Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Shirish Shetty

of COMPS Department, Semester V with

Roll No. 2103164 has completed a course of the necessary

experiments in the subject Software Engineering under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In-Charge 20/10/23

Head of the Department

Date 20th Oct 2023

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

Project Title: Animal Shelter Website

Shirish Shetty -2103164

Maithali Shinde-2103166

Adarsh shukla-2103167

EXPERIMENT NO 1

Aim: -

To prepare a detailed statement of problem for the selected mini project and toidentify a suitable software model for the same.

Problem Title: -

Animal Shelter Website Development .

Problem Statement: -

In recent times, there has been a concerning and gradual increase in the number of missing pets, and unfortunately, the success rates of reuniting them with their distressed owners remain dishearteningly low. The limited range of methods availableto locate lost pets, such as traditional approaches like distributing posters or seeking help through social media, has proven insufficient in addressing the issue effectively. These traditional methods often yield limited results due to their restricted reach and lack of organization. Furthermore, even when missing pets are found, the process of reconnecting them with their rightful owners can be hampered by the absence of proper contact information. Moreover, another significant challenge lies in the scarcity of efficient and functional pet search websites. The lack of well-established and accessible platforms for reporting and searching for missing pets hinders the chances of successful reunions. While some owners may resort to posting their search lists on online platforms, the limited availability of dedicated missing pet websites makes it difficult for users to find an extensive network of resources to aid in their search.

Addressing these issues requires a comprehensive approach that encompasses improved public awareness, effective communication channels between pet owners and animal shelters, and the development of user-friendly online platforms dedicatedto missing pet cases. By establishing a well-coordinated system that incorporates technology, social media, and community engagement, we can work together to increase the chances of reuniting missing pets with their worried and loving owners. Additionally, keeping accurate records of missing pet cases and gathering relevant statistics will not only provide a clearer picture of the scale of the problem but also help garner public attention and support for this important cause. Only through collective efforts can we hope to mitigate the distress caused by missing pets and promote responsible pet ownership across communities.

.

.

Traditional Software Model: -

Traditional model for the development of a pet finder website to address the issue of missingpets is the "Spiral Model."

Justification:

Risk Management: The Spiral Model is well-known for its strong emphasis on risk management. When dealing with an issue like missing pets, there are various uncertainties and challenges that need to be addressed. The Spiral Model allows for the early identification and mitigation of risks through its iterative approach, where each iteration involves risk analysis and risk reduction. This ensures that potential obstacles and challenges related to missing pets can be proactively addressed during the development process.

Iterative and Incremental: Like the Unified Process, the Spiral Model is also iterative and incremental. It involves repetitive cycles of planning, risk analysis, engineering, and testing. This iterative nature is well-suited for the development of a pet finder website, where continuous improvement and refinement are essential to create a functional and user-friendly platform.

User Involvement: The Spiral Model encourages extensive user involvement throughout the development process. In the case of a pet finder website, involving pet owners, shelters, and the community is crucial to understanding their needs and preferences. This user-centric approach ensures that the website meets the requirements of its target audience effectively.

Flexibility and Adaptability: The Spiral Model allows for flexibility and adaptability to changes in requirements. As the development team gathers feedback from users and stakeholders, they can incorporate changes and improvements into subsequent iterations. This enables the website to evolve and cater to emerging needs in the ongoing search for missing pets.

Prototyping: The Spiral Model incorporates the concept of prototyping, where early versions of the product are developed to validate ideas and gather feedback. For a pet finder website, prototyping can be beneficial in visualizing the user interface, testing navigation flow, and refining the website's features based on user feedback.

Efficient Resource Allocation: The Spiral Model focuses on allocating resources effectively by prioritizing high-risk areas and features in the initial iterations. For a pet finder website, this means that the most critical functionalities can be developed and tested early on, ensuring that the core features for reuniting missing pets and their owners are given priority.

In conclusion, the Spiral Model offers an effective and systematic approach to developing a pet finder website by addressing risk management, promoting user involvement, allowing for flexibility, and enabling iterative improvements. By following the Spiral Model, the development team can build a reliable and efficient platform to assist in reuniting missing pets with their owners and contribute to the well-being of the pet-loving community.

Agile Software Model:-

Scrum is an Agile framework that focuses on delivering value incrementally and iteratively. Itis well-suited for complex projects with changing requirements, making it a suitable approach for the development of a pet finder website to address the issue of missing pets.

DESCRIPTION:

In the Scrum model, the development process is divided into time-boxed iterations called "sprints." Each sprint typically lasts two to four weeks and involves a cross- functional team working collaboratively to deliver a potentially shippable product increment. The development work is organized and prioritized in a backlog, which is alist of features and tasks ordered by their importance.

During each sprint, the team selects a set of items from the backlog to work on and commits to completing them within the sprint duration. Daily stand-up meetings areheld to promote transparency, communication, and quick problem-solving. At the end of each sprint, a review meeting is conducted to demonstrate the completed work to stakeholders, gather feedback, and plan the next sprint.

Scrum also emphasizes continuous improvement through retrospectives, where theteam reflects on the past sprint to identify areas for improvement and implement changes in subsequent sprints.

Advantages for the Pet Finder Problem:

Iterative Development: Scrum's iterative approach aligns well with the need for continuous improvement when developing a pet finder website. Each sprint produces a working increment of the website, allowing for regular feedback from users and stakeholders, which helps in refining and improving the platform over time.

Flexibility and Adaptability: As the pet finder problem involves uncertainties and changing requirements, Scrum's flexible nature allows the development team to adapt and adjust their approach based on feedback and emerging needs. This ensuresthat the website remains relevant and effective in addressing the issue of missing pets.

Regular Communication: Scrum encourages frequent communication among team members and stakeholders through daily stand-ups, sprint reviews, and retrospectives. This fosters collaboration and ensures that everyone involved in the

project is aligned and informed about the progress and challenges.

Transparency and Accountability: Scrum provides transparency into the development process and progress through the use of visual boards, burndown charts, and regular meetings. This helps in keeping stakeholders informed about the status of the project and fosters a sense of accountability within the development team.

Continuous Improvement: The pet finder problem may require continuous refinement and enhancement of features based on user feedback. Scrum's iterative nature and the focus on continuous improvement through retrospectives make it well-suited for addressing evolving needs and ensuring the website remains effectivein reuniting missing pets with their owners.

In conclusion, the Scrum model offers a structured yet adaptable approach to the development of a animal shelter website, making it a suitable choice for addressing the issue of missing pets effectively. Its iterative development, flexibility, emphasis oncommunication, transparency, and continuous improvement align well with the requirements of the animal shelter problem statement.

EXPERIMENT 2

Aim: Implementing Agile Methodologies for Animal Shelter website.

Introduction: Agile methodologies have revolutionized project management, emphasizing adaptability, collaboration, and continuous improvement. When applied to complex projects like developing a student marketplace app, Agile principles ensure rapid development and an iterative approach, creating a dynamic and user-friendly platform.

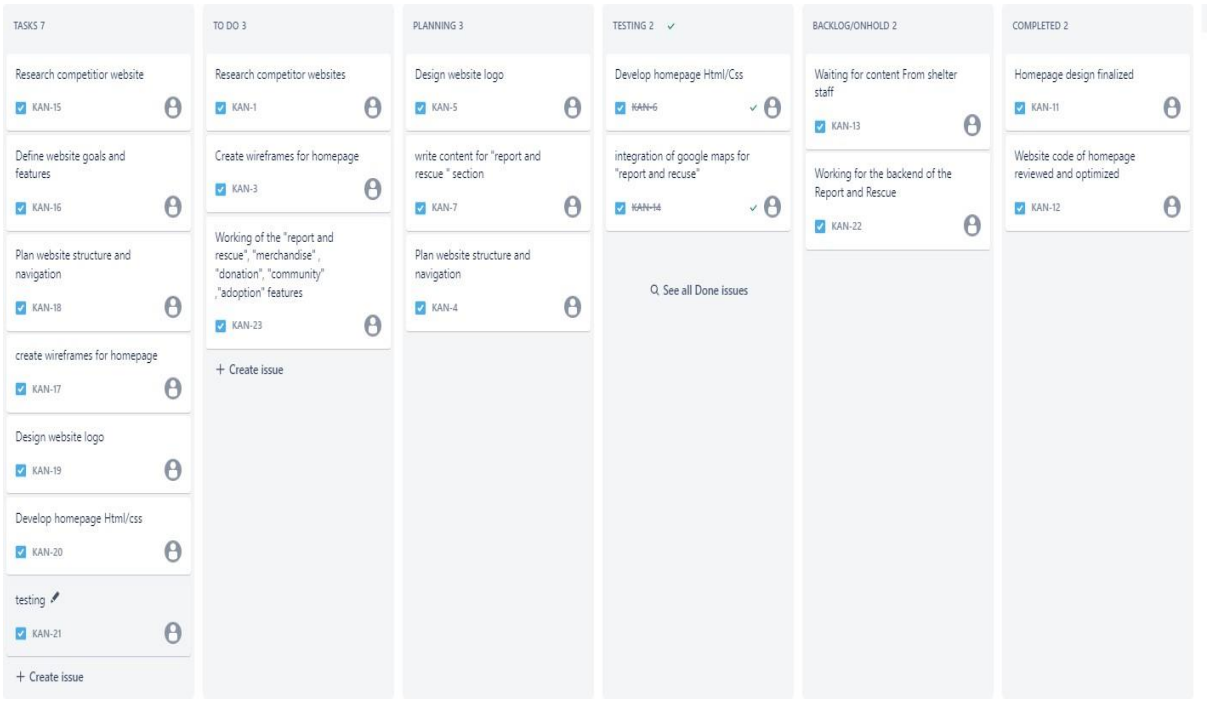
Kanban is a visual management system and methodology for improving workflow efficiency and process management. It originated in Japan and was popularized by Toyota in the 1950s as a part of their lean manufacturing approach. The term "Kanban" itself is Japanese and roughly translates to "visual card" or "signal card." The primary theory behind a Kanban board is to provide a visual representation of work, making it easier to manage and optimize the flow of tasks and activities within a process. Here are the key principles and theories that underpin the Kanban board:

- Visualizing Work: Kanban boards are designed to make work visible. Work items, tasks, or work in progress (WIP) are represented as cards or sticky notes on the board. This visual representation provides a clear picture of what work is in the queue, what is currently being worked on, and what has been completed.
- Limiting Work in Progress (WIP): One of the fundamental principles of Kanban is to limit the amount of work in progress at any given time. This is often done by setting WIP limits for each stage of the process on the Kanban board. By doing this, teams can focus on completing work before taking on new tasks, which helps prevent overloading, bottlenecks, and improves overall flow and productivity.
- Pull System: Kanban operates on a pull-based system. Instead of work being pushed onto team members, work is pulled by team members based on their capacity and the availability of WIP slots on the board. This ensures that work is completed at a sustainable pace and helps maintain a smooth workflow.
- Continuous Improvement (Kaizen): Kanban encourages a culture of continuous improvement. By regularly reviewing and optimizing the Kanban board, teams can identify areas for improvement, reduce cycle times, and increase overall efficiency. This aligns with the broader Lean and Agile principles of constant adaptation and optimization.
- Flow Management: Kanban emphasizes the importance of maintaining a smooth flow of work from start to finish. This involves identifying and addressing bottlenecks or delays in the process to keep work moving at a consistent pace.
- Feedback Loops: Kanban boards provide a clear mechanism for feedback and communication within the team. Team members can easily see the status of work items and collaborate to resolve issues or blockers, promoting transparency and effective teamwork.

- Flexibility: Kanban is adaptable and can be applied to various types of processes, not just software development or manufacturing. Whether you're managing a marketing campaign, IT support, or any other type of work, Kanban can be tailored to your specific needs.
- Empowering Teams: Kanban decentralizes decision-making by allowing team members to self-organize and make choices about what work to pull next. This empowerment can lead to increased motivation and engagement.

In summary, the theory behind a Kanban board is rooted in the principles of visualization, limiting work in progress, maintaining a pull system, and continuously improving processes. It provides a simple yet effective way to manage and optimize workflows, making it a valuable tool for a wide range of industries and teams.

KANBAN BOARD FOR ANIMAL SHELTER WEBSITE:



Software Requirements Specification

for

ANIMAL SHELTER WEBSITE

Version 1.0 approved

Prepared by

Maithili Shinde

Shirish Shetty

Adarsh Shukla

Thadomal Shahani Engineering College

28 July,2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of our website is to rescue and protect the lives of countless dogs and cats facing homelessness and potential harm each year. By creating a robust online community of animal enthusiasts and caretakers, we aim to provide a centralized platform for locating and supporting these vulnerable animals. Through donations, volunteering, and essential resources, our web app strives to reduce the heartbreaking number of animal casualties and ensure that every animal finds a safe and loving forever home.

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 is the team of developers who will be designing and implementing the Pet Finder Website. Also, the document is to be utilized by the testing team who will be testing and evaluating the performance and design of the application. The document consists of all the necessary information that will be required by the team of software engineers who will be working on the project.

1.4 Product Scope

The Animal Shelter Website aims to create a user-friendly platform that simplifies pet adoption, connects pet lovers, and facilitates rescue operations. The web-based application will be mobile responsive, ensuring accessibility on all electronic devices. Key features include a report and rescue function, allowing users to report distressed animals and coordinate rescue efforts. A chatbot will assist visitors in answering queries and guiding them through the adoption process. The website will have a dedicated community page, fostering connections among pet enthusiasts, while a merchandise and donation page will offer opportunities to support the cause. Future enhancements will introduce a payment gateway for transactions and an online pet adoption agreement feature, streamlining the adoption process and promoting responsible pet ownership.

1.5 References

Websites:

- *NodeJs Documentation: https://nodejs.org/en/docs*
- *ExpressJs Documentation: https://expressjs.com/en/5x/api.html*

2. Overall Description

2.1 Product Perspective

The Animal Shelter Website provides a modern and efficient alternative for pet adoption, connecting pet lovers, and coordinating rescue efforts without the need for intermediaries. The application aims to streamline the process of reporting and rescuing distressed animals through its "Report and Rescue" feature. A chatbot will enhance user experience by providing instant assistance and guidance during the pet adoption process. The platform will serve as a vibrant community hub for pet enthusiasts, fostering connections and interactions. Additionally, the website will feature dedicated sections for adopting pets, purchasing merchandise, and making donations, empowering users to support the cause directly. By eliminating intermediaries, the website ensures that interactions and agreements between adopters and shelters are more direct and transparent.

2.2 Product Functions

- "Report and Rescue" function to enable users to report distressed animals and coordinate rescue efforts.
- Chatbot feature providing instant assistance and guidance during the pet adoption process.
- Pet adoption section allowing users to browse and adopt pets from the shelter.
- Community page for pet lovers to connect, share experiences, and engage in discussions.
- Merchandise page offering pet-related products for purchase to support the shelter.
- Donation page to facilitate direct contributions to the shelter's cause.
- Transparent and direct communication between adopters and shelters for clarifications and agreements.
- Mobile responsiveness for convenient access on all electronic devices.
- User-friendly interface for effortless property searching and finding potential pet adopters.
- Streamlined platform that eliminates the need for intermediaries, reducing costs for both parties involved.

2.3 User Classes and Characteristics

The Animal Shelter Website anticipates the following user classes:

1. Pet Seekers and Adopters:Individuals seeking to adopt a pet from the animal shelter. They may have varying technical expertise but should be comfortable navigating the user-friendly interface. Their primary goal is to find and adopt a pet.

2. Animal Welfare Volunteers and Rescuers:Experienced animal welfare volunteers or rescue organizations proficient in using reporting and rescue functionalities. They may require elevated security privileges for coordination.

3. Pet Enthusiasts and Community Members:Users interested in engaging with a supportive pet-loving community. They contribute to discussions and share experiences.

4. Donors and Supporters: Individuals making monetary donations or purchasing merchandise to support

the shelter's cause. They require secure payment options and straightforward procedures.

5. Administrators and Staff:Shelter personnel responsible for website management, processing adoptions, and moderating discussions. They need high-level security privileges and technical expertise.

The most critical user classes for the product are Pet Seekers and Adopters, Animal Welfare Volunteers and Rescuers, and Donors and Supporters, as they directly contribute to the primary goals of pet adoption, rescue operations, and fundraising. Pet Enthusiasts and Community Members add value by fostering a supportive environment, while Administrators and Staff play a vital role in website management and coordination.

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: 64 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

CO-1:
The time allotted for this project is at most 3 months.

CO-2:
The front end of the application will be made using HTML, CSS and JavaScript.

CO-3:
Node.js will be used as the language for the backend of the application and MongoDB will be used for the database of the application.

CO-4:
The website will be in English language. Users who do not know English will face difficulties in using the website.

2.6 User Documentation

The Animal Shelter Website is designed to provide a seamless user experience, and appropriate instructions will be available at every step to ensure users encounter no difficulties while navigating the application. Future enhancements will include the addition of a helpful chatbot, providing guidance to users if they encounter any challenges. Throughout the process of filling out forms, adding photos, and specifying locations, detailed instructions will be provided to facilitate smooth interactions. In case of any inadvertent errors or incorrect information input, the application will display clear and informative error messages to guide users towards corrective actions. Our user documentation is aimed at making the adoption process, community engagement, and supporting the shelter's cause accessible and straightforward for all users.

2.7 Assumptions and Dependencies

AS-1:
It is assumed that users accessing the website have basic computer literacy and internet connectivity to navigate the application.
AS-2:
The successful implementation of the "Report and Rescue" map feature depends on the availability and proper functioning of the Google Maps API. Any changes or disruptions to the API might impact the map functionality.
AS-3:
The application supports only English language. We assume the users of the application will be well versed with English.

AS-4:
The users of the application should have basic knowledge of uploading images and location.

DE-1:
The frontend interface is built using HTML, CSS, and JavaScript, and their proper functioning is dependent on browser compatibility and adherence to web standards.
DE-2:
The "Report and Rescue" map feature requires the integration of Google Maps API to enable users to locate and report distressed animals accurately. The proper functioning of this feature depends on the reliable availability of the Google Maps API.
DE-3:
The backend of the application relies on Node.js and Express for server-side operations, routing, and handling user requests.

3. External Interface Requirements

3.1 User Interfaces

UI-1:
The website will start with a landing page. The landing page will have all information about the animal shelter
UI-2:
There will be a navigation bar at the top of the web page which will help users to navigate to different web pages.
UI-3:
Instructions will be provided to the users on top of forms to be filled.
UI-4:
The user will have to login in order to access the report, rescue, and community page.
UI-5:
The interface will be responsive for all screen sizes as much as possible to provide the users a seamless experience.

3.2 Hardware Interfaces

N/A

3.3 Software Interfaces

- *Browsers: Mozilla Firefox 30+, Google Chrome 27.0+ are the preferred browsers.*
- *Operating System: Android, Windows 7, 8, 10, Mac OS, Linux distributions*

3.4 Communications Interfaces

*The application will be using **HTTPS** protocols.*

4. System Features

4.1 Report and Rescue Feature

4.1.1 Description and Priority:

The "Report and Rescue" feature allows verified users to report injured or distressed animals by uploading relevant details and marking the location on a map, utilizing Google Maps APIs for accuracy. This feature enables the animal shelter to promptly respond to rescue requests and coordinate rescue operations effectively.

4.1.2 Response Sequences:

1. User logs in or signs up as a verified user.

2. User navigates to the "Report and Rescue" section.

3. User provides information about the distressed animal, including species, condition, and any relevant details.

4. User marks the animal's location on the integrated Google Maps.

5. The system verifies the user's credentials and the submitted information.

6. Upon successful verification, the system registers the rescue request and sends a notification to the animal shelter.

4.1.3 Functional Requirements:

REQ-1: The system shall authenticate users before allowing them to access the "Report and Rescue" feature.

REQ-2: Users must provide mandatory details about the distressed animal, such as species, condition, and location, when submitting a rescue request.

REQ-3: The system shall integrate Google Maps APIs to enable users to mark the precise location of the distressed animal on the map.

REQ-4: The system shall validate and verify the submitted information and the user's credentials before processing the rescue request.

REQ-5: If the user is not a verified user, the system shall prompt them to complete the verification process or sign up as a verified user.

REQ-6: In case of invalid or missing information, the system shall display appropriate error messages guiding the user to provide the necessary details.

REQ-7: Once a rescue request is successfully submitted, the system shall send real-time notifications to the animal shelter, notifying them of the emergency.

4.2 Chatbot Feature

4.2.1 Description and Priority:

This feature will help to enhance user experience by offering instant assistance, answering queries, and clearing doubts about the website's features and functionalities.

4.2.2 Response Sequences:

User initiates a conversation with the Chatbot by clicking on the "Chat" button.The Chatbot greets the user and prompts them to ask any questions or seek help. User asks a specific question or requests assistance related to pet adoption, report and rescue, community page, merchandise, donations, or any other feature. The Chatbot provides relevant responses, guiding the user through the requested information or directing them to the appropriate sections of the website. If the Chatbot encounters an ambiguous query or cannot provide an answer, it prompts the user to rephrase the question or suggests contacting support staff for further assistance.

4.2.3 Functional Requirements:

REQ-1: The Chatbot must be continuously updated with the latest information to provide accurate responses to users' questions.

REQ-2: It should offer an option for users to switch to live human support if needed, with appropriate contact details provided.

Note: TBD (To Be Determined) for any missing or unspecified requirements.

4.3 Community Feature:

4.3.1 Description and Priority:

The Community Page feature allows users to interact with other pet lovers through posting and accessing relevant information such as pet blogs and veterinarian details. Users will need to log in to access this section, ensuring a safe and engaging environment for pet enthusiasts to connect and share experiences. The priority for this feature is set to Medium, as it plays a significant role in fostering community engagement and enhancing user experience on the website. However, it is not as critical as features directly related to pet adoption or rescue operations.

4.3.2 Response Sequences:

1. *User logs in to the website.*
2. *User navigates to the Community Page section.*
3. *User posts a message or engages in discussions with other pet lovers.*
4. *User accesses pet blogs, veterinarian information, and other relevant content.*

4.3.3 Functional Requirements:

REQ-1: The website shall have a secure user authentication system to enable login and access to the

Community Page feature.

REQ-2: Users must be able to post messages and engage in discussions on the Community Page.

REQ-3: The website should provide access to pet-related blogs and veterinarian details for logged-in users.

REQ-4: When a user attempts to access the Community Page without logging in, a prompt to log in or create an account should be displayed.

REQ-5: User posts and discussions should be subject to content moderation to ensure a safe and positive user experience.

REQ-6: The website should support real-time updates or notifications for new posts and responses on the Community Page.

4.4 Adoption Page Feature:

4.4.1 Description :

The Adoption Page is a high-priority feature that provides a visually appealing and user-friendly interface showcasing pictures of pets available for adoption. It includes essential information such as breed, age, and other relevant details to help potential adopters make informed decisions. Additionally, the page offers options for users interested in volunteering at the animal shelter.

Priority: High (Benefit: 8, Penalty: 2, Cost: 5, Risk: 4)

4.4.2 Response Sequences:

1. User navigates to the Adoption Page.

2. System displays a grid layout with pictures of available pets and their details.

3. User clicks on a pet to view more information.

4. System presents detailed information about the selected pet, including breed, age, and temperament.

5. User selects the "Volunteer" option.

6. System redirects the user to the volunteering section, prompting them to fill out necessary details for volunteering.

4.4.3 Functional Requirements:

REQ-1: The Adoption Page must display a grid layout of pet pictures with their respective details, including breed, age, and any special notes.

REQ-2: Clicking on a specific pet should lead to a dedicated page with comprehensive information about that pet, assisting users in making an informed adoption decision.

REQ-3: The "Volunteer" option must redirect users to a separate section with a form to capture their details and interest in volunteering at the animal shelter.

REQ-4: The system should handle error conditions gracefully, displaying appropriate error messages if

users encounter issues while browsing or attempting to volunteer.

REQ-5: The Adoption Page should have an intuitive and responsive design to ensure compatibility with various devices and screen sizes.

REQ-6: The adoption details and pet pictures should be regularly updated by the system administrators to reflect the most current availability of pets for adoption.

REQ-7: Users should be able to filter pet listings based on criteria such as species, age, and location to streamline their search process. (TBD - To Be Determined)

4.5 Support Us Feature:

4.5.1 Description and Priority:

The "Support Us" feature allows users to contribute to the animal shelter's cause by either purchasing merchandise or making monetary donations. Users will have the option to click on the "Merchandise" link to be redirected to the merchandise page, where they can purchase pet-related products to support the shelter. Alternatively, users can choose to donate money directly to the shelter's cause.

Priority: Medium

4.5.2 Response Sequences:

1. *User clicks on the "Support Us" option from the main navigation menu.*
2. *The system presents two choices: "Merchandise" and "Donate Money."*
3. *If the user clicks on "Merchandise," they are redirected to the merchandise page.*
4. *If the user selects "Donate Money," the system displays the donation page.*

4.5.3 Functional Requirements:

REQ-1: The system shall provide a clear and accessible "Support Us" option in the main navigation menu.

REQ-2: The system shall offer two choices, "Merchandise" and "Donate Money," within the "Support Us" feature.

REQ-3: If the user clicks on "Merchandise," the system shall redirect them to the merchandise page.

REQ-4: If the user selects "Donate Money," the system shall direct them to the donation page.

REQ-5: The merchandise page shall display a variety of pet-related products available for purchase.

REQ-6: The donation page shall allow users to enter the desired amount they wish to donate.

REQ-7: The system shall process donation transactions securely using an integrated payment gateway.

REQ-8: In case of errors during transactions, the system shall display informative error messages to guide users through the correction process.

REQ-9: The system shall record and store transaction details for administrative purposes and provide donation receipts to users upon successful transactions.

REQ-10: The "Support Us" feature shall have a secure connection (HTTPS) to ensure the privacy and safety of user data during transactions.

Please note that the requirements listed above are placeholders (TBD) and need further specification, especially regarding the integrated payment gateway and transaction processing details.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Performance Requirements for the pet finder website should focus on delivering a fast, reliable, and scalable platform to effectively reunite missing pets with their owners. The system should ensure quick response times during user interactions, smooth handling of simultaneous searches, and efficient data processing. The website must be able to handle a growing user base and increasing search queries without compromising performance. Load times for pages and search results should be optimized to enhance user experience. Additionally, the system should be robust and capable of handling peak traffic periods, ensuring that pet owners and shelters can access and contribute to the platform seamlessly.

5.2 Safety Requirements

- *Conduct regular security audits and vulnerability assessments to identify and address potential security weaknesses.*
- *Backup power supply should be present for server, so that it does not stop functioning in case of power failure.*
- *Limit access to website administration functions only to authorized personnel.*
- *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.*

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*

EXPERIMENT NO 4

AIM: Develop a data flow diagram for the project

THEORY:

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.
It is a graphical tool, useful for communicating with users ,managers and other personnel. it is useful for analyzing existing as well as proposed system.

It provides an overview of

- What data is system processes.
- What transformation are performed.
- What data are stored.
- What results are produced , etc.

Components of DFD:
The Data Flow Diagram has 4 components:

- **Process** Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- **Data Flow** Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
- **Warehouse** The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.
- **Terminator** The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

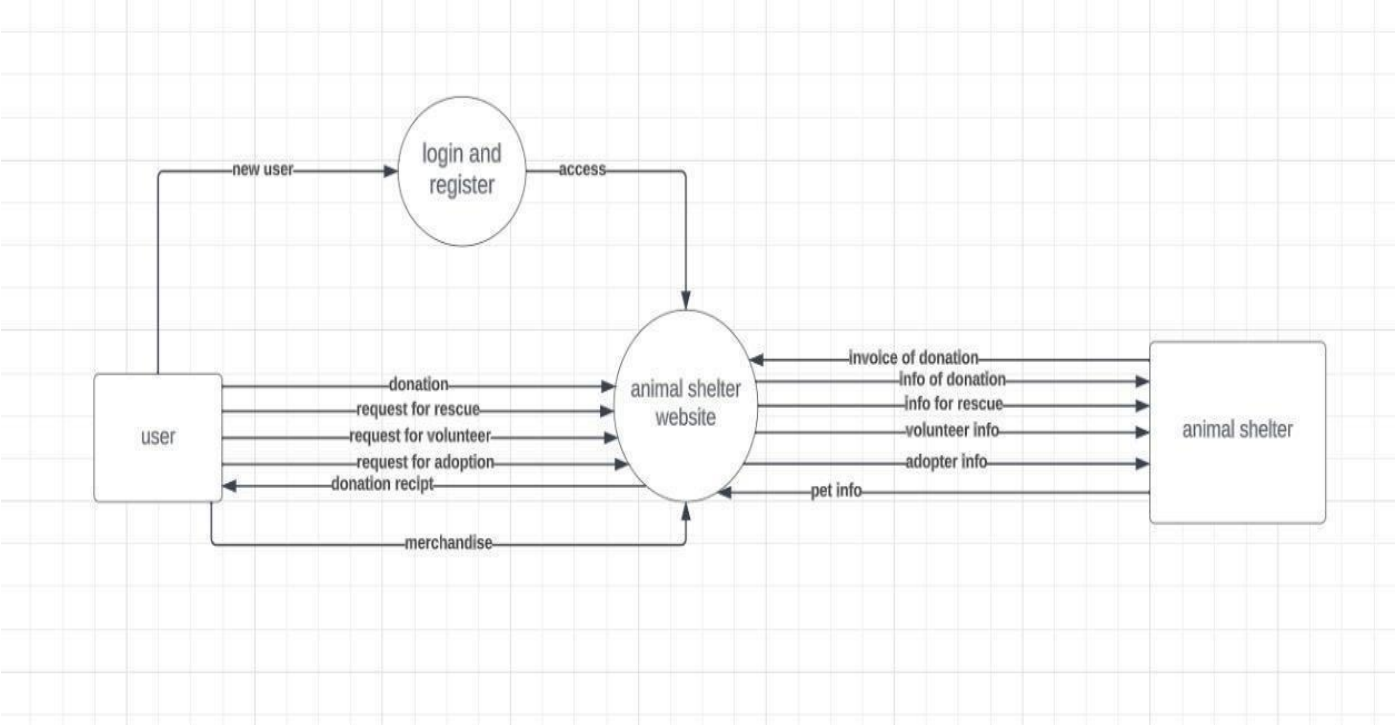
Rules for creating DFD

- The name of the entity should be easy and understandable without any extra assistance(like comments).
- The processes should be numbered or put in ordered list to be referred easily.
- The DFD should maintain consistency across all the DFD levels.
- A single DFD can have a maximum of nine processes and a minimum of three processes.

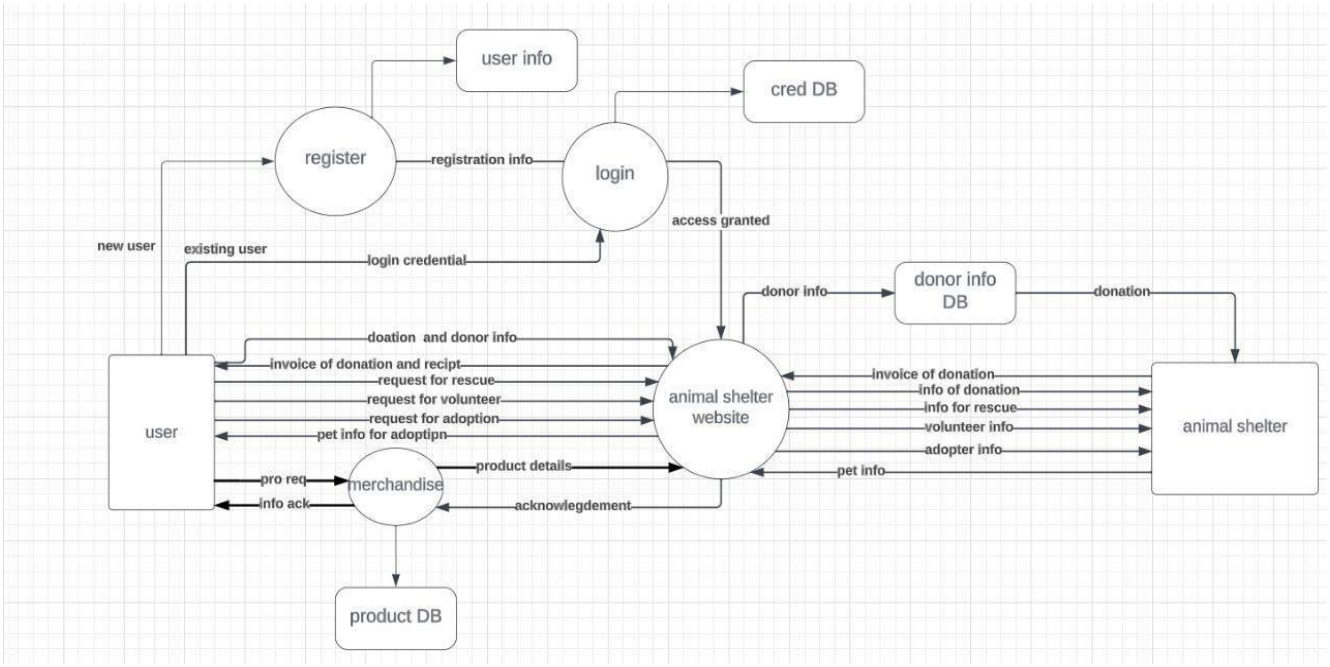
Symbols Used in DFD

- **Square Box:** A square box defines source or destination of the system. It is also called entity. It is represented by rectangle.
- **Arrow or Line:** An arrow identifies the data flow i.e. it gives information to the data that is in motion.
- **Circle or bubble chart:** It represents as a process that gives us information. It is also called processing box.
- **Open Rectangle:** An open rectangle is a data store. In this data is store either temporary or permanently.

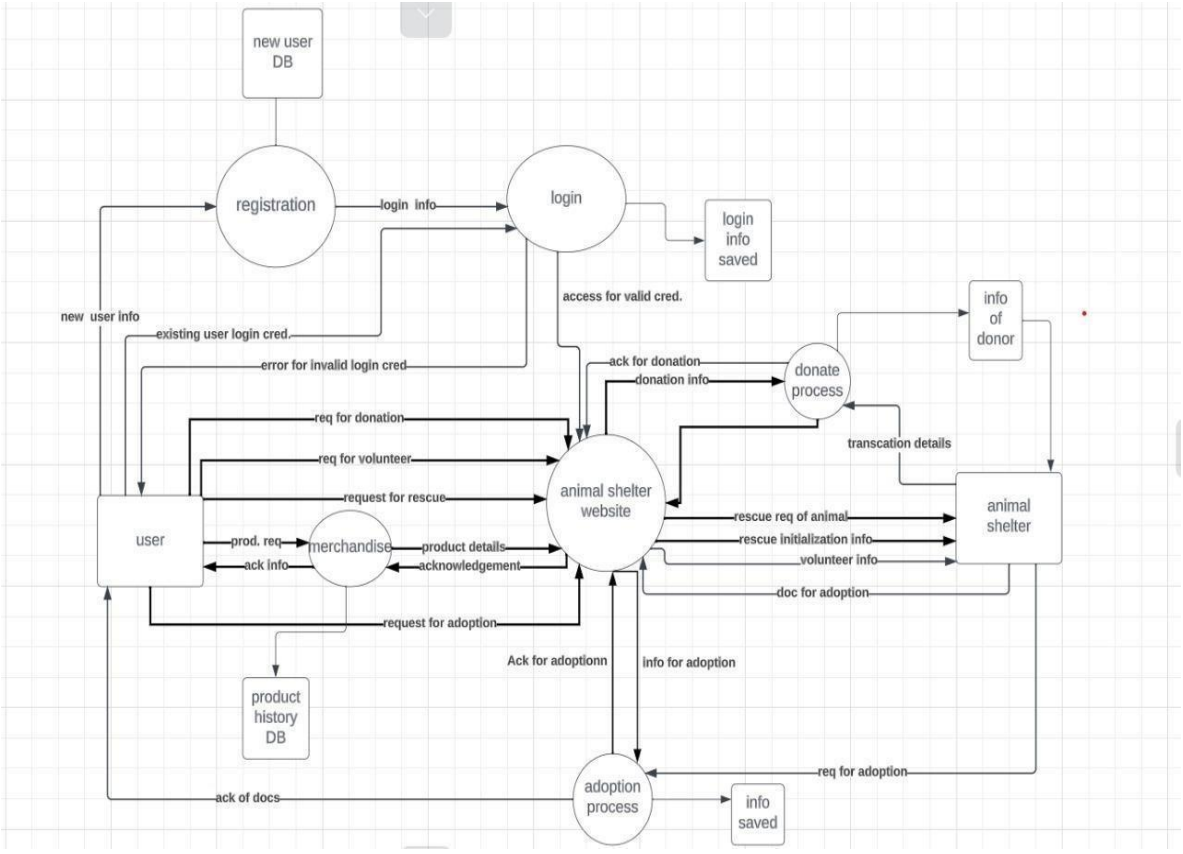
OUTPUT:



1) DFD 0


user info
cred DB
register
registration info
login
access granted
new user
existing user
login credential
donor info
donor info DB
donation
doation and donor info
invoice of donation and recipt
request for rescue
request for volunteer
request for adoption
pet info for adoptipn
user
animal shelter website
invoice of donation
info of donation
info for rescue
volunteer info
adopter info
pet info
animal shelter
pro req
info ack
merchandise
product details
acknowlegdement
product DB

2) DFD 1


new user DB
registration
login info
login
login info saved
access for valid cred.
new user info
existing user login cred.
error for invalid login cred
info of donor
ack for donation
donation info
donate process
req for donation
req for volunteer
request for rescue
transcation details
animal shelter website
animal shelter
user
prod. req
ack info
merchandise
product details
acknowledgement
rescue req of animal
rescue initialization info
volunteer info
doc for adoption
request for adoption
Ack for adoptionn
info for adoption
product history DB
req for adoption
ack of docs
adoption process
info saved

3) DFD 2

EXPERIMENT NO 5

AIM: Develop Activity and State diagram for the project

THEORY:

State-Activity Diagram (SAD) is a graphical representation used in software engineering to depict the behavior and interactions within a system or software application. It provides insight into how different components of the system communicate and the sequence of activities that occur. SAD is particularly useful for communicating with developers, designers, and stakeholders involved in the software development process. It can be applied to both existing systems for analysis and proposed systems for design.

A State-Activity Diagram has four main components:

1. Activity: Activities represent processes or actions within the system. These are depicted as rectangular boxes, ovals, rectangles, or circles, just like in DFD. Each activity is named descriptively to convey its purpose or function within the system. Activities define what the system does and how it processes data.

2. State: States represent the different conditions or modes that the system can be in during its operation. States are often represented as circles or rounded rectangles. They describe the various situations or statuses that the system can transition between. States capture the current state of the system at a given point in time.

3. Transition: Transitions are depicted as arrows or lines and describe how the system moves from one state to another. Transitions indicate the flow of control or data between activities and states. They show the sequence in which activities and states are executed or entered.

4. Store: Similar to the data store in DFD, a store in the State-Activity Diagram represents where data or information is stored within the system. It can be temporary or permanent storage and is typically represented by two horizontal lines. Stores are used to depict data storage and retrieval operations within the system.

Rules for creating a State-Activity Diagram:

1. Clear Naming: Ensure that the names of activities and states are concise and easy to understand without additional explanations. This makes it easier for stakeholders to grasp the diagram's meaning.

2. Sequential Order: If needed, number activities or list them in a logical sequence to aid in understanding the flow of operations within the system.

3. Consistency: Maintain consistency in the representation of components across all levels of the diagram. This consistency helps in avoiding confusion and ensuring clarity.

4. Limit Processes: While there is no strict limit, it's generally advisable to keep the number of activities reasonable to maintain diagram readability. A single diagram might have several activities, but it's essential not to overwhelm the viewer.

Symbols Used in State-Activity Diagram:

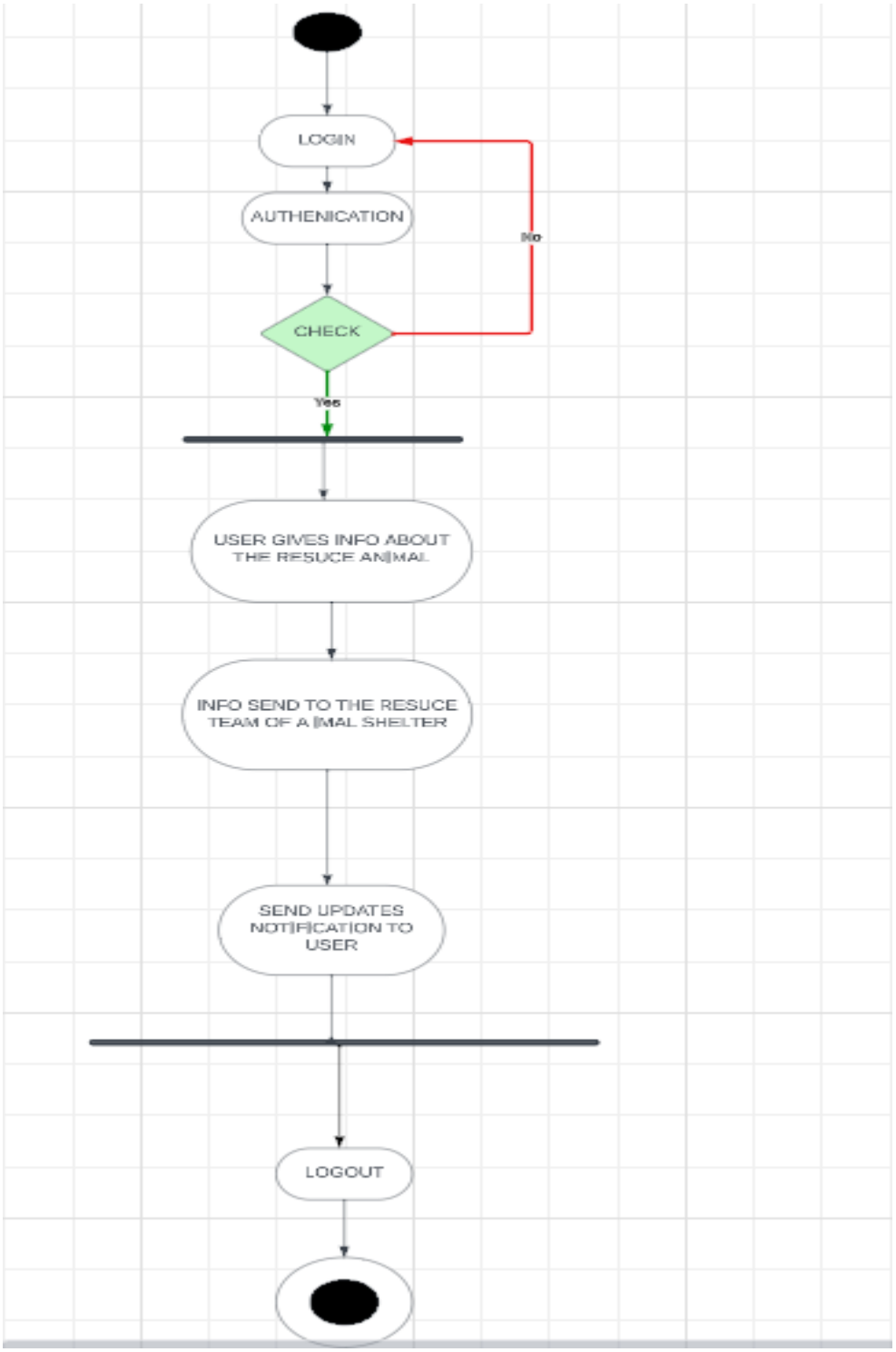
- Rectangle: Represents an activity or process within the system, similar to a DFD process symbol.

- Circle or Rounded Rectangle: Represents a state or condition that the system can be in, similar to DFD states like "data reading" or "data entry."

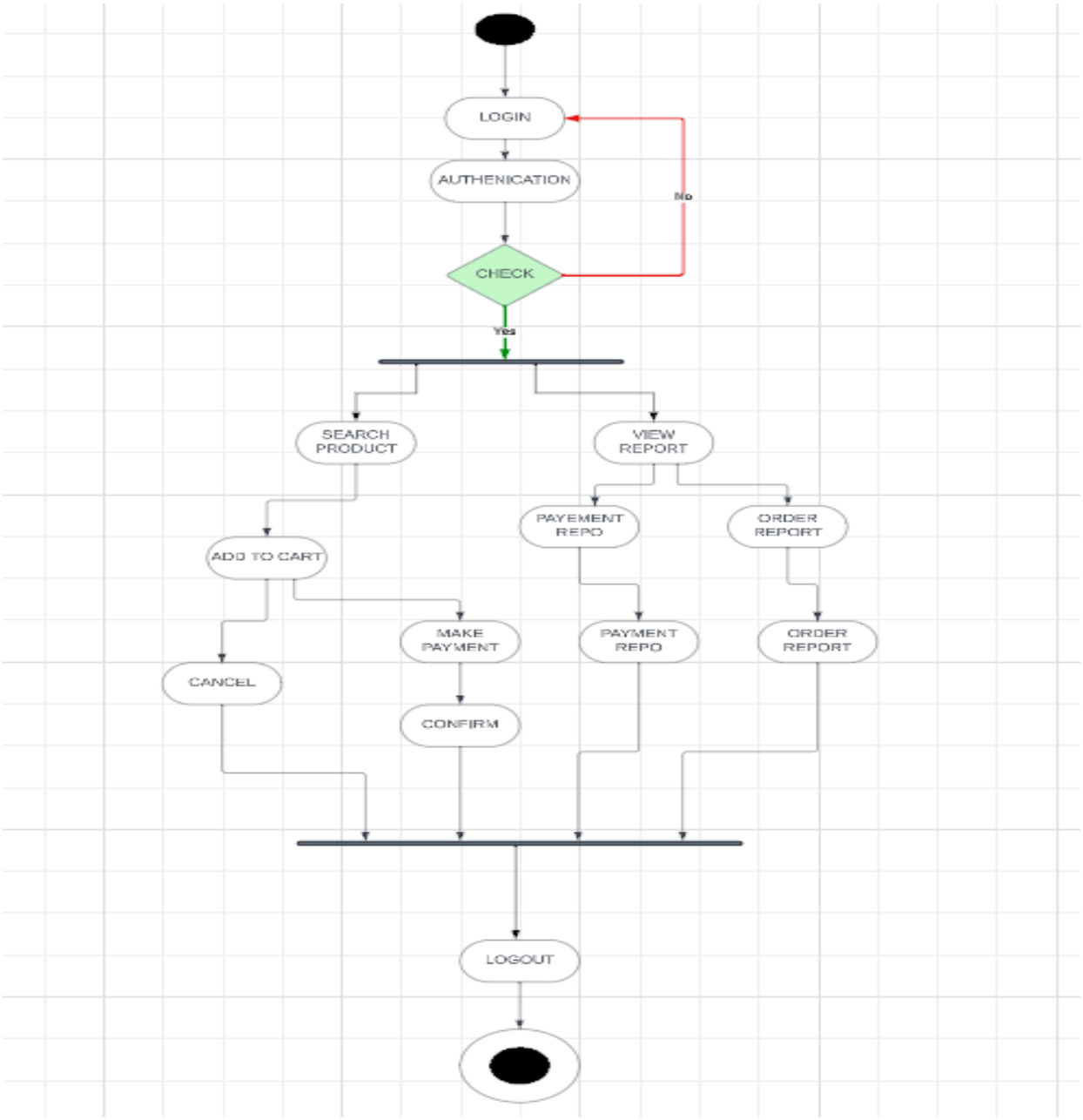
- Arrow or Line: Represents transitions or flow of control/data between activities and states, much like the arrow symbols in DFD.

- Two Horizontal Lines: Depicts a store where data can be temporarily or permanently stored, analogous to a DFD data store.

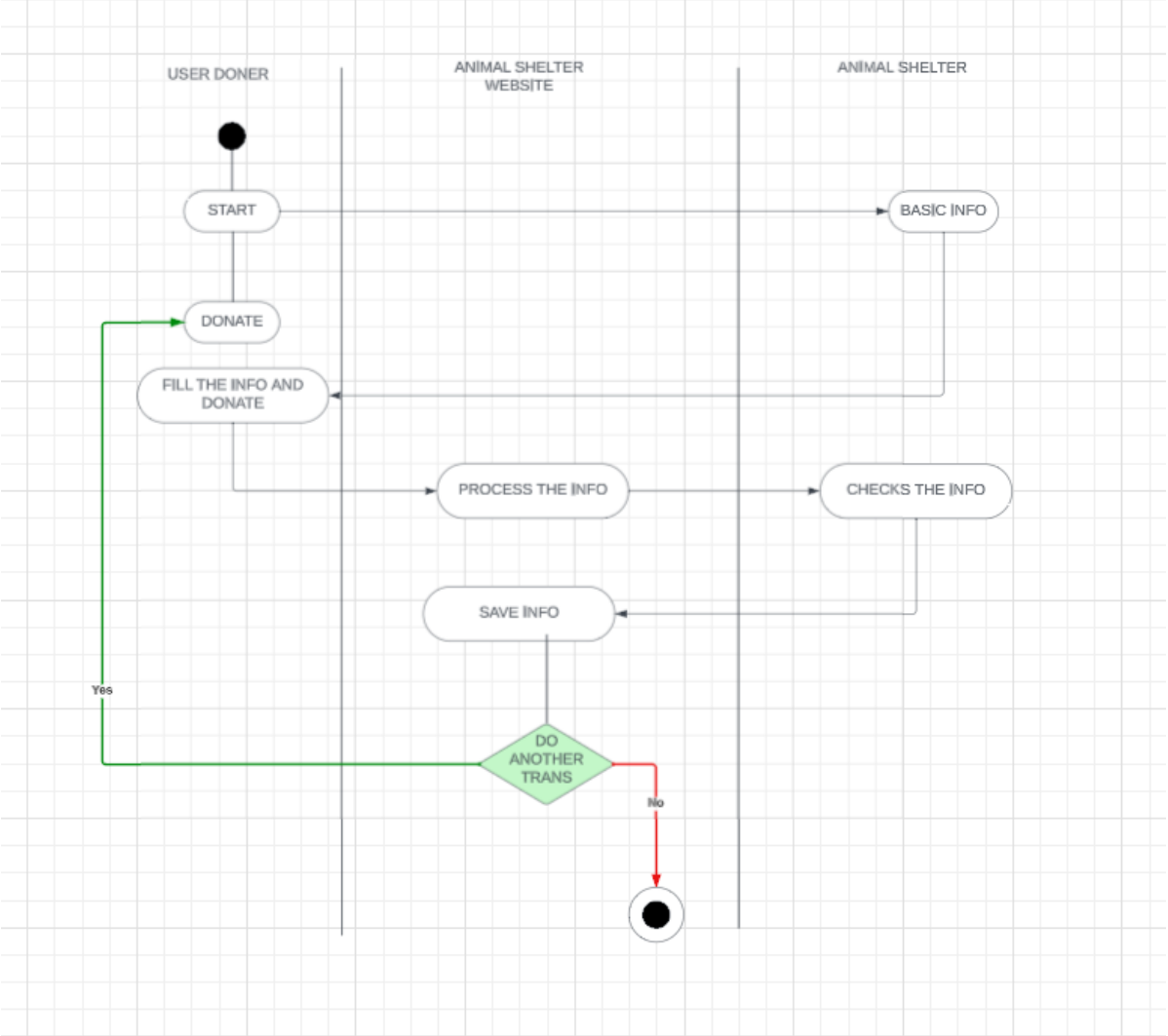
In summary, a State-Activity Diagram is a valuable tool in software engineering for modeling and visualizing the behavior and interactions of a software system. It helps stakeholders understand how data and control flow through the system and how it transitions between different states and activities.

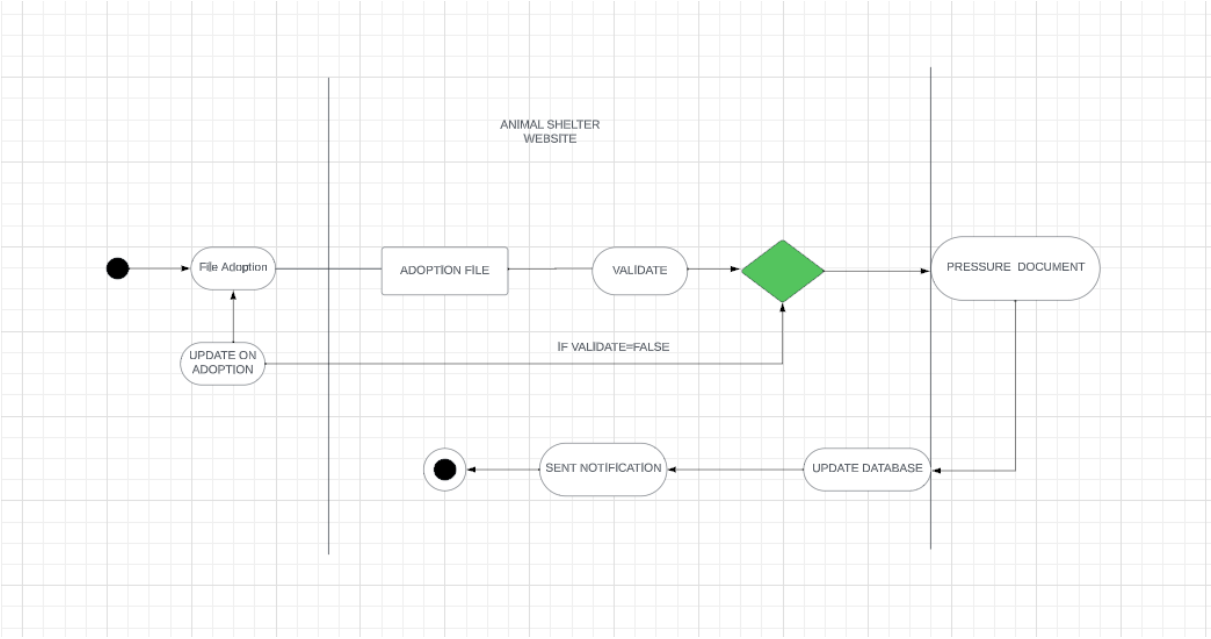


REPORT AND Rescue Activity Diagram

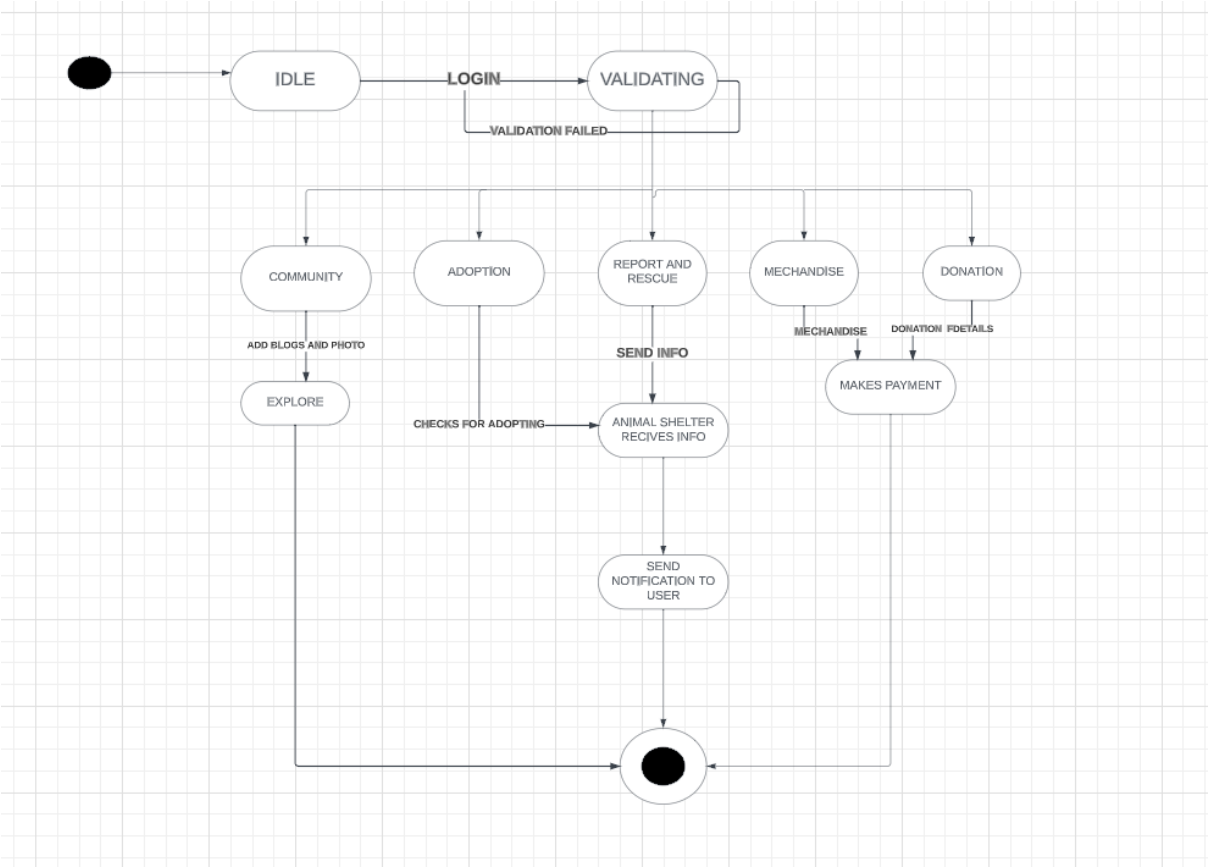


Merchandise ACTIVITY DIAGRAM





Donation and Adoption Activity Diagram



State Diagram

EXPERIMENT NO 6

AIM: Identify scenarios & develop use case diagram for the project

THEORY:

A Use Case Diagram (UCD) is a graphical representation used in software engineering to illustrate the interactions between various actors (users or external systems) and the functionalities or use cases offered by a system or software application. UCDs provide a high-level view of the system's behavior from a user's perspective, highlighting the system's functionality and the roles played by different actors. They are instrumental in communicating system requirements and functionalities to developers, designers, and stakeholders involved in the software development process.

Similar to Data Flow Diagrams (DFD) and State-Activity Diagrams (SAD), Use Case Diagrams have their own set of components and rules:

Components of a Use Case Diagram:

1. Actor: Actors represent the different roles or entities that interact with the system. These could be users, external systems, or even other software applications. Actors are depicted as stick figures or named entities, and they are placed outside the system boundary. Actors are essential for defining who interacts with the system and what their roles entail.

2. Use Case: Use cases represent the specific functionalities or actions that the system provides to its actors. These functionalities describe what the system does and how it responds to different user interactions. Use cases are depicted as ovals or ellipses within the system boundary. Each use case is labeled with a descriptive name.

3. Relationships: Arrows or lines connecting actors and use cases represent the relationships or interactions between actors and the use cases they participate in. These relationships indicate which actors are involved in or initiate specific use cases. Relationships can be categorized into associations, generalizations (inheritance), or dependencies.

4. System Boundary: The system boundary is a rectangle that encloses all the actors and use cases. It defines the scope of the system being modeled in the diagram.

Rules for Creating a Use Case Diagram:

1. Clear Actor Roles: Ensure that the roles of actors are clearly defined and named so that their interactions with the system are easily understood.

2. Descriptive Use Case Names: Name each use case descriptively to convey its purpose or functionality. Use clear and concise language to make it easy for stakeholders to comprehend the system's capabilities.

3. Proper Relationships: Use arrows or lines to connect actors with the use cases they interact with. Choose the appropriate relationship type (association, generalization, or dependency) to accurately represent the nature of the interaction.

4. Keep it Simple: While a Use Case Diagram can represent a complex system, it's essential to maintain simplicity for clarity. Avoid overloading the diagram with too many actors or use cases.

5. Consistency: Ensure consistency in the representation of actors, use cases, and relationships throughout the diagram. This consistency aids in maintaining the diagram's readability.

Symbols Used in a Use Case Diagram:

- Stick Figure or Named Entity: Represents an actor, portraying the roles that interact with the system.

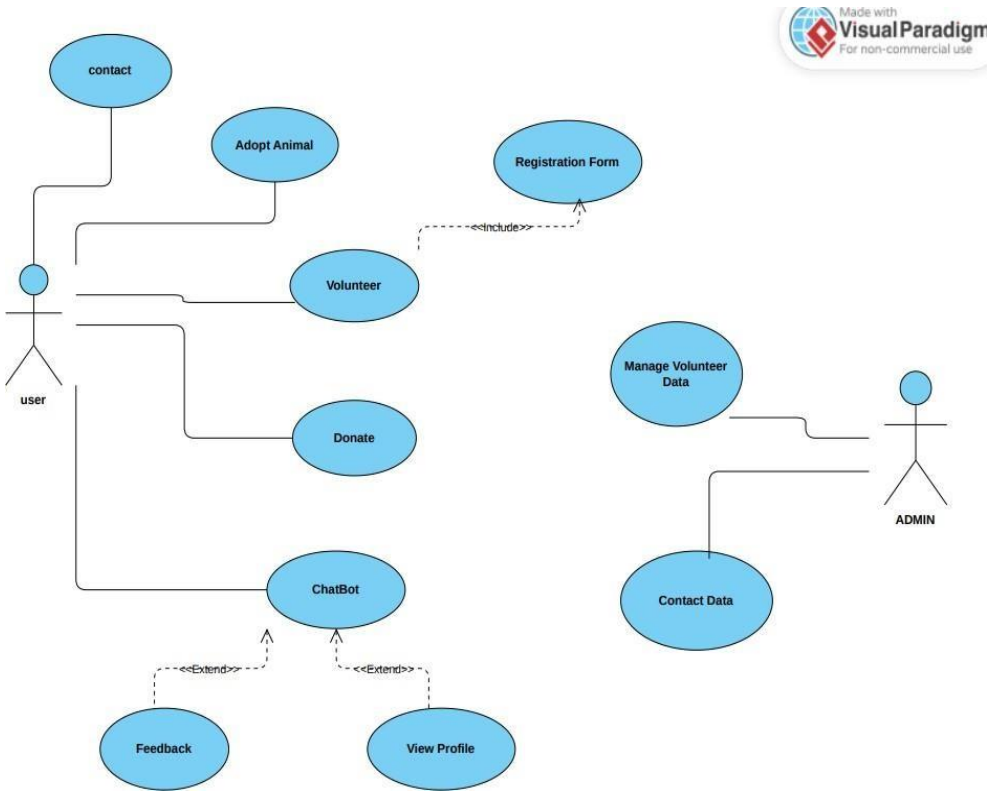
- Oval or Ellipse: Represents a use case, depicting a specific system functionality or action.

- Arrows or Lines: Depict relationships between actors and use cases, indicating the interactions and roles involved.

- Rectangle: Encloses all actors and use cases, defining the system boundary.

In summary, a Use Case Diagram is a valuable tool in software engineering for visualizing the interactions between actors and the functionalities provided by a software system. It helps stakeholders understand how different users or external entities interact with the system and what functionalities the system offers to meet their needs.

OUTPUT:


Made with
Visual Paradigm
For non-commercial use
contact
Adopt Animal
Registration Form
<<Include>>
Volunteer
user
Donate
Manage Volunteer Data
ADMIN
Contact Data
ChatBot
<<Extend>>
<<Extend>>
Feedback
View Profile

EXPERIMENT NO 7

AIM: Create a project schedule using gantt chart

THEORY:

Introduction to Gantt Charts:

Gantt charts are a widely used tool in project management, known for their visual representation of project schedules. Developed by Henry L. Gantt in the early 20th century, these charts provide a comprehensive overview of project tasks, timelines, and dependencies. The primary purpose of Gantt charts is to help project managers and teams plan, track, and manage projects efficiently. This theoretical exploration delves into the key aspects of Gantt charts, their components, and their significance in project management.

Components of a Gantt Chart:

A Gantt chart consists of several key components that collectively provide a detailed depiction of a project schedule:

1. Tasks and Activities: At the core of a Gantt chart are the individual tasks or activities required to complete a project. These tasks are specific, measurable, and time-bound, serving as the building blocks of the chart.

2. Timeline: The horizontal axis of the Gantt chart represents time, typically divided into days, weeks, or months, depending on the project's duration. The timeline serves as the foundation for scheduling tasks.

3. Task Bars: Task bars are horizontal bars that visually represent the start and end dates of each task. The length of these bars corresponds to the duration of the task. They are positioned along the timeline to indicate when each task is scheduled.

Creating a Gantt Chart:

Creating a Gantt chart involves several key steps:

4. Task Identification: The first step is to identify all the tasks and activities required for the project. Tasks should be well-defined, specific, and achievable.

5. Sequencing Tasks: After identifying tasks, project managers need to determine the order in which tasks should be executed. Understanding task dependencies is critical, as some tasks may rely on the completion of others.

6. Estimating Task Durations: Accurate estimation of task durations is essential for creating a realistic schedule. This step often involves input from subject matter experts and historical data.

7. Resource Allocation: Resource allocation involves assigning the necessary resources, including personnel, equipment, and materials, to each task. It's crucial to ensure that resources are available when needed.

8. Constructing the Gantt Chart: Using the timeline as a reference, project managers create the Gantt chart by plotting task bars in the correct sequence, respecting task dependencies, and considering resource availability. This visual representation allows for a clear overview of the project's timeline and tasks.

Monitoring and Managing with Gantt Charts

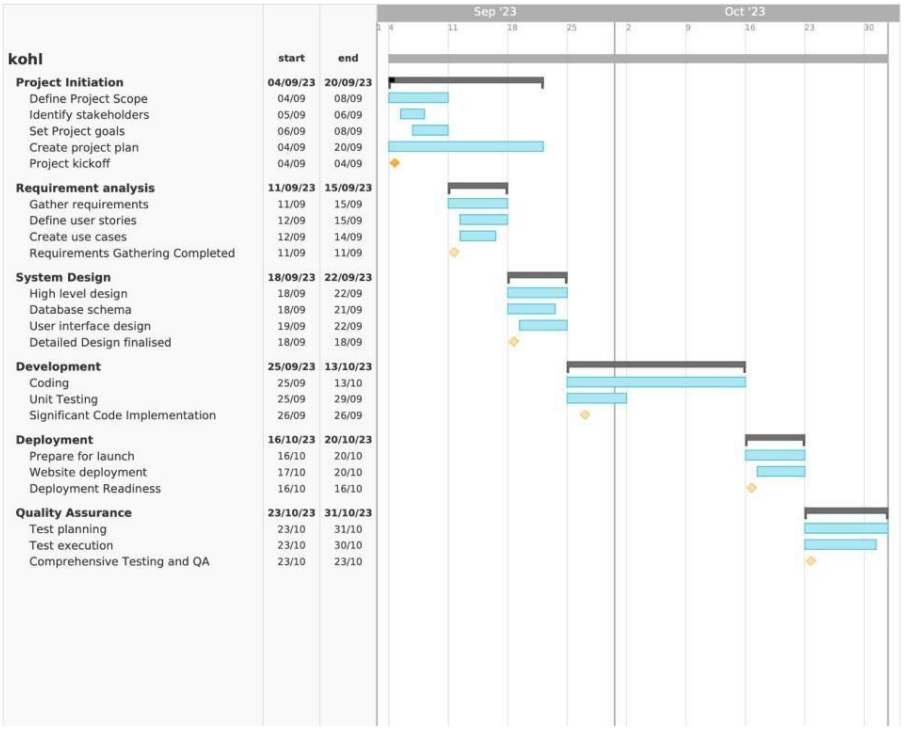
9. Updating the Chart: Gantt charts are dynamic tools that require regular updates as the project progresses. Tasks can be marked as completed, delayed, or rescheduled to reflect real-time project status.

10. Critical Path Analysis: The concept of the critical path, representing the longest sequence of dependent tasks, is instrumental in identifying the project's minimum duration. Monitoring and managing the critical path are essential for keeping the project on track.

11. Resource Management: Effective resource management ensures that resources are utilised efficiently and that potential bottlenecks are addressed promptly.

In conclusion, Gantt charts are indispensable tools for project managers seeking to plan, execute, and control projects successfully. By understanding their components and following a structured approach to create and manage Gantt charts, project teams can improve coordination, mitigate risks, and ultimately ensure that projects are delivered on time and within scope. The ability to create and effectively use Gantt charts is a hallmark of proficient project management, enhancing communication and decision-making in complex projects.

OUTPUT:



kohl	start	end
Project Initiation	**04/09/23**	**20/09/23**
Define Project Scope	04/09	08/09
Identify stakeholders	05/09	06/09
Set Project goals	06/09	08/09
Create project plan	04/09	20/09
Project kickoff	04/09	04/09
Requirement analysis	**11/09/23**	**15/09/23**
Gather requirements	11/09	15/09
Define user stories	12/09	15/09
Create use cases	12/09	14/09
Requirements Gathering Completed	11/09	11/09
System Design	**18/09/23**	**22/09/23**
High level design	18/09	22/09
Database schema	18/09	21/09
User interface design	19/09	22/09
Detailed Design finalised	18/09	18/09
Development	**25/09/23**	**13/10/23**
Coding	25/09	13/10
Unit Testing	25/09	29/09
Significant Code Implementation	26/09	26/09
Deployment	**16/10/23**	**20/10/23**
Prepare for launch	16/10	20/10
Website deployment	17/10	20/10
Deployment Readiness	16/10	16/10
Quality Assurance	**23/10/23**	**31/10/23**
Test planning	23/10	31/10
Test execution	23/10	30/10
Comprehensive Testing and QA	23/10	23/10

Experiment: 8

Aim:

Conduct Function Point Analysis (FPA) for the project.

Theory:

No. of User Inputs (UI):

User registration or login: 1 UI

Reporting distressed animals: 1 UI

Chatbot interactions: 1 UI

Posting messages and engaging in discussions: 1 UI

Selecting pets for adoption: 1 UI

Volunteering for animal shelters: 1 UI

Making merchandise purchases: 1 UI

Making monetary donations: 1 UI

Total UIs: Approximately **8** UIs

No. of User Outputs (UO):

Displaying the results of animal rescue requests: 1 UO

Chatbot responses: 1 UO

Showing community posts and content: 1 UO

Displaying pet adoption details: 1 UO

Providing information on volunteering opportunities: 1 UO

Showing merchandise for purchase: 1 UO

Confirmations of successful donations: 1 UO

Total UOs: Approximately **7** UOs

No. of User Inquiries (UI):

Inquiries to the Chatbot: 1 UI

Total UIs: Approximately **1** UI

No. of Files:

The number of files can vary based on the data storage and organization within your system. However, you might have separate files for user profiles, animal rescue data, community posts, pet adoption details, merchandise information, and potentially others. Let's estimate around **6** files.

No. of External Interface Files (EIFs):

Integration with Google Maps for location marking: 1 EIF

Integration with payment gateways for merchandise purchases and donations:1 EIF

Total EIFs: Approximately **2** EIFs

Counting Function Point (FP):

Step 1:

Backup & Recovery:

Value: 4

Explanation: While there is important data related to animal rescue requests and shelter coordination, the fiscal loss due to data loss may not be as critical as in a financial system.

Data Communications:

Value: 3

Explanation: The project involves communication between users (admin and store owners) and real-time data updates, but it's not highly complex or extensive.

Distributed Processing:

Value: 1

Explanation: The project operates within a single environment, and there's no need for distributed processing.

Performance Critical:

Value: 4

Explanation: Accurate tracking of inventory and cost calculations are crucial to prevent fiscal loss, but it may not be as critical as a financial system.

Existing Operating Environment:

Value: 2

Explanation: The project is not highly dependent on a specific operating environment, but it does require basic CRUD operations.

Online Data Entry:

Value: 3

Explanation: Real-time data updates are essential for efficient inventory management, but the complexity is moderate.

Input Transaction Over Multiple Screens:

Value: 4

Explanation: Simultaneous updates from multiple store owners on different screens require accuracy and real-time synchronization.

Master Files Updated Online:

Value: 4

Explanation: Real-time updates of product orders and billing information are crucial for effective inventory management.

Information Domain Values Complex:

Value: 2

Explanation: The data used in the project is relatively straightforward, with average data volume and relationships. Data transformation is moderate in complexity.

Internal Processing Complex:

Value: 2

Explanation: Internal complexity related to algorithms is minimal, but basic data manipulation and performance optimization are required.

Code Designed for Reuse:

Value: 3

Explanation: Some code can be designed for reuse, such as modularization and encapsulation, to enhance maintainability.

Conversion/Installation in Design:

Value: 4

Explanation: Incorporating installation and data migration considerations in the design is crucial for efficient deployment and cost savings.

Multiple Installations:

Value: 3

Explanation: The project should support installation on different devices and OS environments but may not support multiple views simultaneously for security reasons.

Application Designed for Change:

Value: 4

Explanation: Designing the application to be scalable and adaptable to future requirements is important for long-term viability.

Total Value Adjustment Factor: Σ Fi = 42

Step 3: Calculate Unadjusted Function Point (UFP)

Measurement Parameter	Weighting Factor

	Simple		**Average**		**Complex**	
Number of user input	8x 3	24	8 x 4	32	8 x 6	48
Number of user output	7x 4	28	7 x 5	35	7x 7	49
Number of user inquiries	1 x 3	3	1x 4	4	1x 6	6
Number of files	6x 7	42	6x 10	60	6x 15	90
Number of external interfaces	2 x 5	10	2x 7	14	2x 10	20
	Simple Total	**107**	**Average Total**	**145**	**Complex Total**	**213**

Step 4: Calculate Function Point.

1. **Simple**

   Unadjusted Function Point = 107

   Function point = (Unadjusted Function Point) x (Complexity Adjustment Factor)
   = 107 * [0.65 + 0.01 * 42]
   = 107 * [0.65 + 0.42]
   = 107 * 1.07
   = 114.49
   =114

2. **Average**

   Unadjusted Function Point = 145

   Function point = (Unadjusted Function Point) x (Complexity Adjustment Factor)

   = 145 * [0.65 + 0.01 * 42]
   = 145 * [0.65 + 0.42]
   = 145 * 1.07
   = 155.15
   = 155

3. **Complex**

Unadjusted Function Point = 213

Function point = (Unadjusted Function Point) x (Complexity Adjustment Factor)
= 213 * [0.65 + 0.01 * 42]
= 213 * [0.65 + 0.42]
= 213 * 1.07
= 227.91
=227

1. Productivity Factor Assumption: Let's assume a productivity factor of 4 hours per function point.
2. Hourly Rate Assumption: Let's assume an hourly rate of Rs.50 per hour.

Now, we can calculate the effort, productivity cost per function point, and total cost for each complexity level:

For Simple (107 Function Points):

- Effort = 107 Function Points × 4 hours/FP = 428 hours
- Productivity Cost per Function Point = Total Effort / Total Function Points = 428 hours / 107 FP = 4 hours/FP
- Total Cost = Effort × Hourly Rate = 428 hours × Rs.50/hour = Rs.21,400

For Average (145 Function Points):

- Effort = 145 Function Points × 4 hours/FP = 580 hours
- Productivity Cost per Function Point = Total Effort / Total Function Points = 580 hours / 145 FP = 4 hours/FP
- Total Cost = Effort × Hourly Rate = 580 hours × Rs.50/hour = Rs.29,000

For Complex (213 Function Points):

- Effort = 213 Function Points × 4 hours/FP = 852 hours
- Productivity Cost per Function Point = Total Effort / Total Function Points = 852 hours / 213 FP = 4 hours/FP
- Total Cost = Effort × Hourly Rate = 852 hours × Rs.50/hour = Rs.42,600

EXPERIMENT 9 :

Aim- APPLICATION OF COCOMO MODEL FOR COST ESTIMATION.

Theory-

Types of Software Projects
>Organic
>Semidetached
>Embedded

Organic Mode-Developed in a familiar,Stable environment ,similar to the previously developed projects relatively small and requires little innovation

Semidetached Mode-Intermediate between Organic and embedded

Embedded Mode-Tight, inflexible constraints and interface requirements . The product requires great innovation.

3 stages of Cocomo
-Basic Cocomo
-Intermediate Cocomo
-Complete Cocomo

BASIC COCOMO

$$\text{Effort} = a_1 \times (\text{KLOC})^{a_2}\ \text{PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})^{b_2}\ \text{Months}$$

Where

- KLOC is the estimated size of the software product expressed in Kilo Lines of Code,
- a_1, a_2, b_1, b_2 are constants for each category of software products,
- Tdev is the estimated time to develop the software, expressed in months,
- Effort is the total effort required to develop the software product, expressed in person months (PMs).

Software Project	a_1	a_2	b_1	b_2
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

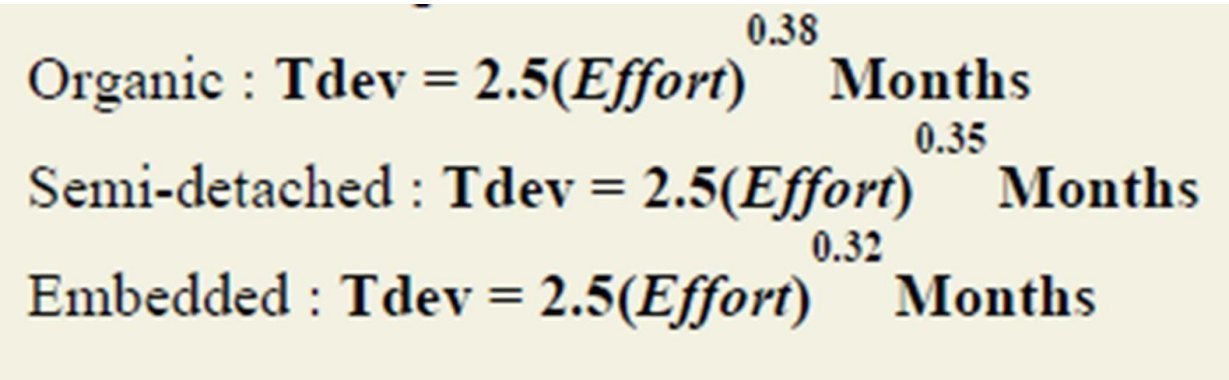
ESTIMATION OF EFFORT

Organic : $\mathbf{Effort = 2.4(KLOC)^{1.05}\ PM}$

Semi-detached : $\mathbf{Effort = 3.0(KLOC)^{1.12}\ PM}$

Embedded : $\mathbf{Effort = 3.6(KLOC)^{1.20}\ PM}$

ESTIMATION OF DEVELOPMENT TIME



QUESTION:

We have determined our project fits the characteristics of Basic, Semi-Detached mode. We estimate our project will have 1,50,000 Delivered Source Instructions. Find the Effort, Schedule, Productivity, and average staffing required for the project.

1. **Organic** :

- Effort = 2.4 * ((KLOC) ^ 1.05) PMs

  = 2.4 * (1,50,000 ^ 1.05) PMs

  = 2.4 * (272204.8268) PMs

  = 653291.5844PMs

  = 653291.5844PMs

- Schedule (Tdev) = 2.5 * (Effort ^ 0.38) Months

  = 2,5 * (653292^ 0.38) Months

  = 2.5 * (162.0842401) Months

  = 405.2106002 Months

  = **405** Months

- Productivity = KLOC / Effort

  = 1,50,000 / 653292

  = **0.23**

- Average Staffing = Effort / Schedule FSP

  = 653292/ 405 FSP

  = 1613.0667 FSP

  = **1613** FSP

2. <u>Semi Detached :</u>

- Effort = 3.0 * ((KLOC) ^ 1.12) PMs

  = 3.0 * (1,50,000 ^ 1.12) PMs

  = 3.0 * (626934.5586) PMs

  = 1880803.676 PMs

  = 1880803 PMs

- Schedule (Tdev) = 2.5 * (Effort ^ 0.35) Months

  = 2,5 * (1880803^ 0.35) Months

  = 2.5 * (157.04358) Months

  = 392.6089Months

  = **393** Months

- Productivity = KLOC / Effort

  = 1,50,000 / 1880803 = **0.079753**

- Average Staffing = Effort / Schedule FSP

= 1880803/ 405 FSP

= 4643.9580 FSP

= **4644** FSP

3. <u>Embedded :</u>

- Effort = 3.6 * ((KLOC) ^ 1.20) PMs

  = 3.6 * (1,50,000 ^ 1.20) PMs

  = 3.6 * (1626707) PMs

  = 5856147.564 PMs

  = 5856147PMs

- Schedule (Tdev) = 2.5 * (Effort ^ 0.32) Months

  = 2,5 * (5856147^ 0.32) Months

  = 2.5 * (146.4319) Months

  = 366.07987Months

  = **366** Months

- Productivity = KLOC / Effort

  = 150,000 / 5856147

  = **0.02561**

- Average Staffing = Effort / Schedule FSP

= 5856147/ 405 FSP

= 14459.622 FSP

= **14459** FSP

INTERMEDIATE COCOMO

Four areas of drivers

-Product itself:

•inherent complexity of the product, reliability requirements etc.

-Computer:

•execution speed required, storage space required, etc.

-Personnel:

•experience level, programming capability, analysis capability, etc.

-Project itself or Development environment:

•development facilities available to developers

Software Project	a1	a2	b1	b2
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

•Product Attributes

-RELY ---Required Software Reliability

The extent to which the software product must perform its intended functions satisfactorily over a period of time.

-DATA --- Data Base Size

The degree of the total amount of data to be assembled for the data base.

-CPLX ---Software Product Complexity

· The level of complexity of the product to be developed.

•Computer Attributes

-TIME --- Execution Time Constraint

. The degree of the execution constraint imposed upon a software product.

-STOR --- Main Storage Constraint

. The degree of main storage constraint imposed upon a software product.

-VIRT -- Virtual Machine Volatility

. The level of the virtual machine underlying the product to be developed.

-TURN --- Computer Turnaround Time

. The level of computer response time experienced by the project team developing the product.

•Personnel Attributes

-ACAP --- Analyst Capability

. The level of capability of the analysts working on a software product.

-AEXP --- Applications Experience

. The level of applications experience of the project team developing the software product.

-PCAP --- Programmer Capability

. The level of capability of the programmers working on the software product.

-VEXP --- Virtual Machine Experience

-LEXP ---Programming Language Experience

•Project Attributes

-MODP --- Modern Programming Practices

. The degree to which modern programming practices (MPPs) are used in developing software product.

-TOOL --- Use of Software Tools

. The degree to which software tools are used in developing the software product.

-SCED--- Required Development Schedule

. The level of schedule constraint imposed upon the project team developing the software product.

COCOMO - COST DRIVERS

	COST DRIVER	RATING V.LOW	LOW	NOMINAL	HIGH	V.HIGH	EX. HIGH
(PRODUCT)	RELY	0.75	0.88	1.00	1.15	1.40	.
..	DATA	.	0.94	1.00	1.08	1.16	.
..	CPLX	0.70	0.85	1.00	1.15	1.30	1.65
(COMPUTER)	TIME	.	.	1.00	1.11	1.30	1.66
..	STOR	.	.	1.00	1.06	1.21	1.56
..	VIRT	.	0.87	1.00	1.15	1.30	.
..	TURN	.	0.87	1.00	1.07	1.15	.
(PERSONNEL)	ACAP	1.46	1.19	1.00	0.86	0.71	.
..	AEXP	1.29	1.13	1.00	0.91	0.82	.
..	PCAP	1.42	1.17	1.00	0.86	0.70	.
..	VEXP	1.21	1.10	1.00	0.90	.	.
..	LEXP	1.14	1.07	1.00	0.95	.	.
(PROJECT)	MODP	1.24	1.10	1.00	0.91	0.82	.
..	TOOL	1.24	1.10	1.00	0.91	0.83	.
..	SCED	1.23	1.08	1.00	1.04	1.10	.

.For our project now we'll calculate the effort ,schedule,productivity and Average staffing for a semidetached project of 1,50,000 KLOC of ours.
The cost drivers are set as follows-
.Product cost drivers-high
.Computer cost drivers-nominal
.Personnel cost drivers-low
.Project cost drivers-high

Solution:

•Product cost drivers (from the table) set **high** = 1.15 x 1.08 x 1.15 = 1.43

•Computer cost drivers (from the table) set **nominal** = 1.00

•Personnel cost drivers (from the table) set **low** = 1.19 x 1.13 x 1.17 x 1.10 x 1.07 = 1.85

•Project cost drivers (from the table) set **high** = 0.91 x 0.91 x 1.04 = 0.86

• **product(cost drivers)** = 1.43 x 1.00 x 1.85 x 0.86 = 2.28

.For a semidetached project of 1,50,000 KLOC : a1=3,a2=1.12,b1=2.5,b2=0.35

EFFORT=(3*(1,50,000)^(1.12))*2.28
=4288232 PM

SCHEDULE=2.5*(4288232)^0.35
=524 months

PRODUCTIVITY=150000/4288232
=0.03497 KLOC/PM

AVERAGE STAFFING=4288232/524
=8183 PM/months

COCOMO 2

QUESTION: let's adapt the question for an animal shelter website project. In this scenario, we'll consider the development of an animal shelter website with 4 main pages, each containing 4 different views, and 10 data tables for managing products, users, orders, and other related information. The website is designed to handle traffic from 1 server and 5 clients. Additionally, the project includes the generation of 3 different reports, each consisting of 7 sections, all of which are derived from the data stored in the 10 data tables. We'll also assume a 20% reuse of object points, and that the developer's experience and capability in a similar environment is Nominal.

Step 1: Determine the number of screens and reports.

- Number of screens = 4 features (Report and Rescue, Chatbot, Community, Adoption) * 1 main page each = 4 screens.
- Number of reports = 3 reports**.**

Step 2: Determine the factors for screens and reports.

For Screens:

- Number of views per screen = Varies for each feature (Not specified)
- Number of data tables per screen = Varies for each feature (Not specified)
- Number of servers = 1
- Number of clients = 5
- Complexity level for each screen = Nominal

For Reports:

- Number of sections per report = 7
- Number of data tables per report = 10
- Number of servers = 1
- Number of clients = 5
- Complexity level for each report = Nominal

Step 3: Assign complexity weights.

Since you've mentioned that the developer's experience is Nominal, we can assign weights as follows:

- Complexity weight for each screen = 5 (Nominal)
- Complexity weight for each report = 5 (Nominal)

Step 4: Calculate Object Point Count

For Screens:
Object Point Count for Screens = (Number of screens * Complexity weight for each screen)
Object Point Count for Screens = (4 screens * 5) = 20 Object Points

For Reports:
Object Point Count for Reports = (Number of reports * Complexity weight for each report)
Object Point Count for Reports = (3 reports * 5) = 15 Object Points

Now, we can calculate the total Object Point Count:
Total Object Point Count = Object Point Count for Screens + Object Point Count for Reports
Total Object Point Count = 20 + 15 = 35 Object Points

Given that there is a 20% reuse of object points, we can calculate the Non-Reused Object Points (NOP):
Non-Reused Object Points (NOP) = [Total Object Points * (100 - % Reuse)] / 100
Non-Reused Object Points (NOP) = [35 * (100 - 20)] / 100
Non-Reused Object Points (NOP) = [35 * 80] / 100
Non-Reused Object Points (NOP) = 28

Step 6: Developer's Experience and Capability
The developer's experience and capability are Nominal.

Using the information given about the developer and productivity rate table:
Productivity rate (PROD) of the given project = 13

Step 7: Effort Calculation
Effort = NOP / PROD
Effort = 28 / 13
Effort ≈ 2.15 person-days (rounded to two decimal places)

Aim: Develop a Risk Migration,monitoring and management plan (RMMM) for a project

1. User Data Security Breach

Risk id : IMS-001	Date: 18/09/23	Probability : 15%	Impact : High
Description : Unauthorized access to sensitive user data can lead to privacy breaches and legal consequences.			
Mitigation : Implement strong encryption, access controls, and regular security audits			
Monitoring : Use intrusion detection systems and real-time log analysis.			
Management : Develop an incident response plan and provide cybersecurity training.			
Current Status : Ongoing security measures in place.			
Originator : Maithili,Shirish,Adarsh		Assigned : Maithili	

2.Server Downtime

Risk id : IMS-002	Date: 18/09/23	Probability : 10%	Impact : Moderate
Description : Server or hosting provider outages can disrupt website availability			
Mitigation : Choose a reliable hosting provider with redundancy.			
Monitoring : Implement server uptime monitoring tools.			
Management : Have a backup server or cloud-based failover solution.			
Current Status : Monitoring tools active.			
Originator : Maithili,Shirish,Adarsh		Assigned : Maithili	

3. Scope Creep

Risk id : IMS-003	Date: 18/09/23	Probability : 20%	Impact : Moderate
Description : Expanding project scope beyond the original plan can lead to delays and increased costs			
Mitigation : Clearly define project scope and use change control processes.			
Monitoring : Regularly review project scope against the plan.			
Management : Ensure proper documentation of scope changes.			
Current Status : Scope changes documented.			
Originator : Maithili,Shirish,Adarsh		Assigned : Maithili	

4. Technical Challenges

Risk id : IMS-004	Date: 18/09/23	Probability : 15%	Impact : High
Description : Technical issues or limitations can delay project progress.			
Mitigation : Conduct technical feasibility studies and engage experts			
Monitoring : Regularly review technical progress.			
Management : Have contingency plans for major technical issues.			
Current Status : No major technical issues.			
Originator : Maithili,Shirish,Adarsh		Assigned : Maithili	

5. User Adoption

Risk id : IMS-005	Date: 18/09/23	Probability : 10%	Impact : Moderate

Description : Low user adoption of the website can affect its success.	
Mitigation : Conduct user testing and gather feedback during development.	
Monitoring : Track user engagement and conduct surveys post-launch.	
Management : Be prepared to make user-centric improvements.	
Current Status : User engagement is moderate.	
Originator : Maithili,Shirish,Adarsh	Assigned : Maithili

6. Budget Overruns

Risk id : IMS-006	Date: 18/09/23	Probability : 15%	Impact : Moderate
Description : Exceeding the allocated project budget can strain resources.			
Mitigation : Carefully track expenses and adjust the project plan if needed.			
Monitoring : Regularly review the project budget			
Management : Seek approval for budget adjustments if necessary..			
Current Status : Budget is within acceptable limits.			
Originator : Maithili,Shirish,Adarsh		Assigned : Shirish	

7. Schedule Delays

Risk id : IMS-007	Date: 18/09/23	Probability : 20%	Impact :High
Description : Project taking longer than planned can lead to missed deadlines			

Mitigation : Develop a realistic project schedule and monitor progress closely.	
Monitoring : Regularly review the project schedule.	
Management : Have contingency plans for schedule delays.	
Current Status : Slight schedule delays.	
Originator : Maithili,Shirish,Adarsh	Assigned :Shirish

8. Vendor Reliability

Risk id : IMS-008	Date: 18/09/23	Probability : 10%	Impact :Moderate
Description : Reliability issues with third-party vendors can disrupt project progress.			
Mitigation : Choose reputable vendors and have alternative vendors in mind.			
Monitoring : Monitor vendor performance.			
Management : Have contingency plans for vendor failures.			
Current Status : Vendors are reliable.			
Originator : Maithili,Shirish,Adarsh		Assigned : Shirish	

9. User Feedback Management

Risk id : IMS-009	Date: 18/09/23	Probability : 10%	Impact :Low
Description : Inefficient handling of user feedback can lead to missed improvements.			
Mitigation : Implement a structured feedback management system.			
Monitoring : Regularly review user feedback.			

Management : Assign responsibilities for reviewing and addressing feedback.	
Current Status : Feedback is being reviewed and addressed.	
Originator : Maithili,Shirish,Adarsh	Assigned :Shirish

10. Performance Optimization

Risk id : IMS-010	Date: 18/09/23	Probability : 15%	Impact : Moderate
Description : Poor website performance can lead to user dissatisfaction..			
Mitigation : Continuously optimize website performance during development.			
Monitoring : Monitor website speed and responsiveness.			
Management : Address performance issues promptly.			
Current Status : Performance optimization ongoing.			
Originator : Maithili,Shirish,Adarsh		Assigned :Shirish	

11) Change Resistance

Risk ID: IMS-011	Date:18/9/23	Probability:10%	Impact:low
Description: Resistance to change within the organization can hinder project adoption.			
Mitigation: Communicate the benefits of the project and involve stakeholders early.			
Monitoring: Gather feedback and address concerns..			
Management: Provide training and support for change.			
Current status: Change resistance is low.			

Originator: Maithili,Shirish,Adarsh	**Assigned:** Adarsh

12) User Training

Risk ID: IMS-012	Date:18/9/23	Probability:15%	Impact:moderate
Description: Inadequate user training can lead to difficulties in using the system.			
Mitigation: Develop comprehensive user training materials and sessions.			
Monitoring: Assess user training needs and provide ongoing support.			
Management: Ensure users have access to training resources.			
Current status: User training is ongoing.			
Originator: Maithili,Shirish,Adarsh		**Assigned:** Adarsh	

13) Deployment Challenges

Risk ID: IMS-013	Date:18/9/23	Probability:10%	Impact:Moderate
Description: Difficulties in deploying the website to production can cause delays.			
Mitigation: Plan for deployment carefully and conduct testing.			
Monitoring: Monitor the deployment process closely.			
Management: Have a rollback plan in case of deployment issues.			
Current status: Deployment plan in place			
Originator: Maithili,Shirish,Adarsh		**Assigned:** Adarsh	

14) Documentation Management

Risk ID: IMS-014	Date:18/9/23	Probability:5%	Impact:low
Description: Inadequate project documentation can lead to confusion.			
Mitigation: Maintain comprehensive project documentation.			
Monitoring: Regularly review and update project documentation.			
Management: Assign responsibility for document management.			
Current status: Documentation is up to date.			
Originator: Maithili,Shirish,Adarsh		**Assigned:** Adarsh	

15) Stakeholder Misalignment

Risk ID: IMS-015	Date:18/9/23	Probability:10%	Impact:moderate
Description: Conflicting expectations among project stakeholders can lead to delays.			
Mitigation: Hold regular stakeholder meetings to align expectations.			
Monitoring: Gather feedback from stakeholders throughout the project.			
Management: Assign a project manager responsible for stakeholder communication.			
Current status: Stakeholders are aligned.			
Originator: Maithili,Shirish,Adarsh		**Assigned:** Adarsh	

EXPERIMENT NO. 11

AIM: Case Study: GitHub for version Control.

THEORY:

Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

Why do we need Configuration management?

The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating.
- It may be a case where multiple version, branches, authors are involved in a software config. project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders

SCM process is also beneficial to control the costs involved in making changes to a system Version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information. Version control is a component of software configuration management.

Benefits of the Version Control System

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

Some key benefits of having a version control system are as follows.

- Complete change history of the file
- Simultaneously working
- Branching and merging
- Traceability

How does version control work?
What is a repository?
You can think of a repository (aka a repo) as a “main folder”, everything associated with a specific project should be kept in a repo for that project. Repos can have folders within them, or just be separate files.
You will have a local copy (on your computer) and an online copy (on GitHub) of all the files in the repository.

1. First of all configure the git username and git email id so that you can perform git commit operations

```
PS C:\Users\complab304pc30\Desktop\gitcase> git config user.name "SDK"
PS C:\Users\complab304pc30\Desktop\gitcase> git config user.email "sdk.1234@gmail.com"
```

2. You can view git commands by typing out below command

```
PS C:\Users\complab304pc30\Desktop\gitcase> git --help
usage: git [-v | --version] [-h | --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.
PS C:\Users\complab304pc30\Desktop\gitcase>
```

3. To initialise git repository use git init command

```
PS C:\Users\complab304pc30\Desktop\gitcase> git init
Initialized empty Git repository in C:/Users/complab304pc30/Desktop/gitcase/.git/
```

4. To check status of your local repository use following command

```
PS C:\Users\complab304pc30\Desktop\gitcase> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

5. Add file to staging area and again check the status

```
PS C:\Users\complab304pc30\Desktop\gitcase> git add .
PS C:\Users\complab304pc30\Desktop\gitcase> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```

6. Now the file is in the staging area it is ready to be committed to save the history and you can use git log to see the history

```
PS C:\Users\complab304pc30\Desktop\gitcase> git commit -m "First Commit"
[master (root-commit) 5997796] First Commit
 1 file changed, 11 insertions(+)
 create mode 100644 index.html
```

EXPERIMENT 12

Aim: To develop test cases for The Pet Finder Website using White Box testing

Theory:

Any engineered product (and most other things) can be tested in one of two ways:

- Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function.
- Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh," that is, internal operations are performed according to specifications and all internal components have been adequately exercised. The first test approach takes an external view and is called black-box testing. The second requires: an internal view and is termed white-box testing. Black-box testing alludes to tests that are conducted at the software interface.

Black-box testing delves into the fundamental aspects of a system, without much concern for how the software is internally structured. In contrast, white-box testing hinges on a meticulous examination of the procedural intricacies. It involves dissecting the logical paths within the software and scrutinizing how various components collaborate. On the surface, it might appear that rigorous white-box testing could yield perfectly error-free programs. The notion is that by defining all possible logical paths, devising comprehensive test cases, and meticulously assessing the results, we could ensure a flawless outcome. However, there's a practical challenge: even for relatively small programs, the number of potential logical paths can become overwhelmingly large. Still, it's essential to note that white-box testing remains a valuable practice. By focusing on a select number of critical logical paths and scrutinizing the integrity of vital data structures, it provides a pragmatic approach to enhance software reliability.

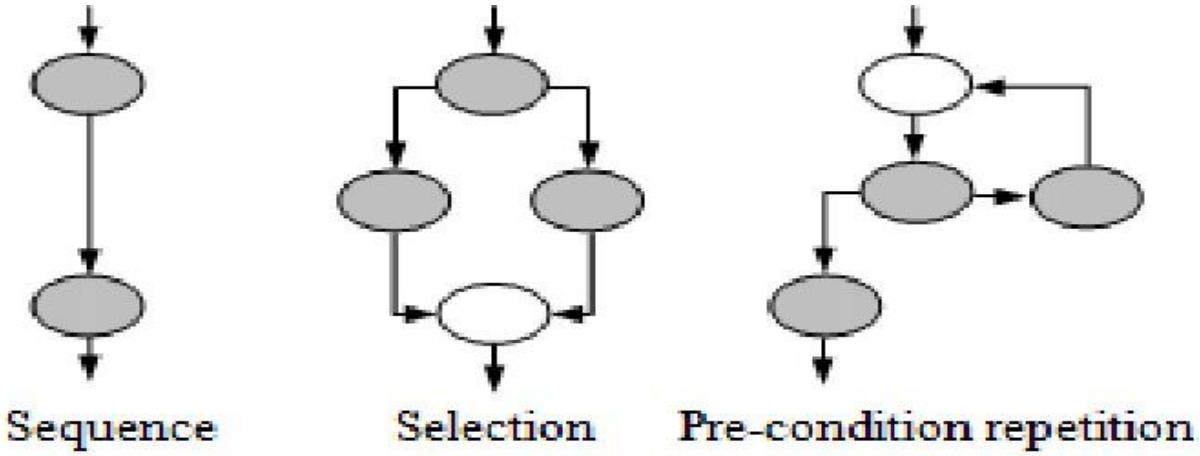
White-box testing, also known as glass-box testing, adopts a test-case design philosophy that leverages the control structure detailed in component-level

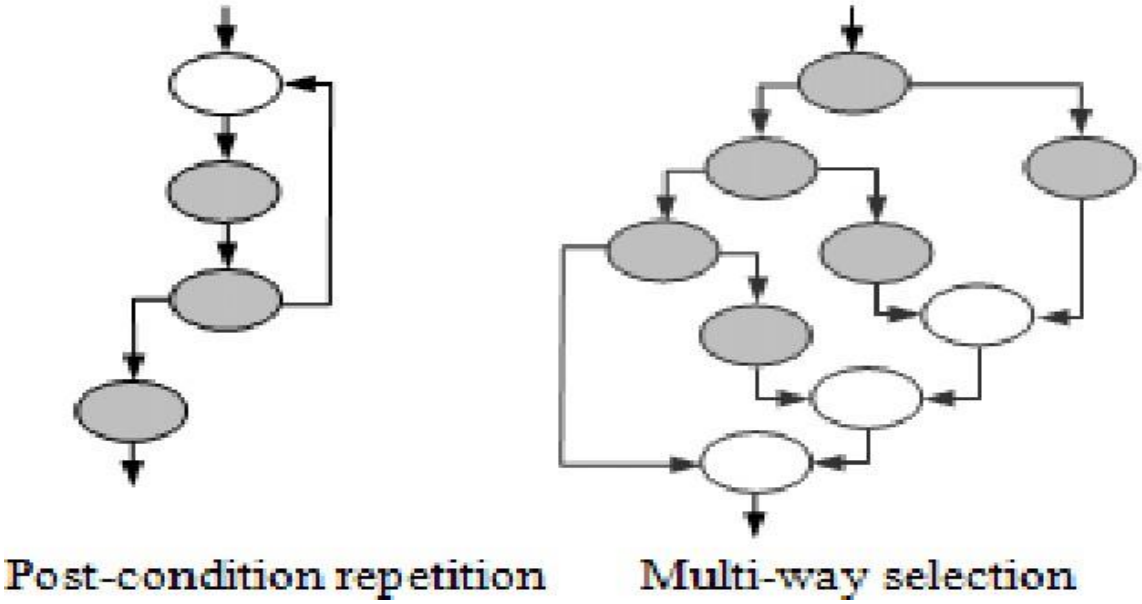
design. This approach facilitates the creation of test cases that serve several critical purposes:

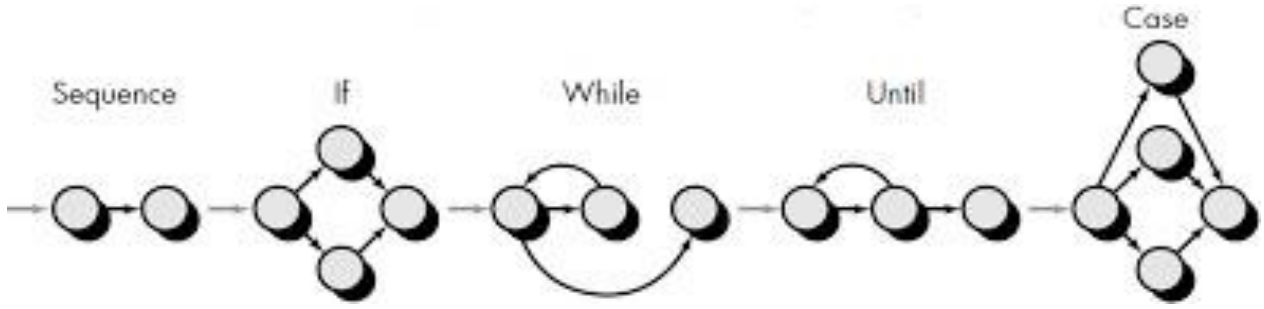
1. **Coverage of All Independent Paths:** It ensures that every independent path within a module is traversed at least once during testing.
2. **Evaluation of Logical Decisions:** It comprehensively assesses all logical decisions from both their true and false perspectives.
3. **Loop Validation:** It verifies the behavior of loops by testing them at their boundaries and within their operational parameters.
4. **Data Structure Examination:** It involves scrutinizing the internal data structures to confirm their integrity and correctness.

In essence, white-box testing provides a methodical way to validate software by diving into its structural intricacies and ensuring that every aspect of its operation is rigorously examined.

Basis Path Testing: Notion of a "basis" has attractive possibilities for structural testing. A basis in terms of a structure called a "vector space", which is a set of elements (called vectors) and which has operations that correspond to multiplication and addition defined for the vectors. Basis path testing is a white-box testing technique first proposed by Tom McCabe [MCC76]. The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one
time during testing.Cyclomatic complexity is a software metric that provides a quantitative measure of the logical complexity of a program. CC defines the number of independent paths in the basis set of a program and Provides us with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once. An independent path is any path through the program that introduces at least one new set of processing statements or a new condition. A simple notation for the representation of control flow, called a flow graph (or program graph) must be introduced. The flow graph depicts logical control flow using the following notation:

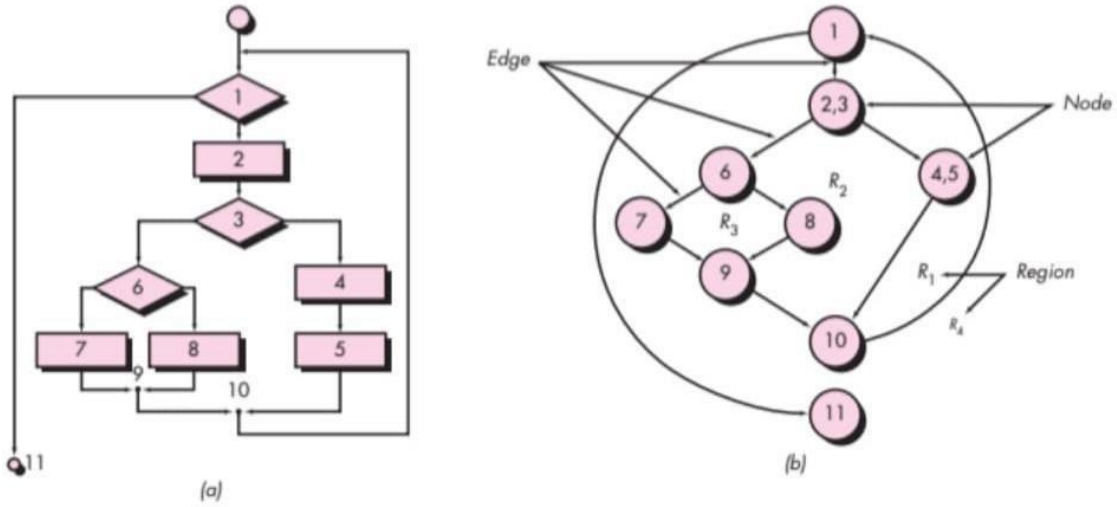






To illustrate the use of a flow graph, consider the procedural design representation in The figure below. Here, a flowchart is used to depict program control structure. Map the flowchart into a corresponding flow graph (assuming

that no compound conditions are contained in the decision diamonds of the flowchart). Each circle, called a flow graph node, represents one or more procedural statements. A sequence of process boxes and a decision diamond can map into a single node. The arrows on the flow graph, called edges or links, represent flow of control and are analogous to flowchart arrows. An edge must terminate at a node, even if the node does not represent any procedural statements (eg, see the flow graph symbol for the if-then-else construct). Areas bounded by edges and nodes are called regions. When counting regions, we include the area outside the graph as a region.



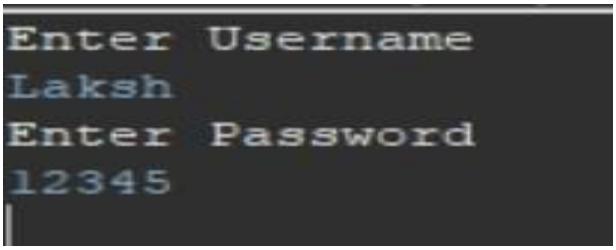
IMPLEMENTATION OF TEST CASE:

```
import static org.junit.Assert.*;
import org.junit.Test;
public class UserAuthenticationTest {
   @Test
   public void testAuthenticateUserValid() {
      // Arrange
      String username = "Laksh";
      String password = "12345";
      boolean isAuthenticated = UserAuthentication.authenticateUser(username,
password);
      assertTrue(isAuthenticated);
   }

   @Test
   public void testAuthenticateUserInvalid() {
      // Arrange
      String username = "Laksh";
      String password = " ib2XPqcFIRHmgVrJ ";

      // Act
      boolean isAuthenticated = UserAuthentication.authenticateUser(username,
password);

      // Assert
```

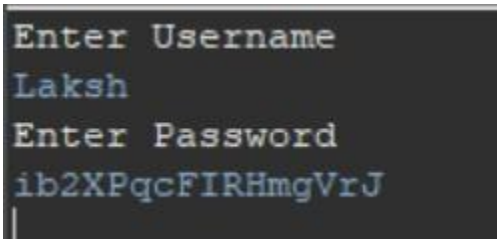
```
        assertFalse(isAuthenticated);
    }
}
```

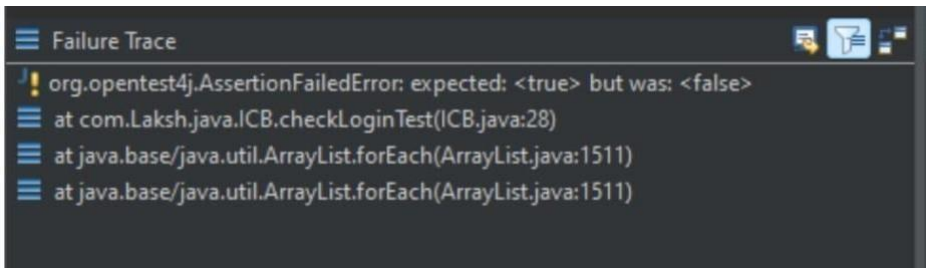
Output :

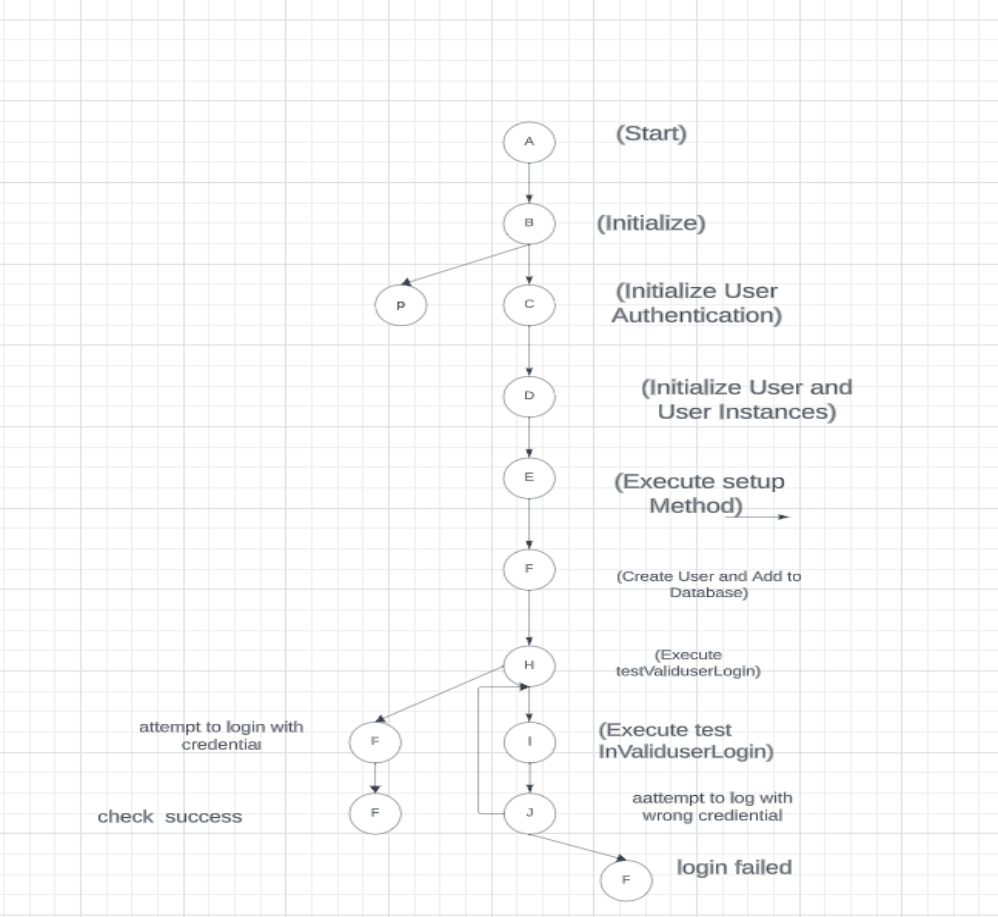
Valid:



Invalid:







Now, let's calculate the Cyclomatic Complexity:

- E (Edges) = 14

- N (Nodes) = 14

- P (Connected Components) = 1 (since the graph is a single connected component)

Using the formula:

V(G) = E - N + 2P

V(G) = 14 - 14 + 2*1

V(G) = 0 + 2

V(G) = 2

The cyclomatic complexity of the flow graph is 2.

C3
C32
2103164.

Assignment No-1 6/10/23

Q.) Architectural Design: Explain in detail each type with example.

⇒ Architectural design in software engineering is about decomposing the system. The software needs the architectural design to represent the design of software. IEEE defines architectural design as "the process of collection of hardware and software components and their interfaces to establish the framework for the development of computer system".
The software that built for computer-based system can exhibit one of these many architectural styles.
Each style will describe a system category that consist of :-

- A set of components that will perform a function required by the system.
- The set of connector will help in coordination, communication and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help designer to understand the overall properities of the system.

The use of architectural style is to establish a structure for all component of the system.

Taxonomy of Architectural styles:-

1.) Data centered architecturs:-

- A data store will reside at the center of this architecture and is accessed frequently by other components that update, add, delete or modify the data present within the store.

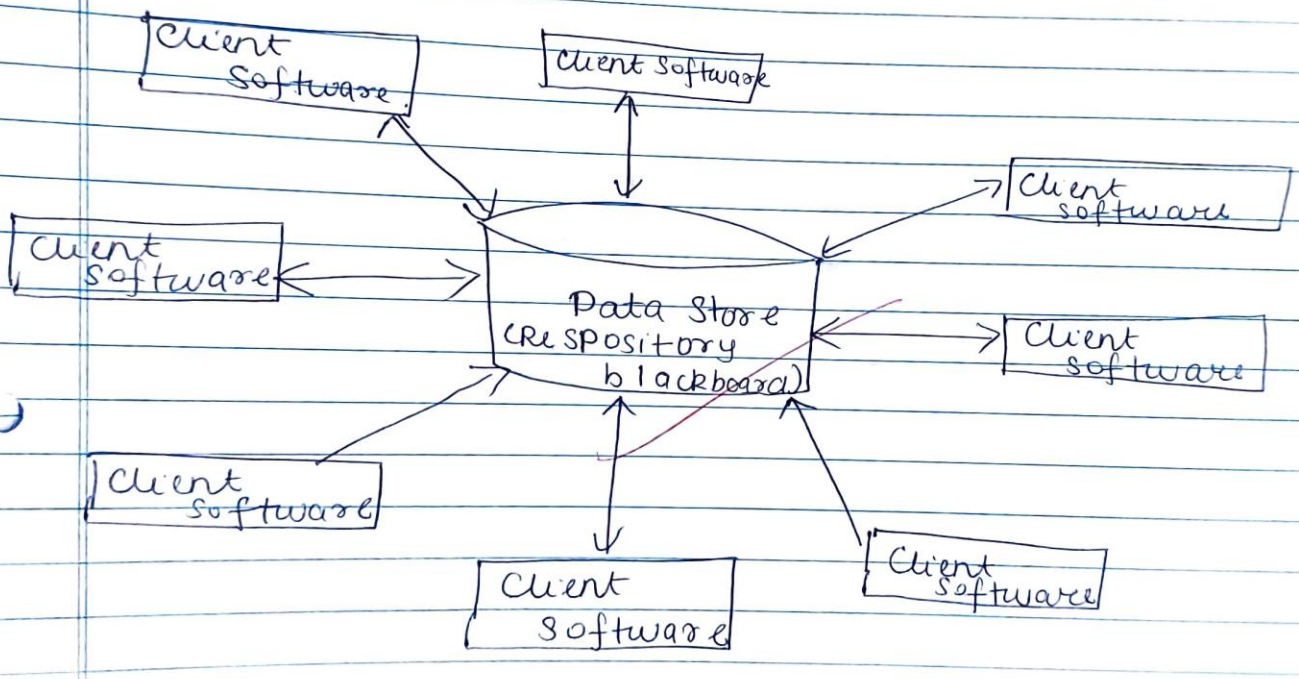
- The figure illustrates a typical data centred style. The client software access a central repository. Variation of this approach are used to transform the repository into blackboard when data related to client or data of interest for the client change the notification to client software

- This data-centered architecture will promote integrability. This means that existing component can be changed and new client component can be added to architecture without permission or concern of other clients.

- Data can be passed among clients using blackboard mechanism

Advantage of Data centred architecture

- Rispository of data is independent of clients
- Client work independent of each other.
- It may be simple to add additional clients
- Modification can be every easy



Data centered architecture

Shruti
C32
C3
103164

2.) Data flow architectures :-

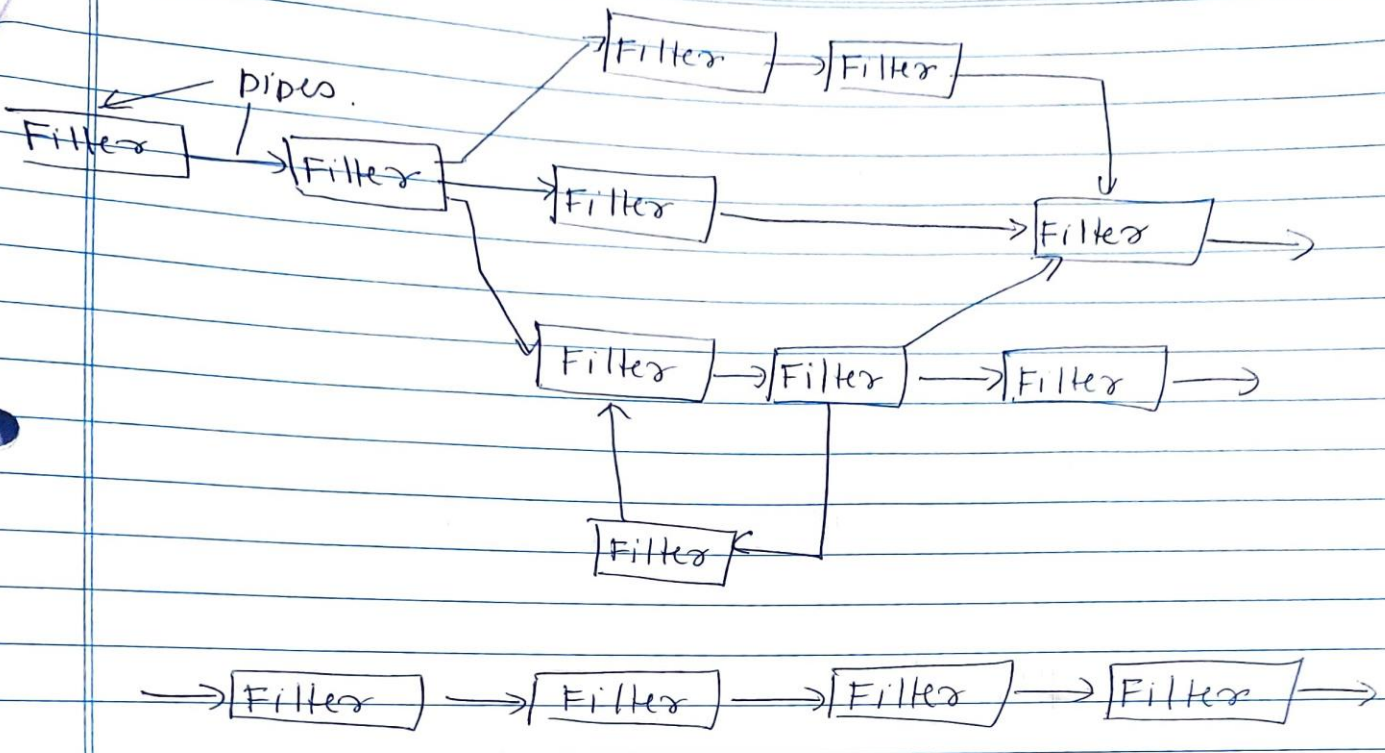
- This kind of architecture is used when input data is transformed into output data through a series of computational manipulative components.
- The figure represents pipe and filter architectural since it uses both pipe and filter and it has a set of components called filters connected by lines
- Pipe are used to transmitting data from one component to next
- Each filter will work idependatly and is designed to take data input of a certain form and produces data output to next filter of a specified form. The filters don't require any knowledge of working of neighboring filters.
- If data flow degenrates into single line to transforms, then it is termed as batch sequential. This structure accepts the batch of data and then applies a series of sequential components to transform it

Advantages :-

1. It encourges unkeep, reuporsing & modification
2. With design, concurrent esecution is supported.

Disadvantages :-

1. It frequently degenerates to batch sequential system
2. It is not easy to coordinate two different but related streams.

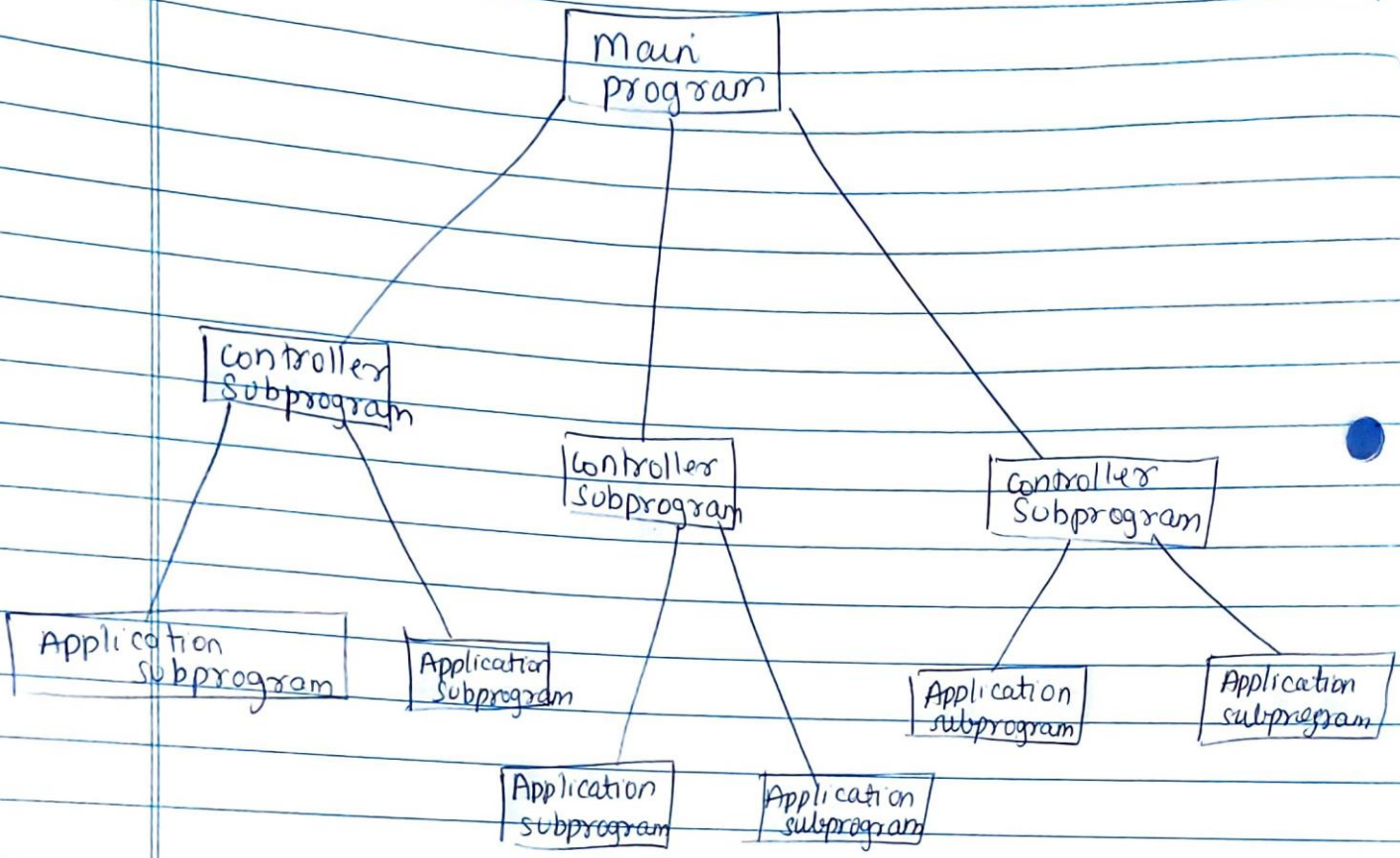


Data flow architecture.

3.) Call and Return architectures :- It is used to creat a program to easy to scal and modify Many sub styles exist within this category Two of them are explained below.

- Remote procedure call architecture :- This component is used to present in a main program or sub program architecture distributed among multipli computer on a network
- Main program or subprogram architecture :- The main program structure decomposes into number of sub programs or function into control hierarchy. Main program contain number of sub program

that can invoke other components.



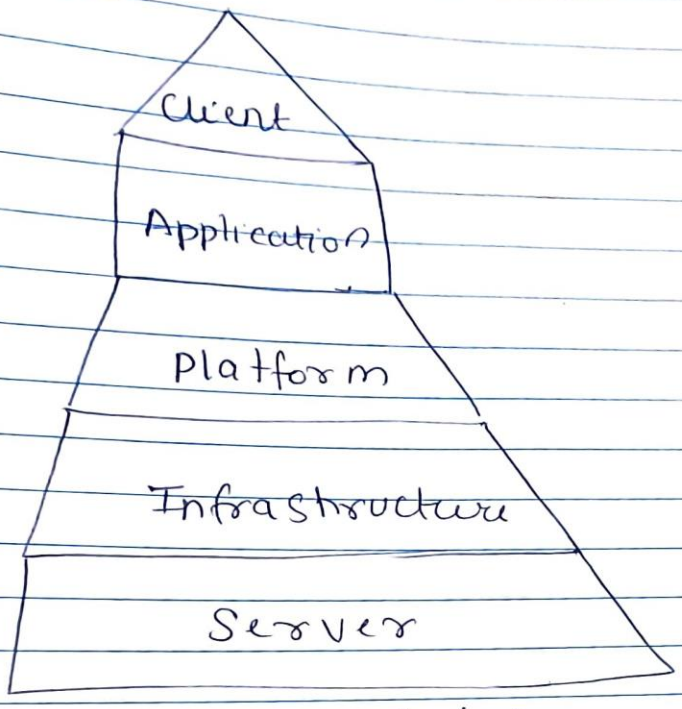
4] Object oriented architecture :- The component of system encapsulate data and operation that must be applied to manipulate the data.

Characteristics :-
- Object protect the system's integrity
- An object unware of depiction of their items.

Advantages :-
- Enables disigner to separate a challenge into collection of autonomous object
- Other object are aware of implementation detail of object allowing changes to be made without having impact on other object.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

5) Layered architecture



Layered architecture

- A number of different layer are defined with each layer performing well defined set of operation Each layer will do some operation that becomes closer to machine instruction set progressively.
- At outer layer, component will recieve the user interface operation at inner layer, component will perform the OS interfacing.
- Intermediate layer to utility services & application software functions.
- One common example of this architectural style is OSI – ISO Communication system.

Shruti C32 C3 2103164

Assignment No - 2

Explain in detail :-

Pawar 6/10/23 (A)

a) Software Maintenance :-

- Maintenance is required to give assurance that the software has ability to stasfy the changing requirement of user
→ Maintenance is important phase for software product which is developed using any SDLC model such as waterfall, spiral or iterative model etc.
→ Due to corrective and non corrective software actions, we need to make changes in software products.

Following are needs of maintenance :-

1. To correct the defects found in software product.
2. To improve the design of software
3. To improve the performance of software product
4. To add new features.
5. To improve communication with other software
6. To transfer legacy software system into new software system and replace old software by new software

There are four types of maintance :-

1. Corrective maintenance :- This type of maintance contains changes performed for fixing of defects found in software product that are affected by end user or tester.

→ In corrective maintance, we perform actions for correcting the defect in our software product that make bad effect on different parts of our software product such as design, logic or code

→ Corrective maintenance has approach to examine the orginal specification to state that what system was orignal designed to do.

20% part of all maintence activities are performed do corrective maintenance.

(2) Adaptive Maintenance :-

→ This category of maintenance contain making changes in software product to maintain software product up date. Adaptive maintenace is used to make change in our software product according to continously modifying world of technology and busines environment.

→ Adaptive maintenance is related with the modification in application which is performed to make the application adaptable with ever changing environment.

→ Adaptive mantance contributes 25% of all maintenance.

(3) Perfective Maintenance :-

→ Perfective maintenance contains changes and updates performed to keep the software usa

for long duration.
→ Perfective Maintenance basically associated with including new or modified user requirement
→ It contain enchacment in both functionality and in efficiency of the source code and modifying the functionalities of the software application according to changing needs of user. Perfective maintenance contributes 50% of total maintance which is largest of all maintenance activities

Preventive Maintenance :-
→ Preventive Maintenance contain changes and updation to avoid problem in future software application.
→ Its objective is solve the problem which are not major at this point but may cause errors issue in future.
→ It includes documentation updating, code optimization & code restructing.
→ Documentation updating includes making changes in document code optimization includes changing the programs. Code restructing includes changing the structure of program for decreasing complexity of source code & making it understandable
→ preventive maintance contributes 5% of all maintance activities.

TSEC ENGINEERING COLLEGE

⑥ Re-engineering :-

→ When we required to update the software application without affecting its functionality so that it can stay in current market, it is known as software re-engineering.

→ Software re-engineering is a process of restructing or rewriting component of old software application without modifying its functionality.

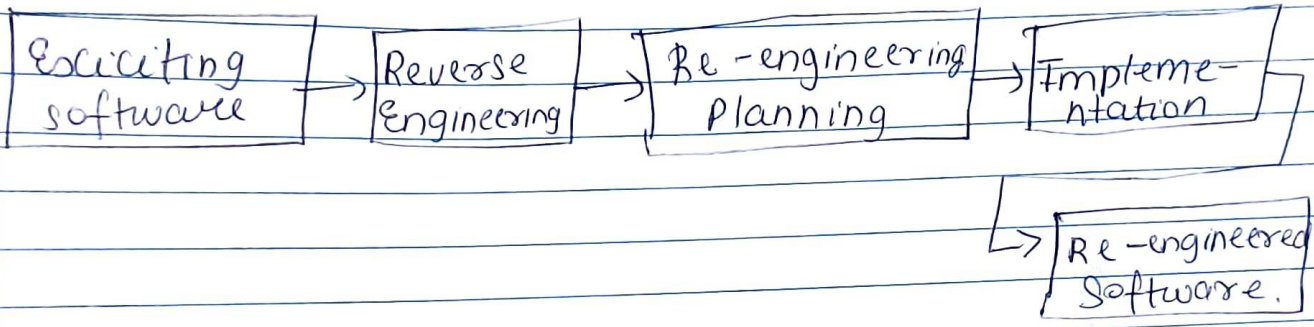
→ Old software system cannot use with latest technology available in market. If we use it perform some to then we may face lot of problem such as hardware become out of date, updating of software becomes a problem. for example, in beginning UNIX was developed using assembly language. When c language was developed, UNIX was re-eginnered in C, as writing source code in assembly language and understanding it becomes difficults.

→ Reengineering procedure includes the following steps:

① Decide which component of software we want to re-engineering. It is complete software or some component of it.

② Do reverse engineering to find out features of existing software

Perform restructuring of source code is needed. for example, modifying function oriented program in object oriented program
4) Perform restructing of data if needed
5) Use forward engineering idea to generate re-engineering-software.



Reverse engineering:-

- It is procedure to get system specification by analyzing and understanding existing system. This procedure can be reverse software development life cycle model. i.e go from maintenance phase to requirement gathering phase.
→ Reverse engineering is procedure that recognize an object a device or technical properties of system through analysis of architecture of system function of system through analysis of architecture of system function and operations.
- Reverse engineering is process of analyzing an object to observe how software work in order to duplicate or add new feature in object

→ Idea of reverse engineering is invented in older industries and now many time it is used in computer hardware & software generation

→ Software reverse engineering includes reversing a machine code of program which is format of sequence of 0s and 1s. It is transformed into source code.