Learning Management System
System Design Document

CSCI-P465/565 (Software Engineering I)

Project Team

Soham Atul Pingat

Rohith Pavuluru

Saicharan Reddy Kotha

Owen Miller

Nihal Shetty

---

1. Introduction

1.1. System Description

The objective of this project is to create a learning management system. System that offers students a centralized hub for course registration, accessing course details, and interacting with professors and peers. It will provide a consolidated view of course information, including assignments, messages, and grades. Additionally, faculty members will have access to features like viewing their courses and registered students. Oversight of student and professor accounts will be managed by the Registrar, serving as the administrator.

1.2. Design Evolution

1.2.1. Design Issues

It is a website which is better when browsed on a computer rather than a phone, despite a phone browser being how most of the people access websites nowadays.

1.2.2. Candidate Design Solutions

The team has opted to utilize a framework like React for the frontend aspect of our project. This decision was primarily influenced by React's component-based nature and its reputation for user-friendly development. By embracing React's component-based approach, we anticipate streamlining our development process, enhancing code modularity, and simplifying maintenance and scalability tasks

Likewise, the backend team has chosen to implement Firebase as our primary database solution. This decision stems from Firebase's well-established reputation for reliability and its real-time database capabilities. Firebase offers a scalable infrastructure that facilitates seamless data storage, synchronization, and retrieval across various platforms and devices.

1.2.3. Design Solution Rationale

Choosing React for the frontend was driven by its component-based architecture and ease of use. This approach enhances code modularity, simplifies development, and improves scalability, meeting our project's requirements efficiently.

Firebase was selected as the primary database for its reliability and real-time capabilities. Its NoSQL database offers flexibility, while real-time synchronization ensures seamless data updates across devices. With built-in authentication and cloud functions, Firebase streamlines backend development, addressing design challenges in data management and integration effectively.

1.3. Design Approach

1.3.1. Methods

We utilized object-oriented design principles to structure our codebase, ensuring modularity and reusability while promoting easier maintenance and scalability. And the customer approved it.

1.3.2. Standards

We're using Camel Naming Convention Our design adhered to industry-standard naming conventions and followed a clear structure and hierarchy of components, ensuring consistency and readability throughout the codebase. Additionally, we incorporated safety standards for data handling and authentication processes to safeguard user information effectively.

1.3.3. Tools

1. VSCode
2. Postman
3. NodeJs
4. ReactJs
5. Github
6. JIRA
7. Firebase
8. Microsoft Teams

2. System Architecture

2.1. System Design

2.1.1. **User Interface (UI):** Front-end interface for students, instructors, and admins, providing access to course materials, announcements, and communication tools.

2.1.2. **Authentication and Authorization:** Manages user authentication and permissions, ensuring secure access to system features based on user roles.

2.1.3. Dashboard and Navigation: Personalized dashboards for users, displaying course information, assignments, and communication options based on their roles.

2.1.4. **Course Management:** Allows instructors to create, manage, and publish courses, including features for adding content, creating assignments, and grading submissions.

2.1.5. **Search and Filtering:** Enables users to search for specific resources within the LMS, with filter options to refine search results based on various parameters.

2.1.6. **Chat and Communication:** Real-time chat functionality for users to engage in private or group chats, check online status, and receive message notifications.

2.2. External Interfaces

2.2.1. OAuth Providers (e.g., Google):

1. Description: Enables user authentication through Google or Facebook accounts.
2. Relationship with System: Facilitates seamless login experience for users.
3. Communication Details: Utilizes OAuth protocol for authentication, with React frontend initiating requests to OAuth providers via Firebase authentication APIs.

2.2.2. Third-Party OTP Provider (e.g., Duo):

1. Description: Enhances security for password reset/recovery through OTP verification.
2. Relationship with System: Adds an extra layer of authentication for account recovery.
3. Communication Details: React frontend communicates asynchronously with the third-party OTP provider's API via NodeJS backend to generate and validate OTPs.

2.2.3. External Database for User Information:

1. Description: Stores user data such as profiles and credentials.
2. Relationship with System: Serves as a centralized repository for user information.
3. Communication Details: NodeJS backend interacts with Firebase database for user data retrieval and storage using Firebase SDKs.

3. Component Design

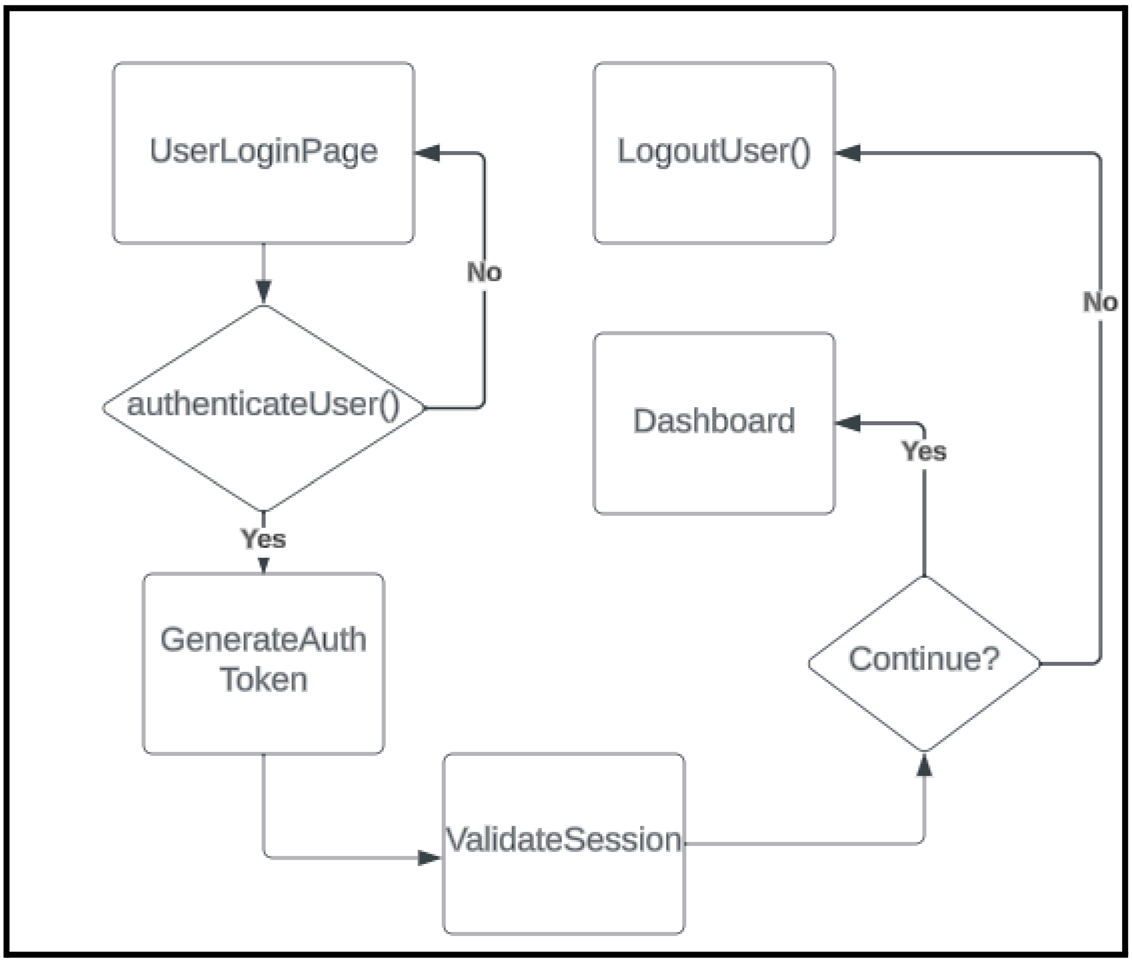
3.1. User Authentication Component

3.1.1. **Component Name:** UserAuthComponent

3.1.2. **Component Description:** Handles user authentication processes, including login verification, session management, and role-based access control. Operational flow involves user authentication, token generation, session validation, and logout. This component operates independently within the system's execution loop.

3.1.3. **Responsible Development Team Member:** Frontend Team (Rohith and Saicharan), Backend Team (Nihal, Owen and Soham), DB Team (Soham)

3.1.4. **Component Diagram:**



3.1.5. **Component User Interface:** Login and Signup

3.1.6. **Component Objects**:

1. UserAuthentication:
    - Data Members: email, password
    - Methods: authenticateUser(), generateAuthToken(), validateSession(), logoutUser()
    - Interface with External Interfaces: Firebase authentication APIs for user authentication.

3.1.7. **Component Interfaces**:

1. Internal Interface: Communicates with Firebase authentication APIs.
2. External Interface: Interacts with Firebase for user authentication and session management.

3.1.8. **Component Error Handling**:

1. Error Case 1: Invalid Credentials - authenticateUser() method verifies input credentials.
2. Error Case 2: Session Timeout - validateSession() method checks session validity.

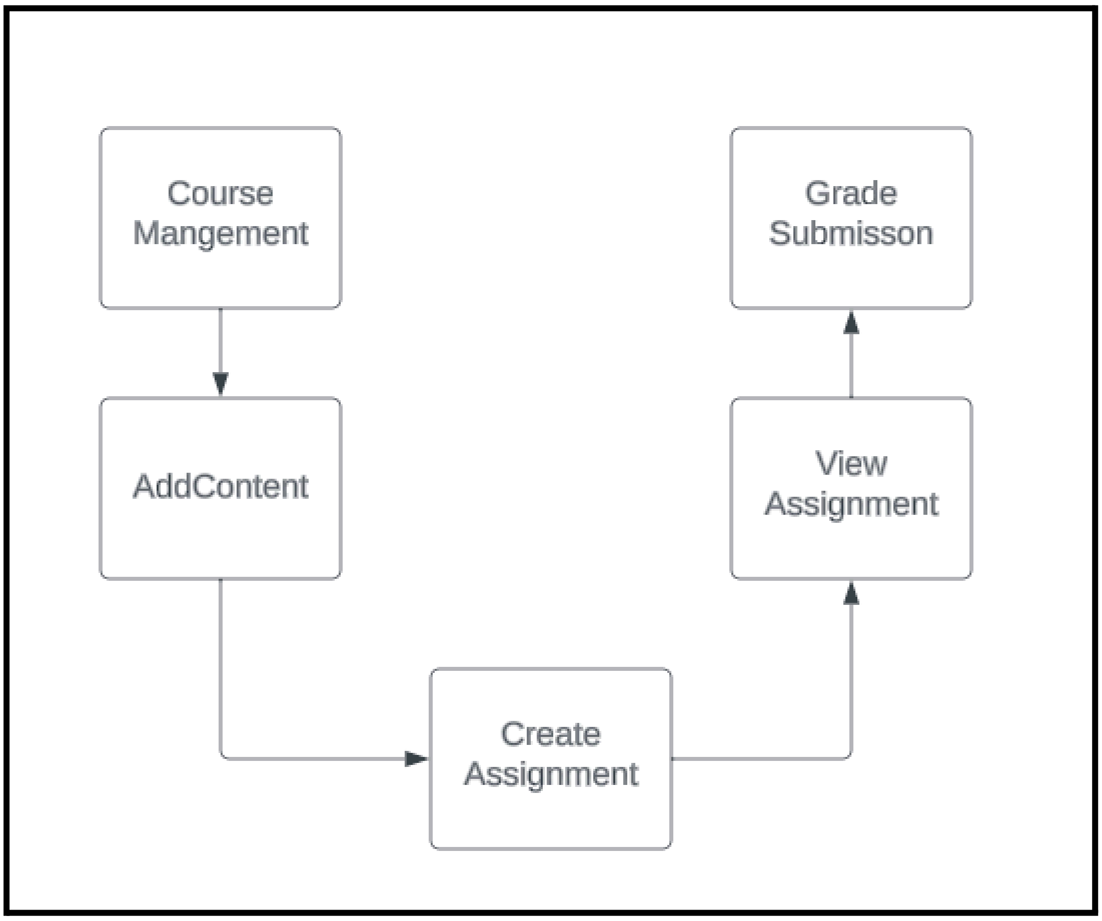
3.2. Course Management Component

3.2.1. **Component Name**: CourseManagementComponent

3.2.2. **Component Description**: Facilitates course creation, content management, assignment creation, student enrollment, and grading. Operational flow includes course setup, content addition, assignment creation, student enrollment, and grading. Operates independently within the main execution loop.

3.2.3. **Responsible Development Team Member**: Frontend Team (Rohith and Saicharan), Backend Team (Nihal, Owen and Soham), DB Team (Soham)

3.2.4. **Component Diagram:**



3.2.5. **Component User Interface:** Course creation, content management, assignment creation, student enrollment, grading.

3.2.6. **Component Objects**:

1. Course:
   - Data Members: courseID, courseTitle, instructorID, startDate, endDate, enrolledStudentsList
   - Methods: createCourse(), addContent(), createAssignment(), enrollStudent(), gradeSubmission()
   - Interface with External Interfaces: Firebase database for data storage.

3.2.7. **Component Interfaces**:
1. Internal Interface: Communicates with Firebase for course data management.
2. External Interface: Interacts with Firebase for course data storage and retrieval.

3.2.8. **Component Error Handling**:
1. Error Case 1: Duplicate Course Creation - createCourse() method prevents duplicate titles.
2. Error Case 2: Invalid Assignment Deadline - createAssignment() validates deadlines.

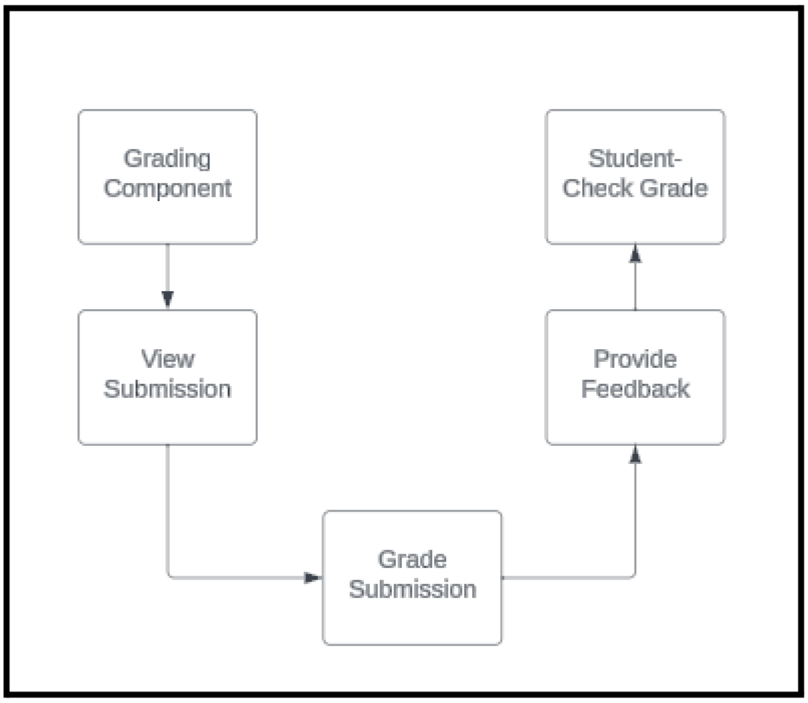
3.3. Grading Component

3.3.1. **Component Name**: GradingComponent

3.3.2. **Component Description**: Manages assignment grading and feedback provision. Operational flow includes viewing assignments, grading submissions, and providing feedback. Operates independently within the main execution loop.

3.3.3. **Responsible Development Team Member**: Frontend Team (Rohith and Saicharan), Backend Team (Nihal, Owen and Soham), DB Team (Soham)

3.3.4. **Component Diagram**:



3.3.5. **Component User Interface**: Submission viewing, grading, feedback provision.

3.3.6. **Component Objects**:
1. Grading:
   - Data Members: assignmentID, studentID, grade, feedback
   - Methods: viewSubmission(), gradeSubmission(), provideFeedback()

- Interface with External Interfaces: Firebase database for data storage.

3.3.7. **Component Interfaces**:

1. Internal Interface: Communicates with Firebase for grading data management.
2. External Interface: Interacts with Firebase for grading data storage and retrieval.

3.3.8. **Component Error Handling**:

1. Error Case 1: Missing Submission - viewSubmission() method handles missing submissions gracefully.
2. Error Case 2: Invalid Grade - gradeSubmission() method ensures grades fall within acceptable ranges.

4. Revision History:

Revision	**Date**	**Change Description**
Initial Design Doc	2/18/2024	None