Team 4

SKILLSBERG ELMS
Software Architecture & Detailed Design Document
(SADD)

Team members:

Bhanu Teja Panguluri

Revanth Maturu

Sai Pranathi Vadrevu

Sudhanshu Dubey

Tarang Nair

INDEX

1. Product Overview

With Skillsberg, learners can acquire new languages efficiently and effectively using a user-friendly platform. It is an ELMS designed to facilitate language learning and fluency development. As part of Skillsberg's architecture, a variety of features and components are included, including course management, content delivery, assessment, and user interaction, all aimed at supporting systematic language acquisition. The system's primary purpose is to provide an immersive and engaging language learning experience, helping users acquire proficiency in various languages through a structured curriculum, interactive activities, and comprehensive tracking of their progress.

2. Architecture Drivers

The following are the high-priority functional and quality features that will be critical in shaping the architecture of the Skillsberg ELMS project, ensuring a user-centric, reliable, and adaptable language learning platform.

Functional Features:

1. **User Registration & Logging In:** Implementing a seamless and secure user registration and login system, allowing users to create accounts and access the platform effortlessly.

2. **Enrolling and Accessing Courses:** Developing a feature that enables users to browse, select, and enroll in language courses with ease. Users will be able to access their enrolled courses conveniently.

3. **Taking Quizzes and Assessments:** Creating a robust assessment system that allows users to take quizzes and assessments related to the language courses they are enrolled in. Ensuring a smooth and engaging experience for learners.

4. **Providing Support Assistance:** Implementing a support system that includes features like a knowledge base(FAQs), and Message Box support to assist users with any questions or issues they encounter during their learning journey.

5. **User Management:** Develop comprehensive user management functionalities, including user profiles, progress tracking, and user role management, ensuring a personalized and effective learning experience.

High-Priority Quality Attributes [Utility Tree]:

Quality attribute	Quality Scenario name	Quality Scenario brief description
RELIABILITY	Low Failure Rate	Ensure a high level of system reliability with low failure rates, minimizing system crashes, downtime, and errors that could disrupt users' language learning activities.
USABILITY	User-Friendly Interface	Prioritizing the development of an intuitive and user-friendly interface, making it easy for users to navigate the platform, access content, and track their language learning progress seamlessly.
MODIFIABILITY	Modular System Design	Implementing a modular system design that allows for the flexible addition of new languages, courses, features, and enhancements to adapt to evolving user needs and business requirements.
AVAILABILITY	High Availability of Content	Focus on providing a robust and highly available content delivery system, ensuring that language courses and learning materials are accessible to users at all times.
MAINTAINABILITY	Documentation and Knowledge Transfer	Emphasize the creation of comprehensive documentation for the system's architecture, codebase, and best practices to facilitate knowledge transfer among development teams and ensure the long-term maintainability of the platform.

3. Architecture Key Decisions and Rationale

Conducting a trade-off analysis was crucial in our decision-making process, allowing us to align architectural choices with our platform's goals and requirements. This analysis helped us evaluate the trade-offs, risks, and implications of technology and design options, ensuring our platform was scalable, secure, and user-centric. It provided a

structured and transparent approach to optimize resource allocation and address challenges, ultimately delivering a high-quality language learning experience.

The Trade-Off Analysis is as follows :

STIMULI: Events that cause the architecture to react to change

1. **Increased User Load:** As user registration grows, the system must efficiently handle a larger number of concurrent users.
2. **Frequent Content Updates:** Frequent updates and additions to language courses and learning materials.
3. **Expansion of Supported Languages:** The need to support new languages as the platform scales.
4. **Security Concerns:** Ongoing security threats and the need to protect user data and content.
5. **Rapid Feature Development:** The requirement to introduce new features and improvements quickly.

RESPONSES: Measurable/observable reactions to stimuli

1. **Scalability:** The system should be able to scale horizontally to handle increased user load.
2. **Content Delivery:** Content updates must be delivered efficiently to users.
3. **Language Integration:** Adding new languages should be a straightforward process.
4. **Security:** Strong security measures should be in place to protect user data.
5. **Development Speed:** The development team should be able to work efficiently.

DECISIONS: Those aspects that directly impact achieving quality attributes

1. **Database Management System (DBMS)**

   Candidates: MySQL, PostgreSQL, NoSQL (e.g., MongoDB, Firebase)
   Decision: Firebase
   Advantages:
   1. Firebase is a real-time database that allows us to sync data across all clients in real-time.
   2. It helps developers to set up the backend services quickly thus accelerating the development process.
   3. Firebase offers a hosting environment and allows developers to easily interact with the serverless cloud functions to develop the backend services

   Disadvantages:
   1. Firebase does not support complex queries as compared to relational databases

2. The cost of operation can significantly rise on high level of usage

2. **Frontend Framework**

   Candidates: React.js, Angular
   Decision: React.js
   Advantages: React offers greater flexibility because it's a library, not a full framework, allowing you to choose and customize tools as needed for your project.
   Disadvantages: React has a steeper learning curve due to its fragmented ecosystem and the need to make more tooling decisions compared to Angular's more opinionated approach.

3. **Backend Framework**

   Candidates: Node JS, Python
   Decision: Node JS
   Advantages: Node JS is lightweight, highly scalable, and well-suited for real-time applications.
   Disadvantages: JavaScript may not be as performant for CPU-intensive tasks.

4. **Cloud Provider**
   Candidates: GCP, AWS, Azure
   Decision: GCP
   Advantages:
   1. GCP doesn't have a steep learning curve, making it easier to use
   2. GCP provides a lot of pre-build features like Firebase real time Database and big Query for data analytics which allows the developer to set up the backend services effortlessly.

   Disadvantages: Smaller market share and ecosystem compared to AWS and Azure, potentially leading to fewer readily available resources and third-party integrations.

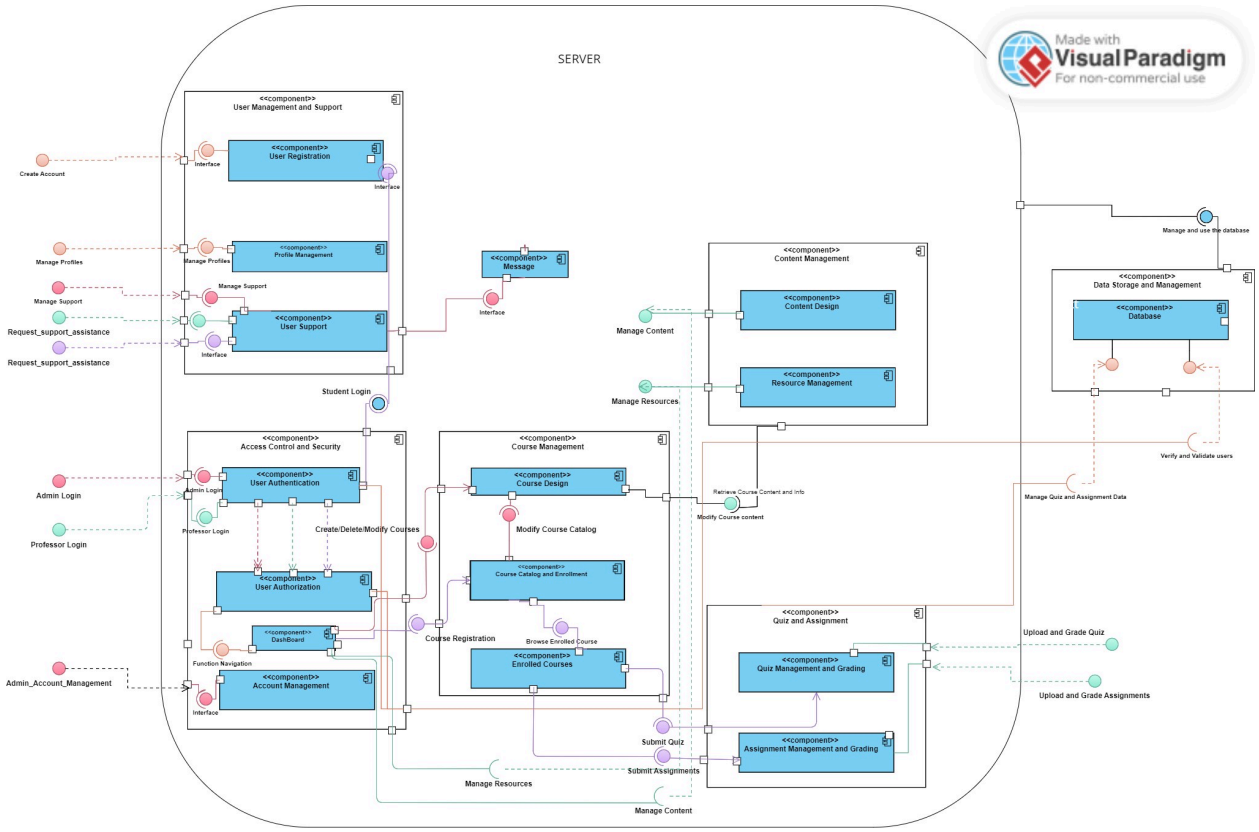
5. **Authentication and Security**

   Candidates: Username and Password, API Key, OAuth 2.0
   Decision: Username and Password
   Advantages: Username and password authentication is familiar and user-friendly, as most users are accustomed to this method, making it easy to implement and use.
   Disadvantages: Passwords can be vulnerable to breaches if not properly managed and they alone may lack the fine-grained access control features provided by more complex methods like OAuth 2.0.

6. **Content Management System (CMS)**

Candidates: Custom-built CMS, Open-source CMS (e.g., WordPress)
Decision: Custom-built CMS
Advantages: Custom-built CMS allows for tailored content management and integration with the ELMS, offering more flexibility.
Disadvantages: Requires more development effort initially, but offers long-term advantages in content customization.

4. Architecture Logical View



Component Name: User Registration
Component Responsibilities:
The User Registration component is responsible for managing the process of registering new users within the system. Its primary responsibilities include:

1. Storing user registration information: It securely stores the user's registration data in the system's database, allowing for easy retrieval and management.

2. Generating user profiles: Upon successful registration, the component generates user profiles, associating them with the "student" role and providing a unique identifier for each new student user.

Provided Interfaces:

The User Registration component provides the following interfaces:

1. Create Account: This interface allows new users to create their accounts by providing the required registration details.
2. Account Management: It provides an interface for administrators to manage user accounts, including updating user information and deactivating accounts if necessary.

Required Interfaces:

The User Registration component requires the following interfaces to fulfill its responsibilities:

1. Database Access: To store and retrieve user registration information, the component relies on the database access interface for data manipulation.

Component Name: Profile Management

Component Responsibilities:

The Profile Management component is responsible for overseeing the user profile management functionality within the system. Its primary responsibilities include:

User Profile Creation and Editing: This component enables users, including students, teachers, and administrators, to create their profiles within the system. Users can input and update their personal information, preferences, and other relevant details.

Provided Interfaces:

The Profile Management component provides the following interface:

Account Management: It provides an interface for administrators to manage user accounts, including updating user information and deactivating accounts if necessary.

Required Interfaces:

The Profile Management component requires the following interface to fulfill its responsibilities:

Manage Profile: This interface allows users with administrative privileges to manage user profiles. users can perform actions such as, editing, deleting profiles for students, teachers, and other users.

Component Name: User Support

Component Responsibilities: The User Support component provides the following interfaces:

Message Interface: This interface enables users (students and teachers) to send messages for requesting support, reporting issues, or seeking assistance.

Provided Interfaces: Message Handling: This component manages and facilitates the exchange of messages between users. Users can use this feature to seek assistance, report issues, or communicate with the system's support team.

Required Interfaces: The User Support component requires the following interfaces to fulfill its responsibilities:

1. Request Support Assistance: This interface is required for students and teachers to initiate support requests. It allows them to submit their support inquiries and issues to the User Support component.
2. Manage Support: This interface is required for administrators to manage support requests. It includes features for reviewing, assigning, and resolving support requests received from students and teachers.

Component Name: User Authentication and Authorization

Component Responsibilities: The User Authentication and authorization components are responsible for ensuring secure and authorized access to the system for different user roles, including students, teachers, and administrators. Its primary responsibilities include:

1. User Authentication: This component verifies the identity of users attempting to log into the system. It authenticates user credentials to ensure that only authorized individuals gain access.
2. Role-Based Access Control: User Authentication enforces role-based access control, ensuring that users are granted appropriate privileges based on their roles as students, teachers, or administrators.

Provided Interfaces: The User Authentication component provides the following interfaces: None: As of this description, the component does not provide external interfaces to other components. Instead, it is primarily responsible for user authentication and access control within the system.

Required Interfaces: The User Authentication component requires the following interfaces to fulfill its responsibilities for different user roles:

1. Admin Login: This interface is required for administrators to log into the system securely, providing access to admin-specific features and functionalities.
2. Professor Login: This interface is required for teachers (professors) to authenticate their identity and access teacher-specific tools and resources.
3. Student Login: This interface is required for students to log in securely and access student-specific features and educational content.

Component Name: Dashboard

Component Responsibilities: The Dashboard component serves as the central hub for users within the system, offering a user-friendly interface and access to various functionalities

Provided Interfaces: User Access Control: The Dashboard component ensures that users can access functionalities appropriate for their roles. It enforces role-based access control for students, teachers, and administrators.

1. Overview and Navigation: It provides users with an overview of their current status, including information about enrolled courses, recent activity, and notifications. Users can navigate to different areas of the system easily.
2. Course Management: The Dashboard enables teachers to manage courses they are responsible for, allowing them to add, edit, or remove course content, resources, and assignments.
3. Content Management: Users can manage educational content, including creating, updating, and organizing course materials within the system.
4. Resource Management: This component facilitates the management of resources, such as documents, multimedia files, and reference materials, for educational purposes.
5. Course Registration: Students can use the Dashboard to search for and enroll in courses, ensuring a streamlined course registration process.
6. Enrolled Courses: It provides students with a list of their currently enrolled courses, making it easy to access course materials and assignments.
7. Account Management: Users can update their account information, such as email addresses and profile details, using the Dashboard.

Required Interfaces:The Dashboard component requires the following interfaces to ensure role-based access control:

1. Admin Login: This interface is necessary for administrators to log into the system securely, granting them access to system administration features.
2. Professor Login: It is required for teachers (professors) to authenticate their identity and access teaching-related functionalities and course management tools.
3. Student Login: This interface is necessary for students to securely log in and access their educational materials, course registrations, and account management.

Component Name: Account Management

Component Responsibilities: User Account Creation: This component allows administrators to create new user accounts within the system, assigning roles, access levels, and account details as needed.

1. Account Information Update: Administrators can use this component to modify user account information, including personal details, contact information, and role assignments.
2. Account Deactivation and Reactivation: It provides the functionality to deactivate and reactivate user accounts, allowing administrators to manage account status as needed, such as suspending or reinstating access.
3. Password Management: Account Management includes tools for administrators to manage user passwords, including resets, changes, and security policy enforcement.
4. Role Assignment: Administrators can assign and modify user roles and permissions within the system, ensuring that users have appropriate access and privileges.

Provided Interfaces: Account Management component does not directly provide external interfaces to other components. Instead, it offers a user interface and functionalities accessible to administrators within the system to perform the above account management tasks.

Required Interfaces: The Account Management component requires the following interface to ensure secure administrator account management:
Admin Account Management Interface:

- Allows administrators to log in securely, granting them access to account management tools.
- Provides the necessary functions and permissions for administrators to create, update, and manage user accounts.

Component Name: Message
Component Responsibilities: The Message Management component is responsible for handling the communication and messaging features within the system. Its primary responsibilities include:

1. Message Creation: This component allows users to create and send messages to other users or groups within the system. Users can compose text-based messages, attach files, and set message preferences.
2. Message Delivery: It ensures the reliable and timely delivery of messages to the intended recipients, whether they are individual users, groups, or specific roles (e.g., students, teachers, administrators).

Provided Interfaces: Create Message Interface: This interface allows users to compose, send, and manage messages, including text, attachments, and message settings.

Required Interfaces: The Message Management component does not have required external interfaces; instead, it interacts with internal system components and user accounts to ensure seamless message communication.

Component Name: Course Design
Component Responsibilities: The Course Design component is responsible for the creation, organization, and maintenance of courses within the system. Its primary responsibilities include:

1. Course Creation: This component enables course designers and administrators to create new courses within the system, defining course details, objectives, and prerequisites.
2. Course Catalog Modification: It provides tools to modify the course catalog, including adding new courses, updating existing courses, or removing courses that are no longer offered.
3. Course Content Design: Course Design allows for the development of course content, including lectures, assignments, assessments, and multimedia materials.

4. Curriculum Organization: The component assists in organizing course content into coherent and structured curricula, ensuring that course materials align with learning objectives.
5. Course Management: It provides tools for the management of course offerings, scheduling, and access control, including setting enrollment limits and access permissions.
6. Collaboration Tools: Course designers can use this component to integrate collaborative features such as discussion boards, forums, and group activities into the course design.

Provided Interfaces: The Course Design component provides the following interface:
Modify Course Catalog Interface: This interface allows administrators and course designers to make changes to the course catalog, including creating new courses, updating existing course information, and removing courses when necessary.

Required Interfaces: The Course Design component requires the following interfaces to ensure comprehensive course creation and management:

1. Modify Course Content Interface: This interface enables course designers to create, edit, and organize the content of individual courses, including lectures, assignments, assessments, and resources.
2. Manage Course Interface: It provides tools for administrators to manage course offerings, schedule courses, and control access to courses, ensuring proper course administration.

Component Name: Message

Component Responsibilities: The Course Catalog and Enrollment component play a crucial role in the system's course management and student enrollment processes. Its primary responsibilities include:

1. Course Catalog Presentation: This component presents a user-friendly and organized course catalog, allowing students to browse available courses and view essential details, such as course descriptions, schedules, and prerequisites.
2. Course Availability: It ensures that the course catalog is up-to-date, displaying only the courses that are currently available for enrollment and removing courses that are no longer offered.
3. Course Enrollment: The component provides the functionality for students to browse, select, and enroll in courses from the catalog, streamlining the enrollment process.

Provided Interfaces: The Course Catalog and Enrollment component provides the following interface:

Browse Enrolled Courses Interface: This interface allows students to view and access the courses they are currently enrolled in, providing quick access to course materials and information.

Required Interfaces: The Course Catalog and Enrollment component requires the following interface to facilitate course selection and enrollment:

Course Registration Interface: This interface is essential for students to browse the course catalog, select courses, and complete the enrollment process.

Component Name: Enrolled Courses

Component Responsibilities: The Enrolled Courses component focuses on providing enrolled students with the necessary tools and features to access and interact with the courses in which they are currently enrolled. Its primary responsibilities include:

1. Course Access: This component grants students access to the courses in which they are enrolled, allowing them to view course materials, lectures, and assignments.
2. Navigation and User Interface: It offers a user-friendly interface for students to easily navigate within their enrolled courses, access course content, and engage in course activities.
3. Submission of Quizzes: The component enables students to submit quizzes and assessments as part of their course requirements, facilitating timely submission and evaluation.
4. Submission of Assignments: It provides the capability for students to submit assignments, projects, and other coursework within the enrolled courses.
5. Progress Tracking: The component allows students to track their progress within each course, including their completed assignments, quiz scores, and overall performance.
6. Communication Tools: Enrolled Courses may include communication features, such as discussion boards or forums, to facilitate interactions between students and instructors.

Provided Interfaces: The Enrolled Courses component provides the following interfaces:

1. Submit Quiz Interface: This interface allows students to submit quizzes and assessments within the courses they are enrolled in, ensuring that their responses are recorded and evaluated.
2. Submit Assignments Interface: Students can use this interface to submit assignments, projects, and coursework within their enrolled courses, providing an efficient means for assessment and grading.

Required Interfaces: The Enrolled Courses component requires the following interface to enable students to access and interact with their enrolled courses:

Browse Enrolled Courses Interface: This interface is essential for students to access their enrolled courses, view course content, and engage in course activities.

Course Registration: The Course Registration interface handles the registration process for students. It enables students to browse available courses, select the ones they wish to enroll in, and complete the registration.

Required Interface:

Course Content and Info: The Course Management component relies on the Course Content and Info interface to access detailed information about courses. This interface provides access to course materials, syllabi, and other content necessary for managing courses and registrations.

Component Name: Quiz and Assignment Management and Grading

Component Responsibilities:

The Quiz and Assignment Management and Grading component is dedicated to the creation, organization, management, and evaluation of quizzes, assignments, and assessments within the system. Its primary responsibilities include:

1. Quiz and Assignment Creation: This component allows educators to create quizzes, assignments, and assessments, specifying questions, instructions, deadlines, and grading criteria.
2. Content Organization: It helps educators organize quizzes and assignments, allowing for categorization and alignment with specific courses and learning objectives.
3. Student Submission Handling: The component provides tools for students to submit their responses to quizzes and assignments, ensuring secure and efficient submission.
4. Grading and Assessment: Educators can use this component to evaluate and grade student submissions, providing feedback and tracking student performance.

Provided Interfaces:

The Quiz and Assignment Management and Grading component provide the following interfaces:

Manage Quiz : This interface enables educators to create quizzes and grade them which includes specifying questions, instructions, deadlines, and grading criteria.

Manage assignment: Educators use this interface to assess and grade student assignments submissions, providing feedback and recording scores.

Required Interfaces:

The Quiz and Assignment Management and Grading component interacts with the following interfaces and components to facilitate quiz and assignment management, submission handling, and grading:

1. Enrolled Courses Interface: Enrolled Courses present quizzes and assignments to students within their enrolled courses and handle the submission process.
2. Content Design Interface: Content Design may include quizzes and assignments in the course content and curriculum, and this component ensures they are accessible and manageable.

Component Name: Database Management
Component Responsibilities: The Database Management component is responsible for the management and organization of data within the system. Its primary responsibilities include:

1. Data Storage: This component stores and manages the system's data, including user information, course content, user interactions, and system configuration.
2. Data Retrieval: It enables efficient retrieval of data for various system components, ensuring that information is accessible when needed.
3. Data Integrity: Database Management maintains the integrity and consistency of data, enforcing data validation rules and ensuring that it meets predefined standards.
4. Data Security: The component incorporates security measures to protect sensitive data, including user credentials, personal information, and confidential records.

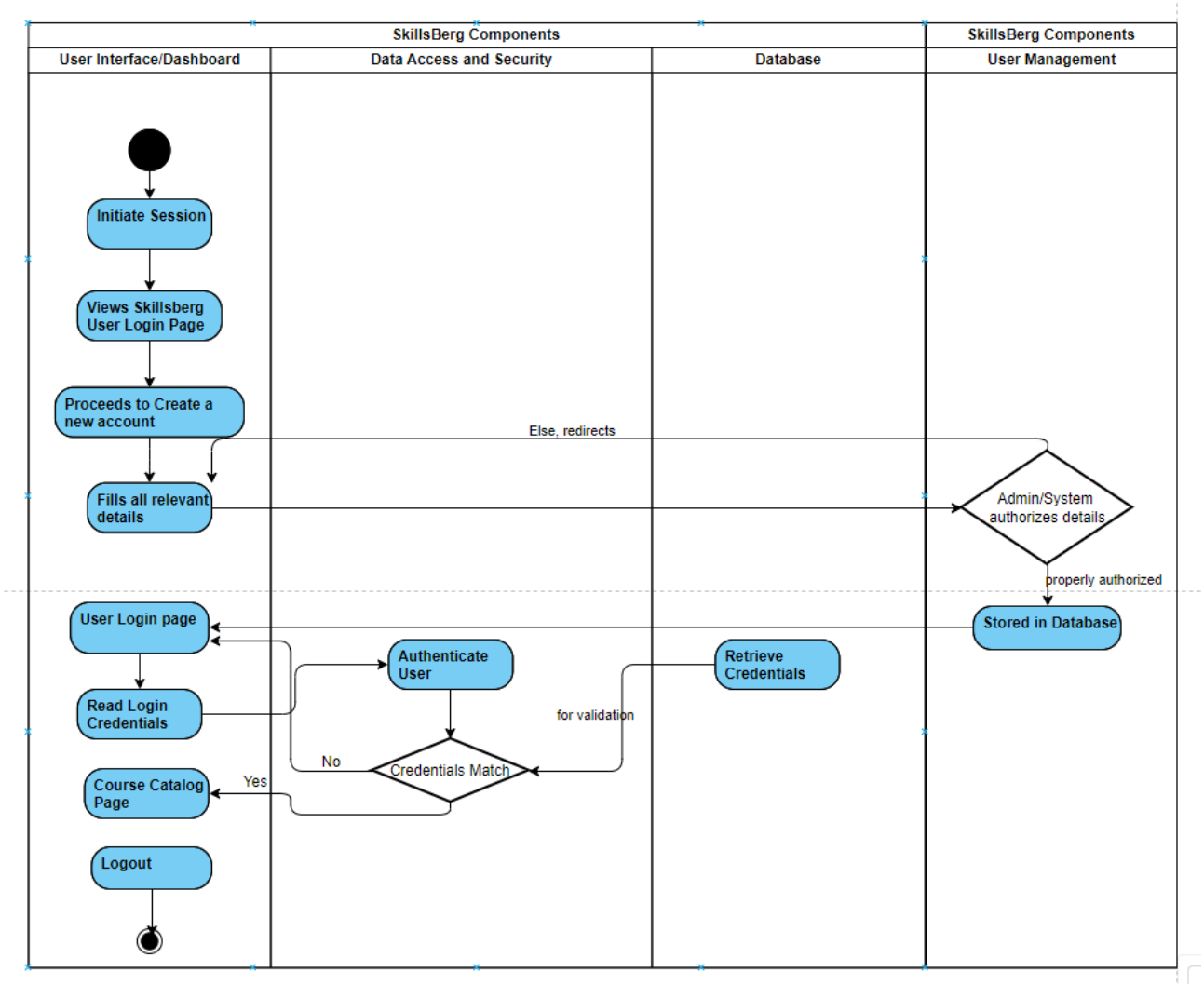
Provided Interfaces:
The Database Management component does not directly provide external interfaces but interacts with other system components to facilitate data storage and retrieval.

Required Interfaces:
The Database Management component relies on the following interfaces to interact with system components that require data storage and retrieval:

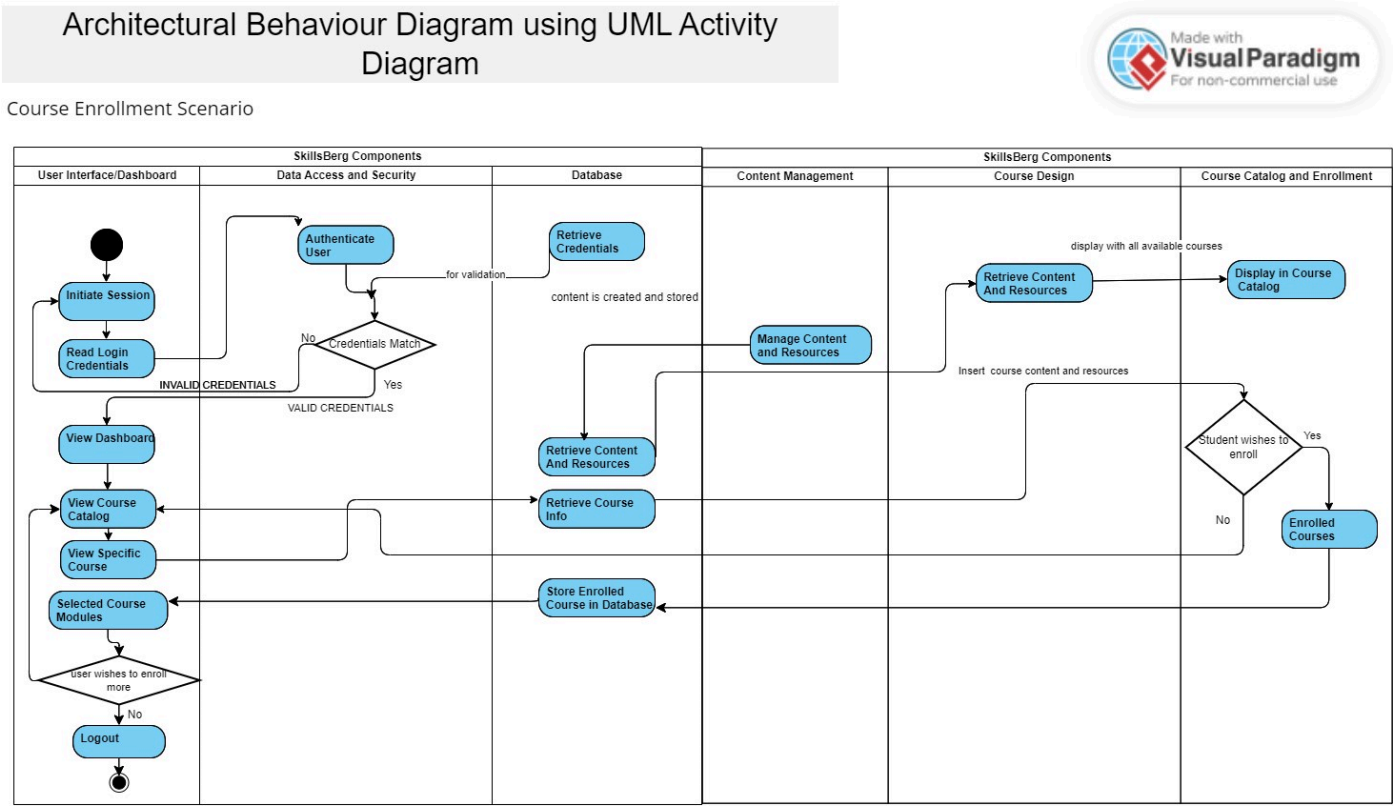
1. User Authentication Interface: User Authentication requires access to user data stored in the database for authentication and access control purposes.
2. Content Design Interface: Content Design needs access to course content and educational materials stored in the database to create, organize, and update content.

5. Architecture Behavior View

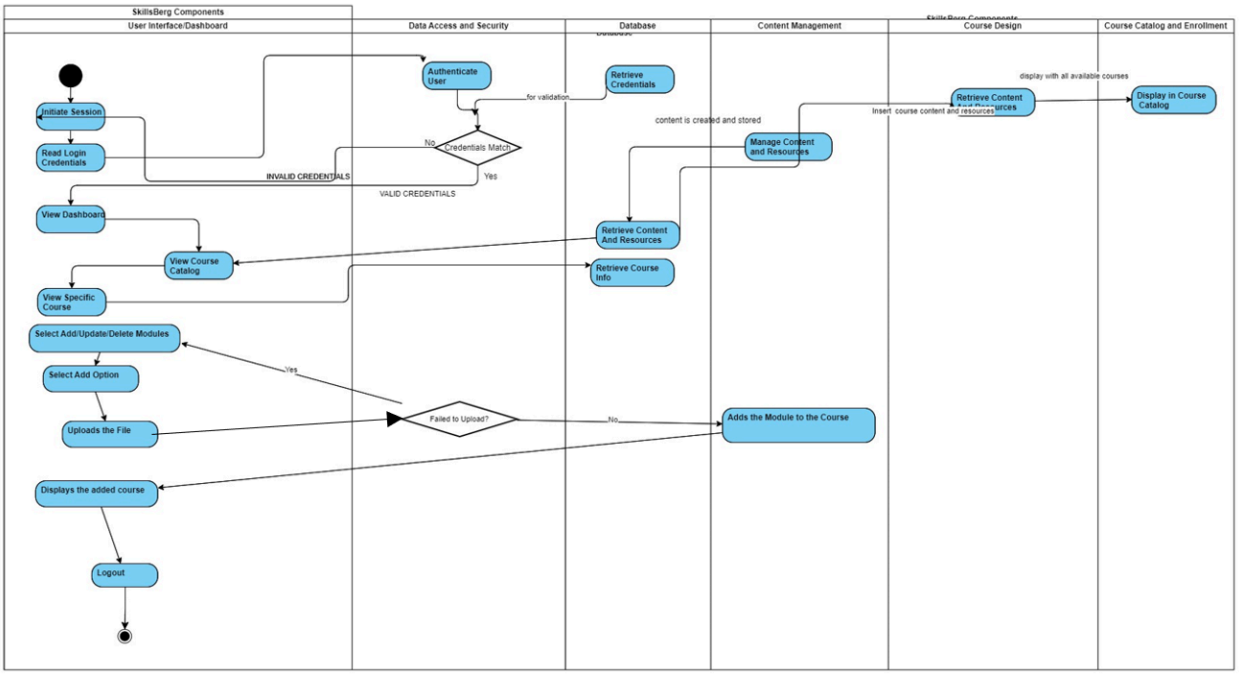
i) Architectural Behaviour Diagram using UML Activity Diagram on User Registration and Logging In by a Student:



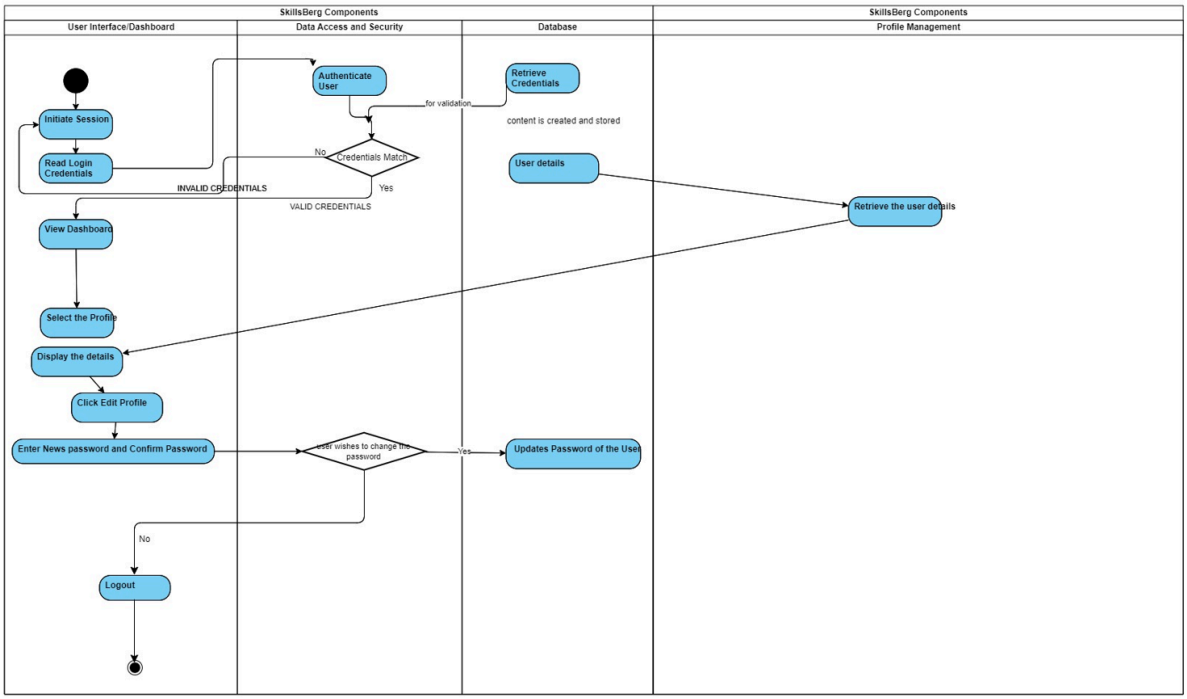
ii) Architectural Behaviour Diagram using UML Activity Diagram on Enrolling to a Course by a Student:



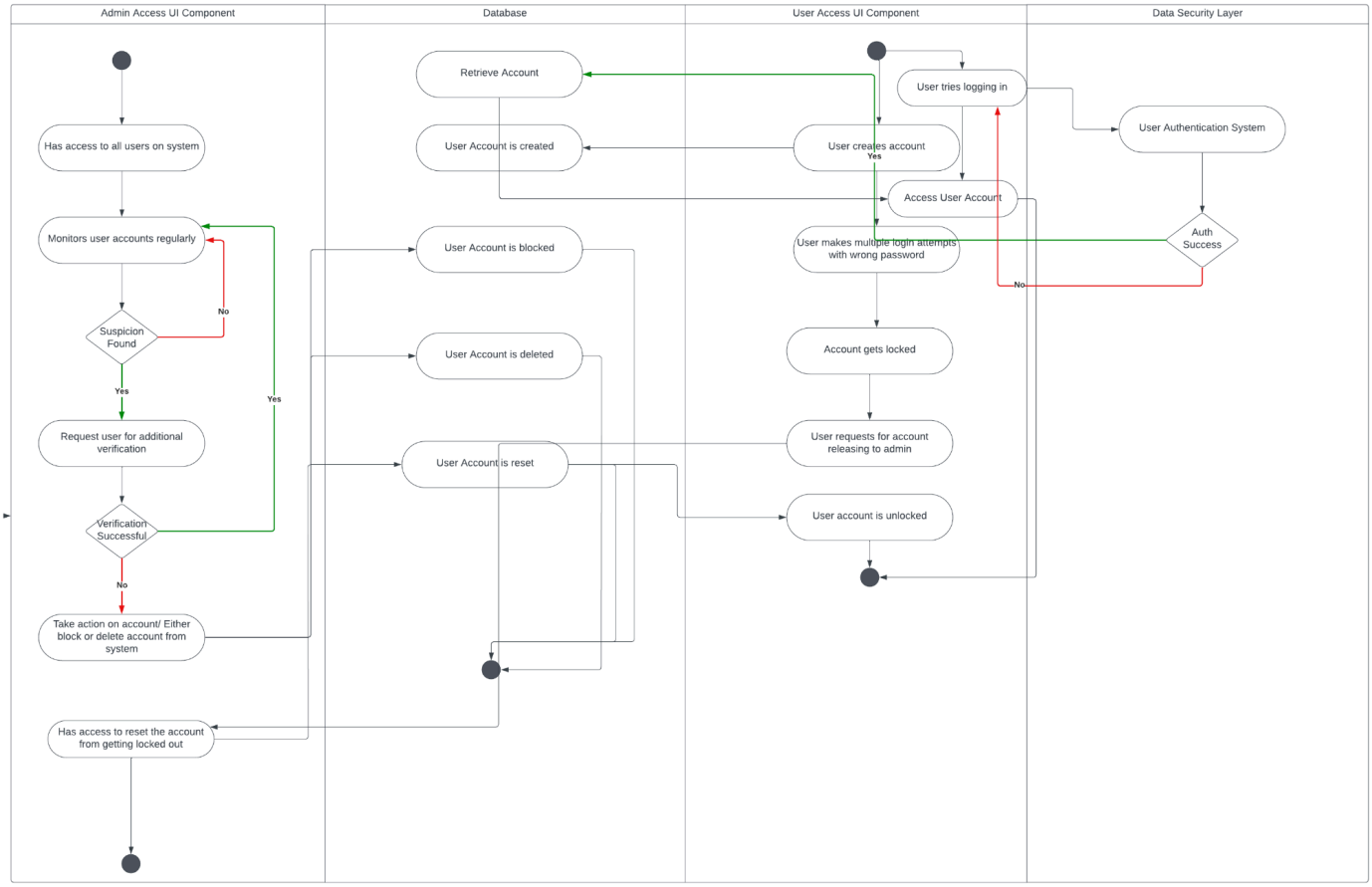
iii) Architectural Behaviour Diagram using UML Activity Diagram on adding a Module to the Course by an Instructor:



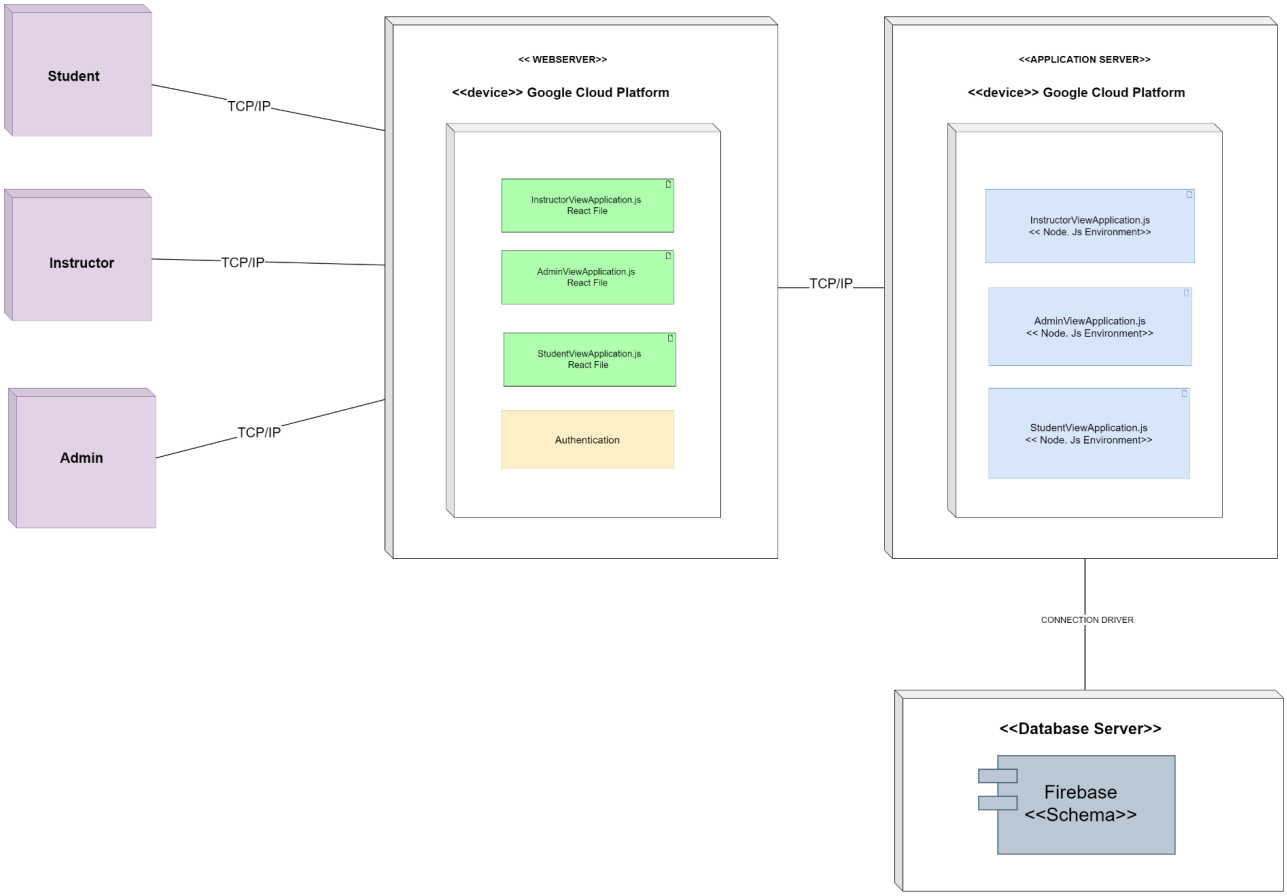
iv) Architectural Behaviour Diagram using UML Activity Diagram on updating password by a student:



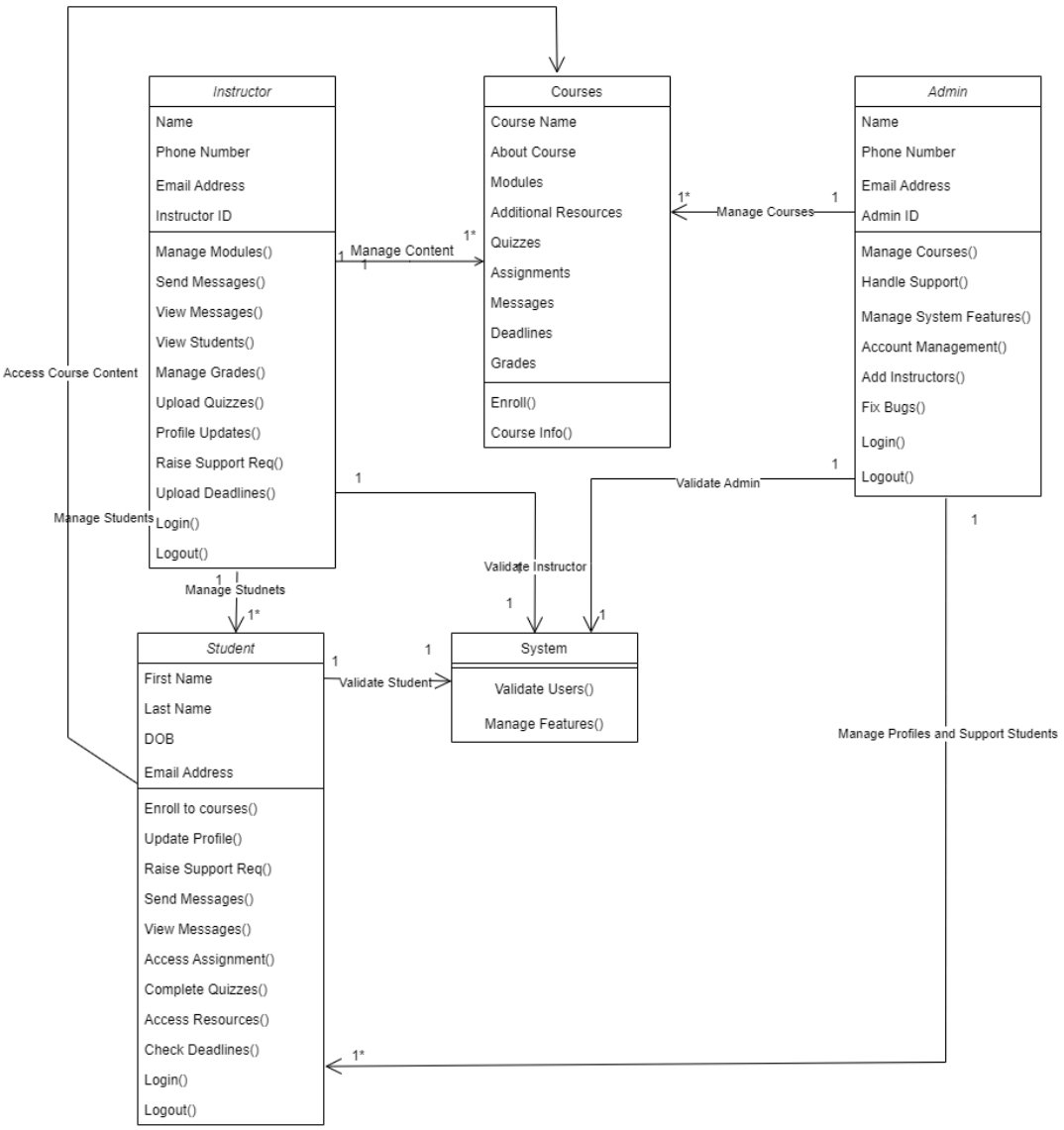
v) Architectural Behaviour Diagram using UML Activity Diagram on Secure User management (Quality Scenario):



6. Architecture Deployment View



7. Information View



8. Requirements Allocation

	User Registration	Profile Management	User Support	Message	User Authentication	User Authorization	Dashboard	Account Management	Course Design	Course Catalog & Enrollment	Enrolled Courses	Quiz Management and Grading	Assignment Management & Grading	Content Design	Resource Management	Database
Student																
Create Account	X															X
Login					X											X
Logout							X									
Forgot Password					X											
Support			X				X									
About US							X									
Course Dashboard							X									
Course Catalog							X			X						
Course Info And Enrollment										X						X
Enrolled Courses							X			X	X					X
Course Modules											X					
Quiz & Assignments Grades												X	X			X
Upload Quiz Assignment												X	X			
Progress Bar							X				X					
Messages				X							X					
Deadlines											X	X	X			X
Profile Page		X					X									X
Update Profile Page		X														X
Upload Disability Certificate	X	X														X
Instructor																

Login					X											
Logout							X									
Forgot Password					X											
Support			X				X									
About US							X									
Course Dashboard							X									
Manage Modules														X	X	X
Manage Resources														X	X	X
Manage Quiz & Assignments												X	X			
Upload Grades												X	X			
Upload Deadlines											X	X	X			
Send Messages				X							X					
Profile Page		X					X				X					X
Update Profile Page		X					X									X
Admin																
Login					X											
Logout							X									
Forgot Password					X											
About US							X									
Support to Stu & Instructors			X				X									
Account Management						X	X	X								X
Manage Courses							X		X							X
Manage Course Catalog							X		X	X						X
Manages User Authorization						X										X
Manage Database																X
Create Instructor Account	X															X
Create Admin Account	X															X
Profile Page		X					X				X					X

Update Profile Page		**X**					**X**								**X**

9. Architecture Work Allocation

	Tarang Nair	**Revanth Reddy Maturu**	**Pranathi Vadrevu**	**Sudhanshu Dubey**	**Bhanu Teja**
User Registration	Development	Testing			
Profile Management					
User Support					
User Authentication				Testing	Development
User Authorization					
Dashboard					
Account Management					
Message		Development	Testing		
Course Design	Testing		Development		
Course Catalog and Enrollment					
Enrolled Courses					
Content Design				Development	Testing
Resource Management					
Quiz Management and Grading	Development	Testing			
Assignment Management and Grading					
Database			Implementation		Testing

10. Detailed Design Key Decisions and Rationale

Key decisions taken [Design Principles chosen]

> **1. Single Responsibility Principle (SRP):** This principle states that a class should have only one reason to change, meaning it should have only one job or responsibility.
>
> **2. Open Close Principle (OCP):** This principle states that OCP suggests that classes/modules should be open for extension but closed for modification. This principle encourages designing systems where new functionalities can be added without altering existing code, promoting scalability and maintainability.

Skillsberg has the following subsystems:

1. User Management and Support
2. Content Management
3. Access Control and Security
4. Course Management
5. Quiz and Assignment
6. Data Storage and Management

Detail Design of these, with an example, their responsibility, application of principle used, rationale for using.

1. **User Management and Support**

   → UserRepository Module: (Example)

   - **Responsibility:** Managing user data storage/retrieval in Firebase.
   - **Application of SRP:** This module should solely handle CRUD operations related to user data in Firebase. It should not be concerned with authentication logic or business rules.
   - **Rationale:** By adhering to SRP, any changes in how user data is stored or retrieved won't impact authentication or other functionalities.

2. **Content Management**

   → ContentService Module: (Example)

- **Responsibility:** Handling content creation, modification, and retrieval.
- **Application of OCP:** Design the module to allow easy extension for new content types (e.g., text-based, video-based, interactive modules) by defining an interface for content creation and leveraging polymorphism to handle different content types.
- **Rationale:** This approach allows for the addition of new content types without altering the existing codebase. For instance, adding a new type of language learning content won't require modifications to the existing content management logic.

3. Access Control and Security

→ Authenticator Module: (Example)

- **Responsibility:** Handling user authentication and access control.
- **Application of SRP:** The module should focus solely on validating user credentials, and managing access permissions.
- **Rationale:** This segregation allows for better management of security-related changes without affecting other aspects of the system. For instance, changes in authentication mechanisms won't impact course management or user support functionalities.

4. Course Management

→ Course Design: (Example)

- **Responsibility:** Handling course catalog and enrollment
- **Application of OCP:** Design the module to allow easy extension for new course content (e.g., modules, quizzes, assignments, resouces) by defining an interface for resource management and leveraging polymorphism to handle different requirements.
- **Rationale:** This approach allows for the addition of new resources without altering the existing codebase. For instance, adding a new type of language learning content won't require modifications to the existing course management logic.

5. Quiz and Assignment

→ QuizHandler : (Example)

- **Responsibility:** Managing quizzes, assignments, and assessments.
- **Application of OCP:** Implement a strategy pattern where different types of quizzes or assignments (e.g., multiple-choice, essay-based) are defined as separate classes implementing a common interface.
- **Rationale:** By following OCP, new types of assessments can be added without modifying the existing codebase. For instance, incorporating a new type of assessment won't require changes to the core quiz-handling logic.

6. Data Storage and Management

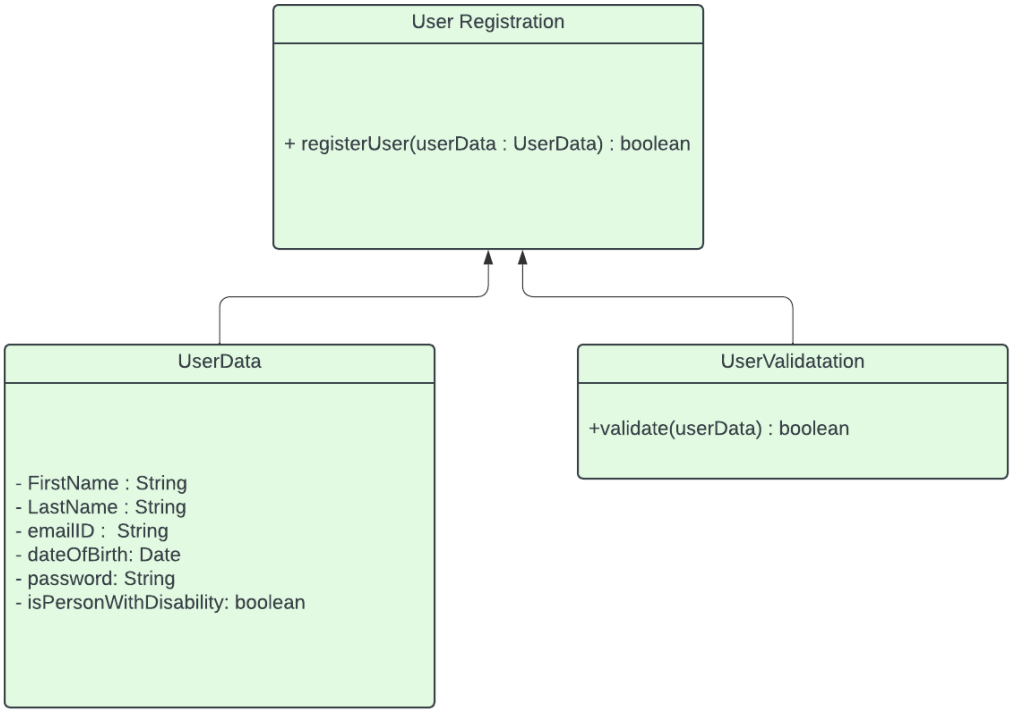
→ DataManager Module: (Example)

- **Responsibility:** Handling data storage and retrieval from Firebase or other data sources.
- **Application of SRP:** Adhering to the Single Responsibility Principle, ensures that this module is solely responsible for managing data storage, retrieval, and possibly caching strategies. It should not handle business logic, authentication, or other unrelated functionalities.
- **Rationale:** By focusing solely on data storage and management concerns, this module becomes easier to maintain, and changes related to data storage methods or optimization won't affect other parts of the system. This separation of concerns ensures better code organization and maintainability.

11. Detailed Design Structure

1. UML Class Diagram, depicting the detailed design of
   Component: 'User Registration'
   Subsystem: 'User Management and Support'

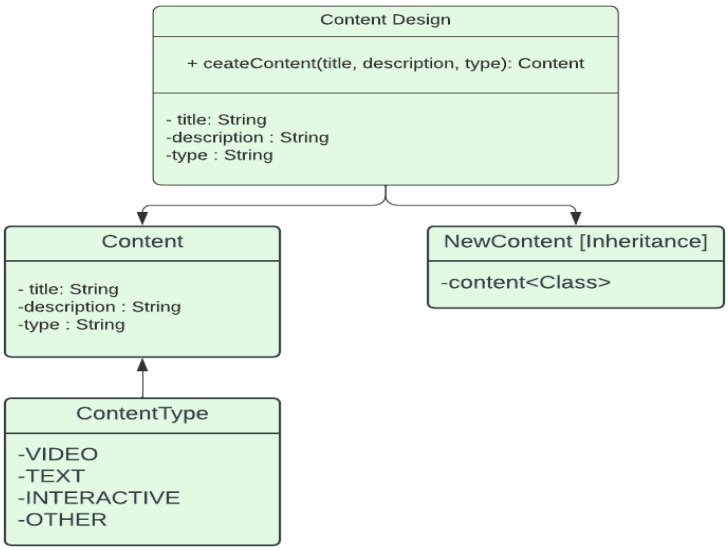
- **User Registration** class remains the central entity responsible for the user registration process.



- **UserData** represents the essential user data required for registration.
- **UserValidation** class handles data validation before the registration process. This separates the responsibility of validating user input from the registration logic.

  Adhering to SRP by separating the validation logic from the registration process, ensured a more modular and maintainable design for user registration

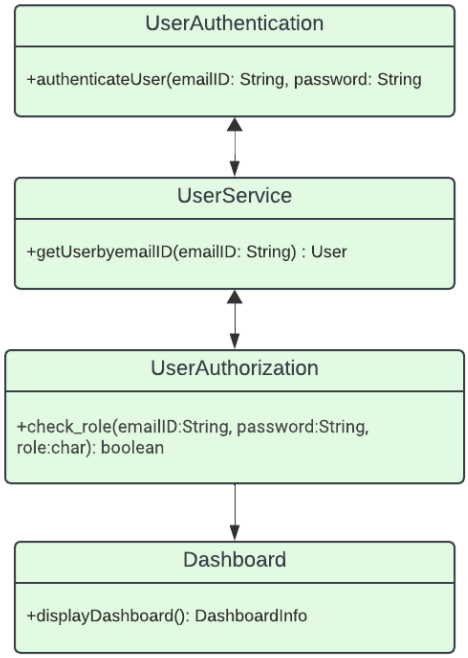
2. UML Class Diagram, depicting the detailed design of
   Component: ‘Content Design’
   Subsystem: ‘Content Management’



- The **ContentDesign** class contains the method **createContent(contentData: ContentData): Content**, responsible for creating content based on the provided **ContentData**.
- **ContentDesign** represents the necessary information (title, description, type) required for creating various types of content.
- **NewContent** class represents the inheritance of class ContentDesign
- **Content** class represents the attributes required for content, containing an associated **ContentType**.
- **ContentType** enumerates different content types available in Skillsberg (e.g., VIDEO, TEXT, INTERACTIVE, OTHER).

→ By utilizing the OCP design strategy, the 'Content Design' component exhibits openness for extension without the need for modifying existing classes, methods, or logic, which aligns with the Open-Closed Principle. New content types can be introduced smoothly by extending existing structures without altering the core content creation functionality encapsulated within the ContentDesign class.

3. UML Class Diagram, depicting the detailed design of
Component: ‘User Authentication and User Authorization’
Subsystem: ‘Access Control and Security’

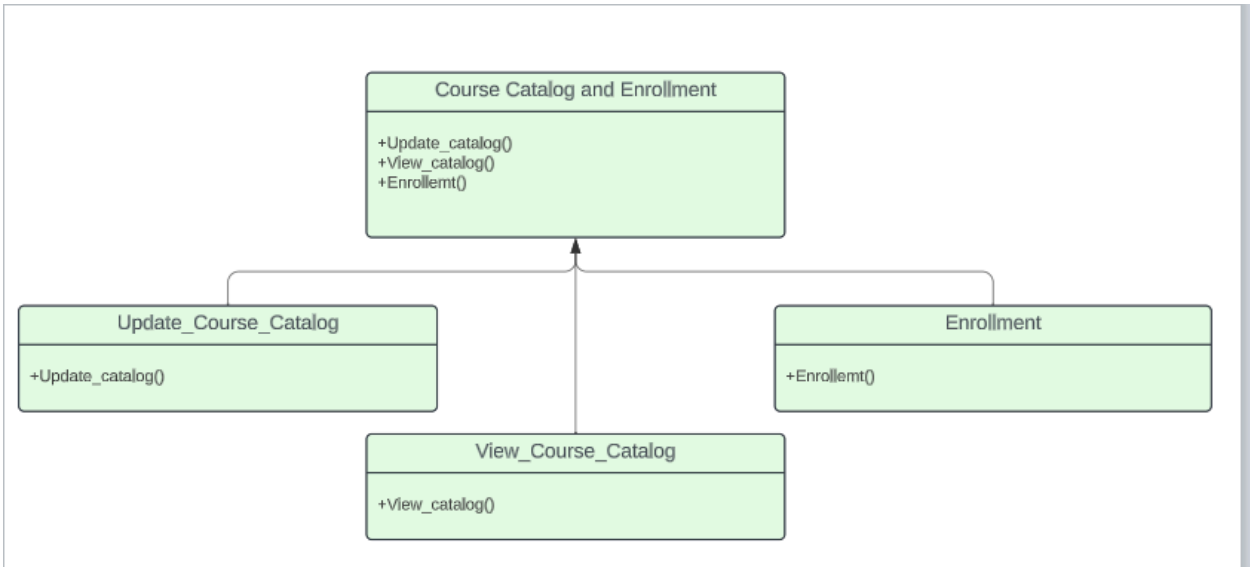


- **UserAuthentication** interface includes the **authenticateUser** method signature responsible for user authentication.
→ The **authenticateUser(username: String, password: String): boolean**: Verifies user credentials by interacting with **UserService**. Returns a boolean indicating authentication success or failure.

- **UserAuthorization** interface includes the **check_role** method signature for user authorization.
→ The **check_role** method validates user authorization by checking user roles against their emailID and password**.**

  → By segregating user authentication and user authorization into separate components, this design adheres to the Single Responsibility Principle. UserAuthentication focuses solely on authenticating user credentials, while UserAuthorization concentrates on determining user access based on roles/permissions, ensuring a more maintainable and robust access control system within the 'Access Control and Security' subsystem.

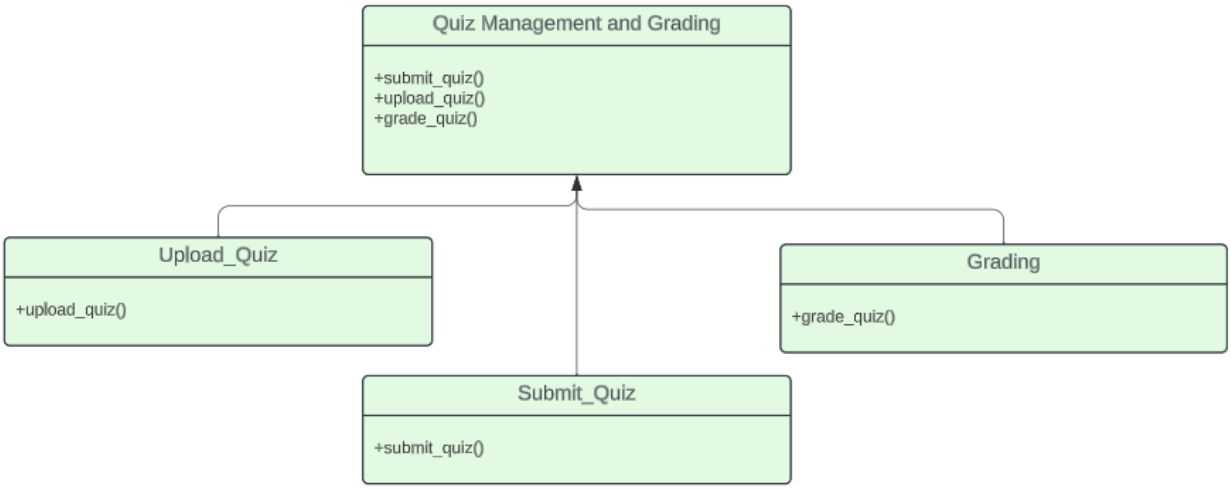
4. UML Class Diagram, depicting the detailed design of
   Component: ‘Course Catalog and Enrollment’
   Subsystem: ‘Course Management’



- **CourseCatalog and Enrollment** interface defines methods related to viewing and updating catalog, enrollment to courses, and accessing enrolled courses. It serves as a contract allowing different implementations.
- **Update_Course_Catalog** class focuses on updating courses, keeping the responsibility of course management separate.
- **View_Course_Catalog** class manages the list of courses available for a user to enroll.
- **Dashboard** class deals with displaying user-specific dashboard information, emphasizing dashboard management.

  → The design allows for extending functionality without modifying existing classes/interfaces. For instance, new implementations (like different ways of viewing courses or enrolling) can be added for **CourseCatalogAbdEnrollment** without altering the existing interface, promoting modularity and flexibility within the 'Course Catalog and Enrollment' subsystem.

5. UML Class Diagram, depicting the detailed design of
   Component: Quiz Management and Grading'
   Subsystem: Quiz and Assignments'



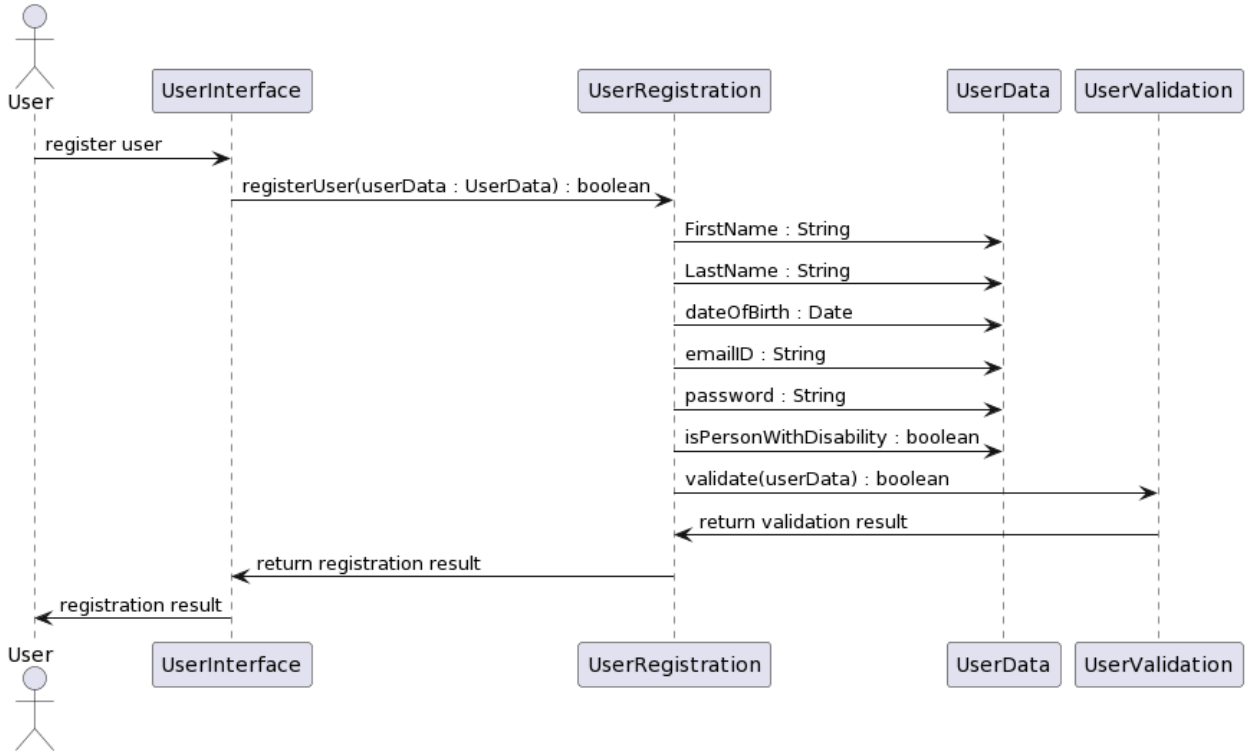
- **Upload Quiz, Submit_Quiz and Grading** are individual sub classes, following SRP design principle under Quiz Management and Grading.

12. Detailed Design Behavior

1)UML Sequence Diagram, depicting the detailed design behavior of

Component: ‘User Registration’

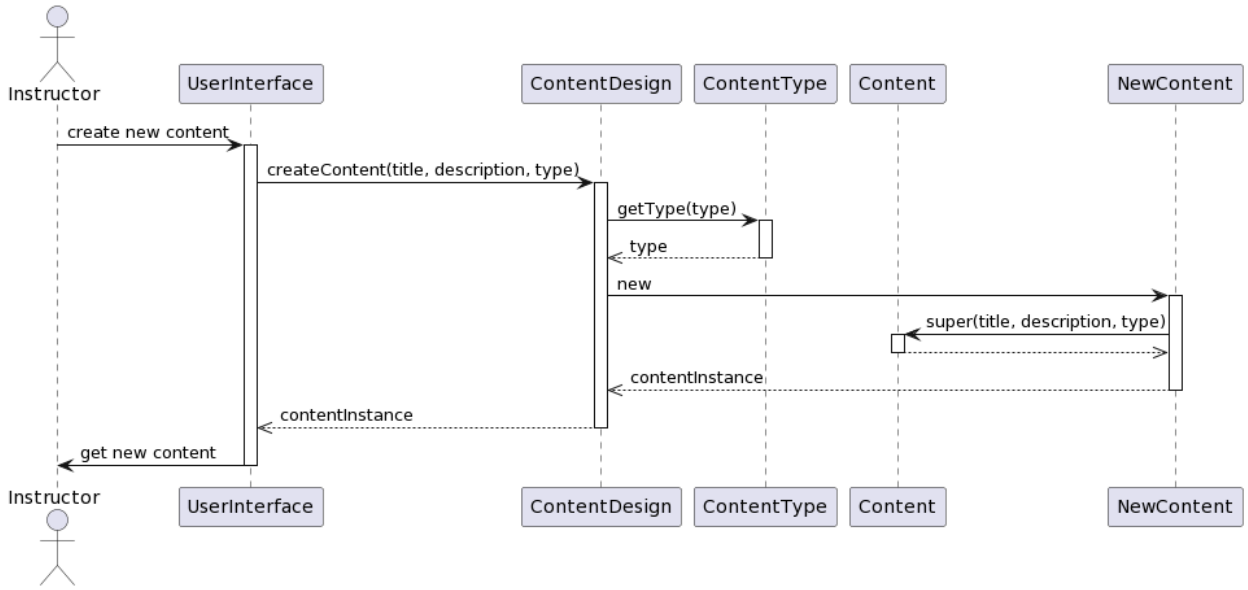
Subsystem: ‘User Management and Support’



2)UML Sequence Diagram, depicting the detailed design behavior of

Component: ‘Content Design’

Subsystem: ‘Content Management’

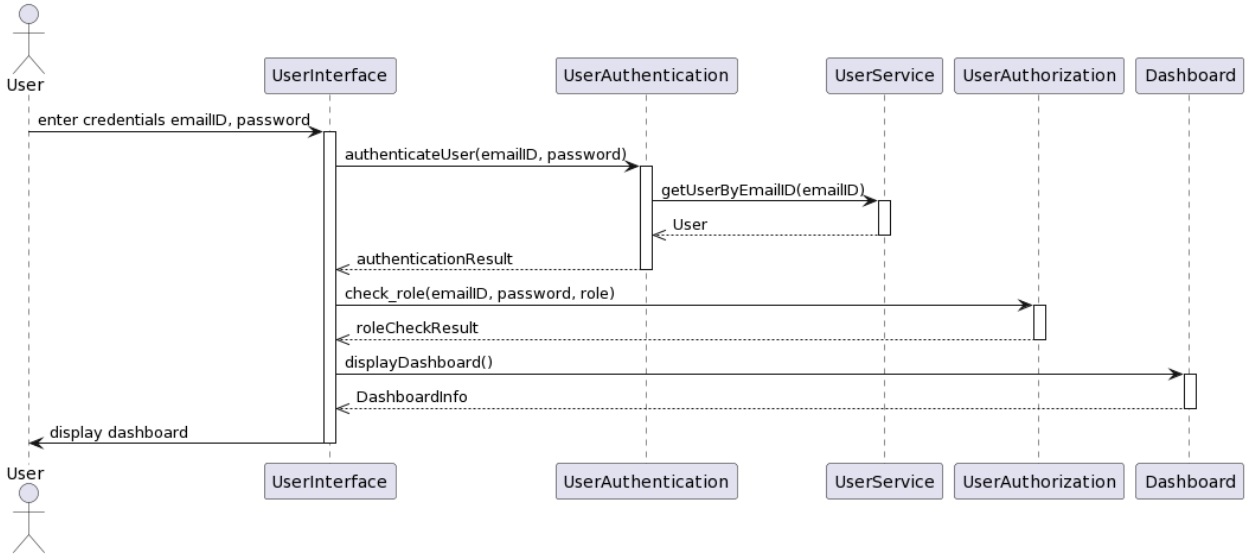


---

3)UML Sequence Diagram, depicting the detailed design behavior of

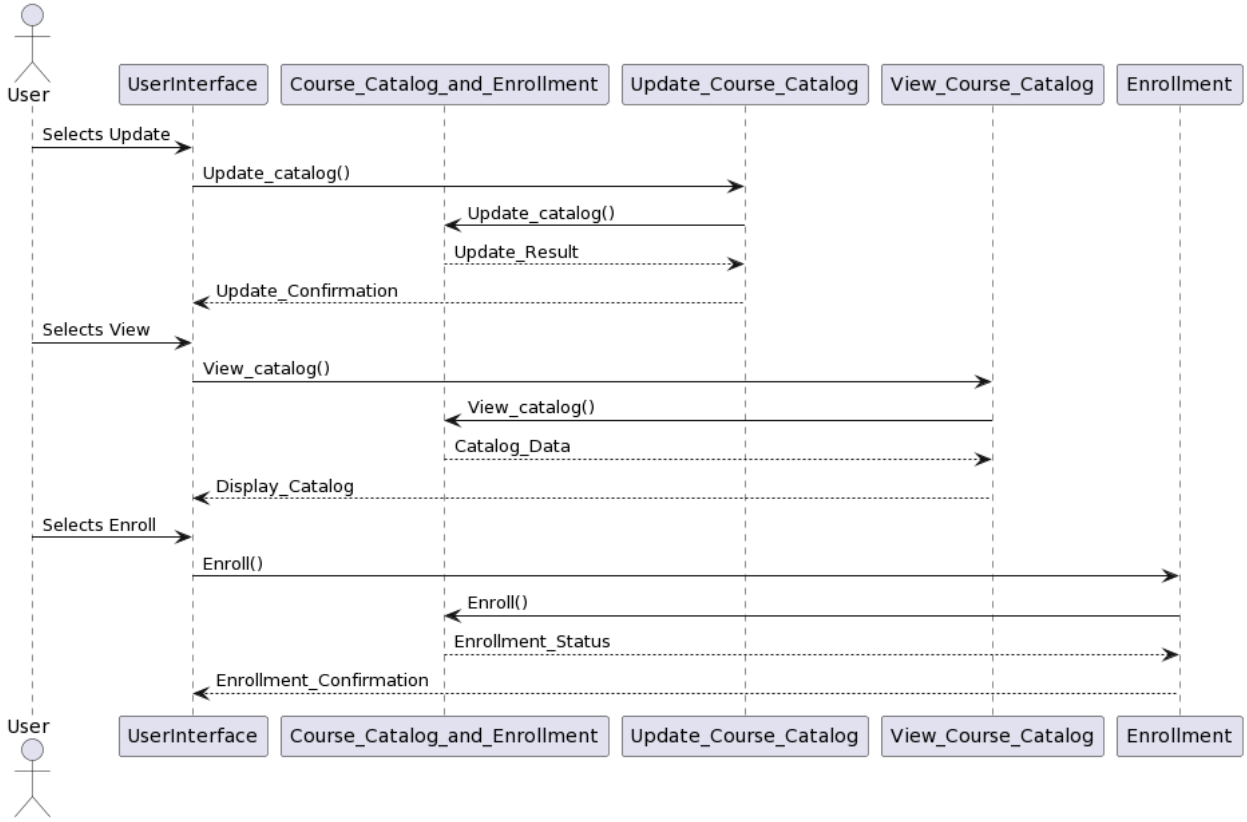
Component: ‘User Authentication and User Authorization’

Subsystem: ‘Access Control and Security’



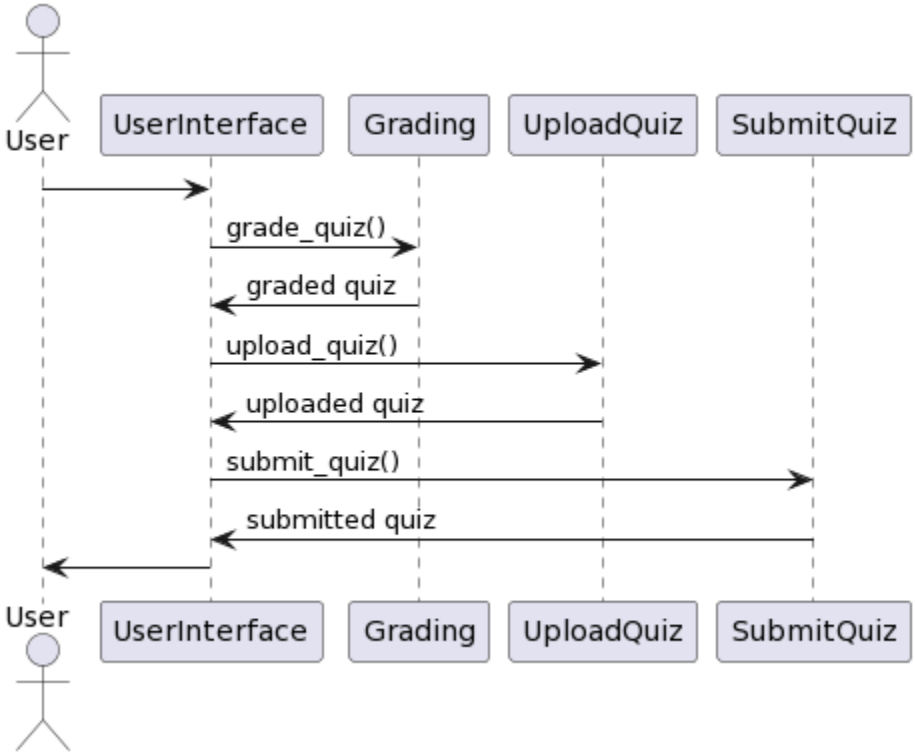
4)UML Sequence Diagram, depicting the detailed design behavior of

Component: ‘Course Catalog and Enrollment’
Subsystem: ‘Course Management’



5)UML Sequence Diagram, depicting the detailed design behavior of

Component: Quiz Management and Grading
Subsystem: Quiz and Assignments'



13. Physical Data Model

For this project we have decided to use Firebase, which is a NoSQL Database.
Some of the reasons for using this is :

- Access to a free plan for initial application development
- A serverless platform
- And faster development time

In Firebase we store data in documents using key-value pairs, which are then organized in collections
There is a possibility of each document to have a subcollection

Data Model for Instructor

> Instructor Collection Document's Data Fields
We would be creating a instructor collection
The collection would hold documents for each instructor

id : String name : String courses : List<String> profileImage : String bio : String address : String email : String phoneNo : String

Data Model For Student

We would be creating a student collection
This collection would hold documents for each instructor

> Student's Collection Document's Data Fields

id : String
name : String
profileImage : String
bio : String
address : String
email : String
phoneNo : String

> Student's Course Subcollection Document's Data Fields

Each Student Document would have a course sub collection whose documents would be hold information about the students enrolled courses

studentId : String
courseId : String
batchId : String
enrolledAt : TimeStamp
totalMarksObtained : Integer
totalPossibleMarks : Integer
gradeObtained : String

> Student's ⇒ Course ⇒ Quiz Subcollection Document's Data Fields

Each Student's course documents would contain a Quiz subcollection whose documents would contain the information of all the student quizzes that were scheduled for him

```
studentId : String
courseId : String
quizId : String
status : String
answers : [{
        questionId : String,
        questionType : String
        answerProvided : String
        optionsSelected : List<Integer>
}]
submittedAt : String
timeTakenInMin : String
attemptCount : Integer
marksObtained : float
```

Here the status field could contain values like SCHEDULED/ACTIVE/ASSESSMENT_PENDING/RESULT_GENERATED depending on the status of the student's Quiz

The question Type could either be MCQ, NUMERICAL or SUBJECTIVE. Based on this either the answer provides or the optionsSelected Field would be present

> Student's ⇒ Course ⇒ Assignment Subcollection Document's Data Fields

Each Student's course documents would contain an Assignment subcollection whose documents would contain the information of all the student assignments that were scheduled for him

```
studentId : String
courseId : String
assignmentId : String
status : String
mediaSubmittedURI : List<String>
submittedAt : TimeStamp
```

```
attemptCount : Integer
marksObtained : float
```

Data Model For Questions

> Question Collection Document Field

```
id : String
courseId : String
questionText : String
type : String
numericalCorrectAnswer : Integer
mcqOptions : [
                    {
                        text : String,
                        isCorrect : bool
                    },
                    ..
            ]
marks : float
```

The type could have values NUMERICAL/MCQ/SUBJECTIVE
Based on the question Type the fields numericalCorrectAnswer or mcqOptions would be available

Data Models for Courses

> Course Collection Document Fields
The documents in the course collection would hold details about the course

id : String
name : String
description : String
credits : Integer

> Course Batch Subcollection Document Field
These documents would contain information about the batches available for the course

courseId : String
batchId : String
batchName : String
studentIds : List<String>

The batches would contain a list of ids of all the students who are enrolled in this course

> **Course => Quiz Sub Collection Document Fields**
The documents in this subcollection would hold information of all the quizzes scheduled for this course

quizId : String
courseId : String
batchId : String
availableDate : TimeStamp
dueDate : TimeStamp
questions : List<String>
timeLimitInMin : Integer
maxMarks : float
createdBy : String
createdAt : TimeStamp
noOfAttempts : Integer

> Course => Assignment Sub Collection Document Fields
The document in this subcollection would hold information of all the assignments scheduled for this course

assignmentId : String
courseId : string
batchId : String
assignmentDescription : String

```
availableDate : Timestamp
dueDate : TimeStamp
media : [{
   type : String
   url : String
}],
maxMarks : float
noOfAttempts : Integer
createdAt : TimeStamp
createdBy : String
```

Here media could be of type IMAGE or VIDEO

> Course => Discussions Subcollection
These documents would hold information about the discussion for each courses

```
courseId : String
discussionId : String
title : String
text : String
```

> Course => Discussions => Comments Subcollection
These document would hold information of all the comments present in the a particular discussion

```
courseId : String
discussionId : String
commentId : String
userId : String
commentText : String
commentMediaURI : List<String>
```

14. Architecture to Detailed Design Tracing

		UserData	User Validation	Content Creation	New Content	User Authentication	User Service	User Authorization	Course Catalog	Course Enrollment	Dashboard	Upload Quiz	Submit Quiz	Grading
ARCHITECTURAL COMPONENT/ INTERFACES	User Registration	X	X											
	Content Design			X	X									
	User Authentication	X	X			X	X							
	User Authorization	X	X				X	X			X			
	Course Catalog and Enrollment								X	X	X			
	Quiz Management and Grading											X	X	X
	Create Account Interface	X	X											
	Student Login Interface	X	X			X		X						
	ProfessorLogin Interface	X	X			X		X						
	Admin	X	X			X		X						

	Login Interface													
	Verify and Validate User					**X**	**X**	**X**						
	Function Navigation Interface		**X**			**X**		**X**			**X**			
	Course Registration									**X**				
	Modify Course Catalog								**X**					
	Browse Enrolled Courses									**X**				
	Submit Quiz										**X**		**X**	
	Upload and Grade Quiz											**X**		**X**
	Manage Quiz and Assignment											**X**	**X**	**X**
	Manage Content			**X**	**X**						**X**			