

**Dissertation / Project / Project Work Title:
Project Management System for Academic
Collaboration**

Course No. : S2-24_SEZG628T

Course Title: Dissertation / Project / Project Work

Dissertation / Project /Project Work Done by:

Student Name: Gadila Sai Nikshay Reddy

BITS ID: 2023mt93269

Degree Program: M.Tech. Software Engineering

Research Area: Web Technologies

Dissertation / Project Work carried out at:

Open Text Technologies India Pvt Ltd, Hyderabad, IND



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,
PILANI
VIDYA VIHAR, PILANI, RAJASTHAN - 333031.**

March 2025



ABSTRACT

The Classroom Project Management System is an innovative platform designed to enhance academic collaboration by integrating project management, real-time communication, task allocation, and evaluation mechanisms into a unified system. The platform addresses the limitations of existing tools like Microsoft Teams and traditional LMS platforms by offering structured workflows tailored for educational settings.

Built on a microservices architecture, the system leverages Spring Boot, React, PostgreSQL, Kafka, Eureka, and API Gateway to provide scalability, security, and efficiency. Key features include role-based task allocation, real-time chat and voice communication, progress tracking dashboards, file storage and management, and automated evaluation and feedback mechanisms. The system employs OAuth2/JWT authentication to ensure secure access and Kafka-based event-driven architecture for asynchronous communication.

The platform enables teachers to assign tasks, track progress, and provide feedback, while students can collaborate within structured groups, access project materials, and receive real-time notifications. This ensures seamless academic workflows, improved accountability, and enhanced learning experiences. The system's modularity allows for future scalability, integrating custom analytics dashboards and advanced AI-driven recommendation engines.

By streamlining academic project management and fostering structured collaboration, this system aims to reduce administrative overhead, improve student engagement, and enhance educational transparency in universities and research institutions.

<div>Signed by:  B4156C04D3C440C...</div>	<div>Signed by:  8ADA725202D1429...</div>
Signature of Student	Signature of Supervisor
Name: Gadila Sai Nikshay Reddy	Name: Sai Goutham Reddy Boddireddy
Date: 23-03-2025	Date: 23-03-2025
Place: Hyderabad	Place: Hyderabad

Contents

1.	SYSTEM ARCHITECTURE	4
1.1	ARCHITECTURAL DIAGRAM	5
2.	DATA FLOW DIAGRAM	7
3.	DESIGN CONSIDERATIONS	7
4.	COMPONENT LEVEL DETAILED DESIGN	7
5.	IMPLEMENTATION DETAILS	12
6.	DATABASE SCHEMA	12
7.	TECHNOLOGY STACK	14
8.	Plan of Work	14

1. SYSTEM ARCHITECTURE

The Smart Classroom Project Management System is built on a microservices-based architecture to ensure scalability, modularity, and efficient communication between different services. The system is designed for real-time collaboration through messaging, voice, and video features while maintaining secure and structured project management workflows.

Key Architectural Components:

a. Client Interface:

- Users interact with the system via a React-based Web Application, which communicates with backend microservices through a REST API.
- The API requests are routed through the Gateway Server, ensuring secure and efficient request handling.

b. API Gateway:

- Serves as the centralized entry point for all API requests.
- Manages authentication and request routing to appropriate microservices.

c. Microservices Layer:

Each core functionality is implemented as an independent microservice, ensuring flexibility, modularity, and easy scaling.

- Authentication Service:
 - Manages user registration, login, and JWT-based authentication.
 - Implements Spring Security with role-based access control (RBAC).
- User Management Service:
 - Handles user profiles, roles, and permission levels.
- Class & Group Service:
 - Manages classroom structures, student-teacher allocations, and group formations for collaboration.
- Project & Task Service:
 - Facilitates project creation, task assignments, and progress tracking.
 - Supports sub-projects and task breakdowns for efficient workflow management.
- Chat Service:
 - Supports real-time text-based chat for collaboration.
 - Manages voice channel creation and storage for team discussions.
 - Integrates OpenVidu platform for real-time voice calls.
- File Storage Service:

- Handles document uploads and storage management.
- Supports project-related and chat-related file sharing.
- Notification Service:
 - Sends task updates, system notifications to users.
 - Uses Kafka Message Broker for event-driven asynchronous notification handling.

d. Databases:

- Each microservice has its own PostgreSQL database, ensuring data isolation, integrity, and independent scaling.

e. Infrastructure Services:

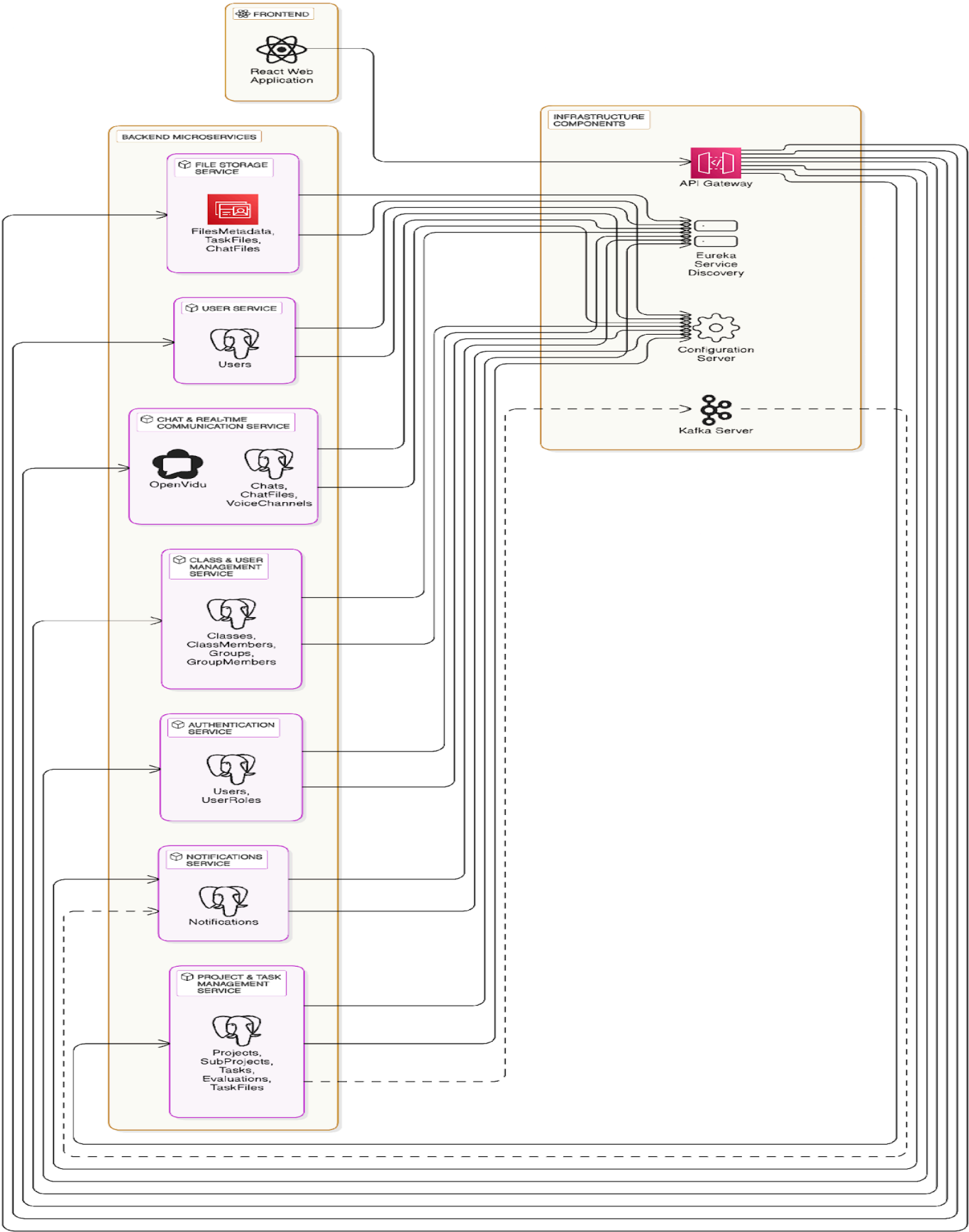
- Config Server:
 - Retrieves configuration properties for microservices from a centralized Git repository.
 - Ensures dynamic updates without manual redeployment.
- Eureka Server (Service Discovery):
 - Enables automatic registration and discovery of microservices.
 - Allows services to dynamically locate and communicate with each other.
- Kafka Message Broker (Asynchronous Communication):
 - Facilitates event-driven interactions between microservices.
 - Used primarily for notifications and project updates.
- OpenVidu platform service (Real-Time Voice Communication):
 - Provides voice chat capabilities.
 - Integrated with Chat Service for seamless user collaboration.

Key Benefits of this Architecture:

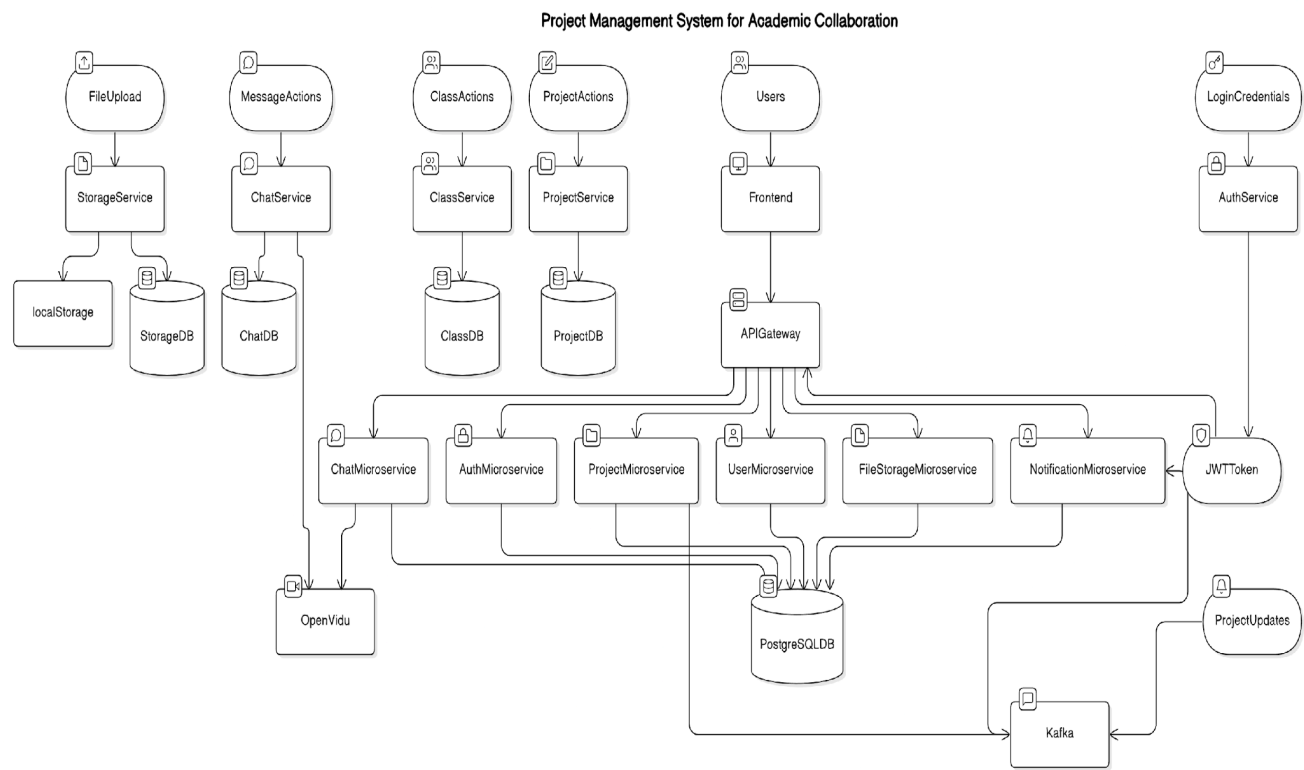
- High Availability: Services remain functional even if individual components fail.
- Scalability: Each microservice can be scaled independently based on workload.
- Security & Role Management: JWT-based authentication and role-based access control ensure secure access.
- Real-Time Communication: OpenVidu and Kafka ensure smooth live interactions.
- Cloud-Ready Deployment: Designed for containerization and deployment in cloud environments.

This architecture ensures a highly efficient, modular, and scalable solution for academic collaboration, enhancing project-based learning in a classroom environment.

1.1 ARCHITECTURAL DIAGRAM



2. DATA FLOW DIAGRAM



3. DESIGN CONSIDERATIONS

Factor	Considerations
Scalability	The system follows a microservices architecture to allow independent scaling of services.
Security	JWT authentication ensures secure access to different modules.
Fault Tolerance	Kafka-based messaging helps in asynchronous communication and event-driven workflows.
Performance Optimization	Efficient indexing in PostgreSQL
User Experience	The UI is designed for clarity, with intuitive navigation and role-based access.

4. COMPONENT LEVEL DETAILED DESIGN

1. Authentication & Authorization Service

- Technology Stack: Spring Boot, Spring Security, JWT, PostgreSQL
- Responsibilities:
 - User registration and login
 - JWT-based authentication token generation.
 - Role-based access control (RBAC) with Spring Security
- Endpoints:

Method	Endpoint	Description
POST	/auth/register	Registers a new user
POST	/auth/login	Authenticates user and generates JWT token

2. User Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Manages user profile information
 - Stores user roles and permissions
- Endpoints:

Method	Endpoint	Description
POST	/users	Creates a new user
GET	/users/{id}	Fetches user details
PUT	/users/{id}	Updates user details

3. Class & Group Management Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Handles class and group creation
 - Manages class and group members
- Endpoints:

Method	Endpoint	Description
POST	/classes	Creates a class
GET	/classes	Fetches all classes
GET	/classes/{id}	Fetches class by ID
PUT	/classes/{id}	Updates a class
DELETE	/classes/{id}	Deletes a class
POST	/groups	Creates a new group

GET	/groups	Fetches all groups
GET	/groups/{id}	Fetches group details by ID
PUT	/groups/{id}	Updates group details by ID
DELETE	/groups/{id}	Deletes a group by ID
POST	/classes/{id}/members	Creates class members
GET	/classes/{id} /members	Fetches all class members
DELETE	/classes/members/{id}	Deletes a class member
GET	/groups/{id} /members	Creates group members
PUT	/groups/{id} /members	Fetches all group members
DELETE	/groups/members /{id}	Deletes a group member

4. Project Management Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Manages projects and sub-projects
 - Assigns projects to classes or groups
- Endpoints:

Method	Endpoint	Description
POST	/projects	Creates a new project.
GET	/projects	Retrieves all projects.
GET	/projects/{id}	Retrieves project details by ID.
PUT	/projects	Updates project details.
DELETE	/projects/{id}	Deletes a project by ID.
POST	/projects/{id}/subprojects	Creates a sub-project under a project.

5. Sub-Project Management Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Manages sub-projects under a parent project
 - Assigns tasks to sub-projects
- Endpoints:

Method	Endpoint	Description
GET	/subprojects	Retrieves all sub-projects.
GET	/subprojects/{id}	Retrieves sub-project details by ID.
PUT	/subprojects	Updates sub-project details.
DELETE	/subprojects/{id}	Deletes a sub-project by ID.
POST	/subprojects/{id}/tasks	Creates a task under a sub-project.

6. Task Management Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Manages tasks within a sub-project
 - Allows file attachments and evaluations
- Endpoints:

Method	Endpoint	Description
GET	/tasks	Retrieves all tasks.
GET	/tasks/{id}	Retrieves task details by ID.
PUT	/tasks	Updates task details.
DELETE	/tasks/{id}	Deletes a task by ID.
POST	/tasks/{id}/evaluations	Creates an evaluation for a task.
GET	/tasks/{id}/evaluations	Retrieves all evaluations for a task.

7. Chat & Voice Channel Service

- Technology Stack: Spring Boot, WebSockets, OpenVidu (for voice calls)
- Responsibilities:
 - Real-time messaging via WebSockets
 - Voice communication using OpenVidu
 - Manages voice channel and messages details for group discussions
- Endpoints:

Method	Endpoint	Description
POST	/chat/sendMessage	Sends a message in a chat.
GET	/chat/getMessages/{chatId}	Retrieves messages for a chat.
POST	/chat/createVoiceChannel	Creates a voice channel.
GET	/chat/getVoiceChannels	Retrieves all voice channels.
PUT	/chat/getVoiceChannels/{id}	Updates Voice channel by ID

8. File Storage Service

- Technology Stack: Spring Boot, PostgreSQL
- Responsibilities:
 - Stores user-uploaded files (e.g., chat files, task attachments)
 - Supports file upload/download
- Endpoints:

Method	Endpoint	Description
POST	/tasks/{id}/files	Uploads a file for a task.
GET	/tasks/{id}/files	Retrieves all files related to a task.
DELETE	/files/{id}	Deletes a file by ID.

9. Notification Service

- Technology Stack: Spring Boot, Apache Kafka (for async messaging)
- Responsibilities:
 - Sends real-time notifications to users
 - Listens to project/task events using Kafka
- Endpoints:

Method	Endpoint	Description
POST	/notifications	Sends a notification.
GET	/notifications	Retrieves all notifications.

10. Gateway Server (API Gateway)

- Technology Stack: Spring Cloud Gateway
- Responsibilities:
 - Central entry point for API requests
 - Routes requests to the respective microservices

11. Eureka Server (Service Discovery)

- Technology Stack: Spring Cloud Netflix Eureka
- Responsibilities:
 - Registers and discovers microservices

12. Configuration Server

- Technology Stack: Spring Cloud Config
- Responsibilities:
 - Manages configurations centrally from a Git repository

5. IMPLEMENTATION DETAILS

1. Backend: Spring Boot microservices for authentication, project management, chat, and notifications.
2. Frontend: React.js for dynamic and responsive UI.
3. Communication: Kafka facilitates asynchronous messaging between all microservices and notifications service, REST APIs are used for communication between all micro services.
4. Configuration Management: Config Server fetches service configurations from a central GIT repository.

5. Service Discovery: Eureka ensures that all microservices can locate and communicate with each other dynamically.
6. Gateway Handling: API Gateway secures and directs traffic to respective microservices.

6. DATABASE SCHEMA

Each microservice in the system operates with its own PostgreSQL database, ensuring modularity and scalability. Relationships are managed using UUIDs, APIs, and event-driven communication via Kafka. Below is an overview of the primary database schema, highlighting the key tables for each microservice:

1. Authentication Service:

- Users (id UUID PK, name, email, password_hash, role ENUM, created_at TIMESTAMP)

Relationships:

- The Users table links to all services, including Class Management, Project Management, Chat, and Evaluation.

2. Class & User Management Service:

- Classes (id UUID PK, name, teacher_id UUID FK -> Users.id, created_at TIMESTAMP)
- ClassMembers (id UUID PK, class_id UUID FK -> Classes.id, student_id UUID FK -> Users.id)
- Groups (id UUID PK, class_id UUID FK -> Classes.id, name, created_at TIMESTAMP)

Relationships:

- Classes link to Projects & Chat Services.
- Groups link to SubProjects within the Project Management Service.

3. Project & Task Management Service:

- Projects (id UUID PK, class_id UUID FK -> Classes.id, title, due_date

TIMESTAMP)

- Tasks (id UUID PK, project_id UUID FK -> Projects.id, assigned_to UUID FK -> Users.id, status ENUM, due_date TIMESTAMP)
- Evaluations (id UUID PK, task_id UUID FK -> Tasks.id, evaluator_id UUID FK -> Users.id, score INT, comments TEXT)

Relationships:

- Projects are linked to academic Classes.
- Tasks are assigned to Users, ensuring structured project tracking.
- Evaluations reference Tasks, allowing teachers to assess student work.

4. Chat Service:

- Chats (id UUID PK, group_id UUID FK -> Groups.id, user_id UUID FK -> Users.id, message TEXT, timestamp TIMESTAMP)

Relationships:

- Messages are linked to Groups or Users.

5. File Storage Service

- FilesMetadata (id UUID PK, owner_id UUID FK -> Users.id, file_url TEXT, file_type ENUM)

Relationships:

- FilesMetadata is referenced in Tasks and Chat Messages, ensuring controlled access to project resources.

6. Notification Service:

- Notifications (id UUID PK, user_id UUID FK → Users.id, message TEXT, read_status BOOLEAN, created_at TIMESTAMP)

Relationships:

- Listens for task updates via Kafka and sends real-time push notifications using Firebase Cloud Messaging.

7. TECHNOLOGY STACK

Component	Technology
Backend Framework	Spring Boot (Java)
Frontend Framework	React.js

Database	PostgreSQL (Relational)
API Gateway	Spring Cloud Gateway
Service Discovery	Eureka Server
Messaging	Apache Kafka
Authentication & Authorization	JWT + Spring Security
File Storage	Local File System
Real-Time Communication	WebSocket, WebRTC (for voice channels)
Configuration Management	Spring Cloud Config Server

8. Plan of Work

Phases	Start Date-End Date	Work to be done	Status
Dissertation Outline	10 Jan 2025 – 18 Jan 2025	Literature Review and prepare Dissertation Outline	Completed
Design & Development	25 Jan 2025 – 20 March 2025	Design & Development Activity	In-Progress
Mid-Semester Progress Report Submission	20 March 2025 – 27 March 2025	Submit progress report with updates on completed features.	Pending
Pending Development and Testing	20 March 2025 – 06 Apr 2025	Pending Development: Voice channels, Notifications and analytics for teacher role. Testing: Software Testing, User Evaluation & Conclusion	In-Progress
Dissertation Review	06 Apr 2025 - 16 Apr 2025	Submit Dissertation to Supervisor & Additional Examiner for review and feedback	Pending
Submission	17 Apr 2025 - 24 Apr 2025	Final Review and submission of Dissertation	Pending