Experiment 1: Apply the knowledge of SRS and prepare Software Requirement Specification (SRS) document in IEEE format for the project

Learning Objective: Students will be able to List various hardware and software requirements, distinguish between functional and nonfunctional requirements, indicate the order of priority for various requirements, analyze the requirements for feasibility.

Tools: IEEE template and MS Word

Theory:

The srs should contain the following Table of Contents

Introduction .....
- Purpose .....
- Intended Audience and Reading Suggestions .....
- Product Scope .....
- Reference

Overall Description .....
- Product Perspective.
- Product Functions .....
- User Classes and Characteristics.....
- Operating Environment .....
- Design and Implementation Constraints .....
- Assumptions and Dependencies.....

External Interface Requirements .....
- User Interfaces .....
- Hardware Interfaces .....
- Software Interfaces.....
- Communications Interfaces.....

System Features .....
- System Feature 1 .....
- Feature 2 (attach further requirements) .....

Other Nonfunctional Requirements .....
- Performance Requirements .....
- Safety Requirements .....
- Security Requirements .....
- Business Rules .....

Other Requirements .....

Appendix A: Glossary.....
- Prepared by,.....

Software Requirement Specification (SRS)

For

Blood Bank Management System

Prepared by:

Harsh Mishra TE COMP B 39

Mihir Gharat TE COMP B 37

Aaryan Pimple TE COMP B 55

Devang Punatar TE COMP B 59

TABLE OF CONTENTS

1. INTRODUCTION

- The project blood bank management system is known to be a pilot project that is designed for the blood bank to gather blood from various sources and distribute it to the needy people who have high Requirements for it.
- The Software is designed to handle the daily transaction of the blood bank and Search the details when required.
- It also helps to register the details of donors, blood collection details as well as blood issued reports.
- The software Application is designed in such a manner that it can suit the needs of all the blood bank requirements in the course of future.
- It will help us to find the Blood group with its most efficient time to take care of the blood and it is more easy to hand over the blood to the hospital to help people to get blood on time.
- This all thing is been stored and been seen in this blood bank management system. To help more people trying best to do so.

1.1 PURPOSE

Blood Bank Management Software is designed and suitable for several Blood Bank either operating as indiviuals organizations or part of organizations covers all blood banking process from donors recruitment, donor management, mobile session component preparation, screening covering all test, blood stock inventory maintenance, patient registration, cross matching, patient issues etc.

1.2 DOCUMENT CONVENTION

1 ER: Entity relationship.

2 This document will use IEEE format. For clarity, acronyms and technical jargon, deemed uncommon by the author, will be annotated and included in the glossary. The format for headings is as followed: Major headings are in bold 18pt font, and concurrent headings in bold 14 pt font. Sections are in the format of x.y, where x and y are real, positive integers.

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

Anybody can use this blood bank management system to donor as well as who need blood e.g., Public, Hospitals, Blood Banks, etc.

1.4 PRODUCT SCOPE

This application is built such a way that it suits for all type of blood bank in future.so every effort is taken to implement this project in this blood bank, on successful implementation in this blood bank, we can target other blood banks in the city.

Main modules of the project:

This project have the following modules, to manage all the requirements of the blood bank.

1. Blood donor details
2. Donor details
3. Recipient details
4. Blood collection details
5. Blood issued details
6. Stock details
7. Camp details
8. Reports

To manage employees in the blood bank it had the following modules:

1. Employee details
2. Employee attendance details
3. Employee salary generation
4. Employee salary payment
5. Report

1.5 REFRENCES

- http://www.bharatbloodbank.com
- http://www.lionbloodbank.net/

2. GENERALDESCRIPTION

2.1 PROJECTPRESPECTIVE

- To provide an efficient donor and blood stock management system to the blood bank by recording the donor and blood details.
- To improve the efficiency of blood stock management by altering the blood bank staff when the blood quantity is low it par level or when the blood stock has expired.
- To provide pure blood with no wastage blood is been collected in different types of packs. They are double, triple, and triple (AS), Quadruple pack.
- They provide synchronized and centralized donor and blood stock database.
- To provide immediate storage and retrieval of data and information.

2.2 Product Function

Class of use cases	Use cases	Descriptions
Use cases related to system authorization of system administrator	1. Login of admin. 2. Change password of admin.	1. Log admin into the system. 2.Change login password of the admin of the system.
Use cases related to registration of donor.	1.Register the donor by himself. 2.Register the donor by system admin.	1. store personal, contact, medical details of donors. 2. store personal, contact, medical details of donors.
Use cases related to system authorization of the donor.	1. Login of donor. 2. Change password of the donor.	1.Log donor into the system. 2.Change login password of the donors of the system.
Use cases related to change the registration details of donor.	1. Change personal, contact details by the donor himself. 2. Change personal, contact details by system admin.	1. Change personal and contact details of donors. 2. Change personal and contact details of donors.
Use cases related to withdraw names from the donor list.	1. withdraw reg. details by the donor. 2. withdraw reg. details by the admin.	1. Delete all details of an exact donors by themselves. 2. Delete all details of an exact donors by the system admin.
Use cases related to inform blood donation details.	Send blood donation details to the relevant donors.	Inform the requirement of the blood group to donors who has same blood group.
Use cases related to replace the older HC Certificates.	Replace donors' HC Certificates.	Override the help condition report details.

Use cases related to inform blood testing to the donor.	Send blood testing details.	Inform disease details to relevant donors. Inform donor details who has diseases, to relevant doctors.
Use cases related to access the database.	Search relevant details from the database.	Search and display relevant details from the database.
Use cases related to print statements.	Print the list of newly registered donors, donation details and list of removed name as statements.	Print the list of newly registered donors, donation details, list of removed names of statements.

2.3 User Classes and Characterstics

In here the system admin & the donor are the system users. According to my assumptions the donor who will register to the system from the website easy questions which are in English language & he/she has the ability to realize small instructions & fill the application without any errors & a small knowledge of computers to upload the health condition certificate to the systems. User is very generous to attend the donation with such a small announcement. (Email & SMS Messages).

2.4 operating environment

Particulars	Client system	server system
Operating system	Windows2000prof/Linux	Linux
Processor	Pentium 4, 1.2GHz	Pentium4, 2GHz
Hard Disk	40GB	100GB
RAM	256MB	512GB

2.5 Design and implementation constraints

Who uses internet connection will be guided through small and clear descriptions. Every donor may get user name and a password in order to log into the will authenticate the accuracy of the donor's mobile numbers through counting the numbers of characters in the entered mobile number system uses the donor registration number and the identity card number to identify each donor separately. Inside the system the administrator has more advance function than the donor. The hospital doctor is not a user of the system. But the doctor connects to the system ina different manner. The doctor mainly has the connection with the system admin

2.6 User Documentation

For user documentation and information please consult section 4 external interface requirement and attached user manual.

2.7 Assumptions and Dependency

There are:

- Every donor has a mobile phone.
- The system doesn’t keep the details of the gathering stock of blood.
- The system database will be accessible in real time.
- The donor doesn’t submit any fake reports to his system.
- The donors who want to contribute to a donation will definitely reply to the request of system.
- The installation of the system to the website server hasn’tconsidered as a process inside the system that process will do by the authorities who controlling the website Therefore, in here the installation the process is considered as a process which is in o3utside of the scope.
- A doctor or a patient can request for an exact blood group. But the request comes through blood bank authorities to the system admin. Therefore, doctor, patient are not directs users of the system.

3. External Interface Requirement

The system is basically running on the official website of the govt, blood bank. Mainly there are 2 actors in the system. The system provides some advance features to the system admin than the donor. If the system admin logs in, the system interface providessome main command buttons to the admin.

- Change login password.
- Edit donor profile details.
- Search donors for a exact blood group and send messages.
- Print statements.
- Update the database.
- Send blood testing details.
- Search details from the database.

If the donors log in, the system will provide another different interface with different commands.

- Change login passwords.
- Edit personal .contact details
- Details related to contributions to donations.
- Future blood donation details.
- Withdraw name from the system.

3.1 User Interfaces

It has been required that every form's interface should e user friendly and simple to use.

3.2 Hardware InterfaceS

- 1GHz or High processor
- 512 MB RAM
- 500 MB Hard Disk

3.3 Software Interface

Smart blood bank

- Windows
- Internet Explorer, Chrome, Firefox

3.4 Communication Interface

There are:

- Should run on 500GHz, 64MB Machine.
- Should have a proper internet connection.
- The response time for occurs a change will be more than 4 seconds.
- The response time for access the database will be no more 5 seconds.

4. SYSTEM FEATURES

There are:

- Donor management-donor registration, managing donor database, recording their physical and medical statistics.
- Inventory management in blood bank for storage and issuance of blood.
- Online transform of blood from one blood bank to another.
- Blood requisition and issuance of blood.
- Discarding of expired and unsuitable blood (less Qty., reactive, clotting, hemolysis).
- Being a webbased system, can be implemented throughout the state. Separate users accounts can be created for each blood bank.
- Patient register/blood sample receiving register, donor register, blood issue register and discarded blood report.
- Fridge wise stock position and printing of fridge stickers.
- List of donors who are eligible for donation on a particular date with contact number.
- Camp wise donor list and printing of donor cards.

4.1. SYSTEM FEATURE 1

4.1.1 Description and priority

- ❖ Information of all blood banks donor details, donate blood with their interest and others will do in future.
- ❖ Interested in donating the blood can register.
- ❖ General users want to contact blood donors by checking interested to donate blood, he can also take the help of this site.
- ❖ Rapid response of urgent request for blood components.
- ❖ Checking pre-transfusion samples and request
- ❖ Assessing of Immunological compatibility between donor and patient.
- ❖ Selecting of suitable blood component of each clinical conditions.
- ❖ Safe delivery and handling of blood components.
- ❖ Inventory and stock management.
- ❖ Interactions with the blood establishment.

4.1.2 Functional Requirements

- Login of admin.
- Blood Donor.
- Change the login password of admin.
- Register the donor by himself.

- Register the donor by system admin.
- Login of the donor.
- Change the login password of donor.
- Change personal, contact details by the donors himself.
- Change personal, contact details by the system admin.
- Withdraw reg. details by the donor.
- Withdraw reg. details by the admin.
- Send blood donation details to the relevant donors.
- Send blood testing details.

4.2 SYSTEM FEATURES 2

There are some features of blood bank management system are:

- ❖ Generating reports on stock- blood group wise, area wise and expiry date wise.
- ❖ Donor database-blood group wise and area wise.
- ❖ Maintain And update unique donor identifications.
- ❖ Track and maintain all the donor types- voluntary, exchange and directed.
- ❖ Improved the Effectiveness and Efficiency of blood bank- faster response time and better control.
- ❖ Accurate database/Record management.
- ❖ Blood cross match and result storage facility.
- ❖ Rejected donor database for donor control and identifications.
- ❖ Blood transfusion related diseases control and preventions.
- ❖ Searched facility for destroyed and expired blood.
- ❖ Comprehensive donor database with search facility.
- ❖ Unique donor ID and patient record ID for managing future list.
- ❖ Improve blood bank processes by providing efficient and continuous software support.

5. OTHER NON-FUNCTIONALREQUIREMENTS

A Characteristics of a quality SRS is that in addition to describing the functional requirements of a system, it will also provide detailed coverage is of the non-functional requirements. In practice, this would entail detailed analysis of issues such as availability, security, usability and maintainability. However, as this document is only an outline specification, It doesn't contain the same degree of rig our that would normally be expected in a formal SRS. Therefore, the sections below should be seen as indicative rather than providing specific (i.e. Testable) requirements.

5.1 Performance Requirements

- The system is interactive and the delays involved are less.
- When connecting to the server the delays is based editing on the distance of the two System configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for the sake of good communications.

5.2 Safety Requirements

Blood bank modules maintains details about the donors and recipients. These blood bank modules is linked to other modules in the software for wards and OT in the hospitals, whereby, any and all blood requirements using surgeries etc. That happens in the hospitals are known to the bank. Important information and parameters such as availability of blood, cross-matching between donor's and recipient's blood groups and blood transfusions reactions are recorded. Also , the interactions with other blood banks within in a hospitals or outside and delivery/recipients of blood bags between these banks or hospitals are recorded and maintained.

- The blood request queue screen, from where all the daily transfusions can be handled.
- Fresh blood and stored blood request processing.
- Blood returns is made easy in the blood bank management system.
- Transfusion detailed and charging.
- Destruction Details.
- Blood bank management system is integrated with lab module for blood cross match and grouping.

5.3 Security Requirements

- The system uses SSL (secured socket layers) in all transactions that include any Confidential customers Information.

- The system must automatically log out all customers after a period of inactivity.

5.4 Software Quality Attributes

The system will posses some quality attributes to the users:

1. Reliability:

   The system has the ability to work all the times without failures apart from network failure. A donor can have the faith on the system. The authorities will keeps the privacy of all donors in a proper manner.

   When the doctors found any disease in the testing stage after providing relevant details to the donor the system keeps the secretively of the donor.

2. Robustness:

   The entire system includes every function which is always help to the system to work correctly and strongly in all conditions.

3. Interoperability:

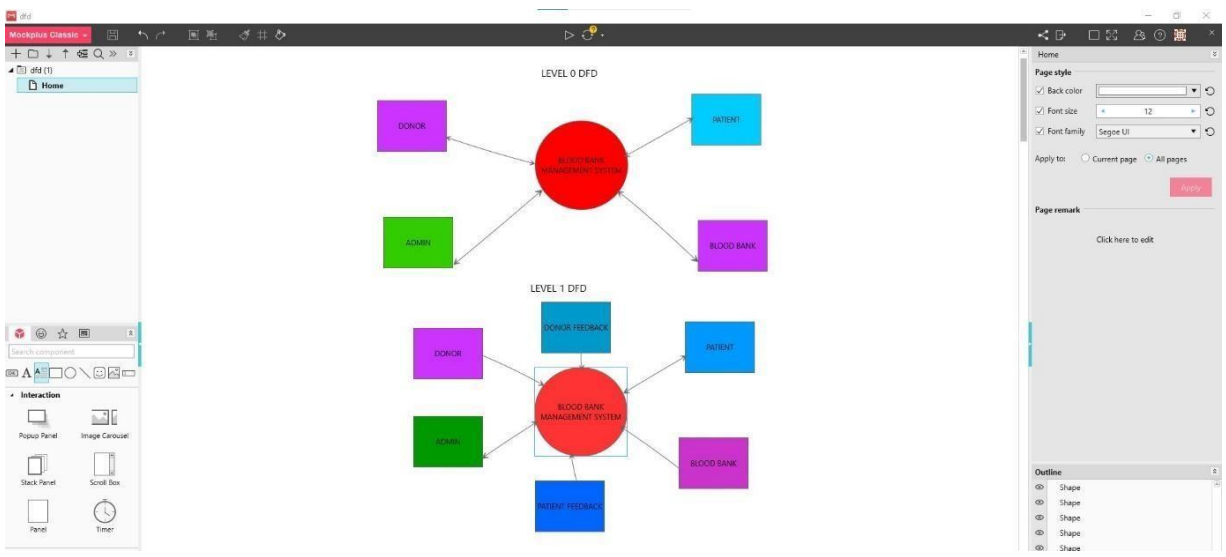
   The system UPAKARA will run on the blood bank website. Therefore the system includes the ability to work with the other applications which are also run on the same website.

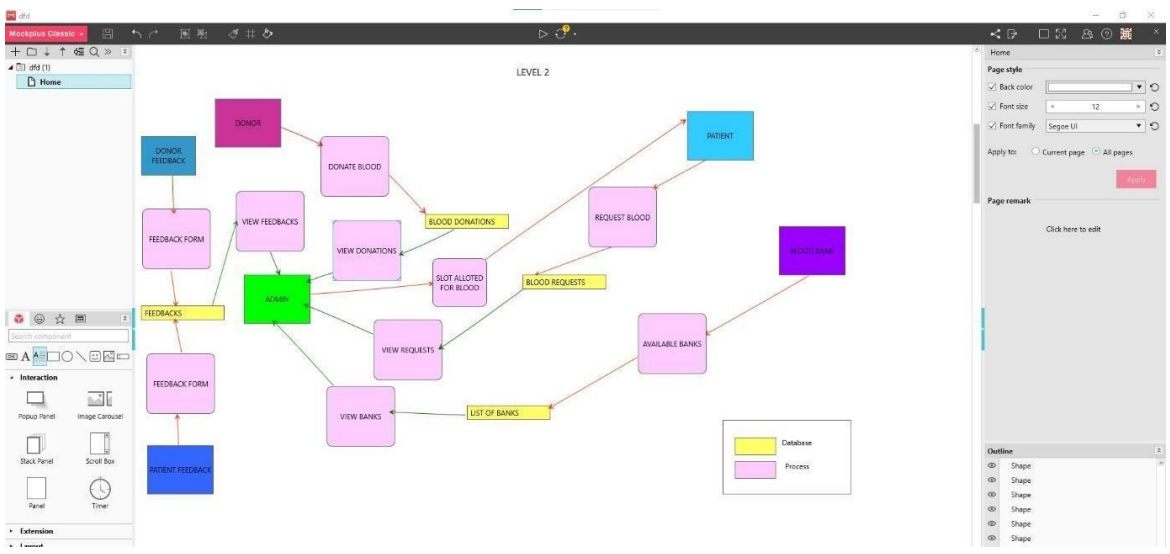
6. OTHER REQUIREMENTS

- ❖ Security: The system doesn't have a tight security system. Because People who log into the system are volunteers who like to donate blood for innocent patients. But the system consists of some security features.
  - Any donor can't see any details of any other donor.
  - If a donor doesn't manage to provide hi user name and a password in three times the user automatically will log out from the website.

DIAGRAMS - Blood Bank Management System

- ✓ Data flow (level 0) Diagram
- ✓ Data flow (level 1) Diagram
- ✓ Data flow (level 2) Diagram


LEVEL 0 DFD
DONOR
PATIENT
BLOOD BANK MANAGEMENT SYSTEM
ADMIN
BLOOD BANK
LEVEL 1 DFD
DONOR FEEDBACK
DONOR
PATIENT
BLOOD BANK MANAGEMENT SYSTEM
ADMIN
BLOOD BANK
PATIENT FEEDBACK


LEVEL 2
DONOR
DONOR FEEDBACK
DONATE BLOOD
PATIENT
BLOOD DONATIONS
REQUEST BLOOD
VIEW FEEDBACKS
FEEDBACK FORM
VIEW DONATIONS
BLOOD BANK
SLOT ALLOTED FOR BLOOD
BLOOD REQUESTS
ADMIN
FEEDBACKS
VIEW REQUESTS
AVAILABLE BANKS
FEEDBACK FORM
VIEW BANKS
LIST OF BANKS
PATIENT FEEDBACK
Database
Process

Learning Outcomes: Students should have the ability to

LO1: List various hardware and software requirements
LO2: Distinguish between functional and nonfunctional requirements
LO3: Indicate the order of priority for various requirements
LO4: Analyze the requirements for feasibility Course

Course Outcomes: Upon completion of the course students will be able to prepare SRS document

Conclusion: We were able to prepare SRS document successfully

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Experiment 2: Use project management tool to prepare schedule for the project.

Learning Objective: Students will able to List the various activities in the project, analyze the various activities for schedule, estimate the time for each activity and develop a Gantt Chart for the activities.

Tools: Gantt Chart using MSExcel

Theory:

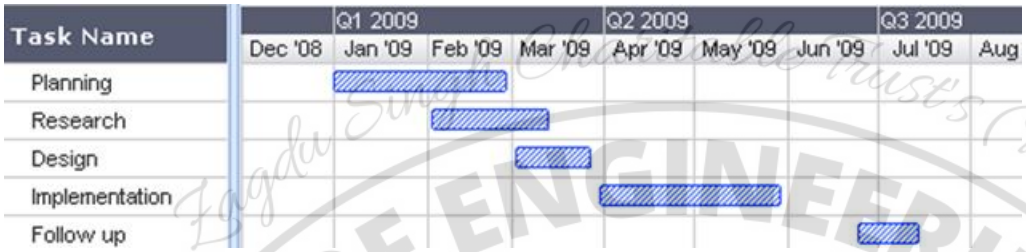
The main aim of PROJECT SCHEDULING AND TRACKING is to get the project completed on time. Program evaluation and review technique (PERT) and Gantt chart are two project scheduling methods that can be applied to software

Split the project into tasks and estimate time and resources required to complete each task. Organize tasks concurrently to make optimal use of workforce. Minimize task dependencies to avoid delays caused by one task waiting for another to complete

Gantt chart:

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are
- When each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much



OUTPUT:

Learning Outcomes: Students should have the ability to

LO1: List the various activities in the project.
LO2: Analyze the various activities for schedule.
LO3: Estimate the time for each activity.
LO4: Develop a Gantt Chart for the activities.

Outcomes: Upon completion of the course students will be able to use project management tool to prepare schedule for the project.

Conclusion: Successfully implemented gant chatt for project using smartsheet.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	

Experiment 3: Draw DFD (upto 2 levels) and prepare Data Dictionary for the project

Learning Objective: Students will able to identify the data flows, processes, source and destination for the project, Analyze and design the DFD upto 2 levels, develop a data dictionary for the project

Tools: Dia, StarUML

Theory:

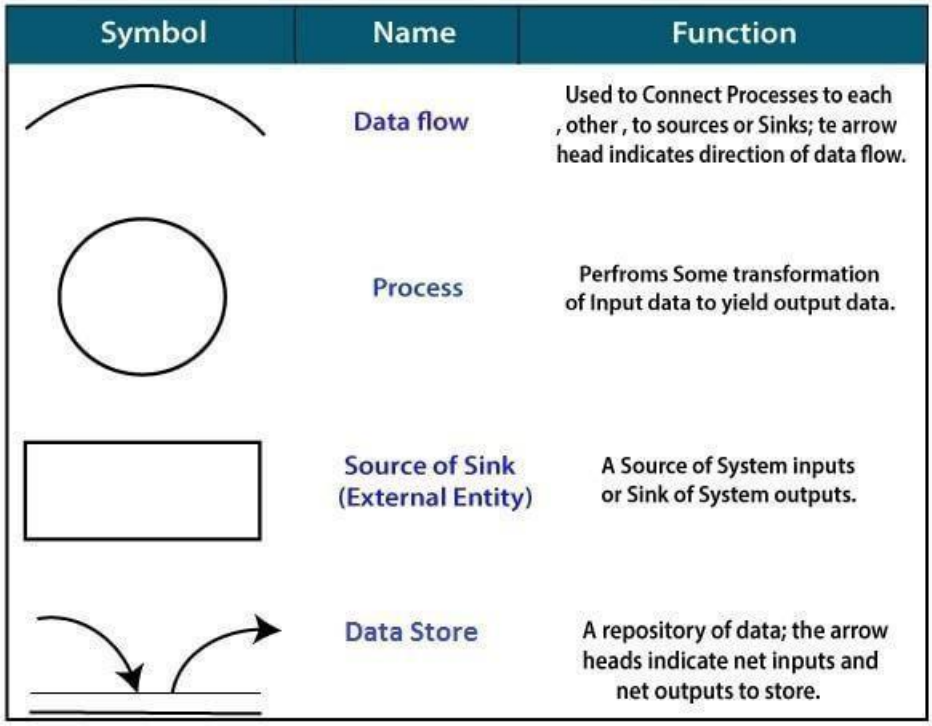
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

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.



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.

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

Standardizing behavior. DFDs are derived from electrical circuit diagram analysis and are shown in fig:

Standardizing behavior. DFDs are derived from electrical circuit diagram analysis and are shown in fig:

Hubbles. 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:

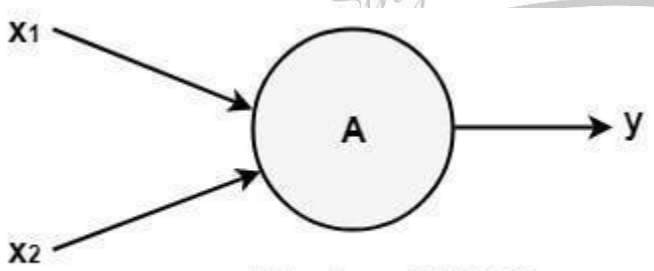


Fig: Level-0 DFD.

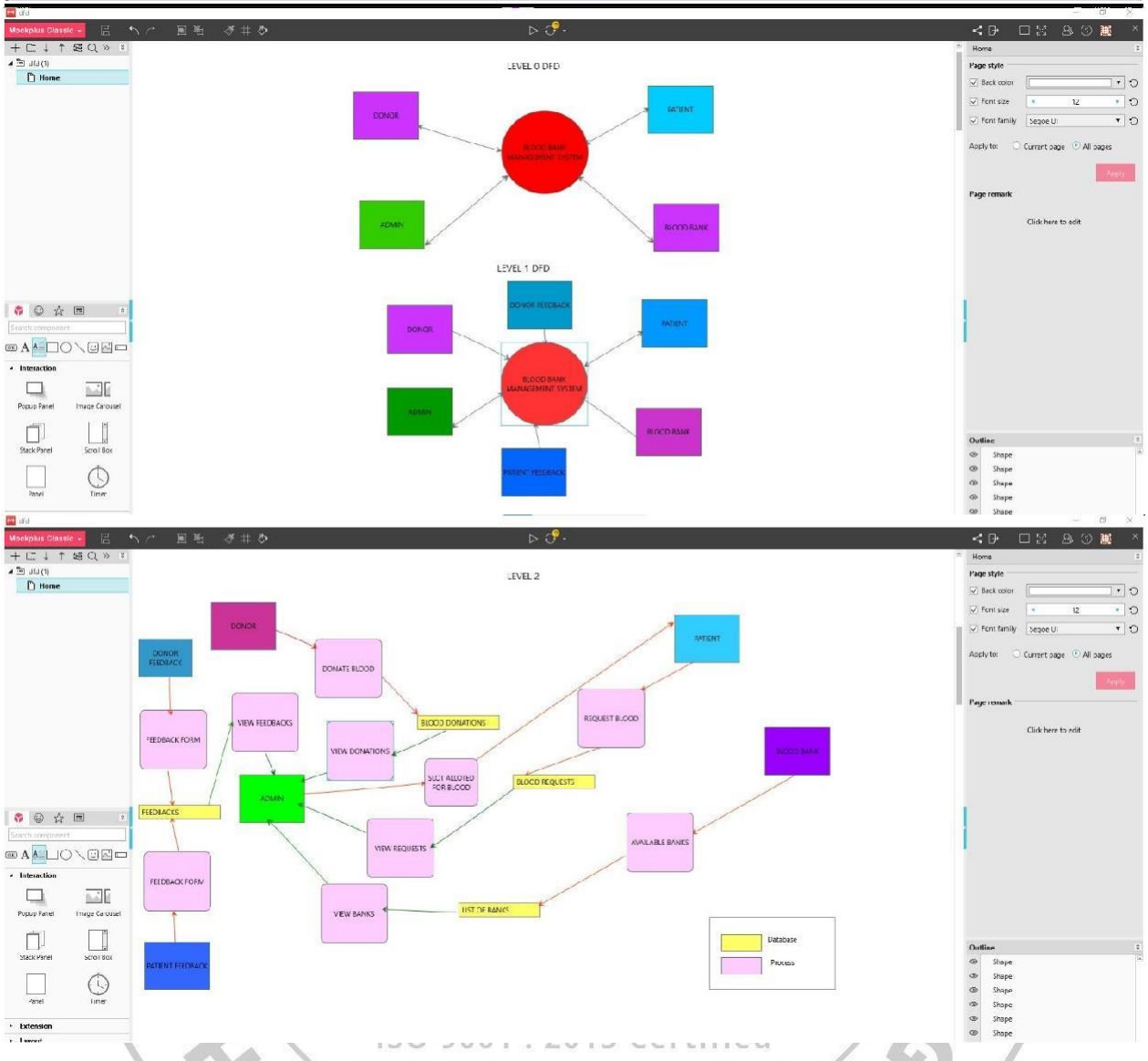
1-level DFD

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.

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.

OUTPUT:



Learning Outcomes: Students should have the ability to

LO1: Identify the dataflows, processes, source and destination for the project.
LO2: Analyze and design the DFD upto 2 levels
LO3: Develop a data dictionary for the project

Outcomes: Upon completion of the course students will be able to prepare Draw DFD (upto 2 levels) and prepare Data Dictionary for the project

Conclusion: Successfully implemented dfd level 0 and 1 using mockplus for project.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment 04- Implement UML Use-case diagram

Learning Objective: To implement UML use-case diagram for the project.

Tools: MS Word, draw.io

Theory:

Use case diagrams

Use case diagrams belong to the category of behavioral diagram of UML diagrams. Use case diagrams aim to present a graphical overview of the functionality provided by the system. It consists of a set of actions (referred to as use cases) that the concerned system can perform one or more actors, and dependencies among them.

Actor

An actor can be defined as an object or set of objects, external to the system, which interacts with the system to get some meaningful work done. Actors could be human, devices, or even other systems.

For example, consider the case where a customer *withdraws cash* from an ATM. Here, customer is a human actor.

Actors can be classified as below:

- **Primary actor**: They are principal users of the system, who fulfill their goal by availing some service from the system. For example, a customer uses an ATM to withdraw cash when he needs it. A customer is the primary actor here.

- **Supporting actor**: They render some kind of service to the system. "Bank representatives", who replenishes the stock of cash, is such an example. It may be noted that replenishing stock of cash in an ATM is not the prime functionality of an ATM.

In a use case diagram primary actors are usually drawn on the top left side of the diagram.

Use Case

A use case is simply a functionality provided by a system.

Continuing with the example of the ATM, *withdraw cash* is a functionality that the ATM provides. Therefore, this is a use case. Other possible use cases include, *check balance*, *change PIN*, and so on.

Use cases include both successful and unsuccessful scenarios of user interactions with the system. For example, authentication of a customer by the ATM would fail if he enters wrong PIN. In such case, an error message is displayed on the screen of the ATM.

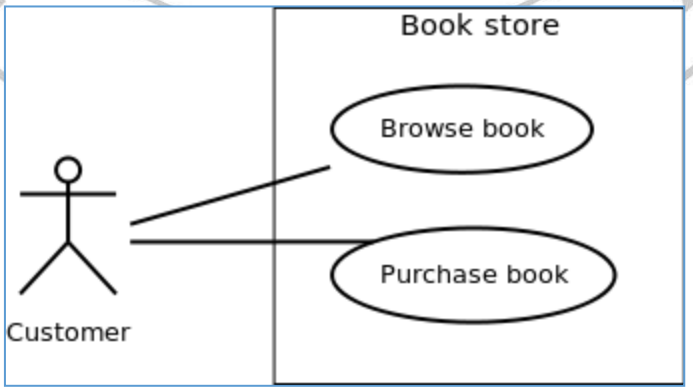
Subject

Subject is simply the system under consideration. Use cases apply to a subject. For example, an ATM is a subject, having multiple use cases, and multiple actors interact with it. However, one should be careful of external systems interacting with the subject as actors.

Graphical Representation

An actor is represented by a stick figure and name of the actor is written below it. A use case is depicted by an ellipse and name of the use case is written inside it. The subject is shown by drawing a rectangle. Label for the system could be put inside it. Use cases are drawn inside the rectangle, and actors are drawn outside the rectangle, as shown in figure - 01.

Figure - 01: A use case diagram for a book store



Association between Actors and Use Cases

A use case is triggered by an actor. Actors and use cases are connected through binary associations indicating that the two communicates through message passing.

An actor must be associated with at least one use case. Similarly, a given use case must be associated with at least one actor. Association among the actors is usually not shown. However, one can depict the class hierarchy among actors.

Use Case Relationships

Three types of relationships exist among use cases:

- Include relationship
- Extend relationship
- Use case generalization

Include Relationship

Include relationships are used to depict common behavior that are shared by multiple use cases. This could be considered analogous to writing functions in a program in order to avoid repetition of writing the same code. Such a function would be called from different points within the program.

Example

For example, consider an email application. A user can send a new mail, reply to an email he has received, or forward an email. However, in each of these three cases, the user must be logged in to perform those actions. Thus, we could have a *login* use case, which is included by *compose mail*, *reply*, and *forward email* use cases. The relationship is shown in figure - 02.

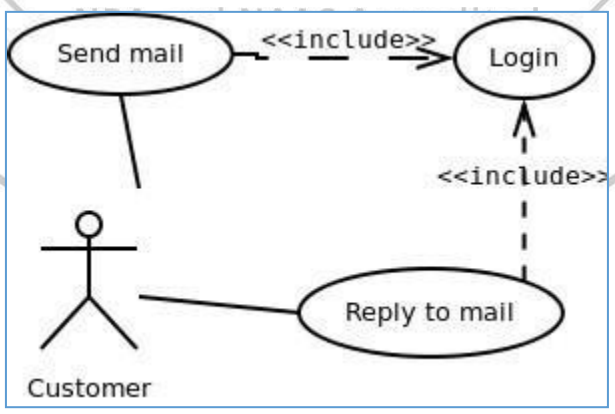


Figure - 02: Include relationship between use cases

Notation

Include relationship is depicted by a dashed arrow with a «include» stereotype from the including use case to the included use case.

Extend Relationship

Use case extensions are used to depict any variation to an existing use case. They are used to the specify the changes required when any assumption made by the existing use case becomes false.

Example

Let's consider an online bookstore. The system allows an authenticated user to buy selected book(s). While the order is being placed, the system also allows specifying any special shipping instructions, for example, call the customer before delivery. This *Shipping Instructions* step is optional, and not a part of the main *Place Order* use case. Figure - 03 depicts such relationship.

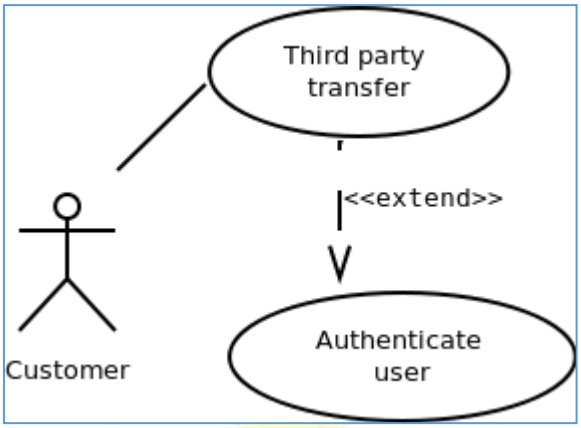


Figure - 03: Extend relationship between use cases

Notation

Extend relationship is depicted by a dashed arrow with a «extend» stereotype from the extending use case to the extended use case.

Generalization Relationship

Generalization relationship is used to represent the inheritance between use cases. A derived use case specializes some functionality it has already inherited from the base use case.

Example

To illustrate this, consider a graphical application that allows users to draw polygons. We could have a use case *draw polygon.* Now, rectangle is a particular instance of polygon having four sides at right angles to each other. So, the use case *draw rectangle* inherits the properties of the use case *draw*

polygon and overrides its drawing method. This is an example of generalization relationship. Similarly, a generalization relationship exists between *draw rectangle* and *draw square* use cases. The relationship has been illustrated in figure - 04.

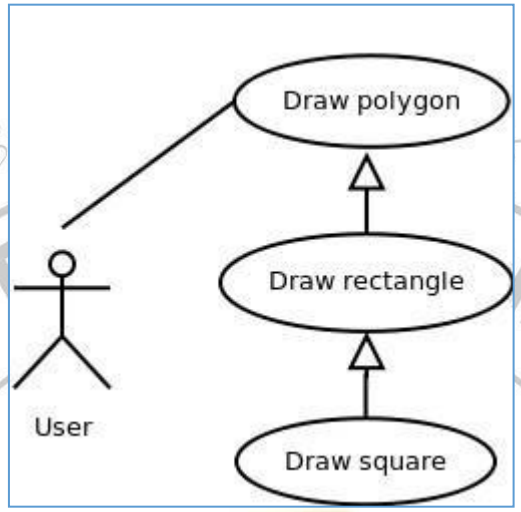


Figure - 04: Generalization relationship among use cases

Notation

Generalization relationship is depicted by a solid arrow from the specialized (derived) use case to the more generalized (base) use case.

Identifying Actors

Given a problem statement, the actors could be identified by asking the following questions :

- Who gets most of the benefits from the system? (The answer would lead to the identification of the primary actor)
- Who keeps the system working? (This will help to identify a list of potential users)
- What other software / hardware does the system interact with?
- Any interface (interaction) between the concerned system and any other system?

Identifying Use cases

Once the primary and secondary actors have been identified, we have to find out their goals i.e. what the functionality they can obtain from the system is. Any use case name should start with a verb like, "Check balance".

Guidelines for drawing Use Case diagrams

Following general guidelines could be kept in mind while trying to draw a use case diagram :

- Determine the system boundary
- Ensure that individual actors have well-defined purpose
- Use cases identified should let some meaningful work done by the actors
- Associate the actors and use cases -- there shouldn't be any actor or use case floating without any connection
- Use include relationship to encapsulate common behavior among use cases , if any

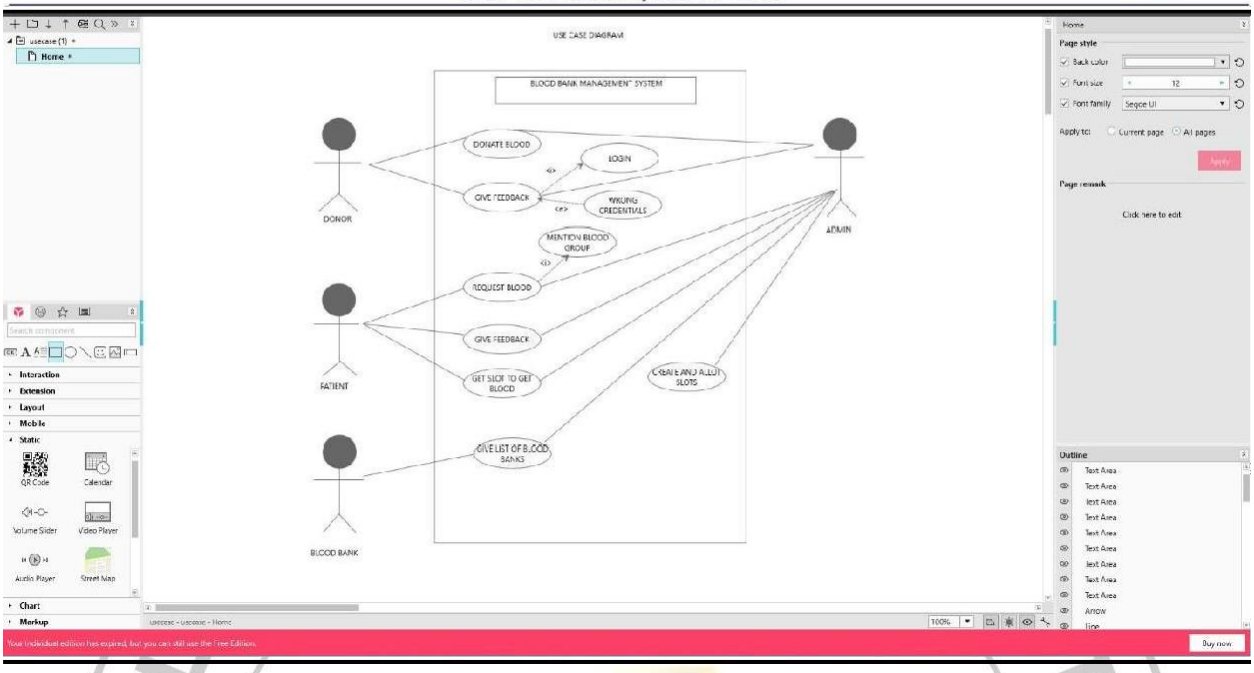
Procedure:

A Use Case model can be developed by following the steps below.

1. Identify the Actors (role of users) of the system.
2. For each category of users, identify all roles played by the users relevant to the system.
3. Identify what are the users required the system to be performed to achieve these goals.
4. Create use cases for every goal.
5. Structure the use cases.
6. Prioritize, review, estimate and validate the users.

Result and Discussion:

Q.1) What is a use-case diagram? Draw at least two use-cases for your projects.



Learning Outcomes: The student should have the ability to:

LO 1: Identify the importance of use-case diagrams.

LO 2: Draw use-case diagrams for a given scenario.

Course Outcomes: Upon completion of the course students will be able to understand and demonstrate use-case diagrams.

Conclusion: Thus, students have understood and successfully drawn use-case diagrams.

Viva Questions:

1. What is use-case diagrams used for?
2. Enumerate on the type of relationships that exists for use-case diagram.

For Faculty Use:

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment 05- Implement UML Class Diagram

Learning Objective: To implement UML class diagram for the project.

Tools: MS Word, draw.io

Theory:

Class Diagrams:

Classes are the structural units in object oriented system design approach, so it is essential to know all the relationships that exist between the classes, in a system. All objects in a system are also interacting to each other by means of passing messages from one object to another.

Elements in class diagram

Class diagram contains the system classes with its data members, operations and relationships between classes.

Class

A set of objects containing similar data members and member functions is described by a class. In UML syntax, class is identified by solid outline rectangle with three compartments which contain

- **Class name**

A class is uniquely identified in a system by its name. A textual string [2]is taken as class name. It lies in the first compartment in class rectangle.

- **Attributes**

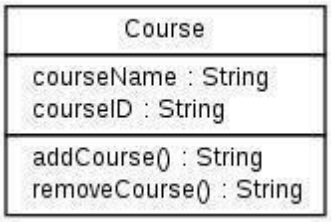
Property shared by all instances of a class. It lies in the second compartment in class rectangle.

- **Operations**

An execution of an action can be performed for any object of a class. It lies in the last compartment in class rectangle.

Example

To build a structural model for an Educational Organization, ‘Course’ can be treated as a class which contains attributes ‘courseName’ & ‘courseID’ with the operations ‘addCourse()’ & ‘removeCourse()’ allowed to be performed for any object to that class.



- **Generalization/Specialization**

It describes how one class is derived from another class. Derived class inherits the properties of its parent class.

Example

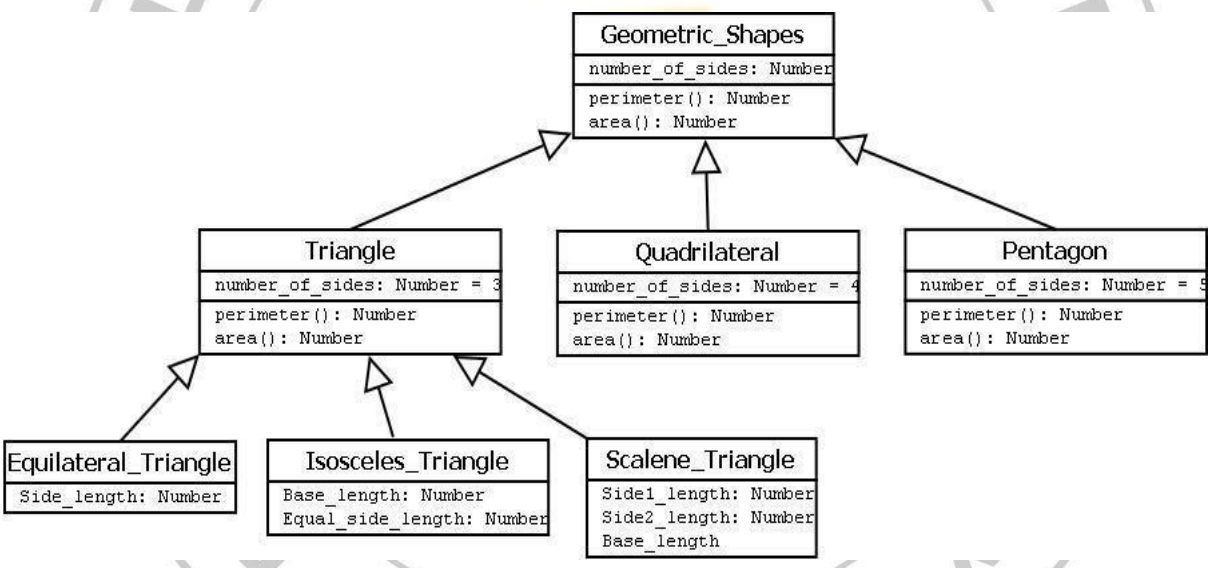


Figure-02:

Geometric_Shapes is the class that describes how many sides a particular shape has. Triangle, Quadrilateral and Pentagon are the classes that inherit the property of the Geometric_Shapes class. So the relations among these classes are generalization. Now Equilateral_Triangle, Isosceles_Triangle and Scalene_Triangle, all these three classes inherit the properties of Triangle class as each one of them has three sides. So, these are specialization of Triangle class.

Relationships

Existing relationships in a system describe legitimate connections between the classes in that system.

- **Association**

It is an instance level relationship that allows exchanging messages among the objects of both ends of association. A simple straight line connecting two class boxes represent an association. We can give a name to association and also at the both end we may indicate role names and multiplicity of the adjacent classes. Association may be uni-directional.

Example

In structure model for a system of an organization an employee (instance of 'Employee' class) is always assigned to a particular department (instance of 'Department' class) and the association can be shown by a line connecting the respective classes.

Figure-03:

- **Aggregation**

It is a special form of association which describes a part-whole relationship between a pair of classes. It means, in a relationship, when a class holds some instances of related class, then that relationship can be designed as an aggregation.

Example

For a supermarket in a city, each branch runs some of the departments they have. So, the relation among the classes 'Branch' and 'Department' can be designed as aggregation. In UML, it can be shown as in the fig. below.

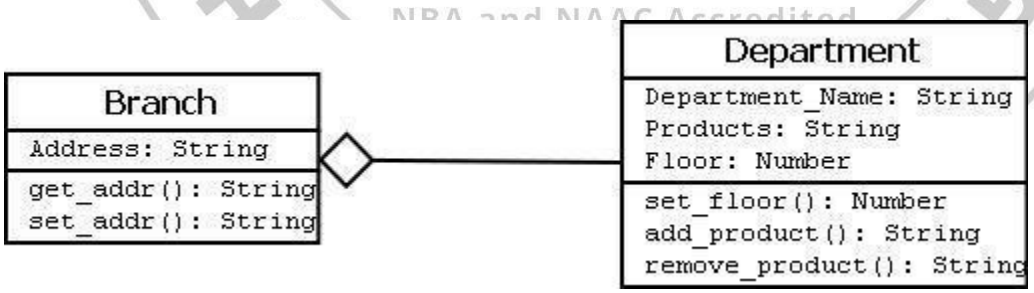


Figure-04:

- Composition

It is a strong from of aggregation which describes that whole is completely owns its part. Life cycle of the part depends on the whole.

Example

Let consider a shopping mall has several branches in different locations in a city. The existence of branches completely depends on the shopping mall as if it is not exist any branch of it will no longer exists in the city. This relation can be described as composition and can be shown as below

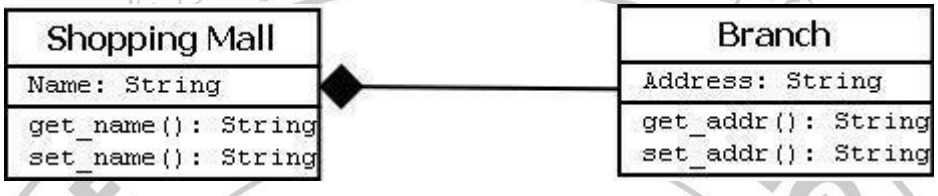


Figure-05:

- **Multiplicity**

It describes how many numbers of instances of one class is related to the number of instances of another class in an association.

Notation for different types of multiplicity:

Single instance	1
Zero or one instance	0..1
Zero or more instance	0..*
One or more instance	1..*
Particular range(two to six)	2..6

Figure-06:

Example

One vehicle may have two or more wheels

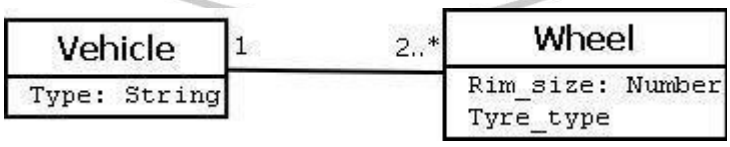


Figure-07:

Procedure:

When required to describe the static view of a system or its functionalities, we would be required to draw a class diagram. Here are the steps you need to follow to create a class diagram.

Step 1: Identify the class names

The first step is to identify the primary objects of the system.

Step 2: Distinguish relationships

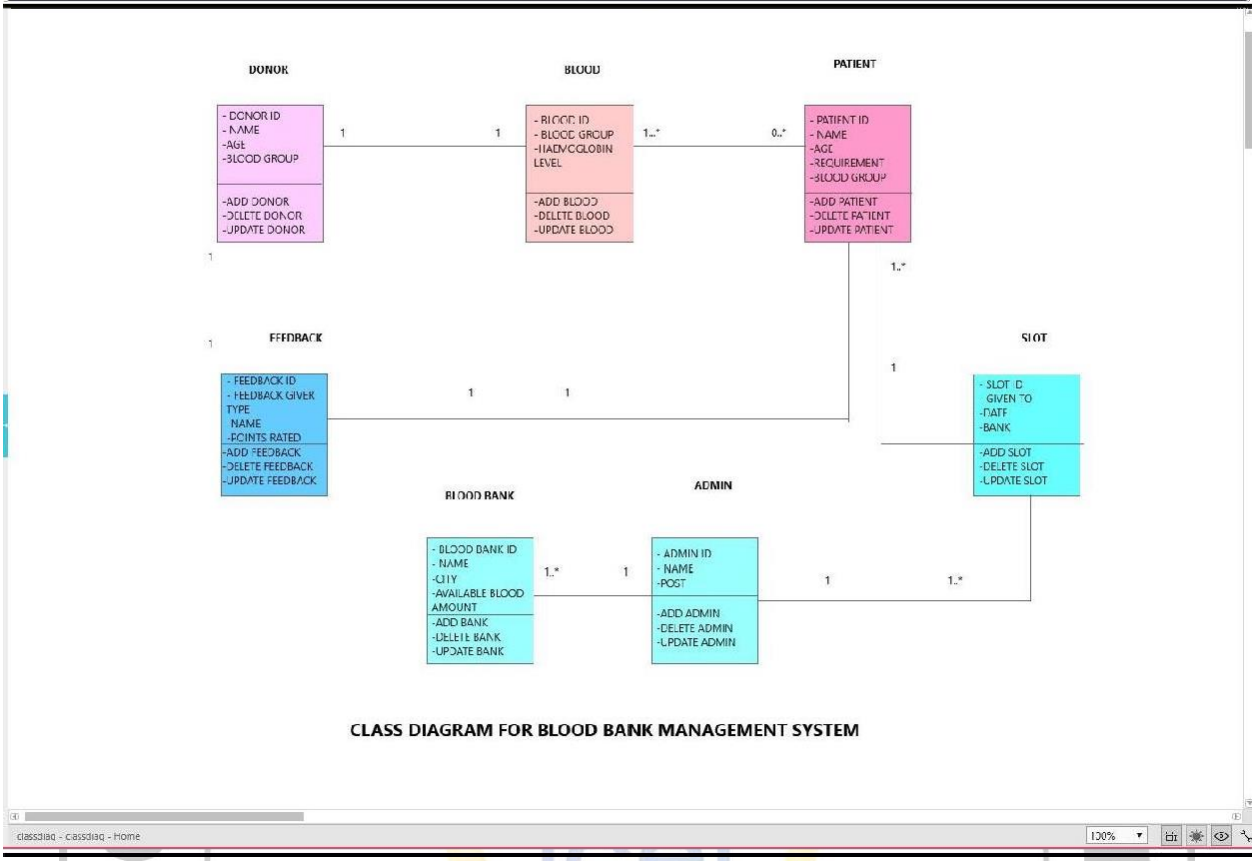
Next step is to determine how each of the classes or objects are related to one another. Look out for commonalities and abstractions among them; this will help you when grouping them when drawing the class diagram.

Step 3: Create the Structure

First, add the class names and link them with the appropriate connectors. You can add attributes and functions/ methods/ operations later.

Result and Discussion:

Q.1) What is a class diagram? Draw at least two class diagram for your projects.



CLASS DIAGRAM FOR BLOOD BANK MANAGEMENT SYSTEM

Learning Outcomes: The student should have the ability to:

LO 1: Identify the importance of class diagrams.

LO 2: Draw class diagrams for a given scenario.

Course Outcomes: Upon completion of the course students will be able to understand and demonstrate class diagrams.

Conclusion: Thus, students have understood and successfully drawn class diagrams.

Viva Questions:

1. What is a class diagrams used for?
2. Enumerate various relationships in a class diagram.

For Faculty Use:

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Experiment_06- Implement_State/Activity diagrams

Learning Objective: To implement dynamic view of a system using Activity/State diagrams.

Tools: MS Word, draw.io

Theory:

Capturing the dynamic view of a system is very important for a developer to develop the logic for a system. State chart diagrams and activity diagrams are two popular UML diagram to visualize the dynamic behavior of an information system.

In this experiment, we will learn about the different components of activity diagram and state chart diagram and how these can be used to represent the dynamic nature of an information system.

State-chart Diagrams

In case of Object Oriented Analysis and Design, a system is often abstracted by one or more classes with some well defined behavior and states. A *state chart diagram* is a pictorial representation of such a system, with all its states, and different events that lead transition from one state to another.

To illustrate this, consider a computer. Some possible states that it could have are: running, shutdown, hibernate. A transition from running state to shutdown state occur when user presses the "Power off" switch, or clicks on the "Shut down" button as displayed by the OS. Here, clicking on the shutdown button, or pressing the power off switch act as external events causing the transition.

State-chart diagrams are normally drawn to model the behavior of a complex system. For simple systems this is optional.

Building Blocks of a State-chart Diagram

State

A state is any "distinct" stage that an object (system) passes through in it's lifetime. An object remains in a given state for finite time until "something" happens, which makes it to move to another state. All such states can be broadly categorized into following three types:

- Initial: The state in which an object remain when created
- Final: The state from which an object do not move to any other state [optional]
- Intermediate: Any state, which is neither initial, nor final

As shown in figure-01, an initial state is represented by a circle filled with black. An intermediate state is depicted by a rectangle with rounded corners. A final state is represented by a unfilled circle with an inner black-filled circle.

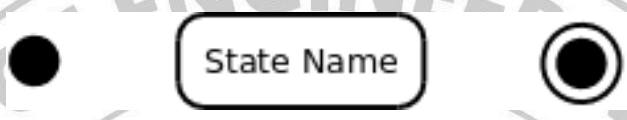


Figure-01: Representation of initial, intermediate, and final states of a state chart diagram

Intermediate states usually have two compartments, separated by a horizontal line, called the name compartment and internal transitions compartment. They are described below:

- Name compartment: Contains the name of the state, which is a short, simple, descriptive string
- Internal transitions compartment: Contains a list of internal activities performed as long as the system is in this state

The internal activities are indicated using the following syntax: action-label / action-expression. Action labels could be any condition indicator. There are, however, four special action labels:

- Entry: Indicates activity performed when the system enter this state
- Exit: Indicates activity performed when the system exits this state
- Do: indicate any activity that is performed while the system remain in this state or until the action expression results in a completed computation
- Include: Indicates invocation of a sub-machine

Any other action label identifies the event (internal transition) as a result of which the corresponding action is triggered. Internal transition is almost similar to self transition, except that the former doesn't result in execution of entry and exit actions. That is, system doesn't exit or re-enter that state. Figure-02 shows the syntax for representing a typical (intermediate) state

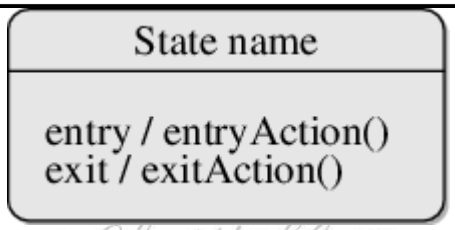


Figure-02: A typical state in a state chart diagram

States could again be either simple or composite. Here, however, we will deal only with simple states.

Transition

Transition is movement from one state to another state in response to an external stimulus (or any internal event). A transition is represented by a solid arrow from the current state to the next state. It is labeled by: event [guard-condition]/[action-expression], where

- Event is the what is causing the concerned transition (mandatory) -- Written in past tense
- Guard-condition is (are) precondition(s), which must be true for the transition to happen
- Action-expression indicate action(s) to be performed as a result of the transition

It may be noted that if a transition is triggered with one or more guard-condition(s), which evaluate to false, the system will continue to stay in the present state. Also, not all transitions do result in a state change. For example, if a queue is full, any further attempt to append will fail until the delete method is invoked at least once. Thus, state of the queue doesn't change in this duration.

Action

An action represents behavior of the system. While the system is performing any action for the current event, it doesn't accept or process any new event. The order, in which different actions are executed, is given below:

1. Exit actions of the present state
2. Actions specified for the transition
3. Entry actions of the next state

Activity Diagrams

Activity diagrams fall under the category of behavioral diagrams in Unified Modeling Language. It is a high level diagram used to visually represent the flow of control in a system. It has similarities with traditional flow charts. However, it is more powerful than a simple flow chart since it can represent various other concepts like concurrent activities, their joining, and so on.

Activity diagrams, however, cannot depict the message passing among related objects. As such, it can't be directly translated into code. These kinds of diagrams are suitable for confirming the logic to be implemented with the business users. These diagrams are typically used when the business logic is complex. In simple scenarios it can be avoided entirely.

Components of an Activity Diagram

Below we describe the building blocks of an activity diagram.

Activity

An activity denotes a particular action taken in the logical flow of control. This could simply be invocation of a mathematical function, alter an object's properties and so on. An activity is represented with a rounded rectangle, as shown in table-01. A label inside the rectangle identifies the corresponding activity.

There are two special types of activity nodes: initial and final. They are represented with a filled circle, and a filled in circle with a border respectively (table-01). Initial node represents the starting point of a flow in an activity diagram. There could be multiple initial nodes, which mean that invoking that particular activity diagram would initiate multiple flows.

A final node represents the end point of all activities. Like an initial node, there could be multiple final nodes. Any transition reaching a final node would stop all activities.

Flow

A flow (also termed as edge or transition) is represented with a directed arrow. This is used to depict transfer of control from one activity to another, or to other types of components, as we will see below. A flow is often accompanied with a label, called the guard condition, indicating the necessary condition for the transition to happen. The syntax to depict it is [guard condition].

Decision

A decision node, represented with a diamond, is a point where a single flow enters and two or more flows leave. The control flow can follow only one of the outgoing paths. The outgoing edges often have guard conditions indicating true-false or if-then-else conditions. However, they can be omitted in obvious cases. The input edge could also have guard conditions. Alternately, a note can be attached to the decision node indicating the condition to be tested.

Merge

This is represented with a diamond shape, with two or more flows entering, and a single flow leaving out. A merge node represents the point where at least a single control should reach before further processing could continue.

Fork

Fork is a point where parallel activities begin. For example, when a student has been registered with a college, he can in parallel apply for student ID card and library card. A fork is graphically depicted with a black bar, with a single flow entering and multiple flows leaving out.

Join

A join is depicted with a black bar, with multiple input flows, but a single output flow. Physically it represents the synchronization of all concurrent activities. Unlike a merge, in case of a join all of the incoming controls **must be completed** before any further progress could be made. For example, a sales order is closed only when the customer has received the product, **and** the sales company has received its payment.

Note

UML allows attaching a note to different components of a diagram to present some textual information. The information could simply be a comment or may be some constraint. A note can be attached to a decision point, for example, to indicate the branching criteria.

Partition

Different components of an activity diagram can be logically grouped into different areas, called partitions or swim lanes. They often correspond to different units of an organization or different actors. The drawing area can be partitioned into multiple compartments using vertical (or horizontal) parallel lines. Partitions in an activity diagram are not mandatory. The following table shows commonly used components with a typical activity diagram.

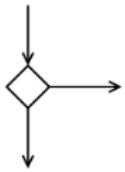
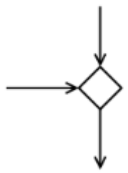
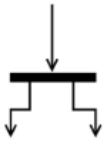
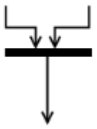
Component	Graphical Notation
Activity	An Activity
Flow	[A Flow]
Decision	
Merge	
Fork	
Join	
Note	A simple note

Table-01: Typical components used in an activity diagram

A Simple Example

Figure-04 shows a simple activity diagram with two activities. The figure depicts two stages of a form submission. At first a form is filled up with relevant and correct information. Once it is verified that there is no error in the form, it is then submitted. The two other symbols shown in the figure are the initial node (dark filled circle), and final node (outer hollow circle with inner filled circle). It may be noted that there could be zero or more final node(s) in an activity diagram.

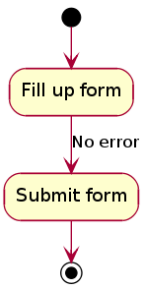


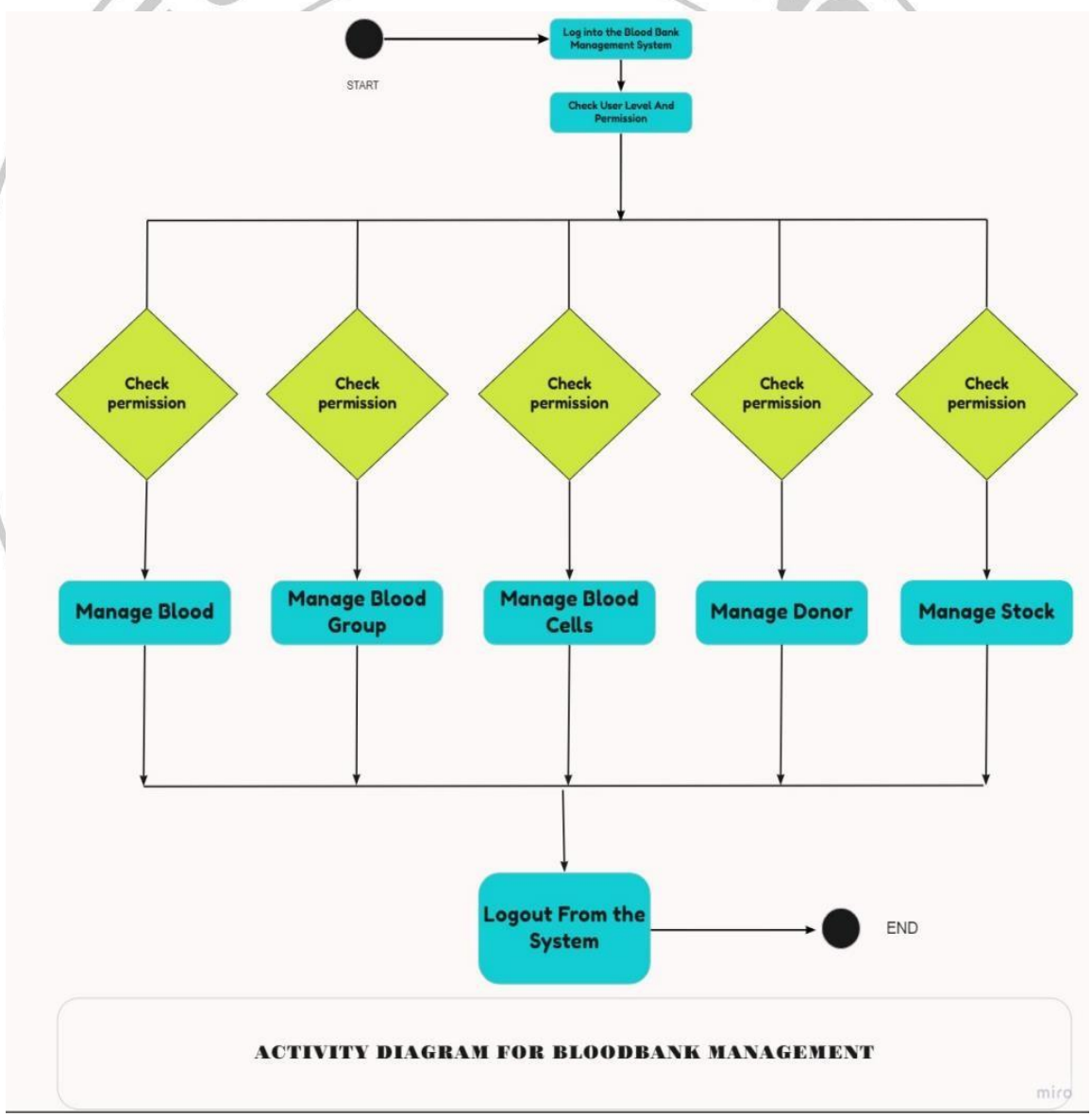
Figure-04: A simple activity diagram.

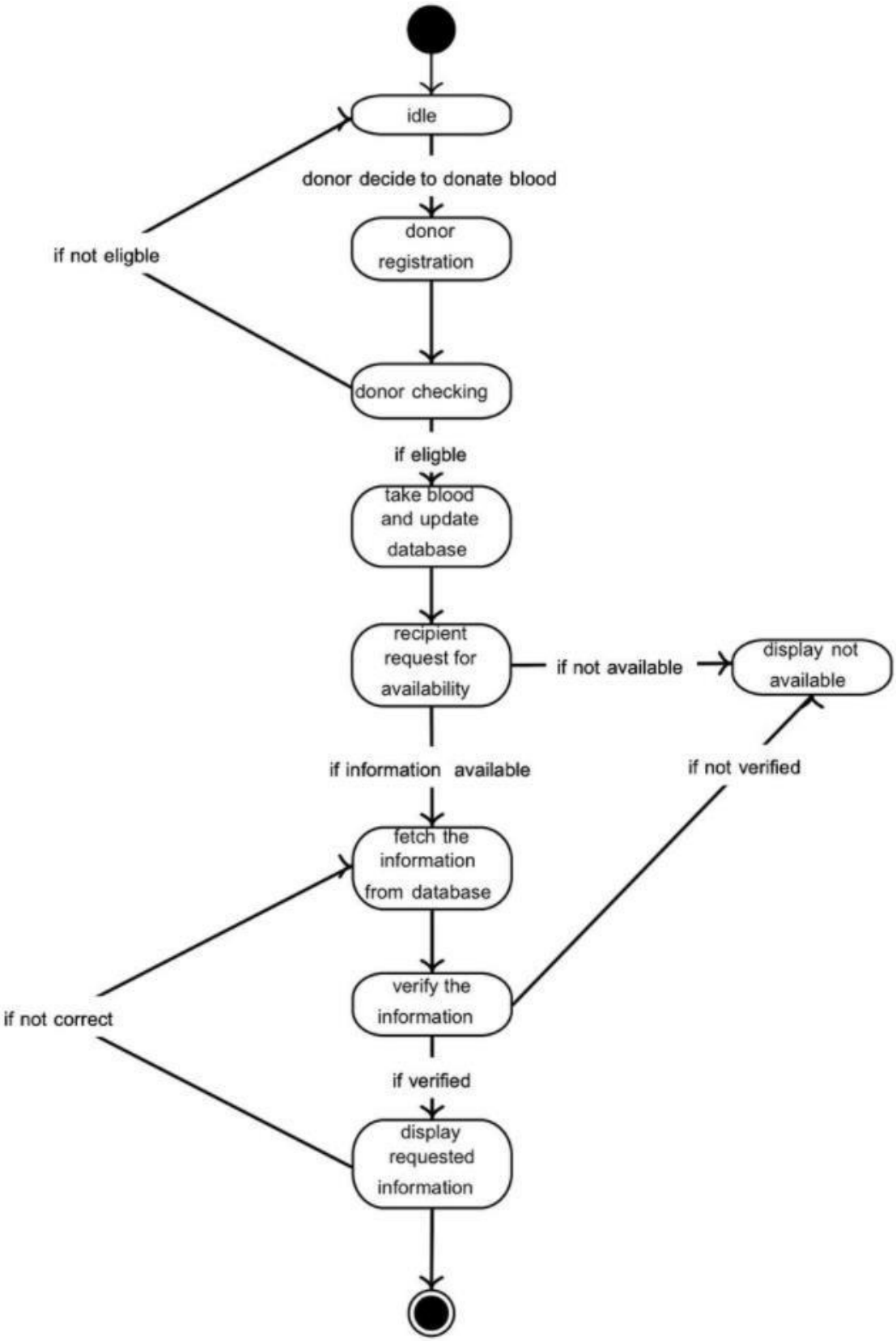
Procedure:

Guidelines for drawing State chart Diagrams

Following steps could be followed, to draw a state chart diagram:

- For the system to developed, identify the distinct states that it passes through
- Identify the events (and any precondition) that cause the state transitions. Often these would be the methods of a class as identified in a class diagram.
- Identify what activities are performed while the system remains in each state





Result and Discussion:

Q.1) what is a dynamic view of a system? Draw at least one state diagram and one activity diagram for your mini project.

Learning Outcomes: The student should have the ability to:

LO 1: Identify the importance of state diagram.

LO 2: Draw activity diagrams for a given scenario.

Course Outcomes: Upon completion of the course students will be able to understand and demonstrate state and activity diagrams.

Conclusion: Thus, students have understood and successfully drawn state and activity diagrams.

Viva Questions:

1. What is a state diagram used for?
2. Enumerate the steps to draw an activity diagram.

For Faculty Use:

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment 7: Sketch Sequence and Collaboration diagram for the project

Learning Objective: Students will able to draw Sequence and Collaboration diagram for the project

Tools: Dia, StarUML

Theory:

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence Diagram representation

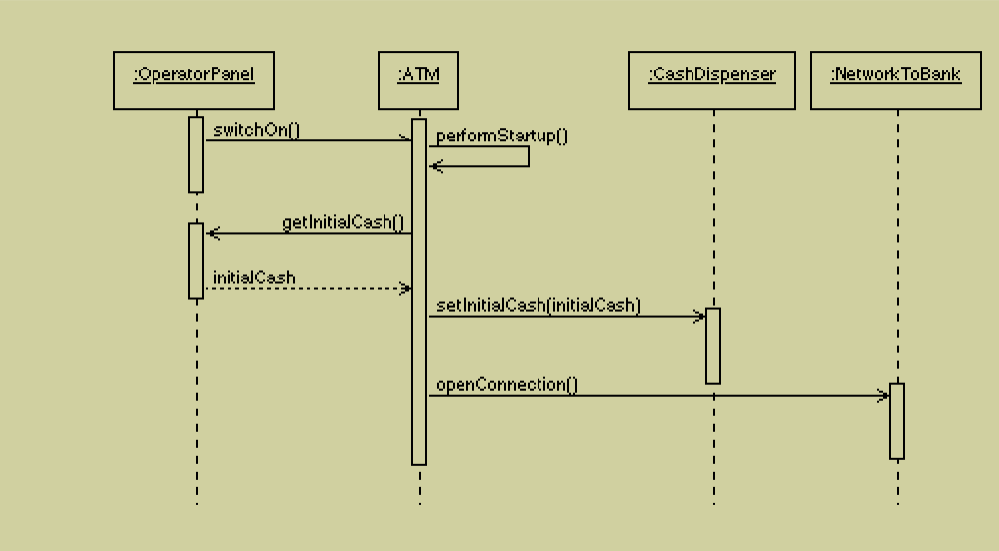
Call Message: A message defines a particular communication between Lifelines of an Interaction.

Destroy Message: Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.

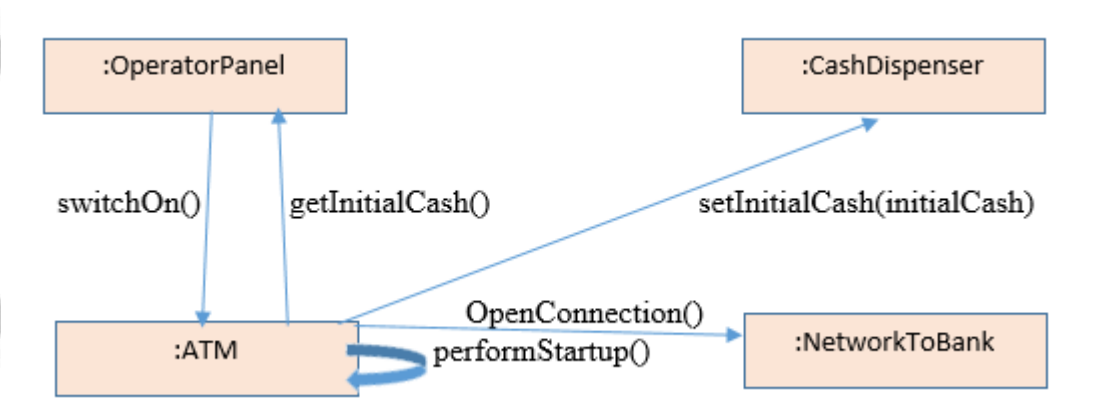
LifeLine: A lifeline represents an individual participant in the Interaction.

Recursive Message: Recursive message is a kind of message that represents the invocation of message of the same lifeline. It's target points to an activation on top of the activation where the message was invoked from.

Sequence Diagram:Example for ATM System startup



Collaboration diagram for ATM System startup

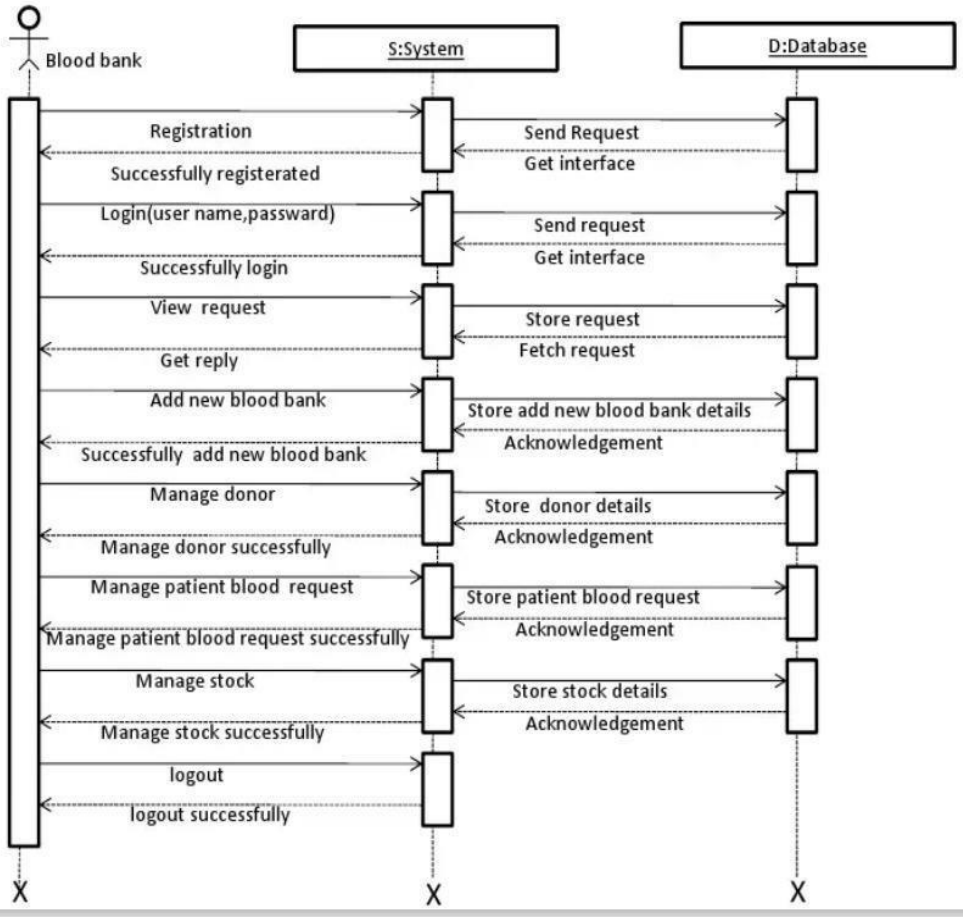


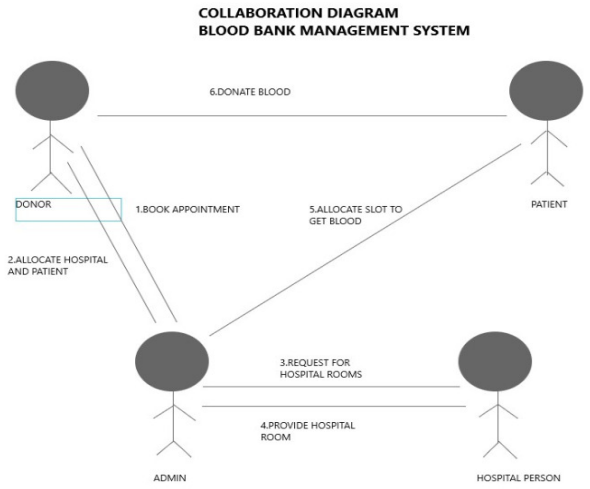
It is clear that sequence charts have a number of very powerful advantages. They clearly depict the sequence of events, show when objects are created and destroyed, are excellent at depicting concurrent operations, and are invaluable for hunting down race conditions. However, with all their advantages, they are not perfect tools. They take up a lot of space, and do not present the interrelationships between the collaborating objects very well.

A collaboration diagram, also known as a communication diagram, depicts the relationships and interactions among software objects in the UML diagrams. Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects.

Collaboration diagrams are used to visualize the structural organization of objects and their interactions. Sequence diagrams, focus on the order of messages that flow between objects.

Blood bank :


Blood bank
S:System
D:Database
Registration
Send Request
Get interface
Successfully registerated
Login(user name,passward)
Send request
Get interface
Successfully login
View request
Store request
Fetch request
Get reply
Add new blood bank
Store add new blood bank details
Acknowledgement
Successfully add new blood bank
Manage donor
Store donor details
Acknowledgement
Manage donor successfully
Manage patient blood request
Store patient blood request
Acknowledgement
Manage patient blood request successfully
Manage stock
Store stock details
Acknowledgement
Manage stock successfully
logout
logout successfully
X
X
X



Learning Outcomes: Students should have the ability to

LO1: Identify the classes and objects.
LO2: Identify the interactions between the objects
LO3: Develop a sequence diagram for different scenarios
LO4: generate the collaboration diagram

Outcomes: Upon completion of the course students will be able to draw the sequence and collaboration diagram for the project.

Conclusion:

We successfully implemented collaboration and sequence diagram.

Viva Questions:

1. **What is a sequence diagram**
2. **Difference between sequence and collaboration diagram?**
3. **What are entities in sequence diagram?**
4. **Explain its relation with the class diagram?**

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment 8: Change specification and use any SCM Tool to make different versions for the project

Learning Objective: Students will able to create versions using Github tool

Tools: Github

Theory:

Software configuration management: The traditional software configuration management (SCM) process is looked upon by practitioners as the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

Software configuration management is a part of software engineering, which focuses mainly on maintaining, tracking and controlling the changes done to the software configuration items.

Configuration management is present in all phase of software development. The configuration items can be all the objects which come as an output of the development process e.g. coding phase produces source code, exes and obj files. The various configuration items can be:

1. Source code,
2. Documents
3. Data used in the programs

In Configuration management, there can be multiple versions created for any configuration item (Source code/ documents). Each version can be identified by unique configuration or an attribute which is associated with each version. E.g. the version number.

Terminologies used in version control

1. SCI – Software configuration items, i.e. the documents and code which will be having version number and saved.

2. Repository- it is the system where all the SCIs will be stored.

3. Check in- to store the tested and qualified source code.

4. Checkout- to get a copy of the stored SCI from the repository.

5. Add – Add to the local repo and keep ready for commit

6. Commit- to save the file in repository and create a version

Advantages

1) The versions are stored in the repository; hence they are available as backups.

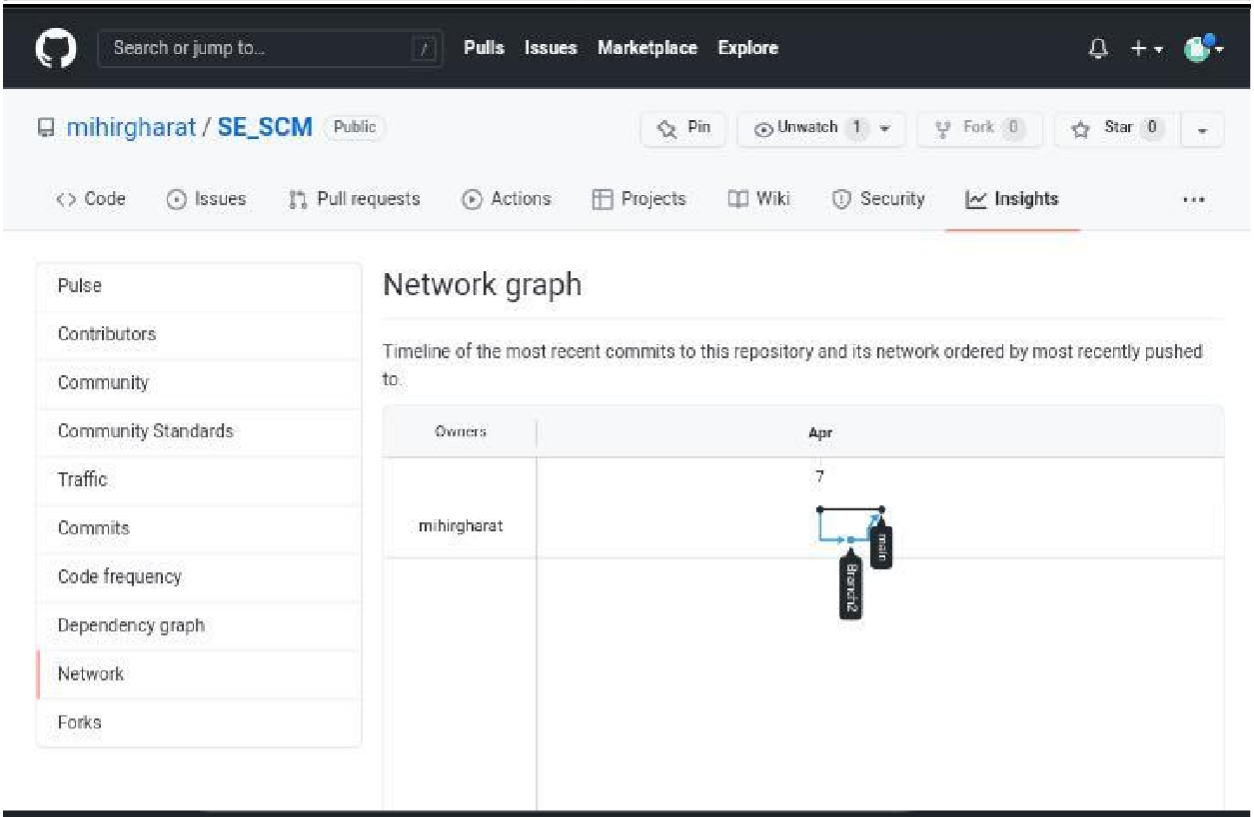
2) Multiple people can work simultaneously on same files/source code, without losing the changes made by other developers

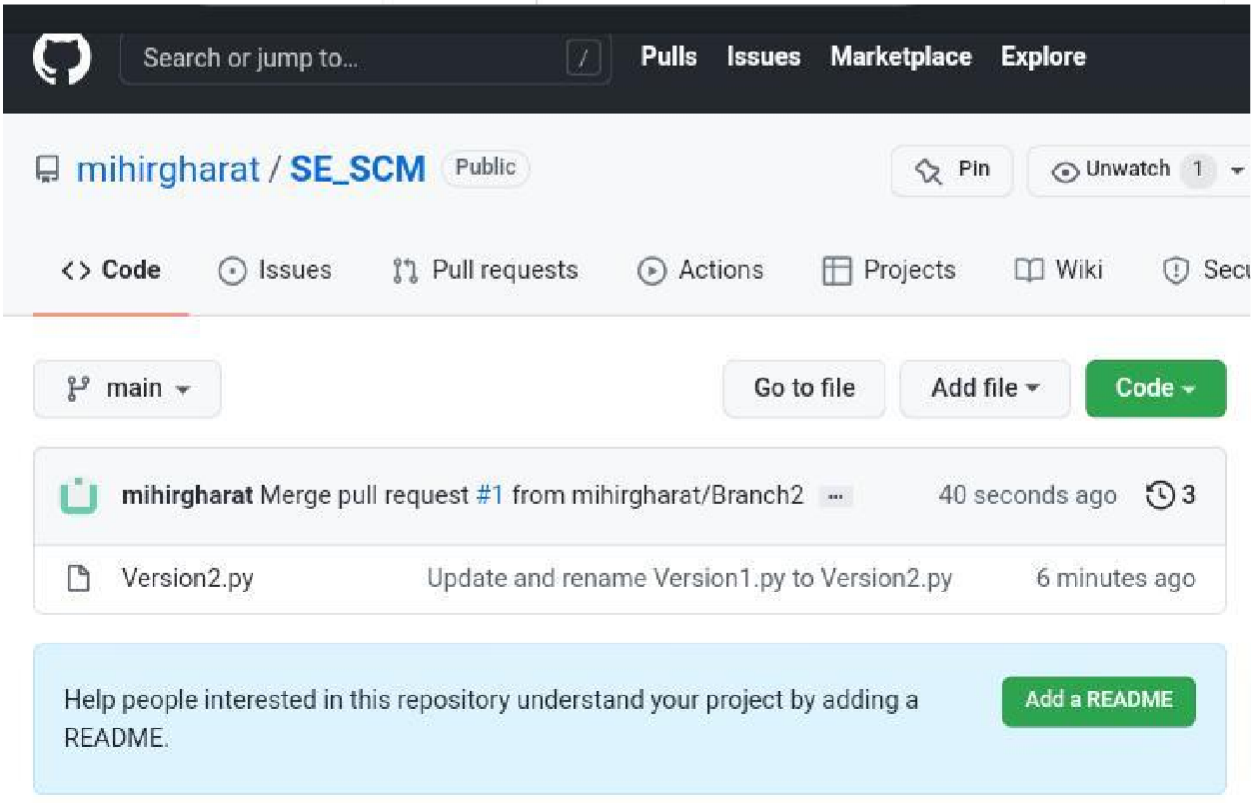
3) It is easy to find the files with specifications as a versions are stored with version numbers

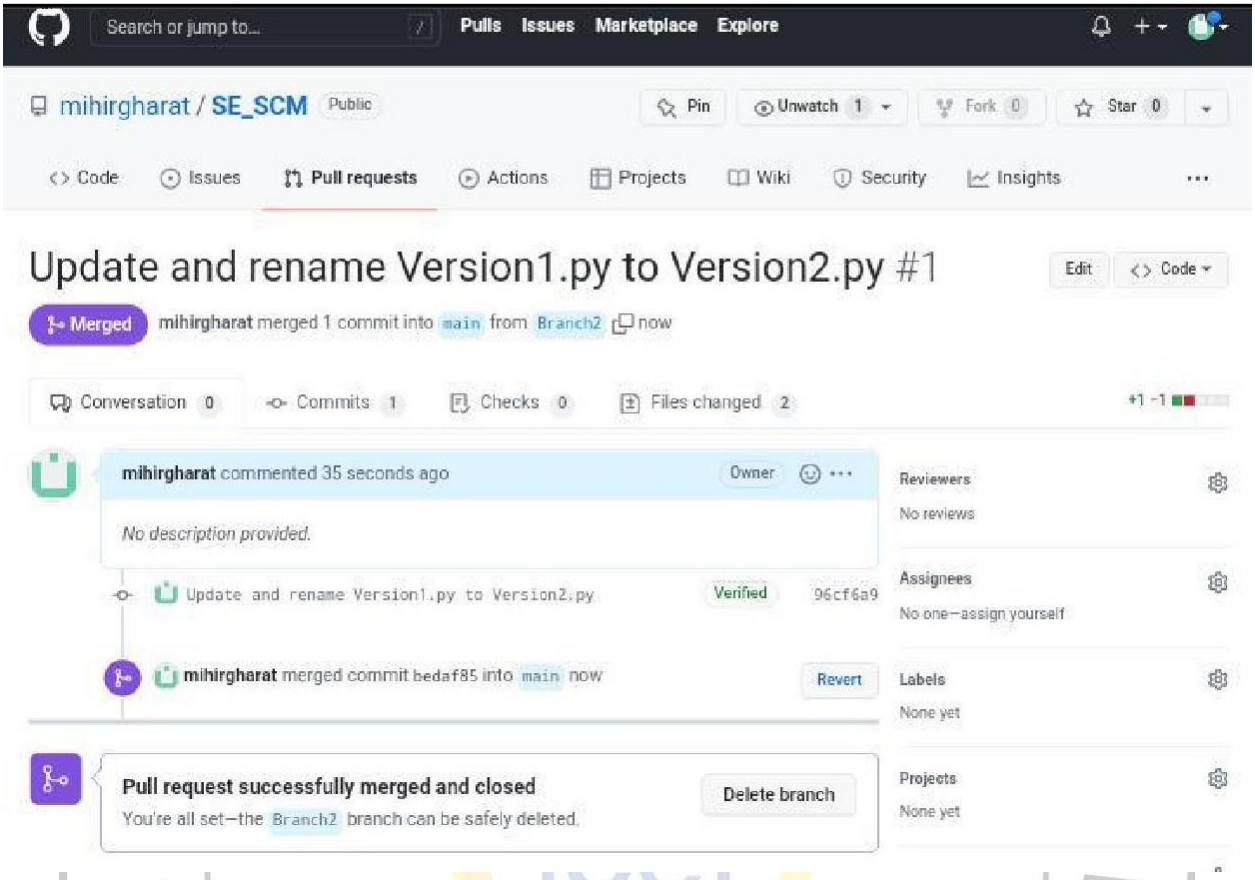
GitHub offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management for every project.

OUTPUT:

https://github.com/mihirgharat/SE_SCM.git


Search or jump to...
Pulls
Issues
Marketplace
Explore
mihirgharat / SE_SCM
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Network graph
Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.
Owners
Apr
7
mihirgharat
main
Branch2


Search or jump to...
Pulls
Issues
Marketplace
Explore
mihirgharat / SE_SCM
Public
Pin
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
main
Go to file
Add file
Code
mihirgharat Merge pull request #1 from mihirgharat/Branch2
40 seconds ago
3
Version2.py
Update and rename Version1.py to Version2.py
6 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README



Learning Outcomes: Students should have the ability to

LO1: to understand the need of doing configuration management.
LO2: Identify the dissimilarity between version and variant
LO3: provide the knowledge of the benefits of using version control
LO4: To understand the types of version control system

Outcomes: Upon completion of the course students will be able to create versions for the project.

Conclusion: Thus,we have successfully implemented and studied SCM tool github and done version updation using it.

Viva Questions:

1. **What is difference between git and Github?**
2. **What is version control? Why is it required?**
3. **What are other tools for version control?**

4. **What are different types of version control?**

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Experiment 9: Apply the knowledge of test cases for the project using white box testing.

Learning Objective: Students will able to create unit test cases

Tools: Junit

Theory:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Unit Testing:

Unit testing focuses on the building blocks of the software system, that is, objects and subsystems. The specific candidates for unit testing are chosen from the object model and the system decomposition. In principle, all the objects developed during the development process should be tested, which is often not feasible because of time and budget constraints. The minimal set of objects to be tested should be the participating objects in the use cases. Subsystems should be tested after each of the objects and classes within that subsystem have been tested individually Unit testing focuses verification effort on the smallest unit of software design—the software component or module.The unit test is white-box oriented. . In Unit testing the following are tested,

1. The module interface is tested to ensure that information properly flows into and out of the program unit under test.
2. The local data structure is examined to ensure that data stored temporarily maintains its integrity.
3. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
4. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.
5. And finally, all error handling paths are tested

Write a program to calculate the square of a number in the range 1-100

```
#include <stdio.h>
int main()
{
    int n, res;
    printf("Enter a number: ");
    scanf("%d", &n);
    if (n >= 1 && n <= 100)
    {
            res = n * n;
            printf("\n Square of %d is %d\n", n, res);
    }
    else if (n<= 0 || n > 100)
            printf("Beyond the range");
```

```
    return 0;
}
```

Sr no	Input	Output
1	-2	Beyond the range
2	0	Beyond the range
3	1	Square of 1 is 1
4	100	Square of 100 is 10000
5	101	Beyond the range
6	4	Square of 4 is 16
7	62	Square of 62 is 3844

Test Cases

Test case 1 : {I1 ,O1}

Test case 2 : {I2 ,O2}

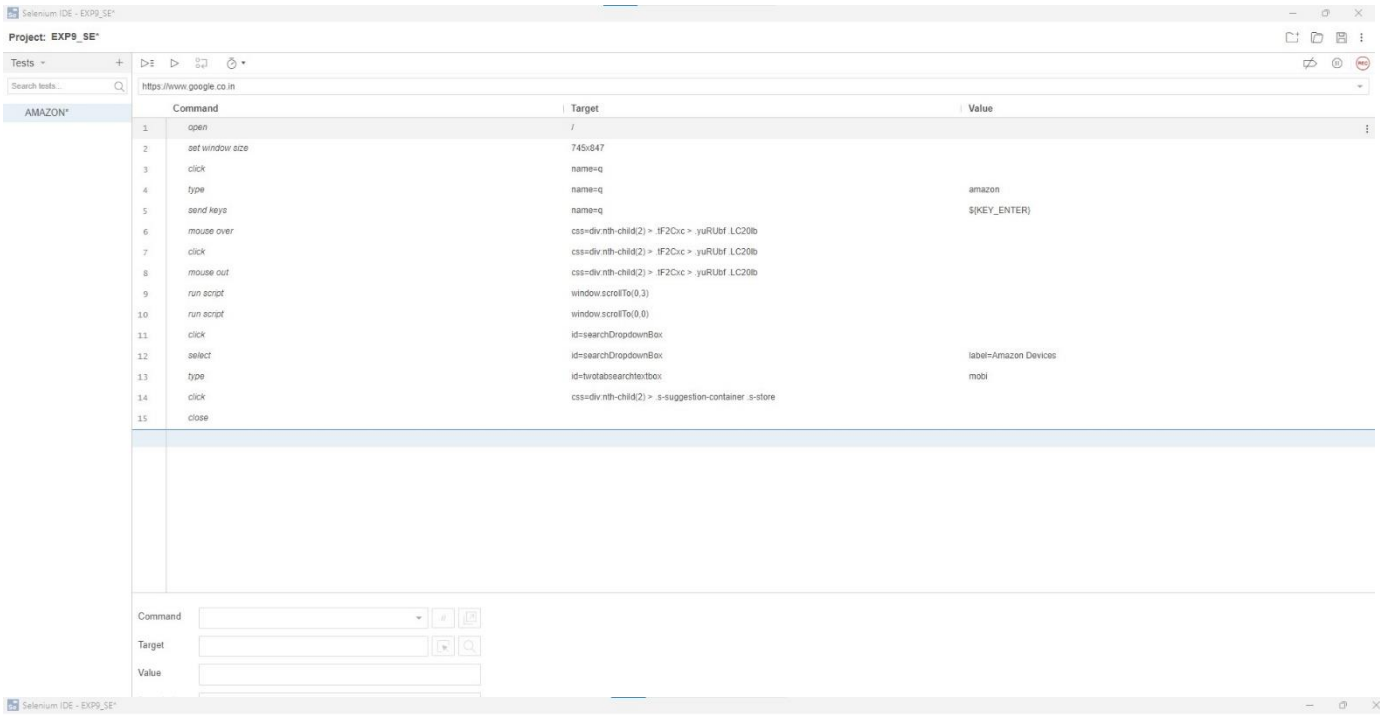
Test case 3 : {I3, O3}

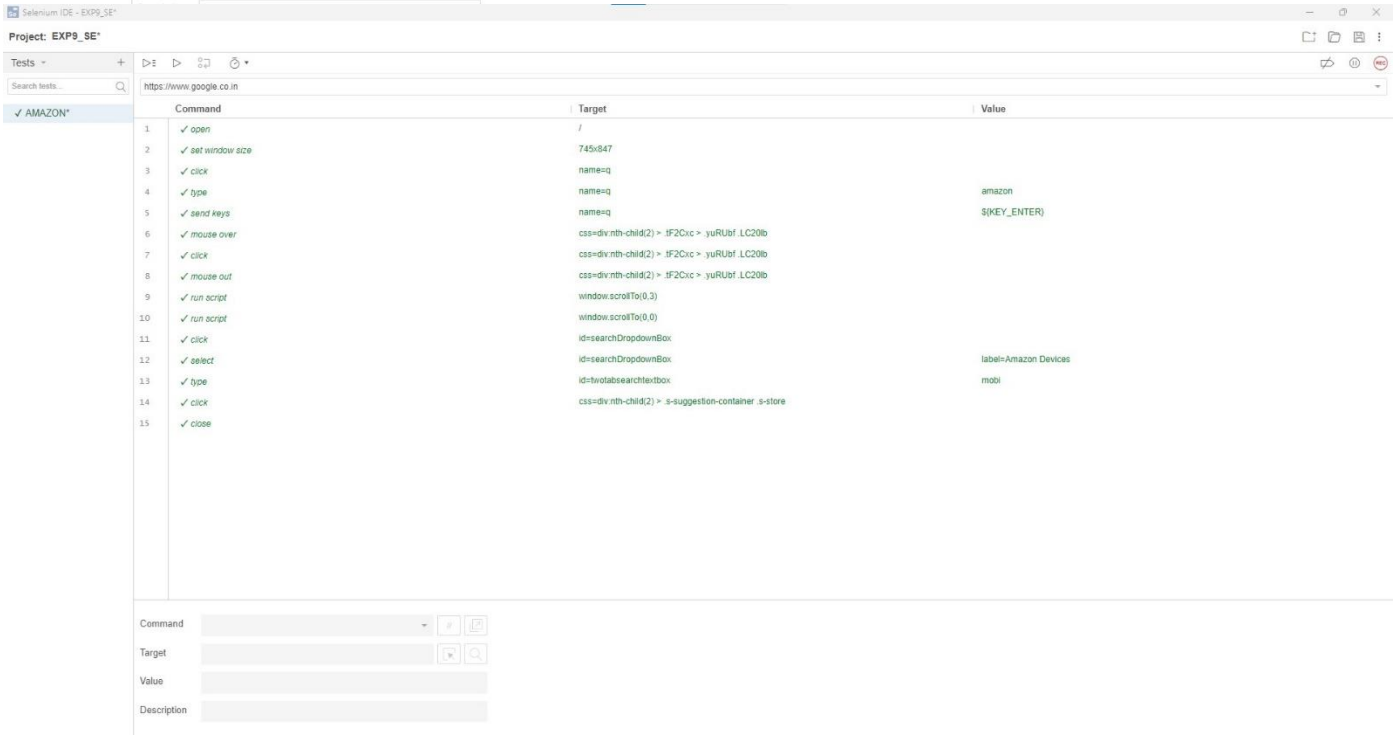
Test case 4 : {I4, O4}

Test case 5 : {I5, O5}

Test case 6 : {I6, O6}

Test case 7 : {I7, O7}





Learning Outcomes: Students should have the ability to

LO1: Students will be able to understand Software Testing Concepts and the various Software standards.
LO2: to test a software with the help of Junit
LO3: create test cases
LO4: To understand different tools for testing

Outcomes: Upon completion of the course students will be able to write test cases for the project.

Conclusion:

We successfully wrote 10 test cases for the project. We successfully tested the website in selenium

Viva Questions:

5. **What is difference between git and Github?**
6. **What is version control? Why is it required?**
7. **What are other tools for version control?**
8. **What are different types of version control?**

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

BLOOD BANK MANAGEMENT SYSTEM

By

GROUP

(Harsh Mishra, TE - B, 39)

(Mihir Gharat, TE – B , 37)

(Aaryan Pimple, TE - B, 55)

(Devang Punatar, TE - B, 59)

Under the Guidance of

Mr. Aniket Mishra
Designation

for the subject
Software Engineering

In

T.E. COMPUTER ENGINEERING

(Academic Year: 2021-22)

TABLE of CONTENTS

1. INTRODUCTION

1.1 MOTIVATION

A blood bank is a central repository in which blood is stored and managed because of blood gathered by collection and donation which are preserved for future use in blood transfusion. There are numbers of online web-based blood bank management system existing for storage of data for blood centers and hospitals to maintain information of donors, blood available, as well as transaction information. Recent research on this topic shows that manual systems as compared to computer-based information system are time consuming, laborious, and costly.

Thus, it also evinces praising computerization as a mechanism of achieving efficiency and effectiveness in this field and pointing out some crucial issues which are left aside such as proper responsibility for administration of the system. In this paper, we introduce to you a new solution blood bank management which is called the Centralized Blood Bank Repository (CBBR). With this system, donors, and other recipients such as patients and hospitals can register into the system. Donors will be able to access information about the various blood banks registered to the system as well as blood donation campaigns organized by blood banks. The blood banks are added into the system by the administrator. Recipients (Patients, hospitals, clinics, etc.) will also have access to important information like type of blood available and at which blood center. Also, continuous track of all transactions in the blood banks will be done by the system to keep efficient log of data and enhance proper report and decision making. With the new CBBR, Blood banks/ Centers, Hospitals, Patients and Blood donors will be brought together to enjoy many functionalities and access a vast amount of information, thereby making blood donation and reception a lot easier and faster.

1.2 APPLICATION

Blood Bank Management System (BBMS) is a web-based system that can assists the information of blood bag during its handling in the blood bank. With this system, the user of this system can key in the result of blood test that has been conducted to each of the blood bag received by the blood bank. The result of test will indicate whether the blood bag can be delivered to patient or not. From this system, there are several type of report that can be generated such as blood stock report, donor's gender report and the total of blood donation according to months and year. The system also can give the information to the donor about blood analysis test result for each time the donor makes contribution. Hence, BBMS will make the blood bank stock more systematic and manageable.

PROBLEM DEFINITION

The percentage of people donating blood is increasing day by day due to awareness to donate blood for those needed. The blood received must be managed thoroughly so that there will be no negative effect to the blood receiver once they received blood. From the observations and interview conducted that have been made during the user requirements phase, it was found out that there is no interaction medium between HSNZ and the public to announce their blood donation schedule. The blood donation event schedule is normally advertised to the public so that they are aware of the blood donation campaign period. At the blood house unit, the staffs and nurses only are informed about the blood donation schedule for each month on the whiteboard at the blood house. So, they are using manual way in informing the schedule. The problem arises when the space provided is not enough. The medium used to inform the staff about the schedule of the month is using whiteboard and it is written by using whiteboard marker. Therefore, the writing tends to become unclear. The public did not have knowledge about blood donation. There are brochures distributed to the donor but not to the public because they are only available at blood donation house. Hence, the public are not getting any details information about blood donation unless they go to the blood donation house. To oversee these, the BBMS interface will be constructed to cater for the blood house staff to post about the blood donation events. These details can be viewed by the public so that they know, and they can allocate some time to go and donate their blood. To ensure that the blood donation event schedule is informed among the blood house staff, there will be an interface for staff to be able to fill in details and list of location of the blood donation events for each month. The data inserted will be displayed to the other blood donation staffs such as nurse so that everyone can be notified about the blood donation event schedule even though the staffs are not available at the HSNZ. By having this function in BBMS, it is easier for the blood staff to make any correction if there are any incorrect details and make any changes if there is any changes in location or specified date. One of the factors of the public afraid to donate their blood is they believe in myths. The myths that they always believe are, if they donate their blood, they will become fat and if they donate their blood, their blood will become less in total of amount, and they will become pale. This BBMS should provide more information to educate the public so that they know blood donation will not give bad effects. By giving awareness to the public, this will increase volunteers to donate their blood.

2. TECHNOLOGY USED

3.1 Hardware and Software Requirements

System Requirements -

Operating System: Windows 7 SP1+

Processor: SSE2 instruction set support

Memory: 1GB RAM

DirectX: Version 10

Storage: 250 MB available space

3.2 Description of Library Used

The backbone of every successful management system is its ability to manage and control the smooth workflow of the various operations in the system. In the Hospital Management System (HMS) the main idea is to have a system that can manage smooth healthcare performance alongside administrative, medical, legal and financial control. The HMS is a self-contained application software that can manage the smooth working of the various services such as bed assignment, patient names, disease/ailment kind, personnel management, administrative issues. Hence there are various stakeholders involved in the proper functioning of the system.

1. **Features:**

I. ADMIN:

- Manage registration for Donors
- Manage Blood bank information like (update, delete)
- Manage Donor Request for Donor
- Mange patient Request for needy people
- Mange Inquiry form for Appropriate Reply
- Mange feedback for Appropriate Reply

II. BLOOD BANK:

- Blood bank information view/update
- View Donor information
- Manage Patient Blood Request

III. DONOR:

- Manage Donor information (profile)
- Add new Donation for Blood

IV. PATIENT:

Manage patient information (profile)

Give the request for patient for blood

2. **User Interfaces:** This includes a sample screen image and GUI. The user interface will contain.
    - Integrate appointment widgets in online Blood bank management systems enabling easy scheduling for patients.
    - Manage all branches with a single platform to access entire data across your branches too.
    - Integrate all the details into a single platform and to add discharge summary.
    - Digital Record of the Donor.
    - Automate Donation scheduling and make use of general wards efficiently through Blood bank management system.

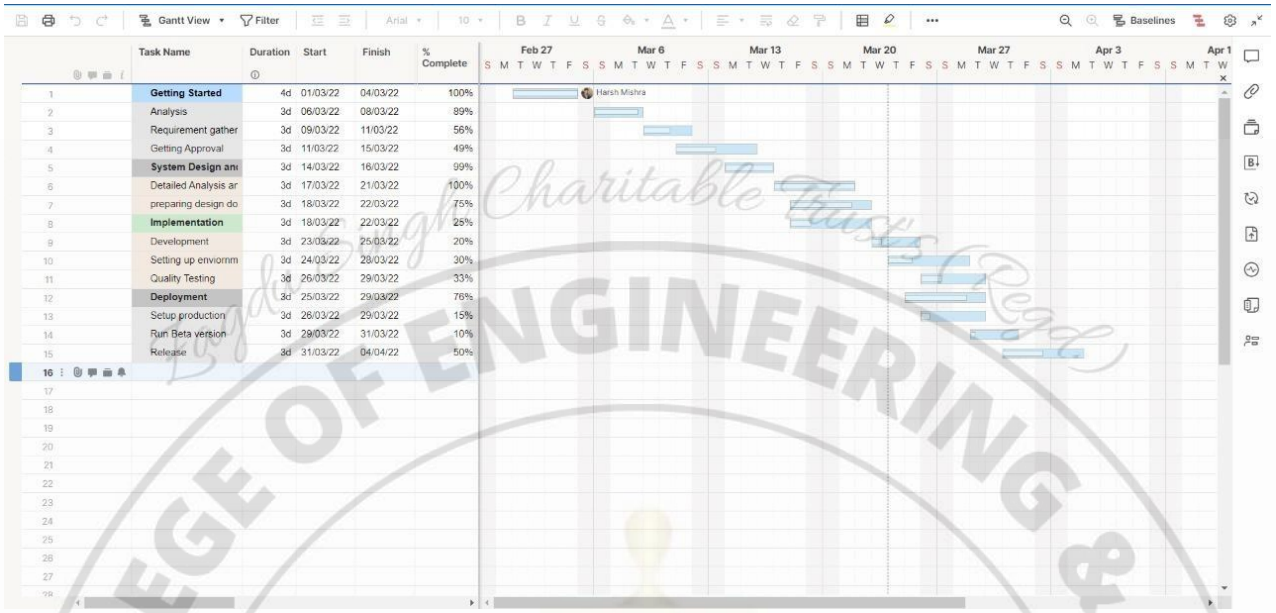
3. **Software interfaces: -** A software interface illustrates the connection between product and software components including databases, tools, operating systems, libraries and integrated components. Identifies and describes the purpose of each data item or message coming in the system. A software interface characterizes all the services needed and nature of communication.

4. **Hardware interfaces:**
    - Laptop/Desktop PC - Purpose of the pc is to give information when Donor asks information about doctors, prescriptions or available laboratory tests.
    - Display Unit (LED/LCD Monitor/TV) - To display information about the BBMS and for displaying the channel number when the patients come to see their consultants.
    - Laser Printer - For printing bills and reports.
    - Wi-Fi router - To be used for internetwork operations inside of a hospital and simply data transmission from pc to server.

5. It is of utmost importance that the BBMS deployed has a very user-friendly OS to remove any form of ambiguity when it comes to the proper management of the services. By doing so there will be fewer hurdles to deal with while carrying out the daily workflow activities. Having an easy to operate OS also ensures that other people within the organization can also perform tasks with ease and training of newer employees is not tedious.

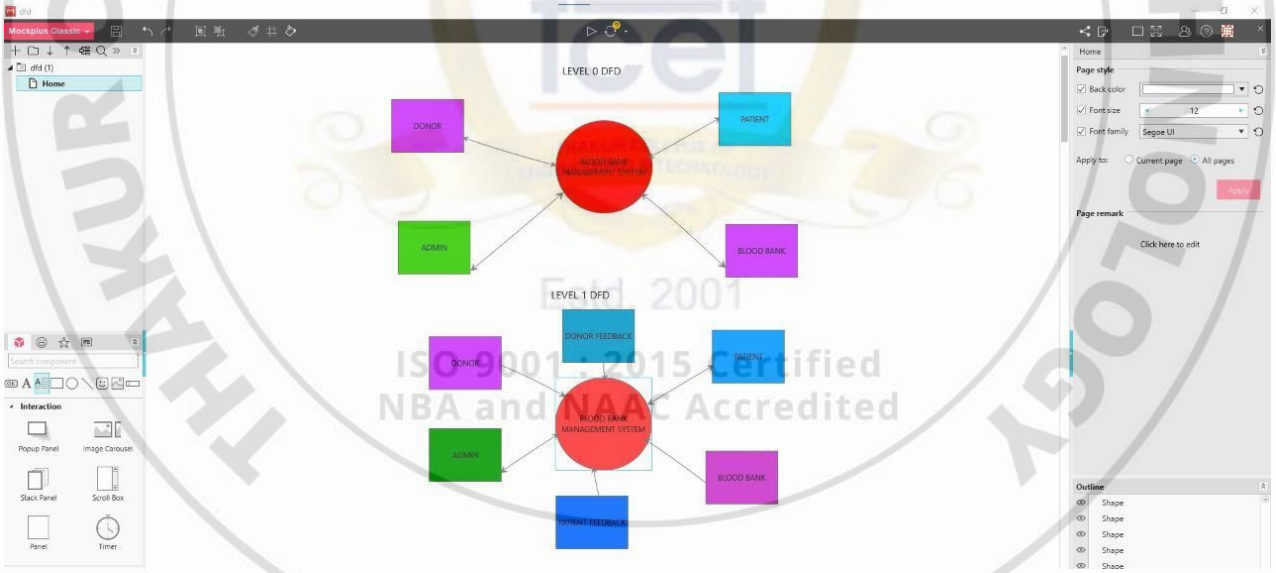
4. IMPLEMENTATION

TIMELINE DIAGRAM

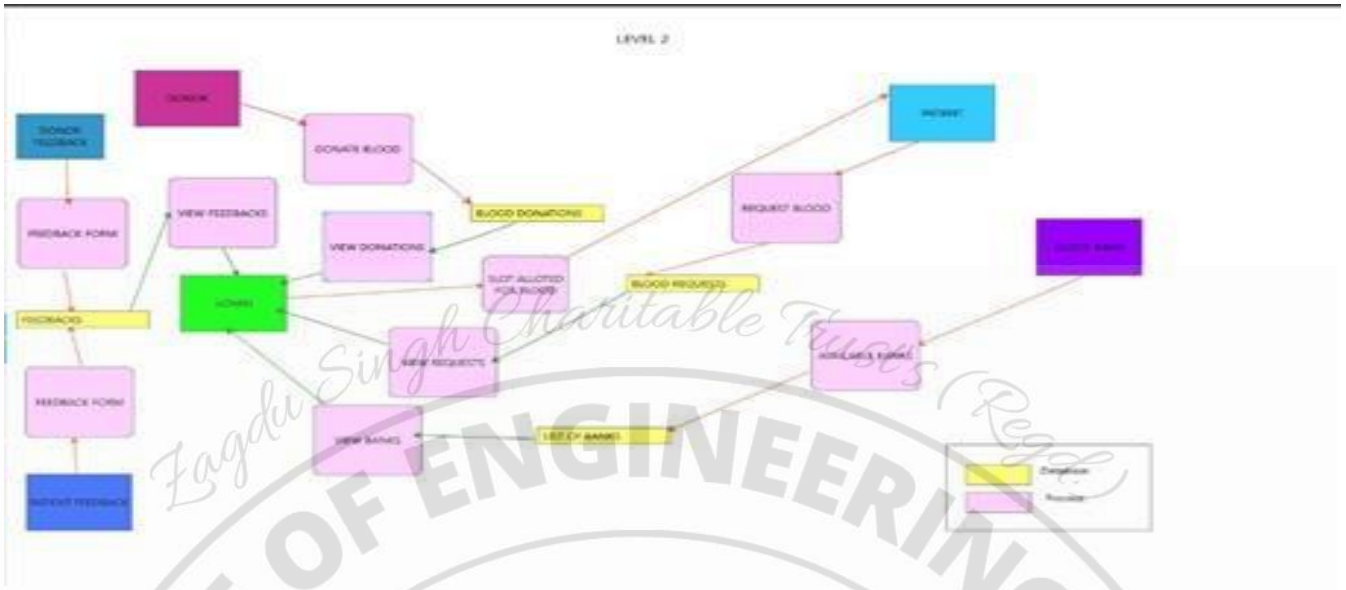


1. DFD Diagram

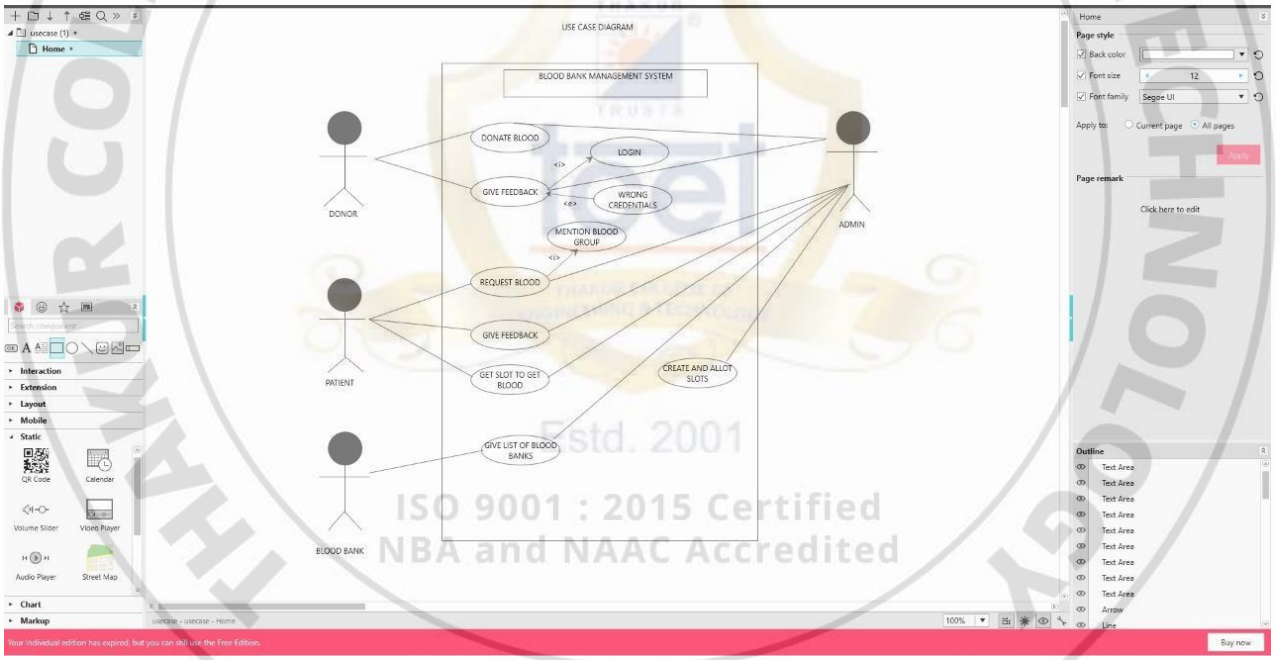
- **Level 0 and Level 1**



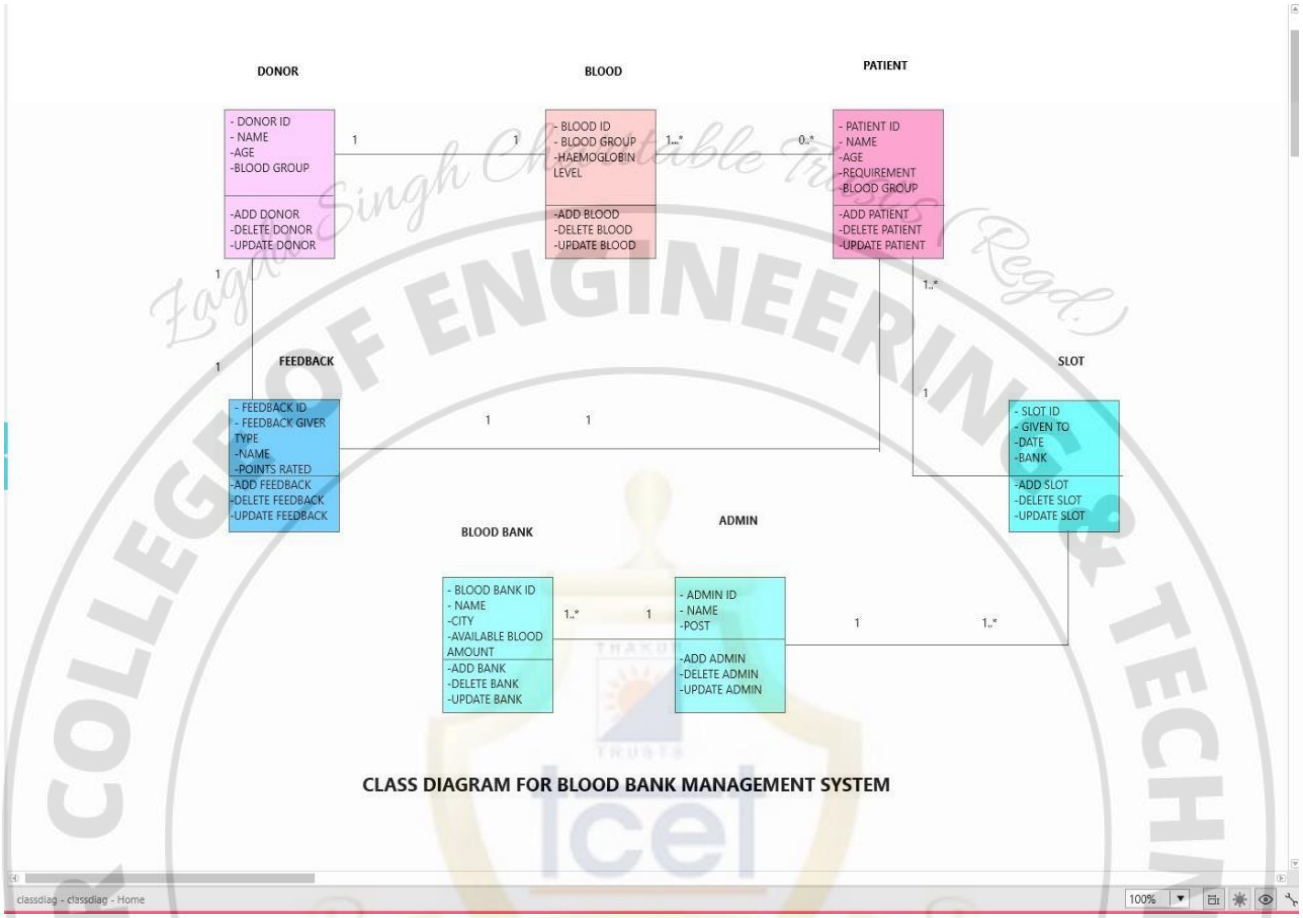
- **Level 2**



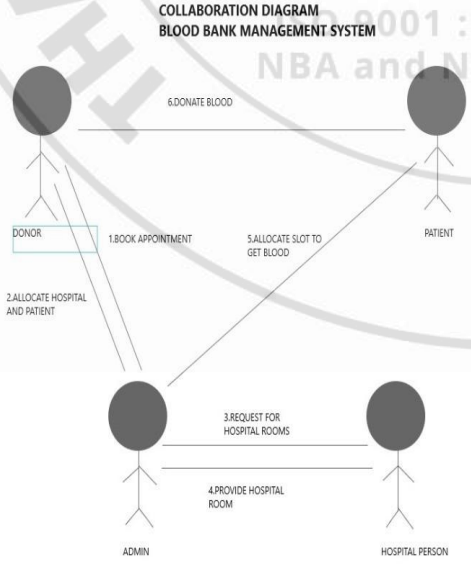
2. UML Diagram



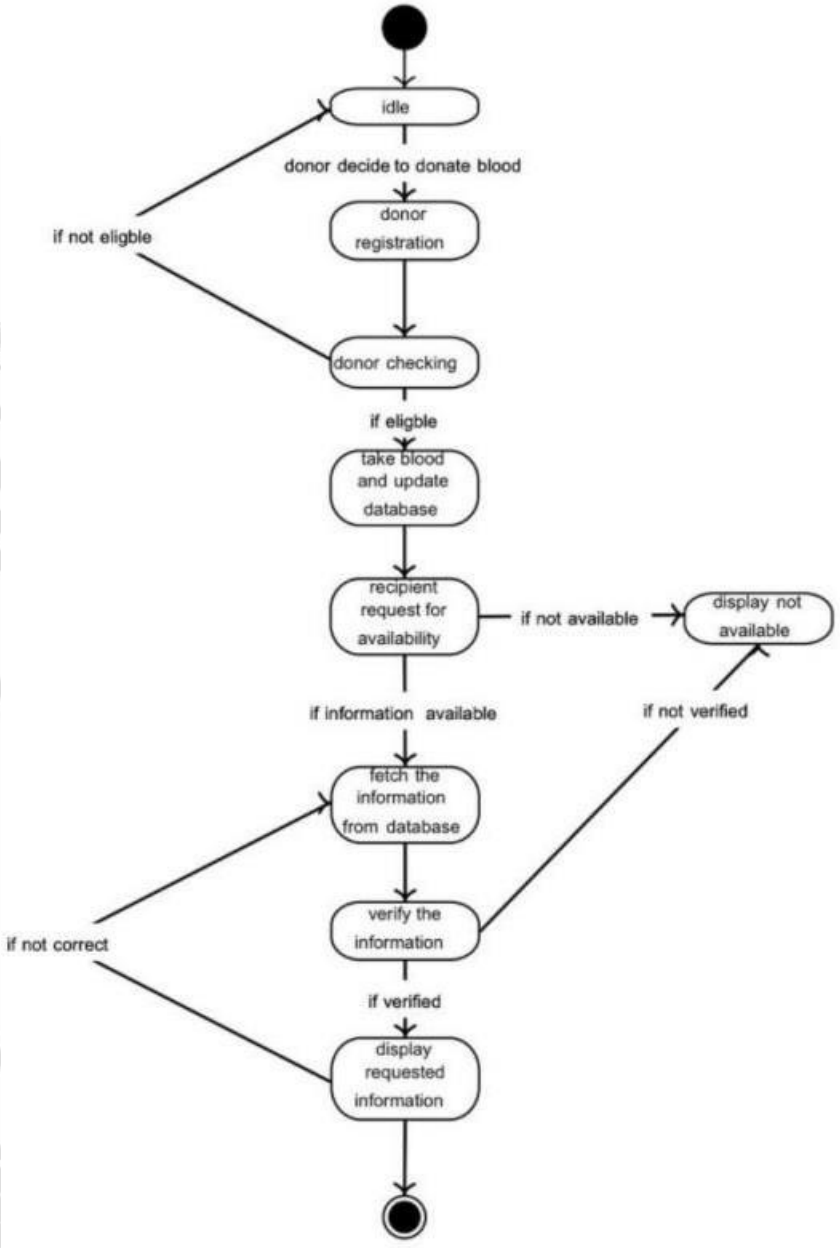
3. Class Diagram:



Collaboration Diagram



4. State Transition Diagram



CONCLUSION AND FUTURE SCOPE

BBMS is a management system that is developed to manage blood bank in HSNZ. The BBMS had been developed in accordance with HSNZ user requirement. This is to make sure that the management of the blood stock became effective, systematic, and meeting user requirements. The functional services provided in the current version are profile management, blood stock management, and blood analysis management. In the next phase we will develop a portable and modified BBMS version based on android OS. The modified version will include a user self-monitoring health profile history.

To conclude, the blood bank management system is the inevitable part of the lifecycle of the modern medical institution.

Developing the blood bank system software is a great opportunity to create the distinct, efficient, and fast delivering healthcare model. Implementation of blood bank management system project helps to store all the kinds of records, provide coordination and user communication, implement policies, improve day-to-day operations, arrange the supply chain, manage financial and human resources, and market hospital services.

LIST OF REFERENCES

Vikas Kulshreshtha, Sharad Maheshwari. (2011).”Blood Bank Management Information System in India”, International Journal of Engineering, 1,2, 260-263.

Rational Unified Process, Best Practices for Software Development Teams. (2012).

Core Workflows Retrieved from www.ibm.com/developerworks/rational/.../1251_bestpractices Noushin Ashrafi, & Hessam Ashrafi. (2008).

Object Oriented Systems Analysis and Design, Pearson Higher Ed USA. Lions Blood Bank & Research Foundation. (2012).

Retrieved from http://www.lionsbloodbank.net/ Blood Bank India. (2012). Retrieved from http://www.bloodbankindia.ne

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

FA-1 SE

Page No: 01
Date:

~~team Name members~~
MIHIR GHARAT TE COMP B
ROLL NO: 37
HARSH MISHRA TE COMP B · 39
AARYAN PIMPLE TECOMP B - 55
DEVANG PUNATAR TECOMP B - 59

MIHIR GHARAT TECOMP B 37

Explain the software project estimation
i) LOC ii) FP
Software project estimations are used in businesses & industry to estime size of softwar, the efforts needed for development and so on. It basically involves: emphirical, decomposition or base estimations.

i) LOC: LOC means total number of lines in soure code.
Units: KLOC: Thousand LOC
NLOC: Non-comment LOC
KDSI: Thousands of delivered source instructions.
The size can be estimated by comparing with existing systems.
*Advantages:
↳ Universally accepted & used in cocomo
↳ Simple to find

*Disadvantage:
↳ different programming language need will account different LOC values.

pp) FP - Function Point counts number of functions operating in a software.
The formula used here is

$$FP = (0.65 + 0.01 * \text{Count total}) *$$
$$FP = (0.65 + 0.01 * \Sigma F) * \text{Count total}$$

Here
ΣF = the sum of scores (out of 5) alloted to each of predefined 14 questions pertaining to software
Count total: The total of measured categorical values pertaining to number of inputs, outputs, external interface, inquiries, files.

DEVANG PUNATAR
TE COMP B 59

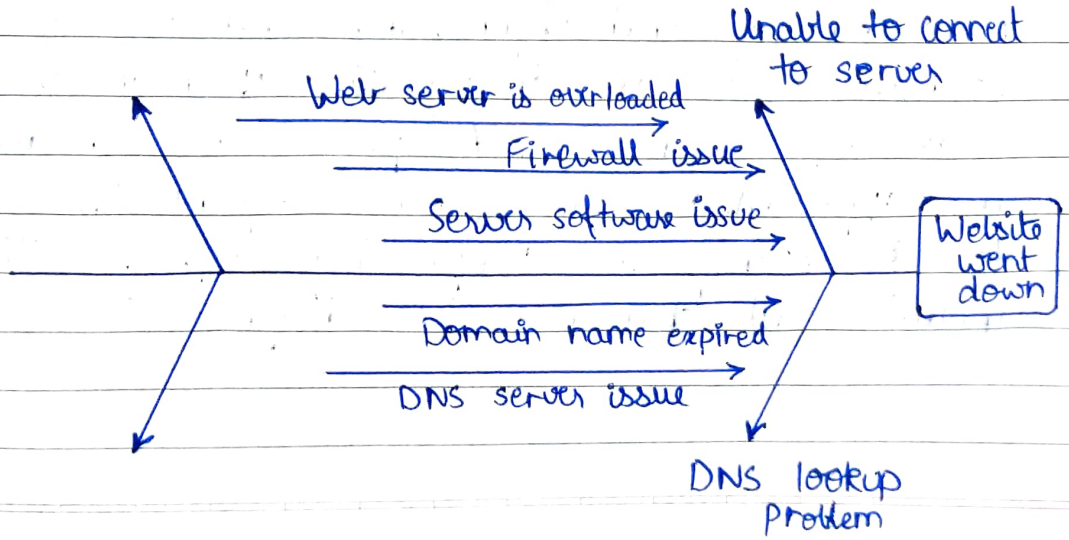
SE FA-1

Q.2) i) Fish Bone Diagram :

A fishbone diagram is a visualization tool for categorizing the potential causes of a problem. This tool is used in order to identify a problem's root causes. Typically used for root cause analysis, a fishbone diagram combines the practice of brainstorming with a type of mind map template. It should be efficient as a test case technique to determine cause and effect.

A fishbone diagram is useful in product development and troubleshooting processes, typically used to focus a conversion around a problem. After the group has brainstormed all the possible causes according to their level of importance and hierarchy. The name comes from the diagram's design, which looks much like a skeleton of a fish. Fishbone diagrams are typically worked right to left, with each large "bone" of the fish branching out to include smaller bones, each containing more detail.

Example of a Fishbone Diagram:



ii) Capability Maturity Model (CMM):

To determine an organization's current state of process maturity, the SEI uses an assessment that results in a five point grading scheme. The grading scheme determines compliance with a capability maturity model (CMM) that defines key activities required at different levels of process maturity. The levels that are defined in the following manner.

- Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

- Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

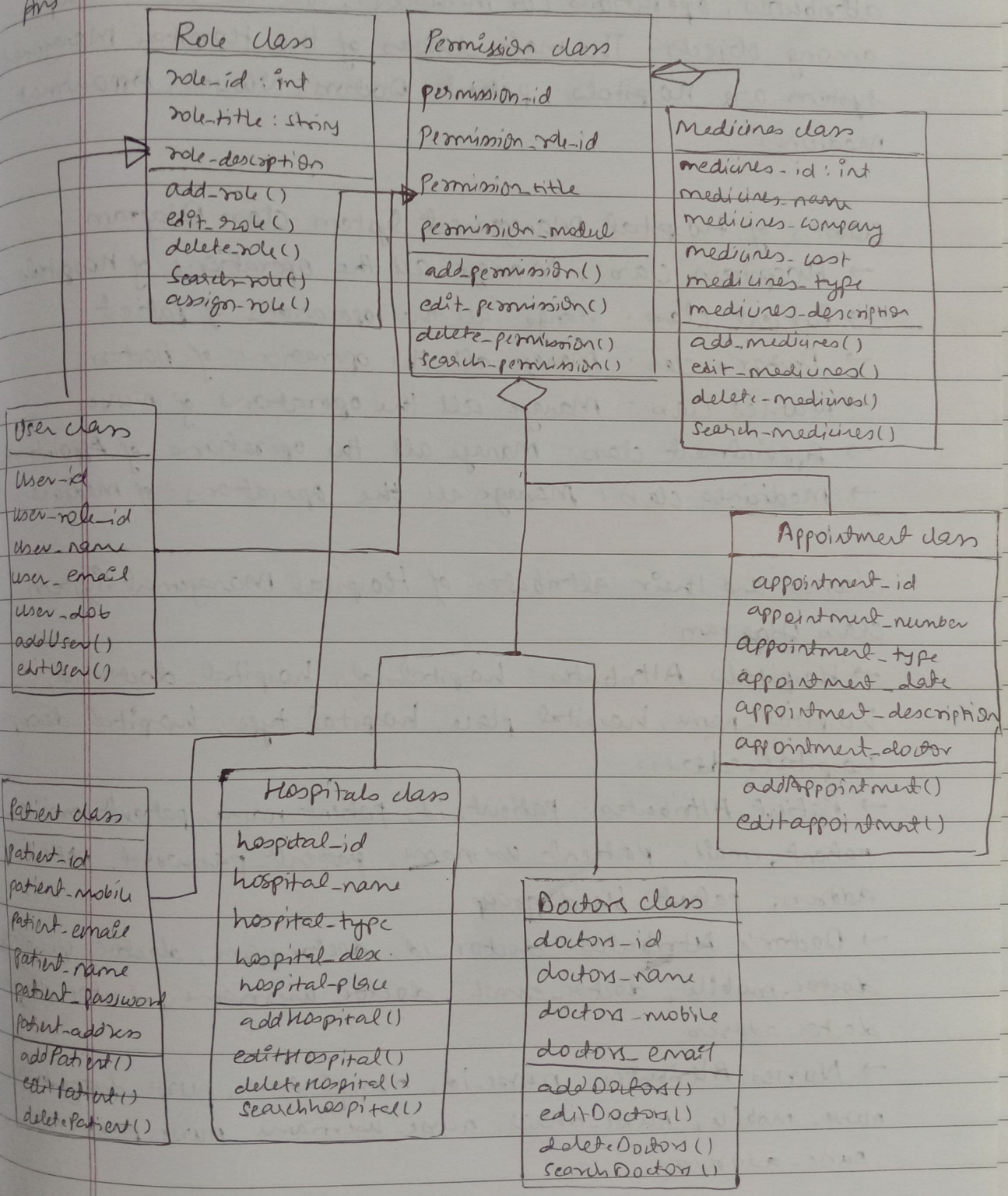
- Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2.

- Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.

- Level 5: Optimizing. Continuous process improvent is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.

Q3 Draw and explain class diagram for Hospital Management System.

Ans



Hospital Mangement System class Diagram describes the structure of a hospital management system classes, their attributes, operations (or methods), and the relationships among objects. The main classes of the Hospital Management system are Hospitals, patient, Doctors, Nurses, Appointment, Medicines.

Classes of Hospital Mangement System class Diagram
→ Hospitals class : Manage all the operations of Hospitals
→ Patient class : mange all the operations of patient
→ Doctor's class : Manage all the operations of Doctors
→ Nurses class : Manage all the operations of Nurses
→ Appointment class : Manage all the operations of Appointments
→ Medicines class : Manage all the operations of Medicines

Classes and their attributes of Hospital Management System class Diagram :
→ Hospitals Attributes : hospital_id, hospital_doctor_id, hospital_name, hospital_place, hospital_type, hospital_descriptio hospital_address
→ Patient Attributes : patient_id, patient_name, patient_mobile, patient_email, patient_username, patient_password, patient_address, patient_blood_group
→ Doctor's Attributes : doctor_id, doctor_name, doctor_specialist, doctor_mobile, doctor_email, doctor_username, doctor_password doctor_address
→ Nurses Attributes : nurse_id, nurse_name, nurse_duty_hour, nurse_mobile, nurse_email, nurse_username, nurse_password nurse_address

→ Appointment's attributes: appointment_id, appointment_doctor_id, appointment_number, appointment_type, appointment_date, appointment_description

→ Medicines Attributes: medicine_id, medicine_name, medicine_company, medicine_composition, medicine_cost, medicine_type, medicine_dose, Medicine_description.

classes and methods of Hospital management system class diagram:

→ Hospital methods: addHospital(), editHospital(), deleteHospital(), updateHospital(), searchHospital().

→ Patients methods: addPatient(), editPatient(), deletePatient(), updateHospital(), searchHospital().

→ Doctors Methods: addDoctors(), editDoctors(), deleteDoctors(), updateDoctors(), searchDoctors()

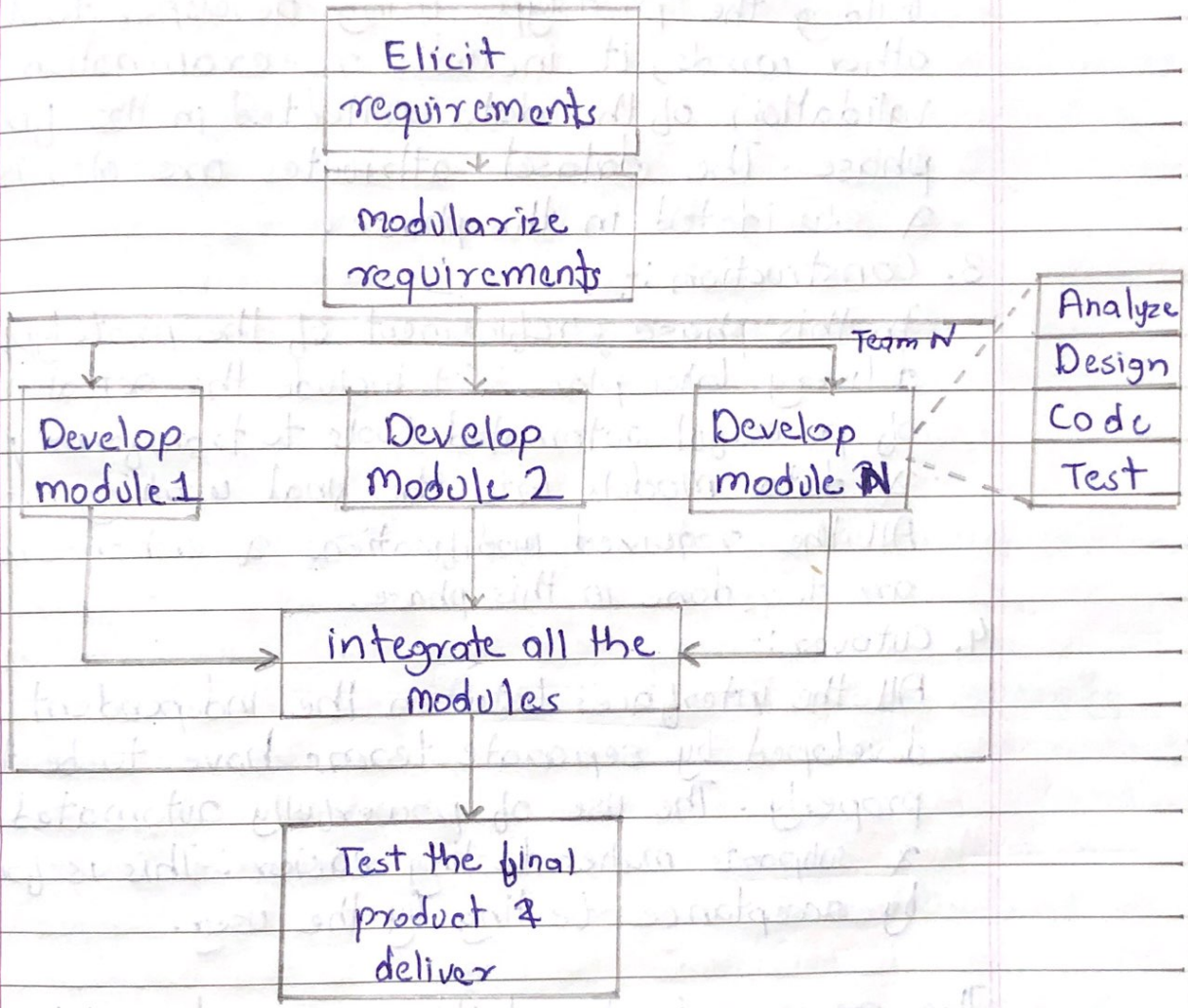
→ Nurse methods: addNurse(), editNurse(), deleteNurse(), updateNurse(), searchNurse()

→ Appointment methods: addAppointments(), editAppointments(), deleteAppointment(), updateAppointment, SearchAppointments()

→ Medicines methods: addMedicines(), editMedicines(), deleteMedicines(), updateMedicines(), saveMedicines().

Q4 Explain RAD & Devops briefly.

Ans RAD (Rapid application development model.



The model consists of 4 basic phases:-

1. Requirement planning :-

It involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it & then processing it to form final refined model.

Date : / /
Page No.:

2. User description :-
This phase consists of taking user feedback & building the prototype using developer tools. In other words, it includes re-examination & validation of the data collected in the first phase. The dataset attributes are also identified & elucidated in this phase.

3. Construction :-
In this phase, refinement of the prototype & delivery takes place. It include the actual use of powerful automated tools to transform process & data models into the final working product. All the required modifications & enhancements are too done in this phase.

4. Cutover :-
All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools & subparts makes testing easier. This is followed by acceptance testing by the user.

The process involves building a rapid prototype, delivering it to the customer & the taking feedback. After validation by the customers, SRS document is developed & the design is finalised.

Date : / /
Page No.:

DevOps:

The DevOps is a combination of two words, one is software development, & second is operations. This allows a single team to handle the entire application lifecycle, from development to testing, deployment & operations. DevOps help you to reduce the disconnection between software developers, quality assurance (QA) engineers, & system administrators. DevOps promotes collaboration between development & operations team to deploy code to production faster in an automated & repeatable way.

DevOps architecture features:

here are some key features of DevOps architecture, such as:

(i) Automation:

Automation can reduce time consumption, especially during the testing & deployment phase. The productivity increases & release are made quicker by automation. This will lead in catching bugs quickly so that it can be fixed easily. For contiguous delivery, each code is defined through automated tests, cloud-based services, & builds. This promotes production using automated deploys.

(ii) Collaboration:

The development & operations team collaborates as a DevOps team, which improves the cultural model as the team become more productive with their productivity, which strengthens accountability & ownership. The team share their responsibilities & work closely in sync, which in turn makes the deployment to production faster.

Date : / /
Page No.:

(iii) Integration :
Applications need to be integrated with other components in the environments. The integration phase is where the existing code is combined with new functionality & then tested. Continous integration & testing enable continous development. The frequency in the releases & micro-services leads to significant operational challenges. To overcome such problems, continous integration & delivery are implemented to deliver in a ~~quik~~ quicker, safe, & reliable manner.

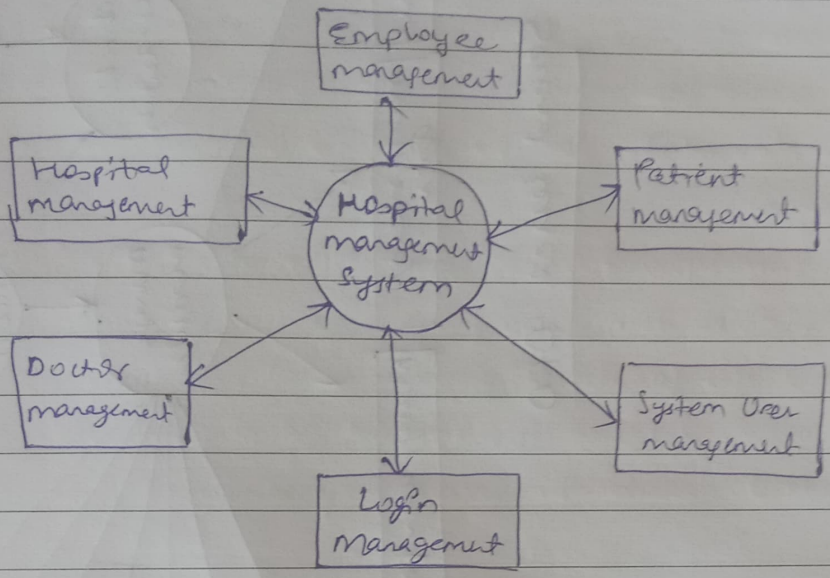
(iv) Configuration management :
It ensures the application to interact with only those resources that are concerned with the environment in which it runs. The configuration files are not created where the ~~extrem~~ external configuration to the application is seperated from the source code. The configuration file can be written during deployment, or they can be loaded at the run time, depending on the environment in which it is running.

q5

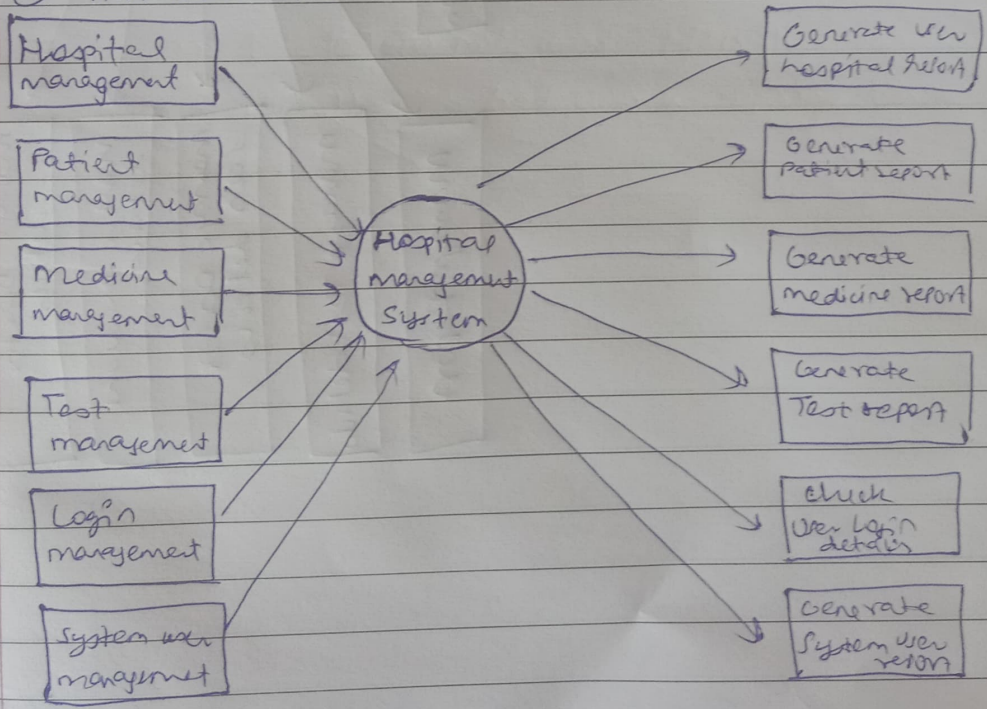
Ans

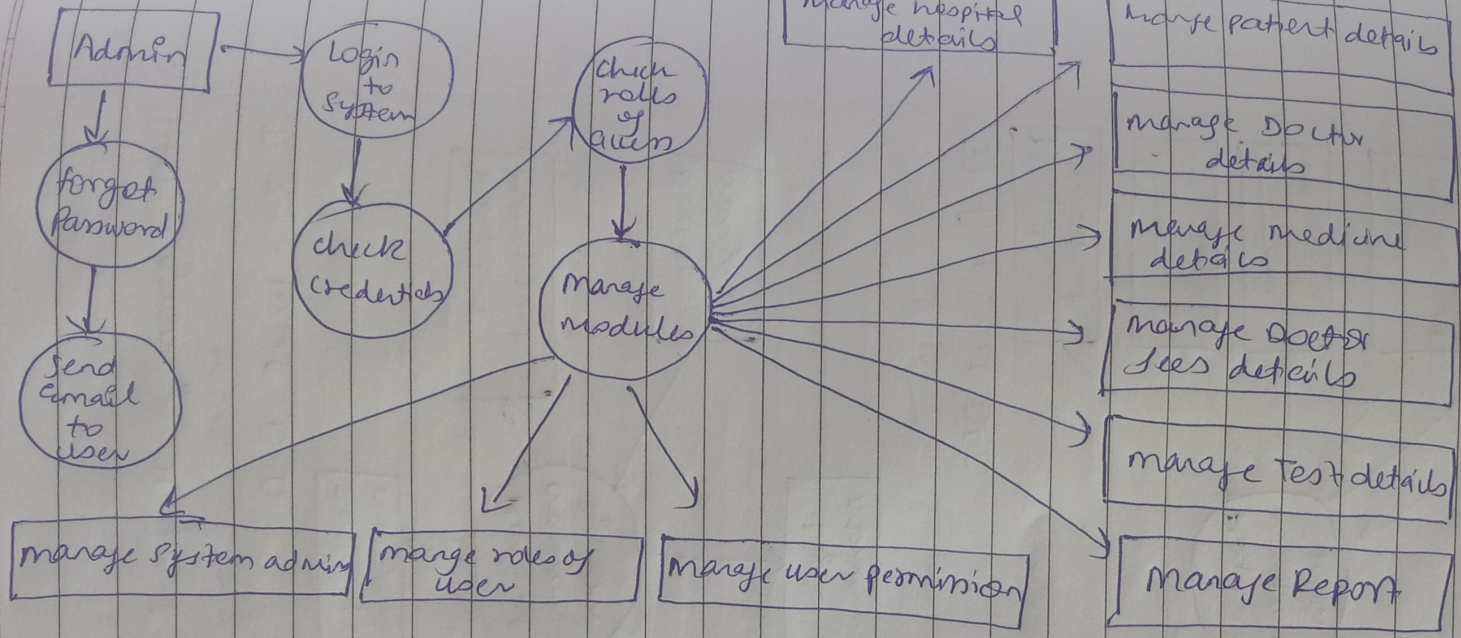
DfD diagram for Hospital Mangement System.

① Zero level DfD



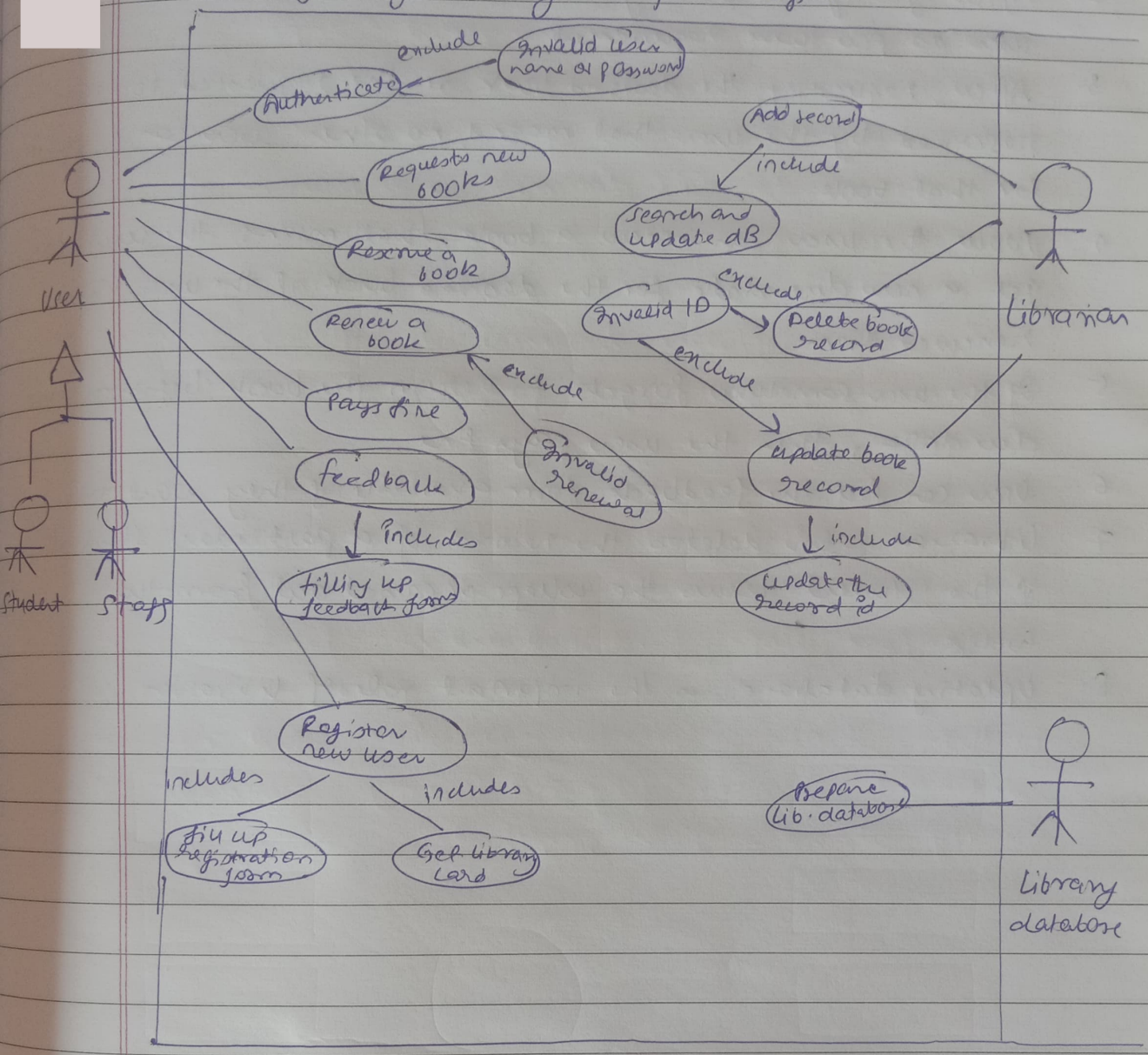
② first level DfD





Second Level DFD

Use case diagram for Library management System



① User who registers himself as a new user initially is regarded as staff or student for the library system.

- for the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.
- After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.

2. After getting the library card, a new book is requested by the user as per their requirement.
3. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
4. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays fine.
6. User can fill the feedback form available if they want to.
7. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college.
8. Updating database is the important role of librarian.