Software Requirements Specification

for

Student Project Management System

Version 1.0 approved

Prepared by

Anand Reddy	**2203031241211**
Bhubnesh Mahrana	**2203031240166**
Tejeswara Rao	**2203031241451**
Neel Thakkar	**2303031247032**

Artificial Intelligence

PIET, Parul University

21-Feb-2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The objective of the Student Project Management System (SPMS) is to automate the process of managing final-year student projects. This system will ease students, mentors, and administrators through the provision of an integrated platform for monitoring project progress, deadlines, and teamwork. The system will also have a Retrieval-Augmented Generation (RAG) chatbot to allow mentors to ask previous project information in natural language, and a plagiarism detector to provide certainty on originality in submissions from students.

This system will help to:

- Offer a centralized project management platform.
- Allow efficient tracking of students' progress and deadlines.
- Provide an intuitive natural language chatbot for helping mentors in retrieving data associated with projects.
- Detect plagiarism by analyzing current projects in comparison to previously submitted projects.

1.2 Document Conventions

- This document adheres to standard conventions:
- Footnotes or bold text will be employed to highlight significant terms.
- Wherever relevant, UML diagrams and flowcharts will be provided to discuss system architecture.

1.3 Intended Audience and Reading Suggestions

The main audience for this document are:

- Developers – System architecture, database schema, and APIs understanding.
- Project Mentors – How to communicate with the chatbot and plagiarism checker.
- Administrators – Student registrations, project approvals, and deadline management.
- Students – Submission process and project tracking understanding.

- Stakeholders – System capabilities and features review.

1.4 Product Scope

The Student Project Management System (SPMS) is a web-based system that seeks to improve final-year project management by offering:

- Project Tracking – Students and mentors can monitor milestones, deadlines, and progress.
- Team Collaboration – Students can create teams, allocate tasks, and track status.
- Mentor Assistance – A RAG-based chatbot enables mentors to ask historical project questions in natural language.
- Plagiarism Detection – The system checks submitted projects against past ones stored in a vector database.
- Admin Management – Admins manage project approvals, student registrations, and mentor allocations.

Technologies Used

- Frontend: Next.js
- Backend: Bun, Prisma, PostgreSQL
- Vector Database: FAISS or Pinecone
- AI Features: LangChain, Hugging Face Embeddings, Transformers, Ollama/Other LLMs
- Plagiarism Checker: NLP-based text similarity checking

1.5 References

- PostgreSQL Documentation: https://www.postgresql.org/docs/
- Next.js Documentation: https://nextjs.org/docs
- Prisma ORM: https://www.prisma.io/docs
- LangChain: https://python.langchain.com/docs/
- FAISS: https://faiss.ai/

- Pinecone: https://www.pinecone.io/

2. Overall Description

2.1 Product Perspective

- Student Project Management System (SPMS) is an online application used to centralize and automate final-year student projects' management. It combines project monitoring, collaboration among team members, mentor guidance through RAG-based chatbot, and plagiarism checking into one platform.
- SPMS is an independent system that communicates with:
    - Database (PostgreSQL + Prisma ORM) – Holds all the project information.
    - Vector Database (FAISS or Pinecone) – Facilitates quick retrieval of historical project information for chatbot and plagiarism detection.
    - Large Language Models (LLMs) through LangChain & Ollama/Hugging Face Transformers – Drives the mentor chatbot.
    - Authentication System (OAuth, JWT, or Custom Authentication) – Handles user roles securely.

2.2 Product Functions

The system offers the following primary functionalities:

1. Project Management:
    - Students can create and manage project information.
    - Mentors can approve, reject, and monitor projects.
    - Progress monitoring with milestone updates and deadlines.
2. Team Collaboration:
    - Students can create teams and allocate roles.
    - Mentors can chat with students and track tasks.
    - File/document uploads for project reports and submissions.

3. Mentor Chatbot (RAG-based):
    - Mentors can pose project-related questions in natural language.
    - The chatbot retrieves relevant past project information using FAISS/Pinecone + LLMs.
    - Gives instant insights on methodologies, reports, and similar past projects.
4. Detection of Plagiarism:
    - Compares new project submissions against historical project data.
    - Uses vector embeddings and NLP similarity models.
    - Identifies potential plagiarism cases for review by mentor/admin.
5. Admin Management:
    - Approves student registrations and assignment of mentors.
    - Establishes deadlines and oversees system-wide activities.
    - Manages plagiarism reports and access by chatbot.

2.3 User Classes and Characteristics

The system accommodates three major user classes: students, mentors, and administrators.

- Students are in charge of developing, organizing, and submitting their projects. They can create teams, delegate roles, and monitor their progress within the system. Their access is restricted to their own project information and assigned mentor comments.

2.4 Operating Environment

SPMS is a web application built with Next.js for the frontend and Bun along with Prisma ORM for the backend. PostgreSQL is used as the database, providing stable storage for structured data. The system also uses FAISS or Pinecone as a vector database for efficient retrieval of previous project data.

The LangChain and LLM-based (e.g., Ollama or Hugging Face Transformers) AI-driven mentor chatbot processes and responds to user input. Authentication and access control are managed via OAuth or JWT-based protocols.

The platform is available through contemporary web browsers like Chrome, Firefox, Edge, and Safari, with support for various devices. It is intended to be deployed on cloud platforms like AWS, Vercel, or DigitalOcean.

2.5 Design and Implementation Constraints

There are several constraints that need to be taken into account while developing and implementing SPMS:

- The database needs to be optimized to support large amounts of project data efficiently. Indexing and query optimization are required to ensure performance, particularly when retrieving historical projects.
- The FAISS or Pinecone vector search system has real-time update limitations, so recently added projects cannot be searched immediately. Synchronization is done periodically to ensure accurate retrieval.
- Hosting an LLM for the mentor chatbot comes with large computational costs and possible latency issues. The system needs to be optimized for fast response speed versus accuracy at the cost of infrastructure expense.
- Security is a significant concern. As the system involves dealing with sensitive student and project information, encryption methods and role-based access control need to be incorporated to avoid unauthorized access.

2.6 User Documentation

To facilitate ease of use, SPMS will include detailed user documentation for various stakeholders:

- A Student Guide will guide students through project creation, team management, and submission operations. It will also detail how progress can be tracked by mentors, how to provide feedback, and how the chatbot is used for accessing information.
- Developers will be provided with API Documentation where they can access information to enable other functionalities or integrate external services.

- A Manual for Plagiarism Checker will detail plagiarism detection mechanisms, similarity result computation, and interpreting marked cases on the part of the mentors.

2.7 Assumptions and Dependencies

Functionality of SPMS depends on some important assumptions and dependencies to work effectively:

- It is assumed that users have a reliable internet connection because the system is cloud-based and needs to access the internet. It depends upon PostgreSQL and Prisma ORM for managing structured data, and FAISS or Pinecone for vector-based search efficiently.
- The accuracy of the mentor chatbot will rely on the quality of the embedded project data and the fine-tuning of the LLM model. Its effectiveness could require periodic updates and enhancements.
- The plagiarism detection will be text-based for the project reports, so other tools could be necessary for detecting plagiarism in code if programming-based projects are a primary component.

3. External Interface Requirements

3.1 User Interfaces

The Student Project Management System (SPMS) will feature an intuitive, responsive web-based UI for ease of use. The UI will be developed using Next.js to create a dynamic, modern experience. It will be accessible via desktop and mobile devices with a responsive design that adapts to various screen sizes.

Primary UI Components:

1. **Login & Authentication Page** – Users log in through OAuth (Google, GitHub, University Login) or email/password login.
2. **Dashboard** – Custom dashboard for students, mentors, and admins showing important project details, deadlines, and notifications.
3. **Project Management Page** – Students can add, edit, and submit projects, and mentors and admins can approve/reject and add feedback.

4. **Mentor Chatbot Interface** – A chat box where mentors may input questions in natural language and receive answers based on previous project data through RAG (Retrieval-Augmented Generation).
5. **Plagiarism Checker Dashboard** – A dashboard for mentors and admins to see reported submissions and similarity reports.
6. **Team Collaboration Module** – A module for students to organize teams, assign roles, and monitor progress.
7. **Admin Panel** – An admin panel where the administrators control user accounts, deadlines, and system-wide settings.

The UI will be developed using Tailwind CSS and React components to provide a modern, consistent, and visually attractive experience.

3.2 Hardware Interfaces

SPMS is a web-based, cloud-hosted application, so it doesn't have direct hardware dependencies for users. However, the server infrastructure on which the system runs needs to have some requirements:

- **Server-Side Hardware Requirements:**
    - Processor: Quad-Core CPU minimum (for simultaneous user request processing).
    - Memory (RAM): 16GB RAM minimum for smooth database and AI model execution.
    - Storage: 500GB SSD (or larger) for storing project reports, documents, embeddings, and chatbot logs.
    - GPU (in case of self-hosting for AI/LLM Processing): NVIDIA A100 or similar for high-speed LLM inference.
- **Client-Side Requirements:**
    - The system will be available on laptops, desktops, tablets, and mobile phones.
    - A recent web browser (Chrome, Firefox, Edge, Safari) with JavaScript enabled is needed.

3.3 Software Interfaces

SPMS combines different software components to function smoothly.

- **Frontend Technologies:**
  - Next.js – React frontend framework for speedy, server-rendered pages.
  - Tailwind CSS – For styling and responsive UI design.
- **Backend Technologies:**
  - Bun – High-performance JavaScript runtime for processing API requests.
  - Prisma ORM – Database interaction layer for efficient queries on PostgreSQL.
- **Database & Storage:**
  - PostgreSQL – Relational database for structured user and project data.
  - FAISS/Pinecone – Vector database for storing historical project embeddings for plagiarism detection and chatbot.
  - Cloud Storage (AWS S3, Google Cloud Storage, or equivalent) – For storing uploaded project documents.
- **AI & Machine Learning Services:**
  - LangChain – Framework for controlling the RAG-based chatbot.
  - Hugging Face Transformers / Ollama – Supplies LLM models for chatbot responses.
  - NLP-based Plagiarism Detection – Custom or third-party plagiarism detection model.
- **Authentication & Security:**
  - OAuth 2.0 / JWT – Authenticate users securely and manage sessions.
  - Role-Based Access Control (RBAC) – Provide different access levels to students, mentors, and admins.
- **API Endpoints:** SPMS will expose RESTful APIs and GraphQL APIs to retrieve data and interact between components.

3.4 Communications Interfaces

SPMS needs reliable communication paths between components.

- **Internal Communication (Backend & Database)**
    - The backend (Bun) communicates with PostgreSQL through Prisma ORM for query-structured communication.
    - The mentor chatbot communicates with FAISS/Pinecone to get project information.
    - LangChain and LLMs execute chatbot queries and provide natural language outputs.
- **External Communication (APIs & Third-Party Services)**
    - Email Notifications – Delivered through SendGrid or AWS SES for deadline reminders and notifications.
    - OAuth Authentication – Utilizes Google, GitHub, or University Login for secure login.
    - Plagiarism API (if third-party integration is employed) – Communicates with Turnitin or an NLP-based text similarity API for plagiarism checks.
- **Client-Server Communication**
    - The frontend (Next.js) sends and receives data from the backend using REST APIs and WebSockets (for real-time updates).
    - Socket.io or Firebase Cloud Messaging (FCM) will be used to handle real-time notifications.
- **Security Measures for Communication**
    - HTTPS encryption will be mandatory for all communications.
    - Encryptions at rest and in transit to secure sensitive data.
    - Rate limiting and API security to avoid abuse and unauthorized access.

4. System Features

The Student Project Management System (SPMS) comprises several features intended to simplify project management, mentor guidance, and plagiarism checking.

Each feature provides an improved user experience by facilitating collaboration, automating processes, and maintaining originality in student projects.

4.1 Project Management & Tracking

- **Description** This feature enables students to initiate, edit, and manage final-year projects. It supports mentors in tracking progress, offering feedback, and accepting or rejecting project milestones.
- **Key Functionalities**
    - Project Creation & Submission – Students can create projects, set objectives, and submit progress reports.
    - Milestone Tracking – Projects will have deadlines and milestones, which can be tracked by mentors.
    - Mentor Approval & Feedback – Mentors can review project submissions, offer feedback, and ask for revisions.
    - Admin Control – Admins can track all projects, approve submissions, and modify deadlines.
- **Actors Involved**
    - Students – Create and control projects.
    - Mentors – Monitor progress, offer feedback, and approve/reject project submissions.
    - Administrators – Track and handle all project information.
- **Constraints & Dependencies**
    - Needs an existing connection to the database (PostgreSQL + Prisma) for storing and fetching project information.
    - System reminders depend on email or in-app notifications to keep users updated about deadlines.

4.2 Team Collaboration & Communication

- **Description** Students usually collaborate in teams for their final-year projects. This aspect enables them to communicate, delegate tasks, and work effectively in teams.

- **Key Functionalities**
    - Team Creation – Students can add team members and define roles.
    - Task Assignment – Each team can allocate tasks with deadlines.
    - Document Sharing – Upload and share files, reports, and presentations.
    - Chat System – In-app chat or integration with Slack/Discord for communication among teams.
- **Actors Involved**
    - Students – Form teams, allocate tasks, and monitor team progress.
    - Mentors – Monitor team progress and chat with students.
- **Constraints & Dependencies**
    - Needs real-time database updating to show team changes.
    - Chat feature may require integration with WebSocket or Firebase for real-time communication.

4.3 Mentor Chatbot (RAG-based AI Assistant)

- **Description** A Retrieval-Augmented Generation (RAG) chatbot enables mentors to bring up historical project information and receive immediate answers regarding methodologies, reports, and related subjects.
- **Key Functionalities**
    - Natural Language Queries – Mentors can pose questions in English.
    - AI-Powered Responses – Leverages LangChain, FAISS/Pinecone, and Hugging Face Transformers to retrieve and construct responses.
    - Project Insights – Retrieves past projects with similar domains or methodologies.
    - Document Summarization – Returns summaries of previous reports or research papers.
- **Actors Involved**
    - Mentors – Utilize the chatbot for support.

- Administrators – Control chatbot access and track performance.

- **Constraints & Dependencies**
  - Dependent on vector database (FAISS/Pinecone) for quick retrieval of historical project data.
  - Dependent on LLM (Ollama, GPT-4, or Hugging Face models) for response generation.
  - Quality of stored project data affects AI performance.

4.4 Plagiarism Detection

- **Description** Guarantees uniqueness in student projects by comparing new submissions to stored projects with text similarity and vector embeddings.

Primary Functionalities

- Automatic Plagiarism Detection – Compares new project reports against stored projects.
- Similarity Score Generation – Generates similarity scores based on NLP-based embeddings.
- Mentor/Admin Check – In the event of detected plagiarism, mentors and admins can check highlighted projects.
- Detailed Plagiarism Reports – Displays matched content and percentage similarities.

Actors Involved

- Students – Upload checked-for-plagiarism projects.
- Mentors – Flagged projects for review.
- Administrators – Oversee plagiarism detection and mediate disputes.

Constraints & Dependencies

- Utilizes FAISS/Pinecone for high-speed vector similarity search.
- Accuracy relies on the quality of saved project embeddings.

- Can possibly incorporate third-party plagiarism APIs for further checking.

4.5 Admin Panel & User Management

- **Description** Centralized Admin Panel enables administrators to manage user roles, deadlines, and system configuration.

Key Functionalities

- User Role Management – Manage roles (Student, Mentor, Admin).
- Deadline & Notification Control – Establish project deadlines and send reminders.
- Project Approval & Oversight – Approve and oversee all project submissions.
- System Analytics – See stats on user interaction, project submissions, and plagiarism reports.

Actors Involved

- Administrators – Maintain the entire system.

Constraints & Dependencies

- Needs secure authentication (OAuth, JWT-based) for admin access.
- Relies on PostgreSQL database for user role management.

4.6 Authentication & Security

- **Description** Guarantees safe access to the platform via role-based authentication and encryption.

Key Functionalities

- OAuth/Email Authentication – Supports Google, GitHub, and university login.
- JWT Token Security – Provides secure API access.
- Role-Based Access Control (RBAC) – Avoids unauthorized access.
- Data Encryption – Secures stored project data and user details.

Actors Involved

- Students, Mentors, Administrators – Access the system securely.

Constraints & Dependencies

- Must use OAuth 2.0 or JWT-based authentication.
- All API calls must be made over HTTPS.

4.7 Notifications & Alerts

- **Description** Automated alerts keep students, mentors, and admins updated about project deadlines, approvals, and plagiarism reports.
- **Key Functionalities**
- Email & In-App Alerts – Deadline, approval, and plagiarism report notifications.
- Real-Time Updates – Live alerts for team collaborations and mentor feedback.
- **Actors Involved**
- Students, Mentors, Administrators – Role-based notifications.
- **Constraints & Dependencies**
- Needs third-party email service (SendGrid, AWS SES) for email notifications.
- In-app notifications require WebSocket or Firebase integration.
-

5. Other Nonfunctional Requirements

- This part presents the nonfunctional requirements that determine the performance, security, quality, and business requirements of the Student Project Management System (SPMS). These requirements will ensure the system works effectively, safely, and fulfills the expectations of its users.

5.1 Performance Requirements

- The system must handle at least 500 simultaneous users (students, mentors, and admins) without significant performance degradation.
- The API response time must be less than 500ms for normal operations such as project retrieval, progress updates, and chatbot queries.
- The mentor chatbot response time must be less than 2 seconds for the majority of queries (may depend on model complexity and data retrieval).

- Plagiarism detection must take 5–10 seconds per submission to perform checks and produce similarity reports.
- Large file uploads (up to 100MB per file) must be supported for project documents and research materials.
- The FAISS/Pinecone vector search system must respond with relevant previous project results in 1 second.
- The database needs to be properly indexed, cached (Redis), and query-optimized to quickly retrieve project information.

5.2 Safety Requirements

- The system must maintain data integrity by ensuring that critical project information is not accidentally deleted or overwritten.
- Backups of the database must be done daily to avoid data loss in the event of failures.
- The system must be GDPR (General Data Protection Regulation) and FERPA (Family Educational Rights and Privacy Act) compliant if required, so that student information is safeguarded.
- In case of a system failure, it must have an auto-recovery feature to return to the previous saved state.
- User activity logs must be kept for auditing and tracking purposes to maintain accountability.

5.3 Security Requirements

Authentication and Access Control

- All users will have to authenticate via OAuth 2.0 (Google, GitHub, or University Login) or JWT-based authentication.
- Role-Based Access Control (RBAC) will provide proper permissions to students, mentors, and administrators.

Data Security & Encryption

- All sensitive information (passwords, project files) must be encrypted at rest and in transit using AES-256 encryption.

- All API calls must be HTTPS (SSL/TLS encryption) secured to avoid data interception.

Protection Against Cyber Threats

- Apply rate limitation and CAPTCHA authentication to avoid brute-force attacks.
- Protect plagiarism detection APIs against unauthorized access to saved projects.
- Regular security audits and penetration testing to detect vulnerabilities.

Logging & Monitoring

- Have detailed logs for user activity to record any unauthorized actions.
- Use SIEM (Security Information and Event Management) tools to monitor security incidents in real-time.

5.4 Software Quality Attributes

Scalability

- The system must be able to handle an increasing number of students, projects, and mentors without a decrease in performance.
- Horizontally scalable by adding more servers (e.g., with Kubernetes or AWS Auto Scaling).

Availability

- The system must provide 99.9% uptime to provide users with constant access.
- Automatic recovery mechanisms should automatically restart failed services.

Usability

- The UI must be intuitive, easy to use, and mobile responsive to provide ease of use for mentors and students.
- All functionalities must be within three clicks of the dashboard.

Maintainability & Extensibility

- The codebase must be modular and documented to ensure future updates are easy.

- APIs must adhere to RESTful or GraphQL standards for seamless integration with third-party services.

Interoperability

- The system must be integrated with university databases, third-party plagiarism detection software, and external storage services (AWS S3, Google Drive, etc.).
- Data must be exportable in various formats (PDF, CSV, JSON) for easy sharing.

5.5 Business Rules

User Registration & Authentication

- Access to the platform is limited to authorized university students and staff.
- Registration has to be done using official university email.

Project Submission Guidelines

- A student can be a member of only one project team.
- The submission has to be accompanied by a report, code (if any), and a final presentation.
- Projects cannot be modified after final submission unless approved by the mentor.

Plagiarism Policy

- Any project with a similarity score over 30% (other than citations and common expressions) will be referred to mentor review.
- Students who are caught copying previous projects without correct citations could be penalized according to university academic rules.

Mentor Responsibilities

- One mentor can supervise up to 5 projects simultaneously.
- Mentors should review and respond in 7 days to a student submission.

Admin Responsibilities

- Admins retain complete control of project approvals, user administration, and system settings.
- Only admins are able to adjust deadlines or bypass project submissions when needed.
-

6. Other Requirements

- This category encompasses any further requirements that don't fit into earlier categories but are essential to the successful deployment and operation of the Student Project Management System (SPMS).

6.1 Compliance Requirements

- The system will have to conform to academic integrity policies established by the university.
- Any personal student information gathered must align with GDPR and FERPA.

6.2 Backup and Disaster Recovery

- The system is required to back up project data automatically on a daily basis to ensure that data will not be lost.
- In the event of server failure, the system must be able to recover from the previous successful backup in 30 minutes.
- Backups must be retained securely in various locations (cloud and on-premises when necessary).

6.3 Integration with University Systems

- The system must be able to integrate with the Learning Management System (LMS) of the university if necessary.
- It must have API-based integrations to synchronize student and mentor information from the university database.

6.4 Legal Considerations

- All projects submitted are owned by students intellectually, but the university has the right to use them for educational purposes.

- The plagiarism feature should be applied only for scholarly purposes and should not infringe on copyright rules.

Appendix A: Glossary

- This page explains the key terms employed within the document.
- **SPMS (Student Project Management System)** – The internet-based system to facilitate students, mentors, and administrators in the management of final-year projects.
- **RAG (Retrieval-Augmented Generation)** – AI method implemented in the mentor chatbot to retrieve pertinent project information prior to responding.
- **FAISS/Pinecone** – Vector databases employed for speedy similarity searches on previous projects.
- **Prisma ORM** – The tool that facilitates database interactions on PostgreSQL.
- **Bun** – Rapid JavaScript runtime applied to backend API operations.
- **OAuth 2.0** – Secure authentication method employed for logins (Google, GitHub, University Login).
- **RBAC (Role-Based Access Control)** – Security control to limit access to the system based on the roles of the users (student, mentor, admin).
- **JWT (JSON Web Token)** – A token-based authentication technique utilized to authenticate API requests.
- **Embedding** – A method to transform text data (projects, reports) into numerical representations for similarity searches.
- **Vector Search** – A search approach that retrieves similar documents based on embeddings.
-
- **Appendix B: Analysis Models**
- This section consists of diagrams and models employed in the system design. You can include:

- **Use Case Diagram** – Illustrates interactions among students, mentors, admins, and system functionalities.
- **ER Diagram (Entity-Relationship Diagram)** – Displays database structure (students, mentors, projects, submissions).
- **Sequence Diagrams** – Shows interactions among system components (e.g., project submission workflow).
- **Flowcharts** – Describes sequential processes (e.g., plagiarism detection, chatbot query processing).
- **State Diagram** – Shows project lifecycle states (Draft → Submitted → Under Review → Approved).