Software Requirements Specification

for

Educational Platform System

Prepared by

Mohammad Asif Mahmud (2254901005)
Nafees Imtiaz (2254901007)
Gazi Mohammad Abrar Zawad (2254901049)
Jablay Noor Rahman (2254901093)
Prithviraj Chowdhury (2254901101)

October 6, 2025

Contents

Revision History

Revision	Date	Author(s)	Description

Chapter 1 Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the Educational Platform System, a web-based application that consolidates course content delivery, MCQ-based examinations, notices, real-time group chat, and a stationery store into a single portal for Students and Administrators. This SRS provides a shared reference for scope, constraints, and acceptance criteria for all stakeholders.

1.2 Product Scope

1.2.1 Purpose

Provide a unified platform where Students can authenticate, access purchased courses and protected content, take timed MCQ exams with instant scoring, receive and save notices, communicate via group chat, and purchase stationery with tracked receipts. Administrators can create and manage courses, upload contents, author MCQs, post notices, manage stationery inventory, and monitor activity from a dashboard.

1.2.2 Benefits and Objective

- **Efficiency:** One portal for academic consumption, communication, and purchases.
- **Transparency:** Instant exam results, order receipts, and clear notice timelines.
- **Accessibility:** Screen-reader-friendly notices and keyboard-navigable UI.
- **Scalability:** Modular features, pluggable storage/CDN for media.
- **Security:** Role-based access, secure sessions, and protected content links.

1.3 Intended Audience

The SRS is designed to address the needs of a diverse audience involved in the development, deployment, and use of the system. The primary stakeholders include:

- **Students:** the main users who will interact with the portal for academic resources, schedules, exams, notices, chat, purchases, and profile management.
- **Administrators:** responsible for managing course catalogs, contents, MCQs, notices, inventory, and dashboards; also accountable for policy enforcement and audits.

- **Project Managers:** overseeing delivery and ensuring the project aligns with scope, timeline, and quality requirements using this SRS for milestones and acceptance criteria.
- **Developers:** implementing features and APIs in accordance with the functional, interface, and data requirements.
- **QA/Testers:** ensuring quality, performance, and reliability through verification of success/failure criteria and non-functional requirements.

By tailoring information to each group's specific needs, the SRS aims to foster a shared understanding among all stakeholders.

1.4 Intended Use

1.4.1 Students

- Use this SRS to understand supported functions, constraints, and expected behaviors (e.g., timed MCQ exams, receipts).
- Provide feedback referencing requirement IDs when reporting issues or change requests.

1.4.2 Administrators

- Use this SRS to manage catalogs, contents, MCQs, notices, and inventory in line with acceptance criteria.
- Plan operational policies (publishing windows, low-stock thresholds) according to constraints and NFRs.

1.4.3 Project Managers / Developers / QA

- Translate requirement IDs into deliverables, test plans, and release checklists.
- Maintain traceability from user stories to implementation and verification.

1.5 Risk Definition

The Software Requirements Specification identifies the risk that users may not fully adhere to platform policies, terms of use, or ethical guidelines. This non-compliance could lead to issues reported to Administrators, impacting the seamless experience intended for all participants and potentially compromising the overall integrity of the Educational Platform System.

1.5.1 User Inactivity

Risk that users (Students and Administrators) may not actively participate, potentially affecting community-building and the overall success of the platform.

1.5.2 Administrator Workload

Risk that Administrators face high workload handling user issues, enforcing policies, moderating content, and maintaining inventory and catalogs.

1.5.3 Limited User Reference

Risk that users may not refer to the SRS directly, potentially resulting in a disconnect between their expectations and the delivered system.

1.5.4 Changes in Project Scope

Risk that unforeseen changes in scope and requirements may not be promptly reflected in the SRS, leading to potential misguidance for Developers, Testers, and Project Managers.

1.5.5 Security Vulnerabilities

Risk that potential security vulnerabilities may arise, posing a threat to user data, financial transactions, and overall system integrity.

Chapter 2 Overall Description

2.1 User Classes and Characteristics

2.1.1 User Class: Students

Characteristics:

- Primary end users; basic web literacy; access via desktop or mobile browsers.
- Tasks: login, view purchased courses, access contents, take MCQ exams, view/pin notices, chat, purchase stationery, manage profile, view receipts.
- Need quick navigation, accessible notices, and clear progress indicators.

2.1.2 User Class: Administrators

Characteristics:

- Operate course catalog; upload/manage contents; author/publish MCQs; post notices; manage stationery; monitor KPIs.
- Require bulk actions (duplicate course setups), audit logs, alerts, and secure role-based access.

2.2 User Needs

2.2.1 Student Needs

- Fast access to purchased courses and contents with clear progress and last activity.
- Accurate, time-stamped notices; ability to pin/save critical items.
- Simple checkout and searchable purchase history with downloadable receipts.
- Reliable real-time chat for course cohorts; file sharing with limits.
- Transparent, instant scoring for MCQ exams and answer review.

2.2.2 Administrator Needs

- Efficient course creation/duplication, content uploads, and question-bank management.
- Controlled publishing (notices, exams) with versioning and audits.
- Inventory thresholds/alerts and concise dashboard KPIs.
- Reports/exports with minimal manual reconciliation.

2.3 Operating Environment

This section outlines the hardware, software, and network conditions required for the software to function correctly, with a focus on a **browser-based web application** that uses a **client–server MySQL database**.

2.3.1 Hardware Platform

The application runs on standard web infrastructure and its performance depends on the capacity of the server and the user's device.

Servers (Application & Database):

- A cloud or on-premise VM/instance capable of hosting the web server and the MySQL database (or separate instances for each where available).
- Sufficient CPU, RAM, and SSD storage to handle peak usage (e.g., exams, enrollment, order processing).
- Optional load balancer to distribute requests across multiple app nodes as traffic grows.

Client Devices (Browsers):

- Desktop and laptop computers with modern browsers.
- Phones and tablets may access the site using their built-in mobile browsers (no native app required).

2.3.2 Operating System and Versions

The system is designed for a web-first user base and is compatible with the following operating systems.

- **Server OS:** Linux LTS (or equivalent) with routine security updates and time synchronization.
- **Client OS:** Current versions of Windows and macOS (and common Linux distributions) using modern web browsers.

2.3.3 Software Components and Applications

The application operates as a browser–server solution and uses a standalone database server. Unlike an embedded database, **MySQL** runs as a managed server process. The following components are critical for operation:

- **Core Stack:** HTML, CSS, and JavaScript (ES6+) for the front end; REST/JSON endpoints consumed by the browser; MySQL as the relational database.
- **Web Browsers:** Current and previous major versions of Chrome, Firefox, Edge, and Safari.
- **Development/Operations Tools:** A standard code editor (e.g., VS Code), version control (e.g., Git), and a web server/runtime suitable for serving static assets and API endpoints.
- **Email/Notifications:** SMTP or a transactional email service for password resets, receipts, and notice alerts.

- **Monitoring:** Centralized logs and basic uptime/health checks for the web and database services.

2.3.4 Database Compatibility

The application uses MySQL as its primary relational database.

- **MySQL:** A server-based, transactional SQL database used for students, courses, notices, MCQs, orders, and results.
- **Data Access:** Parameterized queries or an ORM/query builder to prevent SQL injection and to maintain portability across MySQL minor versions.
- **Large Files:** Videos and receipts are stored as files in web-accessible storage; the database stores only metadata and secure links.
- **Backups:** Automated daily backups with a verified restore process.

2.3.5 Interoperability

While MySQL is a server database and not embedded, the application exposes APIs for integration with other services.

- The system uses RESTful APIs and JSON data formats for compatibility with third-party services (e.g., payment gateway, email provider) and for internal module communication.
- Optional export endpoints provide CSV/PDF output for reports and receipts.

2.3.6 Network Requirements

The core functionality requires an internet connection because the browser communicates with the web and database servers. However, read-only pages may still render cached content if the browser supports it.

- A stable internet connection is required for login, course access, exams, chat, notices, and purchases.
- Typical campus or home broadband/Wi-Fi is sufficient; Ethernet is recommended for administrative work.
- The system supports both wired and wireless network connections.

2.3.7 Security Considerations

Security is essential for a web app backed by a MySQL database, especially where accounts, grades, and transactions are involved.

- The system employs HTTPS/TLS for all pages and APIs to protect data in transit.
- Passwords and sensitive tokens are stored securely using industry-standard hashing and configuration management (no secrets in client code).
- The application follows best practices (e.g., parameterized queries, CSRF protections, input validation, file-type/size checks, and appropriate Content Security Policy headers) to mitigate common web vulnerabilities.

Chapter 3 Requirements

3.1 Functional Requirements

3.1.1 Login

- As a student or an admin, I want to log in using my registered email and password so that I can securely access my personalized dashboard and continue where I left off.

Success Criteria

- Valid credentials authenticate without full page reload; user lands on role-specific dashboard.
- Session persists until logout or idle timeout; protected routes enforce auth; CSRF defenses active.
- "Forgot password" flow sends a reset link and enforces strong new password rules.

Failure Criteria

- Invalid credentials produce a clear inline error; no navigation occurs.
- After N failed attempts, rate-limit/captcha is enforced.
- Expired sessions are redirected to login with a non-destructive message.

3.1.2 Profile Page (Student)

- As a student, I need to view and edit my personal profile so that I can ensure my information is always accurate and up-to-date.
- As a student, I can change my name, contact number, and bio so that my details are current.
- As a student, I can upload, change, or remove my profile picture so that I can be easily identified by peers and admins.
- As a student, I see a confirmation message and my updated profile immediately after saving changes so that I am confident the changes were saved correctly.

Success Criteria

- Profile edits validate and save; UI reflects changes immediately (no full reload).
- Only allowed image types/sizes accepted; previous photo removed when replaced.
- Audit entry recorded for each personal data change.

Failure Criteria

- Invalid formats or missing required fields show inline errors and block save.

- Unauthorized attempts to edit other users' profiles are prevented and logged.

3.1.3 Purchased Courses View

- As a student, I need to see all my purchased courses in one place so that I can quickly resume my learning without delay.
- As a student, I can see my progress (e.g., 25% completed) for each course so that I can track my learning journey.
- As a student, I can access all course materials (videos, PDFs, quizzes) from a single page within each course so that everything I need is organized and easy to find.
- As a student, I can filter my course library by "In Progress", "Completed", or "Not Started" so that I can easily find the course I want to work on.

Success Criteria

- Library lists title, tags, progress, and last activity; filters and search work as expected.
- Opening a course loads the hub with tabs for Content / Quiz / Notes; last location is remembered.

Failure Criteria

- If no purchases exist, an empty state with a CTA to browse the catalog is shown.
- Attempting to access a course without entitlement is blocked and logged.

3.1.4 Upload Course Content (Admin)

- As an admin, I want to upload learning materials like PDFs, videos, and notes to specific courses so that students can access them anytime after purchase.

Success Criteria

- Upload validates file types (PDF/MP4/allowed); metadata saved (title, course, path, size, uploader, timestamp).
- Students entitled to a course can view/play/download materials from the course hub.

Failure Criteria

- Invalid file types/sizes are rejected with clear feedback.
- Failed uploads roll back atomically; partial records are not left behind.

3.1.5 Create MCQs (Admin)

- As an admin, I need to create, edit, delete, and publish multiple-choice questions (MCQs) with options and correct answers, with automatic grading and instant feedback for students, so that the system can store performance data.

Success Criteria

- Question bank is organized per course/topic; images/math supported where necessary.
- Publishing locks correct answers; later edits require a new version; instant scoring posts to result records.

Failure Criteria

- Publishing prevented unless at least one correct option is set.
- Editing a published question without versioning is blocked.

3.1.6 Exam Page (Student MCQ with Timer)

- As a student, I want to take MCQ exams with a visible timer so that I can test my knowledge and see results instantly.

Success Criteria

- Questions and options load from the database; per-question countdown (60 s) is visible and enforced.
- Timeout auto-moves or auto-submits per exam policy; upon submission, score is calculated and stored; review shows correct vs. chosen answers.

Failure Criteria

- Attempt outside the allowed window is blocked with a clear message.
- Unexpected reloads do not reset time beyond the defined tolerance; attempts are uniquely tracked.

3.1.7 Student-View Notices

- As a student, I want to see all campus notices in one place so that I can stay on top of deadlines and exam changes.
- As a student, I want to view notices by course or department so that I can quickly share relevant updates with my section.
- As an admin, I want the platform to be screen-reader friendly so that students can independently access important information.
- As a commuting student, I want notices with clear date and time so that I can make informed decisions.
- As an admin, I want students to be able to save or pin notices so that they can revisit them when needed.

Success Criteria

- Notice feed supports sort by recency and filters by course/department/category.
- Each notice card shows title, summary, publisher, date/time, and attachments/tags; pin/save creates a personal Saved entry.
- Notice list and filters are keyboard navigable with ARIA semantics validated.

Failure Criteria

- Notices missing required metadata cannot be published.
- Students cannot see notices outside their visibility scope (if scoping is configured).

3.1.8 Create Notices (Admin)

- As an admin, I need to create, edit, delete, and publish notices with a title, description, category, and publish date so that students stay updated about exams, events, or important announcements.

Success Criteria

- Draft → Publish workflow; edits are versioned; delete is soft-delete with restore window.
- Publishing triggers in-app/email notifications as configured and updates the student dashboards instantly.

Failure Criteria

- Publishing is blocked if required fields are missing or date is invalid.
- Unauthorized admins cannot modify or delete notices they do not own (without elevated role).

3.1.9 Group Chat (Course Cohorts)

- As a student, I want to send and receive messages in real time so that I can communicate instantly with classmates in the same course.
- As a student, I want to use emojis in my chat so that I can better express emotion and tone.
- As a student, I want to upload and share files in the chat so that I can collaborate on study materials.

Success Criteria

- Real-time delivery with typing indicators and delivery/seen markers; duplicates prevented.
- File share enforces type/size limits; links preview where possible.
- Moderation tools allow admins to delete reported content and mute users in a room.

Failure Criteria

- Temporary network loss re-connects automatically without message duplication.
- Messages from unauthorized users (not in course) are blocked and logged.

3.1.10 Stationery Store (Student)

- As a student, I need to browse and purchase stationery items with details like price and availability, with cart checkout, payment, and order tracking so that I can buy study materials directly from the platform.

Success Criteria

- Catalog provides filters (category, price, availability); item page shows stock and delivery/pickup notes.
- Cart → Checkout → Order confirmation flow completes with an order ID; status timeline updates (Placed → Packed → Ready/Delivered).
- A receipt is generated and stored; student can download it anytime.

Failure Criteria

- Adding out-of-stock items is blocked with a clear message.
- Payment failures roll back the order atomically; user sees guidance to retry.

3.1.11 View Stationery Purchases (Student)

- As a budget-conscious student, I want a searchable purchase history with clear dates and prices so that I can track spending and avoid duplicate buys.
- As a lab/course student, I want itemized details (item, quantity, unit cost, order ID) so that I can submit accurate reimbursements and reports.
- As a forgetful student, I want a quick "already purchased?" check against required supply lists so that I do not re-buy items I own.
- As an admin, I want downloadable digital receipts and monthly spend summaries so that students can manage returns and set a better budget for next term.

Success Criteria

- History table supports search by item/order/date and export of PDF receipts; monthly summary displays totals.
- When browsing required supplies, items previously purchased are flagged with an indicator.

Failure Criteria

- If no purchases exist, an informative empty state with guidance is shown.
- Students cannot access other users' receipts; access control is enforced.

3.1.12 Admin — Create & Manage Courses

- As an admin, I want to create courses with titles, credits, prerequisites, and tags (category/learning outcomes) so that students can discover relevant electives and plan coherent study paths.
- As an admin, I want to edit course data (schedule, capacity) during the term so that information stays accurate without disrupting enrollment.
- As an admin, I want to duplicate prior course setups so that I can roll offerings into the next semester while preserving prerequisites and tags.
- As an admin, I want to archive or retire outdated courses so that the catalog reflects the current curriculum.

Success Criteria

- Course creation validates required fields and relations; duplicate preserves prerequisites/tags.
- Archiving hides courses from catalog while retaining records for audit/reporting.

Failure Criteria

- Invalid or conflicting schedules/capacities are flagged and block publish.
- Deleting a course with enrollments is prevented without an archive flow.

3.1.13 Manage Stationery (Admin)

- As an admin, I need to manage the stationery store's inventory so that students only see items that are available.

- As an admin, I receive a notification when an item's stock falls below the minimum threshold so that I can reorder it before it sells out.
- As an admin, I can add new products, deactivate old ones, or update prices/descriptions so that the catalog stays accurate and relevant.

Success Criteria

- CRUD with validation; deactivated items are hidden from the storefront.
- Low-stock alerts appear in the Admin Dashboard and can be emailed.

Failure Criteria

- SKU conflicts or negative inventory updates are blocked with clear errors.
- Unauthorized changes are prevented and logged.

3.1.14 Admin Dashboard

- As an admin, I want to view a summary of student activities so that I can monitor engagement and platform usage.
- As an admin, I want quick links to important management tools so that I can easily navigate to different sections.

Success Criteria

- Dashboard shows KPIs (active users, enrollments, low-stock items, recent orders, latest notices, recent exam activity) with drill-down links.
- Access obeys role permissions; sensitive widgets hidden from unauthorized admins.

Failure Criteria

- If data sources are unavailable, modules display safe fallbacks without leaking data.
- Insufficient privileges result in an access-denied message and no data exposure.

3.2 Non Functional Requirements

This section defines the quality and operational constraints for the Student Platform System to ensure predictable performance, safety, security, and maintainability across supported environments.

3.2.1 Performance Requirements

- The system should load the dashboard within a few seconds under normal network conditions.
- The system should handle at least 100 concurrent sessions without noticeable degradation.
- Exam timers should continue reliably even if the page is refreshed or the connection briefly drops.

3.2.2 Security Requirements

- All sensitive data (accounts, payments, personal info) must be transmitted using HTTPS/TLS.
- Passwords must be stored securely using industry-standard hashing.
- Users should be automatically logged out after a period of inactivity.

3.2.3 Reliability Requirements

- The system should perform without failure for routine use cases.
- In case of a server failure, data should not be lost and the system should recover quickly.

3.2.4 Availability Requirements

- The system should be available the vast majority of the time, excluding scheduled maintenance.
- During maintenance, students should still be able to access key read-only features where possible (e.g., notices).

3.2.5 Usability Requirements

- The interface should follow consistent UI/UX patterns and be easy to use.
- New users should be able to start using the platform with minimal guidance (within minutes).
- Key actions should be reachable within a small number of clicks/taps.

3.2.6 Portability Requirements

- The web app should run on both Android and iOS devices via modern mobile browsers.
- The system should be compatible with the latest and previous major versions of popular browsers (Chrome, Firefox, Edge, Safari).

3.2.7 Scalability Requirements

- The system architecture should support growth to thousands of users with minimal reconfiguration.
- Additional modules should be easy to integrate without major redesign.

3.2.8 Maintainability Requirements

- The codebase should follow modular design principles to simplify updates and bug fixes.
- APIs and modules should be documented and the code should be well-organized.

3.2.9 Assurance Requirements

- The app should comply with basic accessibility standards so core tasks are usable by all students.
- Support for straightforward, low-friction usage should be considered in every feature design.

Appendix A: Glossary

- RBAC — Role-Based Access Control.
- WCAG — Web Content Accessibility Guidelines.
- CDN — Content Delivery Network.
- ACID — Atomicity, Consistency, Isolation, Durability.