SE491
Capstone Project
(Young Minds Learners Hub (YML Hub)

Prepared By:

Hussain Timah	**4010419**
Khalid Namat Allah	**4110766**
Abdullah Hasan Prwm	**4120243**
Khaled Salem	**4120194**

Supervision By:

Dr. Othman Soufan

I. Anti-Plagiarism Declaration

This is to declare that the above publication produced under the supervision of Dr. Othman Soufan, having the title "Young Minds Learners Hub" is the sole contribution of the authors and no part hereof has been reproduced illegally, which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: 21 Dec 2024

Author(s):

Name:	Hussain Timah	Signature: *Hussain*
Name:	Khalid Namat Allah	Signature: *Khalid*
Name:	Abdullah Hasan Prwm	Signature: *Prwm*
Name:	Khaled Salem	Signature: *Khaled*

II. Acknowledgment

We would like to express our special thanks to Dr. Othman Soufan who helped and guided us to learn new things. Furthermore, many thanks to the Software Engineering Department and College of Computer Science as well for their support.

III. Abstract

The Young Minds Learners Hub is an innovative platform designed to empower Arabic-speaking children by providing an engaging and interactive learning environment. The system offers tailored courses, quizzes, and coding challenges, fostering a deeper understanding of software engineering concepts. With dedicated interfaces for instructors, students, and parents, the platform ensures ease of use, progress tracking, and parental involvement. The platform prioritizes accessibility, usability, and a gamified approach to learning, creating a comprehensive educational experience for the next generation of innovators.

Keywords: Interactive Learning Platform, Arabic Education, Coding for Children, Coding Education.

Table of Contents

IV. List of Diagram

V. List of Tables

VI. Acronym

YML Hub: Young Minds Learners Hub
MVT: Model-View-Template

CHAPTER 1 – INTRODUCTION

1.1 Overview

Today, the quality of education and learning tools has become an integral part of the rapidly innovating world. Young Minds Learners Hub, YML Hub, is a project that aims to fill the void for an interactive learning platform specifically for young learners, especially Arabic-speaking countries. The hub maintains its focus on providing a robust technological interface and user-centric design with the hope of creating a lively, efficient, and scalable educational experience. Through progressive tracking, content management, and gamified learning modules, the YML Hub includes its features regarding global standards aligned with modern education.

The system YML hub is meant to facilitate the process of teaching and learning by providing the instruments that matter to teachers, parents, and students. The system is a platform for role users so easily afford and safe-guarded with multilingual capability to be inclusive and relevant of this different learning-community settings.

1.2 Project Motivation

Educational technology stands as one of the major pillars for the coming future; however, many existing systems do not cater to the specific culture and languages of young learners in the Arab-speaking world. Thus, it brings a motivation to establish a learning hub that will create such a needed bridge towards modern teaching principles.

The motivation behind this project is to improve the whole educational experience by featuring instructiveness in modules as well as customizing content and essential learning tools for teachers and parents. The project YML Hub thus appeals to current issues in accessibility and engagement-the pleasantness and impact of learning.

1.3 Project Objectives

Educational institutions try to fill the apparent voids in learning methodologies to strength and train students with the help of enhanced and effective technology. The Young Minds Learners Hub aims to create a conduit between the conventional mode of education and the prevailing educational paradigm by providing an interactive platform interfacing students, teachers, and parents. The prime objectives of this innovation include:

To analyze some of the platforms that can be explored for their advantages tied into aspects like curriculum management, user engagement, and support in required languages.

To design and realize a scalable and user-friendly platform to serve the varying interests of learners, parents, and instructors.

To develop a complete software architecture for the system reflexive of its functionalities, structural breakdown, and workflows by means of diagrams.

To create an intuitive and interactive learning solution, providing for:

a. Secure user management for administrators, students, parents, and instructors.

b. Facilities for instructors to upload, update, and manage learning materials or lessons, assignments, and grades.

c. Monitoring of student activity and progress through a dashboard, along with the ability to generate reports and send messages among users.

d. Multilingual support, especially in Arabic, to allow easier access for regional users.

The proposed solution envisions a new generation of learning platforms that will not only engage the learning community but also provide effective interfacing with administrative management, thus catering to the educational interests of all stakeholders.

1.4 Project Deliverables

Outputs of the capstone project for the Young Minds Learners Hub (YML Hub) system includes the following:

1 -Documented functional and non-functional requirements.

2 -Use case diagrams and corresponding table descriptions.

3 -Sequence diagrams illustrating system interactions.

4 -Activity diagrams depicting workflow and processes.

Expected outputs from the design phase of this educational platform will comprise:

1 -A full architectural diagram defining the system core structure.

2 -Component diagrams that interpret the modularity of the system.

3 -Class diagrams outlining the object-oriented implementation of the platform.

4 -Entity Relationship (ER) diagrams representing the database schema.

5 -Database schema diagram that expounded on the detailed data model.

6 -User interface (UI) design to manifest how the platform looks and works.

Expected outputs from the implementation phase of this learning management system will include the following:

1- A web-based platform developed with Django as the backend framework.

2- Frontend implementation using HTML, CSS, and JavaScript integrated with React.js for dynamic and interactive design.

3- API endpoints built with Django REST Framework for secure and efficient communication with the frontend and backend.

4- A relational database developed and maintained with PostgreSQL functionalities.

5- Support for multi-language capabilities (in Arabic and English) to serve the wider user base.

Such a structured approach guarantees the functionality, user-friendliness, and expandability of the platform in line with the prescribed requirements and targets.

1.5 Project Timeline

Our project follows a detailed and sequential approach in alignment with the Software Development Life Cycle (SDLC), using the waterfall methodology because of its clear and manageable structure. Figure 1.1 shows that each phase of the project has been allotted certain weeks, with milestones against which progress shall be tracked. This will provide a structured timeline, and every team member knows what is expected of them at any given time; hence, revisiting any phase would not be necessary unless adjustments are needed.

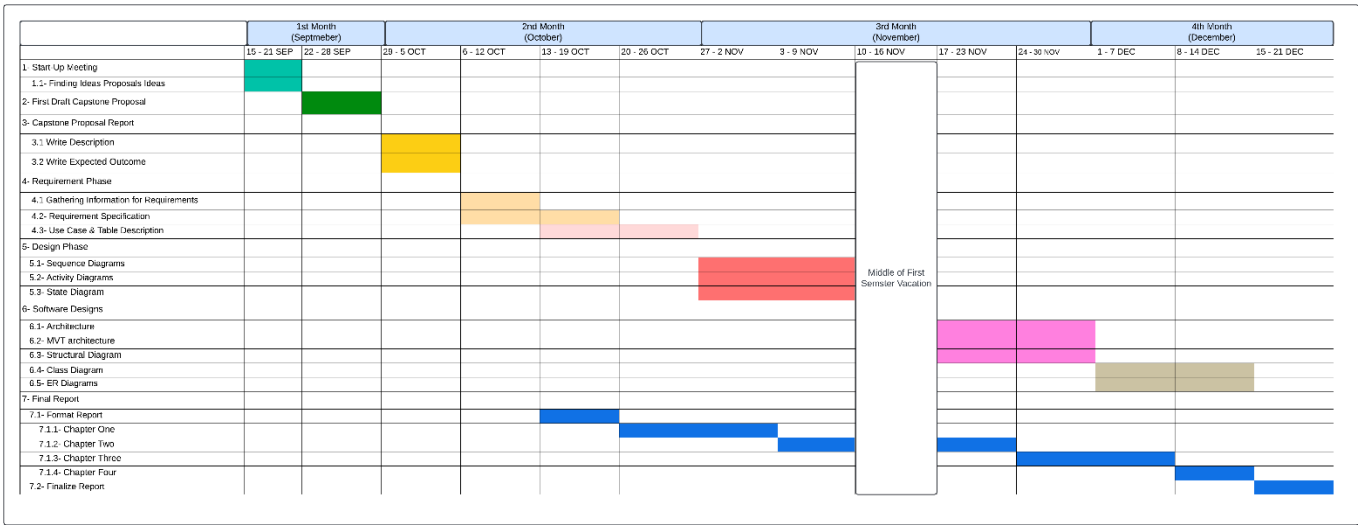


Figure 1.1 - Gantt Chart for 1st Semester

1.6 Summary

In summary, this chapter introduces the project's foundation, a presentation of the purpose and drive to work on it. A key feature in the overview points to the importance of making such an innovative educational platform and its special relevance to Arabic-speaking young learners; in its motivation, it explains what may have been lacking regarding the cultural relevance or actual use of the learning system.

The project objectives put forth are to develop a modular, user-friendly platform with multilanguage support, advanced tracking features, teacher and parent tools, and functionalities for students. This current chapter also specifies the following deliverables for the system: detailed documentation, a diagram of the system itself, and a functional Web-based platform. These components coming together ensure that the focus of the project is within modern educational standards and stakeholder needs.

CHAPTER 2 - SOFTWARE REQUIREMENTS AND ANALYSIS

This chapter outlines the functional and non-functional requirements of the Young Minds Learners Hub system. It also includes analysis models such as use case diagrams, sequence diagrams, and activity diagrams. The chapter demonstrates the requirement engineering process, including elicitation, analysis, specification, and validation.

2.1 Functional and Non-functional Requirements

During the requirements engineering process, information was gathered through research, stakeholder interviews, and surveys. Relevant requirements were organized into clusters, such as user management, course management, progress monitoring, and content management. The prioritized requirements are detailed below.

2.1.1 Functional Requirements

Table 1.1 - Functional Requirements

Requirement Group	Requirement Statement
1 .User Management:	**FR 1.1** - The system shall allow administrators to create, update, and delete user accounts for students, instructors, and parents.
	FR 1.2 - The system should provide an enrollment feature where students can register for available courses.
	FR 1.3 - The system shall ensure secure authentications for all users via a username and password.
2. Course Management:	**FR 2.1** - The system shall allow administrators and instructors to create, modify, and delete and browse courses and courses timetables.
	FR 2.2 - The system shall support the creating of assignment and grading of assignments and quizzes.
3. Progress Monitoring:	**FR 3.1**- The system shall allow students to track their progress on courses via a dashboard.
	FR 3.2- The system shall allow Administrator, instructors and to track students' progress on courses via a dashboard.
	FR 3.3 - The system should generate progress reports for each student.
	FR 3.4- The system shall allow parents to track their children's progress via a dashboard.

4. Content Management:	**FR 4.1** - The system shall allow instructors to upload course materials, such as PDFs, videos, and interactive content.
	FR 4.2 - The platform should clearly define and explain the objectives of learning each programming language to students, emphasizing why they are studying that language and what skills they will gain from it.
	FR 4.3 - The system shall provide instructors with the ability to edit or remove outdated content.
	FR 4.4 - The system shall allow students to submit assignments or homework directly through the platform.
	FR 4.5 - The system should provide instructors with the ability to review, grade, and provide feedback on student submissions.
	FR 4.6 - The system shall notify students when feedback is available or when deadlines are approaching.
5. Interactive Learning Modules:	**FR 5.1** - The system shall support the creation of interactive learning modules that engage children in learning software engineering concepts.
6. User Role Management:	**FR 6.1** - The system shall define and manage multiple user roles such as administrator, Instructor, student, and parent.
	FR 6.2 - The system should provide access control, ensuring that each user role can only access the features and data appropriate for their role.
7. Search and Navigation:	**FR 7.1** - The system shall include a search function that allows users to find courses, learning modules, or assignments.
	FR 7.2 - The system should provide a navigation menu for users to easily access different sections of the platform, such as courses, reports, and communications.
8. User Support and Help:	**FR 8.1** - The system should allow users to submit support tickets or feedback to the administrators for assistance.
9. Multilingual Support:	**FR 9.1** - While Arabic is the primary language and the secondary language is English, the system should allow for future expansion to support other languages if needed.
	FR 9.2 - The system shall provide easy switching between languages where applicable.

2.1.2 Non-functional Requirements

Table 1.2 - Quality Attribute Requirements

Quality Attribute	Requirement Statement	Priority
1. Performance:	**NFR 1.1** - The system shall load pages and interactive modules within 4 seconds under normal load conditions.	**High**
	NFR 1.2 - The system should handle up to 80 concurrent users without significant degradation in performance.	**Medium**
2. Security:	**NFR 2.1** - The system shall encrypt all sensitive data, including user credentials and student performance data.	**High**
	NFR 2.2 - The system should ensure compliance with relevant data protection standards and regulations.	**High**
	NFR 2.3 - The system shall implement secure data transfer mechanisms for communication with external tools and databases.	**High**
3. Usability:	**NFR 3.1** - The system should be intuitive and easy to use for children and non-technical users, with a simple and clear interface.	**Medium**
	NFR 3.2 - The system shall support Arabic as the primary language for all user interfaces.	**High**
5. Availability:	**NFR 4.1** - The system shall have an uptime of at least 99% to ensure availability for students, instructors, and parents.	**Medium**
	NFR 4.2 - The system should be accessible across various devices, such as desktops, tablets, and mobile phones (even though cross-platform software design is out of scope).	**Low**

2.1.3 Requirement Validation and Verification

We concluded requirement validation by checking the characteristics of the functional and non-functional requirements, ensuring their quality through various verification and validation (V&V) techniques. The following criteria were applied:

1. **Realism:**
   The requirements were evaluated to determine if they can be met with current technology and knowledge. This was done by comparing it with other similar educational platforms and seeking advice from e-learning specialists.
2. **Clarity & Crystal**:
   requirements were checked and analyzed to make sure that they are comprehensible and do not have any ambiguities. The use of team reviews also helped reduce the number of interpretations made, making documentation.
3. **Consistency:**
   Efforts were made to review requirements to avoid contradictions. Grouping similar requirements and analyzing dependencies helped detect and resolve potential conflicts.
4. **Verifiability:**
   All requirements were made specific and quantifiable so that they could be easily assessed. A question-based approach was used, where it was clearly stated what specific metrics and conditions should be met to ensure that requirements are fulfilled.
5. **Prioritization:**
   The non-functional requirements were prioritized according to their relevance to the performance of the system, the satisfaction of the users.
6. **Completeness:**
   The use of prototypes and feedback sessions with potential users helped to avoid the exclusion of critical requirements.

To obtain these characteristics, the following V&V techniques were employed: Some of the activities that were implemented included walkthroughs and team reviews with the aim of detecting errors and enhancing the quality of requirements.

2.1.4 Interview

Saad Bin Ghaith Al Ansari School Administration has seasoned with years of experience in managing educational institutions and fostering the adoption of technology in classrooms.

During our meeting with The School Administration, the primary objective was to gather insights on the challenges schools face in teaching programming and leveraging technology for young learners. They emphasized the need for an engaging and easy-to-use platform that caters specifically to Arabic-speaking students while providing flexibility for future expansion.

Key feedback and insights included:

- The importance of interactive and gamified learning modules to maintain student engagement and motivation.
- A need for parental monitoring tools that provide insights into students' progress and achievements.
- Recommendations to include an instructor dashboard to streamline assignment management, grading, and progress tracking.
- A strong emphasis on Arabic is the primary language for the interface, while providing English as an optional language.
- The suggestion to include a tutorial or onboarding system to guide Instructors, parents, and students through the platform's features.

The School Administration praised the system's potential to address gaps in Arabic programming education. He also highlighted the need for regular feedback loops from users to refine the platform further.

With The School Administration's valuable expertise and detailed feedback, we refined the requirements to align better with the needs of educational institutions, ensuring the platform is practical, scalable, and impactful.

2.2 Analysis Models

The Young Minds Learners Hub system analysis is presented using functional and dynamic models. These models illustrate the interactions between actors and the system, its behavior, and its workflows. This section includes use case diagrams, sequence diagrams, and activity diagrams to detail the system's design.

2.2.1 Functional Models

This section showcases our functional model, which utilizes use case diagrams, offering a comprehensive overview of the system. It provides detailed descriptions for three specific use cases: "Add Quizzes," "Add Assignments," and "Add Courses," while also exploring the validation process of the use case diagrams.

2.2.1.1 Use Case Diagrams

Use case diagrams illustrate the interactions between the Young Minds Learners Hub system and its environment. The main focus lies in describing the flow of events between actors and the system. Actors include end-users such as students, Instructors, parents, and administrators, as well as any external systems that interact with the platform.

Here, we have four use case diagrams: Diagram 2.1 provides an overview of the system's functionality, illustrating interactions between all actors and major use cases. Diagram 2.2 details the "Add Courses" use case, demonstrating course creation, modification, and deletion processes. These diagrams are accompanied by textual descriptions to clarify the interactions and system workflows. Diagram 2.3 showcases the "Add Assignments" use case, outlining assignment distribution, submission, and feedback. Diagram 2.4 focuses on the "Add Quizzes" use case, providing a detailed view of quiz creation, participation, and grading.

Diagram 2.1 - Use Case Diagram Young Minds Learners Hub (Top Level)

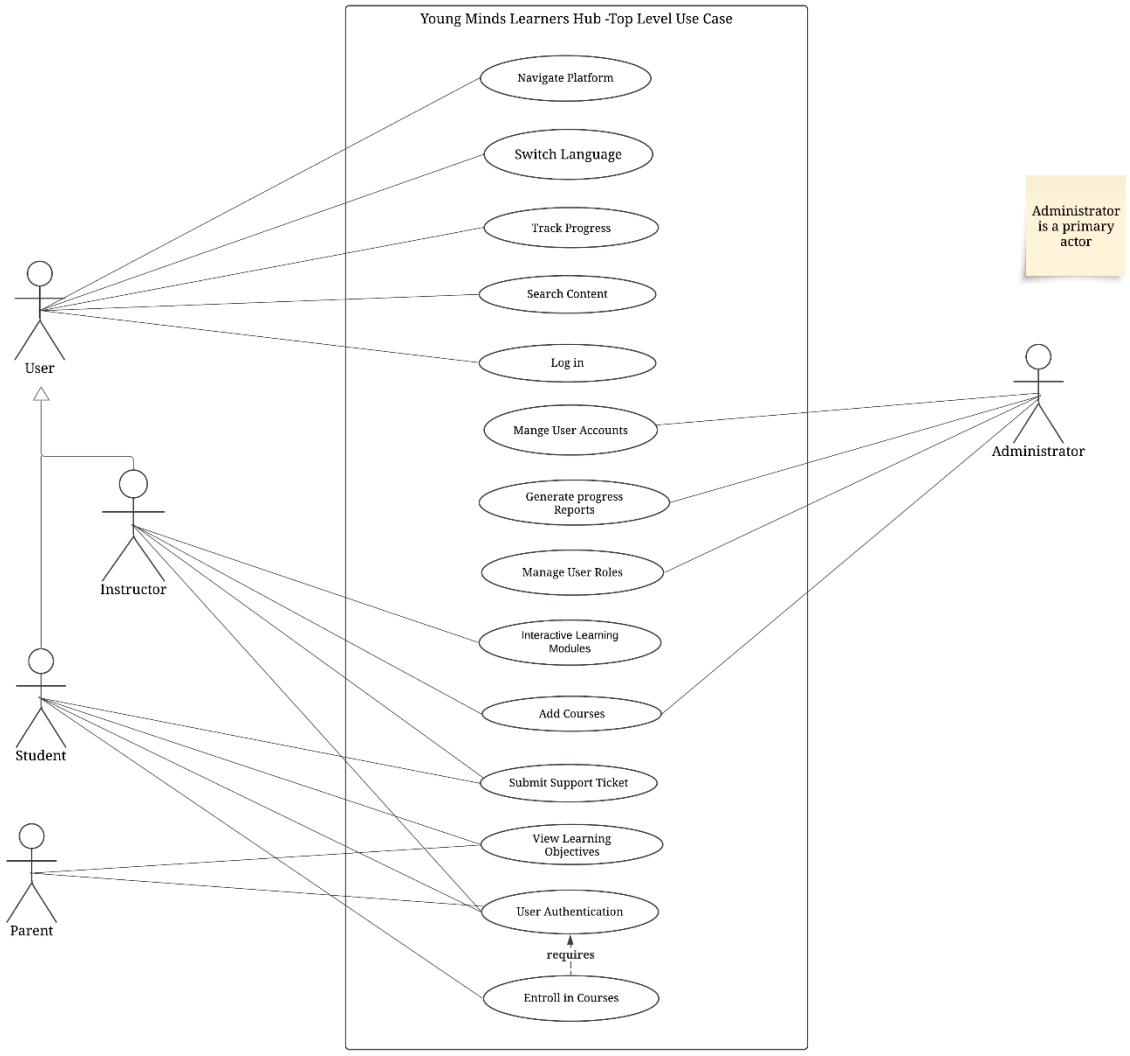


Diagram 2.2 - Use Case Diagram for "Add Courses"

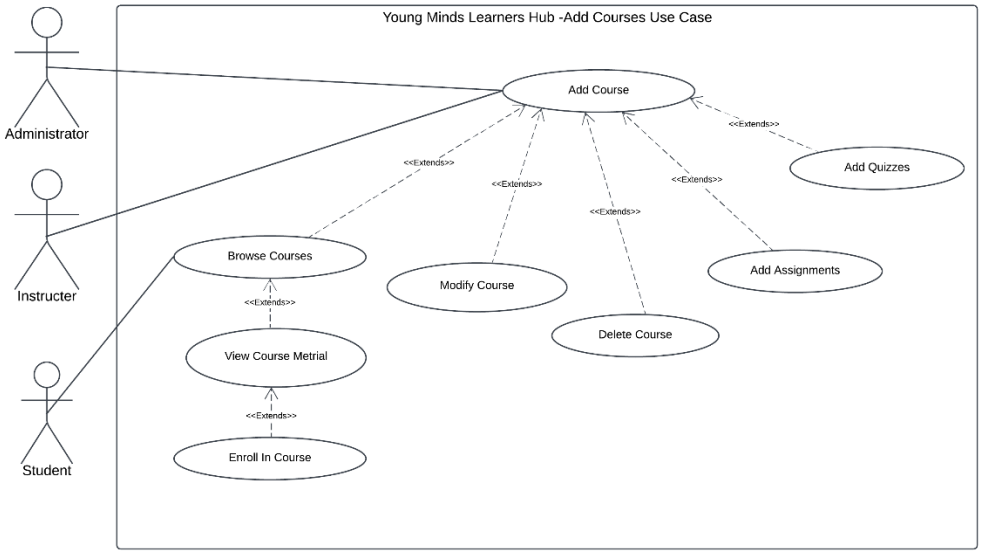


Diagram 2.3 - Use Case Diagram for "Add Assignments"

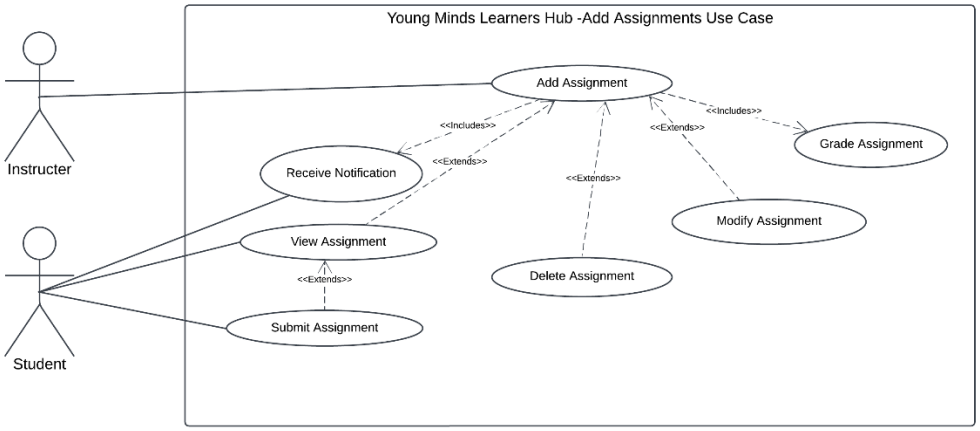
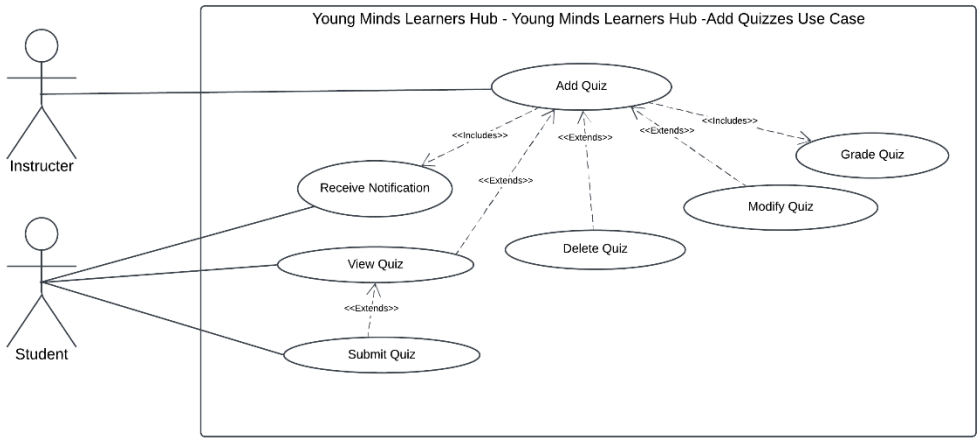


Diagram 2.4 - Use Case Diagram for "Add Quizzes"



2.2.1.2 Use Case Descriptions

We have selected some specific use cases to describe in detail: the "Add Quizzes" use case, the "Add Courses" use case, and the "Interactive Learning Modules" use case, which are key components of the system. The tables labeled "2.1", "2.2", and "5.1" provide detailed descriptions for these use cases.

Table 2.1 - Use Case Description for "Add Quizzes"

Table Description	
ID	UC-2.1
Name	Add Quizzes
Actor	Instructor, Student
Related Use Cases	This use case is part of the "Add Quizzes" process.
Description	The "Add Quizzes" use case allows Instructors to create, edit, and delete quizzes. Students can participate in quizzes, view their results, and track progress.
Preconditions	The user (Instructor or student) must have a valid account and be associated with a course.
Basic Scenario	1. The instructor logs into the system. 2. The instructor selects "Add Quizzes" from the dashboard. 3. The instructor creates or modifies a quiz by adding questions and setting parameters. 4. Students access and attempt the quiz. 5. The system automatically grades the quiz or allows manual grading. 6. The system updates the results and notifies students.
Failure	The quiz cannot be created, modified, or submitted due to system errors.
Failure Conditions	The quiz cannot be saved due to a network error or an invalid configuration.
Postconditions	The quiz is successfully created, completed, or graded, and results are updated in the system.

Table 2.2 - Use Case Description for "Add Courses"

Table Description	
ID	UC-2.2
Name	Add Courses
Actor	Administrator, Instructor
Related Use Cases	This use case is in the use case "Add Courses"
Description	The "Add Courses" use case gives administrators and Instructors the ability to create, edit, delete, and browse courses in the system. Instructors can also associate materials, assignments, and quizzes with courses.
Preconditions	The user must have a valid account with Administrator or Instructor privileges.
Basic Scenario	1. The user logs in to the system. 2. The user selects "Add Courses" from the dashboard. 3. The user chooses to create a new course or edit/delete an existing course. 4. The user enters or updates course details such as title, description, and schedule. 5. The system validates the information and saves the changes. 6. The updated course catalog is displayed.
Failure	The course cannot be added or updated by the system.
Failure Conditions	The database (DB) connection is unavailable due to a network failure.
Postconditions	The course is successfully added, updated, or deleted in the system and is visible in the course catalog.

Table 2.3 - Use Case Description for "Interactive Learning Modules"

Table Description	
ID	UC-5.1
Name	Interactive Learning Modules
Actor	Instructor, Student
Related Use Cases	This use case is part of the "Create and Participate in Interactive Learning Modules" process.
Description	The "Interactive Learning Modules" use case allows Instructors to create engaging, hands-on modules for children to learn software concepts interactively. Students participate in these modules to develop their skills through activities such as coding challenges, quizzes, and simulations. Instructors can design learning activities such as: - Coding Challenges: Exercises to practice programming concepts. - Simulations: Real-world scenarios to apply software principles.
Preconditions	Instructors must have a valid account to create modules. Students must be registered for the course to access the modules.
Basic Scenario	1. The instructor logs into the system. 2. The instructor selects "Create Interactive Module" and enters details such as objectives, activities, and expected outcomes. 3. The system validates and saves the module. 4. Students access the module through their dashboard and engage with the activities. 5. The system tracks the progress and records completion.
Failure	The interactive module cannot be created or accessed due to technical issues.
Failure Conditions	The system fails to save the module due to invalid inputs or a database error, or the module activities fail to load due to network issues.
Postconditions	The interactive learning module is successfully created, accessed, and completed by students, with progress tracked and recorded.

2.2.1.3 Use Case Diagram Validation

The use case diagrams for the Young Minds Learners Hub system were validated through two distinct checks: syntax and semantics.

1- Syntax Check: we ensured the diagrams adhered to UML standards by verifying that actors were depicted as "stickman" icons with appropriate names and positioned outside the system boundary. Use cases were represented by ellipses with verb-like names (e.g., "Submit Assignment") and connected to actors using proper UML relationships, such as «include» and «extend». These relationships were accurately labeled and adhered to UML conventions, ensuring structural correctness.

2- Semantic Validation: we confirmed that all actors represented real-world entities interacting with the system, such as students, teachers, parents, and administrators. The relationships between use cases, such as inclusion and extension, were checked for logical consistency and alignment with system functionalities. Additionally, we ensured that all use case names were meaningful and descriptive, providing clear context for their roles in the system.

2.2.2 Dynamic Models

To illustrate the dynamic behavior of the Young Minds Learners Hub system, two types of UML diagrams have been utilized: interaction diagrams and behavior diagrams. Interaction diagrams focus on the dynamic behavior between system components by showing their interactions through message exchanges. These exchanges often include method executions triggered by one object and may involve sending messages to others, as represented in sequence diagrams. On the other hand, behavior diagrams provide an overview of the internal workflow or state transitions of a single object or a group of objects, such as in activity diagrams.

2.2.2.1 Interaction Diagrams

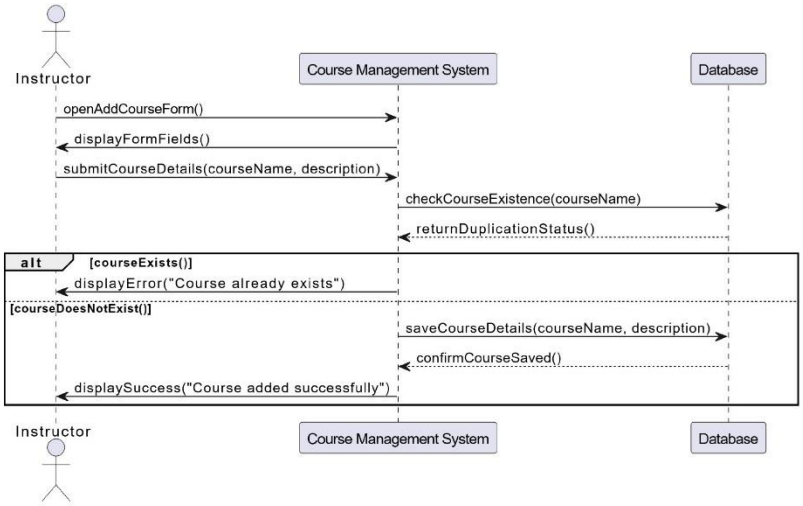
This section delves into interaction diagrams, particularly sequence diagrams, which detail the sequential interactions among system components. Sequence diagrams capture the dynamic interplay of actors, processes, and system elements, making them ideal for describing the flow of control. For demonstration purposes, the "Add Course" use case was selected to highlight the step-by-step process of validating and storing course details within the system.

2.2.2.1.1 Sequence Diagram

The sequence diagram below represents the "Add Course" use case. It demonstrates how an instructor interacts with the system to log in, initiate the addition of a course, and submit course details. The system verifies the instructor's credentials, validates the provided course details, and stores the course data if validation is successful. The diagram also captures alternative flows, such as validation errors, to ensure comprehensive documentation of the process.

Design A

Diagram 2.5 - Sequence Diagram for "Add Course"



Description:

- Instructor Interaction:

The Instructor initiates the process by requesting to open the "Add Course Form" on the Course Management System.
The Course Management System responds by displaying the form fields to the Instructor.

- Form Submission:

The Instructor fills out the course details (e.g., courseName and description) and submits the information to the Course Management System.

- Course Duplication Check:

The Course Management System sends a request to the Database to check if a course with the same name already exists.
The Database processes the request and returns the duplication status to the Course Management System

- Alternative Paths:

If the course already exists:
The Course Management System displays an error message to the Instructor: "Course already exists."

- If the course does not exist:

The Course Management System saves the course details (e.g., courseName and description) into the Database.
The Database confirms that the course details have been saved successfully.

The Course Management System notifies the Instructor with a success message: "Course added successfully."

- End of Interaction:

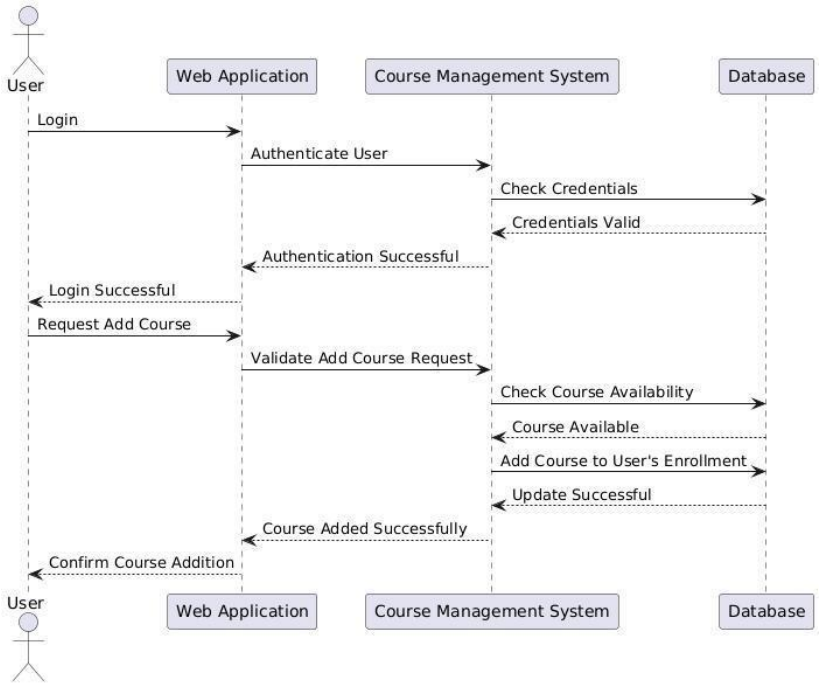
The sequence concludes once the success or error message has been displayed to the Instructor.

Purpose:

This diagram models the process of adding a course while ensuring that no duplicate courses are allowed. It captures key decision points (e.g., whether the course already exists) and interactions between the Instructor, the Course Management System, and the Database.

Design B

Diagram 2.6 - Sequence Diagram for "Add Course"



Description:

- User Authentication:

1. The process begins with the User logging into the system via the Web Application.
2. The Web Application forwards the authentication request to the Course Management System (CMS).
3. The CMS checks the user's credentials against the Database.
4. Upon successful validation, the login confirmation is passed back through the CMS and Web Application to the User.

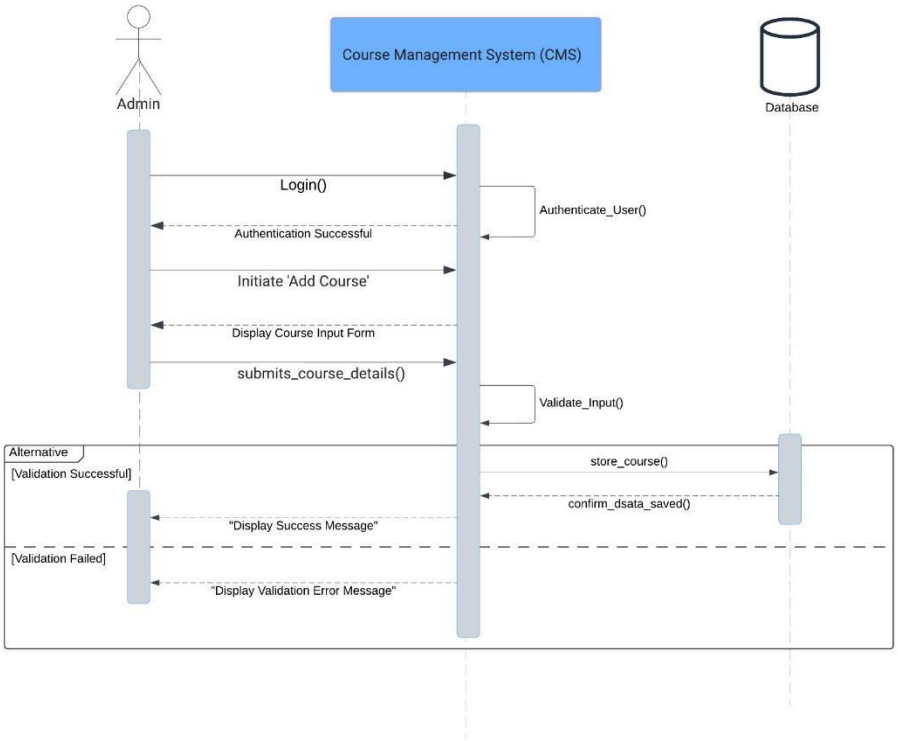
- Course Addition:

1. After successful login, the User requests to add a course through the Web Application.
2. The Web Application sends this request to the CMS for validation.
3. The CMS checks the course availability by querying the Database.
4. If the course is available, the CMS instructs the Database to add the course to the user's enrollment.
5. The Database confirms the successful update.
6. The confirmation of course addition is then passed back through the CMS and Web Application to the User.

This sequence diagram effectively captures the flow of interactions in the course addition process, highlighting the system's attention to user authentication, data validation, and user feedback. It provides a clear visualization of how different components of the YML Hub system interact to fulfill a core functionality.

Design C

Diagram 2.7 - Sequence Diagram for "Add Course"



This sequence diagram illustrates the "Add Course" process in the Course Management System (CMS).

The process begins with the Admin logging in and being authenticated via the Authenticate_User() method.

Upon successful authentication, the Admin initiates the "Add Course" action, and the system displays the course input form.

The Admin submits the course details, which are validated by the system using the Validate_Input() function.

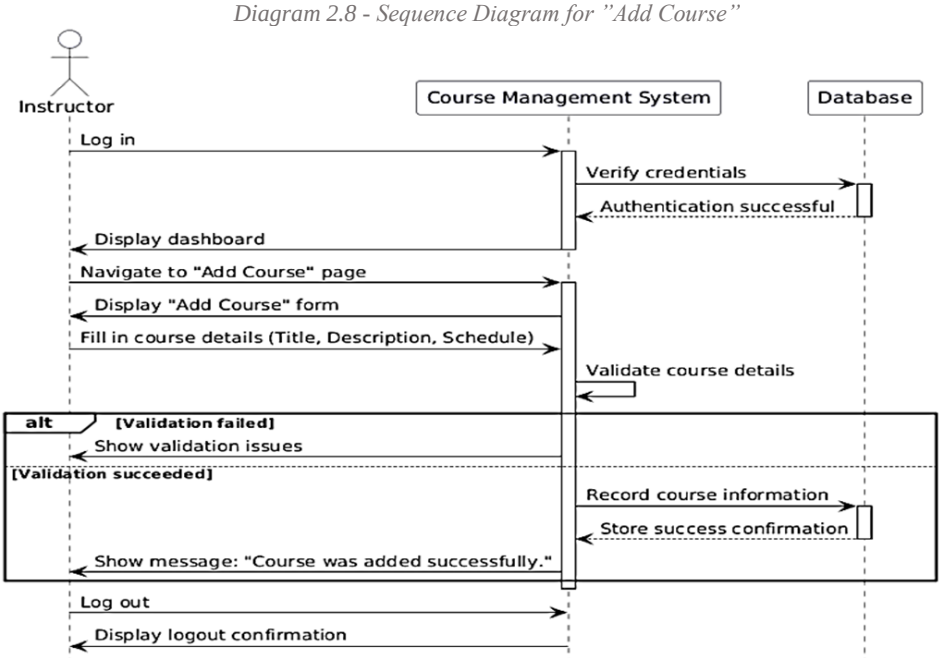
If the validation is successful, the course data is stored in the database using the store_course() method, and a confirmation message is displayed.

In case of validation failure, an error message is returned, prompting the Admin to correct the input.

This diagram highlights the interactions between the Admin, the CMS, and the database, ensuring a clear and logical flow for course creation.

Design D

Diagram 2.8 - Sequence Diagram for "Add Course"



This sequence diagram illustrates the interaction between an Instructor, a Course Management System (CMS), and a Database during the process of logging in, adding a course, and logging out.
Description of Steps:
1- Login Process:
- The Instructor logs into the system.
- The CMS verifies the credentials with the Database.
- Once the credentials are authenticated successfully, the CMS displays the Instructor's dashboard.
2- Navigating to Add Course:
- The Instructor navigates to the "Add Course" page in the CMS.
- The CMS displays the "Add Course" form for the Instructor.
3- Adding Course Details:
- The Instructor fills in the course details, such as Title, Description, and Schedule.
- The CMS validates the entered course details.
4- Validation:
- If validation fails:
. The CMS shows validation issues to the Instructor (e.g., missing, or incorrect fields).
- If validation succeeds:
. The CMS sends the course information to the Database.
. The Database stores the course information and sends a success confirmation back to the CMS.
. The CMS displays a message to the Instructor: "Course was added successfully."
5- Logout Process:
- The Instructor logs out of the CMS.
- The CMS displays a logout confirmation to the Instructor.

2.2.2.1.2 Design Selection Process

The design selection process is a systematic approach to identify and choose the most suitable design among various alternatives. The process ensures that the selected design aligns with the project's objectives and requirements. Below are the steps undertaken in this process:

1. **Define Evaluation Criteria**

   The selection begins by establishing key evaluation criteria relevant to the project. These criteria include:

   - **Clarity**: The ease with which stakeholders can understand the design.
   - **Completeness**: The extent to which the design covers all required interactions and scenarios.
   - **Consistency**: The alignment of the design with existing system architecture and other diagrams.
   - **Compliance**: Adherence to established UML or industry standards.
   - **Efficiency**: The optimization of interactions, ensuring minimal redundancy or unnecessary processes.

2. **Scoring and Ranking Designs**

   Each alternative design is scored against the criteria using a standardized scale (e.g., 1 to 5, where 1 is poor and 5 is excellent). Team members independently evaluate the designs, and the scores are averaged to ensure objectivity.

Table 2.1 Evaluation Criteria

Alternative Design	Clarity	Completeness	Consistency	Compliance	Efficiency	Average Score
Design A (Hussain Timah)	2	3	4	4	3	3.2
Design B (Khalid Namat Allah)	4	3	4	3	5	3.8
Design C (Abdullah Hasan Prwm)	5	4	4	5	4	4.4
Design D (Khaled Salem)	3	3	4	3	4	3.4

3. Based on the scores, **Design C** achieves the highest average and is selected as the optimal choice.

4. **Justification of Selection**

   The selected design, **Design C**, demonstrates superior performance in all critical areas. Its high scores in clarity, completeness, and compliance ensure it meets stakeholder expectations and aligns with industry standards. Additionally, its efficiency optimizes system interactions, making it well-suited for implementation in the Young Minds Learners Hub project.

Validation and Refinement

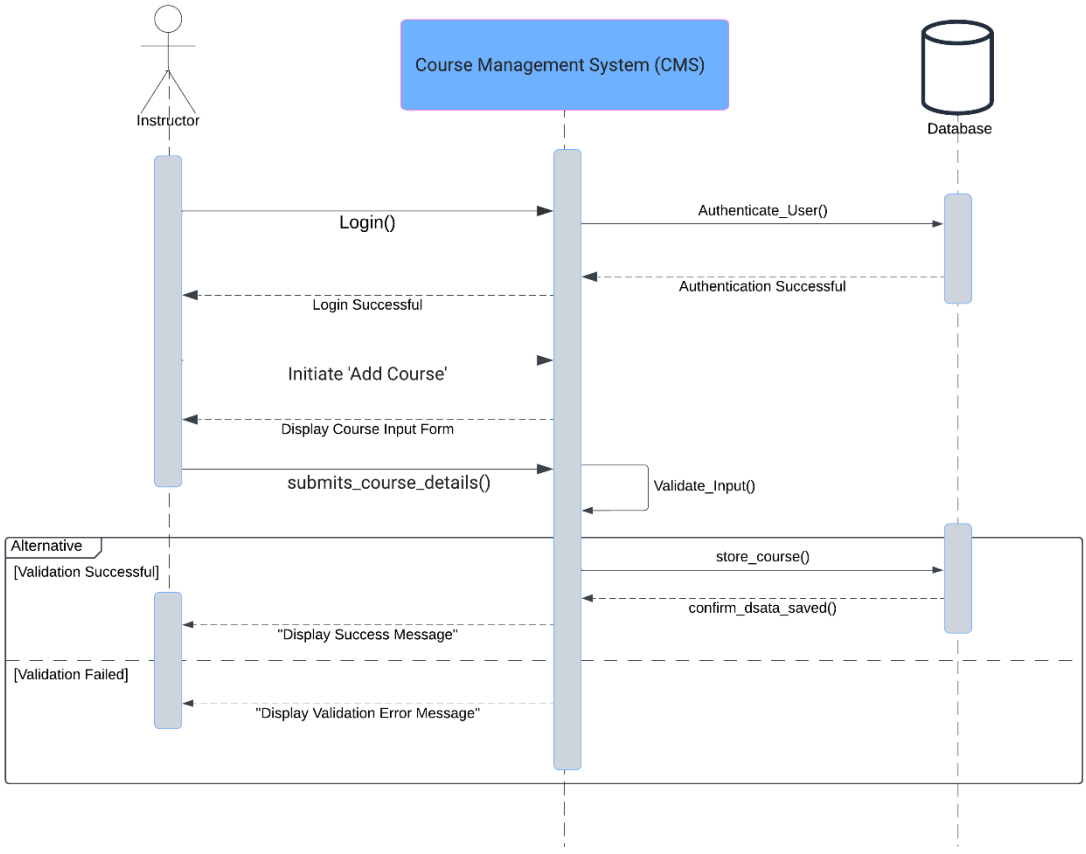
Following the selection, the chosen design undergoes validation to confirm its alignment with system requirements. Any necessary refinements are implemented to address potential gaps or areas for improvement.

2.2.2.1.3 Improvements of Final Sequence Diagram

The final sequence diagram incorporates several improvements and changes compared to Design C.

These modifications enhance the clarity, completeness, and accuracy of the system representation.

Diagram 2.9 - Sequence Diagram for "Add Course"



Key Changes and Improvements:

1. **Clarity and Structure:**

- The flow of interactions is more logically structured, following a clear sequence from login to course addition.

2. **Detailed Authentication Process:**

- The login process is more detailed, showing the flow from the Instructor through the Web Application to the CMS, and then to the Database for credential checking.

3. **Database Interactions:**

- The diagram clearly shows when and how the Database is involved in both the authentication and course addition processes.

Summary:

These enhancements result in a more comprehensive, realistic, and detailed representation of the course addition process, building upon the strong foundation provided by Design C.

2.2.3Behavior Diagram

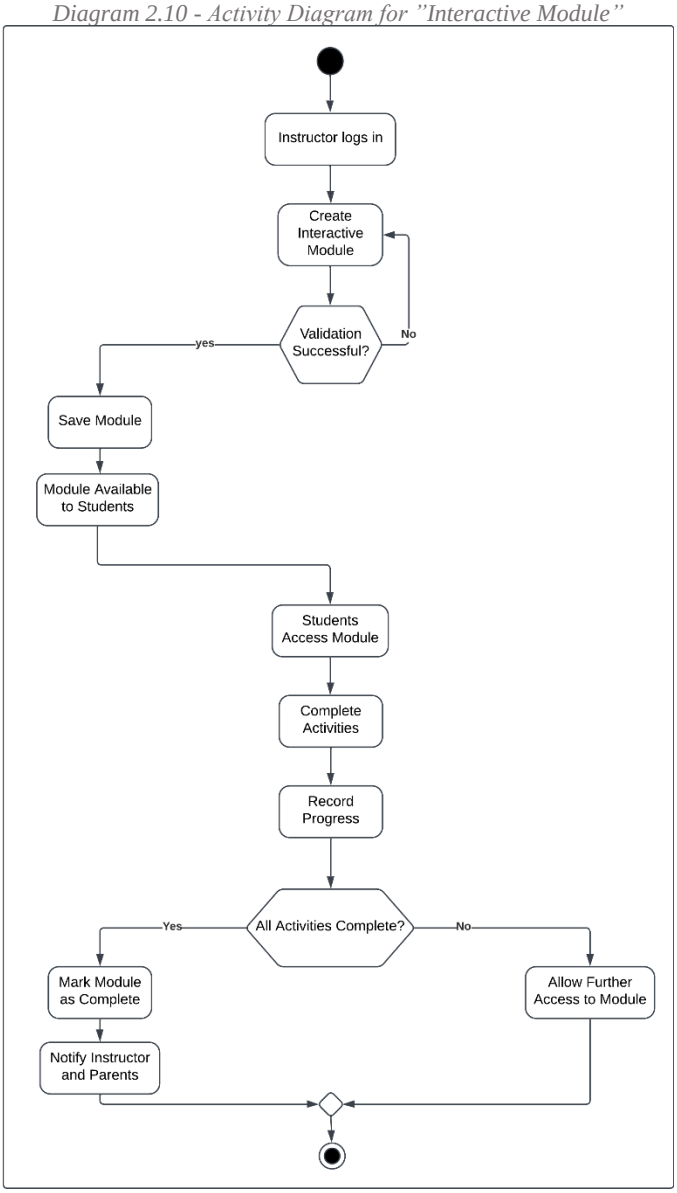
This section presents and validates two key behavior models within the Young Minds Learners Hub system: the Activity Diagram and the State Diagram. These diagrams provide insights into the system's workflows and responses to various events.

2.2.3.1 Activity Diagram

Activity diagrams are used to represent the dynamic aspects of a system, focusing on the sequence of activities and their associated conditions. They are particularly useful for modeling workflows or processes within a system. Unlike sequence diagrams, which emphasize interactions between objects, activity diagrams focus on the logical flow of tasks and decisions.

For this system, the " Interactive Module Workflow" use case is illustrated to highlight its functionality.

Diagram 2.10 - Activity Diagram for "Interactive Module"



2.2.3.2 State Diagram

A state diagram illustrates a system's behavior by showing the transitions between states triggered by specific events or conditions. Unlike sequence diagrams, which focus on object-level interactions, state diagrams emphasize how the system responds to various events during its lifecycle. The "Enrollment Course" process was selected due to its distinct state transitions and importance within the system.

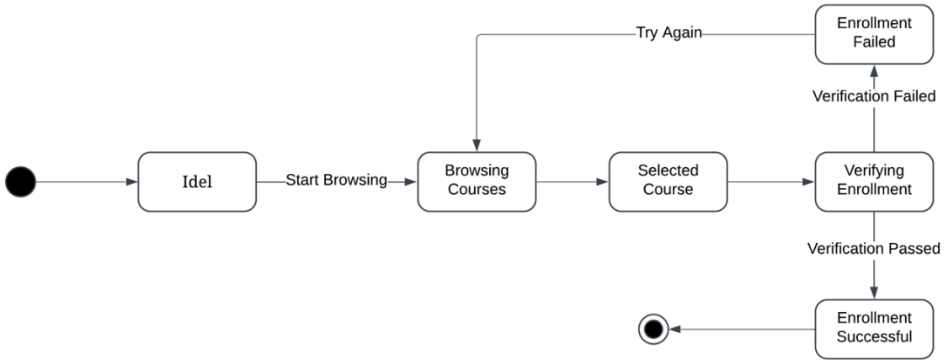
Diagram Description:

The state diagram begins with an Idle state, where the user has not yet initiated any action.

The user progresses through states such as Browsing Courses, Viewing Course Details, and Enrollment Verification, depending on the actions taken.

If verification is successful, the system transitions to the Enrolled state; otherwise, it moves to Enrollment Failed.

Diagram 2.11 - State Diagram for "Enrollment Course"



2.2.3.2.1 State Diagram Validation

In reviewing the state diagram, three distinct perspectives were considered: Syntax, Semantics, and Aesthetic Checks.

1. **Syntax Checks:**
   Verified that the diagram adheres to UML syntax and rules. For example:
   Ensured a single start state, depicted as a filled circle.
   Checked for end states, represented as circles with borders, in Enrollment Failed and Enrolled.
   Verified proper notation for transitions (open arrows) between states.
   Confirmed correct use of decision points for transitions like Verification Successful and Verification Failed.
2. **Semantics Checks:**
   Ensured activity names reflect the actual work carried out by the system and users.
   Verified the logical dependencies between states, such as requiring the user to Browse Courses before moving to Viewing Course Details.
   Ensured that transitions like Verification Successful correctly lead to the Enrolled state.
3. **Aesthetic Checks:**
   Maintained a manageable number of states to ensure clarity and simplicity.
   Limited decision points to no more than two transitions (e.g., Verification Successful and Verification Failed).
   Ensured all elements were placed in a logical and visually clear layout for readability.

CHAPTER 3 - SOFTWARE DESIGNS

This chapter describes the second phase of (SDLC), which is the design phase. This phase is divided into two stages: the initial design stage, which acts as a bridge between the requirements phase and the design phase. This stage assists in understanding how the system may be organized and designed, and here the role of the overall architecture appears. The second stage of design phase is the deep design stage or system design, which is extending from the previous stage, which is the process of developing abstract models of a system, with each model providing a distinct view or perspective on the system. To present the organization of the system's components and their relationships we used: components and class diagrams. Additionally, the design phase also involves creating an ER diagram and a database schema to define the structure and organization of the data that the system will have.

3.1 Overall Architecture

3.1.1 Architecture Analysis

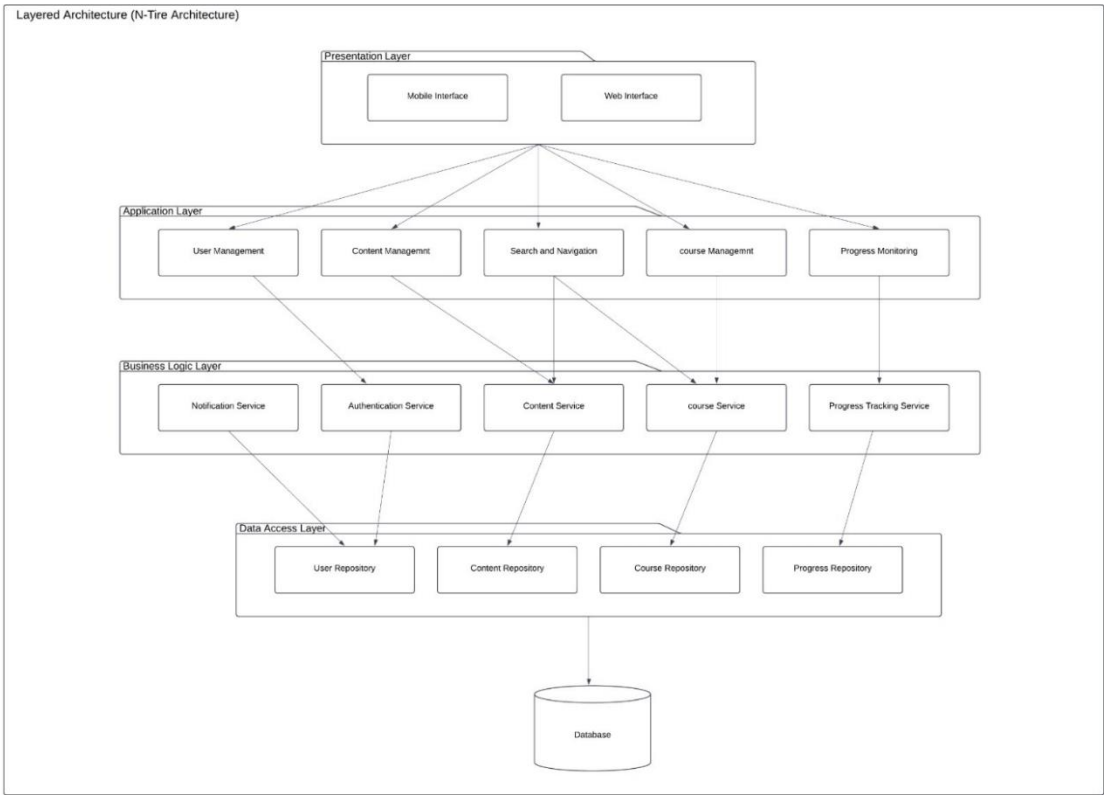
In this section, the first step was to explore the design space and select the most promising architectural structures for our needs. To accomplish this, an examination of a range of different structure/design patterns was conducted, followed by thorough analyses to determine the most suitable one. Table 4.1 has been developed to outline the various structure/ design patterns, along with justifications for choosing them as candidates or omitting them. This ensures the establishment of a solid architecture for our project.

Table 3.1 - Architecture Analysis

Architectural Structure Categories	Patterns	Result	Justification
Component-and-Connector Structures	Model-View-Template (MVT)	Candidate	The MVT pattern aligns well with the system's need for separation of concerns. The **Model** handles data and business logic, the **View** processes user requests and interacts with the Model, and the **Template** renders dynamic user interfaces. This separation improves scalability and maintainability.
	Pipe-and-Filter	Cancel	Pipe-and-Filter is unsuitable because it focuses on processing data streams, which is not the system's primary concern. Instead, MVT provides a better approach to organizing the system's components for handling educational content and user interactions.
	Client-Server	Candidate	The system uses a client-server model inherently as part of its web-based architecture, where clients interact with the server to retrieve data. This supports the modularity and ensures the system can be deployed and scaled easily in distributed environments.
Allocation Structures	Multi-Tier	Candidate	MVT complements a multi-tier architecture by naturally dividing the application into logical tiers: user interface (Template), application logic (View), and data storage (Model). This structure enhances modifiability, scalability, and availability of the system.
	Map-Reduce	Cancel	Map-Reduce is not ideal as the system does not require distributed batch processing of large datasets. Instead, the focus is on real-time user interactions and educational workflows.
Module Structures	Layered Architecture	Candidate	The MVT pattern inherently follows a layered architecture, separating the data layer, logic layer, and presentation layer. This clear division makes the system easier to maintain, test, and scale while supporting the addition of new features without impacting existing functionality.
	Microservices Architecture	Candidate	Microservices can also complement the layered architecture by dividing the system into services corresponding to each layer. For example, user management and progress tracking can function as independent services that communicate via APIs, ensuring flexibility and modularity.

3.1.2 Overall Architecture

Diagram 3.1 N-Tier Architecture



Layered Architecture (N-Tier Architecture) is a software design pattern that organizes an application into distinct layers, each responsible for specific functions. This approach enhances modularity, scalability, and maintainability of the system. In this architecture:

- 'N' represents the number of layers, typically ranging from 3 to 5.
- Each layer has a defined role and interacts primarily with adjacent layers.
- The separation of concerns allows for easier development, testing, and maintenance.
- Layers can be developed and modified independently, promoting flexibility and reusability.

The Young Minds Learners Hub (YML Hub) system utilizes a 5-tier Layered Architecture, which includes:

1. **Presentation Layer:**
    2. Components: Web Interface and Mobile Interface
    3. Purpose: This is the topmost layer that users interact with directly. It is responsible for displaying information and capturing user input.
    4. The Web Interface caters to desktop users, while the Mobile Interface is for mobile users.
    5. Both interfaces connect to all components in the Application Layer, allowing access to all system functionalities.

2. Application Layer:

1. Components: User Management, Course Management, Progress Monitoring, Content Management, and Search and Navigation
2. Purpose: This layer acts as an intermediary between the user interfaces and the core business logic. It orchestrates user actions and data flow.
3. Each component represents a major feature set of the YML Hub system, handling specific aspects of the application's functionality.

3. Business Logic Layer:

1. Components: Authentication Service, Course Service, Progress Tracking Service, Content Service, and Notification Service
2. Purpose: This layer contains the core business rules and data processing logic of the application.
3. Each service handles specific business operations. For example, the Authentication Service manages user login and security, while the Course Service handles course-related operations.

4. Data Access Layer:

1. Components: User Repository, Course Repository, Progress Repository, and Content Repository
2. Purpose: This layer is responsible for data persistence and retrieval. It provides an abstraction for database operations to the upper layers.
3. Each repository corresponds to a specific data domain and interacts with the database.

5. Database:

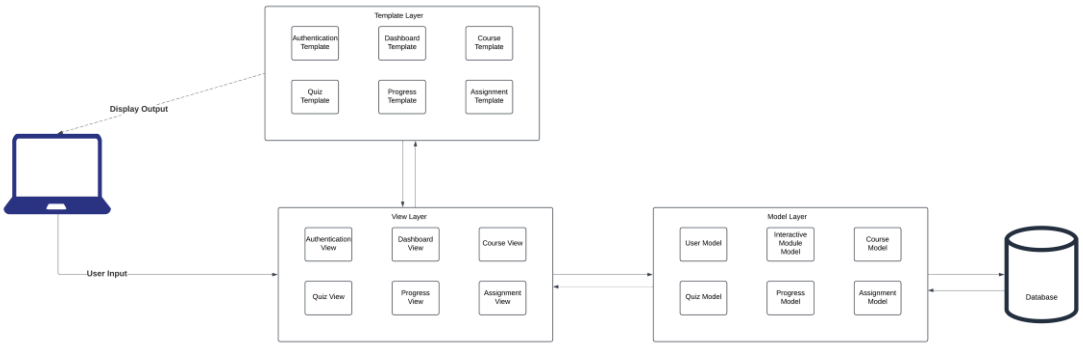
1. Component: SQL Database
2. Purpose: This is where all the application data is stored persistently.

Key Aspects of the Architecture:

1. Separation of Concerns: Each layer has a distinct responsibility, making the system more modular and easier to maintain.
2. Scalability: The layered approach allows for independent scaling of different components as needed.
3. Flexibility: New features can be added to specific layers without affecting the entire system.
4. Security: ensure the Business Logic Layer can implement security measures, while the Data Access Layer can data integrity.
5. Reusability: Common functionalities can be shared across different parts of the application.

This Layered Architecture provides a clear structure for organizing the YML Hub system's components, ensuring a robust, scalable, and maintainable educational platform. It effectively separates the user interface, application logic, business rules, and data management, allowing for easier development, testing, and future enhancements.

Diagram 3.2 Overall Architecture (MVT)



Model-View-Template (MVT) Pattern

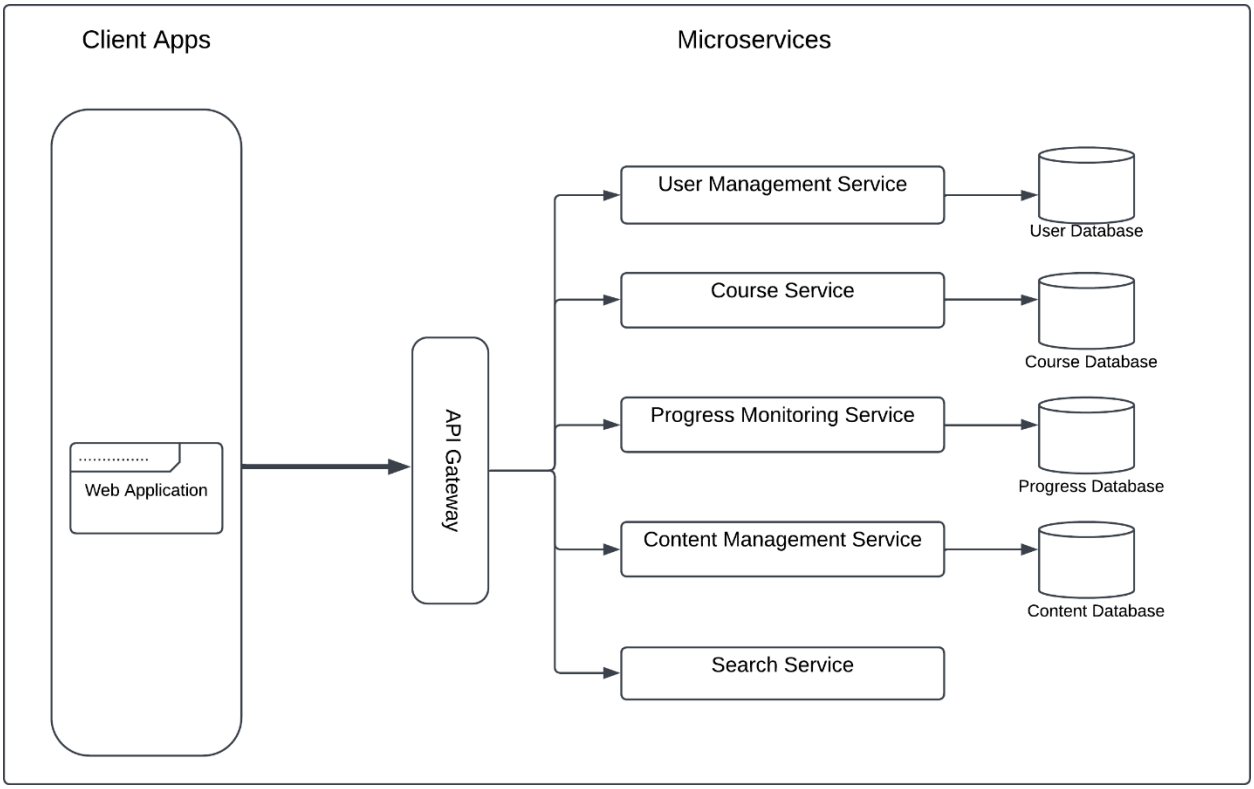
The Model-View-Template (MVT) is an architectural pattern similar to MVC but tailored for web development, especially in frameworks like Django. The acronym MVT stands for Model, View, and Template. Here too, the Model is the data layer of the application. The View is in fact the layer that undertakes the processing logic. The Template is the presentation layer.

This diagram illustrates the **Model-View-Template (MVT)** architecture for the **Young Minds Learners Hub** system.

- **The Model Layer** handles data management and business logic, including components like "User Model," "Course Model," and "Interactive Module Model." It interacts with the Database to store and retrieve information.
- **The View Layer** processes user requests and facilitates communication between the Model and Template layers. It includes views such as "Dashboard View," "Course View," and "Assignment View".
- **The Template Layer** is responsible for rendering user interfaces dynamically using components like "Dashboard Template," "Quiz Template," and "Assignment Template."

The system maintains a clear separation of concerns, ensuring scalability, maintainability, and user-friendly interactions.

Diagram 3.3 - Microservices Architecture



This diagram represents a microservices-based architecture designed to serve a web application. Description of its components:

- Client Apps

. Web Application: The primary interface through which users interact with the system. This application communicates with the backend via the API Gateway.

- API Gateway

. The API Gateway acts as a single-entry point for all client requests. It handles routing, authentication, and aggregation of responses from the microservices, simplifying the client-side interaction.

- Microservices

1- User Management Service:
   - Managed user-related functionality such as registration, authentication, and user profile updates.
   - Connected to the User Database, where all user information is stored.

2- Course Service:
   - Handles the management of course-related operations, such as course creation, and updates.
   - Utilizes the Course Database to store and retrieve course data.

3- Progress Monitoring Service:

- Tracks and provides insights into user progress, such as completed lessons or quizzes.
- Relies on the Progress Database for storing progress-related data.

4- Content Management Service:
- Manages the creation, storage, and retrieval of content for courses.
- Connected to the Content Database, which stores all course content.

5- Search Service:
- Provides search functionality for the application, allowing users to find specific courses, or content easily.

3.1.3 Architecture Selection Process

The architecture selection process is a systematic approach to identify and choose the most suitable architecture among various alternatives. This process ensures the selected architecture aligns with the project's objectives and requirements. Below are the steps undertaken in this process:

1. Define Evaluation Criteria

The process begins by defining key evaluation criteria to assess the suitability of each architecture. The criteria include:

- **Interoperability**: The ability of the architecture to interact seamlessly with other systems, regardless of platform or technology. This includes support for standard protocols (e.g., REST APIs, JSON) and ease of integration with third-party tools.
- **Availability**: The extent to which the system remains operational and accessible to users. This involves redundancy measures, minimized downtime during failures or updates, and disaster recovery strategies like database backups or distributed deployments.
- **Usability**: The ease with which users can interact with the system to accomplish their tasks efficiently. This considers intuitive user interfaces, reduced steps for common tasks, and adaptability to user feedback for improving the experience.
- **Modifiability**: The ease of updating or enhancing the architecture to meet changing requirements. This includes the ability to make modular changes without affecting unrelated components and flexibility to add new features or update existing ones.
- **Security**: The architecture's ability to protect system data and functionality from unauthorized access or attacks. This includes robust authentication and authorization mechanisms, secure data storage and transmission (e.g., encryption), and the ability to identify and address vulnerabilities.

2. Scoring and Ranking Architectures

Each alternative architecture is scored against the defined criteria using a standardized scale (e.g., 1 to 5, where 1 represents poor and 5 represents excellent). The steps include:

1. **Independent Evaluation**: Team members independently evaluate and score each architecture to minimize bias.
2. **Score Aggregation**: The scores are averaged to provide an objective measure of performance across all criteria.

Table 3.1 Evaluation Criteria

Design Alternative	Security	Modifiability	Usability	Availability	Interoperability	Average Score	Comments
Model-View-Template (MVT)	5	5	5	5	4	4.8	Highly clear and complete design; aligns with project needs. Easy to implement and scalable.
Multi-Tier Architecture	4	4	4	4	4	4.0	Moderate scalability and maintainability, but more complex than required for the project.
Microservices Architecture	3	4	3	4	5	3.8	Highly scalable but overly complex for the project's size and scope. Requires advanced expertise.

3. Selection of Architecture

Based on the scoring process, the **Model-View-Template (MVT)** architecture achieves the highest average score and is selected as the optimal choice for the **Young Minds Learners Hub** project.

4. Justification of Selection

The selected **MVT architecture** demonstrates superior performance in the following areas:

- **Clarity**: It provides a simple and easily understandable structure with clear separation of concerns.
- **Maintainability**: The distinct Model, View, and Template layers make updates and modifications straightforward.
- **Scalability**: The architecture supports moderate scalability, sufficient for the project's expected user base.
- **Efficiency**: The MVT pattern optimizes interactions between layers, reducing redundancy and improving performance.

- **Alignment with Project Goals**: Its simplicity and modularity align perfectly with the requirements of an educational platform like **Young Minds Learners Hub**.

5. Validation and Refinement

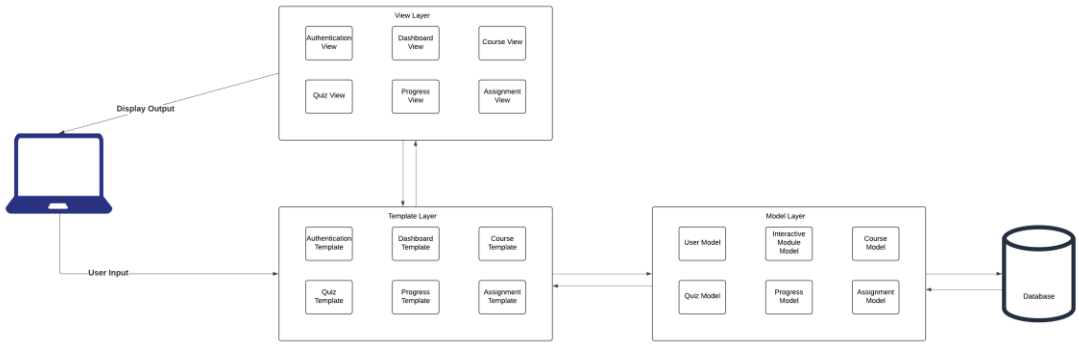
After selecting the **MVT architecture**, it undergoes a validation process to ensure it aligns with system requirements and stakeholder expectations. This includes:

- **Diagram Validation**: Ensuring the architectural diagrams comply with industry standards.
- **Testing for Gaps**: Identifying any potential issues or inefficiencies in the design.
- **Refinements**: Implementing necessary changes to address identified gaps and further enhance the architecture's performance.

By following this process, the **MVT architecture** is confirmed as the optimal choice for the project, balancing simplicity, scalability, and maintainability.

3.1.3.1 Final Architecture Diagram

Diagram 3.2 Overall Architecture (MVT)



3.1.4 Mapping System Design to Quality Attributes

The Young Minds Learners Hub system design aligns with critical quality attributes, ensuring it meets user and business needs effectively. The architecture prioritizes security by incorporating authentication mechanisms and encrypted data transmission to restrict access and protect sensitive information. It enhances usability through user-friendly templates and workflows, minimizing the steps required to complete common tasks and reducing error rates. The use of the MVT design pattern ensures modifiability, allowing new features to be added or existing ones to be updated without impacting unrelated components, thanks to the clear separation of concerns across layers. Availability is addressed through the multi-layered architecture and the integration of a cloud-based database with replication, ensuring minimal downtime and consistent access. Finally, the system achieves interoperability by adopting standardized data formats like APIs, enabling seamless communication with external systems, regardless of their platform or language. These attributes collectively ensure the system's reliability, scalability, and user satisfaction.

3.1.5 Architecture Evaluation

The architecture of the Young Minds Learners Hub was evaluated using the Tactics-Based Questionnaire. Key areas of evaluation include:

- **Security and Privacy:**
  Is sensitive data accessed only by authorized users? Yes, the system enforces authentication to restrict access, and data is encrypted both in transit and at rest.
- **Modifiability:**
  Can new features be added without affecting unrelated parts of the system? Yes, the layered MVT pattern allows for independent development and scaling of components, ensuring low coupling and high cohesion.
- **Usability:**
  Can users' complete system functions with a lower error rate? Yes, the MVT pattern optimizes workflows, grouping related functionalities in user-friendly templates.
- **Availability:**
  Is the system consistently accessible and operational? Yes, the multi-layered architecture, along with cloud database replication, ensures high system availability.
- **Interoperability:**
  Can the system interact with other systems? Yes, the use of JSON and adherence to standard APIs ensures seamless integration with external systems.

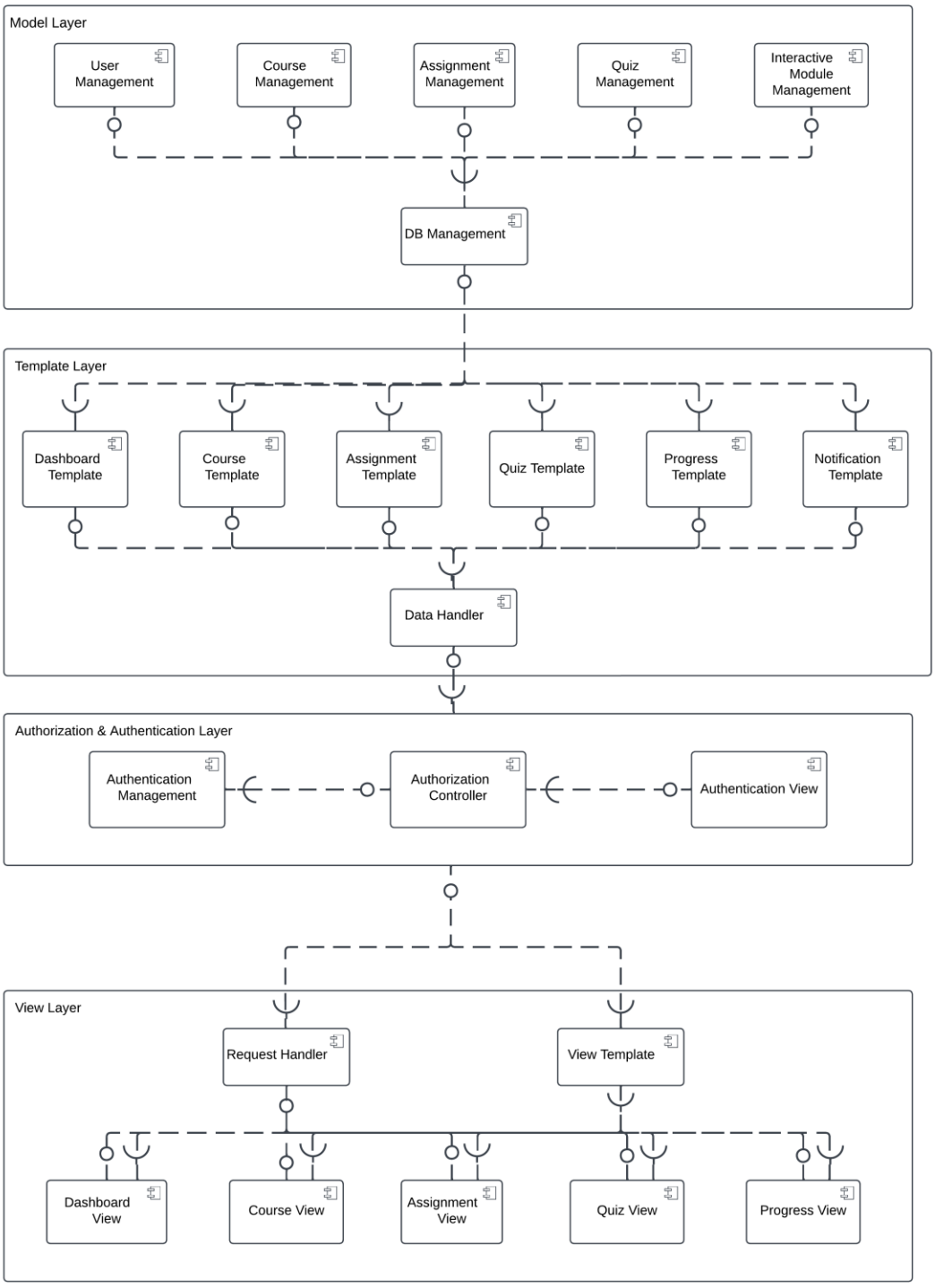
Based on these evaluations, the architecture successfully achieves the business goals of being secure, modifiable, usable, available, and interoperable.

3.2 Structural Diagram

3.2.1 Components Diagram

The system's overall architecture provides a high-level perspective, emphasizing its key components. To enhance understanding and dig deeper into the relationships between the components, we use component diagrams that break down the system into its basic building blocks (components) and put the magnifier on the relationships of the components to illustrate the dependencies they have.

Diagram 3.3 - Components Diagram



The Component Diagram for the Young Minds Learners Hub, as shown in Diagram 3.2, demonstrates a well-organized architecture that promotes loose coupling and scalability. The system incorporates the facade pattern, which simplifies interactions by introducing centralized components: the “ServicesHandler” for accessing services, the “RequestHandler” for managing

user interactions, and the “DB Management” for handling data storage. Each component was designed to either provide or consume specific functionalities, ensuring clear responsibilities and streamlined communication. For example, the Assignment View consumes data from the Assignment Template, which relies on the Assignment Management component in the Model Layer to fetch data. This layered approach ensures separation of concerns, making the system more maintainable and extensible. The diagram highlights the dependencies and relationships among components, ensuring that each layer operates independently while maintaining cohesion within the system. This architecture provides the foundation for a scalable and efficient platform tailored to meet user needs.

3.2.2 Component Diagram Validation

To validate the Component Diagram, three distinct perspectives were considered: Syntax, Semantics, and Aesthetic checks.

1. **Syntax Checks:**
   Ensured that component names are descriptive nouns, making them intuitive and easy to understand (e.g., "Assignment Management," "Quiz View").
   Verified that the direction of arrows accurately represents provider and consumer relationships (e.g., Template Layer components consuming data from Model Layer components).
2. **Semantics Checks:**
   Studied relationships between components to ensure consistency with the overall system architecture. For example, View Template components like "Quiz Template" retrieve data from their corresponding Model Layer components, such as "Quiz Management."
   Ensured that provider components (e.g., DB Management) supply data as expected, and consumer components (e.g., Request Handler) process and utilize this data correctly.
   Checked for meaningful arrangement of dependencies, maintaining a readable flow of relationships from bottom to top and right to left to avoid circular dependencies.
3. **Aesthetic Checks:**
   Introduced only necessary links between components to prevent unnecessary complexity while maintaining essential connections.
   Ensured a manageable number of components in the diagram to strike a balance between detail and clarity. For example, each layer (View, Template, Model) has a clear separation of concerns, reducing coupling while maintaining coherence.

By adhering to these validation steps, the Young Minds Learners Hub Component Diagram reflects a robust, scalable, and maintainable architecture. It effectively captures the interactions and dependencies among components, ensuring the system meets its functional and non-functional requirements.

3.2.3 Class Diagram

A class diagram serves as the foundation for the implementation phase in the SDLC, providing a detailed representation of the system's structure. It breaks down the system into classes, their attributes, operations, and relationships while ensuring modularity and clarity.

As shown in Diagram 3.2, the Young Minds Learners Hub system adopts core software engineering principles such as inheritance, encapsulation, and the single responsibility principle:

The User class serves as a base class for Instructor, Student, and Parent, implementing a clean inheritance hierarchy to promote code reusability.

Each subclass adds unique attributes and operations tailored to its role in the system. For example, the Instructor class can manage courses and assignments, while the Student and Parent classes focus on learning participation and progress tracking, respectively.

The design also showcases composition and aggregation:

The Course class aggregates other components like Assignment, Quiz, and InteractiveModule, representing the relationship between courses and their educational content.

The composition relationship is seen between Course and Assignment, where assignments cannot exist independently of the course.

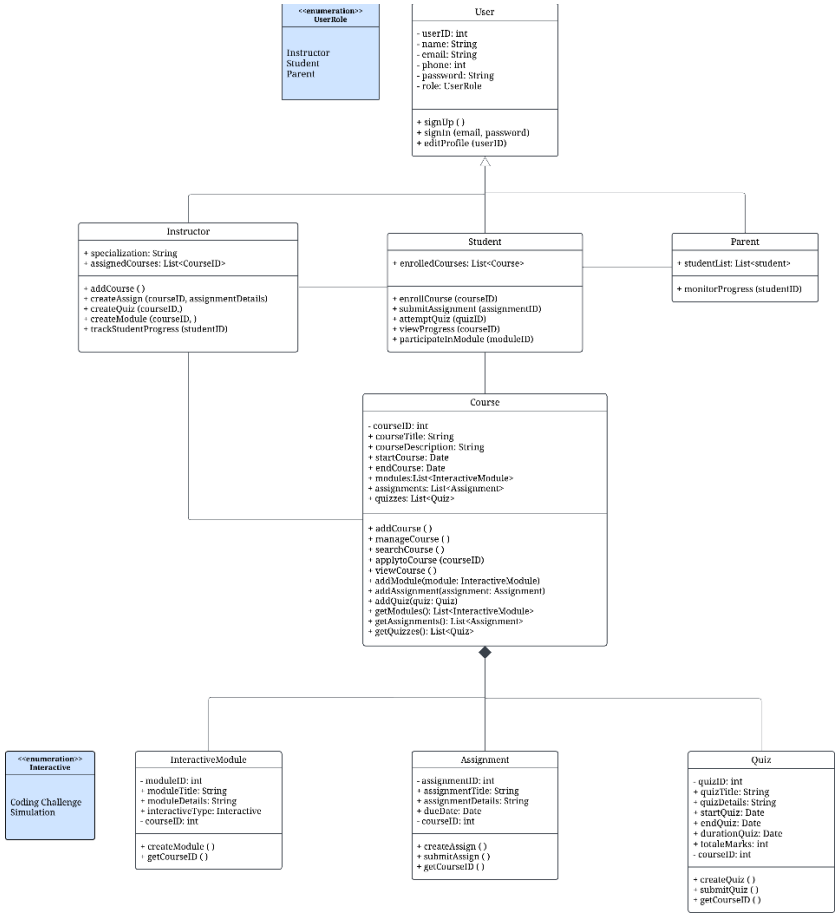
In terms of design patterns:

Loose Coupling: The design reduces dependencies between components, ensuring changes in one module minimally impact others.

Encapsulation: All attributes are private to protect data integrity, and operations are exposed to interact with data when necessary.

The class diagram reflects a modular, maintainable, and scalable architecture for the Young Minds Learners Hub system.

Diagram 3.4 - Class Diagram



3.2.3.1 Class Diagram Validation

To ensure the accuracy, clarity, and usability of the class diagram, the validation process covers Syntax, Semantics, and Aesthetics:

1. **Syntax Checks:**

   Class names are single nouns with an uppercase first letter.

   Attributes are named clearly with data types, and visibility modifiers (+ for public, - for private) are effectively used.

   Operations follow verb-based naming conventions and include necessary method signatures.

   Relationships are validated for correct notation: inheritance (solid line with an arrow), aggregation (hollow diamond), and composition (filled diamond).

2. **Semantics Checks:**

   Class names and attributes are meaningful, ensuring relevance to their purpose.

Encapsulation is strictly enforced by keeping attributes private and exposing operations where necessary.

Operations are scoped to maintain coherence within a single responsibility. Redundancies are avoided by ensuring methods are decomposed effectively.

Relationships (inheritance, aggregation, and composition) are semantically appropriate.

3. **Aesthetic Checks:**

   Classes are balanced to avoid overcrowding; attributes and methods are logically distributed.

   Getter and setter methods are included by default as a note for all classes to maintain clean structure.

   Operations and attributes are listed in decreasing order of visibility for readability.

   No abbreviations are used, ensuring clarity.

   Relationships and associations are carefully arranged to minimize visual clutter and improve diagram clarity.

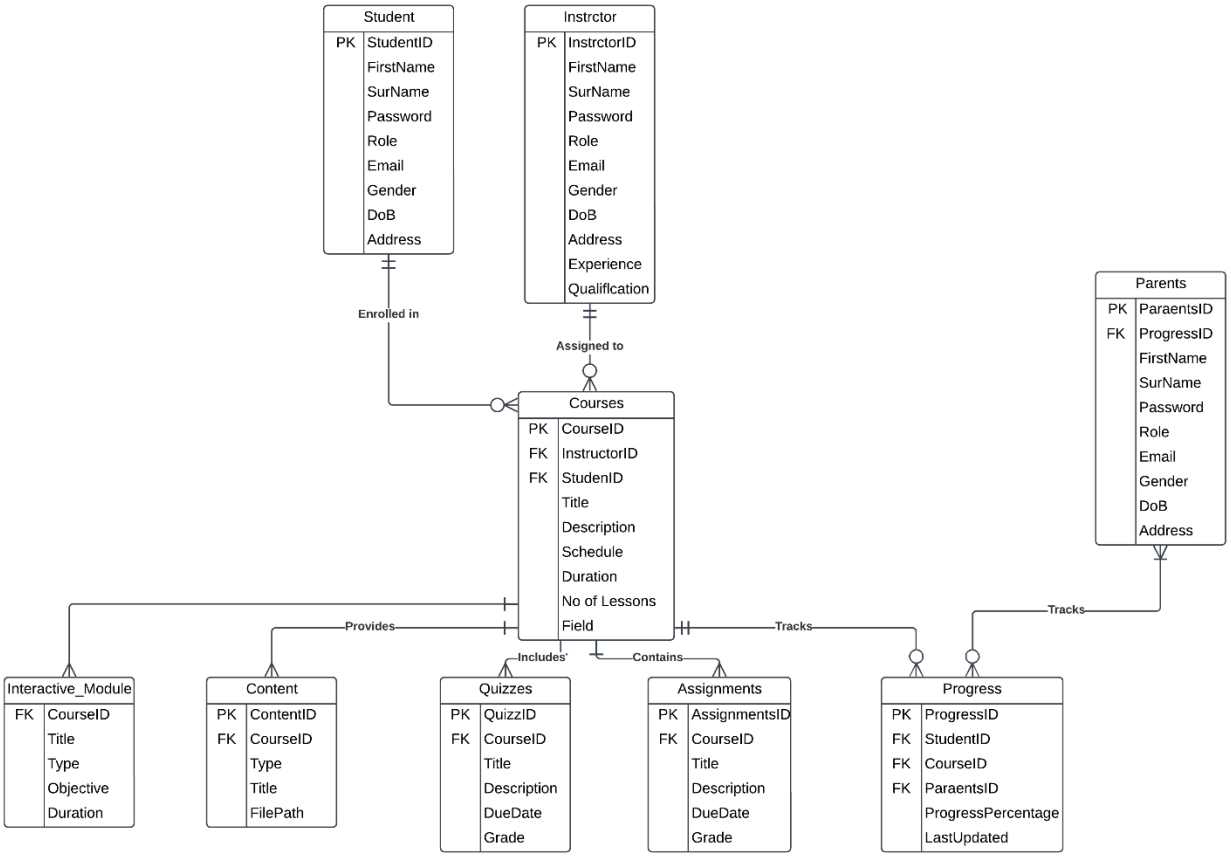
The class diagram is visually structured, adhering to best practices in software design, and effectively maps the system components into an organized hierarchy.

3.2.4 Database Design

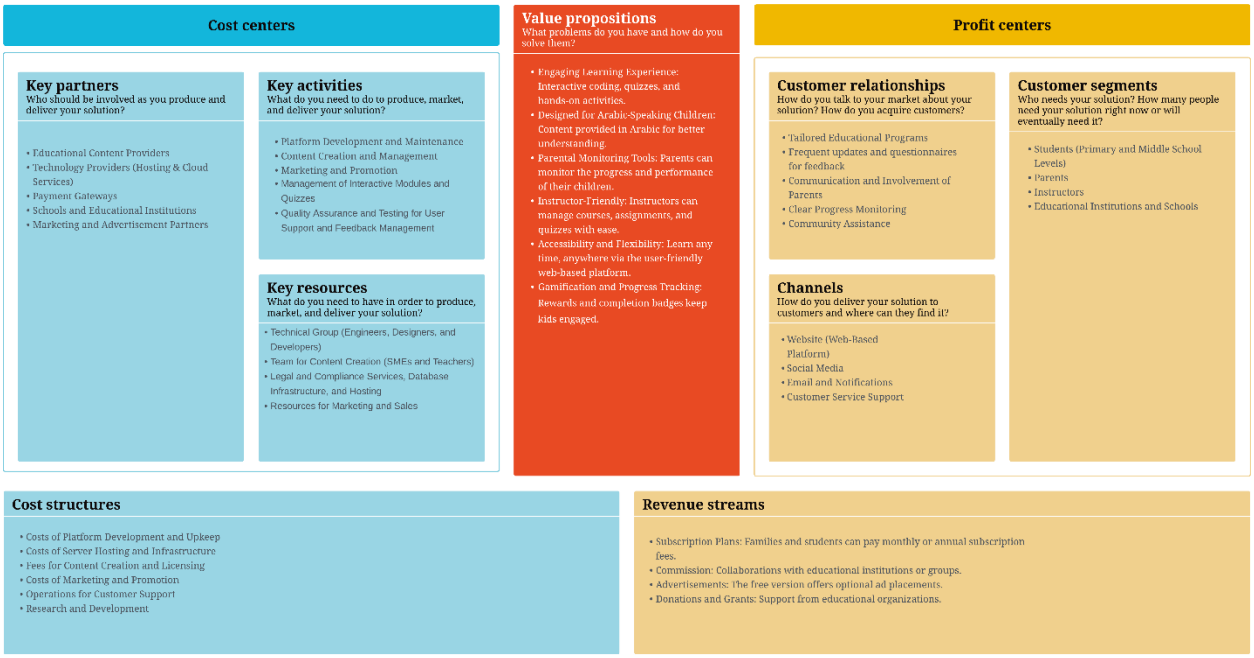
3.3.1 Entity Relation Diagram

The Entity Relationship Diagram (ERD) describes the structure of the database and such entities as students, instructors, courses, and interactive modules. It also prescribes how these entities relate with one another so that data is well structured and retrievable. Due to the capability to provide role-based access and monitor progress, the ERD promotes efficient educational processes. Thanks to its clear visualization, it is easier to develop and guarantees compliance with the requirements of the project. In general, the ERD guarantees scalability, as well as the usability of the platform and the optimal organization of data.

Diagram 3.5 - ER diagram



3.2.5 Business Model



This **Business Model Canvas** outlines the structure of the **Young Minds Learners Hub** system, designed to provide an engaging and accessible learning platform for children. It highlights the key components of the system to ensure sustainability, usability, and value delivery.

1. **Cost Centers**:
    - Focuses on the Key Partners, Activities, and Resources required to build and maintain the platform.
    - Includes educational content providers, technology partners, and marketing efforts, ensuring a high-quality user experience.
2. **Value Propositions**:
    - Defines the platform's core values, such as engaging learning experiences, parental monitoring, accessibility, and interactive tools like quizzes, coding activities, and gamification.
    - Tailored specifically for Arabic-speaking children to bridge learning gaps effectively.
3. **Profit Centers**:
    - **Customer Segments** include students, parents, instructors, and schools as key users.
    - **Customer Relationships** emphasize personalized plans, regular updates, and transparent progress tracking to maintain user trust and engagement.
    - **Channels** ensure effective delivery via a web platform, social media, email, and in-app messaging.

4. **Cost Structures**:
    - Highlights major expenses, such as platform development, hosting infrastructure, content creation, and marketing costs.
5. **Revenue Streams**:
    - Lists the income sources, including subscription plans, partnerships with schools, advertisements, and grants from educational organizations.

This model ensures a clear balance between cost efficiency and value creation, making the "Young Minds Learners Hub" a sustainable and scalable solution for interactive learning.

3.3 Ethical Standards

3.3.1 IEEE Code of Ethics

The IEEE Code of Ethics highlights essential ethical principles, which are directly applicable to the Young Minds Learners Hub (YML Hub) project:

1. **Public Safety and Well-being**: Ensure the platform is safe for children and protects their well-being (IEEE Code of Ethics, Clause 1).
2. **Honesty**: Represent the platform's capabilities and limitations truthfully (Clause 3).
3. **Data Privacy**: Implement measures to safeguard sensitive user data, including student progress and personal information (Clause 8).
4. **Fairness**: Ensure the platform is unbiased and accessible to all users, avoiding discrimination (Clause 10).
5. **Accountability**: Acknowledge and address mistakes or potential harm caused by the platform (Clause 4).
6. **Continuous Learning**: Regularly update skills and platform features to align with the latest standards and technologies (Clause 5).

3.3.2 ACM Code of Ethics and Professional Conduct

The ACM Code emphasizes:

1. **Contributing to Society and Well-being**: Develop educational tools that empower young learners and support societal progress (Section 1.1).
2. **Avoiding Harm**: Minimize risks to users, such as ensuring age-appropriate content and secure interactions (Section 1.2).
3. **Respecting Privacy**: Safeguard personal data in compliance with privacy standards (Section 1.6).
4. **Equity and Inclusivity**: Provide equal access and avoid bias in the platform's design (Section 1.4).
5. **Professional Competence**: Ensure the development team is proficient and follows ethical guidelines (Section 2.2).

3.3.3 Saudi Data & AI Authority (SDAIA)

The SDAIA's ethical guidelines for AI align with YML Hub's objectives:

1. **Transparency**: Clearly communicate how the platform collects and uses data (SDAIA Ethical Principles, Transparency).
2. **Fairness**: Eliminate algorithmic bias and ensure equal access to educational resources (Fairness).

3. **Safety and Security**: Adopt measures to prevent data breaches and unauthorized access (Safety and Security).
4. **Accountability**: Define clear accountability structures to address ethical concerns (Accountability).

3.3.4 Evaluation of Ethical Case Studies

3.3.4.1 Case Study 1: Data Privacy Breach in Educational Tools

Scenario: A platform similar to YML Hub suffered a data breach, exposing personal details of children and educators. **Ethical Conflict**: Breach of data privacy, violating GDPR and COPPA standards.

Resolution:

- Enhanced data encryption and multi-factor authentication.
- Regular third-party security audits.
- User notifications and resources for mitigating risks.

3.3.4.2 Case Study 2: Algorithmic Bias in Learning Systems

Scenario: An educational platform unintentionally favored students from certain demographics due to biased training data. **Ethical Conflict**: Violated equity and inclusivity principles.

Resolution:

- Expanded dataset to include diverse demographics.
- Implemented fairness metrics to detect and address bias.
- Conducted regular reviews to ensure equitable outcomes.

3.3.5 Reflection on Personal Behavior and Decision-making

To align with professional ethics while working on YML Hub:

1. **Adhering to Ethical Standards**: Regularly review IEEE, ACM, and SDAIA guidelines.
2. **Transparent Communication**: Maintain clear documentation on platform features and risks.
3. **Stakeholder Involvement**: Actively include feedback from students, parents, and educators.
4. **Continuous Learning**: Invest in training and workshops on ethical AI and educational tools.

3.3.6 Relevant Regulatory Laws and Guidelines for Engineering Projects

1. **General Data Protection Regulation (GDPR)**:
    - Mandates user consent for data collection and processing (Art. 7).
    - Ensures protection of personal data (Art. 5). **Reference**: GDPR Full Text.
2. **Children's Online Privacy Protection Act (COPPA)**:
    - Requires parental consent for collecting data from children under 13. **Reference**: COPPA Guidelines.
3. **Saudi Personal Data Protection Law**:
    - Governs data collection and processing within Saudi Arabia. **Reference**: Saudi Data & AI Authority.
4. **ISO/IEC 27001**:
    - Framework for managing information security. **Reference**: ISO Standards.
5. **SDAIA Ethical AI Guidelines**:
    - Promote fairness, transparency, and accountability in AI systems. **Reference**: SDAIA Guidelines.

CHAPTER 4 – CONCLUSION

The Young Minds Learners Hub (YML Hub) platform is designed to fill the gap in modern educational technology for Arabic speaking young learners through a novel, scalable and user-friendly platform that bridges critical gaps with multilingual support and advanced user management tools. The platform is developed using the systematic development approach from requirement gathering to analysis, design and implementation.

A comprehensive architecture based on the Model-View-Template (MVT) pattern is used to provide modularity, security and performance, while meeting functional and non-functional requirements. Key deliverables from the project include detailed UML diagrams, a robust relational database design and a dynamic web application that are intended to produce a complete and reliable educational solution.

The YML Hub offers a solid base for future extensions (additional languages, features and technologies) to ensure future adaptability and long-term impact. Therefore, this project contributes to building a more inclusive and interactive learning environment for kids, for teachers and for their parents and it is important in the field of educational technology.

Reference

Van Lamsweerde, A. (2000, June). Requirements engineering in the year 00: A research perspective. In *Proceedings of the 22nd international conference on Software engineering* (pp. 5-19). https://dl.acm.org/doi/abs/10.1145/337180.337184

Macaulay, L. A. (2012). *Requirements engineering*. Springer Science & Business Media. https://books.google.com/books?hl=ar&lr=&id=RojbBwAAQBAJ&oi=fnd&pg=PA1&dq=requirements+engineering&ots=CiqUIMN_N9&sig=6Z6XuzJDENKbwqhwm1fSuFY1FLo

Sommerville, I. (2015, March 24). Software Engineering. Addison-Wesley. Software Engineering - Ian Sommerville - Google Books

Kneuper, R. (2018). Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes. Springer Cham. ISBN 978-3-319-98844-3.

Kaplancali, U. T., & Demirkol, Z. (2017). Teaching coding to children: A methodology for kids 5+. *International Journal of Elementary Education*, *6*(4), 32-37. https://www.academia.edu/download/88563707/10.11648.j.ijeedu.20170604.11.pdf

Richards, M. (2015). Software architecture patterns (Vol. 4, p. 1005). 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Incorporated. https://isip.piconepress.com/courses/temple/ece_1111/resources/articles/20211201_software_architecture_patterns.pdf

Harrington, J. L. (2016). Relational database design and implementation. Morgan Kaufmann. https://books.google.com.sa/books?hl=en&lr=&id=yQgfCgAAQBAJ&oi=fnd&pg=PP1&dq=database+design&ots=qQEuqYPG2v&sig=gi7JTuheJxwzvq-PfEmrQeVxreE&redir_esc=y#v=onepage&q=database%20design&f=false

von der Maßen, T., & Lichter, H. (2002, September). Modeling variability by UML use case diagrams. In Proceedings of the International Workshop on Requirements Engineering for product lines (pp. 19-25). Citeseer. https://d1wqtxts1xzle7.cloudfront.net/73055482/REPL02_International_Workshop_on_Requir20211018-1298-1fw0rne.pdf?1634569672=&response-content-disposition=inline%3B+filename%3DREPL02_International_Workshop_on_Require.pdf&Expires=1734730331&Signature=Ux63oVjrevXMym2JwVSe-dwQ2VDZMRMTTDrg~LEEBjxDt0XZmCtoRftMGL6YUYKFAMXOXeh0A2EpRu2KEJX6cHY6oYBBKNtJX8ArTU7mt6siToljqvTnrLz0Yd9jkNmzUX~uuhkgzdsdHJIEJCI6cbBcvG7zH-qRj4ZpLyDvkVNXpUEkYrY~8ymqPgzikyAiQIfpB3kRre1Ke7dlLXE~6L-5tqtfJ87NRWClN0LxJ23iQ~VmdShyuCsqT1ths92tu2iz0Vbsw27oRclSkqkIy7cGTZ85wre1U4KaWLVBSzIsfQ7CktVmB4sc6YGEwSOC1wp~aLV9w0h-9kvQN16F~g__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA#page=25

Pirsig, R. (2014). From Chapter of Object-Oriented Software Engineering Using UML, Patterns, and Java, Bernd Bruegge, Allen H. Dutoit. Copyright© 2010 by Pearson Education, Inc. All rights reserved. Object-Oriented Software Engineering Bruegge Dutoit 3e, 481. https://opac.atmaluhur.ac.id/uploaded_files/temporary/DigitalCollection/Mjk2MDk4MjU3MzA0YTczNjczYTJhNmRiYWM5MDUzYjM1Nzk3Y2NmNg==.pdf#page=486

Brown, A. (2017). Fundamentals of Project Management. XYZ Publishers .

White, P. (2018). Collaborative Project Management. JKL Publications

Laplante, P. A. (2018). Requirements Engineering for Software and Systems (3rd ed.). CRC Press Taylor & Francis Group, LLC.

Specification, O. A. (2007). Omg unified modeling language (omg uml), superstructure, v2. 1.2. Object Management Group, 70, 16.
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=eb2f430d1fd600cb5adeaf1a64f95adf8a8e7198

GDPR Full Text. (n.d.). Retrieved from
https://gdpr-info.eu/

COPPA Guidelines. (n.d.). Retrieved from
https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa

Saudi Data & AI Authority. (n.d.). Retrieved from
https://sdaia.gov.sa/en/

ISO/IEC 27001: Information Security. (n.d.). Retrieved
https://www.iso.org/isoiec-27001-information-security.html

SDAIA Ethical AI Guidelines. (n.d.). Retrieved from
https://sdaia.gov.sa/en/ai/ethics/

IEEE Code of Ethics. (n.d.). Retrieved from
https://www.ieee.org/about/corporate/governance/p7-8.html

ACM Code of Ethics and Professional Conduct. (n.d.). Retrieved from
https://www.acm.org/code-of-ethics

Case Study 1 References
- European Union's General Data Protection Regulation (GDPR) (Art. 5).
- Children's Online Privacy Protection Act (COPPA), Section 312.5.

Case Study 2 References
- ACM Code of Ethics, Section 1.4.
- IEEE Code of Ethics, Clause 10.

align with professional ethics while working on YML Hub References
- ACM Code of Ethics, Section 1.4.
- IEEE Code of Ethics, Clause 10.

Appendix

(New knowledge)

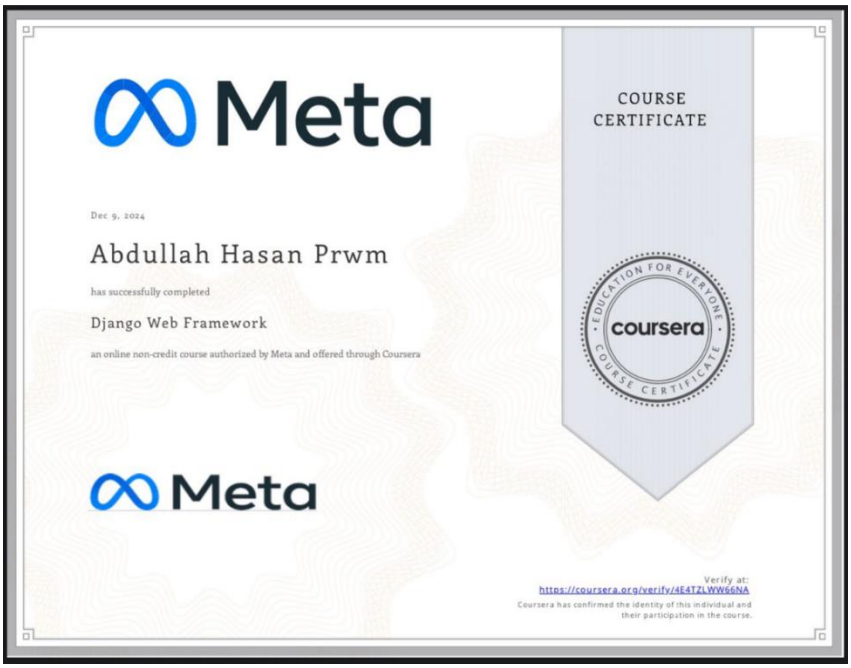
1. **Identifying Knowledge Gaps:**
   Early in the project, it became clear that advanced knowledge in Django was essential for the successful development of our web application’s backend.

2. **Utilizing Credible Resources for Learning:**
   To address this, we enrolled in and completed a detailed online course on Django, which equipped us with a thorough understanding of its key features and architecture. This education was certified by COURSERA, solidifying our proficiency, as we hadn't taken something like Django in our formal studies.

3. **Planning Application of Knowledge:**
   Though We have not yet applied Django in our project, the skills we have developed are crucial for future phases, ensuring that we can enhance the application's backend when implementation begins.

Certificates

- Abdullah Hasan Prwm



- Hussain Timah

- Khaled Salem

- Khalid Namat Allah