Software Requirements Specification

for

Edubuddy

Prepared by:
Sanskriti Saxena & Arnav Rathi
School of Computer Science & Engineering

August 20, 2025

Version 1.0 approved

Contents

Revision History

Name	Date	Reason For Changes	Version
Sanskriti Saxena Arnav Rathi	20/08/2025	Initial Draft with Student and Teacher Roles	1.0

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for **Edubuddy**, version 1.0. This product is a web-based application designed to be a comprehensive academic ecosystem for both students and teachers. Its purpose is to provide a centralized, reliable, and collaborative platform for sharing educational resources, facilitating peer-to-peer learning, and enabling instructor guidance and material verification.

1.2 Document Conventions

This document adheres to the IEEE 830-1998 standard for SRS documentation. All functional requirements are uniquely identified with a tag of the format `REQ-X` for traceability. The term "The system" refers to the Edubuddy software.

1.3 Intended Audience and Reading Suggestions

This document is intended for the following audiences:

- **Developers (Project Team):** To understand the complete feature set that needs to be implemented.
- **Project Manager:** To oversee the project's scope and progress against the defined requirements.
- **Testers:** To develop test cases and quality assurance plans based on the specified requirements.
- **Stakeholders (Students, Teachers):** To review the proposed functionality and ensure it meets their academic needs.

It is recommended to read the document sequentially, paying special attention to the distinct user classes in Section 2.3 and the feature sets in Section 4, which detail role-specific functionalities.

1.4 Product Scope

Edubuddy aims to create a structured digital environment that enhances the learning and teaching experience. The key goals are:

- To build a central repository for course notes, study guides, and other academic materials.
- To distinguish between peer-generated content and instructor-verified materials, adding a layer of authenticity.
- To provide an organized, searchable, and filterable database of all academic resources.
- To foster a community where students can clarify doubts and teachers can provide authoritative answers.
- To reduce the time and effort required for both students and teachers to find and distribute quality educational content.

1.5 References

1. *Edubuddy GitHub Repository: https://github.com/Sanskriti0109/Edubuddy.*

2. *Wiegers, Karl E. Software Requirements Specification Template, 1999.*

3. *MERN Stack (MongoDB, Express.js, React, Node.js) Official Documentation.*

2 Overall Description

2.1 Product Perspective

Edubuddy is a new, self-contained product that will function as a standalone web application. It is designed to be a multi-user platform with role-based access, serving as a vital connection point between students seeking knowledge and instructors providing it.

2.2 Product Functions

The major functions of the system are:

- Role-based user registration and authentication (Student, Teacher).
- Uploading, managing, and sharing of educational resources by all users.
- Verification and endorsement of content by Teacher users.
- A comprehensive search and filter system for browsing resources.
- A collaborative Q&A forum for doubt clarification.
- User profile management.

2.3 User Classes and Characteristics

User Class	**Characteristics**
Student	The primary consumer of content. Students can register, log in, upload their notes, download resources shared by peers and teachers, and participate in the Q&A forum by asking and answering questions. They are expected to have basic web literacy.
Teacher / Instructor	A privileged user responsible for content quality and guidance. Teachers have all the capabilities of a student but can also upload official course materials, verify the accuracy of student-uploaded notes, and provide authoritative answers in the Q&A forum. This role requires subject matter expertise.

2.4 Operating Environment

- **Client-Side:** The system shall be a responsive web application accessible via modern web browsers (e.g., Google Chrome, Mozilla Firefox, Safari) on both desktop and mobile devices.
- **Server-Side:** The application is developed using the MERN stack (MongoDB, Express.js, React, Node.js) and is designed for deployment on a cloud platform (e.g., Vercel, Heroku, AWS).

2.5 Design and Implementation Constraints

- The project must be developed using the specified MERN stack.
- All development must be version-controlled using Git and hosted on the project's GitHub repository.
- The user interface must be intuitive and responsive.
- The system must implement role-based access control to secure teacher-specific functionalities.
- The application depends on a configured MongoDB instance for data persistence.

2.6 User Documentation

- A detailed `README.md` file in the GitHub repository will provide instructions for project setup, configuration, and deployment.
- User-facing help text or tooltips will be integrated into the UI to guide users.

2.7 Assumptions and Dependencies

- It is assumed that there will be a mechanism (manual or automated) to assign the "Teacher" role to legitimate instructors.
- The system's performance depends on the stability of the hosting provider and the MongoDB service.
- Users are assumed to have a stable internet connection.

3 External Interface Requirements

3.1 User Interfaces

The system will feature a clean and modern Graphical User Interface (GUI). Key screens will include:

- **Homepage:** An overview of the platform and its features.
- **Login/Registration Pages:** Forms for user authentication, potentially with role selection.
- **Resource Dashboard:** A central feed displaying all resources, with clear labels for verified or official content.
- **Upload Form:** A modal or page for uploading files and their descriptions.
- **Q&A Forum:** An interface for posting questions and answers.
- **User Profile Page:** For managing personal information and viewing one's own contributions.
- Teacher users will see additional UI elements, such as a "Verify" button on student posts or special markers on their own content.

3.2 Hardware Interfaces

No direct hardware interfaces are required.

3.3 Software Interfaces

- **Database:** The system will interface with a MongoDB database via the Mongoose ODM for all data operations.
- **Node.js Environment:** The backend relies on the Node.js runtime and various npm packages (e.g., Express, JWT, bcryptjs) for its operation.

3.4 Communications Interfaces

All communication between the client and server will occur over the HTTPS protocol to ensure data encryption and security. The frontend will interact with the backend via a RESTful API.

4 System Features

4.1 System Feature 1: User Account Management

4.1.1 Description and Priority

Provides functionality for user registration, authentication, and session management based on roles. Priority: **High**.

4.1.2 Functional Requirements

`REQ-1` The system shall allow a user to register for an account with a name, email, password, and user role (Student or Teacher).

`REQ-2` The system shall securely hash and salt all user passwords before storage.

`REQ-3` The system shall allow registered users to log in to access the platform.

`REQ-4` The system shall use JSON Web Tokens (JWT) for managing user sessions and securing API endpoints.

`REQ-5` The system shall implement role-based access control, restricting certain actions (like content verification) to Teacher users only.

4.2 System Feature 2: Resource Sharing and Management

4.2.1 Description and Priority

The core functionality allowing users to upload, view, and manage academic resources. Priority: **High**.

4.2.2 Functional Requirements

`REQ-6` Any authenticated user shall be able to upload a resource, providing a title, subject/tag, and the associated file.

`REQ-7` All users shall be able to view and download any shared resource.

`REQ-8` Users shall be able to edit or delete the resources they have personally uploaded.

`REQ-9` Teacher users shall have the ability to ”verify” a student-uploaded resource, marking it as accurate and reliable.

`REQ-10` Resources uploaded by Teachers and resources verified by Teachers shall be visually distinguished in the user interface (e.g., with a badge or special highlight).

4.3 System Feature 3: Search and Discovery

4.3.1 Description and Priority

Enables users to efficiently find relevant academic materials. Priority: **High**.

4.3.2 Functional Requirements

`REQ-11` The system shall provide a search bar to find resources based on keywords in the title and subject/tag.

`REQ-12` The system shall provide filters to narrow down resources.

`REQ-13` A key filter option shall allow users to view only ”Teacher-Uploaded” or ”Verified” resources.

4.4 System Feature 4: Collaborative Q&A Forum

4.4.1 Description and Priority

A dedicated space for students to ask questions and receive answers from peers and instructors. Priority: **Medium**.

4.4.2 Functional Requirements

`REQ-14` Any student user shall be able to post an academic question in the forum.

`REQ-15` Any authenticated user shall be able to post a reply to a question.

`REQ-16` A Teacher user shall be able to mark a specific answer as the ”Verified Answer,” which will be highlighted as the recommended solution.

`REQ-17` Users shall be able to view all questions and their corresponding answers.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- Page load times shall be under 3 seconds on a standard broadband connection.
- Search and filter results shall be returned in under 2 seconds for a typical volume of data.
- The system should support at least 100 concurrent users without a noticeable degradation in performance.

5.2 Safety Requirements

Not applicable for this software.

5.3 Security Requirements

- Role-based access control must be strictly enforced on the server-side to prevent unauthorized actions.
- All user-submitted data must be sanitized to prevent common web vulnerabilities like Cross-Site Scripting (XSS).
- All password and session tokens must be handled securely, following industry best practices.

5.4 Software Quality Attributes

- **Usability:** The user interface must be intuitive and require minimal learning for both students and teachers.
- **Maintainability:** The codebase must be modular, well-commented, and follow consistent coding standards to allow for easy future modifications.
- **Reliability:** The system should be robust, handle errors gracefully, and maintain high availability.
- **Scalability:** The application architecture should be able to handle a growing number of users and resources without major redesign.

5.5 Business Rules

- A user must be authenticated to contribute content (upload resources, ask questions).
- The "Teacher" role can only be assigned through a trusted process (e.g., by an administrator) to maintain the integrity of content verification.
- Users are responsible for adhering to copyright laws when uploading materials.