|  |
| --- |
| Luka Nemsitsveridze  10/27/2024 |

|  |
| --- |
|  |
| **Online Course Management System (OCMS)** |
| Technical document |

#### Table of Contents

[Change History](#_Toc156899696)

[1. Introduction](#_Toc156899697)

[2. Roles](#_2._Roles)

[3. Methods](#_2.__მეთოდები)

[4. Used Technologies](#_Toc156899698)

[5. Configuration File](#_Toc156899699)

# History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Comment |
| 1.0.0 | Luka Nemsitsveridze | 10/27/2024 | Initial Version |

# 1. INTRO

The **Online Course Management System (OCMS)** is a platform designed to manage various aspects of online education, enabling users to handle courses, reviews, and authentication securely and efficiently

# 2. Roles

### Student

 **Purpose**: Represents users who are enrolled in courses and want to participate in learning activities.

 **Permissions**:

* Enroll in courses.
* View course content, lessons, and resources.
* See course rating and popularity.
* Provide feedback on courses (e.g., leaving reviews and ratings).

### Instructor

 **Purpose**: Represents users responsible for creating and managing course content, grading, and interacting with students

 **Permissions**:

 Create and manage courses, including adding lessons, assignments, and assessments.

 Update and delete courses they have created.

 View student progress and performance analytics within their courses.

# 3. Methods

Here’s a breakdown of each method in the **Online Course Management System (OCMS)**, detailing the request paths, HTTP methods, expected responses, and behavior:

### AuthenticationController

1. **Register User**
   * **Path**: /auth/register
   * **HTTP Method**: POST
   * **Request Body**: UserDTO (JSON containing username, password, and optional roles)
   * **Response**:
     + 201 Created: if the user is registered successfully.
     + 400 Bad Request: if there’s an issue with the request data.
   * **Description**: Registers a new user based on the provided credentials and roles.
2. **Authenticate User**
   * **Path**: /auth/authenticate
   * **HTTP Method**: POST
   * **Request Body**: UserDTO (JSON containing username and password)
   * **Response**:
     + 200 OK: if authentication is successful, with an AuthenticationResponse containing a JWT token.
     + 401 Unauthorized: if the credentials are invalid.
   * **Description**: Authenticates an existing user and returns a JWT token for secure access to other endpoints.

### EntityController

1. **Get All Entities**
   * **Path**: /entities
   * **HTTP Method**: GET
   * **Response**:
     + 200 OK: List of Entity objects (JSON) representing all reviews.
   * **Description**: Retrieves all entities available in the system.
2. **Get Entity by ID**
   * **Path**: /entities/{id}
   * **HTTP Method**: GET
   * **Path Variable**: id (Long - review identifier)
   * **Response**:
     + 200 OK: Entity object (JSON) representing the review with the specified ID.
     + 404 Not Found: if the entity with the given ID is not found.
   * **Description**: Retrieves a specific entity by its ID.
3. **Create Review**
   * **Path**: /entities
   * **HTTP Method**: POST
   * **Request Body**: Entity (JSON)
   * **Response**:
     + 201 Created: if the Entity is created successfully.
     + 400 Bad Request: if there’s an issue with the request data.
   * **Description**: Adds a new Entity.
4. **Update Entity**
   * **Path**: /entities/{id}
   * **HTTP Method**: PUT
   * **Path Variable**: id (Long - review identifier)
   * **Request Body**: Review (JSON with updated Entity details)
   * **Response**:
     + 200 OK: if the entity is updated successfully.
     + 404 Not Found: if the entity with the given ID is not found.
   * **Description**: Updates an existing entity with new details.
5. **Delete Entity**
   * **Path**: /reviews/{id}
   * **HTTP Method**: DELETE
   * **Path Variable**: id (Long - review identifier)
   * **Response**:
     + 200 OK: if the entity is deleted successfully.
     + 404 Not Found: if the entity with the given ID does not exist.
   * **Description**: Deletes an existing entity by ID.
6. **Search Entity by Criteria**
   * **Path**: /entities/search
   * **HTTP Method**: GET
   * **Query Parameters**:
     + Column1 (String) - optional
     + Column2 (String) - optional
   * **Response**:
     + 200 OK: Set of Entity objects (JSON) matching the provided criteria.
     + 400 Bad Request: if the search criteria are invalid.
   * **Description**: Searches for Entities based on optional criteria of the specific entity Classes, some of which can be searched or grouped by other entities
7. **Entity Examples** 
   * CourseCategory
   * Course
   * Lesson
   * Enrollment
   * Review

# 4. Used Technologies

* OCMS is API written on Java 21.
* OCMS RESTful API was built using Spring Boot 3.3.4
* Maven 3 is used as build too
* API supports Embeded Tomcat and can be started without any configuration
* Log4j2 library is used for logging request information.
* Json Web Token for authentication
* H2 database

# 5. Configuration File

application.properties – file has database credentials for connection, and properties for JWT Authentication

|  |
| --- |
| Conf/application.properties |
| spring.application.name=Online Course Management System  spring.datasource.url=jdbc:h2:mem:coursemanagement spring.datasource.driverClassName=org.h2.Driver spring.datasource.username=username spring.datasource.password=password spring.jpa.database-platform=org.hibernate.dialect.H2Dialect spring.h2.console.enabled=true  jwt.secret=b7acf8204b85ab7cec6e73b91f682677d642d328fdec7788e8cbff11ff1a5b22 jwt.expiration.hours=24  logging.level.ge.OCMS=TRACE |
|  |