Republic of Albania

Faculty of Computer Science and IT

Software Engineering

Software Project Management

Alket Sula

Roksana Ndreca

Miki Bode

Frenkli Koleci

Arbjona Mjeshtri

Enkeled Selami

Metropolitan Career System

Tirane 23.05.2024

I would like to thank everyone who contributed in this project, especially my team. Thank you for the great devotion and understanding of each task. Thank you for showing us that asking for help is the first step towards improvement, and showing me that working on a team is not as bad as it looks. Thanks to the professor's course I.Hakrama. Thank you for guiding us, with this, as a final project, beyond universities borders and expectations. Enjoy!!!

Table of Contents

1. Introduction

1.1 Background of the Project
1.2 Objectives
1.3 Scope
1.4 Significance of the Project

2. Literature Review

2.1 Overview of Alumni Management Systems
2.2 Software Project Management Methodologies
2.3 Best Practices in Web Development
2.4 Relevant Technologies and Tools

3. Methodology

3.1 Software Development Life Cycle (SDLC) Selection
3.2 Project Planning and Scheduling
3.3 Requirements Analysis and Management
3.4 System Design and Architecture
3.5 Implementation Strategies
3.6 Quality Assurance and Testing
3.7 Deployment and Maintenance Plans

4. System Requirements Analysis

4.1 Functional Requirements
4.2 Non-functional Requirements
4.3 User Stories and Use Cases
4.4 Stakeholder Analysis

5. System Design

5.1 System Architecture
5.2 Database Design
5.3 User Interface Design
5.4 Security Design
5.5 Scalability and Performance Considerations

6. Implementation

6.1 Coding Standards and Conventions
6.2 Version Control System Selection and Setup
6.3 Development Environment Configuration
6.4 Code Implementation Details

7. Testing and Quality Assurance

7.1 Test Plan and Strategy
7.2 Unit Testing
7.3 Integration Testing
7.4 System Testing
7.5 User Acceptance Testing
7.6 Bug Tracking and Resolution

8. Project Management

8.1 Team Organization and Roles
8.2 Communication Plan
8.3 Risk Management
8.4 Project Tracking and Reporting
8.5 Change Management

9. Deployment

9.1 Deployment Strategy
9.2 Server Configuration
9.3 Data Migration Plan
9.4 User Training and Documentation

10. Maintenance and Support

10.1 Ongoing Maintenance Plan
10.2 Issue Tracking and Resolution
10.3 Performance Monitoring and Optimization

11. Project Budget

12. Conclusions

12.1 Summary of Achievements
12.2 Lessons Learned
12.3 Future Enhancements

13. Diagrams

14. References

1. Introduction

In today's interconnected world, the value of alumni networks in educational institutions cannot be overstated. These networks serve as invaluable resources for fostering lifelong connections, facilitating professional growth, and enhancing institutional reputation. However, despite its evident benefits, our esteemed institution, University of Metropolitan Tirana(UMT), currently lacks a comprehensive alumni management system to harness the potential of its alumni network.

Recognizing the pivotal role that alumni engagement plays in the holistic development of our university community, a dedicated team of six individuals has embarked on a transformative endeavor to rectify this oversight. This project, undertaken as part of a Software Project Management course, endeavors to conceptualize, design, and implement a robust Alumni Management System tailored to the unique needs and aspirations of UMT.

1.1 Background of the Project

UMT, a bastion of academic excellence and innovation, has nurtured countless individuals who have gone on to achieve remarkable success in their respective fields. However, the absence of a centralized platform for alumni engagement has hindered the realization of the full potential of this rich and diverse network. Alumni remain dispersed, disconnected, and underutilized, depriving both the institution and its alumni of the myriad benefits that effective alumni management entails.

1.2 Objectives

The primary objective of this project is to design and develop a comprehensive Alumni Management System that will serve as the cornerstone of alumni engagement at UMT. Specifically, the project aims to:

- Establish a centralized platform for alumni networking, communication, and collaboration.
- Streamline alumni data management processes, ensuring accuracy, accessibility, and security.
- Facilitate seamless interaction between alumni, students, faculty, and staff members.
- Enhance institutional visibility and reputation through proactive alumni engagement initiatives.
- Cultivate a vibrant and supportive alumni community committed to the advancement of UMT's mission and vision.

1.3 Scope

The scope of the project encompasses the conceptualization, development, testing, deployment, and maintenance of the Alumni Management System. Key features and functionalities to be included in the system include:

- Alumni profiles with comprehensive information about their academic and professional achievements.
- Job posting and application functionalities for both alumni and current students.
- Event management capabilities to organize alumni reunions, networking events, and professional development workshops.
- Discussion forums and groups to facilitate knowledge sharing, mentorship, and collaboration.
- Robust administrative tools for managing user accounts, content moderation, and system configuration.

1.4 Significance of the Project

The successful implementation of the Alumni Management System holds immense significance for UMT and its stakeholders. By fostering meaningful connections, fostering knowledge exchange, and promoting lifelong learning, the system will contribute to the enrichment of the university experience for current students and alumni alike. Furthermore, by harnessing the collective expertise, resources, and goodwill of its alumni community, UMT will be better positioned to achieve its strategic goals and aspirations in the years to come.

2. Literature Review

2.1 Overview of Alumni Management Systems

Alumni Management Systems (AMS) play a pivotal role in fostering lifelong relationships between educational institutions and their graduates. These systems typically encompass a suite of features designed to streamline alumni engagement, communication, and data management processes. Key functionalities of AMS include alumni database management, event coordination, fundraising initiatives, career services, and networking opportunities. By centralizing alumni information and facilitating seamless interaction between alumni and their alma mater, AMS contribute to enhanced institutional visibility, alumni satisfaction, and philanthropic support.

2.2 Software Project Management Methodologies

Effective software project management methodologies are essential for ensuring the successful planning, execution, and delivery of software projects. Various methodologies, such as Waterfall, Agile, Scrum, and Kanban, offer distinct approaches to project management, each tailored to specific project requirements and organizational contexts. In the context of our project, agile methodologies may be particularly relevant, allowing for flexibility and adaptability in response to evolving requirements. By selecting and adhering to an appropriate project management methodology, teams can enhance project transparency, productivity, and stakeholder satisfaction.

2.3 Best Practices in Web Development

Web development encompasses a myriad of practices, techniques, and tools aimed at creating dynamic, responsive, and user-friendly websites and web applications. Best practices in web development encompass areas such as front-end development, back-end development, user experience design, performance optimization, and security. In our project, utilizing phpMyAdmin as the database management tool complements the back-end development process, enabling efficient database administration and interaction. Additionally, robust security measures, including data encryption, input validation, and access control, should be implemented to safeguard the integrity and confidentiality of the database.

2.4 Relevant Technologies and Tools

The successful development of the Alumni Management System relies on leveraging a diverse array of technologies and tools tailored to the project's requirements and constraints. Key technologies include PHP for server-side scripting, HTML, CSS, and JavaScript for front-end development, and Bootstrap for responsive web design. Version control systems such as Git enable collaborative development and code management, while project management tools such as Jira or Trello facilitate task tracking and team communication. Additionally, phpMyAdmin serves as the database management tool, providing an intuitive interface for database administration, query execution, and data manipulation.

3. Methodology

3.1 Software Development Life Cycle (SDLC) Selection

For the development of the Alumni Management System, an Agile Software Development Life Cycle (SDLC) model has been selected. Agile methodologies, such as Scrum, promote iterative development, continuous collaboration with stakeholders, and adaptability to changing requirements. This approach facilitates rapid delivery of functional increments, enabling early feedback and continuous improvement throughout the development process.

3.2 Project Planning and Scheduling

Project planning and scheduling in an Agile context involve breaking down the project into manageable iterations or sprints, each lasting typically 2-4 weeks. The project backlog is prioritized, and user stories are selected for inclusion in each sprint based on their priority and complexity. Sprint planning meetings are conducted to define sprint goals, select user stories, and estimate the effort required for their completion. A sprint backlog is then created, detailing the tasks to be undertaken during the sprint.

3.3 Requirements Analysis and Management

Requirements analysis and management are conducted iteratively throughout the project, with user stories serving as the primary mechanism for capturing and prioritizing requirements. During sprint planning meetings, user stories are refined, elaborated, and estimated by the development team in collaboration with stakeholders. Any ambiguities or uncertainties regarding requirements are clarified through continuous communication and feedback loops.

3.4 System Design and Architecture

System design and architecture in an Agile environment focus on creating flexible, modular, and scalable solutions that can evolve over time. Design decisions are made incrementally, with each sprint delivering incremental improvements to the system's architecture and functionality. Architectural decisions are guided by the project's overall goals, user requirements, and feedback obtained from stakeholders.

3.5 Implementation Strategies

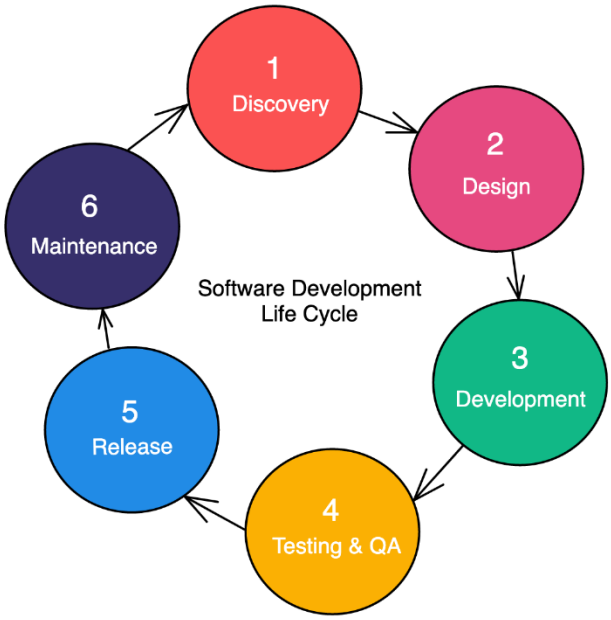
Implementation strategies in Agile development involve iterative coding, testing, and integration of system components. Each sprint culminates in a potentially shippable product increment, which is rigorously tested to ensure its quality and functionality. Pair programming, test-driven development (TDD), and continuous integration (CI) are employed to maintain code quality and facilitate collaboration among team members.

3.6 Quality Assurance and Testing

Quality assurance and testing are integral parts of Agile development, with testing activities conducted continuously throughout the project lifecycle. Sprint testing involves validating user stories against acceptance criteria defined by stakeholders, ensuring that each increment meets the desired quality standards. User Acceptance Testing (UAT) is conducted at the end of each sprint to solicit feedback from stakeholders and verify that the system meets their expectations.

3.7 Deployment and Maintenance Plans

Deployment in an Agile context involves releasing tested and validated increments of the Alumni Management System into production at the end of each sprint. Continuous deployment pipelines automate the deployment process, ensuring that new features and updates are delivered to users seamlessly. Post-deployment, a maintenance plan is established to address any issues or defects identified in production and to prioritize feature enhancements based on stakeholder feedback.



4. System Requirements Analysis

4.1 Functional Requirements

Functional requirements define the specific behaviors, actions, and capabilities that the Alumni Management System must exhibit to fulfill its intended purpose. These requirements typically describe the system's features, functionalities, and interactions with users and other systems. Examples of functional requirements for the Alumni Management System may include:

- **User authentication and authorization:** The system should allow administrators to create and manage user accounts, with different access levels and permissions.
- **Alumni profile management:** Users should be able to create, update, and delete their profiles, including personal information, education history, and employment details.
- **Job posting and application:** The system should facilitate the posting of job opportunities by administrators and allow alumni to browse and apply for jobs.
- **Event management:** Administrators should be able to create, schedule, and manage events, such as alumni reunions, workshops, and networking sessions.
- **Forum functionality:** The system should include discussion forums where users can engage in conversations, ask questions, and share insights on relevant topics.

Functional requirements are typically documented using techniques such as use cases, user stories, and functional specifications, providing a clear understanding of the system's expected behavior from a user's perspective.

4.2 Non-functional Requirements

Non-functional requirements define the quality attributes, constraints, and performance characteristics that the Alumni Management System must adhere to. These requirements encompass aspects such as reliability, scalability, usability, security, and performance. Examples of non-functional requirements for the system may include:

- **Performance:** The system should be able to handle a large volume of concurrent users and maintain acceptable response times under peak loads.
- **Security**: User data should be encrypted during transmission and storage, and access to sensitive information should be restricted based on user roles and permissions.
- **Usability:** The system should have an intuitive user interface, with clear navigation and responsive design to ensure accessibility across devices and platforms.
- **Reliability:** The system should be robust and resilient, with mechanisms in place to handle errors, failures, and interruptions gracefully.
- **Scalability:** The system should be designed to accommodate future growth and expansion, with provisions for scaling resources, such as servers and databases, as needed.

Non-functional requirements are typically documented alongside functional requirements, ensuring that the system meets both functional and quality criteria for successful implementation and operation.

4.3 Users Stories and Use Cases

User stories and use cases provide detailed descriptions of system interactions from the perspective of different stakeholders, including users, administrators, and system integrators. User stories are concise, informal narratives that describe specific user goals or tasks and the desired outcomes. Use cases, on the other hand, are more formalized descriptions of system behavior, depicting interactions between actors (users) and the system to accomplish specific goals or tasks.

For example, a user story for the Alumni Management System may be:

"As an alumni member, I want to be able to update my profile information, including my contact details and employment history, so that my information remains up-to-date and relevant."

A corresponding use case for this user story may involve the following steps:

- The user navigates to the profile settings page and selects the option to edit their profile.
- The system presents the user with a form containing fields for updating personal information, education history, and employment details.
- The user makes the necessary changes to their profile information and submits the form.
- The system validates the user input, updates the user's profile in the database, and displays a confirmation message to the user.

4.3.1 User Stories for Admin:

- **As an admin**, I want to be able to add new job postings, including job titles, descriptions, requirements, and application deadlines, so that alumni and students can view and apply for relevant job opportunities.

Use cases:

- The admin navigates to the job posting management section of the system.
- The admin fills out a form with details of the job posting, including title, description, requirements, and deadline.
- The system validates the input and adds the job posting to the database.
- Upon successful addition, the system notifies users of the new job posting.

- **As an admin**, I want to review and approve alumni profile updates, ensuring that only accurate and relevant information is displayed on the platform.

Use cases:

- The admin receives notifications for pending profile updates.
- The admin accesses the profile management section and reviews the pending updates.
- The admin verifies the accuracy and relevance of the information provided by the alumni.
- If the information is valid, the admin approves the profile update, and the system reflects the changes accordingly.

- **As an admin**, I want to create and manage events, including alumni reunions, workshops, and networking sessions, to foster engagement and collaboration within the alumni community.

Use cases:

- The admin accesses the event management section of the system.
- The admin creates a new event by specifying details such as title, description, date, time, and location.
- The system adds the event to the calendar and notifies users of the upcoming event. ETC.

4.3.2 User Stories for Alumni Members

- **As an alumni member,** I want to update my contact information, including email address, phone number, and mailing address, to ensure that I receive relevant updates and communications from the university.

Use cases:

- The alumni member logs into their account and navigates to the profile settings page.
- The alumni member updates their contact information in the provided form.
- The system validates the input and updates the alumni member's profile with the new information.
- The system sends a confirmation email to the alumni member to verify the changes.

- **As an alumni member,** I want to browse and apply for job postings relevant to my field of expertise and interests, to explore career opportunities and stay connected with the university community..

Use cases:

- The alumni member logs into the system and navigates to the job postings section.
- The alumni member filters job postings based on criteria such as industry, location, and job title.
- The alumni member reviews job descriptions and requirements and selects positions of interest.
- The system facilitates the application process, allowing the alumni member to submit their resume and cover letter to the employer.

- **As an alumni member,** I want to RSVP for upcoming alumni events, such as reunions and networking sessions, to reconnect with former classmates and expand my professional network.

Use cases:

- The alumni member receives notifications for upcoming events via email or in-app notifications.
- The alumni member views the event details and RSVPs by indicating their attendance status.
- The system updates the event attendance list accordingly and sends confirmation to the alumni member.

4.4 Stakeholder Analysis

Stakeholder analysis involves identifying and understanding the individuals, groups, or organizations that have a vested interest in the success of the Alumni Management System. Stakeholders may include university administrators, alumni association members, current students, faculty members, employers, and system administrators.

Each stakeholder group may have distinct interests, needs, and priorities, which must be taken into account during the requirements elicitation and prioritization process. For example, alumni may prioritize features such as job postings and networking opportunities, while university administrators may prioritize administrative functionalities such as user management and event coordination.

Stakeholder analysis helps ensure that the system requirements adequately reflect the needs and expectations of all relevant stakeholders, fostering buy-in, collaboration, and support throughout the project lifecycle.

5. System Design

5.1 System Architecture

The system architecture of the Alumni Management System is designed to ensure modularity, scalability, and maintainability. The system follows a three-tier architecture, consisting of the presentation layer, application layer, and data layer.

- **Presentation Layer:** The presentation layer encompasses the user interface components of the system, including web pages, forms, and interactive elements. The user interface is designed to be intuitive, responsive, and accessible across devices and platforms.
- **Application Layer:** The application layer contains the business logic and functional components of the system, including modules for user authentication, profile management, job postings, events, and forums. This layer is implemented using PHP for server-side scripting and JavaScript for client-side interactions.
- **Data Layer:** The data layer comprises the database management system (DBMS) and storage mechanisms used to persist and retrieve data. The system utilizes MySQL as the relational database management system (RDBMS), managed and accessed through phpMyAdmin for database administration.

5.2 Database Design

The database design of the Alumni Management System is structured to efficiently store and manage data related to alumni profiles, job postings, events, forums, and user interactions. The database schema is normalized to minimize redundancy and ensure data integrity, with tables organized around entities and their relationships.

Key tables in the database schema include:

- **Alumni:** Stores information about alumni profiles, including personal details, education history, employment records, and contact information.
- **Jobs:** Contains details of job postings, including job titles, descriptions, requirements, application deadlines, and employer information.
- **Events:** Stores information about upcoming events, such as alumni reunions, workshops, and networking sessions, including event titles, descriptions, dates, times, and locations.
- **Users:** Manages user accounts and authentication credentials, including usernames, passwords, and access permissions.
- **Forums:** Stores discussion topics, posts, and comments contributed by users, facilitating knowledge sharing and community engagement.

5.3 User Interface Design

The user interface design of the Alumni Management System is crafted to provide a seamless and intuitive user experience. The interface is designed using HTML, CSS, and Bootstrap framework for responsive design, ensuring compatibility with a wide range of devices and screen sizes.

Key design principles and features of the user interface include:

- **Clear Navigation:** The interface includes intuitive navigation menus, buttons, and links, allowing users to easily access different sections and features of the system.
- **Consistent Layout:** Consistency in layout, typography, and color scheme enhances usability and brand identity, providing a cohesive visual experience across the application.
- **Interactive Elements:** Interactive elements such as forms, buttons, and dropdown menus enable users to perform actions and interact with the system seamlessly.
- **Feedback and Notifications**: The interface provides feedback messages and notifications to inform users about successful actions, errors, and important updates, enhancing user engagement and satisfaction.

5.4 Security Design

Security design is paramount in ensuring the confidentiality, integrity, and availability of data within the Alumni Management System. The system implements robust security measures to protect against unauthorized access, data breaches, and malicious attacks.

Key security features and practices include:

- **User Authentication**: The system employs secure authentication mechanisms, such as password hashing and salting, to authenticate users and prevent unauthorized access to accounts.

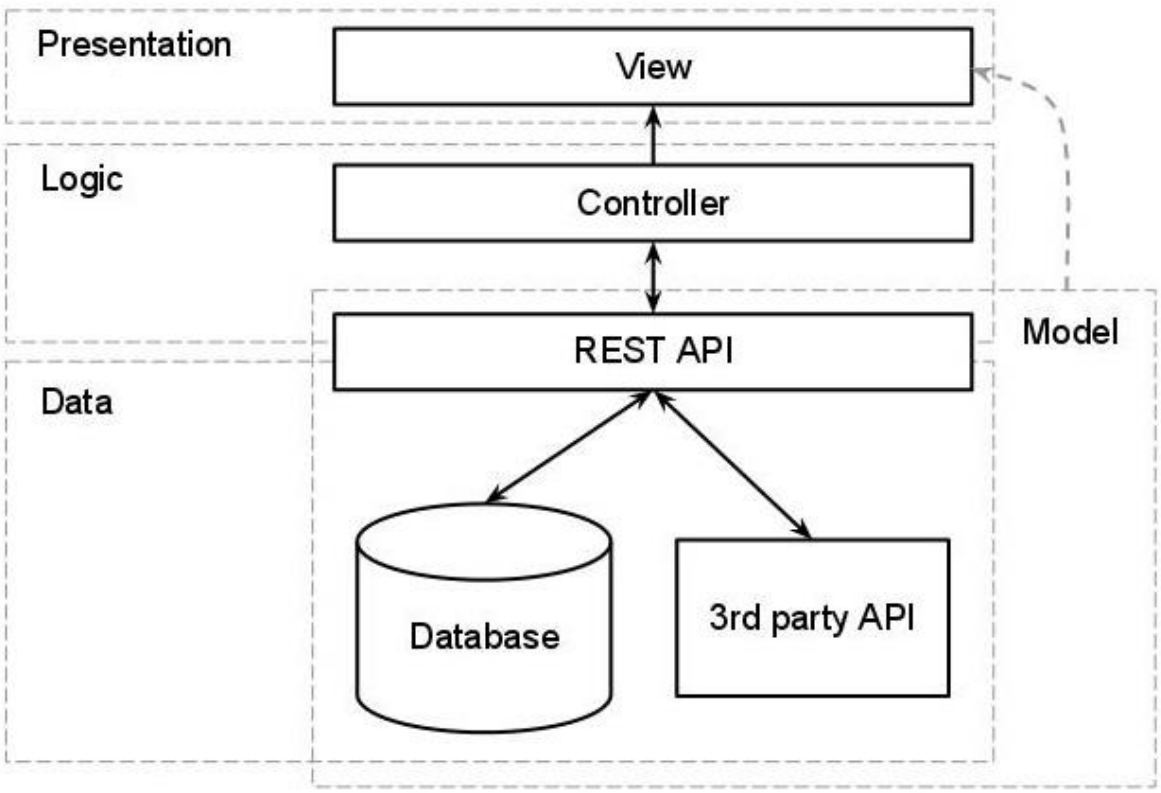
- **Access Control:** Role-based access control (RBAC) is implemented to restrict access to sensitive functionalities and data based on user roles and permissions.
- **Data Encryption:** Sensitive data, such as user credentials and personal information, are encrypted during transmission and storage to prevent unauthorized disclosure.
- **Input Validation:** Input validation techniques are used to sanitize user inputs and mitigate risks associated with SQL injection, cross-site scripting (XSS), and other common security vulnerabilities.

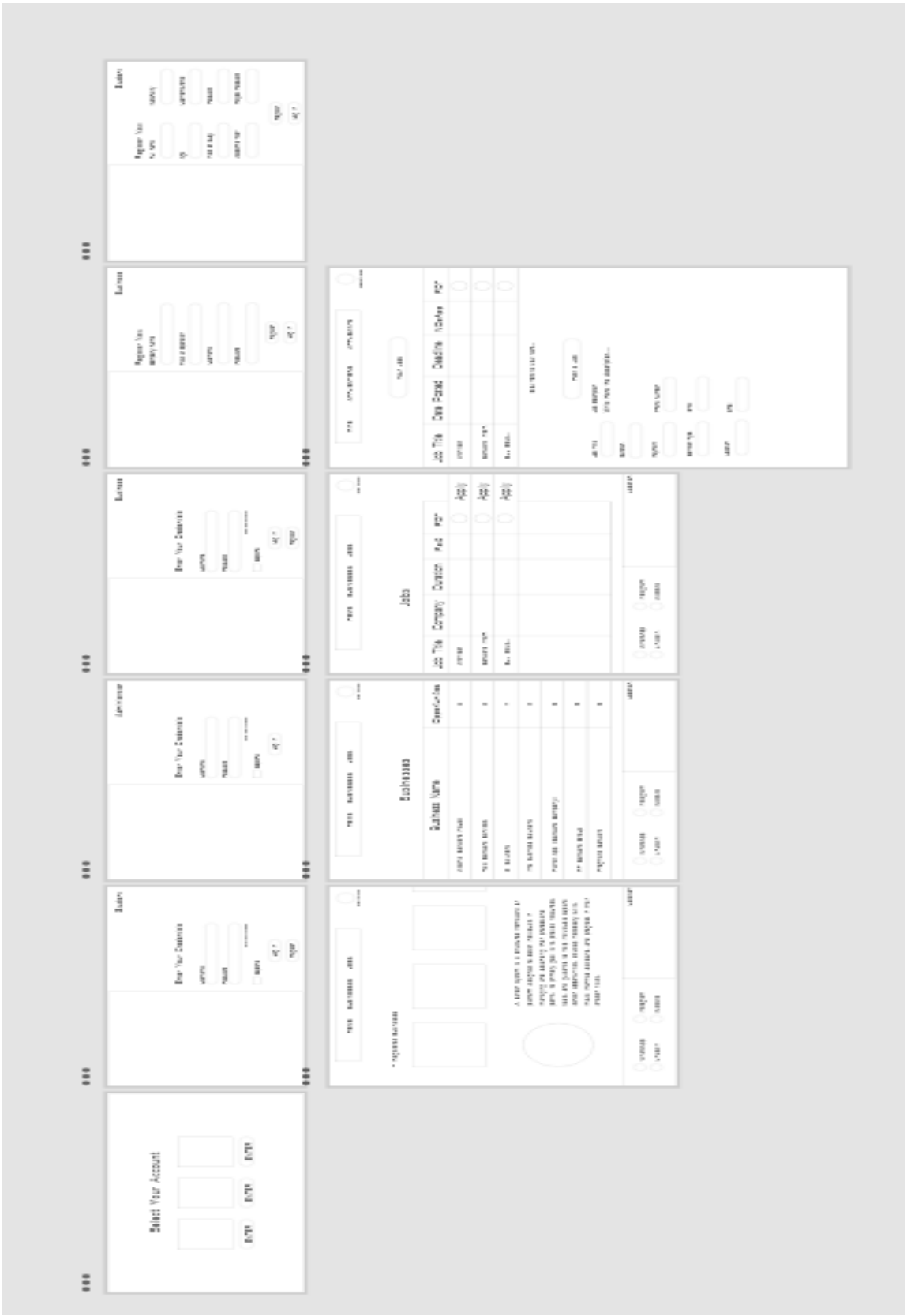
5.5 Scalability and Performance Considerations

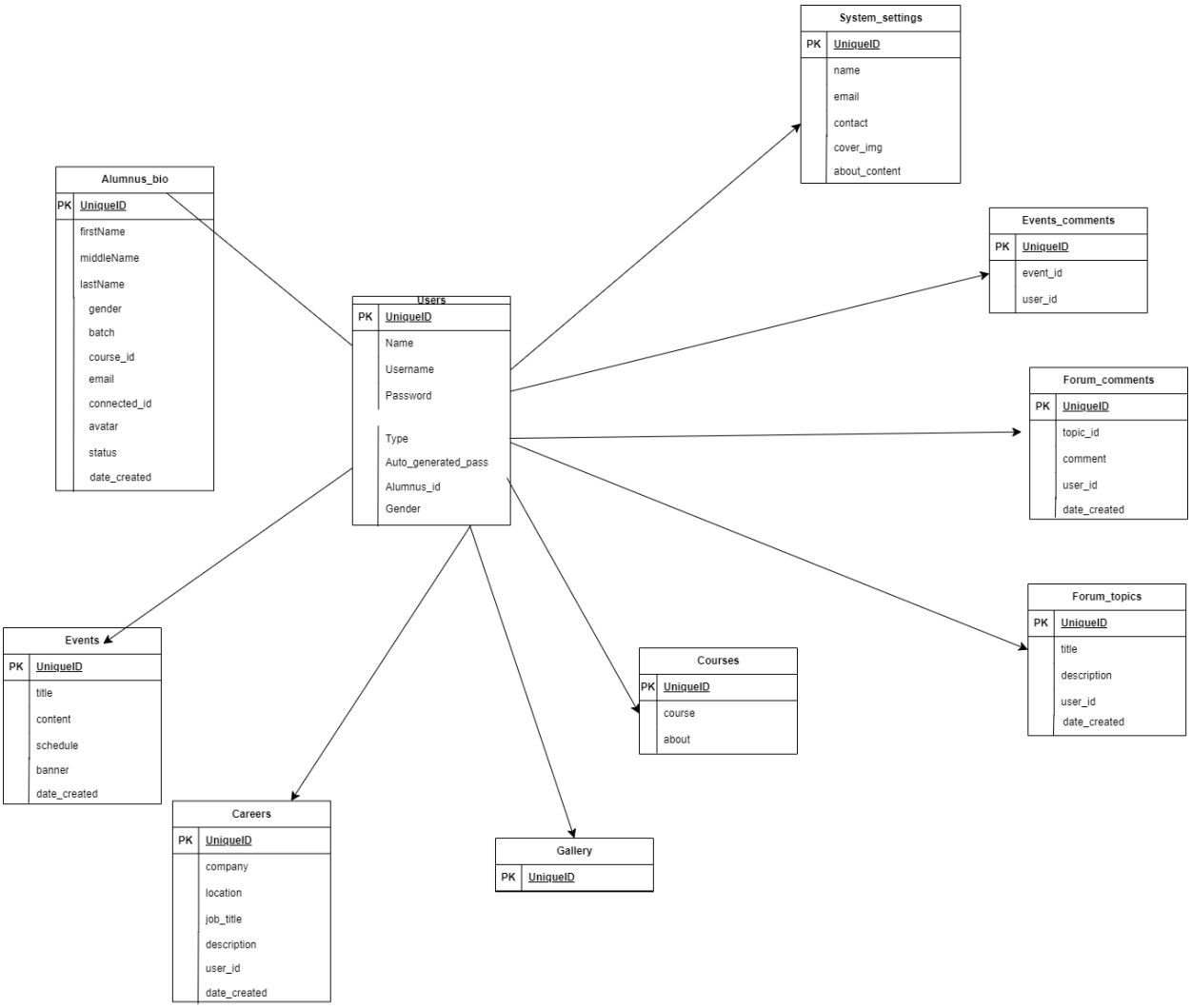
Scalability and performance considerations are essential for ensuring that the Alumni Management System can accommodate growing user bases and handle increasing loads without compromising performance.

Key strategies for scalability and performance optimization include:

- **Load Balancing**: Load balancing techniques distribute incoming traffic across multiple servers, ensuring optimal resource utilization and minimizing response times.
- **Caching:** Caching mechanisms, such as content caching and database query caching, help reduce latency and improve responsiveness by storing frequently accessed data and resources in memory.
- **Database Indexing**: Database indexing improves query performance by creating indexes on frequently queried columns, facilitating faster data retrieval.
- **Code Optimization:** Code optimization techniques, such as code minification, database query optimization, and image compression, enhance application performance by reducing load times and resource consumption.


Presentation
View
Logic
Controller
REST API
Model
Data
Database
3rd party API




System_settings
PK UniqueID
name
email
contact
cover_img
about_content
Alumnus_bio
PK UniqueID
firstName
middleName
lastName
gender
batch
course_id
email
connected_id
avatar
status
date_created
Users
PK UniqueID
Name
Username
Password
Type
Auto_generated_pass
Alumnus_id
Gender
Events_comments
PK UniqueID
event_id
user_id
Forum_comments
PK UniqueID
topic_id
comment
user_id
date_created
Forum_topics
PK UniqueID
title
description
user_id
date_created
Events
PK UniqueID
title
content
schedule
banner
date_created
Courses
PK UniqueID
course
about
Careers
PK UniqueID
company
location
job_title
description
user_id
date_created
Gallery
PK UniqueID

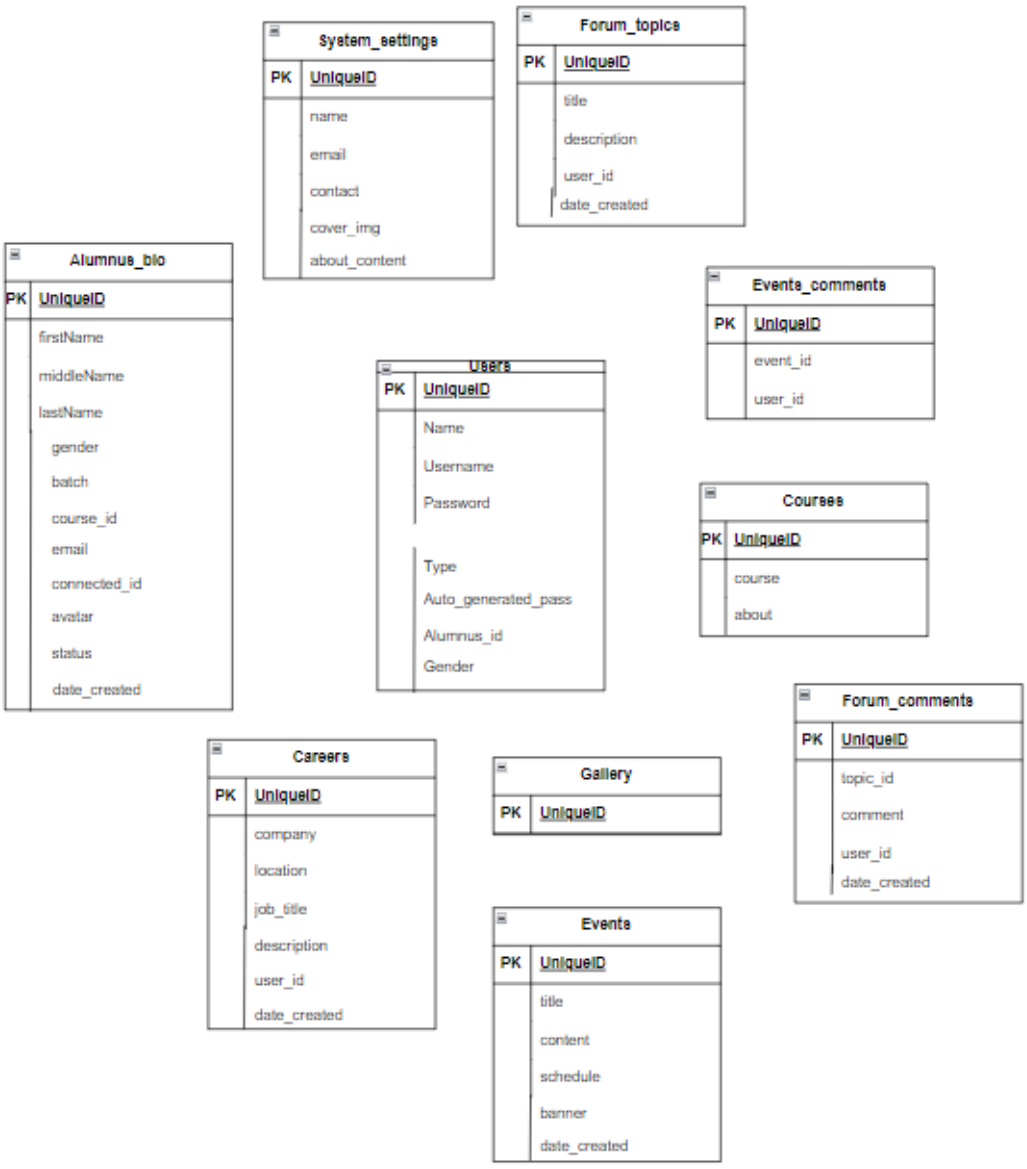
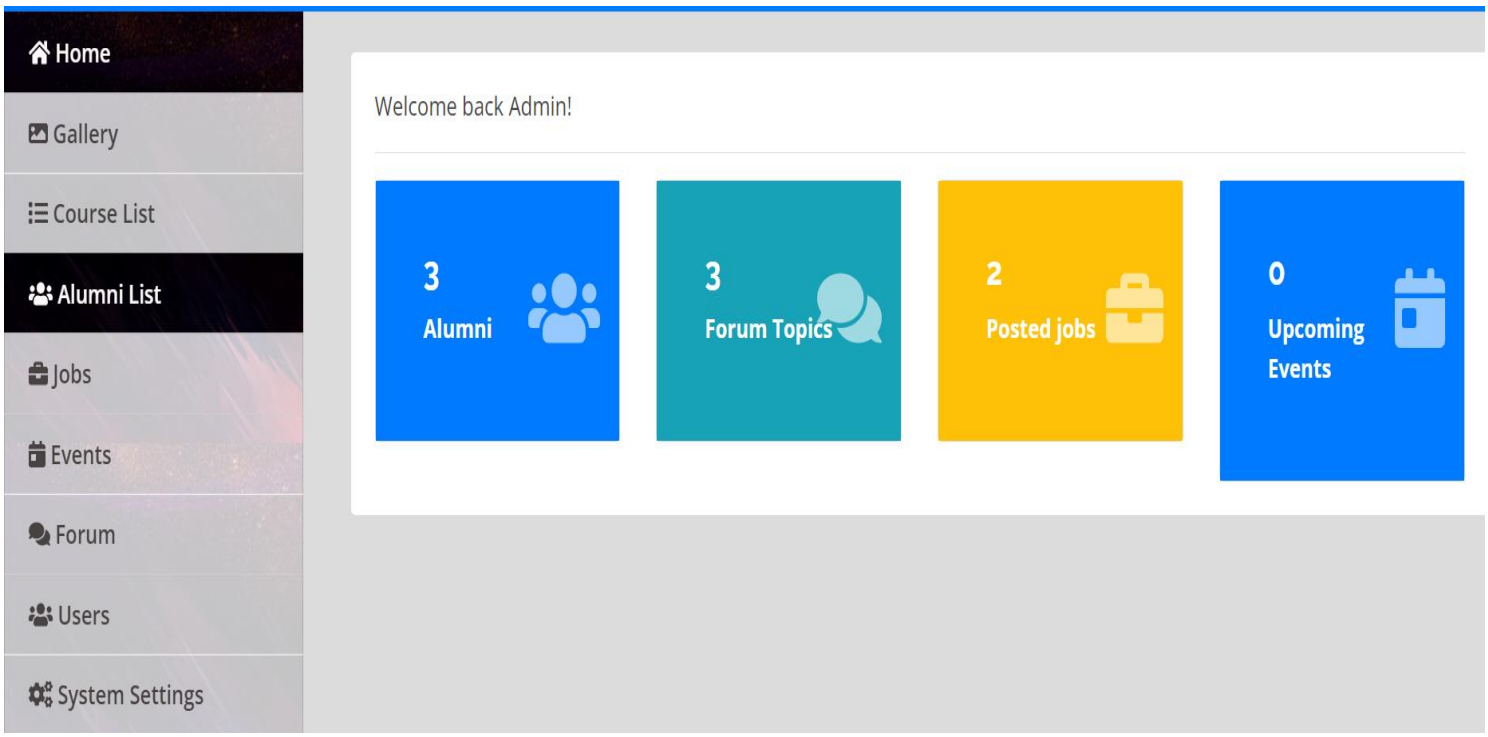

System_settings
PK UniqueID
name
email
contact
cover_img
about_content
Forum_topics
PK UniqueID
title
description
user_id
date_created
Alumnus_bio
PK UniqueID
firstName
middleName
lastName
gender
batch
course_id
email
connected_id
avatar
status
date_created
Events_comments
PK UniqueID
event_id
user_id
Users
PK UniqueID
Name
Username
Password
Type
Auto_generated_pass
Alumnus_id
Gender
Courses
PK UniqueID
course
about
Forum_comments
PK UniqueID
topic_id
comment
user_id
date_created
Careers
PK UniqueID
company
location
job_title
description
user_id
date_created
Gallery
PK UniqueID
Events
PK UniqueID
title
content
schedule
banner
date_created

	Table	Action	Ro
☐	alumnus_bio	Browse Structure Search Insert Empty Drop	
☐	careers	Browse Structure Search Insert Empty Drop	
☐	courses	Browse Structure Search Insert Empty Drop	
☐	events	Browse Structure Search Insert Empty Drop	
☐	event_commits	Browse Structure Search Insert Empty Drop	
☐	forum_comments	Browse Structure Search Insert Empty Drop	
☐	forum_topics	Browse Structure Search Insert Empty Drop	
☐	gallery	Browse Structure Search Insert Empty Drop	
☐	system_settings	Browse Structure Search Insert Empty Drop	
☐	users	Browse Structure Search Insert Empty Drop	
	10 tables	Sum	

Check all With selected:

Console

http://localhost/phpmyadmin/index.php?route=/database/structure&db=alumni_db



6. Implementation

6.1 Coding Standards and Conventions

Adhering to coding standards and conventions ensures consistency, readability, and maintainability of the codebase. The development team follows established coding guidelines, which include:

- **Naming Conventions:** Descriptive and meaningful names for variables, functions, classes, and files following camelCase or snake_case conventions.
- **Formatting Rules:** Consistent indentation, spacing, and line breaks to improve code readability and organization.
- **Comments and Documentation:** Clear and concise comments to explain complex logic, document function parameters and return values, and provide context for future developers.
- **Error Handling:** Proper error handling and exception management to ensure robustness and reliability of the code.

6.2 Version Control System Selection and Setup

The development team utilizes Git as the version control system for managing code changes, collaboration, and versioning. The project repository is hosted on a platform such as GitHub or GitLab, facilitating seamless collaboration and code review among team members.

Key steps in setting up the version control system include:

- **Repository Creation:** Creating a new repository on the chosen platform to host the project codebase.
- **Branching Strategy**: Establishing a branching strategy, such as GitFlow, to manage feature development, bug fixes, and release cycles.
- **Collaboration Workflow:** Defining collaboration workflows, including pull request review processes, code merge policies, and issue tracking.

6.3 Development Environment Configuration

The development environment is configured to replicate the production environment as closely as possible, ensuring consistency and compatibility during code development and testing.

Components of the development environment configuration include:

- **Local Development Setup:** Installing and configuring local development tools such as XAMPP, Visual Studio Code, and phpMyAdmin to facilitate code development, database management, and testing.
- **Dependency Management:** Managing project dependencies using package managers such as Composer for PHP dependencies and npm or Yarn for JavaScript dependencies.
- **Environment Variables:** Using environment variables to manage sensitive information such as database credentials, API keys, and configuration settings.

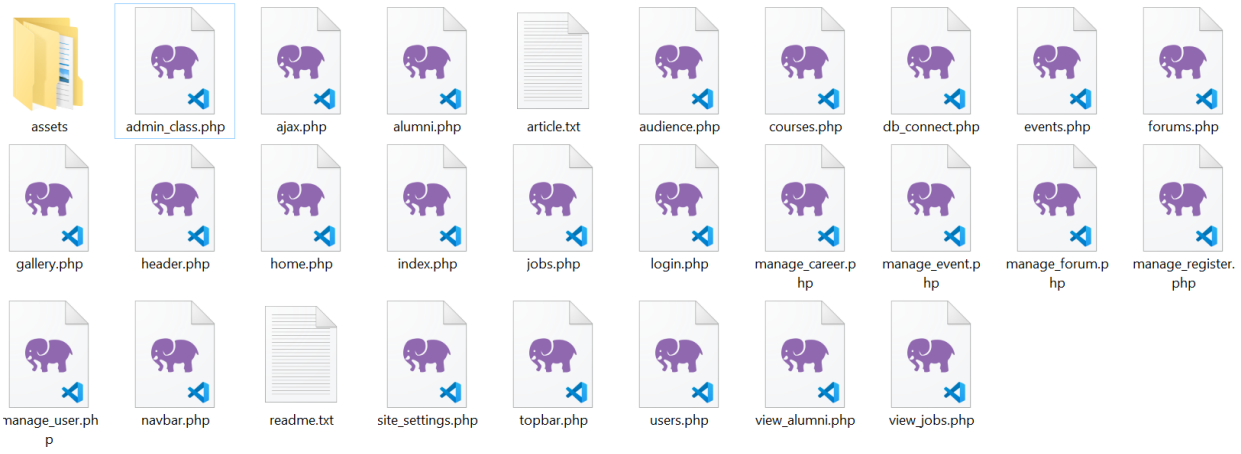
6.4 Code Implementation Details

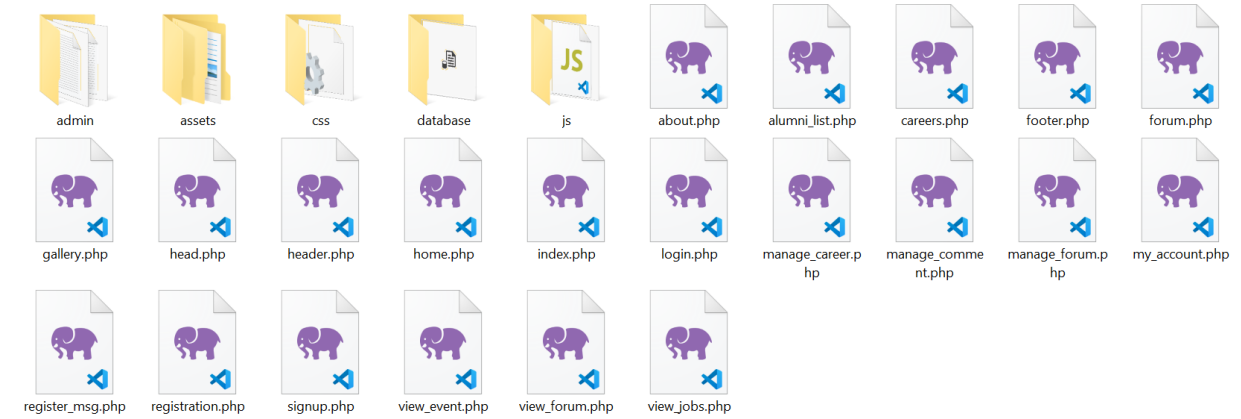
The code implementation phase involves translating system requirements and design specifications into executable code.

Key aspects of code implementation include:

- **Modular Development**: Breaking down the system functionality into smaller, manageable modules or components, each responsible for specific tasks or features.
- **Testing and Debugging:** Writing unit tests and integration tests to validate code functionality and identify and fix bugs and issues.
- **Refactoring and Optimization:** Refactoring code to improve readability, maintainability, and performance, and optimizing critical sections for efficiency and scalability.
- **Documentation:** Documenting code using inline comments, README files, and API documentation to provide insights into code structure, usage, and dependencies.

```php
<?php
session_start();
ini_set('display_errors', 1);
Class Action {
    private $db;

    public function __construct() {
        ob_start();
    include 'db_connect.php';

    $this->db = $conn;
    }
    function __destruct() {
        $this->db->close();
        ob_end_flush();
    }

    function login(){

            extract($_POST);
            $qry = $this->db->query("SELECT * FROM users where username = '".$username."' and password = '".md5($password)."' ");
            if($qry->num_rows > 0){
                foreach ($qry->fetch_array() as $key => $value) {
                    if($key != 'passwords' && !is_numeric($key))
```





7. Testing and Quality Assurance

7.1 Test Plan and Strategy

A comprehensive test plan and strategy are essential for ensuring the quality, reliability, and functionality of the Alumni Management System. The test plan outlines the scope, objectives, methodologies, and resources for various testing activities throughout the development lifecycle.

Key components of the test plan and strategy include:

- **Testing Objectives:** Clearly defined goals and objectives for each testing phase, aligned with project requirements and stakeholder expectations.
- **Testing Methodologies:** Selection of appropriate testing methodologies, such as black-box testing, white-box testing, and exploratory testing, based on the nature of the system and the level of detail required.
- **Test Coverage:** Identification of test scenarios, use cases, and acceptance criteria to ensure comprehensive coverage of system functionalities and user interactions.
- **Testing Tools:** Utilization of testing tools and frameworks, such as PHPUnit for PHP unit testing, Selenium for automated browser testing, and Postman for API testing, to streamline testing processes and enhance efficiency.

7.2 Unit Testing

Unit testing focuses on validating individual components or units of code in isolation to ensure their correctness and functionality. Unit tests are typically written and executed by developers during the coding phase to detect and fix defects early in the development process.

Key aspects of unit testing include:

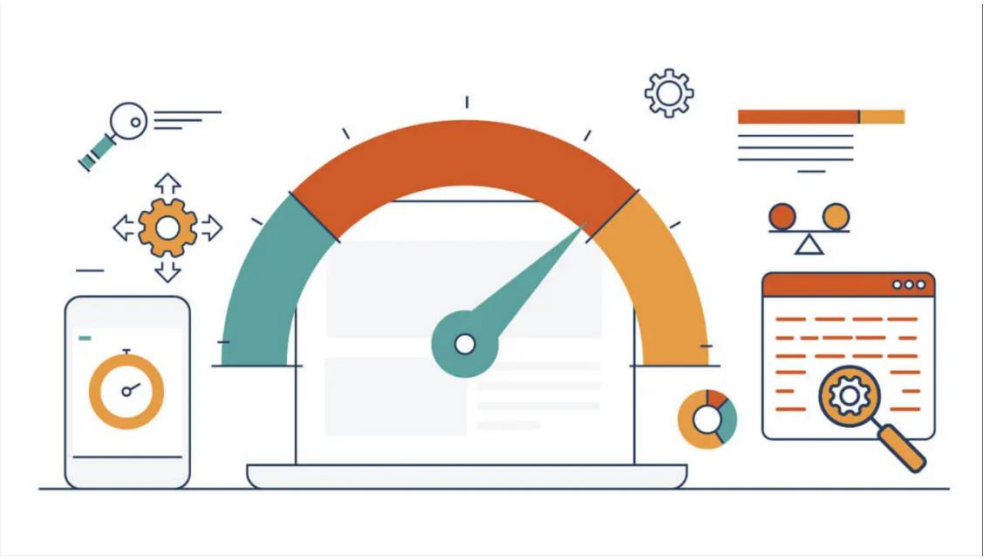
- **Test Case Development:** Writing test cases to verify the behavior and functionality of individual functions, methods, and classes.
- **Test Automation**: Automating unit tests using testing frameworks such as PHPUnit for PHP, Jest for JavaScript, and JUnit for Java to facilitate continuous integration and regression testing.
- **Mocking and Stubbing:** Using mocking and stubbing techniques to simulate dependencies and isolate units under test, enabling thorough and efficient testing.

7.3 Integration Testing

Integration testing verifies the interactions and interoperability between different components or modules of the system. Integration tests are performed to ensure that integrated units function correctly together as a cohesive whole.

Key aspects of integration testing include:

- **Integration Scenarios:** Identifying integration scenarios and defining test cases to validate data flows, communication protocols, and interface contracts between system components.
- **End-to-End Testing:** Conducting end-to-end integration tests to simulate real-world user interactions and validate system behavior across multiple layers and components.
- **Dependency Management:** Managing dependencies and external services using mocking, stubbing, or service virtualization techniques to isolate integration tests and maintain test environment stability.



7.4 System Testing

System testing evaluates the entire system as a whole to validate its compliance with functional and non-functional requirements. System tests are conducted to assess system behavior, performance, security, and usability in a simulated production environment.

Key aspects of system testing include:

- **Functional Testing:** Executing test scenarios and use cases to verify that the system functions as expected and meets specified requirements.
- **Performance Testing:** Assessing system performance under different load conditions, such as stress testing, load testing, and scalability testing, to ensure optimal performance and responsiveness.
- **Security Testing:** Identifying and mitigating security vulnerabilities through techniques such as penetration testing, vulnerability scanning, and code analysis to safeguard sensitive data and prevent unauthorized access.

7.5 User Acceptance Testing

User Acceptance Testing (UAT) involves validating the system against user requirements and expectations to ensure its suitability for deployment and use in the production environment. UAT is typically conducted by end-users or stakeholders to assess the system's usability, functionality, and alignment with business needs.

Key aspects of user acceptance testing include:

- **Test Scenario Definition:** Defining test scenarios and acceptance criteria based on user stories, use cases, and real-world scenarios to guide testing activities.
- **User Involvement:** Involving end-users and stakeholders in the testing process to gather feedback, address usability issues, and prioritize enhancements.

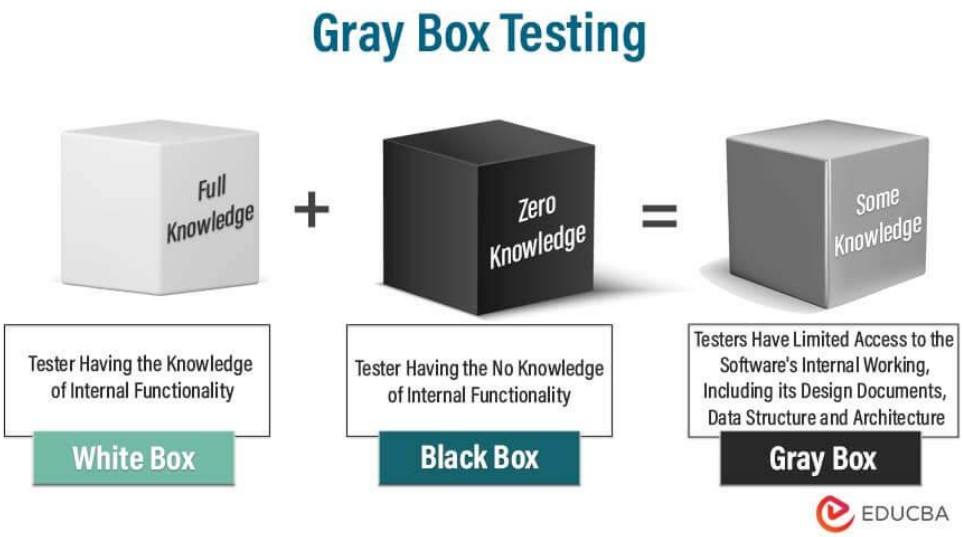
- **Acceptance Criteria Verification:** Verifying that the system meets predefined acceptance criteria and user expectations, addressing any discrepancies or deficiencies identified during testing.

7.6 Bug Tracking and Resolution

Bug tracking and resolution involve identifying, documenting, prioritizing, and resolving defects and issues identified during testing. A robust bug tracking process ensures that defects are effectively managed and addressed to maintain the quality and stability of the system.

Key aspects of bug tracking and resolution include:

- **Issue Logging**: Logging defects and issues in a centralized bug tracking system, including details such as severity, priority, steps to reproduce, and environment information.
- **Prioritization:** Prioritizing bug fixes based on severity, impact on system functionality, and business priorities to allocate resources effectively.
- **Resolution and Verification**: Implementing fixes for identified issues and verifying their resolution through retesting and validation to ensure that fixes are effective and do not introduce new defects.



Output | Performance | VLD | References | Branches

Here you find the average performance (time & memory) of each version. A grayed out version indicates it didn't complete successfully (based on exit-code).

Version	System time (s)	User time (s)	Memory (MiB)
+ 8.3	0.020	0.008	25.92
+ 8.2	0.034	0.009	25.92
+ 8.1	0.032	0.009	25.92

1553.89 ms | **1400** KiB | **21** Q

preferences: ☐ dark mode ☑ live preview

Untitled @ 2024-04-30 15:32:15

```php
<?php
session_start();
ini_set('display_errors', 1);
Class Action {
    private $db;

    public function __construct() {
        ob_start();
    include 'db_connect.php';

    $this->db = $conn;
    }
    function __destruct() {
        $this->db->close();
        ob_end_flush();
    }

    function login(){

            extract($_POST);
            $qry = $this->db->query("SELECT * FROM users where username = '".$username."' and password = '".md5($password)."'
                ");
            if($qry->num_rows > 0){
                foreach ($qry->fetch_array() as $key => $value) {
                    if($key != 'passwors' && !is_numeric($key))
                        $_SESSION['login_'.$key] = $value;
                }
                if($_SESSION['login_type'] != 1){
                    foreach ($_SESSION as $key => $value) {
```

8. Project Management

8.1 Team Organization and Roles

The project team is organized to ensure effective collaboration, communication, and coordination throughout the development lifecycle. Key roles and responsibilities within the team include:

- **Project Manager:** Responsible for overall project planning, coordination, and execution. Manages project timelines, resources, and deliverables, and facilitates communication among team members.
- **Developers**: Responsible for coding, testing, and implementing system functionalities according to project requirements and design specifications. Collaborate with other team members to integrate and maintain codebase.
- **Quality Assurance/Testers:** Responsible for planning, executing, and documenting testing activities to ensure the quality and reliability of the system. Identify and report defects and issues for resolution.
- **UI/UX Designers:** Responsible for designing and implementing the user interface and user experience of the system. Create wireframes, mockups, and prototypes to guide development efforts.
- **Database Administrators:** Responsible for designing, implementing, and maintaining the database schema and ensuring data integrity, security, and performance.

8.2 Communication Plan

For communication channels, the project team utilizes GitHub for code collaboration and version control, as well as a WhatsApp group for quick updates, discussions, and announcements. Additionally, regular group meetings held in class serve as face-to-face communication opportunities for more in-depth discussions and project updates.

8.3 Risk Management

One potential risk identified was the challenge of effective communication due to team members not knowing each other well initially. However, this risk was mitigated through the establishment of communication channels such as GitHub and WhatsApp, as well as regular group meetings. These platforms facilitated open communication, collaboration, and relationship-building among team members, reducing the impact of this risk on project progress. Additionally, GitHub was utilized for project tracking, allowing changes to be tracked and managed effectively throughout the development process.

8.4 Project Tracking and Reporting

Project tracking and reporting involve monitoring project progress, tracking key performance indicators (KPIs), and generating regular reports to stakeholders and project sponsors. Effective project tracking and reporting ensure transparency, accountability, and informed decision-making.

Key elements of project tracking and reporting include:

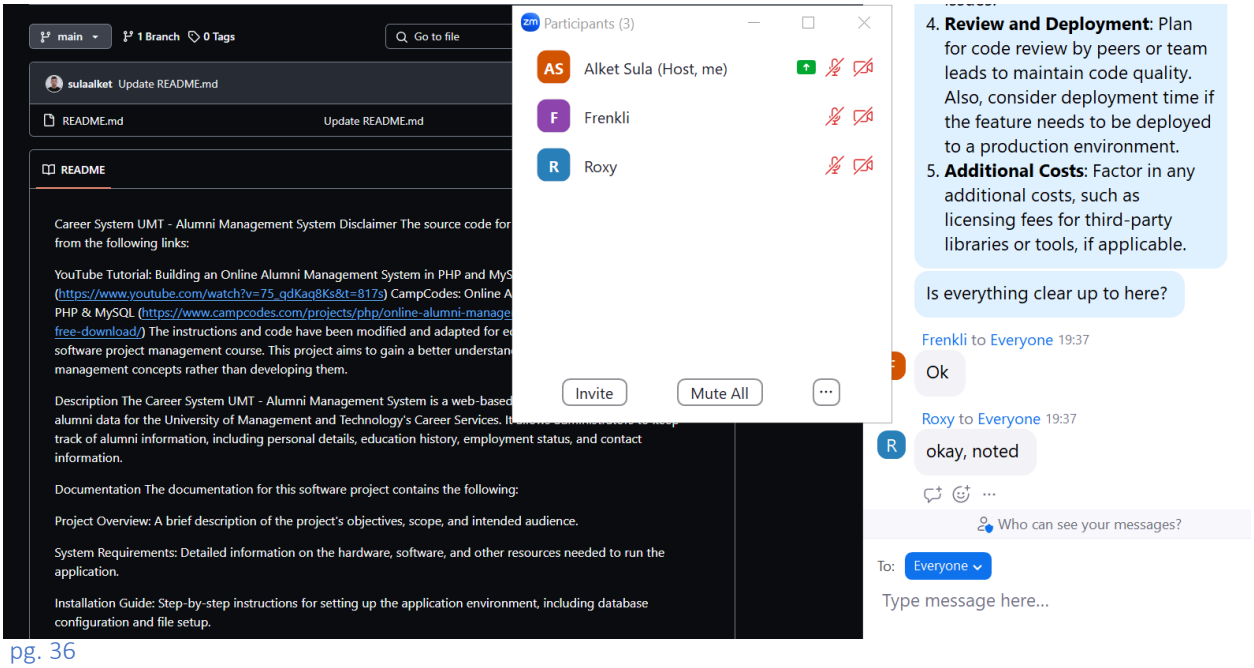
- **Task Tracking:** Monitoring and updating task progress, milestones, and dependencies using project management tools such as Jira, Trello, or Asana.
- **Progress Reporting:** Generating regular progress reports, status updates, and dashboards to communicate project status, achievements, issues, and risks to stakeholders.
- **KPI Measurement:** Tracking and measuring project KPIs, such as budget variance, schedule variance, resource utilization, and quality metrics, to assess project performance and identify areas for improvement.

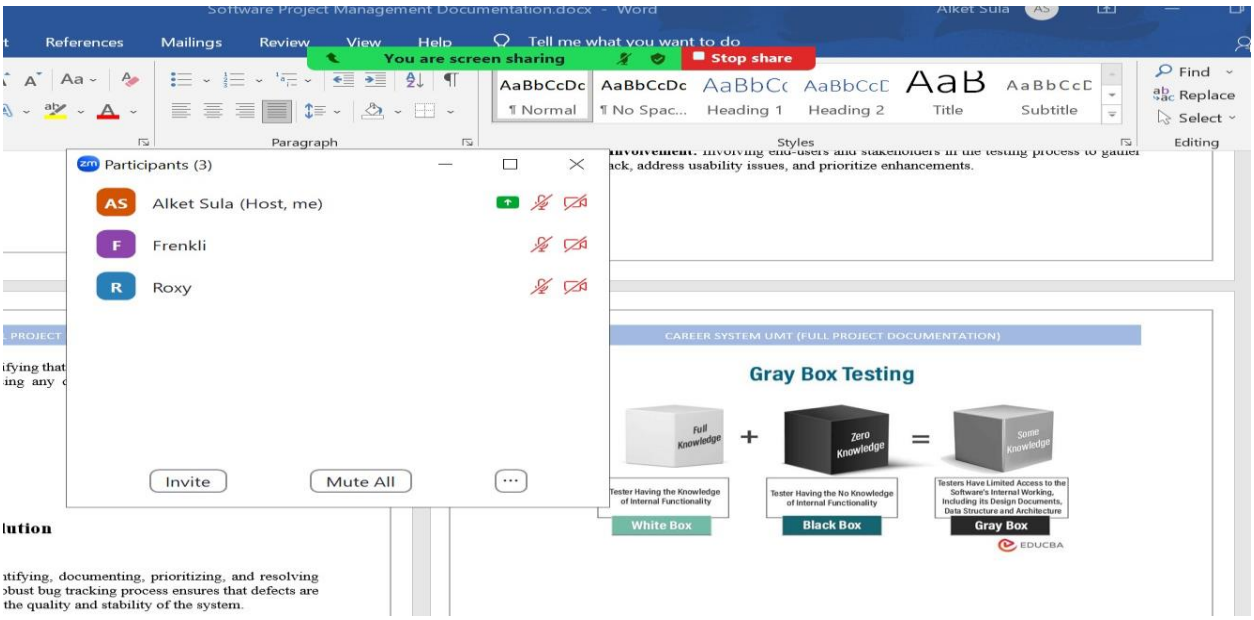
8.5 Change Management

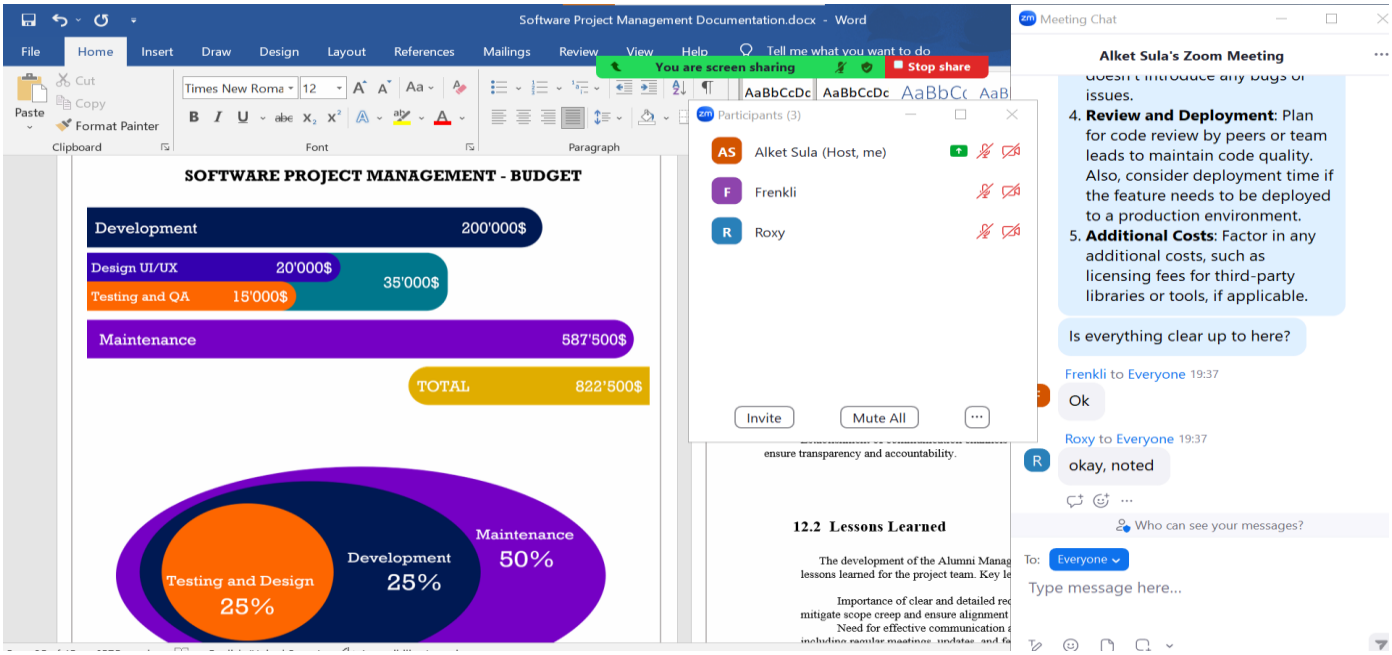
Change management involves managing changes to project scope, requirements, timelines, and resources to minimize disruptions and maintain project alignment with business goals and objectives. A structured change management process helps ensure that changes are properly evaluated, approved, and implemented.

Key components of change management include:

- **Change Request Process**: Establishment of a formal process for submitting, reviewing, and approving change requests. Change requests may originate from stakeholders, team members, or external factors.
- **Impact Assessment**: Evaluation of the impact of proposed changes on project scope, schedule, budget, and resources. Assessments help stakeholders make informed decisions about change prioritization and approval.
- **Change Control Board:** Formation of a change control board comprising project stakeholders and decision-makers to review and approve change requests based on predefined criteria and guidelines.
- **Communication and Documentation:** Communication of approved changes to relevant stakeholders and documentation of changes in project documentation, such as the project charter, requirements specifications, and change log.




Software Project Management Documentation.docx - Word
You are screen sharing
Stop share
Participants (3)
Alket Sula (Host, me)
Frenkli
Roxy
Invite
Mute All
Gray Box Testing
Full Knowledge
Zero Knowledge
Some Knowledge
White Box
Black Box
Gray Box
EDUCBA


Software Project Management Documentation.docx - Word
You are screen sharing
Stop share
SOFTWARE PROJECT MANAGEMENT - BUDGET
Development 200'000$
Design UI/UX 20'000$
Testing and QA 15'000$
35'000$
Maintenance 587'500$
TOTAL 822'500$
Maintenance 50%
Development 25%
Testing and Design 25%
Participants (3)
Alket Sula (Host, me)
Frenkli
Roxy
Invite
Mute All
12.2 Lessons Learned
Meeting Chat
Alket Sula's Zoom Meeting
Is everything clear up to here?
Frenkli to Everyone 19:37
Ok
Roxy to Everyone 19:37
okay, noted
Who can see your messages?
To: Everyone
Type message here...

Full Name	Duty 1	Duty 2	Duty 3
Alket Sula	UML designs	full documentation	full stack coding (html/css/js, framework angular or react js, vue.js, .net , noide.js, express.js, python or java) to be decided, database sql microsoft server),
Roksana Ndreca	user log in interface coding	UI/UX design of the system	non functional requirements
Arbiona Mjeshtri	users log in interface design		non functional requirements
Frenkli Koleci	users log in interface design	functional requirements	
Enkeled Selami	user log in interface coding	. functional requirements	
Miki Bode	UML designs(sequence Diagrams)	UI/UX design of the system	

9. Deployment

9.1 Deployment Strategy

The deployment strategy for the Alumni Management System involves a phased approach to ensure a smooth and seamless transition from development to production environments.

Key components of the deployment strategy include:

- **Staging Environment:** Setting up a staging environment to test the system in a production-like environment before deployment. This allows for thorough testing of system functionality, performance, and compatibility.
- **Incremental Deployment:** Deploying system updates and enhancements in incremental phases to minimize disruptions and mitigate risks. This approach allows for gradual implementation of changes while monitoring system stability and user feedback.
- **Rollback Plan**: Establishing a rollback plan to revert to previous system versions or configurations in case of deployment failures or unforeseen issues. This ensures that the system can be quickly restored to a working state with minimal downtime.

9.2 Server Configuration

The server configuration for hosting the Alumni Management System involves setting up and configuring server infrastructure to support system requirements and accommodate anticipated user traffic.

Key server configuration tasks include:

- **Hardware Provisioning**: Selecting and provisioning appropriate server hardware, such as CPU, memory, and storage, to meet performance and scalability requirements.
- **Operating System Setup:** Installing and configuring the operating system, such as Linux or Windows Server, and applying necessary security updates and patches.

- **Web Server Configuration**: Configuring web server software, such as Apache or Nginx, to serve web pages and handle HTTP requests efficiently.
- **Database Setup**: Installing and configuring the database management system (DBMS), such as MySQL, and optimizing database settings for performance, security, and reliability.

9.3 Data Migration Plan

The data migration plan outlines the process for transferring data from the development or staging environment to the production environment while minimizing downtime and ensuring data integrity.

Key steps of the data migration plan include:

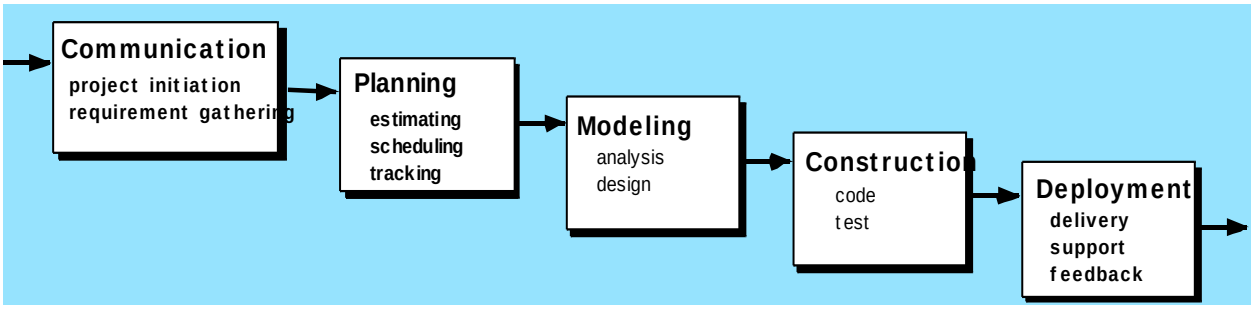
- **Data Backup:** Backing up the database and application files from the development or staging environment to ensure data preservation and disaster recovery capabilities.
- **Data Transfer:** Transferring database backups and application files to the production server using secure transfer protocols such as SSH or SFTP.
- **Database Schema Migration:** Applying database schema changes and updates to the production database, ensuring compatibility with the latest version of the application.
- **Data Verification**: Performing data validation and integrity checks to ensure that data is successfully migrated and consistent between environments.

9.4 User Training and Documentation

User training and documentation are essential for ensuring that end-users understand how to use the Alumni Management System effectively and maximize its benefits.

Key components of user training and documentation include:

- **User Manuals**: Creating user manuals and guides that provide step-by-step instructions on how to navigate the system, perform common tasks, and troubleshoot issues.
- **Training Sessions:** Conducting training sessions or workshops to familiarize users with system features, functionalities, and best practices. Training sessions may be conducted in-person or remotely via video conferencing.
- **Online Help Resources**: Providing online help resources such as FAQs, knowledge bases, and video tutorials to support self-service learning and troubleshooting.
- **User Support Channels:** Establishing user support channels, such as email, helpdesk, or online forums, where users can seek assistance, report issues, and provide feedback.



10. Maintenance and Support

10.1 Ongoing Maintenance Plan

The ongoing maintenance plan ensures the continuous functionality, security, and performance of the Alumni Management System post-deployment.

Key elements of the maintenance plan include:

- **Regular Updates**: Implementing regular updates and patches to address security vulnerabilities, bug fixes, and performance improvements.
- **Backup and Disaster Recovery:** Conducting regular backups of the system data and configurations to prevent data loss and ensure business continuity in case of system failures or disasters.
- **System Monitoring:** Monitoring system health, availability, and performance metrics to identify and address potential issues proactively.
- **User Support:** Providing ongoing user support and assistance through helpdesk services, online forums, and documentation resources to address user inquiries and issues.

10.2 Issue Tracking and Resolution

Issue tracking and resolution involve identifying, prioritizing, and resolving software defects, errors, and user-reported issues to maintain system reliability and usability.

Key components of issue tracking and resolution include:

- **Issue Identification:** Monitoring and triaging incoming bug reports, error logs, and user feedback to identify and categorize issues based on severity and impact.
- **Issue Prioritization**: Prioritizing issue resolution based on factors such as severity, impact on system functionality, and business priorities to allocate resources effectively.

- **Issue Resolution:** Investigating reported issues, identifying root causes, and implementing fixes or workarounds to resolve them. Collaboration among development, QA, and support teams may be necessary to address complex issues.
- **Quality Assurance**: Conducting regression testing and validation to ensure that resolved issues are effectively addressed and do not introduce new defects or regressions.

10.3 Performance Monitoring and Optimization

Performance monitoring and optimization involve continuously monitoring system performance metrics and identifying opportunities to improve system efficiency, responsiveness, and scalability.

Key aspects of performance monitoring and optimization include:

- **Performance Metrics:** Monitoring key performance indicators (KPIs) such as response time, throughput, resource utilization, and error rates to assess system performance and identify bottlenecks.
- **Performance Testing**: Conducting load testing, stress testing, and scalability testing to evaluate system performance under different conditions and workloads.
- **Optimization Strategies**: Implementing performance optimization strategies such as code refactoring, caching mechanisms, database indexing, and resource scaling to improve system responsiveness and scalability.
- **Capacity Planning:** Forecasting future resource requirements based on projected growth and usage patterns to ensure that the system can scale effectively to meet increasing demands.

11. Project Budget

Development Cost:

- Estimated to be around $200,000 based on hourly rates for development efforts.
- Considered factors such as team composition, technology stack, and project complexity.

Design and Testing Costs:

- Design costs estimated at $20,000 based on UI/UX design efforts.
- Testing costs estimated at $15,000 based on QA and testing efforts.
- Assumed design and testing efforts as a percentage of development effort.

Maintenance Cost over 5 Years:

- Estimated annual maintenance cost ranged from $58,750 to $117,500 (assuming 25% to 50% of development cost per year).
- Total maintenance cost over 5 years estimated to be approximately $293,750 based on a 25% maintenance percentage.

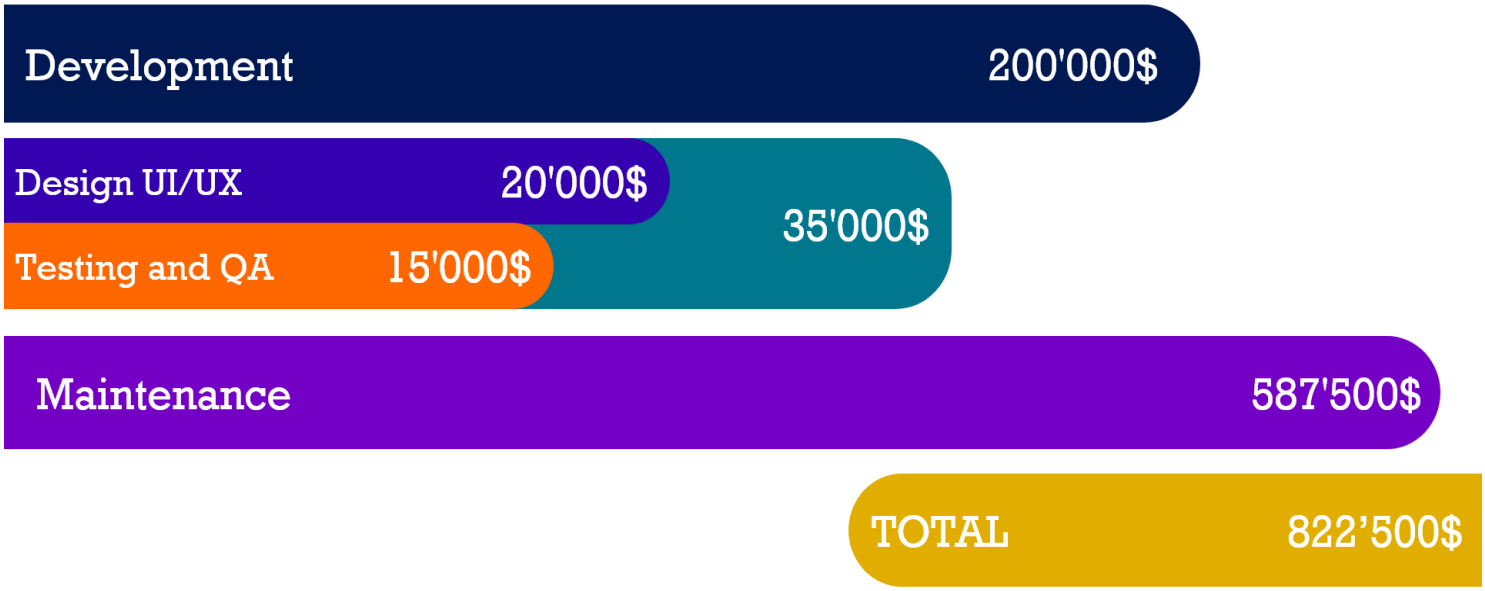
Total Project Budget:

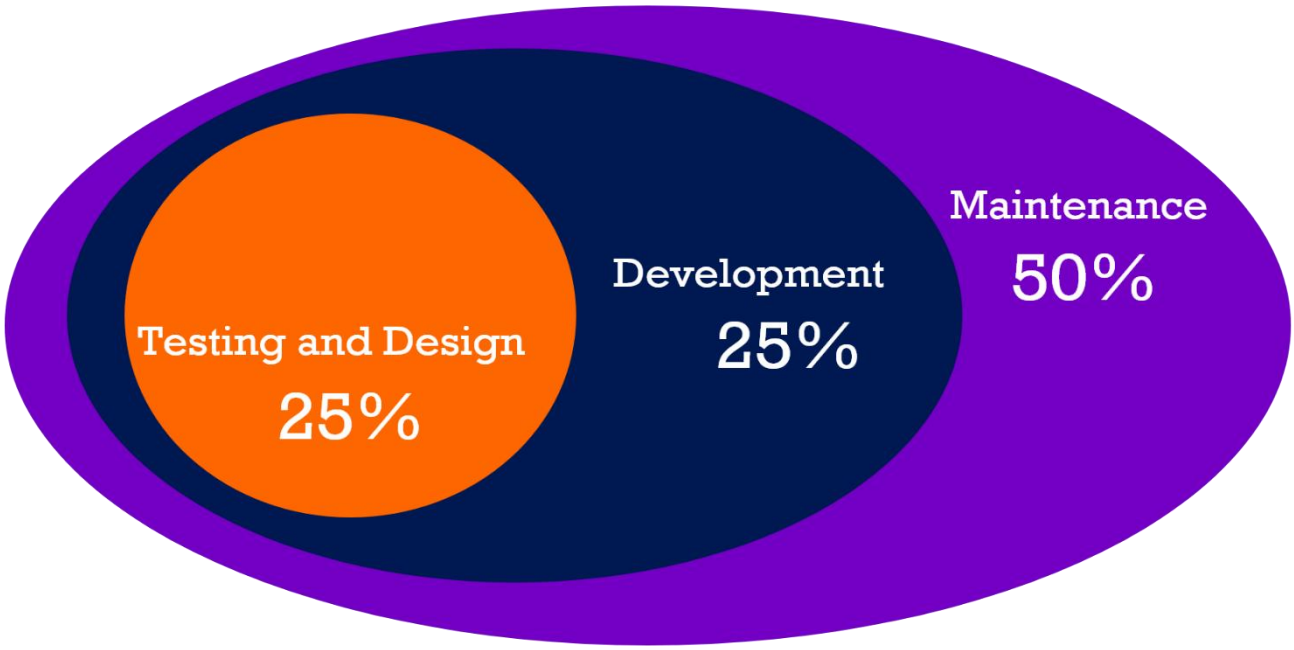
Total estimated cost for development, design, testing, and 5-year maintenance summed up.

Approximated to be around $528'750 to $822'500, depending on the maintenance percentage chosen.

In summary, the total project budget for the Alumni Management System, including development, design, testing, and 5-year maintenance, is estimated to range from approximately ***$528'750 to $822'500***. These estimates are based on various factors such as team composition, hourly rates, and maintenance percentage assumptions.

SOFTWARE PROJECT MANAGEMENT - BUDGET





12. Conclusions

12.1 Summary of Achievements

The development of the Alumni Management System represents a significant achievement for the project team and stakeholders. Key accomplishments include:

Successful implementation of core functionalities such as user authentication, profile management, job listings, events, and forums.

Adherence to software project management methodologies and best practices throughout the development lifecycle.

Collaboration among team members to overcome challenges, meet deadlines, and deliver a functional and user-friendly system.

Establishment of communication channels and effective project tracking mechanisms to ensure transparency and accountability.

12.2 Lessons Learned

The development of the Alumni Management System provided valuable insights and lessons learned for the project team. Key lessons include:

Importance of clear and detailed requirements gathering and documentation to mitigate scope creep and ensure alignment with stakeholder expectations.

Need for effective communication and collaboration among team members, including regular meetings, updates, and feedback sessions.

Value of robust testing and quality assurance processes to identify and address issues early in the development lifecycle, reducing rework and improving overall system reliability.

Recognition of the impact of external factors such as team dynamics, resource constraints, and technology limitations on project outcomes.

12.3 Future Enhancements

While the Alumni Management System has achieved its primary objectives, there are opportunities for future enhancements and improvements. Potential areas for enhancement include:

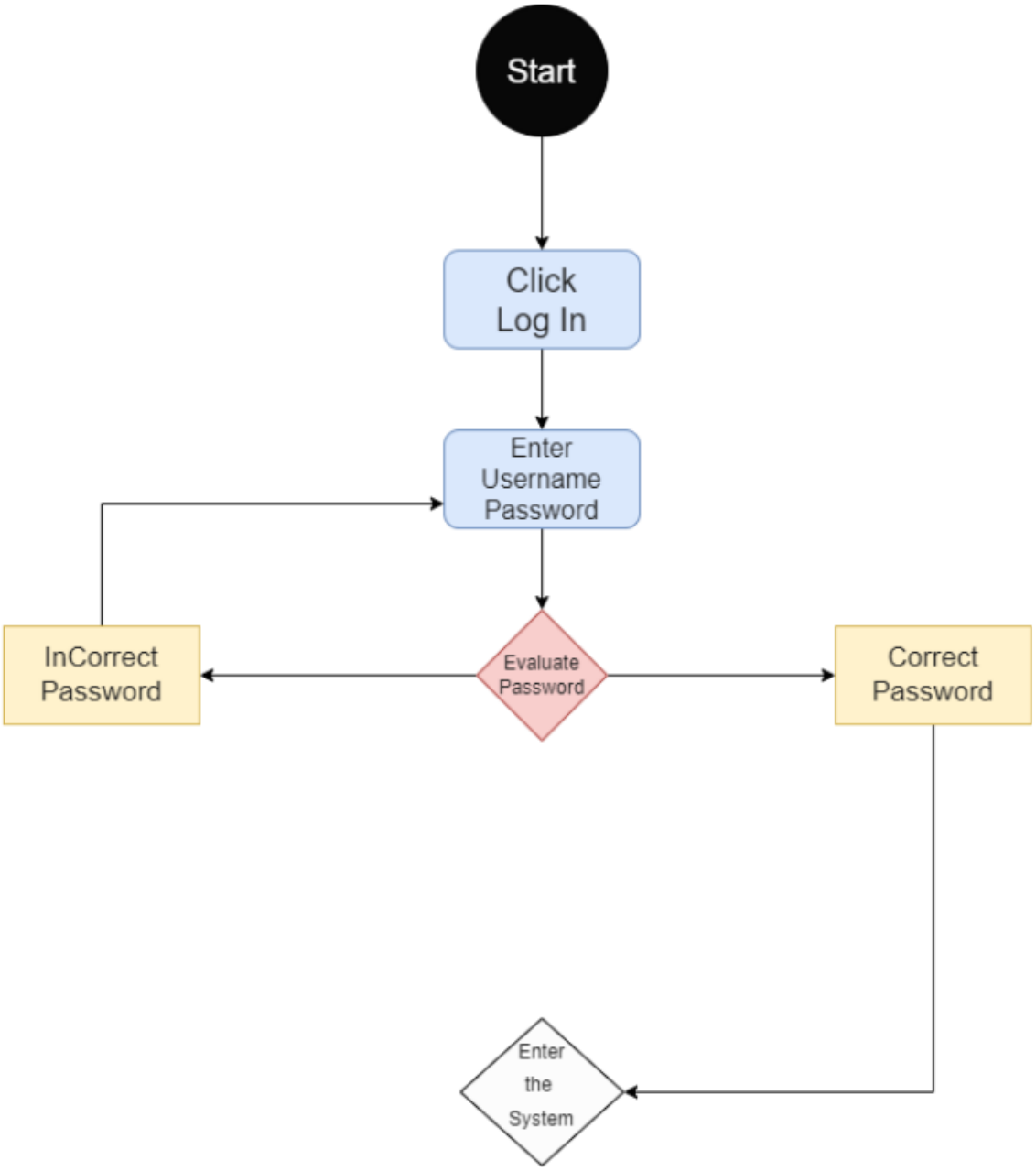
Integration with additional features such as alumni directories, mentorship programs, and fundraising initiatives to enhance user engagement and alumni networking opportunities.

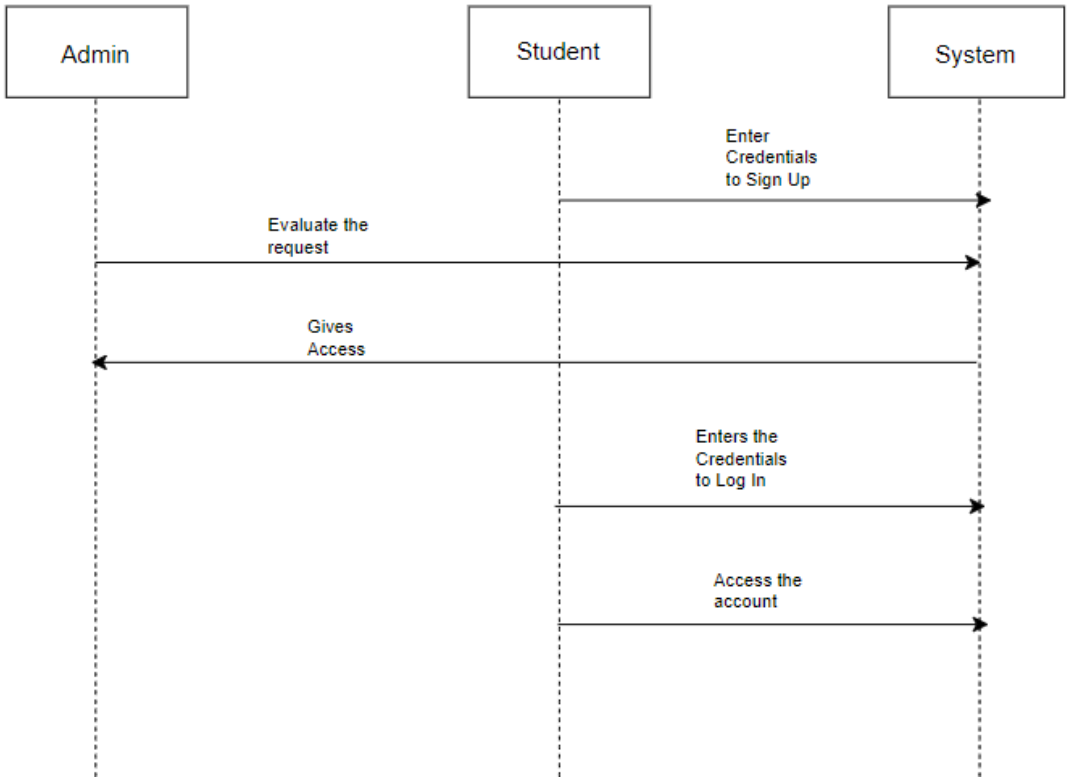
Implementation of advanced analytics and reporting capabilities to provide insights into alumni demographics, engagement metrics, and career pathways.

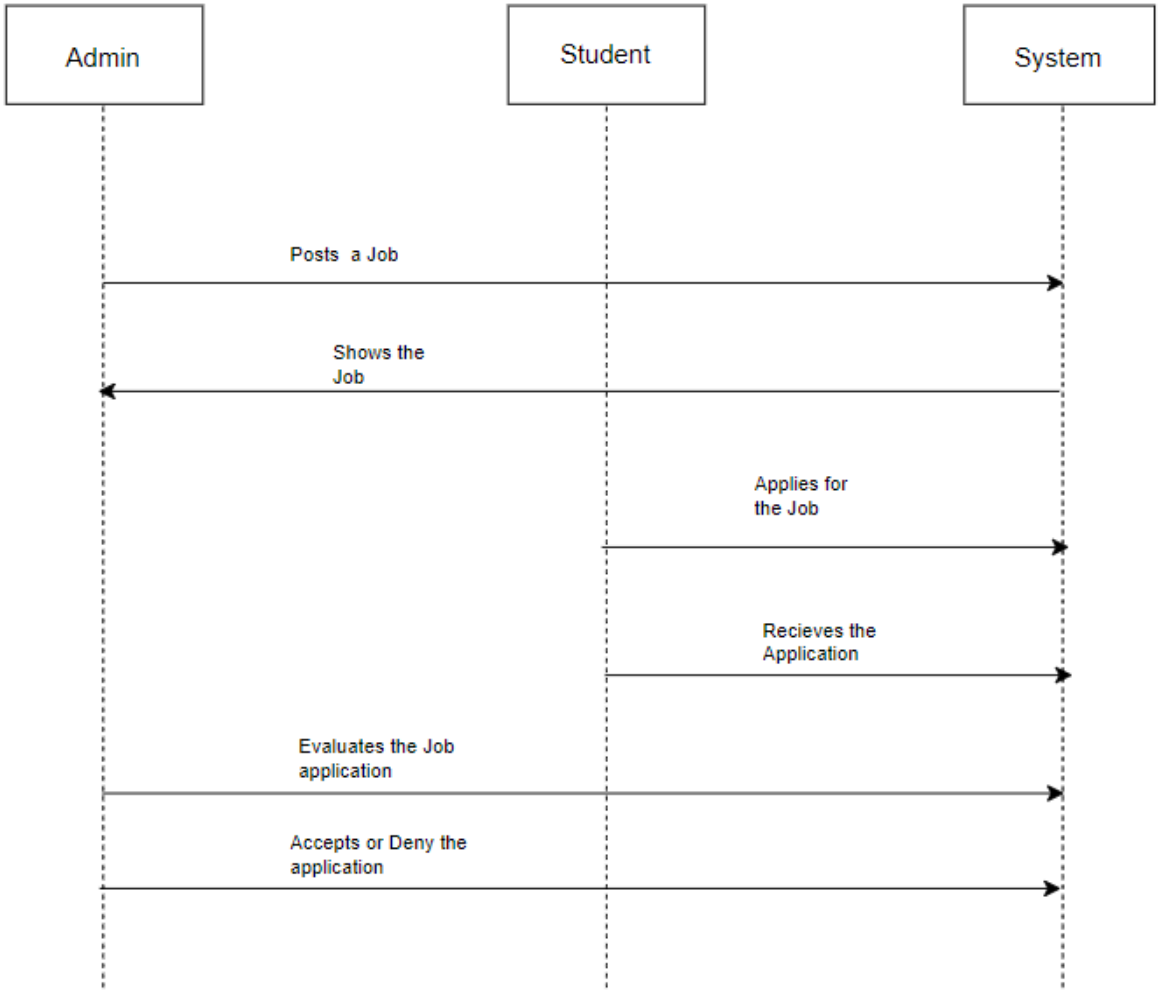
Enhancement of user experience through responsive design, accessibility improvements, and personalized content recommendations.

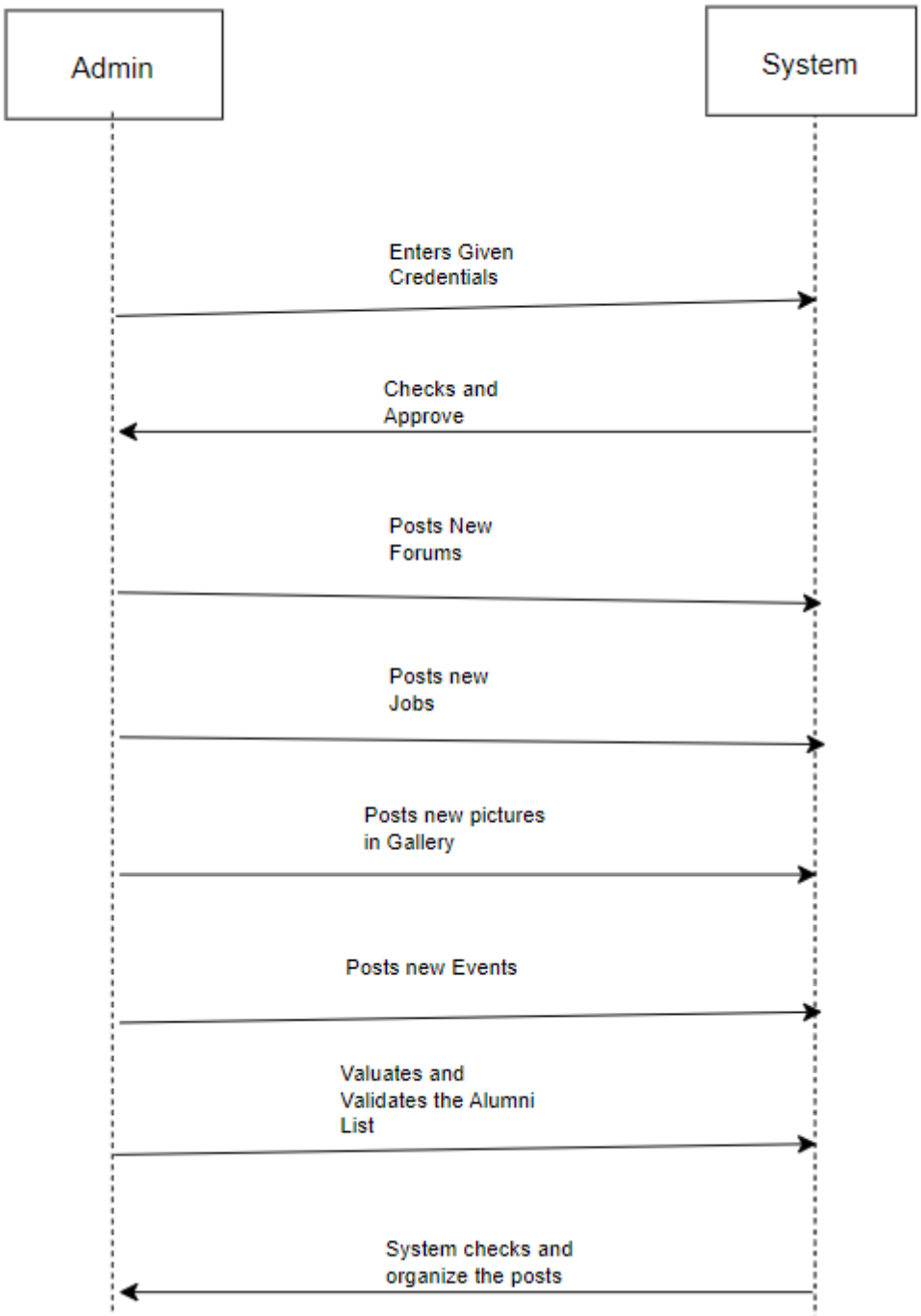
Exploration of emerging technologies such as artificial intelligence (AI), machine learning (ML), and blockchain to enhance system functionality and security.

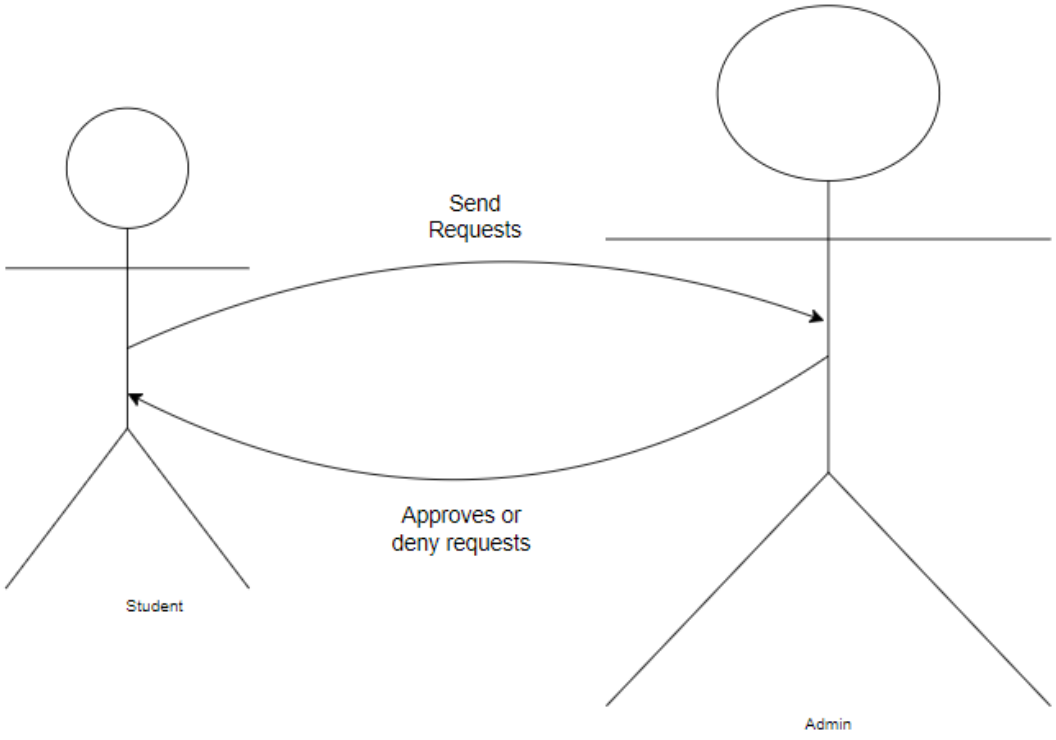
13. Diagrams


Start
Click Log In
Enter Username Password
InCorrect Password
Evaluate Password
Correct Password
Enter the System


Admin
Student
System
Enter Credentials to Sign Up
Evaluate the request
Gives Access
Enters the Credentials to Log In
Access the account


Admin
Student
System
Posts a Job
Shows the Job
Applies for the Job
Recieves the Application
Evaluates the Job application
Accepts or Deny the application


Admin
System
Enters Given Credentials
Checks and Approve
Posts New Forums
Posts new Jobs
Posts new pictures in Gallery
Posts new Events
Valuates and Validates the Alumni List
System checks and organize the posts


Send
Requests
Approves or
deny requests
Student
Admin

14. References

1. BlackBox AI. (n.d.). Retrieved from https://www.blackbox.ai/
2. ChatGPT by OpenAI. (n.d.). Retrieved from https://chat.openai.com/
3. CampCodes. (n.d.). Online Alumni Management System using PHP MySQL. Retrieved from https://www.campcodes.com/projects/php/online-alumni-management-system-using-php-mysql-free-download/
4. YouTube - Software Project Management Tutorial. (n.d.). Retrieved from https://www.youtube.com/watch?v=75_qdKaq8Ks&t=817s
5. Hakrama, I. (n.d.). Lecture Slides of Software Project Management / Web Programming and Software Analysis and Design.
6. Ali, M. (n.d.). Lecture Slides for Software Testing and Verification.
7. Domnori, E. (n.d.). Lecture Slides of Database Management Systems.
8. Sommerville, I. (2020). Software Engineering (10th ed.). Pearson.
9. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.
10. IEEE Computer Society. (n.d.). IEEE Software Engineering Standards. Retrieved from https://www.computer.org/standards/software-engineering

Thank You for Reading This Documentation.