1. INTRODUCTION

1. INTRODUCTION

Our DIY Project Sharing Platform, where creativity meets community! In a world increasingly driven by the desire for personalization and hands-on experiences, our platform serves as a vibrant hub for DIY enthusiasts of all skill levels. Whether you're a seasoned crafter, a weekend warrior, or just starting your journey into the world of do-it-yourself projects, our platform offers a space to explore, share, and inspire.

Here, users can discover a diverse array of projects ranging from home décor and gardening to fashion and technology. Each project is accompanied by detailed instructions, materials lists, and user-generated tips, making it easier than ever to bring your ideas to life. But what truly sets our platform apart is the community aspect; users can connect with fellow DIYers, share their unique creations, and provide feedback and encouragement.

Features:

- Project Database: A comprehensive collection of user-submitted DIY projects across various categories, including home décor, crafts, fashion, gardening, and tech.

- Step-by-Step Tutorials: Each project includes detailed instructions, materials lists, and tips to help users successfully complete their creations.

- User Profiles: Members can create profiles showcasing their skills, interests, and completed projects, allowing others to follow their work and engage with them.

- Community Interaction: Users can comment on projects, ask questions, and provide feedback, fostering a supportive environment for learning and sharing.

- Resource Library: Access to articles, videos, and guides on various DIY techniques, tools, and materials to enhance your skills.

- Inspiration Board: A curated space where users can save their favorite projects and ideas for future reference.

1.1 ABOUT THE PROJECT

Our DIY Project Sharing Platform is designed to connect creative individuals who love to craft, build, and innovate. The platform serves as a central hub for sharing ideas, tutorials, and completed projects, fostering a community of makers who inspire and support one another. Our DIY Project Sharing Platform is designed to connect creative individuals who love to craft, build, and innovate. The platform serves as a central hub for sharing ideas, tutorials, and completed projects, fostering a community of makers who inspire and support one another.

2. SYSTEM STUDY AND ANALYSIS

2.1 REQUIREMENT ANALYSIS

2.1.1 Existing System Details and Disadvantages

Existing System Details:

Many existing DIY project-sharing platforms or content-sharing platforms allow users to post projects, tutorials, or guides, often with a focus on specific types of content like recipes, crafts, or technology. Typically, these platforms offer basic functionality such as uploading text, images, and videos, categorizing content, and allowing users to comment or share projects.

Disadvantages of Existing Systems:

Limited Interactivity: Many existing platforms lack robust interactivity features, making it difficult for users to engage deeply with content beyond basic comments or likes.

Poor User Experience: Some platforms have clunky interfaces that are not user-friendly, leading to poor navigation, difficulty in finding relevant content, and a lack of customization in user profiles.

Inadequate Media Support: Existing systems often have limited support for multimedia uploads, particularly large video files or high-resolution images. This restricts users from fully documenting and sharing their projects.

Lack of Comprehensive Project Management: Many platforms do not provide comprehensive tools for users to manage their projects, such as the ability to edit or delete content after it has been posted, or organize projects under specific categories effectively.

Insufficient Security: Some systems do not enforce strong security measures like password policies or email verification, making them vulnerable to spam, unauthorized access, or user data breaches.

2.1.2 PROPOSED SYSTEM

The proposed DIY Project Sharing Platform is designed to address the shortcomings of existing platforms by providing a more structured, feature-rich, and user-centric approach to project sharing. The system will focus on creating a community of enthusiasts who can easily share and discover new projects while interacting with other users. The key features and modules of the proposed system include:

User Registration and Management:

Secure user registration with strong password policies and email verification to ensure user legitimacy.

Profile management for users to customize their accounts and manage personal information.

Login and Authentication:

Secure login functionality to ensure that only authorized users can access their accounts and project data.

Project Management Module:

Users can create, edit, and delete projects with ease, including the ability to add titles, descriptions, and categories.

Users can upload images, videos, and other media to enhance their project documentation.

Media Upload Module:

Supports the upload of various media types, such as images, videos (via URL), and PDFs, to provide a comprehensive project presentation.

Step-by-Step Instructions Module:

Allows users to create detailed guides with step-by-step instructions, including text, images, and video content, to ensure clarity and ease of understanding.

Category and Tagging Module:

Enables users to categorize and tag their projects for better organization and searchability, making it easier for others to discover relevant content.

2.2 FEASIBILITY STUDY

A feasibility study is crucial in determining whether the proposed system is viable from technical, economic, and operational perspectives. Below is an analysis of the feasibility of the DIY Project Sharing Platform:

2.2.1 Operational Feasibility:

The platform is designed to be user-friendly, with intuitive interfaces that make it easy for users to create and manage their projects.

The proposed system meets the needs of the target audience by providing enhanced project sharing capabilities and a community-driven platform, making it operationally feasible.

The system's ability to scale and adapt to user needs ensures its sustainability and success in an academic environment.

2.2.2 Technical Feasibility:

The project will be built using widely accepted and powerful technologies such as PHP, PostgreSQL, HTML, CSS, JavaScript, and Bootstrap. These technologies are well-supported, have a strong developer community, and are suitable for building scalable web applications.

The use of PostgreSQL ensures a robust, reliable, and secure database system that can handle complex queries and large volumes of data efficiently.

The development tools such as VSCode and Sublime Text provide a conducive environment for coding, testing, and debugging, ensuring the project is technically feasible.

2.2.3 Economic Feasibility:

Since this is a student project, the primary costs involved are related to software and hardware resources, which are typically provided by the educational institution.

Open-source tools and technologies (e.g., PHP, PostgreSQL) are used, reducing the overall cost of the project development.

The project can be completed within the given budget constraints and is economically feasible for academic purposes.

The proposed DIY Project Sharing Platform is a technically and economically viable solution designed to overcome the limitations of existing systems. By implementing advanced project management features, robust media support, and user-friendly interfaces, the platform promises to enhance the user experience and create a vibrant community of project enthusiasts. The feasibility study confirms that the project is feasible within the constraints of a student project, making it a valuable addition to academic learning and skill development.

include government grants, private investments, or public-private partnerships. Cost savings are also considered, considering waste reduction, recycling, and improved efficiency. Risk analysis is conducted to assess potential risks.

2.3 REQUIREMENT SPECIFICATION

2.3.1 SOFTWARE SPECIFICATION

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development

2.3.2 HARDWARE SPECIFICATION

1. Server Hardware

- **Processor**:
  - **Minimum**: Intel Core i5 or AMD Ryzen 5
  - **Recommended**: Intel Xeon or AMD EPYC
  - **Purpose**: Handles server-side processing and ensures smooth operation of PHP scripts, database queries, and user requests.
- **Memory (RAM)**:
  - **Minimum**: 8 GB
  - **Recommended**: 16 GB or more
  - **Purpose**: Supports multiple concurrent users and processes, enhancing performance and responsiveness.

2. Client Hardware

- **Processor**:
  - **Minimum**: Intel Core i3 or AMD Ryzen 3
  - **Recommended**: Intel Core i5 or AMD Ryzen 5
  - **Purpose**: Handles web browsing and interaction with the LMS.
- **Memory (RAM)**:

    - **Minimum**: 4 GB
    - **Recommended**: 8 GB or more
    - **Purpose**: Supports smooth operation of web browsers and multitasking.

2.3.3 SOFTWARE REQUIREMENTS

1. Operating System

- **Windows**:
    - **Minimum**: Windows 10 or Windows Server 2016
    - **Recommended**: Windows 11 or Windows Server 2022

2. Development Environment

- **XAMPP**:
    - **Version**: XAMPP 8.x (latest stable version)
    - **Components**:
        - **Apache**: Web server for hosting PHP applications.
        - **PHP**: Server-side scripting language for application logic.
        - **phpMyAdmin**: Web-based interface for MySQL/MariaDB management (not needed if using PostgreSQL).
- **PostgreSQL**:
    - **Version**: PostgreSQL 13.x or newer
    - **Purpose**: Database management system for handling application data.
    - **Tools**:
        - **pgAdmin**: Web-based tool for managing PostgreSQL databases.
- **Code Editors**:
    - **Sublime Text**: Lightweight and fast text editor for coding.
    - **Visual Studio Code (VSCode)**: Feature-rich code editor with extensions for PHP, HTML, CSS, JavaScript, and PostgreSQL.
- **Web Technologies**:

- **PHP**: Programming language for server-side logic.
- **Bootstrap**: Front-end framework for building responsive web designs.
- **CSS**: Stylesheets for designing the user interface.
- **HTML**: Markup language for structuring web content.
- **JavaScript**: Scripting language for client-side interactions.

- **Web Browser**:
  - **Recommended**:
    - **Google Chrome**
    - **Mozilla Firefox**
    - **Microsoft Edge**
  - **Purpose**: Testing and accessing web applications to ensure compatibility with modern web technologies.

ABOUT TECHNOLOGIES USED:

HTML, CSS, and Bootstrap

HTML (HyperText Markup Language) is the foundational language used to create and structure content on the web. It uses a system of tags and attributes to define elements such as headings, paragraphs, links, images, and more. HTML provides the basic skeleton of a webpage, allowing developers to organize text, multimedia, and forms into a cohesive structure. It is a crucial component of web development, as it establishes the document's semantic structure, enabling browsers to render content appropriately and ensuring accessibility and search engine optimization (SEO).

CSS (Cascading Style Sheets) is a style sheet language used to control the presentation of HTML documents. By applying styles such as colors, fonts, and layouts, CSS enhances the visual appeal and user experience of web pages. CSS enables developers to separate content from design, allowing for cleaner and more maintainable code. Through features like selectors, properties, and media queries, CSS provides flexibility in styling individual elements and creating responsive designs that adapt to various devices and screen sizes. Combined with HTML, CSS forms the backbone of web design, creating visually engaging and user-friendly interfaces.

Bootstrap is a popular front-end framework that simplifies the development of responsive and mobile-first websites. It provides a collection of pre-designed components, such as buttons, forms, modals, and navigation bars, along with a responsive grid system that helps structure web pages efficiently. Bootstrap provides HTML, CSS, and JavaScript to create consistent and professional-looking interfaces quickly. Its extensive library of customizable components allows developers to build complex layouts with minimal effort. Bootstrap's responsiveness and cross-browser compatibility make it a go-to framework for ensuring that web applications look and function well across various devices and platforms.

JavaScript

JavaScript is a versatile programming language primarily used for creating interactive and dynamic web content. It enables developers to add functionality such as form validation, animations, and event handling to web pages. JavaScript can manipulate the DOM (Document Object Model), allowing real-time updates and interactions without requiring a page reload. Modern JavaScript frameworks and libraries like React, Angular, and Vue.js further extend its capabilities, facilitating the development of complex single-page applications (SPAs). As a client-side language, JavaScript enhances user experiences by enabling responsive and engaging interfaces that react to user inputs seamlessly.

PHP (Hypertext Preprocessor) is a widely-used server-side scripting language designed for web development. It is embedded within HTML and executes on the server, generating dynamic content that is sent to the client's browser. PHP is particularly well-suited for building data-driven websites and applications, as it easily integrates with various databases like MySQL and PostgreSQL. Its simplicity, flexibility, and extensive documentation make PHP a preferred choice for creating content management systems (CMS), e-commerce platforms, and other web applications. With its ability to handle form submissions, manage sessions, and interact with databases, PHP is a powerful tool for developing robust and scalable web solutions.

PostgreSQL

PostgreSQL is a powerful, open-source relational database management system (RDBMS) known for its robustness, extensibility, and standards compliance. It supports a wide range of data types and provides advanced features like full-text search, JSON support, and multi-version concurrency control (MVCC). PostgreSQL's adherence to SQL standards ensures compatibility with other databases and tools, while its extensible architecture allows for the addition of custom functions and data types. It is widely used in industries requiring reliable and scalable database solutions, from small-scale applications to large-scale data warehousing.

PostgreSQL's strong emphasis on data integrity and security makes it a reliable choice for mission-critical applications. Its support for ACID (Atomicity, Consistency, Isolation, Durability) transactions ensures that data remains accurate and consistent even in the event of system failures. Additionally, PostgreSQL's extensive indexing capabilities, including B-tree, hash, GiST, SP-GiST, and GIN indexes, optimize query performance. Its active community and documentation provide a wealth of resources for developers, ensuring ongoing improvements and support. As a result, PostgreSQL is a preferred database for developers seeking a versatile, high-performance solution for managing complex data structures and workloads. System Design

Overview: The system design phase involves defining the architecture, components, and interfaces of the DIY Project Sharing Platform. This includes identifying the major modules, database schema, user interfaces, and how they interact to form a cohesive system.

VISUAL STUDIO CODE

Visual Studio Code is a source-code editor that can be used with a variety of program languages, including C, C#, C++, Fortan, Go, Java, JavaScript, Node.js, Python, PHP, Rust, and Julia. It is based on the Electron framework, which is used to develop Node. Visual Studio Code also commonly referred to as VS Code, is a source code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactory and embedded Git. Users can change the theme, keyboard shortcuts, preferences and install extensions that add functionality.

2.3.4 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software-system Component. A function is described as a set of inputs, the behavior and outputs. Functional Requirements may be calculations, technical details, data manipulation and processing and other specific functionality that show how a use case to be fulfilled. Typically, a requirements analyst generates functional requirements after building use case. However, this may have exceptions since software development is an iterative process and sometime certain requirements are conceived prior to the definition of the use case. Both artifacts (use case documents and requirements documents) complement each other in a bidirectional process.

A typical functional requirement will contain a unique name and number, a brief summary, and a rationale. This information is used to help the reader understand why the requirement is needed, and to track the requirement through the development of the system. The core of the requirement is the description of the required behavior, which must be a clear and readable description of the required behavior. This behavior may come from or business rule, or it may be discovered through elicitation sessions with users, stakeholders and other experts within the organization. Software requirements must be clear, correct unambiguous, specific and verifiable. A modular design reduces complexity, facilitates change and results in easier implementation by encouraging parallel development parts of a system. Software with effective modularity is easier to develop because of function may be compartmentalized and interfaces are simplified

- **Functional Requirements:**
  - User Registration and Login: Users must be able to create accounts, log in, and manage their profiles.
  - Project Management: Users should be able to create, edit, and delete their own projects, including adding project details, step-by-step instructions, image, PDFs and video URLs.
  - Project browsing and Searching: Users should be able to search for projects based on various filters such as category type, project name, and project id.
  - Social Features: The platform should allow users to rate, review, and comment on projects.

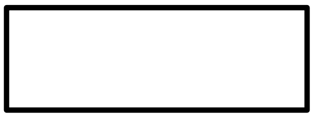
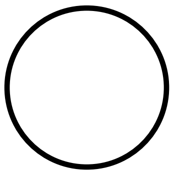
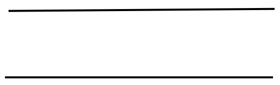
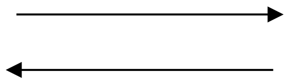
- **Non-Functional Requirements:**
    - Usability: The platform must be easy to navigate with a clean and intuitive user interface.
    - Performance: The platform should load quickly and handle multiple users simultaneously without performance degradation.
    - Security: User data, including personal information and login credentials, must be securely stored and protected against unauthorized access.

2.3.5 DATA FLOW DIAGRAMS

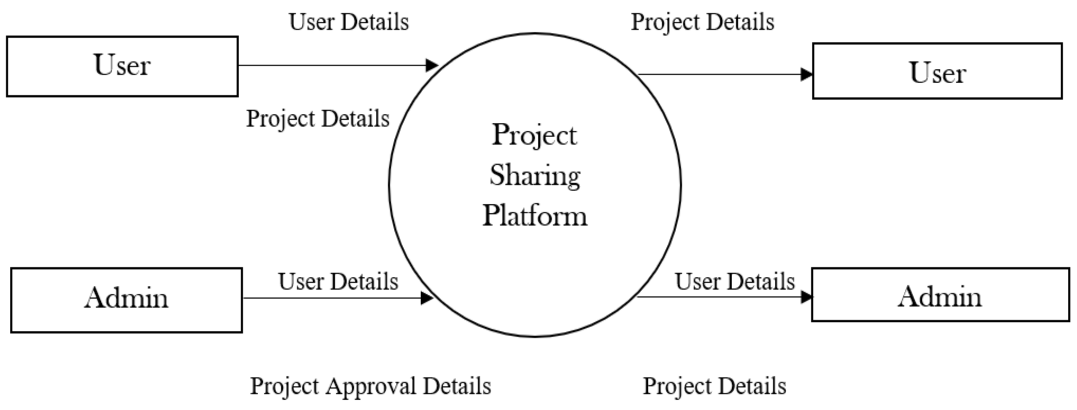
The Data Flow Diagram (DFD) represents a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources. The purpose of DFD is to provide a semantic bridge between users and system developers. The diagram is the basis of structured system analysis. A level 0 DFD, also called a fundamental system model or a context model represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in the next level i.e., Level 1 DFD. Each of the processes represented at Level 1 are sub functions of overall system depicted in the context model. Any processes, which are complex in Level 1, will be further represented into sub functions in the next level, i.e., in level 2. Data flow diagrams illustrate how data is processed by a system in terms of inputs, and output represented by major components or functions with Circles.

Components of Data Flow Diagrams

There are only four symbols that are used in the drawing of data flow diagrams. These are explained below together with the rules that apply to them.

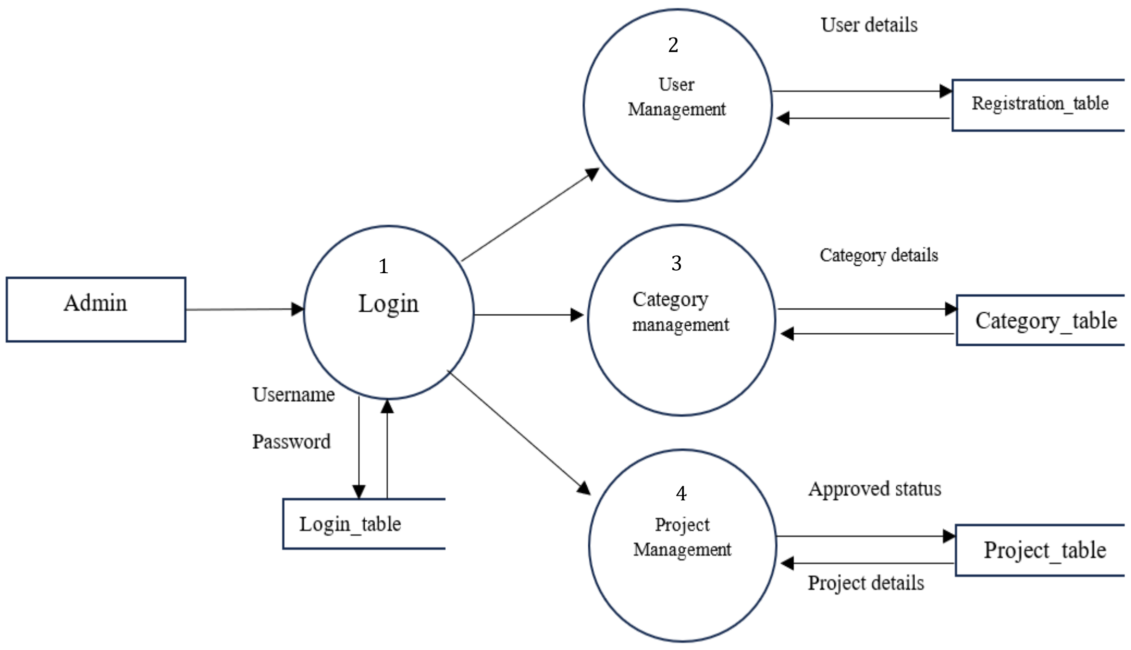
Components	Symbols	Descriptions
External Entities		External entities represent the sources of data that enter the system or the recipients of data that leave the system.
Process		Processes represent activities in which data is manipulated by being stored or retrieved or transformed in some way.
Data store		Data stores represent stores of data within the system. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations.
Data Flow		A data flow shows the flow of information from its source to its destination. A line represents a data flow, with arrowheads showing the direction of flow

CONTEXT DIAGRAM / LEVEL 0 DFD



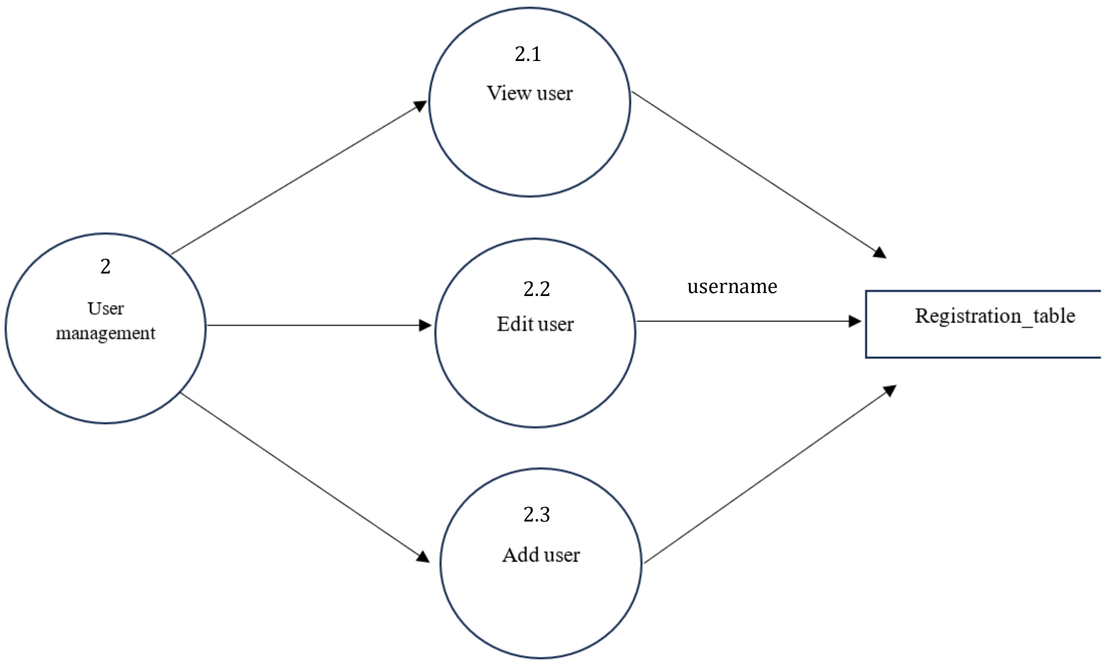
LEVEL 1

DFD FOR ADMIN



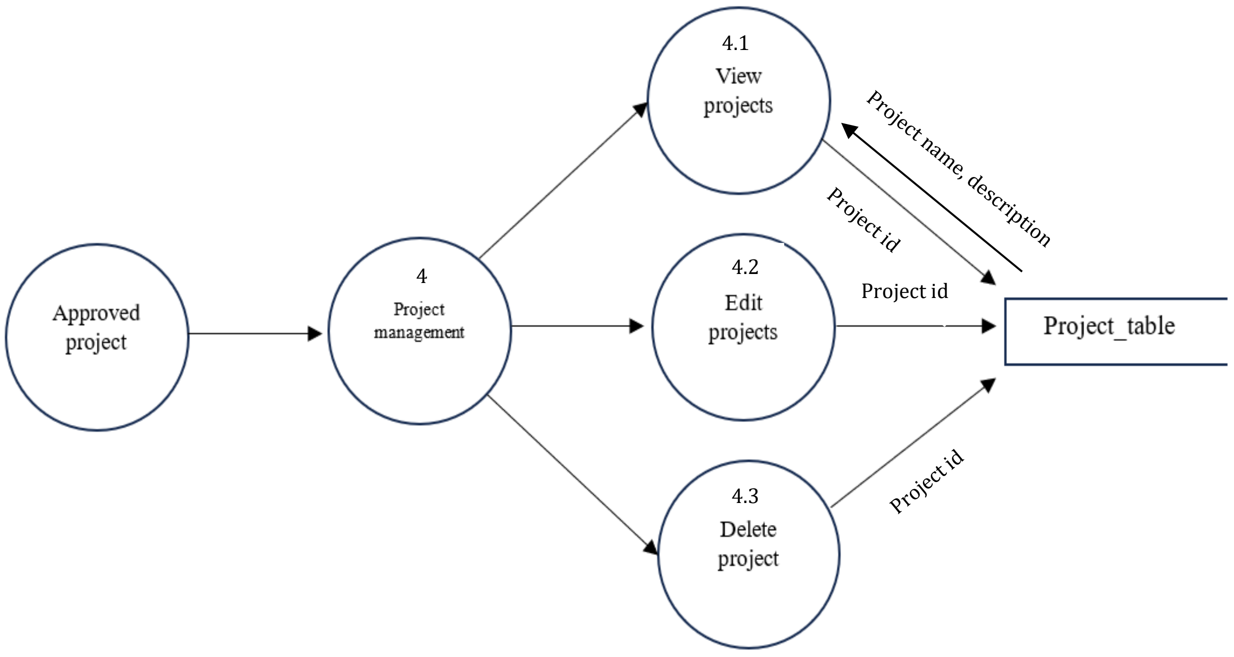
LEVEL 2

DFD FOR USER MANAGEMENT



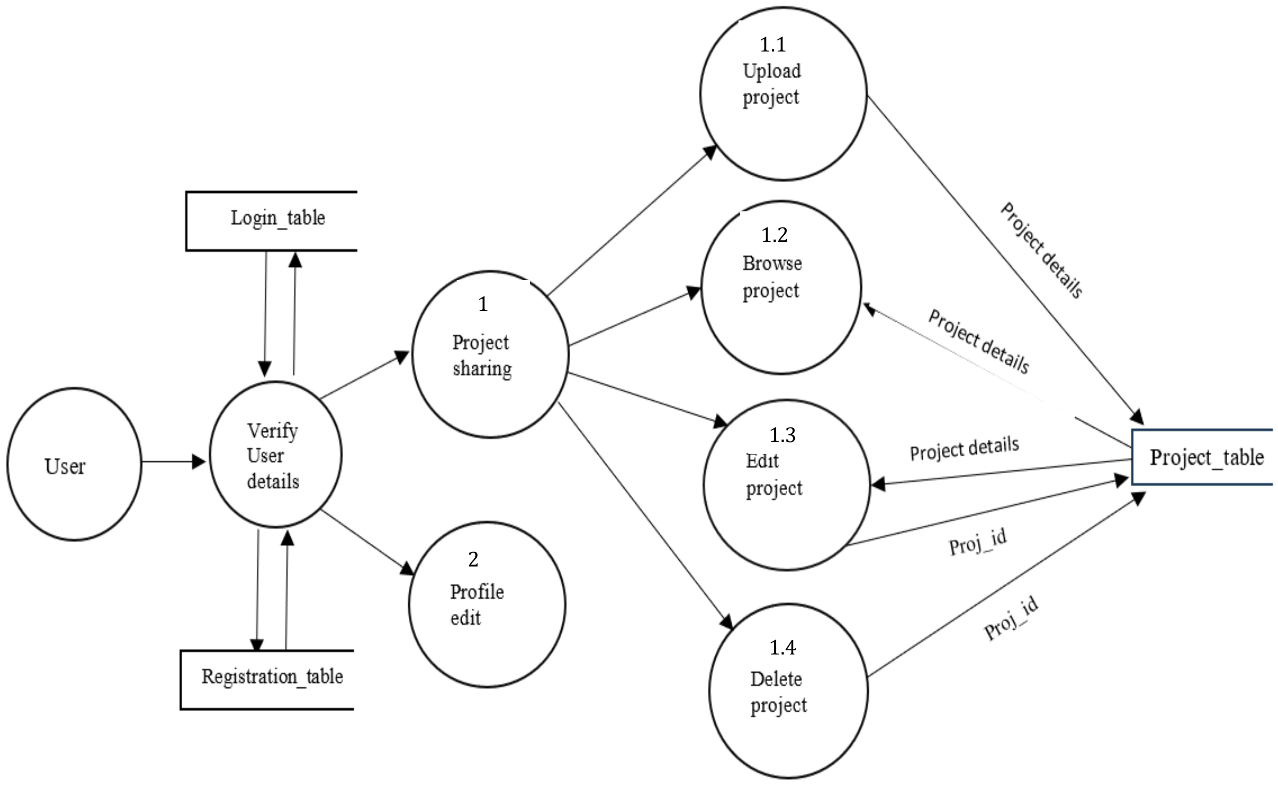
LEVEL 2

DFD FOR PROJECT MANAGEMENT



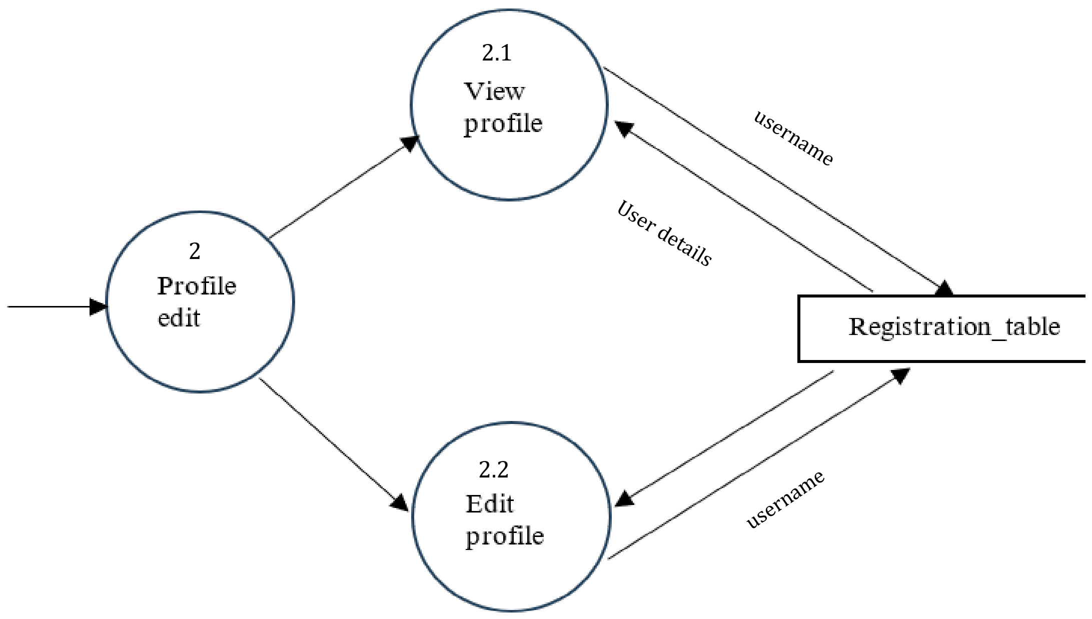
LEVEL 1

DFD FOR USER



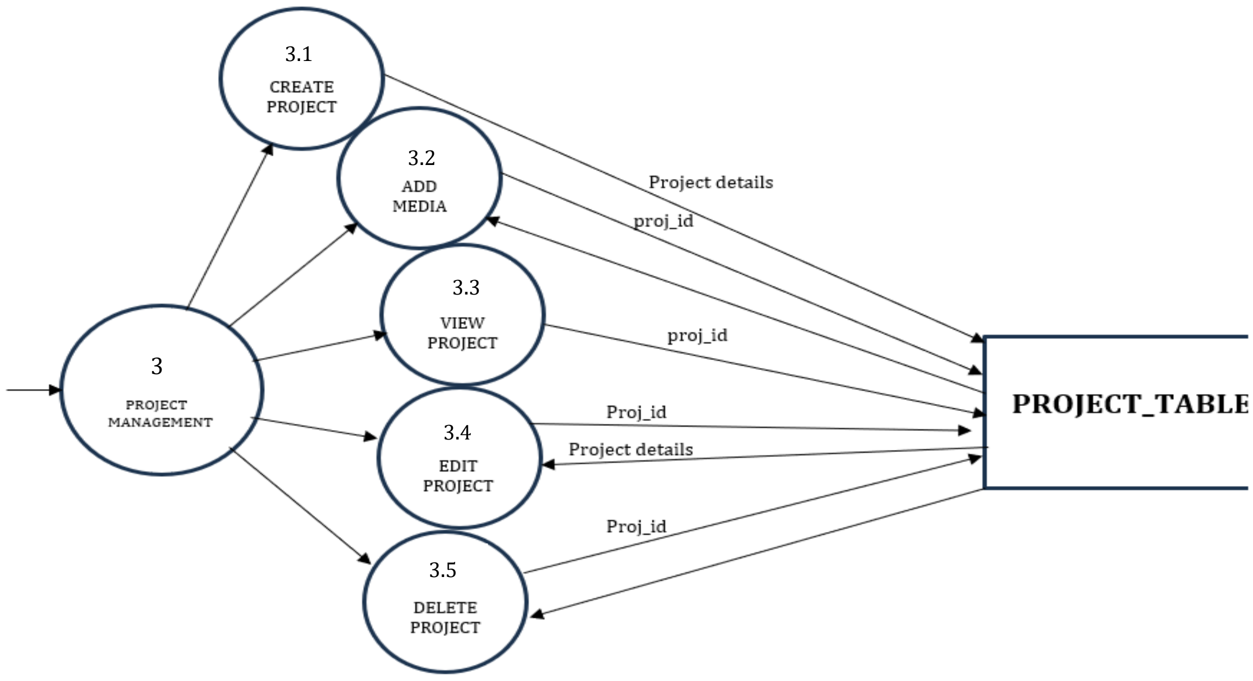
LEVEL 2

DFD FOR USER PROFILE MANAGEMENT



LEVEL 2

DFD FOR USER PROJECT MANAGEMENT



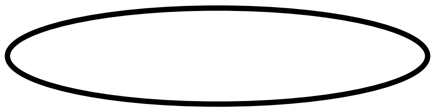
2.3.6 E-R DIAGRAM

Diagrams created to design entities and relationships are called entity– relationship diagrams or ER diagrams. In software engineering, an Entity–Relationship is an abstract way to describe a database. It usually starts with a relational database, which stores data in tables. The overall logical structure of a database can express graphically with an E-R diagram. Rectangle symbol represents entity types, ellipses represent attributes and diamonds represents relationship types. Lines links attributes to entity types and entity types with other relationship types. Primary key attributes are underlined and double Ellipses are used to represent multi-valued attributes.

Entity

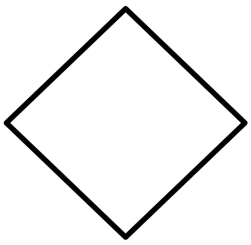
An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. An entity set is a collection of similar type of entities. An entity set may contain entity with attribute sharing similar values.

Attributes



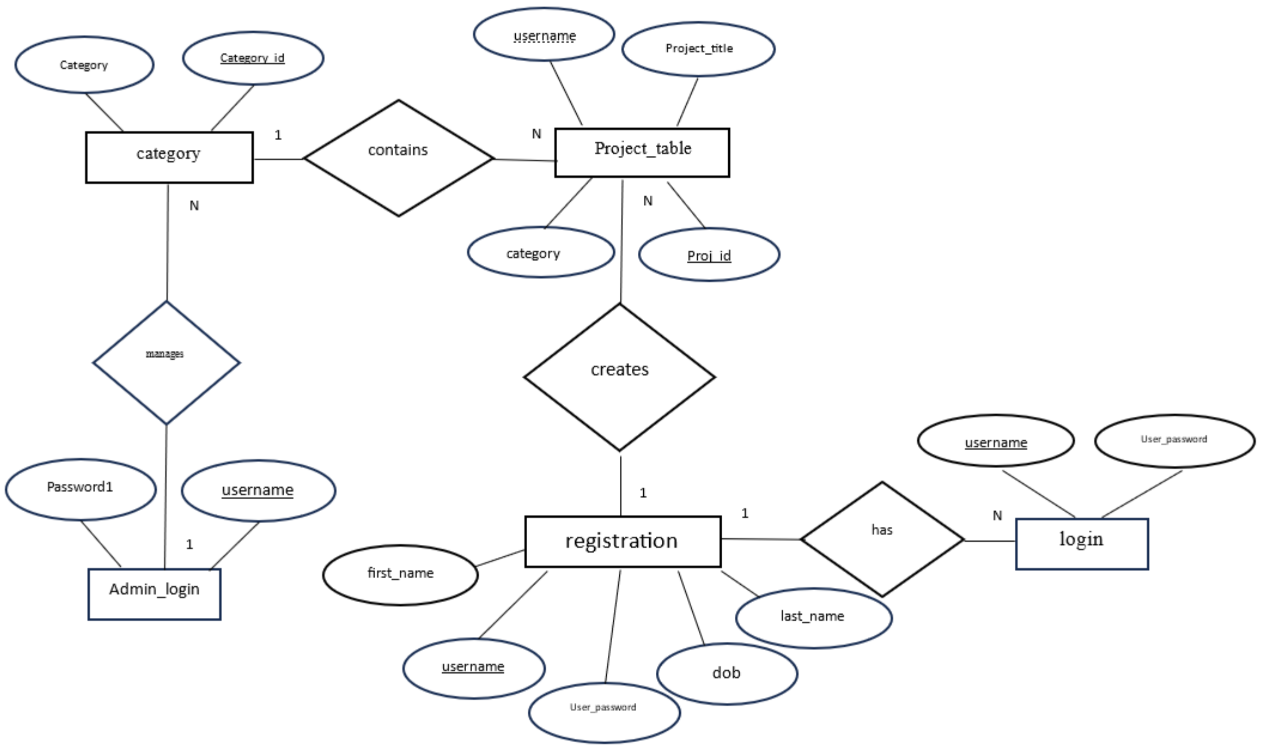
Entities are represented by means of their properties, called attributes. All attributes have values there exist a domain or range of values that can be assigned to attributes.

Relationship



The association among entities is called relationship. A set of relationship of similar type is called a relationship set like entities, a relationship too can have attributes, and these attributes are called descriptive attributes.

ER Diagram of the system



2.3.7 STRUCTURE CHART

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

A structure chart is a top-down modular design tool, constructed of squares representing the different modules in the system, and lines that connect them. The lines represent the connection and or ownership between activities and sub activities as they are used in organization charts.

A structure chart depicts

• The size and complexity of the system.

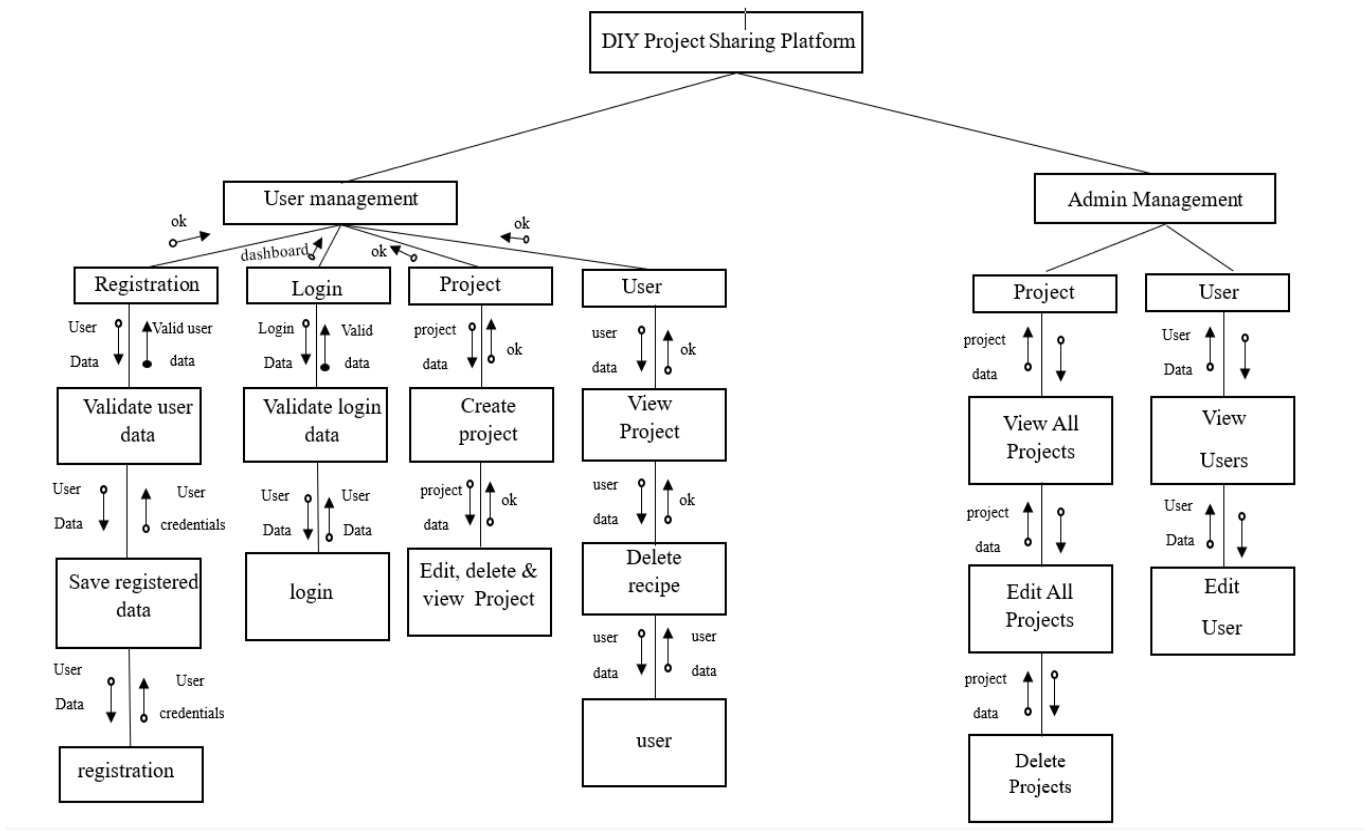
• The number of readily identifiable functions and modules within each function.

• Whether each identifiable function is a manageable entity or should be broken down into smaller components.

Symbols used in structural chart

Modules	▭	Rectangle
Connection	→	Arrow

Structure chart of the system

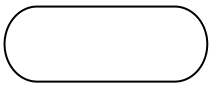
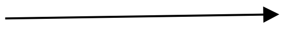
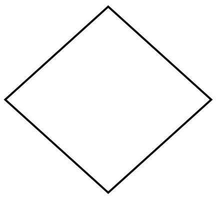


2.3.8 SYSTEM FLOWCHART

Flowchart are used in designing and documenting simple process or program. Like other types of diagrams, they help visualize what is going on and thereby help to understand a process, and perhaps also find flows, bottlenecks, and other less obvious features within it. There are different types of flowchart; each has it own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

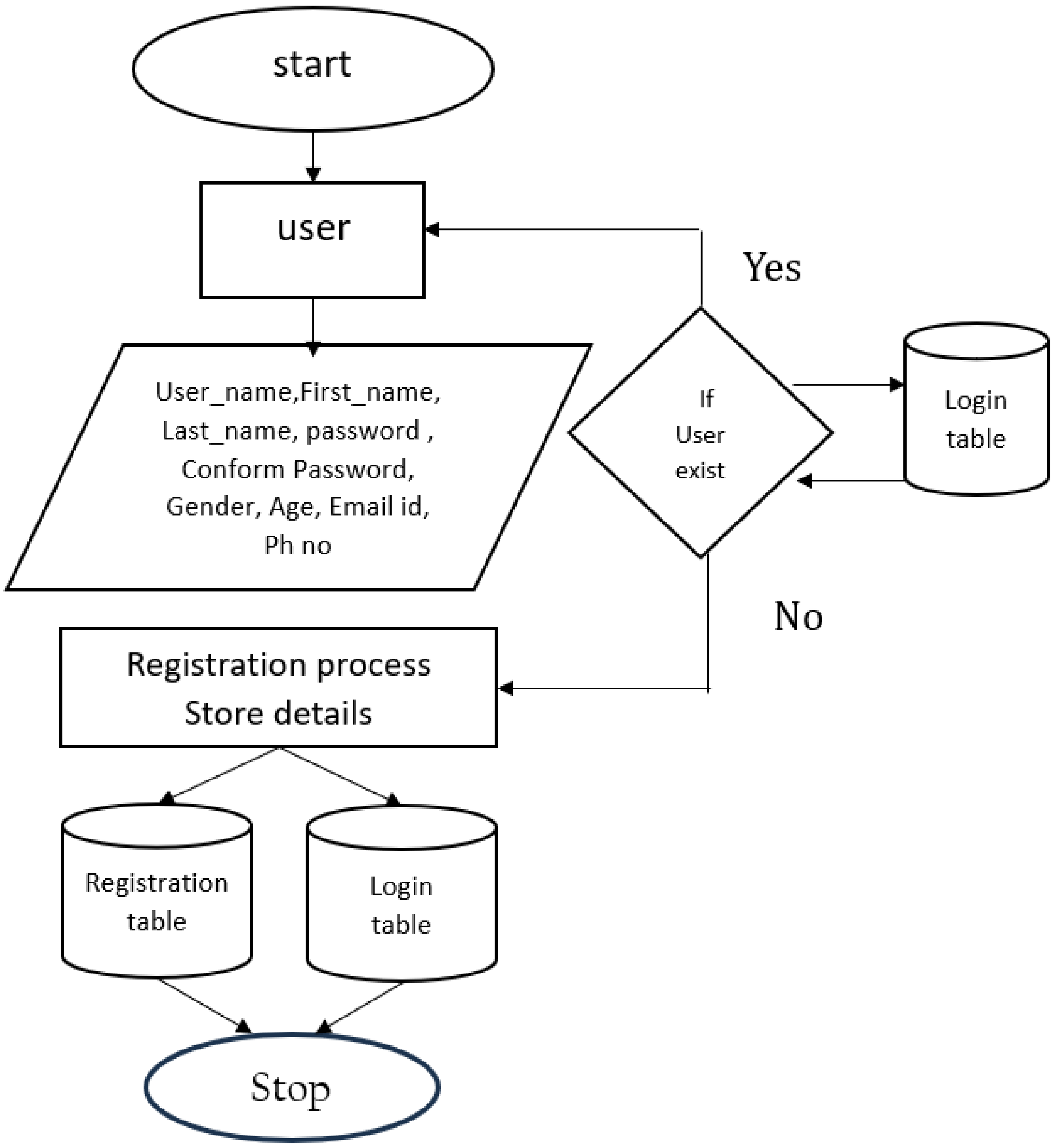
- A processing setup,usually called activity, and denoted as a rectangular box.
- A decision usually denoted as a diamond.

Symbols used in Flow Chart

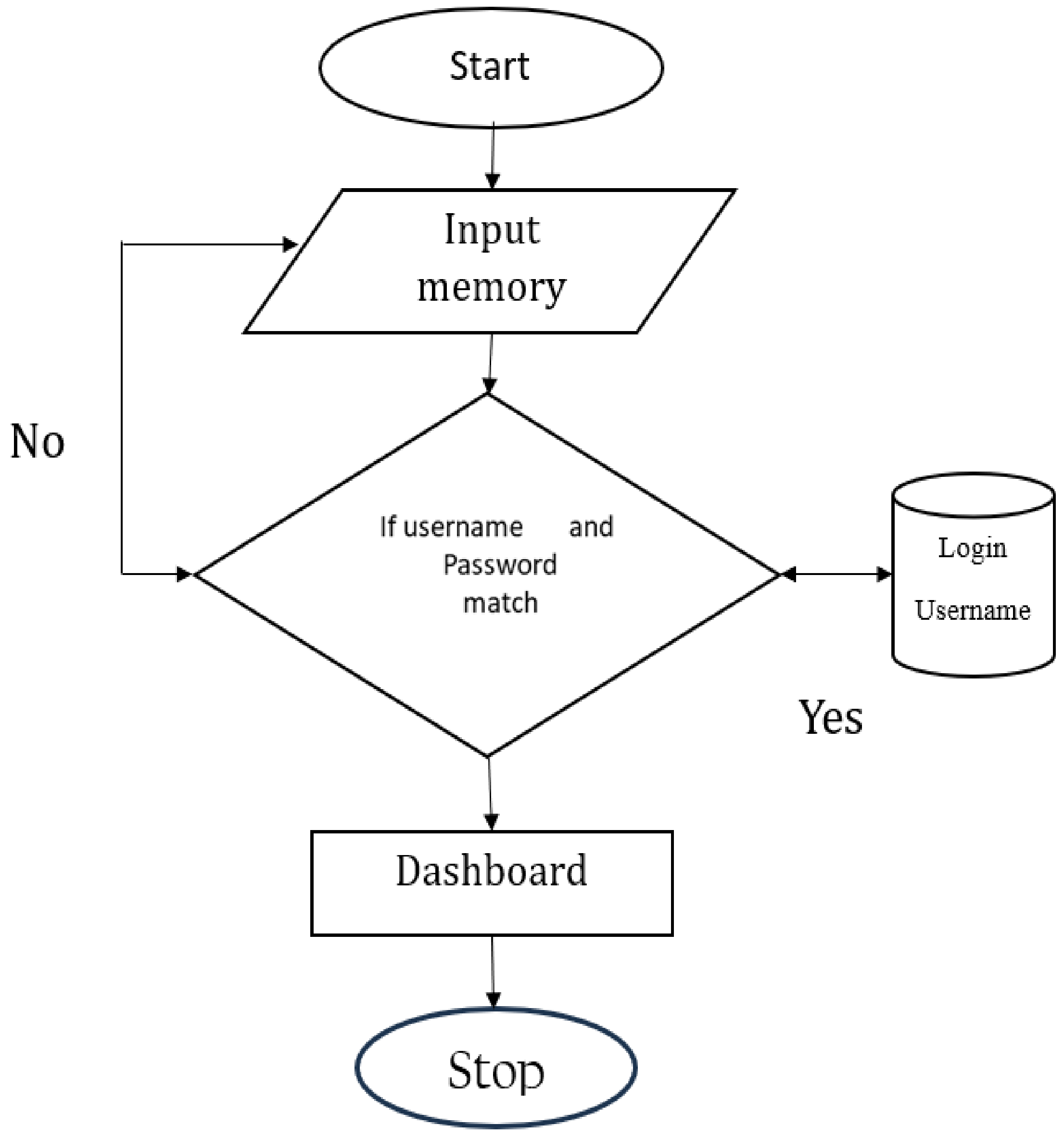
Symbol	Name	Function
	Start/End	An oval represent a start /end
	Arrow	A line is a connector that shows relationships between the representatives shapes
	Input/output	A parallelogram represents input/ouput
	Decision	A rhombus represents a decision
	Process	A rectangle represents a process

Flow chart of the system

Flow chart: Registration


start
user
User_name,First_name, Last_name, password , Conform Password, Gender, Age, Email id, Ph no
If User exist
Yes
Login table
No
Registration process Store details
Registration table
Login table
Stop

Flow chart: Login



3. SYSTEM DESIGN

3. SYSTEM DESIGN

The most creative and challenging phase of the system life cycle is the system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementing the candidate system. It also includes the construction of programs and program testing.

The first step in the system design is to determine how the output is to be produced and in what format. Samples of the output and the inputs are also presented. In the second step, input data and master files are to be designed to meet requirement of the proposed output. The processing phases are handled through program construction and testing, including a list of the programs needed to meet the system's objectives and complete documentation.

System design has two phases:

- Logical design
- Physical design

In the logical design, the designer produces a specification of the major features of the system which meets the objectives. The delivered product of logical design includes current requirements of the following system components:

- Input design.
- Output design.
- Database design.

Physical design takes this logical design blue print and produces the program software, files and a working system. Design specifications instruct programmers about what the system should do. The programmers in turn write the programs that accept input from users, process data, produce reports, and store data in files. Structured design is a data flow based methodology that partitions a program into a hierarchy of modules organized top-down manner with details.

SYSTEM ARCHITECTURE:

The system architecture is based on a client-server model where the client is a web browser and the server is a combination of a web server (Apache or Nginx) and a database server (PostgreSQL). The system is divided into three main layers:

Presentation Layer: This includes the user interface, built using HTML, CSS, and JavaScript. It handles user input and displays data received from the server.
Application Layer: The core logic is implemented in PHP and runs on the server. It processes user requests, interacts with the database, and returns responses to the client.
Data Layer: This consists of the PostgreSQL database, where all project, user, and media data are stored. The database is accessed via SQL queries from the application layer.

Database Schema: The database is designed with several interrelated tables:

Login_table: Stores login credentials.
registration_table: Contains user registration details.
Project_table: Stores information about user projects.
category_table: Contains categories for organizing projects.

MODULAR DESIGN:

Module 1: User Registration and Management

Purpose: Allow users to register, log in, and manage their profiles.
Components:

Registration Form
Login Form

Profile Management Interface

Account Settings Interface

Database Interaction:

Insert new users into the registration_table.

Validate user credentials in login_table.

Update user information in registration_table.

Module 2: Project Management

Purpose: Allow users to create, edit, delete, and view projects.

Components:

Project Creation Form

Project Editing Interface

Project Deletion Functionality

Project Viewing Interface

Database Interaction:

Insert new projects into the project_table.

Update and delete projects in project_table.

Retrieve project data for display.

Module 3: Media Upload

Purpose: Allow users to upload images, videos, and PDFs related to their projects.

Components:

Image Upload Interface

Video URL Submission Form

PDF Upload Interface

Database Interaction:

Store media file paths or URLs in project_table.

Link media to specific projects using proj_id.

Module 4: Step-by-Step Instructions

Purpose: Provide a structured way to document and share project instructions.

Components:

Instruction Creation Interface

Instruction Editing Interface

Instruction Viewing Interface

Database Interaction:

Store step-by-step instructions in project_table.

Link instructions to specific projects using proj_id.

Module 5: Category and Tagging

Purpose: Organize projects into categories and tags for easy discovery.

Components:

Category Selection Interface during Project Creation

Tagging Interface

Search and Filter Interface

Database Interaction:

Store category and tag information in category_table.

Link categories to projects in project_table.

3. Input Design

User Registration Form:

Fields: First Name, Last Name, Email, Date of Birth, Phone, Sex, Username, Password, Experience, Qualification.

Validation: Ensure all mandatory fields are filled, passwords meet security criteria, email format is correct.

Login Form:

Fields: Username, Password.

Validation: Check credentials against login_table and ensure correct login.

Project Creation Form:

Fields: Project Title, Category, Description, Image Upload, Video URL, PDF Upload.

Validation: Ensure the project title is unique, media files meet size and format requirements.

Profile Management Interface:

Fields: First Name, Last Name, Email, Phone, Sex, Profile Picture, Bio, Preferences.

Validation: Ensure correct data formats and allow updates.

4. Output Design

User Profile:

Details Displayed: Name, Email, Profile Picture, Bio, List of Projects.

Interactivity: Option to edit profile details, view project history.

Project Display:

Details Displayed: Project Title, Description, Category, Images, Videos, PDF Files, Creation Date.

Interactivity: Users can view detailed project instructions, download files, and leave comments.

Category Search and Filter Results:

Details Displayed: List of projects within the selected category, including title, brief description, and thumbnail image.

Interactivity: Users can click on a project to view its details.

Dashboard:

Details Displayed: Overview of user activity, including number of projects created, most recent project, and notifications.

Interactivity: Quick links to create new projects, edit existing ones, or manage account settings.

The design of the DIY Project Sharing Platform ensures a user-friendly experience while providing robust project management and sharing capabilities. By organizing the system into well-defined modules and focusing on secure and efficient data handling, the platform will facilitate easy project creation, management, and discovery for all users.

3.1 DATABASE DESIGN

Overview: The database design for the DIY Project Sharing Platform is centered around four primary tables: login_table, registration_table, project_table, and category_table. Each table is designed to store and manage specific aspects of the platform, ensuring data integrity, efficiency, and scalability.

1. login_table

Purpose: Stores the credentials of users for authentication purposes.

Attributes:

username (Primary Key)
user_password

Normalization:

1NF (First Normal Form): Each field contains atomic values, and the table structure is simple with no repeating groups.
2NF (Second Normal Form): The table is already in 1NF, and since there's only one candidate key (username), it satisfies 2NF as well.
3NF (Third Normal Form): There are no transitive dependencies. The user_password is directly dependent on username.

2. registration_table

Purpose: Stores the detailed registration information of users.

Attributes:

first_name
last_name
email
date_of_birth
phone
sex
username (Primary Key, Foreign Key to login_table)
user_password
experience
qualification

Normalization:

1NF (First Normal Form): All attributes are atomic, and there are no repeating groups or arrays.
2NF (Second Normal Form): The table is in 1NF, and every non-key attribute is fully functionally dependent on the primary key (username). There are no partial dependencies.
3NF (Third Normal Form): There are no transitive dependencies in the table. Each attribute is directly dependent on the primary key (username), ensuring that the table is in 3NF.

3. project_table

Purpose: Stores information about the projects uploaded by users.

Attributes:

proj_id (Primary Key)
project_title

category (Foreign Key to category_table)

description

status

upload_pdf

video_url

upload_date

image1

username (Foreign Key to registration_table)

Normalization:

1NF (First Normal Form): Each field is atomic, and there are no repeating groups within the table.

2NF (Second Normal Form): The table is in 1NF, and all non-key attributes are fully functionally dependent on the primary key (proj_id). There are no partial dependencies.

3NF (Third Normal Form): The table is in 2NF, and there are no transitive dependencies. Each attribute is either a primary key or is fully dependent on the primary key (proj_id).

4. category_table

Purpose: Stores the categories that organize the projects.

Attributes:

category_id (Primary Key)

category_name

Normalization:

1NF (First Normal Form): The table has a simple structure, with each field containing atomic values.

2NF (Second Normal Form): Since there is only one candidate key (category_id), and the table is in 1NF, it is automatically in 2NF.

3NF (Third Normal Form): There are no transitive dependencies. category_name is fully functionally dependent on the primary key (category_id).

Summary of Normalization:

1NF: All tables ensure that the values in each column are atomic, meaning they are indivisible.

2NF: All tables eliminate partial dependencies by ensuring that every non-key attribute is fully dependent on the primary key.

3NF: All tables eliminate transitive dependencies, meaning no non-key attribute depends on another non-key attribute.

This normalization process ensures that the database is well-structured, reduces redundancy, and maintains data integrity across all tables. By following these principles, the system is optimized for efficient querying and updates, while also being scalable for future expansions.

TABLES IN THE SYSTEM

Tables in the system

Table Name: login_table

SL.NO	FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS	CRITERIA
1	username	varchar	20	Not null	mandatory
2	user_password	varchar	20	Not null	mandatory

Description-To store the user login details

Tables in the system

Table Name: admin_login_table

Description-To store the admin login details

SL.NO	FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS	CRITERIA
1	Username	Varchar	20	Not null	mandatory
2	Password1	Varchar	20	Not null	mandatory

Tables in the system

Table Name: registration_table

Description-To store the Registration details

SL.NO	FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS	CRITERIA
1	first_name	text	10	Not null	mandatory
2	last_name	text	10	Not null	mandatory
3	email	text	30	Not null	mandatory
4	dob	text		Not null	mandatory
5	phone	numeric	10	Not null	mandatory
6	sex	text	5	Not null	mandatory
7	username	varchar	10	Primary key	mandatory
8	user_password	text	8	Not null	mandatory
9	experience	text	100	Not null	mandatory
10	qualification	text	30	Not null	mandatory

Tables in the system

Table Name: project_table

Description-To store the Project details

SL.NO	FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS	CRITERIA
1	Proj_id	bigint	20	Primary key	mandatory
2	project_title	Char	100	Not null	mandatory
3	category	Char	100	Not null	mandatory
4	description	Varchar	300	Not null	mandatory
5	status	Char	10	Not null	mandatory
6	Upload_pdf	bytea		Not null	mandatory
7	Video_url	varchar	100	Not null	optional
8	Upload_date	date		Not null	mandatory
9	image1	bytea		Not null	optional
10	username	varchar	10	Not null	Optional

Tables in the system

Table Name: category_table

Description-To store the Category type

SL.NO	COLUMN	DATATYPE	SIZE	CONSTRAINTS	CRITERIA
1	category_id	int	10	Primary key	mandatory
2	category_type	text	30	Not null	mandatory

3.2 ARCHITECTURAL DESIGN

The Architectural Design process focuses on the decomposition of a system into different components and their interactions to satisfy functional and non-functional requirements. The basic architecture design process is composed of the following steps:

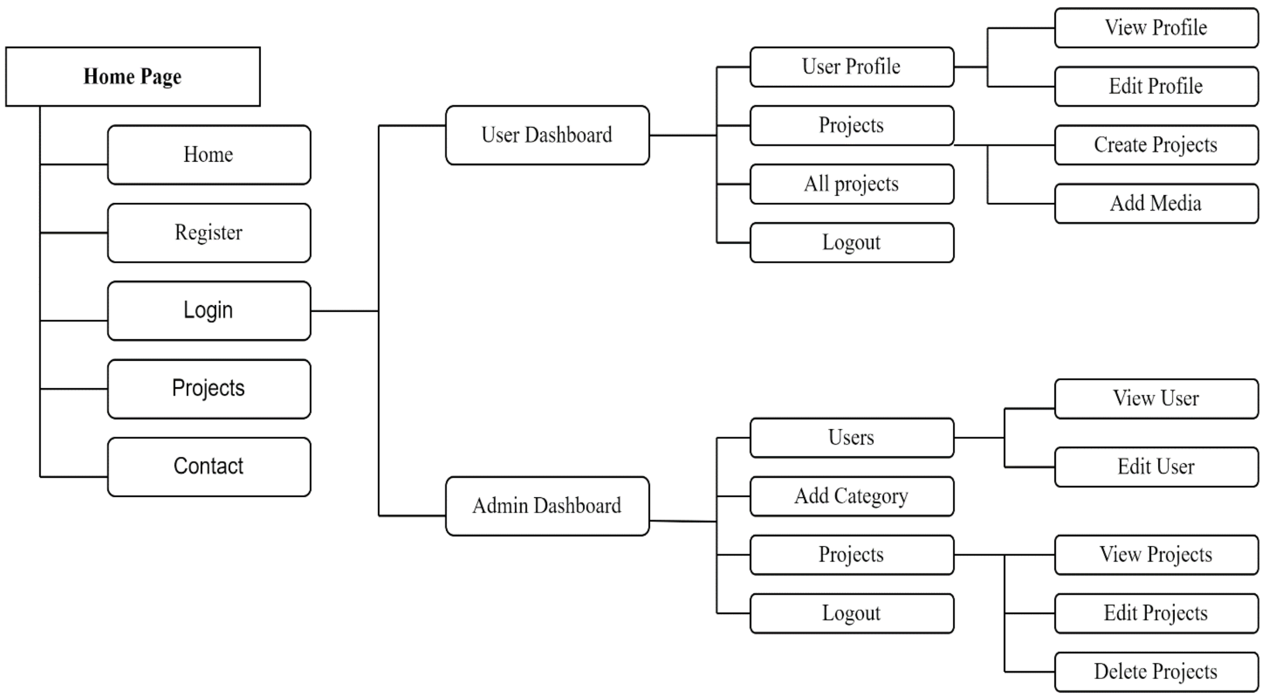
• Identify design elements and their relationships

• Evaluate the Architecture Design

• Transform the Architecture Design

In this project, component based, layered, object-oriented architecture is used. In component, breakdown the application design into reusable functional or logical components that expose well defined communication interfaces. In layered structure divide the concerns into stacked groups into stacked groups (layers). In object oriented based on the division of responsibilities of an application or a system into objects, each containing the data and the behavior relevant to the object.

Menu Tree of the system



3.3 PROCEDURAL DESIGN

Procedural design is best used to model programs that have an obvious flow of data from input to output. It represents the architecture of a program as a set of interacting processes that data pass data from one another. The steps involved in this are:

• Computer procedure: Specify what functions will be carried out on computer, what will be different programs and in what sequence program will be run.

• Non-computer procedure: Specify the manual procedures for feeding input data, receiving outputs etc.

3.4 INTERFACE DESIGN

The purpose of interface design is to communicate effectively through form designs: there are several major requirements:

• The form title must clearly identify its purpose. Columns and rows should be labeled to avoid confusion. The form should be identified by firm name or code number to make it easy to work with.

• The form must be easy to use and fill out. It should be legible, intelligible and uncomplicated.

• The data requested should reflect a logical sequence.

3.5 INPUT DESIGN

The input design is the process of converting the user-oriented inputs in to the computer-based format. The goal of designing input data is to make automation as easy and free from errors as possible. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of the project.
The following points should be considered while designing the input:

- What data to input?
- What medium to use?
- How the data should be arranged or coded?

Inaccurate input data is the most common cause of error in processing data. Errors entered by the data entry operators can be controlled by the input design. The arrangement of messages as well as placement of data, headings and titles on display screens or source document is also a part of input design. The design of input also includes specifying the means by which end user and system operators direct the system what action to take.

3.6 OUTPUT DESIGN

Output generally refers to the results and information that are generated by the system. When output, system analyst must accomplish the following:

Determine what information to present.

- Decide whether to display, print the information and select the output medium.
- Arrange the presentation of information in an acceptable format.
- Decide how to distribute the output to intended recipients.

4. SYSTEM CODING

SYSTEM CODING

The coding step is it process that transforms design into a programming language. It translates a details representation of software into a programming language realization. The translation process continues when a compiler accepts source code as input as produces machine dependent object codes output. Quality of source code can be improved by the use of structures coding techniques; good coding style and readable, consistent code format. During coding, some coding standards are to be allowed. This has to purpose; reducing the chance of making it easier for some time to modify the code later on. Coding phase affects both testing and maintenance profoundly.

In order to begin with the coding section, we must first create the default Flutter project after installing the required technologies and dependencies. So, we have to open terminal to type:

```
Username@computer:~$ flutter create attenz
```

Thus we create a sample project and now we have to add a new folder assets inside it where we add our required fonts, lotties and images to be used in the application. We also creates folders screens and databases to work effectively inside the lib folder.

registration.php

```
<?php
$host = "localhost";
$port = "5432";
$dbname = "DIY_project_sharing_platform";
$user = "postgres";
$password = "1978";

// Connect to PostgreSQL database

$conn = pg_connect("host=$host port=$port dbname=$dbname user=$user
password=$password");

if ($conn)
{
```

```
echo "Connected to the database successfully!";
} else
{
echo "Failed to connect to the database.";
}

if($_SERVER["REQUEST_METHOD"]=="POST")
{
   //RETRIEVE VALUES
   $first_name=$_POST["first_name"];
   $last_name=$_POST["last_name"];
   $email=$_POST["email"];
   $DOB=$_POST["DOB"];
   $phone=$_POST["phone"];
   $sex=$_POST["sex"];
   $username=$_POST["username"];
   $password=$_POST["user_password"];
   $qualification=isset($_POST["qualification"]) ? implode(",", $_POST["qualification"]) :"";
   $experience=$_POST["experience"];

}
// SQL statement for inserting data into the database
$sql = "INSERT INTO
registration_table(first_name,last_name,email,date_of_birth,phone,sex,username,user_passw
ord,qualification,experience)
VALUES ('$first_name', '$last_name', '$email', '$DOB', '$phone', '$sex', '$username',
'$password','$qualification','$experience')";
$sql2 = "INSERT INTO login_table(username,user_password)
VALUES ('$username', '$password')";

$login = "http://localhost/DIY_Project_sharing_platform/HTML/Login_page.html";-
```

```
// Execute SQL query
$result = pg_query($conn, $sql);
$result = pg_query($conn, $sql2);

if ($result)
{
echo "Data inserted successfully!";
header("Location: $login");
}
else
{
echo "Error: " . pg_last_error($conn);
}

pg_close($conn);
?>
```

5.SYSTEM TESTING

5. SYSTEM TESTING

5.1 TESTING

For the DIY Project Sharing Platform, comprehensive testing is essential to ensure the application meets functional, security, and performance requirements. Below is an outline of testing techniques, methods, and test cases tailored to this project.

TESTING TECHNIQUES

5.2 Unit Testing:

Focuses on individual components or functions, such as user registration, project creation, and media upload features.

Ensures that each function works correctly in isolation.

5.3 Integration Testing:

Tests the interactions between different modules, such as how the login module interacts with the user profile management module.

Verifies that data flows correctly between modules and that integrated modules work as intended.

5.4 System Testing:

Involves testing the entire system as a whole to ensure that all components work together seamlessly.

Validates the complete functionality of the DIY project sharing platform, including user registration, project management, and category tagging.

5.5 Acceptance Testing:

Ensures that the application meets the specified business requirements.

This could be done by simulating real-world scenarios to check if the application satisfies user needs.

Performance Testing:

Assesses the application's responsiveness, speed, and stability under various load conditions.

Includes stress testing to determine the breaking point of the application and load testing to see how the system performs with multiple concurrent users.

TESTING METHODS

Black Box Testing:

Focuses on testing the external functionalities of the application without considering the internal code structure.

Used for functional testing, such as verifying user registration, login, and project creation processes.

White Box Testing:

Involves testing the internal code structure, logic, and paths.

Used primarily in unit testing to ensure that each function or method works as intended.

Manual Testing:

Testers execute test cases manually to ensure that the application behaves as expected.

Useful for exploratory testing, usability testing, and testing the UI.

Automated Testing:

Utilizes automated scripts to perform repetitive testing tasks, such as regression testing.

Tools like Selenium or JUnit can be used for automating functional and regression tests.

TEST CASES

User Registration Module

Test Case 1: Valid User Registration

Objective: Verify that a user can register successfully with valid inputs.

Steps:

Navigate to the registration page.
Fill in valid data for all fields (first name, last name, email, etc.).
Submit the registration form.

Expected Result: User is successfully registered, and a confirmation email is sent.

Test Case 2: Registration with Invalid Email Format

Objective: Ensure that the system validates the email format during registration.

Steps:

Enter an invalid email address.
Attempt to submit the registration form.

Expected Result: The system should display an error message indicating an invalid email format.

Login Module

Test Case 1: Valid Login

Objective: Verify that a user can log in with valid credentials.

Steps:

Enter valid username and password.

Click the login button.

Expected Result: User is redirected to the dashboard or home page.

Test Case 2: Invalid Login Attempt

Objective: Verify that the system prevents login with incorrect credentials.

Steps:

Enter an invalid username or password.

Attempt to log in.

Expected Result: The system should display an error message and not allow access.

Project Management Module

Test Case 1: Create New Project

Objective: Ensure that a user can create a new project successfully.

Steps:

Navigate to the project creation page.

Enter project details (title, description, category, etc.).

Submit the project form.

Expected Result: The project is successfully created and appears in the user's project list.

Test Case 2: Edit Existing Project

Objective: Verify that a user can edit an existing project.

Steps:

Navigate to the project list.

Select a project to edit.

Make changes to the project details.

Save the changes.

Expected Result: The project details are updated successfully.

Media Upload Module

Test Case 1: Upload Project Image

Objective: Ensure that a user can upload an image to a project.

Steps:

Create or edit a project.

Upload an image file in the specified format (JPEG, PNG, etc.).

Save the project.

Expected Result: The image is successfully uploaded and displayed in the project details.

Test Case 2: Upload Project Video URL

Objective: Verify that a user can upload a video URL for a project.

Steps:

Create or edit a project.

Enter a valid video URL (e.g., from YouTube).

Save the project.

Expected Result: The video is linked to the project and can be viewed by other users.

Category and Tagging Module

Test Case 1: Add Project Category

Objective: Ensure that a user can select and add a category to their project.

Steps:

Navigate to the project creation/editing page.

Select a category from the dropdown list.

Save the project.

Expected Result: The selected category is saved and associated with the project.

Test Case 2: Search Projects by Tag

Objective: Verify that users can search for projects using tags.

Steps:

Enter a keyword in the search bar.

Search for projects based on the tag.

Expected Result: Projects matching the tag keyword are displayed.

The testing strategy for the DIY Project Sharing Platform involves a combination of various techniques and methods to ensure the application is reliable, secure, and user-friendly. Each module of the application is rigorously tested through predefined test cases to identify and fix any potential issues before deployment.

6. SYSTEM IMPLEMENTATION

6.1 SYSTEM IMPLEMENTATION AND MAINTENANCE

The implementation of the DIY Project Sharing Platform involves several stages, including setting up the development environment, building the backend and frontend, integrating the database, implementing security measures, and testing the application. Below is a detailed description of each phase.

Development Environment Setup

Technologies Used:

Frontend: HTML, CSS, JavaScript, Bootstrap for responsive design.

Backend: PHP for server-side processing.

Database: PostgreSQL for data storage.

Development Tools: XAMPP for local server setup, Sublime Text/VSCode for coding, and GitHub for version control.

Environment Configuration:

Install XAMPP and configure it to run Apache and PostgreSQL.

Set up a version control repository using GitHub.

Configure the local environment to connect to the PostgreSQL database.

DATABASE DESIGN AND INTEGRATION

Database Schema:

Tables Created:

login_table: Stores user login credentials.

registration_table: Stores detailed user information.

project_table: Stores project-related information including media and category details.

category_table: Stores project categories.

Database Connection:

Establish a connection between the PHP backend and PostgreSQL using PHP s PDO (PHP Data Objects) for secure and efficient database interaction.

Implement prepared statements to avoid SQL injection.

Normalization:

Ensure that the database is normalized up to the 3rd Normal Form (3NF) to eliminate redundancy and ensure data integrity.

Backend Development

User Registration and Login:

Implement user registration with input validation, secure password hashing using bcrypt, and email verification.

Develop the login functionality with session management for secure user authentication.

Project Management:

Implement CRUD (Create, Read, Update, Delete) operations for projects.

Include functionality for users to upload media (images and videos) associated with their projects.

Store media files securely and link them to corresponding projects in the database.

Category and Tagging System:

Implement a system where users can select or create categories for their projects.

Use a tagging system to improve searchability, allowing users to filter projects based on categories and tags.

Frontend Development

User Interface (UI):

Design the UI using HTML, CSS, and Bootstrap to ensure a responsive and user-friendly experience.

Create forms for registration, login, project creation, and media uploads.

Project Display and Navigation:

Develop pages to display projects with filtering options based on categories and tags.

Implement pagination for projects to enhance navigation.

User Profile Management:

Create a profile page where users can view and update their personal information, manage their projects, and change their account settings.

SECURITY IMPLEMENTATION

User Authentication:

Use session tokens to manage user sessions securely.

Implement CAPTCHA for user registration to prevent bots.

Input Validation:

Validate all user inputs both on the client side (using JavaScript) and server side (using PHP) to prevent XSS and other injection attacks.

File Upload Security:

Sanitize file names and validate file types before uploading to prevent malicious files from being uploaded.

Store media files outside the webroot to protect them from direct access.

TESTING AND DEBUGGING

Unit Testing:

Write test cases for individual components such as user registration, login, and project creation.

Integration Testing:

Test the interaction between different modules to ensure they work together seamlessly.

User Acceptance Testing (UAT):

Conduct testing with a small group of users to gather feedback on the application s usability and functionality.

Debugging:

Use debugging tools and logs to identify and fix any issues that arise during testing.

MAINTENANCE AND UPDATES

Regular Backups:

Schedule regular backups of the database and media files to prevent data loss.

Feature Updates:

Plan for periodic updates to add new features, improve performance, and enhance security based on user feedback.

Bug Fixes:

Monitor for bugs or vulnerabilities and apply patches as needed to maintain application integrity and security.

The DIY Project Sharing Platform s implementation requires careful planning and execution across multiple stages, including development, security, testing, and deployment. By following the detailed steps outlined above, the platform will be robust, secure, and user-friendly, providing a seamless experience for users to share and discover DIY projects.

MAINTENANCE

It is possible to produce systems of any size which do not need to be changed. Over the lifetime of a system, its original requirements will be modified to reflect the changing user. After implementation, maintenance is the important process. Usually once the system is implemented, the software developers and customer would sign a contract. According to the time mentioned in the contract all errors and requirements would be done free of cost. Once the maintenance period is over all the logical errors will be corrected free of cost were as all extra requirements would be charged. During the contract period we would frequently visit the site where the system is implemented and check the system performance such as response time and also how it works at peak hours. If any problem is found it is corrected. Software development does not freeze at the moment of delivery. Usually, software must grow and change over time.

These activities are collectively referred to as software maintenance. Application updates are part of normal maintenance phase of development life cycle. A modification effort is actually a small project and must proceed through all the phases of development process. There are many reasons for software modification and continued development after the first release. Application may need additional features not discovered during the original analysis and design. Ease if maintenance is a part of every step in development. If the analysis is complete, users will find the most important features in the first release of software.

If the design and coding is done perfectly, then it will be very easy to maintain later. Some of the suggestions are group the changes and deliver another release rather than incremental changes, so that it will force to be more through about new researches. Give more to small requirements. The experience in coding will be an added advantage because integrating a new code with existing code is normally difficult. The working of the system is observed in a local machine and intranet and it was found satisfactory.
The four types of maintenance activities are listed below:

Corrective Maintenance

This is concerned with fixing reported errors in the software. Coding errors are cheap to correct; design errors are more expensive as they may involve the rewriting of several program components. Requirements errors are the most expensive to repair because of the extensive system redesign which may be necessary.

Adaptive Maintenance

This means changing the software to some new environment such as different hardware platform or for use with different operating system. The software functionality does not radically change. Any system that involves JVM can run this software.

Perfective Maintenance

This involves implementing a new functional or non functional system requirement. These are generated by software consumer as their organization or business changes.

Preventive Maintenance

This occurs when software is changed to improve future maintainability or reliability or to provide a better basis for future enhancements.

In the current project, all the above maintenance was implemented.

6.2 FUTURE ENHANCEMENT

The DIY Project Sharing Platform has been designed with scalability and future growth in mind. While the current implementation provides a solid foundation, several enhancements can be considered to improve the platform's functionality and user experience:

Advanced Search and Filtering:

Feature: Implement an advanced search functionality with multi-faceted filters based on project type, difficulty level, materials used, and user ratings.

Benefit: This will allow users to find specific projects more easily, enhancing the platform's usability.

Social Media Integration:

Feature: Integrate social media sharing options so users can share their projects directly to platforms like Facebook, Twitter, and Instagram.

Benefit: This would increase the visibility of projects and attract more users to the platform.

Mobile Application:

Feature: Develop a mobile application for iOS and Android to complement the web platform.

Benefit: A mobile app would provide users with greater accessibility and convenience, encouraging more frequent interaction with the platform.

User Collaboration Features:

Feature: Introduce collaborative project creation where multiple users can contribute to a single project.

Benefit: This would foster community engagement and allow for the creation of more complex and diverse projects.

Enhanced User Profiles:

Feature: Allow users to showcase badges, achievements, and a portfolio of their best projects on their profiles.

Benefit: This would add a gamification element to the platform, motivating users to be more active and contributing to user retention.

AI-Powered Recommendations:

Feature: Implement AI algorithms to recommend projects to users based on their past interactions, preferences, and browsing history.

Benefit: Personalized recommendations would improve user engagement and satisfaction by providing content that is relevant to their interests.

Monetization Options:

Feature: Introduce monetization features, such as premium accounts or the ability for users to sell their project plans and tutorials.

Benefit: This would provide users with an incentive to create high-quality content and could generate additional revenue for the platform.

Internationalization and Localization:

Feature: Add multi-language support and localization features to cater to a global audience.

Benefit: This would broaden the platform's reach and make it accessible to users worldwide.

7.CONCLUSION

7. CONCLUSION

The DIY Project Sharing Platform is designed to be a comprehensive and user-friendly tool for DIY enthusiasts to create, share, and discover projects. The current implementation provides a robust foundation with essential features such as user registration, project management, media uploads, and a category-based organization system.

By focusing on future enhancements like advanced search capabilities, social media integration, mobile app development, and AI-powered recommendations, the platform can evolve to meet the growing demands of its user base. These improvements will not only enhance user experience but also ensure the platform remains competitive in the ever-evolving digital landscape.

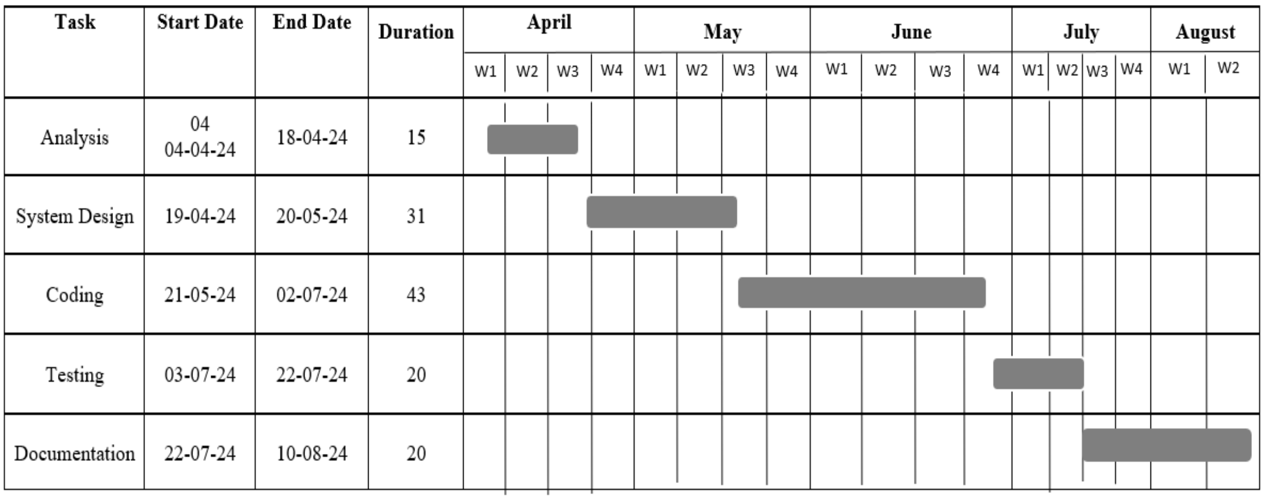
In conclusion, the DIY Project Sharing Platform has the potential to become a leading online community for DIY enthusiasts by continuously adapting to technological advancements and user needs. With careful planning, regular updates, and a focus on user satisfaction, the platform can achieve long-term success and widespread adoption.

8.APPENDIX

8 APPENDIX

8.1 GANTT CHART

The risk and uncertainty raises multirole with respect to the size of the project, even when the project is developed according to set methodologies. There are tools available which aid for effective project management. A few are described using Gantt chart. It was devised by Henry Gantt (1971). It represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities. Tasks that can be completed in parallel:



Task	Start Date	End Date	Duration	April				May				June				July				August	
				W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2
Analysis	04 04-04-24	18-04-24	15	▨	▨	▨															
System Design	19-04-24	20-05-24	31				▨	▨	▨	▨											
Coding	21-05-24	02-07-24	43							▨	▨	▨	▨	▨	▨						
Testing	03-07-24	22-07-24	20												▨	▨	▨				
Documentation	22-07-24	10-08-24	20															▨	▨	▨	▨

8.2 MEETING MINUTES

Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

MINUTES

Date : 04/04/2024

Time : 10.00 AM

Location : IOTIDES PVT LTD, KOLLAM

Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

Our project topic is DIY PROJECT SHARING PLATFORM. And we choose language PHP for doing the project. The language system study and analysis are done in between 01/04/2024 to 07/08/2024

MINUTES

Date : 22/04/2024

Time : 10.00 AM

Location : IOTIDES PVT LTD, KOLLAM

Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

We started design process. The first step is to form design. Many forms are included in project.

MINUTES

Date : 17/05/2024

Time : 10.00 AM

Location : IOTIDES PVT LTD, KOLLAM

Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

Since the form design is over, we decided to do the next step of the project, table creation and design. PHP language is used for coding and for database design MYSQL is used. Also we draw the data flow diagram neatly.

MINUTES

Date :02/07/2024

Time : 10.00 AM

Location : IOTIDES PVT LTD, KOLLAM

Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

About 80% of code design is completed, we started testing process. It can take a long time period to complete the testing process. So we started documentation process.

MINUTES

Date : 15/07/2024

Time : 10.00 AM

Location : IOTIDES PVT LTD, KOLLAM

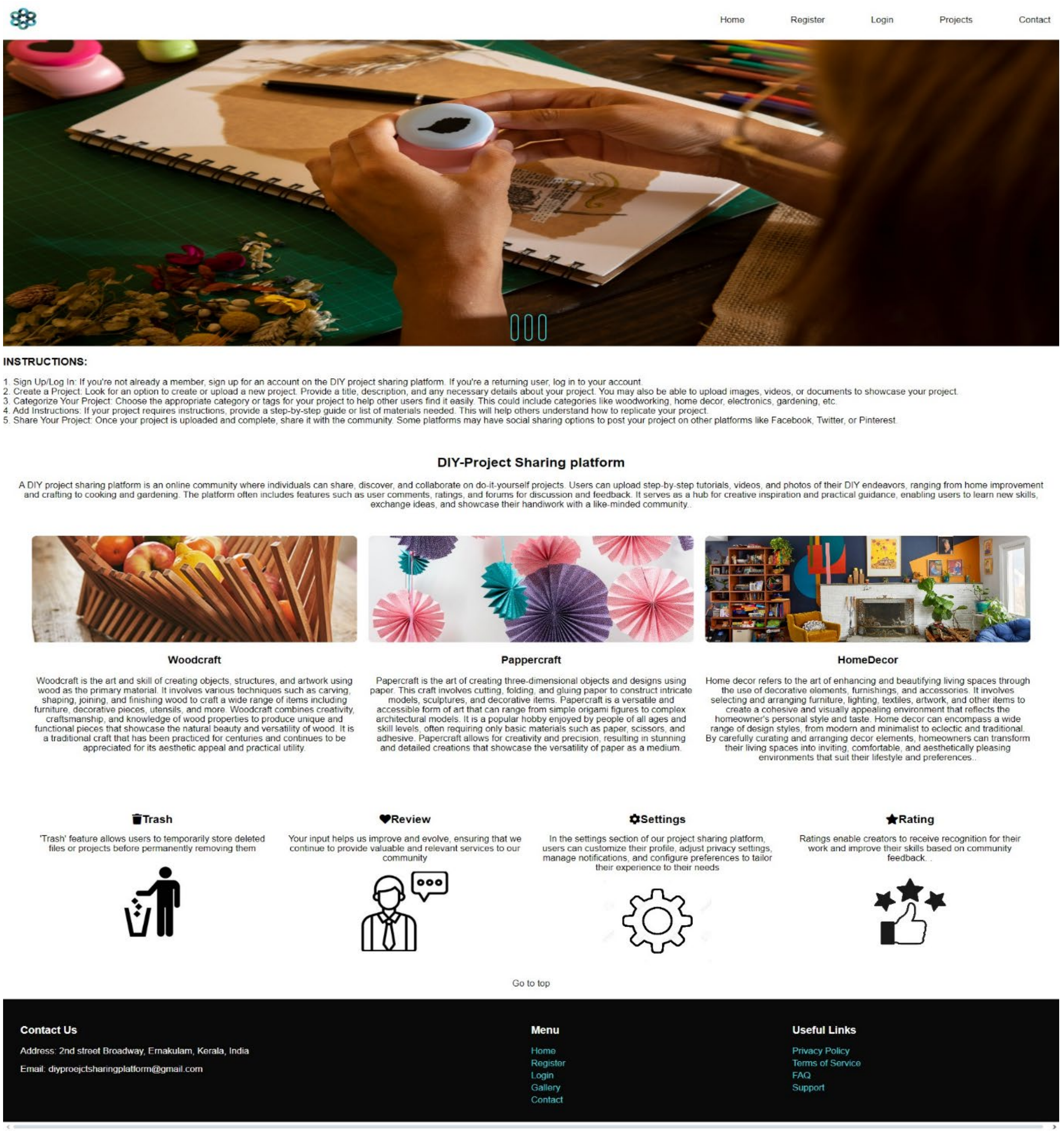
Group Members

AFSAL SHA	(Reg.No:33222959009)
ANANDHU AV	(Reg.No:33222959016)
MUHAMMAD ASLAM	(Reg.No:33222959033)
FIGO KRISHNA	(Reg.No:33222959025)

We completed the documentation on 30/03/2024

8.3 SCREEN LAYOUTS AND REPORTS

Home page



INSTRUCTIONS:

1. Sign Up/Log In: If you're not already a member, sign up for an account on the DIY project sharing platform. If you're a returning user, log in to your account.
2. Create a Project: Look for an option to create or upload a new project. Provide a title, description, and any necessary details about your project. You may also be able to upload images, videos, or documents to showcase your project.
3. Categorize Your Project: Choose the appropriate category or tags for your project to help other users find it easily. This could include categories like woodworking, home decor, electronics, gardening, etc.
4. Add Instructions: If your project requires instructions, provide a step-by-step guide or list of materials needed. This will help others understand how to replicate your project.
5. Share Your Project: Once your project is uploaded and complete, share it with the community. Some platforms may have social sharing options to post your project on other platforms like Facebook, Twitter, or Pinterest.

DIY-Project Sharing platform

A DIY project sharing platform is an online community where individuals can share, discover, and collaborate on do-it-yourself projects. Users can upload step-by-step tutorials, videos, and photos of their DIY endeavors, ranging from home improvement and crafting to cooking and gardening. The platform often includes features such as user comments, ratings, and forums for discussion and feedback. It serves as a hub for creative inspiration and practical guidance, enabling users to learn new skills, exchange ideas, and showcase their handiwork with a like-minded community..

Woodcraft

Woodcraft is the art and skill of creating objects, structures, and artwork using wood as the primary material. It involves various techniques such as carving, shaping, joining, and finishing wood to craft a wide range of items including furniture, decorative pieces, utensils, and more. Woodcraft combines creativity, craftsmanship, and knowledge of wood properties to produce unique and functional pieces that showcase the natural beauty and versatility of wood. It is a traditional craft that has been practiced for centuries and continues to be appreciated for its aesthetic appeal and practical utility.

Pappercraft

Papercraft is the art of creating three-dimensional objects and designs using paper. This craft involves cutting, folding, and gluing paper to construct intricate models, sculptures, and decorative items. Papercraft is a versatile and accessible form of art that can range from simple origami figures to complex architectural models. It is a popular hobby enjoyed by people of all ages and skill levels, often requiring only basic materials such as paper, scissors, and adhesive. Papercraft allows for creativity and precision, resulting in stunning and detailed creations that showcase the versatility of paper as a medium.

HomeDecor

Home decor refers to the art of enhancing and beautifying living spaces through the use of decorative elements, furnishings, and accessories. It involves selecting and arranging furniture, lighting, textiles, artwork, and other items to create a cohesive and visually appealing environment that reflects the homeowner's personal style and taste. Home decor can encompass a wide range of design styles, from modern and minimalist to eclectic and traditional. By carefully curating and arranging decor elements, homeowners can transform their living spaces into inviting, comfortable, and aesthetically pleasing environments that suit their lifestyle and preferences..

Trash

'Trash' feature allows users to temporarily store deleted files or projects before permanently removing them

Review

Your input helps us improve and evolve, ensuring that we continue to provide valuable and relevant services to our community

Settings

In the settings section of our project sharing platform, users can customize their profile, adjust privacy settings, manage notifications, and configure preferences to tailor their experience to their needs

Rating

Ratings enable creators to receive recognition for their work and improve their skills based on community feedback. .

Go to top

Contact Us

Address: 2nd street Broadway, Ernakulam, Kerala, India

Email: diyproejctsharingplatform@gmail.com

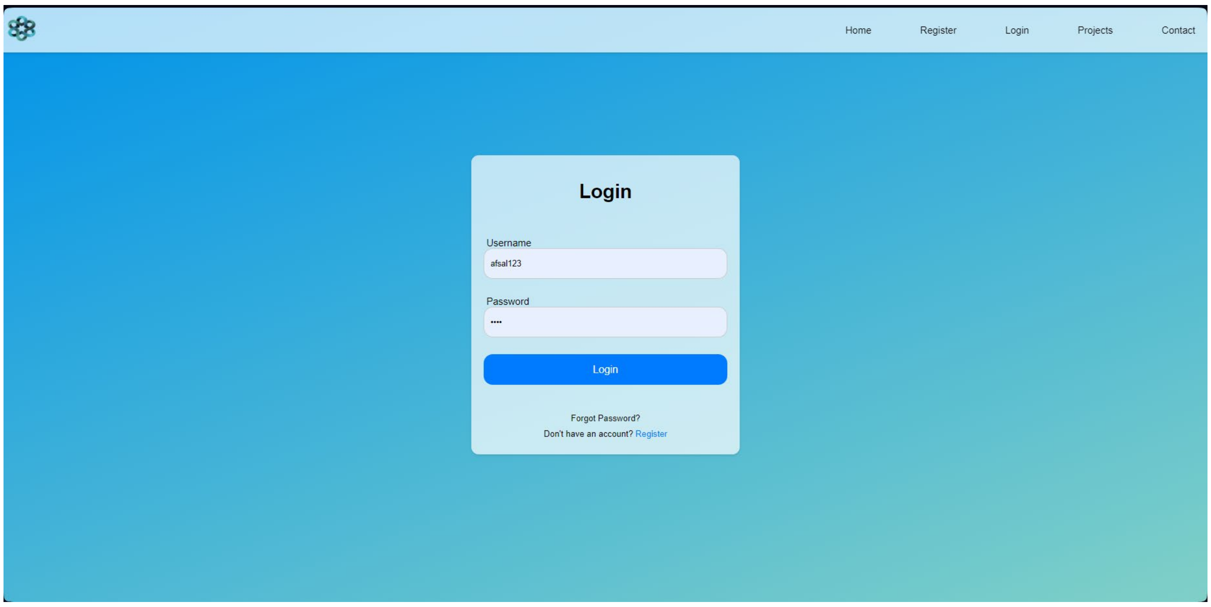
Menu

Home
Register
Login
Gallery
Contact

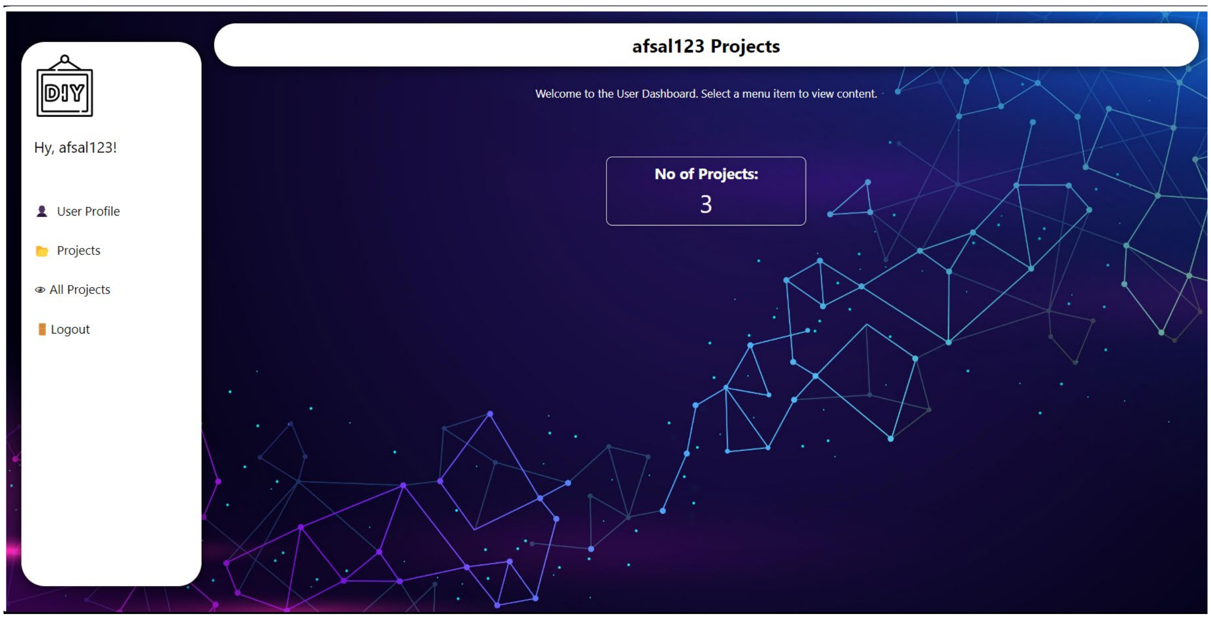
Useful Links

Privacy Policy
Terms of Service
FAQ
Support

Login Page


Home
Register
Login
Projects
Contact
Login
Username
afsal123
Password
••••
Login
Forgot Password?
Don't have an account? Register

User Dashboard

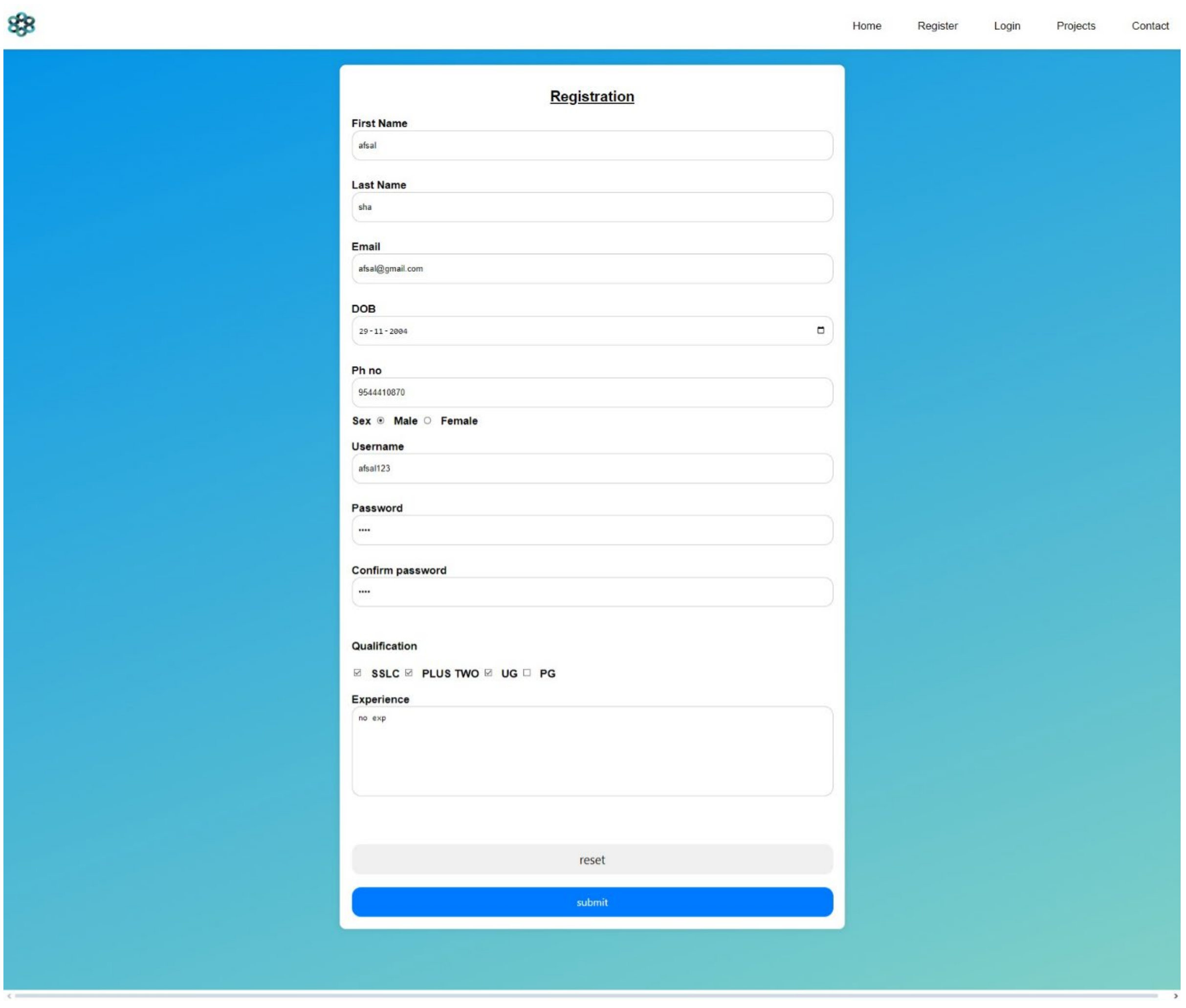


afsal123 Projects

Welcome to the User Dashboard. Select a menu item to view content.

No of Projects:
3

Registration



Registration

First Name: afsal

Last Name: sha

Email: afsal@gmail.com

DOB: 29-11-2004

Ph no: 9544410870

Sex [✓ Male] Female

Username: afsal123

Password: ····

Confirm password: ····

Qualification: [✓ SSLC] [✓ PLUS TWO] [✓ UG] PG

Experience: no exp

9. BIBLIOGRAPHY

9. BIBLIOGRAPHY

Text Book Reference:

- Rajib Mall: Fundamentals of Software Engineering, PHI Learning Private Limited, Third Edition,2009
- Elmasri, Navathe: Fundamentals of Database Systems, Pearson Education, Third Edition, Third Edition, 2000
- Ramakrishnan, Gehre: Database Management Systems, McGraw-Hill Higher Education, Third Edition, 2003
- Pankaj Jalote: An Integrated Approach to Software Engineering Home PVT LTD 2009
- Steven Holzner: The Complete Reference PHP, McGraw-Hill ,2008

Website Reference:

- http://notesofgenius.com/mysql-workbench/
- http://www.roseindia.net/mysql/mysql5/what-is-mysql.shtml