Software Requirements Specification

For Teachtrack

Version 1.1

Prepared by
PROFYZE

Abhiram Ayla	**se22uari005**	**se22uari065@mahindrauniversity.edu.in**
Kandala Mahitha Siri	**se22uari066**	**se22uari066@mahindrauniversity.edu.in**
Aarnav Tandava	**se22uari002**	**se22uari002@mahindrauniversity.edu.in**
Shreya Seshadri	**se22uari159**	**se22uari159@mahindrauniversity.edu.in**
Jotsna Sree	**se22uari123**	**se22uari123@mahindrauniversity.edu.in**

Instructor: Avinash Arun Chauhan

Course: Software Engineering

Lab Section: Tuesday 8:30-10:30 (AI lab)

Teaching Assistant: Murali Krishna Bukkasamudram

Date: 4th March 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being	00/00/00

1. Introduction

1.1. Document Purpose

This is a **Software specific Requirement (SRS)** document that specifies the software requirements for the **Professor Rating Website**, a web-based platform that allows students to rate and review their professors based on various criteria such as teaching quality, grading fairness, and overall experience. The system will provide a structured and reliable method for students to evaluate professors while allowing professors to view feedback and ratings.

This SRS defines the functional and non-functional requirements of the Professor Rating Website, covering all system components, including student interaction, professor verification, review moderation, and administrative control. The document provides a detailed technical foundation for developers, designers, and stakeholders involved in the project.

1.2. Product Scope

This a web-based platform is designed to help students make informed decisions by providing detailed professor reviews and ratings. By offering a transparent and user-driven rating system, the platform ensures that students can evaluate professors based on feedback from their peers. Students can search for professors, view their ratings, check a leaderboard of top-rated professors, and submit anonymous reviews about their professors or to share their learning experiences. Professors, upon verification, can access their ratings and reviews, enabling them to gain insights into student feedback and improve their teaching methods accordingly.

In addition to serving students and professors, the system includes an **administrator dashboard** for managing users, moderating reviews, and ensuring fair usage of the platform. The website will be developed using **HTML5, CSS, and JavaScript for the frontend**, while **PHP (Hypertext Processor) and MySQL** will handle the **backend** operations for managing and securing all student and professor records, ensuring secure data processing and storage. The system is designed to be **scalable, responsive, and user-friendly**, making it accessible across multiple devices while efficiently managing high user engagement.

1.3. Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is intended for **developers, project managers, administrators, professors, and students** who will interact with the **Professor Rating Website**. Developers will use this document to understand the functional and non-functional requirements necessary for building the system. Project managers will oversee if everyone is performing their required roles, ensuring alignment with the defined objectives. Administrators will rely on this document to oversee user management and moderation policies, while professors and students can gain insights into how the platform operates and the features available to them.

Brief Overview of what this SRS document contains:

- **1: Introduction** – Provides an overview of the document, including its purpose, scope, Target audience, and key definitions used for interpretation.
- **2: Overall Description** – Outlines the general functionality of the system, including user roles, key features, and system constraints.
- **3: Specific Requirements** – Details the functional and non-functional requirements, including system interactions, data flow, and expected behavior.
- **4: Other Non-functional Requirements** – Specifies performance, security, usability, and regulatory requirements.
- To end it with a **Data Appendix** and a **Group log**

For optimal understanding, readers should begin with **Section 1 (Introduction)** to grasp the project's purpose and scope. Developers and testers should then focus on **Sections 2 and 3**, as they define system functionality and technical specifications. Administrators and stakeholders may find **Section 4** particularly relevant for

understanding compliance, security, and other Regulatory Software Attributes. The document concludes with a **Data Appendix** containing relevant datasets or system-related data and a **Group Log**, which tracks project progress, decisions, and contributions from team members.

1.4. Definitions, Acronyms and Abbreviations

These are all the terms Necessary for proper interpretation of SRS:

- **Admin** – The system administrator is responsible for managing users, moderating content, and overseeing platform functionality.
- **Backend** – The server-side logic of the system, handling data processing, authentication, and storage.
- **CSS (Cascading Style Sheets)** – A styling language used for designing the front-end interface.
- **Database** – A structured data storage system used to manage professor ratings, student feedback, and user accounts.
- **Frontend** – The user-facing side of the system that interacts with students, professors, and administrators.
- **HTML5 (HyperText Markup Language 5)** – A markup language used to structure content on the web.
- **Leaderboard** – A ranked list of professors based on student ratings and reviews.
- **MySQL** – A relational database management system used for storing and retrieving data.
- **PHP (Hypertext Preprocessor)** – A backend programming language used for server-side processing.
- **Professor** – A registered faculty member who can view ratings and feedback provided by students.
- **Student** – A user who can search for professors, submit ratings, and leave anonymous reviews.
- **User** – A general term referring to students, professors, or administrators who interact with the system.

1.5. Document Conventions

This document adheres to the **IEEE Software Requirements Specification (SRS) formatting guidelines** to ensure clarity, consistency, and readability. The following conventions have been followed throughout the document:

- All text is written in Times font, size 11 or 12, headings and subheadings have a font size of 18 and 24, respectively for clear Readability.
- Section and subsection titles follow a hierarchical numbering system (e.g., 1, 1.1, 1.2) for structured navigation.
- **Bold text** is used for section headings and key terms.
- The document maintains **single-line spacing** with **1-inch margins** on all sides.

1.6. References and Acknowledgments

1.7. PHP Documentation: https://www.php.net/docs.php
1.8. MySQL Documentation: https://dev.mysql.com/doc/
1.9. HTML5 Standards: https://html.spec.whatwg.org/

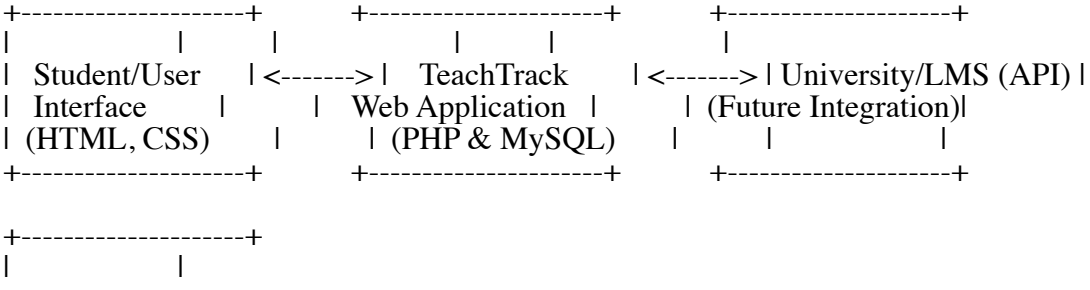
2. Overall Description

2.1. Product Overview

2.1.1. TeachTrack: Product Overview

TeachTrack is a **self-contained web-based platform** that allows students to **rate and review professors** based on various criteria such as **communication skills and course delivery**. The platform features a **student-facing interface**, built using **HTML and CSS**, where students can submit reviews, and an **admin interface** for moderating content to ensure credibility. The backend is developed using **PHP and MySQL**, handling **data management, authentication, and CRUD operations** for reviews and professor records. While TeachTrack is currently an **independent system**, it is designed for **future integration** with university portals or Learning Management Systems (LMS) through **APIs**, enabling enhanced functionalities like **professor data synchronization and authentication via university credentials**. The platform's workflow involves students **submitting ratings**, the **backend processing and storing data**, and **admins moderating reviews** before they are displayed. With **future API-based connectivity**, TeachTrack can evolve into a **scalable academic tool** that provides insights into faculty performance for universities.

2.1.2. System Diagram



| Admin Interface |
| (Moderation) |
| (HTML, CSS, PHP) |
+--------------------+

2.2. Product Functionality-

- **TeachTrack: Product Functionality**
- TeachTrack is designed to provide a **seamless experience** for students to **rate and review professors** while maintaining **moderation and data security**. The platform consists of **three core functional components**:

- **Student/User Interface (Frontend - HTML, CSS)**:
  - Students can **register/login** to access the platform.
  - Ability to **search for professors** and view their ratings.
  - Submit **reviews and ratings** based on multiple criteria (e.g., communication, course delivery).
  - Edit or delete their own reviews within a specified time frame.

- **Backend System (PHP & MySQL)**:
  - Manages **user authentication** (student & admin accounts).
  - Handles **data storage** for professor records, reviews, and ratings.
  - Implements **CRUD operations** (Create, Read, Update, Delete) for reviews and moderation.
  - Ensures **data security** through validation and user role management.

- **Admin Interface (HTML, CSS, PHP)**:
  - Admins can **moderate reviews** by approving, editing, or deleting inappropriate content.
  - View **flagged reviews** and take necessary actions.
  - Manage **professor records** by adding, updating, or removing profiles.
  - Generate **reports/analytics** based on faculty reviews.

2.2.1. Future API Integration Possibilities

- **Sync professor data** with university portals or LMS.
- Enable **authentication via university credentials (SSO integration)**.
- Provide **faculty performance insights** using advanced analytics.

2.3. Design and Implementation Constraints

2.3.1. TeachTrack: Design & Implementation

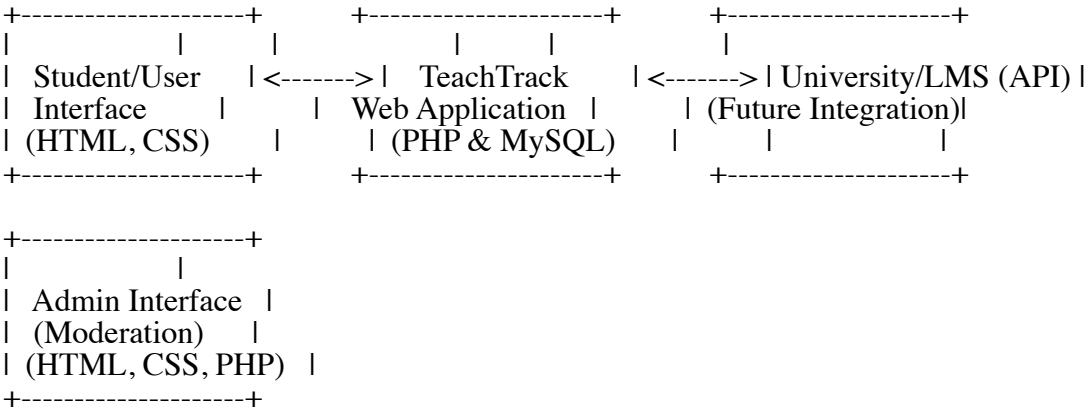
TeachTrack is a **web-based professor rating platform** built using **HTML, CSS (frontend)** and **PHP, MySQL (backend)**. It allows students to **submit and view reviews**, while admins **moderate content** to ensure credibility. The system follows a **three-layer architecture**:

1. **Frontend (HTML, CSS)** – Student and admin interfaces for submitting and managing reviews.
2. **Backend (PHP)** – Handles authentication, data processing, and CRUD operations.
3. **Database (MySQL)** – Stores user details, professor records, and reviews.

2.3.2. Security Features (Beginner-Friendly)

- **SQL Injection Prevention** – Uses **prepared statements** in queries.
- **Password Hashing** – Ensures secure authentication with **password_hash()**.
- **Session Security** – Implements **session_regenerate_id()** to prevent attacks.
- **Input Validation & XSS Protection** – Uses **filter_var()** and **htmlspecialchars()**.
- **Role-Based Access Control (RBAC)** – Restricts features based on user roles.

2.3.3. System Diagram



2.4. Assumptions and Dependencies

Assumptions and Dependencies for TeachTrack:

- **Third-Party Tools and Frameworks**: It is assumed that the selected tech stack—**React, Node.js, and MongoDB**—will remain stable and compatible throughout the project. Any changes or updates to these tools could impact development timelines and performance.

- **APIs for Future Integration**: The project assumes that potential future integration with university portals or Learning Management Systems (LMS) will be feasible through secure API access. Changes in API policies or access restrictions could limit this capability.

- **Security Protocols**: We assume that **SSL/TLS encryption** and current security protocols will suffice to protect user data and communication. Any shifts in security standards could require system upgrades or redesigns.

Assumptions

- Students and admins will use **valid credentials** to access the platform.
- Reviews will be **honest and constructive**, with admins moderating inappropriate content.
- The platform will operate as an **independent system**, with **optional future API integration**.
- Users will have a **stable internet connection** to access the web application.
- PHP and MySQL will be **hosted on a compatible server** for smooth backend operations.

Dependencies

- **Frontend:** HTML, CSS (for UI design).
- **Backend:** PHP (for handling logic and database operations).
- **Database:** MySQL (for storing users, professors, and reviews).
- **Web Server:** Apache/Nginx (to serve the TeachTrack application).
- **Security Measures:** Password hashing, SQL injection prevention, XSS protection.

- **Optional API Integration:** Future dependency on university portals or LMS for authentication and data sync.

3. Specific Requirements

3.1.1.3. Specific Requirements

3.1.1.1. 3.1 External Interface Requirements

3.1.1.1.1. 3.1.1 User Interfaces

- The system will have a **clean and easy-to-use web interface** where students can search for professors, rate them, and leave reviews.
- The site will be **mobile-friendly**, ensuring smooth functionality on desktops, tablets, and smartphones.
- Professors will have a **dashboard** where they can see student feedback but **won't be able to edit or remove reviews**.
- Admins will have a **moderation panel** to manage reported reviews and handle any flagged content.

3.1.1.1.2. 3.1.2 Hardware Interfaces

- The system will be **hosted on cloud platforms** like AWS, GCP, or Azure for better performance and scalability.
- The backend database will be **optimized to handle large amounts of data efficiently**.

3.1.1.1.3. 3.1.3 Software Interfaces

- Secure authentication will be managed using **OAuth 2.0 or JWT tokens**.
- A **relational database (MySQL/PostgreSQL)** will be used to ensure quick searches and efficient data handling.

3.1.1.2. 3.2 Functional Requirements

3.1.1.2.1. 3.2.1 User Management

- **F1**: Students will need to **sign up using their university email** to ensure only verified users can post reviews.
- **F2**: Users can **reset their passwords securely** through an email verification process.
- **F3**: Users will have the option to **delete their accounts and personal data** upon request, in compliance with privacy laws.

3.1.1.2.2. 3.2.2 Professor Rating & Review System

- **F4**: Students will be able to **rate professors** based on different criteria such as:
    - **Teaching clarity**
    - **Grading fairness**
    - **Student engagement**
    - **Course difficulty**
    - **Professor availability**
- **F5**: Each professor's profile will show **a summary of ratings and detailed student reviews**.
- **F6**: Users can **search and filter reviews** by course, rating, or keywords to find relevant feedback easily.
- **F7**: Students will have the ability to **edit or delete their reviews** within a certain timeframe.

3.1.1.2.3. 3.2.3 Moderation & Security

- **F8**: Students can **report inappropriate or fake reviews**, which will be sent for admin review within **24 hours**.
- **F9**: The system will use **automated filters** to detect spam, duplicate reviews, or suspicious activity.
- **F10**: Admins will have the power to **remove flagged content** if it violates the platform's guidelines.
- **F11**: **Role-based access control (RBAC)** will be in place:
    - **Students**: Can submit ratings and reviews.
    - **Professors**: Can view feedback but not edit or delete it.
    - **Administrators**: Can remove inappropriate content and manage reports.
- **F12**: If multiple failed login attempts occur, the system will Say **invalid login details.**

3.1.1.2.4. 3.2.4 System Performance & Reliability

- **F13**: The system will handle **up to 1000 concurrent users** without slowing down.
- **F14**: The website should load within **3 seconds** under normal usage.
- **F15**: Database searches should return results in **under 500 milliseconds**.

3.1.1.2.5. 3.2.5 Scalability & Future Integration

- **F16**: The backend will be structured to **scale efficiently** with increasing users.

- **F17**: Reviews will be **anonymous, but only verified students** will be able to submit them.

4. Other Non-functional Requirements

4.1. Performance Requirements

P1. The website shall load within 3 seconds for users with a standard broadband connection (10 Mbps or higher) under normal traffic conditions.

P2. The system shall support up to 1000 concurrent users without performance degradation. If the number of concurrent users exceeds this limit, the system should auto-scale by deploying additional instances.

P3. Search queries for professors and courses shall return results within 500 milliseconds for a dataset of up to 100,000 professors and 1 million reviews.

P4. The backend API should respond to user requests in less than 300 milliseconds for 90% of requests under typical load conditions.

P5. The system should be optimized to handle high traffic surges, such as during course registration periods, without crashes or excessive delays.

P6. The rating submission feature should allow users to post a review within 2 seconds, even during peak traffic.

P7. Mobile responsiveness should be maintained, with all core functionalities accessible and performing smoothly on Android and iOS devices.

4.2. Safety and Security Requirements

S1. All user passwords shall be hashed and salted before storing them in the database using PBKDF2, bcrypt, or Argon2.

S2. User authentication shall be handled using OAuth 2.0 or JWT tokens to prevent unauthorized access.

S3. The website shall use SSL/TLS encryption to protect user data from man-in-the-middle attacks.

S4. Prevent SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) by implementing secure coding practices and input validation.

S5. Users shall be able to report inappropriate reviews, and flagged content will be reviewed by an admin within 24 hours.

S6. The system shall implement role-based access control (RBAC):

Students: Can submit ratings and comments.

Professors: Can view ratings but cannot modify or delete them.

Administrators: Can remove inappropriate reviews and manage reports.

S7. The system shall log all failed login attempts and notify users of suspicious activity (e.g., multiple failed attempts from different locations).

S8. The system shall comply with GDPR and Indian IT Act regulations, ensuring user data privacy and allowing users to delete their accounts and personal data upon request.

4.3. Software Quality Attributes

4.3.1 Reliability
The system shall have 99.9% uptime, with planned maintenance occurring during non-peak hours.

Implement automated daily database backups to prevent data loss.

A failover system should be in place so that if one server goes down, another takes over automatically.

4.3.2 Usability
The website shall be designed with a simple and intuitive UI, ensuring that first-time users can easily navigate within 2 minutes.

Implement WCAG 2.1 accessibility standards, supporting screen readers and keyboard navigation.

The mobile version shall have a responsive design, ensuring smooth interactions on small screens.

4.3.3 Maintainability
The system shall use a modular architecture, with separate backend and frontend components, allowing for easy upgrades and bug fixes.

All API endpoints and database schemas shall be well-documented.

The codebase shall be structured to follow the MVC (Model-View-Controller) pattern, making it easy to modify or extend features...

4.3.4 Flexibility
The system should support horizontal scaling (adding more servers) to handle increasing loads.

A load balancer should be used to distribute traffic evenly.

The database shall be designed to support sharding and replication if the user base grows significantly.

4.3.5 Testability
All features of the system must be testable through automated unit tests and manual integration testing. We will achieve this by writing unit tests for each module, ensuring that at least 80% code coverage is maintained. Testing environments will be set up to simulate different user loads and scenarios, verifying that the system performs as expected under a variety of conditions. Additionally, continuous integration and deployment (CI/CD) pipelines will be established to automatically run tests and deploy updates safely.

The system shall use a modular architecture, with separate backend and frontend components, allowing for easy upgrades and bug fixes.

4.3.6 Interoperability
The website should be compatible with all major browsers (Chrome, Firefox, Safari, Edge).

The backend should be deployable on cloud platforms such as AWS, GCP, or Azure.

The system should be easily extensible to support a mobile app in the future.

5. Other Requirements

5.1 Database Requirements
The system will use a relational database (e.g., MySQL or PostgreSQL) with optimized read/write operations, encryption for data at rest and in transit, and database indexing for quick access.

5.2 Internationalization Requirements
The system must support multiple languages, with all text stored in configuration files for easy translation. It should also handle time zone and regional formatting differences.

5.3 Legal and Compliance Requirements
The system will comply with data protection regulations (e.g., GDPR), ensuring secure data handling and providing users with access to modify or delete their data.

5.4 Reusability Objectives
Core modules, such as user authentication and the rating system, will be designed for reuse in other university applications, ensuring modularity and scalability.

5.5 Mobile Compatibility
The system must be mobile-friendly and responsive, offering seamless access across devices. A mobile app or PWA can be considered for future development.

5.1.1. Appendix A – Data Dictionary

Variable/ Entity	Description	Type	Possible Values	Related Operations	Requirements
User_ID	Unique identifier for each user (student, professor, admin)	Integer	Auto-incremented	User authentication, account management	Must be unique for each user
User_Type	Defines the role of a user	String	"Student", "Professor", "Admin"	Role-based access control	Determines permissions for users

Email	Email ID for login and communication	String	Valid email format	User registration, login	Must be unique and verified
Password	Encrypted password for authentication	String	Hashed & salted values	Login authentication	Must be securely stored using hashing algorithms
Professor_ID	Unique identifier for each professor	Integer	Auto-incremented	Linking ratings and reviews to professors	Must be unique for each professor
Professor_Name	Full name of the professor	String	Alphanumeric	Displayed on professor profiles	Must match verified faculty records
Course_ID	Unique identifier for a course	Integer	Auto-incremented	Mapping professors to courses	Must be unique per course
Course_Name	Name of the course taught by a professor	String	Alphanumeric	Displayed on course pages	Must match the university's official course list
Review_ID	Unique identifier for each review	Integer	Auto-incremented	Managing user-generated reviews	Must be unique for each review
Review_Text	The content of a student's review	String	User input (max 1000 characters)	Submitted by students, moderated by admins	Must adhere to content guidelines
Rating	Numerical rating given by a student	Float (1 decimal)	1.0 – 5.0	Used to calculate average professor ratings	Must be validated before submission
Timestamp	Date and time of review submission	DateTime	Auto-generated	Used for sorting and moderation	Must be stored in UTC format
Report_Flag	Indicates if a review has been reported	Boolean	True/False	Admins review flagged content	Flagged reviews are subject to moderation
Admin_Action	Action taken on reported reviews	String	"Approved", "Edited", "Deleted"	Moderation of reviews	Actions must be logged for audit
Login_Attempts	Number of failed login attempts	Integer	0 - 10	Used for security enforcement	Account locked after multiple failed attempts
Session_Token	Temporary authentication token for sessions	String	Auto-generated, encrypted	User login and session management	Must expire after a fixed duration
Search_Keyword	Search term entered by a user	String	Alphanumeric	Used for searching professors/courses	Must be processed securely
API_Status	Indicates if API integration is active	Boolean	True/False	Used for future LMS integrations	Controlled by admin settings

Appendix A – Data Dictionary

5.2.Appendix A – Data Dictionary

5.2.1.1. User Management

Variable	Description	Data Type	Possible Values	Operations	Requirements
User_ID	Unique identifier for each user	Integer	Auto-incremented	Create, Read, Update, Delete (CRUD)	Must be unique for each user
User_Type	Defines the type of user	String	"Student", "Professor", "Admin"	Role-based access control (RBAC)	Determines permissions
Email	Email ID for login and verification	String	Valid email format	Registration, authentication	Must be unique and verified
Password	Encrypted user password	String	Hashed & salted values	Authentication, security checks	Must use strong encryption
Session_Token	Temporary authentication token	String	Auto-generated, encrypted	Maintains session after login	Expires after a fixed duration
Login_Attempts	Tracks failed login attempts	Integer	0 - 10	Security enforcement	Account locked after multiple failures

5.2.2. Professor & Course Management

Variable	Description	Data Type	Possible Values	Operations	Requirements
Professor_ID	Unique identifier for each professor	Integer	Auto-incremented	Assign to reviews, link to courses	Must be unique
Professor_Name	Name of the professor	String	Alphanumeric	Displayed in search and review pages	Must match verified faculty records
Course_ID	Unique identifier for a course	Integer	Auto-incremented	Assign to professors	Must be unique per course
Course_Name	Name of the course	String	Alphanumeric	Displayed on course pages	Must match university records

5.2.3.3. Review & Rating System

Variable	Description	Data Type	Possible Values	Operations	Requirements
Review_ID	Unique identifier for each review	Integer	Auto-incremented	CRUD operations	Must be unique
Review_Text	Content of a student's review	String	Max 1000 characters	Submitted, edited, deleted by users	Must follow content guidelines
Rating	Numerical rating given to a professor	Float (1 decimal)	1.0 – 5.0	Affects average professor rating	Must be validated before submission
Timestamp	Date and time of review submission	DateTime	Auto-generated	Sorting and moderation	Stored in UTC format
Report_Flag	Indicates if a review has been reported	Boolean	True/False	Flagging system	Admins review flagged content
Admin_Action	Action taken on reported reviews	String	"Approved", "Edited", "Deleted"	Moderation	Actions must be logged

5.2.4.4. Search & Moderation System

Variable	Description	Data Type	Possible Values	Operations	Requirements
Search_Keyword	Search term entered by a user	String	Alphanumeric	Professor/course search	Must be processed securely
Admin_ID	Unique identifier for administrators	Integer	Auto-incremented	Manage reviews, users	Must be unique
Moderation_Status	Status of review moderation	String	"Pending", "Reviewed", "Removed"	Admin review management	Only admins can modify

5.2.5.5. Security & System Logs

Variable	Description	Data Type	Possible Values	Operations	Requirements
API_Status	Indicates if API integration is active	Boolean	True/False	LMS authentication, data sync	Controlled by admin settings
Security_Log_ID	Unique identifier for security logs	Integer	Auto-incremented	Records security events	Cannot be modified manually
Failed_Attempts	Tracks consecutive failed logins	Integer	0 - 10	Security enforcement	Account locked after threshold
Encryption_Type	Method used for password encryption	String	"bcrypt", "PBKDF2", "Argon2"	User authentication	Must follow security best practices

Appendix B - Group Log

- Abhiram Ayla -Introduction
- Kandala Mahitha Siri – Project Overview

- Aarnav Tandava –Specific Requirements
- Jotsna Sree –other Non-Functional Requirements
- Shreya Seshadri –Other requirements