Software Requirements Specification

For

PROBee

EDA KOCAMAN, DİLA TÜMÜR, YUSUF VURAL

ÇANKAYA UNIVERSITY | SENG272

Table of Contents

1. Introduction

1.1. Purpose

The purpose of PROBee is to streamline and enhance the management of graduation projects within the Çankaya University Software Engineering Department. By providing a centralized platform, PROBee aims to facilitate seamless communication, collaboration, and coordination among students, advisors, and administrators throughout the project lifecycle. With features tailored to project proposal submission, team formation, documentation upload, project monitoring, and evaluation, PROBee empowers stakeholders to efficiently navigate the complexities of project development, ultimately fostering innovation, academic excellence, and successful project outcomes.

1.2. Scope

The scope of PROBee encompasses the comprehensive management of graduation projects within the Çankaya University Software Engineering Department. This includes facilitating the submission and approval of project proposals, forming project teams, tracking project progress through regular monitoring, and facilitating the documentation and reporting processes. Additionally, PROBee provides a platform for collaboration among students, advisors, and administrators, enabling seamless communication and knowledge exchange. The scope extends to integrating with external tools such as GitHub for version control and providing features for task management and collaboration. Ultimately, PROBee aims to streamline the project development process, enhance transparency and accountability, and foster a culture of innovation and academic excellence within the department.

1.3. Definitions, Acronyms and Abbreviations

PROBee: Graduation Project Development Application used by the Çankaya University Software Engineering Department.
MVP: Minimum Viable Product - The basic version of the PROBee application with essential features.
SENG: Software Engineering Department at Çankaya University.
SRS: Software Requirement Specification
API: Application Programming Interface

1.4. References

- This document adheres to the standards outlined in the IEEE SRS template.
- Çankaya University Software Engineering Department. (2023-2024). SENG 491-492 Graduation Project I-II Information Meeting [Course Material].
- Graduation Project Development Application (PROBee Documentation).

1.5. Overview

This document serves as the Software Requirement Specification (SRS) and Software Design Document (SDD) for the PROBee web application, developed for the Çankaya University Software Engineering Department. It outlines the functional and non-functional requirements of the application, as well as its design and architecture. The document is structured to provide a comprehensive understanding of the PROBee application, including its purpose, scope, features, architecture, and implementation details. By following this document, stakeholders can gain insight into the development process and ensure alignment with project objectives and requirements

2. Overall Description

2.1. Product Perspective

The PROBee web application is designed to operate within the context of the Çankaya University Software Engineering Department's graduation project process. It serves as a pivotal tool for managing and facilitating the development of graduation projects from inception to completion. PROBee operates as a standalone system, providing a centralized platform for students, advisors, and administrators to collaborate, track progress, and ensure the successful execution of graduation projects. While PROBee is designed to integrate seamlessly with existing academic and industry practices, it operates independently, offering a tailored solution to the specific needs and requirements of the Çankaya University Software Engineering Department. As such, PROBee functions as an integral component of the graduation project ecosystem, enhancing efficiency, transparency, and collaboration throughout the project lifecycle.

2.2. Product Functions

User Authentication and Authorization:

- Enable users to register and log in with username and password.
- Define user types with respective permissions.

Project Proposal Management:

- Allow students to submit project proposals.
- Provide admin with the capability to approve or reject proposals.
- Record reasons for rejection and track approval status.

Project Calls:

- Include approved projects in project calls.
- Enable students and advisors to apply for projects.
- Facilitate the submission of project acceptance forms by students.

Team Setup:

- Allow students to form project groups and apply for projects.
- Approve project selection forms by students, professors, and department chairs.

Guest Access:

- Allow students from other departments to log in as guests.
- Enable guests to add project proposals and apply for projects.

Project Monitoring:

- Facilitate weekly meetings between project groups and advisors.
- Allow regular entry and approval of project monitoring data by admin.
- Enable users to track project status.

Collaboration Tools:

- Provide an "I need" section for project teams to request help.
- Allow announcements for needs from different departments or sectors.

Documentation and Reporting:

- Enable students to upload reviews/surveys after the first 4 weeks.
- Allow uploading and tracking of work plans and project outputs.
- Facilitate the uploading of SRS and SDD documents before the end of the term.

Project Presentation and Evaluation:

- Allow the upload of project presentation video links.
- Enable advisors to evaluate projects and ask questions.
- Provide separate reporting of statistics for previous projects.

Integration with GitHub:

- Create a GitHub account for each project.
- Integrate with GitHub API for version control.

Task Management:

- Assign and track tasks for project teams.
- Provide Gantt charts or Kanban boards for visualizing project tasks and progress.

These functions collectively support the seamless management of graduation projects within the Çankaya University Software Engineering Department, ensuring efficient collaboration, documentation, and evaluation throughout the project lifecycle.

2.3. User Characteristics

Admin:

- Has full access to all features and functionalities of the PROBee application.
- Responsible for managing project proposals, including approval or rejection.
- Assigns stars to approved project proposals.
- Oversees project calls and team setups, ensuring smooth coordination.
- Monitors project progress, reviews project monitoring data, and updates project status.
- Evaluates project presentations, provides feedback, and reports project statistics.

Student:

- Submits project proposals and participates in project calls.
- Applies for projects and fills out project acceptance forms.
- Forms project groups, applies for projects, and participates in team setups.
- Regularly attends weekly meetings with advisors and enters project monitoring data.
- Uploads reviews/surveys, work plans, and project outputs.
- Uploads SRS and SDD documents before the end of the term.
- Presents project presentations and responds to advisor questions.

Advisor:

- Guides students in the project development process, providing expertise and support.
- Reviews project proposals and provides input during project selection.
- Oversees project teams, conducts weekly meetings, and monitors project progress.
- Evaluates project presentations, asks questions, and provides feedback.

Guest:

- Logged in as a guest, primarily from other departments.
- Can submit project proposals and apply for projects.
- Limited access compared to students and advisors, focusing on participation in the project proposal submission and project application process.

2.4. General Constraints

Time Constraints:

- The development timeline for PROBee is limited to the duration of the academic semester or project timeline.
- Deadlines for project submissions, approvals, presentations, and evaluations must be adhered to.

Resource Constraints:

- Limited resources, including human resources, funding, and infrastructure, may impact the development and deployment of PROBee.
- Availability of skilled developers, project advisors, and administrators may pose challenges.

2.5. Assumptions and Dependencies

Assumptions:

- Users have access to compatible web browsers and stable internet connections to interact with the PROBee application.
- Project proposals submitted by students meet the academic standards and requirements set by the Software Engineering Department.
- Advisors are available and committed to providing guidance and support to project teams throughout the project development process.

- Project teams adhere to the timelines and deadlines specified for project submissions, meetings, and document uploads.
- Integration with external systems and APIs, such as GitHub, operates smoothly without significant technical issues or constraints.
- Users comply with data protection regulations and maintain the confidentiality and security of their login credentials and personal information.

Dependencies:

- PROBee depends on the availability and reliability of the server infrastructure and database management system to ensure continuous operation and data integrity.
- Integration with external services, such as GitHub for version control, depends on the stability and compatibility of the APIs provided by these services.
- The successful adoption and usage of PROBee by users depend on effective communication, training, and support provided by the Çankaya University Software Engineering Department.
- Compliance with legal and regulatory requirements, including data protection laws and academic policies, is a prerequisite for the deployment and usage of PROBee.
- Any changes to the academic curriculum, graduation project guidelines, or institutional policies may impact the functionality and requirements of PROBee and require corresponding updates to the application.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

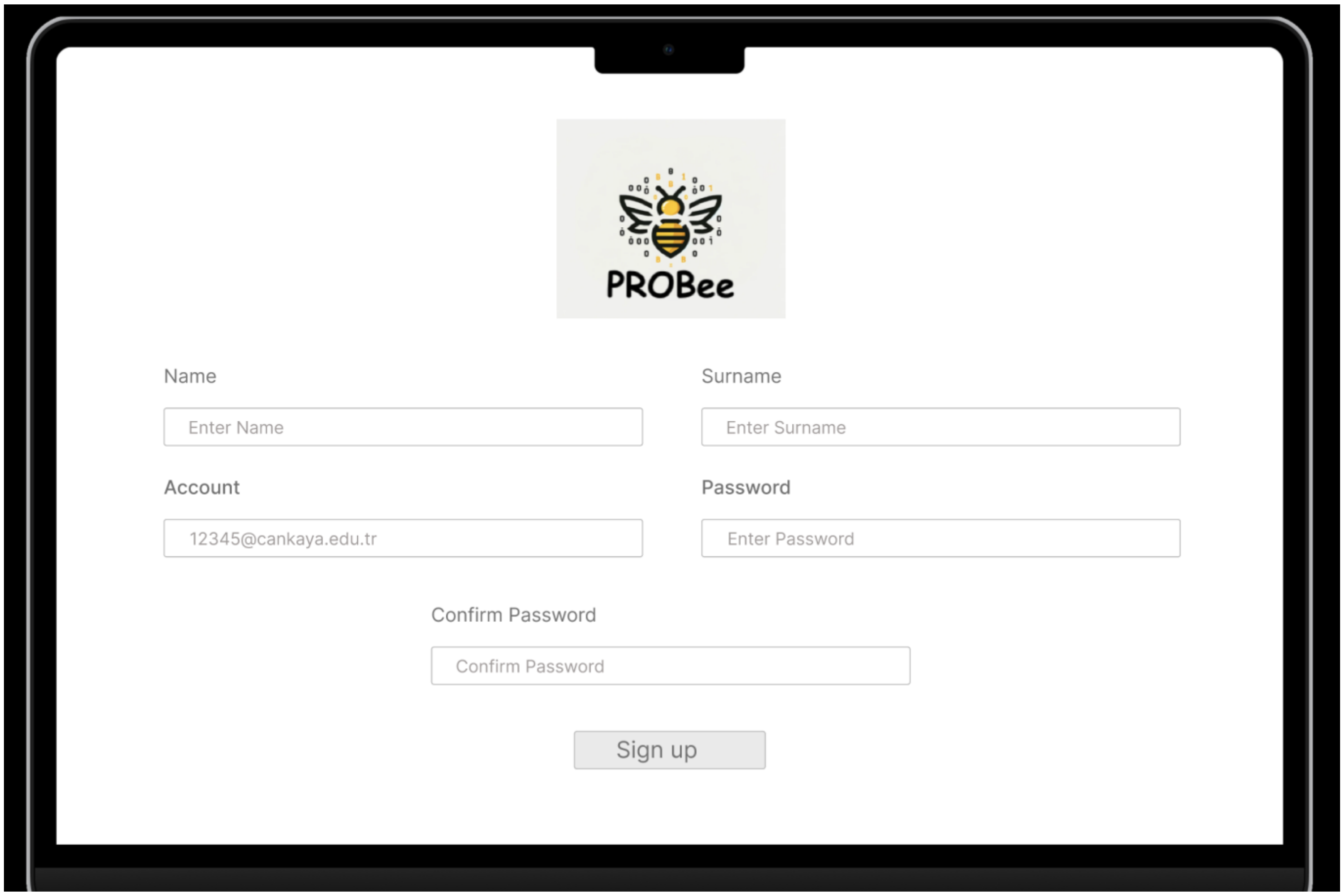


Image1: Register page

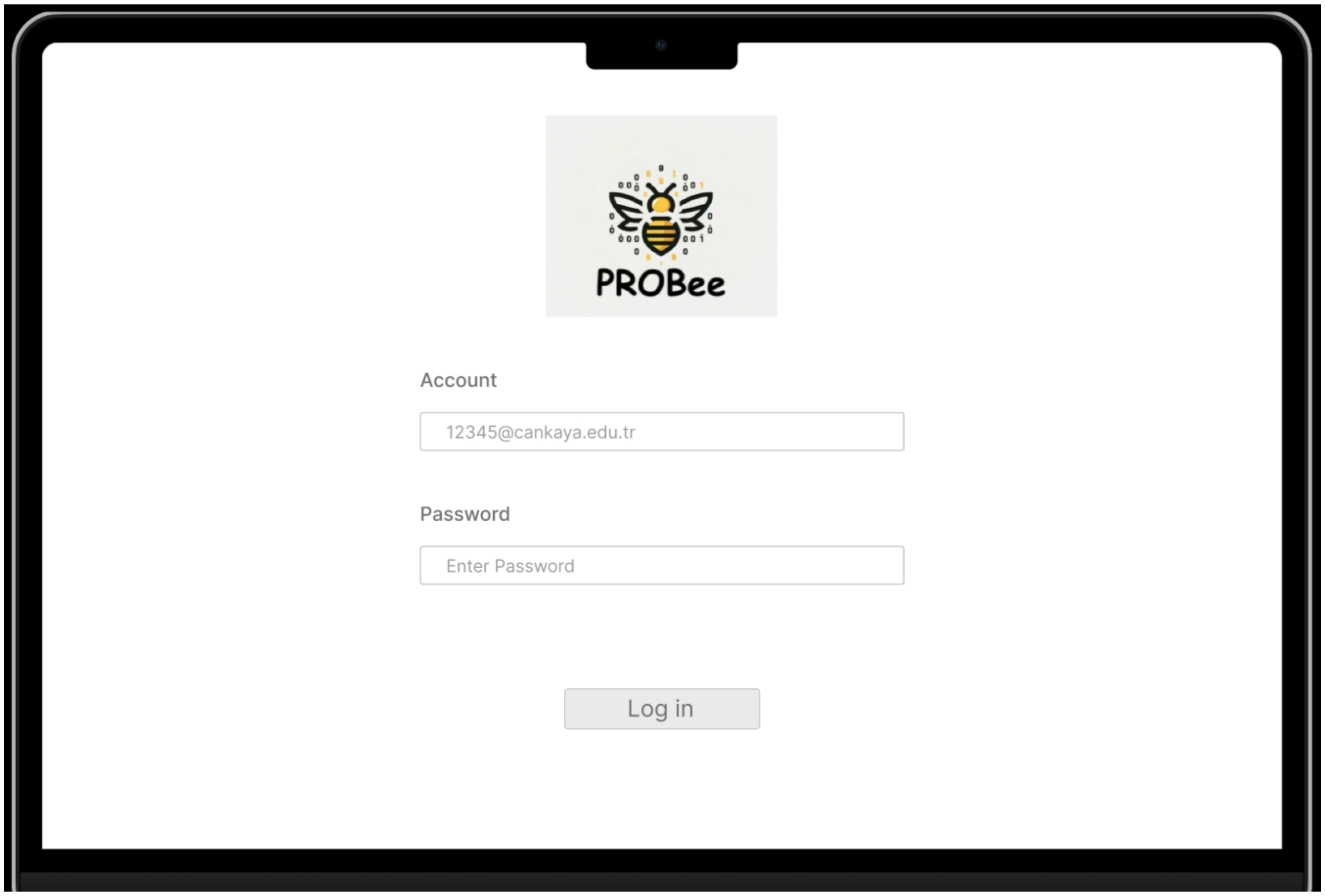


Image2: Login Page

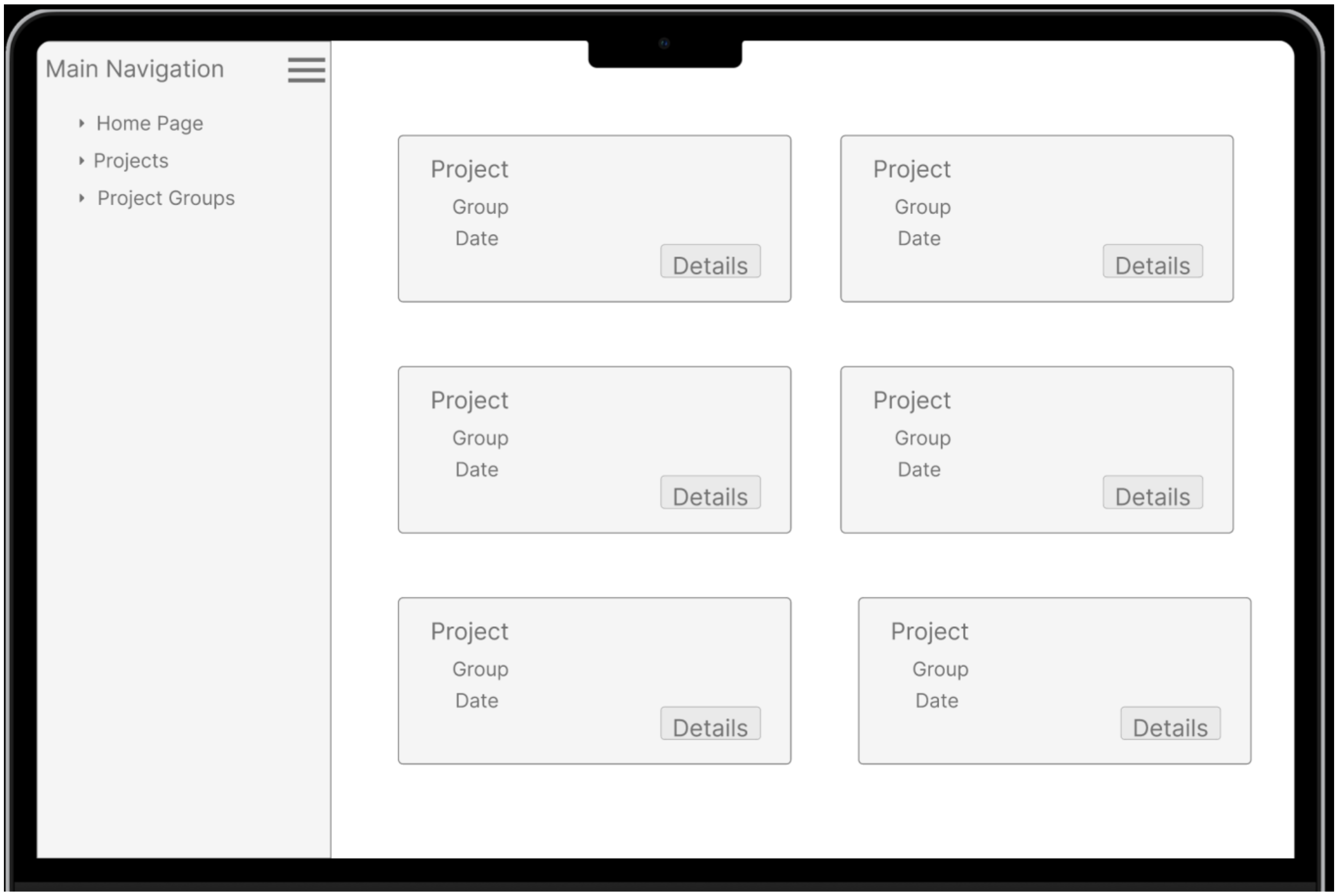


Image3: Projects page

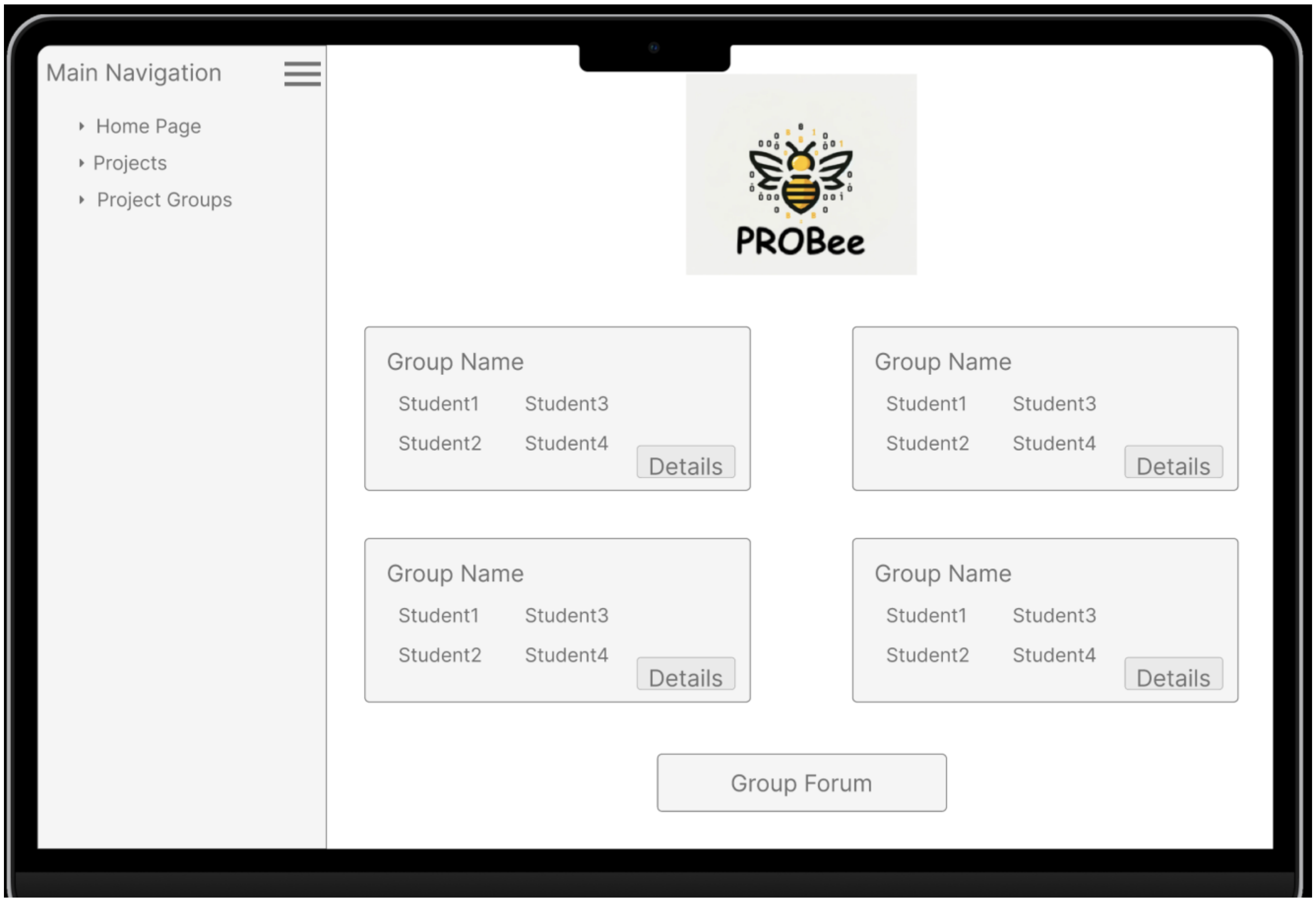
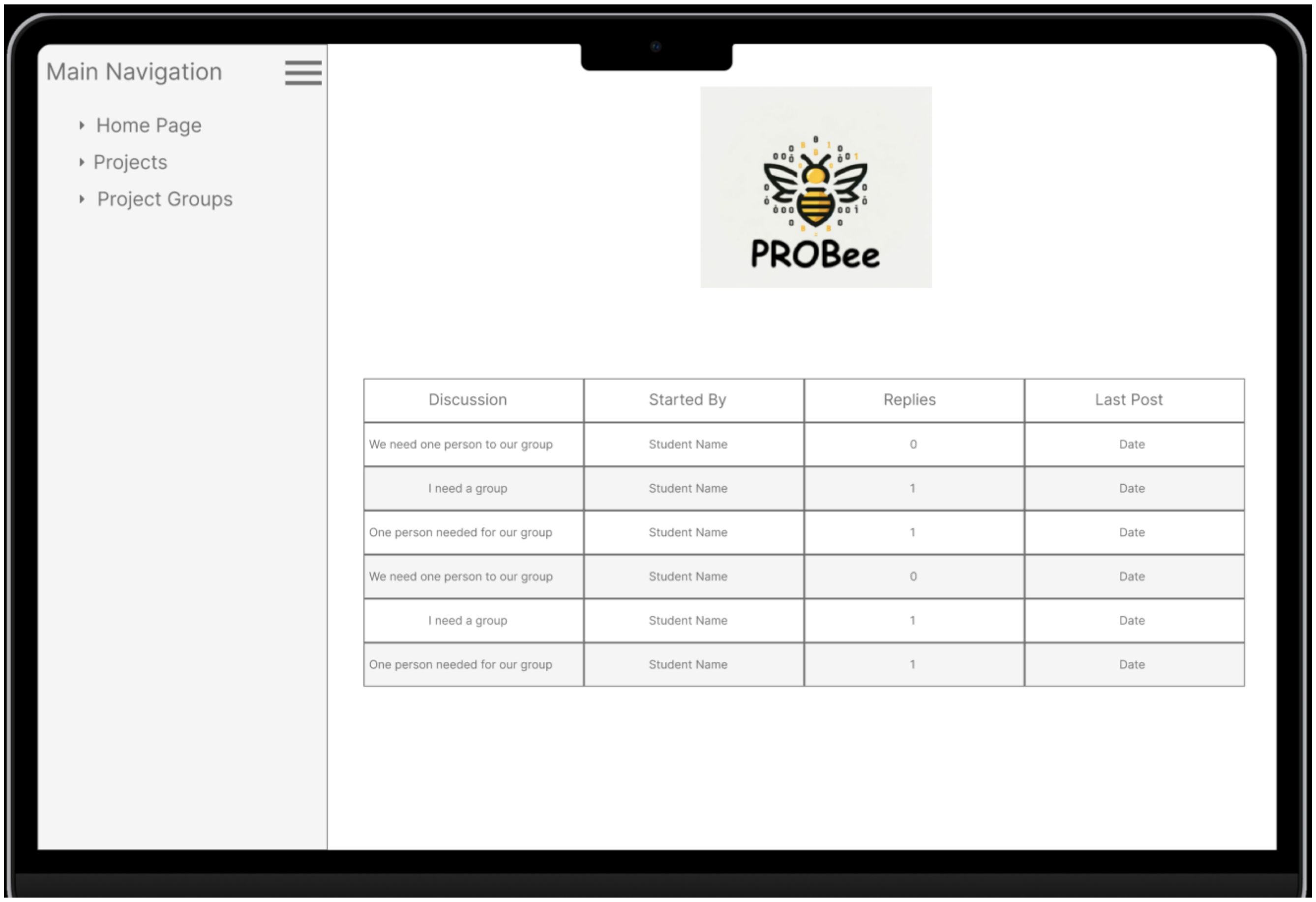


Image4: Groups Page



Discussion	Started By	Replies	Last Post
We need one person to our group	Student Name	0	Date
I need a group	Student Name	1	Date
One person needed for our group	Student Name	1	Date
We need one person to our group	Student Name	0	Date
I need a group	Student Name	1	Date
One person needed for our group	Student Name	1	Date

Image5: Student Forum

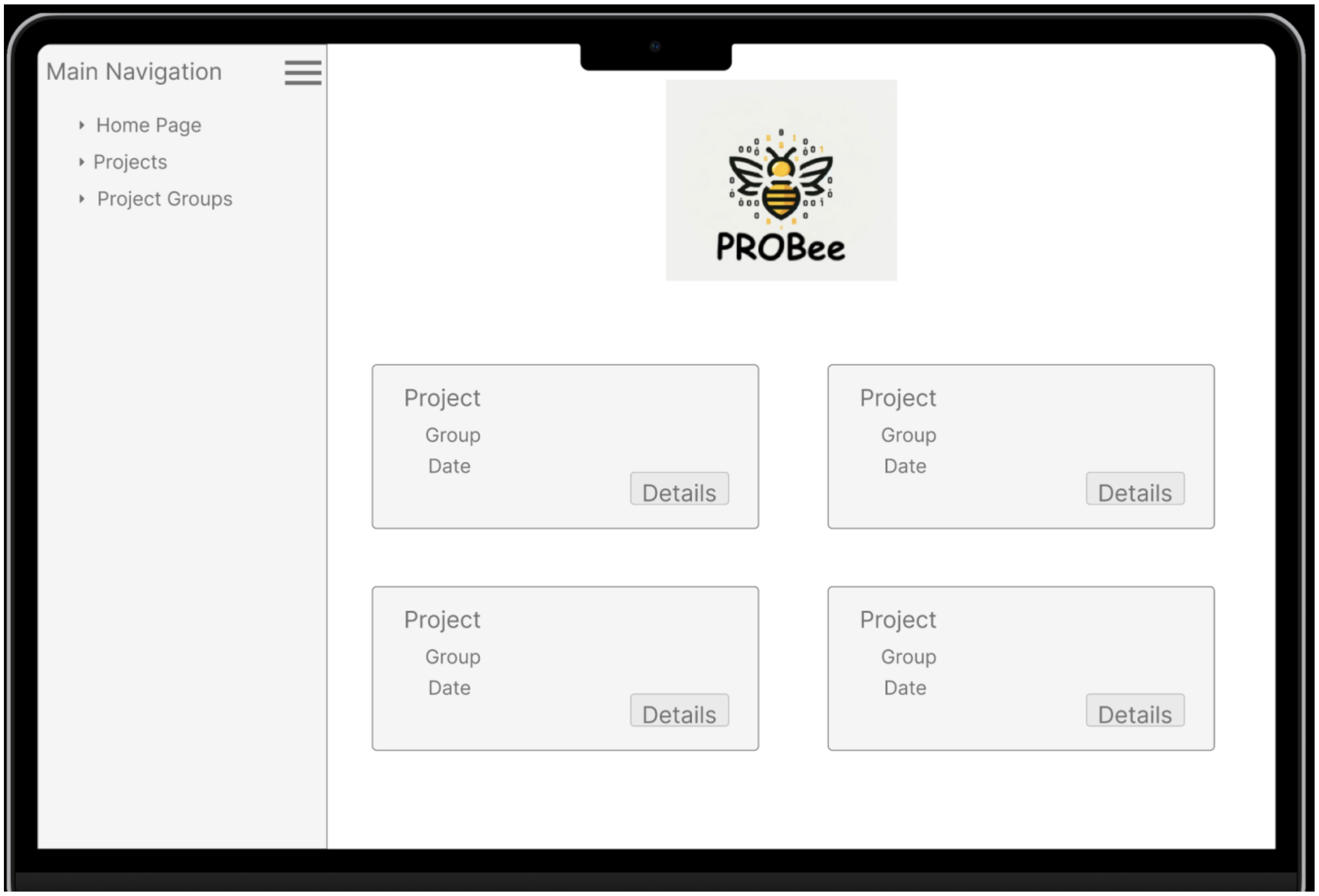


Image6: Teacher Homepage

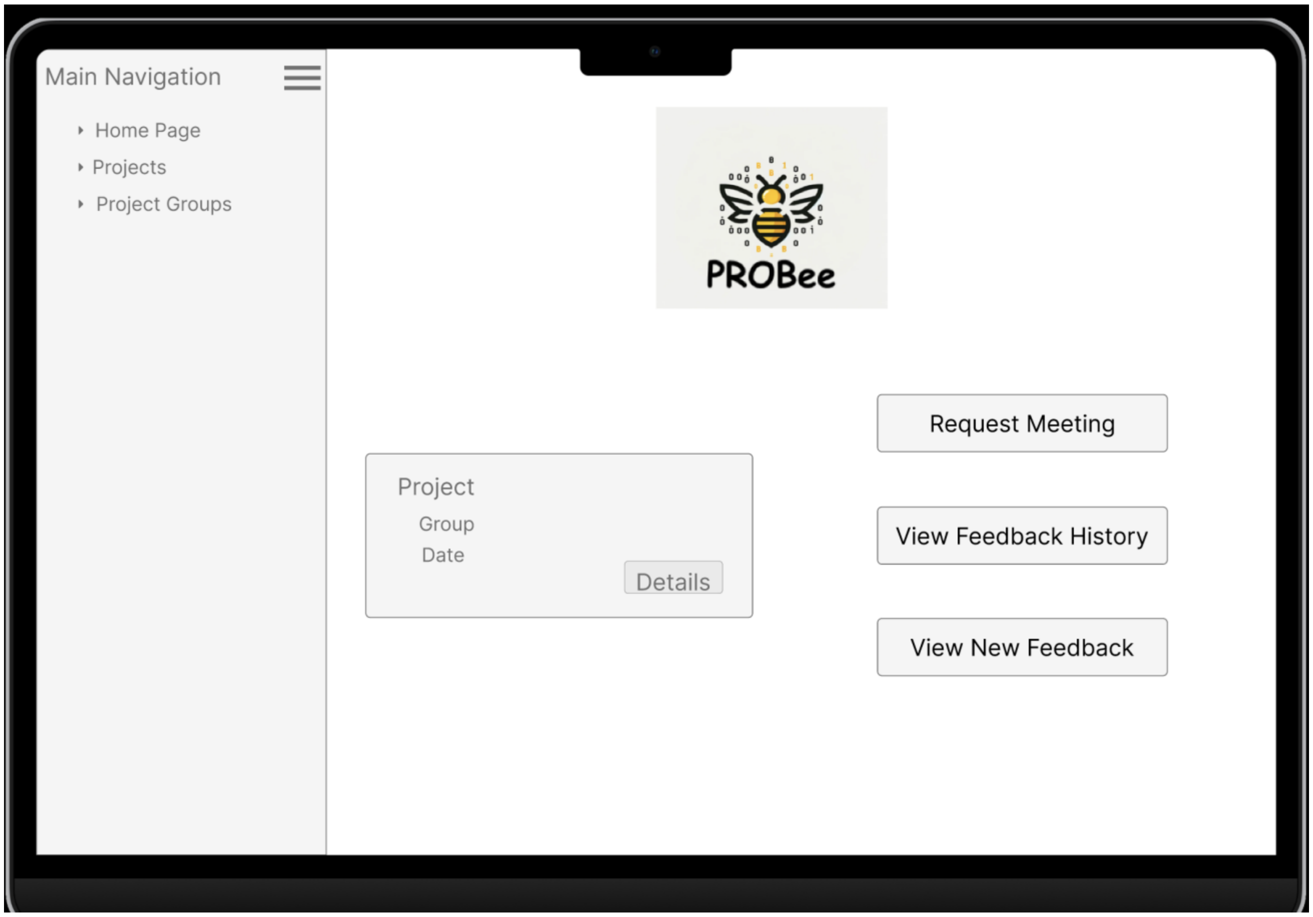


Image7: Student Homepage

3.1.2. Hardware Interfaces

- PROBee requires server hardware to host the web application and database components.
- The server should meet minimum requirements for CPU, memory, storage, and network bandwidth based on expected workload and scalability needs.

3.1.3. Software Interfaces

3.1.3.1. Web Browsers

- The PROBee web application interfaces with various web browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari, ensuring compatibility across different platforms and operating systems. Users can access the application using their preferred web browser, providing flexibility and convenience.

3.1.3.2. Database Management System (DBMS)

- PROBee interacts with a database management system (DBMS) to store and retrieve project-related data, user information, and system configurations. The application supports popular DBMS platforms such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server, ensuring flexibility and scalability in managing project data.

3.1.3.3. API Integration

- PROBee integrates with external APIs (Application Programming Interfaces) to enhance functionality and exchange data with third-party systems or services. For example, integration with the GitHub API enables version control and collaboration features, streamlining project development workflows.

3.1.4. Communication Interfaces

3.1.4.1. HTTPS Protocol

- The PROBee web application communicates with client devices using the Hypertext Transfer Protocol Secure (HTTPS). This protocol facilitates data exchange between the client's web browser and the application server, ensuring reliable and secure communication over the internet.

3.1.4.2. WebSocket Protocol

PROBee utilizes the WebSocket protocol to enable real-time communication between the client and server. WebSocket allows for bidirectional, full-duplex communication channels, enabling efficient exchange of messages and updates without the overhead of traditional HTTP polling. This facilitates interactive features such as instant messaging and live updates within the application.

3.1.4.3. Email Notifications

- The application sends email notifications to users for various events, such as project updates, task assignments, and system notifications. Email communication serves as a reliable means of keeping users informed and engaged, providing timely updates and alerts regarding project activities.

3.2. Functional Requirements

3.2.1. Detailed Description of Each Function

Student Functions:

1. **Register:**

   Description: Allows a new student to create an account by providing necessary details such as name, email, password, and department.

   Steps:

   - Fill out the registration form.
   - Submit the form.
   - Receive a confirmation email.
   - Verify email to activate the account.
2. **Login:**

   Description: Allows a registered student to log in using their email and password.

   Steps:

   - Enter email and password.
   - Click on the login button.
   - Access the student dashboard upon successful authentication.

3. **Submit Project Proposal:**

   Description: Allows a student to submit a new project proposal by filling out a Project Proposal Form.

   Steps:

   - Access the project proposal section.
   - Fill out the Project Proposal Form with details like project title, description, objectives, and technologies to be used.
   - Submit the form for admin approval.

4. **Form Project Groups:**

   Description: Allows students to form groups by inviting other students to join their project.

   Steps:

   - Access the team setup section.
   - Invite other students by entering their emails.
   - Confirm group formation once all members have accepted.

5. **Enter Weekly Data:**

   Description: Enables students to enter project monitoring data from their weekly meetings with advisors.

   Steps:

   - Access the project monitoring section.
   - Enter details of the weekly meetings and progress updates.
   - Submit the data for admin approval.

6. **Upload Work Plans:**

   Description: Enables students to upload their project work plans.

   Steps:

   - Access the documentation upload section.
   - Upload the work plan document.

7. **Upload Project Presentation Video:**

   Description: Enables students to upload a video presentation of their project.

   Steps:

   - Access the project presentation section.
   - Upload the video file or link.

8. **Link GitHub Repository:**

   Description: Allows students to link their project's GitHub repository for version control.

   Steps:

   - Access the GitHub integration section.
   - Enter the GitHub repository URL.
   - Authenticate with GitHub if required.

Teacher Functions:

1. **Login:**

   Description: Allows the admin to log in using their credentials.

   Steps:

   - Enter admin email and password.
   - Click on the login button.
   - Access the admin dashboard upon successful authentication.

2. **Approve/Reject Project Proposal:**

   Description: Allows the admin to review and approve or reject project proposals submitted by students.

   Steps:

   - Access the project proposals section.
   - Review each proposal.
   - Approve or reject the proposal and provide feedback if rejected.

3. **Approve Project Groups:**

   Description: Allows the admin to approve or reject the formation of project groups.

   Steps:

   - Access the project groups section.
   - Review group formation requests.
   - Approve or reject the requests.
4. **Approve Project Data:**

   Description: Enables the admin to review and approve project monitoring data submitted by students.

   Steps:

   - Access the project monitoring section.
   - Review the submitted data.
   - Approve or provide feedback for any changes needed.
5. **Track Project Status:**

   Description: Allows the admin to track the status and progress of all projects.

   Steps:

   - Access the project status section.
   - View the status and updates for each project.

3.2.2. Use Cases

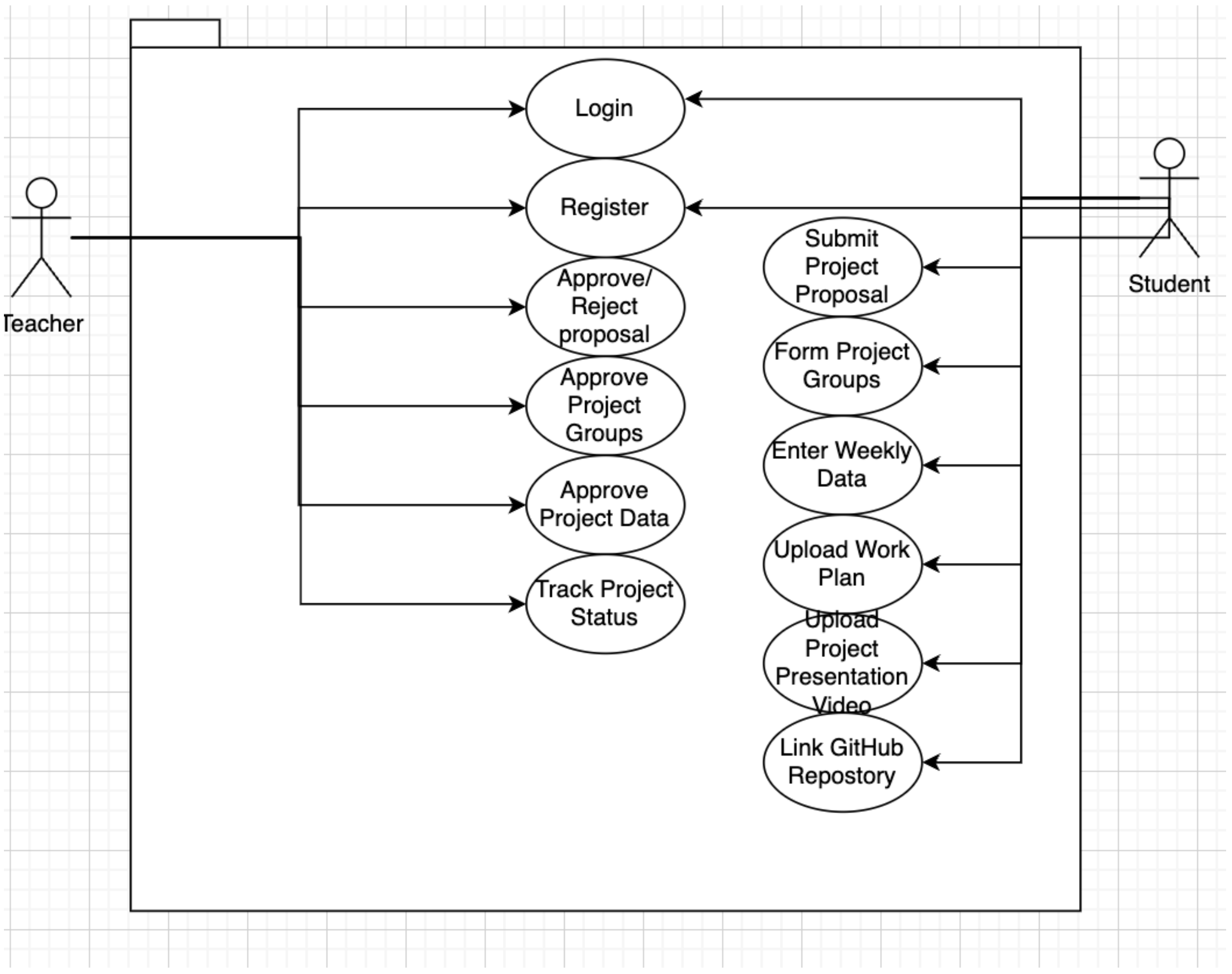


Image8: Use Case Diagram

3.2.3. Activity Diagram

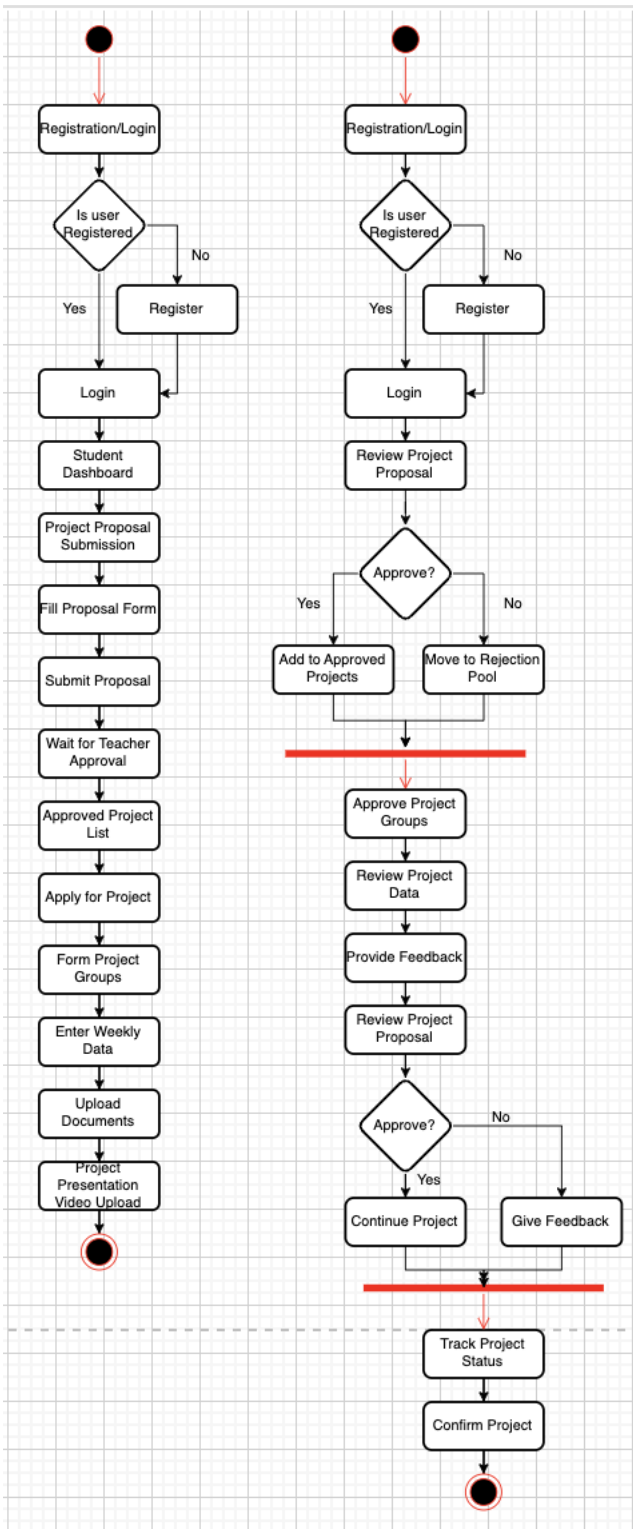


Image9: Activity Diagram

3.2.4. Sequence Diagram

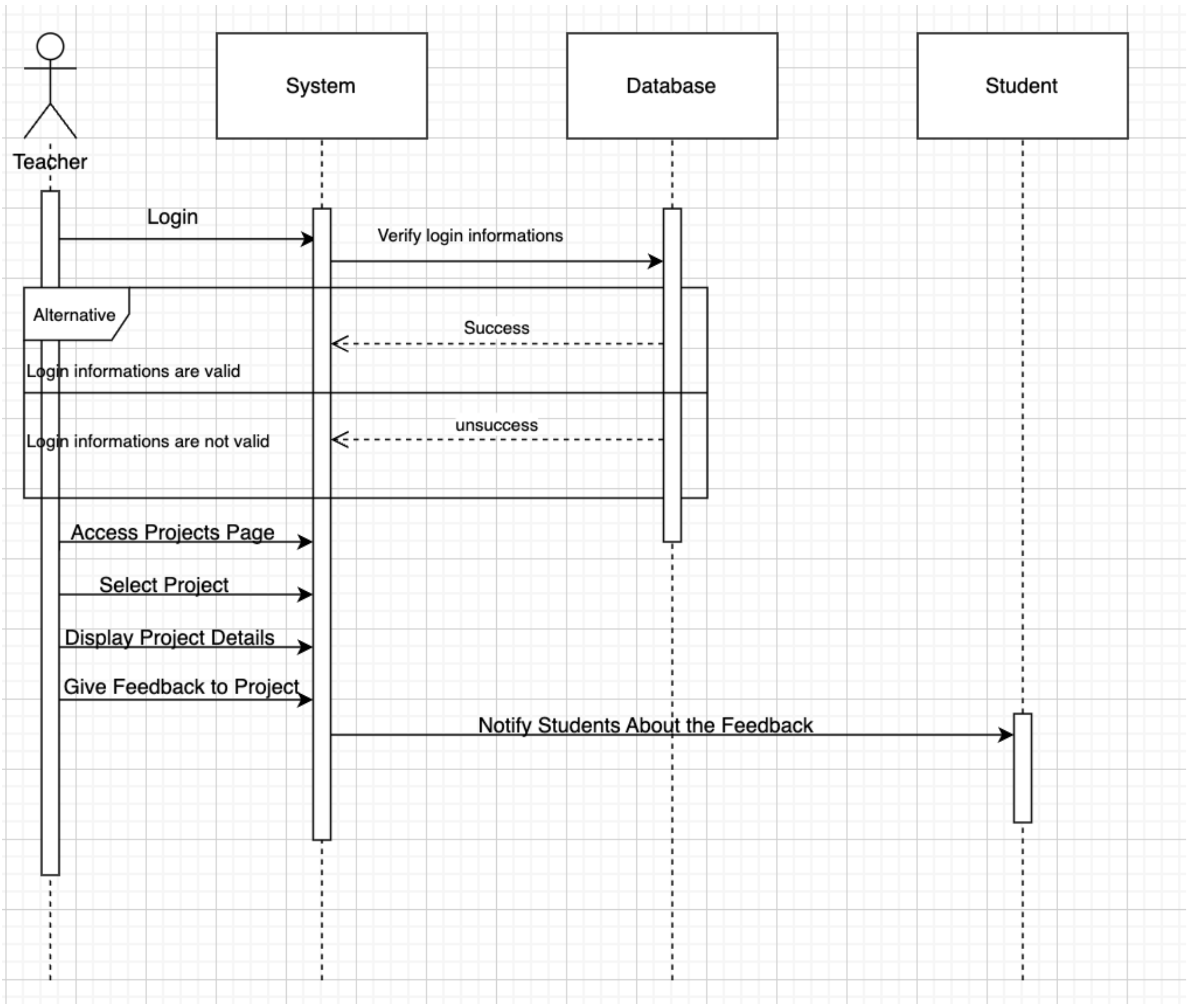


Image10: Sequence Diagram

4. Other Non-Functional Requirements

4.1. Performance Requirements

4.1.1. Response Time

- The system should respond to user actions within a maximum acceptable time limit.
- For example, the response time for loading a page or executing a query should not exceed 2 seconds.

4.1.2. Throughput

- The system should handle a specified number of transactions or requests per unit of time.
- For instance, the system should support a throughput of at least 100 requests per minute during peak usage periods.

4.1.3. Concurrency

- The system should support multiple concurrent users or transactions without experiencing performance degradation.
- It should be capable of handling a specified number of simultaneous connections or operations

4.1.4. Scalability

- The system should scale horizontally or vertically to accommodate increased load or user demand.
- It should be able to handle a growing number of users, data volumes, and transactions without significant performance impact.

4.1.5. Resource Utilization

- The system should optimize the utilization of hardware resources, such as CPU, memory, and disk space.
- It should minimize resource contention and efficiently allocate resources to maximize performance.

4.1.6. Database Performance

- Database queries and operations should be optimized for performance to minimize response times.
- Indexing, query optimization, and database tuning should be performed to enhance database performance.

4.1.7. Network Performance

- The system should maintain optimal network performance, with low latency and high throughput for data transmission.
- It should be resilient to network congestion, packet loss, and latency fluctuations to ensure consistent performance.

4.2. Design Constraints

- Design constraints for the PROBee web application encompass various considerations to ensure optimal performance, accessibility, and compatibility across different platforms and devices. These constraints include platform compatibility, requiring the application to function seamlessly across a diverse range of web browsers, operating systems, and devices. Additionally, responsive design principles are essential to adapt the application's layout and user interface to various screen sizes and resolutions, catering to users accessing the application from smartphones, tablets, and desktop computers. Bandwidth limitations must be addressed to optimize page load times and minimize network requests, particularly for users with limited internet bandwidth. Adherence to accessibility standards is crucial to ensure inclusivity for users with disabilities, necessitating features such as keyboard navigation and screen reader compatibility. Localization and internationalization support are imperative to accommodate users from different regions and language preferences, facilitating easy translation and localization of content. Security measures, including robust authentication mechanisms and data encryption, are vital to protect user data and mitigate security risks. Scalability and performance considerations involve optimizing database queries and caching strategies to maintain responsiveness and scalability under increasing user demand. Finally, seamless integration with external systems and APIs requires adherence to API specifications and error handling mechanisms to ensure robust interoperability with external services.

4.3. Software System Attributes

4.3.1. Scalability

- The system should be capable of handling increased workload or user demand without significant degradation in performance.
- It should scale horizontally by adding more resources or nodes to distribute the load effectively.

4.3.2. Reliability

- The system should consistently perform its intended functions correctly without failures or errors.
- It should be resilient to faults, failures, and unexpected events, ensuring continuous operation.

4.3.3. Availability

- The system should be available and accessible to users whenever needed, with minimal downtime or disruptions.
- It should incorporate redundancy, failover mechanisms, and disaster recovery strategies to maximize availability.

4.3.4. Performance

- The system should deliver optimal performance, responding to user requests promptly and processing data efficiently.
- It should minimize latency, throughput, and resource utilization to ensure responsiveness and scalability.

4.3.5. Security

- The system should protect sensitive data, prevent unauthorized access, and mitigate security risks effectively.
- It should enforce access controls, encryption, authentication, and authorization mechanisms to ensure confidentiality, integrity, and availability.

4.3.6. Maintainability

- The system should be easy to maintain, update, and enhance over time, with clear documentation, modular architecture, and well-structured code.
- It should facilitate code reuse, automated testing, and deployment processes to streamline maintenance activities.

4.3.7. Interoperability

- The system should seamlessly integrate with external systems, services, and APIs to exchange data and functionality.
- It should adhere to industry standards, protocols, and data formats to ensure interoperability and compatibility.

4.3.8. Flexibility

- The system should be flexible and adaptable to accommodate changes in requirements, environments, and technology.
- It should support configuration options, parameterization, and extensibility to enable customization and futureproofing

4.3.9. Usability

- The system should be intuitive and user-friendly, with a well-designed interface and logical navigation.
- It should support user productivity, efficiency, and satisfaction through effective interaction design and usability testing.

4.3.10. Portability

- The system should be portable across different platforms, operating systems, and environments, allowing for deployment flexibility.
- It should minimize dependencies on specific hardware or software configurations to maximize portability and interoperability.

4.4. Security Requirements

- The system shall implement a secure and robust authentication mechanism to verify the identity of users accessing the system.
- User passwords shall be stored securely using industry-standard encryption algorithms.
- The system shall enforce role-based access control (RBAC) to ensure that users can only access information and perform actions appropriate to their roles.
- Administrators shall have the authority to assign and revoke access privileges based on user roles.
- Communication between clients and the server, as well as between internal system components, shall be encrypted using industry-standard protocols (e.g., TLS/SSL).
- The system shall maintain an audit trail to log all significant events, including user logins, data modifications, and security-related incidents.
- User inputs and sensitive data transmitted over the network shall be encrypted to prevent eavesdropping and data interception.
- The system shall implement secure session management to protect against session hijacking and ensure the confidentiality of user sessions.
- The system shall implement mechanisms to ensure the integrity of stored data, preventing unauthorized modification or tampering.
- The system shall enforce password policies, including minimum complexity requirements, expiration periods, and account lockout mechanisms.
- Regular data backups shall be performed to prevent data loss in the event of system failures, and backup procedures shall be periodically tested for reliability.
- The system shall have an incident response plan in place to address security incidents promptly, including procedures for reporting, investigating, and mitigating security breaches.
- Physical access to servers and other critical infrastructure components shall be restricted to authorized personnel, and appropriate physical security measures shall be in place.
- The system shall comply with relevant data protection regulations, and measures shall be in place to ensure the privacy and protection of user data.

4.5. Software Quality Attributes

4.5.1. Reliability

- PROBee should operate reliably without unexpected failures or downtime, ensuring consistent availability and performance.
- It should handle errors gracefully and provide informative error messages to users.

4.5.2. Security

- The application should adhere to stringent security standards to protect user data, prevent unauthorized access, and mitigate security risks.
- Strong encryption, secure authentication mechanisms, and access controls should be implemented to safeguard sensitive information.

4.5.3. Usability

- PROBee should be intuitive and user-friendly, with a clean and responsive user interface that caters to users of all skill levels.
- Navigation should be straightforward, and tasks should be easily accessible and understandable.

4.5.4. Scalability

- The application should be scalable to accommodate a growing user base, increasing data volumes, and additional features.
- Scalability should be achieved through efficient use of resources, horizontal and vertical scaling options, and optimized performance.

4.5.5. Performance

- PROBee should deliver optimal performance, with fast response times and minimal latency, even under peak load conditions.
- Database queries, server-side processing, and client-side rendering should be optimized for speed and efficiency.

4.5.6. Maintainability

- The application should be easy to maintain and update, with well-structured code, modular architecture, and clear documentation.
- Changes and enhancements should be implemented without causing disruptions or introducing regressions.

4.5.7. Compatibility

- PROBee should be compatible with a wide range of web browsers, operating systems, and devices to ensure accessibility for all users.
- Compatibility testing should be conducted regularly to identify and address any compatibility issues.

4.5.8. Interoperability

- The application should seamlessly integrate with external systems, services, and APIs, such as GitHub for version control, to enhance functionality and interoperability.
- Standards-based protocols and data formats should be used to facilitate interoperability and data exchange.

4.5.9. Testability

- PROBee should be designed with testability in mind, with comprehensive test suites covering unit tests, integration tests, and end-to-end tests.
- Automated testing frameworks and continuous integration pipelines should be used to ensure robust test coverage and early detection of defects.

5. Conclusion

PROBee is the cornerstone platform for managing graduation projects within the Çankaya University Software Engineering Department. Seamlessly integrating essential functions such as project proposal submission, team formation, documentation upload, and project monitoring, PROBee streamlines the entire project lifecycle. By providing a centralized hub for students, advisors, and administrators to collaborate, track progress, and ensure timely completion, PROBee elevates the efficiency and effectiveness of graduation projects, fostering a culture of innovation and academic achievement.