and so on. They will help you fitting tool into your organizational cultures, processes, and projects.
- **Plug-in Architecture** : Many users require more and more functionalities to software modeling tools. To meet the requirements, the tool must have well-defined plug-in platform. StarUML provides simple and powerful plug-in architecture so anyone can develop plug-in modules in COM-compatible languages (C++, Delphi, C#, VB, ...)
- **Usability** : Usability is most important issue in software development. StarUML is implemented to provide many user-friend features such as Quick dialog, Keyboard manipulation, Diagram overview, etc.

StarUML is mostly written in Delphi. However, StarUML is *multi-lingual project* and not tied to specific programming language, so any programming languages can be used to develop StarUML. (for example, C/C++, Java, Visual Basic, Delphi, JScript, VBScript, C#, VB.NET, ...)

History of StarUML

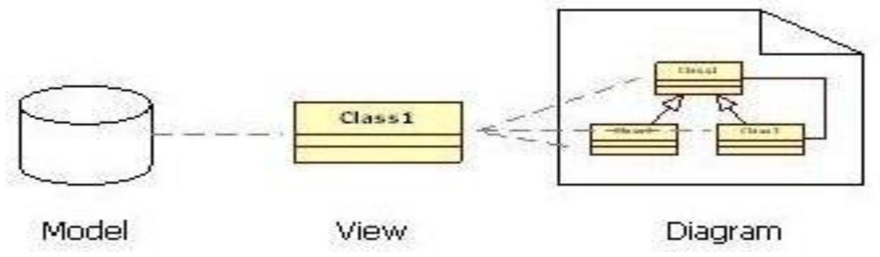
StarUML is formerly known as "Plastic" or "Agora Plastic".

1996	The first version (v0.9) of Plastic was born. It was very simple tool that is used to draw software modules and their dependencies.
1997	Plastic 1.0 released. Freeware, OMT supported, Grand prize of software contest held by Hyundai.
1998	Plastic 1.1 released. UML class diagram supported
1999	Foundation of Plastic Software, Inc. Plastic 2.0 released. UML supported, Java code generation and reverse engineering
2001	Plastic 3.0 released. UML 1.3 fully supported
2003	Plastic 2003 released. Completely redesigned and rewritten, UML 1.4 fully supported, open architecture.
2005	Agora Plastic 2005 released. Internationalized, many features are implemented on extensible platform. 'Good Software' Certified from The Ministry of Information and Communications of Korea.
2005	StarUML 5.0 renamed and released. turned to open-source project, UML 2.0 supported, and notation extension technology was implemented.

Features

When you start a new project, StarUML proposes which approach you want to use: 4+1 (Krutchen), Rational, UML components (from Chessman and Daniels book), default or empty. Depending on the approach, profiles and/or frameworks may be included and loaded. If you don't follow a specific approach, the "empty" choice could be used. Although a project can be managedas one file, it may be convenient to divide it into many units and manage them separately if manydevelopers are working on it together.

StarUML makes a clear conceptual distinction between models, views and diagrams. A Model is an element that contains information for a software model. A View is a visual expression of the information contained in a model, and a Diagram is a collection of view elements that represent the user's specific design thoughts.



StarUML is build as a modular and open tool. It provides frameworks for extending the functionality of the tool. It is designed to allow access to all functions of the model/meta-model and tool through COM Automation, and it provides extension of menu and option items. Also, users can create their own approaches and frameworks according to their methodologies. The toolcan also be integrated with any external tools.

The user interface is intuitive. On the upper right side, a window allows to rapidly navigate between all the content of a project, adopting either a model or a diagram view. Multiple diagramscan be open at the same time and tabs allow switching rapidly between views. The lower right window allows to document the current diagram, either with plain text or attaching an external document. During diagram editing, "wizards" are located around the object that give you the quickshortcuts to main associated tasks with your current operation, like adding an attribute when you create a class for instance. A right-click on the mouse brings the full set of operations at your disposal.

StarUML has also a model verification feature. You can export diagram in different formats (jpg,bmp, wmf). It also supports a patterns approach and import of Rational Rose files.

StarUML Generator is platform module to generate various artifacts (like as Microsoft Word, Excel, PowerPoint, and Text-based artifacts) by templates depending on UML model elements inStarUML. The users can define their own templates and can apply many different kinds of templates to the same UML model, so the users can get various artifacts automatically, easily andfast. The tool supports code generation and reverse engineering for Java, C# and C++.
So, we can summarize the key features of StarUML as follows: -

The key features of StarUML are:

- Multi-platform support (MacOS, Windows and Linux)
- UML 2.x standard compliant
- SysML support
- Entity-Relationship diagram (ERD)
- Data-flow diagram (DFD)
- Flowchart diagram
- Multiple windows
- Modern UX
- Dark and light themes
- Retina (High-DPI) display support
- MacPro Pro's Touch Bar support
- Model-driven development
- Open APIs
- Various third-party extensions
- Asynchronous model validation
- Export to HTML docs
- Automatic updates.

Experiment 1

Aim: Write down the problem statement for a suggested system of relevance.

Description: The problem statement is the initial starting point for a project. It is basically a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem.

The input to requirement engineering is the problem statement prepared by customer.
It may give an overview of the existing system along with broad expectations from the new system.
The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So, from here begins the preparation of problem statement.

So, basically a problem statement describes **what** needs to be done without describing **how**.

Performance Instruction:

1. Choose any one project from given list.
2. Collect all requirements
3. Identify functionalities
4. Write a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level.

Sample Output:

PROBLEM STATEMENT FOR RAILWAY RESERVATION SYSTEM

Software has to be developed for automating the manual railway reservation system. The system should be distributed in nature. It should be designed to provide the functionalities as follows:

Reserve Seat: A traveler should be able to reserve seats in the desired train. A reservation form is to be filled by the traveler and given to the clerk, who then checks for the availabilityof seats for the specified train and date of journey. If seats are available, then entries are madeinto the system regarding the train name, train number, date of journey, boarding station, destination, person name, age, sex and the total fare. Traveler is asked to pay the required fareand the tickets are printed. It should be noted that a single ticket should not reserve more thansix persons at a time and the children below 12 years and the senior citizens should get 50% concession in their respective fare. If the seats are not available, then reservation request is rejected and the Traveler is informed so.

1. **Cancel Reservation:** A traveler wishing to cancel a reservation is required to fill a form. The traveler then submits the form and the ticket to the clerk. The clerk then deletes the corresponding entries in the system and changes the reservation status of that train. The ticket is crossed by hand and considered cancelled. A new cancellation ticket is generated and given to the traveler along with the fare minus the 20% cancellation fees per reservation.
2. **Update Train Information:** Only the administrator can enter any changes related to the train information like change in train name, train number, train route, etc. in the system.
3. **Report Generation:** Provision for generation of different reports should be there in the system. The system should be able to generate reservation chart, monthly train report etc.
4. **Login:** For security reasons all the users of the systems are given a user ID and password. Only when both the entries are correct and they match the user should be allowed to enter the system.
5. **View Reservation status:** All the users should be able to see arrival & departure time and reservation status of a train online. The user needs to enter the train number the PNR number printed on the ticket so that the system can display the current train position like on time, late by specified hours or the reservation status like confirmed, wait listed and RAC.
6. **View Train Schedule:** Provision should be made in the system to see information related to the train schedules for entire train network. The user should be able to see the train name, train number, boarding and destination stations, duration of journey etc.

Conclusion: The problem statement was written successfully by following the steps described above.

Viva - Questions:

Q-1. What is problem statement?

Q-2. What are the benefits of writing problem statement?

Q-3. Writing a problem statement, is really a beneficial for you in proceeding project?

Q-4. Explain 5W's can be used to spark the problem?

Q-5. What are steps that need to follow while writing problem statement?

Experiment 2

Aim: Do requirement analysis and develop Software Requirement Specification Sheet (SRS) for suggested system.

Description: Software Requirement Specification (SRS) is a document that describes the requirements of a computer system from the user's point of view.
An SRS document specifies: The required behavior of a system in terms of: input data, required processing, output data, operational scenarios and interfaces. The attributes of a system including: performance, security, maintainability, reliability, availability, safety requirements and design constraints.

A well-designed, well-written SRS accomplishes four major goals:

- It provides feedback to the customer. An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

Performance Instruction: SRS DOCUMENT TEMPLATE

1. Introduction

1.1 Purpose
1.2 Scope
1.3 Definitions, Acronyms, and Abbreviations

1.4 References
1.5 Overview

2. Overall Description

2.1 Product Perspective

2.1.1 System Interfaces
2.1.2 User Interfaces
2.1.3 Hardware Interfaces
2.1.4 Software Interfaces
2.1.5 Communication Interfaces
2.1.6 Memory Constraints
2.1.7 Operations
2.1.8 Site Adaptation Requirements
2.2 Product Functions
2.3 User Characteristics
2.4 Constraints
2.5 Assumptions and dependencies
2.6 Apportioning of requirements

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Login Screen

Train Info Parameters Screen

Train Information Screen

Passenger Info Parameters Screen

Passenger Information Screen

Passenger’s Train Choice Parameters Screen

Passenger Entry Info Screen

Non-Availability Info Screen

Passenger Entry Screen

Passenger Parameters Screen

Passenger List Report Parameters

RAC List Parameters Screen

3.1.2. Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communications Interfaces

3.2 Software Product Features

3.2.1 Train Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.2.2 Passenger Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.2.3 Ticket Generation

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.2.4 Repot Generation

Passenger List and RAC Report

WL Report

Monthly Passenger List Report

3.2.5 User Accounts Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.1.2. Hardware Interfaces

3.1.5 Software Interfaces

3.1.6 Communications Interfaces

3.3 Software Product Features

3.3.1 Train Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.3.2 Passenger Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.3.3 Ticket Generation

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.3.4 Repot Generation

Passenger List and RAC Report

WL Report

Monthly Passenger List Report

3.3.5 User Accounts Information Maintenance

Description

Validity Checks

Sequencing Information

Error Handling/ Response to Abnormal Situations

3.4 Performance Requirements

3.5 Design Constraints

3.6 Software System Attributes

3.6.1 Security

3.6.2 Maintainability

3.6.3 Portability

3.7 Logical Database Requirements

3.8 Other Requirements

Sample Output:

1. INTRODUCTION

This document aims at defining the overall software requirements for '**RAILWAY RESERVATION SYSTEM**'. Efforts have been made to define the requirements exhaustively and accurately. The final product will be having only features/functionalities mentioned in this document and assumptions for any additional functionality/feature should not be made by any of the parties involved in developing/testing/implementing/ using this product. In case it is required to have some additional features a formal change request will need to be raised and subsequently a new release of this document and/or product will be produced.

1.1 PURPOSE

This specification document describes the capabilities that will be provided by the software application 'RAILWAY RESERVATION SYSTEM'. It also sates the various required constraints by which the system will abide. The intended audiences for this document are the development team, testing team and end users of the product.

1.2 SCOPE

The software product 'RAILWAY RESERVATION SYSTEM' is an application that will be used for ticketing, reservation, cancellation and management of railway system for our government. The application will manage the information about various passengers which travel through the railways, through which train they want to travel, where the want to travel, distance between the two cities, what is the current status and the status of next three days of the reservation, what is the fare, what is the class through they want to travel and their personal records. Printable tickets for the passengers will also be generated.

This application will greatly simplify and speed up the result preparation and management process

.

1.*3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS*

The following abbreviation has been used throughout this document

PNR: Passenger Name Record

RAC: Reservation Against Cancellation.

WL: Waiting List.

1.3 REFERENCES

(i) Website: For more information, log on to www.indianrail.gov.in

1.4 OVERVIEW

The rest of this SRS document describes the various system requirements, interfaces, features and functionalities in detail.

1.OVERALL DESCRIPTION

1.1 PRODUCT PERPECTIVE

The application will be a windows-based, self-contained and independent software product.

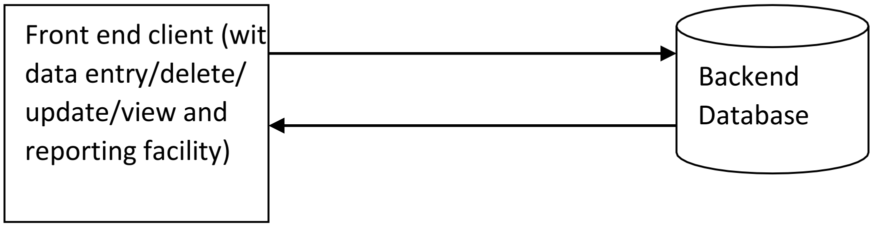


Fig 2.1

1.1.1 <u>SYSTEM INTERFACES</u>

None

1.1.2 <u>USER INTERFACES</u>

The application will have a user-friendly and menu based interface. Following screens will be provided:

(i) A login screen for entering the username, password and role (Reservation Clerk, Administrator, Coordinator) will be provided. Access to different screen will be based upon the role of the user.

(ii) There will be a screen for capturing and displaying information regarding what trains are scheduled during which date, how much is the fare, and what are the classes in the trains.

(iii) There will be a screen for displaying information regarding various passengers.

(iv) There will be a screen for capturing and displaying information regarding which passenger is currently reserved in which train and what is the class of that seat.

(v) There will be a screen that will capture information regarding the discount in the fare of the ticket. Discount will be given on various basis like senior citizen, student concession, scheduled cast etc.

(vi) There will be a screen for capturing and displaying information regarding which all user account exists in the system, thus showing who all can access the system.

The following reports will be generated:

(i) Reservation Chart: Printable report will be generated to show the list of the passengers reserved in a particular train along with the class of their travel.

(ii) RAC Chart: Printable report will be generated to show to show the list of passengers who are getting their seats in RAC in a particular train.

(iii) Waiting List: Printable reports will be generated to show the list of the passengers who are in the waiting list of the reservation for their seats in a particular train.

(iv) Monthly Report: Monthly report will be generated for the railway department to show how many passengers have traveled during the particular month.

1.1.3 <u>HARDWARE INTERFACES</u>

(i) Screen resolution of at least 800x600-required for proper and complete viewing of screen. Higher resolution would not be a problem.

(ii) Support for printer (dot –matrix /Desk Jet /inkjet etc. –any will do)-that is, appropriatedrivers are installed and printer connected printers will be required for printing of reports.

(iii) Standalone system or network based-not a concern, as it will be possible to run the application on any of these.

1.1.4 <u>SOFTWARE INTERFACES</u>

(i) Any window -based operating system (window 95/98/2000/XP/NT).

(ii) MS access 2000 as the DBMS—for database. Future release of the application will aim at upgrading to oracle 8i as the DBMS.

(iii) Crystal reports 8—for generating and viewing pay slips and discharge slips.

(iv) Visual Basic 6—for coding /developing the software.

Software mentioned in pts. (iii) and (iv) above will be required only for development of the application. The final application will be packaged as an independent setup program that will be delivered to the client (hospital in this case).

1.1.5 <u>COMMUNICATION INTERFACES</u>

None

1.1.6 <u>MEMORY CONSTRAINTS</u>

At least 64 MB RAM and 2 GB space on hard disk will be required for running the program.

1.1.7 <u>OPERATIONS</u>

This product release will not cover any automated housekeeping aspects of the database. The DBA at the client site (i.e., Railways) will be responsible for manually deleting old/non-required data. Database backup and recovery will also have to be handled by the DBA. However, the system will

provide a ‘RESET SYSTEM’ function that will delete (upon conformation from the administrator) all the existing information from the database.

1.1.8 <u>SITE ADAPTATION REQUIREMENTS</u>

The terminals at client site will have to support the hardware and software interface specified in above section.

1.2 PRODUCT FUNCTIONS

The system will allow access only to authorized users with specific roles. Depending upon the user’s role he/she will be able to access only specific modules of the system.

A summary of the major functions that the software will perform:

(i) A LOGIN facility for enabling only authorized access to the system.

(ii) User (with role Reservation Clerk) will be able to add/modify/delete information about different passengers that are reserving their ticket in different trains and dates.

(iii) User (with role Reservation Clerk) will be able to add/modify/delete information about different seats that are offered in a train (1AC, 2AC, 3AC, Sleeper). The Reservation list of passengers along with their class should be displayed.

(iv) User (with role Reservation Clerk) will be able to add/modify/delete information about the waiting list of the passengers and their RAC.

(v) User (with role Reservation Clerk) will be able to print the ticket of the passenger.

(vi) User (with role Administrator) will be able to generate printable reports.

(vii) User (with role Administrator) will be able to ‘Reset’ the system – leading to deletion of all existing information from the backend database.

(viii) User (with a role Administrator) will be able to create/ modify/ delete new/ existing user accounts.

1.3 USER CHARACTERISTICS

(i) Educational Level- At least a graduate should be comfortable with English.

(ii) Experience- Should be well informed about the features concerning railways. Technical

expertise-Should be comfortable with general purpose applications of computer.

(iii) Technical expertise-Should be comfortable with general purpose applications of computer.

1.4 CONSTRAINTS

(i) Since the DBMS being used in MS access 2000, which is not a very powerful DBMS it will not be able to store a very huge number of records.

(ii) Due to limited features of DBMS being used performance tuning features will not be applied to the queries and thus the system may become slow with the increase in the records being stored.

(iii) Due to limited features of DBMS being used, database auditing will also not be provided.

(iv) Users at Railway Reservation will have to implement a security policy to safeguard the passenger related information from being modified by unauthorized users (by means of gaining access to the backend database).

1.5 ASSUMPTIONS AND DEPENDENCIES

The numbers of seats in a train are fixed. There should be no additions in the number of births.

1.6 APPORTIONING OF REQUIREMENTS

None.

2.SPECIFIC REQUIREMENTS

This section contains the software requirements to a level of detail sufficient to enable designers to design the system, and testers to test that system.

2.1 EXTERNAL INTERFACE REQUIREMENTS

2.1.1 USER INTERFACES

The following screens will be provided:

Login Screen:

This will be the first screen that will be displayed. It will allow user to access different screens based upon the user’s role. Various fields available on this screen will be:

(i) User ID: Alphanumeric of length upto 10 characters.

(ii) Password: Alphanumeric of length upto 8 characters.

(iii) Role: Will have the following values:
Administrator, Coordinator, Reservation Clerk.

Train Info Parameters Screen:

This screen will be accessible only to user with role Administrator. It will allow the user to enter the name of train for which the user wants to access the train information.

Train Information Screen:

This screen will be accessible only to user with role Administrator. It will allow user to add/modify/delete information about new/existing train(s) for a particular date that was selected in the 'Train Info Parameters' screen. The list of available seats for that train will also be displayed. Various fields available on this screen will be:

(i) Train number: of format T#### (# represent a digit).
(ii) Train Name: Alphanumeric of length upto 50 characters.
(iii) Seats: Number of total seats in each class section of the train.

Passenger Info Parameters Screen:

This screen will be accessible only to user with role Administrator. It will allow the user to enter the train number for which the user wants to access the passenger information. **Passenger Information Screen:**

This screen will be available only to role Administrator. It will allow the user to add/modify/delete information about new/existing student(s) for a particular train number. Train wise list of passengers will also be displayed. Various fields available on these screens will be:

(i) PNR number: of the format PNR########## (# represent Alphanumeric digits).
(ii) Passenger Name: will have only alphabetic letters and length upto 40 characters.
(iii) Sex: will have only one alphabet either 'M' or 'F'.
(iv) Age: will have only three digits.
(v) Train number: of the format T#### (# represent a digit).

Passenger's Train Choice Parameters Screen:

This screen will be accessible only to user with role Administrator. It will allow the user to enter the train number and the class of the travel for which the user wants to access the passenger's train choice information.

Passenger's Train Choice Information Screen:

This screen will be accessible only to user with role Administrator. It will allow the user to add/modify/delete passenger's choices for the trains selected in 'Passenger's Train Choice Parameters' screen. For the selected train it will display the list of seats available in the choices of the passenger. The screen will display the list of passengers who have been allotted the seat. The user will be able to view/add/modify/delete the passenger's choice in the list.

Passenger Entry Info Screen:

This screen will be accessible only to user with role Reservation Clerk. It will allow the user to enter the train number and the class of the train for which the user wants to access the passenger information.

Non-Availability Info Screen:

This screen will be accessible to the user with the role Administrator. It will display the error message to the user about the non-availability of the seats in the current train and class. It allows user to enter another choice for the train number and class. It also allows the user if he wants to continue reserving in the current train and class in the waiting section.

Passenger Entry Screen:

This screen will be accessible only to user with role Reservation Clerk. It will allow user to add/modify/delete information about the seats reserved by different passengers who have been or are going to be allotted seats in the train number and class selected in the 'Passenger Entry Info' screen. The screen will display the list of passengers currently who have been allocated the seats. The user will be able to view/add/modify/delete the passenger information in the list. Various fields available on this screen will be:

(i) PNR number: PNR number of all passengers in the current train.
(ii) Passenger Name: will display the name of passenger.
(iii) Sex: will display the sex of the passenger.
(iv) Age: will display the age of the passenger.
(v) Status: will display the status of the reservation i.e. whether the passenger has been allotted the seat and its seat number or is in RAC or WL. **Passenger Parameters Screen:**

This screen will be accessible only to user with role Reservation Clerk. It will allow the user to enter the PNR number and the Train number of the passenger for whom the user wants to view/print the ticket.

Passenger List Report Parameters:

This screen will be accessible only to user with role Coordinator. It will allow the user to enter the train number for which the user wants to view/print the passenger list report.

RAC List Parameters Screen:

This screen will be accessible only to user with role Coordinator. It will allow the user to enter the train number for which the user wants to view/print the RAC list report.

WL List Parameters Screen:

This screen will be accessible only to user with role Coordinator. It will allow the user to enter the train number for which the user wants to view/print the WL report.

Monthly Passenger List Report Parameters:

This screen will be accessible only to user with role Coordinator. It will allow the user to enter the month for which the user wants to view/print the passenger list report.

3.1.2 **HARDWARE INTERFACES**

As stated in section 2.1.3.

3.1.3 SOFTWARE INTERFACES

As stated in section 2.1.4.

3.1.4 COMMUNICATIONS INTERFACES

None.

2.2 SOFTWARE PRODUCT FEATURES

2.2.1 TRAIN INFORMATION MAINTENANCE

Description:

The system will maintain information about various trains being offered to the passengers. The following information would be maintained for each train:

Train number, train name, train type (superfast, express, passenger, mail etc.), total seats, classes, number of the station the train will pass through.

The system will allow creation/modification/deletion of new/existing trains.

Validity Checks:

(i) Only user with role Administrator will be authorized to access the Train information Maintenance module.
(ii) Each compartment will have a maximum of 72 seats.
(iii) Each train will have atleast two classes.
(iv) Train number will be different for each train.
(v) Train number cannot be blank.
(vi) PNR number cannot be blank.
(vii) Train name cannot be blank.
(viii) Number of seats cannot be zero.

Sequencing Information:

Train info will have to be entered in the system before any info regarding passenger is entered.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error msg. will be prompted to user for doing the needful.

2.2.2 PASSENGER INFORMATION MAINTENANCE

Description:

The system will maintain information about various passengers allotted seats or are waiting to be allotted seats in different trains. The following information would be maintained for each train:

Train number, PNR number, Class, Passenger Info.

The system will allow creation/modification/deletion of new/existing passengers and also have the ability to list all the passengers allotted or are waiting to be allotted seats in a particular train.

Validity Checks:

(i) Only user with role Reservation Clerk will be authorized to access the Passenger Information Maintenance module.
(ii) Every passenger will have a unique PNR number.
(iii) PNR number cannot be blank.
(iv) Passenger name cannot be blank.
(v) Train number cannot be blank.

Sequencing Information:

Train info will have to be entered in the system before any info regarding passenger is entered.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error msg. will be prompted to user for doing the needful.

2.2.3 TICKET GENERATION

Description:

The system will generate ticket for every passenger in different trains.

TICKET WILL HAVE THE FOLLOWING FORMAT:

INDIAN RAILWAYS

NAME OF TRAIN

TRAIN NUMBER

PNR NUMBER:________________NUMBER OF SEATS(s):________

STATUS	SEAT NO.	NAME	AGE	SEX

Fig 2.2

STARTING STATION:_______________DESTINATION: ____________________

DISTANCE IN kms:__________ AMOUNT PAID (in figures): _________

AMOUNT PAID (in words): ____

DATE OF JOURNEY: dd/mm/yyyy STARTING TIME: hh:mm:ss

SIGNATURE OF RAILWAY MINISTER

Fig 2.3

Validity Checks:

(i) Only User with role Coordinator will be authorized to access the Ticket Generation module.

Sequencing Information:

Ticket for a particular passenger can be generated by the system only after PNR number has been entered in the system for a given train number, the passenger info for that ticket has been entered in the system, the choice for the train has been entered in the system, the journey date, and the amount has been paid to the reservation clerk.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error msg. will be prompted to user for doing the needful.

2.2.4 <u>REPORT GENERATION</u>

Passenger List and RAC Report:

For each train a passenger list and a RAC list will be generated containing the list of passengers who have been allotted seats in the train.

INDIAN RAILWAYS

NAME OF TRAIN

TRAIN NUMBER

COACH NO.: ________________

S. No.	PNR NO.	SEAT NO.	NAME	AGE	SEX

Fig 2.5

WL Report:

For each train a WL will be generated containing the list of passengers who are waiting to get the seats allotted in a train.

INDIAN RAILWAYS

NAME OF TRAIN

TRAIN NUMBER

S. No.	PNR NO.	WL STATUS	NAME	AGE	SEX

Fig 2.6

Monthly Passenger List Report:

For each month a passenger list will be generated containing the information about the number of passengers traveling each day and through each train.

INDIAN RAILWAYS

MONTH/YEAR

S. No.	DATE	TRAIN NO.	TRAIN NAME	NO. OF PASSENGERS TRAVELLED

Fig 2.7

2.2.5 USER ACCOUNTS INFORMATION MAINTENANCE

Description: The system will maintain information about various users who will be able to access the system. The following information would be maintained:

User Name, User ID, Password, and Role.

Validity Checks:

(i) Only user with role Administrator will be authorized to access the User Accounts Information Maintenance module.
(ii) User Name cannot be blank.
(iii) User ID cannot be blank.
(iv) User ID should be unique for every user.
(v) Password cannot be blank.
(vi) Role cannot be blank.

Sequencing Information:

User Account for particular user has to be created in order for the system to be accessible to that user. AT system startup, only a default user account for 'Administrator' would be present in the system.

Error Handling/ Response to Abnormal Situations:

If any of the above validations/ sequencing flow does not hold true, appropriate error msg. will be prompted to user for doing the needful.

2.3 PERFORMANCE REQUIREMENTS

None

2.4 DESIGN CONSTRAINTS

None

2.5 SOFTWARE SYSTEM ATTRIBUTES

2.5.1 SECURITY

The application will be password protected. Users will have to enter correct username, password and role in order to access the application.

2.5.2 MAINTAINABILITY

The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules (i.e., new trains, new timings, fare hike).

2.5.3 PORTABILITY

The application will be easily portable on any windows-based system that has MS-Access 2000 installed.

2.6 LOGICAL DATABASE REQUIREMENTS

The following information will be placed in the database:

(i) Passenger Info.
(ii) PNR Number.
(iii) Destination.
(iv) Train Number.

2.7 OTHER REQUIREMENTS

None.

Conclusion: The SRS was written successfully by following the template described above.

Viva - Questions:

Q-1. What are the objectives of requirement analysis?

Q-2. Define different types of requirements?

Q-3. Outline structure of SRS Document?

Q-4. What are benefits of writing SRS document?

Q-5. Define Functional and non-functional requirements

Experiment 3

Aim: To perform the function-oriented diagram: Data Flow Diagram (DFD) and Structured chart.

Description: Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical and logical information systems. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (orexternal entities).

Symbols of DFD:

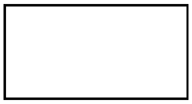
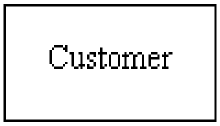
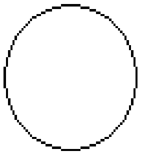
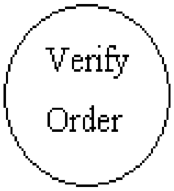
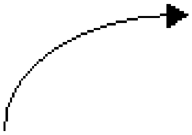
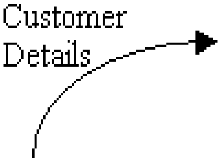
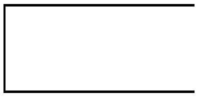
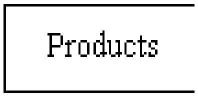
Name	Symbol	Description	Example
Entity		Used to represent people and organizations outside the system. They either input information to the system, accept output information from the system or both	
Process		These are actions that are carried out with the data that flows around the system. A process accepts input data and produces data that it passes on to another part of the DFD	
Data Flow		These represent the flow of data to or from a process	
Data Store		This is a place where data is stored either temporarily or permanently	

Fig 3.1

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.

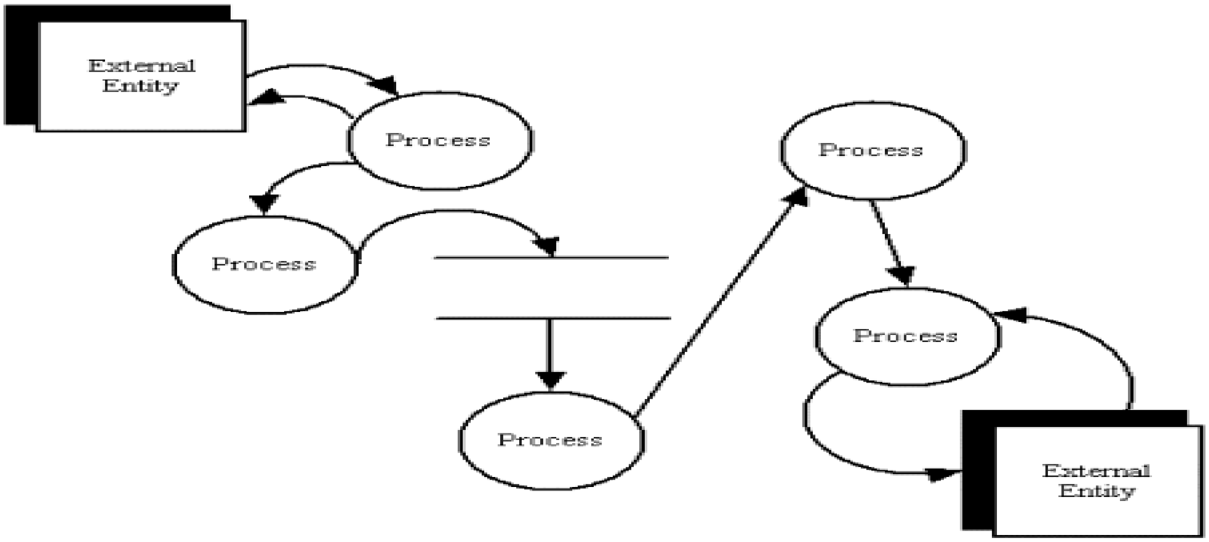


Fig 3.2

Data Flow Diagram Layers

Draw data flow diagrams in several nested layers. A single process node on a high-level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.

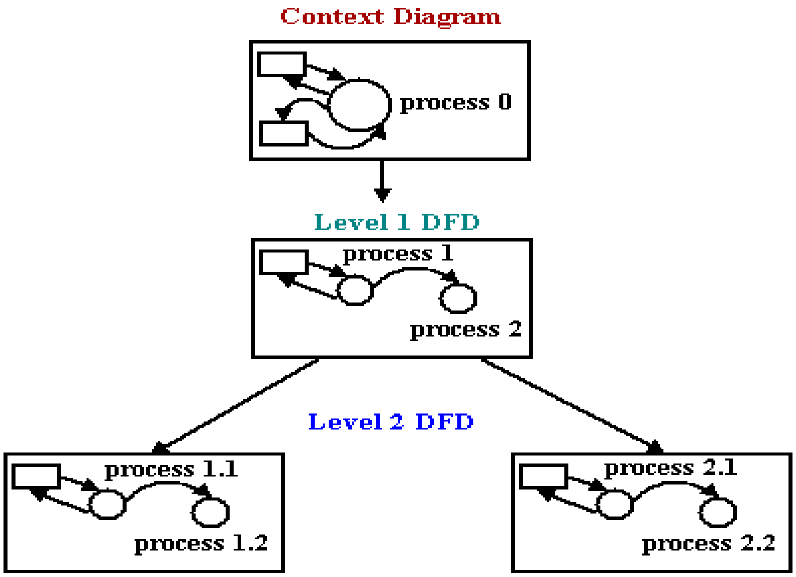


Fig 3.3

A **Structure Chart** (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's

name. The lines represent the connection and or ownership between activities and sub activities as they are used in organization charts. The tree structure visualizes the relationships between modules.

Performance Instruction:

To draw DFD

1) Identify various processes, data store, input, output etc. of the system and analyses it.
2) Use processes at various levels to draw the DFDs.

To draw structured Chart Diagram

3) Identify various modules, input, output etc. of the system.
4) Draw structured chart diagram describing it in form of levels.

Sample Output:

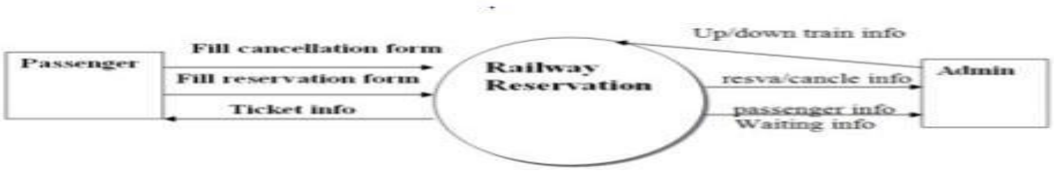


Fig 3.4

Context Diagram / Level 0 DFD

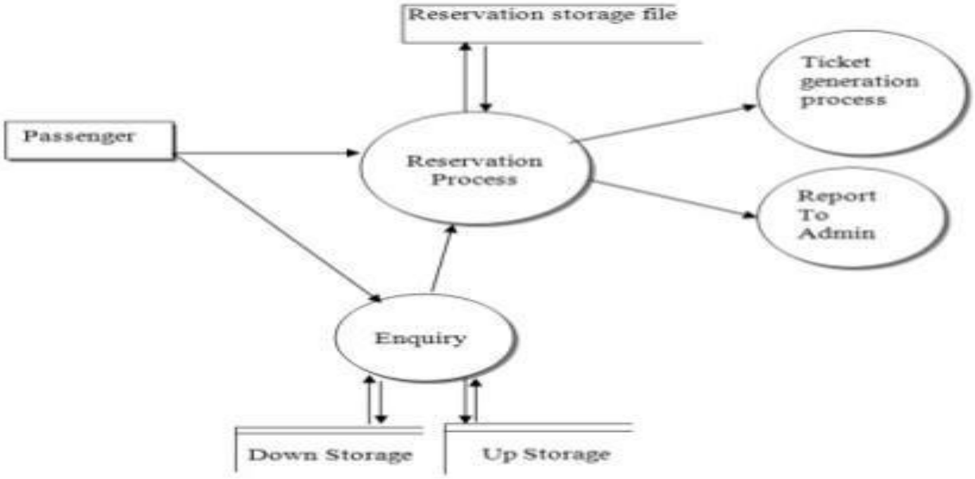


Fig 3.5

Level 1 DFD

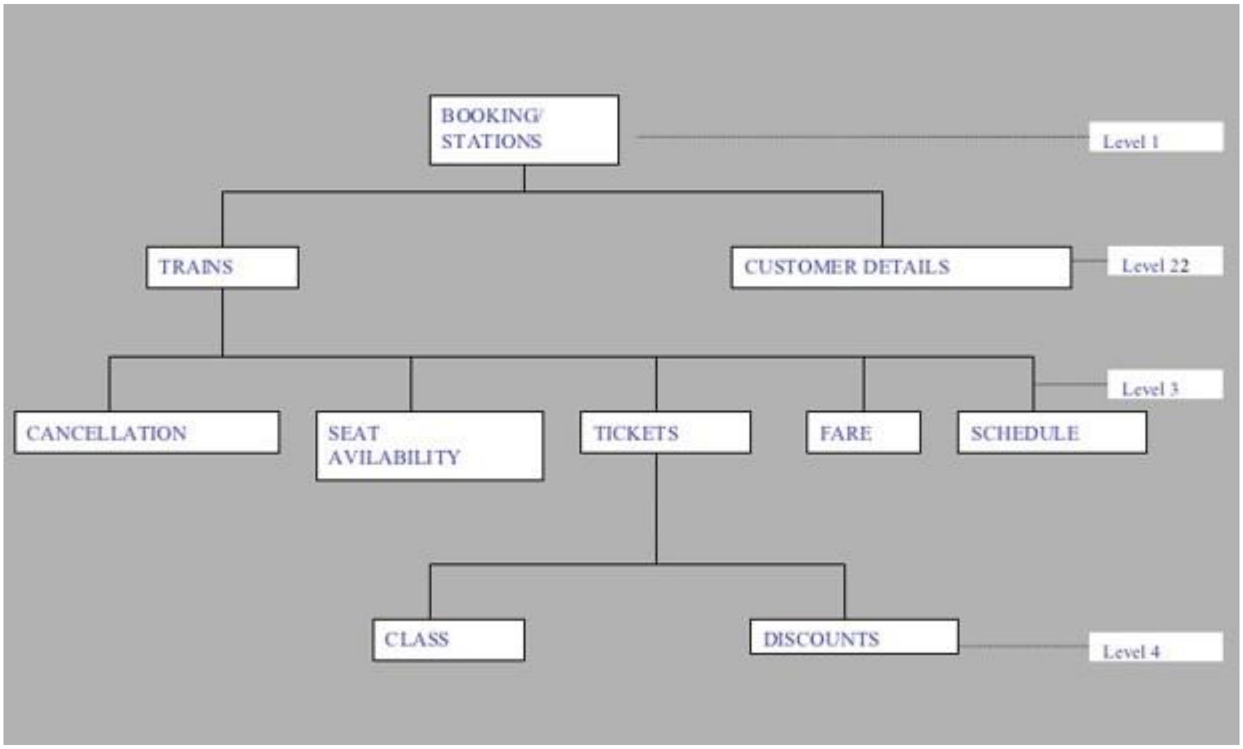


Fig 3.6

Structured Diagram

Conclusion: DFD and Structured Chart diagram was made successfully by following above steps.

Viva - Questions:

Q-1. Define DFD? What are different levels of DFD?

Q-2. Describe symbols used for constructing DFDs?

Q-3. Distinguish between a data flow diagram and a flow chart with example?

Q-4. Explain structured chart diagram?

Q-5. Describe symbols used for constructing structured chart diagram?

Experiment 4

Aim: Draw the entity relationship diagram for the suggested system.

Description: An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.

When to use: ER diagrams are used to model and design relational databases, in terms of logic and business rules (in a logical data model) and in terms of the specific technology to be implemented (in a physical data model.)

Entity:

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

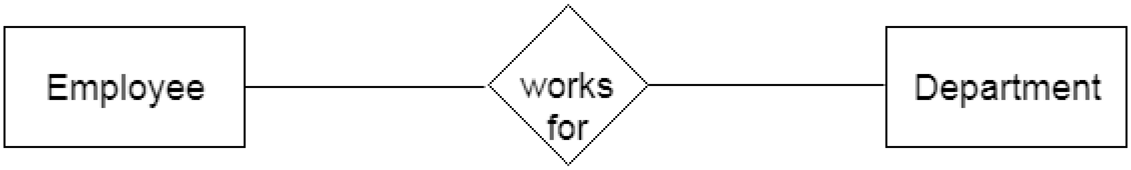


Fig 4.1

Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

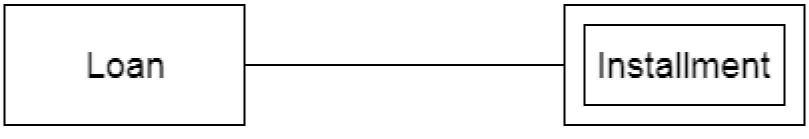


Fig 4.2

Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.

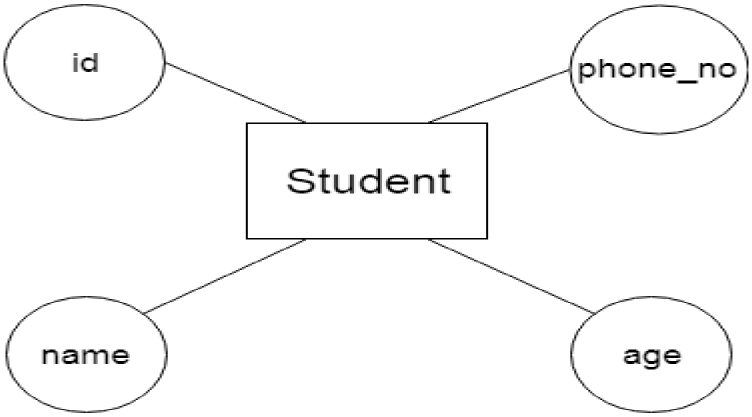


Fig 4.3

Composite Attribute

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

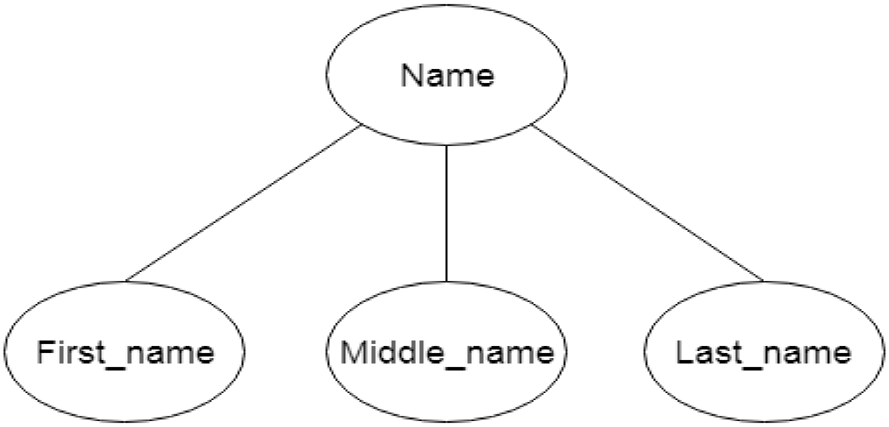


Fig 4.4

Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

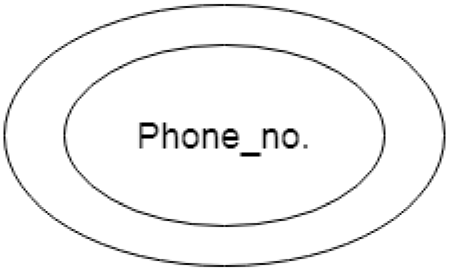


Fig 4.5

Derived Attribute

An attribute that can be derived from another attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.

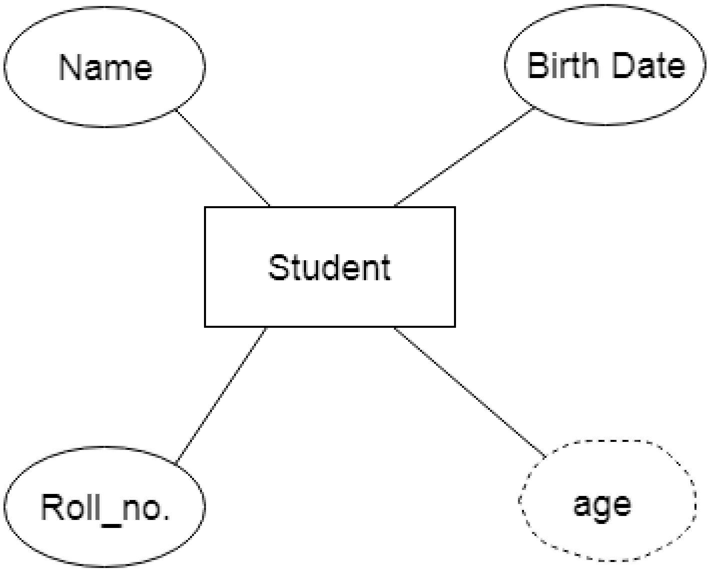


Fig 4.6

Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

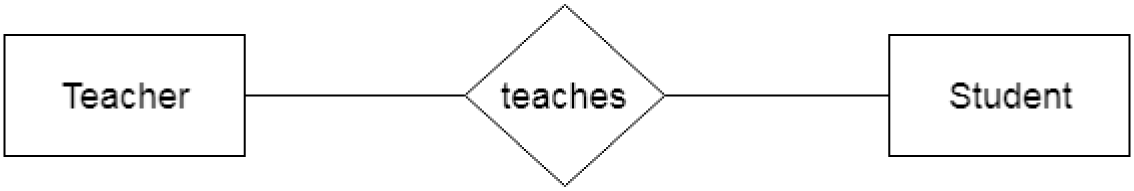


Fig 4.7

Types of Relationships are: -

1. One to one relationship
2. One to many relationship
3. Many to many relationship
4. Many to one relationship

Sample Output

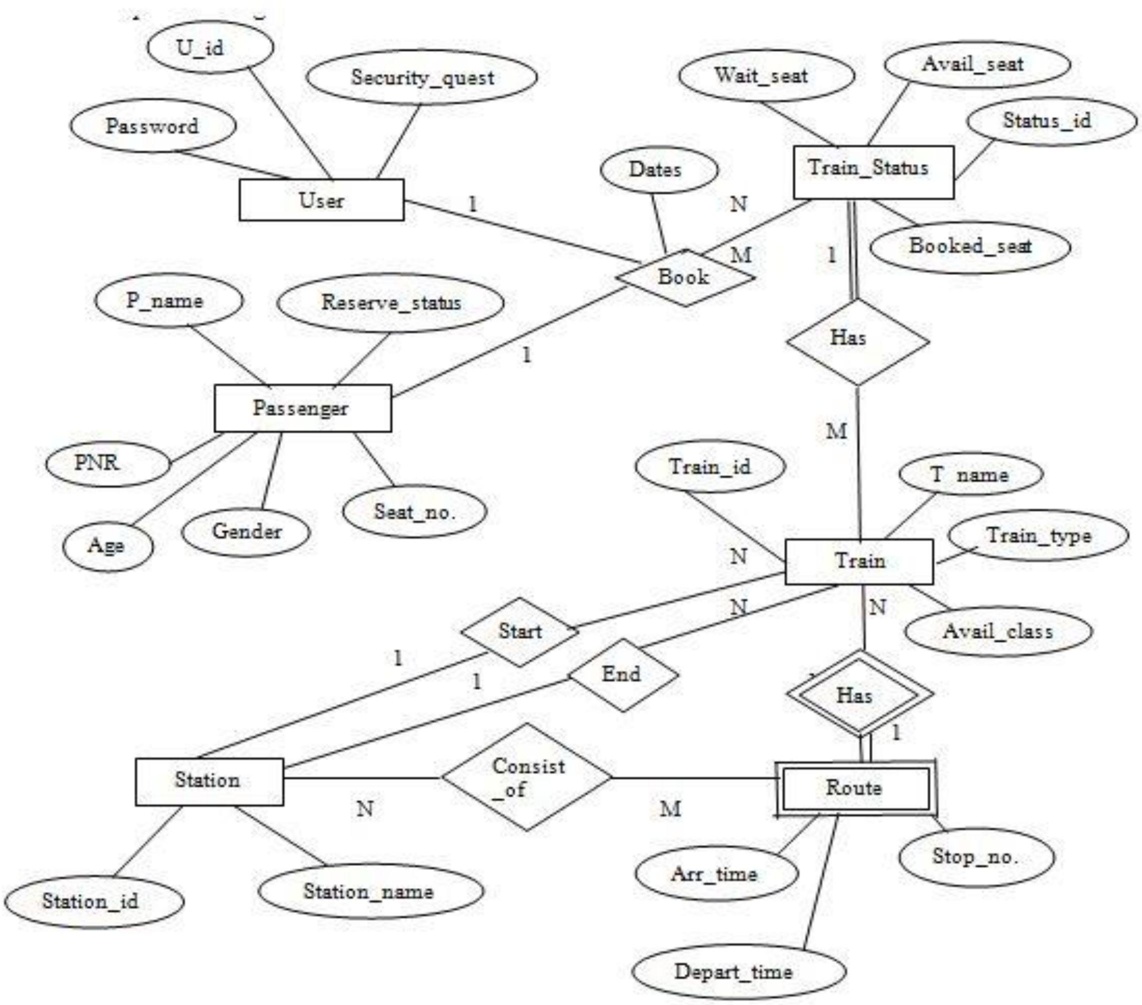


Fig 4.8

Experiment 5

Aim: To perform the user's view analysis for the suggested system: Use case diagram.

Description: The use-case diagram can provide the user's view for designing of the software product. And it can also be tested by matching up the requirements with the use-cases.
When to Use: Use Cases Diagrams
Use cases are used in almost every project. They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but asthe project continues more might become visible.
Actors---Are NOT part of the system – they represent anyone or anything that must interact with the system.
Only input information to the system. Only
receive information from the system.
Both input to and receive information from the system.

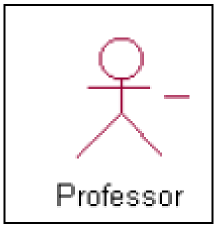


Fig 5.1

Represented in UML as a stickman.

Use Case
A sequence of transactions performed by a system that yields a measurable result of values for a particular actor
A use case typically represents a major piece of functionality that is complete from beginning to end. A use case must deliver something of value to an actor

Use Case Relationships

Between actor and use case.
Association / Communication.
Arrow can be in either or both directions; arrow indicates who initiates communication.
Between use cases (generalization):
Uses: Where multiple use cases share pieces of same functionality.

Performance Instruction:

1) Identify various processes, use-cases, actors etc. of the system and analyze it.
2) Use processes at various levels and draw use case diagram.

Sample Output:

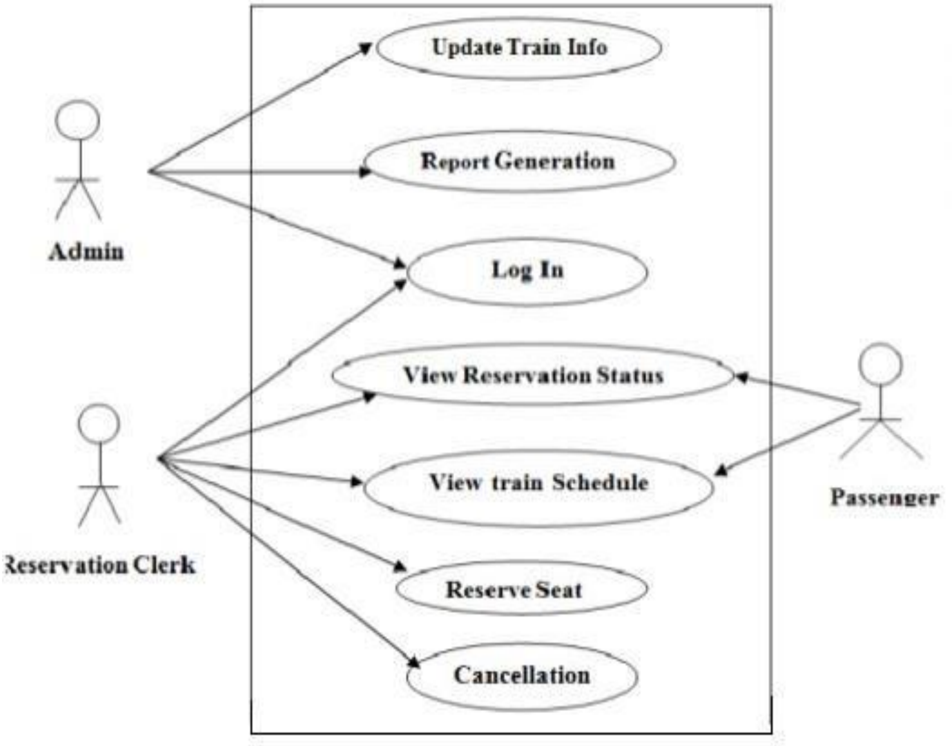


Fig 5.2

Use case Diagram

Conclusion: Use Case diagram was made successfully by following above steps.

Viva - Questions:

Q-1. Explain use case approach of requirement elicitation?

Q-2. Explain term: use-case, use-case scenarios, use-case diagrams?

Q-3. What are actors and use cases?

Q-4. Explain guidelines that should be kept in mind while creating use cases?

Q-5. Name the person who invented use case approach?

Experiment 6

Aim: To draw the structural view diagram for the system: Class diagram, object diagram.

Description: **Class diagrams** are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as
classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation.
Classes are composed of three things: a name, attributes, and operations. Below is an example of a class:

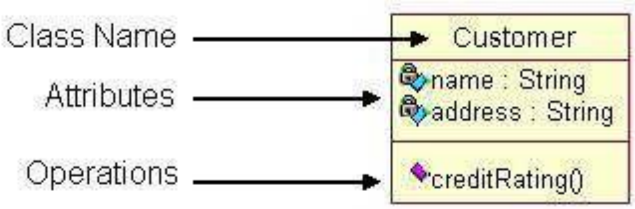


Fig 6.1

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are used to render a set of objects and their relationships as an instance.

Performance Instruction:

To draw class diagram

1) Identify various elements such as classes, member variables, member functions etc. of the class diagram
2) Draw the class diagram as per the norms

To draw object diagram

1) First, analyze the system and decide which instances have important data and association.
2) Second, consider only those instances, which will cover the functionality.
3) Third, make some optimization as the number of instances are unlimited.

Sample Output:

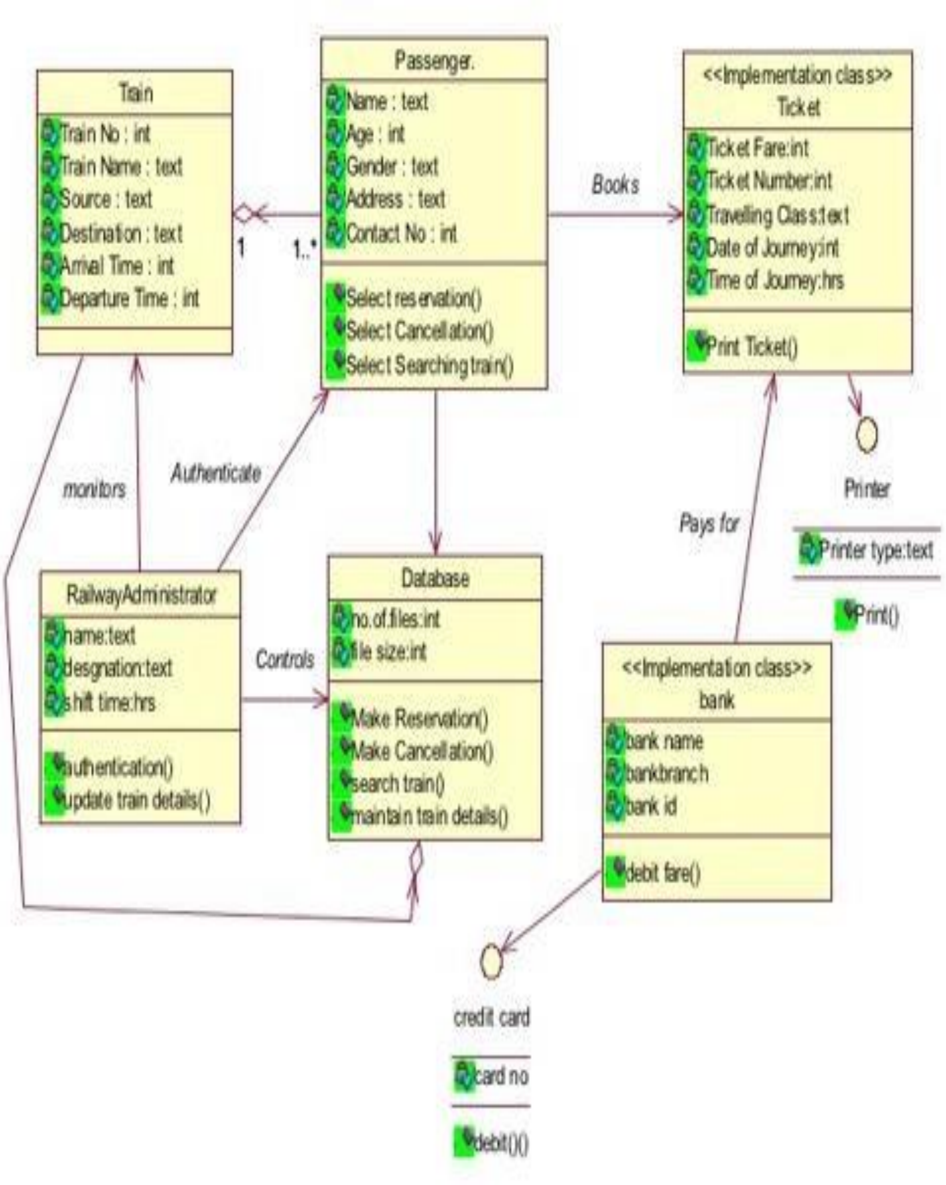


Fig 6.2

Class diagram

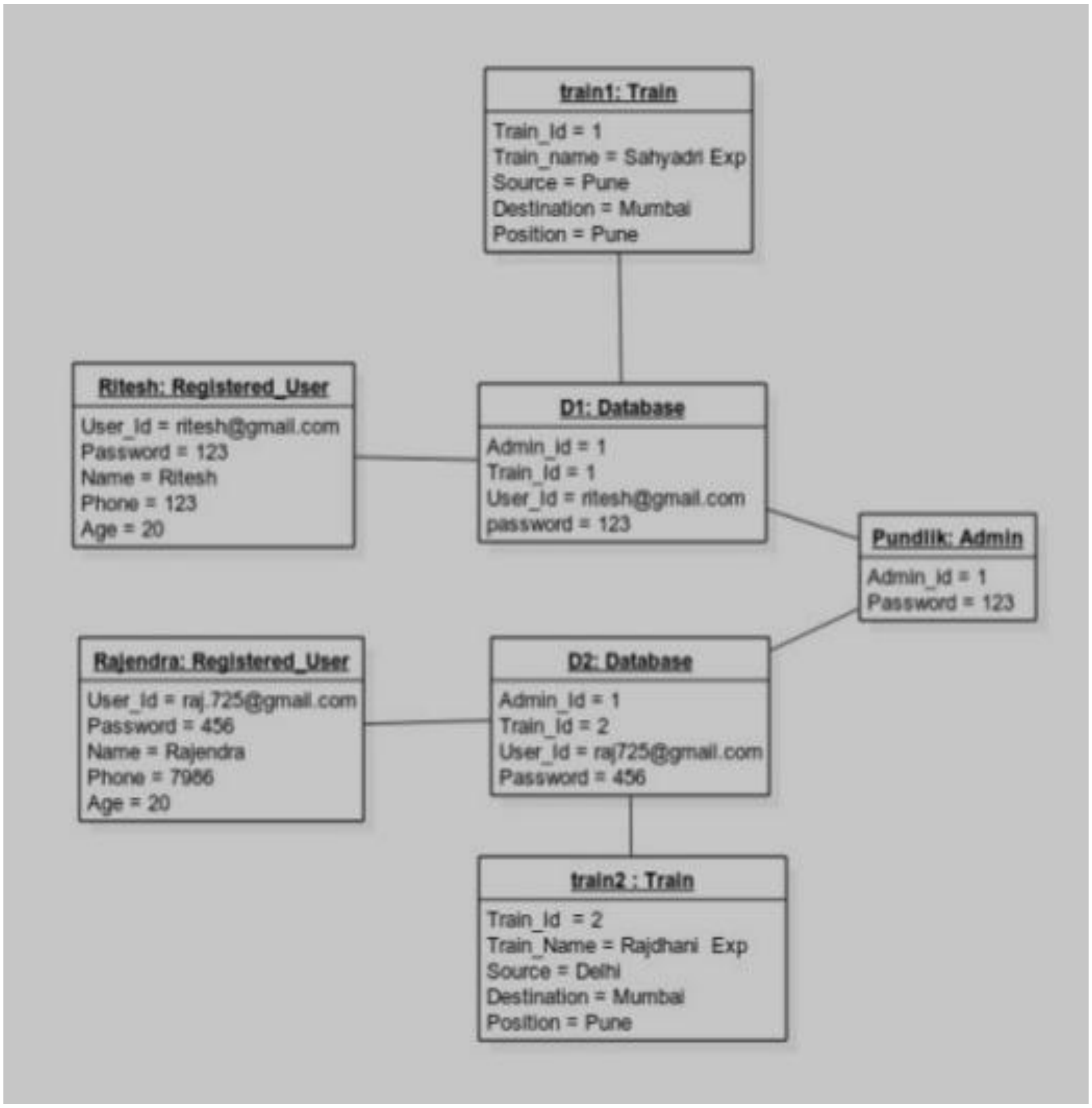


Fig 6.3

Object Diagram

Conclusion: Class diagram and Object diagram were made successfully by following above steps.

Viva - Questions:

Q-1. Explain class diagram?

Q-2. Explain symbols used in it?

Q-3. Explain four types of relationship used in class diagram?

Q-4. Explain terms classes, interfaces, collaborations and dependency?

Q-5. Explain object diagram?

Experiment 7

Aim: To draw the behavioral view diagram: State-chart diagram, Activity diagram

Description: State diagrams are used to describe the behavior of a system. State diagrams describe
all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system.

State chart diagrams represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances.

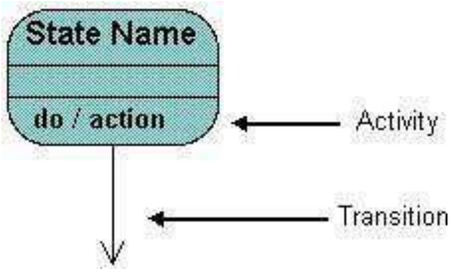


Fig 7.1

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel. Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some points are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.

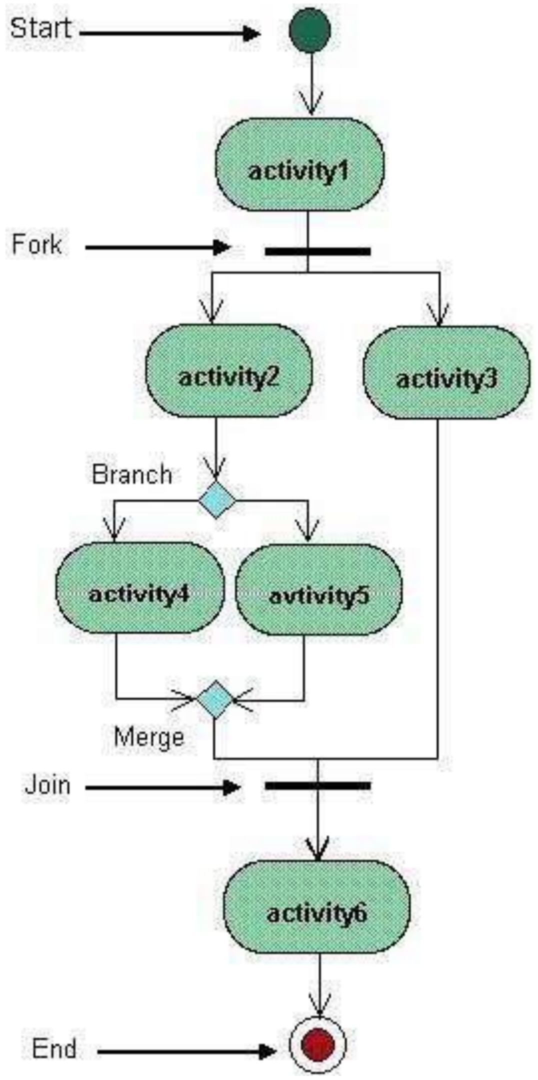


Fig 7.2

Performance Instruction:

To draw state chart diagram

1) Identify various elements states and their different transition of the state-chart diagram
2) Draw the state-chart diagram as per the norms.

To draw activity diagram

1) Identify various elements such as different activity their boundaries etc. of the activity diagram
2) Draw the activity diagram as per the norms.

Sample Output:

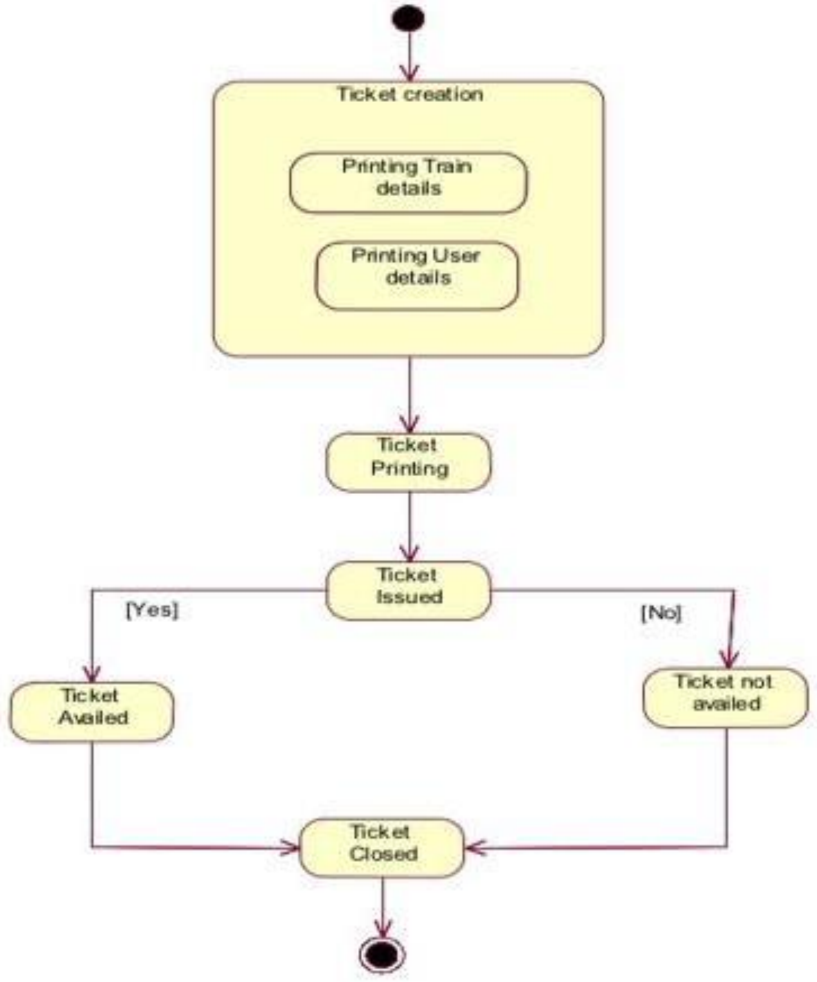


Fig 7.3

<u>State Chart Diagram</u>

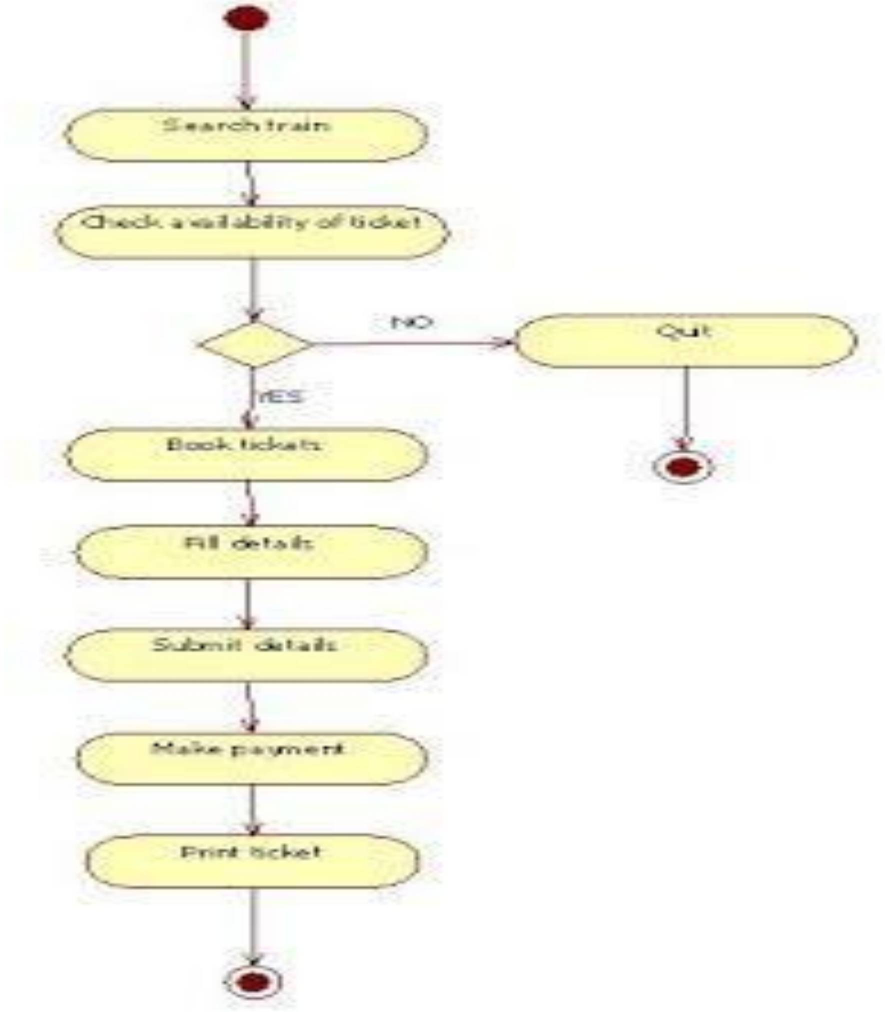


Fig 7.4

Activity diagram for booking ticket

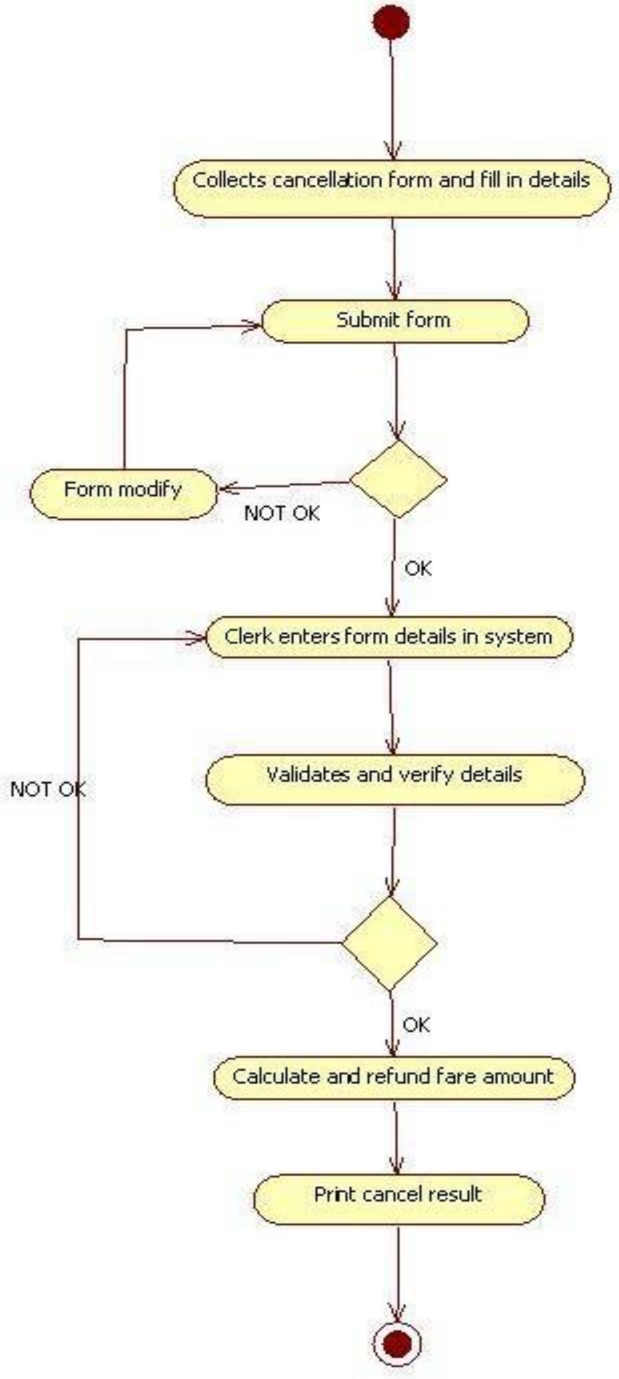

Collects cancellation form and fill in details
Submit form
Form modify
NOT OK
OK
Clerk enters form details in system
Validates and verify details
NOT OK
OK
Calculate and refund fare amount
Print cancel result

Fig 7.5

Activity Diagram for cancel ticket

Conclusion: State Chart and Activity diagram were made successfully by following above steps.

Viva - Questions:

Q-1. How activity diagram explains behavioral view of system?

Q-2. Explain steps followed to construct activity diagram?

Q-3. How state chart diagram explains behavioral view of system?

Q-4. Explain steps followed to construct state chart diagram?’

Q-5. Explain symbols used to construct these diagrams?

Experiment 8

Aim: To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram.

Description: **Sequence diagrams** show potential interactions between objects in the system being defined. Normally these are specified as part of a use case or use case flow and show how the use case will be implemented in the system. They include:

#Objects - oblong boxes or actors at the top - either named or just shown as belonging to a class, from, or to which messages are sent to other objects.

# **Messages** - solid lines for calls and dotted lines for data returns, showing the messages that are sent between objects including the order of the messages which is from the top to the bottom of the diagram.

#Object lifelines - dotted vertical lines showing the lifetime of the objects.

#Activation - the vertical oblong boxes on the object lifelines showing the thread of control in a synchronous system.

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self-explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects. A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

Collaboration Diagram

They are the same as sequence diagrams but without a time axis:

- Their message arrows are numbered to show the sequence of message sending.
- They are less complex and less descriptive than sequence diagrams.
- These diagrams are very useful during design because you can figure out how objects communicate with each other.

Performance Instruction:

To draw Sequence Diagram

1) Identify the class instances (objects) by putting each class instance inside a box.

2) If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line.

3) Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the return value above the dotted line.

To draw collaboration Diagram

1) By simply pressing combination of keys, we can design collaboration diagram from sequence diagram.

Sample Output:

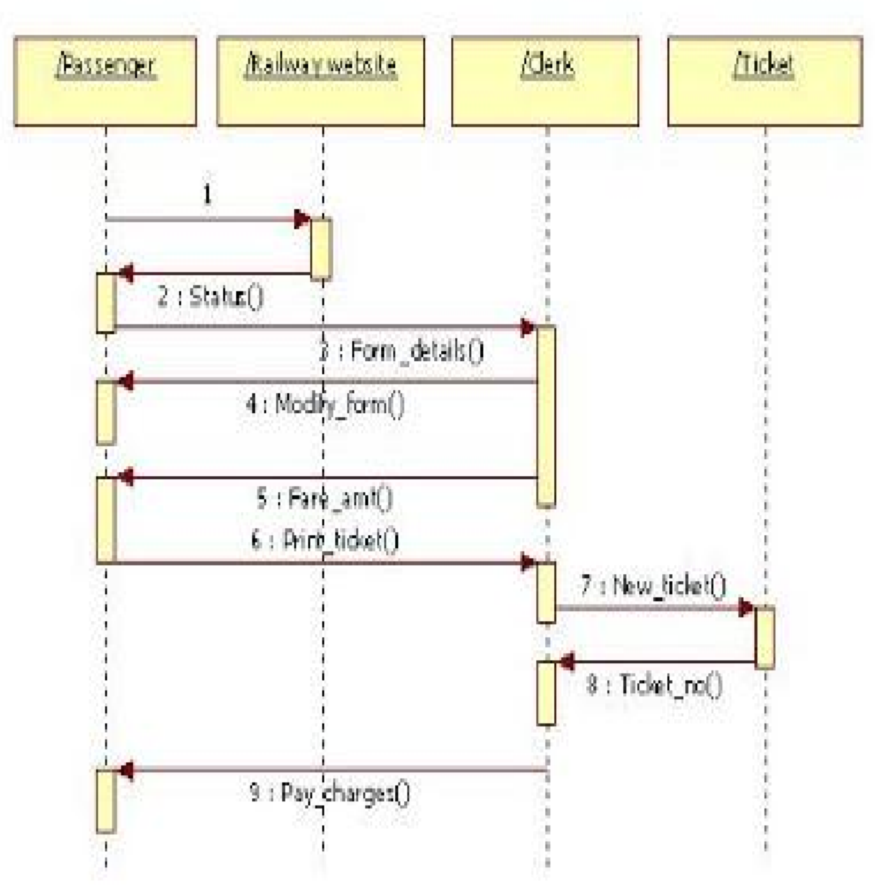


Fig 8.1

Sequence Diagram for booking ticket

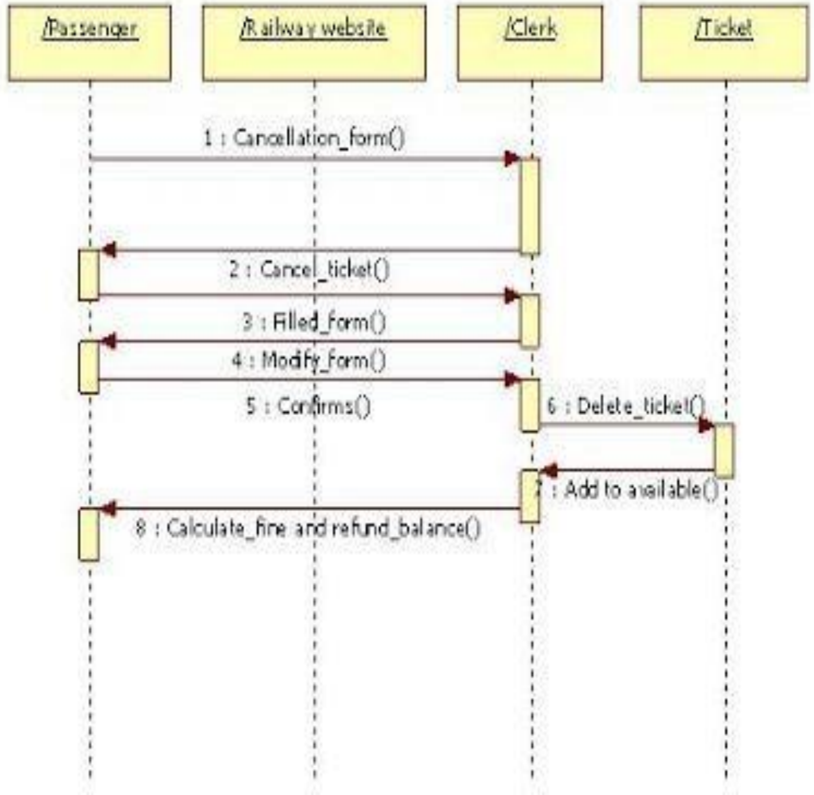


Fig 8.2

Sequence Diagram for cancel diagram

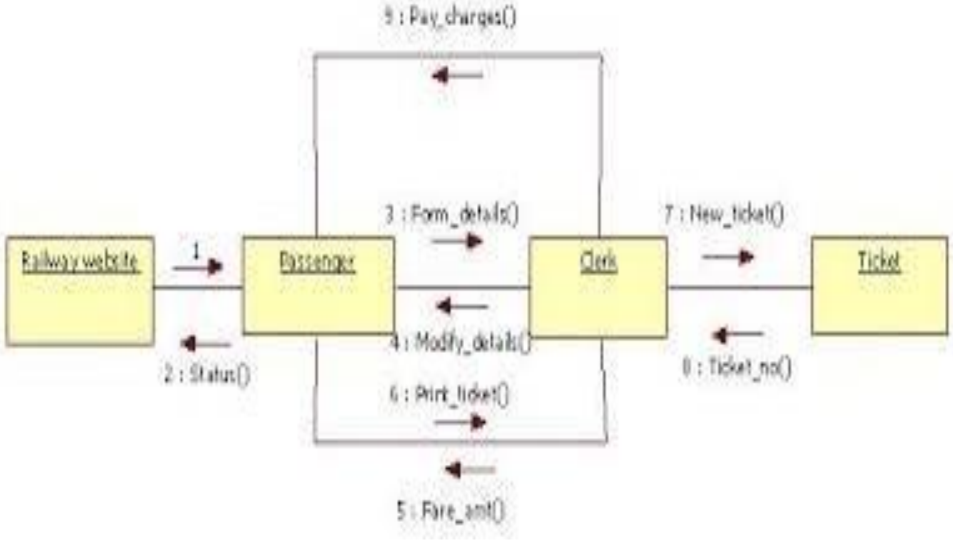


Fig 8.3

Collaboration Diagram for booking ticket

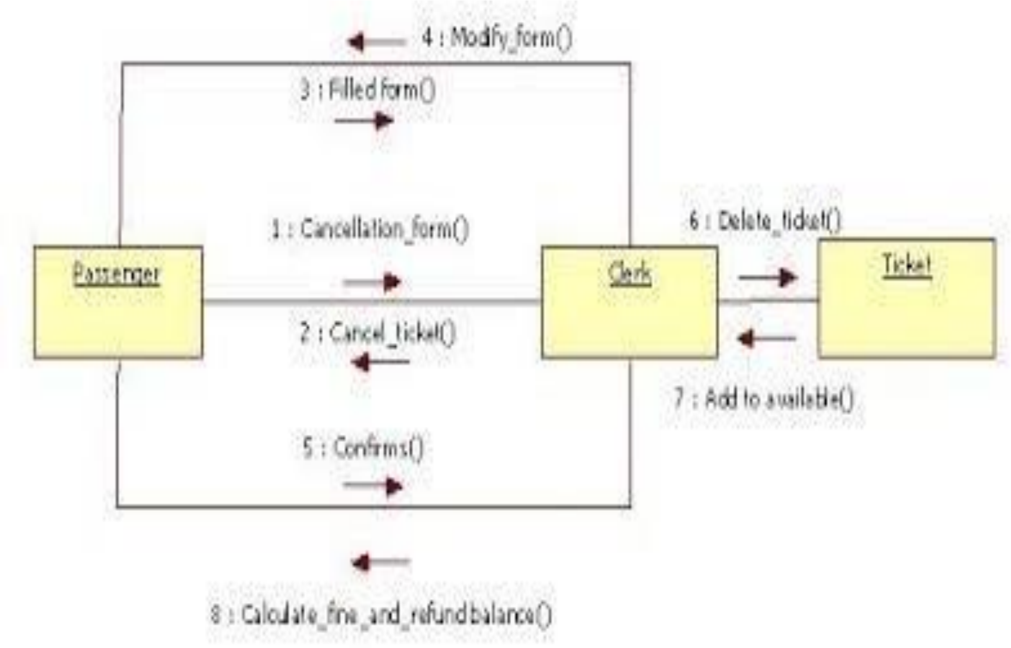


Fig 8.4

Collaboration Diagram for cancel ticket

Conclusion: Sequence diagram and Collaboration Diagram were made successfully by following above steps.

Viva - Questions:

Q-1. Explain use of sequence diagram?

Q-2. Explain steps to draw sequence diagram?

Q-3. Explain need of collaboration diagram?

Q-4. Explain steps to draw collaboration diagram?

Q-5. Explain terms- entity objects, interface objects and control objects?

Experiment 9

Aim: To perform the implementation view diagram: Component diagram for the system.

Description: A component diagram provides a physical view of the system. Its purpose is to show the dependencies that the software has on the other software components (e.g., software libraries) in the system. The diagram can be shown at a very high level, with just the large-grain components, or it can be shown at the component package level. [Note: The phrase component package level is a programming language-neutral way of referring to class container levels such as .NET's namespaces (e.g., System.Web.UI) or Java's packages (e.g., java.util).

Basic Component Diagram Symbols and Notations

Component

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.

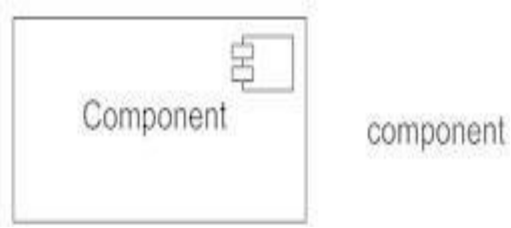


Fig 9.1

Interface

An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.

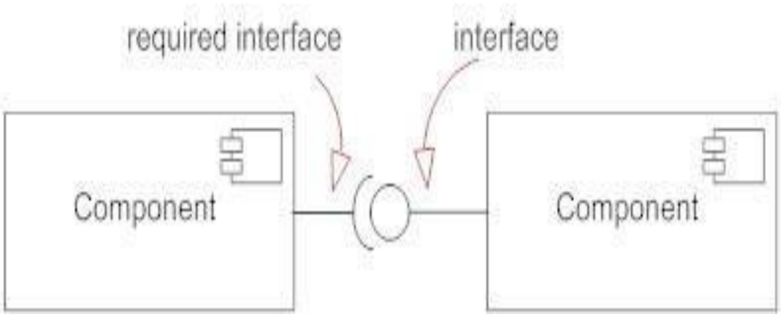
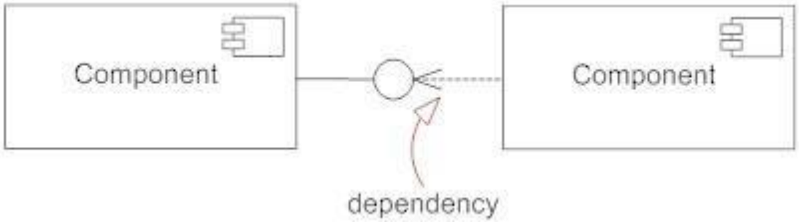


Fig 9.2

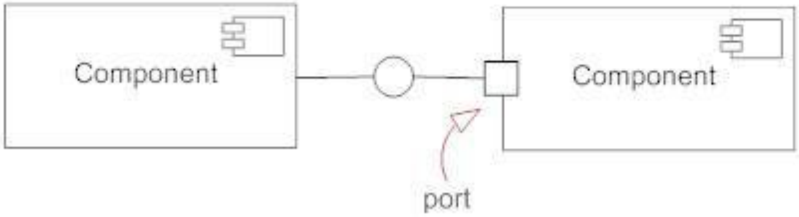
Dependencies

Draw dependencies among components using dashed arrows.



Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



Performance Instruction:

To Draw a Component Diagram

- Take stock of everything needed to implement the planned system. For example, for a simple e-commerce system, you'll need components that describe products, orders, and customer accounts.
- Create a visual for each of the components.
- Describe the organization and relationships between components using interfaces, ports, and dependencies.

Sample Output:

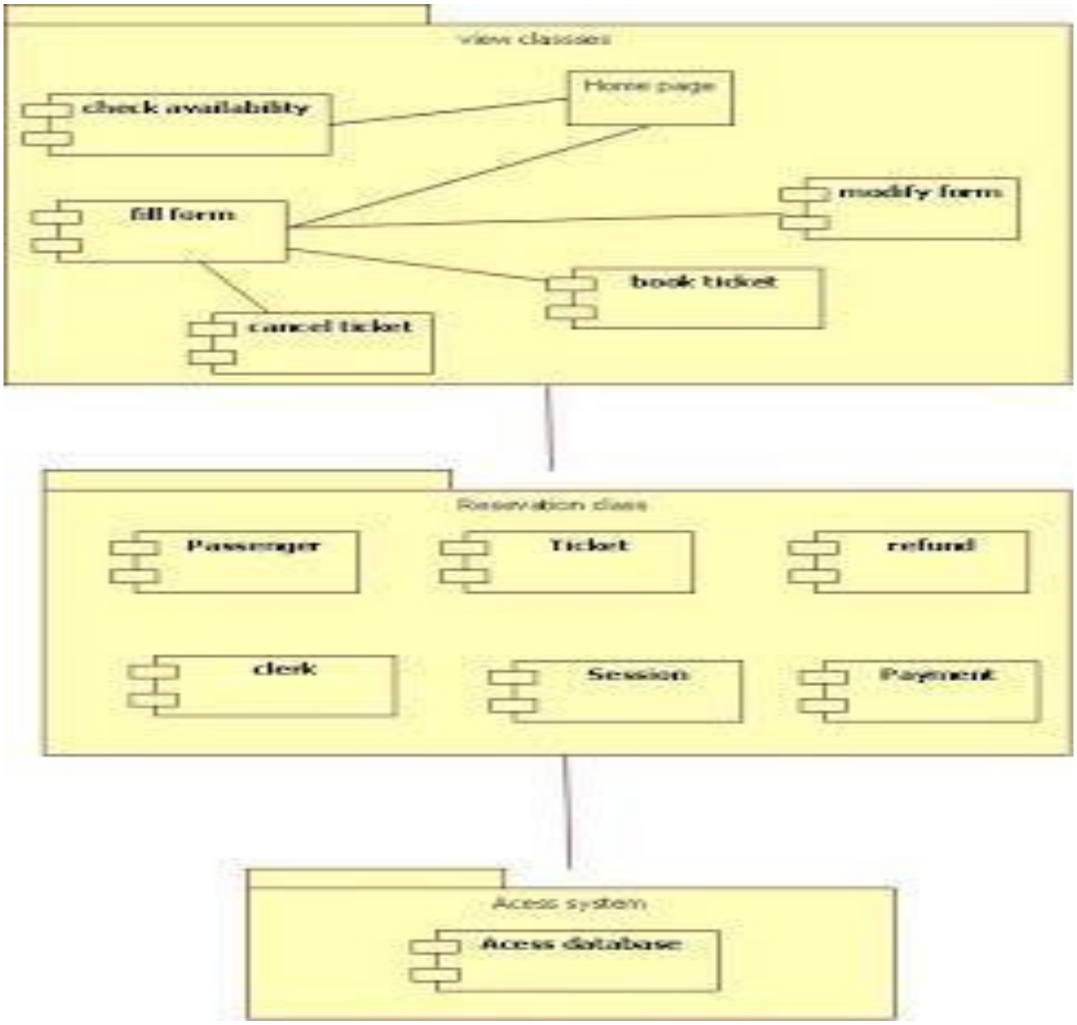


Fig 9.3

Component Diagram

Conclusion: Component diagram was made successfully by following above steps.

Viva - Questions:

Q-1. Explain term component diagram?

Q-2. Component diagram explains which view of system?

Q-3. Explain steps to draw component diagram?

Q-4. What is benefit of drawing component diagram?

Q-5. Explain symbols used to draw component diagram?

Experiment 10

Aim: To perform the environmental view diagram: Deployment diagram for the system.

Description: The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other. Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram.

The notation in a deployment diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node). To model a node, simply draw a three-dimensional cube with the name of the node at the top of the cube.

Performance Instruction:

To create a deployment diagram

1) Identify component
2) Add shapes
3) Connect Nodes
4) Format arrows

Sample Output:

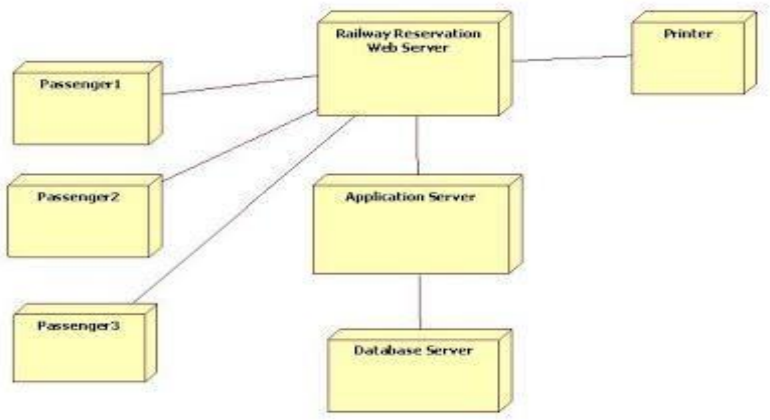


Fig 10.1 Deployment Diagram

Conclusion: Deployment diagram was made successfully by following above steps.

Viva - Questions:

Q-1. What is deployment diagram?

Q-2. Deployment diagram explains which view of system?

Q-3. Explain steps to draw deployment diagram?

Q-4. Explain symbols used to draw deployment diagram?

Q-5. What is benefit of drawing deployment diagram?

Experiment 11

Aim: To perform various testing using the testing tool unit testing, integration testing for a sample code of the suggested system.

Description:

"Testing is the process of executing a program with the intent of finding errors."
The purpose of software testing is
1. To demonstrate that the product performs each function intended.
2. To demonstrate that the internal operation of the product performs according to specification and all internal components have been adequately exercised.
3. To increase our confidence in the proper functioning of the software.
4. To show the product is free from defect.
5. All of the above.

Unit Testing—Checks each coded module for the presence of bugs. Unit testing's purpose is to ensure that each as- built module behaves according to its specification defined during detailed design.

Integration Testing ---Interconnects sets of previously tested modules to ensure that the sets behave as well as they did as independently tested modules. Integration testing's purpose is to ensure that each as-built component behaves according to its specification defined during preliminary design.

Performance Instruction:

Unit test planning—Generate plans and procedures to test each module independently and thoroughly.

Integration Test Planning—Generate plans and procedures to effect orderly system integration.

Sample Problem Statement:

Consider a program for the determination of the nature of roots of a quadratic equation. Its input is a triple of positive integers (say a,b,c) and values may be from interval [0,100]. The program output may have one of the following words. [Not a quadratic equation; Real roots; Imaginary roots; Equal roots] Design the boundary value test cases

Sample Output:

Quadratic equation will be of type:

ax2+bx+c=0

Roots are real if (b2-4ac)>0

Roots are imaginary if (b2-4ac)

Roots are equal if (b2-4ac) =0

Equation is not quadratic if a=0

The boundary value test cases are:

Test Case	a	b	c	Expected Output
1	0	50	50	Not Quadratic
2	1	50	50	Real Roots
3	50	50	50	Imaginary Roots
4	99	50	50	Imaginary Roots
5	100	50	50	Imaginary Roots
6	50	0	50	Imaginary Roots
7	50	1	50	Imaginary Roots
8	50	99	50	Imaginary Roots
9	50	100	50	Equal roots
10	50	50	0	Real Roots
11	50	50	1	Real roots
12	50	50	99	Imaginary Roots
13	50	50	100	Imaginary Roots

Table 11.1

The robust test cases are:

Test case	***a***	***b***	***c***	***Expected Output***
1	-1	50	50	Invalid input`
2	0	50	50	Not quadratic equation
3	1	50	50	Real roots
4	50	50	50	Imaginary roots
5	99	50	50	Imaginary roots
6	100	50	50	Imaginary roots

7	101	50	50	Invalid input
8	50	-1	50	Invalid input
9	50	0	50	Imaginary roots
10	50	1	50	Imaginary roots
11	50	99	50	Imaginary roots
12	50	100	50	Equal roots
13	50	101	50	Invalid input
14	50	50	-1	Invalid input
15	50	50	0	Real roots
16	50	50	1	Real roots
17	50	50	99	Imaginary roots
18	50	50	100	Imaginary roots
19	50	50	101	Invalid input

Table 11.2

Conclusion: Test Cases for given code was made successfully.

Viva - Questions:

Q-1. what is software testing?

Q-2. Explain terms unit testing and integration testing?

Q-3. Why should we test? Who should do the testing?

Q-4. Discuss limitation of testing?

Q-5. Explain term functional and structural testing?

Experiment 12

Aim: To Perform Estimation of effort using FP Estimation for chosen system.

Description: A **function point** is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software.

The principle of Albrecht’s function point analysis (FPA) is that a system is decomposed into functional units.

1. Inputs: information entering the system

2. Outputs: information leaving the system

3. Enquiries: requests for instant access to information

4. Internal logical files: information held within the system

5. External interface files: information held by other system that is used by the system being analyzed.

The FPA functional units are shown in figure given below:

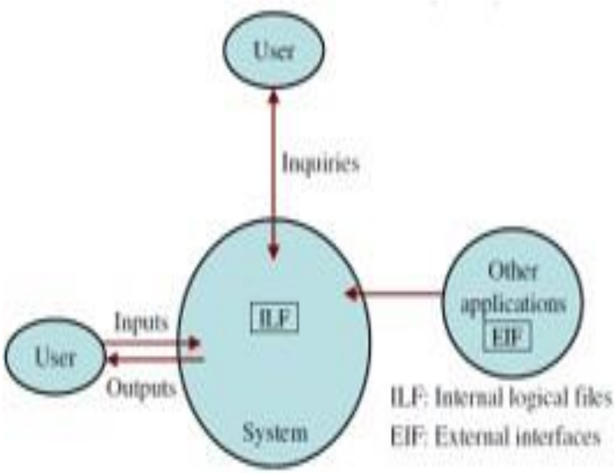


Fig 12.1

The five functional units are divided in two categories:

(i) Data function types

- Internal Logical Files (ILF): A user identifiable group of logical related data or control information maintained within the system.

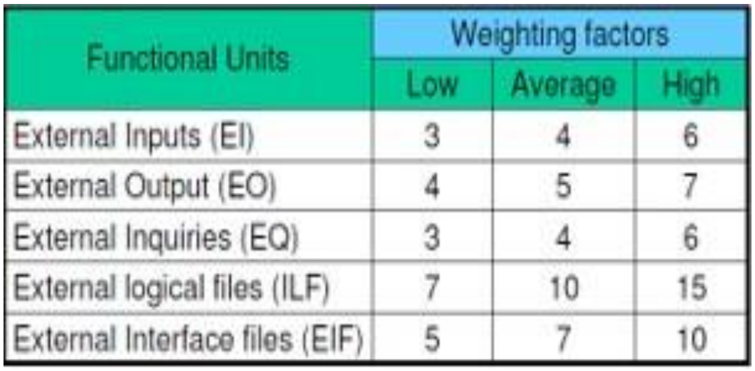
- External Interface files (EIF): A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. Thismeans that EIF counted for one system, may be an ILF in another system.

(ii) Transactional function types

- External Input (EI): An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business.
- External Output (EO): An EO is an elementary process that generate data or control information to be sent outside the system.
- External Inquiry (EQ): An EQ is an elementary process that is made up to an input-output combination that results in data retrieval.

Special features

- Function point approach is independent of the language, tools, or methodologies used for implementation; i.e., they do not take into consideration programming languages, data base management systems, processing hardware or any other data base technology.
- Function points can be estimated from requirement specification or design specification, thus making it possible to estimate development efforts in early phases of development
- Function points are directly linked to the statement of requirements; any change of requirements can easily be followed by a re-estimate.
- Function points are based on the system user's external view of the system, non-technical users of the software system have a better understanding of what function points are measuring.



Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Table 12.1

Performance Instruction:

1.Observe functional units and their weighting factors.
2.Compute them in formula to find value of UFP count.
3.Find value of FP by using formula.

Sample Problem Statement:

Consider a project with the following parameters.

(i)External Inputs:

(a) 10 with low complexity

(b)15 with average complexity

(c) 17 with high complexity

(ii) External Outputs:

(a) 6 with low complexity

(b)13 with high complexity

(iii) External Inquiries:

(a)3 with low complexity

(b)4 with average complexity

(c)2 with high complexity

(iv) Internal logical files:

(a) 2 with average complexity

(b)1 with high complexity

(v) External Interface files:

(a) 9 with low complexity

In addition to above, system requires

i. Significant data communication

ii. Performance is very critical

iii. Designed code may be moderately reusable

iv System is not designed for multiple installation in different organizations.

Other complexity adjustment factors are treated as average. Compute the function points for the project.

Sample Solution: Unadjusted function points may be counted using below table

Functional Units	Count	Complexity	Complexity Totals	Functional Unit Totals
External Inputs (EIs)	10 15 17	Low * 3 Average * 4 High * 6	30 60 102	192
External Outputs (EOs)	6 0 13	Low * 4 Average * 5 High * 7	24 0 91	115
External Inquiries (EQs)	3 4 2	Low * 3 Average * 4 High * 6	9 16 12	37
External logic Files (ILFs)	0 2 1	Low * 7 Average * 10 High * 15	0 20 15	35
External Interface Files (EIFs)	9 0 0	Low * 5 Average * 7 High * 10	45 0 0	45
Total Unadjusted Function Point Count = 424				

Table 12.2

Σ Fi = 3+4+3+5+3+3+3+3+3+3+2+3+0+3=41

CAF = (0.65 + 0.01* Σ Fi)

= (0.65 + 0.01* 41)

= 1.06

FP = UFP * CAF

= 424*1.06

= 449.44

Hence FP = 449

Conclusion: FP estimation was done successfully.

Viva - Questions:

Q-1. Explain five functional units of functional point analysis (FPA)?

Q-2. Explain special features of FPA?

Q-3. Explain three weighting factors of functional units?

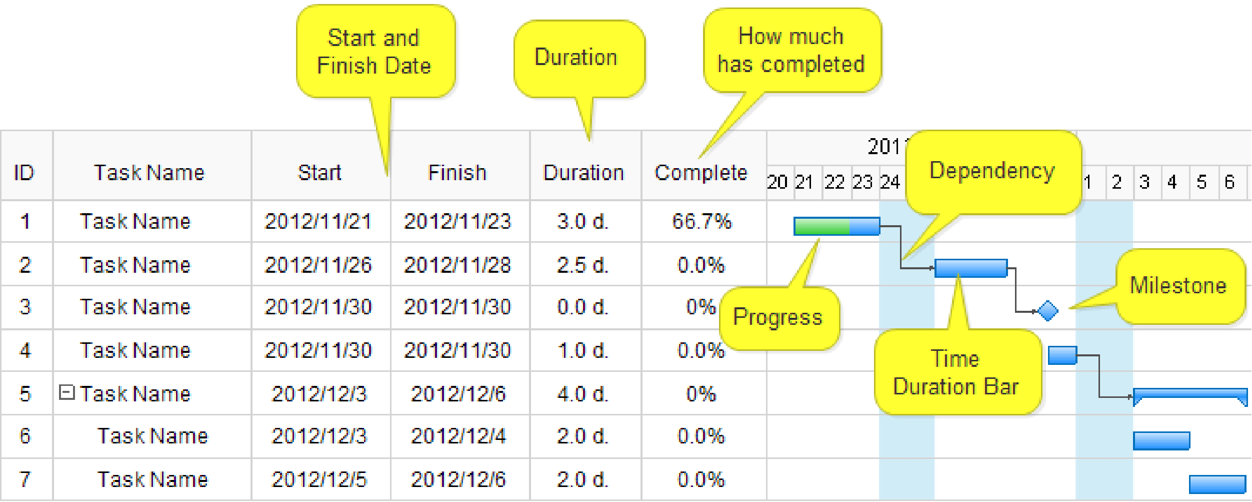
Q-4. Explain term unadjusted function point count (UFP)?

Q-5. What are uses of function point?

Experiment 13

Aim: To Prepare time line chart/Gantt Chart/PERT Chart for selected software project.
Description: A Gantt chart is a graphical depiction for planning and scheduling projects. It breaks a project into smaller pieces of tasks with each of these spread out over time. With it, you can see task dependencies, scheduling constraints, duration of each task and percent complete.

In a **Gantt Chart**, each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. The milestone is represented by a triangle, and the relationship is always shown with an arrow.



ID	Task Name	Start	Finish	Duration	Complete
1	Task Name	2012/11/21	2012/11/23	3.0 d.	66.7%
2	Task Name	2012/11/26	2012/11/28	2.5 d.	0.0%
3	Task Name	2012/11/30	2012/11/30	0.0 d.	0%
4	Task Name	2012/11/30	2012/11/30	1.0 d.	0.0%
5	⊟ Task Name	2012/12/3	2012/12/6	4.0 d.	0%
6	Task Name	2012/12/3	2012/12/4	2.0 d.	0.0%
7	Task Name	2012/12/5	2012/12/6	2.0 d.	0.0%

Table 13.1
Gantt Chart

PERT stands for **Program Evaluation Review Technique**. It was introduced by the United States Navy in the 1950s. Pert chart is also a project management tool used to plan tasks within aproject – making it easier to schedule and coordinate team members accomplishing the work.

Pert chart was initially developed to simplify the planning and scheduling of large and complexprojects. Pert chart looks like a flowchart. It uses boxes and arrows to show the subtasks and their dependencies.

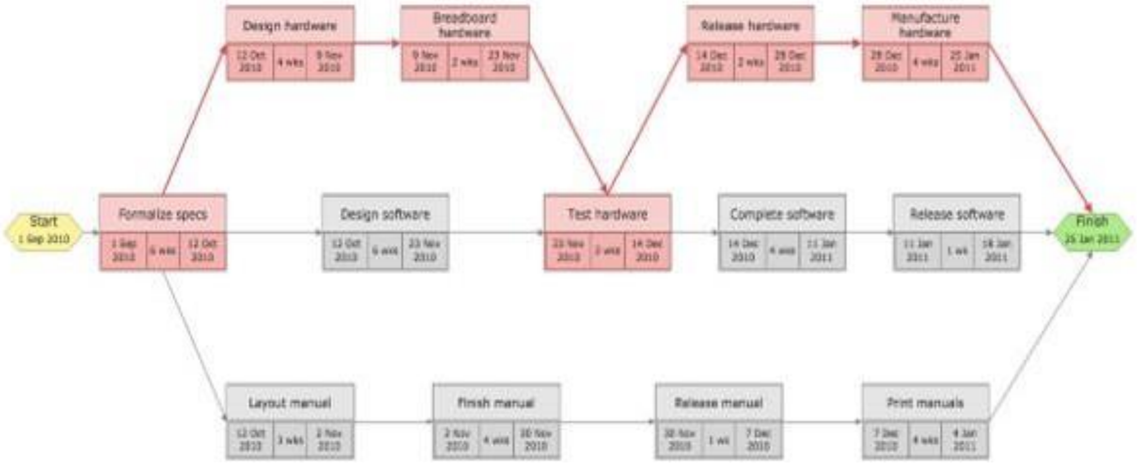


Fig 13.1 PERT Chart

Performance Instruction:

To Create a Gantt Chart

1. **Create a Task Table**

   List each task in your project in start date order from beginning to end. Include the task name, start date, duration, and end date.

2. **Build a Bar Chart**

Click in the empty Series name: form field first, then click on the table cell that reads Start Date.

To Create a PERT Chart.

1. Create a Text Box.

2. With the text box still selected, go to the Format tab under Drawing Tools.

3. Click on the inside of the text box and begin entering any information for a single task.

Sample Output:

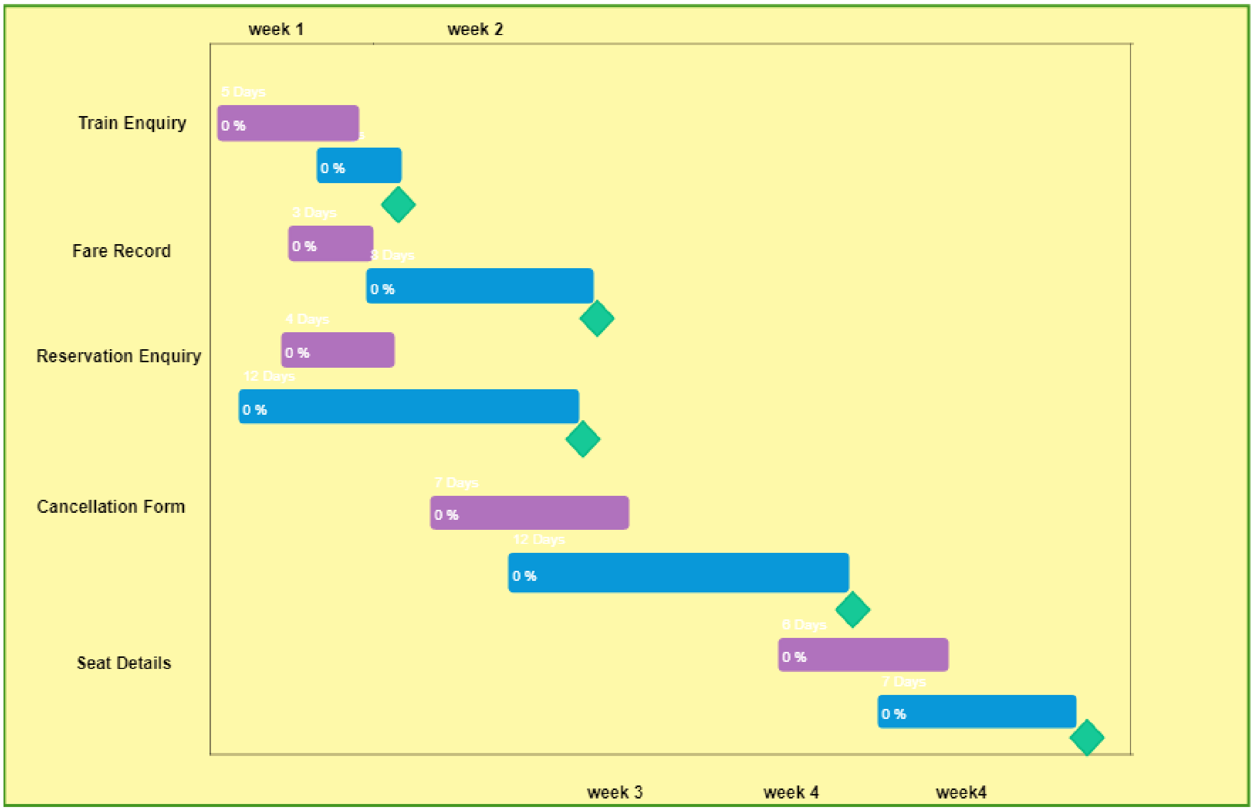


Fig 13.2Gantt Chart

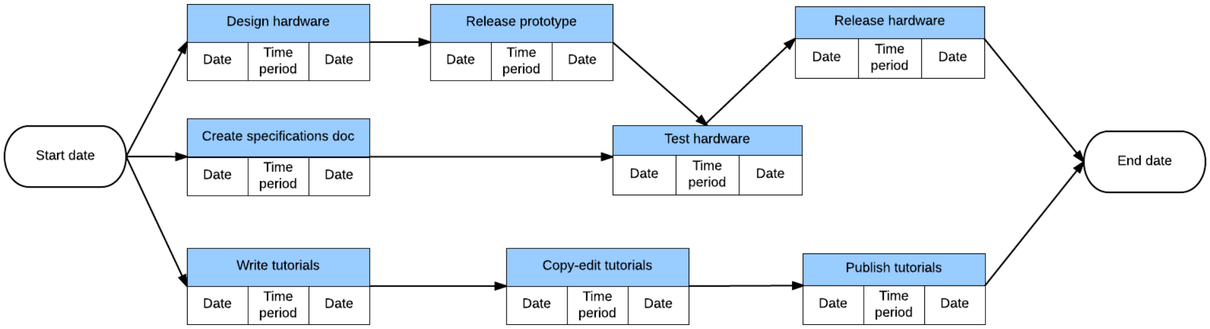


Fig 13.3PERT Chart

Conclusion: Gantt chart and PERT chart were made successfully.

Viva - Questions:

Q-1. Explain term Gantt chart?

Q-2. Explain term PERT chart?

Q-3. What is advantage of using PERT chart?

Q-4. What are advantages of using Gantt chart?

Q-5. What are limitations of Gantt chart?

Experiments beyond the syllabus

Experiment 1

Aim: To give the comparison between the starUML and Rational rose.

Tool Name	Systems	Supports	Code Generation For	Code Reverse Engineering
Rational Rose	Platform Needed for architecture	Class Diagrams, Use case diagrams, Sequence Diagrams, Collaboration Diagrams, State Diagrams, Activity Diagrams,	Java, C, C++	Java, C, C++
Star UML	The Open-Source UML/MDA Platform	Class Diagrams, Use case diagrams, Sequence Diagrams, Collaboration Diagrams, State Diagrams, Activity Diagrams,	Visual Basic, Java Script, Delphi, C#, Python	Visual Basic, Java Script, Delphi, C#, Python

Viva - Questions:

Q-1. Name nine modeling diagrams that are frequently used?

Q-2. What is UML? Explain the SDLC process?

Q-3. What are process workflows?

Experiment 2.

Aim: Why are entity relationship diagrams are used. Show the connectivity of entity relationship diagrams with database. Explain with the help of an example?

Description: An entity relationship model, *also called an* entity-relationship (ER) diagram, *is a* graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

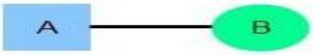
ER Diagram in DBMS is widely used to describe the conceptual design of databases. It helps both users and database developers to preview the structure of the database before implementing the database.

Relationships Between Entities

A relationship is how the data is shared between entities. There are three types of relationships between entities:

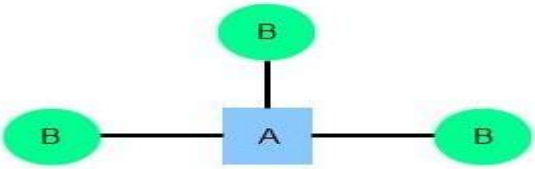
1. One-to-One

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



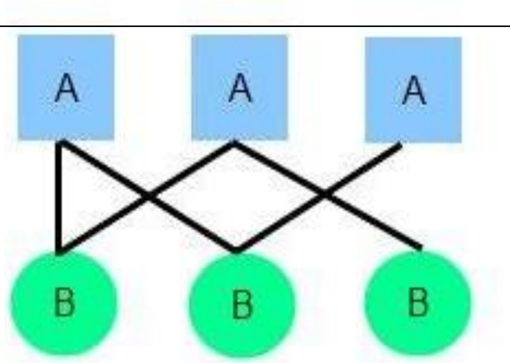
2. One-to-Many

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.

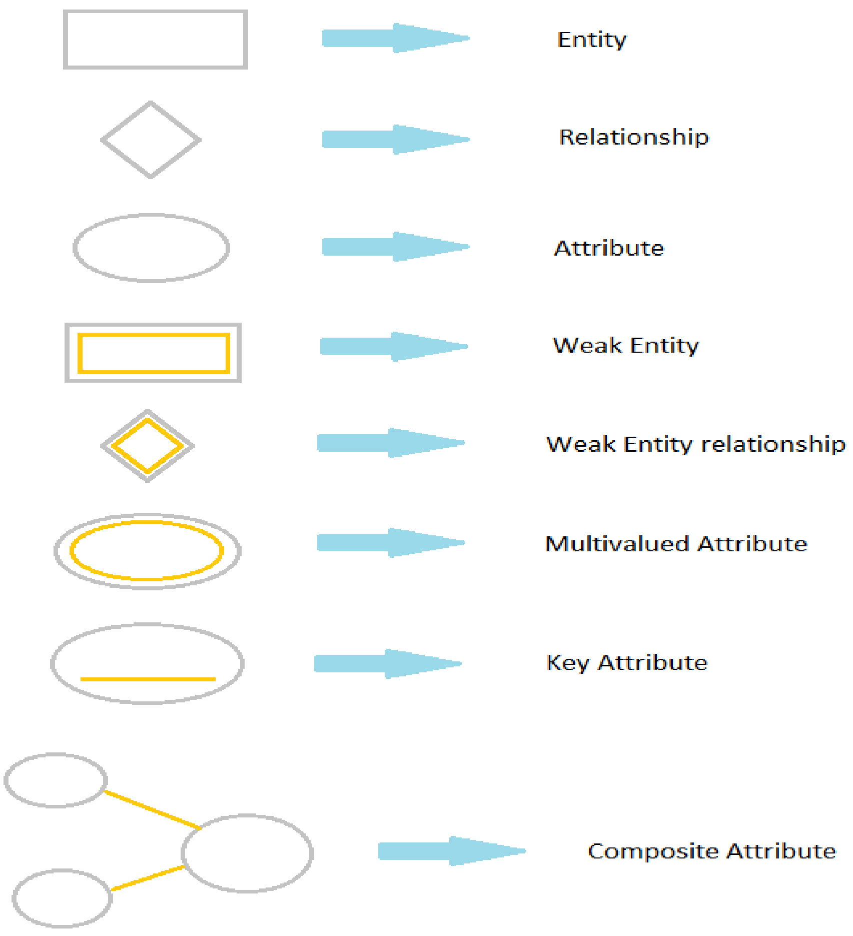


3. Many-to-Many

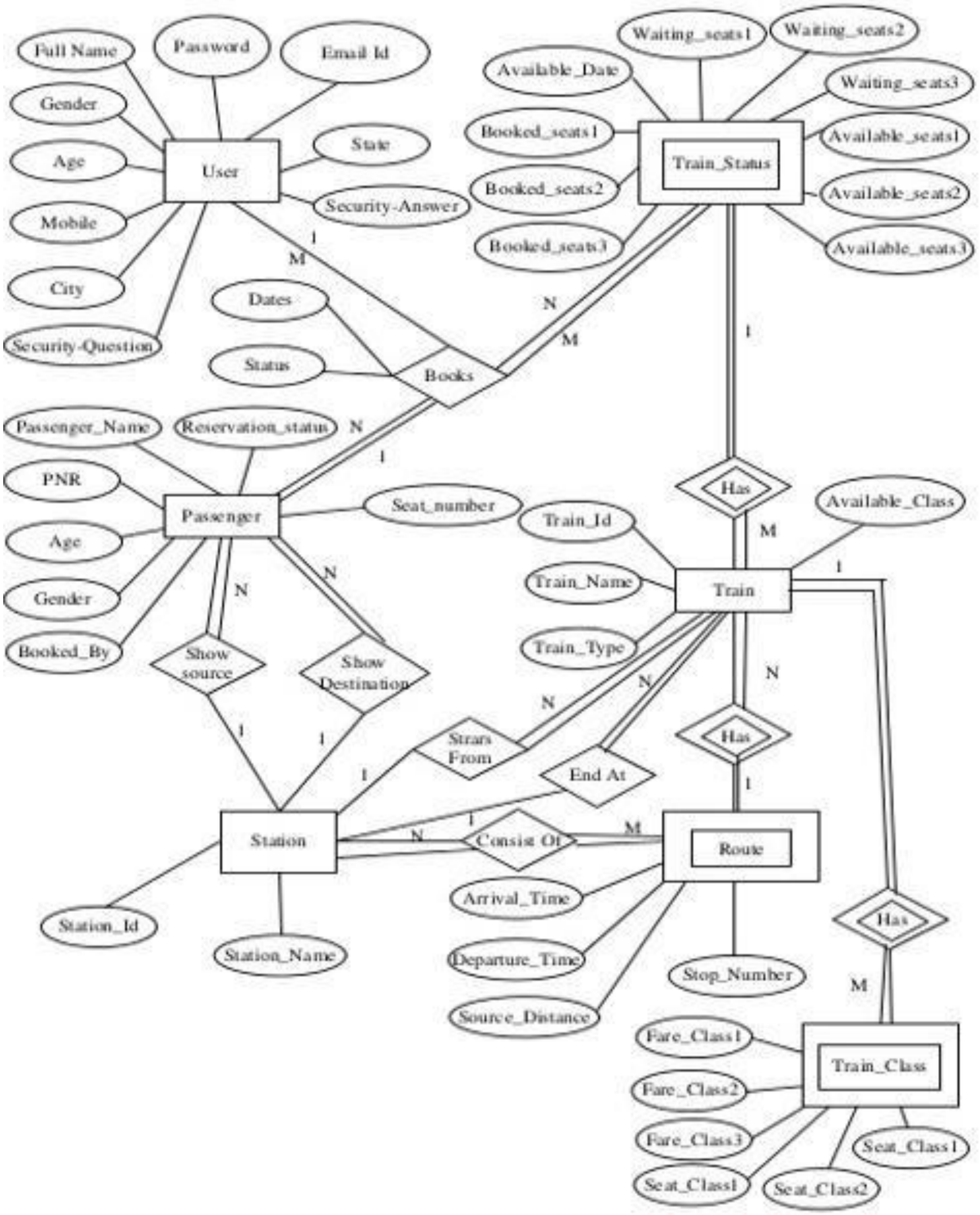
One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



Symbols used to draw ER diagram:



Sample Output:



ER diagram for Railway Reservation System

For example E-R Diagram for the company database

In DBMS, an ER diagram for a company database shows how the databases are connected. It also shows how all the databases are logically related to each other. We can also create an ER diagram by drawing the figure of a different part of the company database, their properties, and how they perform their task.

We can draw an ER diagram of the company database by drawing the design of the database. The sketch of the database became the storage of the database where the data comes and goes.

Details Company Database ER Diagram:

Now we describe the overall function of the Company database in the below table. It is a complete overview of the information about the company database project.

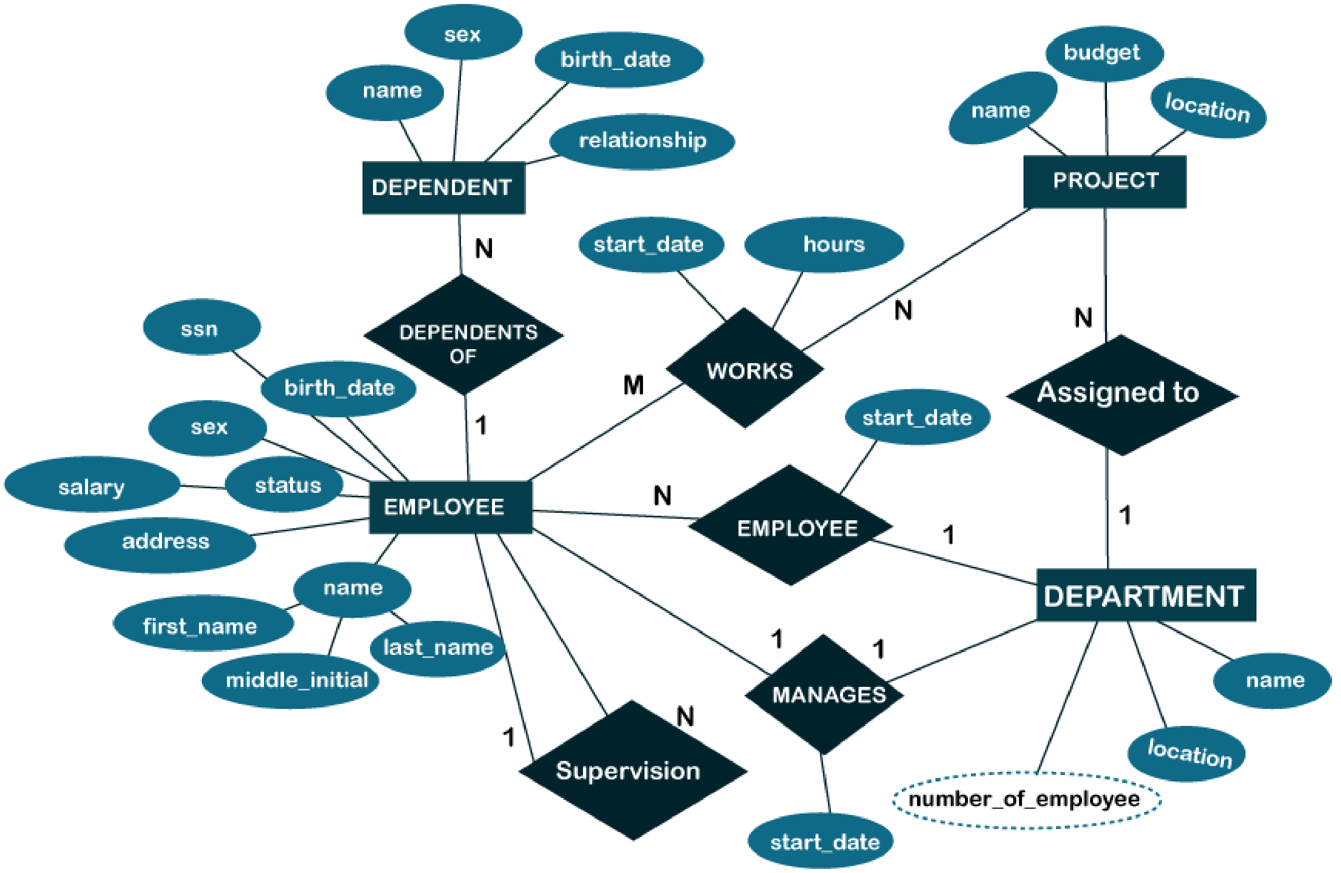
Name:	ER Diagram of the Company Database.
Abstract:	The ER Diagram of the company database shows the relationship between various entities. We can also call it the blueprint of the company database.
Diagram:	We can also call the ER diagram an Entity Relationship diagram.
Tool used:	The ER diagram provides some symbol that is known as the diagramming tool.
Users:	The users of the ER diagram are admin, website, and employee.

The company database is a database that contains all the information about the company and its employee. The information includes details about the company's demographic and firmographics details. The firmographic details of the company include the following things.

- Company name.
- Address of the head office.
- The employee strength of the company.
- Revenue of the company.
- Industry of the company.
- The technology used in the company.
- Link to the company's social media page.
- Funding-related information of the company.
- Website of the company

The ER diagram for the company database, the system data, and their attributes. The data and the attributes are represented by the table, and the table shows how they are related to each other.

Database Design for the Company Database



Conclusion: Connectivity of ER diagram with database was made successfully.

Viva - Questions:

Q-1. Explain ER diagram?

Q-2. What is entity? Explain strong and week entity?

Q-3. What is attributes? Explain different types of attributes?

Q-4. Explain three types of relationship?

Q-5. what is cardinality?

Software Engineering Viva Voice Questions

Q.1 What is SDLC?

Q.2 What are the various models available in SDLC?

Q.3 Explain the term Baseline?

Q.4 What are the responsibilities of a Software Project Manager?

Q.5 What is Cohesion?

Q.6 What is Coupling?

Q.7 Explain the concept of Modularization?

Q.8 What is Software Configuration Management?

Q.9 What are the various phases of SDLC?

Q.10 Provide examples of Project Management tools?

Q.11 What are CASE tools?

Q.12 What is Black box testing?

Q.13 What is White box testing?

Q.14 What is a Feasibility Study?

Q.15 How can you measure Project execution?

Q.16 What are the Functional Requirements?

Q.17 What are Non-Functional Requirements?

Q.18 What is the difference between Quality Assurance and Quality Control?

Q.19 What is the difference between Verification and Validation?

Q.20 Which SDLC model is the best to choose for a Software Product?

Q.21 What do you mean by Software Scope?

Q.22 What is SRS?

Q.23 What is the SDLC model that you have used in your previous project?

Q.24 Explain the Waterfall model in detail?

Q.25 What are the important categories of software?

Q.26 What is the main difference between a computer program and computer software?

Q.27 What is software re-engineering?

Q.28 What are SDLC models available?

Q.30 In software development process what is the meaning of debugging?

Q.31 Name two tools which are used for keeping track of software requirements?

Q.32 According to you which SDLC model is the best?

Q.33 Who is software project manager? What is his role?

Q.34 How to find the size of a software product?

Q.35 What are software project estimation techniques available?

Q.36 How can you measure project execution?

Q.37 What are software requirements?

Q.38 What are functional and non-functional requirements?

Q.39 What is modularization?

Q.40 Mentions some software analysis & design tools?

Q.41 What is mean by level-0 Data flow diagram?

Q.42 What is the major difference between structured English and Pseudo Code?

Q.43 What is Quality Assurance vs. Quality Control?

Q.44 What are CASE tools?

Q.45 Which process model removes defects before software get into trouble?

Q.46 How you can make sure that your written code which can handle various kinds of error situation?

Q.47 Describe the difference between Interface-oriented, Object-oriented and Aspect-oriented programming?

Q.48 Do you think that the maintenance of software is expensive?

Q.49 What does a System Engineering Model accomplish?

Q.50 What are the challenges in software?