## Authentication & Authorization in Dotnet

**Session – 1**

### Question1: Authentication and Authorization in a Document Management System, implementing Basic Authentication

You are tasked with developing an online course management system for a university. The system must authenticate users using Basic Authentication Schema, authorize access to courses based on roles, and track course activities for compliance and auditing purposes.

Create a database named **BasicAuthDb.**

Design and implement two essential endpoints for the system:

**1. Login Endpoint:**

* Create a POST endpoint at **'/api/login'** to handle user authentication.
* Accept JSON payload containing the user's email and password.
* Validate the incoming payload to ensure it is well-formed.
* Authenticate the user by verifying the provided credentials against the database.
* If authentication is successful, return a success message with the user details.
* If authentication fails, return an appropriate error message.

**2.Course Endpoint:**

* Create a POST endpoint at **/api/courses/create** to allow authorized users to add new courses to the system.
* Require authentication, meaning only authenticated users can access this endpoint.
* Validate the incoming payload to ensure all required fields are provided.
* Check if the course already exists to prevent duplicate entries (optional).
* Create a new course record in the database with the provided details.
* Return a success message along with the details of the newly created course.
* Ensure that the course creation endpoint is authorized only for authenticated users with specific roles, such as "**Admin**" or "**CourseManager**".

**User Login - /api/login - POST**

**Payload:**

{

"email": "user@example.com",

"password": "password123"

}

**Course - /api/courses/create - POST**

**Payload:**

{

"courseId": 101,

"courseName": "Introduction to Machine Learning",

"instructor": "Dr. Jane Smith",

"credits": 3,

"semester": "Fall 2024",

"description": "An introductory course on machine learning covering basic algorithms and practical implementations."

}

## Role Based Authorization, implementing Asp.Net Identity

### Exercise 1: Plants Management System

**Implement a Plants Management System using Authentication and Authorization.**

Implement a Plants Management System that allows a Plants Shop owner to post the plants available in their shop, enabling customers to view and purchase them. Assist the Admin in reaching out to customers by providing an intuitive platform for managing plants and user accounts.

**Model Classes**

Create Models folder inside dotnetapp. Inside the model folder create all the classes mentioned below.

**User Class:**

This class stores the user type (**Admin or the Customer**) and all user information.

User Class has the below properties:

* UserId (long) (auto - increments by 1)
* Email (string)
* Password (string)
* Username (string)
* MobileNumber (string)
* UserRole (string)

1. Represents a user in the system with attributes such as UserId, Email, Password, Username, MobileNumber, and UserRole. (UserId is Auto Incremented)

2. Intended for user authentication and authorization.

**Login Model Class:**

* Email (string)
* Password (string)

**Plant Class:**

* PlantId (int) (auto - increments by 1)
* Name (string)
* ScientificName (string)
* Description (string)
* Price (double)

**ApplicationDbContext Class:**

* Create ApplicationDbContext Class for the below-mentioned model class inside the Model class:

1. Users
2. Plants

* Define a class ApplicationDbContext which inherits from DbContext, this class represents the database context for the application.
* Install the necessary packages like **Microsoft.EntityFrameworkCore.SqlServer, Microsoft.AspNetCore.Identity.EntityFrameworkCore.**

Create a database named **IdentityAuthDb.**

**Endpoints to implemented using AuthenticationController:**

**Login Endpoint (Login() method):**

* Accepts a **POST** request to **/api/login.**
* Validates the incoming payload using ModelState.IsValid.
* Invokes the Login() method of the \_authService to authenticate the user.
* If authentication is successful, returns a JWT token in the response body.
* If there's an error during authentication, returns a BadRequest response with an appropriate message.
* Catches any exceptions that occur during the process and logs them.

**Register Endpoint (Register() method):**

* Accepts a **POST** request to **/api/register.**
* Validates the incoming payload using ModelState.IsValid.
* Checks if the user role is either "Admin" or "Customer".
* Invokes the Registration() method of the \_authService to register the user.
* If registration is successful, adds the user to the database using \_context.Users.Add(user) and saves changes.
* Returns a success message in the response body if registration is successful.
* If there's an error during registration or if the user role is invalid, returns a BadRequest response with an appropriate message.
* Catches any exceptions that occur during the process and logs them.

**Authorization:**

The entire controller is not decorated with [Authorize] attribute, meaning that these endpoints are accessible to unauthenticated users.

However, the actual authentication process occurs within the Login() method by generating a JWT token using the \_authService.

Route:

**The controller is mapped to the route /api.**

**Endpoints to implemented using Plant Controller:**

**GET Request (Get() method):**

* Retrieves all plants from the database asynchronously using \_context.Plants.ToListAsync().
* Returns an Ok response with the list of plants.

**POST Request (Post() method):**

* Adds a new plant to the database.
* Uses [FromBody] attribute to bind the incoming JSON payload to the Plant object.
* Checks if the incoming plant object is null and returns a BadRequest response if it is.
* Adds the plant to the \_context and saves changes to the database asynchronously.
* Returns a CreatedAtAction response with the newly created plant.

**Authorization:**

* The entire controller is decorated with **[Authorize]** attribute, meaning that only authenticated users can access its endpoints.
* The **Post()** method is further restricted to users with the role of "**Admin**" using the **[Authorize(Roles = "Admin")]** attribute.

**Route:**

The controller is mapped to the route **/api/plants.**

**IAuthService** (JWT token should be implemented)

* Registeraton(User model, string role)
* Login(LoginModel model)
* GenerateToken(IEnumerable<Claim> claims)

**PayLoads for post**:

1. User Login - **/api/login** - POST

Payload:

{

"Email": "user@example.com",

"Password": "password123"

}

2. User Register - **/api/register** - POST

Payload:

{

"Email": "newuser@example.com",

"Password": "newpassword123",

"Username": "newuser",

"MobileNumber": "1234567890",

"UserRole": "Admin"

}

3. Add Plants - **/api/plants** - POST

Payload:

{

"plantId": 0,

"name": "Sunflower",

"scientificName": "Helianthus annuus",

"description": "Sunflowers are tall, erect plants with large, bright yellow flowers and edible seeds. They are native to North America and are cultivated worldwide

for their beauty and seeds.",

"price": 10.99

}