THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

EXPERIMENT NO. 1

Aim :- To prepare a detailed statement of problem for the selected mini project and to identify a suitable software model for the same.

Problem Title :- A Social media networking website

Problem Statement :-

In the era of social media dominance, social media platforms have revolutionized how people communicate and connect online. However, creating a robust and scalable social media networking website poses several challenges that need to be addressed to ensure a successful software engineering project.

The objective of this project is to develop a social media networking website that consists of core functionalities and seamless user experience. The website should allow users to create profiles, post their thoughts, follow other users, and engage in conversations through likes and comments. Additionally, it should support features such as trending topics.

The main challenges in developing this involve implementing a scalable and efficient architecture to handle a large user base and high traffic, ensuring data integrity and security, designing an

intuitive and responsive user interface, and integrating real-time updates for texts and notifications. Furthermore, addressing potential issues like spam, fake accounts, and maintaining user privacy will be crucial for creating a trustworthy and enjoyable user experience.

By tackling these challenges, the project aims to deliver a fully functional website that demonstrates a deep understanding of software engineering principles and best practices.

Software Model :-

Scrum is a type of Agile framework. It helps teams work together while the scrum we are talking about is most frequently used by the software development teams, and its principles can be applied to all kinds of teamwork.

The framework : People often think about scrum and agile as same thing because scrum is centered around continuous improvement, which is a core principle of agile model. The scrum framework is based on continuous learning and adjustment to fluctuating factors. It acknowledge that team doesn't know everything at the start of a project and will evolve through experience. Scrum is

structured to help teams naturally adapt to the changing conditions and user requirements with re-prioritization built into the process and short release cycles so that the team can constantly learn and improve with time.

A+
26/7/23

EXPERIMENT NO. 2

Aim: Application of Agile Process Model on the project(JIRA). Make Kanban Boards.

Theory:

Kanban is a common framework for agile and DevOps software development that provides transparency of work and team capacity. Kanban boards in Jira Software help teams visualize their workflow, limit work-in-progress, and maximize efficiency. The Jira Software Kanban board is designed to help teams continuously improve cycle time and increase efficiency.
Much more than a task board, the Jira kanban board functions to:
1. Promote transparency
2. Optimize workflow
3. Easily spot bottlenecks
4. Continuously improve Kanban boards give your team full visibility into what's next, so when one work item is completed, the team can quickly move on to the next.
Jira Software kanban boards are ideal for teams who practice agile methodologies; teams of all types can take advantage of the kanban board to facilitate smooth project management.
Here are a few ideas.
1. Software Development
2. Marketing
3. Business & HR Advantages of Kanban boards are:'

a) Better Visibility: Visualization is an important Kanban practice, and the most recognizable feature of the method is the Kanban board. Every project has a backlog of tasks to get through, and a series of process states that a task must pass through before it is delivered. Using the Kanban board, everyone can instantly see how tasks are moving through the process. The simplicity of its visual presentation enables you to easily spot bottlenecks while they are forming.

b) Improved Efficiency: Every product manager wishes they could get more done. Throwing extra resources at a problem is a possible solution when there's some room in the budget, but what if you could do more with what you already have? The most obvious benefit of using Kanban is improved flow efficiency that happens shortly after the method is implemented into your organization.

c) Increased Productivity: Improved efficiency naturally leads to the next Kanban benefit, which is increased productivity. Kanban benefits your productivity by shifting the focus from starting work to finishing work.

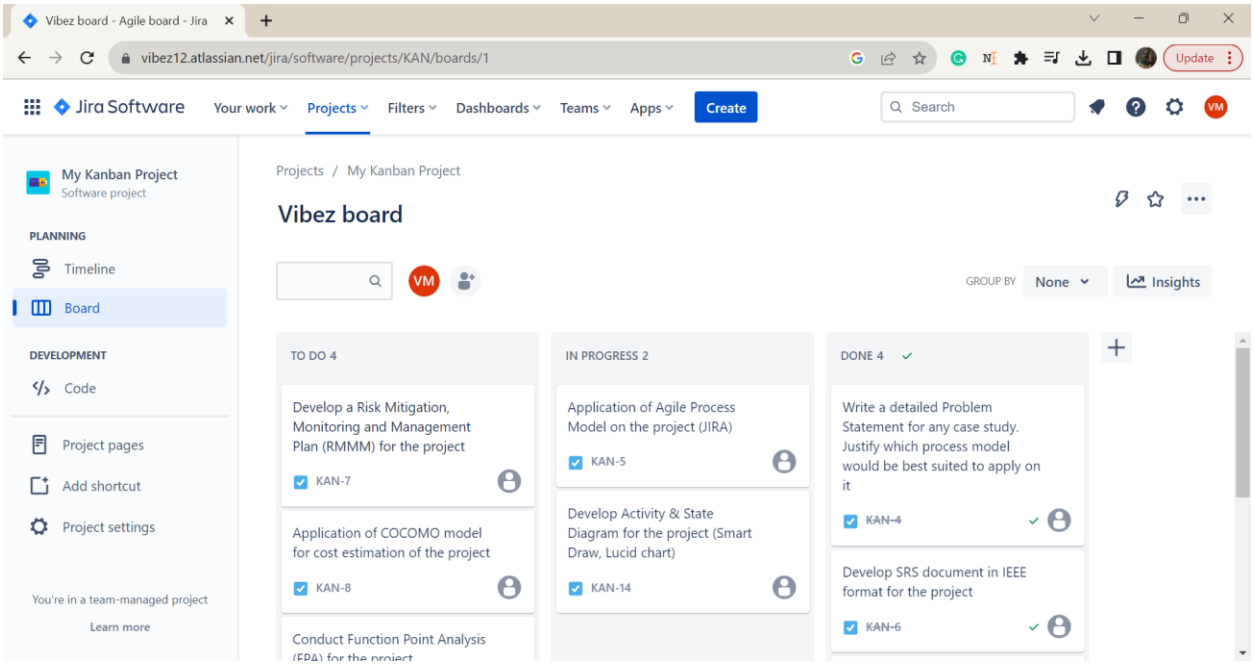
d) Preventing Team Overburden: Traditional management methods rely on planning upfront and pushing the work on to your team. This results in teams struggling with more work than they

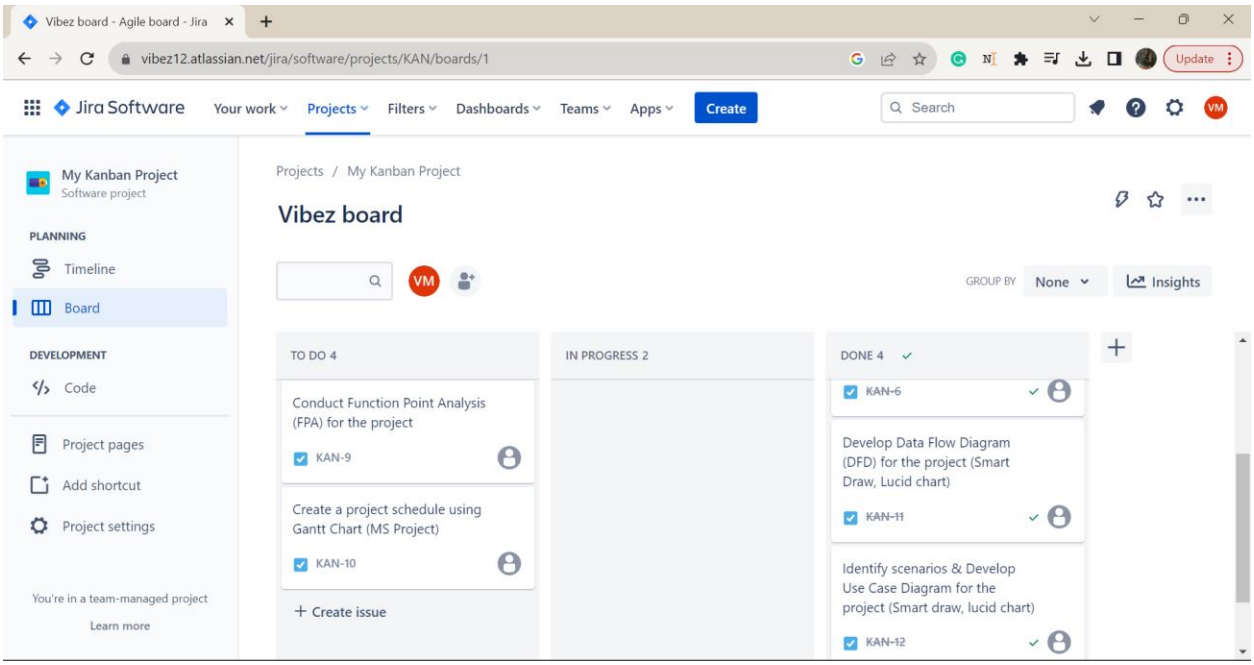
have the bandwidth for. Kanban suggests the implementation of pull system – the team pulls tasks into the workflow only when they have the capacity to do so.

e) Reduced Waste: Eliminating or reducing waste is a cornerstone of lean management. It was an integral feature of the Toyota Production System, the ancestor of modern kanban. Waste is defined as any action that uses resources without adding value. Value refers to something that the customer is willing to pay for. Many activities would not be classed as “value-adding” according to these criteria but are nevertheless essential to delivering quality products. The area to target is a nonessential waste.

f) Flexibility: For many companies, the strive for business agility is driven by the need for flexibility. Especially for early-stage companies, the freedom to respond swiftly and decisively to your customers’ needs and competitors’ actions is essential to successful growth. With no prescribed phase durations, features are released as soon as they are completed. By choosing directions with a Kanban roadmap rather than relying on a rigid general project plan, product managers are free to reassess immediate priorities based on changes in the market. The Kanban Method suggests an approach of backlog management that helps teams become more self-managed while bringing transparency and consistency to the decision-making process.

OUTPUT:





Conclusion : -

Hence, we implemented the application of Agile Process Model on the project (JIRA) using Kanban Boards.

Software Requirements Specification

for

<Vibez>

Prepared by Arnav Malvia

Rishab Mandal

Vivaan Mansukhani

Thadomal Shahani Engineering College

August 2, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of developing Vibez as a project is to create a functional and user-friendly social media networking website. By building this platform, we aim to offer users a familiar experience while adding our unique touch to the interface and features. Vibez will serve as a convenient space for users to connect, share thoughts, engage in conversations, and follow updates in real-time. Moreover, the project allows us to demonstrate our software engineering capabilities, problem-solving skills, and creativity in designing and implementing a scalable web application. Through this project, we seek to provide users with an enjoyable and interactive social media platform, fostering a sense of community and communication in the digital realm.

1.2 Document Conventions

Important points have been underlined to provide emphasis. Headings and Subheadings have been written in bold font to provide emphasis. The points in all sections have been written in the order of their priority, from higher priority points to lower priority points, so that important points are not missed out. Abbreviations are used in some places which will be understood by the developers of the application.

1.3 Intended Audience and Reading Suggestions

The intended audience for the "Vibez" social media networking website would primarily be tech-savvy individuals, social media enthusiasts, and users who enjoy connecting and engaging with others online. It aims to attract a diverse range of users, including students, young professionals, and individuals interested in sharing thoughts, interests, and staying updated with current trends and news. To ensure a successful implementation, the team should consider reading materials covering web development technologies such as HTML, CSS, JavaScript, and frameworks like React or Angular. Additionally, learning about database design and management, user authentication, real-time updates using WebSockets, and scalable server-side technologies (e.g., Node.js, Django) would be beneficial. Reading up on UI/UX design principles and social media platform guidelines will help in creating an appealing and intuitive user interface, encouraging user retention and interaction.

1.4 Product Scope

The product scope for "Vibez" social media networking website includes creating a user-centric and interactive platform for users to share thoughts, connect with others, and engage in discussions. The website will allow users to create profiles, post updates, like, comment, and follow others. It will support real-time updates to ensure timely content delivery and user notifications. The product will focus on ensuring a seamless and intuitive user experience, with responsive design for various devices. Additionally, Vibez will prioritize user privacy and data security, implementing robust authentication and data encryption measures. The platform will be scalable, accommodating a growing user base and potential future features. The scope also encompasses implementing a user-friendly admin panel for content moderation and analytics to monitor user engagement and interactions effectively. Lastly, Vibez will adhere to accessibility guidelines to ensure inclusivity for users with different abilities.

1.5 References

From a product perspective, "Vibez" social media networking website aims to offer a dynamic and user-centered platform that fosters meaningful connections and interactions among its users. The primary focus is on creating an intuitive and visually appealing user interface that encourages seamless navigation and engagement. Vibez will prioritize real-time updates, ensuring users receive timely content and notifications for a more immersive experience. The product perspective also emphasizes robust data management, safeguarding user privacy and implementing effective measures for data security. Additionally, Vibez will provide analytics and insights tools to enable users to track their interactions and engagement on the platform. The website will continuously innovate, incorporating new features and functionalities to stay relevant and meet users' evolving needs. Overall, the product perspective is geared towards delivering a vibrant and inclusive social media platform that captivates users and fosters a strong sense of community.

2. Overall Description

2.1 Product Perspective

From a product perspective, "Vibez" social media networking website aims to offer a dynamic and user-centered platform that fosters meaningful connections and interactions among its users. The primary focus is on creating an intuitive and visually appealing user interface that encourages seamless navigation and engagement. Vibez will prioritize real-time updates, ensuring users receive timely content and notifications for a more immersive experience. The product perspective also emphasizes robust data management, safeguarding user privacy and implementing effective measures for data security. Additionally, Vibez will provide analytics and insights tools to enable users to track their interactions and engagement on the platform. The website will continuously innovate, incorporating new features and functionalities to stay relevant and meet users' evolving needs. Overall, the product perspective is geared towards delivering a vibrant and inclusive social media platform that captivates users and fosters a strong sense of community.

2.2 Product Functions

The key functions of the Vibez include:

1) User registration and login
2) Creating and posting tweets
3) Following other users
4) Like and retweet functionality
5) User timeline display

2.3 User Classes and Characteristics

Vibez will cater to various user classes, including general users, administrators, and moderators. Users will have varying levels of technical expertise and will engage in different activities on the platform.

2.4 Operating Environment

Since the application is a web application it can work on any device having a browser.

- Device: Mobile Phone, Computer, Laptops, Tablets.
- Operating System: Windows, Linux distributions, Mac OS, Android
- RAM: 128 MB or more
- Disk Space: 20 MB or more.
- Browsers: Mozilla Firefox 30+, Google Chrome 27.0+, Microsoft Edge. Other browsers can also be used.
- Internet connection: Strong internet connection with speed of at least 1 Mbps for best experience

2.5 Design and Implementation Constraints

The development of Vibez should adhere to the incremental component-based software model. It will be implemented using primarily React JS, HTML, CSS, JavaScript, and a suitable backend programming language(Firebase) and framework.

2.6 User Documentation

Vibez will come with user documentation, including user manuals and on-site help to guide users through its functionalities.

2.7 Assumptions and Dependencies

Assumptions:

Vibez will be developed using modern web development tools and frameworks.

Users will have a stable internet connection for using the platform.

Dependencies:

Vibez will use the Firebase API for some functionalities, such as user authentication and displaying messages.

3. External Interface Requirements

3.1 User Interfaces

The user interface of Vibez will be designed to be user-friendly and intuitive. It will follow the style guide for social networking websites, ensuring a consistent and familiar user experience.

3.2 Hardware Interfaces

Vibez will interact with standard hardware components, such as web browsers and mobile devices.

3.3 Software Interfaces

Vibez will communicate with the Firebase API to perform actions like user authentication and fetching text.

3.4 Communications Interfaces

The application will utilize standard communication protocols, such as HTTP, for interactions with the backend server and external APIs.

4. System Features

4.1 Authentication and Authorization

4.1.1 Description and Priority:

The application will be having multiple users and so authentication becomes a high priority system feature. When the user creates a new account on the application, they will have to provide their email address and password. The password must be at least 8 characters long and must have at least one uppercase character, one digit and one special character. The passwords in the system will be hashed and stored so that no other person can get to know the password.

4.1.2 Response Sequences:

Once the user registers in the application, they will be guided to a login page where they will have to enter their email address and password to login. After successful login, the user will be redirected to the landing page of the application. There will also be a logout button on the navigation bar. On clicking the logout button, the user will be logged out.

4.1.3 Functional Requirements:

REQ-1: We use google login using firebase module for authentication and authorization functionality. The authentication will be a session-based authentication.

4.2 Post Photographs Feature:

4.2.1 Description and priority:

The user gets to post photographs which he intends to put on his feed. This is again a high priority feature. The user gets the option to choose images using file system or simply drag and drop the pictures into the provided area.

4.2.2 Response Sequences:

The user can add images using the drag and drop functionality or simply choose images from the files that he wants to upload. In case any image exceeds the maximum file size the user will be notified Software Requirements Specification using alerts. On adding the images, the user is redirected to the add location page.

4.2.3 Functional Requirements:

REQ-1: For uploading photos we will use firebase in built functions.

4.3 Add location Feature

4.3.1 Description and Priority:

The user gets to add the location for the image uploaded by using the map. The user will simply have to add a marker by clicking on the location where the picture was clicked. The priority of the feature is moderate.

4.3.2 Response Sequences:

On adding the location, the image is successfully added to the database and the user is redirected to the landing page of the website.

4.3.3 Functional Requirements:

REQ-1: JavaScript APIs will be used for geolocation functionality.

4.4 Search Properties Feature:

4.4.1 Description and Priority:

The user can search different accounts using the search functionality. This feature is of high priority and is primarily for users searching each other.

4.4.2 Response Sequence:

On filling the required information based on desired features, the results will be displayed on the webpage.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Scalability:

The application should be scalable and should perform without any interruption for all the users.

5.2 Safety Requirements

- Backup power supply should be present for server, so that it does not stop functioning in case of power failure.
- API keys of the APIs used should not be made open source.
- Code backup should be taken at regular time intervals.

5.3 Security Requirements

- The passwords of the users are hashed and then stored in the database so that no person can access the passwords of the users.
- The passwords should be at least 8 characters long and must have at least one uppercase character, one digit and at least one special symbol.
- The website should HTTPS protocol for security.
- POST requests are used for transferring information regarding authentication, adding properties, adding advertisements etc. through forms.

5.4 Software Quality Attributes

5.4.1 Usability:

The user interface should be simple to use and not cluttered with a lot of information.

5.4.2 Availability:

- The system should be available at all times.
- The system should be reliable and there should be no loss of data in case the server breaks down when operations are going on.

5.4.3 Maintainability:

The code for the application should be written cleanly and should be

well documented. The code should contain comments to help new

programmers and developers make changes in the application.

5.4.4 Testability:

The code should be written with proper test cases to be tested upon so

that no errors during production take place.

5.5 Business Rules:

The administrator of the application has full permission of controlling the

system.

6.Other Requirements

Appendix A: Glossary

- HTTPS: Hypertext Transfer Protocol Secure
- API: Application Programming Interface
- GUI: Graphical User Interface

EXPERIMENT NO. 4

Aim : - Develop Data Flow Diagram (DFD) for the project (Smart Draw, Lucid chart)

Theory : -

Data Flow Diagrams (DFDs) are powerful tools for visualizing and modeling the flow of data within a system or project. DFDs provide a clear and structured representation of how data moves through processes and entities. In this theory, we will discuss the process of developing a Data Flow Diagram for a project using two popular diagramming tools, SmartDraw and Lucidchart.

Project Understanding and Scope Definition:

Before creating a DFD, it's crucial to have a clear understanding of the project's objectives and scope. Identify the key processes, data sources, and entities involved in the project. Determine the boundaries of your system.

Selecting a Diagramming Tool:

SmartDraw and Lucidchart are two widely used diagramming tools that offer DFD creation capabilities. Choose the tool that best suits your needs based on factors such as your team's familiarity, collaboration requirements, and available features.

Creating a Context Diagram:

Start with a high-level Context Diagram that represents the entire system as a single process. External entities, such as users or other systems, should be depicted interacting with this process. Use rectangles to represent processes and arrows to depict data flow between entities and processes.

Identifying Processes and Data Flows:

Break down the system into its constituent processes. Each process represents a distinct function or action within the system. Identify the data flows between processes and entities, representing the data exchanged in the system.

Drawing DFD Levels:

DFDs can have multiple levels of abstraction. The highest level is the Context Diagram, and subsequent levels provide more detailed views of individual processes. Use SmartDraw or Lucidchart to create additional DFDs for each process, progressively decomposing the system until you reach the desired level of detail.

Labeling and Annotating:

Properly label each process, data store, data flow, and entity in your DFDs. Add descriptions or annotations to clarify the purpose and function of each element. This aids in understanding and documentation.

Validation and Review:

Regularly review and validate your DFDs with project stakeholders to ensure accuracy and alignment with the project's goals. Make revisions as necessary based on feedback and evolving project requirements.

Documentation and Sharing:

Save your DFDs in a format that is easily accessible and shareable with team members and stakeholders. Both SmartDraw and Lucidchart allow you to export diagrams in various formats, including PDF, PNG, or as links for online sharing.

Version Control:

Maintain version control of your DFDs to track changes and revisions over time. This ensures that everyone is working with the latest and most accurate representations of the system.

Continuous Updates:

DFDs are dynamic documents. As the project progresses and evolves, update your DFDs to reflect any changes in processes, data flows, or system architecture.

Level 0 DFD:

The Level 0 DFD, often referred to as the "context diagram," provides an overview of the entire system or project. It depicts the highest-level view of the system, emphasizing its interactions with external entities and the data flows between them. Here are the key characteristics and elements of a Level 0 DFD:

External Entities: External entities, such as users, other systems, or data sources, are represented as rectangles on the diagram. They interact with the system but are not part of it. Arrows depict data flow between these entities and the central process.

Central Process: In the Level 0 DFD, a single central process represents the entire system. This process is responsible for coordinating interactions with external entities and managing the flow of data between them.

Data Flows: Data flows between external entities and the central process are shown as arrows connecting them. These arrows are labeled to indicate the nature of the data being exchanged.

No Internal Processes: The Level 0 DFD does not depict internal processes or detailed subprocesses within the system. It provides a high-level, simplified representation of the system's boundaries and interactions.

Level 1 DFD:

The Level 1 DFD provides a more detailed view of the system by decomposing the central process from the Level 0 DFD into smaller subprocesses. It breaks down the high-level processes into their constituent components and illustrates the data flows among them. Here are the key characteristics and elements of a Level 1 DFD:

Subprocesses: In the Level 1 DFD, the central process from the Level 0 diagram is decomposed into multiple subprocesses or lower-level processes. Each subprocess represents a specific function or task within the system.

Data Stores: Data stores, often depicted as rectangles, represent repositories where data is stored and retrieved. Data flows between processes and data stores, indicating data storage and retrieval operations.

Data Flow Paths: The Level 1 DFD illustrates detailed data flow paths between subprocesses and external entities. It shows how data is processed and transformed as it moves through the system.

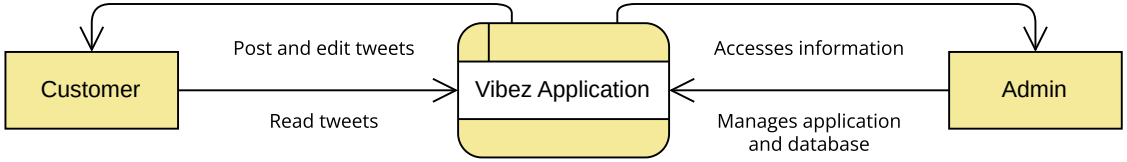
Control Flows: While Level 0 primarily focuses on data flow, Level 1 may also depict control flows, which indicate the sequence and logic of operations within subprocesses.

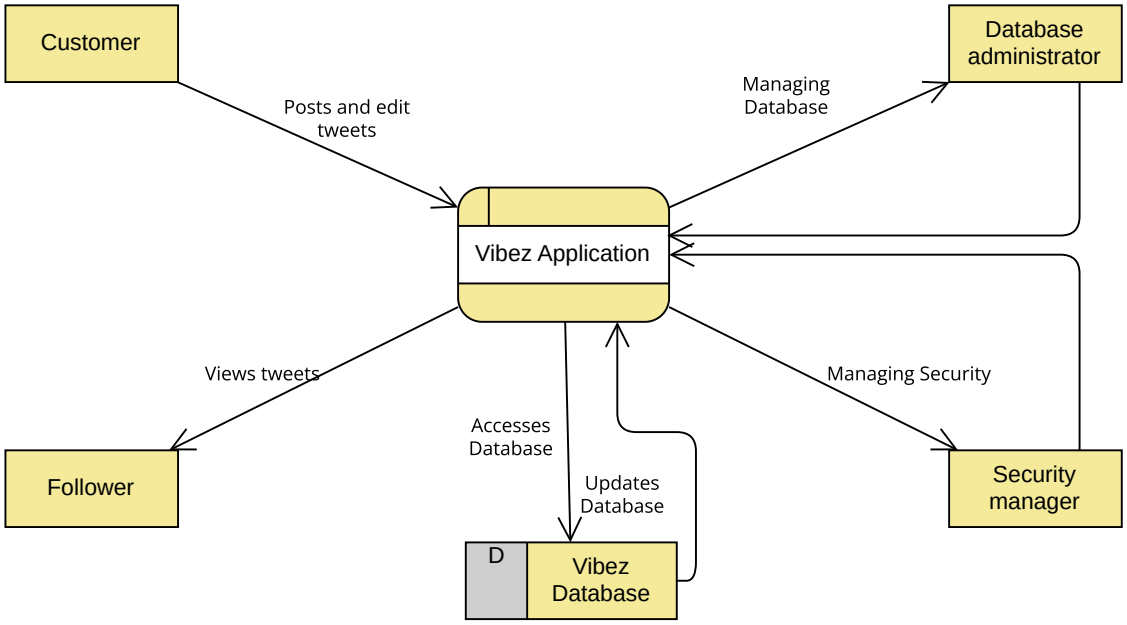
Data Flow Labels: Each data flow path is labeled to specify the type of data and its purpose. These labels provide clarity about the information being transmitted.

Interfaces: External entities may have specific interfaces or interactions with subprocesses. These are shown as additional data flows between external entities and subprocesses.

Conclusion : -

Data Flow Diagrams at Level 0 and Level 1 serve different purposes in system modeling. The Level 0 DFD provides a high-level overview of the system's boundaries and interactions with external entities, while the Level 1 DFD delves into the details of internal processes, data flows, and subprocesses. Together, these two levels help project stakeholders understand the system's architecture, data flow paths, and interactions, forming a solid foundation for further analysis, design, and development activities in system development and documentation.





EXPERIMENT NO. 5

Aim : - Develop Activity & State Diagram for the project (Smart Draw, Lucid chart)

Theory : -

Activity and State Diagrams are essential modeling tools in software engineering used to visualize the behavior and flow of a software system. They help in understanding, analyzing, and documenting the dynamic aspects of a system's functionality. In this theory, we will explore the process of developing Activity and State Diagrams for a software project.

Activity Diagram

Activity Diagrams are part of the Unified Modeling Language (UML) and are particularly useful for representing the workflow and business processes within a software system. They show the sequential and parallel activities, decision points, and transitions in a clear and intuitive manner.

Steps to Develop an Activity Diagram:

Identify Use Cases: Begin by identifying the primary use cases or scenarios within your software project. These could represent user interactions, system processes, or any other relevant activities.

Identify Actors: Determine the actors involved in each use case. Actors are external entities, such as users, systems, or devices, that interact with the system.

Define Activities: For each use case, identify the main activities or steps involved. These activities should represent actions that lead to the accomplishment of the use case's goal.

Sequence Activities: Sequence the activities in a logical order, indicating the flow of execution. Use control flow arrows to connect activities, showing the order in which they are performed.

Decision Points: Add decision points (diamond shapes) to represent choices or conditions that determine the path the workflow will follow. Use conditional flow arrows to illustrate the possible outcomes.

Fork and Join Nodes: Use fork and join nodes to represent parallel execution of activities. Forks split the flow into multiple paths, while joins merge them back together.

Start and End Points: Every activity diagram should have a starting point (usually denoted by a filled circle) and an ending point (usually denoted by a filled circle with a ring). These indicate where the workflow begins and ends.

Activity Labels: Label each activity and decision point with a clear and concise description of the action or condition.

State Diagram

State Diagrams, also part of UML, are used to model the behavior of an individual object or system component over time. They are particularly useful for representing the states, events, and transitions of an entity within a software system.

Steps to Develop a State Diagram:

Identify the Object: Determine the object or system component you want to model. This could be a class, a module, or any entity that exhibits behavior.

Identify States: Identify the possible states that the object can be in. States represent conditions or modes in which the object exists.

Define Events: Determine the events or triggers that cause the object to transition from one state to another. Events could be user actions, system events, or external inputs.

Create Transitions: Connect states with transitions to show how the object moves from one state to another in response to events. Label transitions with event names.

Initial and Final States: Include initial states (filled circle) to represent the starting state of the object and final states (encircled filled circle) to indicate when the object's lifecycle ends.

State Labels: Label each state with a descriptive name that reflects the object's condition or behavior in that state.

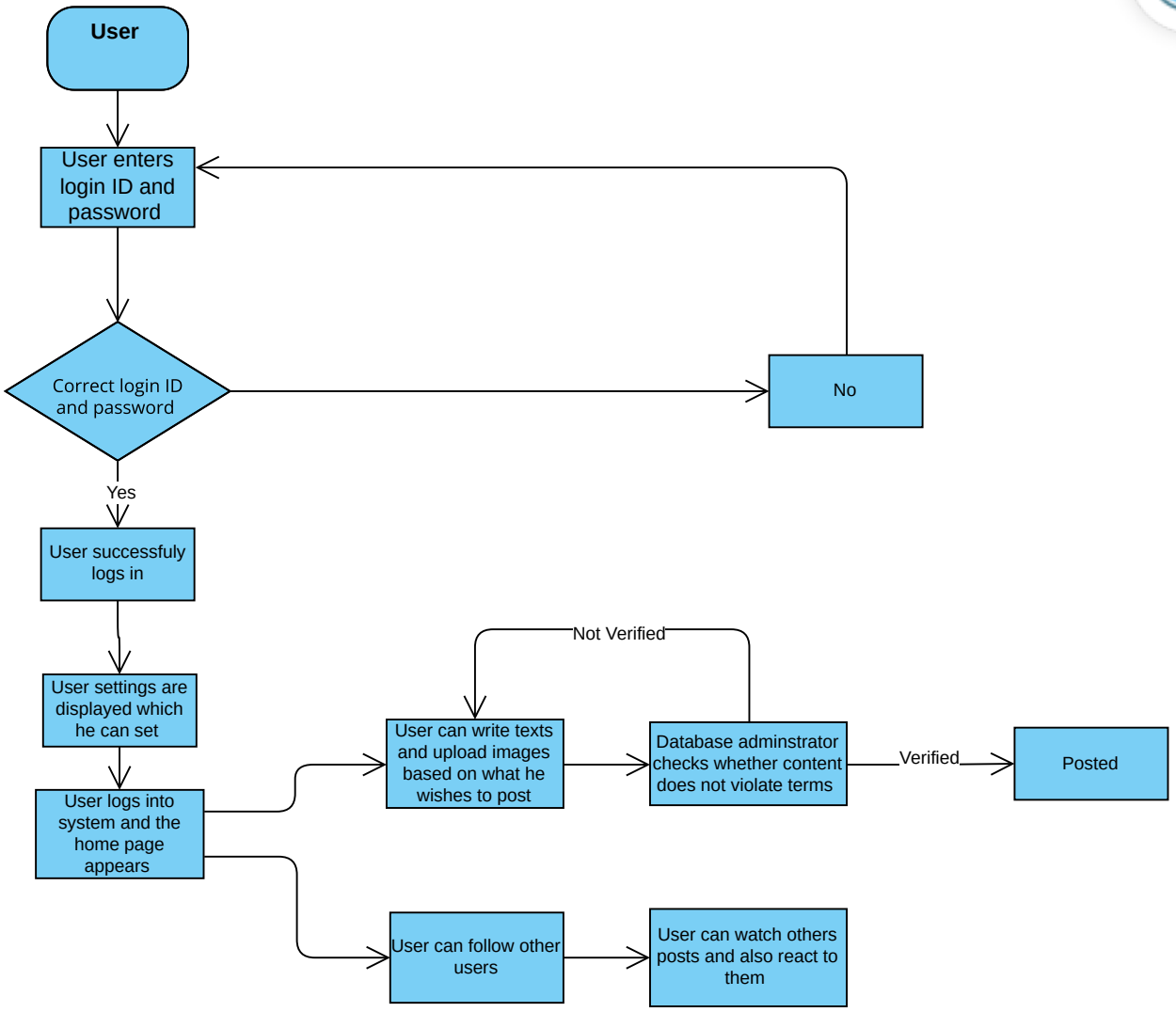
Conditions and Actions: Optionally, you can add conditions or actions associated with transitions, specifying what needs to be true for a transition to occur or what actions should be performed during a transition.

Hierarchy (Optional): For complex objects with nested states, you can create a hierarchical state diagram to represent different levels of states and transitions.

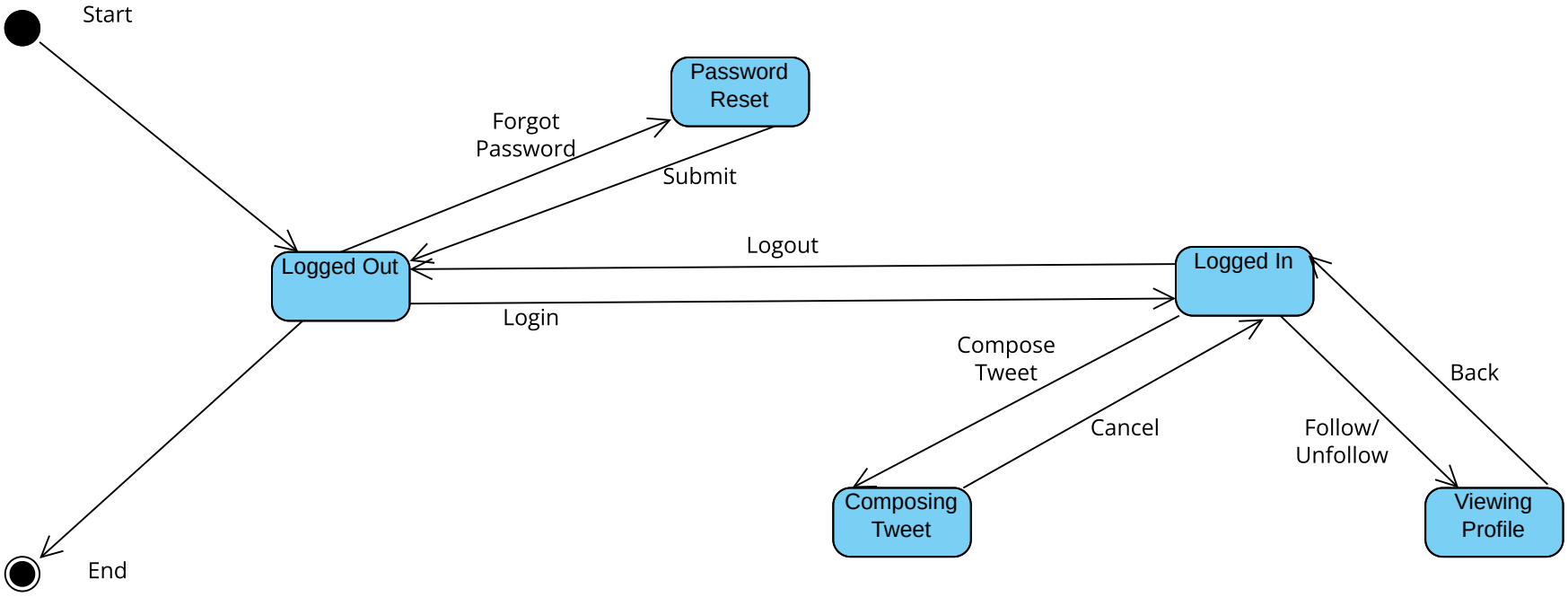
Conclusion : -

Activity and State Diagrams are valuable tools in software engineering for modeling the dynamic behavior of a software system. Activity Diagrams help visualize workflow and processes, while State Diagrams provide insight into the behavior and transitions of individual objects or components. Developing these diagrams systematically ensures a clear understanding of the software's functionality, facilitating communication among project stakeholders, aiding in system design, and serving as a foundation for successful software development.

Vibez



Vibez State Diagram



EXPERIMENT NO. 6

Aim : - Identify scenarios & Develop Use Case Diagram for the project.

Theory : -

Use Case Diagrams are a fundamental tool in software engineering for modeling the interactions between users (actors) and a system. They help identify and define various scenarios or user interactions within a project, providing a visual representation of system functionality from an external perspective. In this theory, we will explore the process of identifying scenarios and developing Use Case Diagrams for a project.

Identifying Scenarios

Before creating a Use Case Diagram, it's essential to identify and understand the scenarios or use cases that describe how users interact with the system. Scenarios represent specific tasks, actions, or processes that users or external systems perform within the software.

Steps to Identify Scenarios:

Stakeholder Analysis: Identify all stakeholders, including end-users, administrators, and external systems, who will interact with the software.

User Interviews and Surveys: Conduct interviews or surveys with users and stakeholders to gather information about their requirements, needs, and expected interactions with the system.

Brainstorming Sessions: Organize brainstorming sessions with the project team to generate a list of potential use cases. Encourage participants to think from different user perspectives.

Requirements Documentation: Review project requirements documents and specifications to identify user stories or tasks that can be converted into use cases.

Contextual Analysis: Analyze the system's context within the organization or industry to identify potential external interactions and scenarios.

Functional Decomposition: Break down the system's functionality into smaller, manageable tasks or features, each of which can become a use case.

Boundary Identification: Clearly define the boundaries of the system and identify what falls within the scope of the project.

Developing Use Case Diagrams

Once scenarios are identified, Use Case Diagrams are created to represent the relationships between actors and use cases, providing a visual overview of how users interact with the system.

Steps to Develop a Use Case Diagram:

Identify Actors: Actors are entities external to the system that interact with it. They can be users, other systems, or hardware devices. List and name all actors involved in the scenarios.

Identify Use Cases: Each use case represents a specific scenario or functionality within the system. List and name the use cases based on the scenarios identified in the previous step.

Establish Relationships: Connect actors to their associated use cases using lines with arrows pointing from actors to use cases. This signifies that an actor interacts with the use case.

Include System Boundary: Draw a boundary around the use cases to represent the system's scope. This boundary separates the system from its external actors.

Extend Relationships (Optional): Use extends and includes relationships to depict how one use case can extend or include another. This helps represent complex interactions and alternative flows.

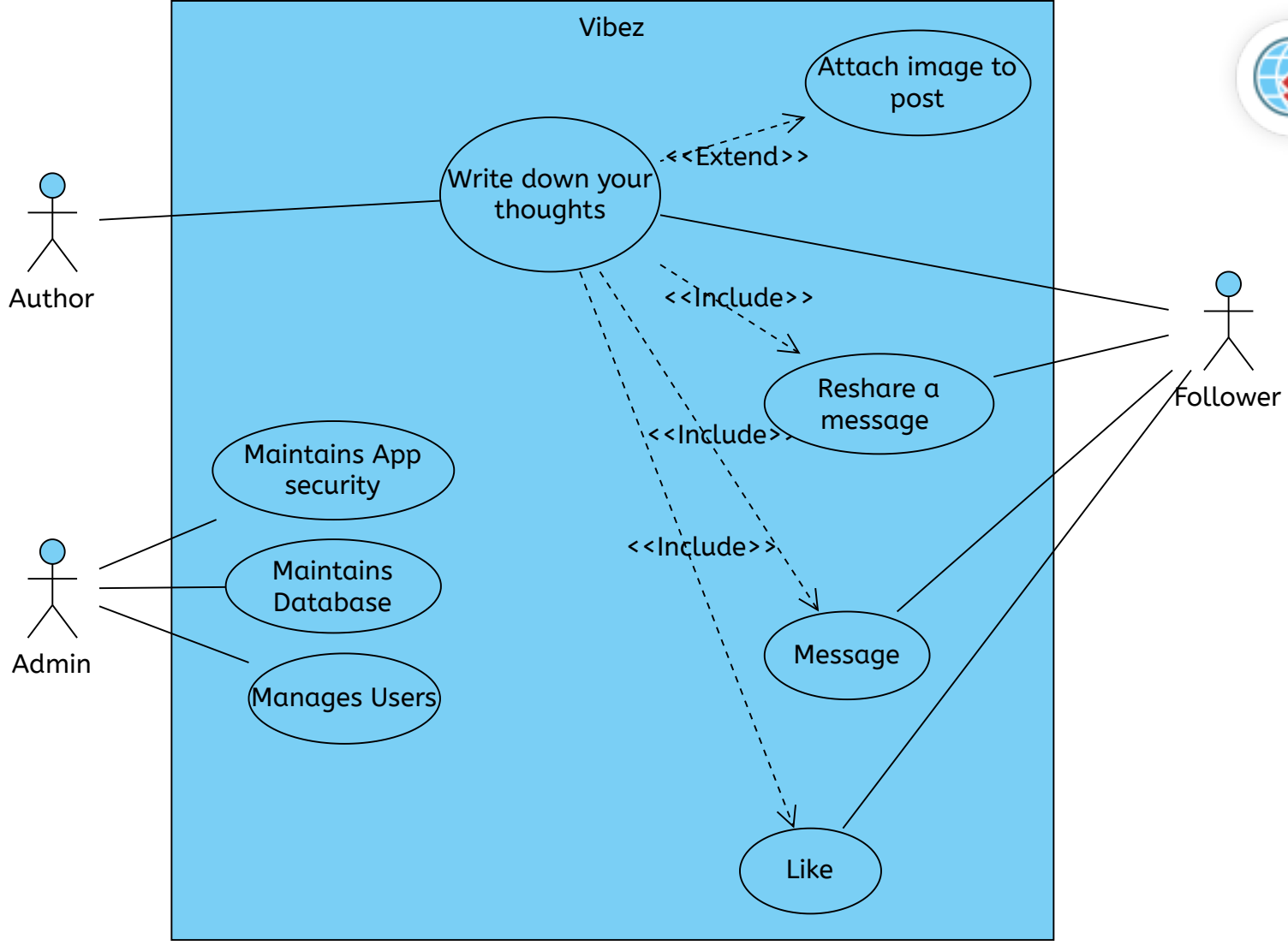
Generalization (Inheritance): Use generalization relationships to show how one use case inherits common behaviors from another. This is useful for modeling variations of a use case.

Document Use Case Descriptions: Add brief descriptions or comments to each use case to provide a clear understanding of its purpose and functionality.

Keep It Simple: Use Case Diagrams should be clear and concise. Avoid including excessive details or technical implementation aspects.

Conclusion : -

Identifying scenarios and developing Use Case Diagrams are essential steps in software engineering, particularly during the requirements analysis and design phases. These diagrams help stakeholders, including developers and project managers, visualize how users interact with the system and understand the software's expected functionality from an external perspective. Properly constructed Use Case Diagrams serve as valuable tools for communication, requirement documentation, and the foundation for subsequent stages of software development, such as system design and implementation.



EXPERIMENT NO. 7

Aim : - Create a project schedule using Gantt Chart (MS Project)

Theory : -

A Gantt Chart is a powerful project management tool that provides a visual representation of a project's tasks, their dependencies, and their scheduled durations. Microsoft Project is a widely-used software application for creating and managing Gantt Charts and project schedules. In this theory, we will explore the process of creating a project schedule using a Gantt Chart in Microsoft Project.

Step 1: Define Project Scope and Objectives

Before creating a project schedule, it's crucial to have a clear understanding of the project's scope, objectives, deliverables, and constraints. This information forms the foundation for your project schedule.

Step 2: Identify Tasks and Activities

Work Breakdown Structure (WBS): Break down the project into smaller, manageable tasks or activities. The WBS is a hierarchical list of all project tasks, starting with the top-level project phases and breaking down into finer levels of detail.

Task List: Create a comprehensive task list that includes all the activities required to complete the project. Each task should have a clear name, description, and an estimated duration.

Step 3: Define Task Dependencies

Identify the dependencies between tasks. Tasks can be dependent on other tasks for various reasons, such as sequential order or resource constraints. Microsoft Project allows you to specify dependencies, including Finish-to-Start (FS), Start-to-Start (SS), Finish-to-Finish (FF), and Start-to-Finish (SF).

Step 4: Estimate Task Durations

For each task, estimate the duration required for completion. You can use historical data, expert judgment, or analogous estimating techniques to arrive at realistic duration estimates. Enter these estimates in Microsoft Project.

Step 5: Assign Resources

Resource Identification: Identify the human and material resources required for each task. Specify who will perform the work and what tools or equipment are needed.

Resource Allocation: In Microsoft Project, assign resources to tasks. This ensures that the right people or materials are available when needed.

Step 6: Set Task Constraints

Specify any constraints on tasks. Constraints can be date constraints (e.g., a task must start on a specific date) or other limitations that affect task scheduling. Microsoft Project allows you to set constraints for tasks.

Step 7: Create the Gantt Chart

Open Microsoft Project: Launch the software and create a new project file.

Task Entry: Enter all tasks and their relevant details, including task names, durations, dependencies, resources, and constraints.

Create the Gantt Chart: Microsoft Project will automatically generate a Gantt Chart based on the information you entered. The Gantt Chart provides a visual timeline of your project, displaying tasks, durations, dependencies, and resource allocations.

Format and Customize: Customize the Gantt Chart to suit your project's needs. You can adjust colors, fonts, and task bars, and add milestones, deadlines, or other project-specific elements.

Step 8: Review and Refine

Carefully review the Gantt Chart to ensure that the project schedule aligns with your project objectives and constraints. Make any necessary adjustments, considering resource availability, critical path analysis, and project risks.

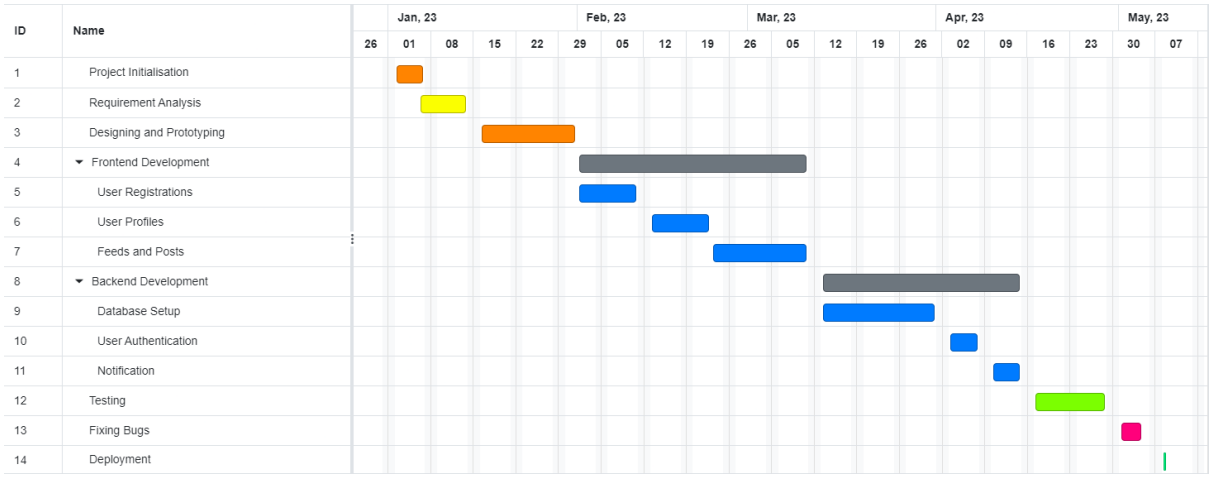
Step 9: Baseline the Schedule

Once you are satisfied with the project schedule, save it as a baseline. This snapshot of the schedule serves as a reference point for monitoring and controlling project progress.

Step 10: Monitor and Update

Regularly update the Gantt Chart to reflect actual progress, including completed tasks, task durations, and any changes in dependencies or resources. Microsoft Project provides tools for tracking progress and generating updated schedules.

Output : -



Conclusion : -

Creating a project schedule using a Gantt Chart in Microsoft Project is a systematic process that involves defining project scope, identifying tasks, estimating durations, setting dependencies, and allocating resources. A well-constructed Gantt Chart not only provides a visual representation of the project schedule but also serves as a valuable tool for project planning, execution, and monitoring. It helps project managers and teams stay organized, meet deadlines, and successfully deliver project objectives. Regularly updating and maintaining the Gantt Chart ensures that the project stays on track and adapts to changes as needed.

Experiment No. 8

Aim : - Conduct Function Point Analysis(FPA) for the project.

Theory : -

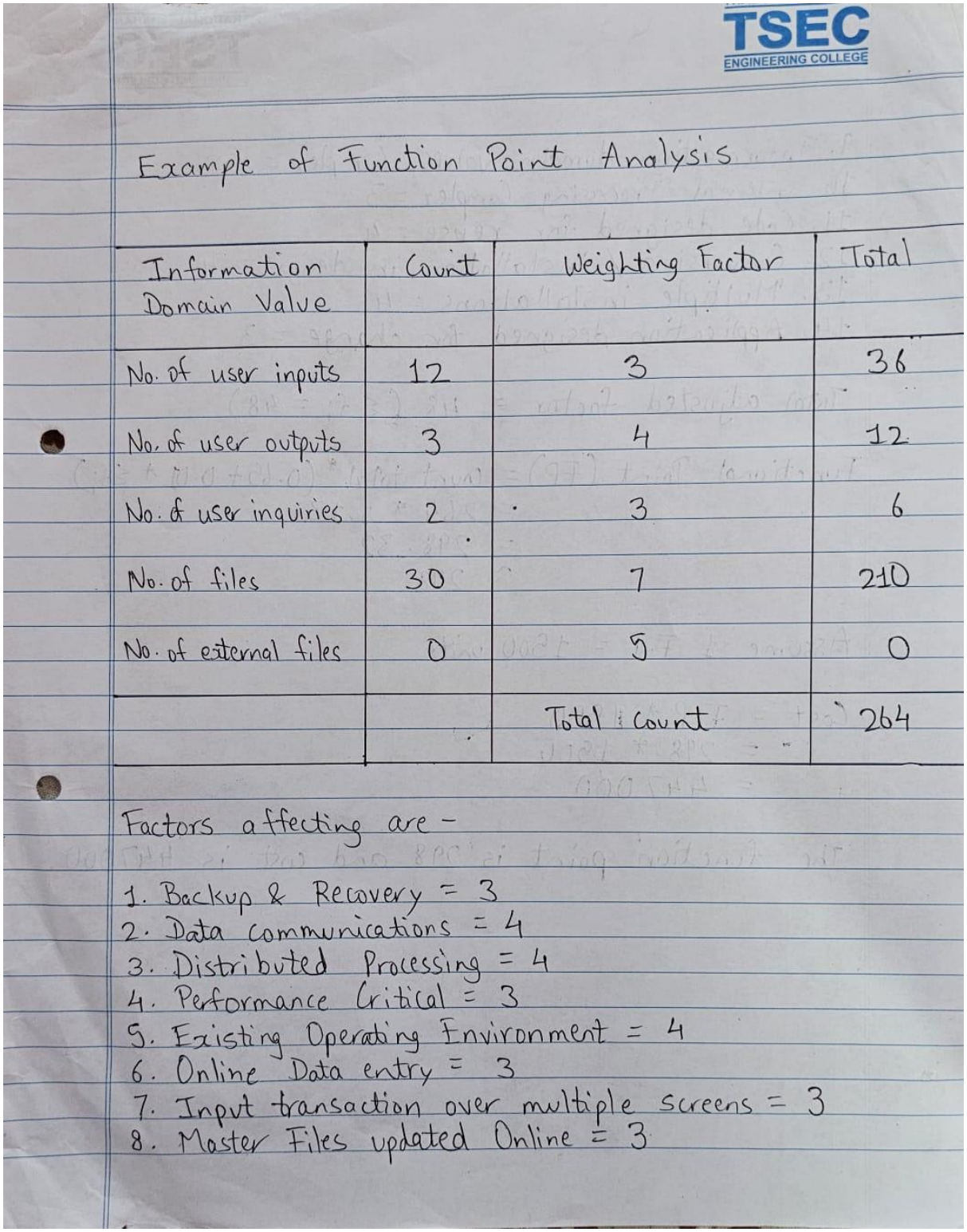
The Functional Point Analysis (FPA) is a one of the most popularly used software estimation technique to measure the functional size of the software work i.e. It is a method or set rules to measure the amount of software functionalities and software size of the software developed product. It depends on logical view flow of the application i.e. the number of functionalities delivered to the users. The Functional Point (FP) is the measurement of functional size of the software application.
When a new project comes to organisation, the organisation starts planning and estimation on that project in terms of schedule time, cost, resources, etc. The estimation process of software product includes the testing phase in terms of every functionality of the application, so the name shows functional point analysis.
There are multiple parameters in functional point analysis process calculation. They are as follows:

- External Inputs(EI)
- External Outputs(EO)
- External Inquiries (EI)
- Internal Logic Files(ILF)
- External Logic Files(ELF)

1. External Inputs(EI): It is a elementary input parameter which is used in functional point analysis process when the input data's to the application are arranged from the external sources like input screens or business documents or external databases or external tables or files i.e. the inputs to the application from outside but within business criteria boundary.
2. External Outputs(EO): In this elementary process the processed data will be sent to the output in externally i.e. the generated response in terms of log reports or files. This report or files are used as an input for other application. The output screen or reports are examples of EO.
3. External Inquiries(EI): It is an elementary parameter, which provides relationship between input and output components to retrieve the response/output. The interrupts are the example of external inquiries.
4. Internal Logic Files(ILF): In this process the group of interrelated or logical data's are stored inside the system with related to the application standard. The input data's are processed by the application to stored in a file with logical structure. The database or directories are the examples of it.
5. External Logic Files(ELF): It is also known as external interface files where the group of logical interrelated external files are stored for reference purpose

and used by the software application. It is externally maintained to validate/get the data from that file. The shared database or shared systems are the examples of it.

Output : -



TSEC
ENGINEERING COLLEGE

Example of Function Point Analysis

Information Domain Value	Count	Weighting Factor	Total
No. of user inputs	12	3	36
No. of user outputs	3	4	12
No. of user inquiries	2	3	6
No. of files	30	7	210
No. of external files	0	5	0
		Total count	264

Factors affecting are -

1. Backup & Recovery = 3
2. Data communications = 4
3. Distributed Processing = 4
4. Performance Critical = 3
5. Existing Operating Environment = 4
6. Online Data entry = 3
7. Input transaction over multiple screens = 3
8. Master Files updated Online = 3

Conclusion : -

Hence we have understood the implementation of Function Point.

Experiment No. 9

Aim: Application of COCOMO model for cost estimation of the project.

Theory:

The COCOMO (Constructive Cost Model) is one of the most popularly used software cost estimation models i.e. it estimates or predicts the effort required for the project, total project cost and scheduled time for the project. This model depends on the number of lines of code for software product development. It was developed by a software engineer Barry Boehm in 1981.

The COCOMO estimates the cost for software product development in terms of effort (resources required to complete the project work) and schedule (time required to complete the project work) based on the size of the software product. It estimates the required number of Man-Months (MM) for the full development of software products. According to COCOMO, there are three modes of software development projects that depend on complexity. Such as:

1. Organic Project: It belongs to small and simple software projects which are handled by a small team with good domain knowledge and few rigid requirements. Example: Small data processing or Inventory management system.
2. Semidetached Project: It is an intermediate (in terms of signs and complexity) project, where the team having mixed experience (both experience and inexperience resources) to deal with rigid/non rigid requirements. Example: Database design or OS implementation
3. Embedded Project: This project having a high level of complexity with a large team size by considering all sets of parameters(software, hardware and operational).

Types of COCOMO mode:

1. The basic COCOMO: It is the one type of static model to estimates software development effort quickly and roughly. It mainly deals with the number of lines of code and the level of estimation accuracy is less as we don't consider the all parameters belongs to the project. The estimated effort and scheduled time for the project are given by the relation: Effort (E) = $a*(KLOC)^b$ MM

Scheduled Time (D) = $c*(E)^d$ Months(M) Where, **E =** Total effort required for the project in Man-Months (MM).

- **D =** Total time required for project development in Months (M).
- **KLOC =** the size of the code for the project in Kilo lines of code.
- **a, b, c, d =** The constant parameters for a software project.

2. The intermediate COCOMO: The intermediate model estimates software development effort in terms of size of the program and other related cost drivers parameters (product parameter, hardware parameter, resource parameter, and project parameter) of the project. The estimated effort and scheduled time are given by the relationship: Effort (E) = $a*(KLOC)^b$ *EAF MM

Scheduled Time (D) = $c*(E)^d$ Months(M) Where,

- **E =** Total effort required for the project in Man-Months (MM).
- **D =** Total time required for project development in Months (M).
- **KLOC =** The size of the code for the project in Kilo lines of code.
- **a, b, c, d =** The constant parameters for the software project.

3.The detailed COCOMO: It is the advanced model that estimates the software development effort like Intermediate COCOMO in each stage of the software development life cycle process.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment :- 9

Aim :- To construct a Cocomo model for cost estimation for the chosen problem statement.

Problem Statement :- Social Media Website

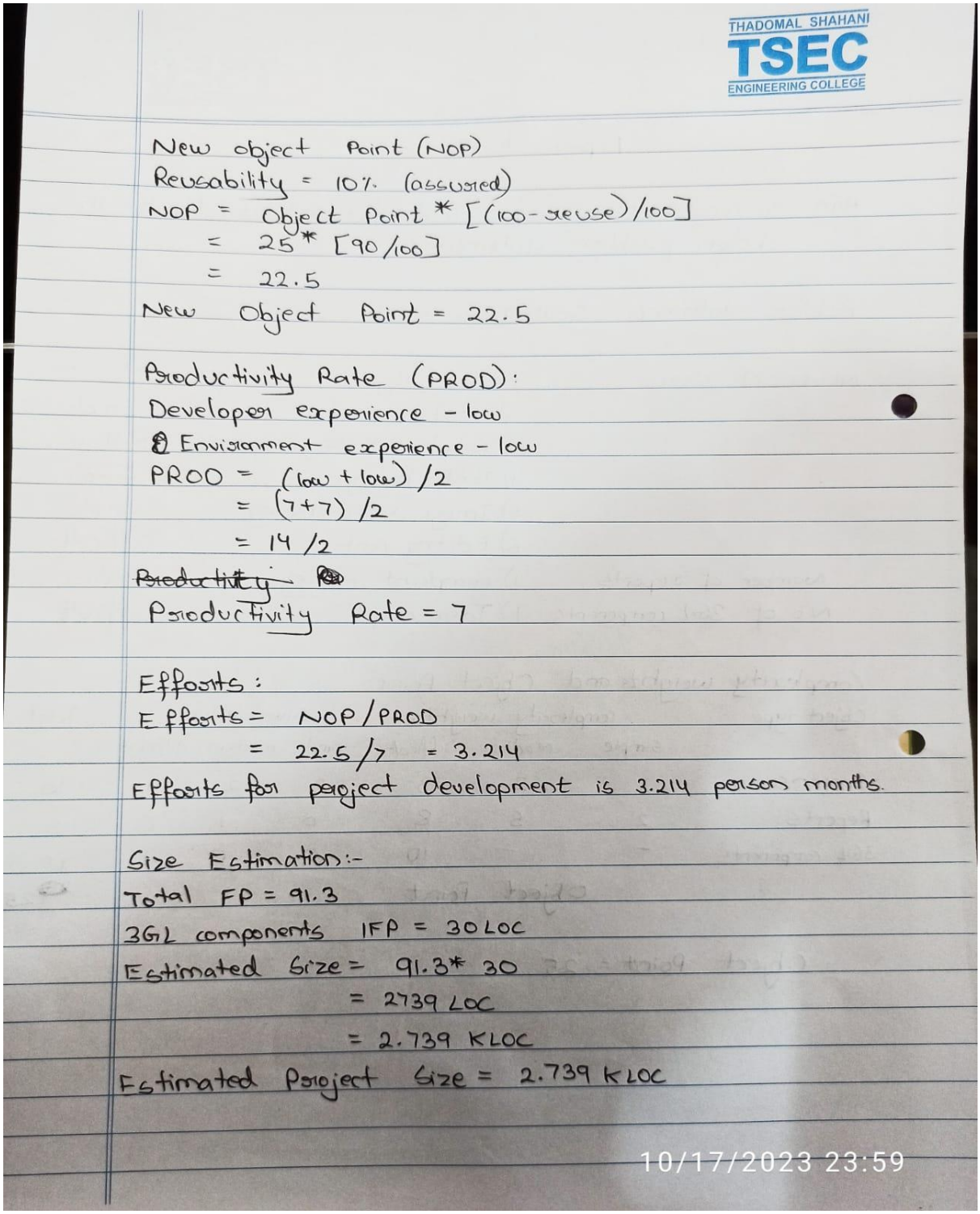
Number of screens	1) Home Page	Medium
	2) Profile	Simple
	3) Complaint History screen	Medium
	4) Text sharing	Simple
	5) Image, video sharing	Simple
	6) Editing content	Difficult
Number of reports	1) Complaint History	Medium
No of 3GL components	1) Javascript	Difficult

Complexity weights and Object Point:

Object type	Complexity weights			Given Value			Total
	simple	Medium	Difficult	simple	medium	difficult	
Screens	1	2	3	3	2	1	10
Reports	2	5	8	0	1	0	5
3GL components	–	–	10	0	0	1	10
			Object Point				25

∴ Object Point = 25

10/17/2023 23:58



Conclusion: Hence, we have understood the implementation of COCOMO model

Experiment No. 10

Aim: To prepare RMMM plan for the Vibez (social media) software

Theory:

Planning the risk management

The proactive strategy for risk estimation is used which helps us in identifying the possible threats that can occur during the project well in advance. Accordingly, steps to avoid, monitor and manage the risk are to be carried out and noted down in the form of RMMM plan.

Example:

THE RMMM PLAN

A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet (RIS) . In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives:

(1) to assess whether predicted risks do, in fact, occur

(2) to ensure that risk aversion steps defined for the risk are being properly applied

(3) to collect information that can be used for future risk analysis. In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project).

Identifying Risks:

Risk ID	Risk Summary	Probability	Impact	Risk Exposure
1	Data Leak	50%	1	Rs. 70,000/month
2	Server/Application Hacked	60%	1	Rs. 20,000/month
3	Financial Loss Due To Malware	20%	2	Rs. 20,000/month
4	Violation Of Privacy	40%	2	Rs. 16,000/month

Impact:

1 – Catastrophic
2 – Critical
3 – Marginal
4 – Negligible

Sorting the risks on the basis of their risk exposure:

Risk ID	Risk Summary	Probability	Impact	Risk Exposure
1	Data Leak	50%	1	Rs. 70,000/month
2	Server/Application Hacked	60%	1	Rs. 20,000/month
3	Financial Loss Due To Malware	20%	2	Rs. 20,000/month
4	Violation Of Privacy	40%	2	Rs. 16,000/month

Impact:

1 – Catastrophic
2 – Critical
3 – Marginal
4 – Negligible

Risk Exposure:

Risk:

Wrong Accusation
The platform is meant to be used by netizens to upload their photos and videos and take part in various activities and collaborations so that people can get to know each other. Data leak is causing this private data to be leaked to various online hackers and competitiors

Risk Probability: 50%

Risk Impact:

The risk impact would be catastrophic as this would defeat the whole purpose of developing such an application. If the number of the cases of wrong accusation increases, this will lead to the wastage of time and the resources.

Risk Information Sheet:

Risk ID: 1	Date: 02/10/2023	Probability: 50%	Impact: Catastrophic
Description: The platform is meant to be used by netizens to upload their photos and videos and take part in various activities and collaborations so that people can get to know each other. Data leak is causing this private data to be leaked to various online hackers and competitors			
Refinement/Context: To solve the problem of data leak we need to ensure that data remains secure and no fake accounts are made to steal this data .			
Mitigation/Monitoring: 1. Higher security should be maintained 2. Accounts of such users will be blocked.			
Management: RE computed to be Rs. 50,000 per month.			
Current Status: Mitigation steps to be initiated.			
Originator: Vivaan Mansukhani		Assigned: Arnav Malvia	

Risk Exposure:

Risk:
If the system gates hacked then all the data of the company and it's users gets compromised and then people are at risk of getting exposed.

Risk Probability: 60%

Risk Impact:

The risk impact would be catastrophic as it would make it impossible for the company to get back the stolen data and anything can be done using it.

Risk Exposure:

RE = 0.6 x Rs. 33,333 = Rs. 20,000/month.

Risk Information Sheet:

Risk ID: 2	Date: 2/10/2023	Probability: 60%	Impact: Catastrophic
Description: If the system gates hacked then all the data of the company and it's users gets compromised and then people are at risk of getting exposed.			
Refinement/Context: Sub condition 1: The risk impact would be catastrophic as it would make it impossible for the company to get back the stolen data and anything can be done using it.			
Mitigation/Monitoring: 1. Usage of better security 2. Training a better algorithm for data 3. Ensureing two factor verification			
Management: RE computed to be Rs. 20,000 per month.			
Current Status: Mitigation steps to be initiated.			
Originator: Arnav Malvia		Assigned: Rishab mandal	

Risk Exposure:

Risk:
Currently malware cannot be detected by traditional way based anti-malware tools due to their polymorphic and/or metamorphic nature. Here we have improvised a current detection technique of malware based on mining Application Programming Interface (API) calls and developed the first public dataset to promote malware research.

Risk Probability: 20%

Risk Impact:

Malware can cause significant loss and incur substantial costs to organizations. The desire to avoid detection coupled with often lucrative nature of malware development means that there is a high probability that new malware is developed it will likely utilise unknown techniques. Though it can be mitigated by using some firewalls to prevent malware attacks via input details.

Risk Exposure:

RE = 0.3 x Rs. 66,666 = Rs. 20,000/month.

Risk Information Sheet:

Risk ID: 3	Date: 02/10/2023	Probability: 20%	Impact: Critical
Description: Malware attacks can cause significant loss to the organization and can lead to several hours of downtime. Downtime will reduce revenue and would also disappoint the users of the application. It may also lead to leakage of sensitive data of the users.			
Refinement/Context: Sub condition 1: Malware can slow down a user's computer and has the ability to crash some websites. It can infect your computer and use it as a server to broadcast various files or attacks. Sub condition 2: Malware can send emails you did not write getting you or your company in trouble which can result in the company's huge loss. To minimize these attacks firewalls are used.			
Mitigation/Monitoring: 1. Use of antivirus, firewalls and anti-malware software. 2. Monitoring should be in place to verify the security state of: a Update your operating system, browsers, and plugins. b Read the emails with eagle eyes. c Don't believe cold callers. d Don't call fake tech support. e Make sure you're on a secure connection. 3. Use strong passwords or password managers.			
Management: RE computed to be Rs. 20,000 per month.			
Current Status: Mitigation steps to be initiated.			
Originator: Vivaan Mansukhani		Assigned: Rishab Mandal	

Risk Exposure:

Risk:

Violation of Privacy.

In today's world, data is the new oil. Therefore companies, organizations and hackers are always on the lookout for more and more data. This has increased the risk of hackers trying to mine important user data.

Risk Probability: 40%

Risk Impact:

Privacy violation may lead to accounts getting hacked, identity theft, impersonation, targeted ads as well as wrong people seeing the information. This could also harm the reputation of the platform and people would soon lose trust in the application. Information about properties could also be used by other competitors. Contact information about users may be used for marketing purposes.

This can be mitigated by outsourcing cloud security to cloud provider such as Cloudfare.

Risk Exposure:

RE = 0.4 x Rs. 40,000 = Rs. 16,000/month.

Risk Information Sheet:

Risk ID: 4	Date: 03/10/2023	Probability: 40%	Impact: Critical
Description: In today's world, data is the new oil. Therefore companies, organizations and hackers are always on the lookout for more and more data. This has increased the risk of hackers trying to mine important user data.			
Refinement/Context: Sub condition 1: Hackers may attempt to steal personal information of users such as email addresses and passwords. Sub condition 2: Activity log of the users can also be targeted with the intention to analyse this data and provide targeted ads to the user. This is a serious breach of privacy.			
Mitigation/Monitoring: 1. Creation of automatic backups of the database. 2. Monitoring should be in place to verify the security state of: a. DNS records b. SSL certificates c. Web server configuration d. Application updates e. User access f. File integrity			
Management: RE computed to be Rs. 16,000 per month.			
Current Status: Mitigation steps to be initiated.			
Originator: Rishab Mandal		Assigned: Arnav Malvia	

Experiment No. 11

Aim : - Case Study: GitHub for version control.

Theory : -

Whenever a software is built, there is always scope for improvement and those improvements brings changes in picture. Changes may be required to modify or update any existing solution or to create a new solution for a problem. Requirements keeps on changing on daily basis and so we need to keep on upgrading our systems based on the current requirements and needs to meet desired outputs. Changes should be analyzed before they are made to the existing system, recorded before they are implemented, reported to have details of before and after, and controlled in a manner that will improve quality and reduce error. This is where the need of System Configuration Management comes. System Configuration Management (SCM) is an arrangement of exercises which controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. In this way, SCM is a fundamental piece of all project management activities.

Processes involved in SCM – Configuration management provides a disciplined environment for smooth control of work products. It involves the following activities:

1. Identification and Establishment – Identifying the configuration items from products that compose baselines at given points in time (a baseline is a set of mutually consistent Configuration Items, which has been formally reviewed and agreed upon, and serves as the basis of further development). Establishing relationship among items, creating

a mechanism to manage multiple level of control and procedure for change management system.

2. Version control – Creating versions/specifications of the existing product to build new products from the help of SCM system.

3. Change control – Controlling changes to Configuration items (CI).

4. A change request (CR) is submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration objects and system functions, and the projected cost of the change. The results of the evaluation are presented as a change report, which is used by a change control board (CCB) —a person or group who makes a final decision on the status and priority of the change. An engineering change Request (ECR) is generated for each approved change. Also, CCB notifies the developer in case the change is rejected with proper reason. The ECR describes the change to be made, the constraints that must be respected, and the criteria for review and audit. The object to be changed is "checked out" of the project database, the change is made, and then the object is tested again. The object is then "checked in" to the database and appropriate version control mechanisms are used to create the next version of the software.

5. Configuration auditing – A software configuration audit complements the formal technical review of the process and product. It focuses on the technical correctness of the configuration object that has been modified. The audit confirms the completeness, correctness, and consistency of items in the SCM system and track action items from the audit to closure.

6. Reporting – Providing accurate status and current configuration data to developers, tester, end users, customers and stakeholders through admin guides, user guides, FAQs, Release notes, Memos, Installation Guide, Configuration guide etc.

System Configuration Management (SCM) is a software engineering practice that focuses on managing the configuration of software systems and ensuring that software components are properly

controlled, tracked, and stored. It is a critical aspect of software development, as it helps to ensure that changes made to a software system are properly coordinated and that the system is always in a known and stable state.

SCM involves a set of processes and tools that help to manage the different components of a software system, including source code, documentation, and other assets. It enables teams to track changes made to the software system, identify when and why changes were made, and manage the integration of these changes into the final product.

Some GitHub features are:

- Commit
- Graphs
- Pull requests
- Issue Tracking
- Email notifications

Example –

Git Version Control

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of

parallel branches running on different systems). Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Since 2005, Junio Hamano has been the core maintainer. As with most other

distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking

abilities, independent of network access or a central server. Git is free and open-source software distributed under GNU General Public License Version 2.

Some git commands:

• Add commands:

$ git add File-name

• Add all files in specific directory to staging area:

$ git add –all

$ git add docs/*.txt

• Committing changes:

$ git commit -m "Add existing file"

• Pushing changes:

$ git push -u origin main

• Pulling changes:

$ git pull

Output : -


Command Prompt
Microsoft Windows [Version 10.0.22621.2361]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Rishab>git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
[--super-prefix=<path>] [--config-env=<name>=<envvar>]
<command> [<args>]
These are common Git commands used in various situations:
start a working area (see also: git help tutorial)
clone Clone a repository into a new directory
init Create an empty Git repository or reinitialize an existing one
work on the current change (see also: git help everyday)
add Add file contents to the index
mv Move or rename a file, a directory, or a symlink
restore Restore working tree files
rm Remove files from the working tree and from the index
examine the history and state (see also: git help revisions)
bisect Use binary search to find the commit that introduced a bug
diff Show changes between commits, commit and working tree, etc
grep Print lines matching a pattern
log Show commit logs
show Show various types of objects
status Show the working tree status
grow, mark and tweak your common history
branch List, create, or delete branches
commit Record changes to the repository
merge Join two or more development histories together
rebase Reapply commits on top of another base tip
reset Reset current HEAD to the specified state
switch Switch branches
tag Create, list, delete or verify a tag object signed with GPG
collaborate (see also: git help workflows)
fetch Download objects and refs from another repository
pull Fetch from and integrate with another repository or a local branch
push Update remote refs along with associated objects
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

```
C:\Users\Rishab\OneDrive\Desktop\react timepass\twitter>git status
On branch main
Your branch is up to date with 'origin/main'.

It took 2.65 seconds to compute the branch ahead/behind values.
You can use '--no-ahead-behind' to avoid this.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
```

```
C:\Users\Rishab\OneDrive\Desktop\react timepass\twitter>git add .
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/index.html.
The file will have its original line endings in your working directory

C:\Users\Rishab\OneDrive\Desktop\react timepass\twitter>
```

Conclusion : -

Hence, we have understood the implementation and use of GitHub for our given project.

Experiment No. 12

Aim : - Develop test cases for the project using **White Box testing (JUnit).**

Theory : -

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software. The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings. Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer. The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Using this method, Software Engineer can derive test cases that:

o Guarantee that all independent paths within a module have been exercised at least once

o Exercise all logical decisions on their true and false sides,

o Execute all loops at their boundaries and within their operational bounds

o Exercise internal data structures to ensure their validity.

- What are test cases?

A test case is exactly what it sounds like: a test scenario measuring functionality across a set of actions or conditions to verify the expected result. They apply to any software application, can use manual testing or an automated test, and can make use of test case management tools.

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related

standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

A test case document includes test steps, test data, preconditions and the post conditions that verify requirements. Test cases define what must be done to test a system, including the steps executed in the system, the input data values that are entered into the system and the results that are expected throughout test case execution. Using test cases allows developers and testers to discover errors that may have occurred during development or defects that were missed during ad hoc tests.

The benefits of an effective test case include:

o Guaranteed good test coverage. o Reduced maintenance and software support costs. o Reusable test cases.

o Confirmation that the software satisfies end-user requirements. o Improved quality of software and user experience.

o Higher quality products lead to more satisfied customers.

o More satisfied customers will increase company profits.

Overall, writing and using test cases will lead to business optimization. Clients are more satisfied, customer retention increases, the costs of customer service and fixing products decreases, and more reliable products are produced, which improves the company's reputation and brand image.

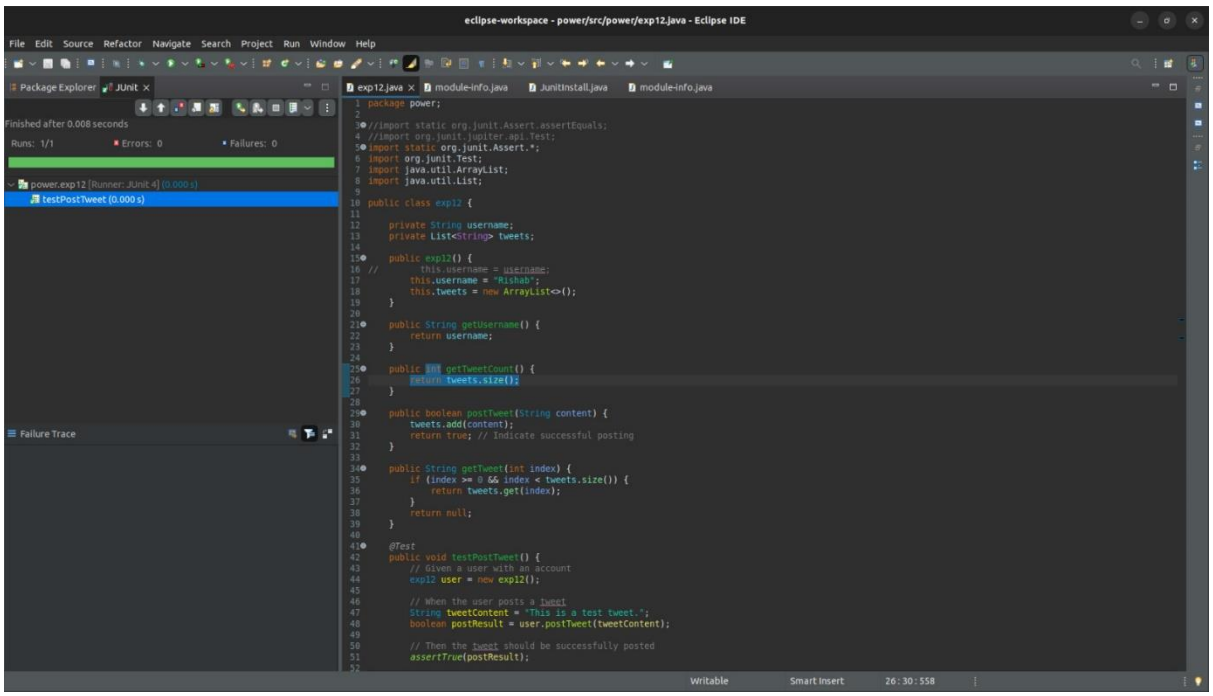
Code:-

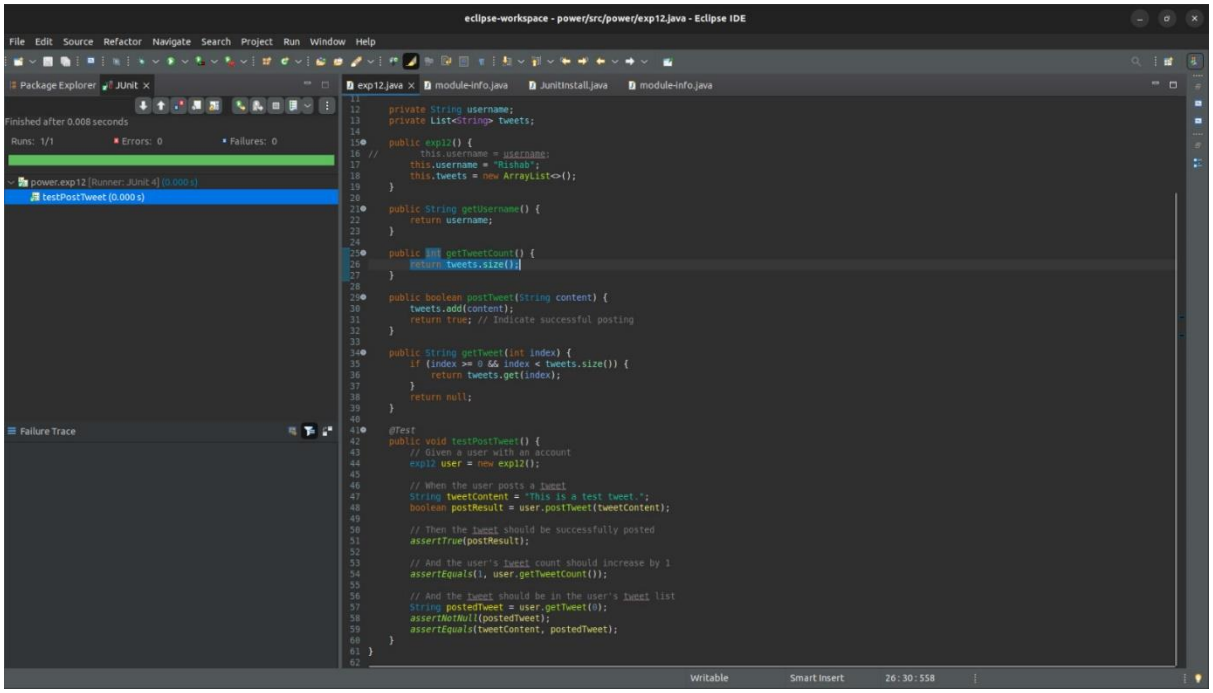
```
package power;


//import static org.junit.Assert.assertEquals;

//import org.junit.jupiter.api.Test;

import static org.junit.Assert.*;

import org.junit.Test;

import java.util.ArrayList;

import java.util.List;


public class exp12 {


    private String username;

    private List<String> tweets;
```

```
    public exp12() {
//        this.username = username;
            this.username = "Rishab";
        this.tweets = new ArrayList<>();
    }

    public String getUsername() {
        return username;
    }

    public int getTweetCount() {
        return tweets.size();
    }

    public boolean postTweet(String content) {
        tweets.add(content);
        return true; // Indicate successful posting
    }

    public String getTweet(int index) {
        if (index >= 0 && index < tweets.size()) {
            return tweets.get(index);
        }
        return null;
    }

    @Test
    public void testPostTweet() {
        // Given a user with an account
        exp12 user = new exp12();
```

```
        // When the user posts a tweet
        String tweetContent = "This is a test tweet.";
        boolean postResult = user.postTweet(tweetContent);

        // Then the tweet should be successfully posted
        assertTrue(postResult);

        // And the user's tweet count should increase by 1
        assertEquals(1, user.getTweetCount());

        // And the tweet should be in the user's tweet list
        String postedTweet = user.getTweet(0);
        assertNotNull(postedTweet);
        assertEquals(tweetContent, postedTweet);
    }
}
```

Output : -





Conclusion : -

Thus, we developed the desired test cases for our existing project using white box testing.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Rishab Mandal
2103110
C23

SE ASSIGNMENT 1

Q. Architectural Design : Explain in detail each type with examples.

→ The software needs the architectural design to represents the design of software. IEEE defines architectural design as "the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system." The software that is built for computer-based systems can exhibit one of these many types of architectural styles.

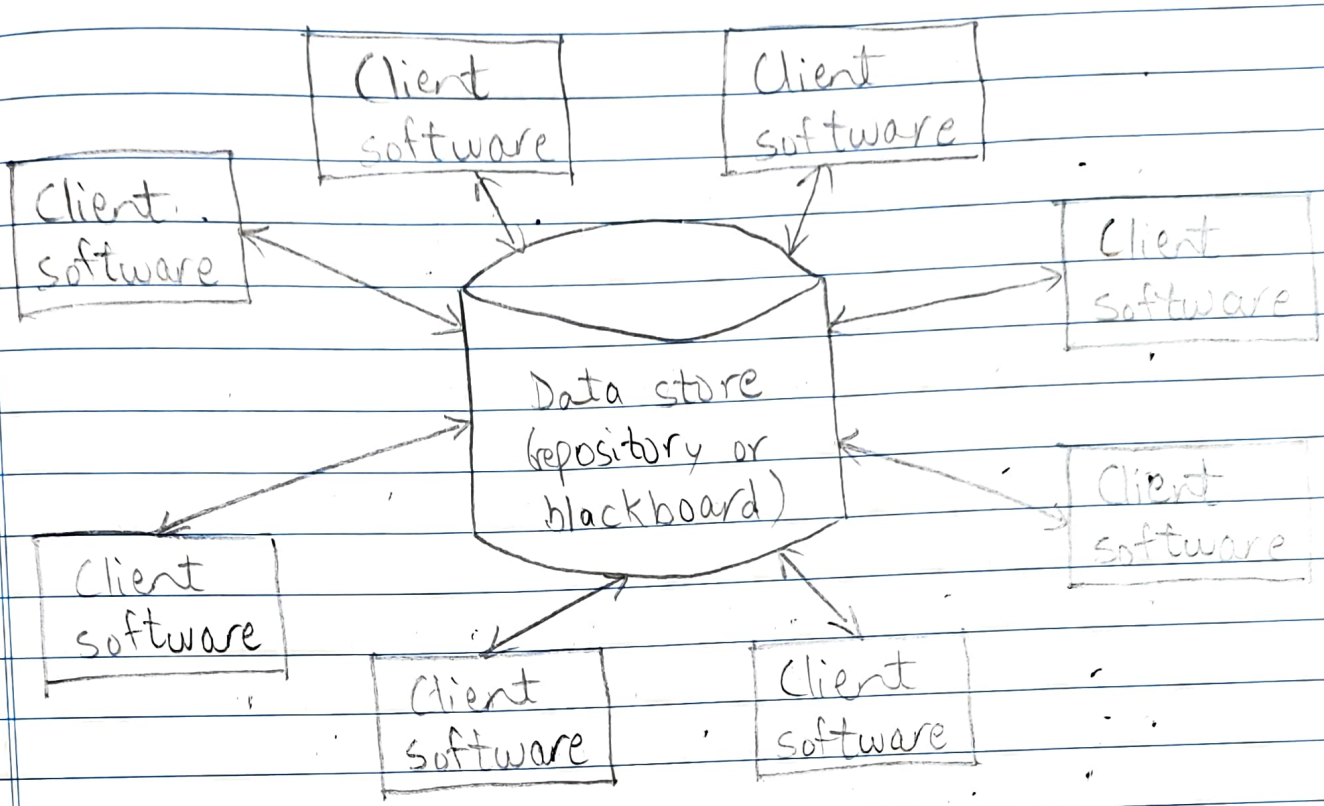
Each style will describe a system category that consists of :

- A set of components (eg. a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in co-ordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

1) Data centered architectures :

A data store will reside at the center of this architecture and is accessed frequently by the other components that update, add, delete or modify the data present within store. The figure illustrates a typical data centered style. The client software access a central repository. Variation of this approach are used to transform the repository into a blackboard when data related to client or data of interest for the client change the notifications to software.
This data-centered architecture will promote integrability. This data means that existing components can be changed and new client components can be added to architecture without permission or concern of clients. Data can be passed among clients using blackboard mechanism. So client components independently executes processes.

Advantages :-
- Repository of data is independent of clients.
- Client work independent of each other.
- It may be simple to add additional clients.
- Modification can be very easy.



2> Data-flow architecture :

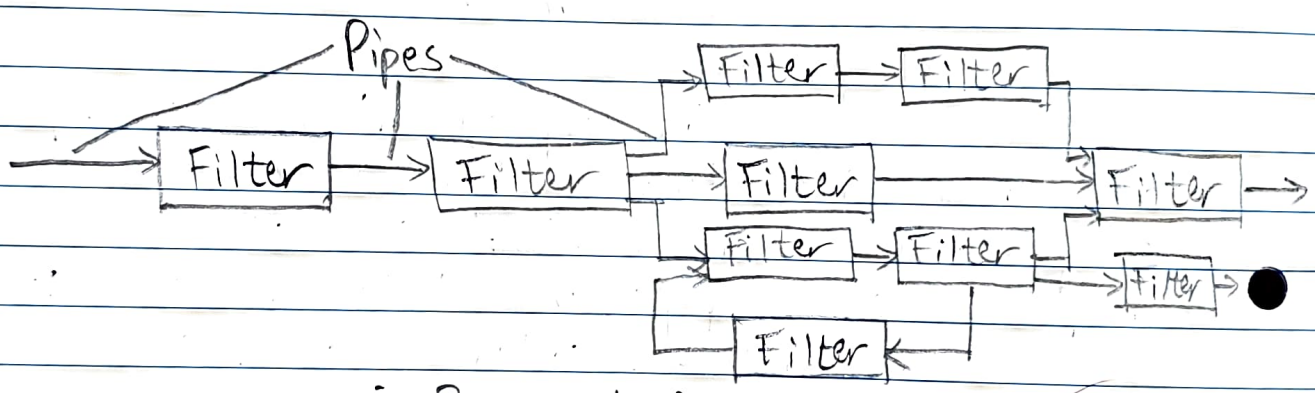
This architecture is applied when the input data are to be transferred through a series of computational or manipulative components into output data. A pipe and filter pattern has a set of components, called filters, connected by pipes that transmit data from component to the next. Each filter works independently and is designed to take data input of a certain form. The filters don't require any knowledge of the working of neighbouring filters.
If the data flow degenerates into a single line of transforms, then it is termed as the batch sequential.

Advantages of Data Flow architecture

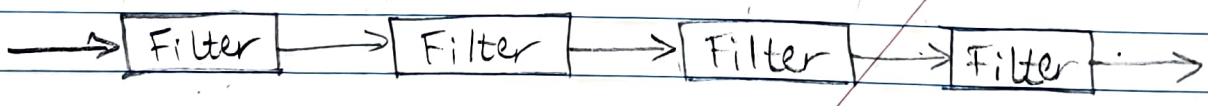
- It encourages upkeep, repurposing, modification.
- With this design, concurrent execution is supported.

Disadvantages of Data Flow architecture

- It frequently degenerates to batch sequential system.
- Data flow architecture does not allow the applications that require greater user engagement.
- It is not easy to coordinate two different but related streams.



Pipes and filters



Batch sequential

3) Call and return architecture :

It is used to create a program that is easy to scale and modify. Many sub-styles exist within this category. Two of them are explained,

- Remote procedure call architecture :
This components is used to present in a main program or sub program architecture distributed among multiple computers on a network.

- Main program or Subprogram architecture :
The main program structure decomposes into number of subprograms or function into a control hierarchy. Main program contains number of subprograms that can invoke other components.

AT
31/10/23

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
Rishab Mandal
C23
2103110

SE ASSIGNMENT 2

Q. Explain in detail : a) Software Maintenance
b) Re-Engineering
c) Reverse Engineering.

→ a] Software Maintenance :-

It refers to the process of modifying and updating a software system after it has been delivered to the customer. This can include bugs, adding new features, improving performance, or updating the software to work with new hardware or software systems. The goal of software maintenance is to keep the software systems working correctly, efficiently and securely and to ensure that it continues to meet the need of the users. Software maintenance is a continuous process, that owns throughout the entire life cycle of the software system. It is important to have a well-defined maintenance process in place, which includes testing and validation, version control and communication with stakeholders.

Several key aspects of Software Maintenance :-

1> Bug Fixing : The process of finding and fixing errors and problems in the software.

2> Enhancements : The process of adding new features or improving existing features to meet

the evolving need of the users.

3) Performance Optimization : The process of improving the speed, efficiency, and reliability of the software.

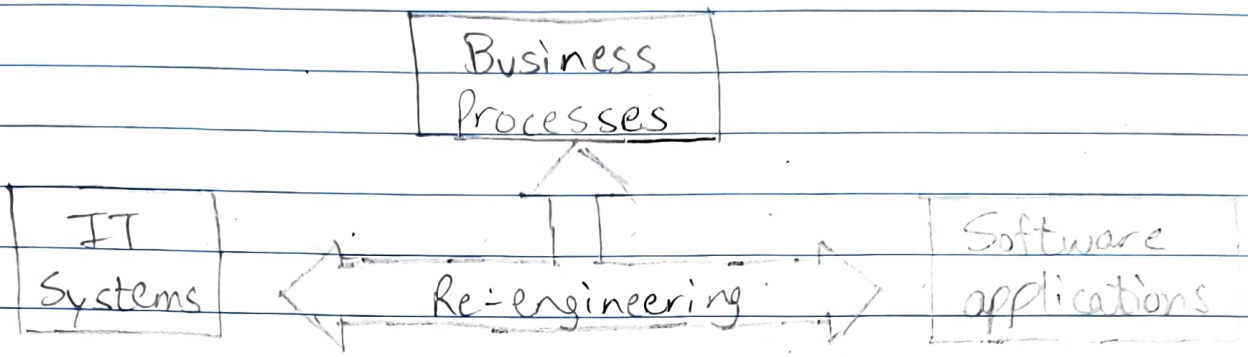
4) Migration : The process of improving and adapting the software to run on new hardware or software platforms.

5) Re-Engineering : The process of improving the design and architecture of the software to make it more maintainable and scalable.

6) Documentation : The process of creating, updating and maintaining the documentation of the software, including user manuals, technical specifications, and design documents.

Software maintenance must be performed in order to :
Correct faults, improve the design, implement enhancement interface with other systems, accomodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
Software maintenance is also an important part of the software Development Life Cycle (SDLC).

b] Re-Engineering :-



Software Re-engineering is a process of software development which is done to improve the maintainability of a system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form. This process encompasses a combination of sub processes like reverse engineering, forward engineering, etc.

The primary goal of software re-engineering is to improve the quality and maintainability of the software system, while minimizing the risks and costs associated with the redevelopment of the system from the scratch. Also, software re-engineering can be initiated for the various reasons, such as :-

1. Improving software quality
2. Updating technology
3. Enhancing functionality
4. Resolving issues

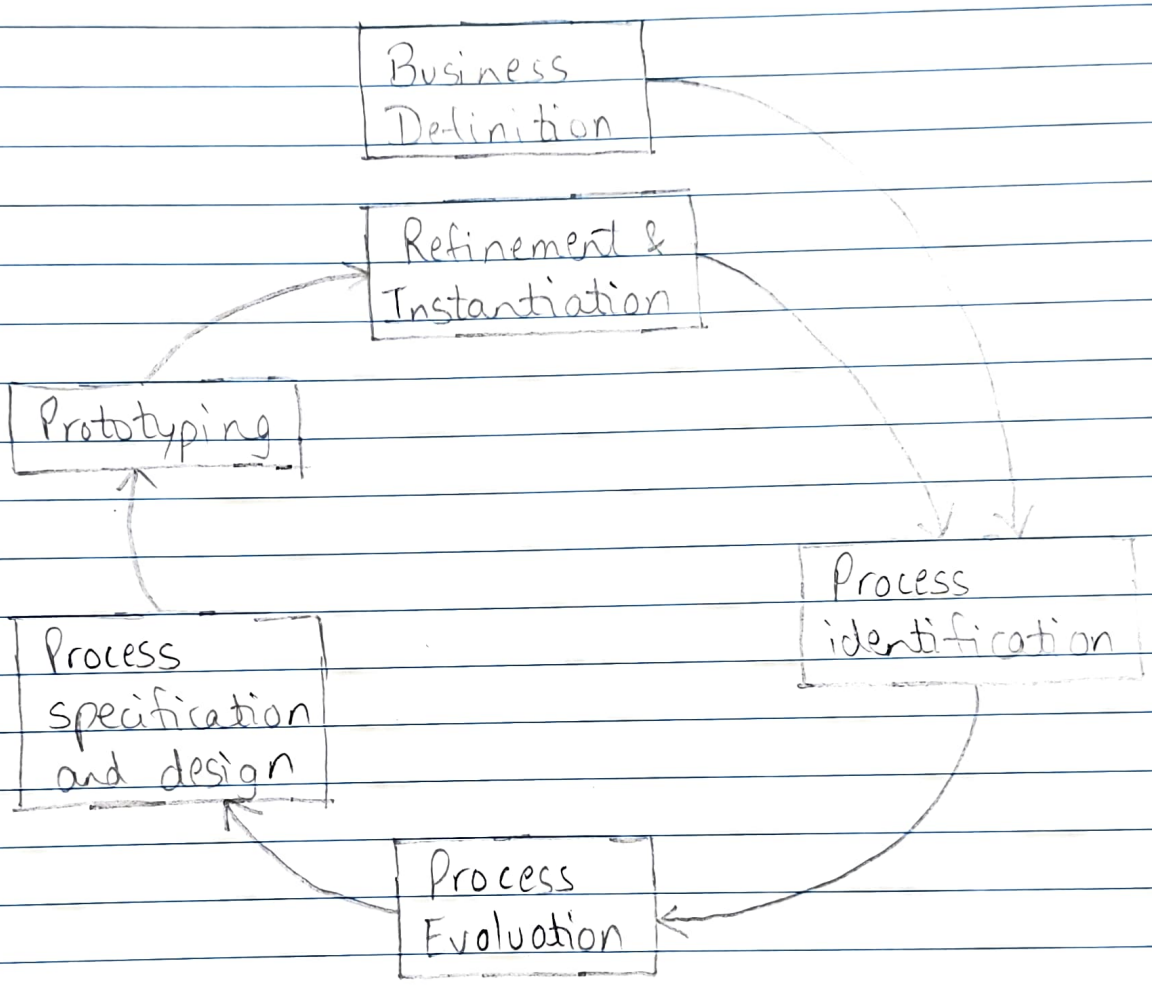
The process of software re-engineering involves the following steps :-

1. Planning :- It helps in identifying the reasons for re-engineering, defining the scope and also establishing the goals and objectives of process.
2. Analysis :- The next step is to analyze the existing system, including the code, documentation and other artifacts.
3. Design :- Based on the analysis, the next step is to implement the changes by modifying the existing code, adding new features, and updating the documentation and other artifacts.
4. Testing :- Once the changes that have been implemented, the software system needs to be tested to ensure that meets the new requirements and specifications.
5. Deployment :- The final step is to deploy the re-engineered software system and make it available to end-users.

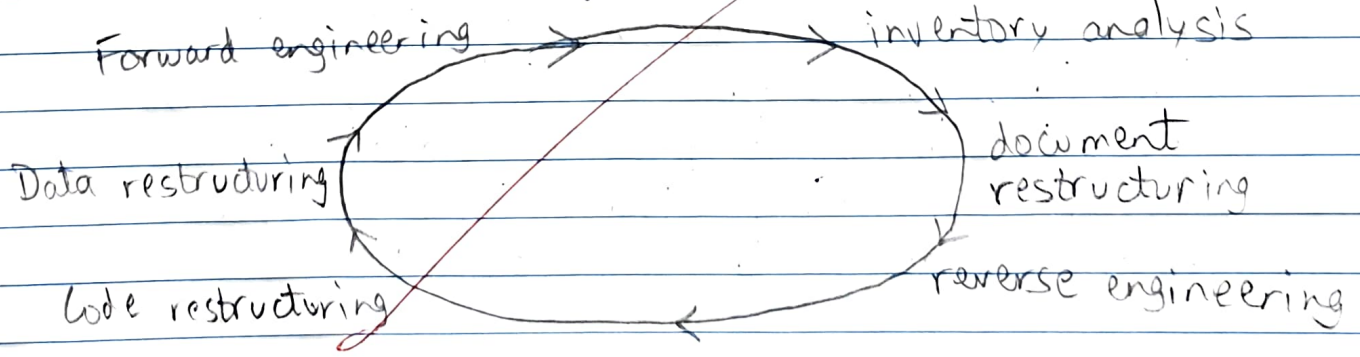
Overall, software re-engineering can be a cost-effective way to improve the quality and functionality of existing software systems, while minimizing the risks and costs associated with starting from scratch.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

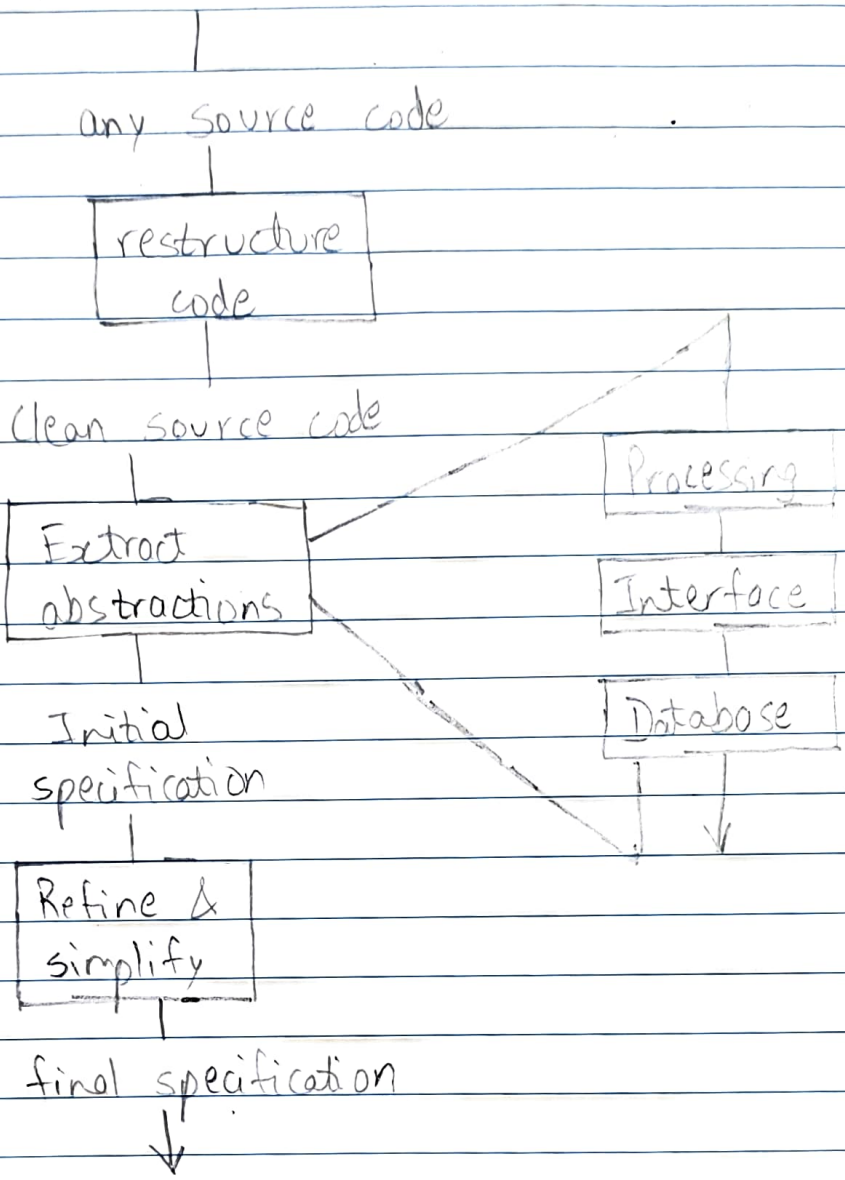
Business Process Re-engineering :- It tells about business definition, process identification, process evaluation, process specification and design, prototyping, refinement.



Software Re-engineering



c] Reverse Engineering :-



Software Reverse Engineering is a process of recovering the design, requirement specifications, and functions of a product from an analysis of its code. It builds a program database and generates information from this. The purpose of reverse engineering is to facilitate the maintenance

work by improving the understandibility of a system and producing the necessary documents for a legacy system.

Reverse Engineering Goals :-

Cope with complexity, recover lost information, detect side effects, synthesize higher abstraction and facilitate reuse.

Steps of Software Reverse Engineering :

1. Collecting Information : This step focuses on collecting all possible information.
2. Examining information : Information ~~caused~~ called in step-1 is studied so as to get familiar with the system.
3. Examining the structure : Identifies program structure in the form of a structure chart, where each node corresponds to some particular routine.
4. Recording the functionality : During this step processing details of each module of structure, charts are recorded using structured language like decision table, etc.
5. Recording data flow : From the information extracted in step-3 and step-4, a set of data flow diagrams is derived to show the flow of data among the processes.

A+
Jain 31/10/23