**Believe in your Project, But never stop improving.**

The project has been submitted to the Computer Science & Engineering Department of the University of Development Alternative (UODA) for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering.

**Submitted by**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | Batch | Semester |
| Md. Zahidul Islam | 011181080 | 52 | 12 |
| Md Mehedi Hasan | 011193012 | 52 | 12 |
| Bijoy Karmokar | 011163015 | 52 | 12 |

**Under The Supervision Of**

**Md. Tareq Hasan**

Assistant professor &

Project Supervisor

Department Of Computer Science & Engineering

University Of Development Alternative (UODA)

A picture containing text

Description automatically generated

**University Of Development Alternative (UODA)**

Date Of Submission: 10/09/2023

**DECLARATION**

To certify that this project is our core work and the results abstracted in this thesis have not been submitted to any other university or institute for awarding a degree. To our best knowledge and belief, the thesis contains no material previously published or written by another person where appropriate references are made in the text. Reproduction of any material in this project has been correctly recognized. We strongly recommend that the completion of each part of this project is the result of our own efforts and hard work.

|  |  |
| --- | --- |
| **Md. Zahidul Islam** | **Md Mehedi Hasan** |
| **Student ID: 011181080** | **Student ID: 011193012** |

|  |
| --- |
| **Bijoy Karmokar** |
| **Student ID: 011163015** |

**APPROVAL**

This project entitled "The Project Hub" has been submitted to the following respected members of the board of examiners, Department of Computer Science & Engineering in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science & Engineering by the following students and has been accepted as satisfactory.

The presentation has been held on 10 September 2023.

**Submitted by**

|  |  |
| --- | --- |
| Name | Student ID |
| Md. Zahidul Islam | 011181080 |
| Md Mehedi Hasan | 011193012 |
| Bijoy Karmokar | 011163015 |

**Approved By**

|  |
| --- |
| **Md. Tareq Hasan**  Assistant professor &  Project Supervisor  Department Of Computer Science &  Engineering  University Of Development Alternative  (UODA) |

|  |  |
| --- | --- |
| **Md. Kawshik Ahmed** | **Md. Ashek-Al-Aziz** |
| Assistant Professor &  Co-supervisor  Department Of Computer Science &  Engineering  University Of Development Alternative  (UODA) | Chairman,  Department Of Computer Science &  Engineering  University Of Development Alternative  (UODA) |

**ACKNOWLEDGMENT**

We would like to begin by expressing our heartfelt gratitude to the Almighty for His blessings, unwavering patience, courage, and strength, which guided us throughout this project's journey, leading us to its successful completion.

Our sincere appreciation extends to our dedicated supervisor, Md. Tareq Hasan, whose continuous support, boundless knowledge, patience, and motivation were instrumental in shaping this project. His unwavering belief in our abilities and tireless guidance served as a constant source of inspiration. Working under his supervision was an immense honor, and we wish him good health, a long and prosperous life, and continued success.

We extend our thanks to our esteemed professors for their invaluable guidance and consistent support over the past few years. Special appreciation goes to the faculty of the engineering department for their crucial logistical assistance. Throughout this graduate project, we have been fortunate to receive help, support, and encouragement from numerous individuals, and we would like to express our gratitude for their unwavering cooperation.

Our sincere appreciation also goes out to our fellow classmates at the University of Development Alternative. We are grateful for the stimulating discussions, the shared sleepless nights spent working towards deadlines, and the memorable moments of fun during the last semester. Additionally, we extend our thanks to all our course mates at UODA who contributed to these discussions during the course of our studies. Your involvement was truly appreciated.

**ABSTRACT**

In the realm of higher education, final-year projects play a pivotal role in shaping the future of students by providing them with an opportunity to apply theoretical knowledge to real-world challenges. “Project Hub” is a pioneering initiative aimed at revolutionizing the way students, educators, and researchers access, share, and collaborate on final-year projects. This abstract encapsulates the essence of Project Hub, a dynamic and versatile library designed to empower the academic community.

Project Hub serves as a centralized platform, connecting students from diverse fields of study with a vast repository of final-year projects spanning various disciplines. Its intuitive interface and robust search capabilities enable users to effortlessly explore and discover projects aligned with their academic interests, fostering an environment of inspiration and collaboration.

**Key features of Project Hub include:**

Diverse Project Collection: Project Hub boasts a comprehensive collection of final-year projects across a multitude of academic disciplines, catering to the wide-ranging interests and ambitions of students worldwide.

Collaborative Environment: Through integrated collaboration tools, students can connect with peers, mentors, and industry experts to seek guidance, feedback, and support throughout their project journey.

Resource Hub: Beyond project listings, Project Hub offers a wealth of resources such as research papers, templates, and tutorials to assist students in project development, ensuring a well-rounded learning experience.

Innovative Project Showcase: Students can showcase their completed projects, promoting their work to potential employers, collaborators, and academic institutions, thereby enhancing their career prospects.

Quality Assurance: Rigorous quality control mechanisms are in place to ensure the accuracy and reliability of project information, assuring users of the trustworthiness of the platform.

Customized Learning Paths: Project Hub uses user data to provide personalized project recommendations, helping students discover projects that align with their academic and career goals.

Real-World Impact: Beyond academia, Project Hub encourages the development of projects that address real-world challenges, fostering a sense of social responsibility and innovation among students.

By bridging the gap between academia and industry, Project Hub serves as a catalyst for educational excellence, professional growth, and knowledge dissemination. It promotes the values of collaboration, innovation, and lifelong learning, making it an indispensable resource for the academic community. Through Project Hub, students embark on a journey of discovery, creativity, and application that transcends the confines of the classroom, ensuring a brighter future for the next generation of thinkers and doers.

**TABLE OF CONTENTS**

Declaration

Approval

Acknowledgement

Abstract

1 INTRODUCTION

* 1. Overview
  2. Goals
  3. Objective
  4. Benefits of using system
  5. Thesis paper outlines

2 SYSTEM STUDY AND ANALYSIS

2.1 Overview

2.2 Methodology

2.3 Initial Investigation

2.4 Feasibility Study

2.5 System Design

2.6 Cost Benefit Analysis

2.7 Hardware requirements

2.8 PC Requirements

2.9 Software Requirements

2.10 Development Tools Required

2.11 Language Requirements

2.12 Functional Requirements

2.13 Non-Functional Requirements

2.14 Schematic diagram of the system

3 SYSTEM DESIGN

3.1 Overview

3.2 Use case of the proposed system

3.3 UML diagram of the system architecture

3.4 Functionalities of proposed system

3.5 Web Design.

3.6 Hardware Design

3.7 Database Diagram/ER Diagram

3.8 Inputs

3.8.1 List of inputs

3.8.2 Input Design.

3.9 Outputs........

3.9.1 List of outputs

3.9.2 Output design

4 CODING AND TESTING

4.1 Coding.

4.1.1 Code Readability

4.1.2 Coding Standards

4.1.3 Coding guidelines

4.2 Testing

4.2.1 Defects and failures

4.2.2 Input combination and preconditions

4.2.3 Testing method.

4.2.4 Testing labels

4.2.5 Testing process

5 IMPLEMENTATION

5.1 Implementation

5.1.1 Algorithmic Complexity

5.1.2 Methodology

5.2 Application User Flowchart .

5.3 Functionalities of proposed system

5.4 Implementation Tools

5.5 Hardware implementation

5.6 Software Implementation

5.7 Debugging

5.8 Screenshot of Input Design

5.9 Screenshot of Output Design

6 DOCUMENTATION & MAINTENANCE

6.1 Software Documentation

6.2 Types of Documentation

6.2.1 Requirements Documentation

6.2.2 Architecture Documentation

6.2.3 Technical Documentation

6.2.4 User Documentation

6.2.5 Composing Software Documentation

6.2.6 Marketing Documentation

6.3 Software Maintenance

6.4 Importance of Software Maintenance

6.5 Software Maintenance Planning

6.6 Software Maintenance Process

6.7 Maintenance to Modify the System's Functionality

6.8 Categories of maintenance

6.9 Factors Affect Maintenance Cost

7 DISCUSSION AND CONCLUSION

7.1 Discussion

7.2 Challenges of the system

7.3 Future Scopes.

7.4 Conclusion

BIBLIOGRAPHY

REFERENCE

URLS

USER GUIDE

APPENDIX (SOURCE CODE).

A white and blue cover with blue text

Description automatically generated

**1.1 Overview**

In the age of information and digital connectivity, education and research are evolving rapidly, and the importance of accessible resources cannot be overstated. "Project Hub" emerges as a comprehensive and innovative digital platform designed to revolutionize the way students, educators, researchers, and enthusiasts access, share, and collaborate on academic projects. This overview provides a glimpse into the transformative potential and features of Project Hub.

Project Hub is a dynamic online repository that serves as a centralized hub for academic projects across diverse disciplines. It caters to a wide-ranging audience, including undergraduate and postgraduate students, researchers, teachers, and industry professionals, fostering an environment where knowledge thrives.

Project Hub emerges as a transformative force, bridging the gap between traditional academia and the evolving demands of a digital world. It promotes the values of collaboration, innovation, and lifelong learning, ensuring that education and research are not confined to the classroom. As users embark on a journey of discovery, creativity, and application, Project Hub paves the way for a brighter future, where knowledge is accessible to all who seek it.

If you are looking for a powerful project management platform to help you improve your project planning, execution, and collaboration, then Project Hub is a great option.

Resource Library: Beyond project listings, Project Hub offers a resource library comprising research papers, templates, tutorials, and other learning materials. These resources enhance the quality and depth of academic endeavors.

Project Showcase: Project Hub provides students with a platform to showcase their completed projects, promoting their work to potential employers, collaborators, and institutions. It serves as a springboard for career opportunities.

Personalized Recommendations: The platform leverages user data to offer tailored project recommendations, enabling students to discover projects aligned with their academic and career goals.

Real-World Impact: Beyond academic pursuits, Project Hub encourages the development of projects that address real-world challenges. This promotes innovation, social responsibility, and practical problem-solving skills.

**1.2 Goals**

Project Hub is a dynamic and innovative project library website dedicated to revolutionizing the way students, educators, and researchers access, share, and collaborate on academic projects. In this essay, we will delve into the specific goals that drive Project Hub's mission and its potential impact on the academic community.

The primary goal of Project Hub is to establish an extensive and diverse collection of academic projects across various fields. This goal serves as the cornerstone of the platform, aiming to provide a comprehensive resource for students and researchers seeking inspiration and valuable project-related content. We are committed to empowering students by offering them a platform to showcase their completed projects. This goal enhances students' visibility to potential employers, collaborators, and academic institutions, ultimately boosting their career prospects.

Project Hub also encourages projects that address real-world challenges. This goal fosters innovation, social responsibility, and practical problem-solving skills among users, emphasizing the platform's potential for real-world impact.

So, our mission is to build an active and engaged community. This goal encourages users to participate in discussions, share knowledge, and collaborate on projects, creating a supportive ecosystem for learning and growth.

**1.3 Objectives**

1. To get knowledge and ideas about the final year project.
2. To collaborate and for knowledge sharing, archives can facilitate collaboration and knowledge sharing among project team members.
3. To encourage research. Project libraries can also encourage research by providing access to primary sources related to a project. This information can be used by researchers to study the project's history, development, and impact.
4. Long-term preservation and sustainability: Project libraries can be designed to ensure the long-term conservation and sustainability of project information. This includes considerations such as appropriate storage formats, data backup, migration strategies, and recovery plans.
5. Creating easy access to previous projects for faculty and students
6. To filter if the project has been done by a previous batch.
7. To generate new project ideas.
8. To make a portfolio of university projects.

Therefore, we proudly introduce to you The Project Hub - A Project Library with a solemn commitment to "Believe In Your Project But Never Stop Improving"

**1.4 Benefits of using "Project Hub"**

1. Comprehensive Project Repository: Project Hub offers a diverse and extensive collection of final-year projects across various fields of study, making it a valuable resource for students
2. Time Savings: Users can save significant time by finding relevant projects and research materials in one centralized platform, eliminating the need for extensive searching and browsing across multiple sources.
3. Educational Resources: In addition to projects, Project Hub provides access to educational resources such as research papers, templates, tutorials, and guides, which can enhance users' learning and project development.
4. Career Advancement: Students can showcase their completed projects on Project Hub, increasing their visibility to potential employers, collaborators, and academic institutions, thereby improving their career prospects.
5. Personalized Recommendations: Through user data analysis, Project Hub offers personalized project recommendations, assisting users in discovering projects aligned with their academic and career interests.
6. Community Engagement: Users can actively engage in discussions, share their knowledge and experiences, and collaborate on projects, creating a vibrant and supportive online community.
7. Support for Educators: Educators can utilize Project Hub as a teaching resource, incorporating projects and resources into their curriculum to promote project-based learning and enhance student engagement.
8. Research Collaboration: The platform fosters research collaboration by allowing users to connect with individuals and teams working on similar research topics, promoting interdisciplinary collaboration.
9. Data-Driven Insights: Users can measure the impact of Project Hub on their academic and professional development through data analytics, helping them make informed decisions for their future projects and careers.

These benefits collectively make **Project Hub** a valuable tool for students, educators, and researchers, promoting collaboration, knowledge sharing, and excellence in academic and research endeavors.

**1.5 Thesis Paper Outline**

The project is divided into seven chapters:

Chapter 1: Introduction

Chapter 2: System Study & Analysis

Chapter 3: System Design

Chapter 4: Coding & Testing

Chapter 5: Implementation

Chapter 6: Documentation & Maintenance

Chapter 7: Conclusion

A white and blue text with blue and white text

Description automatically generated with medium confidence

**2.1 Overview**

Project Hub is a cutting-edge online platform designed to empower final-year students and aspiring innovators with a treasure trove of project ideas and resources. With the goal of fostering creativity, collaboration, and academic excellence, Project Hub is the go-to destination for those seeking inspiration, guidance, and support in their project development journey.

**Why Project Hub?**

* Project Idea Repository: Project Hub boasts an extensive collection of project ideas across various disciplines and domains. Students can easily explore a wide range of ideas, from computer science and engineering to arts and social sciences, ensuring that there's something for everyone.44
* Submission and Sharing: Final-year students and project creators can submit their own project ideas to contribute to the growing database. This collaborative approach encourages knowledge sharing and inspires new innovations.
* Expert Mentorship: Project Hub connects students with experienced mentors and professors who can provide valuable insights, guidance, and feedback on project proposals. This feature ensures that students receive expert support throughout their project journey.
* Search and Filter Functionality: The platform offers advanced search and filter options, allowing users to narrow down project ideas based on their interests, skills, and academic requirements. This ensures that students can find projects that align with their goals and resources.
* Resource Library: Project Hub offers a comprehensive resource library filled with articles, tutorials, templates, and tools to assist students in various project phases, from inception to completion. These resources facilitate skill development and project management.
* Collaboration Tools: Students can form project teams and collaborate seamlessly within the platform. Integrated communication tools, file sharing, and project tracking features ensure effective teamwork and project progress monitoring.
* Project Showcase: Outstanding projects developed by students are showcased on Project Hub, providing recognition and exposure for their hard work. This also serves as inspiration for future project creators.
* Discussion Forums: A dedicated discussion forum allows users to engage in discussions, ask questions, and seek advice from the community. This fosters a sense of community and encourages knowledge exchange.
* Personalized Dashboards: Users can create personalized dashboards to keep track of their project ideas, submissions, and progress. Notifications and updates are easily accessible, ensuring a smooth user experience.
* Feedback and Review System: Students can receive feedback on their project proposals from peers, mentors, and the community. This iterative feedback process helps refine ideas and enhance project quality.

**Benefits:**

* Inspiration and Innovation: Project Hub serves as a constant source of inspiration, encouraging students to explore new ideas and push the boundaries of innovation.
* Efficient Project Planning: Access to resources, mentors, and collaborative tools streamlines the project planning and execution process.
* Community Support: The platform fosters a sense of community among students and project enthusiasts, promoting knowledge sharing and collaboration.
* Recognition: Outstanding projects have the opportunity to gain recognition and exposure, further motivating students to excel in their academic pursuits.

Project Hub is not just a website; it's a vibrant ecosystem where creativity thrives, ideas flourish, and academic excellence is celebrated. Join us on this exciting journey of exploration, innovation, and collaboration, and unlock the potential of your final year project with Project Hub.

**2.2 methodology**

The methodology for developing "Project Hub" aims to create a user-friendly and efficient website that serves as a platform for students to submit and access project ideas. The website's primary goal is to facilitate knowledge sharing and collaboration among students. This methodology outlines the steps and processes involved in developing Project Hub.

**1. Project Planning**

**1.1 Define Project Objectives:**

* + Identify the primary goals of Project Hub, including user needs and project submission criteria.
  + Determine the scope and scale of the project.

**1.2 Stakeholder Analysis:**

* + Identify the key stakeholders, such as students, professors, and administrators.
  + Gather feedback and requirements from these stakeholders.

**2. Requirement Analysis**

**2.1 User Requirements:**

* + Conduct surveys and interviews with final-year students to understand their needs and preferences.
  + Define user personas to guide design decisions.

**2.2 Functional Requirements:**

* + Specify the features of Project Hub, such as user registration, project submission, search and filtering options, commenting, and rating system.
  + Define user roles and access levels.

**2.3 Non-Functional Requirements:**

* + Determine performance, security, and scalability requirements.
  + Establish design guidelines for user interface and user experience (UI/UX).

**3. System Design**

**3.1 Database Design:**

* + Create a database schema to store user data, project information, and related metadata.
  + Choose an appropriate database management system (e.g., MySQL, MongoDB).

**3.2 Application Architecture:**

* + Select a suitable technology stack for web development (e.g., Node.js, React, Python, Django).
  + Define the overall system architecture, including front-end and back-end components.

**3.3 UI/UX Design:**

* + Create wireframes and prototypes for the website's user interface.
  + Ensure the design is user-friendly, responsive, and accessible.

**4. Development**

**4.1 Front-end Development:**

* + Develop the website's front-end components based on the UI/UX design.
  + Implement features such as user registration, project submission forms, and search functionality.

**4.2 Back-end Development:**

* + Build the back-end logic for user authentication, project management, and database integration.
  + Implement security measures to protect user data.

**4.3 Integration and Testing:**

* + Integrate front-end and back-end components.
  + Conduct unit testing, integration testing, and user acceptance testing to identify and resolve bugs and issues.

**5. Deployment**

**5.1 Server Setup:**

* + Choose a reliable hosting environment (e.g., AWS, Heroku).
  + Configure server settings and deploy the application.

**5.2 Continuous Integration/Continuous Deployment (CI/CD):**

* + Set up automated deployment pipelines to streamline updates and maintenance.

**6. User Training and Documentation**

**6.1 User Training:**

* + Provide user training materials and tutorials to guide users on how to use Project Hub effectively.

**6.2 Documentation:**

* + Create documentation for administrators and developers to facilitate future maintenance and enhancements.

**7. User Testing and Feedback**

**7.1 Beta Testing:**

* + Invite a select group of users to participate in beta testing to gather feedback.
  + Make necessary improvements based on user feedback.

**7.2 User Feedback Loop:**

* + Establish a feedback mechanism for ongoing user input and feature requests.

**8. Launch and Monitoring**

**8.1 Launch:**

* + Officially launch Project Hub to the target audience.

**8.2 Monitoring and Maintenance:**

* + Implement monitoring tools to track website performance and user activity.
  + Regularly update and maintain the website to ensure it remains functional and secure.

**9. Evaluation and Improvement**

**9.1 User Analytics:**

* + Analyze user data to assess the website's success in meeting its objectives.

**9.2 Continuous Improvement:**

* + Use user analytics and feedback to make iterative improvements to Project Hub.
  + Stay updated with emerging technologies and trends in web development to enhance the platform.

**2.3 Initial Investigation**

The purpose of this initial investigation is to explore the feasibility and key considerations for the development of "Project Hub," a project library website designed to facilitate the submission and exchange of project ideas among students. This investigation covers several critical aspects of the project, including user registration, user interface, messaging system, user feedback, security, compatibility, legal compliance, and the potential impact of the website.

1. **User Registration:** To provide a seamless and secure user registration process.

**User profiles:** Collect essential information like name, email, university, and major.

**Authentication:** Implement secure authentication mechanisms (e.g., email verification, two-factor authentication).

**Privacy:** Comply with data protection regulations by securing user data and obtaining user consent for data processing.

1. **User Interface:** To create an intuitive and user-friendly interface.

**User experience (UX):** Prioritize ease of navigation, accessibility, and responsiveness.

**Search and filtering:** Include robust search and filtering options to help users find relevant project ideas quickly.

**Project submission:** Develop a straightforward and guided project submission process.

**User-friendly design:** Focus on aesthetics and visual appeal to enhance user engagement.

1. **Messaging System:** To facilitate communication and collaboration among users.

**Real-time messaging:** Implement a messaging system for users to communicate with each other.

**Notifications:** Enable email or in-app notifications for message alerts.

**Moderation:** Establish guidelines and moderation tools to ensure a positive and respectful communication environment.

1. **User Feedback:** To gather user input and continuously improve the platform.

**Feedback mechanisms:** Provide feedback forms, ratings, and comment sections for project ideas.

**User surveys:** Conduct periodic surveys to collect user opinions and preferences.

**Iterative development:** Use feedback to make enhancements and address user concerns.

1. **Security:** To protect user data and maintain platform integrity.

**Data encryption:** Encrypt sensitive data, such as user passwords and personal information.

**Secure coding:** Follow best practices in web development to prevent vulnerabilities like SQL injection and XSS attacks.

**Regular security audits:** Conduct security assessments to identify and address potential threats.

1. **Compatibility:** To ensure the website functions seamlessly across different devices and browsers.

**Cross-browser compatibility:** Test the website on major browsers (Chrome, Firefox, Safari, Edge).

**Mobile responsiveness:** Optimize the user interface for various screen sizes and resolutions.

**Device compatibility:** Verify that the website works well on both desktop and mobile devices.

1. **Legal Compliance:** To adhere to relevant legal regulations and protect user rights.

**Copyright and intellectual property:** Ensure that user-submitted project ideas respect copyright laws.

**Terms of service and privacy policy:** Draft comprehensive legal documents outlining user rights, data handling practices, and platform usage rules.

**Data protection:** Comply with data protection laws, such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act).

1. **Impact:** To assess the potential impact of "Project Hub."

**Educational impact:** Evaluate how the platform enhances learning and collaboration among students.

**User adoption:** Measure the growth in user registration and engagement.

**Long-term sustainability:** Consider the scalability and sustainability of the platform.

The initial investigation for "Project Hub" reveals the critical aspects that need to be addressed during the development process. By focusing on user registration, user interface, messaging, feedback, security, compatibility, legal compliance, and impact assessment, the project can be developed to meet the needs of students effectively while ensuring a secure and compliant platform.

**2.4  Feasibility Study**

The feasibility study for "Project Hub" aims to assess the viability of developing a project library website where final-year students can submit and access project ideas. The study will focus on three key aspects: Economic feasibility, Technical feasibility, and Market feasibility.

**1. Economic Feasibility**

Economic feasibility evaluates whether the project is financially viable and sustainable.

**Development Costs:**

* + Costs associated with web development, including software, hardware, and human resources.
  + Licensing fees for any third-party tools or software.
  + Hosting and infrastructure costs.

**Operational Costs:**

* + Costs related to ongoing maintenance, including server hosting, security updates, and bug fixes.
  + Staffing and administrative expenses, if applicable.
  + Marketing and promotional costs to attract users.

**Revenue Projections:**

* + Revenue streams, such as advertising, premium features, or partnerships.
  + Projected user growth and associated income.
  + Cost recovery timeline and profitability analysis.

**2. Technical Feasibility**

Technical feasibility assesses the project's ability to be developed and maintained.

Technology Requirements

Evaluate the technical infrastructure and technology stack required for "Project Hub." Consider the following:

* + Web development frameworks and programming languages.
  + Database management system (DBMS).
  + Hosting and server requirements.
  + Scalability considerations for potential user growth.

**Development Team**

* + Assess the availability of skilled developers and technical resources required to build and maintain the website. Ensure the team possesses the necessary expertise in web development, security, and database management.

**Technical Challenges**

* + Identify potential technical challenges and risks, such as security vulnerabilities, scalability issues, and compatibility concerns across browsers and devices. Develop strategies to mitigate these challenges.

**3. Market Feasibility**

Market feasibility examines the demand and acceptance of the project in the target market.

**Target Audience**

* + Define the specific target audience for "Project Hub," which, in this case, is final year students seeking project ideas.

**Market Research**

Conduct market research to understand the following:

* + Current demand for project ideas and collaboration platforms among final-year students.
  + Competitor analysis: Identify existing websites or platforms serving a similar purpose.
  + User preferences and pain points through surveys and interviews.

Unique Selling Proposition (USP)

* + Determine what sets "Project Hub" apart from existing solutions. Highlight unique features and benefits that will attract users.

Marketing and User Acquisition

* + Develop a marketing strategy to attract users to "Project Hub." Consider SEO, social media marketing, and partnerships with educational institutions.

Growth Projections

* + Based on market research and competitive analysis, project user growth over time. Estimate the number of users and projects on the platform within specific timeframes.

The feasibility study for "Project Hub" indicates that the project is economically viable, technically feasible, and has a promising market potential. The positive ROI, the availability of technical resources, and the identified market demand suggest that "Project Hub" has the potential to be a successful project library website for students. Further detailed planning and development are recommended to capitalize on this feasibility assessment and bring the project to fruition.

**2.5 System Design**

We have a backend or API service and another one on the web itself. The backend is REST API service. We have followed the SDLC (Systems development life cycle)

* + The backend API Service is responsible for managing all data-intensive work and business logic.

**2.6 Cost Benefit Analysis**

Following is the Breakdown of the costs that we have used for building this project:

1. Hosting Server.
2. Domain
3. CDN
4. Email Service
5. Storage Service

**2.7 Hardware requirements**

* + As this is an app no hardware is required. A smartphone or PC that has a browser will be enough to run this website.

**2.8 PC Requirements**

In this project we have used a PC with the configuration mentioned below OS: Ubuntu 22.04.1 LTS (Jammy Jellyfish)

* + Processor: Core i5
  + RAM: 8GB (4GB recommended)
  + SSD: 512 GB
  + Graphics Card: 2 GB
  + Others: Monitor, Mouse, Keyboard.

**2.9 Software Requirements**

When we examine the comprehensive system requirements for our project, it becomes evident that we need a software development framework to ensure its functionality. We utilized several software tools to accomplish this.

A group of people working on software

Description automatically generated

**IDE**

An Integrated Development Environment, commonly referred to as an IDE, is a specialized software application that provides a comprehensive set of tools and features to assist developers in creating, debugging, and managing software applications. IDEs are essential tools for software development, as they streamline the development process, enhance productivity, and improve code quality.

**VS code**

A white logo in a circle

Description automatically generated

Visual Studio Code (VS Code) is a highly popular and versatile source code editor developed by Microsoft. It has gained immense popularity among developers due to its lightweight yet powerful nature, extensive customization options, and a vibrant ecosystem of extensions. Here are some key features and aspects of Visual Studio Code:

1. Cross-Platform Compatibility: Visual Studio Code is available for Windows, macOS, and Linux, making it a truly cross-platform code editor. This flexibility allows developers to work on their preferred operating system seamlessly.
2. Open Source: VS Code is an open-source project, which means that its source code is freely available and can be modified or extended by the developer community. This openness fosters continuous improvement and innovation.
3. Intuitive User Interface: It features a clean and user-friendly interface that is easy for both beginners and experienced developers to navigate. The interface includes a sidebar for file navigation, a top bar for commands, and a main code editing area.
4. Extensive Language Support: VS Code supports a wide range of programming languages and file types out of the box, thanks to its built-in syntax highlighting, code completion, and code formatting capabilities. It also offers rich support for popular languages like JavaScript, Python, Java, C++, and more.
5. Customizability: One of the standout features of VS Code is its extensibility. Developers can enhance their coding experience by installing extensions from the Visual Studio Code Marketplace. These extensions can add support for specific languages, provide integrations with frameworks, or offer new features and tools.
6. Integrated Terminal: VS Code includes an integrated terminal that allows developers to run command-line tools and scripts directly within the editor. This is especially useful for tasks like running code, managing version control with Git, or performing system operations.
7. Version Control Integration: Git integration is seamlessly built into VS Code, making it easy for developers to manage version control, commit changes, and resolve merge conflicts without leaving the editor.
8. Debugging Capabilities: VS Code comes with a powerful debugging feature that supports multiple programming languages. It allows developers to set breakpoints, inspect variables, and step through code to identify and fix issues efficiently.
9. Live Share: Visual Studio Code offers a Live Share extension that enables real-time collaborative coding. Developers can share their development sessions with colleagues or collaborators, making it easier to collaborate remotely.
10. Continuous Improvement: VS Code is actively maintained and receives regular updates and improvements from the Microsoft team and the open-source community, ensuring that it remains up-to-date with the latest technologies and best practices.
11. Community Support: The VS Code community is active and vibrant, offering support, tutorials, and a wealth of resources for developers of all skill levels.

Whether you are a web developer, data scientist, or working on any other software-related project, Visual Studio Code has become a popular choice due to its flexibility, performance, and extensive ecosystem of extensions. It caters to a wide range of programming needs and has become an indispensable tool for many developers around the world.

**2.10 Language Requirements**

A programming language is a formal system of communication designed to instruct a computer to perform specific tasks or operations. It serves as a bridge between human understanding and

machine execution, allowing programmers to write code that computers can interpret and execute. Programming languages are fundamental tools in the field of software development and play a critical role in shaping the digital world we live in today.

**PHP**

**A blue circle with white text

Description automatically generated**

PHP, which stands for "Hypertext Preprocessor," is a widely used server-side scripting language designed primarily for web development. Created by Rasmus Lerdorf in the mid-1990s, PHP has evolved into a powerful and versatile language that underpins many dynamic websites and web applications. Here's an overview of PHP's key characteristics and its significance in web development:

* 1. Server-Side Scripting: PHP is a server-side scripting language, which means it is executed on the web server rather than the user's browser. It generates dynamic content and communicates with databases, making it essential for building web applications that interact with data and deliver personalized user experiences.
  2. Open Source: PHP is an open-source language, freely available for anyone to use, modify, and distribute. This open nature has contributed to its widespread adoption and a large, active community of developers.

* 1. Embedded in HTML: PHP code is typically embedded directly within HTML, making it easy to mix server-side logic with client-side presentation. PHP code is enclosed within <?php ... ?> tags and can seamlessly integrate with HTML and other web technologies.
  2. Cross-Platform Compatibility: PHP is compatible with various web servers (e.g., Apache, Nginx, Microsoft IIS) and can run on multiple operating systems (Windows, Linux, macOS), ensuring versatility and broad deployment options.
  3. Database Integration: PHP excels at connecting to databases like MySQL, PostgreSQL, SQLite, and others, enabling developers to retrieve, manipulate, and store data, making it indispensable for building dynamic, data-driven websites.
  4. Extensive Library Support: PHP has a rich ecosystem of libraries and extensions, which simplify common web development tasks such as handling forms, working with files, and interacting with APIs.
  5. Frameworks: PHP has a variety of frameworks like Laravel, Symfony, CodeIgniter, and Yii that provide structured development patterns and features for building scalable and maintainable web applications. These frameworks accelerate development and promote best practices.
  6. Security: PHP has evolved to include numerous security features, such as built-in functions for data validation, encryption, and secure coding practices. However, like any programming language, developers must be diligent about security to prevent vulnerabilities.
  7. Community and Documentation: PHP boasts a vast community of developers who contribute to its growth and maintenance. There is extensive documentation, tutorials, and forums available, making it easy for developers to learn and troubleshoot issues.
  8. Popularity and Prevalence: PHP has a long history and is still widely used today. It powers a significant portion of the internet, from small websites to large-scale applications like WordPress, Facebook, and Wikipedia.
  9. Performance: While PHP has made significant performance improvements over the years, it's important to note that its performance may not match that of some other languages like Node.js or Go for certain types of tasks. However, with proper optimization and caching, PHP can handle high-traffic websites effectively.

PHP is a versatile, open-source scripting language that has played a pivotal role in web development for decades. Its ease of use, database integration capabilities, extensive community, and numerous frameworks have made it a preferred choice for developing dynamic web applications. As web technologies continue to evolve, PHP remains relevant, thanks to ongoing updates and the dedication of its developer community.

**Raw PHP**

**A yellow and white rectangular object with a pink sign

Description automatically generated**

Raw PHP" typically refers to using PHP without the assistance of a specific PHP framework or content management system (CMS). It involves writing PHP code directly to create web applications, websites, or web services from scratch. While raw PHP provides complete control and flexibility, it also requires developers to handle many aspects of application development manually. Here are some key points about working with raw PHP:

1. Full Control: When you work with raw PHP, you have full control over your code and project structure. You are not bound by the conventions or limitations imposed by a framework or CMS.
2. Learning Curve: Developing in raw PHP may have a steeper learning curve, especially for beginners, as it requires a deep understanding of PHP language features, database interactions, and security practices.
3. Customization: Raw PHP allows for complete customization. You can design your application's architecture, database schema, and user interface according to your specific requirements.
4. Performance: Since there's no framework overhead, raw PHP applications can be optimized for performance. However, it also means that developers need to handle optimizations themselves.
5. Security: Security is a critical concern when using raw PHP. Developers must be diligent about implementing security measures like input validation, output escaping, and protection against common web vulnerabilities (e.g., SQL injection, and cross-site scripting).
6. Database Integration: Raw PHP can work with various database systems, including MySQL, PostgreSQL, SQLite, and others. Developers need to write their SQL queries and manage database connections directly.
7. File Structure: In raw PHP projects, you have the flexibility to define your file structure and organization. This can be both a benefit and a challenge, as it requires careful planning to maintain a clean and maintainable codebase.
8. Development Time: Developing with raw PHP may take more time compared to using frameworks or CMSs, especially for complex applications. However, it can be more efficient for simple projects where framework features are not needed.
9. Maintenance: Maintenance can be more demanding in raw PHP projects, as developers are responsible for keeping the codebase up to date, fixing bugs, and adding new features without the support of framework updates.
10. Community and Resources: While there is a wealth of resources and documentation available for PHP, raw PHP projects may not benefit from the extensive libraries and plugins that are often available in frameworks or CMS ecosystems.

In summary, working with raw PHP provides the freedom to build web applications from the ground up, tailor-made to specific requirements. It's an excellent choice when you need complete control and are comfortable with PHP's intricacies. However, it also requires a deep understanding of PHP development best practices and diligent attention to security and performance. For larger

and more complex projects, using a PHP framework can help streamline development and provide a foundation of proven practices and tools.

**2.11 Development Tools Required**

A runtime environment, often referred to simply as a "runtime," is a software or hardware infrastructure that provides the necessary resources for executing computer programs or software applications. It is a critical component of the computing environment and plays a central role in enabling the execution of code.

**XAMPP**

**A logo with a black background

Description automatically generated**

XAMPP, which stands for "Cross-Platform (X), Apache (A), MySQL (M), PHP (P), and Perl (P)," is an open-source software package that facilitates the development and testing of web applications on a local environment. It provides a convenient and comprehensive solution for setting up a local web server stack on Windows, macOS, Linux, and other platforms. Here's an overview of XAMPP:

1. Components: XAMPP includes a bundle of software components required to run a web server environment. These typically include:
   * Apache: A popular open-source web server that serves web pages to clients' browsers.
   * MySQL (or MariaDB): A relational database management system (RDBMS) used to store and manage data.
   * PHP: A server-side scripting language used for dynamic web page generation.
   * Perl: A general-purpose scripting language often used in web development.
   * phpMyAdmin: A web-based tool for managing MySQL databases.
   * Mercury Mail: An email server (SMTP, POP3, and IMAP) used for testing email functionality.
2. Cross-Platform: XAMPP is designed to work on various operating systems, making it accessible to developers regardless of their preferred platform.
3. Easy Installation: Installing XAMPP is relatively straightforward, and it provides a user-friendly interface for managing its components.
4. Local Development: XAMPP is primarily used for local web development and testing. It allows developers to create and test web applications on their own computers before deploying them to a production server.
5. Development Environment: XAMPP provides a complete development environment that includes a web server, database server, and scripting languages, ensuring that developers have all the tools they need for web application development.
6. Configuration: Users can easily configure and customize various settings within XAMPP, such as Apache's virtual hosts, PHP extensions, and MySQL databases.
7. Security Considerations: While XAMPP is convenient for development, it is not designed for production environments due to potential security vulnerabilities. Developers should not expose XAMPP installations directly to the internet without proper security measures.
8. Updates and Components: XAMPP developers periodically release updates that include newer versions of its components (e.g., Apache, PHP, MySQL). Users can update their XAMPP installation to benefit from the latest features and security patches.
9. Add-Ons and Extensions: XAMPP can be extended with additional modules and components, allowing developers to incorporate specific technologies or services into their local development environment.
10. Community and Support: XAMPP has an active community of users, and there are various forums, documentation, and tutorials available to assist users with troubleshooting and development tasks.
11. Alternative Stacks: While XAMPP is a popular choice, there are alternative software stacks like WAMP (Windows, Apache, MySQL, PHP) and MAMP (macOS, Apache, MySQL, PHP) that provide similar functionality for specific platforms.

**Database**

A database is a structured collection of data organized in a way that enables efficient storage, retrieval, and management of information. Databases are fundamental to modern computing and play a critical role in various applications and industries.

**MySQL**

**A logo with a dolphin in the middle

Description automatically generated**

MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and scalability. It is one of the most popular database systems in the world and is widely used by developers and organizations for various applications, ranging from small websites to large-scale enterprise systems. Here are the key aspects and features of MySQL:

* 1. Open Source: MySQL is an open-source database, which means it is freely available for use, modification, and distribution under the GNU General Public License (GPL). This open nature has contributed to its widespread adoption and a large, active community of developers.
  2. Relational Database: MySQL is a relational database management system, which means it organizes data into tables with rows and columns. It supports SQL (Structured Query Language) for data manipulation, querying, and management.
  3. ACID Compliance: MySQL adheres to ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring that transactions are processed reliably and safely. This makes it suitable for applications where data consistency is critical.
  4. High Performance: MySQL is known for its speed and efficiency. It is optimized for read-heavy workloads and can handle a large number of concurrent users and queries.

* 1. Scalability: MySQL supports both vertical scaling (adding more resources to a single server) and horizontal scaling (using techniques like replication and clustering to distribute data across multiple servers), making it suitable for a wide range of applications.
  2. Replication: MySQL provides replication features that allow data to be replicated to multiple servers for load balancing, fault tolerance, and data redundancy. It includes options for master-slave and master-master replication configurations.
  3. Partitioning: MySQL offers partitioning support, which allows large tables to be divided into smaller, more manageable pieces. This can improve query performance and maintenance.
  4. Storage Engines: MySQL supports multiple storage engines, including InnoDB (the default, which offers ACID compliance), MyISAM, MEMORY (for in-memory storage), and more. Users can choose the most appropriate engine for their specific use case.
  5. Data Types: MySQL offers a wide range of data types, including numeric, string, date/time, spatial, and JSON, allowing developers to store and manipulate various data formats.
  6. Triggers and Stored Procedures: MySQL supports triggers and stored procedures, enabling developers to define custom logic that automatically executes when certain events occur, such as data changes.
  7. Security: MySQL provides robust security features, including user authentication, access control, and data encryption, to protect sensitive data and control who can access and modify it.
  8. Community and Support: MySQL has a vast community of users and developers, which means that there is extensive documentation, tutorials, forums, and third-party tools available to assist users with troubleshooting and development tasks.
  9. Cloud Integration: MySQL is available as a managed service on various cloud platforms, such as Amazon RDS, Google Cloud SQL, and Azure Database for MySQL, making it easier to deploy and scale in the cloud.
  10. High Availability Solutions: Several high-availability solutions and clustering technologies are available for MySQL, such as MySQL Cluster and Percona XtraDB Cluster, which enhance fault tolerance and availability.
  11. Cross-Platform Compatibility: MySQL is cross-platform and can run on various operating systems, including Windows, macOS, Linux, and others.

**GitHub Actions**

A black background with a black square

Description automatically generated with medium confidence

GitHub Actions is an automation and continuous integration/continuous deployment (CI/CD) platform offered by GitHub, a popular web-based platform for version control and collaboration among software developers. GitHub Actions allows developers to automate various aspects of their software development workflows directly within the GitHub ecosystem.

**unctional Requirements**

1. User Registration and Authentication:

* Users should be able to create accounts with unique usernames and passwords.
* Users should have the option to log in securely.

1. User Profiles:

* Users should have personalized profiles where they can provide their details, including name, email, and profile picture.
* Users should be able to edit and update their profiles.

1. Search and Discovery:

* Users should be able to search for projects and resources within the library.
* Search results should be filterable and sortable by various criteria (e.g., date, category, popularity).
* Users should be able to browse projects by categories and tags.

1. Project Submission:

* Registered users should be able to submit their own projects to the library.
* Project submission forms should include fields for project title, description, tags, category, and links to project files or repositories.

1. Project Management:

* Authors should be able to edit and update their submitted projects.
* Authors should have the option to delete their projects.

1. Rating and Reviews:

* Users should be able to rate and review projects.
* Average project ratings should be displayed alongside project listings.

1. Comments and Discussions:

* Users should be able to comment on projects and engage in discussions.
* Comment threads should be organized and easy to follow.

1. Project Details:

* Each project should have a dedicated page displaying its details, including a description, author information, project files, and links.

1. Notifications:

* Users should receive notifications for actions like comments, reviews, or updates related to their submitted projects.
* Users should have the option to enable or disable notifications.

1. Bookmarking:

* Users should be able to bookmark or "favorite" projects they find interesting for quick access later.

1. Content Reporting:

* Users should be able to report inappropriate or misleading content.
* Administrators should have a mechanism to review and take action on reported content.

1. Admin Dashboard:

* Administrators should have a secure dashboard for managing user accounts, projects, and content.
* Admins should be able to moderate submitted projects and user-generated content.

1. User Roles:

* User roles should include regular users, authors (project submitters), and administrators.
* Authors should have additional privileges to manage their own projects.

1. Accessibility and Responsiveness:

* The website should be accessible to users with disabilities.
* The website should be responsive and usable on various devices and screen sizes.

1. Security:

* User data and passwords should be securely stored and transmitted.
* The website should have measures in place to protect against common web security threats, such as SQL injection and cross-site scripting (XSS).

1. Privacy Settings:

* Users should have control over their privacy settings, including who can view their profile and contributions.

1. Terms of Service and Privacy Policy:

* The website should have clear terms of service and a privacy policy accessible to all users.

1. Analytics and Reporting:

* The website should collect usage data and provide analytics to administrators to understand user behavior and improve the site.

1. Backup and Data Recovery:

* Regular backups of user-generated content and data should be performed to ensure data integrity and availability.

1. Documentation and Help Center:

* The website should offer documentation and a help center to assist users in understanding its features and functionality.

These functional requirements should serve as a foundation for developing the Project Hub library website, allowing users to discover, contribute to, and interact with a diverse range of projects and resources.

**2.13 Non-Functional Requirements**

* + Security
  + Performance
  + Maintainability
  + Reliability

A white and blue cover with blue text

Description automatically generated

**3.1 Overview:**

**The project hub** webpage is a revolutionary platform designed, where all the projects of the University will be stored.

**Add Project,** University students can add projects completed every year here. Later students will get all the projects together very easily.

The **User Profile page**, Sign in with email and password on this website. And with that information will be left when a user signs up. He can see his information himself.

A group of people sitting at desks

Description automatically generated

**3.2 Use case of the Proposed System**

A diagram of a user

Description automatically generated

**3.3 UML diagram of the system architecture**

A diagram of a project

Description automatically generated

**3.4 Functionalities of proposed system**

* Sign up.
* Sign in.
* Add Project.
* Successful Token.
* Search and Filters.
* Rating and Review.
* Main interface.
* List of Project.
* Find specific Project.
* List of book format.

**3.5 Web Design**

The Project Hub (Webpage).

**Backend Service:**

* php.
* Raw php.
* Runtime Environment: Apache.
* Database: MySQL (XAMPP Server).

So, Full webpage design looks like the Design work like the diagram shown below:

A logo of a server

Description automatically generated

**The Project Hub**

A screenshot of a computer

Description automatically generated

A logo of a company

Description automatically generated

**Figure: Technical Stack Diagram (The Project Hub)**

**3.6 Hardware Design**

Hardware design in a project library refers to the process of creating and organizing physical components.

A computer and tablet with different devices

Description automatically generated with medium confidence

**3.7 ER Diagram**

We have used a relational database in this project. Here is the ER Diagram:

A diagram of a project

Description automatically generated

Primary Key

Relational Key

**3.8 Inputs:**

**List of Inputs:**

* Sign up.
* Sign in.
* Add Project.
* Search and Filters.
* Rating and Review.

**Input Discussion:**

1. **Create New Account:** To use these apps the user must create an account.

**Table Name:** User Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Username | E-mail | Password | Batch\_  Number | Role | Created\_date |
| 01 | Md Mehedi Hasan | mehedi35@gmail.com | \*\*\*\*\*\* | 52 | 0 | 02-06-2023 |
| 02 | Md. Zahidul Islam | zahid30@gmail.com | \*\*\*\*\*\* | 52 | 1 | 04-05-2023 |
| 03 | Bijoy Karmokar | bijoy25@gmail.com | \*\*\*\*\*\* | 52 | 2 | 03-05-2023 |
| SIGN UP | | | | | | |

1. **Sign in:** If the user has already created the account. Then next he just has to login with his e-mail and password.

|  |  |
| --- | --- |
| E-MAIL | PASSWORD |
| mehedi35@gmail.com | \*\*\*\*\*\* |
| zahid30@gmail.com | \*\*\*\*\*\* |
| bijoy25@gmail.com | \*\*\*\*\*\* |
| LOGIN | |

1. **Add project:** By coming here, all the teams of each batch can complete the information and submit the project GitHub link.

**Table Name:** Add Project Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Project\_  Name | Batch\_  Number | Project\_  Team\_  Number | Created\_  Date |
| 01 | Lost & Found | 52 | 01 | 28-07-2023 |
| 02 | The Project Hub | 52 | 02 | 05-07-2023 |
| SUBMIT | | | | |

1. **Search and Filters:** In this section, users to search for project ideas based on different criteria. Additionally, provide filters to refine search results.

**Table Name:** Searching Table

|  |  |  |
| --- | --- | --- |
| Project\_  Name | Batch\_number | Project\_  Team\_  Number |
| Lost & Found | 52 | 01 |
| Project2 | 51 | 03 |
| Project3 | 49 | 01 |

1. **Rating and Reviews:** Allow users to rate and review projects. This will help students get feedback and recommendations from their peers.

**Table Name:** Review table

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Rating\_  Number | Comment | Project\_Id |
| 01 | 5 | Good | 01 |
| 02 | 4 | Good | 02 |
| 03 | 5 | Better | 03 |

**3.9 Outputs:**

**List of Outputs**

* Main interface.
* List of Project.
* Find specific Project.
* List of book format.

**Output Discussion**

1. **Main Interface:** This is the structure of the first page of our apps.

A screenshot of a computer

Description automatically generated

1. **List of project:** Here display all batch related projects, that will be added to the app with the batch.

|  |
| --- |
| Batch’s Project List |
| Batch #52: |
| 1. Project team number1 |
| 1. Project team number2 |
| Batch #51: |
| 1. Project team number1 |
| 1. Project team number2 |
| 1. Project team number3 |
| See More… |

1. **Find specific project and suggest search related project:** We can search for specific projects here. Then we will get the GitHub link of all the projects. And with some project related to search will be suggested.

|  |
| --- |
| Available Project |
| 1. Project1. |
| 1. Project2. |
| 1. Project3. See more… |
| Suggest: |
| 1. Project1 |
| 1. Project2 |
| 1. Project3 |
| 1. Project4. See more… |

1. **Project Book list:** Here display all batch related project Book, that will be added to the app with the batch.

|  |
| --- |
| Project Book List |
| Batch #51: |
| 1. Book Town pdf |
| 1. Bengali-literacy pdf |
| Batch #52: |
| 1. Lost & Found pdf |
| 1. The Project Hub pdf |

A white and blue cover with blue text

Description automatically generated

**4.1 Coding**

Coding is the process of converting a system's design into computer language. It's crucial for translating specifications into source code, aiding in verification. Coders, distinct from designers, handle this task, aiming to optimize later stages' costs, like testing and maintenance. Despite various programming languages, all computers ultimately understand them as binary machine code (ones and zeros)

A person sitting at a desk working on a computer

Description automatically generated

**Goals of coding:**

The goals of coding include:

Functionality: Creating code that fulfills the intended functionality and features of the software or system.

Efficiency: Writing efficient code to optimize system performance and resource usage.

Reliability: Ensuring that the code works consistently and reliably under different conditions.

Readability: Writing code that is easy to understand and maintain by others and the coder themselves.

Scalability: Designing code that can adapt to future changes and accommodate growth.

Security: Implementing security measures to protect against vulnerabilities and threats.

Maintainability: Making code easy to update, debug, and enhance over time.

Documentation: Providing clear and comprehensive documentation to assist other developers in understanding and using the code.

Testing: Developing test cases and conducting thorough testing to identify and rectify errors and bugs.

Optimization: Continuously improving and optimizing the code to enhance its performance and efficiency.

**4.1.1 Code readability**

Code readability is an essential but often underestimated aspect of software development. It refers to the ease with which a human can understand, follow, and maintain a piece of code. In this essay, we will explore the importance of code readability, its impact on software development, and best practices to achieve it.

Code readability is not a luxury but a necessity in the world of software development. It significantly impacts collaboration, maintenance, onboarding, bug identification, and the longevity of a project. By adhering to best practices and valuing readability as a core principle, developers can create software that is not only functional but also easier to work with, maintain, and extend. In the end, readable code is a testament to a developer's professionalism and commitment to delivering high-quality software.

Some of these factors include:

* Different indentation styles
* (whitespace) Comments
* Decomposition
* Naming conventions for objects (such as variables, classes, procedures, etc.)

Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display.

**4.1.2 Coding Standards**

General coding standards pertain to how the developer writes code.

**Indentation**

Proper and consistent indentation is important in producing easy-to-read and maintainable programs. Indentation should be used to

1. Emphasize the body of a control statement such as a loop or a select statement
2. Emphasize the body of a conditional statement
3. Emphasize a new scope block

Programming languages like Python make indentation mandatory. This enforces making programs readable. It is a splendid case of form following function in my opinion.

Free-form languages like C/C++, Pascal, Perl, etc. are usually indented to show subordination. But this is not enforced by the language. Some programmers write very poorly formatted code. This makes it harder to understand. It can also be quite obfuscating. For example, you can indent code to falsely indicate subordination when it actually doesn’t exist. Bad business that. Clarity is necessary for maintainability. I have observed that good-looking code is generally good-running and easy-to-maintain code. The programmer cared

for i in range(1,11):

print(i)

if i == 5:

break

**Inline Comments**

Inline comments are comments placed on the same line as a code statement, starting with a hash mark and a single whitespace character. While they should be used sparingly, they can be valuable for elucidating intricate or challenging sections of code.

A comment in code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code. Inline comments explaining the functioning of the subroutine or key aspects of the algorithm are frequently used. Comments are also used to let others understand what has been done in the coding and to remind programmers of what they were thinking while writing the code.

**Structured Programming**

Structured programming (sometimes known as modular programming) is a programming paradigm that emphasizes the use of structured control flow constructs to create clear, organized, and maintainable code. It was developed as a response to the complexity and chaos that often arose in unstructured or spaghetti code.

In simple words, Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection and repetition, block structures, and subroutines.

A diagram of a computer program

Description automatically generated

30:Example Of Structured Programming.

**Object Oriented Programming**

Object-oriented programming (OOP) is a widely used programming paradigm that organizes and structures code around objects, which are instances of classes. It was developed to address the complexity of software development by promoting modularity, reusability, and maintainability. OOP has become a fundamental concept in modern software engineering and is supported by many programming languages, including Java, Python, C++, and C#.

Key principles and concepts of Object-Oriented Programming:

1. Classes and Objects: In OOP, a class is a blueprint or template that defines the structure and behavior of objects. Objects are instances of classes. For example, a "Car" class might define attributes like "color" and "speed" and methods like "accelerate" and "brake." Instances of this class would represent individual cars with their specific characteristics and behaviors.
2. Encapsulation: Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on that data into a single unit, i.e., a class. It hides the internal state and exposes a controlled interface to interact with the object. This promotes data integrity and makes it easier to manage and maintain code.
3. Inheritance: Inheritance is a mechanism that allows a new class (subclass or derived class) to inherit properties and behaviors from an existing class (base class or superclass). This promotes code reuse and the creation of hierarchies. For example, a "Vehicle" class could be a superclass to both "Car" and "Motorcycle" subclasses.
4. Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common superclass. This enables flexibility in code design and the ability to write more generic, reusable code. It includes concepts like method overriding and interfaces.
5. Abstraction: Abstraction is the process of simplifying complex reality by modeling classes based on their essential properties and behaviors. It focuses on the important details while hiding unnecessary complexity. For instance, when modeling a "Bank Account" class, you might abstract away the low-level details of how transactions are processed.
6. Association: Association represents the relationships between objects. It can be one-to-one, one-to-many, or many-to-many. For example, in a library system, there's an association between "Library" and "Book" objects where a library can contain multiple books.
7. Composition: Composition is a stronger form of association where one class contains another class as a part. For example, a "Car" object may contain an "Engine" object as one of its components.

**OOP offers several advantages:**

* Modularity: Code is organized into manageable, reusable units (classes).
* Flexibility: It supports dynamic and polymorphic behavior, making code adaptable to changing requirements.
* Code Reusability: Inheritance and composition facilitate the reuse of existing code.
* Maintenance: Encapsulation helps isolate changes, reducing the risk of breaking other parts of the code.
* Collaboration: OOP models real-world scenarios well, making it easier for teams to collaborate using common concepts.
* However, OOP is not without its challenges, including the potential for overcomplexity, difficulties with large inheritance hierarchies, and performance concerns in certain situations. It's essential to choose the right programming paradigm for the specific problem at hand and balance OOP principles with other programming paradigms when necessary.

A person standing with her hands on her hips

Description automatically generated

Fig 31: OOP Concepts

**Major OOPs Concept**

In this section, we will deep dive into the basic concepts of OOP. We will cover the following topics-

* Class
* Object
* Method
* Inheritance
* Encapsulation
* Polymorphism
* Data Abstraction

A diagram of a diagram

Description automatically generated

Fig 32: Major OOPs Concept

Classes, Subroutines, Functions, and Methods

In certain scenarios, you'll often find the need to reuse specific lines of code in various parts of your program. To address this, you can create a subroutine. Subroutines offer significant advantages, not only in terms of saving time while writing scenarios but also in enhancing code flexibility and reusability. For instance, if you repeatedly use the same lines of code across different sections of your scenario, any modifications will require updating those lines in multiple places. However, by encapsulating these code snippets within a subroutine, you only need to make changes once in the subroutine itself.

In many ways, creating a subroutine resembles defining your custom method. Subroutines can accept arguments and return values, similar to methods. However, unlike methods that are often associated with specific data types, subroutines are not type-bound. In our Built-in Functions section, you'll find methods that aren't tied to specific types. These built-in functions operate much like subroutines, allowing you to pass arguments and retrieve results. They are invoked in a manner similar to calling a subroutine. In fact, some of these built-in functions were introduced because, in our programming efforts and while addressing user queries, we frequently relied on certain subroutines. Consequently, it made sense to incorporate them into the Presentation as built-in functions.

**Source Files**

A source file is a fundamental component in software development. It is a file that contains the human-readable, plain text code written by a programmer in a specific programming language. Source files serve as the foundation for creating computer programs and applications. Here's some essential information about source files:

**Variable Names**

Variable names are fundamental elements in computer programming and play a crucial role in the development of software. They serve as labels or identifiers for storing and manipulating data within a program. Choosing appropriate and meaningful variable names is essential for writing clear, maintainable, and error-free code. In this discussion, we'll explore the significance of variable names, best practices for naming variables, and some common conventions in different programming languages.

**Variable names are used for:**

* Referring to questions in the syntax expressions;
* Naming the data columns in the export data files;
* Referring to the origin of the event in comments files.

**A variable name is a word that consists only of the following:**

* English letters A..Z and a..z;
* Digits 0..9;
* an underscore character “\_”.

No other characters are permitted in the variable name. Specifically, spaces are not permitted in the variable names, as the variable name must be a single word. The variable name may not start with a digit or underscore, and may not end with an underscore. Double underscores are not permitted in a variable name.

**Use of Braces**

Brackets are an important syntactic element in most major programming languages. They may take several forms. The most common are the "{}", "[]", ()" and "<>" brackets. There are several other names for these characters. The "{}" are referred to as curly brackets or braces while "<>" are often called angle brackets or braces. The term "curly braces" is more favored in the U.S., while "brackets" is more widely used in British English. The "()" are also frequently abbreviated 45 as "parens" since they are parentheses characters. These characters are encoded in both ASCII and Unicode.

These brackets define important constructs in a programming language. For example, in C and languages influenced by C, "{}" denotes a code block while "[]" refers to an array subscript. In Perl, the "<>" is referred to as the filehandle operator for reading from and writing to files.

**Compiler Warnings**

Compiler warnings are informative messages generated by a compiler to bring the attention of developers to potential issues within their code. Unlike compilation errors, warnings do not halt the compilation process. They do not represent syntax errors in the programming language but rather highlight areas where the code may have problems or bugs. It's worth noting that many compilers offer customization options that allow developers to continue compilation even in the presence of warnings, but this is not always advisable.

Warnings should not be dismissed or ignored. It is highly recommended to address each warning before initiating software testing. Neglecting warnings can lead to wasted time and effort when debugging later on. Even when the compiler provides explicit warnings, searching for the source of an error in the debugger can be a time-consuming task. Therefore, the goal should be to minimize the number of warnings in your codebase during the project's development phase.

Striving to eliminate all warnings or at least keeping their count to a minimum is a best practice. Fewer warnings make it easier to identify and resolve errors in new code, enhancing the overall quality and maintainability of the software.

**4.1.3 Coding Guidelines**

Good software development organizations want their programmers to maintain some well-defined and standard style of coding called coding standards. They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop. It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

**Line Length**

Avoid lines longer than 80 characters since they’re not handled well by many terminals and tools.

**Spacing**

Don't put more than one statement on a line.

* Use blank lines to separate your code into logical sections. Put a space between all binary operators (e.g., <=, =, +) and their operands. One possible exception is to emphasize precedence. a\*x + b Include a space between a keyword (e.g., while, for, if) and its opening parenthesis.
* Put a space after each statement in a for-loop

for(int i=0;i<n;i++) vs. for (int i = 0; i < n; i++)

* Put a space after each comma in an argument list.
* Put space after each comment delimiter.

//This comment has no space

// This comment has two

//after the delimiter and is

// spaces after the delimiter

//difficult to read.

// and is easier to read

* Do not put spaces before a semicolon.
* Do not put spaces between an object name, the. separator, and a method name.
* Do not put spaces between a method name and its left parenthesis.
* Include blank lines to improve readability by grouping blocks of related code.
* Use spaces to align parallel code whenever it enhances readability.

//int n = Integer.parseInt(args[0]);

// size of population int trials = Integer.parseInt(args[1]);

// number of trials

**Wrapping Lines**

The preferred way of wrapping long lines is by using Python's implied line continuation inside parentheses, brackets, and braces. If necessary, you can add an extra pair of parentheses around an expression, but sometimes using a backslash looks better. Make sure to indent the continued line appropriately. The preferred place to break around a binary operator is after the operator, not before it.

**Variable Declarations**

A declaration of a variable is where a program says that it needs a variable. For our small programs, place declaration statements between the two braces of the main method. The declaration gives a name and a data type for the variable. It may also ask that a particular value be placed in the variable. In a high-level language (such as Java) the programmer does not need to worry about how the computer hardware actually does what was asked. If you ask for a variable of type long, you get it. If you ask for the value 123 to be placed in the variable, that is what happens.

**Program Statements**

Program statements are limited to one per line. Also, nested statements are avoided when possible.

**Use of Parentheses**

It is better to use parentheses liberally. Even in cases where operator precedence unambiguously dictates the order of evaluation of an expression, often it is beneficial from a Readability point of view to include parentheses anyway.

**Meaningful Error Messages**

Error handling is an important aspect of computer programming. This not only includes adding the necessary logic to test for and handle errors but also involves making error messages meaningful.

**4.2 Testing**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects). Software testing can be stated as the process of validating and verifying a computer program/application/product.

Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed.

**importance of Testing**

The primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes the examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and does it what it needs to do? In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed. Every software product has a target audience. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software

product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

**4.2.1 Defects and failures**

Not all software defects are caused by coding errors. One common source of expensive defects is requirement gaps, unrecognized requirements that result in errors of omission by the program designer. Requirement gaps can often be non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following processes. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. A single defect may result in a wide range of failure symptoms.

**4.2.2 Input Combinations and Preconditions**

Not all software defects are caused by coding errors. One common source of expensive defects is requirement gaps, unrecognized requirements that result in errors of omission by the program designer. Requirement gaps can often be non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following processes. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. A single defect may result in a wide range of failure symptoms.

**4.2.3 Testing Method**

**Different methods of testing:**

* White-Box Testing
* Black-box Testing

**White-Box Testing**

White box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) tests the internal structures or workings of a program, as opposed to the functionality exposed to the end user. In white-box testing, an internal perspective of the system,

as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs.

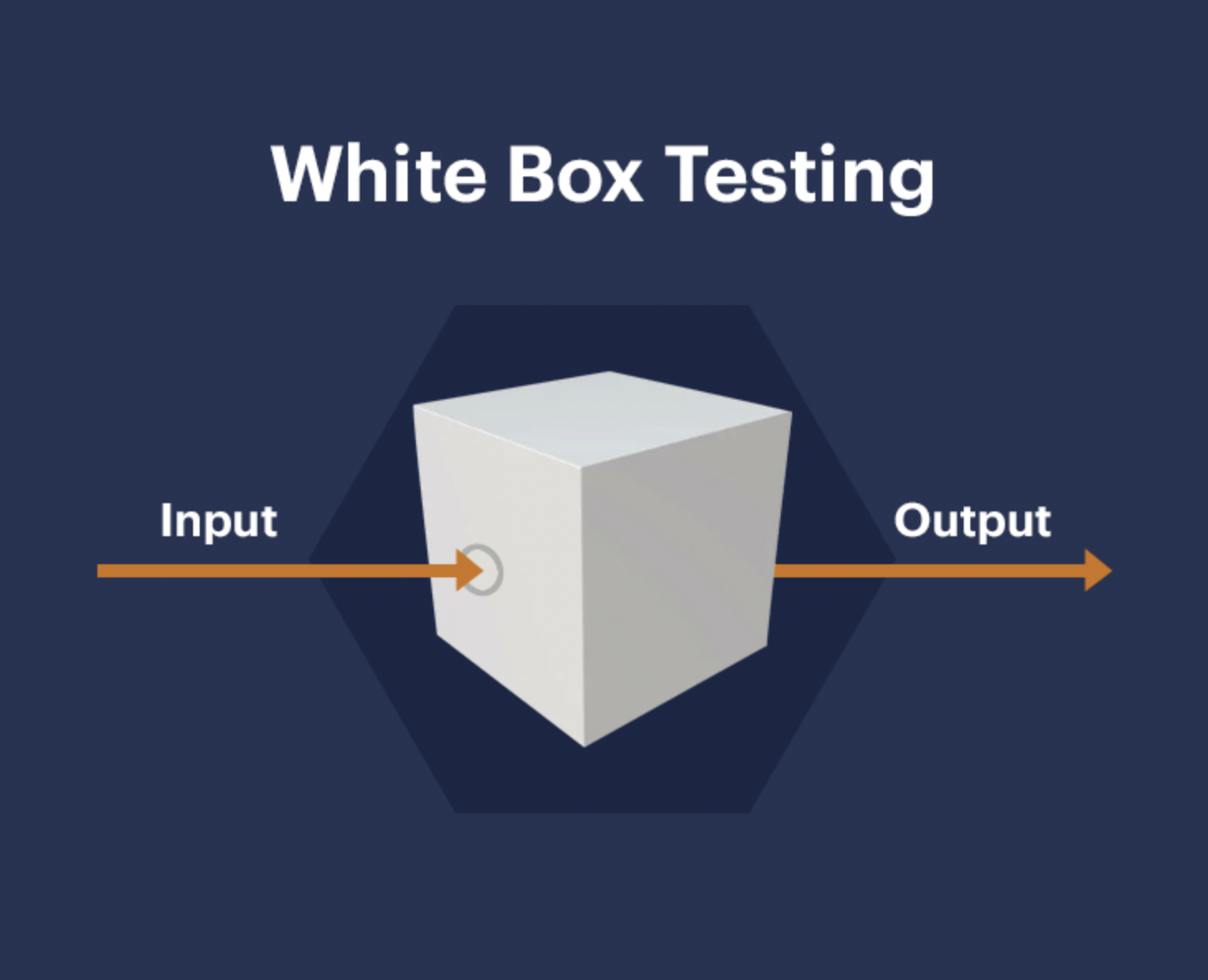


Fig 33: White Box testing.

**Black-Box Testing**

Black-box testing treats the software as a "black box", examining functionality without any knowledge of the internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing.

A diagram of a black box

Description automatically generated

Fig 34: Black Box testing.

**4.2.4 Testing Levels**

There are generally five recognized levels of tests: unit testing, integration testing, user interface testing, system testing, and acceptance testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing which is distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

**Unit Testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

A diagram of a test

Description automatically generated

Unit testing is a software development process that involves the synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted

to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis, and other software verification practices.

**Integration Testing**

Integration testing (sometimes called integration and testing, abbreviated as I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them into larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

A screenshot of a screen

Description automatically generated

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

**User Interface Testing**

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit.



Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit. Component interface testing is a variation of black-box testing, with a focus on the data values beyond just the related actions of a subsystem component.

**System Testing**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

A computer screen with icons

Description automatically generated

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and within the system as a whole.

In addition, the software testing should ensure that the program, as well as working as expected, and does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence).

**Acceptance Testing**

In engineering and its various sub-disciplines, acceptance testing is a test conducted to determine if the requirements of a specifications contract are met. It may involve chemical tests, physical tests, or performance tests.

Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting the transfer of ownership. In the case of software, acceptance testing performed by the customer is known as user acceptance testing (UAT), end-user testing, site (acceptance) testing, or field (acceptance) testing. A smoke test is used as an acceptance test prior to introducing a build to the main testing process.

A diagram of manual software testing

Description automatically generated

**4.2.5 Testing Process**

There are many approaches to software testing but effective testing is essentially a process of investigation, not merely a matter of creating and following a routine procedure. One definition of testing is ‘‘the process of questioning a product in order to evaluate it’’, where the ‘‘question’’ is operations the tester attempts to execute.

A diagram of process processes

Description automatically generated

Fig 34: Testing Process

A white and blue cover with blue text

Description automatically generated

**5.1 Implementation**

In the realm of computer science, an implementation signifies the practical realization of a technical specification or algorithm into a functional program, software component, or another computer system, achieved through the process of computer programming and subsequent deployment. It's worth noting that multiple implementations can coexist for a given specification or standard.

Effectively executing a system implementation entails a multitude of interconnected tasks that must be carried out in a well-organized sequence. Employing a tried-and-tested implementation methodology and seeking expert guidance can be advantageous, yet more often than not, challenges in implementation projects arise not from the inherent complexity of individual tasks but from factors such as the sheer number of tasks, inadequate planning, and insufficient resource allocation.

This phase follows the design stage, during which we illustrate how the designed concepts are put into practice. Depending on the specific requirements and design parameters, a range of tools and technologies, spanning from database systems to front-end software packages, will be employed in this process.

**5.1.1 Algorithmic Complexity**

Both the academic discipline and the practical domain of computer programming share a predominant focus on the exploration and application of highly efficient algorithms tailored to specific problem classes. To facilitate this endeavor, algorithms are categorized into orders through the utilization of the well-known Big O notation. This notation serves to articulate resource utilization, encompassing factors like execution time and memory consumption, relative to the size of an input.

Proficient programmers possess a deep understanding of a diverse range of established algorithms, along with their associated complexities. They leverage this expertise to judiciously select algorithms that align optimally with the particular requirements and constraints of a given context.

**5.1.2 Methodology**

In most formal software development projects, the initial phase typically involves requirements analysis, which is followed by testing for value modeling, implementation, and the crucial process of eliminating failures (commonly referred to as debugging). These essential steps can be approached in various ways, with different methodologies available for each.

For requirements analysis, one widely employed technique is Use Case analysis. When it comes to modeling, Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA) are popular methods, both of which make use of the Unified Modeling Language (UML) for notation. In the realm of database design, Entity-Relationship Modeling (ER Modeling) is a similar technique commonly used.

For software implementation, programmers have a range of options at their disposal, including imperative languages (be they object-oriented or procedural), functional languages, and logic languages, each offering distinct approaches and capabilities to bring the software design to life.

**5.2 Application User Flowchart**

A diagram of a process

Description automatically generated

Fig: Flowchart of the Project Hub

* 1. **Functionalities of the proposed system**
  + Login.
  + Signup
  + Profile
  + Add Project.
  + List of Batch.
  + Search and Filter.
  + Discussion Option.
  + Rating And Reviews.
  + Find a Specific Project.
  + Most Popular Project List.
  + User-friendly Interface / Main Interface.
  + Project Book Format as suggested by Teacher.
  + Project Proposal Format as Suggested by Teacher.

**5.4 Implementation Tools**

User interface design is an iterative process characterized by interactivity. It involves the creation and implementation of a design model, typically in the form of a prototype. This prototype is then evaluated by users who align with the user model established earlier in the process. Subsequent modifications are made based on the feedback received from these users.

To facilitate this dynamic and iterative design approach, a wide range of tools has emerged within the realm of user interface design and prototyping. These tools, often referred to as user interface toolkits or user interface development systems (UIDS), offer a collection of routines or objects that simplify the creation of various components essential for an interactive interface. These components can include windows, menus, user-device interactions, error messages, commands, and more.

Whether through prepackaged software directly usable by designers and implementers or as part of a user interface development system, these tools incorporate built-in mechanisms that streamline the design and implementation process, making it more efficient and responsive to user feedback.

* Managing input devices (such as the mouse or keyboard);
* Validating user input;
* Handling errors and displaying error messages;
* Providing feedback (e.g., automatic input echo);
* Providing help and prompts;
* Handling windows and fields, scrolling within windows;
* Establishing connections between application software and the interface;
* Insulating the application from interface management functions; and
* Allowing the user to customize the interface
* The functions described above can be implemented using either a language-based or a graphical approach.

**5.5 Hardware implementation:**

There is no such hardware implementation in this project The Project Hub.

**5.6 Software implementation:**

We have followed the microservice architecture here.

API (Application Programming Interface)

**Backend Service**

1. Database Management:

* MySQL, PostgreSQL, MongoDB, or another database system to store and retrieve data.

1. User Authentication and Authorization:

* User registration and login system. User roles and permissions management.

1. Session Management:

* Handling user sessions and maintaining state.

1. API Endpoints:

* Building API endpoints for client-server communication. RESTful or GraphQL APIs depending on your requirements.

1. Data Validation and Sanitization:

* Input validation and data sanitization to prevent security vulnerabilities.

1. File Upload and Storage:

* Handling file uploads and managing storage. Amazon S3, local file system, or cloud storage options.

1. Email Services:

* Sending emails for notifications, account verifications, etc. Libraries like PHPMailer or using third-party email services.

1. Caching:

* Implementing caching mechanisms to improve performance. Memcached or Redis for caching data.

1. Security:

* Protection against common security threats like SQL injection, XSS, CSRF, etc. Regular security audits and updates.

1. Error Handling and Logging:

* Centralized error logging for debugging and monitoring. Tools like Monolog for logging.

1. Background Processing:

* Executing time-consuming tasks asynchronously. Tools like RabbitMQ, Gearman, or dedicated job queues.

1. Content Management System (CMS):

* If your website needs content management, consider using a CMS like WordPress or building a custom one.

1. Third-Party Integrations:

* Integrating with external services, APIs, or payment gateways.

1. Performance Optimization:

* Profiling and optimizing code for better performance. Using tools like New Relic or Blackfire for performance monitoring.

1. Scalability and Load Balancing:

* Preparing for traffic spikes and scaling your infrastructure. Load balancers and horizontal scaling.

1. Analytics and Monitoring:

* Implementing analytics tools like Google Analytics or custom solutions. Server monitoring with tools like Nagios, Zabbix, or Prometheus.

1. Version Control:

* Git or another version control system for code management.

1. Deployment and DevOps:

* Continuous Integration/Continuous Deployment (CI/CD) pipelines. Docker containers and container orchestration (e.g., Kubernetes).

1. Database Backups and Maintenance:

* Regular database backups and maintenance routines.

1. Documentation:

* Comprehensive documentation for your backend services and APIs.

**5.7 Debugging**

Debugging plays a pivotal role in the software development process due to the potentially substantial repercussions that can arise from a flawed program, affecting its end users. Certain programming languages are more susceptible to particular types of errors, often due to their specifications not mandating rigorous checks by compilers compared to other languages. Employing static analysis tools can be instrumental in identifying potential issues in the code.

**5.8 Screenshot of Input Design**

* Login
* Sign up.
* Profile
* Project List
* Dashboard
* Add Project

A white and blue cover with blue text

Description automatically generated

**6.1 Software Documentation**

Software documentation is a written piece of text that is often accompanied by a software program. This makes the lives of all the members associated with the project easier. It may contain anything from API documentation, build notes, or just help content. It is a very critical process in software development. It’s primarily an integral part of any computer code development method. Moreover, computer code practitioners are a unit typically concerned with the worth, degree of usage, and quality of the actual documentation throughout the development and its maintenance throughout the total method. Motivated by the requirements of Novatel opposition, a world-leading company developing package in support of worldwide navigation satellite system, and based mostly on the results of a former systematic mapping studies area unit aimed at a higher understanding of the usage and therefore the quality of varied technical documents throughout computer code development and their maintenance.

A computer and a cup of coffee

Description automatically generated

It must fulfill the objective of addressing problems encountered by developers, end-users, or during customer interactions with the knowledge base. Adequate information and descriptions should be documented to accomplish the following objectives:

1. Resolve issues encountered by the developer during the development process
2. Help end-users understand the product
3. Assist customers and the support team to find the information

Effective documentation practices play a pivotal role in ensuring the success of software. Comprehensive documentation should encompass an engaging user experience, well-structured information architecture, and a profound understanding of your target audience

**6.2 Types of Documentation**

A diagram of a product documentation

Description automatically generated

Fig 55: Types of Documentation

* Requirements: These are descriptions that outline the features, abilities, traits, or qualities of a system. They form the fundamental basis for what is planned or has already been developed.
* Architecture/Design: This entails an outline of software that includes its relationship with the surrounding environment and the principles guiding the construction of software components.
* Technical: This includes documentation related to the code, algorithms, interfaces, and APIs used in the system.
* End User: These are manuals intended for various user groups, including end-users, system administrators, and support staff.
* Marketing: This involves strategies for promoting the product and analyzing market demand.

**6.2.1 Requirements Documentation**

Requirements documentation serves as a comprehensive description of the intended functionality of a specific software system. It plays a crucial role throughout the software development process by facilitating clear communication about the system's purpose and features. Additionally, it serves as the basis for agreements regarding the software's functionality. These requirements are essential for all stakeholders involved in the software development lifecycle, including end users, customers, product managers, project managers, sales and marketing teams, software architects, usability engineers, interaction designers, developers, and testers, among others.

Requirements come in a variety of styles, notations, and formality. Requirements can be goal-like (e.g., distributed work environment), close to design (e.g., builds can be started by right-clicking a configuration file and selecting the 'build' function), and anything in between. They can be specified as statements in natural language, as drawn figures, as detailed mathematical formulas, and as a combination of them all.

**6.2.2 Architecture/Design Documentation**

Architecture documentation, also referred to as software architecture descriptions, represents a distinct category within design documentation. In essence, these documents are the culmination of software development, with design documents as the intermediary step and code documents as the initial stage. The content of architecture documents generally doesn't delve into the specifics of the actual codebase. Instead, they prioritize providing in-depth explanations while minimizing detailed implementation information. While architecture documents may hint at potential approaches for lower-level design, they often defer the detailed exploration and trade studies to other supporting documents.

A very important part of the design document in enterprise software development is the Database Design Document (DDD). It contains Conceptual, Logical, and Physical Design Elements. The DDD includes the formal information that the people who interact with the database need. The purpose of preparing it is to create a common source to be used by all players within the scene. The potential users are:

* Database Designer
* Database Developer
* Database Administrator
* Application Designer
* Application Developer

**6.2.3 Technical Documentation**

When programmers refer to software documentation, they typically encompass more than just the code itself. While code is a crucial component, it cannot stand alone. Complementary text is essential to elucidate various aspects of the software's intended functionality. Effective code documentation should be comprehensive without becoming excessively verbose, striking a balance that facilitates maintenance. Within the realm of software documentation, API Writers often create specific how-to guides and overview documents tailored to the particular software application or product. These resources serve a diverse audience, including developers, testers, and end-users who rely on the software for their respective needs.

**6.2.4 User Documentation**

Code documents and user documents serve distinct purposes in software development. User documents primarily focus on explaining how a program is utilized, while code documents delve into the technical details of the program's implementation.

For a software library, these two types of documents might essentially convey the same information and can be beneficial when combined. However, in the context of a general application, it's less common for them to align closely or be interchangeable.

There are three broad ways in which user documentation can be organized

* Tutorial: A tutorial approach is considered the most useful for new users, in which they are guided through each step of accomplishing particular tasks.
* Thematic: A thematic approach, where chapters or sections concentrate on one particular area of interest, is of more general use to an intermediate user. Some authors prefer to convey their ideas through a knowledge-based article to facilitate the user's needs. This approach is usually practiced in a dynamic industry, such as Information technology, where the user population is largely correlated with the troubleshooting demands
* List of Reference: The final type of organizing principle is one in which commands or tasks are simply listed alphabetically or logically grouped, often via cross-referenced indexes. This latter approach is of greater use to advanced users who know exactly what sort of information they are looking for.

**6.2.5 Composing Software Documentation**

Similar to various types of technical documentation, effective software documentation thrives when developed through a structured process. Within the realm of software documentation, this process, commonly observed in the industry, encompasses five distinct steps:

A group of people working on software

Description automatically generated

* User analysis is the basic research phase of the process.
* Planning or the actual documentation phase.
* Draft review is a self-explanatory phase where feedback is sought on the draft composed in the previous step.
* Usability testing, whereby the usability of the document is tested empirically.
* Editing is the final step in which the information collected in steps three and four is
* used to produce the final draft.

**6.2.6 Marketing Documentation**

To effectively engage potential users and capture their interest in a product, it becomes essential to create promotional materials that entice casual observers to delve deeper into understanding the product. These promotional documents serve three primary purposes:

* To excite the potential user about the product and instill in them a desire to become more involved with it.
* To inform them about what exactly the product does, so that their expectations are in line with what they will be receiving.
* To explain the position of this product with respect to other alternatives. One good marketing technique is to provide clear and memorable catchphrases that exemplify the point we wish to convey, and also emphasize the interoperability of the program with anything else provided by the manufacturer.

**6.3 Software Maintenance**

In the realm of software engineering, software maintenance entails the process of altering a software product once it has been delivered, with the aim of rectifying faults, enhancing performance, or addressing other aspects. While many people tend to associate maintenance primarily with bug fixes, it's essential to recognize that a substantial portion, exceeding 80%, of maintenance activities actually involve non-corrective actions.

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanisms must be developed for evaluation, controlling, and making modifications.

**6.4 Importance of Software Maintenance**

Maintenance activities are categorized into four classes:

1. Adaptive – Modifying the system to cope with changes in the software environment (DBMS, OS).
2. Perfective – Implementing new or changed user requirements that concern functional

enhancements to the software.

1. Corrective – Diagnosing and fixing errors, possibly ones found by users.
2. Preventive – Increasing software maintainability or reliability to prevent problems in the

future.

**6.5 Software Maintenance Planning**

Maintenance is a crucial aspect of software development, requiring a well-thought-out plan to be established alongside the initial development process. This plan must outline how users can request changes or report issues, along with budgetary considerations encompassing resources and cost estimates. It's imperative to make informed decisions for each new system feature and define quality objectives.

Software maintenance, a phase that can extend for many years beyond the development stage—sometimes even spanning decades—demands an effective strategy. This strategy should encompass defining the scope of maintenance, tailoring the post-delivery and deployment processes, designating responsible parties for maintenance, and estimating the entire life-cycle costs involved.

**6.6 Software Maintenance Processes**

This section describes the four software maintenance processes:

1. The implementation process contains software preparation and transition activities, such as the conception and creation of the maintenance plan; the preparation for handling problems identified during development; and the follow-up on product configuration management.
2. The process of acceptance of the modification, by confirming the modified work with the individual who submitted the request to make sure the modification provided a solution.
3. The migration process (platform migration, for example) is exceptional and is not part of daily maintenance tasks. If the software must be ported to another platform without any change in functionality, this process will be used and a maintenance project team is likely to be assigned to this task.
4. Finally, the last maintenance process, also an event that does not occur daily, is the retirement of a piece of software.

**6.7 Maintenance to Modify the System’s Functionality**

Maintenance of a system becomes imperative when it must adapt to evolving organizational or business demands and corresponding system requirements. In an ideal scenario, maintenance can be conceptualized as an iterative development process. The ensuing diagram illustrates the stages of the maintenance process.

A diagram of a maintenance process

Description automatically generated

Fig 56: System’s Functionality

**6.8 Categories of Maintenance**

There are three categories of maintenance: corrective, adaptive, and perfection.

1. Corrective maintenance: Reactive modification of a software product performed after delivery to correct discovered problems.
2. Adaptive maintenance: Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment.
3. Perfective maintenance: Modification of a software product after delivery to improve performance or maintainability.

**6.9 Factors Affect Maintenance Cost**

After addressing the fault and completing the necessary repairs, a proactive measure involves revisiting the emergency change at a later point to evaluate the product using an iterative development approach. Additionally, we take into account the various factors that can impact maintenance expenses.

A pie chart with text and numbers

Description automatically generated

Fig 57: Factors Affect Maintenance Cost

Following are some of the factors:

* Understanding how the software is to be used by the user is important.
* If the person who wrote the code is responsible for its maintenance, the cost will be reduced.
* If the external environment of the program is stable, the program will require less maintenance.
* If the program is designed for a particular hardware platform and does not change during the lifetime of the system, the cost will be reduced.

A white and blue background with text

Description automatically generated

**7.1 Discussion**

Building The Project Hub was never an easy task. But thanks to the almighty we have successfully developed the system for our project. By developing this system, we have solved multiple problems and learned many programming logic, ideas, and processes. We have used various tools and Technologies for this system. We have used PHP as a language, also we have also used, XAMPP, HTML, CSS, and SQL for databases. By deploying this We have learned Docker, Ngnix, CICD, CloudFlare, etc. Also from a development point of view, we have tried to practice a solid paradigm, singleton pattern, etc. A framework is a collection of modules and prewritten code that helps developers with particular tasks — in our case, we have worked with Laravel. In today’s scenario, whenever we are developing a website, we will need a database. So we have worked with MySQL.



Building The Project Hub was a journey of learning, unlearning, and relearning. While working we have taken wrong decisions but we quickly made a call to unlearn and relearn. We are so happy that we have done it.

**7.2 Challenges of the System**

Project constraints are limiting factors for your project that can impact quality, delivery, and overall project success. This was our first time developing such a system practically. The time allocated for the project was insufficient to complete or polish all the desired functionalities. So there are some Challenges to this system to which solutions can be provided for future development.



1. **Content Management:** Organizing and maintaining a vast repository of projects can be challenging. Ensuring that all projects are categorized correctly, tagged appropriately, and have up-to-date information can be a time-consuming task.
2. **Data Quality:** Users rely on accurate and reliable information when searching for projects. Ensuring data quality by verifying project details, eliminating duplicates, and addressing inaccuracies can be demanding.
3. **Scalability:** As more projects are added to the library, the platform needs to scale efficiently to handle increased traffic and data storage requirements. This often involves dealing with issues related to server performance and database optimization.
4. **User Engagement:** Encouraging users to actively contribute to projects, reviews, and ratings can be a challenge. You'll need strategies to foster a vibrant community and sustain user interest over time.
5. **Content Moderation:** To maintain the quality and appropriateness of projects, you'll need an effective content moderation system to filter out spam, offensive content, or irrelevant submissions.
6. **Security:** Protecting user data, ensuring secure transactions (if applicable), and preventing unauthorized access or data breaches is crucial. This includes implementing robust authentication and authorization mechanisms.
7. **Search and Recommendation Algorithms:** Developing an efficient search engine and recommendation system to help users find relevant projects can be complex. You'll need to consider factors like user preferences, project metadata, and user behavior.
8. **User Experience (UX):** Creating an intuitive and user-friendly interface is essential for user adoption. Balancing features with ease of use can be a challenge, and user feedback will be crucial for iterative improvements.
9. **Legal and Copyright Issues:** Projects might have intellectual property rights or licensing restrictions that need to be addressed. Ensuring that all projects comply with copyright and licensing regulations is critical.
10. **Funding and Monetization:** Sustaining the platform may require finding viable monetization strategies, such as premium features, ads, or subscriptions. Developing a sustainable business model can be challenging.
11. **Competition:** Depending on your project's niche, you might face competition from other project libraries or platforms. Standing out and offering unique value can be difficult in a crowded market.

**7.3 Future Scopes**

1. **Expand the range of projects:** Currently, most project hubs focus on a specific type of project, such as software development projects or academic projects. In the future, The Project Hub could expand to include a wider range of projects, such as business projects, social impact projects, and creative projects. This would make project hubs more accessible to a wider range of people and organizations.
2. **Improve the search and filtering features:** The search and filtering features of most project hubs are basic. In the future, project hubs could improve these features to make it easier for users to find the projects they are looking for. For example, project hubs could allow users to search by project type, keywords, skills, and other criteria.
3. **Add more features to support project collaboration:** Currently, most project hubs only provide basic features for project collaboration, such as commenting and file sharing. In the future, project hubs could add more features to support project collaboration, such as task management, version control, and chat. This would make it easier for teams to work together on projects.
4. **Make it easier to create and manage projects:** Currently, it can be difficult to create and manage projects on project hubs. In the future, project hubs could make it easier for users to create and manage projects by providing templates, wizards, and other tools. This would make it more accessible for people with less experience to create and manage projects.
5. **Provide more support for project managers:** Project managers are responsible for the success of projects. In the future, project hubs could provide more support for project managers by providing training materials, templates, and other resources. This would help project managers to be more effective in their roles.
6. **Connect project participants:** Project hubs can be a great way to connect project participants, such as team members, clients, and partners. In the future, project hubs could provide more features to connect project participants, such as social networking features and event management features. This would help project participants to stay connected and collaborate more effectively.
7. **AI-Driven Project Recommendations:** Implement advanced recommendation algorithms based on users' project history and preferences. Use AI to suggest relevant projects, tools, and resources to users, making their project discovery more personalized and efficient.
8. **Monetization Features:** Enable project creators to monetize their work by offering premium project versions, access to exclusive content, or through crowdfunding mechanisms. Implement a revenue-sharing model to benefit both creators and the platform.
9. **Educational Resources:** Expand beyond project sharing by offering tutorials, online courses, and educational content related to various project domains. This can attract a wider audience, including students and educators.



These are just a few of the future scopes of a project hub. As technology continues to evolve, project hubs will continue to develop and find new ways to help people with their projects.

**7.4 Conclusion**

In conclusion, Project Hub represents a significant milestone in the world of digital libraries and collaborative project management. This dynamic and innovative platform has seamlessly integrated the principles of convenience, accessibility, and collaboration to create a truly unique and invaluable resource for individuals and teams alike. Throughout the course of this project, we have strived to provide an informative overview of the key features and benefits that Project Hub brings to the table.

One of the standout features of Project Hub is its user-friendly interface. Through intuitive design and thoughtful navigation, users can easily find, access, and contribute to a vast repository of knowledge, resources, and projects. The intuitive search functionality, combined with robust categorization and tagging systems, ensures that users can quickly locate the information they need, fostering efficiency and productivity in their projects.

Furthermore, Project Hub's emphasis on collaboration is truly transformative. By allowing users to create, manage, and collaborate on projects in real time, the platform promotes synergy and enhances the quality of outcomes. This fosters an environment where individuals from diverse backgrounds and expertise can come together, share ideas, and collectively achieve their goals.

The scalability and adaptability of Project Hub are equally commendable. With features that cater to a wide range of users, from students and researchers to professionals and organizations, the platform is poised to become an indispensable tool for countless projects across various domains. Its support for multimedia content, documentation, and version control ensures that it can accommodate a diverse array of project types and formats.

In summary, Project Hub has set a new standard for project libraries by seamlessly merging convenience, collaboration, and security. Its user-centric design, emphasis on collaboration, and adaptability make it a versatile resource for anyone looking to embark on successful projects. As we look ahead to the future, Project Hub's potential for growth and positive impact is boundless. It is poised to empower individuals, teams, and organizations to turn their visions into reality while fostering a global community of knowledge-sharing and collaboration. Project Hub is not just a website; it's a gateway to limitless possibilities.

A white and blue background with black text

Description automatically generated

**References:**

[1] "Research Methods in Education" by Louis Cohen, Lawrence Manion, and Keith Morrison

* A comprehensive guide to research methodologies in education is essential for students and researchers.

[2]"The Craft of Research" by Wayne C. Booth, Gregory G. Colomb, and Joseph M. Williams

* An indispensable resource for mastering the art of research and writing in various academic disciplines.

[3]"Designing Qualitative Research" by Catherine Marshall and Gretchen B. Rossman

* This book provides valuable insights into conducting qualitative research and designing effective studies.

[4]"Statistics for Social Sciences" by R. Mark Sirkin

* A practical introduction to statistical analysis for social science research, with clear explanations and examples.

[5]"The Elements of Style" by William Strunk Jr. and E.B. White

* A timeless guide to writing with clarity and precision, a must-have for anyone working on research papers.

**URLs:**

[www.google.com](http://www.google.com)

<https://www.youtube.com>

<https://www.geeksforgeeks.org>

[www.w3schools.com](http://www.w3schools.com)

<https://www.tutorialspoint.com>

<https://github.com>

<https://uoda.edu.bd>

<https://scholar.google.com>

<https://dl.acm.org>

<https://ieeexplore.ieee.org/Xplore/home.jsp>

A blue and white background with black text

Description automatically generated

**Database Table:**

A screenshot of a computer

Description automatically generated

**Sign up:**

A screenshot of a login form

Description automatically generated

**Sign in:**

A screenshot of a login form

Description automatically generated

**Forget Password:**

A screenshot of a login form

Description automatically generated

**Profile:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Successful Token:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Dashboards:**

A screenshot of a video game

Description automatically generated

**Add Project:**

A screenshot of a computer

Description automatically generated

**Add Project successful Token:**

A screenshot of a computer screen

Description automatically generated

**List of Project:**

A screenshot of a computer

Description automatically generated

**Search Project:**

A screenshot of a computer

Description automatically generated

A blue and white background with black text

Description automatically generated

**Config.php:**

**<?php**

**session\_start();**

**error\_reporting(E\_ALL);**

**ini\_set('display\_errors', 1);**

**// Database configuration**

**$host = "localhost";**

**$username = "root";**

**$password = "";**

**$database = "uv\_project";**

**try {**

**// Create a PDO connection**

**$pdo = new PDO("mysql:host=$host;dbname=$database", $username, $password);**

**// Set PDO attributes**

**$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);**

**} catch (PDOException $e) {**

**die("Connection failed: " . $e->getMessage());**

**}**

**include 'lib/function.php';**

**Index.php:**

**<?php**

**include 'config.php';**

**auth\_check();**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div id="layout-wrapper">**

**<?php include('layouts/top-bar.php'); ?>**

**<!-- ========== App Menu ========== -->**

**<?php include('layouts/left-bar.php'); ?>**

**<!-- ============================================================== -->**

**<!-- Start right Content here -->**

**<!-- ============================================================== -->**

**<div class="main-content">**

**<div class="page-content">**

**<div class="container-fluid">**

**<!-- start page title -->**

**<div class="row">**

**<div class="col-12">**

**<div class="page-title-box d-sm-flex align-items-center justify-content-between">**

**<h4 class="mb-sm-0">PH Dashboard</h4>**

**<div class="page-title-right">**

**<ol class="breadcrumb m-0">**

**<li class="breadcrumb-item"><a href="javascript: void(0);">Dashboards</a></li>**

**<li class="breadcrumb-item active">Dashboard</li>**

**</ol>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="row">**

**<div class="col-xl-6">**

**<div class="card overflow-hidden">**

**<div class="card-body bg-marketplace d-flex">**

**<div class="flex-grow-1">**

**<?php**

**$stmt = $pdo->prepare("SELECT \* FROM projects ORDER BY id DESC limit 1");**

**$stmt->execute();**

**$latest\_project = $stmt->fetch(PDO::FETCH\_ASSOC);**

**?>**

**<h4 class="fs-18 lh-base mb-0">Latest Project (<?= $latest\_project['project\_name'] ?>)</h4>**

**<p class="mb-0 mt-2 pt-1 text-muted">Project Supervisor: <br> <?= $latest\_project['project\_supervisor'] ?></p>**

**<div class="d-flex gap-3 mt-4">**

**<a href="project-view.php?id=<?= $project['id'] ?>" class="btn btn-primary">View</a>**

**<a href="#" class="btn btn-soft-primary">Create Your Own</a>**

**</div>**

**</div>**

**<img src="assets/images/bg-d.png" alt="" class="img-fluid">**

**</div>**

**</div>**

**</div><!--end col-->**

**<div class="col-xl-3 col-md-6">**

**<div class="card card-height-100">**

**<div class="card-body">**

**<div class="d-flex align-items-center">**

**<div class="avatar-sm flex-shrink-0">**

**<span class="avatar-title bg-soft-primary rounded fs-3">**

**<i class="bx bx-wallet text-primary"></i>**

**</span>**

**</div>**

**<div class="flex-grow-1 ps-3">**

**<h5 class="text-muted text-uppercase fs-13 mb-0">Total Project</h5>**

**</div>**

**</div>**

**<div class="mt-4 pt-1">**

**<?php**

**$stmt = $pdo->prepare("SELECT COUNT(\*) AS project\_count FROM projects");**

**$stmt->execute();**

**$result = $stmt->fetch(PDO::FETCH\_ASSOC);**

**$projectCount = $result['project\_count'];**

**?>**

**<h4 class="fs-22 fw-semibold ff-secondary mb-0"><i class="bx bx-wallet text-primary"></i><span class="counter-value" data-target="<?= $projectCount ?>">0</span> </h4>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="col-xl-3 col-md-6">**

**<div class="card card-height-100">**

**<div class="card-body">**

**<div class="d-flex align-items-center">**

**<div class="avatar-sm flex-shrink-0">**

**<span class="avatar-title bg-soft-primary rounded fs-3">**

**<i class="bx bx-dollar-circle text-primary"></i>**

**</span>**

**</div>**

**<div class="flex-grow-1 ps-3">**

**<h5 class="text-muted text-uppercase fs-13 mb-0">Total Student</h5>**

**</div>**

**</div>**

**<div class="mt-4 pt-1">**

**<?php**

**$stmt = $pdo->prepare("SELECT COUNT(\*) AS student\_count FROM project\_student");**

**$stmt->execute();**

**$result = $stmt->fetch(PDO::FETCH\_ASSOC);**

**$projectStudents = $result['student\_count'];**

**?>**

**<h4 class="fs-22 fw-semibold ff-secondary mb-0">$<span class="counter-value" data-target="<?= $projectStudents ?>">0</span> </h4>**

**</div>**

**</div>**

**</div>**

**</div><!--end col-->**

**</div>**

**</div>**

**<!-- container-fluid -->**

**</div>**

**<!-- End Page-content -->**

**</div>**

**<!-- end main content-->**

**</div>**

**<!-- END layout-wrapper -->**

**<!--start back-to-top-->**

**<button onclick="topFunction()" class="btn btn-primary btn-icon" id="back-to-top">**

**<i class="ri-arrow-up-line"></i>**

**</button>**

**<!--end back-to-top-->**

**<!--preloader-->**

**<div id="preloader">**

**<div id="status">**

**<div class="spinner-border text-primary avatar-sm" role="status">**

**<span class="visually-hidden">Loading...</span>**

**</div>**

**</div>**

**</div>**

**<!-- <div class="customizer-setting d-none d-md-block">**

**<div class="btn-primary btn-rounded shadow-lg btn btn-icon btn-lg p-2" data-bs-toggle="offcanvas" data-bs-target="#theme-settings-offcanvas" aria-controls="theme-settings-offcanvas">**

**<i class='mdi mdi-spin mdi-cog-outline fs-22'></i>**

**</div>**

**</div> -->**

**<!-- Theme Settings -->**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**</body>**

**</html>**

**Registration.php:**

**<?php**

**include 'config.php';**

**if (count($\_POST) > 0) {**

**$inputData = array(**

**"email" => $\_POST['email'],**

**"username" => $\_POST['username'],**

**"password" => md5($\_POST['password']),**

**// "batch" => '52',**

**"created\_at" => date('Y-m-d H:i:s')**

**);**

**$stmt = $pdo->prepare("INSERT INTO users (email, name, password, created\_at) VALUES (:email, :username, :password, :created\_at)");**

**$stmt->bindParam(':email', $inputData['email']);**

**$stmt->bindParam(':username', $inputData['username']);**

**$stmt->bindParam(':password', $inputData['password']);**

**// $stmt->bindParam(':batch', $inputData['batch']);**

**$stmt->bindParam(':created\_at', $inputData['created\_at']);**

**if ($stmt->execute()) {**

**header('location: login.php');**

**}**

**}**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div class="auth-page-wrapper pt-5">**

**<!-- auth page content -->**

**<div class="auth-page-content">**

**<div class="container">**

**<div class="row justify-content-center">**

**<div class="col-md-8 col-lg-6 col-xl-5">**

**<div class="card mt-4 card-bg-fill">**

**<div class="card-body p-4">**

**<div class="text-center mt-2">**

**<h5 class="text-primary">Create New Account</h5>**

**<p class="text-muted">Get your free Project Hub account now</p>**

**</div>**

**<div class="p-2 mt-4">**

**<form class="needs-validation" novalidate action="register.php" method="post">**

**<div class="mb-3">**

**<label for="useremail" class="form-label">Email <span class="text-danger">\*</span></label>**

**<input type="email" class="form-control" name="email" id="useremail" placeholder="Enter email address" required>**

**<div class="invalid-feedback">**

**Please enter email**

**</div>**

**</div>**

**<div class="mb-3">**

**<label for="username" class="form-label">Username <span class="text-danger">\*</span></label>**

**<input type="text" class="form-control" name="username" id="username" placeholder="Enter username" required>**

**<div class="invalid-feedback">**

**Please enter username**

**</div>**

**</div>**

**<div class="mb-3">**

**<label class="form-label" for="password-input">Password</label>**

**<div class="position-relative auth-pass-inputgroup">**

**<input type="password" name="password" class="form-control pe-5 password-input" onpaste="return false" placeholder="Enter password"**

**id="password-input" aria-describedby="passwordInput" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" required>**

**<button class="btn btn-link position-absolute end-0 top-0 text-decoration-none text-muted password-addon" type="button" id="password-addon"><i class="ri-eye-fill align-middle"></i></button>**

**<div class="invalid-feedback">**

**Please enter password**

**</div>**

**</div>**

**</div>**

**<div class="mb-4">**

**<p class="mb-0 fs-12 text-muted fst-italic">By registering you agree to the Velzon <a href="#" class="text-primary text-decoration-underline fst-normal fw-medium">Terms of Use</a></p>**

**</div>**

**<div id="password-contain" class="p-3 bg-light mb-2 rounded">**

**<h5 class="fs-13">Password must contain:</h5>**

**<p id="pass-length" class="invalid fs-12 mb-2">Minimum <b>8 characters</b></p>**

**<p id="pass-lower" class="invalid fs-12 mb-2">At <b>lowercase</b> letter (a-z)</p>**

**<p id="pass-upper" class="invalid fs-12 mb-2">At least <b>uppercase</b> letter (A-Z)</p>**

**<p id="pass-number" class="invalid fs-12 mb-0">A least <b>number</b> (0-9)</p>**

**</div>**

**<div class="mt-4">**

**<button class="btn btn-success w-100" type="submit">Sign Up</button>**

**</div>**

**<div class="mt-4 text-center">**

**<div class="signin-other-title">**

**<h5 class="fs-13 mb-4 title text-muted">Create account with</h5>**

**</div>**

**<div>**

**<button type="button" class="btn btn-primary btn-icon waves-effect waves-light"><i class="ri-facebook-fill fs-16"></i></button>**

**<button type="button" class="btn btn-danger btn-icon waves-effect waves-light"><i class="ri-google-fill fs-16"></i></button>**

**<button type="button" class="btn btn-dark btn-icon waves-effect waves-light"><i class="ri-github-fill fs-16"></i></button>**

**<button type="button" class="btn btn-info btn-icon waves-effect waves-light"><i class="ri-twitter-fill fs-16"></i></button>**

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**<!-- end card body -->**

**</div>**

**<!-- end card -->**

**<div class="mt-4 text-center">**

**<p class="mb-0">Already have an account ? <a href="login.php" class="fw-semibold text-primary text-decoration-underline"> Signin </a> </p>**

**</div>**

**</div>**

**</div>**

**<!-- end row -->**

**</div>**

**<!-- end container -->**

**</div>**

**<!-- end auth page content -->**

**</div>**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**</body>**

**</html>**

**Log in.php:**

**<?php**

**include 'config.php';**

**if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {**

**$email = $\_POST['email'];**

**$password = md5($\_POST['password']);**

**// Basic validation**

**if (empty($email) || empty($password)) {**

**$error = "Both email and password are required.";**

**} else {**

**$stmt = $pdo->prepare("SELECT \* FROM users WHERE email = :email AND password = :password");**

**$stmt->bindParam(':email', $email);**

**$stmt->bindParam(':password', $password);**

**$stmt->execute();**

**if ($stmt->rowCount() > 0) {**

**$user = $stmt->fetch(PDO::FETCH\_ASSOC);**

**// You can set a session or redirect to a dashboard page**

**session\_start();**

**$\_SESSION['user\_id'] = $user['id'];**

**$\_SESSION['is\_auth'] = true;**

**header('location: index.php');**

**exit();**

**} else {**

**$error = "Invalid email or password.";**

**}**

**}**

**}**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div class="auth-page-wrapper pt-5">**

**<!-- auth page content -->**

**<div class="auth-page-content">**

**<div class="container">**

**<div class="row justify-content-center">**

**<div class="col-md-8 col-lg-6 col-xl-5">**

**<div class="card mt-4 card-bg-fill">**

**<div class="card-body p-4">**

**<div class="text-center mt-2">**

**<h5 class="text-primary">Welcome Back !</h5>**

**<p class="text-muted">Sign in to continue to The Project Hub.</p>**

**</div>**

**<div class="p-2 mt-4">**

**<form action="" method="post">**

**<div class="mb-3">**

**<label for="username" class="form-label">Email</label>**

**<input type="text" class="form-control" name="email" id="username" placeholder="Enter email">**

**</div>**

**<div class="mb-3">**

**<div class="float-end">**

**<a href="forgot-password.php" class="text-muted">Forgot password?</a>**

**</div>**

**<label class="form-label" for="password-input">Password</label>**

**<div class="position-relative auth-pass-inputgroup mb-3">**

**<input type="password" class="form-control pe-5 password-input" name="password" placeholder="Enter password" id="password-input">**

**<button class="btn btn-link position-absolute end-0 top-0 text-decoration-none text-muted password-addon" type="button" id="password-addon"><i class="ri-eye-fill align-middle"></i></button>**

**</div>**

**</div>**

**<div class="form-check">**

**<input class="form-check-input" type="checkbox" value="" id="auth-remember-check">**

**<label class="form-check-label" for="auth-remember-check">Remember me</label>**

**</div>**

**<div class="mt-4">**

**<button class="btn btn-primary w-100" type="submit">Sign In</button>**

**</div>**

**<div class="mt-4 text-center">**

**<div class="signin-other-title">**

**<h5 class="fs-13 mb-4 title">Sign In with</h5>**

**</div>**

**<div>**

**<button type="button" class="btn btn-primary btn-icon waves-effect waves-light"><i class="ri-facebook-fill fs-16"></i></button>**

**<button type="button" class="btn btn-danger btn-icon waves-effect waves-light"><i class="ri-google-fill fs-16"></i></button>**

**<button type="button" class="btn btn-dark btn-icon waves-effect waves-light"><i class="ri-github-fill fs-16"></i></button>**

**<button type="button" class="btn btn-info btn-icon waves-effect waves-light"><i class="ri-twitter-fill fs-16"></i></button>**

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**<!-- end card body -->**

**</div>**

**<!-- end card -->**

**<div class="mt-4 text-center">**

**<p class="mb-0">Don't have an account ? <a href="register.php" class="fw-semibold text-primary text-decoration-underline"> Signup </a> </p>**

**</div>**

**</div>**

**</div>**

**<!-- end row -->**

**</div>**

**<!-- end container -->**

**</div>**

**</div>**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**</body>**

**</html>**

**Log out.php:**

**<?php**

**session\_start();**

**if (isset($\_SESSION['is\_auth'])) {**

**unset($\_SESSION['is\_auth']);**

**header('location: login.php');**

**}**

**Project add.php:**

**<?php**

**include 'config.php';**

**auth\_check();**

**if ($\_SERVER["REQUEST\_METHOD"] == "POST") {**

**$validationMessages = array(); // Store validation messages**

**// Validate and sanitize the input data**

**$project\_name = isset($\_POST["project\_name"]) ? validate\_input($\_POST["project\_name"]) : "";**

**$batch = isset($\_POST["batch"]) ? (int) validate\_input($\_POST["batch"]) : "";**

**$project\_link = isset($\_POST["project\_link"]) ? validate\_input($\_POST["project\_link"]) : "";**

**$project\_supervisor = isset($\_POST["project\_supervisor"]) ? validate\_input($\_POST["project\_supervisor"]) : "";**

**$project\_book = 'test';**

**$created\_at = date('Y-m-d H:i:s');**

**$students = isset($\_POST["student\_name"]) ? $\_POST["student\_name"] : array();**

**$reg\_no = isset($\_POST["reg\_no"]) ? $\_POST["reg\_no"] : array();**

**$roll\_no = isset($\_POST["roll\_no"]) ? $\_POST["roll\_no"] : array();**

**// Validate each input field and store validation messages**

**if ($\_SERVER["REQUEST\_METHOD"] == "POST") {**

**if (empty($project\_name)) {**

**$validationMessages[] = "Project Name is required.";**

**}**

**if (empty($batch)) {**

**$validationMessages[] = "Batch is required.";**

**}**

**if (empty($students)) {**

**$validationMessages[] = "At least one Student Name is required.";**

**}**

**if (empty($reg\_no)) {**

**$validationMessages[] = "Registration No is required.";**

**}**

**if (empty($roll\_no)) {**

**$validationMessages[] = "Roll No is required.";**

**}**

**if (empty($project\_link)) {**

**$validationMessages[] = "Project Link is required.";**

**}**

**if (empty($project\_supervisor)) {**

**$validationMessages[] = "Project Supervisor is required.";**

**}**

**if (empty($validationMessages)) {**

**try {**

**// Insert data into the MySQL database using prepared statements**

**$stmt = $pdo->prepare("INSERT INTO projects (project\_name, batch, project\_link, project\_supervisor, project\_book, created\_at) VALUES (:project\_name, :batch, :project\_link, :project\_supervisor, :project\_book, :created\_at)");**

**$stmt->bindParam(':project\_name', $project\_name);**

**$stmt->bindParam(':batch', $batch);**

**$stmt->bindParam(':project\_link', $project\_link);**

**$stmt->bindParam(':project\_supervisor', $project\_supervisor);**

**$stmt->bindParam(':project\_book', $project\_book);**

**$stmt->bindParam(':created\_at', $created\_at);**

**$stmt->execute();**

**$project\_id = $pdo->lastInsertId(); // Get the inserted project ID**

**// Insert student names into a separate table, assuming you have a table named 'project\_student'**

**for ($i = 0; $i <= count($students) - 1; $i++) {**

**$student\_name = validate\_input($students[$i]);**

**$student\_roll = (int) validate\_input($roll\_no[$i]);**

**$student\_reg = (int) validate\_input($reg\_no[$i]);**

**$stmt = $pdo->prepare("INSERT INTO project\_student (project\_id, batch, reg\_no, roll\_no, student\_name, created\_at) VALUES (:project\_id, :batch, :reg\_no, :roll\_no, :student\_name, :created\_at)");**

**$stmt->bindParam(':project\_id', $project\_id);**

**$stmt->bindParam(':student\_name', $student\_name);**

**$stmt->bindParam(':batch', $batch);**

**$stmt->bindParam(':reg\_no', $student\_reg);**

**$stmt->bindParam(':roll\_no', $student\_roll);**

**$stmt->bindParam(':created\_at', $created\_at);**

**$stmt->execute();**

**}**

**$data\_insert = 'success';**

**} catch (PDOException $e) {**

**echo "Error: " . $e->getMessage();**

**}**

**}**

**}**

**}**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**<link rel="stylesheet" href="assets/libs/dropzone/dropzone.css" type="text/css" />**

**<link href="assets/libs/sweetalert2/sweetalert2.min.css" rel="stylesheet" type="text/css" />**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div id="layout-wrapper">**

**<?php include('layouts/top-bar.php'); ?>**

**<!-- ========== App Menu ========== -->**

**<?php include('layouts/left-bar.php'); ?>**

**<!-- ============================================================== -->**

**<!-- Start right Content here -->**

**<!-- ============================================================== -->**

**<div class="main-content">**

**<div class="page-content">**

**<div class="container-fluid">**

**<!-- start page title -->**

**<div class="row">**

**<div class="col-12">**

**<div class="page-title-box d-sm-flex align-items-center justify-content-between">**

**<h4 class="mb-sm-0">PH Dashboard</h4>**

**<div class="page-title-right">**

**<ol class="breadcrumb m-0">**

**<li class="breadcrumb-item"><a href="javascript: void(0);">Dashboards</a></li>**

**<li class="breadcrumb-item active">Dashboard</li>**

**</ol>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="row">**

**<div class="col-lg-12">**

**<div class="card">**

**<div class="card-header align-items-center d-flex">**

**<h4 class="card-title mb-0 flex-grow-1">Add Project Details</h4>**

**</div>**

**<div class="card-body">**

**<form action="project-add.php" method="post" enctype="multipart/form-data">**

**<div class="live-preview">**

**<?php**

**// Display validation messages at the bottom**

**if (!empty($validationMessages)) {**

**echo "<div class='error-messages'>";**

**echo "<ul>";**

**foreach ($validationMessages as $message) {**

**echo "<li>$message</li>";**

**}**

**echo "</ul>";**

**echo "</div>";**

**}**

**?>**

**<div class="row align-items-center g-3 pb-3">**

**<div class="col-md-6">**

**<div class="form-floating">**

**<input type="text" class="form-control" id="project\_name" name="project\_name" placeholder="Enter project Name">**

**<label for="project\_name">Project Name</label>**

**</div>**

**</div>**

**<div class="col-md-3">**

**<div class="form-floating">**

**<input type="number" class="form-control" id="batch" name="batch" placeholder="Enter batch">**

**<label for="batch">Batch</label>**

**</div>**

**</div>**

**</div>**

**<div class="row align-items-center g-3 pb-3 mt-3">**

**<div class="row student\_row mb-2">**

**<div class="col-12 col-md-4">**

**<div class="form-floating">**

**<input type="text" class="form-control" id="student\_name" name="student\_name[]" placeholder="Enter student name">**

**<label for="student\_name">Student Name</label>**

**</div>**

**</div>**

**<div class="col-12 col-md-3">**

**<div class="form-floating">**

**<input type="number" class="form-control" id="reg\_no" name="reg\_no[]" placeholder="Enter registration no">**

**<label for="reg\_no">Registration No</label>**

**</div>**

**</div>**

**<div class="col-12 col-md-3">**

**<div class="form-floating">**

**<input type="number" class="form-control" id="roll\_no" name="roll\_no[]" placeholder="Enter roll no">**

**<label for="roll\_no">Roll No</label>**

**</div>**

**</div>**

**<div class="col-12 col-md-2">**

**<div class="form-floating">**

**<button type="button" class="btn btn-danger remove-section">-</button>**

**</div>**

**</div>**

**</div>**

**<div class="col-12">**

**<button type="button" class="btn btn-success add-section">+</button>**

**</div>**

**</div>**

**<div class="row align-items-center g-3 pb-3">**

**<div class="col-md-6">**

**<div class="form-floating">**

**<input type="text" class="form-control" id="project\_link" name="project\_link" placeholder="Enter project Link">**

**<label for="project\_link">Project Link</label>**

**</div>**

**</div>**

**<div class="col-md-6">**

**<div class="form-floating">**

**<input type="text" class="form-control" id="project\_supervisor" name="project\_supervisor" placeholder="Enter project supervisor">**

**<label for="project\_supervisor">Project supervisor</label>**

**</div>**

**</div>**

**</div>**

**<div class="row align-items-center g-3 mb-3">**

**<div class="col-12 col-sm-6">**

**<div class="dropzone">**

**<label for="project\_file">Project Book</label>**

**<div class="fallback">**

**<input name="project\_book" type="file" multiple="multiple">**

**</div>**

**<div class="dz-message needsclick">**

**<div class="mb-3">**

**<i class="display-4 text-muted ri-upload-cloud-2-fill"></i>**

**</div>**

**<h4>Drop files here or click to upload.</h4>**

**</div>**

**</div>**

**<ul class="list-unstyled mb-0" id="dropzone-preview">**

**<li class="mt-2" id="dropzone-preview-list">**

**<!-- This is used as the file preview template -->**

**<div class="border rounded">**

**<div class="d-flex p-2">**

**<div class="flex-shrink-0 me-3">**

**<div class="avatar-sm bg-light rounded">**

**<img data-dz-thumbnail class="img-fluid rounded d-block" src="assets/images/new-document.png" alt="Dropzone-Image" />**

**</div>**

**</div>**

**<div class="flex-grow-1">**

**<div class="pt-1">**

**<h5 class="fs-14 mb-1" data-dz-name>&nbsp;</h5>**

**<p class="fs-13 text-muted mb-0" data-dz-size></p>**

**<strong class="error text-danger" data-dz-errormessage></strong>**

**</div>**

**</div>**

**<div class="flex-shrink-0 ms-3">**

**<button data-dz-remove class="btn btn-sm btn-danger">Delete</button>**

**</div>**

**</div>**

**</div>**

**</li>**

**</ul>**

**</div>**

**</div>**

**<div class="col-12">**

**<button class="btn btn-primary" type="submit">Submit form</button>**

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**</div>**

**<!--end col-->**

**</div>**

**</div>**

**<!-- container-fluid -->**

**</div>**

**<!-- End Page-content -->**

**</div>**

**<!-- end main content-->**

**</div>**

**<!-- END layout-wrapper -->**

**<!--start back-to-top-->**

**<button onclick="topFunction()" class="btn btn-primary btn-icon" id="back-to-top">**

**<i class="ri-arrow-up-line"></i>**

**</button>**

**<!--end back-to-top-->**

**<!--preloader-->**

**<div id="preloader">**

**<div id="status">**

**<div class="spinner-border text-primary avatar-sm" role="status">**

**<span class="visually-hidden">Loading...</span>**

**</div>**

**</div>**

**</div>**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**<script src="assets/js/jquery.js"></script>**

**<script src="assets/libs/dropzone/dropzone-min.js"></script>**

**<script src="assets/js/pages/form-file-upload.init.js"></script>**

**<script src="assets/libs/sweetalert2/sweetalert2.min.js"></script>**

**<script>**

**$(document).ready(function() {**

**// Add Section**

**$(".add-section").click(function() {**

**var sectionClone = $(".row.student\_row").first().clone();**

**sectionClone.find("input").val(""); // Clear input values**

**$(".row.student\_row").last().after(sectionClone);**

**});**

**// Remove Section**

**$(document).on("click", ".remove-section", function() {**

**if ($(".row.student\_row").length > 1) {**

**$(this).closest(".row.student\_row").remove();**

**}**

**});**

**<?php**

**if (isset($data\_insert)) { ?>**

**Swal.fire({**

**html: '<div class="mt-3"><lord-icon src="https://cdn.lordicon.com/lupuorrc.json" trigger="loop" colors="primary:#0ab39c,secondary:#405189" style="width:120px;height:120px"></lord-icon><div class="mt-4 pt-2 fs-15"><h4>Well done !</h4><p class="text-muted mx-4 mb-0">Project has been listed to database.</p></div></div>',**

**showCancelButton: !0,**

**showConfirmButton: !1,**

**cancelButtonClass: "btn btn-primary w-xs mb-1",**

**cancelButtonText: "Back",**

**buttonsStyling: !1,**

**showCloseButton: !0**

**})**

**<?php } ?>**

**});**

**</script>**

**</body>**

**</html>**

**Project list.php:**

**<?php**

**include 'config.php';**

**auth\_check();**

**$stmt = $pdo->prepare("SELECT batch from projects group by batch");**

**$stmt->execute();**

**$batch = $stmt->fetchAll(PDO::FETCH\_ASSOC);**

**$project\_query = 'SELECT projects.\*,COUNT(project\_student.project\_id) as total\_student from projects join project\_student on projects.id = project\_student.project\_id';**

**if (isset($\_GET['search\_text']) && !empty($\_GET['search\_text'])) {**

**$search\_text = $\_GET['search\_text'];**

**$project\_query .= " AND projects.project\_name like '%$search\_text%'";**

**}**

**if (isset($\_GET['batch']) && !empty($\_GET['batch'])) {**

**$project\_batch = $\_GET['batch'];**

**$project\_query .= " AND projects.batch = '$project\_batch'";**

**}**

**$project\_query .= " GROUP by project\_student.project\_id";**

**$stmt = $pdo->prepare($project\_query);**

**$stmt->execute();**

**$projects = $stmt->fetchAll(PDO::FETCH\_ASSOC);**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**<style>**

**.pr-3 {**

**margin-right: 15px;**

**}**

**</style>**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div id="layout-wrapper">**

**<?php include('layouts/top-bar.php'); ?>**

**<!-- ========== App Menu ========== -->**

**<?php include('layouts/left-bar.php'); ?>**

**<!-- ============================================================== -->**

**<!-- Start right Content here -->**

**<!-- ============================================================== -->**

**<div class="main-content">**

**<div class="page-content">**

**<div class="container-fluid">**

**<!-- start page title -->**

**<div class="row">**

**<div class="col-12">**

**<div class="page-title-box d-sm-flex align-items-center justify-content-between">**

**<h4 class="mb-sm-0">PH Dashboard</h4>**

**<div class="page-title-right">**

**<ol class="breadcrumb m-0">**

**<li class="breadcrumb-item"><a href="javascript: void(0);">Dashboards</a></li>**

**<li class="breadcrumb-item active">Dashboard</li>**

**</ol>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="row">**

**<div class="col-xl-12">**

**<div class="card">**

**<div class="card-header align-items-center d-flex">**

**<h4 class="card-title mb-0 flex-grow-1">Project List's</h4>**

**</div><!-- end card header -->**

**<div class="card-body">**

**<div class="live-preview">**

**<div class="table-responsive table-card">**

**<table class="table align-middle table-nowrap mb-0">**

**<thead class="table-light">**

**<tr>**

**<th scope="col" style="width: 46px;">**

**<div class="form-check">**

**<input class="form-check-input" type="checkbox" value="" id="cardtableCheck">**

**<label class="form-check-label" for="cardtableCheck"></label>**

**</div>**

**</th>**

**<th scope="col">ID</th>**

**<th scope="col">Name</th>**

**<th scope="col">Batch</th>**

**<th scope="col">Project Supervisor</th>**

**<th scope="col">Created at</th>**

**<th scope="col" style="width: 150px;">Action</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr>**

**<td colspan="7">**

**<form action="project-list.php" method="get">**

**<div class="row">**

**<div class="col-12 d-sm-flex">**

**<div class="form-group pr-3">**

**<input type="text" class="form-controll h-100 form-controll-lg" name="search\_text"**

**value="<?php (isset($\_GET['search\_text'])) ? print($\_GET['search\_text']) : '' ;?>"**

**placeholder="Enter Search key">**

**</div>**

**<div class="form-group pr-3">**

**<select name="batch" id="batch" class="foro-controll h-100">**

**<option value="">select batch</option>**

**<?php foreach ($batch as $key => $value) { ?>**

**<option value="<?= $value['batch']; ?>" <?php (isset($\_GET['batch']) && $\_GET['batch'] == $value['batch']) ? print('selected'): '';?>><?= $value['batch']; ?></option>**

**<?php } ?>**

**</select>**

**</div>**

**<button type="submit" class="btn btn-primary">**

**<i class=" las la-search"></i>**

**</button>**

**</div>**

**</div>**

**</form>**

**</td>**

**</tr>**

**<?php if (count($projects) > 0) {**

**foreach ($projects as $key => $value) { ?>**

**<tr>**

**<td>**

**<input type="checkbox" name="delete\_id" id="delete\_id" value="<?= $value['id'] ?>">**

**</td>**

**<td><?= $value['id'] ?></td>**

**<td><?= $value['project\_name'] ?></td>**

**<td><?= $value['batch'] ?></td>**

**<td><?= $value['project\_supervisor'] ?></td>**

**<td>**

**<?php**

**$originalDate = $value['created\_at'];**

**$formattedDate = date('d F, Y', strtotime($originalDate));**

**echo $formattedDate;**

**?>**

**</td>**

**<td>**

**<a href="project-view.php?id=<?= $value['id'] ?>" class="btn btn-sm btn-light">Details</a>**

**</td>**

**</tr>**

**<?php }**

**} ?>**

**</tbody>**

**</table>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**<!-- container-fluid -->**

**</div>**

**<!-- End Page-content -->**

**</div>**

**<!-- end main content-->**

**</div>**

**<!-- END layout-wrapper -->**

**<!--start back-to-top-->**

**<button onclick="topFunction()" class="btn btn-primary btn-icon" id="back-to-top">**

**<i class="ri-arrow-up-line"></i>**

**</button>**

**<!--end back-to-top-->**

**<!--preloader-->**

**<div id="preloader">**

**<div id="status">**

**<div class="spinner-border text-primary avatar-sm" role="status">**

**<span class="visually-hidden">Loading...</span>**

**</div>**

**</div>**

**</div>**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**</body>**

**</html>** **<?php**

**include 'config.php';**

**auth\_check();**

**$stmt = $pdo->prepare("SELECT batch from projects group by batch");**

**$stmt->execute();**

**$batch = $stmt->fetchAll(PDO::FETCH\_ASSOC);**

**$project\_query = 'SELECT projects.\*,COUNT(project\_student.project\_id) as total\_student from projects join project\_student on projects.id = project\_student.project\_id';**

**if (isset($\_GET['search\_text']) && !empty($\_GET['search\_text'])) {**

**$search\_text = $\_GET['search\_text'];**

**$project\_query .= " AND projects.project\_name like '%$search\_text%'";**

**}**

**if (isset($\_GET['batch']) && !empty($\_GET['batch'])) {**

**$project\_batch = $\_GET['batch'];**

**$project\_query .= " AND projects.batch = '$project\_batch'";**

**}**

**$project\_query .= " GROUP by project\_student.project\_id";**

**$stmt = $pdo->prepare($project\_query);**

**$stmt->execute();**

**$projects = $stmt->fetchAll(PDO::FETCH\_ASSOC);**

**?>**

**<!doctype html>**

**<html lang="en" data-layout="vertical" data-topbar="light" data-sidebar="dark" data-sidebar-size="lg" data-sidebar-image="none" data-layout-mode="light" data-body-image="img-1" data-preloader="disable">**

**<head>**

**<?php include('layouts/head.php'); ?>**

**<style>**

**.pr-3 {**

**margin-right: 15px;**

**}**

**</style>**

**</head>**

**<body>**

**<!-- Begin page -->**

**<div id="layout-wrapper">**

**<?php include('layouts/top-bar.php'); ?>**

**<!-- ========== App Menu ========== -->**

**<?php include('layouts/left-bar.php'); ?>**

**<!-- ============================================================== -->**

**<!-- Start right Content here -->**

**<!-- ============================================================== -->**

**<div class="main-content">**

**<div class="page-content">**

**<div class="container-fluid">**

**<!-- start page title -->**

**<div class="row">**

**<div class="col-12">**

**<div class="page-title-box d-sm-flex align-items-center justify-content-between">**

**<h4 class="mb-sm-0">PH Dashboard</h4>**

**<div class="page-title-right">**

**<ol class="breadcrumb m-0">**

**<li class="breadcrumb-item"><a href="javascript: void(0);">Dashboards</a></li>**

**<li class="breadcrumb-item active">Dashboard</li>**

**</ol>**

**</div>**

**</div>**

**</div>**

**</div>**

**<div class="row">**

**<div class="col-xl-12">**

**<div class="card">**

**<div class="card-header align-items-center d-flex">**

**<h4 class="card-title mb-0 flex-grow-1">Project List's</h4>**

**</div><!-- end card header -->**

**<div class="card-body">**

**<div class="live-preview">**

**<div class="table-responsive table-card">**

**<table class="table align-middle table-nowrap mb-0">**

**<thead class="table-light">**

**<tr>**

**<th scope="col" style="width: 46px;">**

**<div class="form-check">**

**<input class="form-check-input" type="checkbox" value="" id="cardtableCheck">**

**<label class="form-check-label" for="cardtableCheck"></label>**

**</div>**

**</th>**

**<th scope="col">ID</th>**

**<th scope="col">Name</th>**

**<th scope="col">Batch</th>**

**<th scope="col">Project Supervisor</th>**

**<th scope="col">Created at</th>**

**<th scope="col" style="width: 150px;">Action</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr>**

**<td colspan="7">**

**<form action="project-list.php" method="get">**

**<div class="row">**

**<div class="col-12 d-sm-flex">**

**<div class="form-group pr-3">**

**<input type="text" class="form-controll h-100 form-controll-lg" name="search\_text"**

**value="<?php (isset($\_GET['search\_text'])) ? print($\_GET['search\_text']) : '' ;?>"**

**placeholder="Enter Search key">**

**</div>**

**<div class="form-group pr-3">**

**<select name="batch" id="batch" class="foro-controll h-100">**

**<option value="">select batch</option>**

**<?php foreach ($batch as $key => $value) { ?>**

**<option value="<?= $value['batch']; ?>" <?php (isset($\_GET['batch']) && $\_GET['batch'] == $value['batch']) ? print('selected'): '';?>><?= $value['batch']; ?></option>**

**<?php } ?>**

**</select>**

**</div>**

**<button type="submit" class="btn btn-primary">**

**<i class=" las la-search"></i>**

**</button>**

**</div>**

**</div>**

**</form>**

**</td>**

**</tr>**

**<?php if (count($projects) > 0) {**

**foreach ($projects as $key => $value) { ?>**

**<tr>**

**<td>**

**<input type="checkbox" name="delete\_id" id="delete\_id" value="<?= $value['id'] ?>">**

**</td>**

**<td><?= $value['id'] ?></td>**

**<td><?= $value['project\_name'] ?></td>**

**<td><?= $value['batch'] ?></td>**

**<td><?= $value['project\_supervisor'] ?></td>**

**<td>**

**<?php**

**$originalDate = $value['created\_at'];**

**$formattedDate = date('d F, Y', strtotime($originalDate));**

**echo $formattedDate;**

**?>**

**</td>**

**<td>**

**<a href="project-view.php?id=<?= $value['id'] ?>" class="btn btn-sm btn-light">Details</a>**

**</td>**

**</tr>**

**<?php }**

**} ?>**

**</tbody>**

**</table>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**</div>**

**<!-- container-fluid -->**

**</div>**

**<!-- End Page-content -->**

**</div>**

**<!-- end main content-->**

**</div>**

**<!-- END layout-wrapper -->**

**<!--start back-to-top-->**

**<button onclick="topFunction()" class="btn btn-primary btn-icon" id="back-to-top">**

**<i class="ri-arrow-up-line"></i>**

**</button>**

**<!--end back-to-top-->**

**<!--preloader-->**

**<div id="preloader">**

**<div id="status">**

**<div class="spinner-border text-primary avatar-sm" role="status">**

**<span class="visually-hidden">Loading...</span>**

**</div>**

**</div>**

**</div>**

**<!-- JAVASCRIPT -->**

**<?php include('layouts/script.php'); ?>**

**</body>**

**</html>**