CHAPTER 3

METHODOLOGY

**3.0 Overview**

This project is intended to use the Agile Software Development Methdology with Scrum as the chosen framework. Agile emphasizes flexibility, iterative development, as well as colaboration, making it ideal for projects that need to adapt to changing requirements and foster continuous improvement. The Scrum framework, a subset of Agile, is employed to structure the project’s development process into a series of iterative cycles called Sprints. Each Sprint includes stages of planning, development, testing and review. This iterative process allows the team to deliver increments of the product frequently, gather feednack, and adjust the course of development as needed. The Agile approach with Scrum is particularly beneficial for projects where requirements may evolve over time or where iterative feedback is crucial. Agile methodologies like scrum are recommended for projects that require flexibility, adaptability and continuous deleivery of functional increments. [SchwaberSutherland2020],

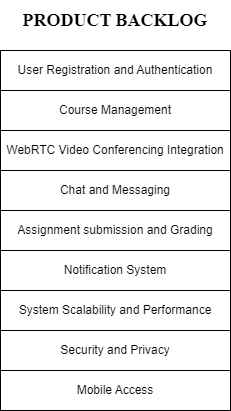
**3.1 Product Backlog Creation**

The Product Backlog is a crucial artifact in Scrum, representing a dynamic list of work that needs to be done on the project. It includes everything from new features and enhancements to bug fixes and technical debt. The management of the backlog, involves prioritizing items based on their value to the project and their alignment with the overall project goals. This prioritization is essential for ensuring that the team is always working on the most important tasks first. As the project evolves, so does the backlog, with items being added, removed, or re-prioritized based on feedback and changing requirements. (Schwaber & Sutherland, 2020).

The product backlog for the e-learning platform project is structured around a set of prioritized features and functionalities designed to advance the research objectives of enhancing user engagement and improving learning outcomes. Central to this backlog is the development of a responsive user interface capable of adapting to a wide range of devices, thereby ensuring accessibility and usability across different platforms. The project also emphasizes the implementation of a content management system, which is crucial for facilitating the seamless creation and management of courses. Additionally, the integration of interactive components, such as a video conferencing feature, is prioritized to actively engage students in the learning process.

Beyond these core functionalities, the backlog includes the development of a secure user authentication system, which is essential for protecting user data and maintaining the integrity of the platform. The incorporation of real-time analytics is another key feature, enabling the tracking of student progress and providing valuable insights that can be used to tailor educational experiences. Furthermore, the support for multimedia content is recognized as a critical element in enhancing the overall learning experience, allowing for a richer and more varied educational environment.

In the figure below, we see how our product backlog is structured. This initial list of items are at the “Epic Level” which means they are too big and vague for us to be able to structure them based on technical infrastructure and other dependencies.

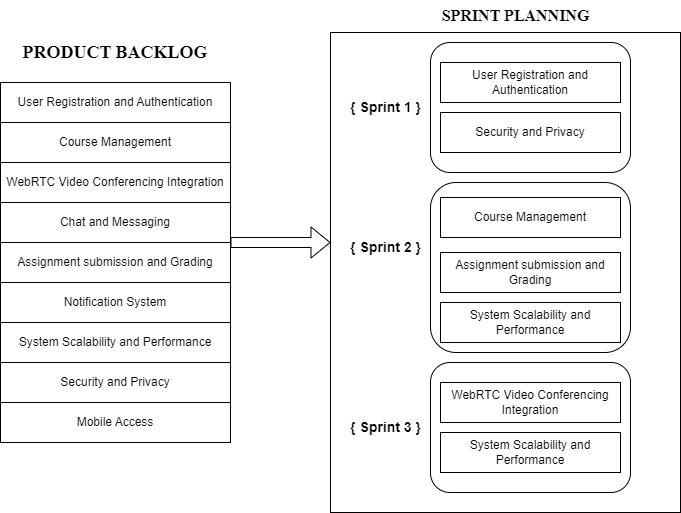


In the development of our Learning Management System (LMS), several key features and functionalities are prioritized to enhance user experience and system efficiency, which have made it into the product backlog. For User Registration and Authentication, users need the capability to register for accounts and log in securely to access their dashboards, while administrators require tools to manage user roles and permissions to control access across the LMS. In Course Management, administrators must be able to create, edit, and delete courses, assign lecturers, and manage course schedules, while instructors need to manage their assigned courses and students should be able to register and enroll in courses to engage in the learning process. The integration of WebRTC Video Conferencing is crucial, enabling instructors to schedule and conduct live sessions, students to participate in these sessions, and all users to share screens and record sessions for later access. The Chat and Messaging system facilitates real-time communication, allowing students to collaborate and instructors to send announcements. For Assignment Submission and Grading, students need to submit their work online and instructors must review and grade these submissions. The Notification System ensures users stay informed about video sessions, deadlines, and announcements, with administrators configuring settings to manage notification overload. System Scalability and Performance are vital, requiring the ability to handle high volumes of concurrent video sessions and optimize streaming quality based on network conditions. Security and Privacy measures include implementing encryption for video streams and enforcing data privacy policies. Lastly, Mobile Access must be optimized to ensure that users can participate in courses and video conferencing from various devices, with developers ensuring the LMS is responsive and consistent across different screen sizes.

**3.2 Sprint Planning**

Sprint Planning is the event that kicks off each Sprint, where we selects items from the top of the Product Backlog to work on during the upcoming Sprint. The goal is to determine what can be delivered in the Sprint and how we will accomplish that work. The outcome of Sprint Planning is a Sprint Goal, a clear objective that the team commits to achieving, and a Sprint Backlog, which includes the tasks required to meet that goal. This structured planning process helps to ensure that the team is focused and aligned on what needs to be accomplished within the Sprint timeframe (Schwaber & Sutherland, 2020).

In our upcoming sprint, we will focus on several crucial items from the product backlog to drive our LMS project's progress. Our primary objective is to enhance user registration and authentication processes, ensuring secure login and comprehensive management of user roles and permissions by administrators. We will also prioritize the development of robust course management features, enabling administrators to create and manage courses, assign lecturers, and establish schedules, while facilitating student registration and enrollment. Integrating WebRTC video conferencing is essential, allowing instructors to schedule and conduct live sessions, with features for screen sharing and session recording. Additionally, we will implement chat and messaging capabilities to support real-time communication and announcements. The sprint will address assignment submission and grading workflows, streamline notifications for users, and configure settings to prevent notification overload. We will also focus on ensuring system scalability and optimizing performance to handle concurrent video sessions and varying network conditions. Security measures will include encryption for video streams and adherence to data privacy policies. Finally, we will enhance mobile access to ensure a consistent and responsive user experience across devices.



3.3 Sprint Execution

During Sprint Execution, the development team works on the tasks identified in the Sprint Backlog to achieve the Sprint Goal. This phase is characterized by intense collaboration among team members, who frequently communicate to ensure progress is on track and to quickly address any issues that arise. (Beck et al., 2001).

In this sprint execution phase, our focus will be on systematically addressing the prioritized tasks from the product backlog to ensure efficient delivery of key features. We will start by implementing and testing enhancements to the user registration and authentication processes, focusing on secure login and comprehensive management of user roles by administrators. Next, we will develop and refine course management functionalities, including the creation, editing, and scheduling of courses, and enabling student enrollment. Concurrently, we will integrate WebRTC video conferencing, enabling instructors to schedule live sessions, share screens, and record sessions, while ensuring the feature supports smooth real-time interaction for students. The chat and messaging system will be developed to facilitate effective communication and announcements. We will also work on the assignment submission and grading features to streamline the process for both students and instructors. The notification system will be configured to provide timely updates while avoiding overload, and performance optimizations will be implemented to handle high volumes of concurrent sessions and varying network conditions. Security measures will be integrated, focusing on encryption and data privacy compliance. Finally, we will ensure the LMS is responsive and functional on mobile devices, providing a consistent user experience. Regular stand-ups and reviews will ensure progress tracking and address any issues promptly, aligning with our sprint goals and timelines.

3.3.1 Requirements

The requirements of the project has been put into two parts. We have the hardware requirements anad Software requirements. Hardware requirements refer to the specific physical components and specifications needed for a computer system or device to run a particular software application, perform a specific function, or meet a certain level of performance. These requirements ensure that the hardware is capable of supporting the software and tasks intended to be executed on it. However it is advisable to have hardware that exceeds these requirements for the optimal performance of services and applications. [hardware req] On the other hand, the software requirements refer to the technical specifications that explain the circumstances and capabilities that a software system must have in order to perform properly. [software req].

**Hardware requirements**

1. Processor: 13th Gen Intel Core i7 13620 H
2. RAM: 16GB DDR4 Random Access Memory (RAM)
3. Storage: 256GB Solid State Drive (SSD)
4. WebCam
5. Microphone
6. Router

**Software Requirements**

The various software applications used in the design and implementation of the project includes

1. **Operating System**

The operating system provides the needed environment for software development and execution. For our project, we used Windows 11 Home. This is because it is compatible with a wide selection of development tools and applications, it offers enhanced performance and stability, the UI provides an intuitive and efficient workspace, which can improve productivity during development and testing phases. And because the scope of our project does not require enterprise-level features, Windows 11 Home offers a cost-effective solution without compromising on the essential functionalities needed for development.

1. **Web Browser**

A web browser is a software application used to access information on the World Wide Web [web browser def]. For our project, Google Chrome was selected for its performance, developer tools, compatibility, and strong security features, all of which contribute to a more efficient and effective development process.

1. **Text Editor**

The text editor of choice is the Microsoft Visual Studio Code (VS Code). VS Code supports a wide range of programming languages and frameworks, making it suitable for diverse development tasks. It also has a rich ecosystem of extensions and plugins that enhance functionality, including support for debugging, code linting, version control, among others. It has an integrated terminal, it is cross-platform and has a user-friendly interface.

1. **MongoDB**

MongoDB was chosen as our Database Management System (DBMS) for the project because of its flexibility and scalability. It is a NoSQL database that stores data in JSON-like format, allowing for dynamic schema design as well as the handling of unstructured or semi-structured data. It is also well integrated in our programming language of choice, JavaScript.

1. **MongoDB Compass**

MongoDB compass is a powerful Graphical user interfae ( GUI ) for querying, aggregating and analysing your mongodb data in a visual environment. It allows users to connect to MongoDB deployment hosted on a remote server or on the users’ local machine. MongoDB compass provides real-time data and performance statistic visualisations, which allows us to better in understanding database behaviour and how to ways for query optimisation.

1. **JavaScript & NodeJs**

Node.js is an important component of this project's technology stack, serving as a runtime environment for executing JavaScript code on the server. Node.js is based on Chrome's V8 JavaScript engine, which provides a non-blocking, event-driven architecture that is ideal for asynchronous processes and scalable network applications. It also provides a solid foundation for developing server-side apps and APIs, by leveraging the npm ecosystem to integrate a diverse set of libraries and modules that simplify development processes.

1. **PeerJS & WebRTC**

PeerJS is used to simplify the project's implementation of peer-to-peer communication. PeerJS provides an easy-to-use API that encapsulates the difficulties of setting up WebRTC peer connections, allowing clients to exchange real-time audio, video, and data. It manages the signalling process, peer finding, and connection management, allowing developers to concentrate on application logic rather than underlying infrastructure. PeerJS's capabilities are critical for developing real-time communication features like video chat and file sharing without having to deal with the intricacies of WebRTC protocols.

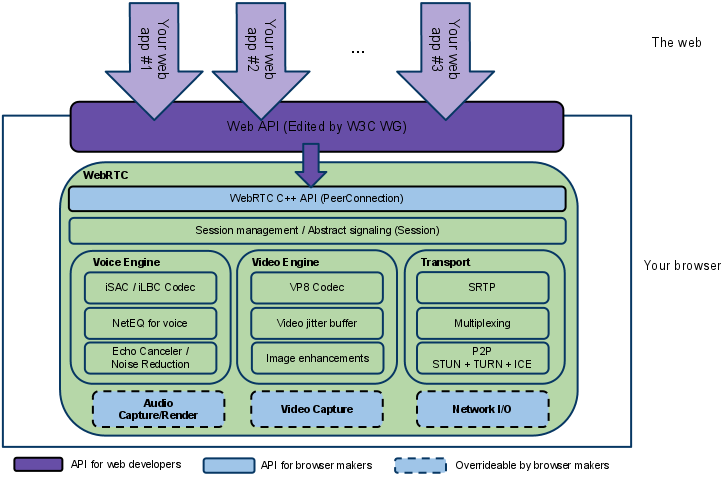


Figure is from (W3C WG, 2022)

1. **EJS (Embedded JavaScript)**

EJS (Embedded JavaScript) is used to render HTML templates on the server side of and we will use this as our templating engine for rendering dynamic pages. EJS allows us to embed JavaScript code directly into HTML pages, making it easier to generate dynamic content based on server-side data. This templating engine supports template inheritance and partials, which make it easier to create reusable and maintainable HTML components. EJS works smoothly with Node.js applications, giving us an efficient way to render views and manage dynamic content.

1. **Cascading Style Sheets ( CSS )**

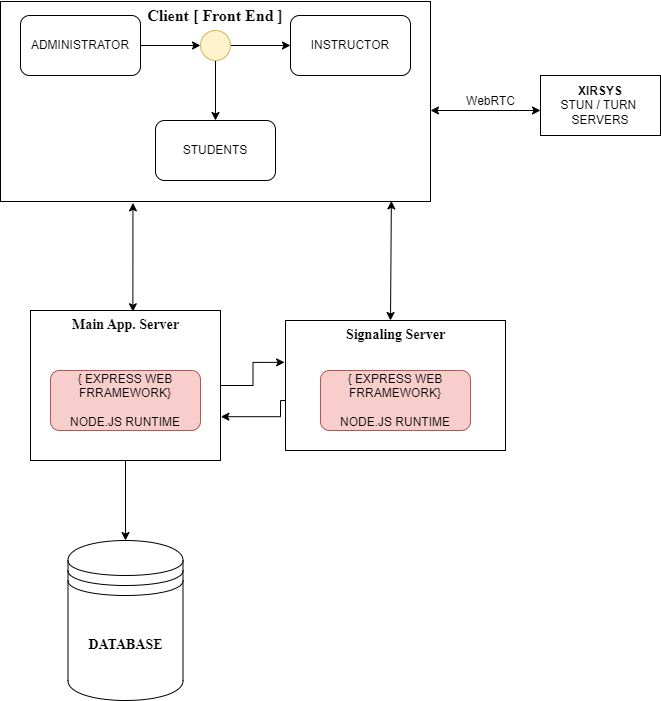
CSS (Cascading Style Sheets) is used to specify the visual appearance and layout of the project's web pages. CSS separates content from design, allowing developers to apply consistent styling across various pages and components. CSS allows developers to customise the look and feel of their applications, including colour schemes, typography, spacing, and responsive design.

**3.3.2 Design and Implementation**

In this section we will illustrate using diagrams, the architecture of our system, the flow charts for various features of the application such as user account registration and authentication as well as how a video call is setup. We also illustrate various sequence diagrams for the system, data flow diagrams as well as entity relationship diagrams.

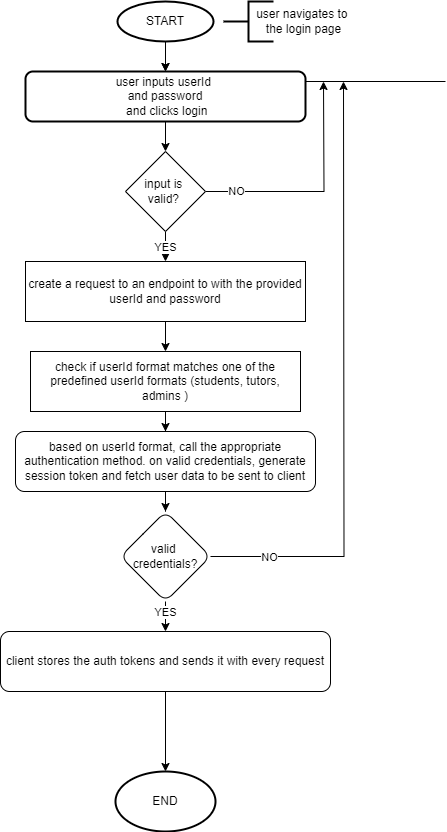
**3.3.3 System Architecture**

The system architecture diagram provides a comprehensive overview of the LMS's core components and their interactions, serving as a high-level abstraction to facilitate understanding of the system's overall structure. At its core, the diagram illustrates the main elements of the system: the user interface, application server, WebRTC server, and database, detailing how these components are interconnected and communicate with each other. The user interface represents the front-end layer, where users interact with the system. It includes all the elements that users see and interact with, such as dashboards, course management tools, and video conferencing interfaces. This layer ensures a seamless user experience and interfaces with the application server to handle user requests and present data. The application server functions as the central processing unit of the system. It manages business logic, handles user requests, and facilitates communication between the user interface and the database. It processes authentication, authorization, course management, assignment submissions, and other critical operations, ensuring that the system functions smoothly and efficiently. The WebRTC server is integral for real-time communication, enabling live video sessions, screen sharing, and recording functionalities. It supports the interactive aspects of the LMS, allowing instructors and students to engage in real-time video conferencing and collaborate effectively. The database component stores all persistent data, including user information, course details, assignment submissions, and chat messages. It interacts with the application server to retrieve and store data as needed, ensuring data integrity and accessibility.



**3.3.4 User Registration and Login**

The process begins when a user accesses the registration or login page of the platform. The login process is straightforward. Users must input their user id (student id for students, admin id for administrators and lecturer id for lecturers ) and password. The system then checks these credentials against the database to authenticate the user. If the details match, the user is redirected to their dashboard, where they can access various features of the platform. In the case of an authentication failure, an error message is displayed, prompting the user to retry or recover their password.

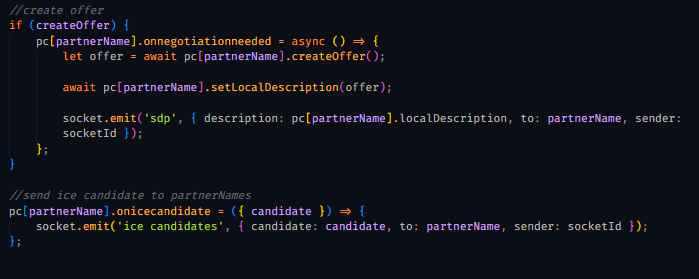


**3.3.5 Video Conferencing Flow Chart**

Upon the initiation of a session, a participant’s identification and course details are validated. This is facilitated through a structured URL pattern that determines the user type (either lecturer or student) and redirects the user to the appropriate meeting room.

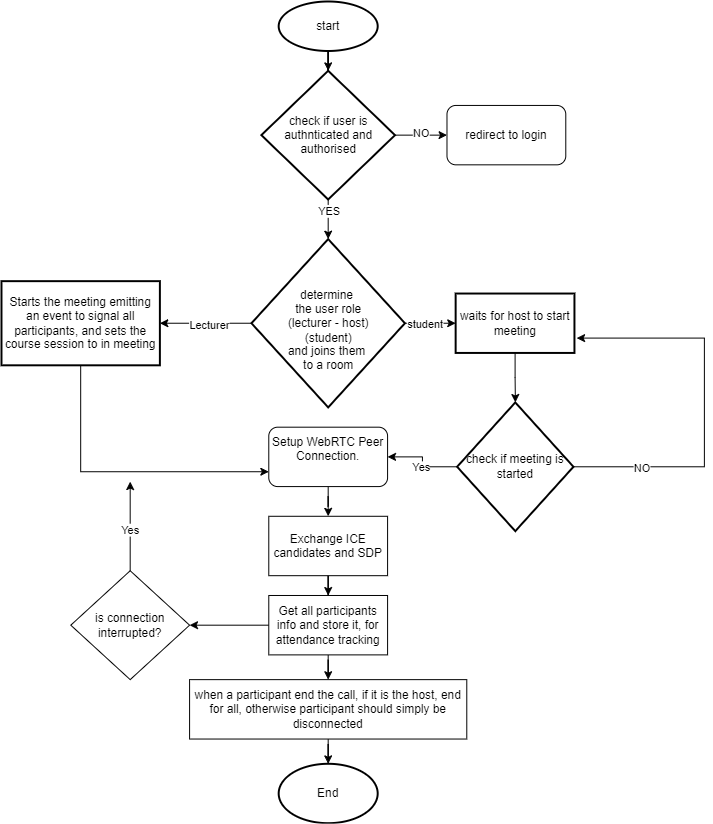
The participant's credentials and course information are stored temporarily in the session storage to ensure continuity across the session.Once the user accesses the meeting room, the system retrieves the course and user information to ensure proper authentication and authorization. Participants' data is then used to set up the communication environment, where the information about the participants, such as user IDs and their roles (e.g., lecturer or student), is crucial for managing permissions and interactions during the session.

The system establishes peer-to-peer connections between participants using the WebRTC framework. This involves setting up peer connections, exchanging session descriptions (SDP), and handling ICE (Interactive Connectivity Establishment) candidates, which are essential for establishing the media streams over varying network conditions. The negotiation process, initiated either by the host or other participants, involves creating and setting session descriptions, which detail the media formats and parameters supported by each peer.



This ensures compatibility and seamless media exchange. Participants' media streams, whether video or audio, are then captured and transmitted across these peer connections. The system ensures that each participant's media is rendered on the interfaces of other participants, allowing for real-time interaction. The handling of media streams is dynamic, enabling participants to toggle between different media sources, such as sharing their screens or switching between audio and video streams. This is particularly useful in a classroom setting, where a lecturer might need to present material via screen sharing.

Throughout the session, the system monitors the connection state and signaling state of each peer connection, ensuring that any disruptions, such as disconnections or failures, are managed promptly. The interface dynamically adjusts to reflect the current state of the connections, ensuring that participants are kept informed of any issues, such as a peer leaving the session. Additionally, the platform provides functionality for managing the session lifecycle, including the ability for the host (typically the lecturer) to terminate the session for all participants. This action is accompanied by an update of the course meeting information, which ensures that the session details are accurately recorded and that participants are appropriately redirected once the session ends.



**3.3.6 Level 0 Data Flow Diagram (DFD) for Video Conferencing Feature**

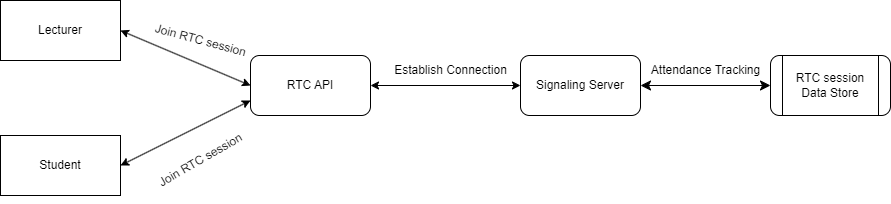
The Level 0 Data Flow Diagram (DFD) provides an overview of the video conferencing feature within the e-learning platform, illustrating the main processes, data stores, and external entities involved. It serves as a high-level representation that abstracts detailed internal processes, emphasizing primary data flows and interactions.

* Components:

1. External Entities:
   1. Lecturer: Initiates and participates in the RTC session.
   2. Student: Joins and participates in the RTC session.
2. Processes:
   1. RTC API: Serves as the central process, coordinating the initiation and participation in RTC sessions.
   2. User Authentication: Ensures that the user (lecturer or student) is authenticated before accessing the RTC API.
   3. Signaling Server: Establishes and manages the WebRTC connection between participants.
3. Data Stores:
   1. Data: Repository for storing session data, such as attendance tracking and session metadata.

* Data Flows:

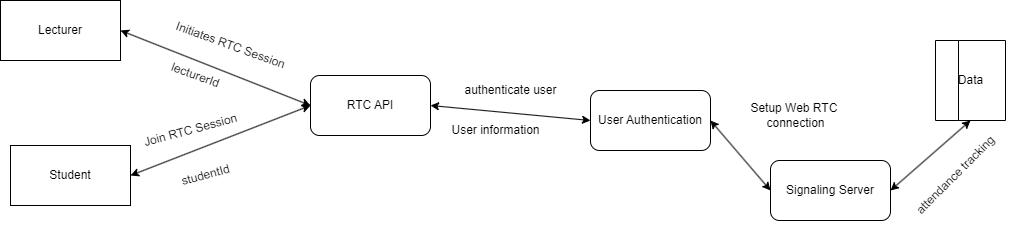
1. Initiates RTC Session: The lecturer initiates a Real-Time Communication (RTC) session by sending a request to the RTC API along with their lecturerId.
2. Join RTC Session: Students join the RTC session by sending a request to the RTC API along with their studentId.
3. User Information: The RTC API forwards user information to the User Authentication process to authenticate the user.
4. Authenticate User: User Authentication validates the user credentials and sends an authentication status back to the RTC API.
5. Setup WebRTC Connection: The RTC API interacts with the Signaling Server to establish the WebRTC connection.
6. Attendance Tracking: The Signaling Server sends session data, including attendance tracking, to the Data store.



**3.3.7 Level 1 Data Flow Diagram (DFD) for Video Conferencing Feature**

The Level 1 Data Flow Diagram (DFD) delves deeper into the internal processes of the video conferencing feature, detailing the specific functions within the RTC API and Signaling Server processes. It provides a more granular view of data interactions and flow within the system.

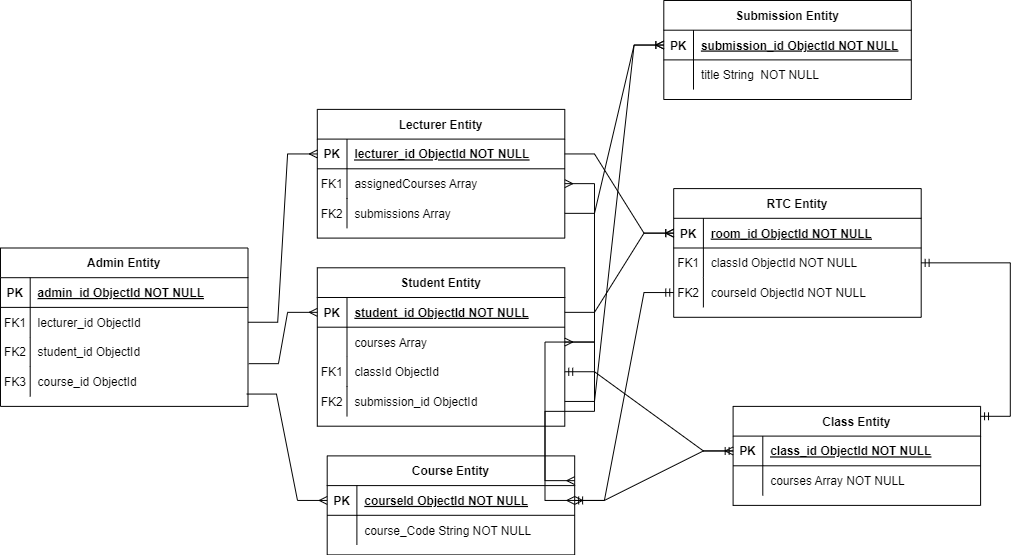
1. Components:
2. External Entities:
   1. Lecturer: Continues to initiate and participate in RTC sessions.
   2. Student: Continues to join and participate in RTC sessions.
3. Processes:
   1. RTC API: Further decomposed into functions handling session initiation, management, and participation.
   2. Signaling Server: Manages the signaling and connection establishment for WebRTC, along with attendance tracking.
4. Data Stores:
   1. RTC Session Data Store: Stores detailed session data, including connection logs, attendance records, and session metadata.
5. Data Flows:
6. Initiates RTC Session: The lecturer sends a request to the RTC API to initiate an RTC session, containing the lecturerId.
7. Join RTC Session: Students send requests to the RTC API to join the RTC session, containing their studentId.
8. Establish Connection: The RTC API processes the session initiation and participation requests, then interacts with the Signaling Server to establish the WebRTC connection.
9. Attendance Tracking: The Signaling Server tracks attendance and sends detailed session data to the RTC Session Data Store for storage and later retrieval.
10. Detailed Process Descriptions:
11. RTC API:
    1. Session Management: Handles the initiation and management of RTC sessions, ensuring that requests from lecturers and students are processed appropriately.
    2. Connection Coordination: Coordinates with the Signaling Server to establish and manage WebRTC connections.
12. Signaling Server:
    1. Connection Establishment: Manages the signaling process required to establish WebRTC connections between participants.
    2. Attendance Tracking: Monitors and records attendance, sending this data to the RTC Session Data Store for persistent storage.



The Level 0 and Level 1 Data Flow Diagrams for the video conferencing feature in the e-learning platform provide a structured and detailed view of the system's data interactions. These diagrams help in understanding the primary processes involved, the flow of data between them, and the interactions with external entities. The level of detail provided in Level 1 further clarifies the internal workings of the RTC API and Signaling Server, ensuring a comprehensive understanding of the video conferencing feature's operation.

**3.3.8 Entity Relationship Overview for the Video Conferencing feature**

The Entity-Relationship Diagram (ERD) illustrates the data structure of the video conferencing feature within the e-learning platform. It showcases the various entities involved, their attributes, and the relationships between them. This ERD is pivotal for understanding how the system manages courses, submissions, and real-time communication (RTC) sessions.



The Admin Entity is the administrative core of the system, uniquely identified by the admin\_id attribute. It maintains references to lecturers, students, and courses through foreign keys, ensuring streamlined management of these entities. The Lecturer Entity, identified by lecturer\_id, includes arrays for assigned Courses and submissions, reflecting the courses they manage and the submissions they oversee. Similarly, the Student Entity, identified by student\_id, contains arrays for courses and a foreign key for classId, linking students to their classes and courses. The Course Entity, with courseId as its primary key, holds vital information about each course, including a unique course code and a reference to the associated class. The Class Entity aggregates courses through an array, highlighting the collective structure of classes within the platform. The RTC Entity, identified by room\_id, is crucial for real-time communication, linking each RTC session to specific classes and courses, thereby facilitating seamless virtual interactions. The Course Entity, with courseId as its primary key, holds vital information about each course, including a unique course code and a reference to the associated class. The Class Entity aggregates courses through an array, highlighting the collective structure of classes within the platform. The RTC Entity, identified by room\_id, is crucial for real-time communication, linking each RTC session to specific classes and courses, thereby facilitating seamless virtual interactions.

**3.3.9 Course Management and Submission Management**

Upon successful login, students and lecturers access the course management area, which provides a tailored list of courses based on the user’s level, program, and faculty. For students, the registration process involves using system-generated registration codes to enroll in courses. Once registered, the courses are added to the student's courses , effectively integrating the student into the course structure. Lecturers, on the other hand, are tasked with managing their assigned courses. This involves adding lessons and materials to course chapters and initiating live class sessions, thus centralizing course material management.

The submission management system is meticulously designed to streamline the creation and handling of assignments. Lecturers initiate this process by entering essential details such as the course code, title, instructions, and timelines. The system organizes these details into structured objects and links them to the appropriate course using the course code. If no existing submission record is found, a new entry is created in the Submissions collection. For existing records, the system employs the $addToSet operator to append new submissions, ensuring no duplication. Students interact with this system to access and submit assignments. Upon submission, the system associates the uploaded files with the student’s record in the student-submissions field of the Submissions collection. This organized storage mechanism ensures easy retrieval and review. Post-submission, lecturers can review and grade the work, with the system updating each entry's status to indicate whether it has been graded. Additional functionalities, such as bulk download of submissions and robust error handling, enhance the platform’s utility, ensuring smooth interactions and efficient data management.

The ERD and the detailed workflows of course management and submission management provide a comprehensive blueprint of the e-learning platform’s data interactions. This structured approach ensures efficient management of academic activities, fostering seamless communication and collaboration between lecturers and students. The robust design facilitates effective course management, organized submission handling, and real-time communication, thereby enhancing the overall educational experience.

**3.4 Sprint Review and Retrospective**

The Sprint Review is held at the end of each Sprint to inspect the work done and gather feedback from stakeholders. The team demonstrates the completed work, and discussions focus on what was accomplished versus what was planned. This meeting is not just about showcasing the work but also about fostering a collaborative environment where stakeholders can provide input, which may lead to adjustments in the Product Backlog. Following the Sprint Review, the team conducts a Sprint Retrospective, a meeting dedicated to reflecting on the Sprint. The Retrospective is an opportunity for the team to discuss what went well, what didn't, and what could be improved in the next Sprint. These meetings are essential for continuous improvement, allowing the team to identify and implement changes that enhance their effectiveness in future Sprints (Schwaber & Sutherland, 2020; VersionOne, 2021).