Software Requirements Specification

for

Flex Student Portal Project

Version 1.0 approved

Prepared by 22k5024 Hadi

22k5018 Aheed,22k5144 Lucky

12.5.2024

Table of Contents

1. Introduction

The Student Portal Website is designed to facilitate effective interaction between students and teachers through distinct login modules. It provides features like attendance tracking, academic marks distribution, and timetable management. This SRS describes the foundational requirements for Version 1.0 of the portal, focusing on core functionalities for students and teachers.

a. Purpose :

The purpose of this document is to outline the functional and nonfunctional requirements of the Student Portal Website. It serves as a guide for developers, stakeholders, and project managers to ensure the product meets its intended goals. This SRS covers both the student and teacher login modules but excludes administrative functionalities, which will be part of a future release.

b. Document Conventions

- ***Font Style and Size**:*
- ***Headings**: Arial, Bold, 14 pt*
- ***Body Text**: Times New Roman, Regular, 12 pt*
- ***Priorities**: Requirements are categorized into:*
- ***High Priority**: Core functionalities*
- ***Medium Priority**: Enhancements*
- ***Low Priority**: Additional features for future releases*
- ***Terminology**:*
- *"User" refers to students or teachers unless specified otherwise.*
- *"Portal" refers to the Student Portal Website.*

c. Intended Audience and Reading Suggestions

- *This document is intended for:*
- ***Developers**: To understand technical and functional requirements.*
- ***Project Managers**: For planning and resource allocation.*
- ***Users (Students and Teachers)**: To comprehend the intended functionality.*
- ***Documentation Writers**: For user manuals and guides.*

d. Product Scope

The Student Portal Website aims to enhance educational efficiency by providing an online platform for academic interactions.

1.4.1 Key Benefits:

- *Streamlined attendance management and grading.*
- *Easy communication of marking between students and teachers.*
- *Transparent performance tracking for students.*

1.4.2 Goals:

- *Improve accessibility to academic resources.*
- *Provide secure data management for students and teachers.*
- *Align with institutional objectives of fostering digital education.*

- *Allow students to access their performance anytime.*
- *This portal supports the institution's vision of adopting technology for modernized learning.*

2. Overall Description

a. Product Perspective

*The Student Portal Website is a **new, self-contained product** designed to support academic interactions within an institution. It does not replace any existing system but rather aims to integrate seamlessly with institutional services. This integration enables synchronization of student data, grades, and class schedules.*

b. Product Functions

i. ***Dynamically Saving and Storing Data**: Allows real-time data entry and updates, ensuring seamless user interaction and data persistence in the database.*

ii. ***Image Saving in Database**: Facilitates storing and retrieving student or teacher profile pictures securely in a Binary format.*

iii. ***Hashing Password**: Secures user passwords using hashing algorithms ensuring stored passwords are not readable.*

iv. ***Timetable Management for Students**: Provides personalized timetables based on enrolled courses, displaying class schedules, timings, and locations.*

v. ***Showing Student Data**: Displays comprehensive student profiles, including personal information, academic history, and enrolled courses.*

vi. ***Course Registration for Students and Teachers**: Enables students to enroll in courses and teachers to assign themselves to courses for a given term.*

vii. ***Managing and Viewing Attendance**: Tracks and displays attendance records, allowing students and teachers to monitor attendance history and compliance.*

viii. ***Managing Transcript**: Generates and displays official academic transcripts, showing completed courses, grades, and GPA.*

ix. ***Marks Distribution for Students**: Provides a breakdown of marks for assignments, quizzes, and exams, offering a clear view of performance in each course.*

x. ***Challan Information**: Generates and tracks fee challans for tuition and other expenses, allowing students to view and download payment details.*

xi. ***Changing Password for Website**: Allows users to update their passwords securely via the portal.*

xii. ***Drop and Add Courses for Teachers**: Enables teachers to modify their assigned courses by adding new ones or dropping current ones during the registration period.*

xiii. ***Withdraw Courses for Students**: Allows students to withdraw from courses within a specified period, updating their academic records accordingly.*

xiv. ***Managing Course Feedback**:*

Collects and organizes feedback on courses and instructors from students, aiding in quality improvement.

c. User Classes and Characteristics

i. ***Students***:
Access their grades, attendance, and assignments.
Typically have basic technical knowledge.

ii. ***Teachers***:
Manage student performance data and post resources.
May vary in technical expertise.

d. Operating Environment

i. ***Hardware***:
Any standard desktop, laptop, tablet, or mobile device.

ii. ***Software***:
Operating System: Windows 10+, macOS 11+, Android 10+, iOS 13+.
Browser: Chrome 90+, Firefox 80+, Edge 91+, Safari 14+.

iii. ***Server Requirements***:
Database: PostgreSQL.
Packages:react.js,express.js,cors,bcrypt,multer,node.js

e. Design and Implementation Constraints

i. ***Technologies***: *The portal will use* ***ReactJS*** *for the front end and* ***Node.js*** *with Express for the back end.*

ii. ***Scalability***: *Design should allow future expansions, like administrator features or advanced analytics.*

iii. ***Performance***: *Handle up to many concurrent users.*

f. User Documentation

i. **Tutorials**: Video tutorials available on the portal's help page.

g. Assumptions and Dependencies

i. ***Assumptions***:
1. *Users will have access to the internet and compatible devices.*

ii. ***Dependencies***:
1. *Availability of third-party libraries (e.g., React, Node.js).*
2. *Institutional IT infrastructure for hosting the system.*

h. Process Model

For the Student Portal Website development, the **Agile Development Process Model** is selected to ensure flexibility, iterative progress, and responsiveness to changes.

Phases in the Agile Process:

- **Concept and Inception**:
  - Gather requirements from colleagues and group members.
  - Define scope and objectives.
  - Develop initial use cases and project proposals.
- **Iteration Planning**:

- Break down requirements into small, manageable user stories.

- **Design and Prototyping**:
  - Create a blueprint for key interfaces (login, dashboard, assignment management).
- **Incremental Development**:
  - Step by step develop every page and link the backend to pull the data dynamically.
- **Testing and Integration**:
  - Perform system testing to ensure all components work together seamlessly.
- **Deployment and Maintenance**:
  - Deploy the portal to the local host and connect with the server.
  - Provide ongoing support and updates based on feedback from group members and other colleagues.

i. Project Plan

i. Project Milestones and Timeline

1. Use-Case Diagram:1 week
2. ERD:1 week
3. Database Implementation: 1 week
4. Making Front-end: 2-3 weeks
5. Connecting Database with back-end: 1-2 weeks
6. Connecting Front-end with back-end: 3-4 weeks
7. Finalizing & Testing: 2-3 weeks

j. Feasibility report

2.10.1 **Technical Feasibility**

The Student Portal Website is technically feasible, leveraging modern web technologies:

- **Frontend**: ReactJS for dynamic, responsive interfaces.
- **Backend**: Node.js for scalable server-side logic.
- **Database**: MySQL/PostgreSQL for robust data storage and retrieval.
- **Hosting**: Cloud platforms like AWS or Azure ensure high availability.
  Integration with existing institutional systems (LMS/SIS) is supported via APIs.

2.10.2 **Operational Feasibility**

- The portal aligns with the institution's goal of digitizing academic processes.
- Users (students/teachers) can adapt easily due to intuitive design and training materials.

2.10.3 **Economic Feasibility**

- Initial development cost includes hosting, software licensing, and manpower.
- Cost savings include reduced paper use, centralized administration, and improved academic efficiency.

2.10.4 **Schedule Feasibility**

- The system can be developed and deployed within six months with a standard Agile approach.

k. Use case descriptions

Use Case: Login

Actors: Student, Teacher, Backend
Description: Enables users to securely log into the portal.
Precondition: User credentials must be registered in the system.
Basic Flow:

1. User enters username and password.
2. System verifies credentials against the database.
3. Users are redirected to their respective dashboard.
   Postcondition: User gains access to their personalized functionalities.

Use Case: Change Password

Actor: Student
Description: Allows students to change their login password securely.
Precondition: User must be logged in.
Basic Flow:

1. User selects the "Change Password" option.
2. Enter the current password and a new password.
3. System updates the password after verification.
   Postcondition: Password is updated successfully.

Use Case: View Attendance

Actor: Student
Description: Students can view their attendance record.
Precondition: Attendance data must exist in the system.
Basic Flow:

1. Students select the "View Attendance" option.
2. System fetches and displays attendance records.
   Postcondition: Attendance details are displayed to the student.

Use Case: View Marks

Actor: Student
Description: Allows students to view marks for their courses.
Precondition: Teachers must have uploaded marks to the system.
Basic Flow:

1. Student selects "View Marks."
2. System retrieves and displays marks for relevant courses.
   Postcondition: Marks are displayed to the student.

Use Case: Withdraw from Course

Actor: Student
Description: Enables students to withdraw from a course.
Basic Flow:

1. Student selects a course to withdraw from.
2. Confirms withdrawal.
3. System removes the student from the course.
   Postcondition: Student is unenrolled from the course.

Use Case: View Challan

Actor: Student
Description: Students can view and download their fee challan.
Precondition: Fee challan details must be generated by the system.
Basic Flow:

1. Students navigate to "View Challan."
2. System retrieves the fee details and displays them.
   Postcondition: Fee challan is displayed.

Use Case: Manage Students

Actor: Teacher
Description: Teachers can manage student information for their assigned courses.
Precondition: Teacher must have access to relevant course data.
Basic Flow:

1. Teacher selects "Manage Students."
2. Adds, updates, or removes student details as needed.
   Postcondition: Student information is updated successfully.

Use Case: Manage Attendance

Actor: Teacher
Description: Teachers can upload, edit, or review attendance records for students.

Precondition: The teacher must have assigned classes.
Basic Flow:

1. Teacher navigates to "Manage Attendance."
2. Updates or reviews attendance for selected classes.
   Postcondition: Attendance records are saved.

Use Case: Generate Marks Report

Actor: Teacher
Description: Teachers can generate and download detailed marks reports for their classes.
Precondition: Marks must already be uploaded to the system.
Basic Flow:

1. Teacher selects the "Generate Marks Report" option.
2. System compiles the data and generates a report.
   Postcondition: Marks report is generated and available for download.

Use Case: Register Student

Actor: Backend
Description: Backend user registers new students into the portal.
Precondition: Student details must be provided.
Basic Flow:

1. Backend enters student information.
2. System assigns a unique ID and creates login credentials.
   Postcondition: Student is successfully registered.

Use Case: Manage Feedback

Actors: Student, Teacher
Description: Collects and manages feedback for courses or instructors.
Basic Flow:

1. Students provide feedback via the portal.
2. Teacher reviews feedback for evaluation.
   Postcondition: Feedback is stored and accessible for analysis.

Use Case: Show Timetable

Actor: Student
Description: Displays a student's class timetable.
Precondition: Timetable must exist in the database.
Basic Flow:

1. Student selects "Show Timetable."
2. System retrieves and displays timetable details.
   Postcondition: Timetable is displayed successfully.

l. Video of working software

3. External Interface Requirements

a. User Interfaces

The Student Portal Website provides intuitive, user-friendly interfaces for both students and teachers, following modern UI/UX principles.

i. ***General Features**:*
   1. ***Standard Layout**:*
   2. ***Header**: Displays the portal name, user role, and logout button.*
   3. ***Navigation Bar**: Contains links for Home, Assignments, Grades, Attendance, Marks,Transcript,Challan,Feedback,Withdraw,Timetable and Course Management.*
   4. ***Consistency**: Standardized buttons for "Submit," and “Remove” and consistency in features such as marks distribution and attendance management.*
   5. ***Accessibility**: keyboard navigation.*

ii. ***Screens**:*
   1.
   2. ***Login Page***
      a. ***Purpose: Allows users (teachers/students) to log in using a username and password. Implements password hashing for security.***
   3. ***Home Page***
      a. ***Purpose: Displays user-specific information such as name, role (teacher/student), and recent activity.***
   4. ***Course Registration Page***
      a. ***Purpose: Lists available courses for students or teachers.***
      b. ***Allows students to register for courses or teachers to assign themselves to unassigned courses.***
   5. ***Marks Distribution Page***

a. ***Purpose: Enables teachers to assign, update, and manage marks for quizzes, assignments, sessionals, and finals.***
b. ***Dynamic columns for quizzes and assignments.***
c. ***Includes options for adding or updating marks for specific evaluations.***

6. ***Attendance Management Page***

a. ***Purpose: Lets teachers mark and manage attendance for specific courses and sections.***
b. ***Displays a list of students and allows attendance marking (Present/Absent).***
c. ***Teachers can add new attendance sessions.***

7. ***Profile Page***

a. ***Purpose: Displays and allows updating of user profile information.***
b. ***Includes a feature to upload and display profile pictures saved in the database.***

8. ***Marks Viewing Page (Student-Specific)***

a. ***Purpose: Displays detailed marks for quizzes, assignments, sessionals, finals, and grand total.***
b. ***Allows students to review their performance across all evaluations.***

9. ***Attendance Viewing Page (Student-Specific)***

a. ***Purpose: Displays attendance records for students, including total attendance percentage and session details.***

10. ***Course Management Page***

a. ***Purpose: Allows teachers to view and manage courses they teach, including viewing assigned sections and students.***
b. ***Provides options to update or withdraw from courses.***

iii. ***Error Messaging***:
1. *Display clear, user-friendly error messages like "Invalid login credentials" or "File not chosen" when attempting to upload a picture without choosing it.*
2. *Highlight erroneous fields in red and provide tooltips for corrections.*

iv. ***Standards and Guidelines***:
1. *Follows Material Design principles.*
2. *Consistent fonts, colors, and component behaviors across pages.*

b. Hardware Interfaces

i. ***Supported***:
1. *Desktops, laptops, tablets, and smartphones.*
2. *Minimum screen resolution: 1024x768 pixels.*

ii. ***Input Devices***:
1. *Standard keyboards, touchscreens, and pointing devices like mice or touchpads.*

c. Software Interfaces

i. ***Databases***:
   1. ***Type***: *pgAdmin 14.*
   2. ***Interaction***:
      a. *Stores user data, assignment submissions, grades, and attendance records.*

ii. ***Libraries and Frameworks***:
   1. ***Frontend***: *ReactJS with Redux for state management.*
   2. ***Backend***: *Node.js with Express for REST API services.*

d. Communications Interfaces

i. **Local Host Communication**:
   1. Communication between host and server through get() and post() to retrieve database information or insert into database.

4. System Features

This section elaborates on the system features based on the provided requirements, incorporating dynamic functionality and specific capabilities.

4.1 System Feature 1: Dynamically Saving and Storing Data

4.1.1 Description and Priority

This feature allows real-time data entry, modification, and retrieval, ensuring that data is consistently updated and stored in the database for seamless user interaction.

- **Priority**: High

4.1.2 Stimulus/Response Sequences

1. **Stimulus**: A user inputs data (e.g., new student or attendance updates).
   Response: The system dynamically saves the data to the database and confirms success.

4.1.3 Functional Requirements

- **REQ-1**: The system shall allow users to input and modify data dynamically in real-time.
- **REQ-2**: The system shall ensure data persistence in the database and maintain backups.
- **REQ-3**: The system shall provide immediate feedback on data entry status (success or failure).

4.2 System Feature 2: Image Saving in Database

4.2.1 Description and Priority

This feature enables secure storage and retrieval of user profile pictures in a binary format within the database.

- **Priority**: Medium

4.2.2 Stimulus/Response Sequences

1. **Stimulus**: A user uploads a profile picture.
   Response: The system converts the image to binary format and securely stores it in the database.

4.2.3 Functional Requirements

- **REQ-1**: The system shall allow users to upload images in common formats (e.g., JPEG, PNG).
- **REQ-2**: The system shall convert and store images securely in binary format.
- **REQ-3**: The system shall retrieve and display profile pictures for authorized users.

4.3 System Feature 3: Hashing Password

4.3.1 Description and Priority

This feature ensures secure password storage by hashing user passwords with cryptographic algorithms.

- **Priority**: High

4.3.2 Stimulus/Response Sequences

1. **Stimulus**: A user sets or updates their password.
   Response: The system hashes the password and stores the hashed value securely in the database.

4.3.3 Functional Requirements

- **REQ-1**: The system shall hash all passwords using secure algorithms (e.g., SHA-256).
- **REQ-2**: The system shall ensure password data is stored in a non-reversible format.
- **REQ-3**: The system shall compare hashed passwords during authentication without storing plaintext passwords.

4.4 System Feature 4: Timetable Management for Students

4.4.1 Description and Priority

This feature provides personalized timetables based on enrolled courses, displaying class schedules, timings, and locations.

- **Priority**: High

4.4.2 Stimulus/Response Sequences

1. **Stimulus**: A student requests to view their timetable.
   Response: The system retrieves and displays the student's personalized schedule.

4.4.3 Functional Requirements

- **REQ-1**: The system shall generate timetables dynamically based on student course enrollment.
- **REQ-2**: The system shall include class timings, locations, and instructor details.
- **REQ-3**: The system shall notify students of timetable changes.

4.5 System Feature 5: Managing and Viewing Attendance

4.5.1 Description and Priority

This feature tracks and displays attendance records, enabling students and teachers to monitor attendance history and compliance.

- **Priority**: High

4.5.2 Stimulus/Response Sequences

1. **Stimulus**: A teacher updates attendance for a class.
   Response: The system stores the attendance data in real-time and updates student records.
2. **Stimulus**: A student requests to view attendance history.
   Response: The system retrieves and displays attendance records.

4.5.3 Functional Requirements

- **REQ-1**: The system shall allow teachers to update and track attendance dynamically.
- **REQ-2**: The system shall allow students to view their attendance records.
- **REQ-3**: The system shall send notifications to students with low attendance.

4.6 System Feature 6: Marks Distribution for Students

4.6.1 Description and Priority

This feature provides a detailed breakdown of marks for assignments, quizzes, and exams.

- **Priority**: Medium

4.6.2 Stimulus/Response Sequences

1. **Stimulus**: A teacher inputs student marks.
   Response: The system calculates and updates the marks distribution.
2. **Stimulus**: A student views the breakdown of marks for a course.
   Response: The system displays the detailed marks distribution.

4.6.3 Functional Requirements

- **REQ-1**: The system shall calculate and store marks distribution for each student.
- **REQ-2**: The system shall display a detailed breakdown of marks by component (e.g., assignments, exams).
- **REQ-3**: The system shall notify students of updated marks.

4.7 System Feature 7: Challan Information

4.7.1 Description and Priority

This feature generates and tracks fee challans for tuition and other expenses.

- **Priority**: Medium

4.7.2 Stimulus/Response Sequences

1. **Stimulus**: A student requests a challan for payment.
   Response: The system generates and displays the requested challan.

4.7.3 Functional Requirements

- **REQ-1**: The system shall generate fee challans dynamically based on tuition and other fees.
- **REQ-2**: The system shall allow students to download challan PDFs.
- **REQ-3**: The system shall track payment status for issued challans.

4.8 System Feature 8: Managing Transcript

4.8.1 Description and Priority

This feature generates and displays academic transcripts for students, including grades and GPA.

- **Priority**: High

4.8.2 Stimulus/Response Sequences

1. **Stimulus**: A student requests their transcript.
   Response: The system generates and displays the official transcript.

4.8.3 Functional Requirements

- **REQ-1**: The system shall generate academic transcripts dynamically.
- **REQ-2**: The system shall display completed courses, grades, and GPA.
- **REQ-3**: The system shall allow students to download transcripts in PDF format.

4.9 System Feature 9: Course Registration for Students and Teachers

4.9.1 Description and Priority

This feature allows students to register for courses and teachers to assign themselves to courses during the registration period.

- **Priority**: High

4.9.2 Stimulus/Response Sequences

1. **Stimulus**: A student selects courses to enroll in.
   Response: The system validates and updates enrollment.
2. **Stimulus**: A teacher assigns themselves to a course.
   Response: The system updates the course-teacher assignment.

4.9.3 Functional Requirements

- **REQ-1**: The system shall allow students to select available courses for enrollment.
- **REQ-2**: The system shall allow teachers to assign themselves to courses.

4.10 System Feature 10: Withdraw Courses for Students

4.10.1 Description and Priority

This feature allows students to withdraw from courses, ensuring that their academic records are updated accordingly.
Priority: High

4.10.2 Stimulus/Response Sequences

- **Stimulus**: A student initiates a course withdrawal request.
  Response: The system checks withdrawal eligibility and updates the student's course enrollment accordingly.

- **Stimulus**: A student confirms course withdrawal.
  Response: The system updates the student's academic record.

4.10.3 Functional Requirements

- **REQ-1**: The system shall allow students to initiate the course withdrawal process.
- **REQ-2**: The system shall update the student's academic record and course enrollment status after a successful withdrawal.
- **REQ-3**: The system shall ensure that withdrawing from a course does not result in the student violating academic policies (e.g., dropping all courses).

4.11 System Feature 11: Managing Course Feedback

4.11.1 Description and Priority

This feature allows students to provide feedback on courses and instructors, and enables the system to store and organize this feedback for quality improvement purposes.
Priority: Medium

4.11.2 Stimulus/Response Sequences

- **Stimulus**: A student submits feedback for a course or instructor.
  Response: The system stores the feedback and acknowledges receipt.

- **Stimulus**: An instructor or administrator reviews feedback received for a course.
  Response: The system retrieves and displays the feedback data in a structured format.

4.11.3 Functional Requirements

- **REQ-1**: The system shall allow students to provide feedback for courses and instructors.
- **REQ-2**: The system shall store the feedback in a secure database.
- **REQ-3**: The system shall allow authorized users (e.g., instructors, administrators) to view and analyze the feedback.

4.12 System Feature 12: Add and Remove Courses for Instructors

4.12.1 Description and Priority

This feature allows instructors to add new courses or remove existing courses from database. It helps instructors manage their course offerings and ensures that course assignments remain up to date.
Priority: High

4.12.2 Stimulus/Response Sequences

- **Stimulus**: An instructor adds a new course to their teaching schedule.
  Response: The system validates the instructor's permissions and adds the new course to their list of assigned courses.

- **Stimulus**: An instructor removes an existing course from their teaching schedule.
  Response: The system validates the instructor's permissions and removes the course from their list of assigned courses, ensuring all associated data (e.g., student enrollments) is updated.

4.12.3 Functional Requirements

- **REQ-1**: The system shall ensure that when a course is removed, all related student enrollments are also updated accordingly.
- **REQ-2**: The system shall allow instructors to view a list of their current courses and make necessary adjustments.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

- *The system should respond to user actions (e.g., logging in, submitting forms) within **2 seconds** under normal load.*
- *The database should handle up to **10,000 concurrent users** without performance degradation.*
- *Attendance and marks data queries must execute within **1 second** for up to **1,000 records**.*
- *Image uploads for profile pictures should complete within **5 seconds**, even under high server load.*
- *The application must maintain **99.9% uptime** over a rolling 30-day period.*

5.2. Safety Requirements

- *Prevent data corruption or loss during server crashes by implementing regular database backups every **12 hours**.*
- *Ensure that invalid or incomplete data cannot be submitted (e.g., missing fields in attendance or marks forms).*
- *Ensure secure handling of sensitive user data (e.g., passwords, profile pictures) to prevent unauthorized access.*
- *Adherence to regulatory safety requirements, such as **GDPR** or **FERPA** compliance, depending on the user base.*

5.3. Security Requirements

- *All communication between client and server must be encrypted using **SSL/TLS** protocols.*
- *Passwords must be hashed using a secure algorithm (e.g., **bcrypt**) before storage.*
- *Implement role-based access control (RBAC) to restrict teacher, student, and admin actions.*
- *Maintain audit logs of actions such as marks updates, attendance modifications, and course registrations.*
- *Limit user sessions to **1 hour** of inactivity, requiring reauthentication thereafter.*
- *Ensure profile pictures and other uploaded files are scanned for malware before storage.*
- *Compliance with industry standards for student data protection (e.g., **FERPA** in the U.S.).*

5.4. Software Quality Attributes

- ***Usability:** The system must be intuitive and require no more than **5 minutes of training** for basic tasks.*
- ***Reliability:** System downtime must not exceed **43 minutes per month**.*

- ***Maintainability:*** *Code should be modular, with clear documentation to enable updates within* ***2 weeks*** *for any new features.*
- ***Interoperability:*** *APIs must allow integration with other learning management systems (LMS).*
- ***Scalability:*** *The application must support scaling up to* ***50,000 active users*** *across multiple regions.*
- ***Testability:*** *The system must pass automated unit and integration tests with* ***90% code coverage*** *before deployment.*
- ***Portability:*** *The application must run on all major modern browsers (Chrome, Firefox, Edge, Safari).*
- ***Robustness:*** *The system must gracefully handle invalid inputs or network failures without crashing.*

5.5. Business Rules

- *Only instructors can update marks, attendance, or course details.*
- *Students can only view their own marks, attendance, and profile data.*
- *Admins can add/remove users, assign instructors to courses, and view all system data.*
- *Course registrations for students must close* ***1 week*** *after the semester starts.*
- *Attendance data must be locked for editing after* ***48 hours*** *of submission to ensure integrity.*

6. Other Requirements

- ***Database***
    - *Use PostgreSQL with optimized queries, foreign key constraints, and 12-hour backups retained for 30 days.*
- ***Internationalization***
    - *Support English with a future option for multiple languages. Adhere to ISO 8601 for date/time formats.*
- ***Legal***
    - *Comply with GDPR, FERPA, and local data protection laws. Obtain user consent for storing personal data.*
- ***Reuse***

    Modular architecture with reusable UI components and RESTful backend for integration in future projects.

- ***Scalability***
    - *Support new features like online exams and real-time notifications with minimal changes.*
- ***Audit and Logging***
    - *Maintain secure logs of user actions and errors, retained for 6 months.*

Appendix A: Glossary

General Terms

1. ***Frontend**: The part of the application that the user interacts with directly, typically built using technologies like React, HTML, CSS, and JavaScript.*
2. ***Backend**: The server-side logic and database interactions that power the application, typically built using Node.js, Express.js, and PostgreSQL.*
3. ***Database**: A structured system to store, manage, and retrieve data. In this project, PostgreSQL is used as the database.*
4. ***HTTP Request**: A communication mechanism used to send data from the frontend to the backend or vice versa.*
5. ***API (Application Programming Interface)**: A set of rules that allows communication between the frontend and backend. APIs use endpoints for specific functionalities.*
6. ***CRUD Operations**: Basic database operations: Create, Read, Update, Delete.*

Specific to the Project

1. ***Course Management**: A feature that allows teachers to manage courses they are assigned to, including accepting courses, viewing course details, and managing marks or attendance.*
2. ***Evaluation**: Refers to different methods of assessing student performance, such as quizzes, assignments, sessional exams, and final exams.*
3. ***Dynamic Fields**: Fields in the frontend that can be added or removed dynamically, e.g., adding quizzes or assignments for a course.*
4. ***Marks Distribution**: A feature where teachers assign and update marks for quizzes, assignments, sessionals, and finals.*
5. ***Attendance Management**: A feature that allows teachers to mark and manage student attendance.*

Database Tables

1. ***Course_Marks**: A table that stores information about students' marks related to specific courses and evaluations.*
    - ***Fields**: `Std_id`, `Course_ID`, `Instructor_ID`, `Evaluation_ID`.*
    - ***Evaluation_ID**: Used to identify whether the data corresponds to quizzes, assignments, sessionals, or finals.*
2. ***Marks_Evaluation**: A table that maps evaluation IDs to specific question IDs.*
    - ***Fields**: `Evaluation_ID`, `Question_ID`.*

3. ***Question****: A table containing detailed information about each question in quizzes, assignments, or exams.*
    - ***Fields****:* `Question_ID`*,* `Question_Number`*,* `Weightage`*,* `Obtained_Marks`*,* `Total_Marks`*.*

Acronyms and Abbreviations

1. ***ID****: Identifier – A unique value assigned to identify an entity (e.g., student, course, instructor).*
2. ***CRUD****: Create, Read, Update, Delete – Basic operations for handling data.*
3. ***HTTP****: Hypertext Transfer Protocol – The protocol used for communication between the frontend and backend.*
4. ***SQL****: Structured Query Language – A language used to interact with relational databases like PostgreSQL.*
5. ***JSON****: JavaScript Object Notation – A lightweight data format used for transferring data between the frontend and backend.*
6. ***API****: Application Programming Interface – A way for two software systems to communicate.*
7. ***DOM****: Document Object Model – A programming interface for HTML and XML documents.*

Relevant Technologies

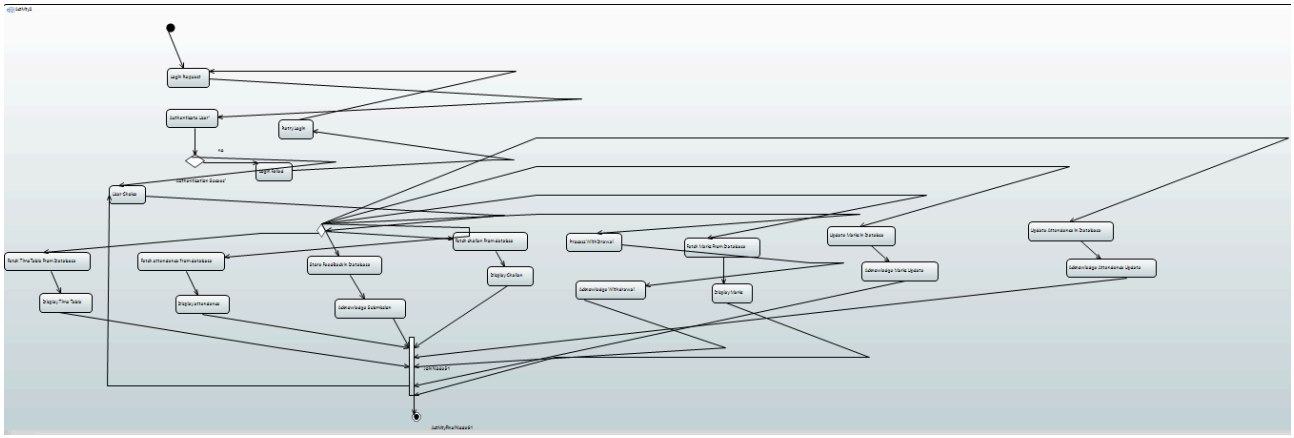
1. ***React.js****: A JavaScript library for building user interfaces.*
2. ***Node.js****: A runtime environment for executing JavaScript on the server.*
3. ***Express.js****: A web application framework for Node.js to build APIs.*
4. ***PostgreSQL****: A powerful, open-source relational database system.*
5. ***Axios****: A JavaScript library for making HTTP requests from the frontend to the backend.*
6. ***CORS (Cross-Origin Resource Sharing)****: A security mechanism allowing controlled access to resources on a server from different origins.*
7. ***REST (Representational State Transfer)****: An architectural style for designing APIs.*

Error Handling Terms

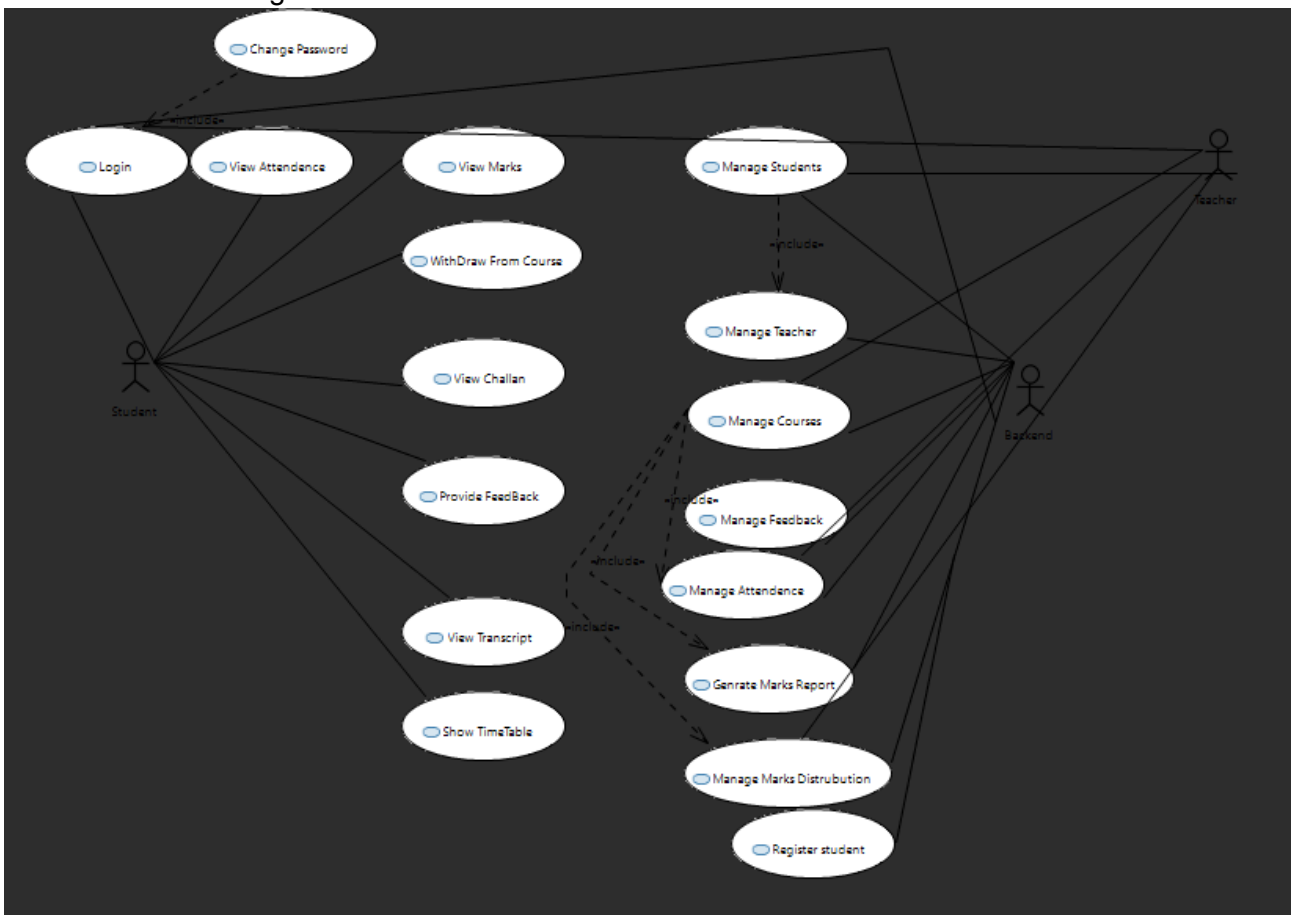
1. ***Error 500 (Server Error)****: Indicates an internal server issue.*
2. ***Error 400 (Bad Request)****: Indicates an issue with the client request, such as missing or incorrect data.*
3. ***ON CONFLICT****: A PostgreSQL clause to handle duplicate key violations during* `INSERT` *queries.*
4. ***EXCLUDED****: A PostgreSQL keyword used in* `ON CONFLICT` *clauses to reference the conflicting rows.*

Appendix B: Analysis Models

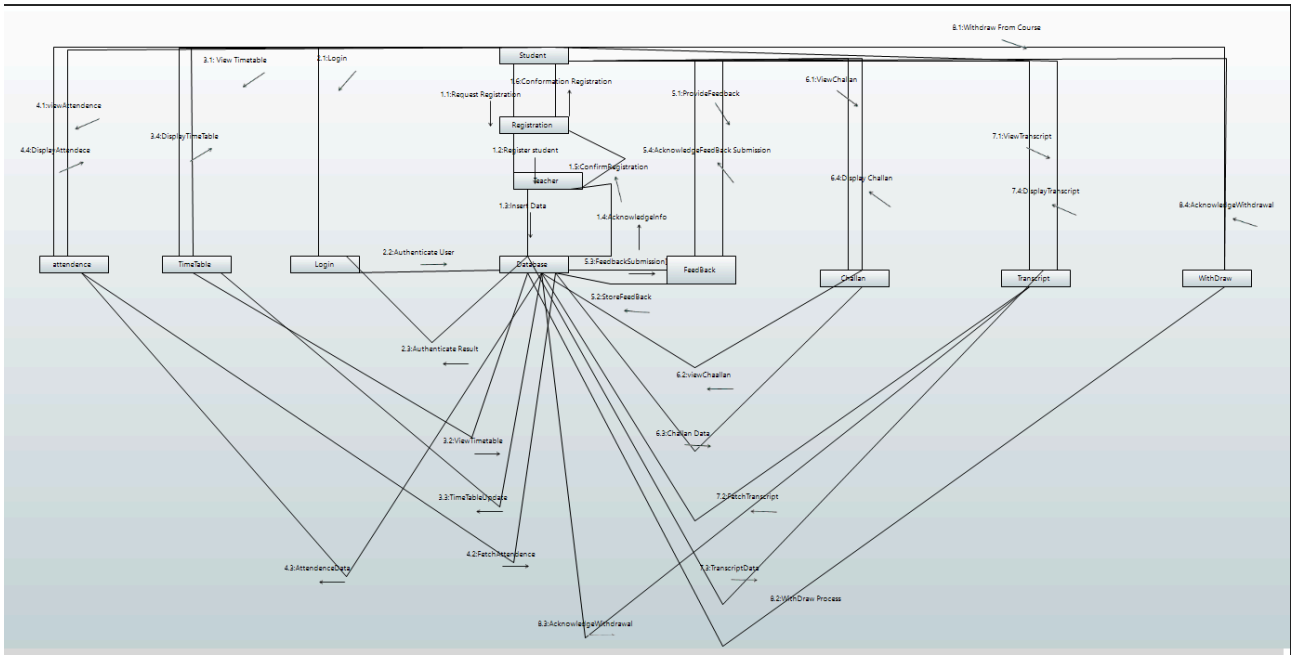
Activity Diagram:



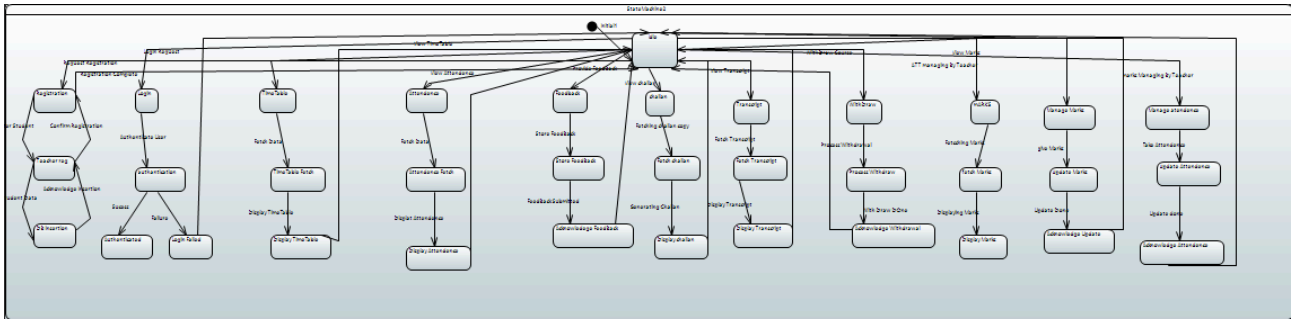
UseCase Diagram:



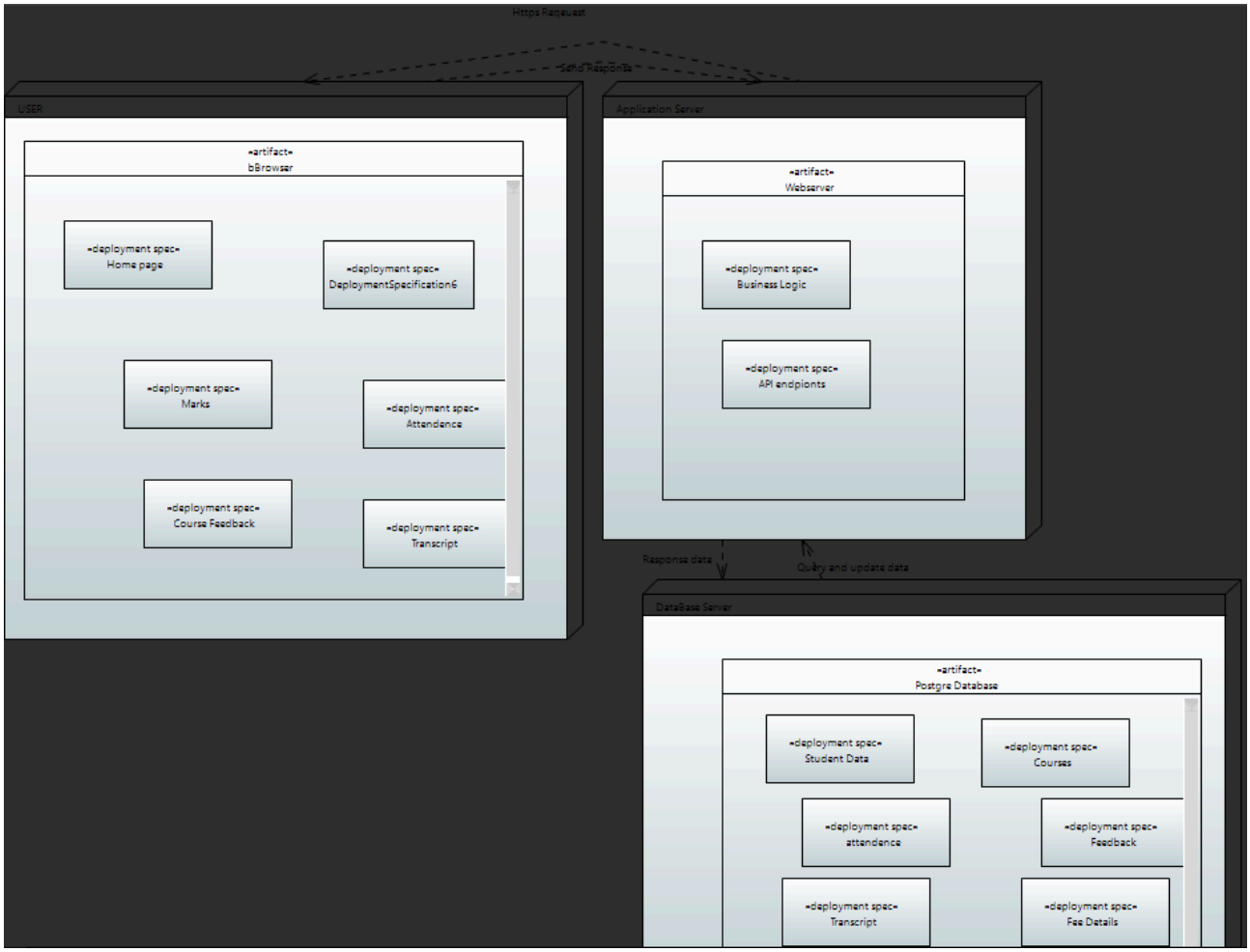
Communication Diagram:



State Machine Diagram:



Deployment Diagram:



Appendix C: To Be Determined List

- ***TBD #1: Authentication Mechanism***
    - *Determine the authentication method (e.g., OAuth, JWT, session-based authentication) to secure user access and ensure proper session management.*
- ***TBD #2: Role Hierarchy and Permissions***
    - *Finalize the roles (Admin, Teacher, Student) and their associated permissions to define system access and actions clearly.*
- ***TBD #3: Dynamic Field Limits for Marks Distribution***
    - *Define the maximum number of dynamic fields (e.g., quizzes, assignments) a teacher can add for marks distribution to prevent performance or usability issues.*
- ***TBD #4: Data Retention Policy***
    - *Establish how long attendance records, marks data, and profile pictures will be stored in the database before archiving or deletion to comply with data regulations.*