CAMPUS AMBASSADOR PORTAL

A Project Report
submitted in partial fulfillment of the requirements
For the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

CHIRAG KAMRA, 2101610100073
DISHA,2101610100082

Under the Esteemed Guidance of

Mr. PRASHANT NARESH

Krishna Engineering College, Ghaziabad- 201007

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our guide and staff **Mr. Prashant Naresh, Asst Prof. Department of CSE**, for his vital support, valuable guidance and for providing us with all facility and guidance for presenting assisting us in times of need.

We would also take this opportunity to express our heartfelt gratitude to **Dr. Sachin Malhotra, Head of the Department of Computer Science**, for his valuable support and cooperation in the presentation of this paper.

We are thankful to our friends for their lively discussion and suggestions. Finally, we would like to thank the almighty who has given us all that is required for the successful completion of my seminar.

Sincerely
CHIRAG KAMRA

(Roll no: 2101610100073)

DISHA

(Roll no: 2101610100082)

Date: **01/12/2023**

Place: **Ghaziabad**

CERTIFICATE

Certified that seminar work entitled "**Campus Ambassador Portal"** is a Bonafide work carried out in the V semester by "**Chirag Kamra & Disha"** in partial fulfillment for the award of Bachelor of Computer Science and Engineering from Krishna Engineering College Ghaziabad during the academic year 2023- 2024.

SIGNATURE

STUDENT's NAME:

CHIRAG KAMRA

DISHA

ROLL NO-

2101610100073

2101610100082

MENTOR SIGNATURE

Table of Contents

Introduction

- Introduction to Project: The Campus Ambassador Portal project signifies a transformative

approach to managing campus ambassador programs. It introduces a dynamic web portal designed to streamline collaboration among educational institutions, companies, and student representatives. By enhancing communication, simplifying task management, and fostering a cohesive community, this project addresses the limitations of current systems, promising a more efficient and engaging experience for all stakeholders.

- Project Category: This project falls within the category of an internet-based system, specifically focusing on the development of a responsive web portal using modern technologies.
- Objectives: The primary objectives include optimizing program efficiency, fostering seamless collaboration, and introducing unique features that elevate the overall campus ambassador experience.
- Existing System: Current systems lack efficient communication channels and task management capabilities. The proposed Campus Ambassador Portal aims to overcome these limitations by offering an intuitive and feature-rich platform.
- Proposed System: The proposed system developed using Node.js and Express.js for the backend, and EJS, JavaScript, and CSS for the frontend, promises to revolutionize campus ambassador program management with enhanced communication and task assignment tools.
- Unique Features of the System: Unique features include real-time task assignment, performance tracking, and a collaborative environment that bridges the gap between educational institutions, companies, and student representatives.

Requirement Analysis and System Specification

- **Feasibility study:**
    1. Technical Feasibility: The technical feasibility of the Campus Ambassador Portal is

robust, supported by the utilization of well-established and widely adopted technologies such as Node.js, Express.js, EJS, JavaScript, and CSS. These technologies offer scalability, flexibility, and a sturdy foundation for the development of a dynamic web portal. The availability of skilled developers proficient in the chosen stack ensures the technical viability of the project. Additionally, the modular architecture allows for future enhancements and adaptability to emerging technologies.

2. Economic Feasibility: From an economic perspective, the project demonstrates feasibility. The development costs are justifiable considering the potential benefits and revenue streams that the Campus Ambassador Portal can generate. The scalability of the platform ensures cost-effectiveness as the user base expands, contributing to a positive return on investment. The economic viability is further supported by market research, indicating a demand for efficient campus ambassador program management solutions and collaboration platforms.
3. Operational Feasibility: The operational feasibility of the Campus Ambassador Portal is evident in its ease of integration into existing campus ambassador programs. The user-friendly interface and comprehensive features facilitate a seamless adoption process for educational institutions, companies, and organizations. The platform enhances operational efficiency by centralizing communication, task assignment, and collaboration. Stakeholders can efficiently manage their engagement, contributing to the overall success and sustainability of the project.

- **Software Requirement:**

1. Data Requirement:

- Define the types of data the system will handle (e.g., user profiles, task assignments, performance metrics).
- Specify data sources, data formats, and any data conversion requirements.
- Detailed data storage and retrieval mechanisms, including database requirements.

2. Functional Requirement:

- List and describe the core functions of the Campus Ambassador Portal.
- Specify user roles and their corresponding permissions.
- Define use cases, scenarios, and expected interactions with the system.

3. Performance Requirement:

- Outline performance expectations, such as response time for user interactions.
- Specify system scalability requirements to accommodate potential growth.
- Identify any performance testing criteria and benchmarks.

4. Dependability Requirement:

- Detail the system's reliability and availability expectations.
- Specify backup and recovery mechanisms to ensure data integrity.
- Define any failover or redundancy requirements.

5. Maintainability Requirement:

- Describe measures for easy maintenance and updates.

- Specify coding standards and documentation practices.
- Detailed the process for system updates and version control.

6. Security Requirement:

- Outline security measures to protect user data and system integrity.
- Define user authentication and authorization protocols.
- Specify encryption and data privacy requirements.

7. Look and Feel Requirement:

- Describe the desired user interface aesthetics and layout.
- Specify any branding or style guidelines to be followed.
- Include considerations for user experience and ease of navigation.

This SRS document serves as a comprehensive guide for the development team, ensuring a clear understanding of the project's requirements and expectations across various dimensions. Each section addresses critical aspects of the software development process, contributing to the successful implementation of the Campus Ambassador Portal.

- **Validation**: Validation for the Campus Ambassador Portal is a comprehensive process aimed at confirming the system's functionality, performance, security, and user experience.
  - **Functional Validation:**
    - **Use Case Verification:** Ensuring all defined use cases and interactions are accurately implemented.
    - **User Acceptance Testing (UAT):** Engaging users to validate the system aligns with their expectations.
    - **Task Assignment and Tracking:** Confirming the accurate assignment and tracking of tasks within the portal.
    - **Communication Module Testing:** Validating the effectiveness of communication channels among stakeholders.
  - **Non-Functional Validation:**
    - **Performance Testing:** Assessing responsiveness and scalability under various loads.
    - **Security Testing:** Identifying and addressing potential vulnerabilities for data protection.
    - **Usability Testing:** Evaluating user interface and experience for ease of navigation.
    - **Compatibility Testing:** Verifying correct functionality across different browsers and devices.
  - **Iterative Validation:**

        - Ongoing feedback collection and refinement throughout the development lifecycle.
        - Continuous testing, feedback integration, and adjustments to meet user expectations.

- **Expected hurdles:**
    - Potential challenges may arise during integration of front-end and back-end technologies.
    - Ensuring seamless communication between Node.js, Express.js, and EJS components.
    - Addressing potential scalability issues as the user base expands.
    - Overcoming compatibility challenges across various web browsers.
    - Mitigating security concerns and ensuring robust data protection measures.
    - Adapting to evolving technology trends and updates.
    - Managing user experience complexities, especially in real-time task assignment.
    - Incorporating feedback and making iterative improvements during development.
    - Ensuring effective collaboration among stakeholders, including administrators, ambassadors, and collaborating entities.
    - Testing and validating system functionality across diverse use cases and scenarios.
    - SDLC: The project follows an Agile Software Development Life Cycle (SDLC) model, allowing for iterative development and continuous adaptation to evolving requirements, aligning well with the project's dynamic nature.

System Design

- **Design Approach: Object-Oriented:** The design approach for the Campus Ambassador Portal is object-oriented. Emphasizing modularity, reusability, and scalability, this approach employs encapsulation, inheritance, and polymorphism. Objects, represented using Unified Modeling Language (UML), facilitate independent development, data integrity, and seamless addition of new features. The object-oriented design ensures a structured foundation for the system's complexity and growth.

- **Detail Design**

The detailed design of the Campus Ambassador Portal involves a systematic breakdown of the system's components to guide the implementation phase effectively.

- **Component Breakdown:**
    - **User Management Module:** Responsible for handling user authentication, role assignment, and profile management.
    - **Task Assignment Module:** Manages the assignment and tracking of tasks, ensuring efficient communication between administrators and ambassadors.
    - **Communication Module:** Facilitates real-time communication, including notifications and updates, enhancing collaboration.
    - **Data Analytics Module:** Gathers and analyzes performance metrics to provide insights for program optimization.
- **Data Flow Diagrams (DFDs):**
    - DFDs illustrate the flow of data between modules, clarifying the interactions within the system. For instance, DFDs highlight the flow of task-related information between the Task Assignment and Communication modules.
- **Data Dictionary:**
    - The data dictionary outlines key data elements, such as user profiles, task details, and performance metrics. For instance, it defines user attributes like username, email, and role.
- **Structured Charts:**
    - Structured charts visually represent the hierarchy of processes within modules. For example, the Task Assignment Module's structured chart delineates the step-by-step process of task assignment and tracking.
- **Flowcharts:**
    - Flowcharts detail specific processes within modules. An example is the flowchart illustrating the process of real-time communication, demonstrating how notifications are initiated, delivered, and received.
- **Unified Modeling Language (UML):**
    - Class diagrams outline the structure and relationships of key objects within modules. Sequence diagrams depict the chronological flow of tasks between the Task Assignment and User Management modules.

Objectives of Detail Design:

- **Modular Structure:**

    - Designing modular components ensures that each module can be independently developed, facilitating ease of maintenance and future updates.
- **Data Integrity:**
    - Implementing encapsulation principles to protect data integrity and maintain consistency across the system.
- **Scalability:**
    - The design accounts for scalability, allowing for seamless integration of new features and accommodating growth in user interactions and data.
- **User Interaction:**
    - Defining the user interface components ensures a user-friendly experience, focusing on intuitive navigation and clear communication features.
- **Security Measures:**
    - Incorporating security measures includes defining access controls and encryption protocols to safeguard user data and system functionality.

In summary, the detail design phase provides a comprehensive plan for the implementation of the Campus Ambassador Portal, detailing components, data flows, and interactions to guide the development process effectively.

User Interface Design:

Login Page:

- The login page features a prominent sign-in section requiring a login ID and password for both administrators and users.
- A separate sign-up page gathers essential details from students, including name, email, LinkedIn profile, course duration, college name, and resume upload.

Admin Portal:

- **Home Page:**
    - The admin home page provides an overview of program statistics, task status, and important announcements.
    - Quick links to other admin portal sections enhance navigation.
- **Notify Page:**
    - The notify page allows administrators to send announcements and notifications to all

ambassadors.
    - Features include a message composition area and a list of previous announcements.
- **Task Assignment:**
    - This page enables administrators to assign tasks to ambassadors, specifying details such as description, deadline, and priority.
    - A task list displays the current assignments and their status.
- **Task Completed:**
    - Administrators can review and verify completed tasks on this page.
    - Task details, submission timestamps, and feedback options are available.
- **Ambassadors List:**
    - A page listing the names and profiles of ambassadors.
    - Includes filters for easy identification and selection.

User Portal:

- **Home Page:**
    - The user home page offers personalized content, including program updates and individual achievements.
    - Quick access to other user portal sections for seamless navigation.
- **Notifications:**
    - Ambassadors receive real-time notifications about new tasks, announcements, and program updates.
    - Notifications are organized for easy reference.
- **Task Assigned:**
    - This page displays tasks assigned to the ambassador, including descriptions, deadlines, and priority levels.
    - Ambassadors can view and manage their tasks efficiently.
- **Task Completed:**
    - A page displaying a history of completed tasks, including details and feedback.
    - Provides a comprehensive overview of individual accomplishments.
- **Feedback:**
    - Ambassadors can provide feedback on tasks, fostering open communication.
    - Feedback forms include ratings and comment sections.

Database Design:

- User database stores login credentials, personal details, and resumes.
- Task database manages task-related information, including assignment details and completion status.

- Communication database records notifications, announcements, and messages.
- Feedback database stores ambassador feedback on tasks.
- Admin database manages administrator accounts and access permissions.

This user interface and database design ensure a user-friendly experience while efficiently managing essential data for the Campus Ambassador Portal.

Database Normalization:

- Database normalization is a crucial process in the design of the Campus Ambassador Portal's database, aiming to reduce data redundancy, improve data integrity, and streamline data maintenance. The following represents a normalized structure for key entities in the system:
- **Entities:**
  - **Users:**
    - Attributes: UserID (Primary Key), Name, Email, LinkedIn Profile, Course Duration, College Name
    - Normalization: 1st Normal Form (1NF) - All attributes are atomic and there are no repeating groups.
  - **Login Credentials:**
    - Attributes: UserID (Foreign Key referencing Users), LoginID, PasswordHash
    - Normalization: 1NF - Non-prime attributes are fully functionally dependent on the primary key.
  - **Tasks:**
    - Attributes: TaskID (Primary Key), Description, Deadline, Priority, Status
    - Normalization: 1NF - Attributes are atomic and uniquely identified by the primary key.
  - **Task Assignments:**
    - Attributes: AssignmentID (Primary Key), TaskID (Foreign Key referencing Tasks), UserID (Foreign Key referencing Users), AssignmentDate
    - Normalization: 1NF - All attributes are atomic and uniquely identified by the primary key.
  - **Task Feedback:**
    - Attributes: FeedbackID (Primary Key), AssignmentID (Foreign Key referencing Task Assignments), Rating, Comments
    - Normalization: 1NF - All attributes are atomic and uniquely identified by the primary key.
  - **Normalization Steps:**
    - **First Normal Form (1NF):**

        - Ensures that each attribute in a table is atomic and that there are no repeating groups.
        - Achieved by breaking down data into the smallest, indivisible units.
    - **Second Normal Form (2NF):**
        - Ensures that no non-prime attribute is partially dependent on the primary key.
        - Achieved by removing partial dependencies by creating separate tables.
    - **Third Normal Form (3NF):**
        - Ensures that no non-prime attribute is transitively dependent on the primary key.
        - Achieved by removing transitive dependencies.

- **Benefits of Normalization:**
    - Reduced Data Redundancy: Data is stored efficiently, minimizing redundant information.
    - Improved Data Integrity: Updates and modifications are less error prone.
    - Simplified Data Maintenance: Changes to the database structure are more straightforward.
- **Considerations:** Denormalization might be considered for certain scenarios where query performance is a critical factor.
- The normalized database structure ensures efficient data management, integrity, and scalability for the Campus Ambassador Portal, providing a solid foundation for the system's functionality.

Database Manipulation

Database manipulation in the Campus Ambassador Portal involves the efficient retrieval, insertion, updating, and deletion of data. The following outlines key aspects of database manipulation:

- **Retrieve Data:**

    - **Task Retrieval:** Retrieve tasks assigned to a specific ambassador for display on their user portal.
    - **Notification Retrieval:** Retrieve real-time notifications for users and administrators for immediate display.
- **Insert Data:**
    - **User Registration:** Insert new user data during the registration process, capturing details such as name, email, and course duration.
    - **Task Assignment:** Insert new tasks assigned by administrators into the tasks table.
- **Update Data:**
    - **Task Status Update:** Update the status of tasks as they are completed by ambassadors.
    - **User Profile Update:** Allow users to update their profiles, including changes to their LinkedIn profiles or email addresses.
- **Delete Data:**
    - **User Deactivation:** Delete user data in case of account deactivation or closure.
    - **Task Removal:** Remove tasks that are no longer relevant or necessary from the tasks table.
- **Transaction Management:**

  Implement transaction management to ensure data consistency during multi-step operations. For example, when marking a task as completed, the system should simultaneously update task status and record feedback.

Database Connection Controls and Strings Methodology:

- **Connection Pooling:**
    - Utilize connection pooling to manage a pool of database connections, reducing overhead and improving performance.
    - Set maximum connection limits and connection timeouts to prevent resource exhaustion.
- **Secure Connection String Handling:**

- **Parameterized Queries:** Use parameterized queries to prevent SQL injection attacks, ensuring secure interaction with the database.
- **Connection Strings Encryption:** Encrypt connection strings to protect sensitive information, such as database credentials, during transmission.

- **Error Handling:**
  - Implement robust error handling mechanisms to gracefully manage database connection errors.
  - Log detailed error messages for diagnostic purposes while presenting user-friendly error messages to end-users.
- **Connection String Storage:**

  Safely store connection strings, ensuring they are not hard coded within application code.

  Leverage secure configuration management practices, such as environment variables or configuration files, to store and retrieve connection strings.
- **Connection Timeout and Pooling Parameters:**
  - Set appropriate connection timeout values to manage connection duration effectively.
  - Configure pooling parameters, such as minimum and maximum pool size, to optimize resource utilization.

Database manipulation in the Campus Ambassador Portal is designed to be secure, efficient, and adaptable, ensuring smooth interactions between the application and the underlying database. Connection controls and strings methodology are implemented with a focus on security, reliability, and performance.

Implementation, Testing, and Maintenance

Introduction to Languages, IDEs, Tools, and Technologies for Campus Ambassador Portal Implementation

The implementation of the Campus Ambassador Portal involves a carefully chosen to set of programming languages, integrated development environments (IDEs), tools, and technologies to ensure a robust, scalable, and user-friendly platform.

- **Backend Development:**
  - **Node.js:**

    - Purpose*:* Node.js is employed for server-side development, providing a non-blocking, event-driven architecture that enhances the portal's scalability and real-time capabilities.
    - Advantages*:* Efficient handling of concurrent connections, enabling asynchronous operations crucial for a responsive user experience.
  - **Express.js:**
    - Purpose*:* Express.js, a web application framework for Node.js, streamlines the development of the backend by providing a robust set of features for routing, middleware, and template engines.
    - Advantages*:* Simplifies the creation of RESTful APIs and ensures organized code structure.
- **Frontend Development:**
  - **EJS (Embedded JavaScript):**
    - Purpose*:* EJS is utilized for dynamic content rendering on the client-side, allowing the seamless integration of server-side variables into HTML templates.
    - Advantages*:* Enhances code readability and maintainability while facilitating the creation of dynamic and data-driven web pages.
  - **JavaScript:**
    - Purpose*:* JavaScript is the scripting language employed for client-side functionality, enhancing the interactivity and responsiveness of the user interface.
    - Advantages*:* Allows the creation of dynamic, asynchronous features and seamless integration with backend processes.
  - **CSS (Cascading Style Sheets):**
    - Purpose*:* CSS is utilized for styling and layout, ensuring a visually appealing and consistent user interface across different devices.
    - Advantages*:* Facilitates the separation of presentation from content, enabling efficient design modifications.
- **Development Environment:**
  - **Visual Studio Code:**
    - Purpose*:* Visual Studio Code serves as the primary integrated development environment for coding, debugging, and version control.
    - Advantages*:* Lightweight, extensible, and supports a variety of programming languages, enhancing the development workflow.

- **Database:**
  - **MongoDB:**
    - Purpose*:* MongoDB is chosen as the NoSQL database for its flexibility in handling unstructured data, making it suitable for storing user profiles, task details, and program analytics.
    - Advantages*:* Scalable, document-oriented database that aligns with the dynamic nature of the portal's data.

Coding Standards and Language Used:

Language Used: JavaScript (Node.js, Express.js, EJS)

- **Coding Standards:**
  - JavaScript coding standards are implemented following the Airbnb JavaScript Style Guide for consistency and readability.
  - Variable Naming: Variables are named descriptively, adhering to camelCase. Example: **taskAssignmentDate**.
  - Function Naming: Functions follow a verb-noun pattern for clarity. Example: **assignTask()**.
  - Indentation: Consistent use of two spaces for indentation throughout the codebase.
  - Comments: Comments are used to explain complex logic and provide context for code segments.
- **Linting:**
  - ESLint is integrated into the development workflow to enforce coding standards and identify potential issues.
  - ESLint configurations are aligned with the Airbnb style guide to ensure consistency.

Project Scheduling:

Tools: PERT, Gantt Charts, OpenProj

- **PERT (Program Evaluation and Review Technique):**
  - Task: Develop User Authentication Module
    - Optimistic Time: 3 days

        - Pessimistic Time: 7 days
        - Most Likely Time: 5 days
    - Critical Path: User Authentication -> Task Assignment -> Communication Module
- **Gantt Charts:**
    - Gantt chart depicts a timeline for key project tasks:
        - User Authentication (Jan 1 - Jan 10)
        - Task Assignment (Jan 5 - Jan 15)
        - Communication Module (Jan 10 - Jan 20)
- **OpenProj:**
    - OpenProj is utilized for dynamic project management, allowing for adjustments in timelines, resource allocations, and task dependencies.
    - Resource Allocation: Developers assigned to specific modules with defined work hours.

Testing Techniques and Test Plans:

Testing Techniques:

- **Unit Testing:**
    - Example Test: **testCalculateTaskPriority()** in the Task Assignment Module.
- **Integration Testing:**
    - Example Test: Validate communication between Task Assignment and Communication Module.
- **End-to-End (E2E) Testing:**
    - Example Test: Simulate user interaction from login to task completion.

Test Plans:

- **Test Objective: Verify User Authentication Module**
    - Test Case 1: Verify successful user registration.
    - Test Case 2: Confirm proper handling of invalid login credentials.
- **Test Objective: Validate Task Assignment Module**

- Test Case 1: Assign a task and verify its appearance on the user portal.
- Test Case 2: Test error handling for task assignment failures.

Automation:

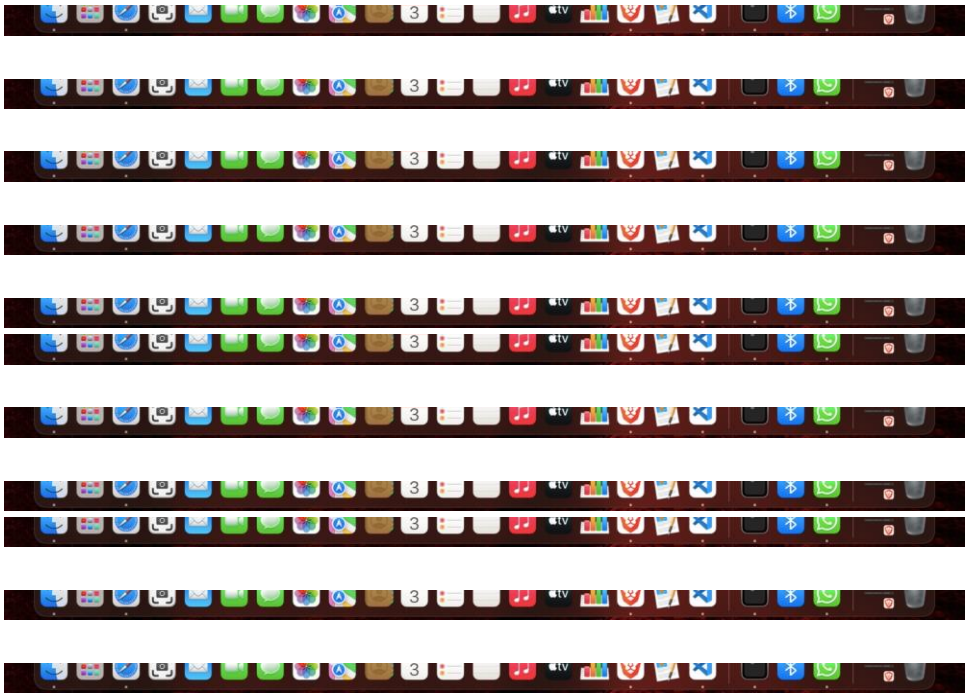
- Automated test scripts using Jest for unit testing and Cypress for end-to-end testing.
- CI/CD pipeline integration ensures automated testing with every code commit.

Regression Testing:

- Conducted after each development cycle to ensure new features do not impact existing functionalities.
- Test Case: Verify that changes in the Communication Module do not affect the Task Assignment Module

Results and Discussions

- **User Interface Representation**



Conclusion and Future Scope

Conclusion:

- The development of the Campus Ambassador Portal represents a significant stride towards creating an efficient, collaborative, and streamlined system for managing campus ambassador programs. The project successfully leverages JavaScript, Node.js, Express.js, EJS, and MongoDB to build a dynamic and responsive platform. Adhering to industry-standard coding practices ensures a maintainable and scalable codebase.
- The implementation of project scheduling tools like PERT, Gantt charts, and OpenProj facilitates effective planning, coordination, and resource allocation. These tools contribute to a well-organized development process, ensuring that tasks are completed within defined timelines.
- The testing strategies employed, including unit testing, integration testing, and end-to-end testing, validate the robustness of the system. Automation of test cases enhances efficiency, allowing for rapid and reliable identification of issues throughout the development lifecycle.

Future Scope:

- **Enhanced Analytics:**
  - Implement advanced analytics to provide administrators with deeper insights into ambassador performance, engagement trends, and program effectiveness.
- **Gamification Elements:**
  - Introduce gamification features to motivate ambassadors through rewards, badges, and leaderboards, fostering a competitive and engaging environment.
- **Machine Learning Integration:**
  - Explore the integration of machine learning algorithms for predictive analytics, allowing the system to anticipate trends and recommend personalized tasks for ambassadors.
- **Mobile Application:**
  - Develop a mobile application to provide ambassadors and administrators with on-the-go access to the portal, enhancing user convenience and accessibility.
- **Integration with External Platforms:**
  - Integrate the portal with external platforms, such as social media and communication tools, to streamline information sharing and enhance collaboration.
- **User Feedback Enhancements:**
  - Expand the feedback module to include more granular insights, allowing

ambassadors to provide detailed feedback on specific aspects of tasks and program dynamics.

- **Security Augmentation:**
  - Continuously enhance security measures, including encryption protocols, to safeguard user data and maintain the integrity of the portal.
- **Global Expansion:**
  - Explore opportunities for global expansion by adapting the portal to cater to diverse educational institutions and organizations worldwide.

In conclusion, the Campus Ambassador Portal serves as a foundational platform with the potential for continuous improvement and expansion. The identified future scope areas align with evolving trends in technology and user expectations, ensuring the portal remains innovative and valuable in the long term.

References:

- Flanagan, D. (2011). *JavaScript: The Definitive Guide.* O'Reilly Media.
- Node.js. (n.d.). Retrieved from https://nodejs.org/
- Express.js. (n.d.). Retrieved from https://expressjs.com/
- MongoDB. (n.d.). Retrieved from https://www.mongodb.com/
- EJS (Embedded JavaScript). (n.d.). Retrieved from https://ejs.co/
- GanttProject. (n.d.). Retrieved from https://www.ganttproject.biz/
- OpenProj. (n.d.). Retrieved from http://openproj.org/
- Jest - JavaScript Testing Framework. (n.d.). Retrieved from https://jestjs.io/
- Cypress - JavaScript End to End Testing Framework. (n.d.). Retrieved from https://www.cypress.io/
- Heroku. (n.d.). Retrieved from https://www.heroku.com/
- Git - Version Control System. (n.d.). Retrieved from https://git-scm.com/

Bibliography:

- Ambler, S. W. (2002). *Introduction to UML 2 Activity Diagrams.* Retrieved from http://www.agilemodeling.com/artifacts/activityDiagram.htm
- Sommerville, I. (2011). *Software Engineering.* Addison-Wesley.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach.* McGraw-Hill Education.
- Cockburn, A. (2002). *Agile Software Development: The Cooperative Game.* Addison-Wesley.