Wisdom Arena

Software Design

CSCI-P465/565 (Software Engineering I)

Project Team

Abigail Lindvall
William Cesaretti
Prithvi Amin
Sai Praneeth Chapala
Venkata Naga Sreya Kolachalama

---

1. Introduction

The design approach for our Learning Management System (LMS), developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js), focuses on creating a robust, scalable, and user-friendly platform. This LMS is aimed at providing an intuitive and comprehensive online learning environment for educators and learners.

1.1 System Description

The LMS is designed to address the growing need for accessible, efficient, and flexible online education solutions. It allows educators to create, manage, and deliver educational content, and enables students to access learning materials, submit assignments, and interact with peers and instructors.

1.2 Design Evolution

1.2.1 Design Issues

The design of the LMS was driven by several key requirements and constraints, including:

Scalability: The need to accommodate an increasing number of users and courses without degradation of performance.

User Accessibility: Ensuring that the system is easily navigable and accessible remotely from various devices and browsers.

Interactivity: Facilitating interactive learning experiences through discussion forums, real-time feedback, and multimedia resources.

Data Security: Protecting user data and course content with robust security measures.

1.2.2 Candidate Design Solutions

We looked into three main ways to build our LMS:

All-in-One System: Everything is built together as one big program, which makes it easier to set up and manage but can be a bit rigid as it grows.

Microservices: Instead of one big program, we build lots of smaller, independent pieces that work together. This is great for growing and changing parts of the system without affecting everything else.

Combined Approach: We use a mix of the first two methods. We keep some parts together for simplicity and break out other parts into smaller pieces for flexibility.

1.2.3 Design Solution Rationale

After considering our options, we decided on the combined approach for a few reasons:

Ease of Use: The all-in-one part keeps it simple where we don't need much change, making it easier to manage day-to-day operations.

Flexibility: For parts of the LMS that we expect to grow or need to update often, like video content or chats, we use the microservices approach. This lets us update and scale these features independently, which is really handy.

Best of Both Worlds: This way, we get the simplicity of a single system where it benefits us and the flexibility of microservices where it counts, making our LMS robust yet adaptable.

1.3 Design Approach

1.3.1 Methods

Prototyping

Prototyping is crucial for visualizing the user interface and experience early in the development process. It allows us to create preliminary versions of the LMS to explore and validate concepts, design choices, and usability with potential users.

Object-Oriented Design (OOD)

By modeling components as objects that encapsulate both data and behavior, we enhance code reusability and make the system more manageable. OOD facilitates clear separation of concerns, making the system easier to develop, maintain, and scale over time

Agile Methodology

This approach aligns with our use of prototypes and allows for continuous feedback and adaptation of the design, ensuring that the final product meets user needs and can evolve with emerging requirements.

1.3.2 Standards

The design adheres to several standards, including:

Web Accessibility Standards: Ensuring that the system is accessible to all users, including those with disabilities.
Responsive Design Standards: Guaranteeing a seamless user experience across different devices and screen sizes.
RESTful API Standards: For the development of web services, ensuring they are stateless, scalable, and easily consumable.
Coding and Naming Conventions: Following best practices for code readability, maintainability, and consistency across the development team.

1.3.3 Tools

The development team plans to use a range of tools, including:

Visual Studio Code: For coding, with integrated support for debugging and version control.
Postman: For designing, testing, and documenting APIs.
Docker: For containerization, ensuring consistent environments across development, testing, and production.
GitHub: For version control and collaboration.
JIRA: For project management and tracking tasks and issues.
Zoom, Slack: Communication within the team.

2. System Architecture

2.1 System Design

The high-level design of our Learning Management System (LMS) adopts a layered architecture, ensuring separation of concerns and facilitating scalability and maintenance. The system is composed of several major components, including:

Client Layer (Frontend): Developed using React.js, this layer provides the user interface through which users interact with the LMS. It includes pages and components for course management, content delivery, assignments, and user profiles. It will be styles with one globally applied CSS file to overall styling and each page will have a corresponding CSS file for a few specific changes.
Server Layer (Backend): Utilizing Node.js and Express.js, this layer handles business logic, authentication, authorization, data processing, and serves API requests from the frontend.
Database Layer: MongoDB is used for data storage, including user information, course content, assignments, and grades.
External Interfaces: These include integrations with third-party services such as OTP on email for authorization, Google and Facebook OAuth,and Cloudinary for course videos.

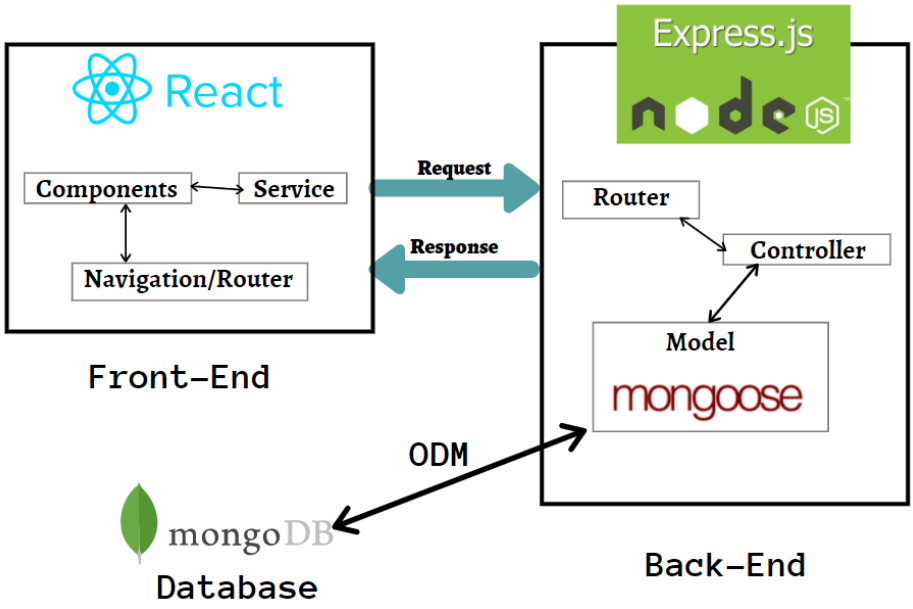

Figure 1

2.2 External Interfaces

OTP on Email for Authorization
Description: Sends one-time passwords to users' emails for secure authorization actions within the LMS.
Communication: Utilizes SMTP to send emails containing OTPs. The LMS backend sends a request with the user's email and OTP content to the email service provider, which then delivers the email.

Google and Facebook OAuth
Description: Enables users to sign in using their Google or Facebook accounts, streamlining the login process.
Communication: Implements OAuth flow by redirecting users to authenticate with Google or Facebook, then exchanges authorization codes for access tokens to retrieve user profile information.

Cloudinary for Course Videos
Description: Manages and streams course videos, offering a scalable solution for video hosting.
Communication: The LMS backend uses Cloudinary's API for video uploads and storage, sending video files and metadata via HTTP POST requests. For streaming, it embeds Cloudinary-generated video URLs in course materials.

3. Component Design

- **Component Name**
  USER MANAGEMENT COMPONENT

- **Component Description**
  The User Management Component is crucial for handling all aspects related to user accounts within the Learning Management System (LMS), including registration, authentication, authorization, profile management, and role assignment. It ensures secure access to the system, enforcing role-based access controls to different functionalities and resources. This component integrates with external authentication services like Google and Facebook OAuth for login convenience and supports OTP via email for secure operations.

  It operates as part of the main execution loop and interacts closely with the Database Layer for storing and retrieving user data, and with External Interfaces for authentication services and email notifications.

  Operational Flow:
  New users register, providing necessary details and choosing authentication methods. Users log in to the system through either standard authentication or external OAuth

services. User may forget password and use a OTP code to change their password. User may then log in with the new password. Google login is working, pending Facebook.
The system manages session tokens for active users, handling session expiration and renewal.
Administrators assign or modify user roles (e.g., student, instructor) and manage user access.

- **Responsible Development Team Member**
  Prithvi

- **Component Diagram**

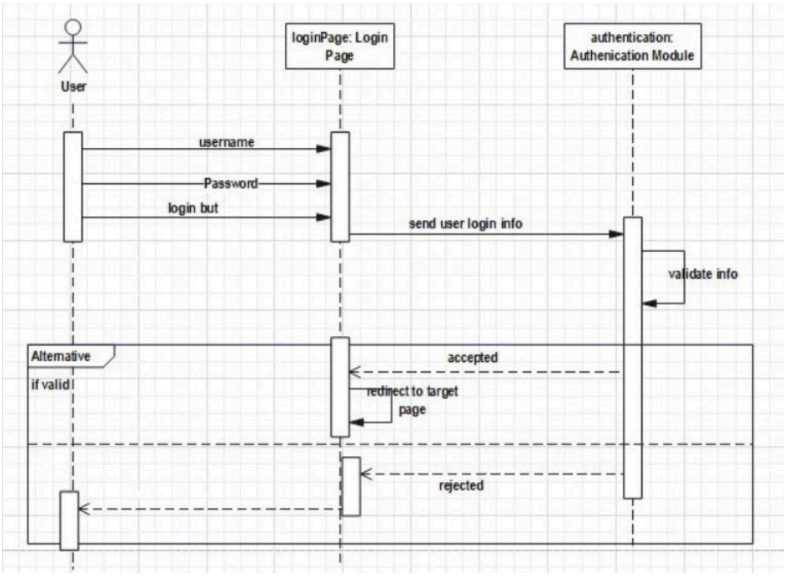

Figure 2

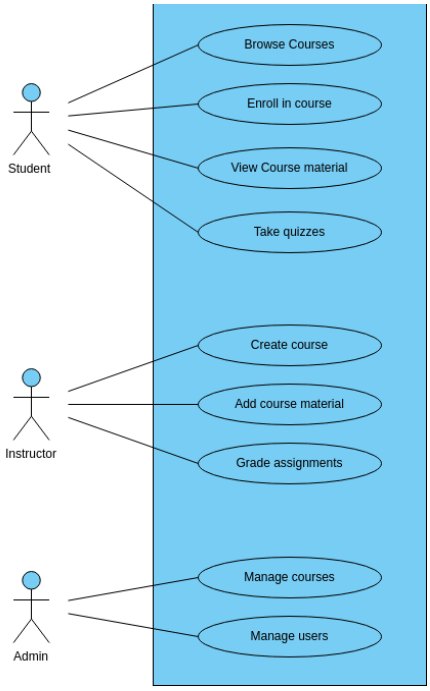

Figure 3

- ***Component User Interface***
  This component's interface features include login and registration forms, user profile management pages, and admin interfaces for user role management.

  Login/Registration Screen: Secure forms for user authentication and new user sign-up.

  Profile Management: Users can update their personal information and preferences.

  Admin User Management: Interface for admins to assign roles, deactivate accounts, and monitor user activities.

- **Component Objects**
  User: Includes data members such as userID, name, email, password, and role. Methods cover create, update, delete, and authenticate.

  Role: Manages role definitions and permissions, with methods to assign and modify roles.

  Authentication: Handles login logic, session management, and integration with external OAuth providers.

  Interactions:
  Interacts with the Database Layer for CRUD operations on user data.

Utilizes External Interfaces for OAuth authentication and sending OTP emails.
Explicitly calls methods of the Session Manager object for handling user sessions.

- **Component Interfaces (internal and external)**
  Uses RESTful APIs for communication with front-end components, exchanging user data in JSON format.
  Integrates with OAuth providers for external authentication, handling access tokens and user profile fetching.
  Interacts with an email service provider for sending OTPs, using SMTP protocols.

- **Component Error Handling**

  Input Validation: Ensure all user input meets security standards to prevent injection attacks.

  Role Authorization Checks: Verify user roles before granting access to protected resources or functionalities.

  JWT error checks: Checks if the token is expired or if it's a wrong token.

- **Component Name**
  COURSE MANAGEMENT COMPONENT

- **Component Description**
  The Course Management Component is responsible for all operations related to course content within the LMS. It handles the creation, update, deletion, and retrieval of course information, including syllabus, modules, lessons, and associated resources. This component supports multimedia content, enabling instructors to enrich their course offerings with videos, documents, and interactive tools.

  It operates within the main execution loop, interfacing with both the Database Layer for data persistence and External Interfaces for content like videos and documents.

  Operational Flow:
  Admin can create new courses, specifying details like title, description, and resources.
  Instructors can see the courses assigned and also the students enrolled.
  Students can enroll, drop and view the courses.
  Courses can be updated with new content or modified based on feedback.

Instructors can delete courses that are no longer offered.
Students retrieve course content for their enrolled classes.

Content organization uses a tree structure algorithm for efficient management and navigation. Access control checks ensure that users can only interact with courses according to their roles.

- **Responsible Development Team Member**
  Sai Praneeth Chapala

- **Component Diagram**

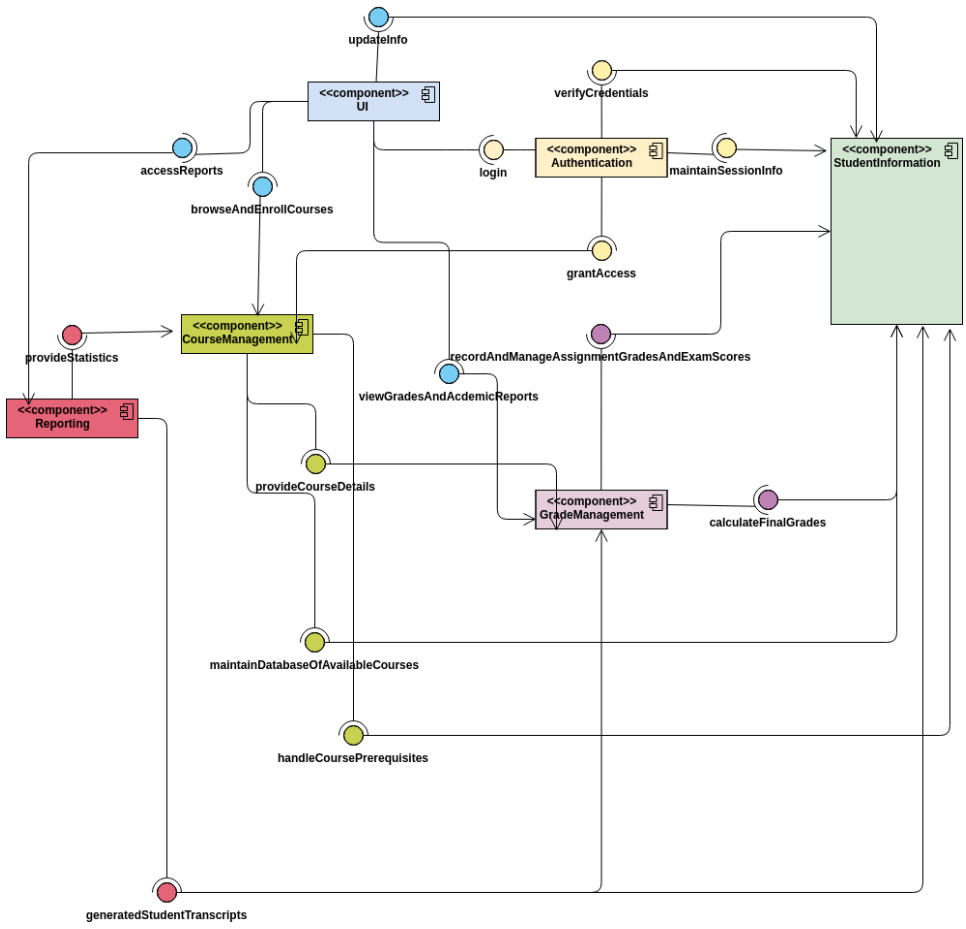

Figure 4

- **Component User Interface**
  The interface allows instructors to manage courses through a dashboard, where they

can add, update, and delete courses and content. Students access courses through a separate view, displaying available resources and modules.

Instructor Dashboard: Create/update/delete courses, upload content.
Student Course View: List of courses, access to content, download resources

- **Component Objects**

  Course: Data members include course ID, title, description, and list of modules. Methods include create, update, delete, and retrieve.
  Module/Lesson/Resource: Similar structure to Course but tailored to their respective content types.

  Interactions:
  Creates/alters/deletes files when updating course content.
  Calls methods of the Database Layer object for data persistence.
  Receives data from the User Management Component for role-based access.

- **Component Interfaces (internal and external)**
  Interfaces with the Database Layer using MongoDB queries for CRUD operations. Exchanges JSON structures with External Interfaces like Cloudinary for video content management. Utilizes RESTful API conventions for communication with other system components.

- **Component Error Handling**

  Input Validity Check: Verifies course and content details to be within accepted ranges and formats.

  Access Control: Ensures users can only perform actions aligned with their roles.

  Data Persistence: Implements transactional operations to prevent data corruption during updates.

- **Component Name**
  FEEDBACK MANAGEMENT COMPONENT

- **Component Description**
  The Feedback System component is designed to collect, manage, and display feedback from users (both students and instructors) about courses, lessons, and overall learning experience within the Learning Management System (LMS). It enables users to rate content, write reviews, and submit suggestions.

  This component integrates into the LMS's main execution loop, facilitating real-time feedback that instructors can use to improve course content and students can use to make informed decisions about their learning paths.

- **Responsible Development Team Member**
  Venkata Naga Sreya Kolachalama

- **Component Diagram**

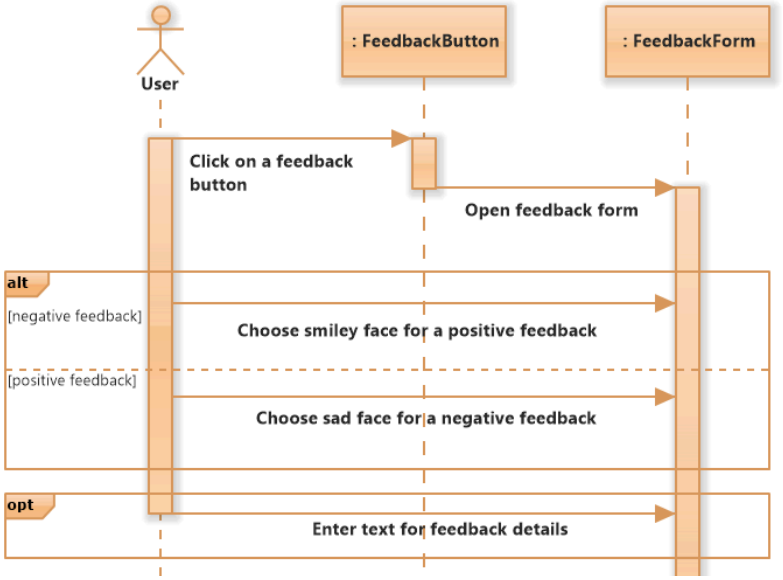

Figure 5

- **Component User Interface**
  Feedback Submission Form: Allows users to rate courses/lessons and leave comments.

  Feedback Dashboard: For instructors, displays received feedback for their courses, including average ratings and student comments.

  Notifications: Users receive notifications via email or dashboard for submitted feedback acknowledgment and when their feedback leads to changes in courses.

- **Component Objects**

Feedback Entry: Represents a single feedback item. Data members include userID, courseID, rating, comment. Methods include submit, edit, and delete.

Feedback Manager: Manages operations related to feedback collection and retrieval.

Interactions:
Submits feedback to the Database Layer (creates/alters feedback entries).
Calls the User Management Component to verify user identity and permissions.
Sends notifications through External Interfaces.

- **Component Interfaces (internal and external)**
  RESTful API endpoints for feedback submission and retrieval.
  Database queries for storing and accessing feedback data.
  SMTP protocol for sending email notifications regarding feedback.

- **Component Error Handling**

  Invalid Feedback Submission: Ensures that all feedback submissions include valid ratings and non-empty comments. If validation fails, the user is prompted to correct the input.

  Unauthorized Access Attempt: Verifies user permissions before allowing feedback submission or editing, preventing unauthorized feedback manipulation.

  Database Operation Failure: Implements retry logic and transactional integrity checks for database operations to prevent data corruption and ensure feedback is accurately recorded and retrieved.

- **Component Name**
  GRADING AND ASSESSMENT COMPONENT

- **Component Description**
  The Grading and Assessment System component facilitates the submission of assignments by students, the grading of these submissions by instructors, and the management of grades within the Learning Management System (LMS). It supports various types of submissions (text, files, links) and grading schemes (points, percentages, letter grades).

  This component integrates with the LMS's core functionality, allowing for seamless interaction between course content, student submissions, and feedback.
  .

- **Responsible Development Team Member**
  William Cesaretti

- **Component Diagram**

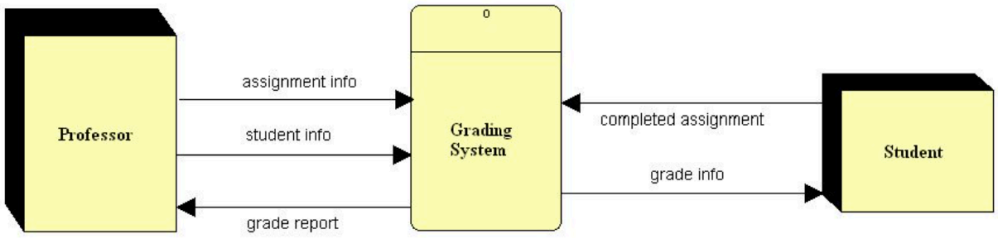

Figure 6

- **Component User Interface**
  Instructor can view can create assignments
  Assignment Submission Page: Allows students to submit their work, view submission deadlines, and track their submission history.

  Grading Interface: Enables instructors to review submissions, assign grades, and provide feedback.

  Gradebook: A dashboard for both students and instructors to view grades, feedback, and overall progress in a course.

- **Component Objects**

  Assignment Submission: Represents a student's submission, including studentID, assignmentID, submissionType (text/file/link), submissionContent, and timestamp. Methods include submit, update, and withdraw.

  Grade: Manages grading information, with data members for assignmentID, studentID, gradeValue, and feedback. Methods include assignGrade and editGrade.

  Interactions:
  Submits assignments and grades to the Database Layer.
  Verifies user roles and permissions via the User Management Component.
  Sends out notifications of submission deadlines and grade postings through External Interfaces.

- **Component Interfaces (internal and external)**
  RESTful APIs for handling submissions, grading operations, and accessing grades.
  Database queries for CRUD operations related to submissions and grades.
  Email and SMS notifications for reminding about deadlines and informing about new grades.

- **Component Error Handling**

  Submission Deadline Passed: Prevents submission of assignments past their deadline, providing an error message to the student.

  Unauthorized Grading Attempt: Checks instructor permissions before allowing grade entries or modifications to ensure only authorized personnel can grade assignments.

  Incomplete Submission Data: Validates submission completeness and format, prompting the user to correct missing or invalid data before submission can proceed.

- **Component Name**
  COLLABORATIVE COMMUNICATION COMPONENT

- **Component Description**
  The Collaborative Communication System component is designed to support chats and group discussions within the Learning Management System (LMS), enhancing communication among students and between students and instructors. It enables the creation of group discussions for specific courses, facilitates private messaging, and supports the organization of study groups or project teams.

  This component is integral to fostering a collaborative learning environment, allowing for immediate feedback and peer interaction.
  .
- **Responsible Development Team Member**
  Abigail Lindvall

- **Component Diagram**
  A rough higher level diagram of the chat component.

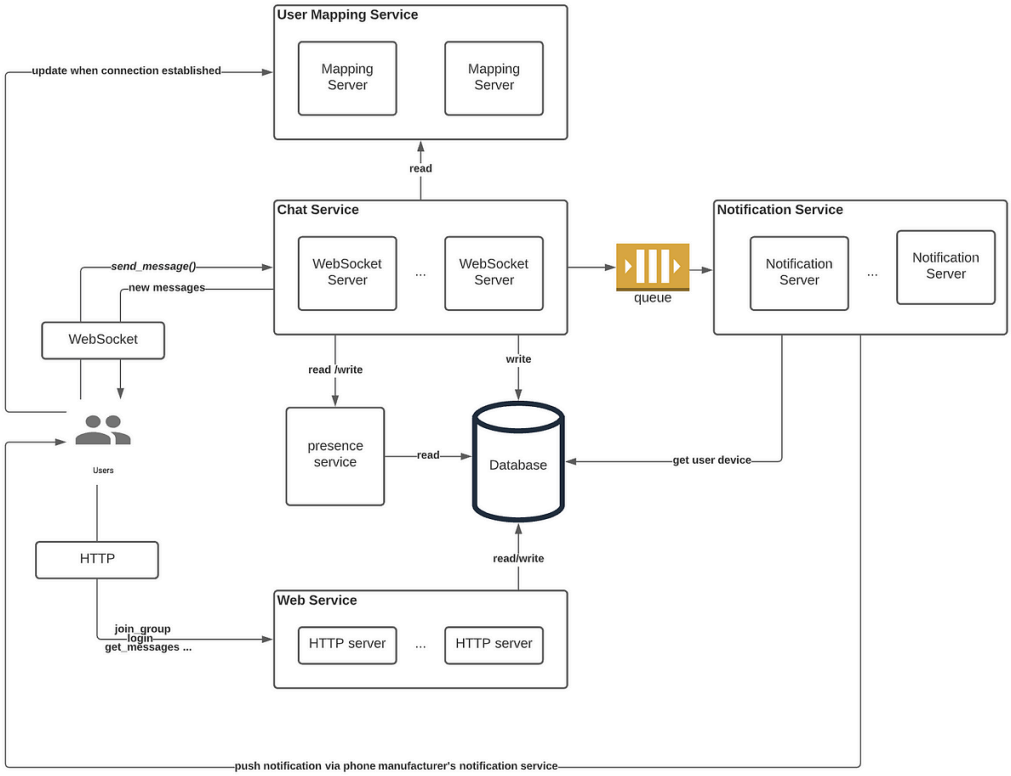

Figure 7

- **Component User Interface**
  Chat Interface: Provides a messaging platform for users, with features for private and group chats, including multimedia support.

  Group Management Dashboard: Enables users to create, join, or manage groups related to their courses, including setting group objectives and sharing resources.

  Notifications Panel: Alerts users to new messages and group activities, ensuring they stay informed and engaged.

- **Component Objects**

  Chat Message: Represents individual messages, with attributes like senderID, receiverID/groupID, messageContent, timestamp, and messageType. Methods include send, edit, and delete.

  Group: Manages details of discussion groups, including groupID, members, courseID, and groupSettings. Methods encompass createGroup, joinGroup, and manageGroup.

  Interactions:

Manages message sending and retrieval through interactions with the Database Layer.
Utilizes the User Management Component to authenticate users and authorize participation in chats and groups.
Engages External Interfaces for dispatching instant notifications to users about chat and group updates.

- **Component Interfaces (internal and external)**
  WebSockets for real-time bidirectional communication between clients and the server.
  Database operations for storing and querying chat messages and group metadata.
  API endpoints for managing group memberships and settings, and for fetching chat history.

- **Component Error Handling**

  Unauthorized Group Access: Ensures that only members of a group or authorized instructors can view or participate in the group's discussions.

  Message Delivery Failure: Implements retry logic for message delivery in case of temporary network issues, and notifies the sender of persistent failures.

  Invalid Message Content: Validates message content for compliance with the platform's guidelines, blocking inappropriate content and alerting the sender.

Revision History

Revision	Date	Change Description
Updates	3/1/24	Included more details for the implementation to reflect the work we have done so far. I have italicized work that is completed to show the progress we have made.
Updates	3/24/24	Included more details for the implementation to reflect the work we have done so far. I have italicized work that is completed to show the progress we have made.

Page Author/Creator: Adeel Bhutta
Last Modified: 8/23/2016