Software Requirements Specification (SRS) for “iEngage”

1. Introduction

1.1 Purpose

iEngage is a web-based platform designed to foster collaboration, networking, and engagement among university students, alumni, and faculty. This Software Requirements Specification (SRS) document defines the technical and functional requirements for iEngage, outlining its objectives, scope, functionalities, and constraints to ensure clear communication among stakeholders and developers. The platform enables users to participate in discussions, share knowledge, collaborate on projects, and organize or join events within a secure, interactive space.

Technical Highlights:

- Developed using modern web technologies to ensure performance and scalability.
- Integrated with secure university authentication systems for user verification.
- Designed with modular architecture to support future enhancements seamlessly.

1.2 Document Conventions

- Abbreviations:
    - UI: User Interface
    - UX: User Experience
    - Q&A: Questions and Answers
- Formatting:
    - Functional requirements are listed in bullet points for clarity.
    - Key features are organized into subsections with technical details highlighted.

1.3 Intended Audience and Reading Suggestions This document is intended for:

- Developers: To understand feature requirements, architecture, and technical constraints.
- Testers: To design test cases and validation scenarios based on functional requirements.

- Project Managers: To ensure that development aligns with project milestones and goals.
- Stakeholders: To validate that the platform's features meet user expectations and objectives.

1.4 Project Scope

iEngage aims to bridge the communication and collaboration gap within the university community. By integrating features such as Collaboration, Events, Queries, and Volunteer spaces, the platform facilitates interaction, knowledge sharing, and personal growth. The initial release focuses on core functionalities, with provisions for scalability and user-driven enhancements in future iterations.

Key Objectives:

- Networking: Foster connections among students, alumni, and faculty.
- Collaboration: Provide tools for teamwork on academic, professional, and volunteer projects.
- Event Management: Enable efficient planning and participation in university events.
- Knowledge Sharing: Establish a centralized space for academic and career-oriented discussions.

1.5 References

1. HTML & CSS Resources: W3Schools
2. Image Resources: Google Images
3. University Reference: Independent University, Bangladesh
4. Technical Guidance: ChatGPT and relevant developer documentation

2. Overall Description

2.1 Product Perspective

iEngage is a standalone web-based application tailored specifically for university communities. It functions as a centralized platform for academic, professional, and social engagement, offering an integrated suite of features that cater to diverse user needs. The system's modular design ensures scalability and ease of maintenance, making it adaptable to evolving requirements.

2.2 Product Features

- Volunteer Work: Facilitate opportunities for community service and skill-building projects.
- Event Calendar: Centralized tracking and management of university events.
- Collaboration: Dedicated spaces for teamwork and project management.
- Query: Q&A platform for academic and career-related discussions.
- Messaging: Direct and group messaging for user communication.
- User Profiles: Customizable profiles with privacy settings.
- Groups/Communities: Interest-based groups to foster engagement and collaboration.

2.3 User Classes and Characteristics

- Students: Engage in learning, volunteer work, events, and collaborations.
- Alumni: Network with peers, contribute to discussions, and mentor students.
- Faculty: Share knowledge, guide collaborative efforts, and participate in events.

2.4 Operating Environment

- Hardware: Desktop, laptop, and mobile devices with internet access.
- Software: Supported on modern browsers such as Chrome, Firefox, and Safari.
- Hosting: Deployed on cloud-based servers with scalable infrastructure to accommodate user growth.

2.5 Design and Implementation Constraints

- Authentication: Secure two-factor authentication for all users.
- Compliance: Adherence to university IT policies and data privacy regulations.
- Scalability: Designed to support increasing user demand without performance degradation.

2.6 User Documentation

- User Manual: Comprehensive guide accessible via the platform.
- FAQs: Contextual FAQs integrated with user workflows.

- Onboarding Tutorials: Step-by-step tutorials to assist first-time users**.**

2.7 Assumptions and Dependencies

- Users have access to stable internet connections and basic familiarity with web applications.
- Integration with the university's existing authentication systems is supported.
- External services (e.g., cloud storage) are reliable and meet performance expectations.

3. System Features

3.1 Volunteer Work

- **Description and Priority**: Core feature to enable users to participate in or organize volunteer opportunities. High priority.
- **Technical Details**:
    - **Backend Implementation**: Developed using RESTful APIs to handle creation, management, and tracking of volunteer opportunities.
    - **Frontend Features**: Interactive forms for creating opportunities, with validation and dynamic updates using JavaScript frameworks (e.g., React).
    - **Database Schema**: Tables for "Volunteer Opportunities", "Participants", and "Organizer Details" with relational mapping.
- **Functional Requirements**:
    - Create and manage volunteer opportunities.
    - Sign up for available opportunities and track participation.

3.2 Event Calendar

- **Description and Priority**: Centralized calendar for tracking and managing events. High priority.
- **Technical Details**:
    - **Backend Implementation**: Calendar and event data managed via a structured database with API endpoints for CRUD operations.

    - **Frontend Features**: Calendar component with date pickers and filters implemented using libraries like Full Calendar.
    - **Integration**: Notification system linked to user profiles for RSVP reminders.
- **Functional Requirements**:
    - Display upcoming events with detailed information.
    - Integrate RSVP functionality and reminders.
    - Allow users to filter and search for events by type or date.

3.3 Collaboration

- **Description and Priority**: Feature for creating and managing collaborative projects. High priority.
- **Technical Details**:
    - **Backend Implementation**: Collaboration spaces supported by a microservices architecture ensuring scalability.
    - **Frontend Features**: Dynamic project dashboards with real-time updates powered by WebSocket connections.
    - **File Management**: Integration with cloud storage services (e.g., AWS S3 or Google Drive) for file sharing.
- **Functional Requirements**:
    - Initiate and manage collaboration spaces for projects.
    - Approve membership requests to join collaborations.
    - Share resources, set milestones, and maintain project discussions.

3.4 Query

- **Description and Priority**: Q&A feature for academic and career-related discussions. Medium priority.
- **Technical Details**:
    - **Backend Implementation**: Designed as a message board system using a NoSQL database (e.g., MongoDB) for flexibility.

    - **Frontend Features**: User-friendly interface with sorting options for upvotes and accepted answers.
    - **Search Functionality**: Full-text search capabilities integrated for quick query retrieval.
- **Functional Requirements**:
    - Post questions and reply to others.
    - Upvote, downvote, and mark answers as accepted.

4. External Interface Requirements

4.1 User Interfaces

- Web-based UI designed using responsive design principles to ensure compatibility with various devices, including desktops, tablets, and smartphones.
- Intuitive dashboard layout featuring modular widgets for personalized user interaction.
- Enhanced accessibility features, including keyboard navigation, screen reader support, and ARIA labels.
- Interactive elements built with modern frontend technologies such as React.js, ensuring dynamic content rendering and minimal load times.

4.2 Hardware Interfaces

- The platform does not require specialized hardware; it operates efficiently on standard user devices.
- Compatible with devices featuring a minimum resolution of 1024x768 pixels and above.

4.3 Software Interfaces

- Seamless integration with university authentication systems via Single Sign-On (SSO) protocols such as OAuth 2.0 and SAML.
- Compatibility with third-party calendar APIs (e.g., Google Calendar, Microsoft Outlook) to facilitate event synchronization.

- Backend API endpoints designed following RESTful architecture, ensuring interoperability with future external services.

4.4 Communications Interfaces

- All data transmissions are encrypted using HTTPS protocols with TLS 1.3 for enhanced security.
- Real-time communication features (e.g., messaging and notifications) implemented using WebSocket protocols.
- Email notifications and reminders powered by integration with SMTP servers and third-party email services such as SendGrid.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Core pages load in under 2 seconds under typical network conditions (50 Mbps download speed).
- Scalable architecture is designed to support up to 1,000 concurrent users, with future plans to accommodate more based on demand.
- Optimized database queries and caching mechanisms to minimize server response times.

5.2 Safety Requirements

- Daily automated backups of user and system data stored in geographically redundant locations.
- Failover mechanisms are implemented with secondary servers to ensure minimal downtime in case of primary server failure.

5.3 Security Requirements

- All sensitive data, including user credentials, stored using AES-256 encryption.
- Regular security audits conducted following OWASP guidelines to identify and address vulnerabilities.
- Multi-factor authentication (MFA) enforced for all users, with options for SMS, email, or authenticator apps.

5.4 Software Quality Attributes

- **Maintainability**: Modular codebase following clean coding practices, with extensive inline documentation to facilitate updates.
- **Portability**: Cross-browser compatibility verified on Chrome, Firefox, Safari, and Edge; mobile optimization ensures smooth operation on Android and iOS.
- **Usability**: UI/UX design tested with diverse user groups to ensure ease of navigation and functionality.

6. Other Requirements

- Scalability to support a growing user base, with load balancers and horizontal scaling capabilities integrated into the system architecture.
- Multilingual support planned for future phases, starting with widely spoken languages in the user base.

Appendices

A. Glossary

- **RSVP**: Responding to event invitations.
- **Two-Factor Authentication (2FA)**: A security measure requiring two forms of verification.
- **OAuth 2.0**: An open standard for access delegation commonly used for token-based authentication.
- **SAML**: Security Assertion Markup Language, used for Single Sign-On (SSO) solutions.

B. Analysis Models

- Detailed use case diagrams, sequence diagrams, and entity-relationship models to be included in subsequent iterations.

C. Issues List

- Pending finalization of hosting provider.
- Integration testing with university systems to ensure seamless functionality.