Software Requirements Specification (SRS)

For

Intelligent Peer Learning Platform

Prepared by

Shreyashi Singh - 2301320100141

Shriya Shukla – 2301320100142

Shivam Kumar – 2301320100xxx

CSE 3C

24, Sept 2025

Table of Contents

Software Requirements Specification (SRS)

Intelligent Peer Learning Platform

1. Introduction

1.1 Purpose

This document specifies the requirements for the Intelligent Peer Learning Platform (IPLP), which aims to provide a scalable, secure, and AI-powered online peer learning ecosystem. The platform enables personalized peer matching, AI-supported assistance, and real-time progress analytics, fostering an inclusive, interactive, and engaging learning community.

1.2 Scope

IPLP will support a diverse user base including students, educators, and administrators, offering features such as multimedia course content management, chat-enabled mentorship, AI-driven peer matching, predictive analytics, and role-based security. It targets learners worldwide and integrates modern web technologies (MERN stack) with machine learning models.

1.3 Definitions, Acronyms, Abbreviations

- MERN: MongoDB, Express.js, React.js, Node.js
- ML: Machine Learning
- NLP: Natural Language Processing
- JWT: JSON Web Tokens
- OAuth: Open Authorization
- CDN: Content Delivery Network

2. Overall Description

2.1 Product Perspective

IPLP is an independent web application built on a microservices architecture, containerized with Docker, and orchestrated with Kubernetes for scalability. It integrates AI inference endpoints and third-party authentication providers.

2.2 Product Functions

- User registration, login, and profile management.
- Course creation, editing, and multimedia content upload.
- Real-time chat interface for peer mentorship and AI chatbot interaction.
- AI-powered peer matching based on learning behaviour and profiles.
- Analytics dashboard for real-time tracking of progress and predictive alerts.
- Role-based access and administrative controls.

2.3 User Characteristics

Users include students (learners), teachers (mentors/content providers), and administrators.

2.4 Constraints

- Must comply with GDPR and data privacy regulations.
- Responsive design supporting desktop and mobile platforms.
- Cloud deployment with reliable uptime and security.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Management

- Users can register/login using email or third-party (Google OAuth).
- Password encryption and two-factor authentication (2FA) offered.
- Role assignment: student, teacher, admin.

3.1.2 Course Management

- Teachers can create, update, delete, and organize course materials.
- Support uploading multimedia content (videos, documents, quizzes).
- Students can enroll, access materials, and submit assignments.

3.1.3 Peer Matching

- ML algorithms dynamically match learners with peers or mentors.
- Matching criteria includes skills, learning styles, past interactions.

3.1.4 Communication

- Real-time chat between peers and with AI assistants powered by NLP.
- Group discussions, private messages, and notifications supported.

3.1.5 Analytics & Reporting

- Dashboards showing progress metrics, engagement, and performance trends.
- Predictive alerts for at-risk learners sent to students and teachers.

3.1.6 Security

- Session management with JWT tokens.
- Secure API access with HTTPS.
- Role-based authorization controls.

3.2 Non-Functional Requirements

3.2.1 Performance

- System should handle concurrent users with minimal latency (<200ms response).

3.2.2 Scalability

- Support horizontal scaling with Kubernetes and cloud resources.

3.2.3 Usability

- Intuitive, accessible UI compliant with WCAG 2.1 standards.

3.2.4 Reliability

- 99.9% uptime, fault tolerance through container orchestration.

3.2.5 Maintainability

- Modular codebase with documentation and CI/CD pipelines.

3.2.6 Security

- Data encryption in transit and at rest.
- Compliance with GDPR and industry best practices.

4. System Architecture Overview

- React.js frontend communicates with Node.js/Express.js backend APIs.
- MongoDB Atlas as the primary data store.
- TensorFlow.js and Hugging Face models for AI/ML services.
- AWS S3 and CDN for multimedia content delivery.
- Docker for containerization and Kubernetes for orchestration.