ClassNest

Your Nest for Seamless Learning

SOFTWARE DESIGN AND IMPLEMENTATION

ENPM - 613

Software Architecture and Detailed Design

TEAM -11

Aanchal Pandey

Anantha Sridhar

Balakrishna Nair

Piyush Anand

Trishan Koyedi

INDEX

1. DETAILED DESIGN KEY DECISIONS AND RATIONALE

One of the fundamental SOLID principles of object oriented design implemented in ClassNest is the **Single Responsibility Principle (SRP).** This principle ensures that each class has only one reason to change, making the code more modular with minimal interdependencies.Each of these services performs a specific task independently. The modularity in ClassNest is achieved by dividing it into distinct components:

1. **User Management Independence:**
   - Changes to authentication or authorization mechanisms don't impact assignment management
   - User Related functionality modifications remain isolated from course content management
   - Profile management changes don't affect enrollment processes

2. **Assignment Management Isolation:**
   - Changes to assignment grading or submission processes don't impact other system areas
   - Updates to assignment deadlines or requirements remain contained within the service
   - Modifications to grading algorithms don't affect course content or user management

3. **Course Content Separation:**
   - Updates to course materials don't interfere with enrollment processes
   - Changes in content organization remain isolated from user authentication
   - Course structure modifications don't impact assignment submissions

Chain of Responsibility Pattern

This pattern is implemented in ClassNest to create a flexible and extensible system.:

1. **Course Management Chain:**
   - Different handlers manage course creation, updates, and archiving
   - Specialized handlers for different course types or academic levels
   - Sequential processing of course related requests

2. **Assignment Management Chain:**

- Separate handlers for assignment creation, submission, and grading

3. **User Management Chain:**
   - Distinct handlers for registration, authentication, and profile management
   - Rolebased access control handlers
   - Sequential processing of user related requests

Facade Pattern Implementation

ClassNest implements the Facade Pattern to:

1. **Simplify System Interaction:**
   - Provides unified interfaces for students and instructors
   - Hides complex subsystem interactions
   - Streamlines common operations

2. **Enhance Maintainability:**
   - Clear separation between interface and implementation
   - Easier system updates and modifications
   - Reduced dependencies between components

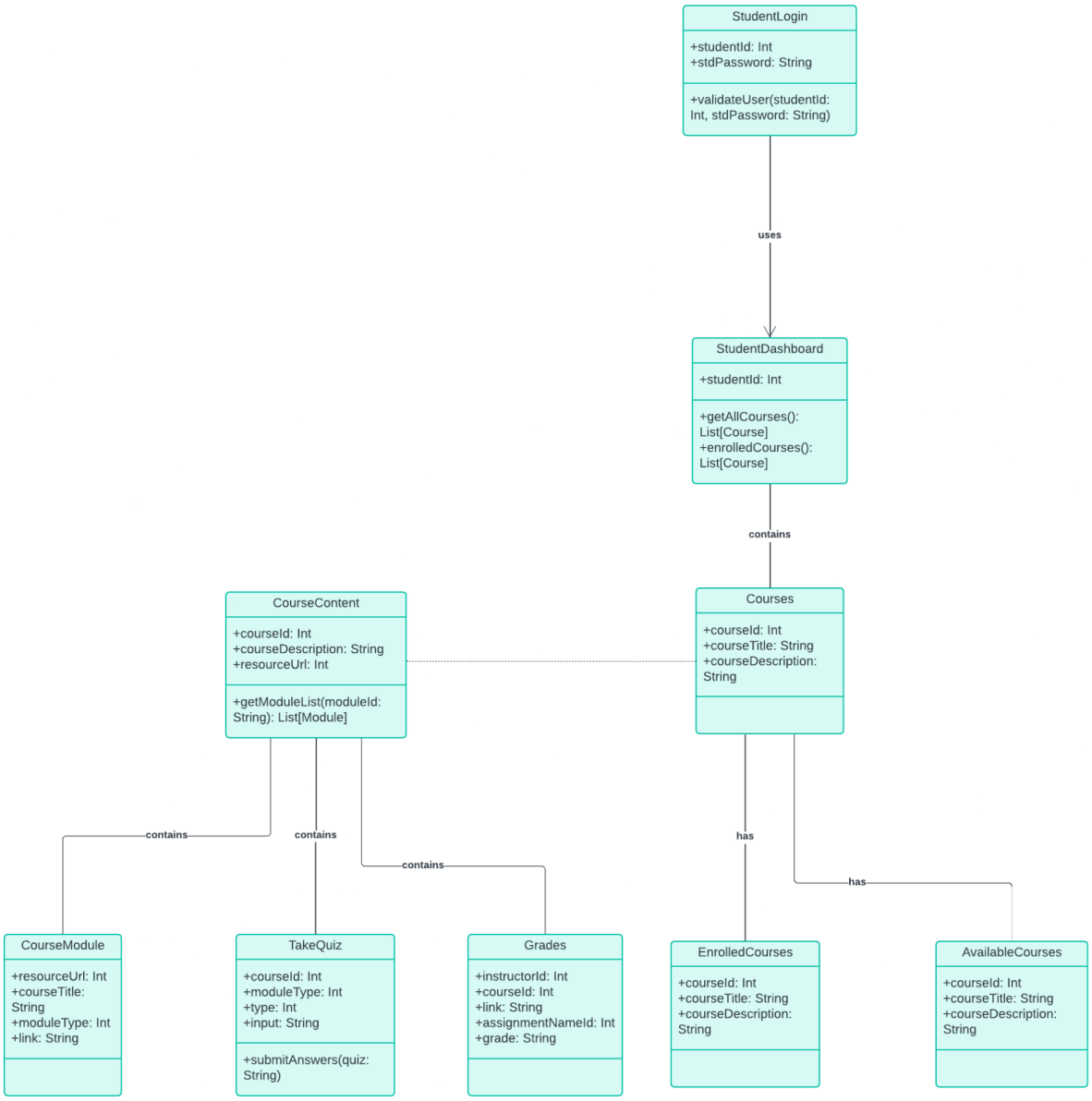
3. **Improve User Experience:**
   - Simplified access to course materials
   - Streamlined assignment submission process
   - Intuitive course management for instructors

4. **Support System Evolution:**
   - Easy addition of new features
   - Simple integration of new components
   - Flexible adaptation to changing requirements

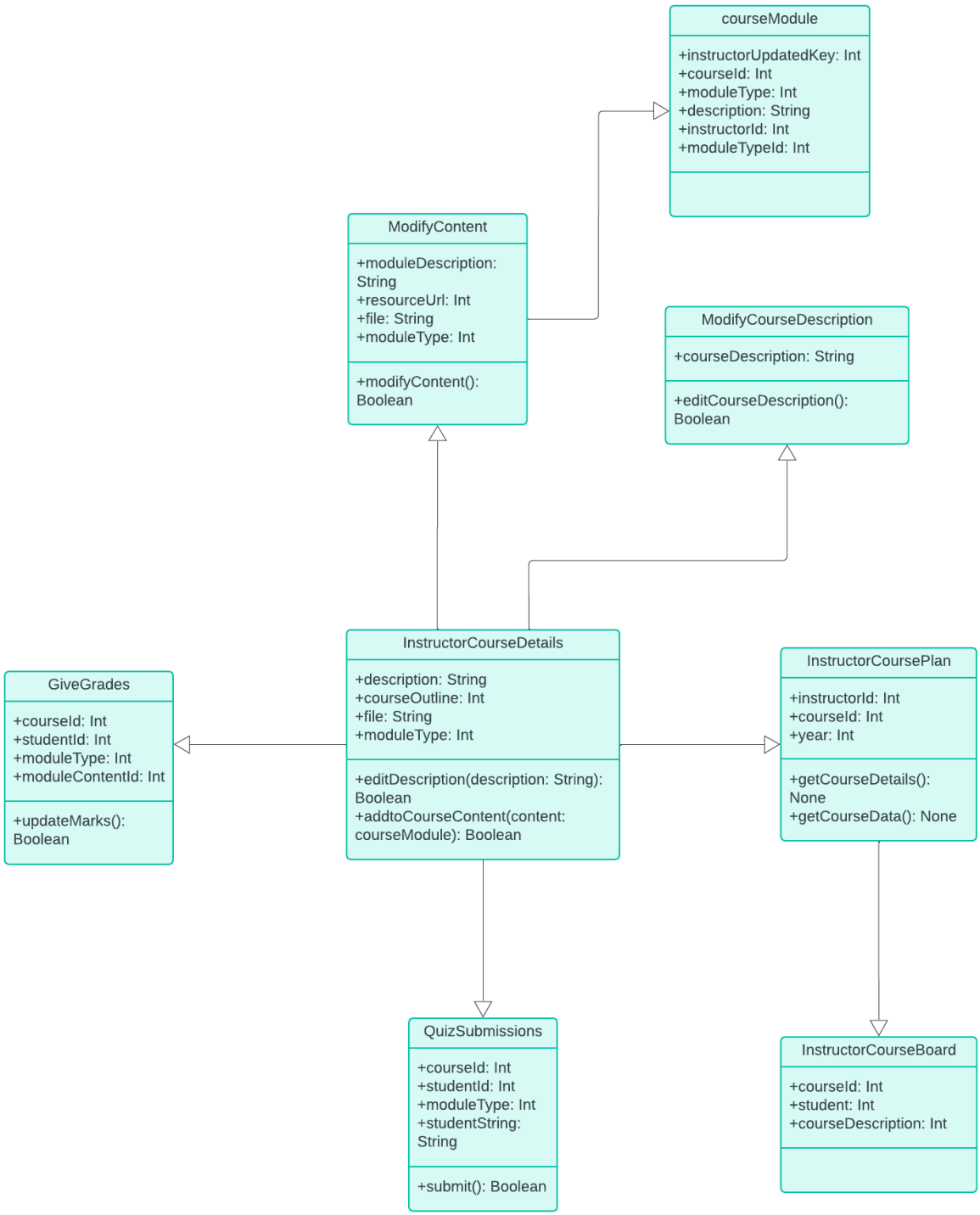
This architectural approach ensures ClassNest remains maintainable, scalable, and user friendly while adhering to solid design principles.

2. DETAILED DESIGN STRUCTURE

Student UI :



Instructor UI:

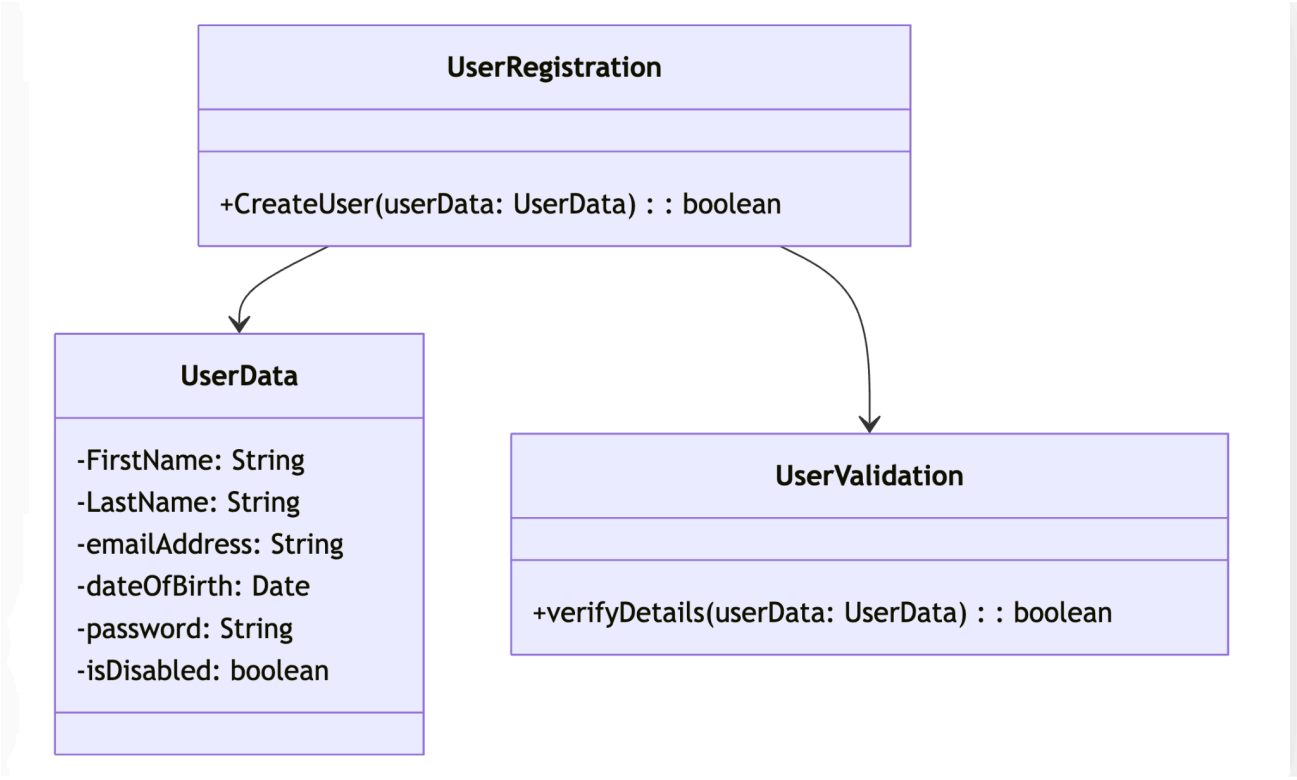


1. UML Class Diagram, depicting the detailed design of

Component: ‘User Registration’

Subsystem: ‘User Management and Support’

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

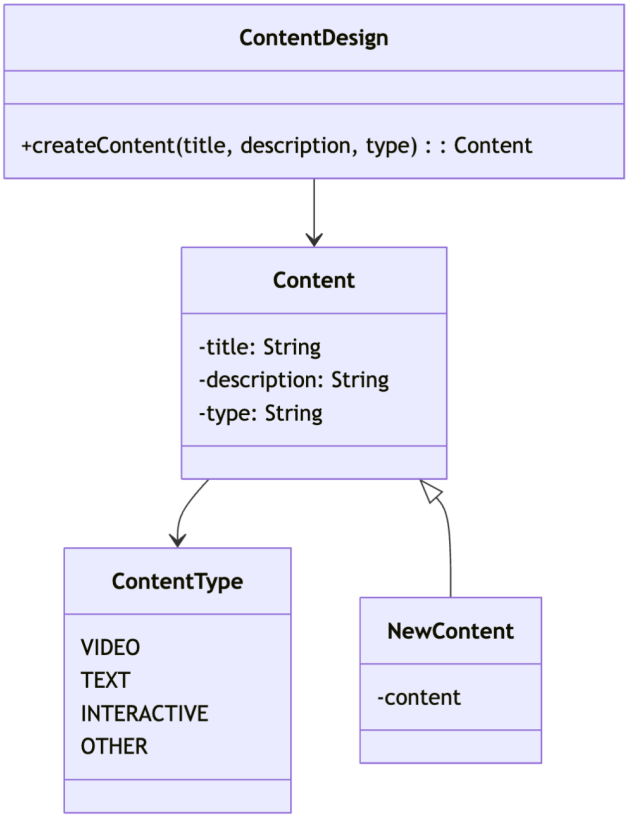


UserData represents the essential user data required for registration.

● **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’



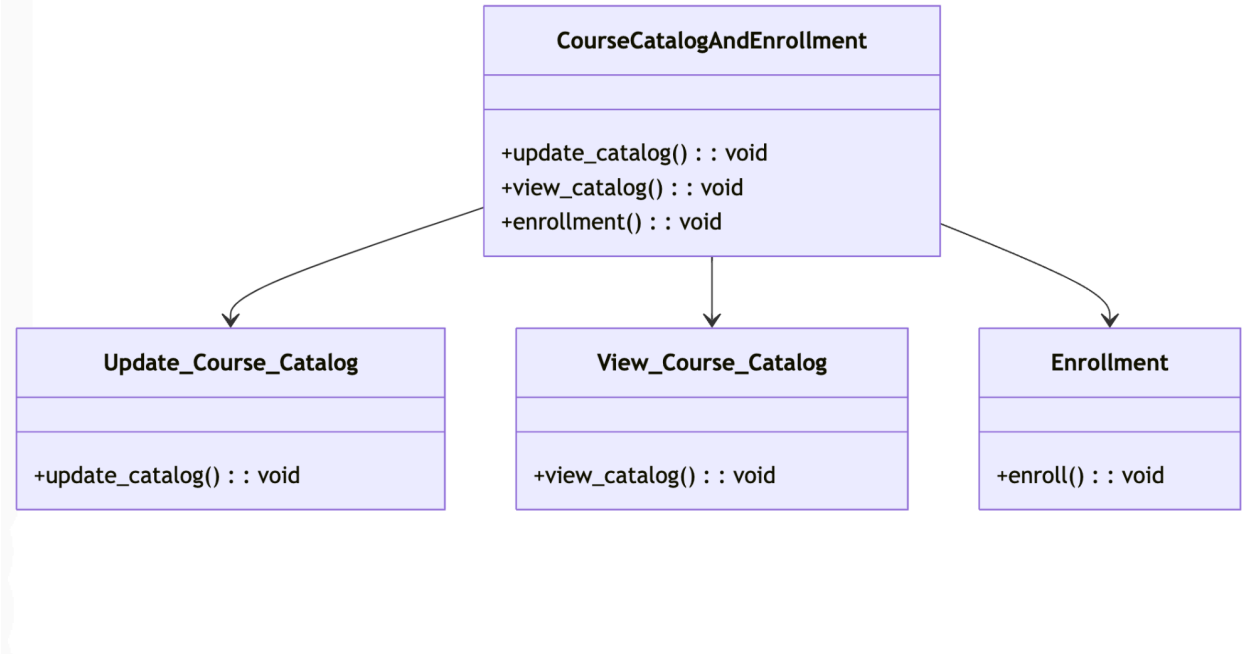
The **ContentDesign** class contains the method **createContent(contentData: ContentData): Content**, responsible for creating content based on the provided **ContentData**.

- **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

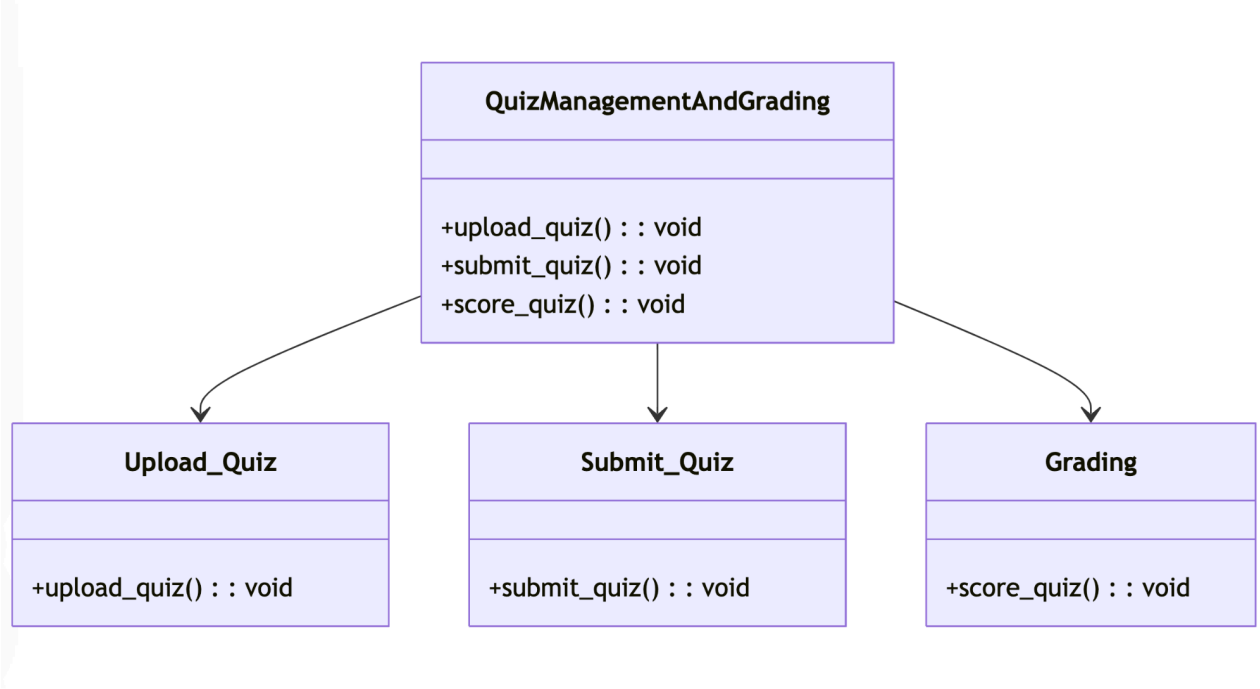
smoothly by extending existing structures without altering the core content creation
functionality encapsulated within the ContentDesign class.

3. UML Class Diagram, depicting the detailed design of
Component: ‘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.

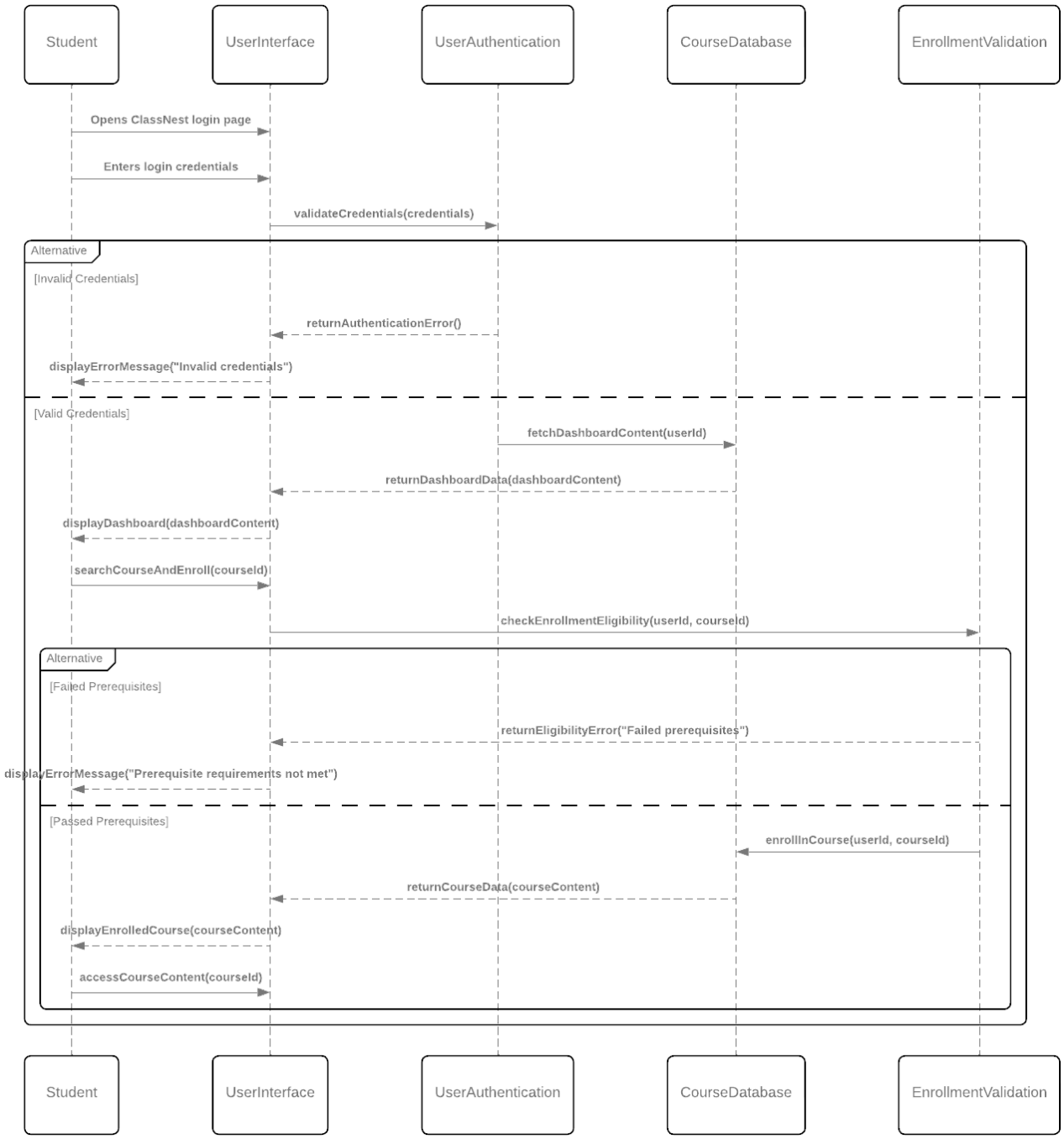
4. UML Class Diagram, depicting the detailed design of Component: Quiz Management and Grading'
Subsystem: Quiz and Assignments'



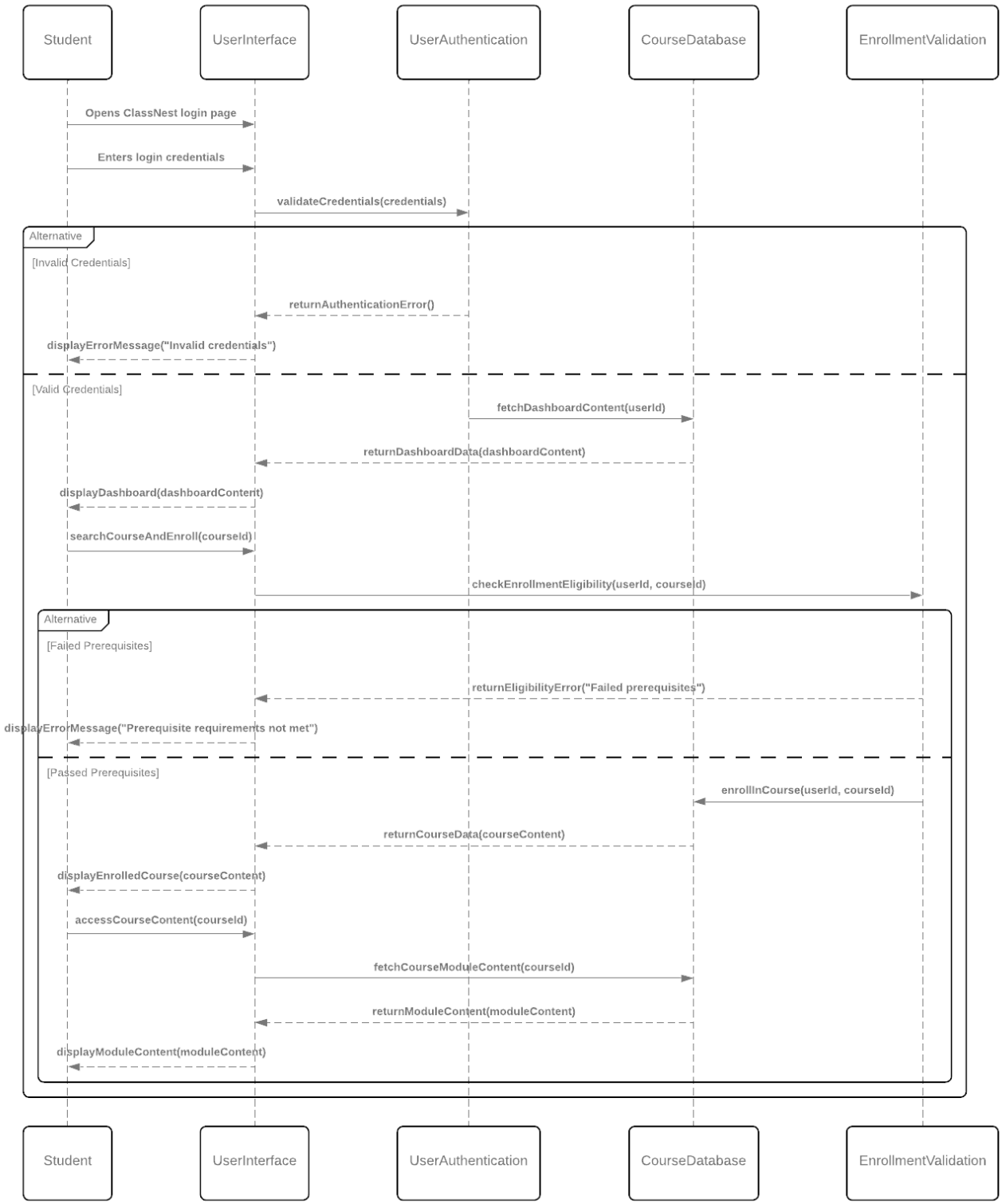
Upload Quiz, Submit _Quiz and Grading are individual sub classes, following SRP design principle under Quiz Management and Grading.

3. DETAILED DESIGN BEHAVIOR

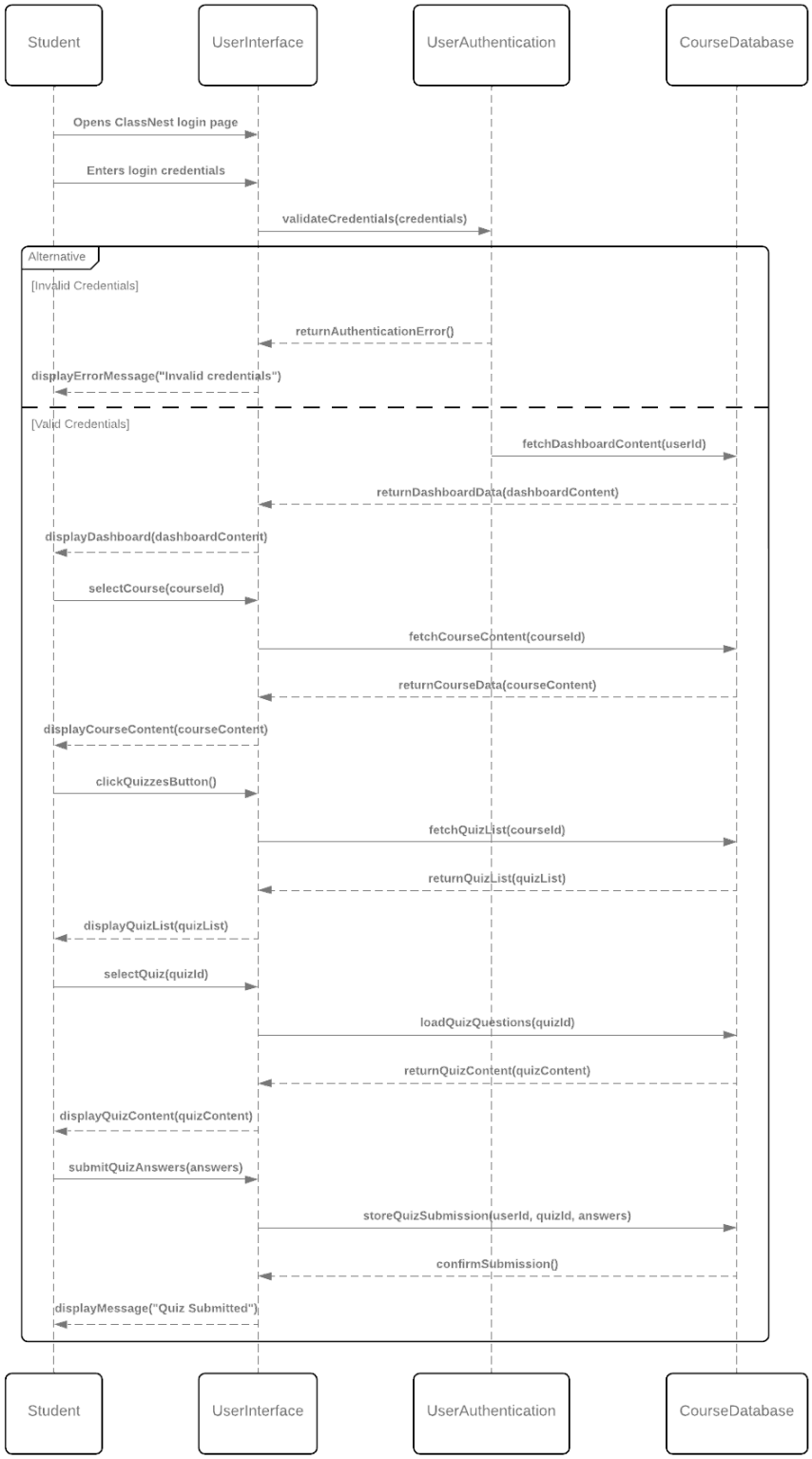
Enrolling in a course:



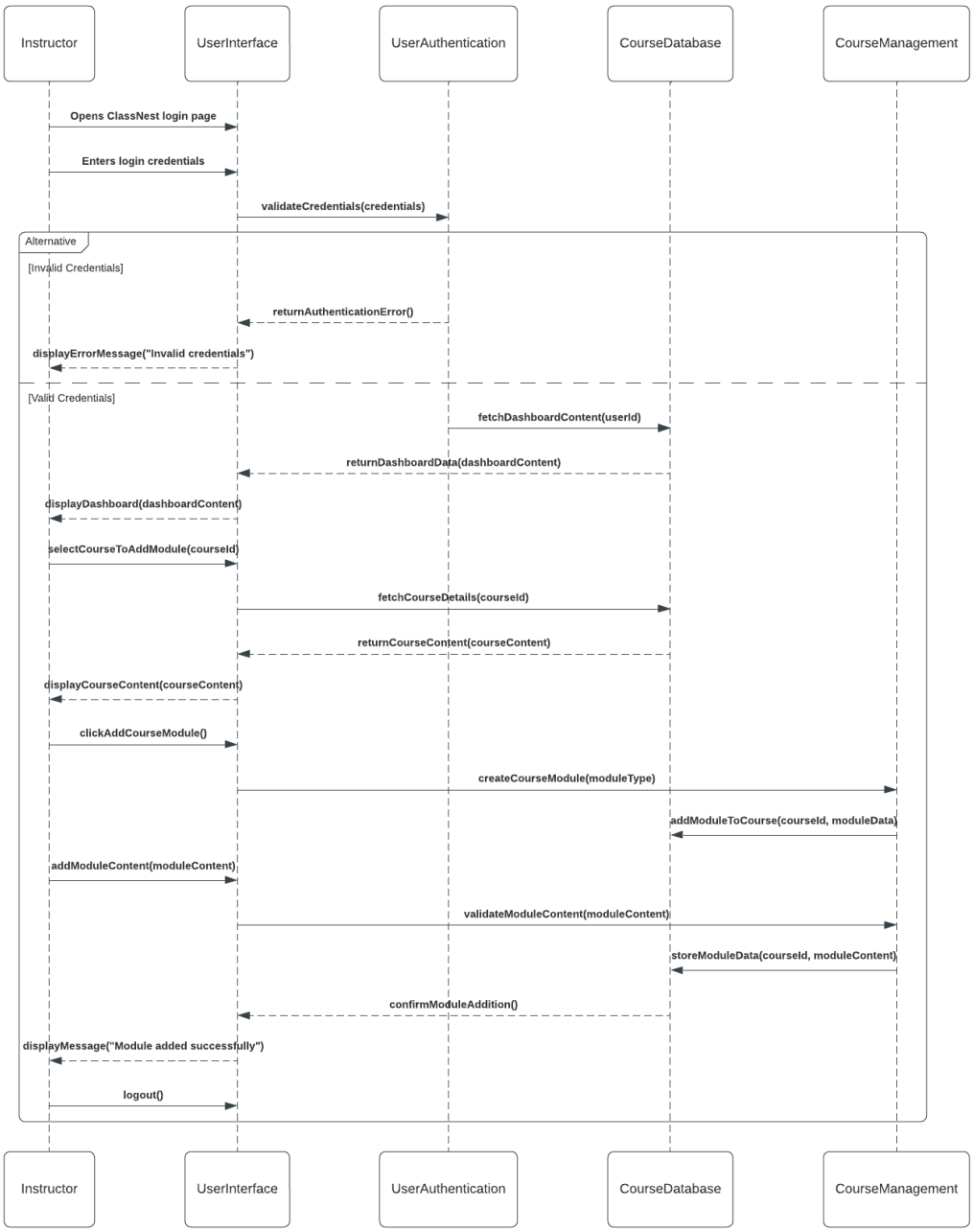
Viewing course content:



Taking a quiz:

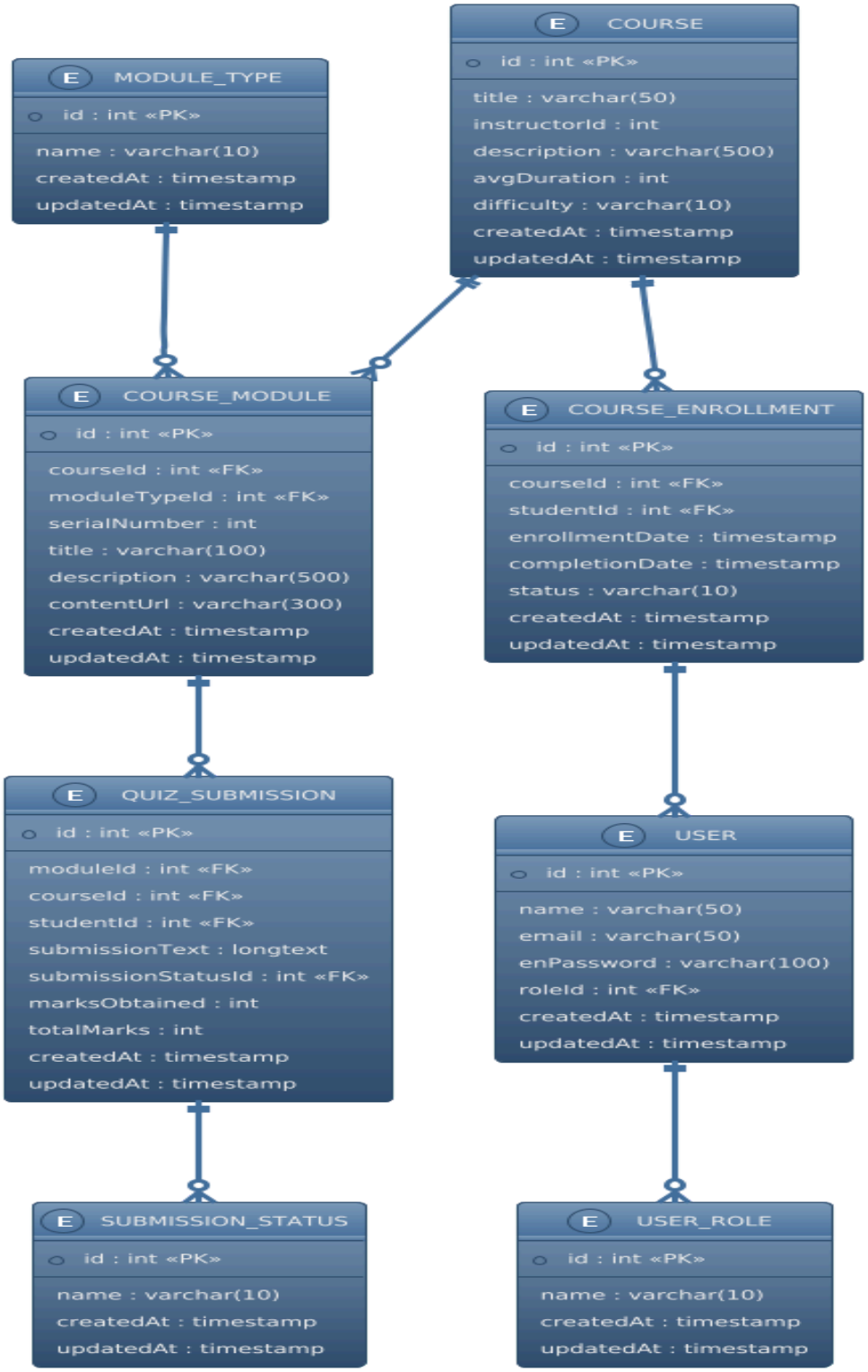


Adding course module:



Student feedback & progress track:

4. PHYSICAL DATA MODEL



ER Diagram Description for the Database for ClassNest

The ER Diagram provides a structured Information View of the **ClassNest** database, emphasizing efficient data organization and relationships.

Entities:

1. **`USER`**
   This entity stores information about the system users.
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each user.
   - **`name`**: Name of the user.
   - **`email`**: Email address of the user.
   - **`enPassword`**: Encrypted password of the user.
   - **`roleId`** (**Foreign Key**): References the **`id`** attribute of the **`USER_ROLE`** entity.
   - **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update
2. **`USER_ROLE`**
   This entity defines the roles a user can have (e.g., Student, Instructor).
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each role.
   - **`name`**: Name of the role.
   - **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update
3. **`COURSE`**
   This entity stores information about courses offered in the system.
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each course.
   - **`title`**: Title of the course.
   - **`instructorId`** (**Foreign Key**): References the **`id`** attribute of the **`USER`** entity.
   - **`description`**: Description of the course.
   - **`avgDuration`**: Average duration of the course (in hours).
   - **`difficulty`**: Difficulty level of the course (e.g., Easy, Medium, Hard).
   - **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update
4. **`COURSE_ENROLLMENT`**
   This entity stores the enrollment details of users in various courses.
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each enrollment.
   - **`courseId`** (**Foreign Key**): References the `id` attribute of the **`COURSE`** entity.
   - **`studentId`** (**Foreign Key**): References the `id` attribute of the **`USER`** entity.
   - **`enrollmentDate`**: Date of enrollment.
   - **`completionDate`**: Date of course completion.
   - **`status`**: Status of the enrollment (e.g., Active, Completed).

- **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update

5. **`COURSE_MODULE`**
   This entity represents the different modules within a course.
   Attributes:
   - **`id`** (**Primary Key)**: Unique identifier for each module.
   - **`courseId`** (**Foreign Key):** References the `id` attribute of the **Entities and Attributes**
6. **`MODULE_TYPE`**
   This entity defines the various types of modules (e.g., Quizzes, Study Materials).
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each module type.
   - **`name`**: Name of the module type.
   - **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update
7. **`QUIZ_SUBMISSION`**
   This entity stores students' quiz submissions.
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each quiz submission.
   - **`moduleId`** (**Foreign Key**): References the `id` attribute of the `COURSE_MODULE` entity.
   - **`courseId`** (**Foreign Key**): References the `id` attribute of the `COURSE` entity.
   - **`studentId`** (**Foreign Key**): References the `id` attribute of the `USER` entity.
   - **`submissionText`**: Text of the quiz submission.
   - **`submissionStatusId`** (**Foreign Key**): References the `id` attribute of the `SUBMISSION_STATUS` entity.
   - **`marksObtained`**: Marks scored by the student.
   - **`totalMarks`**: Total marks for the quiz.
   - **`createdAt`, `updatedAt`**: Timestamps for record creation and update.
8. **`SUBMISSION_STATUS`**
   This entity defines the statuses for quiz submissions (e.g., Not Graded, Graded).
   Attributes:
   - **`id`** (**Primary Key**): Unique identifier for each status.
   - **`name`**: Name of the status.
   - **`createdAt`**, **`updatedAt`**: Timestamps for record creation, and update

Relationships:

1. **USER and USER_ROLE**
    - A user can have only one role.
    - A role can be assigned to many users.
2. **USER and COURSE_ENROLLMENT**
    - A student can enroll in many courses.
    - A course enrollment is associated with only one student.
3. **USER and COURSE**
    - An instructor can teach multiple courses.
    - A course is taught by only one instructor.
4. **USER and QUIZ_SUBMISSION**
    - A student can have multiple quiz submissions.
    - A quiz submission is associated with only one student.
5. **COURSE and COURSE_ENROLLMENT**
    - A course can have multiple enrollments.
    - An enrollment is associated with only one course.
6. **COURSE and QUIZ_SUBMISSION**
    - A course can have multiple quiz submissions.
    - A quiz submission is associated with only one course.
7. **QUIZ_SUBMISSION and SUBMISSION_STATUS**
    - A quiz submission has one status.
    - A status can be associated with multiple quiz submissions.
8. **COURSE_MODULE and QUIZ_SUBMISSION**
    - A module can have multiple quiz submissions.
    - A quiz submission is associated with only one module.
9. **COURSE and COURSE_MODULE**
    - A course can have multiple modules.
    - A module is associated with only one course.
10. **MODULE_TYPE and COURSE_MODULE**
    - A module can have only one type.
    - A type of module can be associated with many course modules

5. ALGORITHMS ENCRYPTION

Algorithm Used for Encryption - AES

Implementing AES encryption for ClassNest provides several key benefits, especially for a platform dedicated to secure data handling in education:

1. **User Trust and Privacy Assurance:** ClassNest deals with sensitive data such as student profiles, progress reports, and other personal details. AES encryption helps secure this information, building trust among educators, students, and guardians who rely on the platform for a safe learning environment.
2. **Broad Compatibility and Adaptability:** Given ClassNest's tech stack of HTML, CSS, JavaScript, Django, and SQL/PostgreSQL, AES is ideal because it seamlessly integrates across web applications and various database systems, without compromising on performance or security.
3. **Alignment with Educational Data Security Standards:** AES encryption is widely recognized as a best practice for protecting sensitive information. Using AES in ClassNest demonstrates a commitment to established data security standards, essential in educational platforms where privacy and security are paramount.
4. **Support for Personalized Learning Experiences:** ClassNest offers a personalized learning experience, which involves securely handling individual user data. AES encryption enables safe storage and retrieval of data, allowing ClassNest to deliver tailored educational resources to its users without compromising confidentiality.
5. **Reliability and Long-Term Data Security:** As ClassNest grows and accumulates more data, ensuring the integrity and security of stored information becomes increasingly important. AES provides a reliable, long-term solution for securing sensitive data, scaling with the platform's growth.

This encryption approach with AES not only reinforces the security of ClassNest but also aligns with industry standards and enhances user confidence by protecting their educational data effectively.

6. ARCHITECTURE TO DETAILED DESIGN TRACING

Part 1: Mapping Components and Interfaces										
Component/Interface	**User Data**	**User Service**	**User Model**	**User Role**	**Authentication**	**Authorization**	**Course Content**	**Course Service**	**Enrollment**	**Assignment**
User Management Service	X	X	X	X	X	X				
Course Content Management Service							X	X		
Assignment Management Service										X
Enrollment Management Service				X					X	
User Interface	X	X		X	X	X	X	X	X	X

Part 2: Mapping Interfaces to User Roles and Functionalities							
Component/Interface	**Student Interface**	**Instructor Interface**	**Admin Interface**	**Content Upload**	**Content Access**	**Submission**	**Grading**
User Management Service	X	X	X				
Course Content Management Service		X		X	X		
Assignment Management Service	X	X				X	X
Enrollment Management Service	X	X	X				
User Interface	X	X	X	X	X	X	X

In ClassNest, our Architecture to Detailed Design Tracing tables map key components to their detailed functionalities. In the first table, we show that each service—like User Management, Course Content, and Enrollment—handles specific responsibilities. For example, the User Management Service oversees authentication, roles, and access control, while Course Content Management focuses on delivering materials.

The second table connects these services to user roles and core functionalities. For instance, students, instructors, and admins interact with tailored interfaces, each accessing content, submissions, and grading as needed. This clear mapping supports modularity and keeps each service focused, ensuring a seamless user experience and efficient data handling