**Project Title: OnlineLearnHub**

**Project Description:**  
OnlineLearnHub is a simplified online learning platform developed using ASP.NET Core and SQLite. It enables instructors to create and manage courses while allowing students to enroll in those courses. The platform employs ASP.NET Identity for user authentication, supporting distinct roles for Admin, Instructor, and Student.

**Architecture Overview:**

* **Frontend:** ASP.NET Razor Views (MVC Pattern)
* **Backend:** ASP.NET Core MVC and REST API
* **Database:** SQLite (lightweight, serverless, file-based database)
* **Authentication:** ASP.NET Core Identity for role-based authentication

**Technologies Used:**

* **Backend:** ASP.NET Core MVC
* **Frontend:** Razor Pages (HTML, CSS, Bootstrap)
* **Database:** SQLite (local file database)
* **Authentication:** ASP.NET Identity (role-based authentication)

**Challenges Faced:**

* **Database Selection:** While SQLite was chosen for its simplicity, it presented challenges with scaling and executing complex queries. For production environments, other databases like SQL Server or MySQL may be more appropriate.
* **API Creation:** I encountered difficulties in creating a fully functioning API with secure authentication due to limited technical expertise. However, leveraging ASP.NET MVC with Razor pages simplified interactions between the frontend and backend. The Enrollment API specifically posed challenges.
* **Identity Integration:** Integrating ASP.NET Identity with a custom User model was complex. The solution involved extending the IdentityUser class and configuring appropriate relationships, which required extensive research.
* **Data Relationships:** Managing the complex relationships between Courses and Enrollments was challenging, requiring a solid understanding of Entity Framework's navigation properties.
* **Scaffolding Issues:** The initial days of the project involved resolving errors during scaffolding, which took approximately three days. This was a significant learning curve and impacted the overall timeline.
* **Coding Challenges:** As a non-technical person, I found it difficult to write code, leading to instances of borrowing snippets from online resources. This raised concerns about the integrity and understanding of the codebase.
* **Controller Issues:** There are existing issues with the Enrollment Controller, where the API is not functioning as intended. Due to time constraints, I opted not to modify the code further to avoid introducing additional errors.

**Improvements for the Future:**

* **API Enhancements:** With more time, I would enhance the API by adding more detailed endpoints and implementing better separation of concerns.
* **Error Handling:** Implementing comprehensive error handling and logging to capture issues dynamically would improve the robustness of the application.
* **Database Migration:** Exploring migration to a more robust database system like SQL Server or PostgreSQL could enhance scalability.

**Access Points:**

* **Web Application:** <https://localhost:5124>
* **Swagger UI:** <https://localhost:5124/swagger>
* **API Base URL:** <https://localhost:5125/api/courseApi>

**Required Packages:**

dotnet add package Microsoft.AspNetCore.Mvc

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.Sqlite

dotnet add package Microsoft.EntityFrameworkCore.Tools

dotnet add package Microsoft.AspNetCore.Identity

dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore

dotnet add package Swashbuckle.AspNetCore

dotnet add package Microsoft.Extensions.Configuration.Json

**How to Set Up Locally:**

1. Ensure the .NET Core SDK is installed to run this project.
2. To apply database migrations, run:

dotnet ef database update

dotnet run

Access Swagger at: <http://localhost:5154/swagger/index.html>

API End points: Retrieves a list of all courses.

http://localhost:5154/api/coursesApi

**GET /api/coursesApi**: Retrieves a list of all courses.A screenshot of a computer program

Description automatically generated

**GET /api/CoursesApi/{id}:** Retrieves a specific course by ID.

A screenshot of a computer

Description automatically generated

**POST /api/CoursesApi:** Creates a new course.

{

"title": "New Course Title",

"description": "This is a description of the new course",

"createdAt": "2024-10-01T00:00:00",

"updatedAt": "2024-10-01T00:00:00"

}

A screenshot of a computer

Description automatically generated

 **PUT /api/CoursesApi/{id}:** Updates an existing course.

A black and white striped background

Description automatically generated

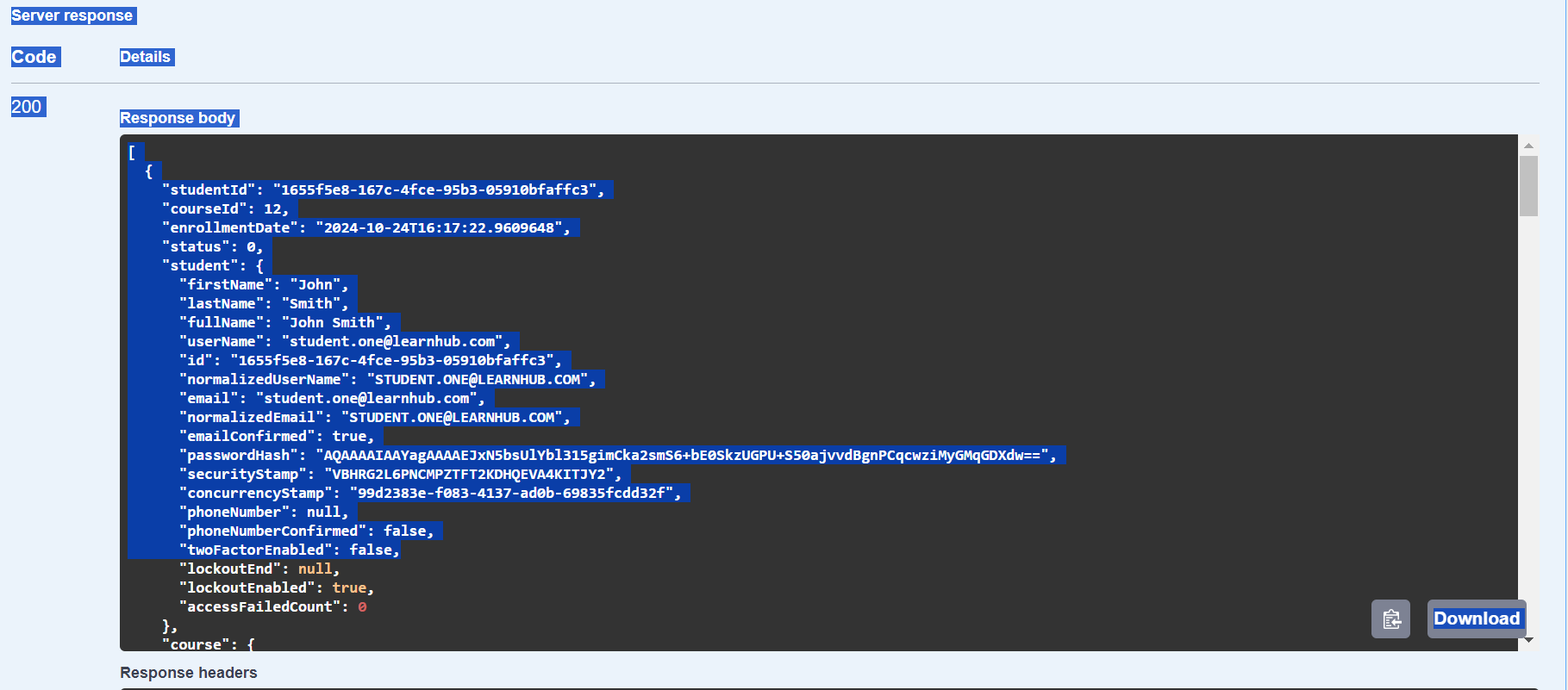
**DELETE /api/CoursesApi/{id}:** Deletes a course.

A screenshot of a computer

Description automatically generated

API End points: Retrieves a list of all courses.

**GET: http://localhost:5154/api/enrollmentsApi**



A black and white striped background

Description automatically generated

Get: <http://localhost:5154/api/enrollmentsApi/student/5b49b92a-cbcc-4bad-8390-dc59045fe70f/course/2>

A screenshot of a computer

Description automatically generated