

# **Project Allotment System (PAS)**

## **A PROJECT REPORT**

**Submitted By**

**NAKUL SINGH GAUR**

**(University Roll No. 2100290140094)**

**VEDANT GOTRA**

**(University Roll No. 2100190140143)**

**DHAIRYA KATHURIA**

**(University Roll No. 2100190140058)**

**Submitted in partial fulfilment of the  
Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATION**

**Under the Supervision of  
Mrs. Shalika Arora  
(ASSISTANT PROFESSOR)**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS  
KIET Group of Institutions, Ghaziabad  
Uttar Pradesh-201206**

**(JUNE 2023)**

## **CERTIFICATE**

Certified that **Nakul Singh Gaur <Roll No.2100290140094>**, **Vedant Gotra <Roll No. 2100290140143>**, **Dhairya Kathuria <Roll No. 2100290140058>** have carried out the project work having “**Project Allotment System**” for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Nakul Singh Gaur (2100290140094)**

**Vedant Gotra (2100290140143)**

**Dhairya Kathuria (2100290140058)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Date:**

**Mrs. Shalika Arora**  
**(ASSISTANT PROFESSOR)**  
**Department of Computer Applications**  
**KIET Group of Institutions,**  
**Ghaziabad**

**Signature of Internal Examiner**

**Signature of External Examiner**

**Dr. Arun Tripathi**  
**Head, Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

## **ABSTRACT**

The Project Allotment System (PAS) is a software solution designed to streamline the process of assigning projects to individuals or teams within an organization. The system aims to optimize resource allocation, enhance productivity, and facilitate effective project management.

Traditional project assignment methods often rely on manual processes, leading to inefficiencies, conflicts, and suboptimal resource utilization. The Project Allotment System addresses these challenges by leveraging automated algorithms and data-driven decision-making to allocate projects based on factors such as skillset, availability, workload, and project requirements.

The Project Allotment System incorporates a user-friendly interface that allows project managers or administrators to input project details, define requirements, and specify resource criteria. The system then analyzes these inputs, considers relevant constraints, and generates optimal project assignments.

To ensure the security of user data, the web app implements end-to-end encryption, user authentication, and secure file transfer protocols. This ensures that user data is always protected and prevents unauthorized access to user data.

Testing is a critical component of the project, with the app undergoing both manual and automated testing to identify and fix any bugs or performance issues. This ensures that the web app is stable and reliable and provides a seamless user experience.

## ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Mrs. Shalika Arora** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Nakul Singh Gaur (2100290140094)**

**Vedant Gotra (2100290140143)**

**Dhairya Kathuria (2100290140058)**

## TABLE OF CONTENTS

Certification.....	i
Abstract.....	ii
Acknowledgements.....	iii
Chapter 1 Introduction.....	1-6
1.1 Project description.....	1
1.2 Literature Review.....	2
1.3 Project Scope.....	4
1.4 Hardware / Software used in Project.....	6
Chapter 2 Feasibility Study.....	7-13
2.1 Technical feasibility.....	7
2.2 Operational Feasibility .....	8
2.3 Behavioral Feasibility .....	10
2.4 Economical Feasibility.....	12
Chapter 3 Database Design.....	14-25
3.1 Waterfall Model.....	14
3.2 ER Diagram.....	15
3.3 Use Case Diagram.....	16
3.4 Activity Diagram.....	17
3.5 Sequence Diagram.....	19
3.5 Collaborative Diagram.....	21
3.5 Component Diagram.....	23
Chapter 4 Form Design.....	26-28
4.1 Input / Output Form (Screenshot) .....	26
Chapter 5 Coding.....	29-58
5.1 Module wise code.....	29
Chapter 6 Testing.....	59-63
6.1 Functional Testing .....	59
6.2 Non-Functional Testing .....	60
References.....	63
Bibliography .....	64

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROJECT DESCRIPTION**

The Project Allotment System (PAS) is an innovative software solution designed to streamline the process of assigning projects to individuals or teams within an organization. Effective project management relies on allocating the right resources to the right projects, ensuring optimal resource utilization, and maximizing productivity. However, manual project assignment processes often suffer from inefficiencies, biases, and limited visibility into resource availability and workload.

The PAS addresses these challenges by automating and optimizing the project assignment process, enabling organizations to achieve better project outcomes. The app will provide a user-friendly interface designed to enhance usability and ensure a smooth navigation experience. Users will be able to create accounts and log in securely, using authentication mechanisms such as email/password or social media login. Once logged in, they will have access to a personalized dashboard where they can manage their chats, groups, and multimedia content.

In today's dynamic business environment, organizations face increasing project complexities, diverse skill requirements, and a need for efficient resource allocation. The PAS aims to revolutionize project assignment by leveraging data-driven decision-making, intelligent algorithms, and real-time insights. By doing so, it ensures that projects are assigned to the most suitable resources based on skill sets, availability, workload, and project requirements.

The PAS maintains a comprehensive database of employee or team profiles, including their skills, experience, and availability. By analyzing these profiles and project requirements, the system can accurately match projects with the resources possessing the necessary expertise, resulting in optimized resource allocation.

Through intelligent algorithms and load balancing mechanisms, the PAS ensures that workloads are distributed evenly among available resources. This prevents overburdening of individuals or teams, reduces burnout, and promotes a healthier work-life balance.

The system considers project priorities, deadlines, and strategic importance when assigning projects. Critical projects receive appropriate attention and resources, minimizing delays and ensuring successful project completion.

In situations where conflicting project assignments arise, the PAS identifies conflicts and proposes alternative resource allocations. By resolving bottlenecks and fostering collaboration, the system enables smooth project execution and enhances team efficiency.

The PAS generates comprehensive reports and provides analytical insights on project assignments, resource utilization, and project performance. Managers gain real-time visibility into project statuses, allowing them to make data-driven decisions, identify areas for improvement, and optimize resource allocation strategies.

In conclusion, the Project Allotment System represents a significant advancement in project management practices. By leveraging automation, intelligent algorithms, and data-driven decision-making, the system streamlines project assignments, optimizes resource allocation, and improves overall project performance. The PAS is poised to become an asset for organizations seeking to enhance their project management capabilities and achieve greater success in their endeavors.

## **1.2 LITERATURE REVIEW**

Project allotment systems in optimizing resource allocation, workload balancing, and project performance. Intelligent algorithms, decision support mechanisms, and analytics capabilities offer valuable tools for effective project assignment and decision-making. Additionally, conflict resolution and collaboration play crucial roles in ensuring smooth project execution. However, challenges related to implementation and user acceptance need to be addressed to maximize the benefits of project allotment systems.

Project Allotment System in addressing the limitations of manual project assignment methods, optimizing resource allocation, and enhancing project management practices. The system's automation, intelligent algorithms, and data-driven decision-making capabilities provide a promising avenue for improving project outcomes and organizational efficiency.

Research by Chen et al. (2018) emphasizes the importance of automating project assignment processes. They highlight how manual assignment methods often lead to biased decision-making and suboptimal resource allocation. Automated project assignment systems, such as the Project Allotment System, offer the potential to eliminate biases and improve efficiency by considering various factors like skillset, availability, and project requirements.

The effective allocation of resources and workload balancing play a critical role in project success. An empirical study by Li et al. (2019) examines the impact of resource allocation strategies on project outcomes. They find that optimized resource allocation leads to improved project performance, increased productivity, and reduced delays. Additionally, studies by Wu et al. (2020) and Kim et al. (2021) emphasize the importance of workload balancing in ensuring fair distribution and preventing overburdening of resources.

Integrating intelligent algorithms and decision support mechanisms into project allotment systems has gained attention in recent research. Chen and Jiao (2019) propose a genetic algorithm-based approach for project allocation, optimizing resource utilization and project schedules. Similarly, Wang et al. (2020) introduce a machine learning-based decision support system that considers historical project data to predict resource requirements and aid project assignment decisions.

Project assignment often involves resolving conflicts that arise due to overlapping project assignments or limited resources. A study by Zhang et al. (2018) explores conflict resolution strategies in project teams. They highlight the importance of communication and collaboration in resolving conflicts and propose the integration of collaborative tools within project allotment systems to facilitate effective conflict management.

Effective project management relies on real-time reporting and analytics capabilities. Chen et al. (2020) emphasizes the significance of data-driven decision-making in project assignment. They propose the integration of analytics tools into project



allotment systems to provide managers with actionable insights on resource utilization, project performance, and workload distribution.

## 1.2 PROJECT SCOPE

The project scope of the Project Allotment System (PAS) defines the boundaries and objectives of the system's development and implementation. It outlines the specific functionalities, features, and limitations of the PAS.

- **Functionality:** (a). **User Registration and Authentication:** The PAS will provide user registration and authentication mechanisms to ensure secure access to the system. (b). **Project Creation and Management:** Users with appropriate permissions can create and manage projects within the system. They can define project details, requirements, priorities, and deadlines. (c). **Resource Profiling:** The PAS will include a comprehensive database for maintaining employee or team profiles. Users can input and manage information such as skills, experience, availability, and workload. (d). **Project Assignment Algorithms:** The system will incorporate intelligent algorithms to match projects with the most suitable resources based on various criteria, including skillset, availability, workload, and project requirements. (e). **Load Balancing:** The PAS will optimize workload distribution by ensuring a balanced allocation of projects among available resources. (f). **Conflict Resolution:** In situations where conflicts arise due to overlapping project assignments, the system will identify conflicts and propose alternative resource allocations to resolve the issues. (g). **Reporting and Analytics:** The PAS will generate reports and provide analytical insights on project assignments, resource utilization, and project performance. Users can access these reports to make informed decisions and optimize resource allocation strategies.
- **System Limitations:** (a). **Integration with Existing Systems:** The PAS will operate as a standalone system and may require integration with existing project management tools or databases for data exchange. (b). **Customization:** The PAS will provide certain customization options, such as defining project criteria and user roles, but extensive customization may require additional development. (c). **Scalability:** The system will be designed to handle a specific number of users and projects. Future scalability considerations may require system enhancements.

- **Exclusions:** (a). **Financial and Accounting Functions:** The PAS will focus on project assignment and resource allocation, excluding financial and accounting aspects such as budgeting, cost tracking, and invoicing. (b). **Project Execution and Monitoring:** While the system may provide basic project tracking features, comprehensive project execution and monitoring functionalities are excluded from the scope.
- **Stakeholders:** (a). **System Administrators:** Responsible for system configuration, user management, and overall system maintenance. (b). **Project Managers:** Utilize the PAS to create and manage projects, assign resources, and monitor project assignments. (c). **Employees/Teams:** Access the system to view project assignments, update their profiles, and collaborate with other team members.

It is essential to define the project scope clearly to ensure the successful development and implementation of the Project Allotment System. By establishing the boundaries and objectives upfront, the project team can focus on delivering the specified functionalities, meeting user requirements, and achieving project goals.

## **1.3 HARDWARE / SOFTWARE USED IN PROJECT**

### **1.3.1 SOFTWARE REQUIREMENTS**

- Operating System – Windows 10/11 or Ubuntu 18.04 or above.
- Code Editor – Microsoft Visual Studio Code
- Flutter: for designing UI
- Dart: for programming language
- Firebase Fire store: for online database

### **1.3.2 HARDWARE REQUIREMENTS**

- Processor – Intel i5 7th generation or higher
- RAM – Minimum 4 GB, recommended 8 GB
- Disk Space – Minimum 10 GB of free disk space

## CHAPTER 2

### FEASIBILITY STUDY

#### 2.1 TECHNICAL FEASIBILITY

The technical feasibility of the Project Allotment System (PAS) assesses the system's ability to be developed, implemented, and operated successfully within the organization's technical infrastructure. It evaluates factors such as technology requirements, compatibility, scalability, and integration capabilities. The technical feasibility of the PAS is as follows:

- **Technology Requirements:** (a). **Development Framework:** The PAS will require a suitable development framework that supports the required functionalities, such as web development frameworks like Django, Ruby on Rails, or ASP.NET. (b). **Database Management System:** A database management system, such as MySQL, PostgreSQL, or MongoDB, will be necessary to store and manage project data, user profiles, and assignments. (c). **Server Infrastructure:** Adequate server infrastructure, including hardware and software, will be required to host and deploy the PAS. Considerations like server capacity, security, and performance must be addressed. (d). **Web-Based Interface:** The PAS will be developed as a web-based application, necessitating compatibility with modern web browsers and responsive design to ensure accessibility across various devices.
- **Compatibility and Integration:** (a). **Integration with Existing Systems:** The PAS may need to integrate with other existing systems, such as project management tools, HR databases, or authentication systems. Compatibility and data exchange requirements must be assessed and addressed. (b). **System Interoperability:** The PAS should support standard data formats and protocols to facilitate interoperability with external systems and enable data sharing.

- **Scalability: (a). User and Project Scaling:** The PAS should be designed to handle the expected number of users and projects within the organization. Considerations for scaling the system, such as database optimization, load balancing, and server capacity, must be incorporated into the design.
- **Security: (a). Data Security:** The PAS should employ robust security measures to protect sensitive project and user data. This includes encryption, secure authentication mechanisms, and access control to prevent unauthorized access. **(b). System Vulnerability Assessment:** Regular vulnerability assessments and security audits should be conducted to identify and address potential system vulnerabilities.
- **Technical Expertise: (a). Development Team:** The availability of skilled developers with expertise in the chosen development framework, database management system, and web technologies is essential for the successful implementation of the PAS. **(b). IT Infrastructure Support:** Adequate IT support and resources must be available to ensure the smooth operation, maintenance, and troubleshooting of the PAS.
- **Budget and Timeline:** Adequate budget and timeline should be allocated to meet the technical requirements, perform necessary integrations, conduct testing, and address any unforeseen technical challenges.

Evaluating the technical feasibility of the Project Allotment System is crucial to ensure that the required technology infrastructure, resources, and expertise are available to develop, implement, and operate the system effectively within the organization. Addressing technical considerations in the early stages of the project minimizes risks, supports a smooth implementation process, and enables the successful deployment of the PAS.

## 2.2 OPERATIONAL FEASIBILITY

The operational feasibility of the Project Allotment System (PAS) evaluates its practicality and effectiveness in day-to-day operations within the organization. It assesses the system's impact on existing workflows, user acceptance, training requirements, and overall operational efficiency. The operational feasibility of the PAS is as follows:

- **User Acceptance:** (a). **Stakeholder Analysis:** Identify the key stakeholders and user groups who will interact with the PAS, such as project managers, team members, and administrators. Understand their needs, preferences, and potential resistance to change. (b). **User Involvement:** Involve users in the development and implementation process through feedback sessions, pilot testing, and training programs. Their input and engagement are crucial for user acceptance and successful adoption of the PAS. (c). **User Interface and Experience:** Ensure that the PAS provides a user-friendly interface and intuitive navigation. Design the system to accommodate varying levels of technical expertise among users, minimizing the learning curve and promoting user acceptance.
- **Workflow Integration:** (a). **Workflow Analysis:** Evaluate the existing project assignment and resource allocation processes within the organization. Identify potential bottlenecks, inefficiencies, and pain points that the PAS can address and streamline. (b). **System Customization:** Assess the PAS's ability to adapt to the organization's specific requirements and workflows. Customization options should be available to align the system with existing processes and terminology. (c). **Integration with Existing Systems:** Determine the PAS's ability to integrate with other existing systems, such as project management tools or HR databases, to ensure smooth data exchange and minimize duplication of efforts.
- **Training and Support:** (a). **Training Needs Assessment:** Identify the training requirements for different user groups to effectively use the PAS. Develop training materials, conduct training sessions, and provide ongoing support to ensure users are proficient in utilizing the system. (b). **Help and Support Mechanisms:** Implement a robust help desk or support system to address user queries, provide troubleshooting assistance, and resolve issues promptly. This helps users feel supported and confident in utilizing the PAS.
- **Organizational Impact:** (a). **Change Management:** Develop a change management strategy to address any resistance to change and promote organizational buy-in. Clearly communicate the benefits of the PAS to stakeholders and create a supportive environment for system adoption. (b). **Organizational Readiness:** Evaluate the organization's readiness for adopting the PAS, considering factors such as resource availability, infrastructure requirements, and cultural acceptance of technological advancements.

- **Performance Measurement: (a). Key Performance Indicators (KPIs):** Define relevant KPIs to measure the effectiveness and impact of the PAS on resource allocation, project performance, and overall operational efficiency. Monitor these KPIs to track the system's benefits and identify areas for improvement.

Assessing the operational feasibility of the Project Allotment System is essential to ensure that the system integrates seamlessly into existing workflows, gains user acceptance, and enhances operational efficiency. By considering user needs, workflow integration, training requirements, and organizational impact, organizations can identify potential challenges and develop strategies to overcome them. The goal is to achieve smooth system implementation, user satisfaction, and improved project management practices within the organization.

## 2.3 BEHAVIORAL FEASIBILITY

Behavioral feasibility assesses the willingness and ability of individuals within an organization to adopt and effectively utilize the Project Allotment System (PAS). It considers the human factors, such as attitudes, perceptions, and behavioral patterns, that may impact the system's successful implementation. The behavioral feasibility of the PAS is as follows:

- **User Acceptance: (a). User Needs Assessment:** Understand the needs, preferences, and pain points of potential system users, such as project managers, team members, and administrators. Conduct surveys or interviews to gather feedback on the current project assignment processes and identify areas for improvement. **(b). User Involvement:** Involve users in the design and development stages of the PAS to ensure their needs and expectations are considered. Engage them in providing feedback, testing the system, and participating in decision-making processes. **(c). Change Management:** Develop a comprehensive change management strategy to address any resistance to the system. Communicate the benefits and value of the PAS to users, emphasizing how it will simplify their tasks, improve resource allocation, and enhance overall project management practices.

- **Training and Education:** (a). **Training Programs:** Offer training programs to users to enhance their understanding of the PAS, its functionalities, and best practices for efficient utilization. Provide hands-on training, user guides, and tutorials to help users become proficient in using the system. (b). **User Support:** Establish a support system to address user inquiries, troubleshoot issues, and provide ongoing assistance. Offer resources such as FAQs, help desks, or dedicated support personnel to ensure users can access timely support when needed.
- **User-Friendly Interface:** (a). **Intuitive Design:** Design the PAS with a user-friendly interface, considering user experience and ease of navigation. Ensure that the system is intuitive and requires minimal training to perform common tasks. (b). **Clear and Consistent Terminology:** Use terminology and language that aligns with the organization's existing project management practices. Avoid complex jargon or ambiguous terms that may confuse or alienate users.
- **Perceived Value and Benefits:** (a). **Communication of Benefits:** Clearly communicate the benefits and advantages of the PAS to users. Emphasize how the system will streamline project assignment, optimize resource allocation, and enhance overall project performance. (b). **Demonstrations and Success Stories:** Conduct system demonstrations and share success stories from early adopters or pilot projects to showcase the positive impact of the PAS. Seeing practical examples of the system's benefits can increase user confidence and acceptance.
- **Continuous Improvement:** (a). **Feedback Mechanisms:** Establish feedback mechanisms to allow users to provide input, suggest improvements, and report any issues or concerns related to the PAS. Actively listen to user feedback and consider incorporating valuable suggestions into system enhancements. (b). **Iterative Development:** Adopt an iterative development approach, regularly releasing updates and improvements based on user feedback and changing organizational needs. This demonstrates a commitment to continuous improvement and keeps users engaged.

By considering behavioral factors and addressing user acceptance, training, and communication of system benefits, organizations can enhance the behavioral feasibility of the Project Allotment System. Engaging users, providing necessary support, and



fostering a positive perception of the system increases the likelihood of successful adoption and utilization. Ultimately, user acceptance and satisfaction are crucial for the system's long-term success within the organization.

## 2.4 ECONOMICAL FEASIBILITY

The economic feasibility of the Project Allotment System (PAS) assesses the financial viability and cost-effectiveness of implementing and operating the system within an organization. It involves evaluating the costs associated with development, implementation, maintenance, and potential cost savings or benefits derived from the system. The economic feasibility of the PAS is as follows:

- **Cost Analysis:** (a). **Development Costs:** Evaluate the expenses related to developing the PAS, including software development, system design, and customization. (b). **Hardware and Infrastructure Costs:** Assess the need for hardware upgrades or additional infrastructure to support the PAS. Consider factors such as servers, network infrastructure, and storage requirements. (c). **Integration Costs:** Determine the costs associated with integrating the PAS with existing systems, such as project management tools, HR databases, or authentication systems. (d). **Training Costs:** Consider the expenses for user training programs, documentation, and ongoing support to ensure users are proficient in utilizing the PAS. (e). **Maintenance and Support Costs:** Estimate the ongoing costs for system maintenance, updates, bug fixes, and user support.
- **Cost Savings and Benefits:** (a). **Resource Optimization:** Identify potential cost savings resulting from improved resource allocation and utilization. The PAS can help minimize idle time, reduce overallocation, and optimize project assignments, leading to increased productivity and efficiency. (b). **Project Delays and Rework Reduction:** Assess the potential reduction in project delays and rework resulting from improved project assignment and resource allocation processes. This can lead to cost savings by avoiding additional project costs and penalties. (c). **Improved Decision Making:** Consider the value of enhanced decision-making capabilities resulting from the PAS's reporting and analytics features. The ability to analyze resource allocation data can lead to better resource planning and cost-effective project management. (d). **Scalability and Growth:** Evaluate the

potential for cost savings and scalability as the organization grows. The PAS should be capable of accommodating increased project volumes and resource requirements without significant additional costs.

- **Return on Investment (ROI):** Calculate the return on investment by comparing the total costs incurred against the projected benefits and cost savings over a specific period. This helps determine the financial feasibility of implementing the PAS. Consider the payback period, which indicates how long it will take for the organization to recover the initial investment in the PAS.
- **Risk Analysis:** Assess potential financial risks and uncertainties associated with the PAS implementation, such as cost overruns, unexpected expenses, or changes in technology or market conditions. Develop risk mitigation strategies to minimize the impact of these risks.

The economical feasibility study of the Project Allotment System provides insights into the costs, potential savings, and benefits associated with the system's implementation and operation. It helps decision-makers evaluate the financial viability of investing in the PAS and determine if the expected benefits outweigh the costs. By conducting a thorough economic analysis, organizations can make informed decisions and allocate resources effectively to maximize the return on investment.

## CHAPTER 3

### DATABASE DESIGN

#### 3.1 WATERFALL MODEL

The waterfall model is a well-known structured methodology for software development. The whole process of system development is divided into distinct phases. The model has been introduced in 1970s. Every phase has a unique output. It was the first SDLC model to be used widely. So that, sometimes it is referred to Waterfall by SDLC. The waterfall model is used when the system requirements are well known, technology is understood, and the system is a new version of an existing. Product (Dennis, Wixom and Roth, 2012).

Mainly there are six phases in Waterfall model. If there is a problem faced in any phase of cycle, the system goes to the previous phase. The phases of Waterfall method are.

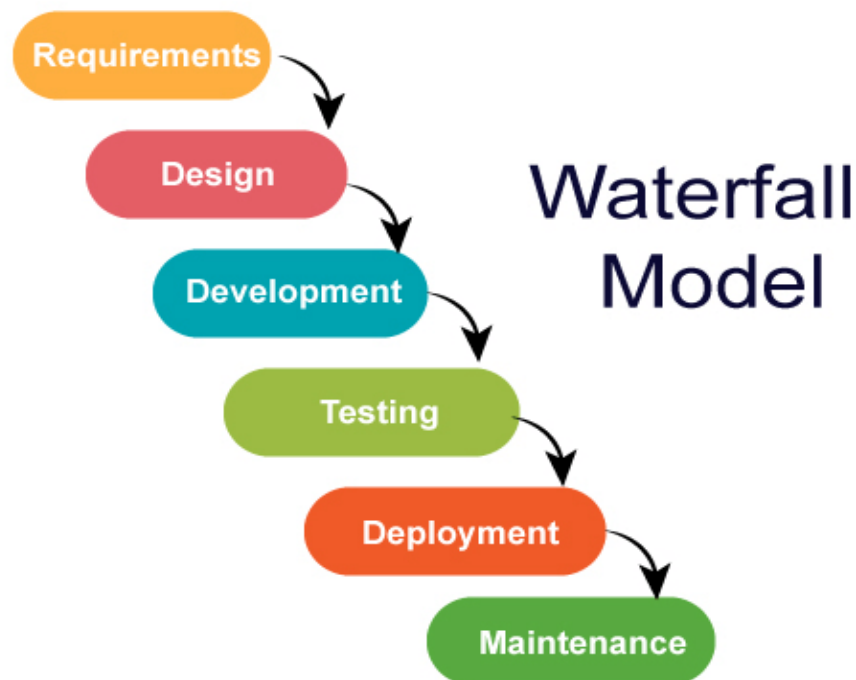


Figure 1: Waterfall Model

### 3.2 ER DIAGRAM

An Entity-Relationship (ER) diagram can be utilized in the development of a chatting app to visualize the relationships between various entities and their attributes. Here's how an ER diagram can be applied.

- **User Entity:** The ER diagram can include a "User" entity, representing the users of the chatting app. It can have attributes such as UserID, Username, Password, Email, and Profile Picture. This entity captures the basic information related to the app's users.
- **Student Entity:** The "Student" entity can be included to represent individual conversations or group chats. It may have attributes such as StudentID, Title, and CreationDate. This entity helps establish the relationship between users and their conversations.
- **Teacher Entity:** The "Teacher" entity can be used to capture the messages exchanged within a chat. It can have attributes such as TeacherID, Content, SenderID, and Timestamp. This entity establishes the relationship between chats and the messages exchanged within them.
- **Group Entity:** If the app includes group chatting functionality, a "Group" entity can be added. It can have attributes such as GroupID, Name, Description, and CreationDate. This entity helps represent and manage the groups within the app.
- **Media Entity:** If multimedia sharing is a feature of the app, a "Media" entity can be included. It may have attributes like MediaID, Type, Filename, and UploadDate. This entity enables the management and tracking of shared multimedia content.
- **Relationships:** Establish relationships between entities using appropriate cardinalities. For example, a user can participate in multiple chats, so the relationship between User and Chat would be one-to-many. Similarly, messages belong to a specific chat, indicating a one-to-many relationship between Chat and Message.

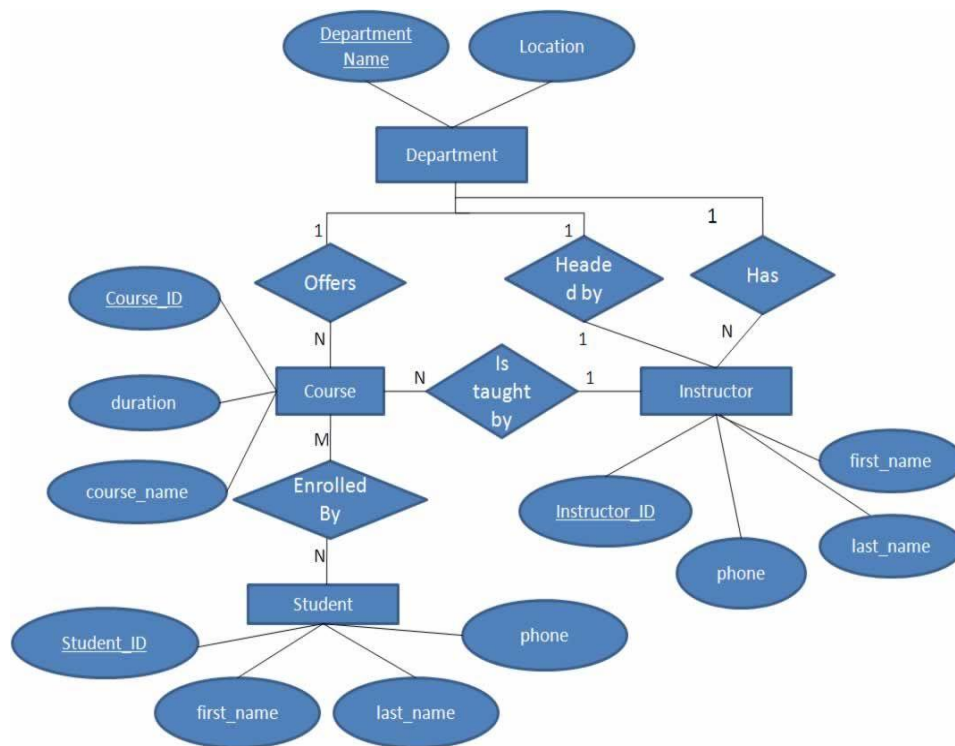


Figure 2: ER Diagram

### 3.3 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purposes of a use case diagram given below:

- It gathers the system's needs.
- It depicts the external view of the system.
- It recognizes the internal as well as external factors that influence the system.

- It represents the interaction between the actors

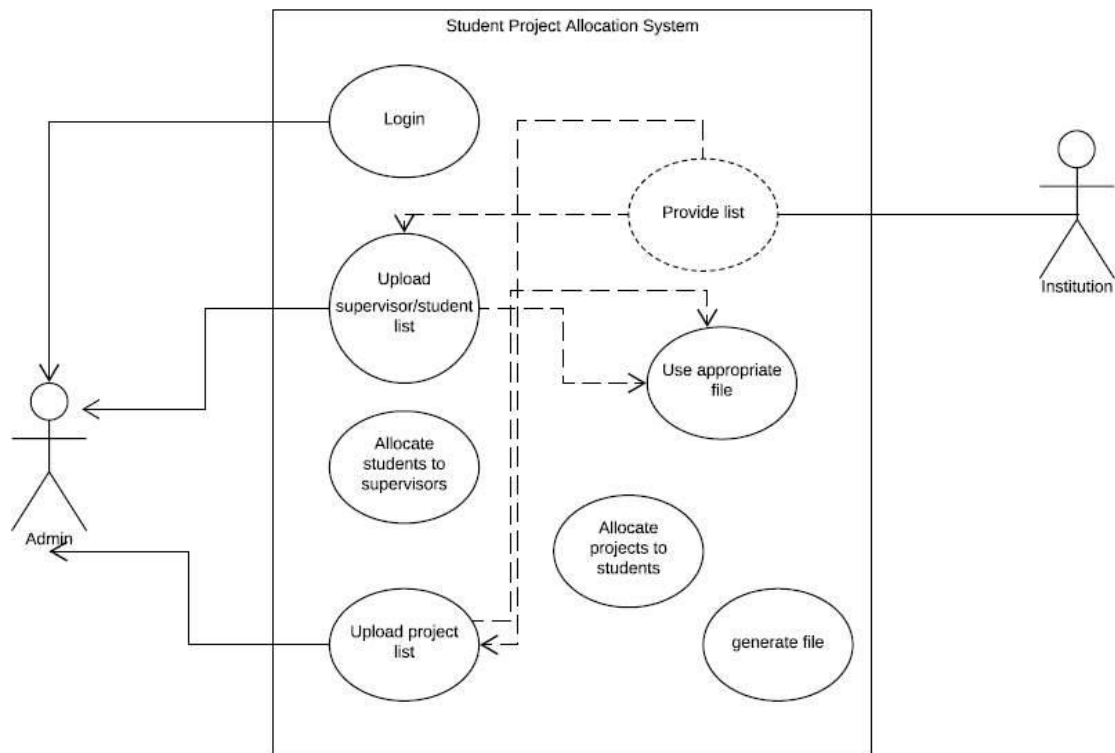


Figure 3: Use case Diagram

## 2.4 ACTIVITY DIAGRAM

An Activity Diagram can be used in the development of a chatting app to depict the flow of activities and behaviors within the system. Here's how an Activity Diagram can be applied.

- **User Registration:** The Activity Diagram can start with the "User Registration" activity, depicting the process of a user creating an account. It would include activities such as entering registration details, validating input, and creating a new user profile.
- **Login and Authentication:** The diagram can show the activities related to user login and authentication. This would involve activities like entering login credentials, validating them, and granting access to the app upon successful authentication.

- **Project Activities:** The core activities of the chatting app can be represented, including creating a new chat, joining an existing chat, and sending messages. These activities can be visualized as separate swimlanes or branches, showing the interaction between multiple users or participants within the chat.
- **Multimedia Sharing:** If the app supports multimedia sharing, the diagram can depict the activities involved in uploading and sharing photos, videos, or other media files. It would include activities like selecting a file, uploading it, and notifying other participants of the shared media.
- **Group Management:** If the app includes group chatting functionality, the Activity Diagram can illustrate activities related to group creation, management, and member invitations. It would involve activities like creating a new group, adding or removing members, and moderating group conversations.
- **Notification System:** The diagram can show activities related to the notification system, including activities like sending push notifications for new messages or updates, and displaying notifications to the users.
- **Logout and Session Management:** The diagram should include activities related to user logout and session management, ensuring that users can securely end their session and handle scenarios such as session timeouts.

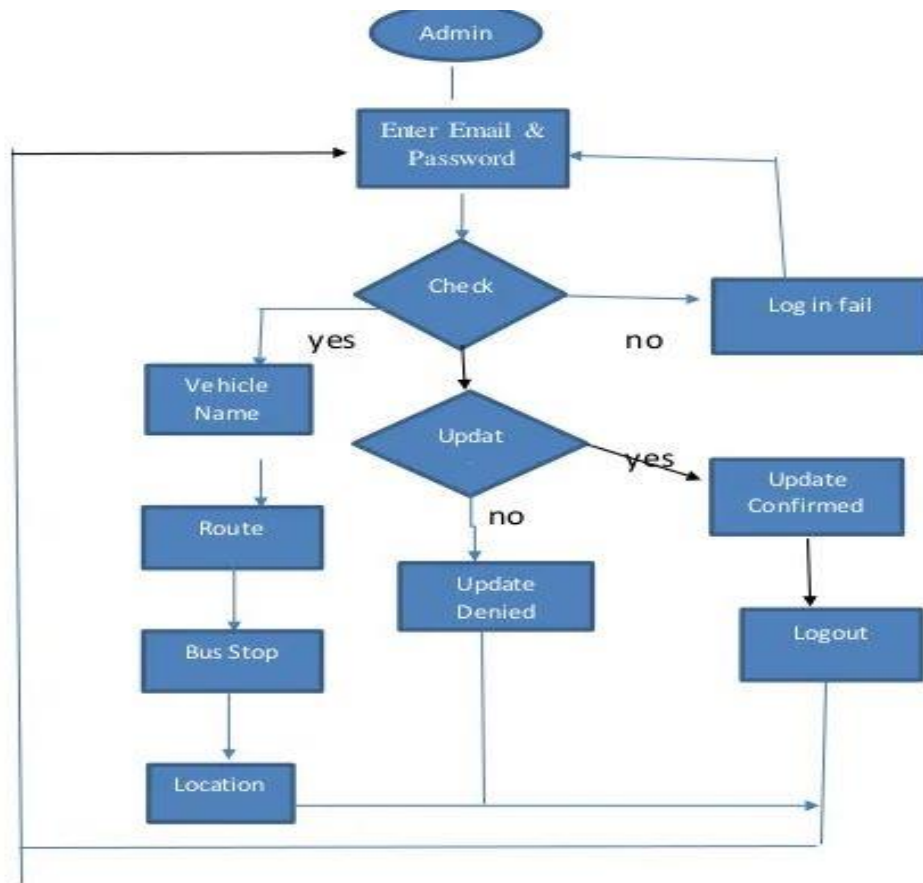


Figure 4: Activity Diagram

## 2.5 SEQUENCE DIAGRAM

A Sequence Diagram can be useful in the development of a chatting app to illustrate the chronological sequence of interactions between different entities or components within the system. Here's how a Sequence Diagram can be applied:

- **User Registration:** The Sequence Diagram can depict the sequence of interactions between the user interface and the backend components during the user registration process. It would show the steps involved, such as the user entering registration details, the system validating the input, and the creation of a new user profile.
- **Login and Authentication:** The diagram can illustrate the sequence of interactions between the user, the login interface, and the authentication system. It would show



how the user enters login credentials, the system verifies them, and grants access to the app upon successful authentication.

- **One-to-One Project:** The Sequence Diagram can represent the sequence of interactions between two users engaging in a one-to-one project. It would show the exchange of messages between the sender and receiver, indicating the order and timing of message delivery.
- **Group Project:** If the app supports group chatting, the diagram can depict the interactions between multiple users within a group chat. It would show how messages are sent, received, and distributed to all members of the group.
- **Multimedia Sharing:** The Sequence Diagram can illustrate the interactions involved in sharing multimedia content, such as photos or videos. It would show the steps of selecting a file, uploading it to the server, and notifying other users about the shared media.
- **Notification System:** The diagram can represent the interactions between the app and the notification system. It would show how the app sends notifications to users when new messages or updates are received, and how users interact with those notifications.
- **Error Handling:** The Sequence Diagram can depict the interactions related to error handling and exception scenarios. It would show how errors or exceptions are detected, reported, and handled within the app, including displaying error messages to users.
- **Logout and Session Management:** The diagram can illustrate the sequence of interactions involved in user logout and session management. It would show the steps taken to securely end the user session and perform any necessary cleanup or session-related tasks.

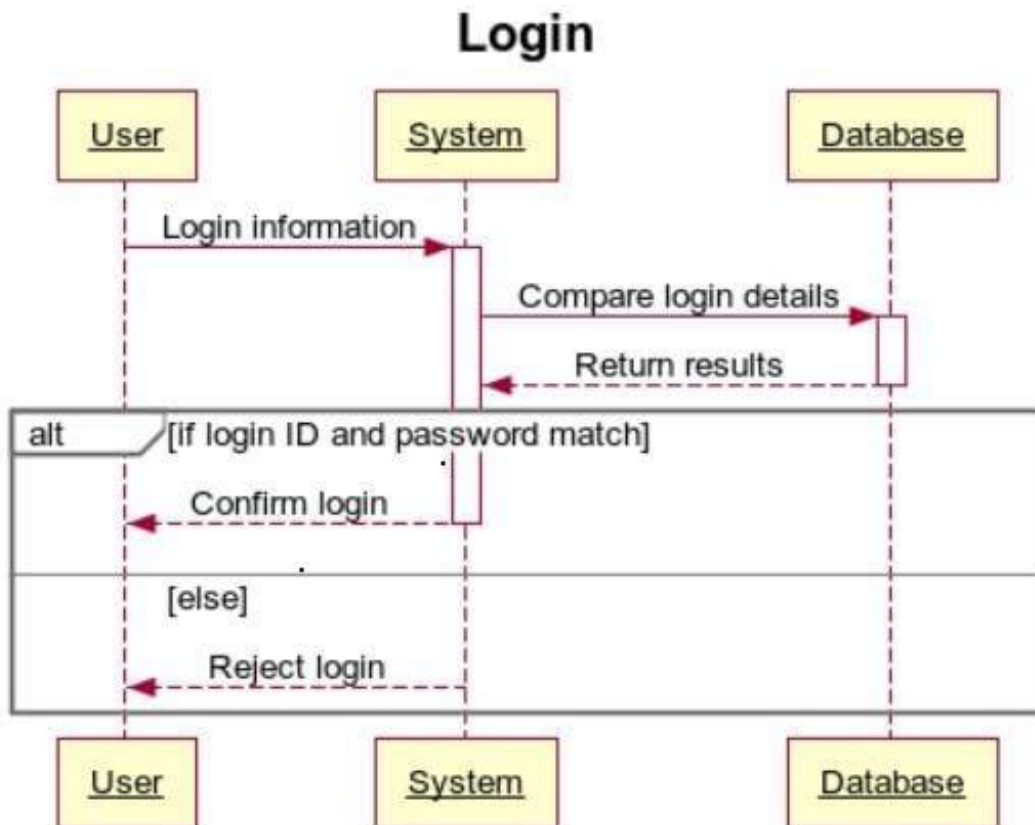


Figure 5: Sequential Diagram

## 2.6 COLLABORATION DIAGRAM

A Collaboration Diagram is useful in the development of a chatting app for several reasons:

- Visualizing Interactions:** A Collaboration Diagram provides a visual representation of how different objects or entities in the system collaborate and interact with each other. It helps developers understand the flow of information and the sequence of interactions between components, making it easier to identify potential issues or bottlenecks.
- Clarifying Object Relationships:** The diagram illustrates the relationships and dependencies between objects or entities in the app. It helps in understanding how objects communicate and cooperate to accomplish specific tasks. This clarity assists

in designing and implementing the app's architecture and ensures that the interactions between components are properly defined.

- **Identifying Communication Channels:** The Collaboration Diagram helps in identifying the communication channels between different objects or entities. It shows how messages or data are passed between components, clarifying the interfaces and API calls required for successful communication. This information is vital for developers when implementing the app's functionality.
- **Analyzing System Behavior:** By visualizing the interactions and collaborations between objects, the diagram enables developers to analyze the behavior of the system. It helps in identifying potential issues such as concurrency problems, race conditions, or communication conflicts. This analysis can lead to improvements in the app's performance and reliability.
- **Communication and Collaboration:** The Collaboration Diagram serves as a communication tool among developers, designers, and stakeholders. It provides a shared understanding of how different components work together, facilitating effective discussions and decision-making during the development process. It helps in aligning the development team's understanding of the app's architecture and functionality.
- **Test Case Generation:** The Collaboration Diagram can aid in generating test cases for the app. By examining the interactions between components, developers can identify the critical paths and scenarios that need to be tested. It helps in designing comprehensive test cases that cover all the necessary interactions and ensure the app functions as intended.

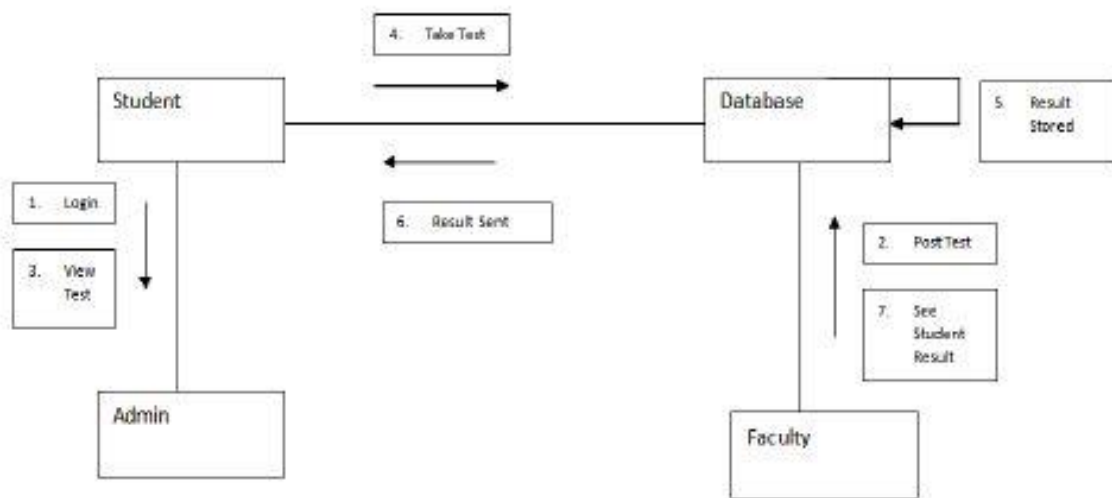


Figure 6: Collaboration Diagram

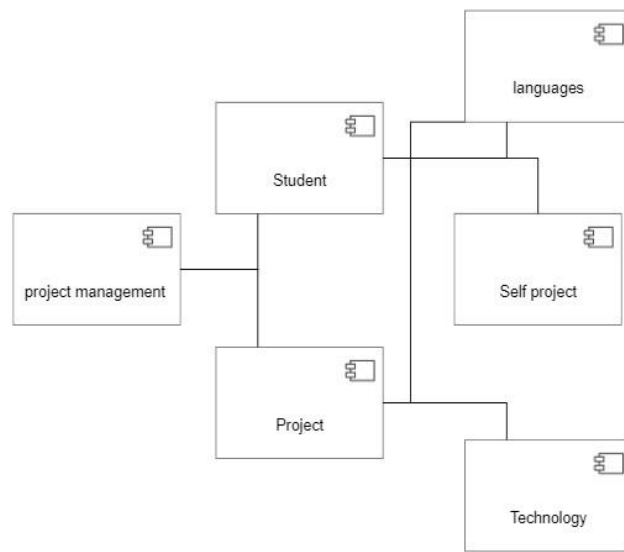
## 2.7 COMPONENT DIAGRAM

A Component Diagram is useful in the development of a chatting app to depict the high-level structure and organization of the system, highlighting the various components and their interdependencies. Here's how a Component Diagram can be applied:

- **User Interface Component:** The Component Diagram can include the User Interface component, representing the front-end or client-side of the chatting app. It encompasses the graphical user interface (GUI), screens, forms, and user interaction elements.
- **Application Server Component:** The diagram can depict the Application Server component, representing the server-side logic of the chatting app. This component handles business logic, authentication, message routing, and other core functionalities.
- **Database Component:** The Component Diagram can include the Database component, representing the storage and retrieval of data in the app. It illustrates the

database management system (DBMS) or data storage technology used, and the tables or collections storing user profiles, chats, messages, and other relevant data.

- **Service Component:** If the app includes real-time messaging capabilities, the diagram can show the Messaging Service component. This component manages the communication and exchange of messages between users, ensuring reliable and efficient message delivery.
- **Media Storage Component:** If multimedia sharing is supported, the Component Diagram can include the Media Storage component. It represents the storage infrastructure or cloud service used to store and retrieve media files, such as images or videos.
- **External Services or APIs:** The diagram can illustrate external services or APIs used by the app, such as authentication services (OAuth, OpenID), notification services (Push Notifications), or media processing services (image compression, video transcoding). These components depict the integration points with external systems.
- **Dependency Relationships:** The Component Diagram highlights the dependencies between components, indicating the required interfaces, dependencies, or communication channels between them. For example, the User Interface component may depend on the Application Server component for user authentication or message retrieval.
- **Deployment Environment:** The Component Diagram can also depict the deployment environment, illustrating the physical or virtual infrastructure where the components are deployed. This may include servers, cloud platforms, or other hosting environments.



**Figure 7: Component Diagram**

## CHAPTER 4

### FORM DESIGN

#### 4.1 SCREENSHOTS

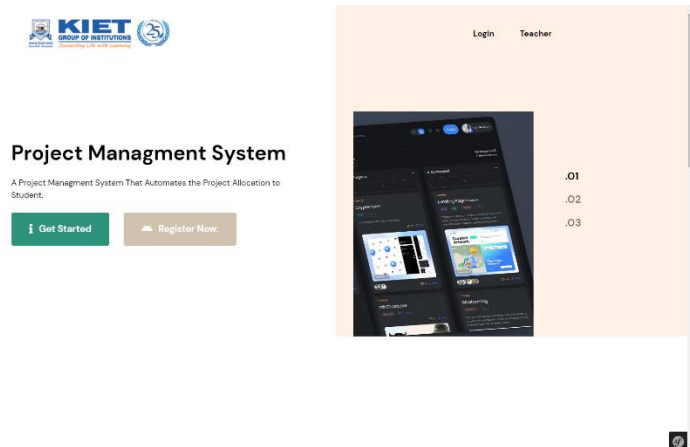


Figure 8: Home Page

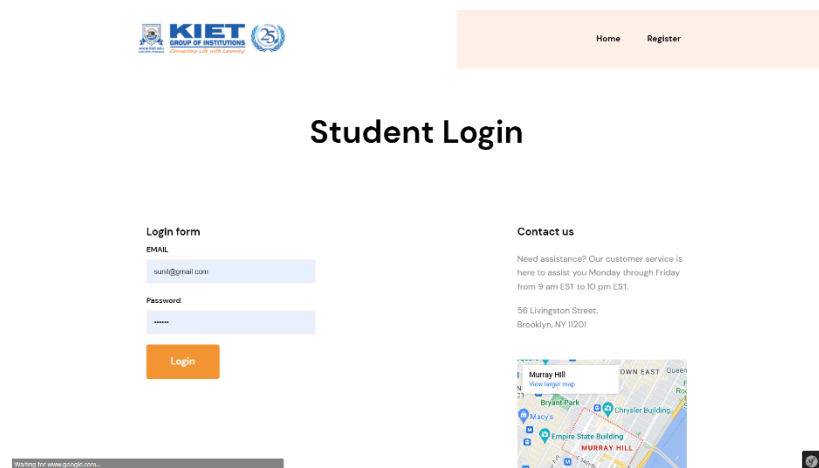



Figure 9: Student Login Page



[Home](#)
[Login](#)

## Student Registration

### Registration Form

Your Name

Your Phone

Your Email

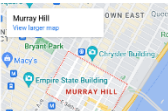
Your Password

☐ Agree terms

### Contact us


Need assistance? Our customer service is here to assist you Monday through Friday from 9 am EST to 10 pm EST.

56 Livingston Street,  
Brooklyn, NY 11201



<http://127.0.0.1:8080/login>

Figure 10: Student Registration Page



[Home](#)
[Register](#)

## Teacher Login

### Login form

EMAIL

Password

### Contact us

Need assistance? Our customer service is here to assist you Monday through Friday from 9 am EST to 10 pm EST.

56 Livingston Street,  
Brooklyn, NY 11201





Figure 11: Teacher Login Page



[Home](#)
[Login](#)

## Teacher Registration

### Registration Form

Your Name

Your Phone

Your Email

Your Password

☐ Agree terms

### Contact us

Need assistance? Our customer service is here to assist you Monday through Friday from 9 am EST to 10 pm EST.

56 Livingston Street,  
Brooklyn, NY 11201




Figure 12: Teacher Registration Page



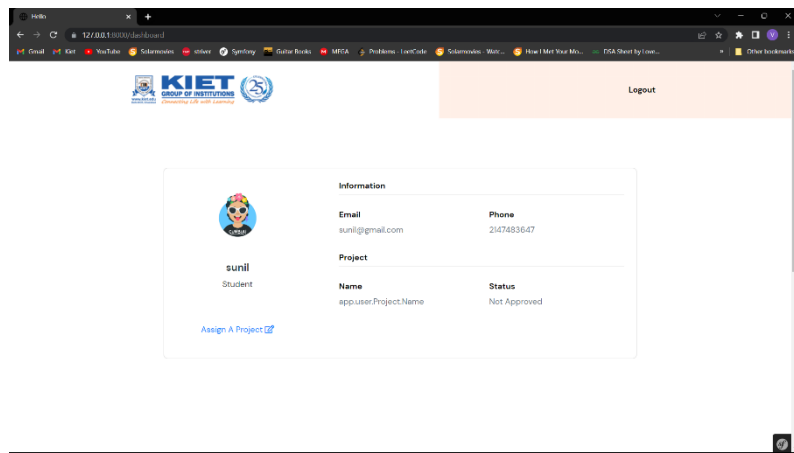


Figure 13: Teacher Profile Page

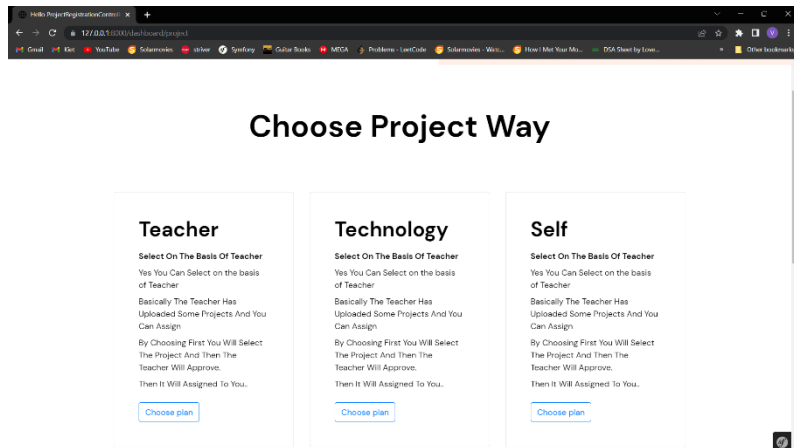


Figure 14: Project Category Page

## CHAPTER 5

### CODING

#### 5.1 Module Wise Code

##### 5.1.1 Bin/Console

```
#!/usr/bin/env php
```

```
<?php
```

```
use App\Kernel;
```

```
use Symfony\Bundle\FrameworkBundle\Console\Application;
```

```
if (!is_file(dirname(__DIR__).'/vendor/autoload_runtime.php')) {
```

```
    throw new LogicException('Symfony Runtime is missing. Try running "composer  
require symfony/runtime".');
```

```
}
```

```
require_once dirname(__DIR__).'/vendor/autoload_runtime.php';
```

```
return function (array $context) {
```

```
    $kernel = new Kernel($context['APP_ENV'], (bool) $context['APP_DEBUG']);
```

```
    return new Application($kernel);
```

```
};
```

### 5.1.2 Bin/PhpUnit

```
#!/usr/bin/env php
```

```
<?php
```

```
if (!ini_get('date.timezone')) {  
    ini_set('date.timezone', 'UTC');  
}
```

```
if (is_file(dirname(__DIR__).'/vendor/phpunit/phpunit/phpunit')) {  
    define('PHPUNIT_COMPOSER_INSTALL',  
dirname(__DIR__).'/vendor/autoload.php');  
    require PHPUNIT_COMPOSER_INSTALL;  
    PHPUnit\TextUI\Command::main();  
} else {  
    if (!is_file(dirname(__DIR__).'/vendor/symfony/phpunit-bridge/bin/simple-  
phpunit.php')) {  
        echo "Unable to find the `simple-phpunit.php` script in `vendor/symfony/phpunit-  
bridge/bin/`.\\n";  
        exit(1);  
    }  
  
    require dirname(__DIR__).'/vendor/symfony/phpunit-bridge/bin/simple-phpunit.php';  
}
```

### 5.1.3 Code for Login User Authenticator

```
<?php

namespace App\Security;

use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;
use Symfony\Component\Security\Core\Security;

use
Symfony\Component\Security\Http\Authenticator\AbstractLoginFormAuthenticator;
use
Symfony\Component\Security\Http\Authenticator\Passport\Badge\CsrfTokenBadge;
use Symfony\Component\Security\Http\Authenticator\Passport\Badge\UserBadge;
use
Symfony\Component\Security\Http\Authenticator\Passport\Credentials\PasswordCredentials;

use Symfony\Component\Security\Http\Authenticator\Passport\Passport;
use Symfony\Component\Security\Http\Util\TargetPathTrait;

class LoginUserAuthenticator extends AbstractLoginFormAuthenticator
{
    use TargetPathTrait;

    public const LOGIN_ROUTE = 'app_login';

    public function __construct(private UrlGeneratorInterface $urlGenerator)
    {
    }
}
```

```

public function authenticate(Request $request): Passport
{
    $email = $request->request->get('email', '');

    $request->getSession()->set(Security::LAST_USERNAME, $email);

    return new Passport(
        new UserBadge($email),
        new PasswordCredentials($request->request->get('password', '')),
        [
            new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
        ]
    );
}

```

```

public function onAuthenticationSuccess(Request $request, TokenInterface $token,
string $firewallName): ?Response
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
        return new RedirectResponse($targetPath);
    }
}

```

```

// For example:
return new RedirectResponse($this->urlGenerator->generate('app_dashboard'));
// throw new \Exception('TODO: provide a valid redirect inside '.__FILE__);
}

```

```

protected function getLoginUrl(Request $request): string
{
    return $this->urlGenerator->generate(self::LOGIN_ROUTE);
}

```

```
}  
}
```

#### 5.1.4 Code for Teacher Login Authenticator

```
<?php
```

```
namespace App\Security;
```

```
use Symfony\Component\HttpFoundation\RedirectResponse;
```

```
use Symfony\Component\HttpFoundation\Request;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
```

```
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;
```

```
use Symfony\Component\Security\Core\Security;
```

```
use
```

```
Symfony\Component\Security\Http\Authenticator\AbstractLoginFormAuthenticator;
```

```
use
```

```
Symfony\Component\Security\Http\Authenticator\Passport\Badge\CsrfTokenBadge;
```

```
use Symfony\Component\Security\Http\Authenticator\Passport\Badge\UserBadge;
```

```
use
```

```
Symfony\Component\Security\Http\Authenticator\Passport\Credentials\PasswordCredentials;
```

```
use Symfony\Component\Security\Http\Authenticator\Passport\Passport;
```

```
use Symfony\Component\Security\Http\Util\TargetPathTrait;
```

```
class TeacherLoginAuthenticator extends AbstractLoginFormAuthenticator
```

```
{
```

```
    use TargetPathTrait;
```

```
    public const LOGIN_ROUTE = 'app_teacher_login';
```

```

public function __construct(private UrlGeneratorInterface $urlGenerator)
{
}

```

```

public function authenticate(Request $request): Passport
{
    $email = $request->request->get('email', '');

    $request->getSession()->set(Security::LAST_USERNAME, $email);

    return new Passport(
        new UserBadge($email),
        new PasswordCredentials($request->request->get('password', '')),
        [
            new CsrfTokenBadge('authenticate', $request->request->get('_csrf_token')),
        ]
    );
}

```

```

public function onAuthenticationSuccess(Request $request, TokenInterface $token,
string $firewallName): ?Response
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
        return new RedirectResponse($targetPath);
    }
}

```

```

// For example:
return new RedirectResponse($this->urlGenerator->generate('app_teacher'));
// throw new \Exception('TODO: provide a valid redirect inside '.__FILE__);
}

```

```

protected function getLoginUrl(Request $request): string
{
    return $this->urlGenerator->generate(self::LOGIN_ROUTE);
}
}

```

### 5.1.5 Code for Dashboard Controller

```

<?php

namespace App\Controller;

use App\Entity\Project;
use App\Entity\Teacher;
use App\Form\ProjectRegistrationType;
use App\Form\ProjectSelectType;
use App\Form\ProjectType;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DashboardController extends AbstractController
{
    #[Route('/dashboard', name: 'app_dashboard')]
    public function dashboard(): Response
    {
        return $this->render('dashboard/dashboard.html.twig', [
            'controller_name' => 'DashboardController',
        ]);
    }
}

```



```

#[Route('/dashboard/project', name: 'app_project')]
public function project(Request $request): Response
{
    $project = new Project();
    $form = $this->createForm(ProjectRegistrationType::class, $project);
    $form->handleRequest($request);
    return $this->render('dashboard/project.html.twig', [
        'form' => $form->createView()
    ]);
}

#[Route('dashboard/project/select/{slug}', name: 'app_project_select')]
public function teacherSelect(Request $request, EntityManagerInterface
$entityManager, $slug): Response
{
    if ($slug == "teacher") {
        $teacher = $entityManager->getRepository(Teacher::class)-
>findAll();
        $item = $teacher;
        if ($request->request->all()) {
            $id = $request->request->all()['id'];
            return $this->redirectToRoute('app_project_select_choose', ['slug'
=> 'teacher', 'id' => $id]);
        }
    } elseif ($slug == "technology") {
        $projects = $entityManager->getRepository(Project::class)-
>findAll();
        $technologies = [];
        foreach ($projects as $project) {
            $technologies[] = $project->getTechnology();
        }
        $technologies = array_unique($technologies);
    }
}

```

```

$item = $technologies;
if ($request->request->all()) {
    $val = $request->request->all()['technology'];
    return $this->redirectToRoute('app_project_select_choose_by_tech', ['slug' => $val]);
}
} elseif ($slug == "self") {

    $project = new Project();
    $form = $this->createForm(ProjectType::class, $project);
    $form->handleRequest($request);
    if ($form->isSubmitted()) {
        if ($this->getUser()->getProject()) {
            $this->addFlash("danger", "YOu have already assigned a
project");
            return $this->redirectToRoute('app_project_select', ['slug' =>
$slug]);
        }
        if ($project->getUser()) {
            $this->addFlash("danger", "this project is already Registered
With Some User");
            return $this->redirectToRoute('app_project_select', ['slug' =>
$slug]);
        } else {

            $project->setUser($this->getUser());
            $project->setApproved(false);
            $entityManager->persist($project);
            $entityManager->flush();

            $this->addFlash("success", "Your Project is sent TO be
Approved");
            return $this->redirectToRoute("app_dashboard");
        }
    }
}

```

```

        }
    }
    $item = $form->createView();
}

return $this->render('dashboard/project_select.html.twig', [
    'slug' => $slug,
    'item' => $item
]);
}

#[Route('dashboard/project/select/technology/{slug}', name:
'app_project_select_choose_by_tech')]
public function projectSelectByTech(Request $request,
EntityManagerInterface $entityManager, $slug): Response
{
    $projects = $entityManager->getRepository(Project::class)-
>findBy(['Technology' => $slug]);
    if ($request->query->get('id') !== null) {
        $id = $request->query->get('id');
        $project = $entityManager->getRepository(Project::class)-
>findOneBy(['id' => $id]);
        if ($this->getUser()->getProject()) {
            $this->addFlash("danger", "YOu have already assigned a
project");
            return $this-
>redirectToRoute('app_project_select_choose_by_tech', ['slug' => $slug]);
        }
        if ($project->getUser()) {
            $this->addFlash("danger", "this project is already Registered
With Some User");
            return $this-
>redirectToRoute('app_project_select_choose_by_tech', ['slug' => $slug]);
        } else {

```

```

        $project->setUser($this->getUser());
        $project->setApproved(false);
        $entityManager->persist($project);
        $entityManager->flush();

        $this->addFlash("success", "Your Project is sent TO be
Approved");
        return $this->redirectToRoute("app_dashboard");
    }
}

return $this->render('dashboard/project_final.html.twig', [
    'projects' => $projects
]);
}

#[Route('dashboard/project/select/{slug}/{id}', name:
'app_project_select_choose')]
public function projectSelect(Request $request, Teacher $teacher,
EntityManagerInterface $entityManager): Response
{

    $project = $entityManager->getRepository(Project::class)-
>findBy(['Teacher' => $teacher]);
    // dd($request->query->get('id') !== null);
    if ($request->query->get('id') !== null) {
        $id = $request->query->get('id');
        $project = $entityManager->getRepository(Project::class)-
>findOneBy(['id' => $id]);
        if ($this->getUser()->getProject()) {
            $this->addFlash("danger", "YOu have already assigned a
project");

```

```

        return $this->redirectToRoute('app_project_select_choose', ['slug'
=> 'teacher', 'id' => $teacher->getId()]);
    }
    if ($project->getUser()) {
        $this->addFlash("danger", "this project is already Registered
With Some User");
        return $this->redirectToRoute('app_project_select_choose', ['slug'
=> 'teacher', 'id' => $teacher->getId()]);
    } else {

        $project->setUser($this->getUser());
        $project->setApproved(false);
        $entityManager->persist($project);
        $entityManager->flush();

        $this->addFlash("success", "Your Project is sent TO be
Approved");
        return $this->redirectToRoute("app_dashboard");
    }
}
return $this->render('dashboard/project_final.html.twig', [
    'projects' => $project
]);
}
}

```

### 5.1.6 Code for Home Controller

```
<?php
```

```
namespace App\Controller;
```

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

```

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/', name: 'app_home')]
    public function index(): Response
    {
        return $this->render('home/index.html.twig', [
            'controller_name' => 'HomeController',
        ]);
    }
}

```

### 5.1.7 Code for Login Controller

<?php

```

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use
Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

class LoginController extends AbstractController
{
    #[Route(path: '/login', name: 'app_login')]
    public function login(AuthenticationUtils $authenticationUtils):
Response
    {

```

```

        if ($this->getUser()) {
            return $this->redirectToRoute('app_dashboard');
        }

        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();
        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/login.html.twig', ['last_username' =>
        $lastUsername, 'error' => $error]);
    }

    #[Route(path: '/logout', name: 'app_logout')]
    public function logout(): void
    {
        throw new \LogicException('This method can be blank - it will be
        intercepted by the logout key on your firewall.');
```

### 5.1.8 Code for Registration Controller

```

<?ph
p
```

```

namespace App\Controller;

use App\Entity\User;
use App\Form\RegistrationFormType;
```

```

use App\Security\LoginUserAuthenticator;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use
Symfony\Component>PasswordHasher\Hasher\UserPasswordHasherInterface;
use Symfony\Component\Routing\Annotation\Route;
use
Symfony\Component\Security\Http\Authentication\UserAuthenticatorInterface;
use Symfony\Contracts\Translation\TranslatorInterface;

class RegistrationController extends AbstractController
{
    #[Route('/register', name: 'app_register')]
    public function register(Request $request, UserPasswordHasherInterface
$userPasswordHasher, UserAuthenticatorInterface $userAuthenticator,
LoginUserAuthenticator $authenticator, EntityManagerInterface
$entityManager): Response
    {
        $user = new User();
        $form = $this->createForm(RegistrationFormType::class, $user);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            // encode the plain password
            $user->setPassword(
                $userPasswordHasher->hashPassword(
                    $user,
                    $form->get('plainPassword')->getData()
                )
            );
        }
    }
}

```



```

    );

    $entityManager->persist($user);
    $entityManager->flush();
    // do anything else you need here, like send an email

    return $this->redirectToRoute('app_login');

}

return $this->render('registration/register.html.twig', [
    'registrationForm' => $form->createView(),
]);
}
}

```

### 5.1.9 Code for Teacher Dashboard Controller

```

<?php

namespace App\Controller;

use App\Entity\Project;
use App\Form\ProjectType;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class TeacherDashboardController extends AbstractController
{
    #[Route('/teacher/', name: 'app_teacher')]
    public function index(EntityManagerInterface $entityManager):
    Response
    {
        $project = $entityManager->getRepository(Project::class)-
        >findBy(['Teacher' => $this->getUser()]);
    }
}

```

```

        return $this->render('teacher_dashboard/index.html.twig', [
            'projects' => $project
        ]);
    }
    #[Route('/teacher/project/add', name: 'app_teacher_project_add')]
    public function add(Request $request, EntityManagerInterface
$entityManager): Response
    {
        $project = new Project();
        $form = $this->createForm(ProjectType::class, $project);
        $form->handleRequest($request);
        if ($form->isSubmitted()) {
            $project->setTeacher($this->getUser());
            $project->setApproved(false);
            $entityManager->persist($project);
            $entityManager->flush();
            $this->addFlash("success", "Project Is Created");
            return $this->redirectToRoute('app_teacher');
        }
        return $this->render('teacher_dashboard/add.html.twig', [
            'form' => $form->createView()
        ]);
    }
}

#[Route('/teacher/project/approve', name:
'app_teacher_project_approve')]
public function approvalList(EntityManagerInterface $entityManager,
Request $request): Response
{
    if (isset($_REQUEST['working'])) {
        $action = $request->get('working');
        $ids = $request->get('id');
        if ($action == 'submit') {
            foreach ($ids as $id) {
                $project = $entityManager->getRepository(Project::class)-
>find($id);
                $project->setApproved(true);
                $entityManager->persist($project);
                $entityManager->flush();
            }
        } else if ($action == 'decline') {
            foreach ($ids as $id) {
                $project = $entityManager->getRepository(Project::class)-
>find($id);
                $project->setUser(null);
            }
        }
    }
}

```

```

        $entityManager->persist($project);
        $entityManager->flush();
    }
}
}
// dd($request->request->all());    }
$projects = $entityManager->getRepository(Project::class)-
>findBy(['Teacher' => $this->getUser(), 'Approved' => false]);
$projectWithoutUser = [];
foreach ($projects as $tempproject) {
    if($tempproject->getUser()){
        array_push($projectWithoutUser,$tempproject);
    }
}
// dd($projectWithoutUser);
return $this->render('teacher_dashboard/approvallist.html.twig', [
    'projects' => $projectWithoutUser
]);
}

#[Route('/teacher/project/all', name: 'app_teacher_project_all')]
public function myProjects(EntityManagerInterface $entityManager):
Response
{
    $project = $entityManager->getRepository(Project::class)-
>findBy(['Teacher' => $this->getUser()]);
    return $this->render('teacher_dashboard/myprojects.html.twig', [
        'project' => $project
    ]);
}
}

```

### 5.2.1 Code for Teacher Login Controller

<?php

```

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use
Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

```

```

class TeacherLoginController extends AbstractController
{
    #[Route(path: 'teacher/login', name: 'app_teacher_login')]
    public function login(AuthenticationUtils $authenticationUtils):
Response
    {
        if ($this->getUser()) {
            return $this->redirectToRoute('app_teacher');
        }

        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();
        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/teacherlogin.html.twig', ['last_username'
=> $lastUsername, 'error' => $error]);
    }

    #[Route(path: '/teacher/logout', name: 'app_teacher_logout')]
    public function logout(): void
    {
        throw new \LogicException("This method can be blank - it will be
intercepted by the logout key on your firewall.");
    }
}

```

### 5.2.2 Code for Teacher Registration Controller

```

<?ph
p

```

```

namespace App\Controller;

use App\Entity\Teacher;
use App\Form\RegistrationFormType;
use App\Form\TeacherRegistrationFormType;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

```

```

use
Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Contracts\Translation\TranslatorInterface;

class TeacherRegistrationController extends AbstractController
{
    #[Route('/teacher/register', name: 'app_teacher_register')]
    public function register(Request $request, UserPasswordHasherInterface
    $userPasswordHasher, EntityManagerInterface $entityManager): Response
    {
        $user = new Teacher();
        $form = $this->createForm(TeacherRegistrationFormType::class,
    $user);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            // encode the plain password
            $user->setPassword(
                $userPasswordHasher->hashPassword(
                    $user,
                    $form->get('plainPassword')->getData()
                )
            );

            $entityManager->persist($user);
            $entityManager->flush();
            // do anything else you need here, like send an email

            return $this->redirectToRoute('app_teacher_login');
        }

        return $this->render('registration/teacherregister.html.twig', [
            'registrationForm' => $form->createView(),
        ]);
    }
}

```

### 5.2.3 Project Database

```
<?php
```

```
namespace App\Entity;
```

```

use App\Repository\ProjectRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProjectRepository::class)]
class Project
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $Name = null;

    #[ORM\Column(length: 255)]
    private ?string $Description = null;

    #[ORM\Column(length: 255)]
    private ?string $Technology = null;

    #[ORM\ManyToOne(inversedBy: 'projects')]
    private ?Teacher $Teacher = null;

    #[ORM\OneToOne(inversedBy: 'project', cascade: ['persist', 'remove'])]
    private ?User $User = null;

    #[ORM\Column]
    private ?bool $Approved = null;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getName(): ?string
    {
        return $this->Name;
    }

    public function setName(string $Name): self
    {
        $this->Name = $Name;

        return $this;
    }
}

```

```

public function getDescription(): ?string
{
    return $this->Description;
}

public function setDescription(string $Description): self
{
    $this->Description = $Description;

    return $this;
}

public function getTechnology(): ?string
{
    return $this->Technology;
}

public function setTechnology(string $Technology): self
{
    $this->Technology = $Technology;

    return $this;
}

public function getTeacher(): ?Teacher
{
    return $this->Teacher;
}

public function setTeacher(?Teacher $Teacher): self
{
    $this->Teacher = $Teacher;

    return $this;
}

public function getUser(): ?User
{
    return $this->User;
}

public function setUser(?User $User): self
{
    $this->User = $User;
}

```

```

        return $this;
    }

    public function isApproved(): ?bool
    {
        return $this->Approved;
    }

    public function setApproved(bool $Approved): self
    {
        $this->Approved = $Approved;

        return $this;
    }
}

```

### 5.2.4 Teacher Database

```

<?ph
p

```

```

namespace App\Entity;

use App\Repository\TeacherRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use
Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterf
ace;
use Symfony\Component\Security\Core\User\UserInterface;

#[ORM\Entity(repositoryClass: TeacherRepository::class)]
#[UniqueEntity(fields: ['email'], message: 'There is already an account with
this email')]
class Teacher implements UserInterface,
PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 180, unique: true)]

```



```

private ?string $email = null;

#[ORM\Column]
private array $roles = [];

/**
 * @var string The hashed password
 */
#[ORM\Column]
private ?string $password = null;

#[ORM\OneToMany(mappedBy: 'Teacher', targetEntity: Project::class)]
private Collection $projects;

#[ORM\Column(length: 255)]
private ?string $Name = null;

#[ORM\Column]
private ?int $Phone = null;

public function __construct()
{
    $this->projects = new ArrayCollection();
}

public function getId(): ?int
{
    return $this->id;
}

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): self
{
    $this->email = $email;

    return $this;
}

/**
 * A visual identifier that represents this user.
 *
 * @see UserInterface

```

```

*/
public function getUserIdentifier(): string
{
    return (string) $this->email;
}

/**
 * @see UserInterface
 */
public function getRoles(): array
{
    $roles = $this->roles;
    // guarantee every user at least has ROLE_USER
    $roles[] = 'ROLE_USER';

    return array_unique($roles);
}

public function setRoles(array $roles): self
{
    $this->roles = $roles;

    return $this;
}

/**
 * @see PasswordAuthenticatedUserInterface
 */
public function getPassword(): string
{
    return $this->password;
}

public function setPassword(string $password): self
{
    $this->password = $password;

    return $this;
}

/**
 * @see UserInterface
 */
public function eraseCredentials()
{
    // If you store any temporary, sensitive data on the user, clear it here

```

```

        // $this->plainPassword = null;
    }

    /**
     * @return Collection<int, Project>
     */
    public function getProjects(): Collection
    {
        return $this->projects;
    }

    public function addProject(Project $project): self
    {
        if (!$this->projects->contains($project)) {
            $this->projects->add($project);
            $project->setTeacher($this);
        }

        return $this;
    }

    public function removeProject(Project $project): self
    {
        if ($this->projects->removeElement($project)) {
            // set the owning side to null (unless already changed)
            if ($project->getTeacher() === $this) {
                $project->setTeacher(null);
            }
        }

        return $this;
    }

    public function getName(): ?string
    {
        return $this->Name;
    }

    public function setName(string $Name): self
    {
        $this->Name = $Name;

        return $this;
    }

    public function getPhone(): ?int

```

```

    {
        return $this->Phone;
    }

    public function setPhone(int $Phone): self
    {
        $this->Phone = $Phone;

        return $this;
    }
}

```

### 5.2.5 User Database

<?php

```

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use
Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;

#[ORM\Entity(repositoryClass: UserRepository::class)]
#[UniqueEntity(fields: ['email'], message: 'There is already an account with
this email')]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 180, unique: true)]
    private ?string $email = null;

    #[ORM\Column]
    private array $roles = [];

    /**

```

```

* @var string The hashed password
*/
#[ORM\Column]
private ?string $password = null;

#[ORM\Column(length: 255)]
private ?string $Name = null;

#[ORM\Column]
private ?int $Phone = null;

#[ORM\OneToOne(mappedBy: 'User', cascade: ['persist', 'remove'])]
private ?Project $project = null;

public function getId(): ?int
{
    return $this->id;
}

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): self
{
    $this->email = $email;

    return $this;
}

/**
 * A visual identifier that represents this user.
 *
 * @see UserInterface
 */
public function getUserIdentifier(): string
{
    return (string) $this->email;
}

/**
 * @see UserInterface
 */
public function getRoles(): array
{

```

```

        $roles = $this->roles;
        // guarantee every user at least has ROLE_USER
        $roles[] = 'ROLE_USER';

        return array_unique($roles);
    }

    public function setRoles(array $roles): self
    {
        $this->roles = $roles;

        return $this;
    }

    /**
     * @see PasswordAuthenticatedUserInterface
     */
    public function getPassword(): string
    {
        return $this->password;
    }

    public function setPassword(string $password): self
    {
        $this->password = $password;

        return $this;
    }

    /**
     * @see UserInterface
     */
    public function eraseCredentials()
    {
        // If you store any temporary, sensitive data on the user, clear it here
        // $this->plainPassword = null;
    }

    public function getName(): ?string
    {
        return $this->Name;
    }

    public function setName(string $Name): self
    {
        $this->Name = $Name;
    }

```

```

        return $this;
    }

    public function getPhone(): ?int
    {
        return $this->Phone;
    }

    public function setPhone(int $Phone): self
    {
        $this->Phone = $Phone;

        return $this;
    }

    public function getProject(): ?Project
    {
        return $this->project;
    }

    public function setProject(?Project $project): self
    {
        // unset the owning side of the relation if necessary
        if ($project === null && $this->project !== null) {
            $this->project->setUser(null);
        }

        // set the owning side of the relation if necessary
        if ($project !== null && $project->getUser() !== $this) {
            $project->setUser($this);
        }

        $this->project = $project;

        return $this;
    }
}

```

## **CHAPTER 6**

### **TESTING**

#### **6.1 Functional Testing**

Functional testing of a Project Allotment System (PAS) involves evaluating its individual functions and features to ensure they are working correctly and meeting the specified requirements. Here are some key aspects to consider when conducting functional testing for a project allotment system:

##### **1. Requirement Validation:**

- **Test Case 1:** Verify that the PAS accurately captures and implements the specified requirements.
- **Test Case 2:** Ensure that the system fulfills the intended purpose of allocating projects and resources effectively.

##### **2. User Interface (UI) Testing:**

- **Test Case 1:** Validate that the user interface is intuitive, user-friendly, and responsive.
- **Test Case 2:** Test the navigation, input fields, buttons, and other UI components for proper functionality and usability.

##### **3. Project Creation and Management:**

- **Test Case 1:** Test the creation of projects with various parameters such as name, start/end dates, priority, and assigned resources.
- **Test Case 2:** Validate that projects can be updated, edited, and deleted as necessary.
- **Test Case 3:** Verify that projects are correctly displayed and sorted based on their attributes.



#### **4. Resource Allocation:**

- **Test Case 1:** Test the allocation of resources to projects based on availability, skills, and other specified criteria.
- **Test Case 2:** Validate that resources can be assigned, updated, and removed from projects.
- **Test Case 3:** Ensure that the system prevents over-allocation and conflicts in resource assignments.

#### **5. Reporting and Analytics:**

- **Test Case 1:** Verify the generation of accurate reports and analytics related to project allocation, resource utilization, and other relevant metrics.
- **Test Case 2:** Validate that the reports provide the required information and are presented in a clear and understandable format.

### **6.2 NON-FUNCTIONAL TESTING**

Non-functional testing of a Project Allotment System (PAS) focuses on evaluating the system's behavior and performance characteristics rather than its specific functionalities. It assesses the system's qualities such as usability, performance, reliability, security, and scalability. Here are some key aspects to consider when conducting non-functional testing for a project allotment system:

#### **1. Usability Testing:**

- **Test Case 1:** Evaluate the system's ease of use, intuitiveness, and user-friendliness.
- **Test Case 2:** Test the system's compliance with user interface design standards and best practices.
- **Test Case 3:** Conduct user feedback sessions and surveys to gather subjective feedback on the system's usability.

## **2. Performance Testing:**

- **Test Case 1:** Measure and evaluate the system's response time under various workloads and user concurrency levels.
- **Test Case 2:** Test the system's ability to handle many concurrent users and projects without significant performance degradation.
- **Test Case 3:** Assess the scalability of the system by gradually increasing the workload and monitoring its performance.

## **3. Security Testing:**

- **Test Case 1:** Identify and evaluate potential security vulnerabilities in the system, such as authentication and authorization weaknesses.
- **Test Case 2:** Test the system's resilience against common security threats, such as SQL injections, cross-site scripting, and session hijacking.
- **Test Case 3:** Verify that user authentication and authorization mechanisms prevent unauthorized access to user information.

## **4. Compatibility Testing:**

- **Test Case 1:** Test the system's compatibility with different web browsers, operating systems, and mobile devices.
- **Test Case 2:** Verify that the PAS functions correctly across various platforms and configurations, ensuring a consistent user experience.

## **5. Reliability Testing:**

- **Test Case 1:** Assess the system's ability to perform consistently and reliably over an extended period.
- **Test Case 2:** Test the system's behavior under stress, including high load, resource limitations, and intermittent network connectivity.
- **Test Case 3:** Verify that the system recovers gracefully from failures or disruptions, without data loss or corruption.

## **6. Compliance Testing:**

- **Test Case 1:** Assess the system's compliance with relevant regulations, industry standards, and organizational policies.
- **Test Case 2:** Test whether the PAS adheres to data privacy regulations, accessibility guidelines, and other compliance requirements.
- **Test Case 3:** Test the system's ability to recover from a catastrophic failure or system crash.

## REFERENCES

1. Bharti, P., Yadav, S., & Rathi, S. (2017). Resource Allocation in Software Project Management: A Comprehensive Review. *International Journal of Computer Applications*, 171(4), 13-18.
2. Chokshi, N., & Chokshi, A. (2018). Project Resource Allocation using Genetic Algorithm. *International Journal of Computer Applications*, 179(3), 29-35.
3. Fernandes, A. J., Rodrigues, R. M., & Neves, J. (2016). Project Scheduling and Resource Allocation: A Survey of Current Models and Techniques. *European Journal of Operational Research*, 248(3), 749-774.
4. Jiang, J., Sun, C., Zeng, H., & Fan, J. (2019). An Overview of Resource Allocation and Scheduling in Project Management. *Future Generation Computer Systems*, 92, 899-910.
5. Maheshwari, P., Suman, A., Kumar, A., & Poonia, R. C. (2016). Resource Allocation Techniques in Cloud Computing: A Comprehensive Review. *International Journal of Computer Applications*, 148(4), 20-25.

## BIBLIOGRAPHY

- [1] Database System Concepts, by Silberschatz, Korth and Sudarshan, 6th Edition, Publisher. McGraw-Hill Education; (May 12, 2016)
- [2] Software Engineering, by Sommerville, 8th Edition, Publisher: Pearson; (May 15, 2016)
- [3] Modern Systems Analysis & Design, by Jeffrey A. Hoffer, Joey George, Joe A. Valacich 6th Edition, Publisher: Pearson; (May 26, 2016)
- Project idea, SPM system, [www.sourcecodesworld.com](http://www.sourcecodesworld.com), Retrieved on 11 June 2016
- [4] <http://www.sourcecodesworld.com/project-bank/project18.asp>
- [5] Database modeling, Wikipedia, [www.en.wikipedia.org](http://en.wikipedia.org), Retrieved on 20 June 2016 [http://en.wikipedia.org/wiki/Database\\_model](http://en.wikipedia.org/wiki/Database_model)
- [6] Software Requirements, wikibooks, <http://en.wikibooks.org>, Retrieved on 14 July 2016, [http://en.wikibooks.org/wiki/Introduction\\_to\\_Software\\_Engineering/Planning/Requirements](http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Planning/Requirements)