Diploma Engineering

Laboratory Manual

Introduction to Software Engineering (4340702)

Diploma Computer Engineering Semester 4

Enrolment No	
Name	
Branch	
Academic Term	
Institute	

Directorate of Technical Education Gandhinagar Gujarat

DTE’s Vision:

- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls’ education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical.

Institute’s Vision:

Institute’s Mission:

Department’s Vision:

Department’s Mission:

Certificate

This is to certify that Mr./Ms…………………………………………Enrollment No. …………………….. of……… Semester of Diploma in……………… ……………………………………………………………………………..

Of Institute,……………..………………………………………………….

……………… (Gtu Code: …………...) has completed the term work satisfactorily in Subject Introduction to software engineering - 43040702 for the academic year : …….. Term: ……… as prescribed in the curriculum.

Place: ……………

Date: ……………

Subject Faculty **Head of the Department**

Preface

Main motto of any laboratory/Practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome based curriculum -2021 (COGC-2021) for engineering diploma programmes. In that more time allotted to practical work than theory .It shows importance of enhancement of skills amongst students and it pays attention to utilize every seconds of time allotted for practical amongst students, instructors and Lecturers to achieve relevant outcomes by performing rather than writing practice in study type. It is must for effective implementation of competency focused outcome- based green curriculum-2021, every practical has been keenly designed to serve as a tool to develop & enhance relevant industry needed competency in each and every student. This psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can read procedure one day advance before actual performance day of practical experiment which creates interest and also they have idea of judgement of magnitude prior to performance .This in turn enhances pre-determined outcomes amongst students. Each and every experiment /Practical in this manual begins by competency, industry relevant skills, course outcomes as well as practical outcomes which serve as a key role for doing the practical.

This manual also provides guidelines to lecturers to facilitate student-centered lab activities through each practical/experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve outcomes. It also gives an idea that how students will be assessed by providing Rubrics.

Introduction to software engineering course provides basic understating of software development process models, SRS and analyze software requirements, software design using DFD and object oriented UML diagrams, Project scheduling, test cases to test software. This course will be helpful to students to know about process of software development currently used in industry.

Although we try our level best to design this lab manual, but always there are chances of improvement. We welcome any suggestions for improvement.

Programme Outcomes (POs) to be achieved through Practical of this Course

Following programme outcomes are expected to be achieved through the practical of the course:

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods

3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs

4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements

5. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

6. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

7. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

Practical Outcome - Course Outcome matrix

Course Outcomes (COs):

a. CO1: Compare various software development process models.

b. CO2: Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

c. CO3: Prepare software development plan using project scheduling.

d. CO4: Prepare test-cases to test software functionalities.

Sr. No.	Practical Outcome	CO1	CO2	CO3	CO4
1.	Explain various software development models with appropriate diagram	√	-	-	-
2.	Write problem statement to define the project title with bounded scope of the project	√	-	-	-
3.	Select relevant process model to define activities and related tasks set for assigned project	√	-	-	-
4.	Gather application specific requirements	-	√		-
5.	Prepare broad SRS (software requirement software) for the above selected project	-	√	-	-
6.	Develop data designs using DFDs (data flow diagram) and E-R (entity-relationship) diagram	-	√	-	-
7.	Prepare use-cases and draw use case diagram	-	√	-	-
8.	Develop a class diagram for selected project	-	√	-	-
9.	Develop Sequence diagram for selected project	-	√	-	-
10.	Develop the activity diagram to represent flow from one activity to another for software development	-	√	-	-
11.	Evaluate size of the project using Function point metric for the assigned project.	-	-	√	-
12.	Estimate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach for the assigned project.	-	-	√	-
13.	Use flow chart and Gantt charts to track progress of the assigned project. (Use Sprint burn down chart if agile model is selected).	-	-	√	-
14.	Prepare various test case for selected project	-	-	-	√

Industry Relevant Skills

The following industry relevant skills are expected to be developed in the students by performance of experiments of this course.

1. Prepare SRS document.
2. Design various UML and project scheduling diagrams.
3. Prepare various test cases.

Guidelines to Teachers

1. Teacher should provide the guideline with demonstration of practical to the students.
2. Teacher is expected to refer complete curriculum document and follow guidelines for implementation strategies.
3. Teacher shall explain prior concepts and industrial relevance to the students before starting of each practical
4. Involve all students in performance of each experiment and should give opportunity to students for hands on experience.
5. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Finally give practical quiz as per the instructions. Utilise 2 hrs of lab hours effectively and ensure completion of write up with quiz also on same day.

Instructions for Students

1. Listen carefully the lecture, curriculum, learning structure, skills to be developed.
2. Organize the work in the group and make record of all observations.
3. Students shall develop maintenance skill as expected by industries.
4. Student shall attempt to develop related hand-on skills and build confidence.
5. Student shall develop the habits of evolving more ideas, innovations, skills etc.
6. Student shall refer technical magazines and data books.
7. Student should develop habit to submit the practical on date and time.
8. Student should well prepare while submitting write-up of exercise.

Continuous Assessment Sheet

Enrolment No: **Name**

Name: **Term:**

Sr. No.	**Practical Outcome/Title of experiment**	**Page**	**Date**	**Marks (25)**	**Sign**
1	Explain various software development models with appropriate diagram				
2	Write problem statement to define the project title with bounded scope of the project.				
3	Select relevant process model to define activities and related tasks set for assigned project				
4	Gather application specific requirements- Requirement gathering				
5	Prepare broad SRS (software requirement software) for the above selected project				
6	Develop data designs using DFDs (data flow diagram) and E-R (entity-relationship) diagram.				
7	Prepare use-cases and draw use case diagram				
8	Develop a class diagram for selected project				
9	Develop Sequence diagram for selected project				
10	Develop the activity diagram to represent flow from one activity to another for software development.				
11	Evaluate size of the project using Function point metric for the assigned project.				
12	Estimate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach for the assigned project.				
13	Use flow chart and Gantt charts to track progress of the assigned project. (Use Sprint burn down chart if agile model is selected).				
14	Prepare various test case for selected project.				
Total					

Practical No.1: Explain various software development models with appropriate diagram

A. **Objective:** Software Process model gives graphical representation of system to be built. Modeling contributes to a successful software organization. Modeling is a proven and well accepted engineering technique.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problem.

C. **Expected Skills to be developed based on competency:**

This practical is expected to develop the following skills for the industry identified competency: 'Select the relevant software process model for the given problem.'

1. Software Model Selection skills
2. Apply appropriate Software model for the given problem

D. **Expected Course Outcomes(Cos)**

Compare various software development process models.

E. **Practical Outcome(Pro)**

1. Recommend the relevant software solution for the given problem.
2. Select the relevant software process model for the given problem statement with justification.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Software development process model

The term **software** specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.

A software process is the set of activities and associated outcome that produce a software product. Software engineers mostly carry out these activities. These are four key process activities, which are common to all software processes. These activities are:

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software development:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

The Software process model.

A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering. Some examples of the types of software process models that may be produced are:

1. **A workflow model:** This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
2. **A dataflow or activity model:** This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.
3. **A role/action model:** This means the roles of the people involved in the software process and the activities for which they are responsible.

There are several various general models or paradigms of software development:

1. **The waterfall approach:** This takes the above activities and produces them as separate process phases such as requirements specification, software design, implementation, testing, and so on. After each stage is defined, it is "signed off" and development goes onto the following stage.
2. **Evolutionary development:** This method interleaves the activities of specification, development, and validation. An initial system is rapidly developed from a very abstract specification.
3. **Formal transformation:** This method is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program. These transformations are 'correctness preserving.' This means that you can be sure that the developed programs meet its specification.
4. **System assembly from reusable components:** This method assumes the parts of the system already exist. The system development process target on integrating these parts rather than developing them from scratch.

H. Practical related Quiz.

1. List various software process models.
2. Explain each software process model in detail with diagram.

I. Assessment-Rubrics

Practical 1: Explain various software development models with appropriate diagram			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.2: Write problem statement to define the project title with bounded scope of the project.

A. **Objective:** Student will explore different areas in field of software engineering for problem definition and select project appropriately. Also it will develop ability to manage the real world problems in the process of software development.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems

C. **Expected Skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency: 'Define the project title with bounded scope of the project.'

1. Identify real world problem definition.

D. **Expected Course Outcomes(Cos)**

Explain fundamentals of Software Engineering

E. **Practical Outcome(Pro)**

1. Identify real word problem and define in terms of software engineering.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

What is a Project Description?

A ***project description*** is a high-level overview of why you're doing a project. The document explains a project's objectives and its essential qualities

How to Write a Project Description

Although writing a project description will vary somewhat depending on the type of project, the basic process is the same. The following 10 steps are key to writing a good project description.

1. **Summarize:** Write a one- or two-paragraph explanation of what the project aims to accomplish. Avoid delving deep into background or past projects. A good project summary will not only serve as your elevator speech, but will also help you clarify larger issues with your plan.
2. **Define:** Describe the problem or opportunity and how the project will address it.
   - **Set goals:** Identify SMART project objectives, defined as follows:
   - **Specific:** Answer who, what, when, where, and why.
   - **Measurable:** Include metrics for defining success.
   - **Achievable:** Set goals that are possible to accomplish with the available resources.
   - **Relevant:** Goals should be aligned with your organization's mission.
   - **Time-bound:** Include intermediate and final deadlines for each goal.
3. **Explain:** Briefly explain your methodology. Include any key technologies or project management techniques you'll use and why they're appropriate.
4. **Measure:** Identify the project deliverables. How will you measure success and evaluate the project?
5. **Schedule:** Include a general timeline, with project phases and milestones. Be sure to note any important deadlines.

6. **Budget:** Include the total estimated cost of the project and how much you have budgeted. (Note that this shouldn't be a line item budget.) Use a project budget template for a more detailed breakdown of budgeted and actual project expenses.
7. **Get feedback:** Seek feedback from key stakeholders, customers, and anyone impacted by the project for feedback. Ask them to explain the project in their own words to get a sense of how clearly you've communicated the vision.
8. **Peer Review:** Have someone else read the project description. In addition to spelling and grammatical errors, ask them to look for missing details that are significant to the project.
9. **Revise:** Update and revise the document as the project progresses. Treat the project description as a living document.

H. Practical related Quiz.

1. Select and Write your system (project) definition with brief introduction.
2. Write abstract, scope of the selected project and need of your system.

I. Assessment-Rubrics

Practical 2: Write problem statement to define the project title with bounded scope of the project.			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.3: Select relevant process model to define activities and related tasks set for assigned project

A. Objective: Software Process model gives graphical representation of system to be built. Modeling contributes to a successful software organization. Modeling is a proven and well accepted engineering technique.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems

C. Expected skills to be developed based on competency

This practical is expected to develop the following skills for the industry identified competency: 'Define the project title with bounded scope of the project.'

1. Software Model Selection skills
2. Apply appropriate Software model for the given problem

D. Expected Course Outcomes(Cos)

Explain fundamentals of Software Engineering

E. Practical Outcome(Pro)

1. Recommend the relevant software solution for the given problem.
2. Select the relevant software process model for the given problem statement with justification.

F. Expected Affective domain Outcome(ADos)

1. Work as a leader/ a team member
2. Follow ethical practice.

G. Prerequisite Theory:

How to select the right software process model.

Selecting the right software process model is a process in itself that the organization can implement internally or consult for. There are some steps to get the right selection.

STEP 1: Learn the about software process model.

STEP 2: Assess the needs of Stakeholders

We must study the business domain, stakeholders concerns and requirements, business priorities, our technical capability and ability, and technology constraints to be able to choose the right SDLC against their selection criteria.

STEP 3: Define the criteria

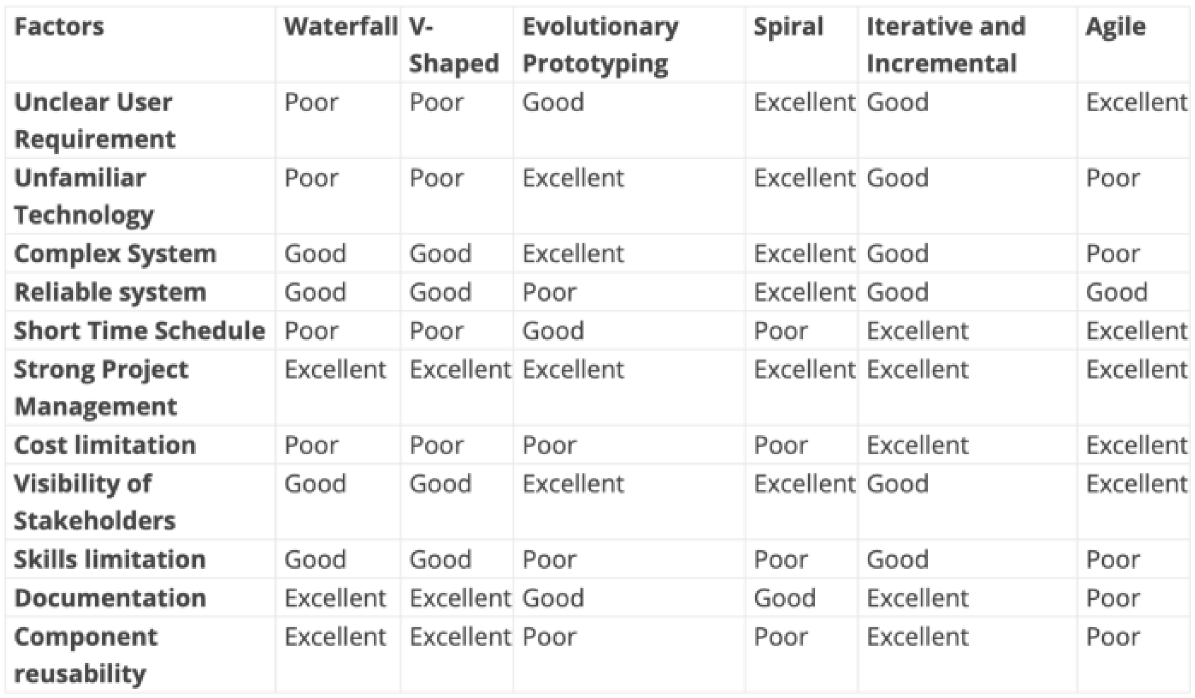
Some of the selection criteria or arguments that you may use to select software process model are:

- Is the software process model suitable for the size of our team and their skills?
- Is the software process model suitable for the selected technology we use for implementing the

solution?

- Is the software process model suitable for client and stakeholders concerns and priorities?
- Is the software process model suitable for the geographical situation (distributed team)?
- Is the software process model suitable for the size and complexity of our software?
- Is the software process model suitable for the type of projects we do?
- Is the software process model suitable for our software engineering capability?
- Is the software process model suitable for project risk and quality insurance?
- What are the criteria?

Here are some recommended criteria.



Factors	**Waterfall**	**V-Shaped**	**Evolutionary Prototyping**	**Spiral**	**Iterative and Incremental**	**Agile**
Unclear User Requirement	Poor	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Poor	Good	Poor	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Good	Excellent	Excellent	Good	Excellent
Skills limitation	Good	Good	Poor	Poor	Good	Poor
Documentation	Excellent	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Poor	Excellent	Poor

STEP 4: Decide

When you define the criteria and the arguments you need to discuss with the team, you will need to have a decision matrix and give each criterion a defined weight and score for each option. After analyzing the results, you should document this decision in the project artifacts and share it with the related stakeholders.

STEP 5: Optimize

You can always optimize the software process model during the project execution, you may notice upcoming changes do not fit with the selected software process model, it is okay to align and cope with the changes. You can even make your own SDLC model which optimum for your organization or the type of projects you are involved in.

H. Practical related Quiz.

1. Identify software process model for your system and justify your selection of particular process model.

I. Assessment-Rubrics

Practical 3: Select relevant process model to define activities and related tasks set for assigned project.			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.4: Gather application specific requirements- Requirement gathering.

A. **Objective:** Requirements Gathering is the process of generating a list of requirements (functional, system, technical, etc.) from all the stakeholders (customers, users, vendors, IT staff) that will be used as the basis for the formal definition of what the project is.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency:**

This practical is expected to develop the following skills for the industry identified competency:

1. Gather application specific requirements- Requirement gathering.

D. **Expected Course Outcomes(Cos)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pro)**

1. Identify software requirement for the given problem.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Requirements are an essential part of any software project and the foundation on which all projects should be built. The gathering of and compiling of requirements for a software project is very much a partnership between the user of the software and the developer. Obviously, the customer or software user needs to communicate to the developer what they need, but at the same time the developer needs to be able to anticipate needs and ask the right questions during the requirements gathering phase of a project.

Some techniques for gather requirements include

One-on-one Interviews

Sit down with the client and ask them what they need. Asking open-ended questions and getting the client talking about their hopes for the project is a good way to understand their needs.

Group Interviews

Similar to the one-on-one, but with a group of people, the same types of questions are asked, but with more of the users in the room, they may pick up on other's statements and expand or correct them. This can be helpful, but too many cooks spoil the broth and these meetings can stray from the goal.

Questionnaires

Providing questionnaires to the customer to fill out can be a good starting point or supplemental approach to interviewing.

Use Cases

A use case if a story about how a certain process in the software should work, they may be easier for users to communicate clearly.

There are many more techniques and methods of gathering good requirements the above are only a few.

H. Practical related Quiz.

1. Write down set of questions to collect functional requirements form various stake holders for your system.
2. Write all the users of your system.
3. Write and analyze the task of the users of your system.

I. Assessment-Rubrics

Practical 4: Gather application specific requirements- Requirement gathering.			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.5: Prepare broad SRS (software requirement software) for the above selected project.

A. **Objective:** An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods

3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs

4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements

5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze functional and non-functional requirements and prepare SRS document for the project (system).

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Prepare broad SRS (software requirement software) for the above selected project.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Basic concept of SRS

- SRS is the output of requirement gathering and analysis activity which is created by system analyst.

- SRS is a detailed description of the software that is to be developed. It describes the complete behaviour the system.
- **SRS describes '*what'* the proposed system should do without describing '*how'* the software will do (what part, not how).**
- The SRS document is known as black-box specification, because:
    - ➢ In SRS, internal details of the system are not known (as SRS doesn't specify how the system will work).
    - ➢ Only its visible external (i.e. input/output) behaviour is documented.

Benefits of SRS (Features of SRS)

- It works as an input to the design phase.
- It enhances communication between customer and developer because user requirements are expressed in natural language.
- Developers can get the idea what exactly the customer wants.
- Format of forms and rough screen prints can also be represented in SRS.
- As it is working as an agreement between user and developer, we can get the partial satisfaction of the end user for the final product.

Contents of the SRS document

- An SRS should clearly document the following three things:

i. Functional requirements of the system

ii. Non-functional requirements of the system

iii. Constraint (restriction) on the system

Characteristics of a good SRS: It should be concise, complete, consistent, structured, conceptual integrity, black box view, verifiable, adaptable, maintainable, portable, unambiguous and traceable.

As we have seen that, there are main three contents of SRS, but here in practical purpose, we will concentrate only on 'Functional requirements' of the system, and we will ignore the 'non-functional requirements and constraints'.

➢ **Example of SRS (functional requirements): operations at ATM**

Different functional requirements (likely) of ATM system are listed below:

- o Withdraw cash
- o Deposit cash
- o Check balance
- o Link aadhaar card
- o Print mini statement
- o Change pin number Etc.

For example, in ATM system, here we are considering "**withdraw cash**" as requirement. Now let's write step by step process for that requirement.

R1: withdraw cash

Description: this function first determines the type of operation that user wants to do (i.e. withdraw), then determines the account type and amount that the user has for his transaction. It checks the balance to determine whether the requested amount is available in the account. If yes then it outputs the required cash else it generates an error message.

R 1.1: enter the card

Input: ATM card

Output: user prompted to enter PIN showing the display with various operations

R 1.2: enter PIN

Input: valid PIN

Output: showing the display with various operations

R 1.3: select operation type

Input: select proper option (withdraw amount option)

Output: user prompted to enter the account type

R 1.4: select account type

Input: user option (for example: saving account or current account)

Output: user prompted to enter amount

R 1.5: get required amount

Input: amount to be withdrawn

Output: requested cash and printed transaction

H. Practical related Quiz.

1. List out all the functional requirements of the system.
2. List some non-functional requirements of your system and analyze them in brief. (Here you have to write some important non-functional requirements and for each of them you have to write one to two statements how your system concerned with them)

I. Assessment-Rubrics

Practical 5: Prepare broad SRS (software requirement software) for the above selected project**.**			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.6: Develop data designs using DFDs (data flow diagram) and E-R (entity-relationship) diagram**.**

A. **Objective:** The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The purpose of E-R diagram are the database analyst/designer gains a better understanding of the information to be contained in the database through the process of constructing the E-R Diagram. The E-R Diagram serves as a documentation tool. Finally, The E-R Diagram is used to communicate the logical Structure of the database to users. In particular the logic of the database to users.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze the system and prepare DFD
2. Analyze the system and prepare ER diagram.

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Develop data designs using DFDs (data flow diagram) and E-R (entity-relationship) diagram**.**

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

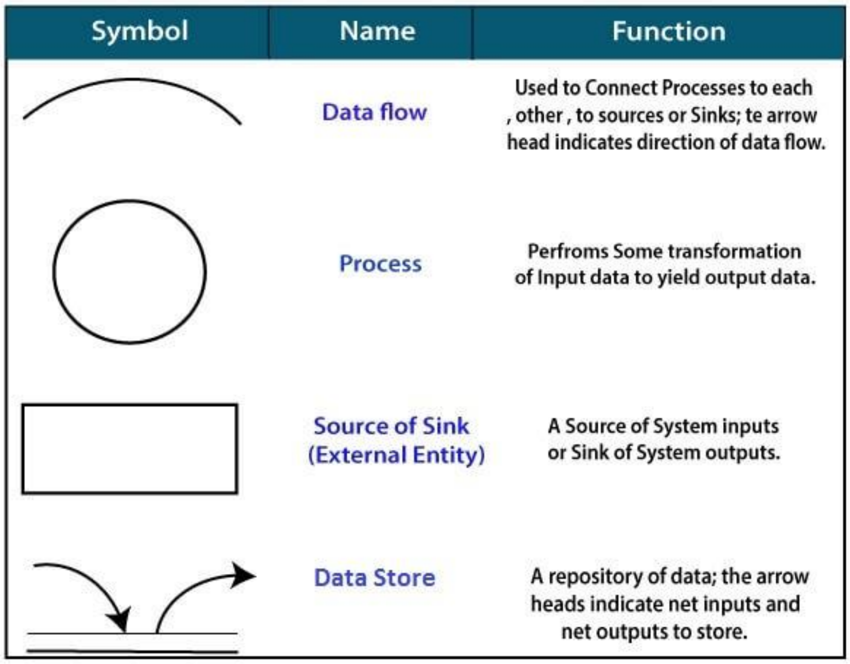
It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0-level DFDM

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x_1 and x_2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

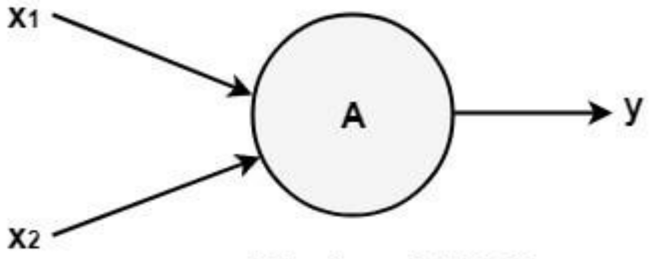


Fig: Level-0 DFD.

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.

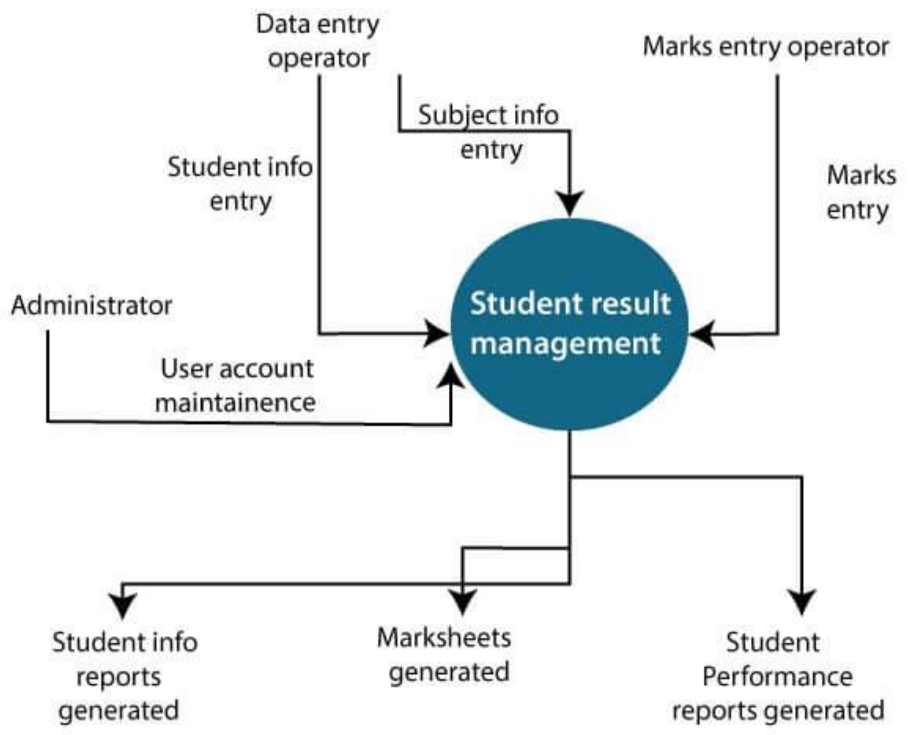


Fig: Level-0 DFD of result management system

1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

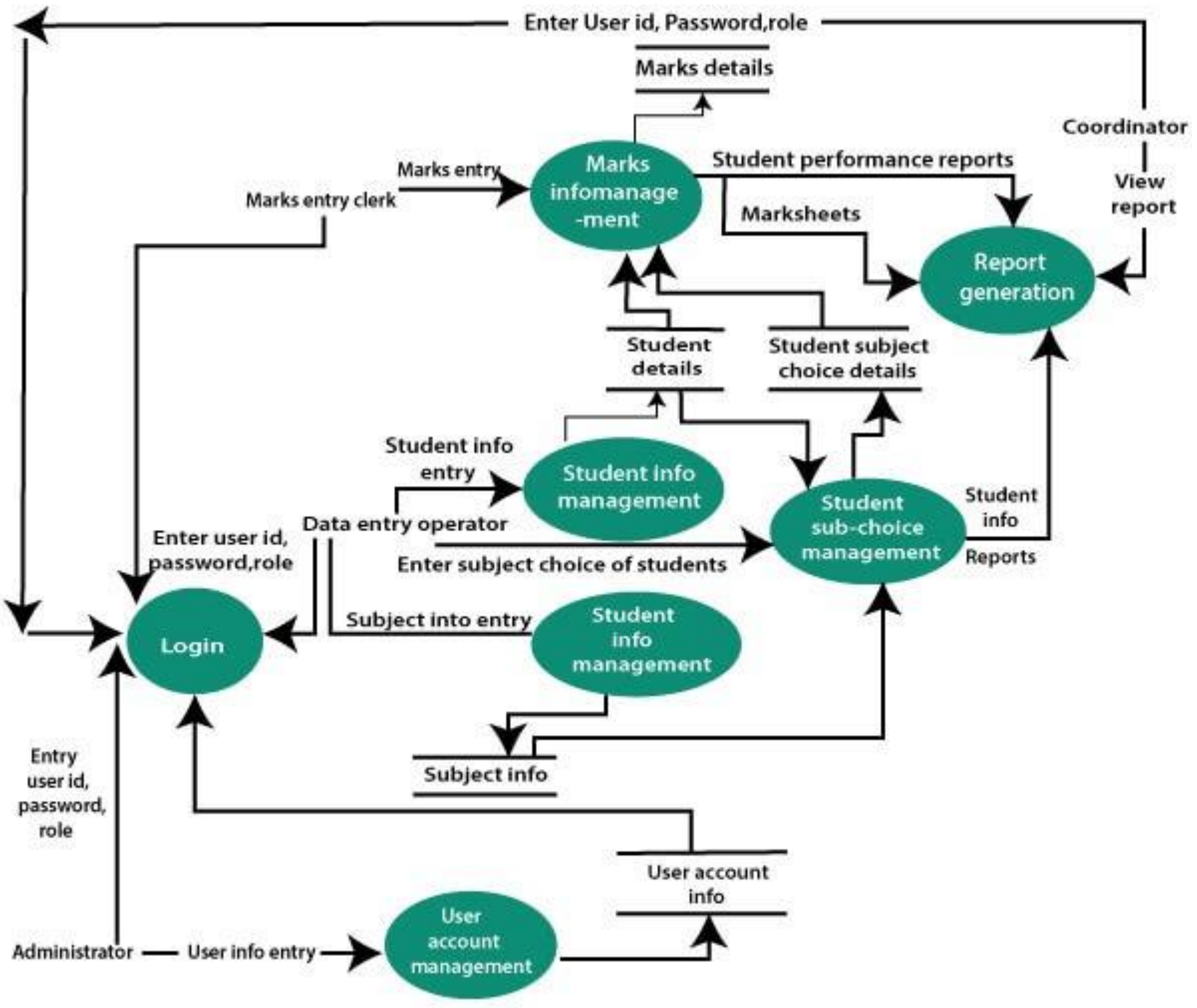
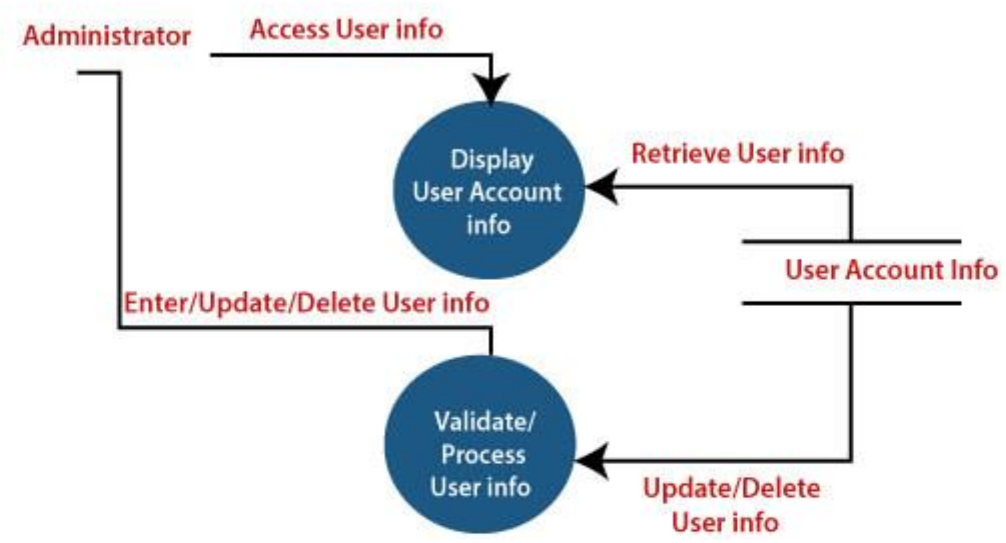


Fig: Level-1 DFD of result management system

2-Level DFD

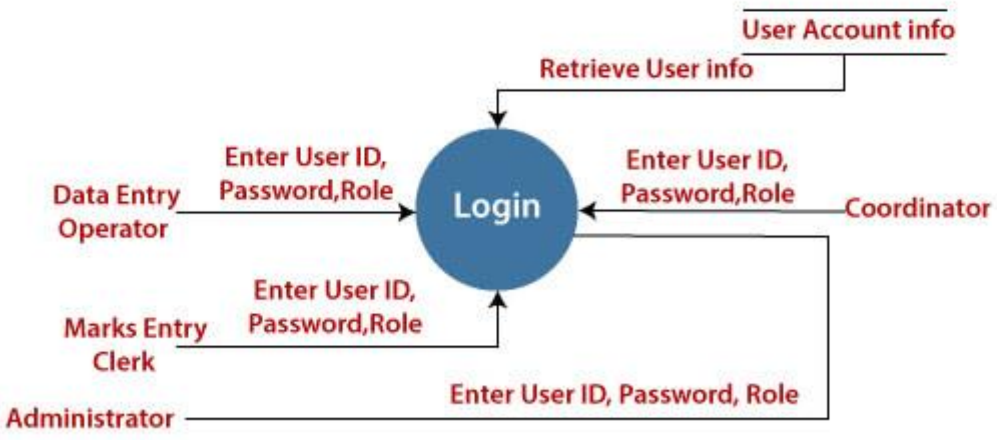
2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.

1.User Account Maintenance

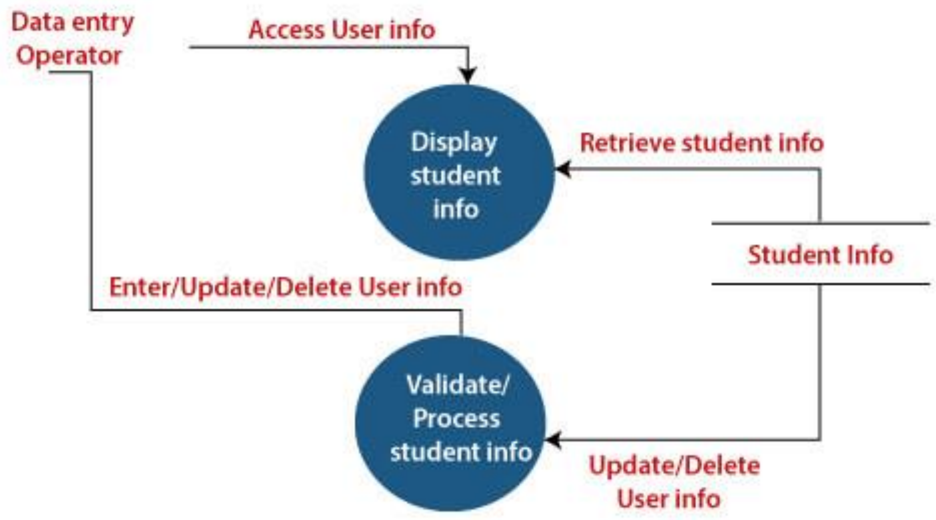


2. Login

The level 2 DFD of this process is given below:

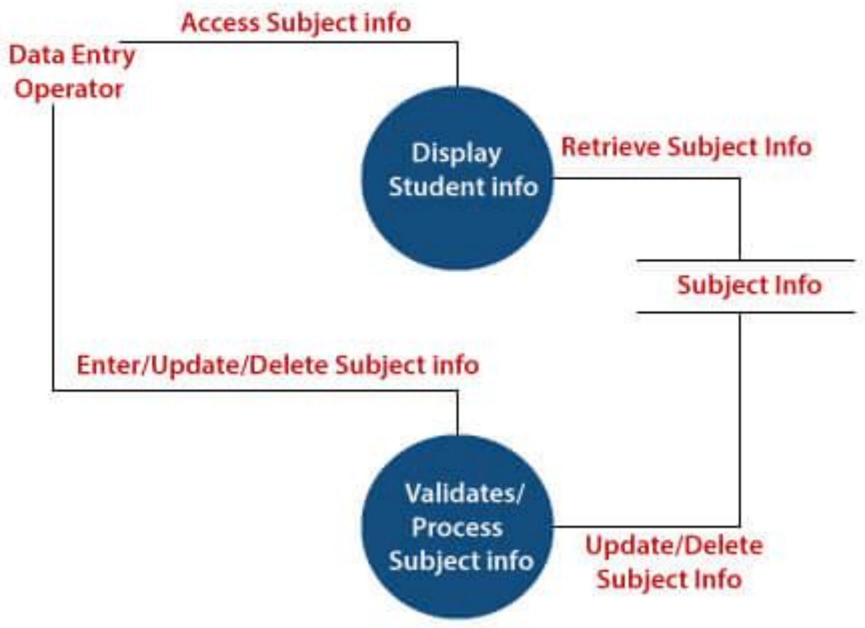

User Account info
Retrieve User info
Enter User ID, Password,Role
Data Entry Operator
Login
Enter User ID, Password,Role
Coordinator
Enter User ID, Password,Role
Marks Entry Clerk
Enter User ID, Password, Role
Administrator

3. Student Information Management



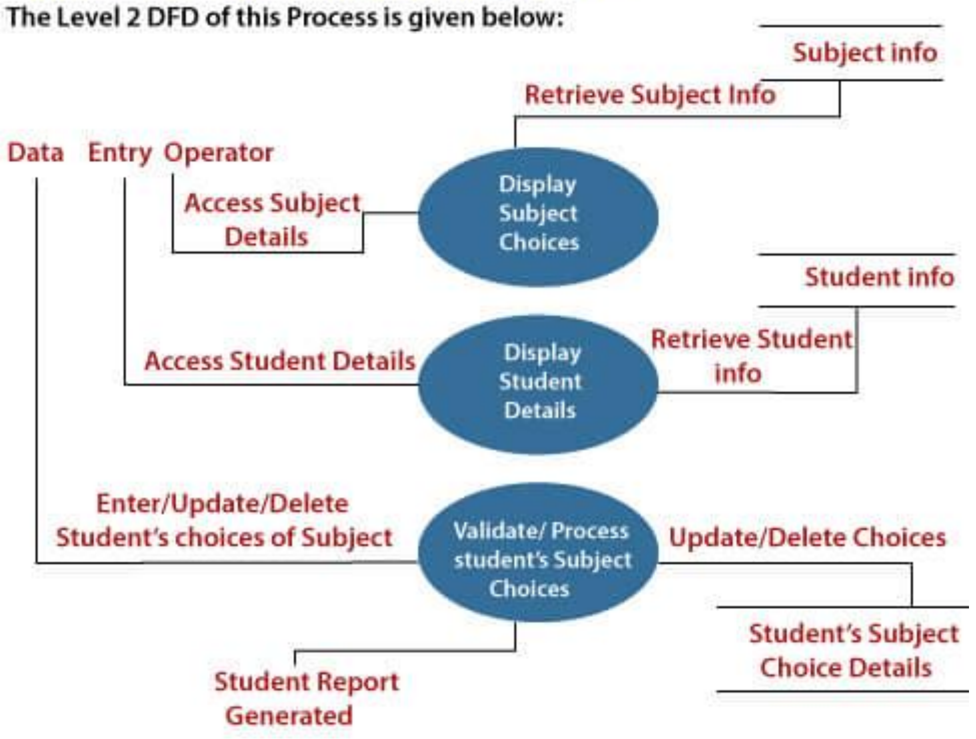
4. Subject Information Management

The level 2 DFD of this process is given below:



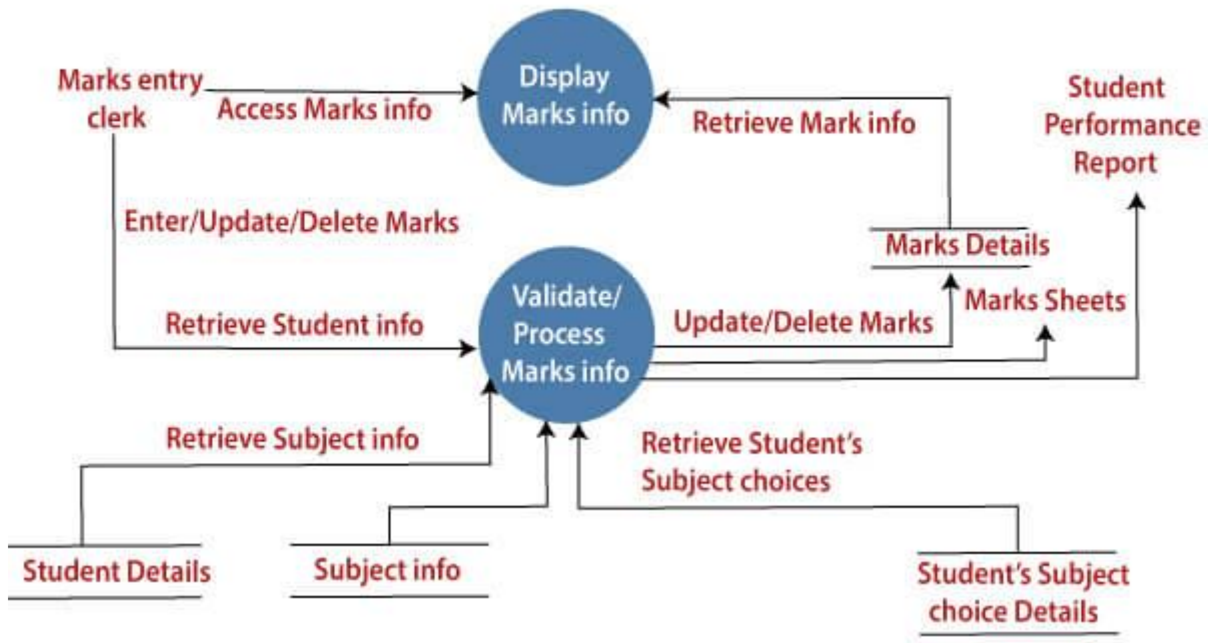
5. Student's Subject Choice Management

The Level 2 DFD of this Process is given below:



6. Marks Information Managment

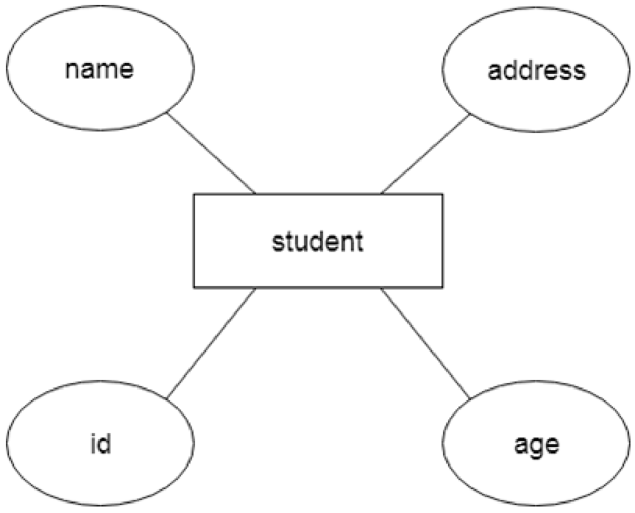
The Level 2 DFD of this Process is given below:



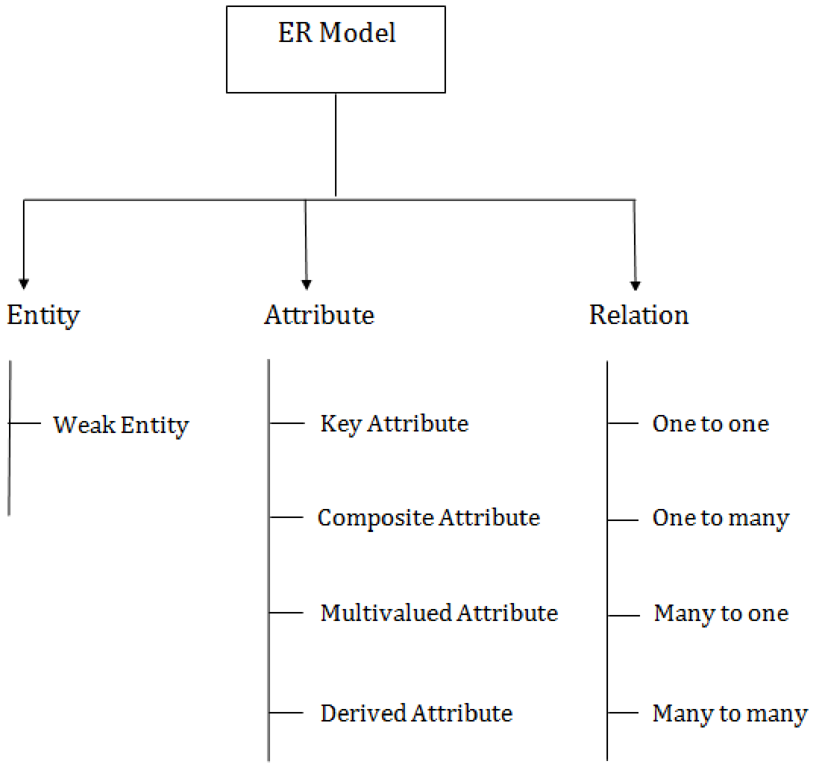
ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



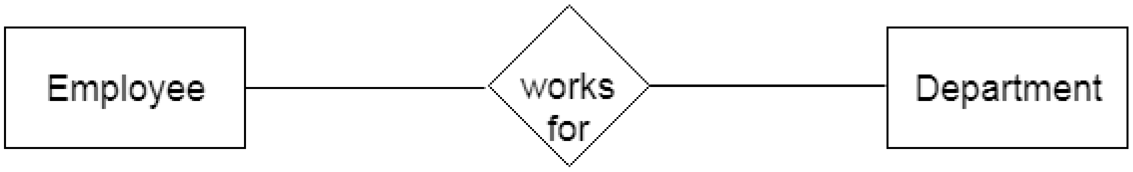
Component of ER Diagram



1. Entity:

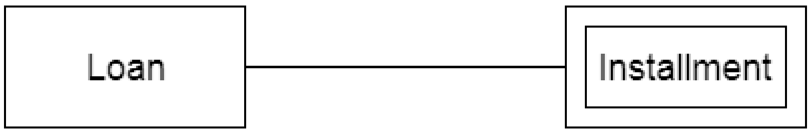
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

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



a. Weak Entity

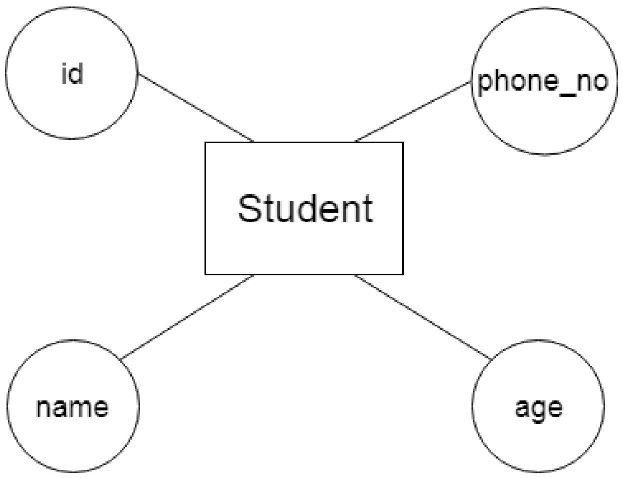
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

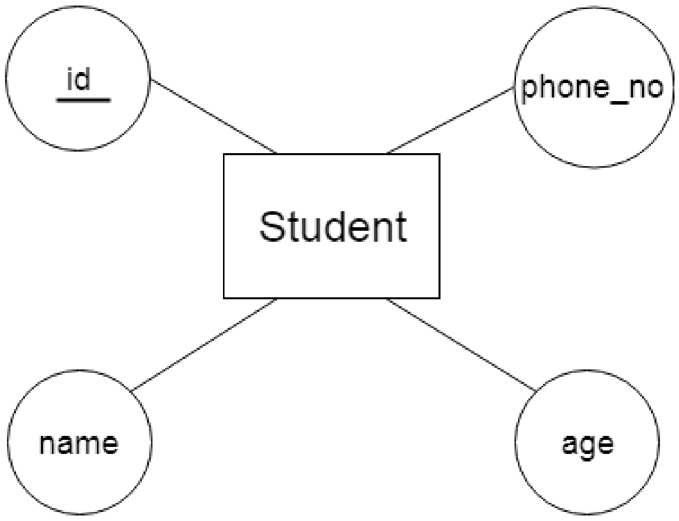
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

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



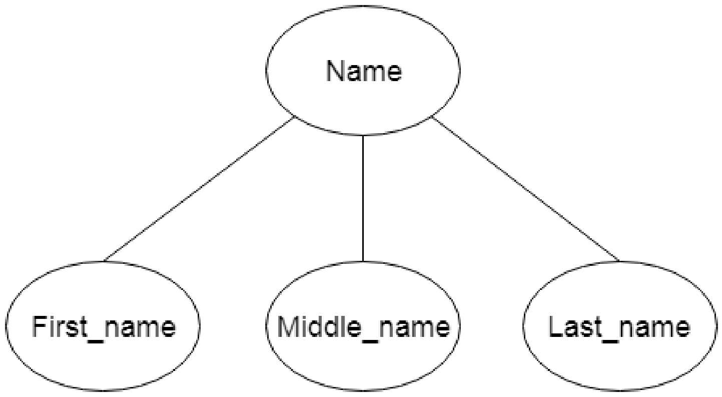
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

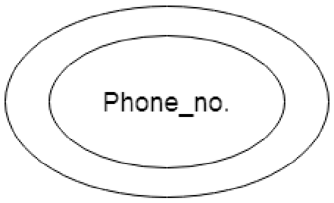
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. 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.

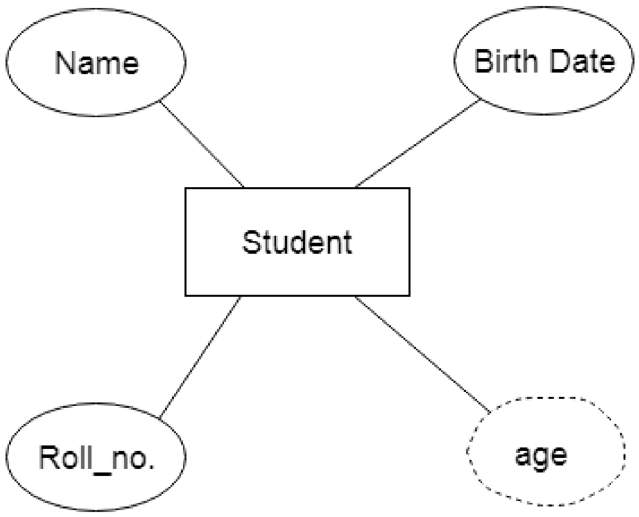
For example, a student can have more than one phone number.



d. Derived Attribute

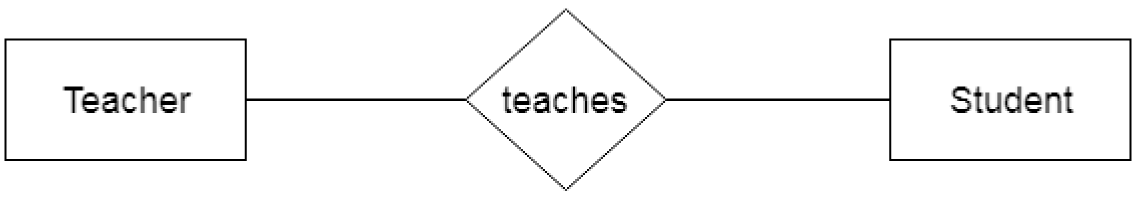
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

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



3. Relationship

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

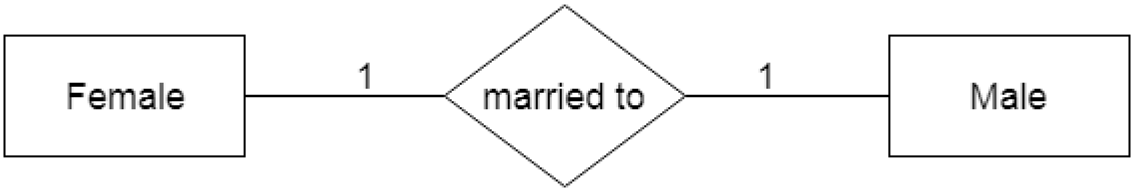


Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

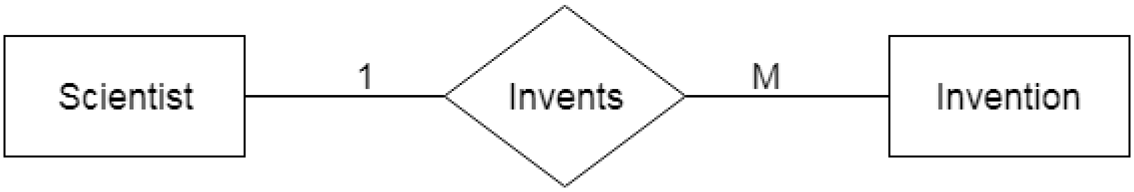
For example, A female can marry to one male, and a male can marry to one female.



b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

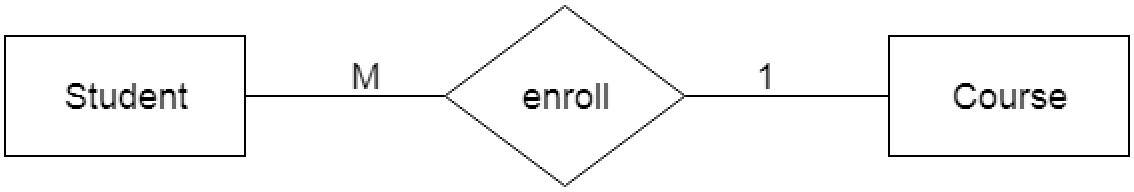
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

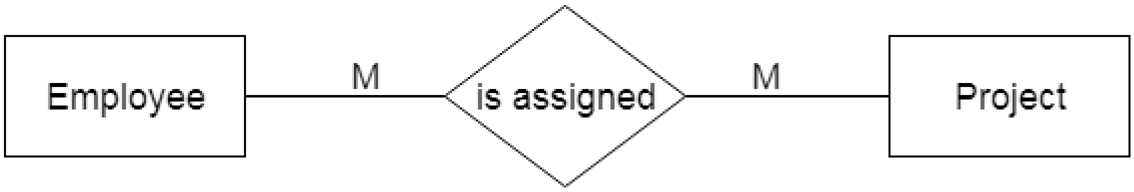
For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



H. Practical Related Quiz:

1. Prepare DFD diagram for above selected system.
2. Prepare ER Diagram for above selected system.

I. Assessment-Rubrics

Practical 6: Develop data designs using DFDs (data flow diagram) and E-R (entity-relationship) diagram.			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.7: Prepare use-cases and draw use case diagram

A. **Objective:** The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze the system and prepare use case diagram for it.

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Prepare use-cases and draw use case diagram

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

UML Use Case Diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purpose of Use Case Diagrams

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

How to draw a Use Case diagram?

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

After that, we will enlist the actors that will interact with the system. The actors are the person or a thing that invokes the functionality of a system. It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected. It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

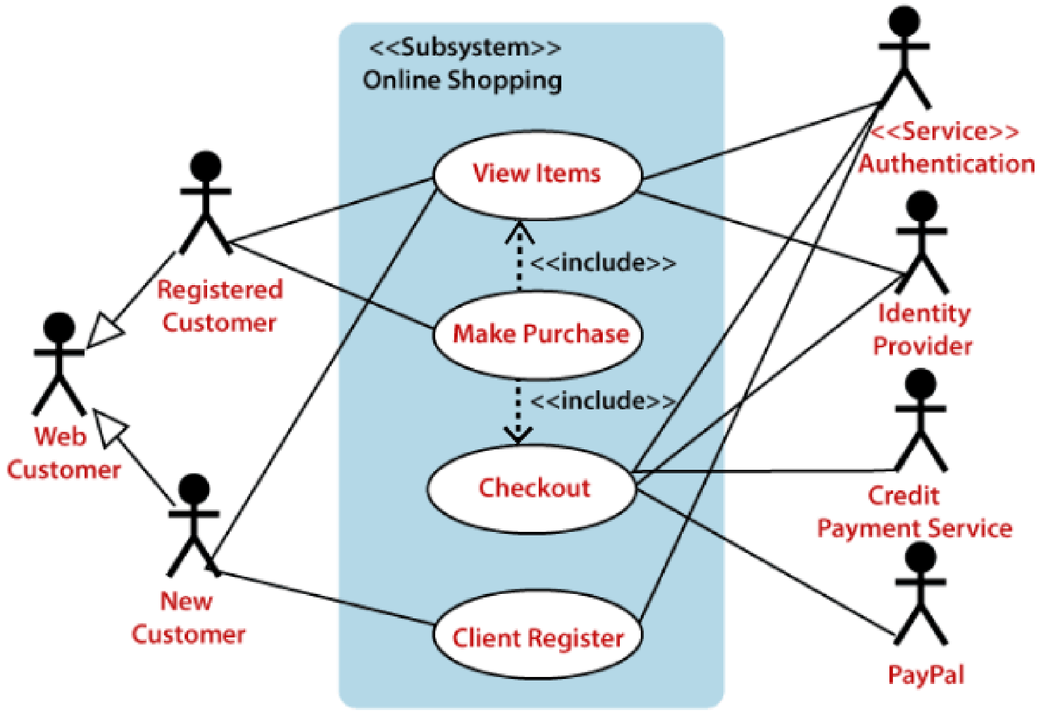
Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

Example of a Use Case Diagram

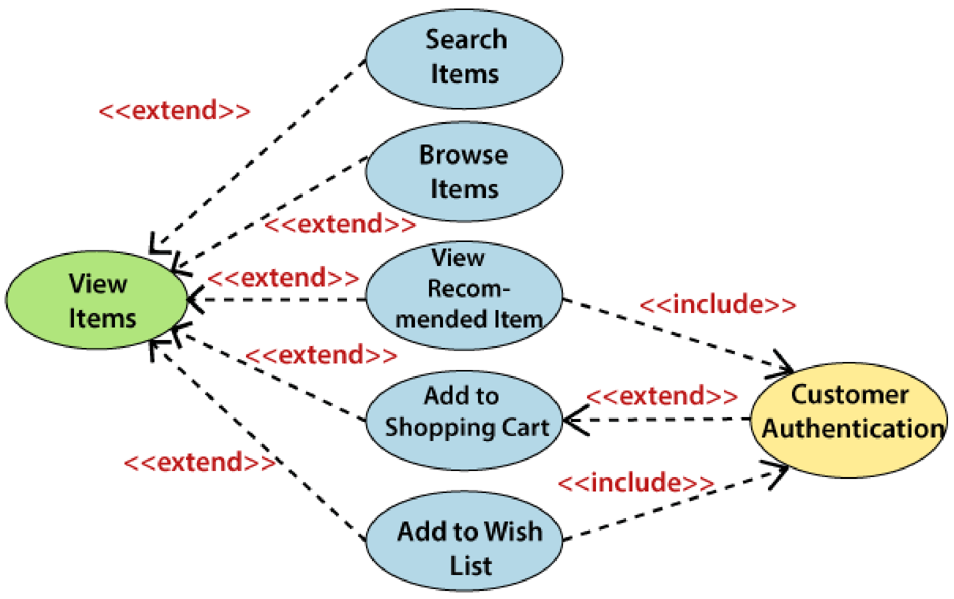
A use case diagram depicting the Online Shopping website is given below.

Here the Web Customer actor makes use of any online shopping website to purchase online. The top-level uses are as follows; View Items, Make Purchase, Checkout, Client Register. The **View Items** use case is utilized by the customer who searches and view products. The **Client Register** use case allows the customer to register itself with the website for availing gift vouchers, coupons, or getting a private sale invitation. It is to be noted that the **Checkout** is an included use case, which is part of **Making Purchase,** and it is not available by itself.



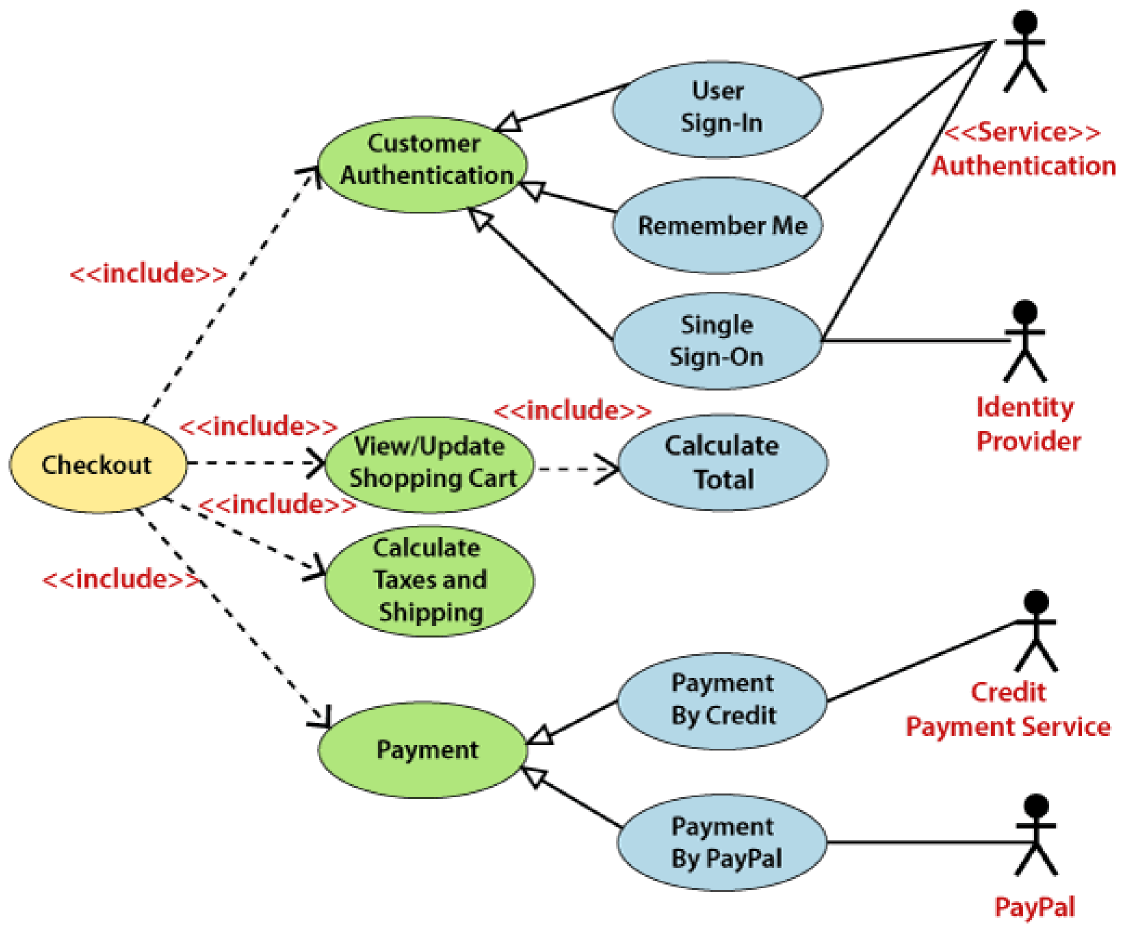
The **View Items** is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these extended use cases provide some functions to customers, which allows them to search for an item. The View Items is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these extended use cases provide some functions to customers, which allows them to search for an item.

Both **View Recommended Item** and **Add to Wish List** include the Customer Authentication use case, as they necessitate authenticated customers, and simultaneously item can be added to the shopping cart without any user authentication.



Similarly, the **Checkout** use case also includes the following use cases, as shown below. It requires an authenticated Web Customer, which can be done by login page, user authentication cookie ("Remember me"), or Single Sign-On (SSO). SSO needs an external identity provider's participation, while Web site authentication service is utilized in all these use cases.

The Checkout use case involves Payment use case that can be done either by the credit card and external credit payment services or with PayPal.



Important tips for drawing a Use Case diagram

Following are some important tips that are to be kept in mind while drawing a use case diagram:

1. A simple and complete use case diagram should be articulated.
2. A use case diagram should represent the most significant interaction among the multiple interactions.
3. At least one module of a system should be represented by the use case diagram.
4. If the use case diagram is large and more complex, then it should be drawn more generalized.

H. Practical Related Quiz:

1. Prepare USE case diagram for above selected system.

I. Assessment-Rubrics

Practical 7: Prepare use-cases and draw use case diagram			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.8: Develop a class diagram for selected project

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

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze the system and prepare class diagram of it.

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Develop class diagram for the selected project.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

UML Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

1. It analyses and designs a static view of an application.
2. It describes the major responsibilities of a system.
3. It is a base for component and deployment diagrams.
4. It incorporates forward and reverse engineering.

Benefits of Class Diagrams

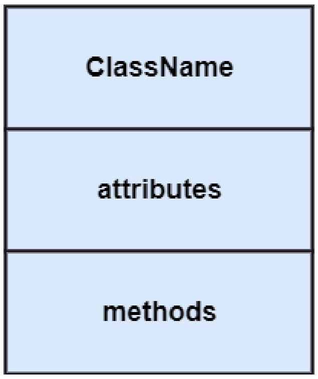
1. It can represent the object model for complex systems.
2. It reduces the maintenance time by providing an overview of how an application is structured before coding.
3. It provides a general schematic of an application for better understanding.
4. It represents a detailed chart by highlighting the desired code, which is to be programmed.
5. It is helpful for the stakeholders and the developers.

Vital components of a Class Diagram

The class diagram is made up of three sections:

- **Upper Section:** The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships, attributes, operations, and semantics. Some of the following rules that should be taken into account while representing a class are given below:
    a. Capitalize the initial letter of the class name.
    b. Place the class name in the center of the upper section.
    c. A class name must be written in bold format.
    d. The name of the abstract class should be written in italics format.
- **Middle Section:** The middle section constitutes the attributes, which describe the quality of the class. The attributes have the following characteristics:

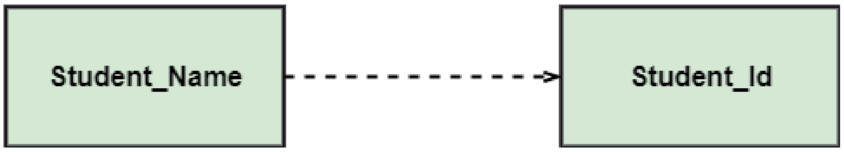
. The attributes are written along with its visibility factors, which are public (+), private (-), protected (#), and package (~).
    a. The accessibility of an attribute class is illustrated by the visibility factors.
    b. A meaningful name should be assigned to the attribute, which will explain its usage inside the class.
- **Lower Section:** The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.



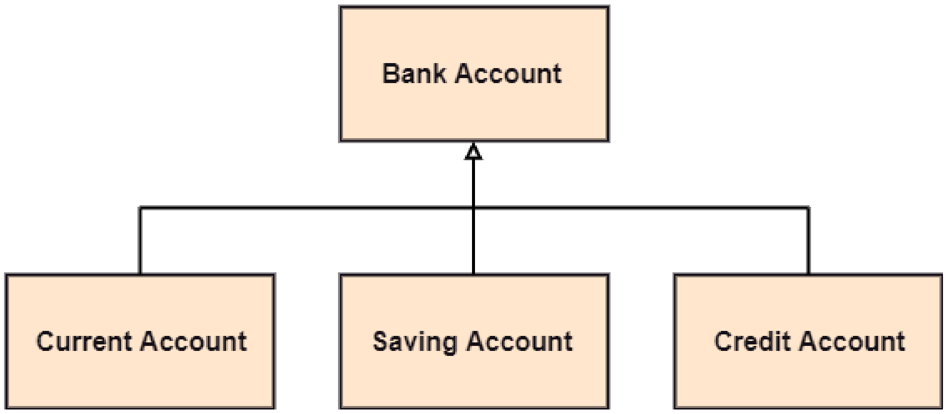
Relationships

In UML, relationships are of three types:

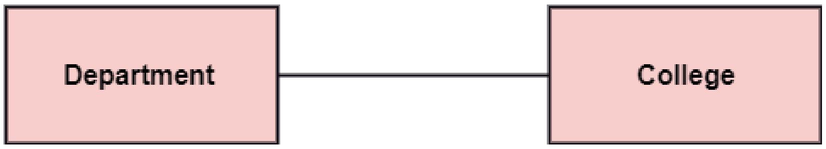
- **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship. In the following example, Student_Name is dependent on the Student_Id.



- **Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class. For example, The Current Account, Saving Account, and Credit Account are the generalized form of Bank Account.

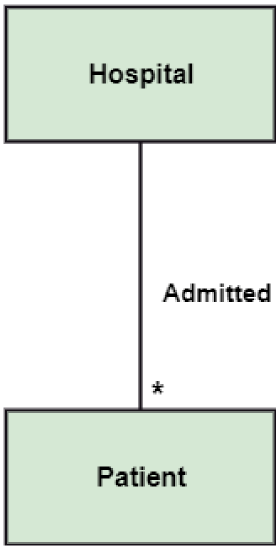


- **Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship. For example, a department is associated with the college.



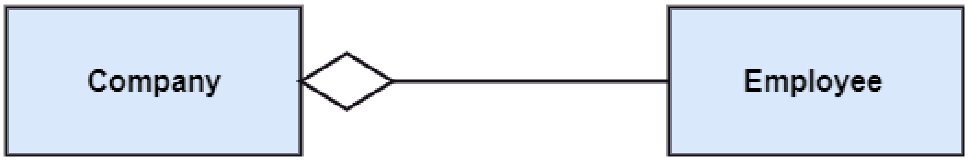
Multiplicity: It defines a specific range of allowable instances of attributes. In case if a range is not specified, one is considered as a default multiplicity.

For example, multiple patients are admitted to one hospital.



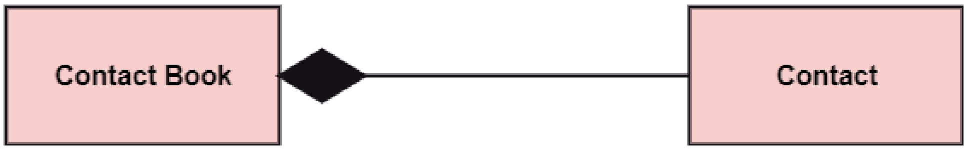
Aggregation: An aggregation is a subset of association, which represents has a relationship. It is more specific then association. It defines a part-whole or part-of relationship. In this kind of relationship, the child class can exist independently of its parent class.

The company encompasses a number of employees, and even if one employee resigns, the company still exists.



Composition: The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.

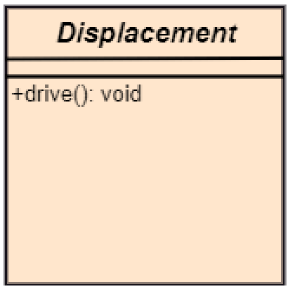
A contact book consists of multiple contacts, and if you delete the contact book, all the contacts will be lost.



Abstract Classes

In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects.

Let us assume that we have an abstract class named **displacement** with a method declared inside it, and that method will be called as a **drive ()**. Now, this abstract class method can be implemented by any object, for example, car, bike, scooter, cycle, etc.



How to draw a Class Diagram?

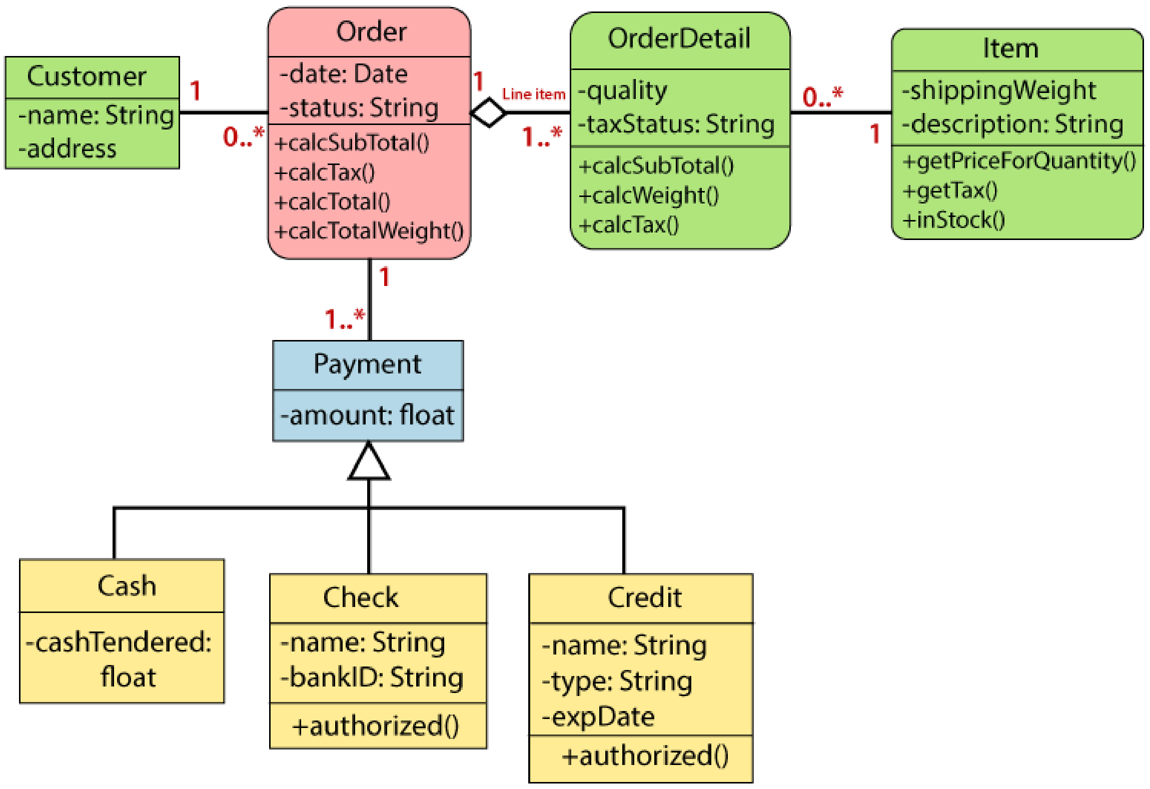
The class diagram is used most widely to construct software applications. It not only represents a static view of the system but also all the major aspects of an application. A collection of class diagrams as a whole represents a system.

Some key points that are needed to keep in mind while drawing a class diagram are given below:

1. To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.
2. The objects and their relationships should be acknowledged in advance.
3. The attributes and methods (responsibilities) of each class must be known.
4. A minimum number of desired properties should be specified as more number of the unwanted property will lead to a complex diagram.
5. Notes can be used as and when required by the developer to describe the aspects of a diagram.
6. The diagrams should be redrawn and reworked as many times to make it correct before producing its final version.

Class Diagram Example

A class diagram describing the sales order system is given below.



Usage of Class diagrams

The class diagram is used to represent a static view of the system. It plays an essential role in the establishment of the component and deployment diagrams. It helps to construct an executable code to perform forward and backward engineering for any system, or we can say it is mainly used for construction. It represents the mapping with object-oriented languages that are C++, Java, etc. Class diagrams can be used for the following purposes:

1. To describe the static view of a system.
2. To show the collaboration among every instance in the static view.
3. To describe the functionalities performed by the system.
4. To construct the software application using object-oriented languages.

H. Practical Related Quiz:

1. Develop a class diagram for selected project

I. Assessment-Rubrics

Practical 8: Develop a class diagram for selected project			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.9: Develop Sequence diagram for selected project.

A. **Objective:** The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze the system and prepare sequence diagram of it.

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Develop Sequence diagram for selected project.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Sequence Diagram

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

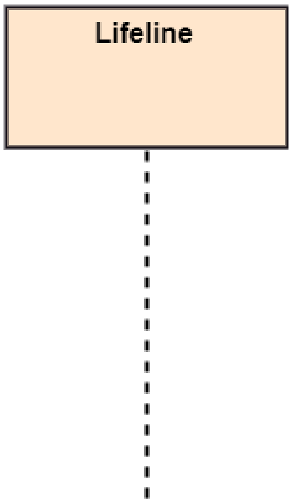
Purpose of a Sequence Diagram

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

Notations of a Sequence Diagram

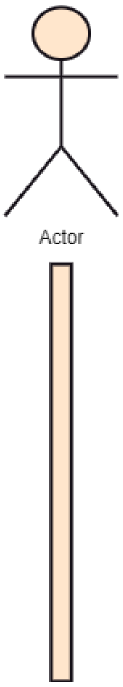
Lifeline

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



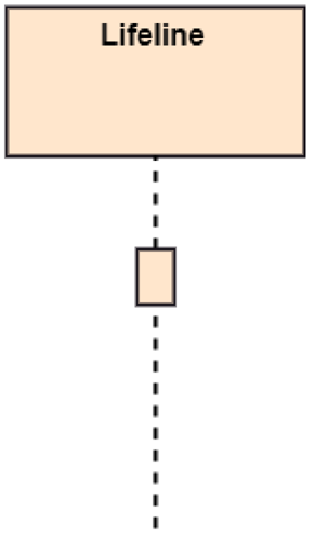
Actor

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



Activation

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

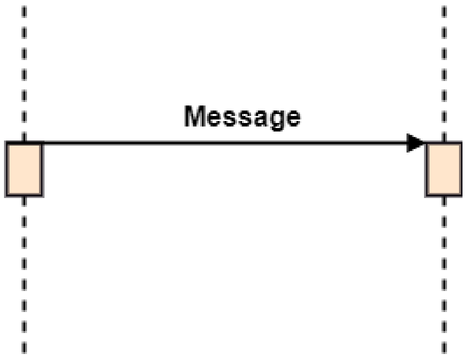


Messages

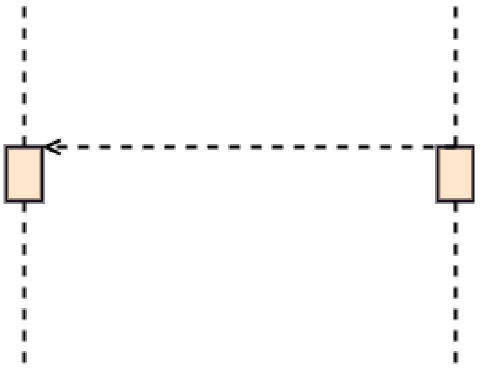
The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

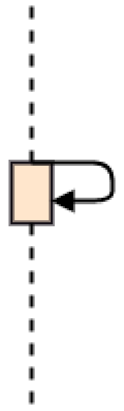
- **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.



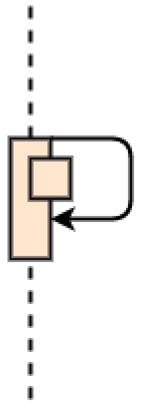
- **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.



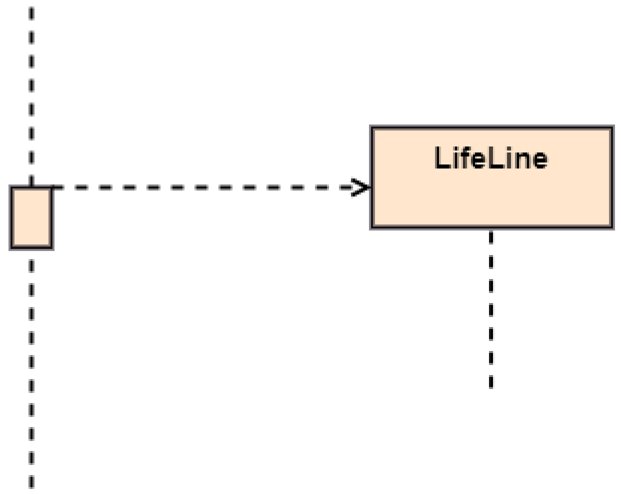
- **Self Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.



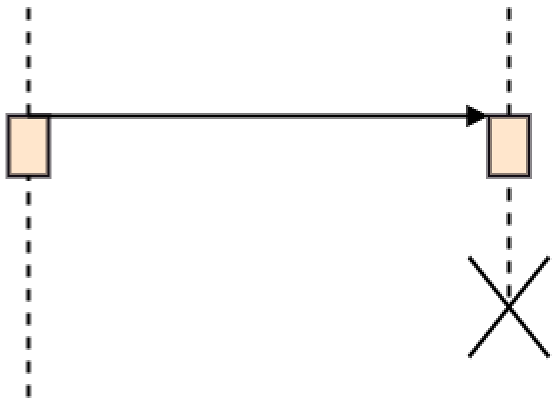
- **Recursive Message:** A self message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self message as it represents the recursive calls.



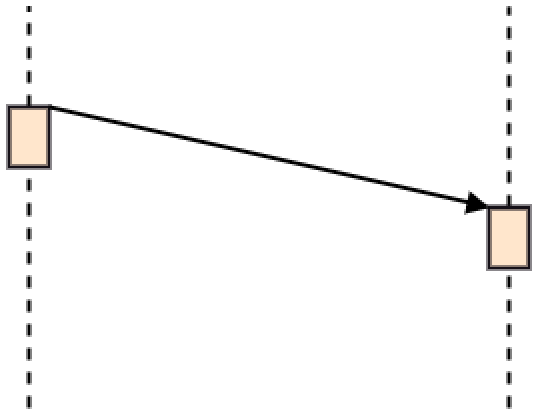
- **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.



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

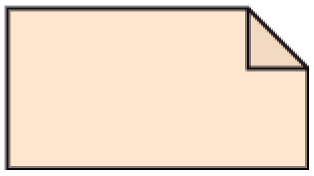


- **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modeling a system.



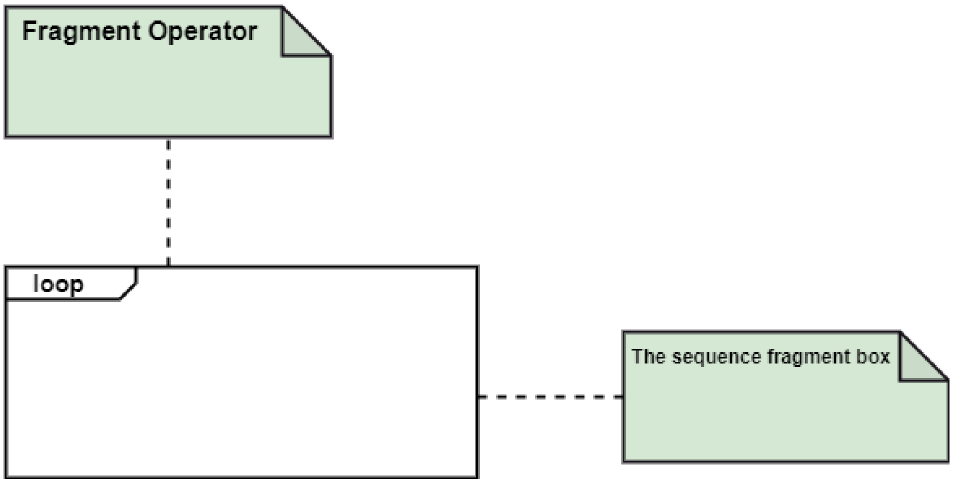
Note

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



Sequence Fragments

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



Types of fragments

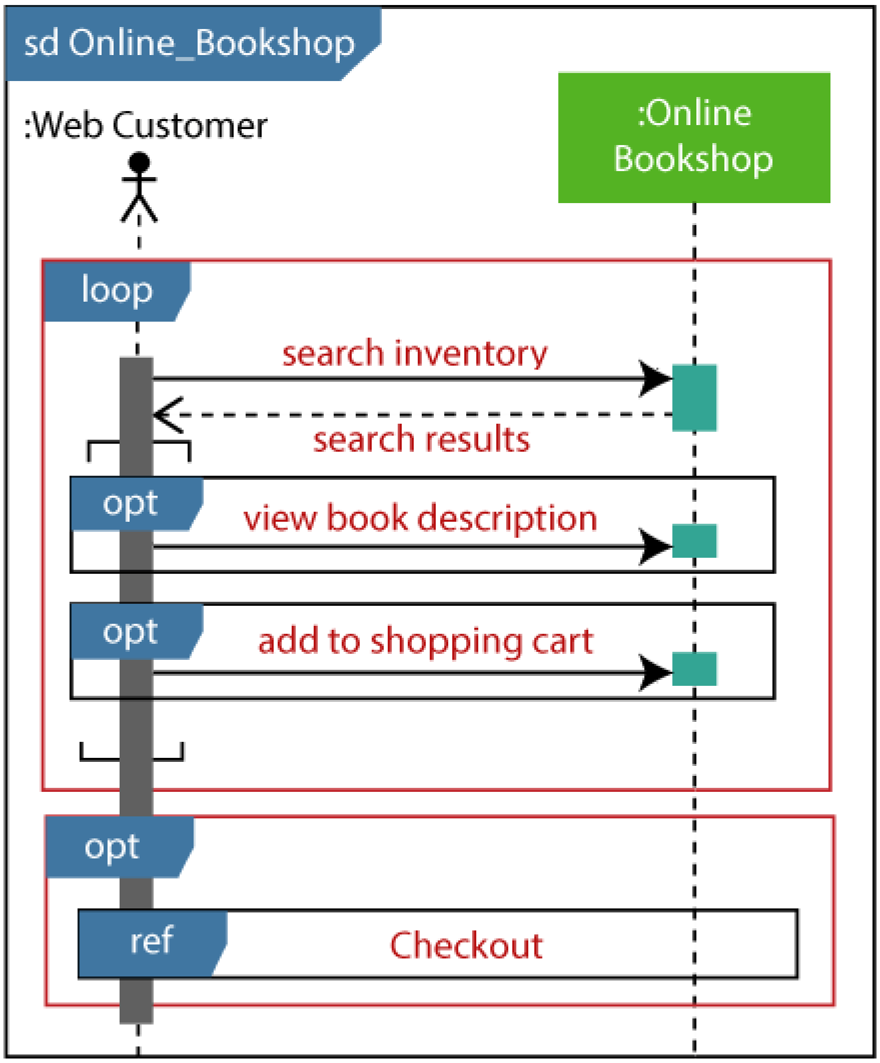
Following are the types of fragments enlisted below;

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

Example of a Sequence Diagram

An example of a high-level sequence diagram for online bookshop is given below.

Any online customer can search for a book catalog, view a description of a particular book, add a book to its shopping cart, and do checkout.



Benefits of a Sequence Diagram

1. It explores the real-time application.
2. It depicts the message flow between the different objects.
3. It has easy maintenance.
4. It is easy to generate.
5. Implement both forward and reverse engineering.
6. It can easily update as per the new change in the system.

The drawback of a Sequence Diagram

1. In the case of too many lifelines, the sequence diagram can get more complex.
2. The incorrect result may be produced, if the order of the flow of messages changes.
3. Since each sequence needs distinct notations for its representation, it may make the diagram more complex.
4. The type of sequence is decided by the type of message.

H. Practical Related Quiz:

1. Develop a sequence diagram for selected project

I. Assessment-Rubrics

Practical 9: Develop a sequence diagram for selected project			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.10: Develop the activity diagram to represent flow from one activity to another for software development.

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

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Analyze the system and develop the activity diagram of it.

D. **Expected Course Outcomes(COs)**

1. Prepare software analysis and design using SRS, DFD and object oriented UML diagrams.

E. **Practical Outcome(Pros)**

1. Develop the activity diagram to represent flow from one activity to another for software development.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

UML Activity Diagram

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

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

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

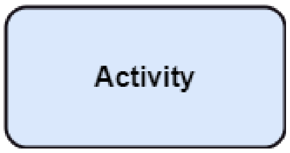
Components of an Activity Diagram

Following are the component of an activity diagram:

Activities

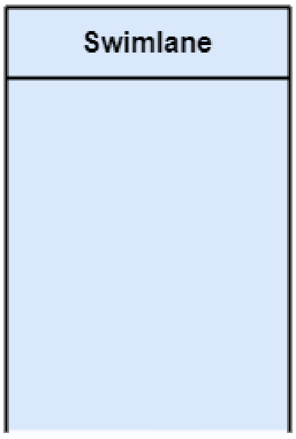
The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.

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



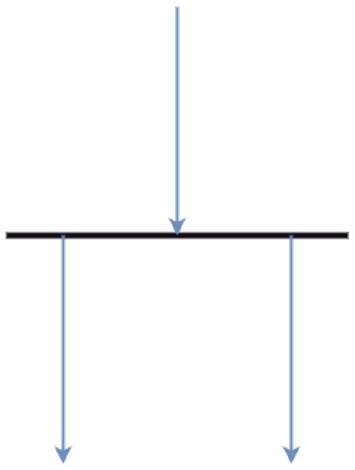
Activity partition /swimlane

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



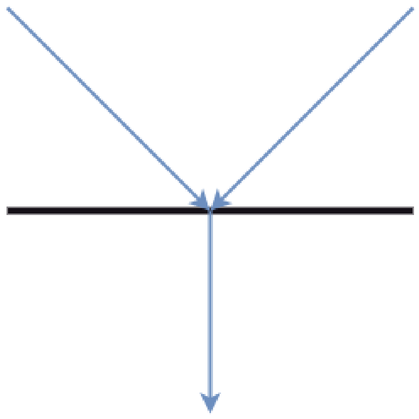
Forks

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



Join Nodes

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



Pins

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

Notation of an Activity diagram

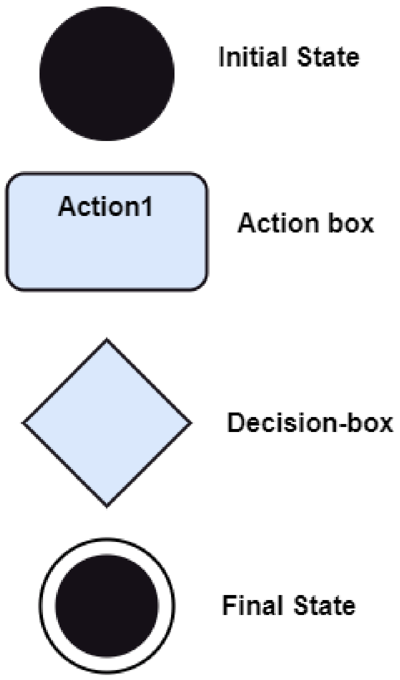
Activity diagram constitutes following notations:

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

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

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

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



Why use Activity Diagram?

An event is created as an activity diagram encompassing a group of nodes associated with edges. To model the behavior of activities, they can be attached to any modeling element. It can model use cases, classes, interfaces, components, and collaborations.

It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering. It does not include the message part, which means message flow is not represented in an activity diagram.

It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities.

How to draw an Activity Diagram?

An activity diagram is a flowchart of activities, as it represents the workflow among various activities. They are identical to the flowcharts, but they themself are not exactly the flowchart. In other words, it can be said that an activity diagram is an enhancement of the flowchart, which encompasses several unique skills.

Since it incorporates swimlanes, branching, parallel flows, join nodes, control nodes, and forks, it supports exception handling. A system must be explored as a whole before drawing an activity diagram to provide a clearer view of the user. All of the activities are explored after they are properly analyzed for finding out the constraints applied to the activities. Each and every activity, condition, and association must be recognized.

After gathering all the essential information, an abstract or a prototype is built, which is then transformed into the actual diagram.

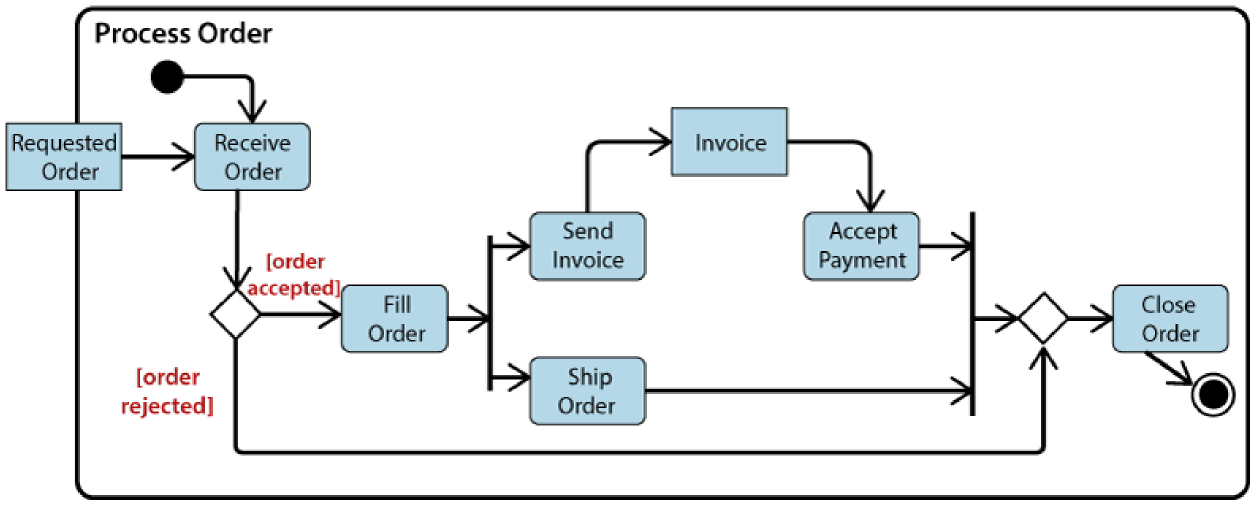
Following are the rules that are to be followed for drawing an activity diagram:

1. A meaningful name should be given to each and every activity.
2. Identify all of the constraints.
3. Acknowledge the activity associations.

Example of an Activity Diagram

An example of an activity diagram showing the business flow activity of order processing is given below.

Here the input parameter is the Requested order, and once the order is accepted, all of the required information is then filled, payment is also accepted, and then the order is shipped. It permits order shipment before an invoice is sent or payment is completed.



When to use an Activity Diagram?

An activity diagram can be used to portray business processes and workflows. Also, it used for modeling business as well as the software. An activity diagram is utilized for the followings:

1. To graphically model the workflow in an easier and understandable way.
2. To model the execution flow among several activities.
3. To model comprehensive information of a function or an algorithm employed within the system.
4. To model the business process and its workflow.
5. To envision the dynamic aspect of a system.
6. To generate the top-level flowcharts for representing the workflow of an application.
7. To represent a high-level view of a distributed or an object-oriented system.

H. Practical Related Quiz:

1. Develop the activity diagram to represent flow from one activity to another for software development.

I. Assessment-Rubrics

Practical 10: Develop the activity diagram to represent flow from one activity to another for software development.

Criteria	M	Rubrics	Marks
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.11: Evaluate size of the project using Function point metric for the assigned project.

A. **Objective:** The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Evaluate size of the project using Function point metric for the assigned project.

D. **Expected Course Outcomes(COs)**

1. Prepare software development plan using project scheduling.

E. **Practical Outcome(Pros)**

1. Evaluate size of the project using Function point metric for the assigned project.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. **Prerequisite Theory:**

Functional Point (FP) Analysis

FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product. However, functional point analysis may be used for the test estimation of the product. The functional size of the product is measured in terms of the function point, which is a standard of measurement to measure the software application.

The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request. Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

Following are the points regarding FPs

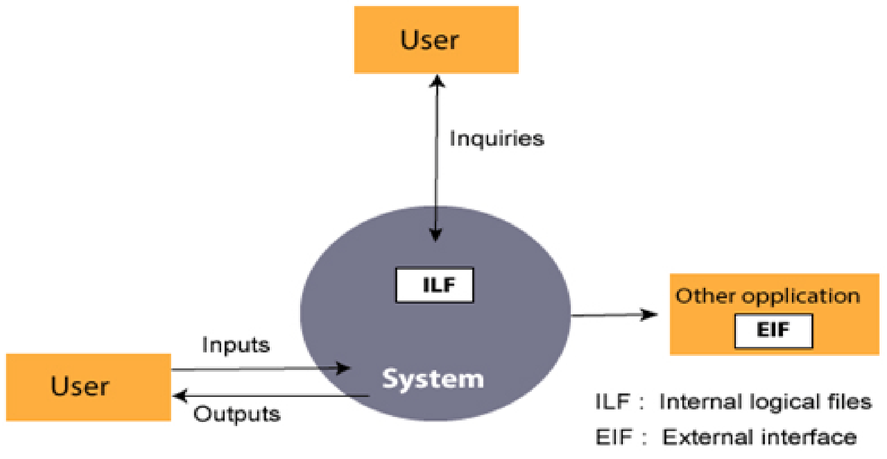
1. FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown in Table:

Types of FP Attributes

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

All these parameters are then individually assessed for complexity.

The FPA functional units are shown in Fig:



FPAs Functional Units System

2. FP characterizes the complexity of the software system and hence can be used to depict the project time and the manpower requirement.

3. The effort required to develop the project depends on what the software does.

4. FP is programming language independent.

5. FP method is used for data processing systems, business systems like information systems.

6. The five parameters mentioned above are also known as information domain characteristics.

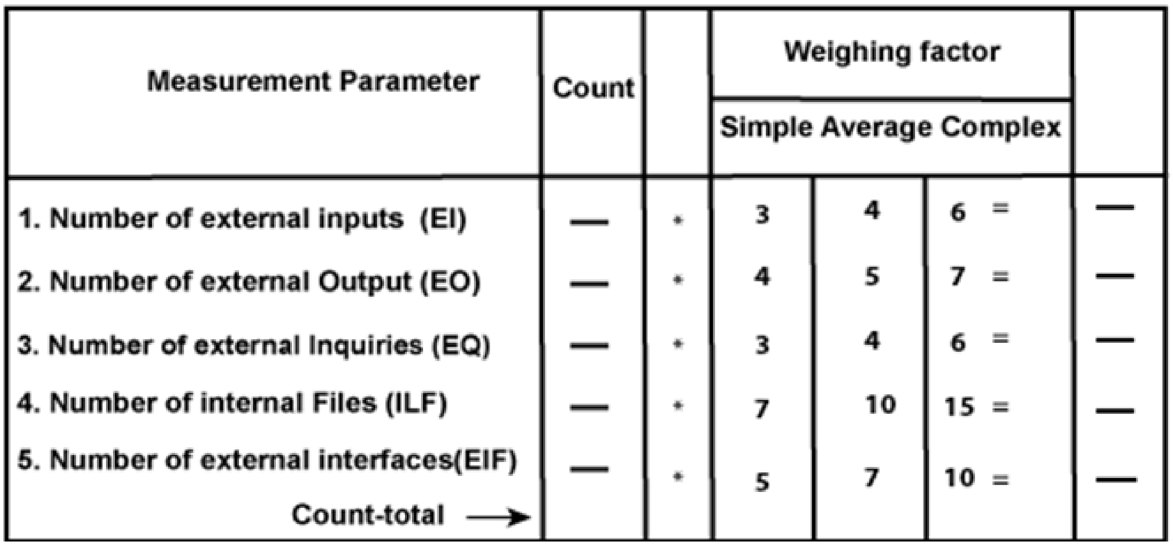
7. All the parameters mentioned above are assigned some weights that have been experimentally determined and are shown in Table

Weights of 5-FP Attributes

Measurement Parameter	Low	Average	High
1. Number of external inputs (EI)	7	10	15
2. Number of external outputs (EO)	5	7	10
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	4	5	7
5. Number of external interfaces (EIF)	3	4	6

The functional complexities are multiplied with the corresponding weights against each function, and the values are added up to determine the UFP (Unadjusted Function Point) of the subsystem.

Computing FPs



Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	—	*	3	4	6 =	—
2. Number of external Output (EO)	—	*	4	5	7 =	—
3. Number of external Inquiries (EQ)	—	*	3	4	6 =	—
4. Number of internal Files (ILF)	—	*	7	10	15 =	—
5. Number of external interfaces(EIF)	—	*	5	7	10 =	—
Count-total ⟶						

Here that weighing factor will be simple, average, or complex for a measurement parameter type.

The Function Point (FP) is thus calculated with the following formula.

$$\mathbf{FP = Count\text{-}total * [0.65 + 0.01 * \sum(f_i)]}$$
$$\mathbf{= Count\text{-}total * CAF}$$

where Count-total is obtained from the above Table.

$$\mathbf{CAF = [0.65 + 0.01 * \sum(f_i)]}$$

and **∑(f_i)** is the sum of all 14 questionnaires and show the complexity adjustment value/ factor-CAF (where i ranges from 1 to 14). Usually, a student is provided with the value of ∑(f_i)

Also note that **∑(f_i)** ranges from 0 to 70, i.e.,

$0 <= \sum(f_i) <=70$

and CAF ranges from 0.65 to 1.35 because

a. When **∑(f_i)** = 0 then CAF = 0.65

b. When **∑(f_i)** = 70 then CAF = 0.65 + (0.01 * 70) = 0.65 + 0.7 = 1.35

Based on the FP measure of software many other metrics can be computed:

a. Errors/FP

b. $/FP.

c. Defects/FP

d. Pages of documentation/FP

e. Errors/PM.

f. Productivity = FP/PM (effort is measured in person-months).

g. $/Page of Documentation.

8. LOCs of an application can be estimated from FPs. That is, they are interconvertible. **This process is known as backfiring**. For example, 1 FP is equal to about 100 lines of COBOL code.

9. FP metrics is used mostly for measuring the size of Management Information System (MIS) software.

10. But the function points obtained above are unadjusted function points (UFPs). These (UFPs) of a subsystem are further adjusted by considering some more General System Characteristics (GSCs). It is a set of 14 GSCs that need to be considered. The procedure for adjusting UFPs is as follows:

a. Degree of Influence (DI) for each of these 14 GSCs is assessed on a scale of 0 to 5. (b) If a particular GSC has no influence, then its weight is taken as 0 and if it has a strong influence then its weight is 5.

b. The score of all 14 GSCs is totaled to determine Total Degree of Influence (TDI).

c. Then Value Adjustment Factor (VAF) is computed from TDI by using the formula:

VAF = (TDI * 0.01) + 0.65

Remember that the value of VAF lies within 0.65 to 1.35 because

a. When TDI = 0, VAF = 0.65

b. When TDI = 70, VAF = 1.35

c. VAF is then multiplied with the UFP to get the final FP count: **FP = VAF * UFP**

Example: Compute the function point, productivity, documentation, cost per function for the following data:

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8

4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	**Count**		**Weighing factor**
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all f_i ($i \leftarrow 1$ to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

$$\begin{aligned} FP &= \text{Count-total} * [0.65 + 0.01 * \textstyle\sum(f_i)] \\ &= 378 * [0.65 + 0.01 * 43] \\ &= 378 * [0.65 + 0.43] \\ &= 378 * 1.08 = 408 \end{aligned}$$

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Differentiate between FP and LOC

FP	LOC
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP (backfiring)

H. Practical Related Quiz:

1. What is estimation techniques?
2. What are Function points in software engineering?
3. How many Information Domain Values are used for Function Point Computation?
4. Evaluate size of the project using Function point metric for the assigned project.

I. Assessment-Rubrics

Practical 11: Evaluate size of the project using Function point metric for the assigned project.			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.12: Estimate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach for the assigned project.

A. **Objective:** The purpose of COCOMO predicts the efforts and schedule of a software product based on the size of the software.

B. **Expected Program Outcomes (POs)**

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods

3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs

4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements

5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. **Expected skills to be developed based on competency**

This practical is expected to develop the following skills for the industry identified competency:

1. Apply basic and fundamental knowledge to calculate size using COCOMO technique

D. **Expected Course Outcomes(COs)**

1. Prepare software development plan using project scheduling.

E. **Practical Outcome(Pros)**

1. Estimate cost of the project using COCOMO/COCOMO II approach for the assigned project.

F. **Expected Affective domain Outcome(ADos)**

1. Work as a leader/ a team member
2. Follow ethical practice.

G. Prerequisite Theory:

Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KLOC).
2. Determine a set of 15 multiplying factors from various attributes of the project.
3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KLOC as the measure of the size. To determine the initial effort E_i in person-months the equation used is of the type is shown below

$$E_i=a*(KDLOC)b$$

The value of the constant a and b are depends on the project type.

In COCOMO, projects are categorized into three types:

1. Organic
2. Semidetached
3. Embedded

1.Organic: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects. **Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.**

2. Semidetached: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience in related systems but may be unfamiliar with some aspects of the order being developed. **Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.**

3. Embedded: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist.

For Example: ATM, Air Traffic control.

For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.

According to Boehm, software cost estimation should be done through three stages:

1. Basic Model
2. Intermediate Model
3. Detailed Model

1. Basic COCOMO Model: The basic COCOMO model provide an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

$$Effort=a_1*(KLOC)a_2 \text{ PM}$$
$$Tdev=b_1*(efforts)b_2 \text{ Months}$$

Where **KLOC** is the estimated size of the software product indicate in Kilo Lines of Code, a_1,a_2,b_1,b_2 are constants for each group of software products,

Organic: Effort = 2.4(KLOC) 1.05 PM

Semi-detached: Effort = 3.0(KLOC) 1.12 PM

Embedded: Effort = 3.6(KLOC) 1.20 PM

Estimation of development time

For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic: Tdev = 2.5(Effort) 0.38 Months

Semi-detached: Tdev = 2.5(Effort) 0.35 Months

Embedded: Tdev = 2.5(Effort) 0.32 Months

Some insight into the basic COCOMO model can be obtained by plotting the estimated characteristics for different software sizes. Fig shows a plot of estimated effort versus product size. From fig, we can observe that the effort is somewhat superliner in the size of the software product. Thus, the effort required to develop a product increases very rapidly with project size.

Example1: Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

Solution: The basic COCOMO equation takes the form:

Effort=a_1*(KLOC) a_2 PM
Tdev=b_1*(efforts)b_2 Months
Estimated Size of project= 400 KLOC

(i)Organic Mode

E = 2.4 * (400)1.05 = 1295.31 PM
D = 2.5 * (1295.31)0.38=38.07 PM

(ii)Semidetached Mode

E = 3.0 * (400)1.12=2462.79 PM
D = 2.5 * (2462.79)0.35=38.45 PM

(iii) Embedded Mode

E = 3.6 * (400)1.20 = 4772.81 PM
D = 2.5 * (4772.8)0.32 = 38 PM

H. Practical Related Quiz:

1. Explain COCOMO II Model.
2. What is cost estimation?
3. What is LOC in cost estimation?
4. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded of your selected system.

I. Assessment-Rubrics

Practical 12: Estimate cost of the project using COCOMO (Constructive Cost Model) / COCOMO II approach for the assigned project.

Criteria	M	Rubrics	Marks
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.13: Use flow chart and Gantt charts to track progress of the assigned project. (Use Sprint burn down chart, if agile model is selected).

A. Objective: The purpose of scheduling charts is used for project planning: it's a useful way of showing what work is scheduled to be done on specific days.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
2. **Problem analysis**: Identify and analyze well-defined engineering problems using codified standard methods
3. **Design/ development of solutions** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs
4. **Engineering Tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements
5. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
6. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes.

C. Expected skills to be developed based on competency

This practical is expected to develop the following skills for the industry identified competency:

1. Apply basic and fundamental knowledge of gantt chart, flow chart and Sprint burn down chart to track progress of the project.

D. Expected Course Outcomes(COs)

1. Prepare software development plan using project scheduling.

E. Practical Outcome(Pros)

1. Use flow chart and Gantt charts to track progress of the assigned project. (Use Sprint burn down chart, if agile model is selected).

F. Expected Affective domain Outcome(ADos)

1. Work as a leader/ a team member
2. Follow ethical practice.

G. Prerequisite Theory:

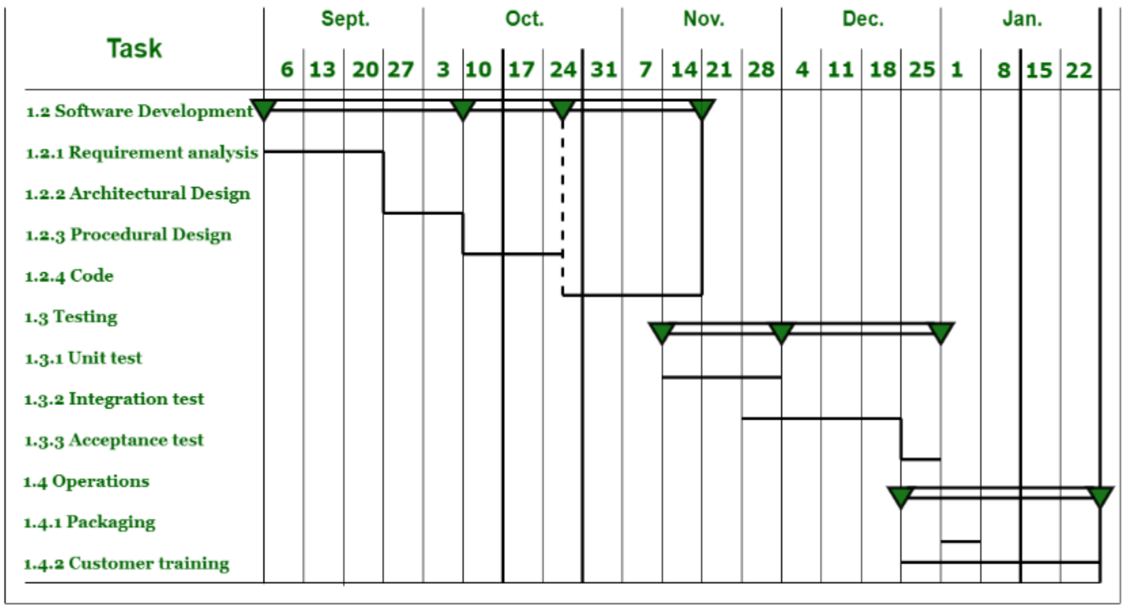
Gantt chart

A **Gantt chart** is a visual view of tasks scheduled over time. **Gantt charts** are used for planning **projects** of all sizes and they are a useful way of showing what work isscheduled to be done on a specific day. They also help you view the start and end dates ofa **project** in one simple view. This practical is useful for tracking progress of the project.

Elements of a gantt chart

Gantt charts may seem complicated at first, but we can start by breaking themdown into 4 sections.

- **Group and task names** A project is made up of several tasks, and related tasks can be organized into groups.
- **Group and task bars** group and task bars that correspond to the group and task names. Each bar represents when the task will start and end.
- **Milestones** A milestone is an important goal, event, or deliverable in your project, such as a kickoff meeting or major deadline. Using milestones in your project plan can help you monitor progress and identify potential delays. Milestones are signified by a gold diamond on the Gantt chart.
- **Dependencies** A dependency links tasks together to ensure work getsdone in the right order. a dependency shows up as a light gray line connecting tasks on your gantt chart.



Gantt Chart

Burndown Chart:

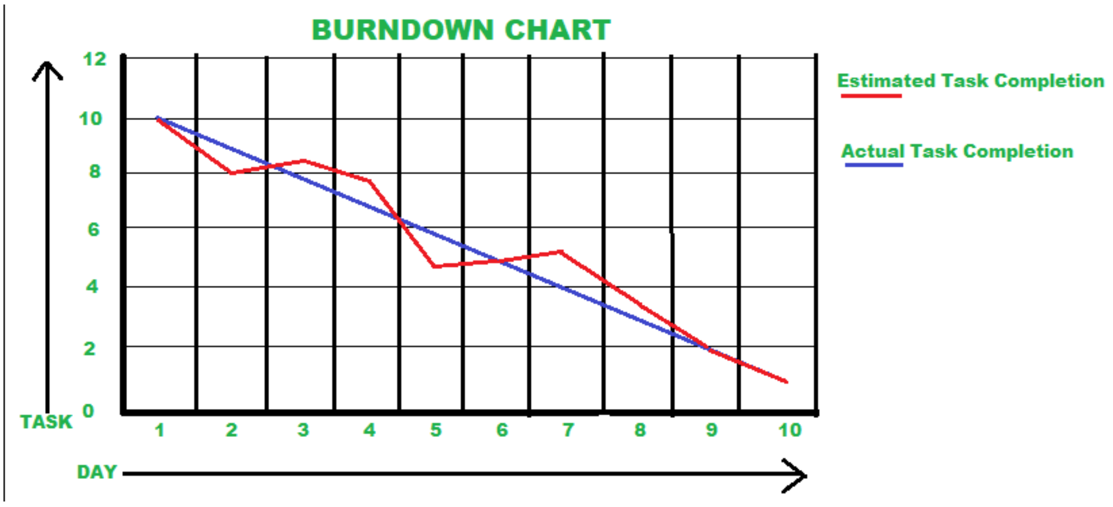
Burndown chart is a major parameter used in agile software development and scrum to detect how much work remains to be completed. It is the graphical representation of showing the left-out portion of the task versus time. It is highly used when a project is going to be done. The company gets the knowledge of 'How the team members are working", and "Can determine the accomplishment of the task".

Steps to Create Burndown Chart:

From Burndown chart project leader gets an idea that it is the graphical representation of work done, work to be done, the time required for pending work done. So these are the main components that are involved during burndown chart scrum creation steps.

- **Estimating Work:** Estimating work means from the backlog deciding what are tasks need to be done and how much time is required to complete this task. More specifically how many user stories in how many days.
- **Estimating Time:** Estimating remaining time means how much time is left for a sprint.
- **Estimating Effort:** Estimating effort means managing effort accordingly looking at the work left and time left and tracking the burndown slope(means work completion track). In Y-axis a point(work needs to be done) and in X-axis a point(time remaining), the straight line from the Y point to the X point is the burn down slope.
- **Tracking Progress:** Tracking daily progress includes comparing the actual daily progress with the expected progress as per effort estimation. So if the daily progress line is below the burndown slope then the team is ahead of schedule, if the daily progress line is above the burndown slope then the team is far behind from the target and if the daily progress line and burndown slope are lined up then it is in the right track.

Example:



H. Practical Related Quiz:

1. What is Gantt chart, flow chart and Sprint burn down chart?
2. What are the Advantages of Gantt Charts, flow chart and sprint burn down chart?
3. Prepare Gantt chart/ Flow chart / burndown chart for your selected system.

I. Assessment-Rubrics

Practical 13: Use flow chart and Gantt charts to track progress of the assigned project. (Use Sprint burn down chart, if agile model is selected).			
Criteria	**M**	**Rubrics**	**Marks**
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date

Practical No.14: Prepare various test cases for selected project.

A. Objective: The purpose of software testing is to validate the working of an application as per the specification. It is to analyze an application.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

C. Expected skills to be developed based on competency

This practical is expected to develop the following skills for the industry identified competency:

1. Apply basic and fundamental knowledge of gantt chart, flow chart and Sprint burn down chart to track progress of the project.

D. Expected Course Outcomes(COs)

1. Prepare test-cases to test software functionalities

E. Practical Outcome(Pros)

1. Prepare various test cases for selected project.

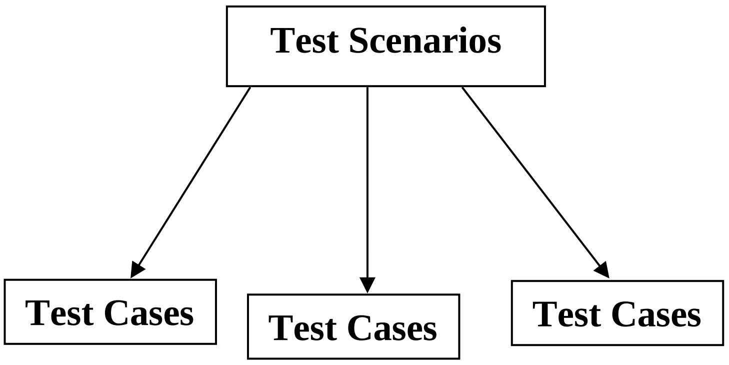
F. Expected Affective domain Outcome(ADos)

1. Work as a leader/ a team member
2. Follow ethical practice.

G. Prerequisite Theory:

Test Case

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.



Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.

Example:

Test scenarios are can cover a wide range of possibilities or specific values.

For a Test Scenario: Check Login Functionality there many possible test cases are:

- Test Case 1: Check results on entering valid User Id & Password
- Test Case 2: Check results on entering Invalid User ID & Password
- Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

The format of Standard Test Cases

Below is a format of a standard login Test cases example.

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check response when valid email and password is entered	Email: gtu@email.com Password: 123423	Login should be successful	Login was successful	Pass

Step 1: Test Case ID

Test cases should all bear unique IDs to represent them. In most cases, following a convention for this naming ID helps with organization, clarity, and understanding.

Step 2: Test Description

This description should detail what unit, feature, or function is being tested or what is being verified.

Step 3: Test Data

This relates to the variables and their values in the test case. In the example of an email login, it would be the username and password for the account.

Step 4: Expected Result

This indicates the result expected after the test case step execution. Upon entering the right login information, the expected result would be a successful login.

Step 5: Actual Result

As compared to the expected result, we can determine the status of the test case. In the case of the email login, the user would either be successfully logged in or not.

Step 6: Pass/Fail

Determining the pass/fail status depends on how the expected result and the actual result compare to each other.

Same result = Pass
Different results = Fail

H. Practical Related Quiz:

1. Design all possible test cases of your selected system

I. Assessment-Rubrics

Practical 14: Prepare various test cases for selected project.

Criteria	M	Rubrics	Marks
Regularity	5	Low: (1-2 marks) Poor (0-40%)	
		Medium: (3-4 marks) Moderate (40-70%)	
		High: (5 marks) High (>70%)	
Problem Analysis	5	Low:(1-2 marks) Very Less Identification of the Problem	
		Medium(3-4 marks) Limited Identification of the Problem	
		High: (5 marks) Apt & Full Identification of the Problem	
Development of the Solution	5	Low:(1-2 marks)- Very Less Solution for the Problem	
		Medium: (3-4 marks) Incomplete Solution for the Problem	
		High: (5 marks) -Complete Solution for the Problem	
Testing of the Solution	5	Low:(1-2 marks) Very less correct solution for the problem	
		Medium(3-4 marks) Partially Correct Solution for the Problem	
		High: (5 marks) Correct Solution as required	
Mock viva test	5	Low: (1-2 marks) -Very few questions answered correctly	
		Medium: (3-4 marks) Delayed & partially correct response	
		High: (5 marks)All questions responded correctly	
		Marks for Practical	

Sign with Date