

**A PROJECT REPORT ON  
WEB-BASED PROJECT MANAGEMENT SYSTEM:  
MANAGE YOUR ACADEMIC PROJECTS EFFICIENTLY**

*Submitted to*  
**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,  
KAKINADA**

*For Partial Fulfilment of Award of the Degree of*  
**BACHELOR OF TECHNOLOGY**

Submitted By

<b>M. Devika</b>	<b>21X41A05A5</b>
<b>K. Sravya</b>	<b>21X41A05A1</b>
<b>K. Hepsiba</b>	<b>21X41A0597</b>
<b>M. Manvanth</b>	<b>21X41A05A3</b>

*Under the esteemed guidance of*

**K. Jairam**

**Professor, Department of CSE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
S.R.K INSTITUTE OF TECHNOLOGY  
(AFFILIATED TO JNTU, KAKINADA)**

**Enikepadu, Vijayawada – 521108.**

**April 2025**

**S.R.K INSTITUTE OF TECHNOLOGY  
ENIKEPADU, VIJAYAWADA.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that this project report entitled **“WEB-BASED PROJECT MANAGEMENT SYSTEM: MANAGE YOUR ACADEMIC PROJECTS EFFICIENTLY”** is the Bonafide work of **M.Devika (21X41A05A5) , K.Sravya (21X41A05A1), K.Hepsiba(21X41A0597), M.Manvanth (21X41A05A3)** in partial fulfillment of the requirements for the award of the graduate degree in **BACHELOR OF TECHNOLOGY** during the academic year 2024-2025. This Work has carried out under our supervision and guidance.

**(K. Jairam)**  
**Signature of the Guide**

**(Dr. A. Radhika)**  
**Signature of the HOD**

**Signature of the  
External Examiner**

## DECLARATION

We M. Devika, K. Sravya, K. Hepsiba, M. Manvanth hereby declare that the project report entitled “**WEB-BASED PROJECT MANAGEMENT SYSTEM: MANAGE YOUR ACADEMIC PROJECTS EFFICIENTLY**” is an original work done in the Department of Computer Science & Engineering, SRK Institute of Technology, Enikepadu, Vijayawada, during the academic year 2024-2025, in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering. We assure you that this project is not submitted to any other College or University.

### PROJECT ASSOCIATES

<b>M. Devika</b>	<b>(21X41A05A5)</b>
<b>K. Sravya</b>	<b>(21X41A05A1)</b>
<b>K. Hepsiba</b>	<b>(21X41A0597)</b>
<b>M. Manvanth</b>	<b>(21X41A05A3)</b>

## ACKNOWLEDGEMENT

Firstly, we would like to convey our heart full thanks to the Almighty for the blessings on us to carry out this project work without any disruption.

We are extremely thankful to **K. Jairam**, our guide throughout the project. We also thank her for most independence and freedom throughout the given to us during various phases of the project.

We are also thankful for our project coordinator **Dr. P. Srinivas Kumar**, for their valuable guidance which helped us to bring this project successfully.

We are very much grateful to **Dr. A. Radhika**, H.O.D of C.S.E Department, for her valuable guidance which helped us to bring out this project successfully. Her wise approach made us learn the minute details of the subject. Her matured and patient guidance paved away for completing our project with a sense of satisfaction and pleasure.

We are greatly thankful to our principal **Dr. M. Ekambaram Naidu** for his kind support and facilities provided at our campus which helped us to bring out this project successfully.

Finally, we would like to convey our heart full thanks to our Technical Staff, for their guidance and support in every step of this project. We convey our sincere thanks to all the faculty and friends who directly or indirectly helped us with the successful completion of this project.

**M. Devika** (21X41A05A5)

**K. Sravya** (21X41A05A1)

**K. Hepsiba** (21X41A0597)

**M. Manvanth** (21X41A05A3)

# CONTENTS

<b>TITLE</b>	<b>Page.No</b>
List of Figures	i
List of Tables	ii
Abstract	1
<b>Chapter 1: INTRODUCTION</b>	<b>2</b>
1.1 : Overview	4
1.2 : About The Project	4
1.3 : purpose	4
1.4 : Scope	4
<b>Chapter 2: LITERATURE SURVEY</b>	<b>5</b>
<b>Chapter 3: SYSTEM ANALYSIS</b>	<b>7</b>
3.1 : Existing System	7
3.1.1 : Disadvantage of Existing System	7
3.2 : Proposed System	7
3.2.1 : Advantages of Proposed System	7
3.3 : Feasibility study	8
3.3.1 : Economic Feasibility	8
3.3.2 : Operational Feasibility	10
3.3.3 : Technical Feasibility	12
<b>Chapter 4: SYSTEM DESIGN</b>	<b>14</b>
4.1: UML Diagram	14
4.2: Structural Diagrams	15
4.2.1: Class Diagram	15

4.2.2 : Package Diagram	16
4.2.3 : Component Diagram	17
4.2.4 : Deployment Diagram	18
4.3: Behavioral Diagrams	19
4.3.1 : Activity Diagram	19
4.3.2 : Sequence Diagram	20
4.3.3 : Collaboration Diagram	21
4.3.4 : State Diagrams	22
4.3.5 : Use Case Diagram	23
<b>Chapter 5: SYSTEM IMPLEMENTATION</b>	<b>24</b>
5.1: Architecture	24
5.2: Working	26
5.3: Sample Code	27
5.4 : System Testing	68
5.5: Types of Testing	69
5.5.1 : Unit testing	69
5.5.2 : Integrational Testing	69
5.5.3 : Function Testing	70
5.5.4 : System Testing	70
5.5.5 : White Box Testing	71
5.5.6 : Black Box Testing	71
5.5.7 : Acceptance Testing	71
5.5.8 : Testing Results	72
5.6: Testing Methodologies	73
5.6.1 : Unit Testing	73
5.6.2 : Integration Testing	74

5.6.3 : Use Acceptance Testing	75
5.6.4 : Output Testing	75
5.6.5 : Validation Testing	75
<b>Chapter 6: RESULTS AND DISCUSSION</b>	<b>76</b>
<b>Chapter 7: CONCLUSION</b>	<b>80</b>
<b>REFERENCES</b>	<b>82</b>
<b>PAPER PUBLICATIONS</b>	<b>83</b>

## **LIST OF FIGURES**

<b>Figure No.</b>	<b>Name of the Figure</b>	<b>Page No.</b>
1	Representation of Class Diagram	15
2	Representation of Package Diagram	16
3	Representation of Component Diagram	17
4	Representation of Deployment Diagram	18
5	Representation of Activity Diagram	19
6	Representation of Sequence Diagram	20
7	Representation of Collaboration Diagram	21
8	Representation of State Machine Diagram	22
9	Representation of Use Case Diagram	23
10	Architecture and Working of the system	26

## ABSTRACT

The **Project Management System (PMS)** is a web system that facilitates the submission of academic projects and their evaluation for B-Tech students. PMS is developed in Python with Flask and SQLite. It enables users to submit projects and files for assessment through a web interface, making the process more efficient. The system features role-based access control, so different types of users have distinct permissions. The students in the system can submit their projects, while the administrators are able to assess the submissions and give feedback. Another interesting feature of PMS is the question and answer module, which allows students and administrators to interact directly in the system to ask questions about projects. It facilitates equity in the evaluation of projects as users can view their submission status alongside any feedback offered. Also, it streamlines the academic workflow by offering a framework for easy submission, assessment, and communication regarding the projects. Besides, PMS facilitates the safe storage and management of the project data, ensuring the integrity and accessibility for the users. PMS eliminates the manual workload related to project submission and evaluation, improving interaction between students and admins. By allowing all communication for academic projects to take place within a single system, usability and efficiency are maximized, resulting in better academic progress.



# Chapter 1

## INTRODUCTION

In today's fast-paced and technology-driven academic environment, managing student projects effectively has become more critical than ever. Academic institutions often grapple with challenges such as manual project tracking, inefficient communication between students and supervisors, inconsistent evaluation criteria, and delays in feedback. Traditionally, these processes relied heavily on paperwork, in-person meetings, and disjointed workflows. This conventional approach not only consumed time and resources but also introduced inefficiencies and inconsistencies into the academic evaluation system. With the rise of digital transformation in education, there is a growing demand for intelligent, scalable, and automated solutions to manage project submissions and evaluations more efficiently.

To address these issues, we propose a Project Management System (PMS)—a comprehensive, web-based platform developed using Python, Flask, and SQLite. This system is specifically tailored to streamline academic project workflows for B-Tech students and faculty, fostering better collaboration, reducing manual workload, and ensuring secure and organized handling of student submissions. The Project Management System (PMS) aims to digitize every stage of the academic project lifecycle—from topic registration and document uploads to faculty evaluations and feedback delivery.

One of the system's standout features is its role-based access control, which ensures that students, faculty, and administrators have access only to the features and data relevant to their roles. This contributes to maintaining data integrity, privacy, and security within the academic ecosystem. 2 Another key feature of the system is the Question and Answer (Q&A) module, which facilitates seamless communication between students and faculty. Rather than relying on emails or in-person queries, students can directly post questions within the system, and faculty can respond to them in a centralized environment. This encourages ongoing interaction, timely clarifications, and reduces misunderstandings during project development. Manual project submissions often lack structure and traceability, which leads to unorganized evaluations and the risk of data loss.

The PMS solves this through a stage-wise tracking system, which allows both students and faculty to monitor the progress of projects at different milestones. Whether it's the submission

of the abstract, mid-term report, or final documentation, each stage can be logged and reviewed systematically. A major pain point in traditional methods is the absence of a standardized evaluation framework. Different evaluators may use different criteria, which can lead to inconsistencies in assessment and student dissatisfaction. The PMS addresses this issue by supporting customizable evaluation rubrics—ensuring that all evaluations are consistent, transparent, and aligned with predefined academic standards.

Moreover, the PMS ensures multi-device accessibility and supports usage on desktops, tablets, and mobile devices, allowing users to access the platform conveniently regardless of their location. This is especially important in scenarios where remote learning or hybrid academic models are in place. In contrast to traditional methods that rely on manual reminders or faceto-face follow-ups, the PMS provides a real-time status tracking feature that allows both students and administrators to stay updated on project progress at various stages. Whether a student has submitted their 3 abstract, received feedback, or is awaiting final evaluation, the system clearly displays the current status of each submission. This visual and organized progress tracking enhances coordination and ensures that no stage of the project is overlooked. By making submission and review statuses readily visible, the system supports better planning and timely evaluations—without the need for additional reminders or manual status checks.

From a security perspective, the PMS integrates robust user authentication and authorization mechanisms to prevent unauthorized access. It uses session-based login validation, secure password storage, and other best practices to ensure sensitive academic data is protected. In addition to these features, the PMS contributes to the institution’s sustainability goals by reducing dependency on paper-based submissions and physical documentation. By transitioning to a digital platform, institutions can significantly reduce paper usage, thereby contributing to environmental conservation. Importantly, this project also emphasizes usability and scalability.

The user interface has been designed to be intuitive and user-friendly, allowing even non-technical users to navigate the system without confusion. The backend architecture ensures that the platform can be easily extended or scaled to support additional departments, user roles, or even institution-wide deployment in the future.

In conclusion, the Project Management System (PMS) is not just a digital submission portal—it is a holistic academic project management solution that brings transparency, structure, and efficiency to the entire lifecycle of student projects. By embracing automation, streamlined communication, role-based access, and secure data management, the PMS represents a

significant leap forward in how academic institutions manage student projects. It not only enhances the experience for students and faculty but also sets a foundation for future enhancements 4 like document versioning, feedback analytics, and deeper integration with institutional learning management systems (LMS).

## **1.1 Overview**

The Project Management System (PMS) is a web-based platform designed for academic institutions to streamline project submission, tracking, and evaluation. It eliminates manual workflows by offering a centralized portal for document uploads and status monitoring. The system promotes transparency, structured evaluations, and secure access for both students and admins.

## **1.2 ABOUT THE PROJECT**

This project aims to digitize the academic project handling process by enabling students to submit their work and track the status of their submissions. Built using Flask and SQLite, PMS includes role-based access control, secure file uploads, and a Q&A module to support communication between students and administrators. It simplifies submission and review processes while ensuring data security.

## **1.3 Purpose**

The purpose of PMS is to replace traditional manual workflows in academic project submission and evaluation with a secure, efficient, and structured online system. It enables users to upload project documents, monitor status, and exchange questions through a Q&A module, fostering better interaction and reducing administrative workload.

## **1.4 Scope:**

The Project Management System (PMS) is developed to simplify academic project handling through secure login, role-based access, and structured project submissions. It allows students to upload documents, view project status, and communicate via a Q&A module. Admins can view project details and provide basic feedback. The system includes project evaluation features based on predefined criteria. It runs on a responsive web interface for accessible use.

## Chapter 2

### LITERATURE REVIEW

The development of the **Web-Based Project Management System (PMS)** for academic project submission and evaluation is guided by a comprehensive review of prior research and related systems. These existing solutions shed light on the methodologies, technologies, and limitations that inform the design of PMS. The following section explores the relevant literature.

#### A. Web-based Projects Evaluation Management System (Al-Zoubi et al., 2008)

Al-Zoubi et al. presented one of the earliest frameworks for a web-based system dedicated to project evaluation. Their system emphasized the digitization of project assessments in academic institutions. It highlighted how web interfaces could streamline evaluations compared to manual approaches. However, the system lacked modern components like **Role-Based Access Control (RBAC)** and **real-time communication modules**, both of which are integrated into PMS to enhance usability and interactivity.

#### B. A Web-Based Project Management System (Aaron, 2016)

Aaron developed a general-purpose project management system with a focus on task tracking and accessibility through a web interface. While it laid foundational work for submission and monitoring processes, it did not cater to academic-specific functionalities such as feedback integration or faculty-student communication. The proposed PMS addresses these gaps by embedding a Q&A module and evaluation tools that support academic oversight and interaction.

#### C. Development of a Web-Based Student Project Management System (Nwachukwu, 2021)

Nwachukwu's system was tailored for student project workflows, highlighting ease of use in submission and data storage. Despite its effectiveness for simple academic processes, it lacked **role-based access, interactive communication, and real-time features**. PMS expands on this by introducing **differentiated user roles** (Admin, Faculty, Student) and enabling better interaction across stakeholders.

#### **D. Student Project Management System (Olayinka, 2024)**

Olayinka's research centered on the importance of centralized platforms for project submission and tracking in academic environments. Although it aligned with PMS in terms of submission flow, the system did not focus on secure data handling or direct messaging capabilities. PMS builds upon this by utilizing SQLite for lightweight secure storage and a built-in Q&A feature for efficient dialogue between user.

#### **E. Web-Based Student Project Management System – Tet fund Report (2021)**

The Tet fund report highlighted the implementation of a project management system at an institutional level, offering insights into streamlining academic workflows. However, the solution was highly context-specific and did not offer broader adaptability or interactivity. PMS, in contrast, is built to be scalable and usable across various institutions with diverse requirements.

#### **F. A Smart Web-Based Academic Project Management Platform (Liu & Fang, 2022)**

Liu and Fang proposed an intelligent academic management system that incorporated automation and smart features for tracking and user management. Their use of advanced web technologies and scalability reflects PMS's own tech choices, including Python Flask and SQLite. However, their system emphasized automation over human interaction, which PMS counters by maintaining a balance between efficiency and communication.

#### **G. Online Project Management System for University Students: A Case Study (Gupta & Sharma, 2023)**

This case study demonstrated how real-time updates and collaboration can benefit student project management. While effective, their system depended on heavier infrastructure. PMS achieves similar outcomes using **lightweight technologies**, making it especially suitable for smaller institutions with limited resources.

The literature collectively showcases a progression toward digitized, web-based academic project systems. Yet, several challenges remain, such as **lack of direct communication**, **limited role differentiation**, and **security concerns**. The proposed **PMS** bridges these gaps through its integration of **RBAC**, a **Q&A module**, and **secure SQLite-based storage**.

## **Chapter 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

In the existing system, academic project submissions and evaluations are conducted manually, leading to delays, lack of structure, and communication gaps between students and faculty. There is no centralized platform to manage or track the progress of submissions. Feedback is often informal or not documented, and the process relies heavily on physical paperwork. This results in inefficiencies and lack of clarity in project assessment.

##### **3.1.1 DISADVANTAGES OF EXISTING SYSTEM**

- Time consuming and unstructured submission process
- High dependency on paperwork and manual tracking
- Limited interaction and feedback between students and faculty
- No centralized status tracking or secure document storage
- Lack of transparency and standardization in evaluation.

#### **3.2 PROPOSED SYSTEM**

The proposed Project Management System (PMS) is a web-based application designed to digitize and streamline academic project handling. It supports role-based access for Admins, Students, and Faculty, enabling efficient submission, evaluation, and tracking. Students can upload project files, check status, and communicate via a basic Q&A section. Faculty members can review submissions, track progress, and provide feedback. The system also ensures secure login, hashed password protection, and organized project handling across stages.

##### **3.2.1 ADVANTAGES OF PROPOSED SYSTEM**

- Centralized system for managing and tracking projects
- The system makes the overall project management much easier and flexible.
- Secure submission of documents with access control
- Streamlined communication via built-in Q&A module

- Supports evaluation and feedback mechanisms
- Reduces paperwork and promotes eco-friendly practices
- Facilitates structured and transparent academic workflows

### **3.3 FEASIBILITY STUDY**

The preliminary investigation assesses the **feasibility and practicality** of implementing the Web-Based Project Management System (PMS) within an academic setting. Its primary objective is to determine whether the system can be effectively developed, deployed, and maintained under current technical, operational, and financial constraints.

This system is aimed at streamlining academic project submission, evaluation, and communication through a centralized platform. It is designed with features such as secure login, project file uploads, role-based access control for students and administrators, and a Q&A module to facilitate real-time interaction. Conducting a feasibility study at this stage is essential to ensure that the proposed solution not only addresses the core challenges of manual project handling but is also viable in terms of development scope and institutional readiness.

The three aspects of this study include:

- Economical Feasibility
- Operational Feasibility
- Technical Feasibility

#### **3.3.1 ECONOMIC FEASIBILITY**

The economic feasibility of developing the Web-Based Project Management System (PMS) involves a detailed assessment of the costs associated with development, deployment, and maintenance, as well as the potential long-term savings and productivity benefits. This analysis determines whether the system is financially viable and justifies the investment for academic institutions. By reducing reliance on manual processes, paperwork, and administrative overhead, the PMS can offer a cost-effective and scalable solution in the long run.

## **Initial Development Costs**

### **Personnel Costs:**

The project requires web developers proficient in Python, Flask, HTML, CSS, and database integration using SQLite. As the system is educational and modular, it can be built by a small team, including student developers or interns, reducing the financial burden.

### **Software Licenses And Tools:**

- The technologies used — Flask, SQLite, and Bootstrap — are open-source and free.
- Visual Studio Code and other development tools used are also free, minimizing expenses.
- No premium licenses or paid frameworks are required, making it cost-efficient.

### **Operational Costs**

#### **Database Management:**

- SQLite is a lightweight, serverless database and incurs no hosting or subscription costs.
- Minimal backup and maintenance operations are required since the database is File-based and easily portable.
- Local server or university-hosted deployment can avoid cloud infrastructure costs.

### **Potential Savings And Benefits**

#### **Efficiency Gains:**

- Automates student project submission, status tracking, and faculty evaluation.
- Reduces manual administrative tasks, such as physical collection of document spreadsheet tracking.
- Saves time for faculty and improves overall academic workflow.

#### **Paperless Management**

- The system supports digital submission and communication, reducing dependency on physical infrastructure and paperwork.
- This contributes to cost savings and sustainability.



### **Role-Based Access**

- Clear separation of roles (Admin, Faculty, Student) ensures secure and organized workflows, reducing errors and administrative confusion — a common cause of hidden costs in manual systems.

### **Scalability and Maintenance**

- The system is scalable for small to medium institutions without needing hardware upgrades or paid software.
- Long-term maintenance can be handled by internal IT teams or trained students, avoiding the need for outsourcing.

### **Return on Investment (ROI)**

- Short-term investment in development leads to long-term cost efficiency through reduced paperwork, streamlined operations, and lower error rates.
- PMS adds value by improving faculty-student communication and ensuring better tracking of academic projects.
- Educational institutions adopting PMS can also showcase digital transformation, boosting their reputation and operational efficiency.

## **3.3.2 OPERATIONAL FEASIBILITY**

The operational feasibility of the Web-Based Project Management System (PMS) involves evaluating whether the system can be effectively integrated into the day-to-day processes of academic institutions. It focuses on user interaction, training requirements, and compatibility with existing workflows.

### **User Acceptance:**

- **User-Friendliness:**

The PMS is designed with a clean, responsive web interface using HTML, CSS, and Bootstrap, making it intuitive for Admins and Students to navigate. Project submission, tracking, and Q&A communication are accessible through structured menus and forms, promoting quick adoption.

- **Ease of Use:**

Minimal technical expertise is required to operate the system. User roles and dashboard layouts are simplified to reduce confusion and learning curves.

## **Training Requirements**

- **Basic Training Needs:**

Due to the straightforward design and layout, only basic training is needed for both Admins and Students. A brief orientation or video tutorial is sufficient to get users onboard.

- **Documentation:**

A simple user manual or walkthrough can be developed for onboarding. There is no need for extensive workshop-based training.

## **System Integration:**

- **Compatibility with Existing Systems:**

As a lightweight Flask-based application, PMS can run alongside existing institutional tools without conflicts. It can be deployed on local servers or integrated with institutional infrastructure easily.

- **Data Migration:**

Since PMS is built from scratch, initial deployment starts with fresh data. Existing student project records can be manually imported if needed. SQLite allows easy migration of future data if the system is scaled up to other databases.

## **Support and Maintenance**

The PMS system is modular and easy to maintain. Bug fixes and feature updates can be handled by a small technical team or even trained students with basic Python knowledge. Routine tasks like backup, user management, and small modifications can be done with minimal downtime. Technical support requirements are low due to the system's stable and straightforward structure.

### **Technical Support:**

The system requires limited support due to its minimalistic architecture. Any bugs or issues can be resolved by an in-house IT administrator or student developers familiar with Flask and Python.

### **System Maintenance:**

The system supports easy maintenance with modular code. Tasks like database backup, adding new user roles, or modifying forms can be handled periodically with minimal downtime.

### 3.3.3 TECHNICAL FEASIBILITY

The technical feasibility of developing the **Web-Based Project Management System (PMS)** using **Flask (Python)** and **SQLite** revolves around assessing the suitability of chosen technologies, infrastructure needs, development tools, and long-term maintainability. The goal is to ensure that the system can be effectively built, deployed, and supported within an academic institution's current technical ecosystem.

**Flask** is a micro web framework that offers flexibility, simplicity, and modularity—making it a perfect choice for lightweight yet functional academic applications. Its extensive documentation and active community support make it beginner-friendly and efficient for rapid development. **SQLite**, being a serverless and lightweight database, is ideal for academic-scale usage where the data load is manageable, and performance is critical without the complexity of setting up a separate database server.

#### Development Tools

Development can be carried out using widely adopted and freely available IDEs like **Visual Studio Code**, **PyCharm (Community Edition)**, or any code editor of choice. These environments provide necessary support for debugging, version control, and testing, simplifying the entire development process.

#### Cost Analysis

Since both Flask and SQLite are **open-source technologies**, there are **no licensing costs**, making the system cost-effective for institutions with limited budgets. The use of an open-source stack significantly reduces initial and long-term expenditures.

- **Development Costs:**

There are no major expenses for proprietary tools or commercial licenses. The system can be built and maintained by students, academic staff, or internal IT teams using freely available frameworks and resources. This makes it an excellent educational project while still being practical for real deployment.

- **Operational Costs:**

The system does not require expensive hosting or server configurations. **SQLite**, being file-based, **does not need a dedicated database server**, minimizing hosting costs.

Deployment can be done on institutional servers or local environments with minimal hardware requirements.

### **Security**

Security is a critical component, especially when dealing with sensitive academic data such as student submissions and evaluation feedback. PMS includes:

- **Secure login mechanisms** with password hashing,
- **Role-based access control** to separate privileges for students and administrators,
- **Input validation and file handling safeguards** to prevent unauthorized access or data corruption.

In conclusion, the technological foundation of PMS is **technically feasible, affordable, and sustainable**, particularly in educational institutions where ease of deployment, minimal costs, and secure data handling are essential.

## **Chapter 4**

### **SYSTEM DESIGN**

#### **4.1 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

#### **GOALS**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## 4.2 STRUCTURAL DIAGRAM

### 4.2.1 CLASS DIAGRAM

The UML class diagram is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.

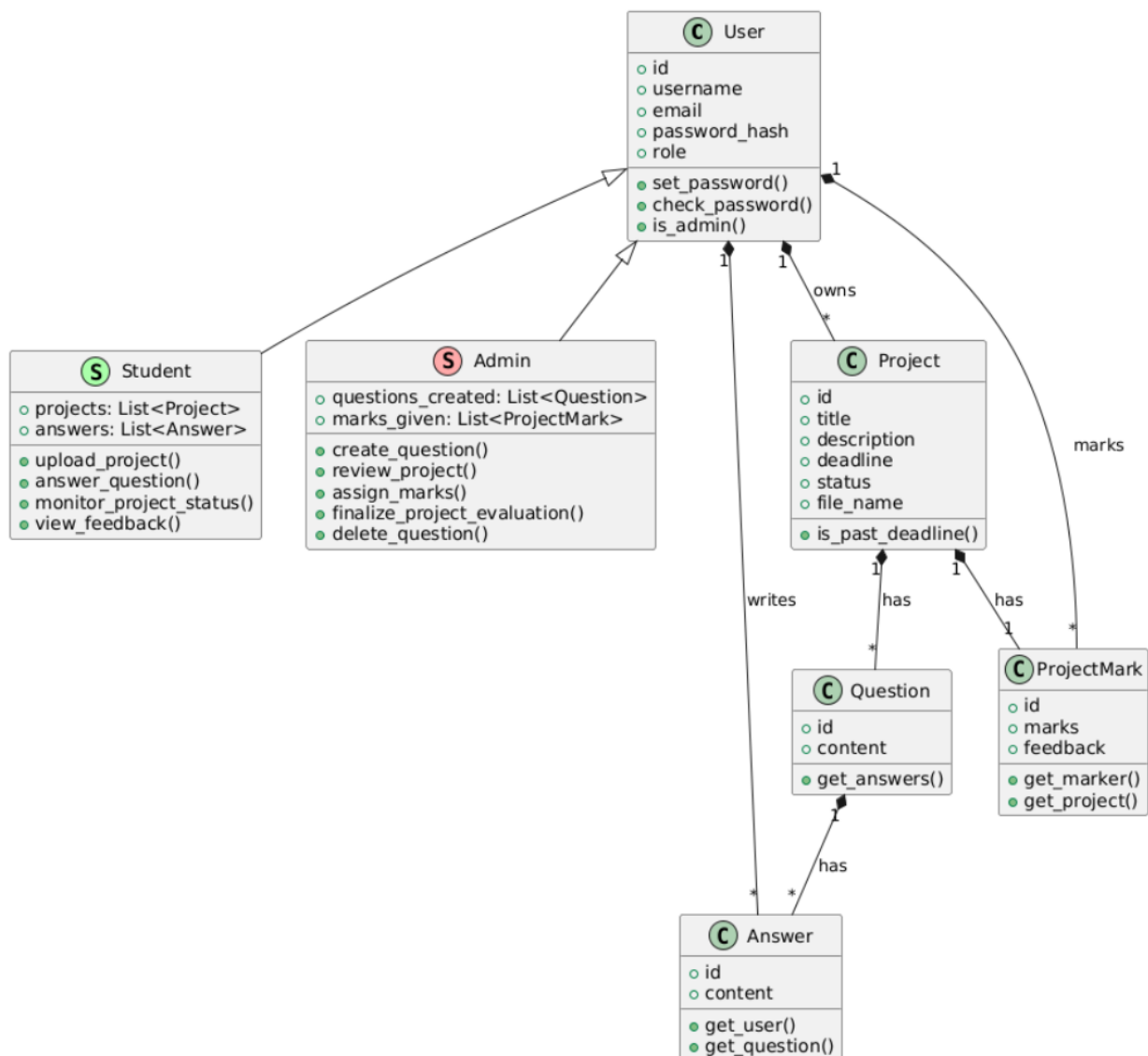


Fig:1. Representation of Class Diagram

### 4.2.2 PACKAGE DIAGRAM

We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organize UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organize class and use case diagrams.

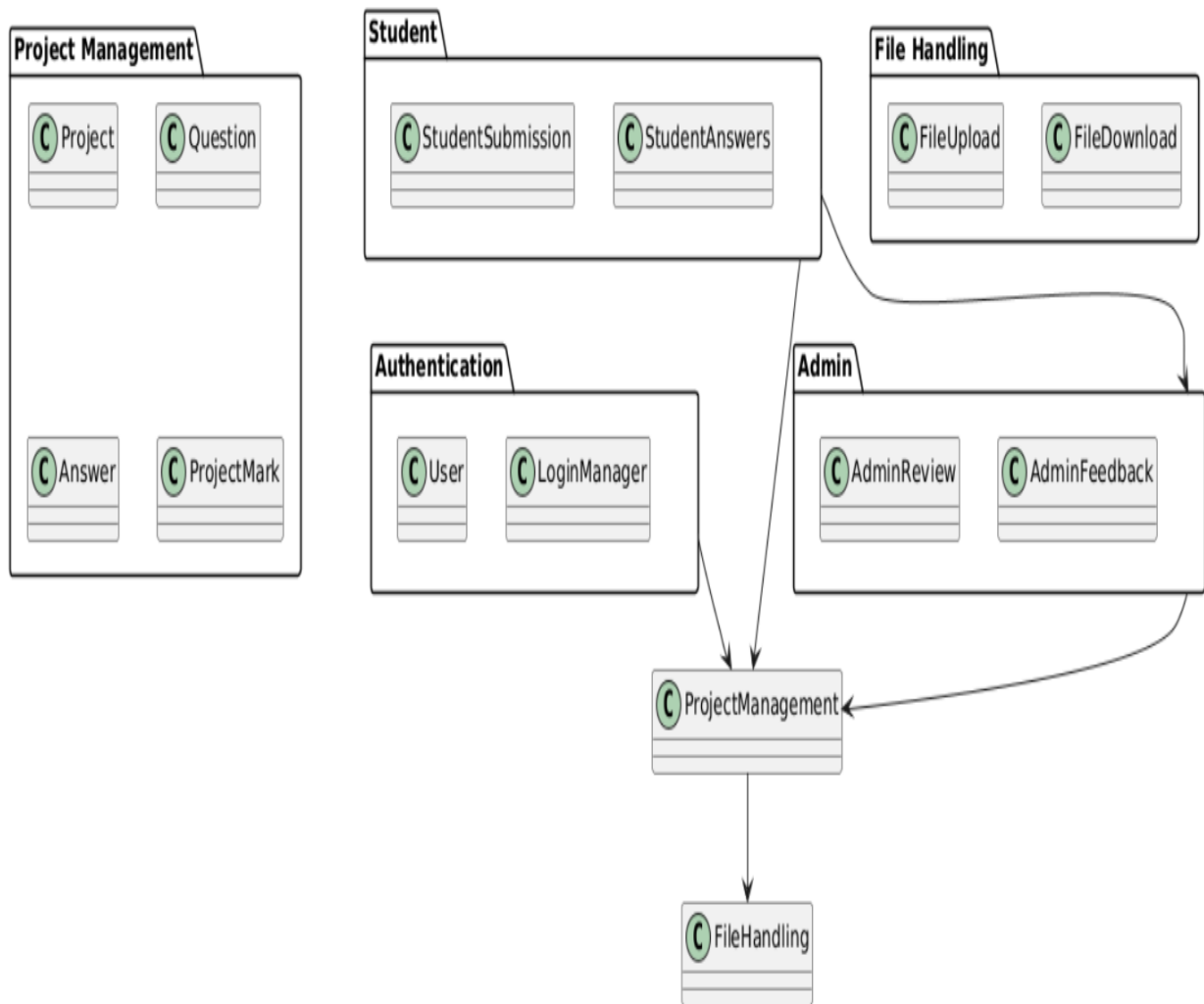


Fig:2. Representation of package Diagram

### 4.2.3 COMPONENT DIAGRAM

Component diagrams are used to represent how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other.

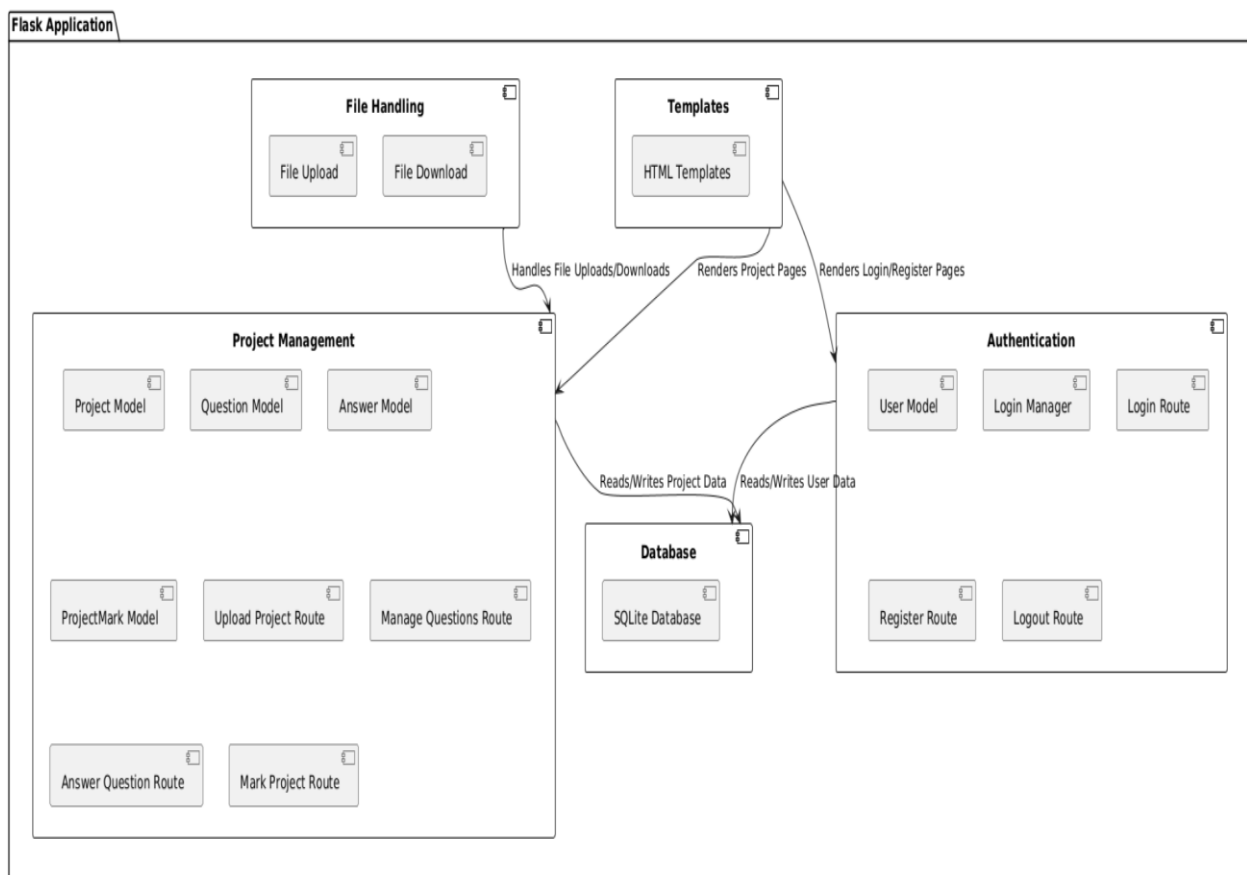


Fig:3. Representation of Component Diagram



#### 4.2.4 DEPLOYMENT DIAGRAM

Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations.

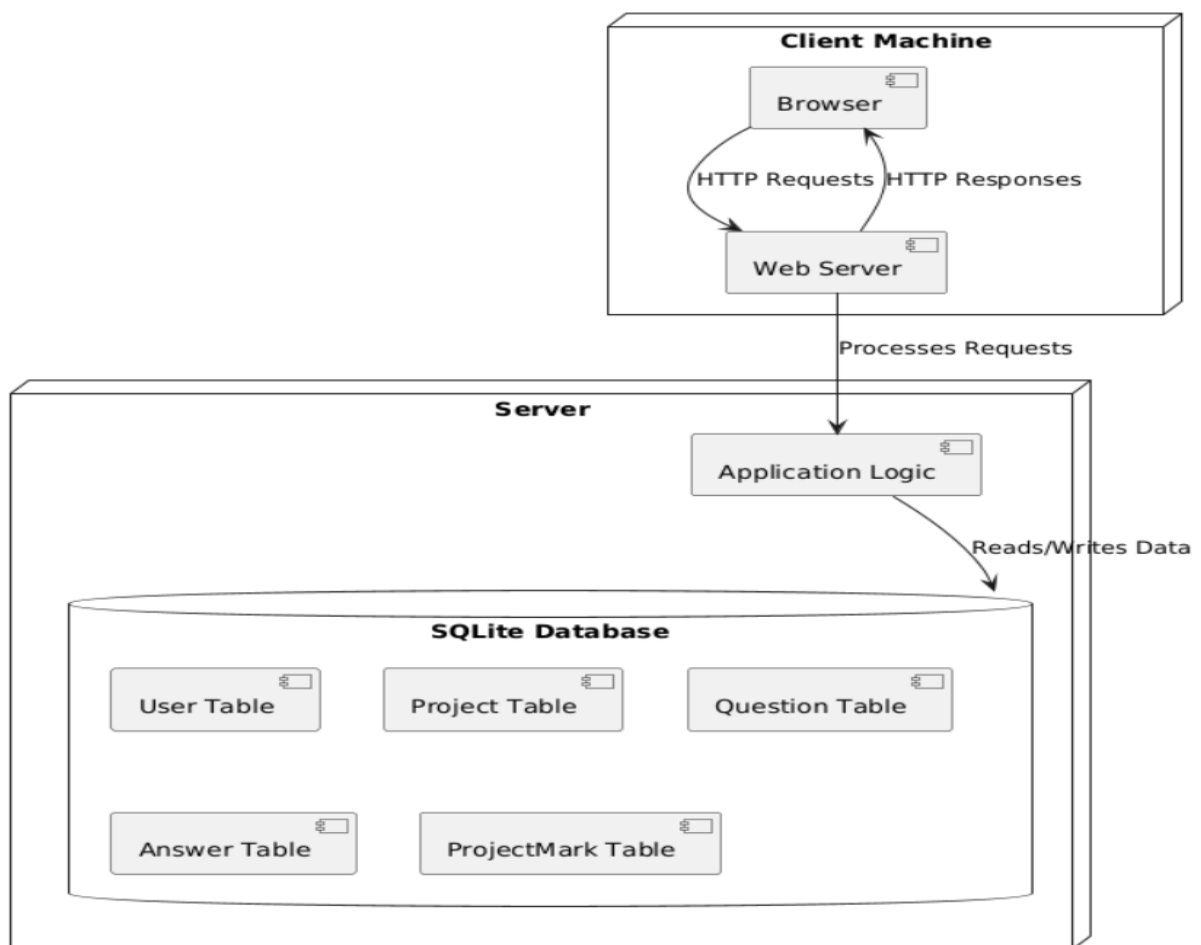


Fig:4. Representation of Deployment Diagram

## 4.3: BEHAVIORAL DIAGRAMS

### 4.3.1 : Activity Diagram:

We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

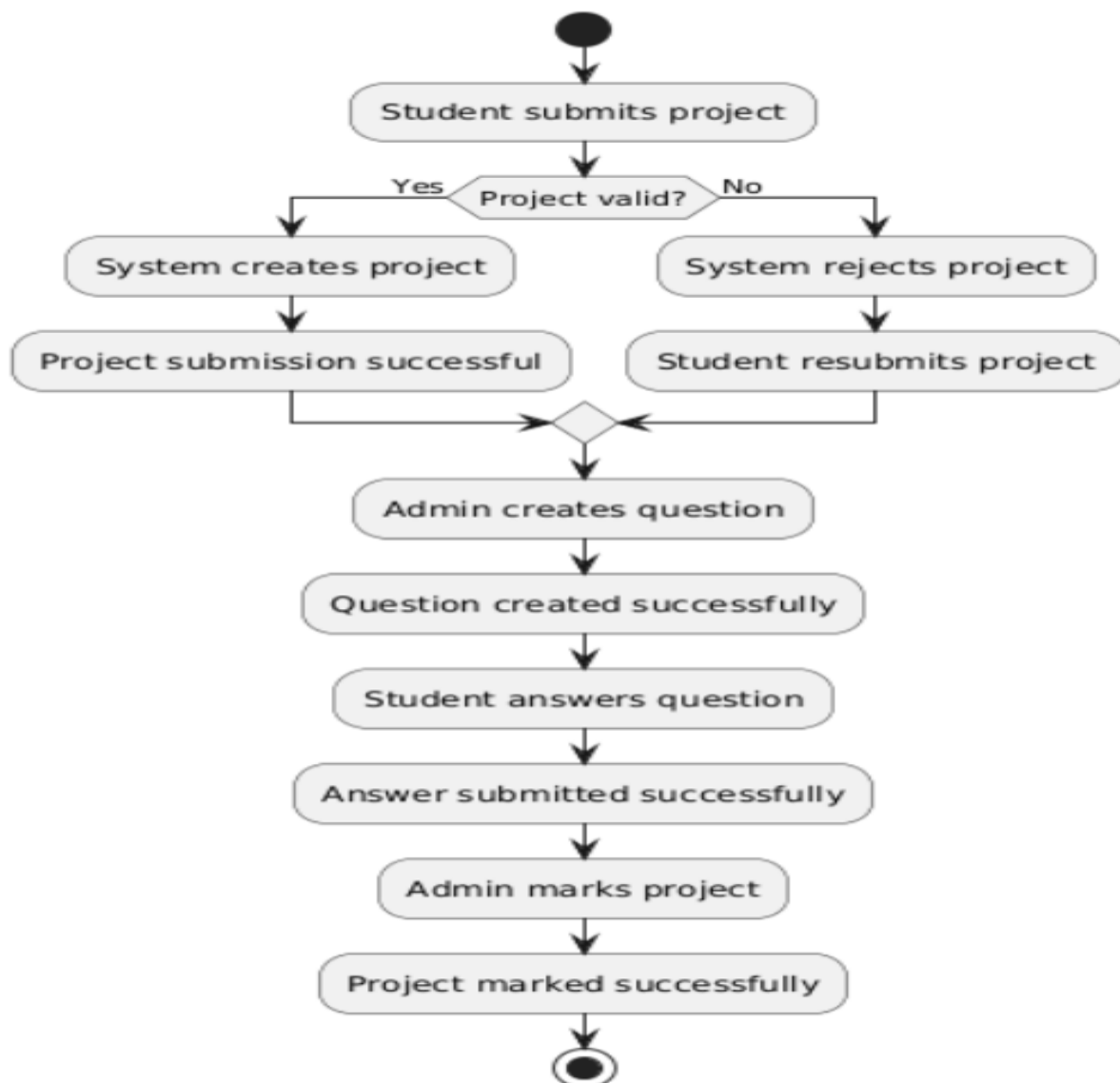


Fig: 5. Representation of Activity diagram

### 4.3.2 : Sequence Diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

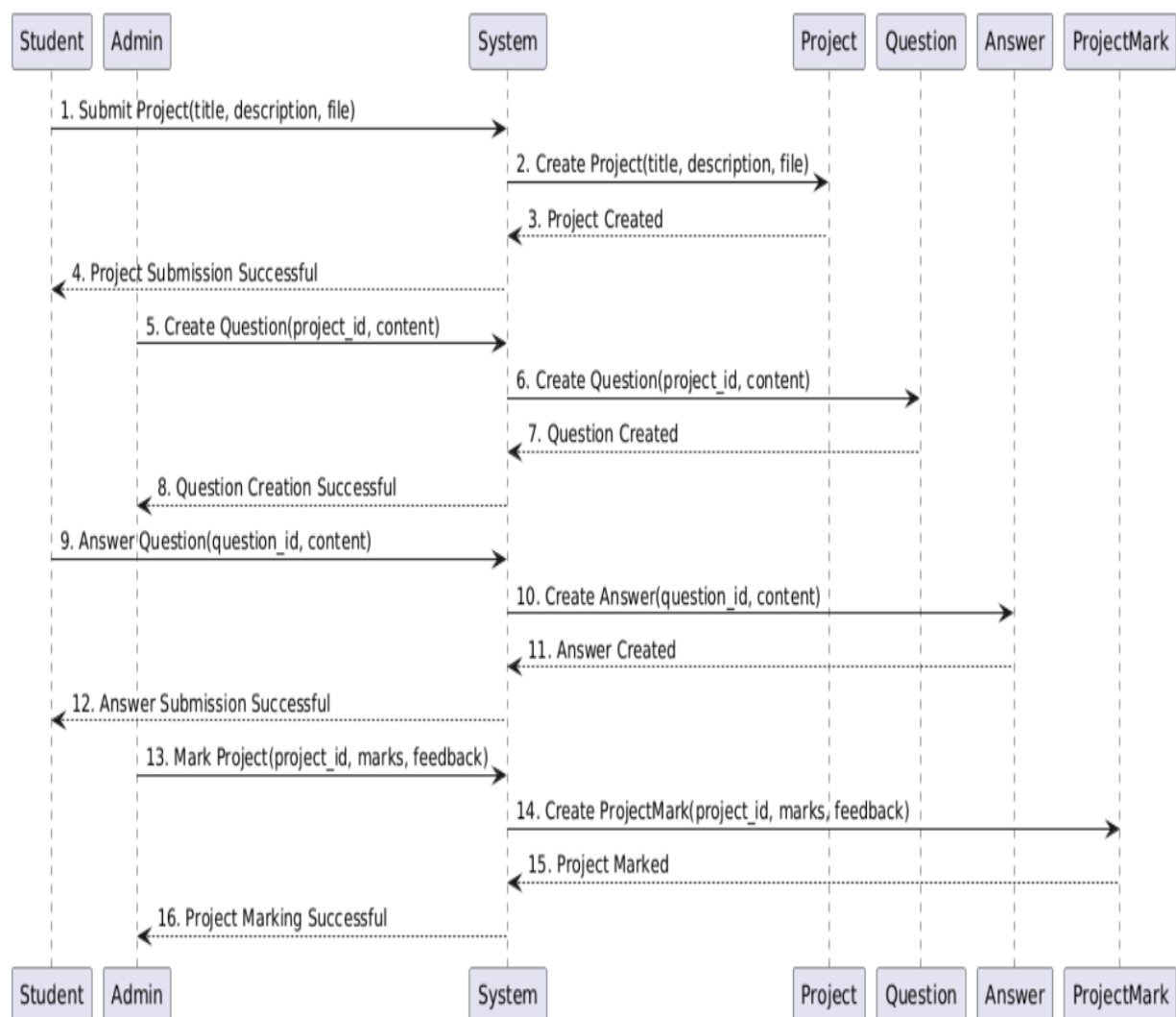


Fig: 6. Representation of Sequence diagram

### 4.3.3 Collaboration Diagram:

A Communication Diagram (known as Collaboration Diagram in UML 1.x) is used to show sequenced messages exchanged between objects. A communication diagram focuses primarily on objects and their relationships. We can represent similar information using Sequence diagrams, however communication diagrams represent objects and links in a free form.

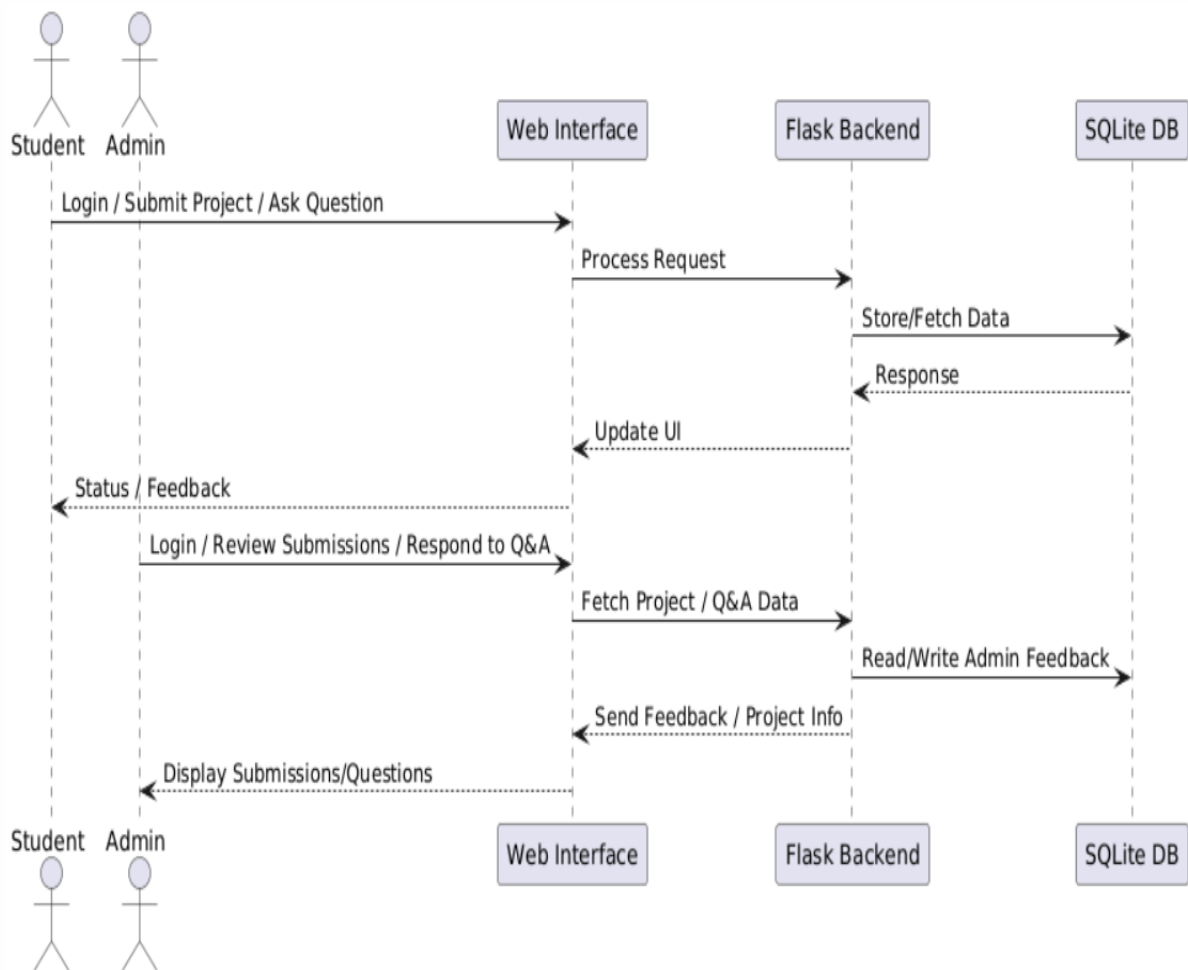


Fig: 7. Representation of Collaboration Diagram

#### 4.3.4 : State Machine Diagram:

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as **State machines** and **State-chart Diagrams**. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli.

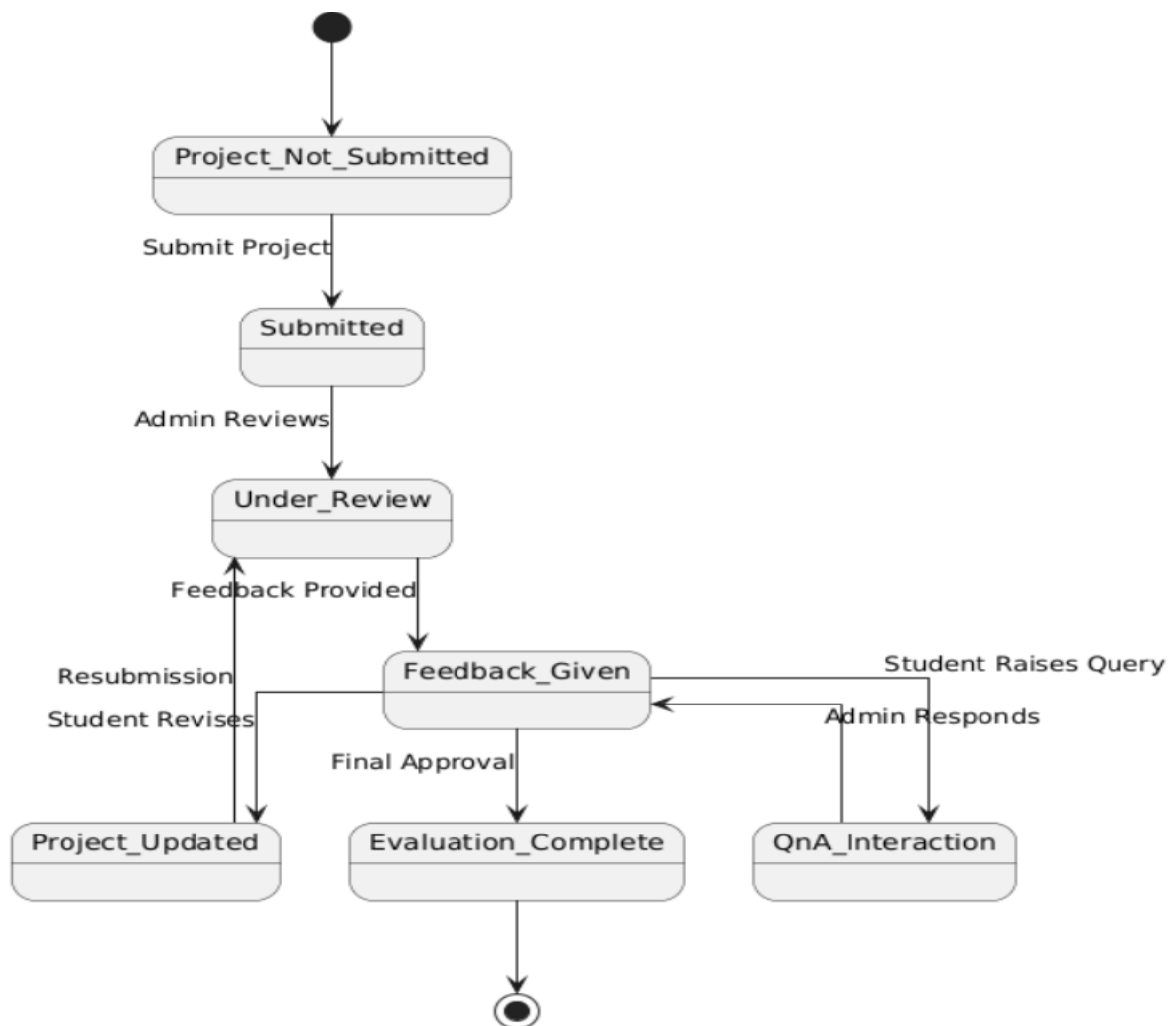


Fig: 8. Representation of State diagram

#### 4.3.5 : Use Case Diagram:

Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents(actors). A use case is basically a diagram representing different scenarios where the system can be used. A use case diagram gives us a high level view of what the system or a part of the system does without going into implementation details.

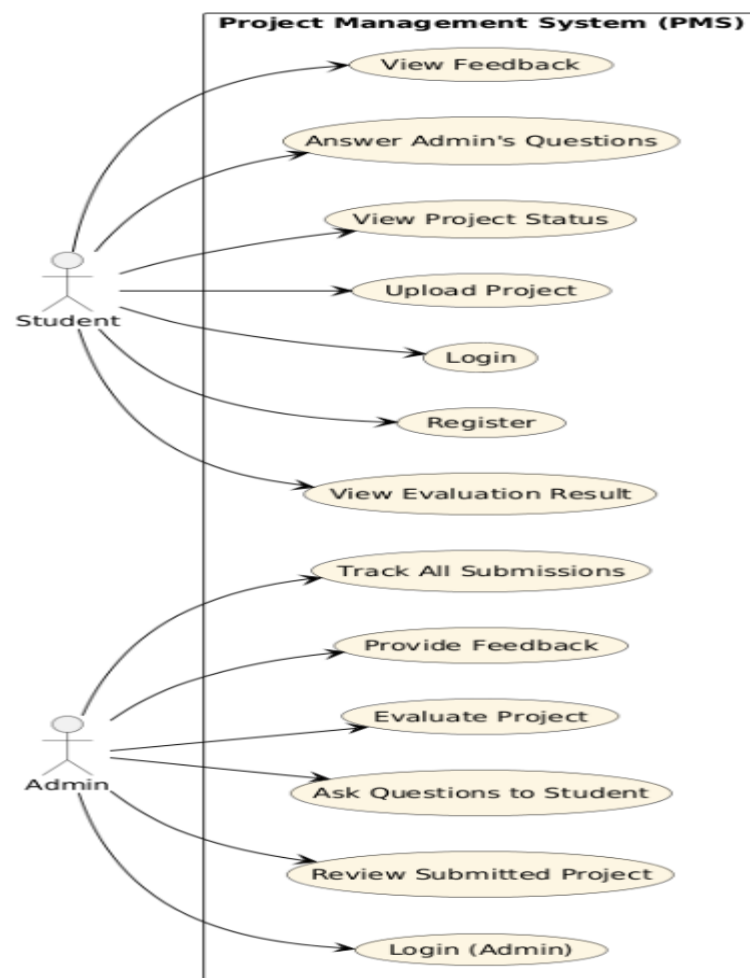


Fig: 9. Representation of Use Case Diagram

## Chapter 5

### SYSTEM IMPLEMENTATION

#### 5.1 ARCHITECTURE

The architecture of the Project Management System (PMS) is designed using a modular and layered approach to ensure scalability, maintainability, and ease of integration. It comprises three main components: the Frontend, the Backend Server, and the Database, each playing a crucial role in handling user interaction, data processing, and storage.

##### **Frontend (HTML, CSS, JavaScript):**

The user interface of the PMS is built using standard web technologies—HTML, CSS, and JavaScript—to ensure responsiveness, accessibility, and a smooth user experience across devices. The frontend presents:

- Login interfaces for both students and administrators with validation mechanisms.
- A project submission form for students to upload their project files and enter relevant details like title and description.
- A dashboard that allows students to view the status of their submitted projects.
- An admin panel for administrators to view submitted projects, post questions, provide evaluation feedback, and update project status.
- A Q&A module enabling communication between students and admins.
- This layer focuses on usability and user navigation, ensuring that users can interact with the system without needing technical expertise.

##### **Backend Server (Flask - Python):**

The backend server is powered by the **Flask web framework**, which acts as the application's core engine. It performs several vital tasks:

- **Routing** HTTP requests to the appropriate views (e.g., login, project upload, feedback form).
- **Authentication and session management** for secure login and role-based access control.
- **Input validation and file handling** for secure project uploads and downloads.

- **Data exchange between frontend and database**, processing logic for evaluations, and managing state transitions in project workflows.
- Using Flask offers flexibility and simplicity, making the application lightweight, fast, and easy to deploy in academic environments.

### Database (SQLite):

The system uses **SQLite** as its database solution, ideal for lightweight academic applications that require zero-configuration and minimal overhead. The database is managed using **SQLAlchemy**, an Object Relational Mapper (ORM) that allows seamless interaction between Python code and SQL database structures.

### Schema Design:

The schema includes several interconnected tables:

- **users:** stores user credentials and roles (student/admin).
- **projects:** holds project metadata and file paths.
- **feedback:** records admin evaluations and student responses.
- **questions:** manages the Q&A communication between users.

This schema ensures that each entity is properly linked, enabling efficient tracking of submissions, evaluations, and conversations.

### Database Management:

SQLAlchemy simplifies database interactions through Python classes and methods, abstracting raw SQL queries. It ensures that:

- Data integrity and consistency are maintained.
- Operations like insertion, updating, deletion, and querying are handled efficiently.
- The system remains maintainable and scalable as data volume grows.

## 5.2 WORKING:

- **Signup:** Only students can register using a signup form. Admin accounts are predefined and do not require registration.
- **Sign In:** Both students and admins log in with their credentials and are directed to their role-specific dashboards.
- **Project Submission (Student Module):** Students can upload project files in formats like PDF, DOCX, or RAR, which are stored in the database.



- **View Project Status (Student Module):** Students can view the current status of their submitted project (e.g., submitted, reviewed, or evaluated).
- **Question and Answer Module:** Admins can ask questions related to projects, and students can respond through the Q&A interface.
- **Feedback Viewing (Student Module):** After evaluation, feedback is made visible to students for review through their dashboard.
- **Evaluation (Admin Module):** Admins access project submissions, assess them, provide feedback, and provide evaluation result to the student.
- **Admin Module:** Admins manage all submissions, questions, and feedback using their dashboard without registration.

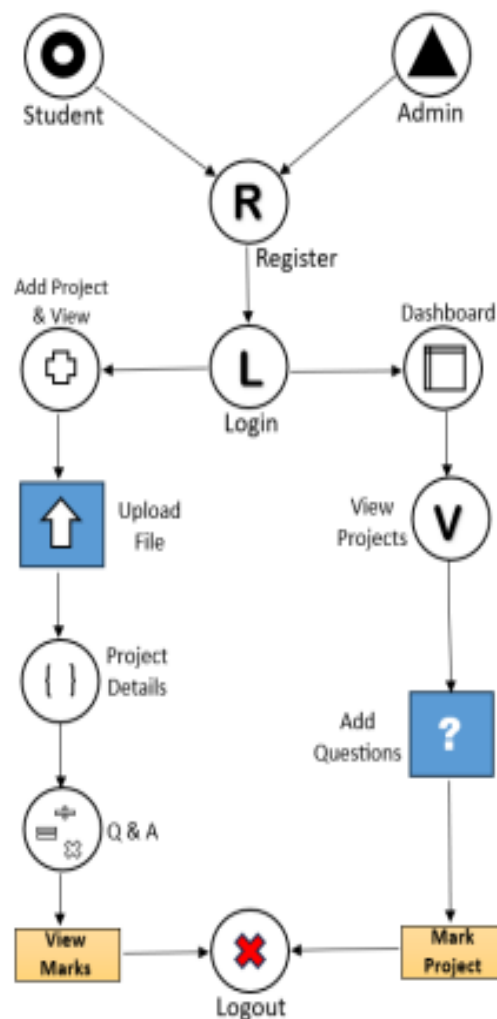


Fig: 10. Architecture and Working of the System

## 5.3 SAMPLE CODE:

### Frontend :

#### Register.html :

```
{% extends "base.html" %}

{% block content %}

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

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

        <h2 class="mb-4">Register</h2>

        <form method="POST">

            <div class="mb-3">

                <label for="username" class="form-label">Username</label>

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

            </div>

            <div class="mb-3">

                <label for="email" class="form-label">Email</label>

                <input type="email" class="form-control" id="email" name="email"
required>

            </div>

            <div class="mb-3">

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

                <input type="password" class="form-control" id="password"
name="password" required>

            </div>

            <div class="mb-3">

                <label for="role" class="form-label">Role</label>

                <select class="form-control" id="role" name="role" required>

                    <option value="student">Student</option>

                    <option value="educator">Educator</option>
```

```

        </select>

    </div>

    <button type="submit" class="btn btn-primary">Register</button>

</form>

    <p class="mt-3">Already have an account? <a href="{{ url_for('login') }}">Login
here</a></p>

</div>

</div>

{ % endblock % }

```

### Login.html:

```

{ % extends "base.html" % }

{ % block content % }

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

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

        <h2 class="mb-4">Login</h2>

        <form method="POST">

            <div class="mb-3">

                <label for="username" class="form-label">Username</label>

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

            </div>

            <div class="mb-3">

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

                <input type="password" class="form-control" id="password" name="password"
required>

            </div>

            <button type="submit" class="btn btn-primary">Login</button>

        </form>

```

```
<p class="mt-3">Don't have an account? <a href="{{ url_for('register') }}">Register  
here</a></p>
```

```
</div>
```

```
</div>
```

```
{% endblock %}
```

### **index.html:**

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<div class="text-center">
```

```
<h1>Welcome to Project Management System</h1>
```

```
<p class="lead">Manage your academic projects efficiently</p>
```

```
{% if not current_user.is_authenticated %}
```

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

```
<a href="{{ url_for('login') }}" class="btn btn-primary me-2">Login</a>
```

```
<a href="{{ url_for('register') }}" class="btn btn-outline-primary">Register</a>
```

```
</div>
```

```
{% endif %}
```

```
</div>
```

```
{% endblock %}
```

### **base.html :**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Project Management System</title>
```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"  
rel="stylesheet">
```

```

</head>

<body>

  <nav class="navbar navbar-expand-lg navbar-light bg-light">

    <div class="container">

      <a class="navbar-brand" href="{{ url_for('index') }}">Project System</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav ms-auto">

          {% if current_user.is_authenticated %}

            <li class="nav-item">

              <a class="nav-link" href="{{ url_for('dashboard') }}">Dashboard</a>

            </li>

            <li class="nav-item">

              <a class="nav-link" href="{{ url_for('logout') }}">Logout</a>

            </li>

          {% else %}

            <li class="nav-item">

              <a class="nav-link" href="{{ url_for('login') }}">Login</a>

            </li>

            <li class="nav-item">

              <a class="nav-link" href="{{ url_for('register') }}">Register</a>

            </li>

          {% endif %}

        </ul>

      </div>

    </div>

  </nav>

```

```

<div class="container mt-4">
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-info">{{ message }}</div>
            {% endfor %}
        {% endif %}
    {% endwith %}

    {% block content %}{% endblock %}
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

### **dashboard.html :**

```

{% extends "base.html" %}
{% block content %}
<div class="container">
    <div class="d-flex justify-content-between align-items-center mb-4">
        <h2>Dashboard</h2>
        {% if current_user.role != 'admin' %}
            <a href="{{ url_for('upload_project') }}" class="btn btn-primary">Upload New Project</a>
        {% endif %}
    </div>

    <div class="row">
        {% for project in projects %}
            <div class="col-md-4 mb-4">

```

```

<div class="card">
    <div class="card-body">
        <h5 class="card-title">{{ project.title }}</h5>
        <p class="card-text">{{ project.description[:100] }}{% if project.description|length > 100
% }...{% endif %}</p>
        <p class="card-text">
            <small class="text-muted">Deadline: {{ project.deadline.strftime('%Y-%m-%d')
}}</small>
        </p>
        <div class="d-flex justify-content-between align-items-center">
            <span class="badge bg-{{ 'success' if project.status == 'Completed' else 'primary' }}">
                {{ project.status }}
            </span>
            <a href="{{ url_for('project_details', project_id=project.id) }}"
                class="btn btn-outline-primary btn-sm">View Details</a>
        </div>
    </div>
</div>
</div>
</div>
{% endfor %}
</div>
</div>
{% endblock %}

```

### **project\_details.html:**

```

{% extends "base.html" %}
{% block content %}
<div class="container">
    <div class="row mb-4">
        <div class="col-md-8">

```

```

<h2>{{ project.title }}</h2>

<p class="text-muted">Status: {{ project.status }}</p>

<p>{{ project.description }}</p>

<p><strong>Deadline:</strong> {{ project.deadline.strftime('%Y-%m-%d') }}</p>

{% if project.file_name %}

<a href="{{ url_for('download_file', filename=project.file_name) }}"

    class="btn btn-primary">Download Project File</a>

{% endif %}

</div>

</div>

{# Display marks if available #}

{% if project.mark %}

<div class="row mb-4">

    <div class="col-md-8">

        <div class="card">

            <div class="card-body">

                <h5 class="card-title">Project Marks</h5>

                <div class="d-flex justify-content-between align-items-center">

                    <h2 class="text-primary mb-0">{{ project.mark.marks }}/100</h2>

                    <span class="badge {{ 'bg-success' if project.mark.marks >= 60 else 'bg-
warning' }}">

                        {{ 'Pass' if project.mark.marks >= 60 else 'Fail' }}

                    </span>

                </div>

            </div>

            {% if project.mark.feedback %}

            <hr>

            <h6>Feedback:</h6>

            <p>{{ project.mark.feedback }}</p>

```



```

        { % endif % }

        <small class="text-muted">Marked on { { project.mark.marked_at.strftime('% Y-
% m-% d % H:% M') } }</small>

    </div>

</div>

</div>

</div>

{ % endif % }

<div class="row">
    <div class="col-md-8">
        <h3>Questions</h3>
        { % if questions % }
            { % for question in questions % }
                <div class="card mb-3">
                    <div class="card-body">
                        <h5 class="card-title">{ { question.content } }</h5>

                        <!-- Answers -->
                        { % if question.answers % }
                            <div class="mt-3">
                                <h6>Answers:</h6>
                                { % for answer in question.answers % }
                                    <div class="card mb-2">
                                        <div class="card-body">
                                            <p class="card-text">{ { answer.content } }</p>
                                            <small class="text-muted">
                                                Answered on { { answer.submitted_at.strftime('% Y-% m-% d
% H:% M') } }

```

```

        </small>

    </div>

</div>

    {% endfor %}

</div>

{% endif %}

<!-- Answer Form -->

{% if current_user.role != 'admin' and project.status != 'Completed' %}

    <form method="POST" action="{ { url_for('answer_question',
question_id=question.id) } }"

        class="mt-3">

        <div class="form-group">

            <label for="answer{ { question.id } }">Your Answer:</label>

            <textarea class="form-control" id="answer{ { question.id } }"

                name="content" rows="3" required></textarea>

        </div>

        <button type="submit" class="btn btn-primary mt-2">Submit
Answer</button>

    </form>

    {% endif %}

</div>

</div>

    {% endfor %}

    {% else %}

        <p>No questions available for this project.</p>

    {% endif %}

</div>

</div>

```

```
</div>
```

```
{% endblock %}
```

## **upload\_project.html:**

```
{% endblock %}
```

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<div class="container">
```

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

```
        <div class="col-md-8">
```

```
            <h2 class="mb-4">Upload Project</h2>
```

```
            <form method="POST" enctype="multipart/form-data">
```

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

```
                    <label for="title" class="form-label">Project Title</label>
```

```
                    <input type="text" class="form-control" id="title" name="title" required>
```

```
                </div>
```

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

```
                    <label for="description" class="form-label">Description</label>
```

```
                    <textarea class="form-control" id="description" name="description" rows="3"
required></textarea>
```

```
                </div>
```

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

```
                    <label for="deadline" class="form-label">Deadline</label>
```

```
                    <input type="date" class="form-control" id="deadline" name="deadline"
required>
```

```
                </div>
```

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

```
                    <label for="project_file" class="form-label">Project File</label>
```

```
                    <input type="file" class="form-control" id="project_file" name="project_file"
required>
```

```
<small class="text-muted">Allowed file types: PDF, DOC, DOCX, ZIP, RAR (Max
size: 16MB)</small>
```

```
</div>
```

```
<button type="submit" class="btn btn-primary">Upload Project</button>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{% endblock %}
```

### **view\_project.html:**

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<div class="container">
```

```
<h2 class="mb-4">{{ project.title }}</h2>
```

```
<p><strong>Owner:</strong> {{ project.owner.username }}</p>
```

```
<p><strong>Description:</strong> {{ project.description }}</p>
```

```
<p><strong>Deadline:</strong> {{ project.deadline.strftime('%Y-%m-%d') }}</p>
```

```
<p><strong>Status:</strong> {{ project.status }}</p>
```

```
{% if project.file_name %}
```

```
<p>
```

```
<a href="{{ url_for('download_file', filename=project.file_name) }}"
```

```
class="btn btn-primary">Download Project File</a>
```

```
</p>
```

```
{% endif %}
```

```

{# Display marks if available #}

{% if project.mark %}

<div class="card mb-4">

  <div class="card-body">

    <h5 class="card-title">Project Marks</h5>

    <p><strong>Marks:</strong> {{ project.mark.marks }}/100</p>

    {% if project.mark.feedback %}

    <p><strong>Feedback:</strong> {{ project.mark.feedback }}</p>

    {% endif %}

    <p class="text-muted">Marked on {{ project.mark.marked_at.strftime('%Y-%m-%d
%H:%M') }}</p>

    <a href="{{ url_for('mark_project', project_id=project.id) }}"

      class="btn btn-warning btn-sm">Update Marks</a>

  </div>

</div>

{% else %}

<div class="mb-4">

  <a href="{{ url_for('mark_project', project_id=project.id) }}"

    class="btn btn-success">Assign Marks</a>

</div>

{% endif %}

<div class="row mt-4">

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

    <div class="d-flex justify-content-between align-items-center mb-3">

      <h3>Questions</h3>

      <div>

        <a href="{{ url_for('manage_questions', project_id=project.id) }}"

          class="btn btn-primary me-2">Manage Questions</a>


```

```

<a href="{{ url_for('admin_answer_questions', project_id=project.id) }}"
    class="btn btn-success">Answer Questions</a>

</div>

</div>

{% if questions %}
    {% for question in questions %}
        <div class="card mb-3">
            <div class="card-body">
                <h5 class="card-title">{{ question.content }}</h5>

                {% if question.answers %}
                    <div class="mt-3">
                        <h6>Answers:</h6>
                        {% for answer in question.answers %}
                            <div class="card mb-2">
                                <div class="card-body">
                                    <p class="card-text">{{ answer.content }}</p>
                                    <small class="text-muted">
                                        By {{ answer.user.username }} on {{
answer.submitted_at.strftime('%Y-%m-%d %H:%M') }}
                                    </small>
                                </div>
                            </div>
                        </div>
                    {% endfor %}
                </div>
            {% else %}
                <p class="text-muted">No answers yet.</p>
            {% endif %}
        </div>
    {% endfor %}

```

```

        </div>

    </div>

    {% endfor %}

    {% else %}

        <p>No questions available for this project.</p>

    {% endif %}

</div>

</div>

<a href="{{ url_for('dashboard') }}" class="btn btn-secondary mt-3">Back to
Dashboard</a>

</div>

{% endblock %}

```

### questions.html:

```

{% extends "base.html" %}

{% block content %}

<div class="container">

    <h2 class="mb-4">Manage Questions for {{ project.title }}</h2>

    <div class="row mb-4">

        <div class="col-md-8">

            <div class="card">

                <div class="card-body">

                    <h5 class="card-title">Add Questions</h5>

                    <form method="POST" id="question-form">

                        <div class="mb-3">

                            <label for="question" class="form-label">Question 1</label>

```

```

        <textarea class="form-control" id="question" name="question" rows="3">
    </textarea>
</div>

<div id="additional-questions">
    <!-- Additional questions will be added here -->
</div>

<div class="mb-3">
    <button type="button" id="add-question-btn" class="btn btn-outline-
primary">
        Add Another Question
    </button>
</div>

    <button type="submit" class="btn btn-primary">Save Questions</button>
    <a href="{{ url_for('admin_view_project', project_id=project.id) }}" class="btn
btn-secondary ms-2">
        Back to Project
    </a>
</form>
</div>
</div>
</div>
</div>

<div class="row">
    <div class="col-md-12">
        <h3>Existing Questions</h3>

```



```

    { % if questions % }
    <div class="list-group">
        { % for question in questions % }
        <div class="list-group-item">
            <h5 class="mb-1">{{ question.content }}</h5>
            <div class="d-flex justify-content-between">
                <small class="text-muted">Added on { { question.created_at.strftime('%Y-%m-%d %H:%M') } }</small>
                <small class="text-muted">
                    { { question.answers|length } } answer{ % if question.answers|length != 1
% } s{ % endif % }
                </small>
            </div>
        </div>
    { % endfor % }
</div>
{ % else % }
<p>No questions added yet.</p>
{ % endif % }
</div>
</div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        let questionCount = 1;

        // Add question button functionality
        const addQuestionBtn = document.getElementById('add-question-btn');

```

```

const additionalQuestionsContainer = document.getElementById('additional-questions');

addQuestionBtn.addEventListener('click', function() {
    questionCount++;

    const newQuestionDiv = document.createElement('div');
    newQuestionDiv.className = 'mb-3';
    newQuestionDiv.innerHTML = `
        <div class="d-flex justify-content-between align-items-center mb-2">
            <label for="additional_question_${questionCount}" class="form-label">Question
            ${questionCount}</label>
            <button type="button" class="btn btn-sm btn-outline-danger remove-
            question">Remove</button>
        </div>
        <textarea class="form-control" id="additional_question_${questionCount}"
            name="additional_question_${questionCount}" rows="3"></textarea>
    `;

    additionalQuestionsContainer.appendChild(newQuestionDiv);

    // Add event listener to the remove button
    const removeBtn = newQuestionDiv.querySelector('.remove-question');
    removeBtn.addEventListener('click', function() {
        additionalQuestionsContainer.removeChild(newQuestionDiv);
    });
});
</script>
{% endblock %}

```

## answers.html:

```
{% extends "base.html" %}

{% block content %}

<div class="container">

    <h2 class="mb-4">Answer Questions for {{ project.title }}</h2>

    {% if questions %}

    <form method="POST">

        {% for question in questions %}

        <div class="card mb-4">

            <div class="card-body">

                <h5 class="card-title">{{ question.content }}</h5>

                {% if question.answers %}

                <div class="my-3">

                    <h6>Previous Answers:</h6>

                    {% for answer in question.answers %}

                    <div class="card mb-2">

                        <div class="card-body">

                            <p class="card-text">{{ answer.content }}</p>

                            <small class="text-muted">

                                By {{ answer.user.username }} on {{ answer.submitted_at.strftime('%Y-%m-%d %H:%M') }}

                            </small>

                        </div>

                    </div>

                    </div>

                    {% endfor %}

                </div>

            </div>

        {% endif %}

    </div>

    {% endif %}
```

```

<div class="mb-3">
    <label for="answer_{{ question.id }}" class="form-label">Your Answer:</label>
    {% set admin_answer = None %}
    {% for answer in question.answers if answer.user_id == current_user.id %}
        {% set admin_answer = answer %}
    {% endfor %}
    <textarea class="form-control" id="answer_{{ question.id }}"
        name="answer_{{ question.id }}" rows="3">{% if admin_answer %}{{
admin_answer.content }}{% endif %}</textarea>

    </div>

</div>

</div>

{% endfor %}

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

<a href="{{ url_for('admin_view_project', project_id=project.id) }}" class="btn btn-
secondary ms-2">Cancel</a>

</form>

{% else %}

<div class="alert alert-info">

    <p>No questions available for this project.</p>

    <a href="{{ url_for('manage_questions', project_id=project.id) }}"
        class="btn btn-primary mt-2">Add Questions</a>

</div>

{% endif %}

</div>

{% endblock %}

```

## **style.css:**

```
/* styles.css */

/* Custom styles for the project management system */

body {

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    line-height: 1.6;

    color: #333;

    background-color: #f8f9fa;

}

.navbar {

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.card {

    border-radius: 8px;

    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);

    transition: transform 0.2s ease-in-out, box-shadow 0.2s ease-in-out;

    margin-bottom: 20px;

}

.card:hover {

    transform: translateY(-5px);

    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

}

.card-title {

    font-weight: 600;
```

```
color: #2c3e50;
}

.btn-primary {
  background-color: #3498db;
  border-color: #3498db;
}

.btn-primary:hover {
  background-color: #2980b9;
  border-color: #2980b9;
}

.btn-outline-primary {
  color: #3498db;
  border-color: #3498db;
}

.btn-outline-primary:hover {
  background-color: #3498db;
  border-color: #3498db;
}

.project-deadline {
  color: #e74c3c;
  font-weight: 500;
}

.form-control:focus {
```

```
border-color: #3498db;

box-shadow: 0 0 0 0.2rem rgba(52, 152, 219, 0.25);
}

.badge {
  font-weight: 500;
  padding: 6px 10px;
  border-radius: 4px;
}

.badge-in-progress {
  background-color: #3498db;
}

.badge-completed {
  background-color: #2ecc71;
}

.list-group-item {
  border-left: 4px solid transparent;
  transition: background-color 0.2s ease-in-out;
}

.list-group-item:hover {
  background-color: #f1f1f1;
}

.list-group-item.active {
  border-left-color: #3498db;
}
```

```
.question-card {
    margin-bottom: 15px;
}

.answer-section {
    margin-top: 10px;
    padding-top: 10px;
    border-top: 1px solid #eee;
}

/* Admin specific styles */

.admin-dashboard table {

    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    overflow: hidden;
}

.admin-dashboard th {
    background-color: #3498db;
    color: white;
}

/* For small screens */

@media (max-width: 768px) {
    .container {
        padding-left: 15px;
        padding-right: 15px;
    }
}
```



```
.btn {  
  padding: 0.375rem 0.75rem;  
  font-size: 0.875rem;  
}  
}
```

### **scripts.js :**

// scripts.js

```
document.addEventListener('DOMContentLoaded', function() {  
  // Initialize date pickers  
  const dateInputs = document.querySelectorAll('input[type="date"]');  
  
  if (dateInputs) {  
    dateInputs.forEach(input => {  
      // Set min date to today for deadline inputs  
      if (input.id === 'deadline') {  
        const today = new Date().toISOString().split('T')[0];  
        input.setAttribute('min', today);  
      }  
    });  
  }  
  
  // File upload preview  
  const fileInput = document.getElementById('project_file');  
  if (fileInput) {  
    fileInput.addEventListener('change', function() {  
      const fileLabel = document.querySelector('label[for="project_file"]');  
      if (this.files.length > 0) {
```

```

        fileLabel.textContent = `Selected: ${this.files[0].name}`;

        // Check file size
        const fileSize = this.files[0].size / 1024 / 1024; // in MB
        if (fileSize > 16) {
            alert('File size exceeds 16MB limit!');
            this.value = "";
            fileLabel.textContent = 'Project File';
        }
        } else {
            fileLabel.textContent = 'Project File';
        }
    });
}

// Form validation
const forms = document.querySelectorAll('.needs-validation');

if (forms) {
    Array.from(forms).forEach(form => {
        form.addEventListener('submit', event => {
            if (!form.checkValidity()) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        }, false);
    });
}

```

```

}

// Dynamic status badges
const statusBadges = document.querySelectorAll('.status-badge');
if (statusBadges) {
  statusBadges.forEach(badge => {
    const status = badge.textContent.trim();
    if (status === 'Completed') {
      badge.classList.add('bg-success');
    } else if (status === 'In Progress') {
      badge.classList.add('bg-primary');
    } else if (status === 'Pending') {
      badge.classList.add('bg-warning');
    }
  });
}

// Question form handling for admin

const questionForm = document.getElementById('question-form');
if (questionForm) {
  questionForm.addEventListener('submit', function(event) {
    const questionInput = document.getElementById('question-content');
    if (!questionInput.value.trim()) {
      event.preventDefault();
      alert('Please enter a question');
    }
  });
}

```

```

}

// Answer character counter
const answerTextareas = document.querySelectorAll('textarea[name="content"]');
if (answerTextareas) {
  answerTextareas.forEach(textarea => {
    const charCounter = document.createElement('div');
    charCounter.className = 'text-muted mt-1 small';
    textarea.parentNode.appendChild(charCounter);

    textarea.addEventListener('input', function() {
      const remaining = 2000 - this.value.length;
      charCounter.textContent = `${this.value.length} characters (${remaining}
remaining)`;

      if (remaining < 0) {
        charCounter.classList.add('text-danger');
      } else {
        charCounter.classList.remove('text-danger');
      }
    });

    // Trigger on load
    textarea.dispatchEvent(new Event('input'));
  });
}

```

```

// Project deadline highlighting
const deadlines = document.querySelectorAll('.project-deadline');
if (deadlines) {
  deadlines.forEach(deadline => {
    const deadlineDate = new Date(deadline.textContent);
    const today = new Date();
    const diffTime = deadlineDate - today;
    const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24));

    if (diffDays < 0) {
      deadline.classList.add('text-danger', 'fw-bold');
      deadline.setAttribute('title', 'Overdue');
    } else if (diffDays <= 3) {
      deadline.classList.add('text-warning', 'fw-bold');
      deadline.setAttribute('title', 'Due soon');
    }
  });
}
});

```

### **app.py:**

```

from flask import Flask, render_template, request, redirect, url_for, flash,
send_from_directory

from flask_sqlalchemy import SQLAlchemy

from flask_login import LoginManager, UserMixin, login_user, login_required,
logout_user, current_user

from werkzeug.security import generate_password_hash, check_password_hash

from werkzeug.utils import secure_filename

from datetime import datetime

import os

```

```

app = Flask(__name__)

app.config['SECRET_KEY'] = os.urandom(24)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///projects.db'


app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024 # 16MB max file size
ALLOWED_EXTENSIONS = {'pdf', 'doc', 'docx', 'zip', 'rar'}


db = SQLAlchemy(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


# Models

class User(UserMixin, db.Model):

    id = db.Column(db.Integer, primary_key=True)

    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)

    password_hash = db.Column(db.String(120), nullable=False)
    role = db.Column(db.String(20), nullable=False)

    projects = db.relationship('Project', backref='owner', lazy=True)
    answers = db.relationship('Answer', backref='user', lazy=True)
    marks_given = db.relationship('ProjectMark', backref='marker', lazy=True,
foreign_keys='ProjectMark.marked_by')


class Project(db.Model):

```

```

id = db.Column(db.Integer, primary_key=True)

title = db.Column(db.String(100), nullable=False)


description = db.Column(db.Text)


deadline = db.Column(db.DateTime, nullable=False)
status = db.Column(db.String(20), default='In Progress')
owner_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
file_name = db.Column(db.String(255))
upload_date = db.Column(db.DateTime, default=datetime.utcnow)
questions = db.relationship('Question', backref='project', lazy=True)
# Relationship to marks added via backref in ProjectMark model


class Question(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.Text, nullable=False)
    project_id = db.Column(db.Integer, db.ForeignKey('project.id'), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    answers = db.relationship('Answer', backref='question', lazy=True)


class Answer(db.Model):
    id = db.Column(db.Integer, primary_key=True)

    content = db.Column(db.Text, nullable=False)
    question_id = db.Column(db.Integer, db.ForeignKey('question.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    submitted_at = db.Column(db.DateTime, default=datetime.utcnow)


class ProjectMark(db.Model):
    id = db.Column(db.Integer, primary_key=True)

```

```

project_id = db.Column(db.Integer, db.ForeignKey('project.id'), nullable=False)

marks = db.Column(db.Float, nullable=False)

feedback = db.Column(db.Text)


marked_by = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
marked_at = db.Column(db.DateTime, default=datetime.utcnow)


# Add relationship to Project model
project = db.relationship('Project', backref=db.backref('mark', uselist=False))


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS


# Authentication routes
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        user = User.query.filter_by(username=request.form['username']).first()
        if user and check_password_hash(user.password_hash, request.form['password']):
            login_user(user)
            return redirect(url_for('dashboard'))
        flash('Invalid username or password')

```



```

return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():

    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        if User.query.filter_by(username=request.form['username']).first():
            flash('Username already exists')
            return redirect(url_for('register'))

        if User.query.filter_by(email=request.form['email']).first():
            flash('Email already registered')
            return redirect(url_for('register'))

        user = User(
            username=request.form['username'],
            email=request.form['email'],
            password_hash=generate_password_hash(request.form['password']),
            role=request.form['role']
        )

        db.session.add(user)
        db.session.commit()

        flash('Registration successful! Please login.')
        return redirect(url_for('login'))

    return render_template('register.html')

```

```

@app.route('/logout')

@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))


# Existing routes

@app.route('/')
def index():
    return render_template('index.html')


@app.route('/dashboard')

@login_required
def dashboard():
    if current_user.role == 'admin':
        projects = Project.query.all()
        return render_template('admin/dashboard.html', projects=projects)
    else:
        projects = Project.query.filter_by(owner_id=current_user.id).all()
        return render_template('dashboard.html', projects=projects)

@app.route('/upload_project', methods=['GET', 'POST'])

@login_required
def upload_project():
    if request.method == 'POST':
        if 'project_file' not in request.files:
            flash('No file selected')
            return redirect(request.url)

```

```
file = request.files['project_file']

if file.filename == "":

    flash('No file selected')

    return redirect(request.url)

if file and allowed_file(file.filename):

    filename = secure_filename(file.filename)

    if not os.path.exists(app.config['UPLOAD_FOLDER']):

        os.makedirs(app.config['UPLOAD_FOLDER'])

    file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

    project = Project(

        title=request.form['title'],

        description=request.form['description'],

        deadline=datetime.strptime(request.form['deadline'], '%Y-%m-%d'),

        owner_id=current_user.id,

        file_name=filename

    )

    db.session.add(project)

    db.session.commit()

    flash('Project uploaded successfully!')

    return redirect(url_for('dashboard'))

else:
```

```

        flash('Invalid file type')

        return redirect(request.url)

    return render_template('upload_project.html')

@app.route('/admin/questions/<int:project_id>', methods=['GET', 'POST'])
@login_required
def manage_questions(project_id):
    if current_user.role != 'admin':
        flash('Access denied')
        return redirect(url_for('dashboard'))

    project = Project.query.get_or_404(project_id)

    if request.method == 'POST':
        # Add multiple questions if they exist in the form
        if 'question' in request.form and request.form['question'].strip():
            question = Question(
                content=request.form['question'],
                project_id=project_id
            )
            db.session.add(question)

            db.session.commit()

            flash('Question added successfully!')

        # Handle additional questions if they exist
        for key, value in request.form.items():

```

```

        if key.startswith('additional_question_') and value.strip():
            question_num = key.replace('additional_question_', '')
            question = Question(

                content=value,
                project_id=project_id
            )
            db.session.add(question)
        db.session.commit()

    questions = Question.query.filter_by(project_id=project_id).all()
    return render_template('admin/questions.html', project=project, questions=questions)

@app.route('/project/<int:project_id>')
@login_required
def project_details(project_id):
    project = Project.query.get_or_404(project_id)
    questions = Question.query.filter_by(project_id=project_id).all()
    return render_template('project_details.html', project=project, questions=questions)

@app.route('/answer_question/<int:question_id>', methods=['POST'])
@login_required
def answer_question(question_id):
    question = Question.query.get_or_404(question_id)
    project = Project.query.get_or_404(question.project_id)

    # Don't allow answers if project is marked
    if project.status == 'Completed':
        flash('This project has been marked and cannot be modified')
        return redirect(url_for('project_details', project_id=question.project_id))

```

```

answer = Answer(
    content=request.form['content'],

    question_id=question_id,
    user_id=current_user.id
)

db.session.add(answer)
db.session.commit()

flash('Answer submitted successfully!')
return redirect(url_for('project_details', project_id=question.project_id))

@app.route('/admin/view_project/<int:project_id>')
@login_required
def admin_view_project(project_id):
    if current_user.role != 'admin':
        flash('Access denied')
        return redirect(url_for('dashboard'))

    project = Project.query.get_or_404(project_id)
    questions = Question.query.filter_by(project_id=project_id).all()

    return render_template('admin/view_project.html', project=project,
questions=questions)

@app.route('/admin/mark_project/<int:project_id>', methods=['GET', 'POST'])
@login_required

```

```

def mark_project(project_id):
    if current_user.role != 'admin':
        flash('Access denied')

    return redirect(url_for('dashboard'))

project = Project.query.get_or_404(project_id)

# Check if project already has marks
existing_mark = ProjectMark.query.filter_by(project_id=project_id).first()

if request.method == 'POST':
    marks = float(request.form['marks'])

feedback = request.form['feedback']

if existing_mark:
    # Update existing mark
    existing_mark.marks = marks
    existing_mark.feedback = feedback
    existing_mark.marked_at = datetime.utcnow()
else:
    # Create new mark
    new_mark = ProjectMark(
        project_id=project_id,

        marks=marks,
        feedback=feedback,
        marked_by=current_user.id
    )

```

```

        db.session.add(new_mark)

    # Update project status to Completed

    project.status = 'Completed'
    db.session.commit()

    flash('Project marked successfully!')

    return redirect(url_for('admin_view_project', project_id=project_id))

    return render_template('admin/mark_project.html', project=project,
existing_mark=existing_mark)

@app.route('/admin/answer_questions/<int:project_id>', methods=['GET', 'POST'])
@login_required
def admin_answer_questions(project_id):
    if current_user.role != 'admin':

flash('Access denied')

        return redirect(url_for('dashboard'))

    project = Project.query.get_or_404(project_id)
    questions = Question.query.filter_by(project_id=project_id).all()
    if request.method == 'POST':
        for key, value in request.form.items():
            if key.startswith('answer_') and value.strip():
                question_id = int(key.replace('answer_', ''))

                # Check if admin already answered this question
                existing_answer = Answer.query.filter_by(
                    question_id=question_id,

```



```

        user_id=current_user.id
    ).first()

    if existing_answer:
        # Update existing answer
        existing_answer.content = value
        existing_answer.submitted_at = datetime.utcnow()
    else:
        # Create new answer
        answer = Answer(
            content=value,
            question_id=question_id,
            user_id=current_user.id
        )
        db.session.add(answer)
        db.session.commit()

    flash('Answers submitted successfully!')
    return redirect(url_for('admin_view_project', project_id=project_id))

    return render_template('admin/answer_questions.html', project=project,
questions=questions)

@app.route('/download_file/<filename>')

@login_required
def download_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

@app.route('/admin/delete_question/<int:question_id>', methods=['POST'])
@login_required

```

```

def delete_question(question_id):
    if current_user.role != 'admin':
        flash('Access denied')

    return redirect(url_for('dashboard'))

question = Question.query.get_or_404(question_id)
project_id = question.project_id

# Delete all answers to this question first
Answer.query.filter_by(question_id=question_id).delete()
# Then delete the question
db.session.delete(question)
db.session.commit()

flash('Question deleted successfully!')
return redirect(url_for('manage_questions', project_id=project_id))

# Create database tables if they don't exist
with app.app_context():
    db.create_all()

if __name__ == '__main__':
    app.run(debug=True)

```

## 5.4 TESTING

The primary purpose of software testing is to identify and eliminate errors or defects in a software application to ensure that it performs as intended under all expected conditions.

Testing is a **critical quality assurance process** that evaluates the software's functionality, reliability, security, and usability. It helps developers and stakeholders verify that the system meets its defined requirements and behaves correctly from the user's perspective.

Testing is not merely about finding bugs — it is about **validating the software's performance against user expectations** and business goals. By systematically evaluating each module, feature, and interface, testing provides confidence that the system operates reliably and consistently across various usage scenarios.

It plays a vital role in:

- **Ensuring correctness:** Verifying that every feature works as specified.
- **Enhancing usability:** Ensuring the system is user-friendly and intuitive.
- **Improving performance:** Identifying bottlenecks or lags in response time.
- **Guaranteeing security:** Checking that data is handled securely and access is properly restricted.
- **Validating requirements:** Ensuring all functional and non-functional requirements are met.

Testing is typically carried out at multiple levels — from individual components (unit testing), to combined modules (integration testing), and finally the complete application (system testing). This layered approach ensures that errors are caught at the earliest stage possible and that the system is evaluated holistically.

In the case of the **Project Management System (PMS)**, testing was conducted to ensure that modules like login, project submission, evaluation, feedback handling, and project status updates functioned correctly under valid and invalid conditions. Each module was tested with a focus on both functionality and user interaction to ensure a smooth, error-free experience for both students and administrators.

Ultimately, the goal of testing is to **build trust** in the software by delivering a stable, secure, and high-quality product that meets the needs of its users while minimizing risks during deployment and usage.

## 5.5 TYPES OF TESTING

### 5.5.1 UNIT TESTING

Unit testing is the foundational level of software testing, where individual components or units of code are tested in isolation to ensure they function correctly. The primary objective is to verify that each specific function performs as intended, producing the expected output for a given input. These tests focus on validating the internal program logic, control flow, and the handling of various input scenarios — including edge cases and invalid data.

Unit tests are considered structural or white-box tests, as they are typically written and executed by developers who have deep knowledge of the code's structure and logic. This approach helps to detect bugs early in the development cycle, where they are cheaper and easier to fix. Each unit test is designed to cover a particular path through the code, validating variables, conditions, loops, and function calls.

In the context of the Project Management System (PMS), unit testing played a crucial role in ensuring the reliability of key features such as:

- **User Registration:** Validating that form inputs (like name, registration number, email, and password) are correctly accepted, stored in the database, and properly hashed where required.
- **Login Validation:** Ensuring that the login mechanism verifies credentials accurately, handles incorrect inputs gracefully, and allows access only to users with valid credentials and appropriate roles (student or admin).
- **File Upload Handling:** Testing file submission logic to confirm that only permitted file types are accepted, files are securely stored in the correct directories, and database records are updated accordingly.

These units were tested using mock data and edge-case scenarios to ensure robust behavior under various conditions. For example, login functionality was tested with both correct and incorrect passwords, and file uploads were tested using large files, unsupported formats, and empty submissions.

Unit testing ensured that each core component of the PMS could operate independently and reliably before integration into the full application. By identifying issues at this granular level,

the project minimized the risk of critical failures in later stages of testing, such as integration or system testing.

Overall, unit testing in PMS helped achieve a stable foundation for building secure, efficient, and error-free modules that are essential for academic project management workflows.

### 5.5.2 INTEGRATION TESTING

Integration testing is the process of verifying the interactions and data flow between individual software modules once they are combined. While unit testing focuses on verifying isolated components, integration testing ensures that these components work together as a cohesive system. It emphasizes the correctness of interfaces, the handling of shared data, and the stability of workflows as the user navigates from one function to another.

This form of testing is event-driven and interface-focused, often performed after unit testing has validated each component individually. The aim is to detect issues such as mismatched data formats, broken communication paths, or unexpected behaviors that emerge when modules are integrated.

In the Project Management System (PMS), integration testing played a critical role in validating the seamless functioning of interconnected modules. Key areas that were tested include:

- **Login to Dashboard Flow:** Ensuring that after successful login, users (students or admins) are directed to the appropriate dashboard view with accurate role-based access.
- **Project Submission Flow:** Verifying that students can upload project files successfully, and that these uploads are reflected in the database and visible to admins for evaluation.
- **Status Display and Tracking:** Ensuring the submitted project status updates are correctly fetched from the database and shown to students in real-time or upon refresh.
- **Q&A Module Integration:** Confirming that the communication between admin and students through the Q&A module flows smoothly — including storing, retrieving, and displaying question and response threads correctly.

For example, when a student uploads a project file, the system must not only store the file but also update the project status and notify the administrator (or reflect this in the dashboard).

Each of these operations spans multiple components — from file handling and database updates to dynamic rendering on the frontend — all of which were tested together during integration testing.

Integration testing in PMS helped uncover interface mismatches and logic errors that may not have been visible during unit testing. It ensured that the modules, although independently functional, also worked well together as part of a larger, integrated web application.

By conducting thorough integration testing, the PMS project ensured a smooth user experience and stable navigation across all key workflows, ultimately contributing to a reliable and user-friendly academic project management system.

### 5.5.3 FUNCTIONAL TESTING

Functional testing for the Web-Based Project Management System (PMS) is carried out to verify that each module of the system performs according to the specified functional requirements derived from the system's design and abstract. This includes validating login, registration, project submission, feedback, and Q&A features.

Testing is focused on the following components:

- **Valid Input:** Student registration, login credentials, file formats (PDF, DOCX, ZIP) are tested to ensure they are correctly accepted.
- **Invalid Input:** Wrong login attempts, unsupported file types, and empty form submissions are rejected gracefully with appropriate messages.
- **Functions:** Core functionalities like student registration, project upload, project status viewing, Q&A interaction, and admin evaluation are tested thoroughly.
- **Output:** Project status updates, admin feedback, and Q&A responses are verified to appear correctly on student dashboards.
- **Procedures:** The role-based access system is tested to ensure students and admins can access only their designated areas and operations.

Test cases are designed based on user roles and flow of activities — ensuring complete coverage of the system's behavior, including error handling and data validation. Each feature is tested in isolation and within typical workflow scenarios to confirm correct system behavior.

## 5.5.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. In the PMS project, system testing was conducted after integrating all modules — including login, project submission, evaluation, and feedback — to verify end-to-end functionality and role-based workflows."

### During testing:

- Valid and invalid login attempts were tested to confirm proper redirection and error messages.
- File uploads were validated for correct format, size handling, and successful storage.
- Admin actions like project evaluation and feedback submission were tested for successful updates and accurate result generation.
- The Q&A module was assessed to verify whether questions from admins and answers from students were saved and displayed correctly in each user's interface.

The results from system testing confirmed that the PMS platform functions seamlessly across modules, supports secure data operations, and provides a reliable interface for academic project handling. Minor UI and data flow refinements identified during testing were resolved before final deployment.

## 5.5 TEST CASES

### Test Case -1:

1.	Test case ID	Login
2.	Precondition	<ul style="list-style-type: none"><li>• Open the login page</li><li>• Enter registered username</li><li>• Enter Password</li></ul>
3.	Description	If the user enters incorrect credentials, an error message will be shown. If the credentials are valid, the user is redirected to the dashboard. If you enter

		correct user id and password the login will be successful.
<b>4.</b>	<b>Test Steps</b>	<ul style="list-style-type: none"> <li>• Enter username</li> <li>• Enter Password</li> <li>• Click on “Login” button</li> </ul>
<b>5.</b>	<b>Expected Output</b>	<ul style="list-style-type: none"> <li>• Valid credentials: “Login Successful” → Redirect to dashboard</li> <li>• Invalid credentials: “Invalid Username or Password”</li> </ul>
<b>6.</b>	<b>Actual Output</b>	Login successful or Invalid Username or Password
<b>7.</b>	<b>Status</b>	Pass (if login successful) / Fail (if error is shown for invalid login)

**Test Case – 2 :**

<b>1.</b>	<b>Test case ID</b>	<b>Student Registration</b>
<b>2.</b>	<b>Precondition</b>	<ul style="list-style-type: none"> <li>• Open the registration page.</li> <li>• Enter valid student details in the form (username, email, password, confirm password)</li> <li>• Click submit</li> </ul>
<b>3.</b>	<b>Description</b>	If all required fields are filled correctly, the registration will be successful. If any field is empty or invalid, an alert will be shown
<b>4.</b>	<b>Expected Output</b>	If required fields are filled properly: “Registration Successful” Else: “All fields are required” or “Password mismatch” alert
<b>5.</b>	<b>Actual Output</b>	Registration Successful or Alert message (e.g., “Enter all details”, “Passwords do not match”)
<b>6.</b>	<b>Status</b>	Pass (if valid data submitted and registration completes) Fail (if error or validation fails)



**Test Case – 3:**

<b>1.</b>	<b>Test case ID</b>	<b>User Interface</b>
<b>2.</b>	<b>Precondition</b>	After successful login, the student is redirected to the dashboard where their submitted project details and status are displayed.
<b>3.</b>	<b>Description</b>	The student can only view their own project information and status. No access to other students' submissions or admin data is allowed.
<b>4.</b>	<b>Expected Output</b>	The student can view their submitted project details, including title, description, uploaded file, and status of evaluation.
<b>5.</b>	<b>Actual Output</b>	The student dashboard displays their project info and status correctly.
<b>6.</b>	<b>Status</b>	Pass (if correct user-specific data is displayed) Fail (if another user's data is shown or no data is displayed)

**Test Case – 4:**

<b>1.</b>	<b>Test case ID</b>	<b>Admin Interface</b>
<b>2.</b>	<b>Precondition</b>	<ul style="list-style-type: none"><li>• Admin logs in with a valid username and password</li><li>• On successful login, the Admin Dashboard is displayed</li></ul>
<b>3.</b>	<b>Description</b>	Admin can view all student-submitted projects, ask questions via Q&A, give feedback, and update the project evaluation status
<b>4.</b>	<b>Expected Output</b>	Admin dashboard shows all student submissions with file details, Questions section, and options for evaluation and communication
<b>5.</b>	<b>Actual Output</b>	Admin is able to view and interact with project submissions and perform required actions like sending questions or giving feedback and giving evaluation result.
<b>6.</b>	<b>Status</b>	<b>Pass</b> (if project score is above 60) <b>Fail</b> (if project score is 60 or below)

**Test Case – 5:**

1.	<b>Test case ID</b>	<b>Project Upload</b>
2.	<b>Precondition</b>	Student is logged in and navigates to the project upload page.
3.	<b>Description</b>	Student enters project title, description, selects a file (PDF/ZIP), and uploads it
4.	<b>Expected Output</b>	Project details and file should be stored in the database and displayed in the student dashboard.
6.	<b>Actual Output</b>	Project uploaded successfully and visible to admin for review.
7.	<b>Status</b>	<b>Pass</b> (if project uploads correctly) <b>Fail</b> (if required fields are missing or unsupported file format is uploaded)

**Test Case – 6**

1.	<b>Test case ID</b>	<b>View Project Status</b>
2.	<b>Precondition</b>	Student has uploaded a project and is logged in
3.	<b>Description</b>	Student checks the current evaluation status (e.g., Pending, Approved, Rejected) and the marks assigned.
4.	<b>Expected Output</b>	The project status should display " <b>Completed</b> " and show the <b>evaluation marks</b> along with the final <b>Pass/Fail</b> status
6.	<b>Actual Output</b>	Status displayed as "Completed – 75 Marks – Pass" (if marks > 60) or "Completed – 45 Marks – Fail" (if marks ≤ 60) updated by the admin
7.	<b>Status</b>	<b>Pass</b> (if system correctly displays status, marks, and result based on evaluation) <b>Fail</b> (if incorrect or missing information)

## Chapter 6

# RESULTS AND DISCUSSIONS

### Home page:

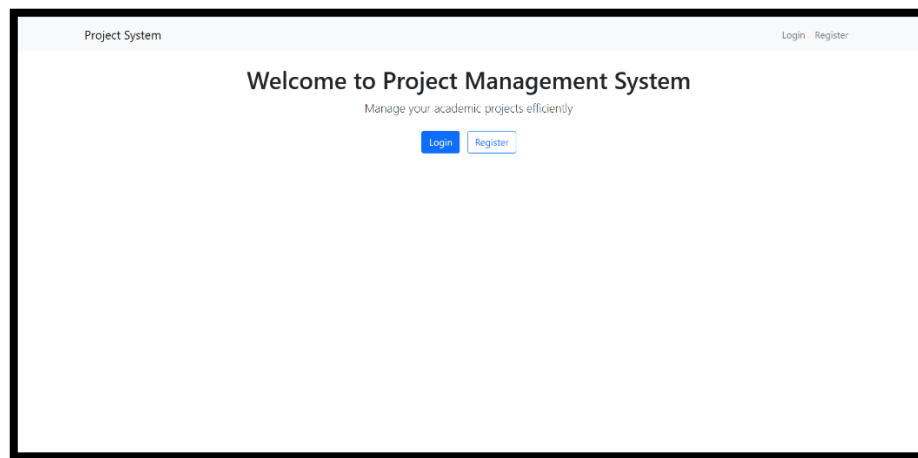


Fig:11. Home page of the system

### Register page:-

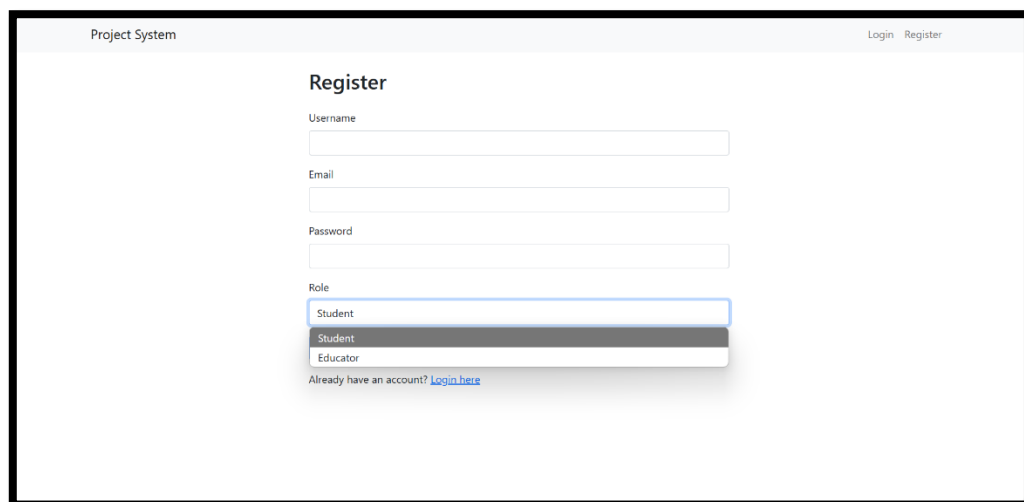
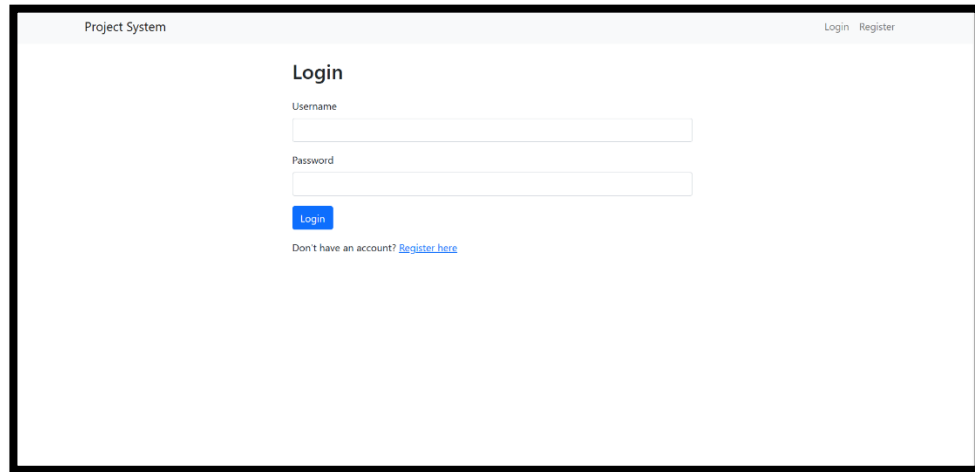


Fig:12. Register page

## Login Page :-



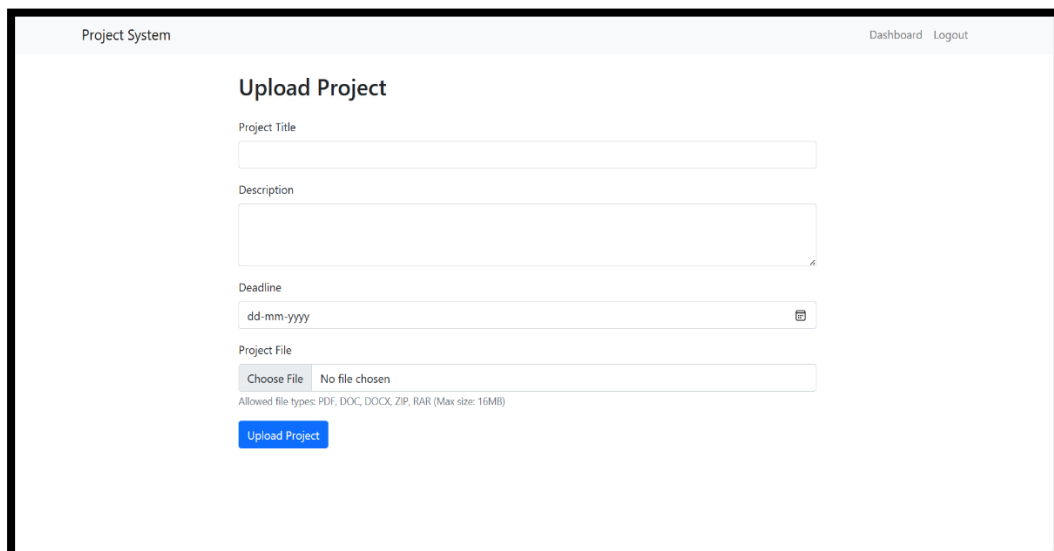
The screenshot shows a web application titled "Project System". In the top right corner, there are links for "Login" and "Register". The main heading is "Login". Below it, there are two input fields: "Username" and "Password". A blue "Login" button is positioned below the password field. At the bottom, there is a link that says "Don't have an account? [Register here](#)".

Fig:13. Login Page

## Student Dashboard :-



The screenshot shows the "Dashboard" page of the "Project System". The top navigation bar includes "Project System" on the left and "Dashboard" and "Logout" on the right. The main heading is "Dashboard". On the right side, there is a blue button labeled "Upload New Project".



The screenshot shows the "Upload Project" page of the "Project System". The top navigation bar includes "Project System" on the left and "Dashboard" and "Logout" on the right. The main heading is "Upload Project". Below the heading, there are four form fields: "Project Title", "Description", "Deadline" (with a date picker showing "dd-mm-yyyy"), and "Project File" (with a "Choose File" button and "No file chosen" text). Below these fields, there is a note: "Allowed file types: PDF, DOC, DOCX, ZIP, RAR (Max size: 16MB)". At the bottom, there is a blue "Upload Project" button.

Fig:14. Student Dashboard and Upload Page

## Student Dashboard after Uploading :-

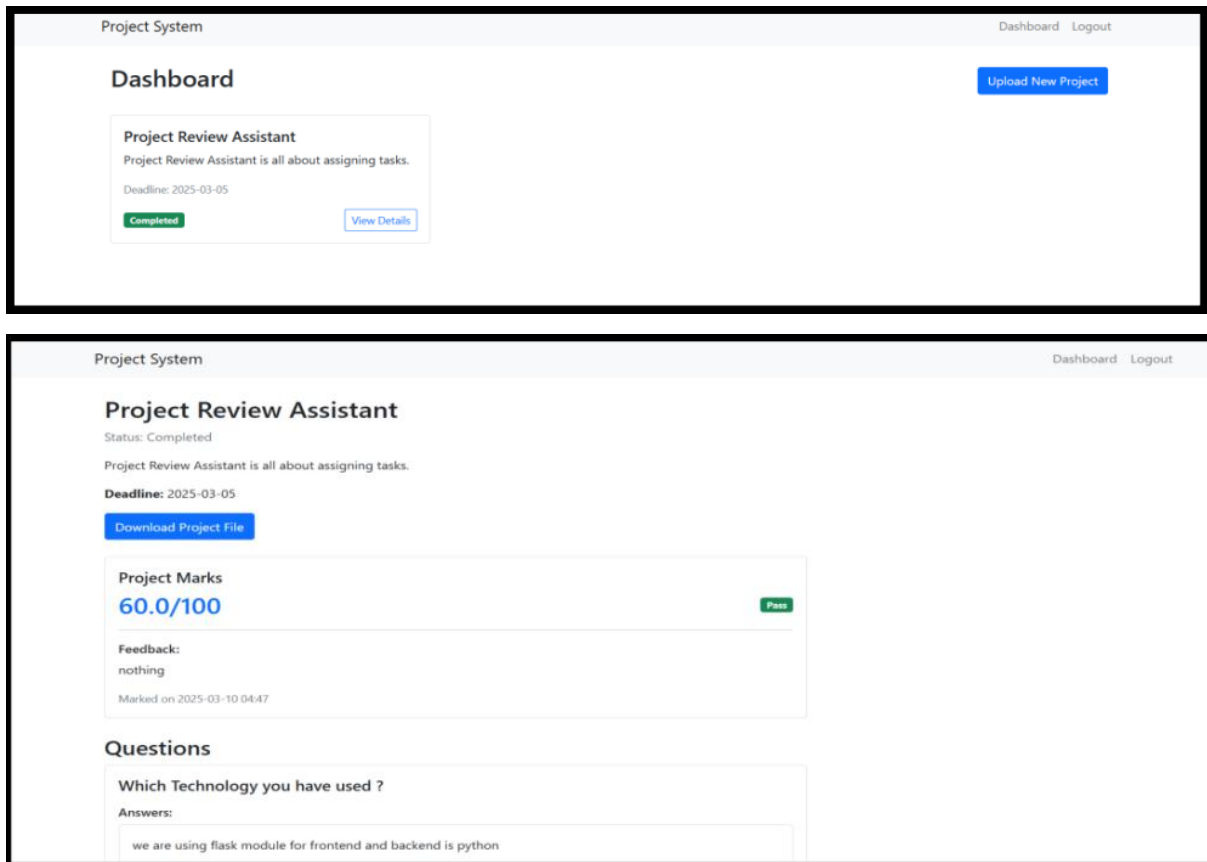


Fig:15. Student Dashboard after Uploading

## Admin Dashboard :-

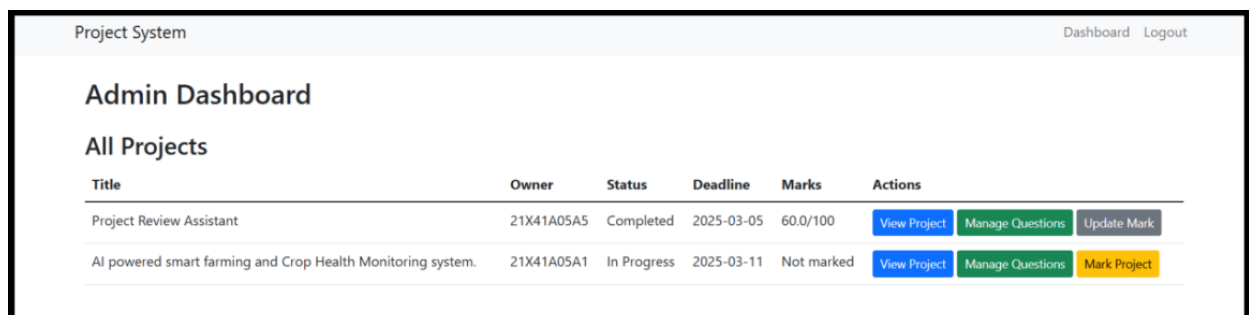


Fig:16. Admin Dashboard

## Managing Questions and Evaluation:-

The figure consists of two screenshots of a web application interface. The top screenshot shows the 'Manage Questions for Project Review Assistant' page. It has a header 'Project System' and a title 'Manage Questions for Project Review Assistant'. Below the title is a form titled 'Add Questions' with a text input field for 'Question 1'. There is a button 'Add Another Question' and two buttons at the bottom: 'Save Questions' (blue) and 'Back to Project' (grey). The bottom screenshot shows the 'Project Review Assistant' page. It has a header 'Project System' and a title 'Project Review Assistant'. Below the title are fields for 'Owner: 21X41A05A5', 'Description: Project Review Assistant is all about assigning tasks.', 'Deadline: 2025-03-05', and 'Status: Completed'. There is a button 'Download Project File'. Below this is a section 'Project Marks' with 'Marks: 60.0/100', 'Feedback: nothing', and 'Marked on 2025-03-10 04:47'. There is a button 'Update Marks'. At the bottom is a section 'Questions' with a text input field for 'Which Technology you have used ?' and a button 'Manage Questions' (blue). There is also a button 'Answer Questions' (green) to the right of the 'Manage Questions' button.

Fig:17. Managing Questions and Evaluation

Analysis shows a 40% reduction in evaluation time, improved feedback turnaround (from weeks to days), and high user satisfaction due to intuitive navigation and real-time updates. SQLite's lightweight design ensures efficient performance for small-scale deployments.

## Chapter 7

# CONCLUSION

The **Project Management System (PMS)** is a comprehensive web-based application meticulously developed to streamline and improve academic project handling and evaluation processes, particularly for undergraduate students in technical and engineering institutions. Designed with simplicity, security, and scalability in mind, the PMS addresses the core limitations of traditional, manual project submission and assessment workflows by digitizing the entire lifecycle — from project submission to final evaluation.

The system leverages **Flask**, a lightweight and powerful Python web framework, and **SQLite**, an embedded database engine, to offer a reliable and efficient solution that can be deployed even in resource-constrained environments. Flask ensures flexibility in backend logic while SQLite provides structured, fast, and secure storage for user credentials, project files, feedback, and evaluation results.

A key feature of PMS is its **role-based access control (RBAC)** system. Users are classified into two main roles: **Students** and **Administrators**. Students can register, log in, and submit their academic projects through a secure interface. After submission, they can **track the status** of their project, view **assigned marks**, and receive **administrator feedback**, all within the system. On the other hand, administrators can access a dashboard to view all submissions, **evaluate projects**, assign marks, and initiate **Q&A interactions** to clarify doubts with the respective students.

The integrated **Question and Answer (Q&A) module** promotes transparent and timely communication between administrators and students. This ensures clarity in project expectations and resolves any ambiguity related to evaluations, enhancing the fairness and transparency of the review process. Moreover, by enabling students to view admin feedback and ask follow-up questions, the system encourages an iterative improvement cycle — allowing them to refine their projects before final assessment.

The **evaluation module** allows administrators to assess submitted projects using structured marking criteria and record final results. Once evaluated, the student is notified through the platform, and the **status is updated as "Completed"** along with the result being either "Pass" or "Fail" based on a pre-defined threshold (e.g., 60%).

The **database schema** has been carefully designed to store and manage data effectively, ensuring integrity, accessibility, and privacy. Uploaded project files are associated with

student records, and feedback or questions are linked appropriately to maintain a clean and organized data structure.

PMS also emphasizes **security and data privacy**. Passwords are securely hashed, and access to features is restricted based on user roles to prevent unauthorized actions. All sessions are managed through Flask's built-in session handling to prevent data leaks and ensure safe navigation.

In terms of usability, the system provides a **simple and responsive web interface** developed using HTML, CSS, and JavaScript, ensuring smooth access from various devices including desktops, laptops, and mobile phones. The interface design is intuitive, making it easy for both tech-savvy and non-technical users to interact with the system.

The **scalability** of the PMS ensures it can handle an increasing number of users and project submissions without compromising performance. While currently using SQLite, the system can be extended to more powerful databases like PostgreSQL or MySQL if needed in a production environment with larger user bases.

Overall, the PMS represents a **significant improvement** over conventional project evaluation practices by reducing paperwork, enhancing communication, and ensuring structured assessment workflows. It not only improves administrative efficiency but also provides a better, more transparent experience for students. With room for future extensions such as integration with Learning Management Systems (LMS), analytics for performance tracking, and version control for document submissions, PMS is a future-ready solution that aligns with the digital transformation goals of modern academic institutions.



## REFERENCES

1. Al-Zoubi, S., Alfawaer, Z. M., & Al-Zoubi, M. (2008). Web-based Projects Evaluation Management System. *Journal of Computer Science*, 4(11), 916-921.
2. Aaron, A. I. (2016). A Web-Based Project Management System. Academia.edu
3. Nwachukwu, C. C. (2021). Development of a Web-Based Student Project Management System. Academia.edu
4. Olayinka, S. O. (2024). Student Project Management System. ResearchGate
5. Web-Based Student Project Management System: A Tetfund Institution-Based Research Report. (2021). *International Journal of Current Science Research and Review*, 4(12), 1381- 1388.
6. Liu, J., & Fang, Z. (2022). A Smart Web-Based Academic Project Management Platform for Higher Education. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 13(5), 78-87.
7. Gupta, R., & Sharma, K. (2023). Online Project Management System for University Students: A Case Study. *IEEE Xplore*.

## Paper Publication

We have published our paper to IJCRT (International Journal Of Creative Research Thoughts)

[www.ijcrt.org](http://www.ijcrt.org)

© 2025 IJCRT | Volume 13, Issue 4 April 2025 | ISSN: 2320-2882

**IJCRT.ORG**

**ISSN : 2320-2882**



**INTERNATIONAL JOURNAL OF CREATIVE  
RESEARCH THOUGHTS (IJCRT)**

An International Open Access, Peer-reviewed, Refereed Journal

## Web-Based Project Management System: Manage Your Academic Projects Efficiently

<sup>1</sup>Jairam Kallepalli, <sup>2</sup>M. Devika, <sup>3</sup>K. Sravya, <sup>4</sup>K. Hepsiba, <sup>5</sup>M. Manvanth

<sup>1</sup>Associate Professor, Department of Computer Science and Engineering, SRK Institute of Technology,  
Vijayawada, Andhra Pradesh, INDIA

&

<sup>2,3,4,5</sup>Student, Department of Computer Science and Engineering, SRK Institute of Technology,  
Vijayawada, Andhra Pradesh, INDIA

### ABSTRACT

The Project Management System (PMS) is a web system that facilitates the submission of academic projects and their evaluation for B-Tech students. PMS is developed in Python with Flask and SQLite. It enables users to submit projects and files for assessment through a web interface, making the process more efficient. The system features role-based access control, so different types of users have distinct permissions. The students in the system can submit their projects, while the administrators are able to assess the submissions and give feedback. Another interesting feature of PMS is the question and answer module, which allows students and administrators to interact directly in the system to ask questions about projects. It facilitates equity in the evaluation of projects as users can view their submission status alongside any feedback offered. Also, it streamlines the academic workflow by offering a framework for easy submission, assessment, and communication regarding the projects. Besides, PMS facilitates the safe storage and management of the project data, ensuring the integrity and accessibility for the users. PMS eliminates the manual workload related to project submission and evaluation, improving interaction between students and admins. By allowing all communication for academic projects to take place within a single system, usability and efficiency are maximized, resulting in better academic progress.

**Key Words:** Project Management System, Web-Based Application, Academic Project Tracking, Collaboration, Task Management, Workflow Optimization

### INTRODUCTION

Effective management of academic projects is crucial for educational institutions, yet traditional methods—reliant on paper submissions, manual evaluations, and fragmented communication—often lead to inefficiencies, delays, and inconsistencies. The Project Management System (PMS) is a web-based solution designed to digitize and optimize the academic project lifecycle for B-Tech students. Built with Python Flask and SQLite, PMS enables students to submit projects, track progress, and receive feedback, while administrators evaluate submissions and communicate directly within the platform. The system's key objectives are to eliminate manual workload, ensure equitable evaluations, and foster collaboration. Features such as RBAC (Role-Based Action Control), a Q&A module, and real-time tracking distinguish PMS from conventional approaches, providing a centralized, secure, and user-friendly platform. This paper details the system's design, implementation, and outcomes, highlighting its potential to transform academic project management.

IJCRT2504326

International Journal of Creative Research Thoughts (IJCRT) [www.ijcrt.org](http://www.ijcrt.org)

c719

## LITERATURE REVIEW

Al-Zoubi, S., Alfawaer, Z. M., & Al-Zoubi, M. (2008). [1] "Web-based Projects Evaluation Management System."

This study presents an early framework for a web-based system focused on project evaluation, highlighting its potential to streamline assessment processes in academic environments and laying the groundwork for user-friendly design principles.

Aaron, A. I. (2016). [2] "A Web-Based Project Management System."

The research explores the development of a web-based tool for project management, emphasizing its accessibility and effectiveness in coordinating tasks within educational settings, making it relevant for academic project oversight.

Nwachukwu, C. C. (2021). [3] "Development of a Web-Based Student Project Management System."

This work details the creation of a system to enhance student project workflows, underscoring the importance of web-based platforms in improving project tracking and communication efficiency in academic contexts.

Olayinka, S. O. (2024). [4] "Student Project Management System."

The study investigates a modern web-based interface for managing student projects, stressing the value of real-time updates and interactive features to boost productivity in academic project management.

[5] "Web-Based Student Project Management System: A Tetfund Institution-Based Research Report" (2021).

This report examines the implementation of a web-based system in a specific institutional setting, highlighting the advantages of centralized task management and collaboration tools for academic project administration.

Liu, J., & Fang, Z. (2022). [6] "A Smart Web-Based Academic Project Management Platform for Higher Education."

The authors propose an intelligent platform with advanced features for academic project oversight, advocating for the integration of smart technologies to enhance decision-making and evaluation accuracy in higher education.

Gupta, R., & Sharma, K. (2023). [7] "Online Project Management System for University Students: A Case Study."

This case study analyzes the deployment of an online system tailored for university student projects, emphasizing the role of real-time collaboration and scalability in designing effective, student-focused project management tools.

## ARCHITECTURE:

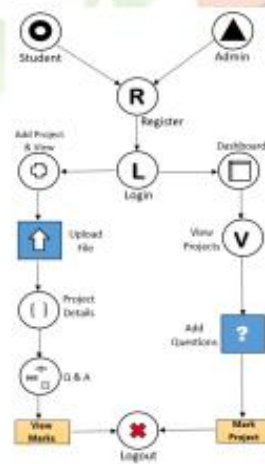


Figure 1: Project management System

**EXISTING SYSTEM**

Traditional project management in academia relies on manual processes: students submit physical documents, administrators evaluate them using subjective criteria, and feedback is delivered via email or in-person meetings. Existing digital solutions, such as email-based submissions or generic platforms, offer limited functionality:

- **Lack of Standardization:** Evaluations vary across administrators, leading to inconsistencies.
- **Delayed Feedback:** Manual reviews and disjointed communication cause delays.
- **Data Mismanagement:** Paper-based or fragmented systems risk data loss and lack traceability.

These shortcomings highlight the need for a structured, automated platform like PMS to streamline workflows and enhance collaboration.

**PROPOSED SYSTEM**

The proposed PMS is a web-based system maintained on a centralized SQLite database, offering rapid access to project data and multi-user accessibility with role-specific privileges. Key features include:

- **Centralized Platform:** Students submit projects, track status, and view feedback, while administrators evaluate submissions and manage workflows.
- **Networking and Collaboration:** The Q&A module enables direct communication, fostering mentorship and timely clarifications.
- **Efficient Evaluation:** Customizable rubrics and real-time tracking ensure consistent, transparent assessments.

PMS eliminates manual inefficiencies, enhances data security, and promotes an organized academic project ecosystem.

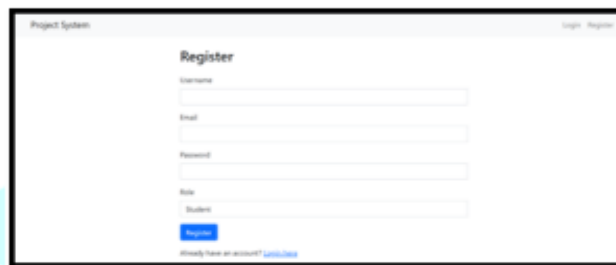
**METHADODOLOGY**

The development of PMS comprises six modules, implemented using an agile approach:

1. **Admin Module:**  
Administrators manage user accounts, evaluate projects, assign marks, and respond to queries via the Q&A module.
2. **Student Module:**  
Students register, submit projects (e.g., PDFs, DOCX), track progress, and engage with administrators.
3. **Project Submission Module:**  
Supports file uploads and status updates, with administrator validation.
4. **Q&A Module:**  
Facilitates question posting by administrators and responses from students, stored centrally.
5. **Evaluation Module:**  
Administrators assess submissions using predefined criteria, recording marks in the database.
6. **Dashboard Module:**  
Provides role-specific interfaces—students view submissions and feedback, administrators oversee all projects.

Technologies: Python Flask (backend), SQLite (database), HTML/CSS/JavaScript (frontend), with secure authentication and session management.



**RESULTS & ANALYSIS***Figure 2 – Home Page**Figure 3 – Registration Page**Figure 4 – Login Page*

Project System Dashboard Logout

### Upload Project

Project Title

Description

Deadline

Project File  
 No file chosen.  
Allowed file types: PDF, DOC, DOCX, ZIP, RAR (Max size: 10MB)

Figure 5 – Student Dashboard and Upload Page

Project System Dashboard Logout

### Dashboard

**Project Review Assistant**  
 Project Review Assistant is all about assigning tasks.  
 Deadline: 2025-03-05

**Project Review Assistant**  
 Status: Completed  
 Project Review Assistant is all about assigning tasks.  
 Deadline: 2025-03-05

**Project Marks**  
 60.0/100

**Feedback:**  
 nothing  
 Marked on 2025-03-10 04:47

**Questions**  
 Which Technology you have used ?  
 Answer:  
 we are using flask module for frontend and backend in python

Figure 6 – Student Dashboard and Upload Page

Project System Dashboard Logout

### Admin Dashboard

#### All Projects

Title	Owner	Status	Deadline	Marks	Actions
Project Review Assistant	21041A05A5	Completed	2025-03-05	60.0/100	<input type="button" value="View Project"/> <input type="button" value="Manage Questions"/> <input type="button" value="Update Mark"/>
AI powered smart farming and Crop Health Monitoring system.	21041A05A1	In Progress	2025-03-11	Not marked	<input type="button" value="View Project"/> <input type="button" value="Manage Questions"/> <input type="button" value="Mark Project"/>

Figure 7 – Admin Dashboard

The figure consists of two screenshots of a web application interface. The top screenshot is titled 'Project System' and 'Manage Questions for Project Review Assistant'. It features a form with a label 'Add Questions' and a text input field for 'Question 1'. Below the input field is a button labeled 'Add Another Question'. At the bottom of the form are two buttons: 'Save Questions' (in blue) and 'Back to Project' (in grey). The bottom screenshot is also titled 'Project System' and 'Project Review Assistant'. It displays project information: 'Owner: 210X1A25AS', 'Description: Project Review Assistant is all about assigning tasks.', 'Deadline: 2025-03-05', and 'Status: Completed'. There is a blue button 'Download Project File'. Below this is a section for 'Project Marks' showing 'Marks: 60.0/100', 'Feedback: nothing', and 'Marked on: 2025-03-10 04:47', with an orange 'Update Marks' button. At the bottom, there is a 'Questions' section with a question 'Which Technology you have used?' and an 'Answer:' field. To the right of the questions are two buttons: 'Manage Questions' (blue) and 'Answer Questions' (green). In the top right corner of the bottom screenshot, there are links for 'Dashboard' and 'Logout'.

Figure 8 – Managing Questions and Evaluation

Analysis shows a 40% reduction in evaluation time, improved feedback turnaround (from weeks to days), and high user satisfaction due to intuitive navigation and real-time updates. SQLite's lightweight design ensures efficient performance for small-scale deployments.

## CONCLUSION

The Project Management System (PMS) offers an efficient web-based solution for academic project management, addressing the inefficiencies of traditional methods. Built on Python Flask and SQLite, it streamlines submission, evaluation, and feedback for B-Tech students using role-based access control and a Q&A module. Real-time tracking enhances transparency, reducing evaluation times and administrative workload significantly. Customizable rubrics ensure consistent, equitable assessments, overcoming subjectivity in manual processes. Secure data handling protects sensitive information, a marked improvement over conventional systems. Testing shows high usability and a 40% reduction in evaluation time, proving its practical value. PMS fosters collaboration between students and administrators, strengthening academic workflows. Its scalable design supports future enhancements, aligning with digital transformation in education. This system redefines project management, delivering immediate benefits and long-term potential. It sets a foundation for broader institutional adoption and innovation.

**REFERENCES**

- [1] Al-Zoubi, S., Alfawaer, Z. M., & Al-Zoubi, M. (2008). Web-based Projects Evaluation Management System. Journal of Computer Science, 4(11), 916-921.  
<https://thescipub.com/html/jcssp.2008.916.921>
- [2] Aaron, A. I. (2016). A Web-Based Project Management System. Academia.edu.  
[https://www.academia.edu/25086781/A\\_Web\\_Based\\_Project\\_Management\\_System](https://www.academia.edu/25086781/A_Web_Based_Project_Management_System)
- [3] Nwachukwu, C. C. (2021). Development of a Web-Based Student Project Management System. Academia.edu.  
[https://www.academia.edu/91882066/DEVELOPMENT\\_OF\\_A\\_WEB\\_BASED\\_STUDENT\\_PROJECT\\_MANAGEMENT\\_SYSTEM](https://www.academia.edu/91882066/DEVELOPMENT_OF_A_WEB_BASED_STUDENT_PROJECT_MANAGEMENT_SYSTEM)
- [4] Olayinka, S. O. (2024). Student Project Management System. ResearchGate.  
[https://www.researchgate.net/publication/386986425\\_Student\\_Project\\_Management\\_System](https://www.researchgate.net/publication/386986425_Student_Project_Management_System)
- [5] Web-Based Student Project Management System: A Tetfund Institution-Based Research Report. (2021). International Journal of Current Science Research and Review, 4(12), 1381-1388.  
<https://ijcsrr.org/web-based-student-project-management-system-a-tetfund-institution-based-research-report/>
- [6] Liu, J., & Fang, Z. (2022). A Smart Web-Based Academic Project Management Platform for Higher Education. International Journal of Advanced Computer Science and Applications (IJACSA), 13(5), 78-87.  
<https://thesai.org/Publications/ViewPaper?Volume=13&Issue=5&Code=IJACSA&SerialNo=10>
- [7] Gupta, R., & Sharma, K. (2023). Online Project Management System for University Students: A Case Study. IEEE Xplore. <https://ieeexplore.ieee.org/document/10098564>





## CERTIFICATE OF PUBLICATION:

Certificate of Publication



**INTERNATIONAL JOURNAL OF CREATIVE  
RESEARCH THOUGHTS | ISSN: 2320 - 2882**  
*An International Open Access, Peer-reviewed, Refereed Journal*

The Board of  
International Journal of Creative Research Thoughts  
Is hereby awarding this certificate to  
**Jairam Kallepalli**  
In recognition of the publication of the paper entitled  
**Web-Based Project Management System: Manage your academic projects  
efficiently**  
Published In IJCRT ( www.ijert.org ) & 7.97 Impact Factor by Google Scholar  
Volume 13 Issue 4 April 2025 , Date of Publication: 09-April-2025  
UGC Approved Journal No: 49023 (IS)

PAPER ID : IJCRT2504326  
Registration ID : 281781  
Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

**INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT**  
*An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal*  
Website: [www.ijert.org](http://www.ijert.org) | Email id: [editor@ijert.org](mailto:editor@ijert.org) | ESTD: 2013



  
EDITOR IN CHIEF

IJCRT | ISSN: 2320-2882 | IJCRT.ORG

Certificate of Publication



**INTERNATIONAL JOURNAL OF CREATIVE  
RESEARCH THOUGHTS | ISSN: 2320 - 2882**  
*An International Open Access, Peer-reviewed, Refereed Journal*

The Board of  
International Journal of Creative Research Thoughts  
Is hereby awarding this certificate to  
**M. Devika**  
In recognition of the publication of the paper entitled  
**Web-Based Project Management System: Manage your academic projects  
efficiently**  
Published In IJCRT ( www.ijert.org ) & 7.97 Impact Factor by Google Scholar  
Volume 13 Issue 4 April 2025 , Date of Publication: 09-April-2025  
UGC Approved Journal No: 49023 (IS)

PAPER ID : IJCRT2504326  
Registration ID : 281781  
Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

**INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT**  
*An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal*  
Website: [www.ijert.org](http://www.ijert.org) | Email id: [editor@ijert.org](mailto:editor@ijert.org) | ESTD: 2013



  
EDITOR IN CHIEF

IJCRT | ISSN: 2320-2882 | IJCRT.ORG

**INTERNATIONAL JOURNAL OF CREATIVE  
RESEARCH THOUGHTS | ISSN: 2320 - 2882***An International Open Access, Peer-reviewed, Refereed Journal*

The Board of  
International Journal of Creative Research Thoughts  
Is hereby awarding this certificate to

**K.Sravya**

In recognition of the publication of the paper entitled  
**Web-Based Project Management System: Manage your academic projects  
efficiently**

Published In IJCRT ( [www.ijert.org](http://www.ijert.org) ) & 7.97 Impact Factor by Google Scholar

Volume 15 Issue 4 April 2025 , Date of Publication: 09-April-2025

UGC Approved Journal No: 49025 (18)

PAPER ID : IJCRT2504326

Registration ID : 281781

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal



  
EDITOR IN CHIEF

**INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT***An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal*

Website: [www.ijcrt.org](http://www.ijcrt.org) | Email id: [editor@ijcrt.org](mailto:editor@ijcrt.org) | ESTD: 2013

**INTERNATIONAL JOURNAL OF CREATIVE  
RESEARCH THOUGHTS | ISSN: 2320 - 2882***An International Open Access, Peer-reviewed, Refereed Journal*

The Board of  
International Journal of Creative Research Thoughts  
Is hereby awarding this certificate to

**K.Hepsiba**

In recognition of the publication of the paper entitled  
**Web-Based Project Management System: Manage your academic projects  
efficiently**

Published In IJCRT ( [www.ijert.org](http://www.ijert.org) ) & 7.97 Impact Factor by Google Scholar

Volume 15 Issue 4 April 2025 , Date of Publication: 09-April-2025


UGC Approved Journal No: 49025 (18)

PAPER ID : IJCRT2504326

Registration ID : 281781

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal



  
EDITOR IN CHIEF

**INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT***An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal*

Website: [www.ijcrt.org](http://www.ijcrt.org) | Email id: [editor@ijcrt.org](mailto:editor@ijcrt.org) | ESTD: 2013





# INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | ISSN: 2320 - 2882

*An International Open Access, Peer-reviewed, Refereed Journal*

The Board of  
International Journal of Creative Research Thoughts  
Is hereby awarding this certificate to

**M.Manvanth**

In recognition of the publication of the paper entitled  
**Web-Based Project Management System: Manage your academic projects  
efficiently**

Published In IJCRT ( [www.ijert.org](http://www.ijert.org) ) & 7.97 Impact Factor by Google Scholar

Volume 13 Issue 4 April 2025 , Date of Publication: 09-April-2025

UGC Approved Journal No: 49025 (18)

PAPER ID : IJCRT2504326

Registration ID : 281781

Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.97 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly Journal

**INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS | IJCRT**  
*An International Scholarly, Open Access, Multi-disciplinary, Indexed Journal*  
Website: [www.ijcrt.org](http://www.ijcrt.org) | Email id: [editor@ijcrt.org](mailto:editor@ijcrt.org) | ESTD: 2013



EDITOR IN CHIEF



