Madhuben & Bhanubhai Patel Institute of Technology

Computer Engineering/IT Department

Lab manual

Semester: 6th

Subject Name: SOFTWARE ENGINEERING (2160701)

INDEX

```
public Hashtable countAlphabet(String aString){
   Hashtable table = new Hashtable();
   If (aString.length > 4000) return table;
   StringBuffer buffer = new StringBuffer(aString);
   While (buffer.length() > 0){
         String firstChar = buffer.substring(0, 1);
         Integer count = (Integer)table.get(firstChar);
         if (count == null){
               count = new Integer(1);
         } else{
               count = new Integer(count.intValue() + 1);
         }
         table.put(firstChar, count);
         buffer.delete(0, 1);
   }
   return table;
}
```

1. Guarantees that all independent execution path is exercised at least once;
2. Guarantees that both the true and false side of all logical decisions are exercised;
3. Executes the loop at the boundary values and within the boundaries.

Sketch out Design control flow diagram and Apply Cyclomatic complexity for given Code. Identify numbers of Independence path require for testing.

EXPERIMENT-10

Subject Project: For below mentioned Systems and other systems assign a mini-project two a group of students to prepare.

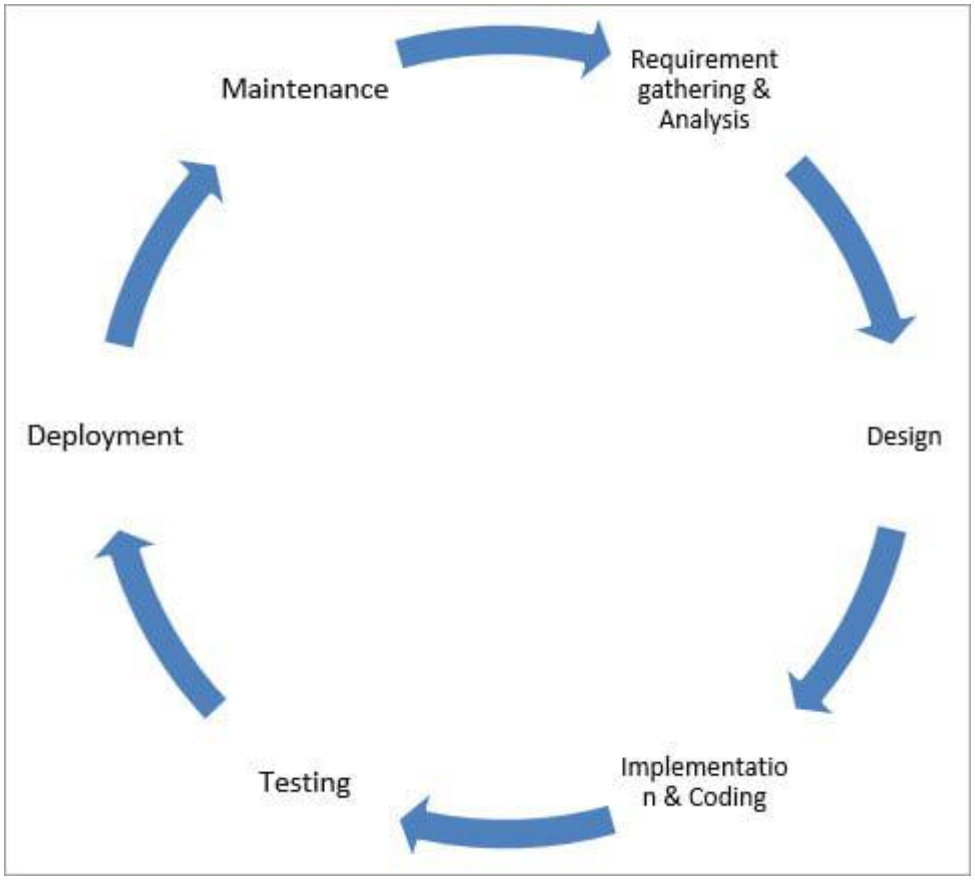
1. Library Information System
2. Villager Telephone System
3. Waste Management Inspection Tracking System (WMITS)
4. Flight Control System

EXPERIMENT-1

Aim: Study the complete Software Development Life Cycle (SDLC) and analyse various activities conducted as a part of various phases. For each SDLC phase, identify the objectives and summaries outcomes.

What is SDLC

- Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.
- Purpose of SDLC is to deliver a high-quality product which is as per the customer's requirement.
- The following figure is a graphical representation of the various stages of a typical SDLC.



1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

In our project, customer wants to have an application which involves the flower classification. In this case, the requirement has to be clear like what kind of operation will be done, how it will be done, by using which method it will be done, etc.
Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.

Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer's expectation. Testers refer SRS document to make sure that the software is as per the customer's standard.

5) Deployment

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.
In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

EXPERIMENT-2

Aim: Do requirement analysis and develop SRS for your project.

SRS (System Requirement Specification):

- The software requirements provide a basis for creating the software requirements Specification (SRS).
- The SRS is useful in estimating cost, planning team activities, performing tasks, and tracking the team's progress throughout the development activity.

Table of Contents

5. Non-functional Requirements

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to build a flower recognition model using machine learning that will identify name of flower with maximum accuracy.

1.2 DOCUMENT CONVENTIONS

This document uses the following conventions.

DS	Data set
Global features	Like
Local features	like

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the flower classification system and it is useful across the world. This has been implemented under the guidance of college professors. This project is useful for the all ages of people as well as for scientist and researchers.

1.4 PROJECT SCOPE

The flower recognition system is to be developed to give the name of any flower given by the user with accurate accuracy.

- This system will also provide login facility.
- The system provides the members with the option to check their account and/or change their options like password of the account whenever needed all through the day.
- In this system the user can choose the image of flower from gallery or either they can Take the photo through the camera provide in phone.

1.5 REFERENCES

- https://www.researchgate.net/publication/320746968
- https://www.researchgate.net/publication/323725215
- https://www.researchgate.net/publication/325544411_A_Review_on_Flower_Classification_Using_Neural_Network_Classifier

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

The flower classification system is used to predict the name of flower with accurate accurarcy. This system is used by researcher, scientist as well as for aryuvedic students to know the name of unknown flower. As these application will display the name of flower within the second so it will also helpful in saving time.

2.2 PRODUCT FEATURES

It will predict the name of any flower that will be given by the user as input.

2.3 USER CLASS and CHARACTERISTICS

User on this system can first login with their email and password. After login they can select image either from gallery or can take the pictures through camera. After giving image as input the system will apply some algorithm in back-end and predict the name of the flower and give the name of the flower with input image to users as output.

2.4 OPERATING ENVIRONMENT

Operating environment for the flower recognition system is as follow:

- Data set
- Operating system: Windows.
- Platform: Python, Anaconda or Spider.

2.5 ASSUMPTION DEPENDENCIES

- Quality of input image

3. SYSTEM FEATURES

3.1 DESCRIPTION and PRIORITY

- First priority goes to the welcome form. When user will login the system there will be welcome screen which will assure to the use either he/she wants to enter or exit.
- After the welcome form there will be login form. Through this form only authenticated users can login the system just by entering their name and password.
- Second priority goes to the menu form where the contents, manuals, and some others Functions of the system. From the menu form the user can go any form of the system.
- Similarly if the user wants to just show the report then he will go to the report section And present the report in the desired format

3.2 FUNCTIONAL REQUIREMENTS

- **Description:** In this system the user can scan as well as upload image of any flower and get the name of the flower with accurate results from the Data set.
- **Risk**: Due to low internet connections there might be a problem while using the system.
- **Technical Issues:** Due to accuracy problem it will not predict right label.

:

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

- The design or layout of every form will be very clear and very interactive to the user.
- In the login section the user can easily entered the desired password and login name. Then it will give the successfully login message.
- In every android & Mac there is help and support option is present for the ease of user.
- The user will be able to search any data from the record by using proper guideline shown in the android & Mac.
- This software will be easily understandable and operable by the user

4.2 HARDWARE INTERFACES

- Android phone

.

4.3 SOFTWARE INTERFACES

Following are the software or languages used for the flower classification system

Software or language used	Description
Operating system	We have chosen Windows operating system for its best support and user-friendliness.
Python	To implement the project we have chosen python language for its more interactive support.
Android	For making application of flower classification

4.4 COMMUNICATION INTERFACES

This project supports all types of platforms. It will runs on android as well as on mac.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

- This software is not breakdown suddenly in any disaster like power failure.
- The timeline of this software must be in our mind.
- The performance of the functions and every module must be well.
- At every step the output of the one phase is the input of the other phase and it will be Reliable and accurate.

- The risk factor must be taken at initial step for better performance of the software.
- For individual function the performance will be well.
- For login to the software password and user name will be matched to the password And name saved in the database and thus only authenticated users are allowed to the login.

- There will be various ways of retrieving data and it takes less time.

5.2 SECURITY REQUIREMENTS

- There will be proper security regarding to the accessing of data.
- The external security can be provided by given the login authentication.
- The external security can be provided by given the login authentication.
- The data that are stored in the database must be private.
- There is also required a user authentication.
- There is also the facility that the admin can lock his private data that will not be Accessed by anyone.

- The whole software is secure from the outside accessing.

5.3 SOFTWARE QUALITY ATTRIBUTES

- **Adaptability**: This software is adaptable by any organization.
- **Availability**: The availability of the software is easy and for everyone.
- **Correctness**: The results of the function are pure and accurate.
- **Flexibility**: The operation may be flexible and reports can be presented in many ways.
- **Maintainability**: After the deployment of the project if any error occurs then it can be Easily maintain by the software developer.

- **Portability: The** software can be deployed at any machine.
- **Reliability** :The performance of the software is better which will increase the reliability Of the software.

- **Re-usability** :The data and record that are saved in the database can be reused if needed

EXPERIMENT-3

Aim: Draw DFD, Structured chart, Use case diagram, Class diagram.

Data Flow Diagram:

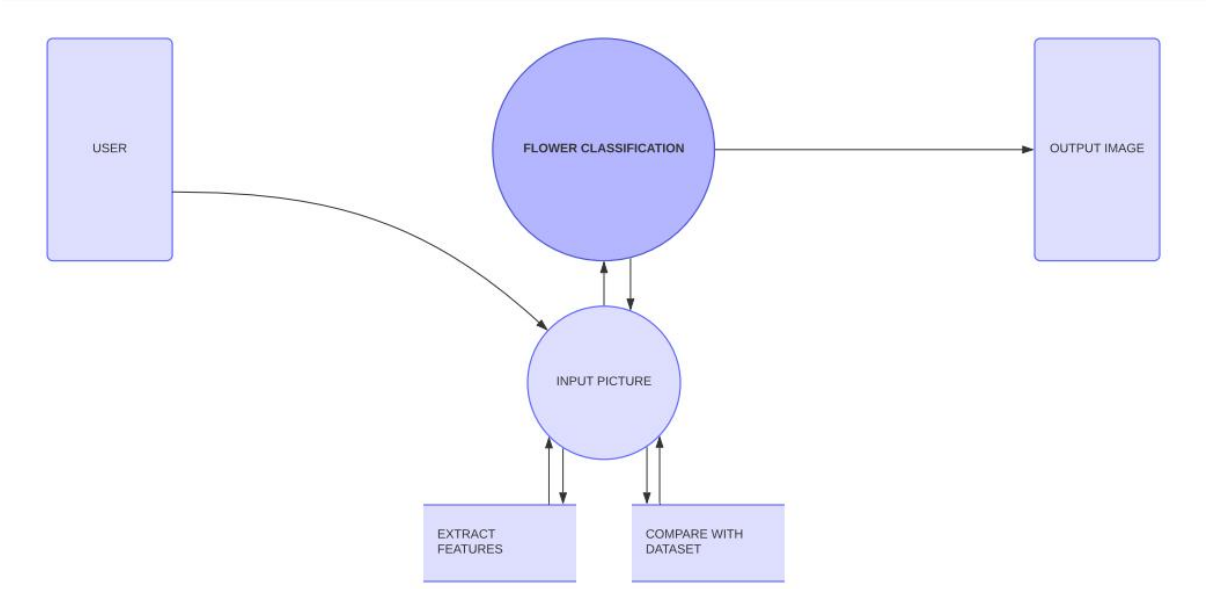
Data Flow Diagram is a means of representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources/destinations. The purpose of data flow diagrams is to provide a semantic bridge between users and systems developers.
The purpose of data flow diagrams is to provide a semantic bridge between users and systems developers. The diagrams are:

- graphical, eliminating thousands of words;
- logical representations, modelling WHAT a system does, rather than physical models showing HOW it does it;
- hierarchical, showing systems at any level of detail; and
- Jargon less, allowing user understanding and reviewing.

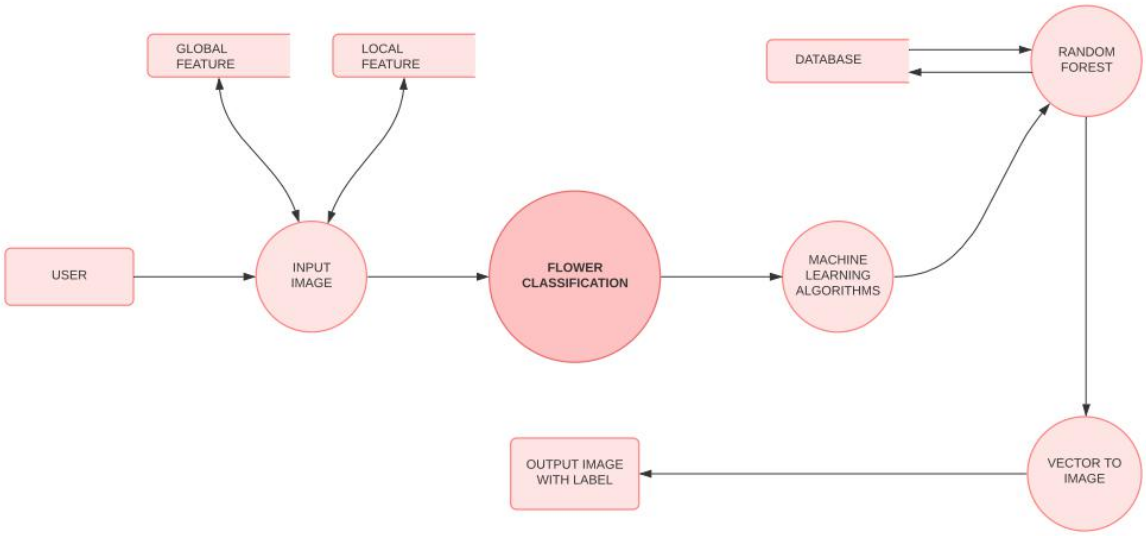
Data Flow Diagram for flower classification:

DFD level 0:

DFD level 1:

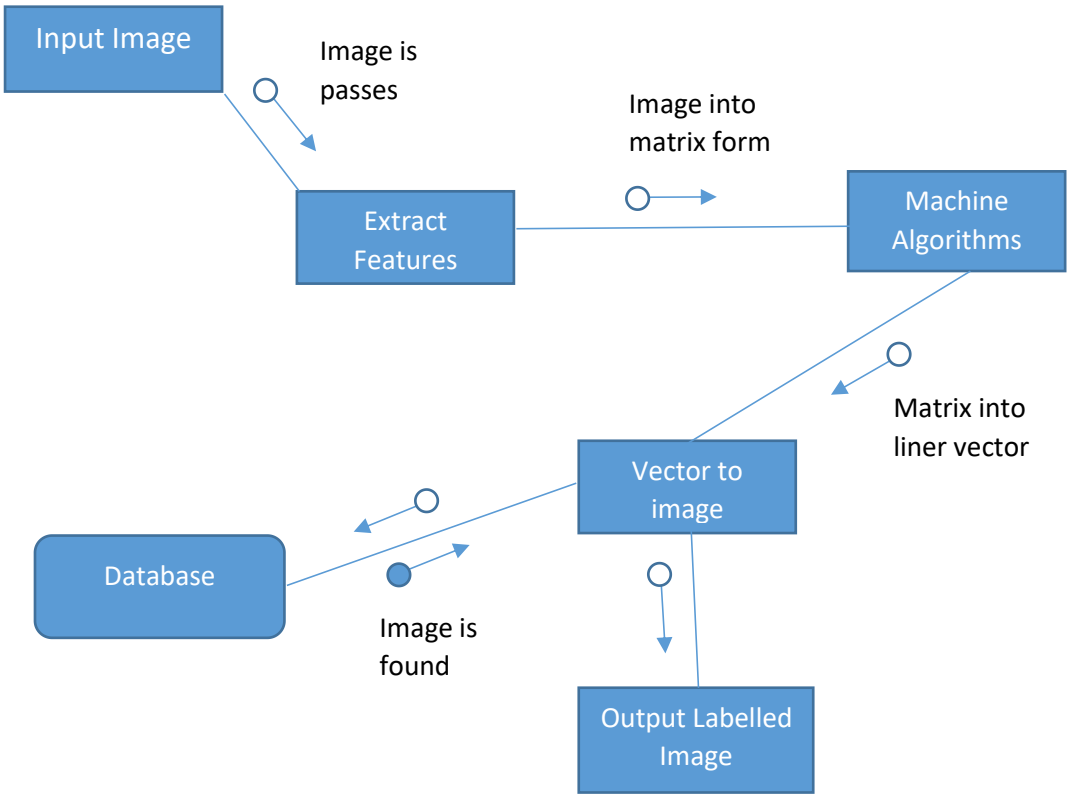


DFD level 2:



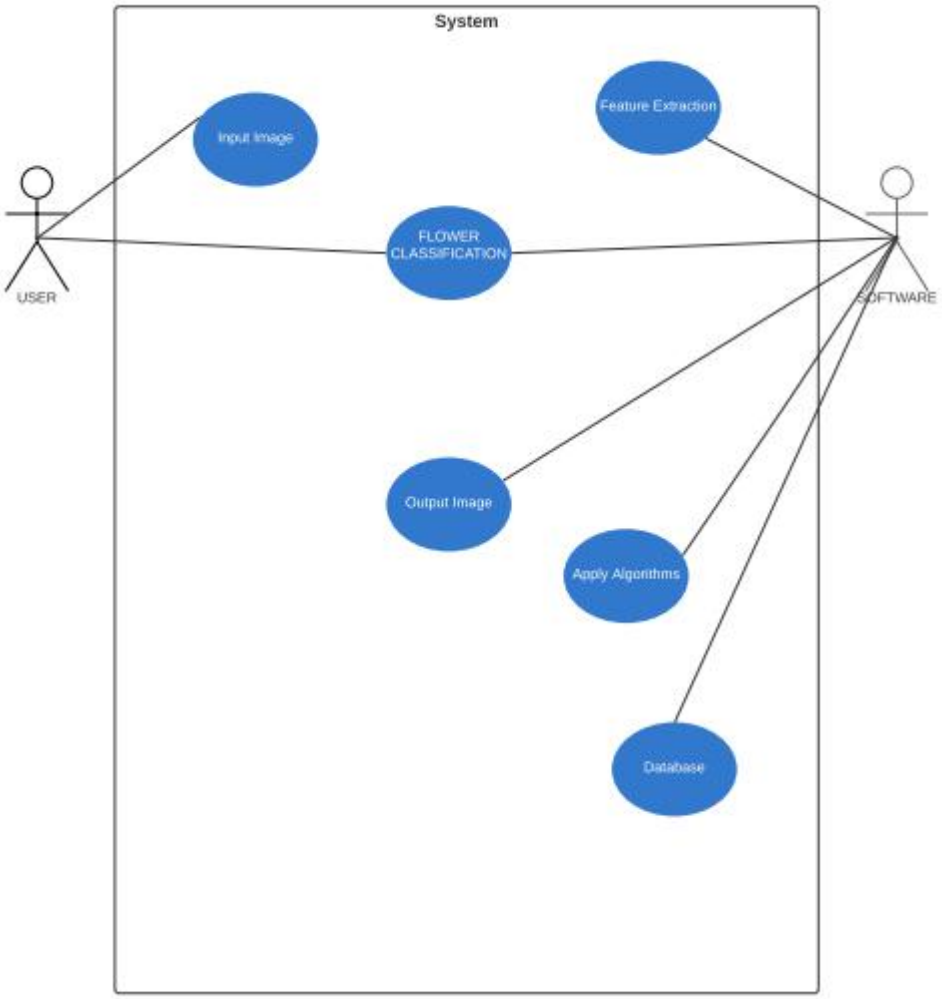
Structure Chart:

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



Use Case Diagram:

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how).

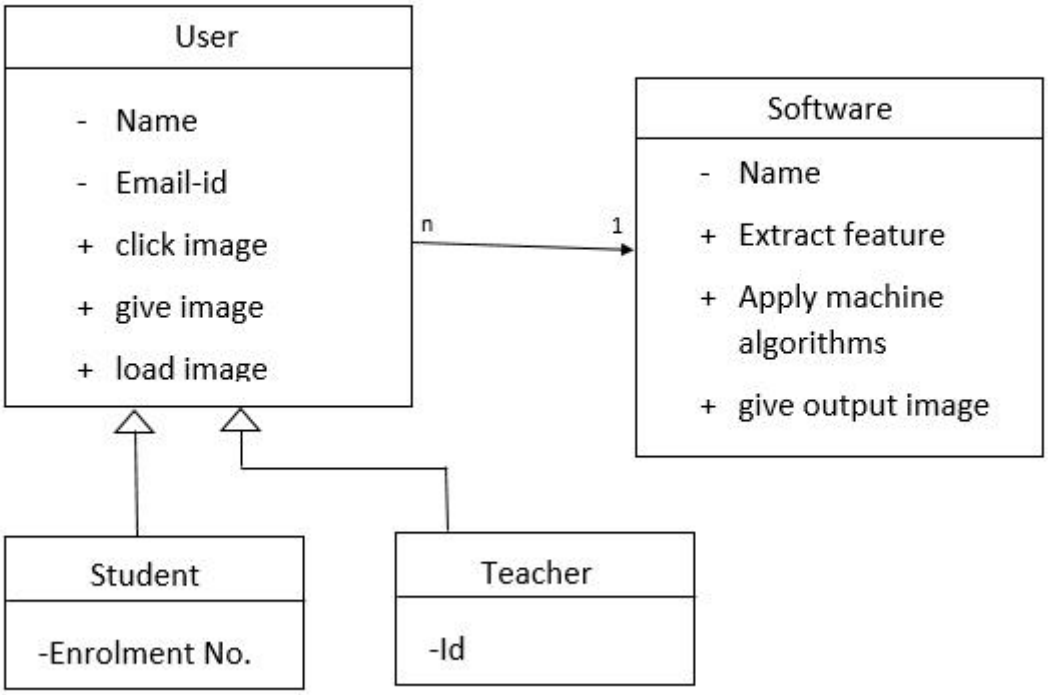


Class diagram:

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

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

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



EXPERIMENT-4

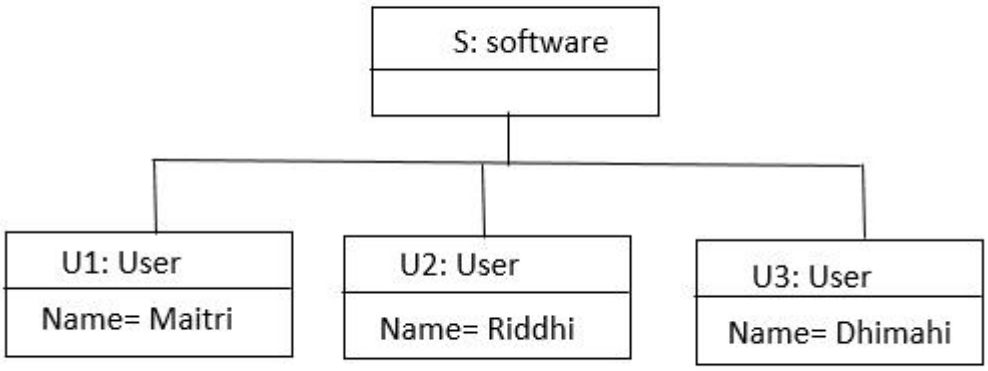
Aim: Draw object diagram, Sequence diagram, Collaboration diagram.

Object Diagram:

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

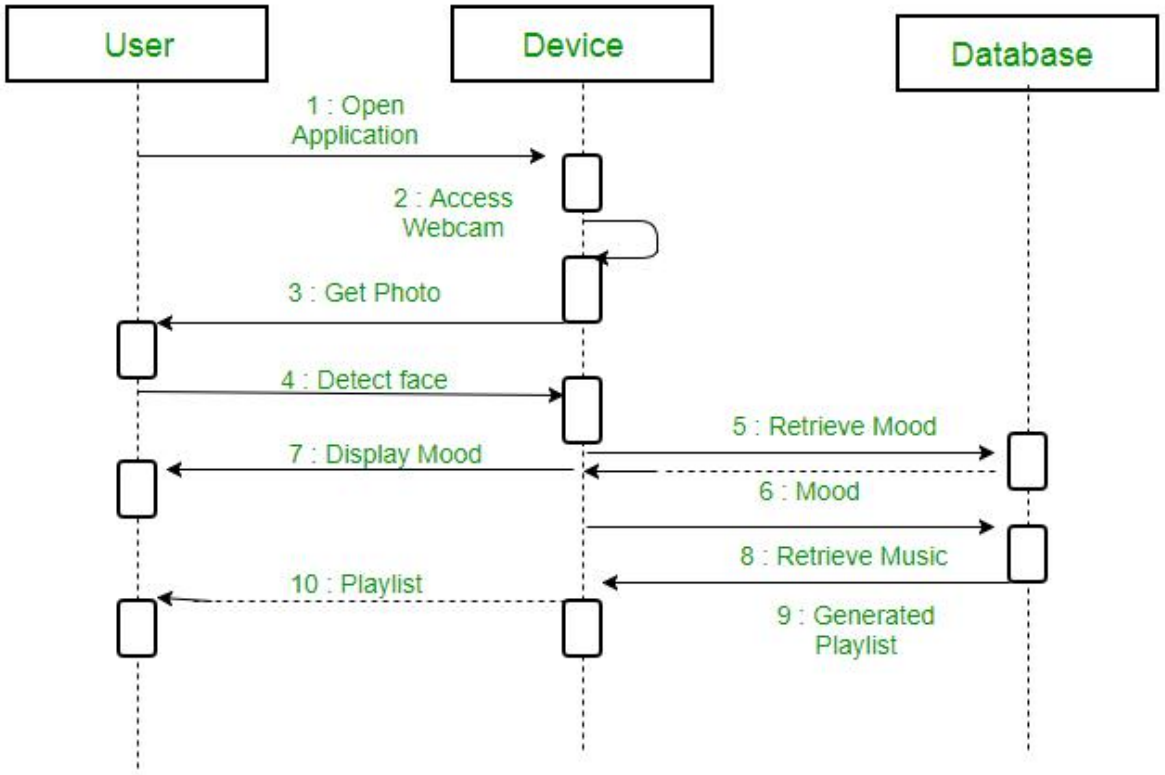
Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

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



Sequence Diagram:

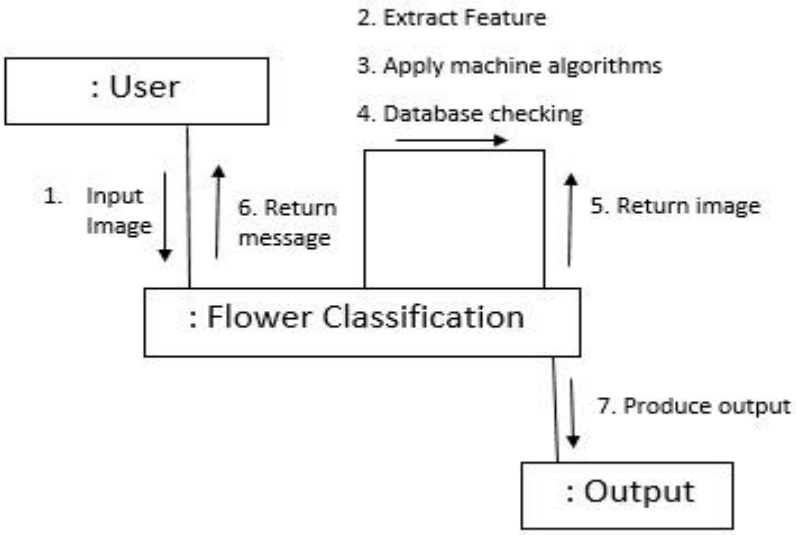
Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.



Collaboration Diagram:

Collaboration Diagram places a priority on mapping the object interactions to the object links (drawing the participating objects in an Object Diagram format and laying the message parallel to the object link).

Collaboration Diagram emphasizes the effect of the object structures on the interactions.



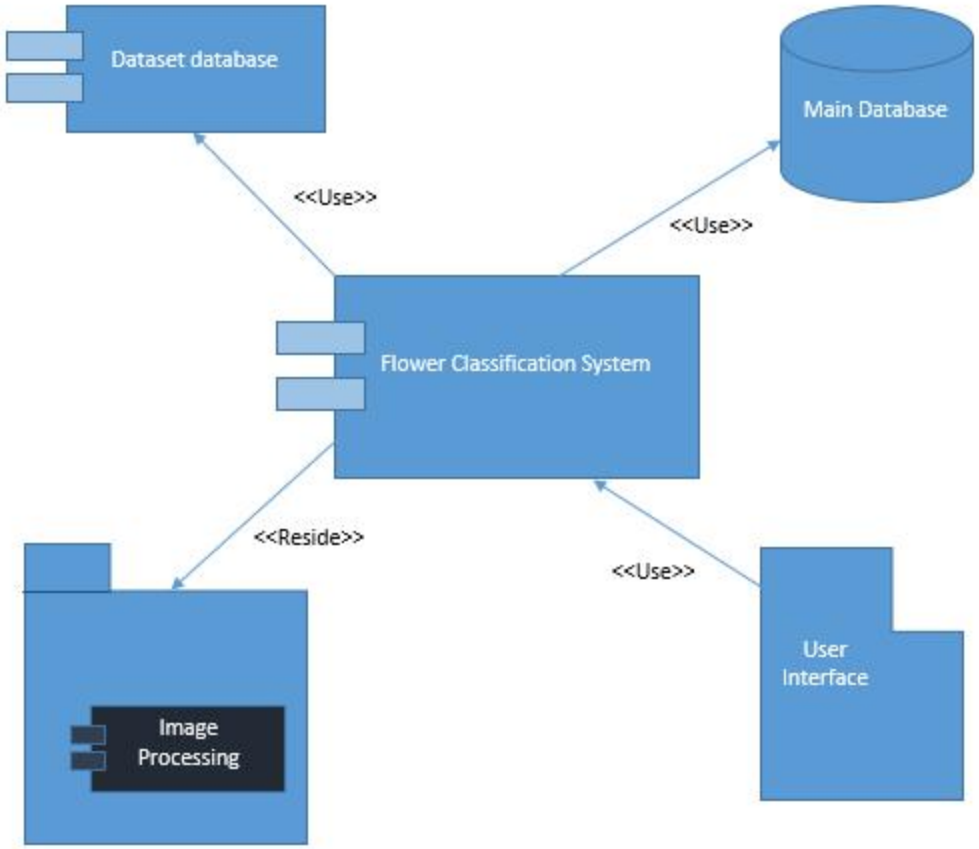
EXPERIMENT-5

Aim: Draw Component diagram and Deployment diagram.

Component diagram:

Component diagrams are different in terms of nature and behaviour.

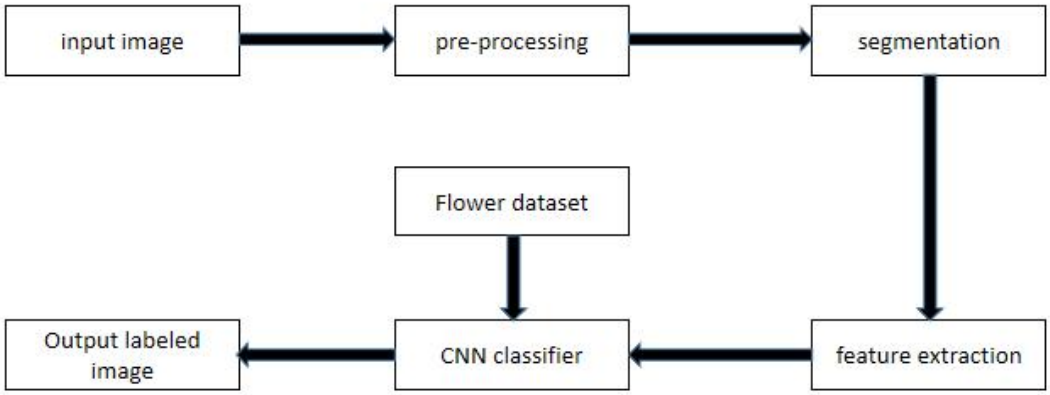
Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.



Deployment diagram:

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

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



EXPERIMENT-6

Aim: Estimate efforts using FP Estimation for chosen system.

Floating Point is an element of software development which helps to approximate the cost of development early in the process. It may measures functionality from user's point of view.

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.

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.

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

FP = Count-total * [0.65 + 0.01 * ∑ (f_i)]
= Count-total * CAF

Where Count-total is obtained from the above Table.

CAF = [0.65 + 0.01 *∑ (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)

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

- Errors/FP
- $/FP.
- Defects/FP
- Pages of documentation/FP
- Errors/PM.
- Productivity = FP/PM (effort is measured in person-months).

- $/Page of Documentation.

In Our project, the values of following function is given below:

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, and 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 ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *∑ (f_i)]
= 378 * [0.65 + 0.01 * 43]
= 378 * [0.65 + 0.43]
= 378 * 1.08 = 408

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

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

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

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

EXPERIMENT-7

Aim: To perform various testing using the testing Methods i.e. unit testing, integration testing.

Unit Testing

Unit testing ensures that each part of the code developed in a component delivers the desired output. In unit testing, developers only look at the interface and the specification for a component. It provides documentation of code development as each unit of the code is thoroughly tested standalone before progressing to another unit.

Unit tests support functional tests by exercising the code that is most likely to break. If you use functional tests without unit tests, you may experience several smells:

- It's hard to diagnose failed tests
- Test fixtures work around known issues rather than diagnosing and fixing them

Unit Testing Tools

There are several automated tools available to assist with unit testing.

1. Junit: Junit is a free to use testing tool used for Java programming language. It provides assertions to identify test method. This tool test data first and then inserted in the piece of code.
2. NUnit: NUnit is widely used unit-testing framework use for all .net languages. It is an open source tool which allows writing scripts manually. It supports data-driven tests which can run in parallel.
3. JMockit: JMockit is open source Unit testing tool. It is a code coverage tool with line and path metrics. It allows mocking API with recording and verification syntax. This tool offers Line coverage, Path Coverage, and Data Coverage.
4. EMMA: EMMA is an open-source toolkit for analysing and reporting code written in Java language. Emma support coverage types like method, line, basic block. It is Java-based so it is without external library dependencies and can access the source code.
5. PHPUnit: PHPUnit is a unit testing tool for PHP programmer. It takes small portions of code which is called units and test each of them separately. The tool also allows developers to use pre-

define assertion methods to assert that a system behave in a certain manner.

We have developed our project flower classification on python. So we have used PHPUnit for Unit testing.

EXPERIMENT-8

Aim: Assume that you are Software Architect or Project Manager in organization. You have been assigned the task of constructing a website for a specific company with your team. Design and priorities the test cases using test case templates for this project.

Test Plan is the most important task of Test Management Process.

Follow the seven steps below to create a test plan

1) Analyze the product

The product under test is company website. First we should research clients and the end users to know their needs and expectations from the application

- Who will use the website?
- What is it used for?
- How will it work?
- What are software/ hardware the product uses?

2) Develop Test Strategy

A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines:

- The project's **testing objectives** and the means to achieve them
- Determines testing **effort** and **costs**

We need to develop Test Strategy for testing that banking website. You should follow steps below

3) Define Test Objective

Test Objective is the overall goal and achievement of the test execution. The objective of the testing is finding as many software defects as possible; ensure that the software under test is **bug free** before release.

To define the test objectives, we should do 2 following steps

1. List all the software features (functionality, performance, GUI…) which may need to test.
2. Define the **target** or the **goal** of the test based on above features

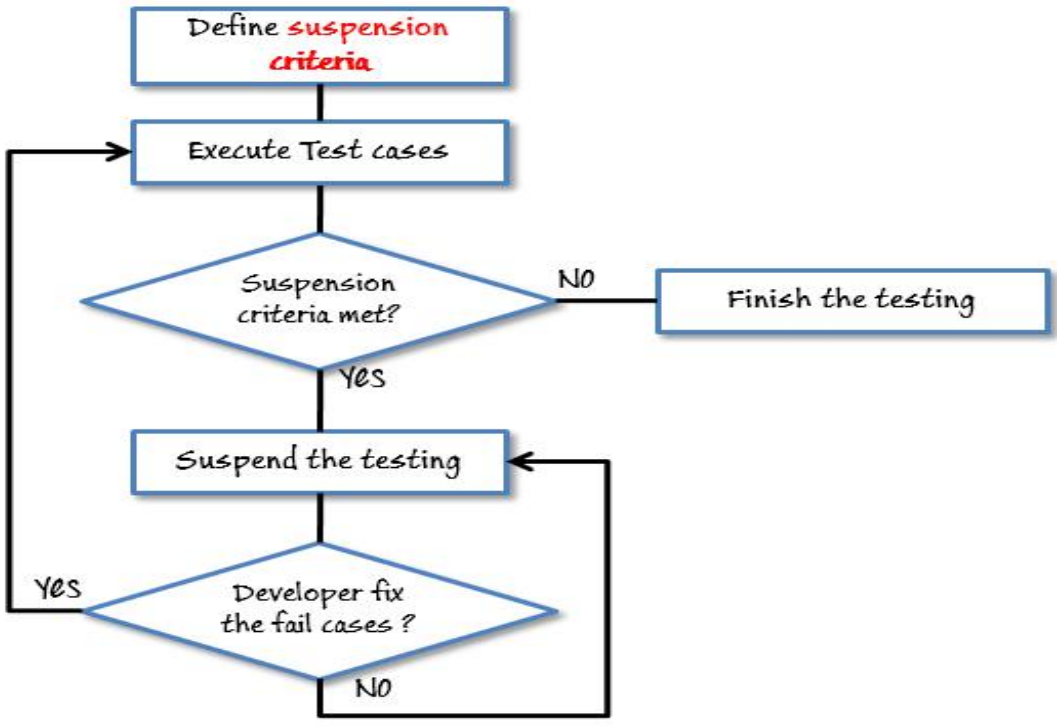
4) Define Test Criteria

Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There're 2 types of test criteria as following

Suspension Criteria

Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**.

Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.



Exit Criteria

It specifies the criteria that denote a **successful** completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: **95%** of all critical test cases must pass.

Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.

- Run rate is ratio between **number test cases executed/total test cases** of test specification. For example, the test specification has total 120 TCs, but the tester only executed 100 TCs, so the run rate is 100/120 = 0.83 (83%)
- Pass rate is ratio between **numbers test cases passed / test cases executed**. For example, in above 100 TCs executed, there're 80 TCs that passed, so the pass rate is 80/100 = 0.8 (80%)

This data can be retrieved in Test Metric documents.

- **Run** rate is mandatory to be **100%** unless a clear reason is given.
- **Pass** rate is dependent on project scope, but **achieving high pass rate** is a goal.

Example: Your Team has already done the test executions. They report the test result to you, and they want you to confirm the **Exit Criteria.**

5) Resource Planning

Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project

The resource planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment…) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

6) Plan Test Environment

A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front end running environment.

We can ask the questions to the developer team:

- What is the maximum user connection which this website can handle at the same time?
- What are hardware/software requirements to install this website?
- Does the user's computer need any particular setting to browse the website?

7) Schedule & Estimation

We should include that estimation as well as the schedule to the Test Planning

Making schedule is a common term in project management. By creating a solid schedule in the Test Planning, the Test Manager can use it as tool for monitoring the project progress, control the cost overruns.

To create the project schedule, the Test Manager needs several types of input as below:

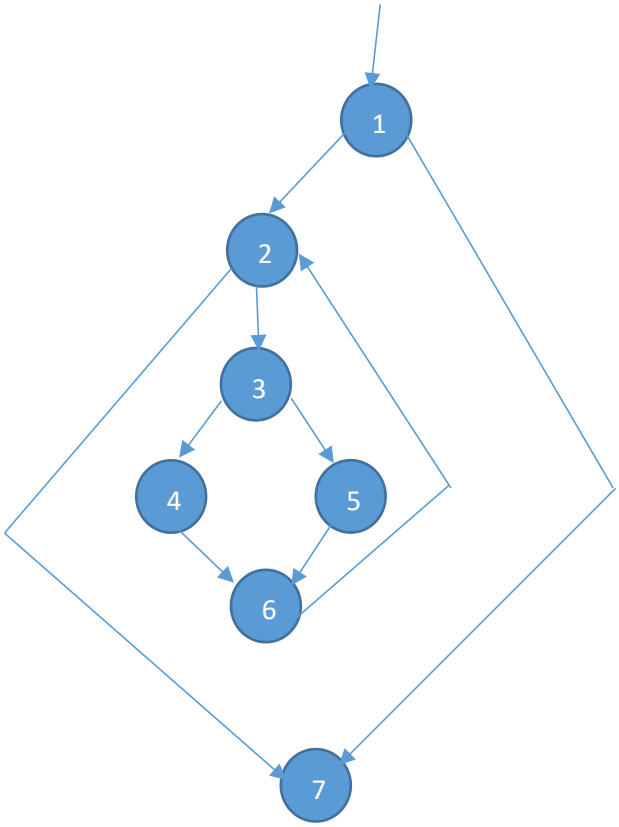
- **Employee and project deadline**: The working days, the project deadline, resource availability are the factors which affected to the schedule
- **Project estimation**: Base on the estimation, the Test Manager knows how long it takes to complete the project. So he can make the appropriate project schedule
- **Project Risk** : Understanding the risk helps Test Manager add enough extra time to the project schedule to deal with the risks

EXPERIMENT-9

Aim: Consider the following Java code segment.

```
public Hashtable countAlphabet(String aString){
   Hashtable table = new Hashtable();
   If (aString.length > 4000) return table;
   StringBuffer buffer = new StringBuffer(aString);
   While (buffer.length() > 0){
         String firstChar = buffer.substring(0, 1);
         Integer count = (Integer)table.get(firstChar);
         if (count == null){
               count = new Integer(1);
         } else{
               count = new Integer(count.intValue() + 1);
         }
         table.put(firstChar, count);
         buffer.delete(0, 1);
   }
   return table;
}
```

Flow graph notation for given program:



Cyclomatic complexity = E – N + 2*P
Where,

E = represents a number of edges in the control flow graph.

N = represents a number of nodes in the control flow graph.

P = represents a number of nodes that have exit points in the control flow graph.

The graph shows seven shapes (nodes), seven lines (edges), hence Cyclomatic complexity is 7-7+2 = 2.