Software Requirements Specification

for

EduCom

Version 1.0 approved

Anmol Kumari (2212141)
Pawan Mahesh (2212263)

Shaheed Zulfikar Ali Bhutto Institute of Science & Technology (SZABIST)

10/22/2025

Table of Contents

1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 2
1.3 Intended Audience and Reading Suggestions 3
1.4 Product Scope 3
1.4.1 Product Description 3
1.4.2 Purpose 3
1.4.3 Key Benefits: 3
1.4.4 Objectives and Goals: 4
1.4.5 Business Strategy Alignment: 4
1.5 References 4
2. Overall Description 5
2.1 Product Perspective 5
2.2 Product Functions 6
2.3 User Classes and Characteristics 6
2.4 Operating Environment 8
2.5 Design and Implementation Constraints 8
2.5.1 Organizational and Policy Constraints 8
2.5.2 Technical Constraints 9
2.5.3 Security Constraints 9
2.5.4 Development and Maintenance Constraints 9
2.5.5 External Interface Constraints 9
2.6 User Documentation 10
2.6.1 User Manuals 10
2.6.2 Online Help and Tutorials 10
2.6.3 Technical Documentation 10
2.6.4 Documentation Format and Delivery 10
2.7 Assumptions and Dependencies 10
2.7.1 Assumptions 11
2.7.2 Dependencies 12
3. External Interface Requirements 12
3.1 User Interfaces 12
3.1.1 General Interface Characteristics 12
3.1.2 Interface Standards and Conventions 13
3.1.3 Screen Layouts (Logical Description) 13
3.1.4 Error Handling and User Feedback 14
3.1.5 Accessibility Considerations 14
3.1.6 User Interface Components 14
3.2 Hardware Interfaces 15
3.2.1 Server Hardware Interface 15
3.2.2 Client Hardware Interface 15
3.2.3 AI Moderation Hardware Interaction 16

3.2.4 Peripheral Devices (Optional) ..... **Error! Bookmark not defined.**
3.2.5 Summary of Hardware Interfaces 16
3.3 Software Interfaces 16
3.4 Communications Interfaces 18
4. System Features 20
4.1 Sign In 20
4.1.1 Stimulus/Response Sequence 21
4.1.2 Functional Requirements 21
4.2 Course & Community Management 22
4.2.1 Stimulus/Response Sequences 22
4.2.2 Alternate Flow 23
4.2.3 Exception Flow 23
4.2.4 Functional Requirements 23
4.3 Messaging & Chat System 24
4.3.1 Stimulus/Response Sequences 24
4.3.2 Alternate Flow (Anonymous Mode) 25
4.3.3 Functional Requirements 25
4.4 Assignment Management System 25
4.4.1 Stimulus/Response Sequences 26
4.4.2 Functional Requirements 27
4.5 Admin Panel & Management System 28
4.5.1 Stimulus/Response Sequences 28
4.5.2 Functional Requirements 29
4.6 Marketplace System 30
4.6.1 Stimulus/Response Sequences 30
4.6.2 Alternate Flow (Listing Management) 31
4.6.3 Exception Flow 31
4.6.4 Functional Requirements 31
4.7 Notification & Announcement System 32
4.7.1 Stimulus/Response Sequences 32
4.7.2 Alternate Flow (Targeted Announcements) 33
4.7.3 Exception Flow 33
4.7.4 Functional Requirements 33
4.8 Anonymous Feedback & Reporting System 34
4.8.1 Stimulus/Response Sequences 34
4.8.2 Alternate Flow (Feedback Categories) 35
4.8.3 Functional Requirements 35
5. Other Nonfunctional Requirements 35
5.1 Performance Requirements 36
5.2 Safety Requirements 36
5.3 Security Requirements 37
5.4 Software Quality Attributes 38
5.5 Business Rules 38

6. Other Requirements 39
7. Appendix A Glossary ..……41
8. Appendix B……..……..……...…….43

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

To develop a comprehensive **academic communication and transparency platform** that connects **students, faculty, and administrators** within the university, focusing on the **Computer Science (CS)** and **Business Administration (BBA)** departments. The platform — **EduCom** — aims to provide structured academic communication, community interaction, and administrative transparency across defined academic departments. Unlike automated systems, EduCom will utilize **manually extracted schedule and user data** from ZABDESK to ensure accuracy, control, and compliance with university policies. The platform emphasizes **real-time communication**, **anonymity**, and **content transparency**, providing a digital environment that enhances trust, accountability, and accessibility. A defining feature of EduCom is its **Anonymity and Transparency Mechanism**, allowing students to share feedback or report issues anonymously while maintaining visible accountability for teachers and administrators.

Following are Some major functionalities:

- **Departmental Communities:** Separate communication environments for CS and BBA departments, organized by courses and sections.
- **Communication Tools:** Real-time chat, announcements, and discussion forums limited to enrolled users.
- **Anonymity and Transparency System:** Controlled anonymity for users to ensure honest feedback and transparent administrative review.
- **AI-Powered Content Moderation:** Automatic detection and reporting of inappropriate or non-academic content.
- **Polling and Feedback:** Tools for surveys and opinion collection within academic departments.
- **Payment Gateway (Phase 2 Feature):** Secure microtransaction module (Rs1 gift model) for internal student-level exchanges.

Product Version

This SRS applies to EduCom Version 1.0, which includes the core communication, transparency, and anonymity features for web-based use within the CS and BBA departments. Future versions may expand to additional departments and introduce features such as mobile application access, department-wide analytics, or external platform integrations.

Subsystems Covered

- **User Authentication & Role Management:** Secure login for students, teachers, and administrators, with manual data verification from ZABDESK.
- **Departmental Communication:** Controlled discussion channels for each course or section within CS and BBA.

- **Anonymity & Transparency Module:** System enabling anonymous communication while maintaining institutional accountability.
- **AI Moderation System:** Automated detection and flagging of inappropriate or sensitive content.
- **Polling & Feedback Module:** Custom polls and academic surveys to gather department feedback.
- **Payment Gateway Integration:** Secure, session-based transaction handling (no sensitive data stored).
- **Notification System:** Alerts for messages, polls, and updates relevant to each department.

1.2 Document Conventions

This Software Requirements Specification (SRS) follows the IEEE 830-1998 standard for structuring requirements documents. The following conventions and standards are used throughout this document to ensure clarity and consistency:

- **Font and Formatting:**
  - **Headings:** Bold and numbered according to their hierarchy (e.g., 1, 1.1, 1.1.1).
  - **Body Text:** Standard font (Calibri, 11 pt) for readability.
  - **Keywords and Definitions:** Italicized when first introduced.
  - **System Components, Modules, or Interfaces:** Written in **Title Case** and bold for easy identification (e.g., **Login Module**, **Chat Interface**).
- **Requirement Identification:**
  Each requirement is labeled with a unique identifier for traceability, e.g.,
  - FR-01 for Functional Requirements
  - NFR-01 for Non-Functional Requirements
- **Priority Levels:**
  Each requirement includes a priority level:
  - **High (H):** Must be implemented for system success.
  - **Medium (M):** Important but can be implemented after high-priority features.
  - **Low (L):** Desirable but optional or future enhancement.
- **Inheritance of Priorities:**
  Higher-level requirements' priorities are **not automatically inherited** by their detailed sub-requirements. Each requirement has its own explicitly defined priority.
- **Terminology:**
  - The terms "system", "application", and "EduCom" refer to the same software product.
  - The term "user" refers to any actor interacting with the system, including students, teachers, and administrators.

1.3 Intended Audience and Reading Suggestions

- **Developers:** For system architecture, communication flow, and feature implementation.
- **Project Supervisor:** To plan development phases, milestones, and resource allocation.
- **University Administration:** To monitor compliance, data transparency, and operational integrity.
- **Faculty and Students:** To understand how EduCom enhances communication and feedback002E

1.4 Product Scope

1.4.1 Product Description

EduCom is a **web-based academic communication platform** built to improve **inter-departmental communication and feedback transparency** within the **CS** and **BBA** departments of the university. The system provides authenticated access for teachers, students, and administrators — based on **manually gathered schedules** — and facilitates structured, secure, and moderated communication channels. It features **AI-powered content moderation**, **anonymous feedback options**, and a **department-based community model** to ensure effective and responsible communication within academic boundaries.

1.4.2 Purpose

The primary purpose of EduCom is to **create a transparent, secure, and collaborative academic communication environment** within select departments. The system promotes a balance between **anonymity and accountability**, allowing students to voice opinions or report issues safely, while ensuring that communication remains academic and traceable. By focusing on controlled access, verified participation, and AI moderation, EduCom aims to enhance trust and discipline in university communication systems.

1.4.3 Key Benefits:

- **For Faculty:** Efficient and transparent communication with students within defined course sections.
- **For Students:** Safe and moderated communication space with optional anonymity for feedback or reporting.
- **For Administration:** Improved visibility and monitoring of communication activities across departments.

1.4.4 Objectives and Goals:

- Develop a structured communication system for CS and BBA departments.
- Implement controlled anonymity to promote honest, fear-free communication.
- Ensure transparency and accountability through role-based access and monitoring.
- Use AI-based moderation to filter inappropriate or non-academic content.
- Provide scalability for additional departments in future versions.

1.4.5 Business Strategy Alignment:

EduCom aligns with the university's digitalization and transparency initiatives, promoting ethical communication, paperless administration, and data-driven oversight.
By introducing privacy-aware and transparent academic interaction systems, EduCom supports the institution's goal of fostering responsible communication, academic integrity, and student engagement within a modern digital framework

1.5 References

[1] M. Kaur, S. Kumar, and A. Singh, "Design and Implementation of Role-Based Access Control in Web Applications," *International Journal of Computer Applications*, vol. 182, no. 27, pp. 1–6, 2021.

[2] A. Rahman, A. Alamri, and F. Ahmad, "Real-Time Communication Systems Using Node.js and Socket.IO," *Journal of Web Engineering and Technology*, vol. 19, no. 3, pp. 245–260, 2022.

[3] H. Chen, J. Li, and X. Zhou, "Anonymous Feedback Collection Framework for Institutional Systems," *IEEE Access*, vol. 10, pp. 95632–95645, 2022.

[4] T. Nguyen, P. Do, and K. Le, "AI-Based Content Moderation for Online Communication Platforms," *ACM Transactions on Internet Technology*, vol. 23, no. 2, pp. 1–19, 2023.

[5] S. Anwar, M. J. Khan, and N. Hussain, "Developing Secure Educational Communication Platforms Using Role Hierarchies," *Education and Information Technologies*, vol. 28, pp. 15213–15229, 2023.

[6] L. Torres and B. Chowdhury, "Building Smart University Systems with Integrated Chatbots and Notification Services," *IEEE Transactions on Learning Technologies*, vol. 17, no. 2, pp. 120–131, 2024.

[7] A. K. Sharma and M. Patel, "Integrating Email Notifications in Modern Web-Based Communication Systems," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 3, pp. 145–153, 2024.

[8] J. Wang, C. Liu, and H. Lin, "Monitoring and Logging Mechanisms in Distributed Web Platforms," *Journal of Information Systems Engineering*, vol. 21, no. 4, pp. 211– 225, 2023.

P. O'Connor, E. B. Santos, and M. Gomez, "AI-Driven Transparency in University Communication Systems," Computers & Education: Artificial Intelligence*, vol. 5, p. 100181, 2024.*

2. Overall Description

2.1 Product Perspective

EduCom is a standalone, web-based platform for SZABIST that enables real-time academic communication, collaboration, and internal marketplace activities. It includes:

- **Admin Management**: User roles, course allocation, notifications
- **Academic Communication**: Messaging, announcements, assignments
- **Anonymity & Moderation**: Anonymous feedback, AI content filtering
- **Marketplace**: OLX-style buy/sell with PayFast integration

Built with **Node.js**, **Express.js**, **Socket.io**, **React.js**, and **PostgreSQL**, plus a Python-based AI module. It runs independently on SZABIST's network, with optional RESTful API integration.

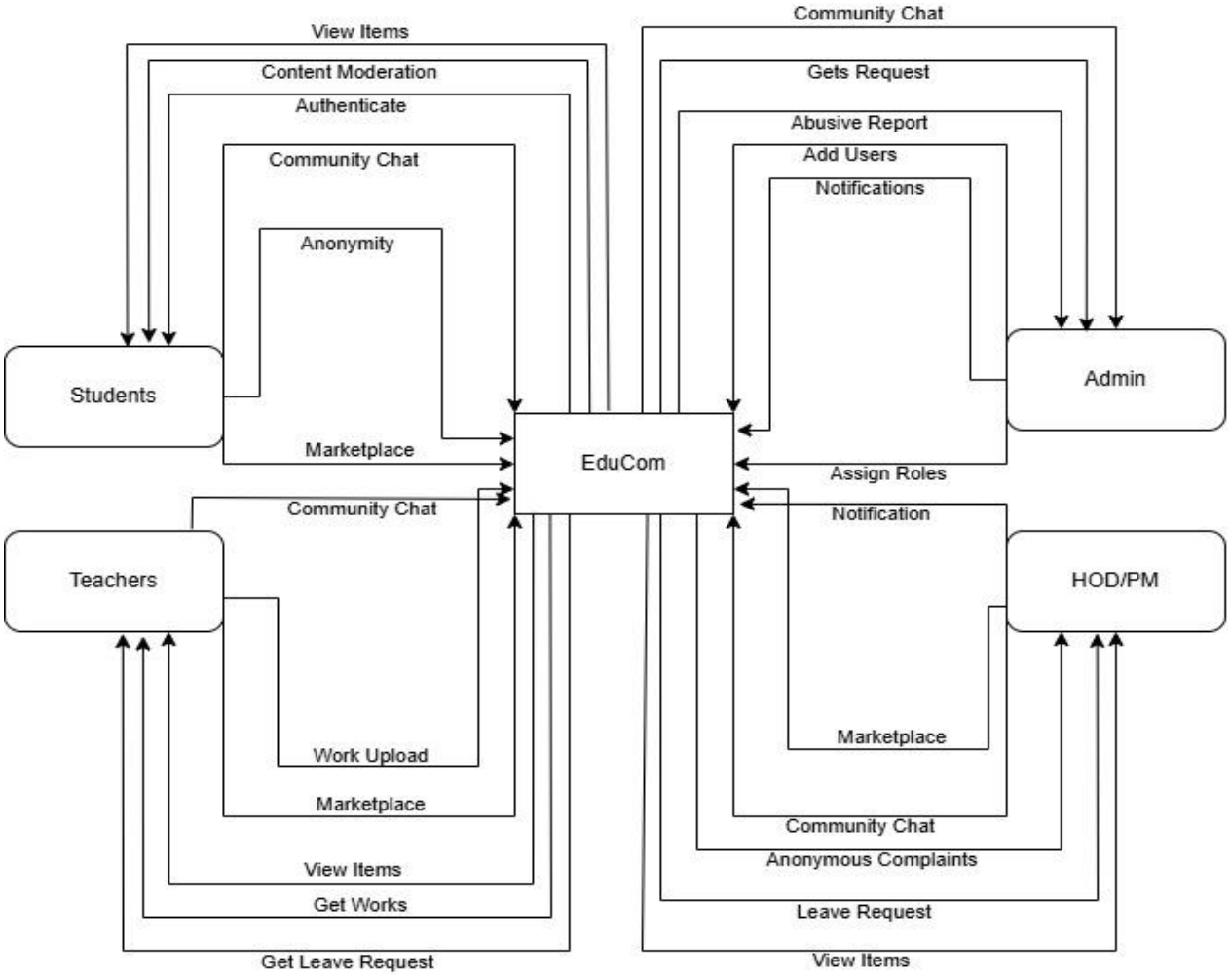

Fig 2.1.0

2.2 Product Functions

At a high level, **EduCom** provides the following core functions:

Admin Functions

- Add, edit, or remove users (students, faculty, staff).
- Assign or modify roles such as Teacher, HOD, or PM.
- Create courses and assign teachers and students.
- Send announcements or notifications to specific or all communities.
- Review flagged or abusive content from the moderation system.
- Approve or remove marketplace listings.

Teacher Functions

- View assigned courses and enrolled students.
- Post announcements, class materials, and assignments with deadlines.
- Communicate with students through real-time chat.
- Receive anonymous feedback from students.
- Review and grade submitted assignments.

Student Functions

- Join course communities automatically upon enrollment.
- Communicate with peers and teachers in real time.
- Submit assignments before the due date.
- Send anonymous feedback or complaints.
- Report abusive or inappropriate messages.
- Access and participate in the EduCom Marketplace.

Marketplace Functions

- Post items for sale (books, notes, devices, etc.).
- Search, view, and purchase items listed by others.
- Conduct secure transactions using **PayFast** integration.
- Receive notifications about approvals, sales, and messages.

Anonymity & Moderation Functions

- Provide optional anonymity for user feedback and complaints.
- Use NLP-based AI to detect offensive or irrelevant content.
- Automatically flag, filter, or block harmful messages.
- Generate abuse reports for admin review.

2.3 User Classes and Characteristics

EduCom will be used by four primary user classes within the SZABIST environment. Each class has distinct responsibilities, privileges, and access levels within the system.

User Class	Description & Responsibilities	Access Level / Privileges	Technical Expertise
Administrator (Admin)	The Admin manages the overall system, including adding and removing users, assigning roles (Teacher, HOD, PM, Student), creating and managing communities, and overseeing the marketplace and moderation reports.	Full system control — can access all modules and user data.	Moderate to high — requires knowledge of data management and basic web system operation.
Teacher / Faculty Member	Teachers are responsible for managing assigned course communities, posting announcements, uploading materials, creating assignments with deadlines, and communicating with students. They can view anonymous feedback and moderate their class discussions.	Access limited to assigned courses and related student communities.	Moderate — comfortable with educational tools and digital platforms.
Student	Students interact within their enrolled course communities, submit assignments, send messages, and provide anonymous feedback. They also access the integrated marketplace for buying or selling items.	Access limited to assigned courses and the public marketplace module.	Basic — general familiarity with mobile and web interfaces.
Head of Department (HOD) / Program Manager (PM)	Senior faculty members who supervise academic activities within a specific program or department. They receive student queries (anonymous or direct), monitor teacher performance, and can post departmental announcements.	Access to departmental communities, student queries, and oversight dashboards.	Moderate — academic staff familiar with communication systems.

Primary Users:

- Admin and Teacher roles are the most critical for system operation and content management.
- Student users represent the largest user base and interact most frequently with the platform.
- HOD/PM users serve in a supervisory capacity and access system functions less frequently but are essential for institutional oversight.

Usage Frequency:

- **Students:** Daily use for communication, assignments, and marketplace access.
- **Teachers:** Regular use (weekly or daily) for course management.
- **Admins:** Continuous or periodic use for user management and monitoring.
- **HOD/PM:** Occasional use for oversight and communication.

2.4 Operating Environment

EduCom is designed to function as a Progressive Web Application (PWA), ensuring cross-platform accessibility and minimal installation requirements. The system operates entirely on web technologies, enabling access through standard browsers across multiple devices.

Hardware Platform:

- **Server Side:**
  - Minimum: 8 GB RAM, Quad-core processor, stable internet connection.
  - Recommended: Cloud-hosted server (e.g., Render, Vercel, or AWS EC2 instance).
- **Client Side:**
  - Any desktop, laptop, tablet, or smartphone capable of running a modern web browser.

Software Environment:

- **Operating System (Server):**
  - Windows Server 2019 or later.
- **Operating System (Client):**
  - Windows, macOS, Android, or iOS.
- **Web Browsers:**
  - Chrome (latest), Firefox, Safari, or Edge — supporting modern JavaScript and PWA features.

Software Components and Dependencies:

- **Frontend Framework:** React.js with CSS.
- **Backend Framework:** Node.js with Express.js.
- **Database:** MongoDB / PostgreSQL for structured data management.
- **Real-Time Communication:** Socket.io for instant messaging and updates.
- **AI Moderation Module:** Python (spaCy / TensorFlow) integrated via RESTful API.
- **Payment Gateway:** PayFast API for secure online transactions.
- **Version Control:** Git & GitHub for collaborative development.

2.5 Design and Implementation Constraints

The development and implementation of **EduCom** are influenced by several organizational, technical, and operational limitations that define the scope, tools, and standards to be followed during the system's design and deployment.

2.5.1 Organizational and Policy Constraints

- The system will operate **exclusively within SZABIST's internal network**, adhering to institutional data protection and IT policies.
- User data (students, teachers, and admin) must be stored and handled according to **institutional privacy guidelines**.
- Integration with third-party services such as **PayFast** must comply with respective **API usage, encryption, and data security policies**.
- All content and communications must remain strictly **academic and professional**, in accordance with SZABIST's communication ethics policy.

2.5.2 Technical Constraints

- The application will be developed using a **fixed technology stack** as approved by the project scope:
  - **Frontend:** React.js, Tailwind CSS
  - **Backend:** Node.js with Express.js
  - **Database:** MongoDB or PostgreSQL
  - **AI Module:** Python (spaCy / TensorFlow)
  - **Real-Time Messaging:** Socket.io
  - **Payment Gateway:** PayFast API
- **Hardware Requirements:**
  - Minimum Server: 8 GB RAM, Quad-core processor, and stable internet connection.
  - Client Devices: Must support a modern web browser capable of running JavaScript and PWA features.
- **Performance Requirements:**
  - Real-time communication should maintain low latency (less than 2 seconds message delivery).
  - The NLP moderation service must process content asynchronously to prevent message delays.

2.5.3 Security Constraints

- All user credentials must be **encrypted** using secure hashing algorithms (e.g., bcrypt) and transmitted only via **HTTPS**.
- The system must implement **role-based access control (RBAC)** to restrict unauthorized access.
- Sensitive payment data must comply with **PayFast's PCI DSS (Payment Card Industry Data Security Standards)**.
- The **AI moderation system** must anonymize and delete temporary message logs after review to ensure data privacy.

2.5.4 Development and Maintenance Constraints

- Developers must follow **modular and RESTful architecture principles** to ensure scalability and ease of maintenance.
- Source code must adhere to **standard naming conventions** and **commenting practices**.
- Version control will be maintained using **Git and GitHub**, with proper branching and commit documentation.
- Only **licensed or open-source dependencies** will be used to prevent future legal or licensing conflicts.
- The software must be maintainable by SZABIST's **internal IT team or future FYP batches**, requiring clear technical documentation and readable code structure.

2.5.5 External Interface Constraints

- The communication between **Node.js backend** and the **AI moderation module (Python)** will use **JSON-based REST APIs**.

- The **PayFast API** will be accessed through **secure HTTPS endpoints** with authentication tokens.
- Any additional third-party integration (e.g., email notifications) must follow **asynchronous communication protocols** to avoid system delays.

2.6 User Documentation

The **EduCom** system will include a comprehensive set of user documentation materials to ensure effective onboarding, usage, and maintenance for all user classes (Admin, Teacher, Student, HOD/PM). Documentation will be delivered in multiple formats for accessibility and ease of understanding.

2.6.1 User Manuals

- **Admin User Manual:** Includes guidelines on user management, role assignments, course creation, community management, and marketplace approvals.
- **Teacher User Manual:** Provides instructions for course management, announcements, assignments, and communication tools.
- **Student User Manual:** Explains login, course access, chat features, assignment submissions, and marketplace use.

Each manual will include:

- Detailed step-by-step procedures
- Screenshots of user interfaces
- Common troubleshooting FAQs

2.6.2 Online Help and Tutorials

- **In-App Help Section:** Integrated "Help" menu within the platform providing on-screen guidance for key features.
- **Tutorial Videos:** Short visual tutorials for onboarding users and explaining major functionalities.
- **Interactive Tooltips:** Built-in guidance for new users highlighting important UI elements and workflows.

2.6.3 Technical Documentation

- **System Architecture Document:** Detailed explanation of API routes, database design, and module dependencies.
- **Installation and Deployment Guide:** Instructions for environment setup, backend installation, and deployment on servers.
- **Maintenance Documentation:** Notes on updating dependencies, handling database migrations, and debugging system issues.

2.6.4 Documentation Format and Delivery

- All user manuals will be provided in **PDF format**
- Technical and installation documentation will be stored in the **GitHub repository** alongside the codebase for developers.

2.7 Assumptions and Dependencies

This section outlines the assumptions and external dependencies that may influence the design, development, deployment, or operation of the **EduCom** system. These factors are expected to remain valid throughout the project lifecycle; any change in them could impact system functionality, timelines, or overall project feasibility.

2.7.1 Assumptions

1. **Stable Internet Connectivity:**
   It is assumed that all users (students, teachers, and administrators) will have reliable internet access to use the EduCom platform effectively, as real-time communication and cloud-based operations depend on constant connectivity.
2. **User Device Compatibility:**
   Users are expected to have access to **modern devices** (PC, laptop, tablet, or smartphone) capable of running updated web browsers supporting JavaScript and Progressive Web Applications (PWA).
3. **Institutional IT Support:**
   The SZABIST IT department will provide necessary technical support for system hosting, deployment, and basic maintenance after initial development.
4. **Data Accuracy:**
   Admin-entered information, including user details, roles, and course assignments, will be accurate and verified. The system relies on this data for role-based access and community formation.
5. **Responsible User Behavior:**
   Users will adhere to institutional communication guidelines. Although AI moderation is implemented, it is assumed that users will generally act responsibly within the platform.
6. **Third-Party Service Availability:**
   External services such as **PayFast API** for payments and **cloud hosting platforms (e.g., Render or Vercel)** will remain operational and maintain backward compatibility with existing integration versions.
7. **Open-Source Tool Stability:**
   It is assumed that open-source frameworks (React.js, Node.js, Express.js, MongoDB/PostgreSQL, Socket.io, and Python libraries for NLP) will continue to receive community or vendor support during the project lifecycle.
8. **Browser and OS Compatibility:**
   Supported web browsers (Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari) will maintain standard compliance with modern JavaScript and PWA technologies.

2.7.2 Dependencies

1. **PayFast API Integration:**
   The payment feature in the EduCom Marketplace depends on PayFast's external API. Any change in their API structure, service availability, or pricing policy may affect marketplace operations.
2. **Cloud Hosting and Server Dependencies:**
   System deployment relies on cloud services (Render, Vercel, or equivalent). Service downtime, policy changes, or cost variations could affect uptime and scalability.
3. **AI Moderation Module (Python-based):**
   The text moderation system depends on Python libraries such as **spaCy** or **TensorFlow**. Changes in their APIs, model availability, or compatibility with Node.js could impact the moderation process.
4. **Email or Notification Services:**
   If the system incorporates external email or push notification APIs (e.g., Nodemailer, Firebase), functionality will depend on the availability and uptime of these services.
5. **Institutional Approval and Usage Policies:**
   System deployment and operation depend on SZABIST's administrative approval, IT infrastructure policies, and data protection standards.
6. **Maintenance and Future Development:**
   Future updates or bug fixes may rely on SZABIST's internal IT department or future FYP teams. Proper documentation and version control are critical for continuity.

3. External Interface Requirements

3.1 User Interfaces

The **EduCom** platform will provide a **web-based graphical user interface (GUI)** accessible through modern browsers on both desktop and mobile devices. The interface will be **intuitive, consistent, and responsive**, designed to ensure ease of use for all user roles — Admin, Teacher, Student, and HOD/PM. The system will follow a **uniform layout design** across all pages, ensuring visual consistency, accessibility, and compliance with SZABIST's institutional branding.

3.1.1 General Interface Characteristics

- **Design Framework:**
  The frontend will be developed using **React.js** and styled with **Tailwind CSS**, ensuring responsive design and fast rendering.
- **Interface Structure:**
  The system will follow a **modular dashboard layout**, where each user role sees specific menus, widgets, and components according to their permissions.
- **Navigation:**
  A **side navigation bar** will be present on all dashboards containing links to core modules (e.g., Dashboard, Courses, Messages, Marketplace, Notifications).

- **Top Bar (Header):**
  Displays the logged-in user's name, profile picture, notifications, and a quick-access "Help" icon.
- **Footer (Optional):**
  Displays copyright notice, institutional contact information, and version number.
- **Color Scheme & Branding:**
  Consistent with SZABIST's theme (e.g., blue, white, and gray palette). Icons and UI components will follow modern flat design standards.

3.1.2 Interface Standards and Conventions

- **Font Family:** Sans-serif fonts such as Inter or Poppins for readability.
- **Navigation Controls:** Persistent sidebar for major sections; breadcrumbs for sub-navigation.
- **Standard Buttons:**
  - **Primary Actions:** "Submit", "Send", "Upload", "Approve".
  - **Secondary Actions:** "Cancel", "Edit", "Delete".
- **Confirmation Dialogs:** Required for all destructive actions (e.g., user deletion, post removal).
- **Error/Success Messages:** Displayed via toast notifications or modals at the top-right corner of the screen.
- **Help Button:** Present on every page, linking to in-app help documentation or tutorials.
- **Responsive Design:** Layout auto-adjusts for mobile, tablet, and desktop views.

3.1.3 Screen Layouts (Logical Description)

- **1. Admin Dashboard Interface**
- **Sections:** User Management, Role Assignment, Course Creation, Notifications, Marketplace Moderation, and Reports.
- **Features:**
  - Search bar for locating users or communities.
  - Data tables for user and course management.
  - Buttons for "Add", "Edit", "Delete", and "Assign Role".
  - Graphical overview of system activity (e.g., number of users, flagged messages, transactions).

2. Teacher Dashboard Interface

- **Sections:** My Courses, Assignments, Announcements, Chat, Feedback.
- **Features:**
  - Course cards displaying course name, semester, and student count.
  - Chat window with real-time messaging (Socket.io).
  - Assignment creation form with deadline picker.
  - Anonymous feedback panel (hidden sender information).

3. Student Dashboard Interface

- **Sections:** My Communities, Assignments, Messages, Marketplace, Notifications.
- **Features:**
  - Real-time community chat interface.
  - File upload option for assignment submissions.
  - Marketplace tab for browsing or posting items.

- Anonymous feedback form with "Send" button (no sender info stored on UI).

4. HOD/PM Interface

- **Sections:** Departmental Overview, Feedback Reports, Faculty List, Announcements.
- **Features:**
  - View and respond to anonymous student feedback.
  - Monitor class performance and teacher activity.
  - Send departmental messages or updates.

5. Marketplace Interface

- **Sections:** Browse Items, My Listings, Sell Item, Transaction History.
- **Features:**
  - Product cards showing title, price, and seller info.
  - "Buy Now" and "Chat with Seller" buttons.
  - Integration with PayFast payment gateway.
  - Admin approval badge for verified listings.

3.1.4 Error Handling and User Feedback

- All forms will include **input validation** (e.g., email format, required fields, file size limits).
- Invalid inputs trigger **inline error messages** in red text below the respective field.
- Successful actions (e.g., message sent, item listed) will trigger **green toast notifications**.
- In case of server errors or connectivity issues, a **standard error modal** will inform the user and suggest retry options.

3.1.5 Accessibility Considerations

- The interface will comply with **WCAG 2.1 accessibility standards** where possible.
- High-contrast color themes and scalable text will ensure readability for all users.
- Keyboard navigation and ARIA labels will be supported for accessibility tools.

3.1.6 User Interface Components

Component	Used By	Purpose
Dashboard Layout	All users	Central navigation and overview
Chat Window	Students, Teachers	Real-time messaging
Assignment Form	Teachers	Create tasks with deadlines
Submission Upload	Students	Submit assignments before deadlines
Marketplace Panel	All users	Buy/sell products via PayFast
Moderation Panel	Admin	Review flagged messages
Notification Center	All users	View system-wide or class notifications
Anonymous Feedback Form	Students	Send feedback anonymously

3.2 Hardware Interfaces

The **EduCom** platform is a **web-based application** that primarily operates within a **client-server architecture**. It interacts with standard hardware components used by both end users and the hosting environment. Since the system is implemented as a **Progressive Web Application (PWA)**, it is **device-independent**, requiring only compatible hardware capable of running a modern web browser.

3.2.1 Server Hardware Interface

The EduCom backend services will be hosted on a **dedicated or cloud-based server**, which provides processing, data storage, and communication functionality for all client devices.

a) Physical Characteristics:

- Minimum 8 GB RAM and Quad-core processor.
- Minimum 100 GB SSD storage.
- High-speed internet connection (minimum 50 Mbps upload/download).
- Secure hosting with SSL certificates for encrypted communication.

b) Logical Characteristics:

- The server handles HTTP/HTTPS requests from clients and returns responses using RESTful APIs.
- Real-time communication occurs through **WebSocket connections (Socket.io)**.
- The server communicates with the **AI moderation module** (Python) via local or remote REST API calls.
- Payment interactions are handled through secure connections with the **PayFast API** using HTTPS.

c) Communication Protocols:

- **HTTP/HTTPS** for general web communication.
- **WebSocket (WSS)** for real-time chat and notifications.
- **TCP/IP** as the fundamental network protocol.
- **SSL/TLS** for secure data transmission.

3.2.2 Client Hardware Interface

The client-side interface operates through **desktop and mobile devices** using a compatible web browser.

a) Supported Device Types:

- Desktop Computers and Laptops
- Tablets
- Smartphones (Android and iOS)

b) Minimum Hardware Requirements:

- **Processor:** Dual-core CPU or higher.
- **Memory:** Minimum 2 GB RAM.
- **Storage:** Minimum 200 MB available space for browser cache and offline PWA data.
- **Display:** Minimum 720p resolution (responsive layout for smaller screens).
- **Network:** Stable internet connection (minimum 1 Mbps).

c) Control and Data Interaction:

- Users interact with the system through mouse, keyboard, or touchscreen input.
- Uploaded files (assignments, images, marketplace listings) are transmitted securely to the server.

- Notifications and chat messages are delivered through **WebSocket** channels in real time.
- Browser-based local storage may temporarily hold user session tokens for seamless navigation.

3.2.3 AI Moderation Hardware Interaction

- The AI moderation service runs as a **Python-based microservice** either on the same host machine or a separate server.
- Interactions occur via **REST API calls** — text data is sent to the AI module, which returns a filtered or approved response.
- This process does not require specialized hardware but benefits from **GPU acceleration** when using large NLP models (optional).

3.2.4 Summary of Hardware Interfaces

Component	Type	Interaction	Protocol/Interface
Application Server	Cloud / Local Machine	Hosts backend APIs and manages client requests	HTTP/HTTPS, WebSocket
Client Devices	Desktops, Laptops, Tablets, Smartphones	Access EduCom PWA via browsers	HTTPS
Database Server	Cloud/Local	Stores user data, chats, and logs	MongoDB/PostgreSQL driver
AI Moderation Module	Local/Remote Microservice	Filters and classifies messages	REST API (JSON)
Payment Gateway (PayFast)	External API	Handles secure transactions	HTTPS (SSL/TLS)

3.3 Software Interfaces

3.3 Software Interfaces

The **EduCom** platform integrates several software components to provide a **secure**, **real-time**, and **scalable** academic communication and marketplace system. Each component works cohesively to manage user interactions, AI moderation, and online payment services.

Database

PostgreSQL serves as the sole data storage system, ensuring data consistency, integrity, and reliability. It stores structured information for all system entities, maintaining relational links between users, courses, communities, and transactions.

Key Data Stored:

- User accounts, roles, and permissions
- Course details and assigned teachers/students
- Messages, announcements, and anonymous feedback
- Assignments, submissions, and deadlines
- Marketplace items, listings, and transaction logs

Integration:
The backend uses the **Sequelize ORM** for database management, schema definition, and query optimization, allowing efficient interaction with PostgreSQL.

Backend Framework

The backend is built using **Node.js (v20)** and **Express.js**, ensuring a modular, event-driven architecture optimized for real-time communication and API routing.

Key Modules and Libraries:

- **Sequelize:** Database ORM for PostgreSQL
- **JWT (JSON Web Token):** Secure, stateless user authentication
- **bcrypt.js:** Password encryption and hashing
- **Socket.io:** Enables real-time chat and notification delivery
- **Nodemailer:** Sends email alerts, password resets, and announcements
- **Axios:** Handles asynchronous API communication

Responsibilities:

- Manage business logic, authentication, and authorization
- Handle CRUD operations for users, courses, messages, and marketplace items
- Communicate with the AI moderation module and PayFast API via RESTful endpoints

Frontend Framework

Developed using **React.js** and **Tailwind CSS**, the frontend provides a dynamic, responsive, and accessible user experience. It interacts with backend services using **RESTful APIs** for standard operations and **WebSocket (WSS)** for live communication.

Key Features:

- Built as a **Progressive Web Application (PWA)** for accessibility across devices
- Uses **Context API / Redux** for state management
- Exchanges all data in **JSON format** for consistency and efficiency

AI Moderation Service

EduCom integrates a **Python-based AI module** developed using **spaCy** or **TensorFlow** to analyze user messages and feedback content in real time.

Integration Details:

- Connected to the Node.js backend via **REST API (HTTPS)**
- Receives JSON data (text content, sender role, timestamp)
- Returns moderation results (approved, rejected, or flagged)
- Operates asynchronously to preserve real-time message flow

Purpose:
To automatically detect and filter inappropriate or non-academic content, ensuring a professional communication environment.

Payment Gateway (PayFast Integration)

EduCom integrates the **PayFast API** to facilitate secure and authenticated transactions in the internal marketplace.

Key Features:

- Supports product purchases and sales between users
- Uses **HTTPS (SSL/TLS)** for all payment requests
- Returns payment confirmation tokens and transaction statuses

- No sensitive card or financial data is stored within EduCom — all payment handling occurs on PayFast's secure servers

Integration:

The system exchanges transaction data (item ID, amount, user ID, and redirect URLs) through secure JSON-based API calls to PayFast.

Data Exchange

All communication between EduCom's components follows a **JSON-based data exchange model**, ensuring interoperability and security.

Data Exchanged:

- **Authentication:** Login credentials and JWT tokens
- **Messages:** Text, file links, and timestamps
- **Assignments:** Submission metadata and deadlines
- **Marketplace:** Product details and transaction confirmations
- **Feedback:** Anonymous reports and AI moderation results

Configuration & Security

Configuration data and environment variables are securely managed through .env files. Security measures are enforced across all layers of the system.

Key Security Features:

- Passwords encrypted with bcrypt.js
- JWT-based authentication for all sessions
- HTTPS enforcement for all endpoints
- Role-Based Access Control (RBAC) for user segregation
- RESTful API architecture for scalability and maintainability

3.4 Communications Interfaces

The **EduCom** platform relies on multiple communication interfaces to support **real-time messaging**, **data exchange**, **notifications**, and **secure online transactions**. These communication functions ensure seamless interaction between users, the backend system, the AI moderation service, and the integrated payment gateway.

Web Communication

EduCom operates as a **web-based platform**, communicating through standard **HTTP/HTTPS protocols**. All interactions between the frontend and backend occur over **RESTful APIs**, while **WebSocket (WSS)** is used for real-time data transmission.

Standards and Protocols:

- **HTTP/1.1 and HTTPS (TLS/SSL encryption):** Used for all API and form submissions.
- **WebSocket (WSS):** Enables real-time, bi-directional communication for chat, notifications, and live updates.
- **JSON (JavaScript Object Notation):** The standard data format used for all request and response payloads.

Purpose:

To deliver fast, reliable, and encrypted communication between user interfaces and backend services.

Messaging and Real-Time Communication

EduCom uses **Socket.io** (built on WebSocket protocol) to manage **instant messaging** and **real-time notifications** among users and system components.

Functional Requirements:

- Supports one-to-one and group chat within course communities.
- Maintains live synchronization of chat messages and notifications.
- Uses **event-based communication** for broadcasting updates (e.g., assignment posted, new feedback received).

Security Measures:

- All WebSocket connections operate over **WSS (Secure WebSocket)** to ensure encrypted real-time data flow.
- User sessions are verified through **JWT tokens** before establishing a socket connection.

Email Communication

EduCom integrates **Nodemailer** for sending automated emails to users for system alerts, account verification, and password resets.

Email Features:

- Sends HTML-formatted emails for clarity and branding.
- Used for:
    - Password reset links
    - Account registration confirmation
    - Administrative notifications
    - Marketplace transaction confirmations

Protocols:

- **SMTP (Simple Mail Transfer Protocol)** is used for all outgoing email communication.
- Email credentials and API keys are stored securely in environment variables (.env).

AI Moderation Communication

The AI-based content moderation module communicates with the Node.js backend through **RESTful API requests**.

Communication Details:

- **Protocol:** HTTPS (secured REST API)
- **Request Format:** JSON { "text": "message content", "role": "student", "timestamp": "..."}
- **Response Format:** JSON {"status": "approved/flagged/rejected", "reason": "detected keyword/offensive content" }
- Operates asynchronously to prevent delays in real-time chat.

Security:

All requests between the backend and AI service are authenticated using API keys and transmitted over SSL/TLS.

Payment Communication

EduCom integrates with **PayFast API** for secure online payment processing within the internal marketplace.

Communication Flow:

1. EduCom sends a **payment request** containing order details (amount, user ID, item ID, redirect URLs) to PayFast via HTTPS.
2. PayFast processes the transaction securely on its platform.

3. A **confirmation callback** is sent back to EduCom with the transaction reference and payment status.

Standards and Protocols:

- **HTTPS (SSL/TLS):** Encrypted communication between EduCom and PayFast.
- **RESTful API:** JSON-formatted request and response handling.

Security Requirements:

- EduCom must not store or log sensitive financial data.
- Only payment confirmation and transaction IDs are retained in the database.

Data Encryption and Security

All communication within EduCom must follow **end-to-end encryption** and **data integrity** standards.

Encryption Protocols:

- **TLS/SSL:** For securing all HTTP, WebSocket, and API traffic.
- **bcrypt:** For encrypting user passwords before database storage.
- **JWT (JSON Web Token):** For secure, stateless session management.

Security Objectives:

- Protect user credentials and private messages.
- Prevent unauthorized data interception or modification.
- Ensure confidentiality of all payment and feedback transactions.

Synchronization Mechanisms

- **Socket.io events** synchronize real-time updates (messages, assignment submissions, notifications).
- **Database timestamps** ensure ordered message delivery and accurate history tracking.
- **Automatic retry mechanisms** re-establish socket connections on network interruptions.

Summary of Communication Standards

Communication Type	Protocol / Standard	Data Format	Purpose
Web APIs	HTTPS / RESTful	JSON	General data exchange
Real-Time Messaging	WebSocket (WSS)	JSON	Live chat and notifications
Email Service	SMTP	HTML / Plain Text	System alerts and password resets
AI Moderation	HTTPS (REST API)	JSON	Message filtering and classification
Payment Gateway	HTTPS (PayFast API)	JSON	Secure online transactions

4. System Features

4.1 Sign In

Section	Details
Feature Name:	**Sign In**
Description:	The Sign In feature allows all authorized users (Admin, Teachers, Students, HODs, and PMs) to securely log in to the EduCom platform using their credentials assigned by the Admin. This feature ensures authenticated access and role-based dashboard redirection.
Priority:	**High**

Actors:	Admin, Teacher, Student, HOD, PM
Goal:	To authenticate registered users and provide access to their respective dashboard interfaces.
Pre-Conditions:	1. User must already exist in the system (created by Admin). 2. User must have valid login credentials (email or registration ID and password).
Post-Conditions:	1. User is successfully logged in and redirected to their dashboard. 2. A secure JWT token is generated to maintain an active session.

Table 4.1

4.1.1 Stimulus/Response Sequence

Steps	User Action	System Response
1	User navigates to the home page and clicks the **"Sign In"** button.	The system displays a login form requesting credentials (Email/Reg ID and Password).
2	User enters valid credentials and clicks **"Login"**.	The system validates input and checks credentials in the PostgreSQL database.
3	Credentials are valid.	The system generates a secure JWT token, authenticates the user, and redirects to the appropriate dashboard (Admin/Teacher/Student).
4	Credentials are invalid	The system displays an error message: "Invalid credentials. Please try again."
5	Network or server failure	The system displays: "System temporarily unavailable. Please try again later."
6	User submits empty fields or incorrect data.	The system highlights invalid inputs and shows: "Please enter valid credentials."

Table 4.1.1

4.1.2 Functional Requirements

Req ID	Description	Priority	Status
REQ-1	The system shall display a secure login form with fields for Email/Reg ID and Password.	High	Pending
REQ-2	The system shall validate user credentials against stored data in PostgreSQL.	High	Pending
REQ-3	The system shall authenticate users using JWT-based authentication.	High	Pending
REQ-4	The system shall redirect users to their respective dashboards based on their assigned roles.	High	Pending

REQ-5	The system shall display appropriate error messages for invalid or missing credentials.	High	Pending
REQ-6	The system shall maintain user sessions using secure JWT tokens until logout or expiration.	Medium	Pending
REQ-7	The system shall restrict access to unauthorized users and expired sessions.	Medium	Pending

Table 4.1.2

4.2 Course & Community Management

Section	Details
Feature Name:	**Course & Community Management**
Description:	This feature allows Admins and Teachers to manage academic courses and their corresponding communities. When a teacher is assigned to a course, a dedicated community is automatically generated. All students enrolled in that course are added to the respective community. Within each community, members can communicate, share resources, and collaborate in real time.
Priority:	**High**
Actors:	Admin, Teacher, Student
Goal:	To automate the creation and management of course-based communities where teachers and students can interact, collaborate, and share academic materials.
Pre-Conditions:	1. Users (teachers and students) must be added to the system by the Admin. 2. Courses must exist in the database.
Post-Conditions:	1. Communities are automatically created and linked to assigned courses. 2. Users can access only their enrolled course communities. 3. Any updates (add/remove user, change teacher) reflect instantly in the community.

Table no 4.2

4.2.1 Stimulus/Response Sequences

Step	User Action	System Response
1	Admin assigns a teacher to a course and adds students to that course.	The system automatically generates a community for that course.
2	Teacher logs in and accesses "My Courses."	The system displays a list of courses assigned to the teacher, each linked to its respective community.
3	Student logs in and accesses "My Communities."	The system displays course-based communities the student is enrolled in.
4	Teacher posts a message or upload in a community.	The system broadcasts it to all enrolled students in real time.

5	Admin modifies course assignment (adds/removes student or teacher).	The system updates the corresponding community membership dynamically.

Table 4.2.1

4.2.2 Alternate Flow

User Action	System Response
Admin edits community members manually (e.g., removing a user).	The system updates the member list and reflects changes immediately.
Teacher archives a completed course.	The system marks the community as inactive and restricts posting.

Table 4.2.2

4.2.3 Exception Flow

Condition	System Response
Admin assigns a course to a non-existing teacher/student.	The system displays: "User not found. Please add user before assigning course."
Database update failure during community creation.	The system shows: "Error creating community. Please try again later."

Table 4.2.3

4.2.4 Functional Requirements

Req ID	Description	Priority	Status
REQ-8	The system shall allow Admin to assign teachers and students to specific courses.	High	Pending
REQ-9	The system shall automatically create a course community when a new course assignment is made.	High	Pending
REQ-10	The system shall allow teachers and students to view only the communities related to their assigned courses.	High	Pending
REQ-11	The system shall allow real-time posting and message exchange within each community.	High	Pending
REQ-12	The system shall allow Admin to update course and community memberships dynamically.	Medium	Pending
REQ-13	The system shall restrict access to non-enrolled users attempting to enter a course community.	High	Pending
REQ-14	The system shall store all messages and community interactions securely in the database.	High	Pending
REQ-15	The system shall allow archiving or deactivating a community once a course concludes.	Medium	Pending

Table 4.2.4

4.3 Messaging & Chat System

Section	Details
Feature Name:	**Messaging & Chat System**
Description:	The Messaging & Chat System enables real-time communication between students and teachers within their respective course communities. It includes features such as anonymity, content moderation, and AI-based message filtering to maintain a professional environment. Messages are exchanged securely using WebSocket connections and stored in the PostgreSQL database for recordkeeping.
Priority:	**High**
Actors:	Teachers, Students, Admin
Goal:	To facilitate secure, moderated, and real-time communication between course participants while preserving academic and professional integrity.
Pre-Conditions:	1. User must be authenticated through the Sign In feature. 2. User must be part of an existing course community. 3. WebSocket connection must be active.
Post-Conditions:	1. Messages are delivered in real time to all members of the community. 2. Messages flagged as inappropriate are filtered or reported to Admin. 3. Chat history is stored securely for future reference.

Table 4.3

4.3.1 Stimulus/Response Sequences

Step	User Action	System Response
1	User opens the community chat interface.	The system fetches and displays recent chat messages from the database.
2	User sends a message (anonymous or identified).	The message is transmitted through a secure WebSocket connection to the server.
3	System receives the message.	The system sends the message text to the AI moderation module for filtering.
4	AI module evaluates the message.	The AI returns a decision (Approved / Flagged / Rejected).
5	Message approved.	The system broadcasts it in real time to all community members.
6	Message flagged or rejected.	The system prevents the message from being posted and optionally notifies Admin.
7	User receives new messages.	Messages are pushed to all connected clients instantly.
8	Admin receives report of abusive or flagged content.	Admin reviews the message in the moderation panel.

Table 4.3.1

4.3.2 Alternate Flow (Anonymous Mode)

User Action	System Response
User chooses to send a message anonymously.	The system hides the sender's name and replaces it with a generic identifier (e.g., "Anonymous"). Only the system logs retain the sender's identity for security purposes.

Table 4.3.2

4.3.3 Functional Requirements

Req ID	Description	Priority	Status
REQ-16	The system shall enable real-time messaging using WebSocket technology.	High	Pending
REQ-17	The system shall integrate an AI moderation service to filter inappropriate or offensive content.	High	Pending
REQ-18	The system shall allow users to send messages anonymously within a community.	High	Pending
REQ-19	The system shall store all chat messages securely in the PostgreSQL database.	High	Pending
REQ-20	The system shall notify the Admin of messages flagged by the moderation system.	Medium	Pending
REQ-21	The system shall maintain chat history and retrieve recent messages upon community load.	Medium	Pending
REQ-22	The system shall display real-time message updates for all active users.	High	Pending
REQ-23	The system shall prevent users not enrolled in a course from accessing its chat.	High	Pending
REQ-24	The system shall support automatic reconnection during temporary network loss.	Medium	Pending
REQ-25	The system shall ensure that all message transmissions are encrypted and secure.	High	Pending

Table 4.3.3

4.4 Assignment Management System

Section	Details
Feature Name:	**Assignment Management System**
Description:	The Assignment Management System enables teachers to create, distribute, and manage coursework within each course community. Students can view assignments, submit their work before the deadline, and receive notifications

	regarding their submissions. Late submissions are automatically restricted after the deadline expires. This feature provides transparency, automation, and accountability for both teachers and students.
Priority:	**High**
Actors:	Teacher, Student
Goal:	To facilitate seamless creation, submission, and tracking of assignments with deadline-based control within each course community.
Pre-Conditions:	1. The teacher must be assigned to a course. 2. Students must be enrolled in the corresponding course community. 3. The user must be authenticated via the Sign In feature.
Post-Conditions:	1. Assignments are created and visible to all enrolled students. 2. Submitted assignments are stored securely in the system. 3. Students cannot submit or edit work after the submission deadline.

Table 4.4.1.1

4.4.1 Stimulus/Response Sequences

Step	User Action	System Response
1	Teacher navigates to the "Assignments" tab in the course community.	The system displays the assignment creation form with fields for title, description, file attachments, and deadline.
2	Teacher enters assignment details and clicks **"Post Assignment."**	The system saves the assignment and displays it in the course community for all students.
3	Student logs in and opens the course community.	The system displays the list of active assignments and their deadlines.
4	Student uploads and submits their work.	The system validates the submission (file type and size) and stores it in the PostgreSQL database.
5	Student tries to submit after the deadline.	The system prevents submission and displays: "Deadline has passed. Submission closed."
6	Teacher views submissions.	The system lists all student submissions with timestamps for review and grading.
7	Teacher deletes or edits an assignment.	The system updates all associated records and notifies students automatically.

Table 4.4.1.1

4.4.1.1

User Action	System Response
Teacher extends assignment deadline.	The system updates the deadline and notifies all students in the community.
Student uploads an incorrect file and re-submits before the deadline.	The system replaces the previous file with the latest submission.

Table 4.4.1.1

4.4.1.2 Exception Flow

Condition	System Response
Teacher attempts to post an assignment without a deadline.	The system prompts: "Please specify a valid deadline."
File upload exceeds maximum allowed size.	The system rejects the submission and displays: "File too large. Please upload a smaller file."
Database or network error during upload.	The system displays: "Upload failed. Please try again later."

Table 4.4.1.2

4.4.2 Functional Requirements

Req ID	Description	Priority	Status
REQ-26	The system shall allow teachers to create assignments with titles, descriptions, and deadlines.	High	Pending
REQ-27	The system shall notify all enrolled students when a new assignment is posted.	High	Pending
REQ-28	The system shall allow students to submit assignments in supported file formats before the deadline.	High	Pending
REQ-29	The system shall prevent submission after the assignment deadline has passed.	High	Pending
REQ-30	The system shall store assignment submissions securely in PostgreSQL with timestamps.	High	Pending
REQ-31	The system shall allow teachers to view, download, and review student submissions.	Medium	Pending
REQ-32	The system shall allow teachers to edit or delete assignments, with automatic notifications to students.	Medium	Pending
REQ-33	The system shall support deadline extension or re-submission functionality before expiry.	Medium	Pending
REQ-34	The system shall validate file type and size before accepting submission uploads.	Medium	Pending

REQ-35	The system shall handle upload errors gracefully and prompt users to retry.	Medium	Pending

Table 4.4.2

4.5 Admin Panel & Management System

Section	Details
Feature Name:	**Admin Panel & Management System**
Description:	The Admin Panel is the core control center of EduCom, allowing administrators to manage all platform operations. Admins can add, remove, or update users (students, teachers, HODs, PMs), assign courses, modify user roles, send notifications, and moderate flagged content. The Admin Panel ensures structured management of academic data and enforces system-wide control and security.
Priority:	**High**
Actors:	Admin
Goal:	To enable system administrators to manage users, communities, courses, and overall platform activities effectively.
Pre-Conditions:	1. Admin must be logged into the system. 2. All necessary database tables (users, courses, roles, communities) must exist.
Post-Conditions:	1. User and course data are updated in the database. 2. Notifications or role updates are reflected immediately for all affected users. 3. Admin actions (edits/deletes) are logged for accountability.

Table 4.5.0

4.5.1 Stimulus/Response Sequences

Step	User Action	System Response
1	Admin logs into the dashboard.	The system loads the Admin Panel displaying user, course, and community management options.
2	Admin adds a new teacher or student record.	The system stores the user in the PostgreSQL database and sends a credential email automatically.
3	Admin assigns a teacher to a course and adds enrolled students.	The system updates the course table and automatically generates or updates the course community.
4	Admin changes a faculty member's role (e.g., promote to HOD or PM).	The system updates role permissions instantly.
5	Admin sends a notification to a specific class or all communities.	The system broadcasts the message via WebSocket and stores it in the notification log.
6	Admin reviews flagged or reported	The system displays message details and allows

	messages.	Admin to delete, warn, or block the sender.
7	Admin removes a user from the system.	The system deletes the user record and updates all related communities.

Table 4.5.1

4.5.1.1 Alternate Flow (Bulk Operations)

User Action	System Response
Admin uploads a CSV file to add multiple students or teachers at once.	The system validates and imports records into the database, skipping duplicates and reporting errors.
Admin archives completed courses.	The system marks related communities as "inactive" and hides them from active dashboards.

Table 4.5.1.1

4.5.1.2 Exception Flow

Condition	System Response
Admin attempts to add a user with an existing ID or email.	The system displays: "User already exists."
Admin sends a notification with an empty message.	The system prompts: "Please enter a message before sending."
Database connection error during role or course update.	The system displays: "Operation failed. Please try again later."

Table 4.5.1.2

4.5.2 Functional Requirements

Req ID	Description	Priority	Status
REQ-36	The system shall allow the Admin to add, update, or remove users (students, teachers, HODs, PMs).	High	Pending
REQ-37	The system shall allow the Admin to assign courses to teachers and enroll students in those courses.	High	Pending
REQ-38	The system shall allow the Admin to modify user roles and permissions dynamically.	High	Pending
REQ-39	The system shall automatically create or update a community when course assignments change.	High	Pending
REQ-40	The system shall allow the Admin to send notifications to specific or all communities.	High	Pending
REQ-41	The system shall allow the Admin to view and act upon flagged or reported messages.	High	Pending
REQ-42	The system shall log all admin actions (add, edit, delete) for auditing purposes.	Medium	Pending
REQ-	The system shall support bulk import of user records through CSV	Medium	Pending

43	upload.		
REQ-44	The system shall validate all inputs and prevent duplicate entries.	High	Pending
REQ-45	The system shall display confirmation prompts before deleting any user or course data.	Medium	Pending

Table 4.5.2

4.6 Marketplace System

Section	Details
Feature Name:	**Marketplace System**
Description:	The Marketplace System allows users within the EduCom platform to buy and sell products such as books, notes, accessories, and other academic-related or personal items. Transactions are handled securely through **PayFast**, ensuring that payments are processed safely. Admins monitor listings to prevent misuse or the posting of inappropriate content. This feature encourages community engagement and provides a trusted trading environment within the campus ecosystem.
Priority:	**Medium–High**
Actors:	Student, Teacher, Admin
Goal:	To enable authenticated users to list, browse, and purchase items securely within the EduCom marketplace using integrated PayFast payments.
Pre-Conditions:	1. User must be authenticated through the Sign In feature. 2. User must have access to the Marketplace module. 3. PayFast service must be active and connected via API.
Post-Conditions:	1. Products are listed in the marketplace and visible to all users. 2. Completed transactions are recorded and confirmed. 3. Admin can view all listings and take moderation actions when needed.

Table 4.6.0

4.6.1 Stimulus/Response Sequences

Step	User Action	System Response
1	User opens the **Marketplace** tab.	The system displays available products, filters, and a search bar.
2	User clicks **"Sell Item."**	The system opens a listing form requesting title, description, price, and product image upload.
3	User submits listing for review.	The system validates input data, stores it in the PostgreSQL database, and displays the pending approval status.
4	Admin reviews and approves the product listing.	The product becomes visible to all users in the marketplace.

5	Another user clicks **"Buy Now."**	The system redirects to PayFast's secure payment page to process the transaction.
6	Payment is completed successfully.	PayFast sends a confirmation callback; EduCom marks the product as sold and stores transaction details.
7	Admin views or removes inappropriate listings.	The system updates the listing status and notifies the item owner.

Table 4.6.1

4.6.2 Alternate Flow (Listing Management)

User Action	System Response
Seller edits product details (price, description) before approval.	The system updates the listing record in PostgreSQL.
Seller deletes a product before it's sold.	The system removes the listing and frees associated storage space.
Buyer cancels payment before completion.	The system receives cancellation status and marks the order as "Cancelled."

Table 4.6.2

4.6.3 Exception Flow

Condition	System Response
Seller attempts to post an incomplete listing (missing image or price).	The system displays: "All required fields must be filled before submission."
PayFast service unavailable during checkout.	The system displays: "Payment gateway unavailable. Please try again later."
Admin removes a listing already marked as sold.	The system logs the action and notifies both buyer and seller.

Table 4.6.3

4.6.4 Functional Requirements

Req ID	Description	Priority	Status
REQ-46	The system shall allow users to create product listings with details such as title, description, price, and image.	High	Pending
REQ-47	The system shall store product listings in PostgreSQL and display them in the marketplace after admin approval.	High	Pending
REQ-48	The system shall integrate PayFast API for secure transaction processing.	High	Pending
REQ-49	The system shall record and store all completed transactions with payment reference IDs.	High	Pending

Table 4.6.4

REQ-50	The system shall allow admins to review, approve, or delete any product listing.	High	Pending
REQ-51	The system shall allow users to edit or delete their listings before approval or sale.	Medium	Pending
REQ-52	The system shall mark items as sold once a transaction is completed.	Medium	Pending
REQ-53	The system shall handle payment failure or cancellation gracefully and update order status accordingly.	Medium	Pending
REQ-54	The system shall restrict posting of inappropriate or unrelated items.	High	Pending
REQ-55	The system shall notify buyers and sellers of order updates or admin actions.	Medium	Pending

4.7 Notification & Announcement System

Section	Details
Feature Name:	**Notification & Announcement System**
Description:	The Notification & Announcement System enables Admins to send important updates, alerts, or announcements to specific course communities or the entire platform. Notifications are exclusively sent via **email** to ensure professional, reliable, and recordable communication. This feature is designed for class updates, cancellations, reschedules, deadline reminders, and system-wide announcements.
Priority:	**High**
Actors:	Admin, Teacher, Student
Goal:	To provide an efficient and reliable email-based notification system for communication between Admins, Teachers, and Students.
Pre-Conditions:	1. User must be registered in the system with a valid email address. 2. Admin must be logged in to access the notification panel.
Post-Conditions:	1. Email notifications are successfully delivered to selected recipients. 2. Notification logs are stored in the database for record-keeping.

Table 4.7.0

4.7.1 Stimulus/Response Sequences

Step	User Action	System Response
1	Admin opens the "Notification Center" from the dashboard.	The system displays options to select recipients (specific course, community, or all users).
2	Admin composes an announcement	The system validates that required fields are filled.

	message (subject, body).	
3	Admin clicks **"Send Notification."**	The system uses **Nodemailer (SMTP)** to send the announcement email to all selected recipients.
4	Notification sent successfully.	The system displays: "Emails sent successfully" and logs the event in the database.
5	Email delivery fails for specific users.	The system marks failed addresses in the log and shows: "Some emails could not be delivered."

Table 4.7.1

4.7.2 Alternate Flow (Targeted Announcements)

User Action	System Response
Admin selects a specific community or course to receive the message.	The system filters recipients based on course/community membership and sends the email only to those users.
Admin chooses to notify only Teachers or only Students.	The system filters recipients by user role and sends accordingly.

Table 4.7.2

4.7.3 Exception Flow

Condition	System Response
Admin sends an empty message or subject.	The system prompts: "Subject and message body cannot be empty."
Invalid or missing email addresses detected.	The system skips invalid addresses and logs them for admin review.
SMTP server not responding.	The system displays: "Email service unavailable. Please try again later."

Table 4.7.3

4.7.4 Functional Requirements

Req ID	Description	Priority	Status
REQ-56	The system shall allow the Admin to compose and send announcements through email only.	High	Pending
REQ-57	The system shall allow selection of recipients (specific community, course, or all users).	High	Pending
REQ-58	The system shall use the **Nodemailer (SMTP)** service for sending all email notifications.	High	Pending
REQ-59	The system shall store notification logs (date, recipients, message subject) in the database.	Medium	Pending
REQ-	The system shall validate message subject and content before	Medium	Pending

60	sending.		
REQ-61	The system shall handle and log failed email deliveries for review.	Medium	Pending
REQ-62	The system shall allow Admin to resend or forward previous announcements if required.	Low	Pending

Table 4.7.4

4.8 Anonymous Feedback & Reporting System

Section	Details
Feature Name:	**Anonymous Feedback & Reporting System**
Description:	The Anonymous Feedback & Reporting System allows students to send feedback, suggestions, or queries to Teachers, HODs, or PMs while keeping their identity hidden. It also enables users to report inappropriate or abusive content within communities. Reported content is forwarded to the Admin for review and necessary action. This feature encourages open communication while maintaining user privacy and promoting a respectful environment.
Priority:	**High**
Actors:	Student, Teacher, HOD, PM, Admin
Goal:	To allow students to communicate concerns and feedback anonymously, and to provide a reporting system for maintaining a positive and safe community.
Pre-Conditions:	1. The user must be authenticated. 2. The user must belong to a valid community or course.
Post-Conditions:	1. Feedback messages are securely stored and sent to the intended recipient. 2. Reported messages are logged and displayed to the Admin for review. 3. User identity remains hidden for anonymous feedback.

Table 4.8.0

4.8.1 Stimulus/Response Sequences

Step	User Action	System Response
1	Student opens the "Anonymous Feedback" form in their dashboard.	The system displays a form with fields for recipient (Teacher, HOD, PM) and message content.
2	Student writes and submits the message anonymously.	The system stores the message in PostgreSQL, masking the sender's identity.
3	Recipient (Teacher/HOD/PM) checks their feedback inbox.	The system displays received messages labeled as "Anonymous."
4	Recipient replies to the feedback (optional).	The system sends the response without revealing either party's identity.

5	User reports an inappropriate message or post.	The system flags the message and notifies the Admin for review.
6	Admin opens the moderation panel.	The system displays flagged messages with details and action options (Delete, Warn, Block).

Table 4.8.1

4.8.2 Alternate Flow (Feedback Categories)

User Action	System Response
Student selects "Complaint," "Suggestion," or "General Feedback."	The system categorizes the message accordingly for better filtering.
Teacher sends acknowledgment or reply to feedback.	The system routes the response back anonymously through the platform.

Table 4.8.2

4.8.3 Functional Requirements

Req ID	Description	Priority	Status
REQ-63	The system shall allow students to send feedback or queries anonymously to Teachers, HODs, or PMs.	High	Pending
REQ-64	The system shall store anonymous feedback securely without revealing sender identity.	High	Pending
REQ-65	The system shall allow Teachers, HODs, or PMs to receive and review anonymous feedback in their dashboards.	High	Pending
REQ-66	The system shall provide a "Report Message" option for users to flag inappropriate or abusive content.	High	Pending
REQ-67	The system shall notify the Admin automatically upon receipt of a reported message.	High	Pending
REQ-68	The system shall allow the Admin to take moderation actions (Delete Message, Warn User, Block User).	High	Pending
REQ-69	The system shall categorize feedback types (Complaint, Suggestion, General).	Medium	Pending
REQ-70	The system shall prevent access to sender identity for all anonymous messages.	High	Pending
REQ-71	The system shall allow optional recipient replies without revealing identities.	Medium	Pending
REQ-72	The system shall store all feedback and reports in PostgreSQL for auditing and record-keeping.	Medium	Pending

Table 4.8.3

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Response Time:

- User login and dashboard loading should complete within **2–3 seconds**.
- Community messages should appear in **real time (≤2 seconds)**.
- Assignment uploads (≤25 MB) should complete within **5 seconds**.
- Marketplace listings or searches should load within **3–4 seconds**.
- Email notifications (via Nodemailer) should be sent within **5 seconds** of initiation.

System Load Handling:

- The platform should support at least **500 concurrent users** without significant performance degradation.
- API responses should return within **1 second** on average.
- Chat system (Socket.io) should handle up to **50 active messages per second**.

Availability:

- The system should maintain **99% uptime** during academic hours.
- Automatic recovery and session restoration should occur after unexpected downtime or connection loss.

5.2 Safety Requirements

- **Data Protection:**
  User data (personal, academic, and marketplace-related) must be securely stored in **PostgreSQL** with **AES encryption** for sensitive information.
- **Error Handling:**
  The system must prevent accidental deletion or modification of critical data by using **confirmation prompts** and **automated daily backups**.

- **System Reliability:**
  Failures or crashes must not cause **data corruption**; recovery mechanisms and backup routines must ensure full data restoration.

- **Transaction Safety:**
  All payment operations through **PayFast** must use **HTTPS** and be validated through secure tokens.

- **Anonymity Assurance:**
  The anonymity system must ensure sender identity cannot be revealed, even to Teachers or Admins, except under authorized investigation.

5.3 Security Requirements

- **User Authentication:**
  All users must log in using valid credentials; passwords are **hashed using bcrypt** and verified securely.

- **Data Privacy:**
  Personal and academic data must be **encrypted** in storage and during transmission via **SSL/TLS**.

- **Access Control:**
  Role-based permissions (Admin, Teacher, Student, HOD, PM) must be enforced to restrict access to sensitive features and data.

- **Compliance:**
  The system must follow standard **data protection policies** and institutional IT security guidelines.

- **Audit & Monitoring:**
  All login attempts, course modifications, and admin actions must be logged for **auditing and breach detection**.

- **Session Management:**
  JWT tokens must expire automatically after inactivity, requiring users to reauthenticate securely.

5.4 Software Quality Attributes

- **Usability:**
  The interface should be **intuitive and responsive**, enabling users to perform core actions (login, chat, submit assignments) with minimal guidance.
- **Reliability:**
  EduCom should maintain **99% operational uptime**, handling network errors gracefully without data loss.
- **Maintainability:**
  The codebase should follow **modular architecture**, ensuring ease of debugging, updates, and feature expansion.
- **Scalability & Flexibility:**
  The platform should support an increasing number of users, communities, and marketplace listings without major redesign.
- **Portability & Interoperability:**
  EduCom should be compatible with all major browsers (Chrome, Firefox, Edge, Safari) and devices (desktop, tablet, mobile).
- **Security & Robustness:**
  The system must resist unauthorized access, handle invalid inputs gracefully, and sanitize all user data to prevent injection attacks.

5.5 Business Rules

- **User Roles:**
  Only **registered users** (added by Admin) can access the system. Admins have additional privileges to manage content, courses, and accounts.
- **Community Access:**
  Users can only view and participate in **communities** related to their assigned courses.
- **Assignment Rules:**
  Students can submit assignments only **before the specified deadline**; late submissions are automatically blocked.
- **Marketplace Rules:**
  Users can post and purchase products only within the EduCom community; all transactions occur through **PayFast** for security.

- **Data Integrity:**
  Users cannot edit or delete other users' data. All changes are **logged and time-stamped** for accountability.
- **Feedback and Reporting:**
  Anonymous feedback must remain identity-protected, and reported messages must be reviewed by Admin before action.
- **Compliance:**
  All operations must follow **institutional policies** on privacy, academic integrity, and acceptable use of communication platforms.

6. Other Requirements

Category	Details
6.1 Database Requirements	- The system shall use **PostgreSQL** as the primary relational database. - Database schema shall support **role-based access control**, course-community relationships, and anonymous message storage. - Sensitive information (passwords, payment data, personal details) must be **encrypted** using strong hashing and AES standards. - All transactions (assignment submissions, payments, community posts) shall be **logged with timestamps** for tracking and auditing. - Regular **database backups** shall occur daily, and recovery scripts must ensure minimal data loss during failure.
6.2 Internationalization and Localization	- The system's text and messages shall be stored in a **language-independent format**, enabling easy translation in future updates. - Default language is **English (UK)**. - Date and time formats shall follow the **local timezone (PKT)**, adaptable if deployed elsewhere.
6.3 Legal and Compliance Requirements	- EduCom must comply with **institutional data protection policies** and applicable **IT regulations** governing user data and privacy. - Payment operations via **PayFast** must comply with **PCI DSS** (Payment Card Industry Data Security Standard). - The system must respect copyright and intellectual property rights for shared content within communities. - Any collection or storage of personal information must be done with **user consent** and protected against unauthorized disclosure.
6.4 Backup and Recovery Requirements	- Automatic backups of databases and uploaded files shall be scheduled daily. - The system shall maintain at least **three recent backup copies** stored securely. - Recovery mechanisms must ensure data restoration within **30 minutes** in case of system failure.
6.5 Software Reuse and Extensibility	- The system shall be designed with **modular architecture** to allow reuse of components such as chat, authentication, and notification modules in future projects. - APIs and services shall follow **RESTful design principles** to ensure interoperability with future mobile or third-party integrations.

6.6 Future Enhancement Possibilities	- Integration with **ZABDESK API** if access becomes available for automated course synchronization. - Addition of **AI-driven content moderation** for detecting inappropriate behavior more accurately. - Implementation of **mobile app support** for Android and iOS. - Enhancement of marketplace features with bidding, product ratings, and delivery tracking.
6.7 Environmental and Deployment Considerations	- The system shall operate on cloud-based infrastructure (e.g., AWS, Azure, or VPS) for scalability and reliability. - It must support **HTTPS** for all network communications. - Deployment pipelines (CI/CD) should be configured for automated testing and version control (e.g., using GitHub Actions or Jenkins).

Table 6.1

Appendix A: Glossary

Term / Acronym	***Definition***
EduCom	*The academic communication and management platform connecting students, faculty, HODs, PMs, and Admins for learning, collaboration, and information sharing.*
Admin	*The highest-level user responsible for managing the system, including adding/removing users, assigning roles, managing communities, reviewing reports, and sending announcements.*
HOD (Head of Department)	*A designated faculty member responsible for departmental supervision, reviewing student feedback, and addressing academic queries.*
PM (Program Manager)	*A role that manages program-level operations, class scheduling, and communication between departments and students.*
Teacher / Faculty	*A user responsible for managing course-related activities, creating assignments, grading submissions, and interacting with students in course communities.*
Student	*A user who participates in assigned course communities, submits assignments, interacts with teachers, and can send anonymous feedback.*
Community	*A virtual group automatically created for each course, allowing enrolled users to chat, share resources, and collaborate.*
Anonymous Messaging	*A feature that allows users to communicate or report issues without revealing their identity to maintain privacy and openness.*
Assignment Module	*A feature that enables teachers to upload assignments, set deadlines, and prevent submissions after the due date.*
Marketplace / Buy & Sell	*A module that enables EduCom users to post, sell, and purchase products securely within the platform.*
PayFast	*A secure third-party payment gateway used for processing all marketplace transactions within EduCom.*
Nodemailer	*A backend module that handles automated email notifications for announcements, class updates, and administrative alerts.*
PostgreSQL	*The relational database used to store all user, course, assignment, and marketplace data.*
REST API	*A standardized web protocol (Representational State Transfer) that enables seamless communication between EduCom's frontend and backend.*
JWT (JSON Web Token)	*A secure token-based authentication method used to verify and manage user sessions without maintaining server-side session storage.*
AES (Advanced Encryption Standard)	*A strong encryption algorithm used for protecting confidential information such as passwords and payment data.*
Socket.io	*A real-time communication framework enabling instant messaging and live updates across communities and chat modules.*
CI/CD (Continuous Integration / Continuous Deployment)	*Automated software deployment pipelines used to streamline updates, testing, and releases.*

HTTPS	*Hypertext Transfer Protocol Secure — used to encrypt and protect data transmitted between users and the EduCom server.*
Data Integrity	*Ensuring that all data remains accurate, consistent, and reliable throughout system operations and updates.*
Encryption	*The process of securing data by converting it into a coded form that can only be accessed with a decryption key.*
User Role	*Defines access privileges for each user type (Admin, HOD, PM, Teacher, Student) within the system.*
Usability	*A measure of how easily users can interact with the EduCom interface to complete key tasks.*
Anonymity	*A privacy-focused feature ensuring that user identity remains hidden during feedback or certain communications.*
Scalability	*The ability of EduCom to handle increasing numbers of users, messages, and communities without performance loss.*
Backup	*A copy of system and database data created regularly to prevent data loss and allow recovery in case of failure.*
Audit Log	*A record of system activities (e.g., logins, updates, reports) maintained for monitoring and security auditing.*
API (Application Programming Interface)	*A defined set of protocols and tools that allow software components to interact or integrate with other systems.*
Frontend	*The user-facing interface built with React.js where users interact with EduCom's features.*
Backend	*The server-side logic developed in Node.js and Express.js that handles data processing, security, and communication with the database.*
Access Control	*A security mechanism restricting user actions based on their assigned roles and permissions.*
Moderator Panel	*A special section of the Admin dashboard for reviewing reported messages, handling abuse reports, and taking moderation actions.*
Notification System	*The feature used by Admins to send email announcements or alerts to specific users or communities.*
Community Chat	*A real-time communication feature enabling users to message, share files, and discuss course-related topics.*
Deadline Enforcement	*A rule in the assignment module that automatically blocks submissions after the due date has passed.*
Error Handling	*Mechanisms that detect, log, and display user-friendly error messages when unexpected issues occur.*
Database Backup	*The automated daily process that stores a secure copy of PostgreSQL data for disaster recovery.*
Feedback Inbox	*The interface used by Teachers, HODs, and PMs to receive anonymous messages or feedback from students.*
Role Management	*A feature that allows Admins to assign, update, or revoke user roles dynamically.*

Appendix B: To Be Determined List

TBD No.	Description	Responsible Role	Expected Resolution Stage
TBD-1	Final decision on the **maximum file size limit** for assignment uploads (currently estimated at 25 MB).	Development Team	During system testing phase
TBD-2	Confirmation of **maximum concurrent users** supported under deployment infrastructure (currently estimated at 500).	Project Manager / DevOps	After load testing
TBD-3	Final design and **UI layout** for the Teacher and Student dashboards.	UI/UX Designer	Before frontend implementation
TBD-4	Selection of **email server provider** (e.g., Gmail SMTP, SendGrid, or institutional mail server) for notification delivery.	Admin / Development Team	Before deployment
TBD-5	Decision on whether to implement **AI-based content moderation** or rely solely on rule-based filters for abusive messages.	Project Lead / Developers	Phase 2 enhancement
TBD-6	Determination of **cloud hosting provider** (e.g., AWS, Azure, or DigitalOcean) for production environment.	Project Manager / IT Admin	Deployment stage
TBD-7	Definition of **retention policy** for stored messages, feedback, and report logs (e.g., 6 months, 1 year).	Admin / Legal Advisor	Before production rollout
TBD-8	Final **grading mechanism** or evaluation criteria for assignment submissions (if required in future phases).	Academic Team / Developers	Future enhancement
TBD-9	Confirmation of **PayFast merchant account credentials** and test sandbox integration details.	Finance / Development Team	Before marketplace module release
TBD-10	Decision on **mobile application development** timeline (Android/iOS) and integration approach with the existing backend.	Project Manager / Developers	Future expansion phase
TBD-11	Finalization of **email template designs** for notifications and announcements (branding, logos, layout).	UI/UX Designer / Admin	Before beta testing
TBD-12	Selection of **backup storage location** (local server or cloud repository) for daily PostgreSQL backups.	DevOps Engineer	Deployment phase
TBD-13	Clarification on **institutional policies** for anonymous feedback handling and escalation.	Admin / HOD / Legal Advisor	Policy definition stage
TBD-14	Specification of **access control hierarchy** for Admin sub-roles (e.g., moderators, content reviewers).	Project Lead	Implementation stage
TBD-15	Final **email verification method** for user authentication (OTP vs. verification link).	Backend Developer	Development phase