

DOCONNECT

CAPSTONE PROJECT

WIPRO NGA - .NET FULL STACK ANGULAR

FY26 – BATCH C2

DOCONNECT

SARAVANAN M

WIPRO NGA - .NET FULL STACK ANGULAR

FY26 – BATCH C2

DOCONNECT

MENTOR

JOYTI S PATIL

WIPRO NGA - .NET FULL STACK ANGULAR

FY26 – BATCH C2

INTRODUCTION

The rise of technology and online learning has increased the need for platforms where people can share knowledge and solve problems together. **DoConnect** is a community-driven Q&A platform, similar to Stack Overflow, built as a capstone project to demonstrate full-stack development skills.

It supports two roles: **User** and **Admin**. Users can register, post questions, give answers, upload images, and search content. Admins moderate by approving/rejecting posts and managing accounts, ensuring quality and security.

PROBLEM STATEMENT

DoConnect is a Q&A platform where users can ask and answer questions related to various technical topics. The application has two types of users:

- ADMIN
- USER

OBJECTIVE

The objective of the DoConnect project is to design and implement a community-driven Question & Answer platform that enables effective knowledge sharing and problem-solving. The system aims to provide secure user participation, efficient content management, and role-based access control while showcasing full-stack development skills through a modern technology stack and best software engineering practices.

SOFTWARE REQUIREMENTS

LANGUAGES USED:

- ANGULAR
- ASP.NET
- C#
- TYPESCRIPT
- SQL

IDE USED:

- VS CODE
- SSMS
- SWAGGER
- GITHUB

HARDWARE REQUIREMENTS

- Processor : Intel Core i5 (or equivalent AMD)
- RAM : 8 GB minimum (16 GB recommended for smooth multitasking with IDEs, DB, and server)
- Storage : 250 GB SSD (for faster build, run, and DB operations)
- Graphics : Integrated graphics sufficient (no heavy GPU required)
- Network : Stable internet connection (for Git, package managers, and API testing)

ANGULAR ARCHITECTURE

- Modules
- Components
- Templates
- Directives
- Services
- Routing
- Data Binding
- HTTP Client
- Dependency Injection

ASP.NET ARCHITECTURE

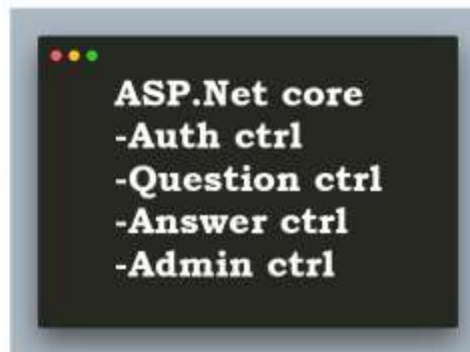
- Presentation Layer
- Business Layer
- Data Access Layer
- Database Layer

PROJECT OVERVIEW DIAGRAM

FRONTEND



BACKEND

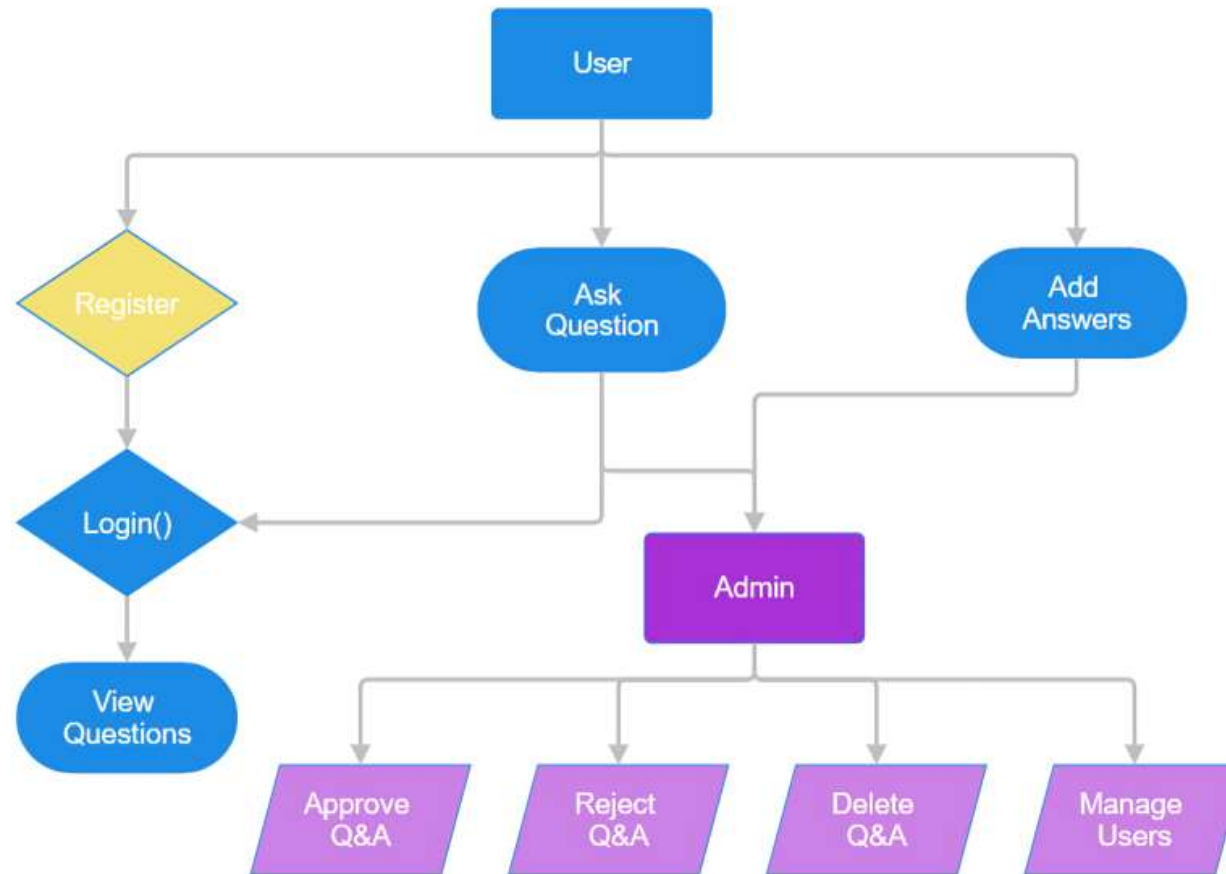


DATABASE

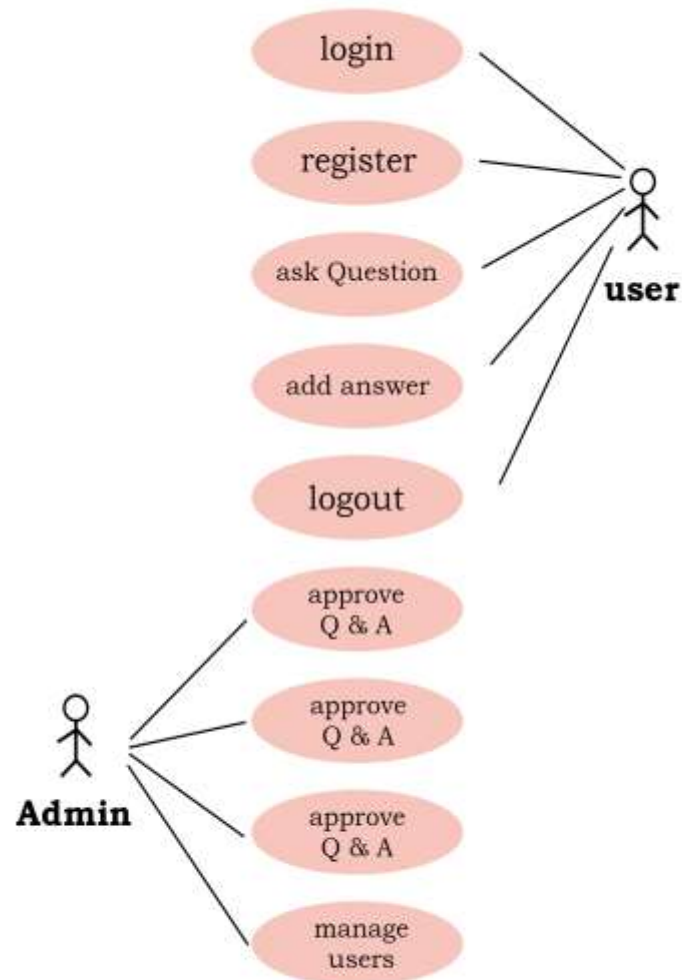


Project Overview Diagram

SYSTEM OVERFLOW DIAGRAM

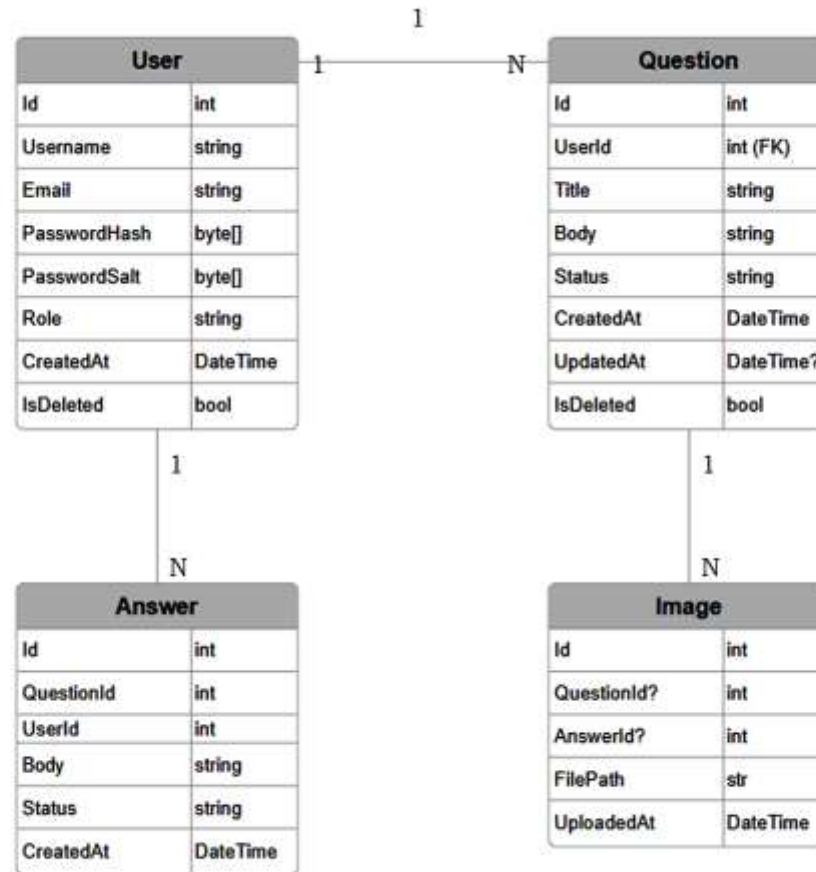


USE CASE DIAGRAM



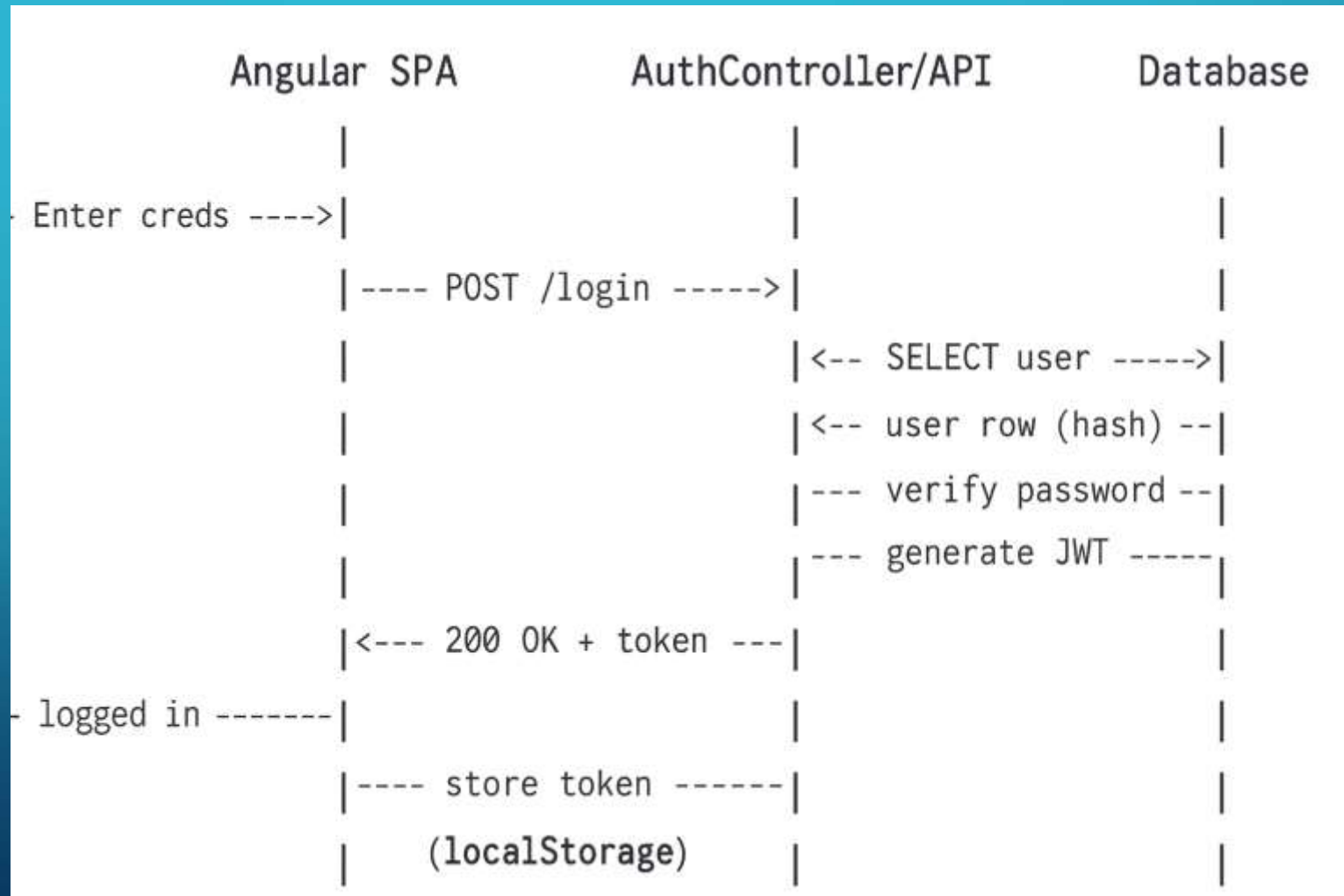
Use Case Diagram

CLASS DIAGRAM

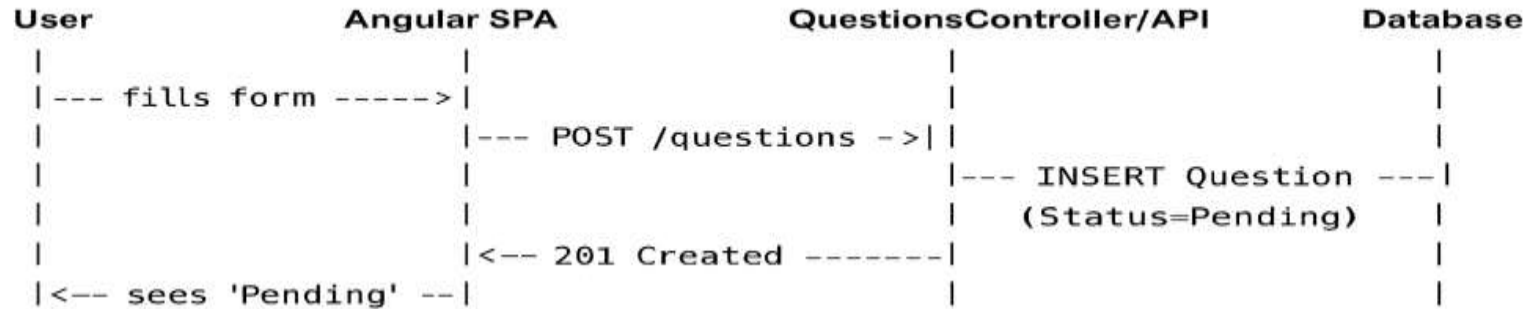


Entity Relationship Diagram

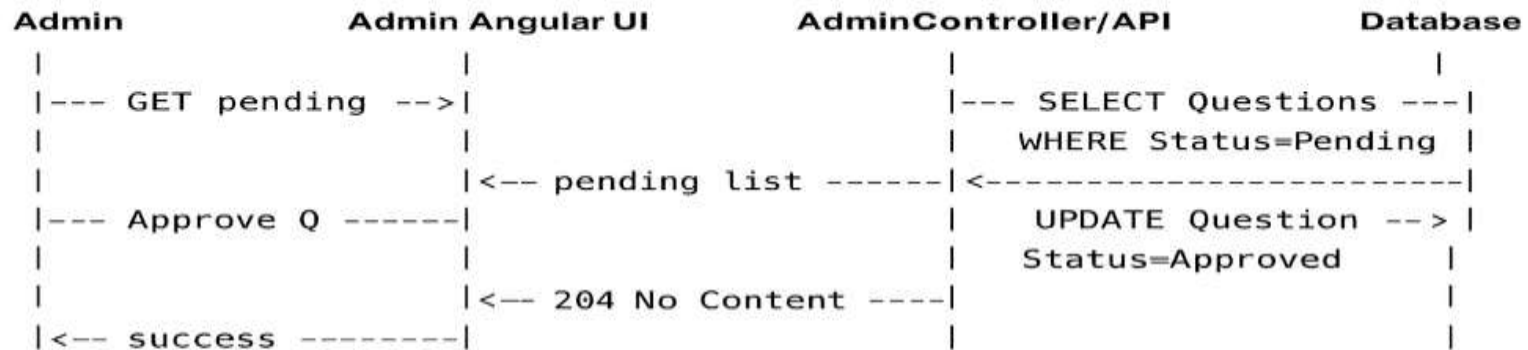
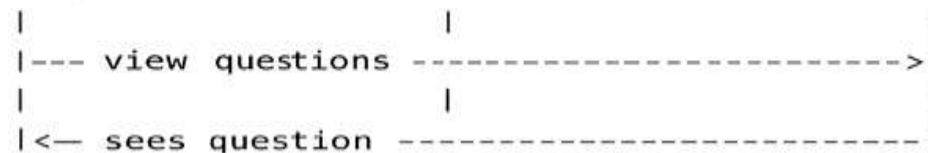
LOGIN FLOW DIAGRAM



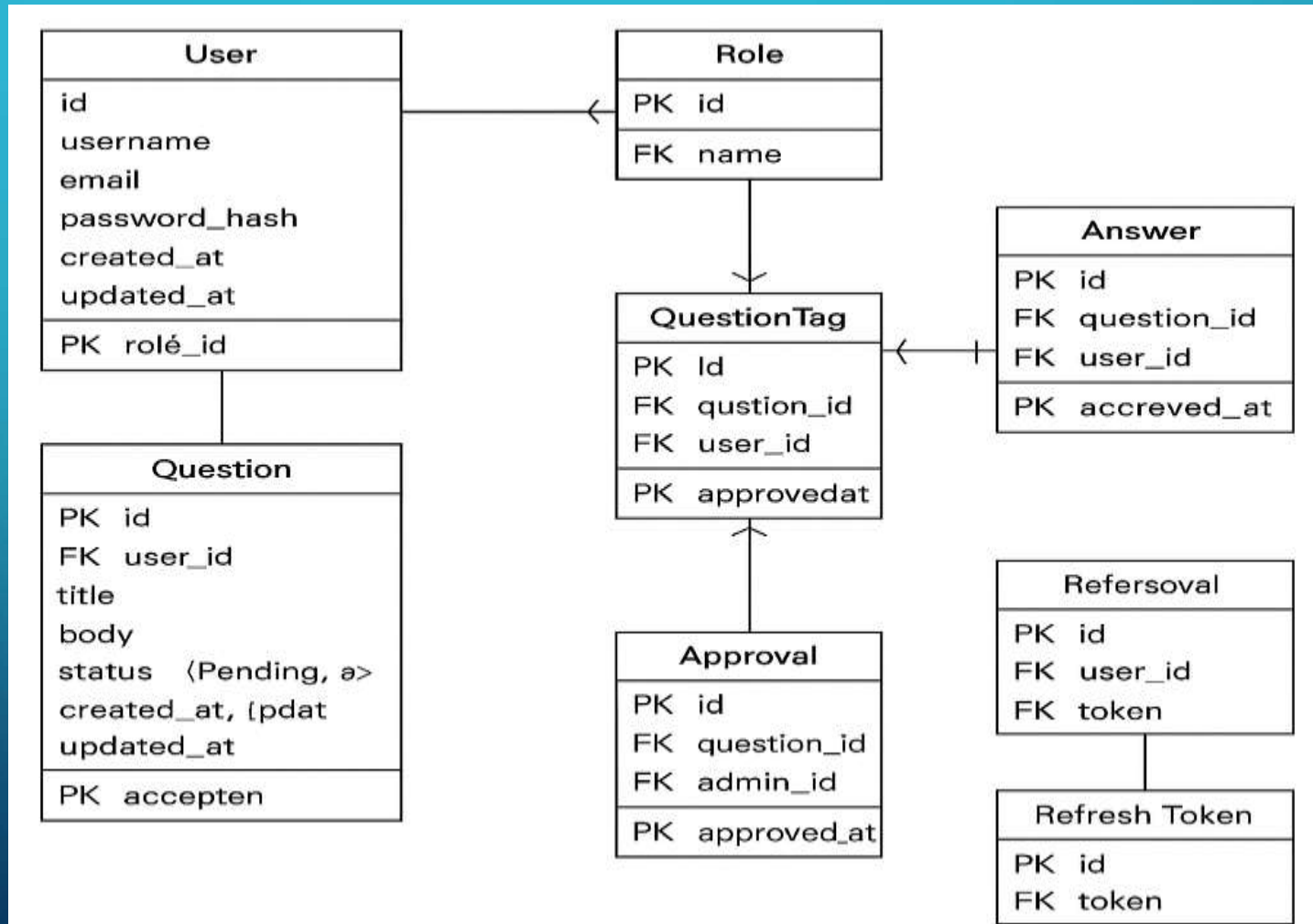
ADMIN FLOW DIAGRAM



(Later...)

**User/All**

ER DIAGRAM

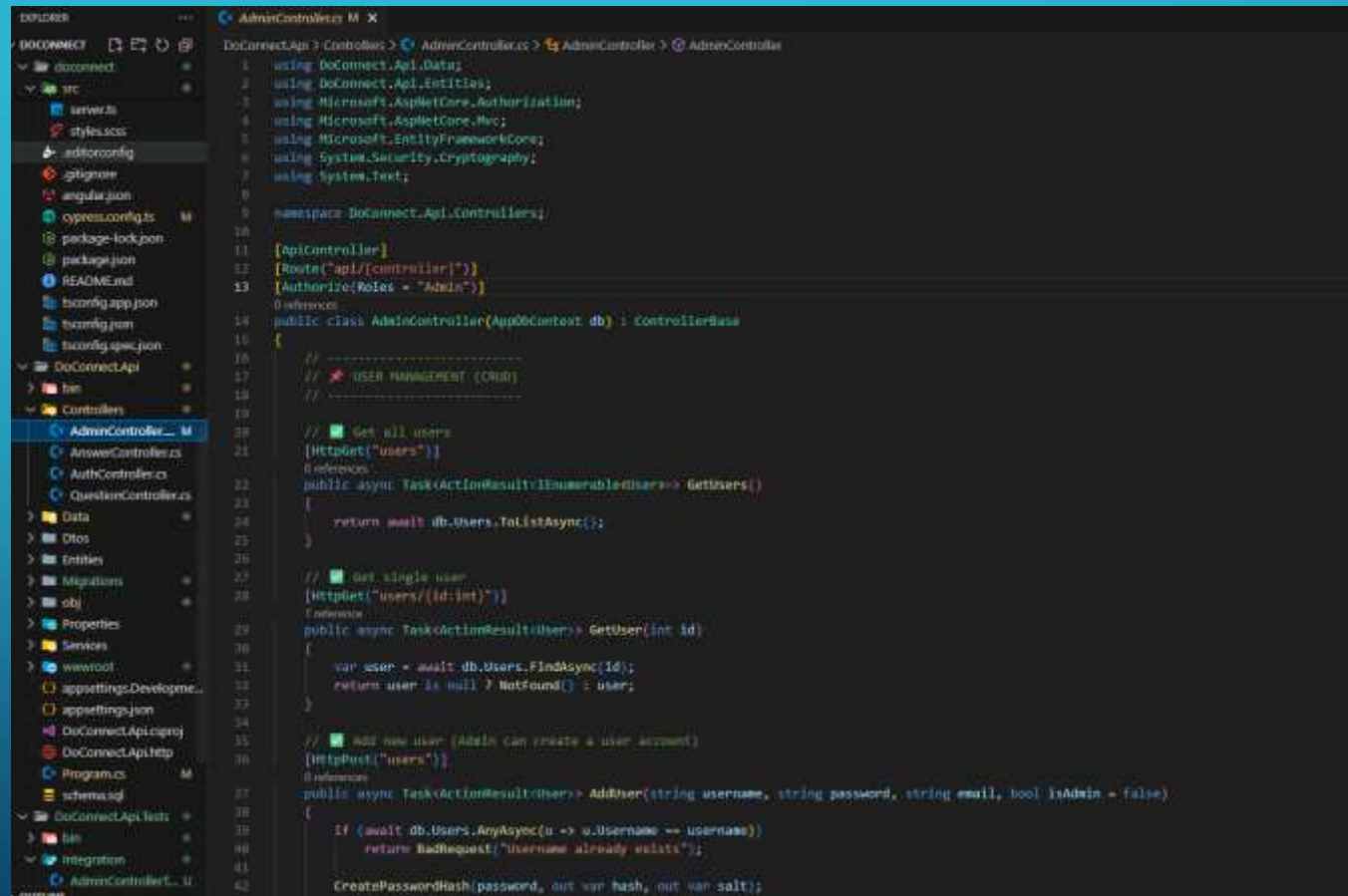


PROJECT SCREENSHOTS

PROJECT SCREENSHOTS

DOCONNECT

FRONTEND CODE

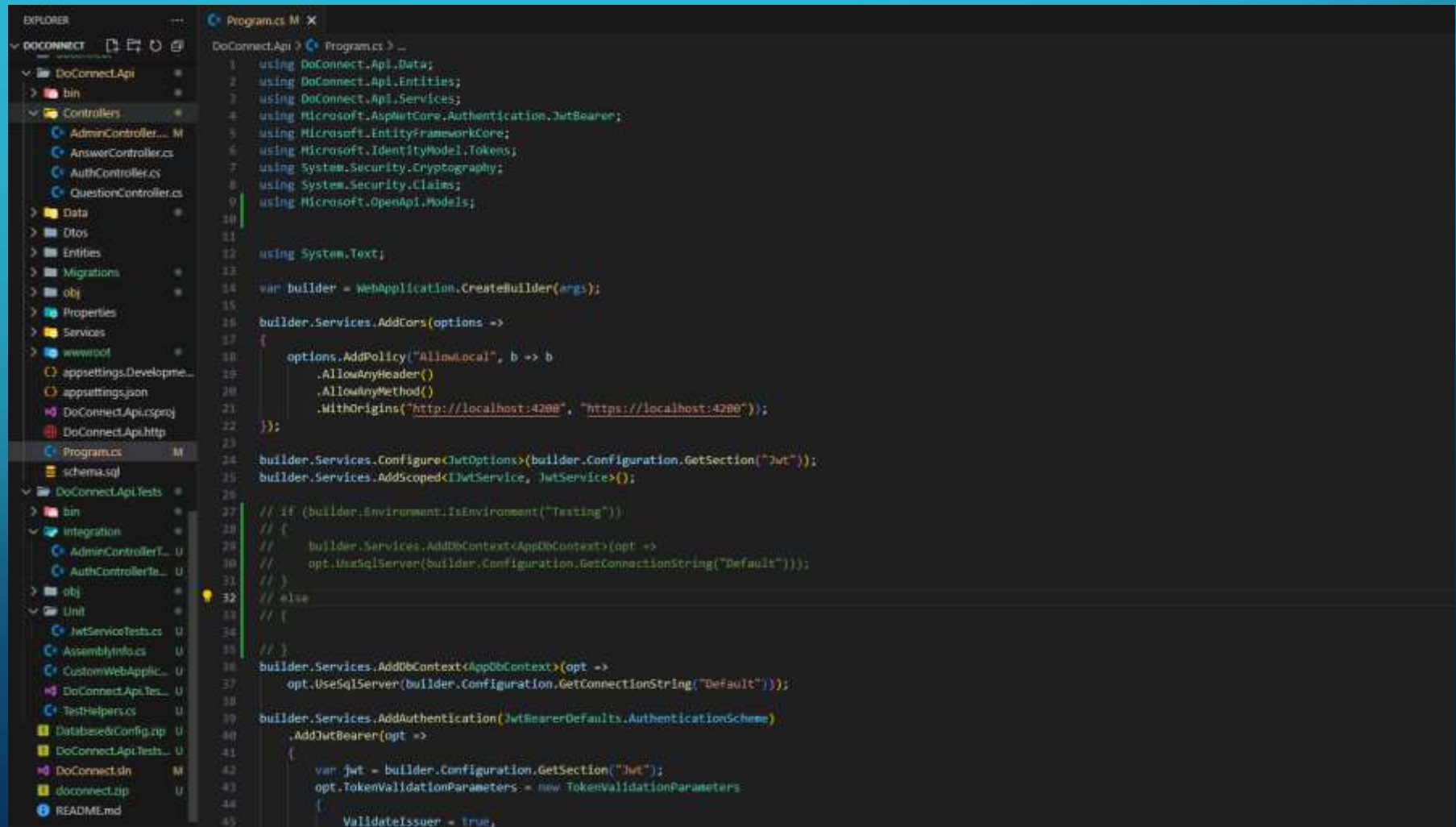


```
1 using DoConnect.Api.Data;
2 using DoConnect.Api.Entities;
3 using Microsoft.AspNetCore.Authorization;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.EntityFrameworkCore;
6 using System.Security.Cryptography;
7 using System.Text;
8
9 namespace DoConnect.Api.Controllers;
10
11 [ApiController]
12 [Route("api/[controller]")]
13 [Authorize(Roles = "Admin")]
14 public class AdminController(ApplDbContext db) : ControllerBase
15 {
16     // -----
17     // * USER MANAGEMENT (CRUD)
18     // -----
19
20     // [GET] Get all users
21     [HttpGet("users")]
22     public async Task GetUsers()
23     {
24         return await db.Users.ToListAsync();
25     }
26
27     // [GET] Get single user
28     [HttpGet("users/{id:int}")]
29     public async Task
```

PROJECT SCREENSHOTS

DOCONNECT

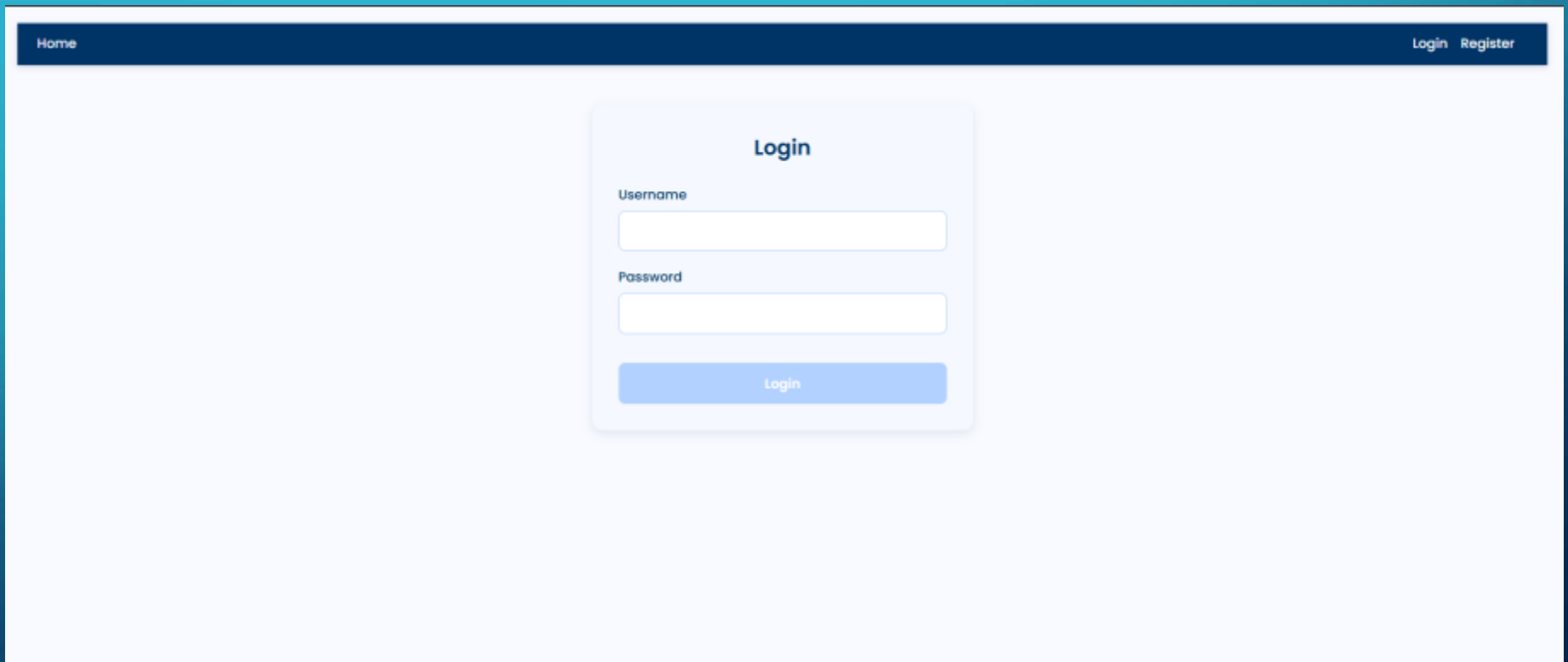
BACKEND CODE



The screenshot displays the Visual Studio IDE interface. On the left, the 'EXPLORER' pane shows the project structure for 'DOCONNECT'. The 'DoConnect.Api' folder is expanded, revealing subfolders like 'bin', 'Controllers', 'Data', 'Dtos', 'Entities', 'Migrations', 'obj', 'Properties', 'Services', and 'wwwroot'. The 'Program.cs' file is selected in the 'Controllers' folder. The main editor area shows the code for 'Program.cs', which is a standard ASP.NET Core startup file. It includes various using statements for namespaces like 'DoConnect.Api.Data', 'DoConnect.Api.Entities', 'DoConnect.Api.Services', 'Microsoft.AspNetCore.Authentication.JwtBearer', 'Microsoft.EntityFrameworkCore', 'Microsoft.IdentityModel.Tokens', 'System.Security.Cryptography', 'System.Security.Claims', and 'Microsoft.OpenApi.Models'. The code then configures the web application builder, adding CORS policies for 'http://localhost:4200' and 'https://localhost:4200', and setting up JWT authentication and database context.

```
1 using DoConnect.Api.Data;
2 using DoConnect.Api.Entities;
3 using DoConnect.Api.Services;
4 using Microsoft.AspNetCore.Authentication.JwtBearer;
5 using Microsoft.EntityFrameworkCore;
6 using Microsoft.IdentityModel.Tokens;
7 using System.Security.Cryptography;
8 using System.Security.Claims;
9 using Microsoft.OpenApi.Models;
10
11
12 using System.Text;
13
14 var builder = WebApplication.CreateBuilder(args);
15
16 builder.Services.AddCors(options =>
17 {
18     options.AddPolicy("AllowLocal", b => b
19         .AllowAnyHeader()
20         .AllowAnyMethod()
21         .WithOrigins("http://localhost:4200", "https://localhost:4200"));
22 });
23
24 builder.Services.Configure<JwtOptions>(builder.Configuration.GetSection("Jwt"));
25 builder.Services.AddScoped<IJwtService, JwtService>();
26
27 // if (builder.Environment.IsEnvironment("Testing"))
28 // {
29 //     builder.Services.AddDbContext<AppDbContext>(opt =>
30 //         opt.UseSqlServer(builder.Configuration.GetConnectionString("Default")));
31 // }
32 // else
33 // {
34 // }
35 // }
36
37 builder.Services.AddDbContext<AppDbContext>(opt =>
38     opt.UseSqlServer(builder.Configuration.GetConnectionString("Default")));
39
40 builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
41     .AddJwtBearer(opt =>
42     {
43         var jwt = builder.Configuration.GetSection("Jwt");
44         opt.TokenValidationParameters = new TokenValidationParameters
45         {
46             ValidateIssuer = true,
```

LOGIN PAGE



The screenshot displays the login interface of the Doconnect application. At the top, a dark blue navigation bar contains a 'Home' link on the left and 'Login' and 'Register' links on the right. The main content area is white and features a central, light blue login card with rounded corners and a subtle shadow. The card is titled 'Login' in bold. It contains two input fields: 'Username' and 'Password', both with yellow backgrounds and blue borders. Below these fields is a blue 'Login' button. The overall design is clean and modern.

Home Login Register

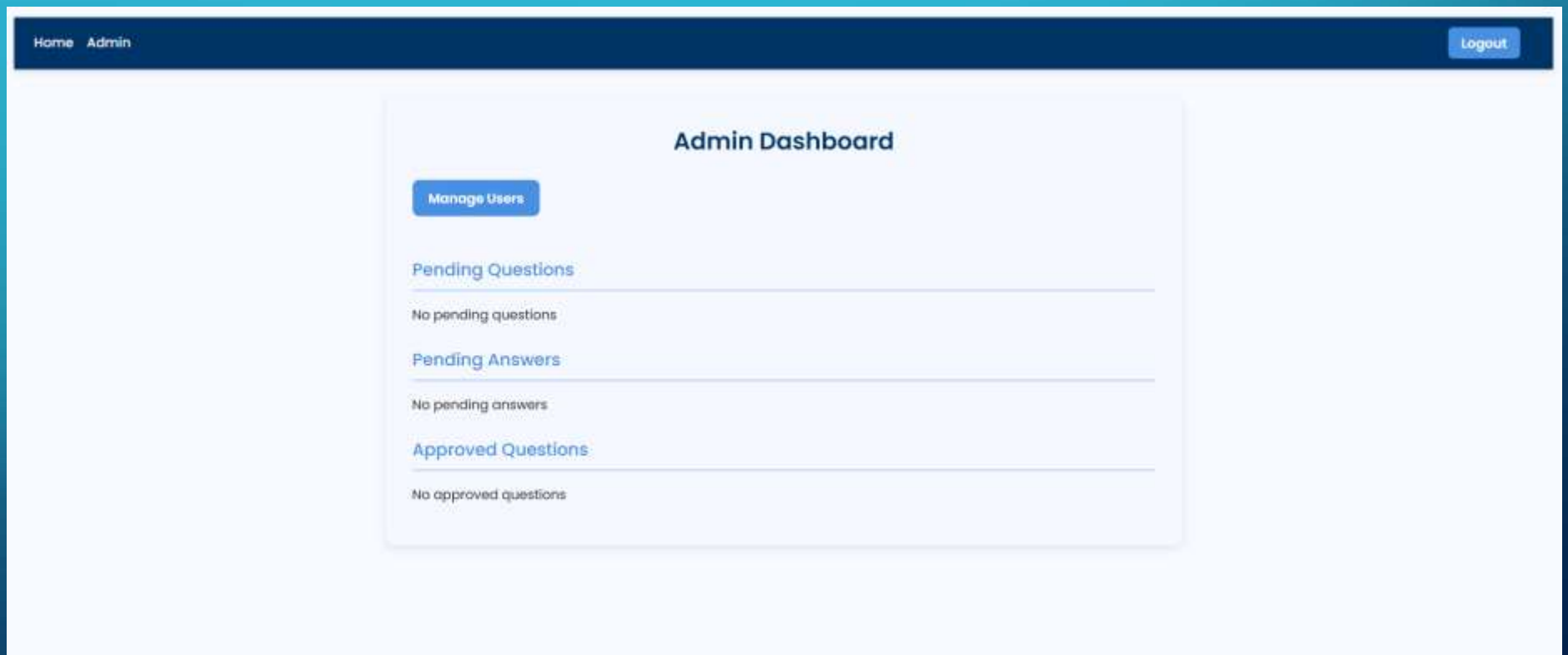
Login

Username

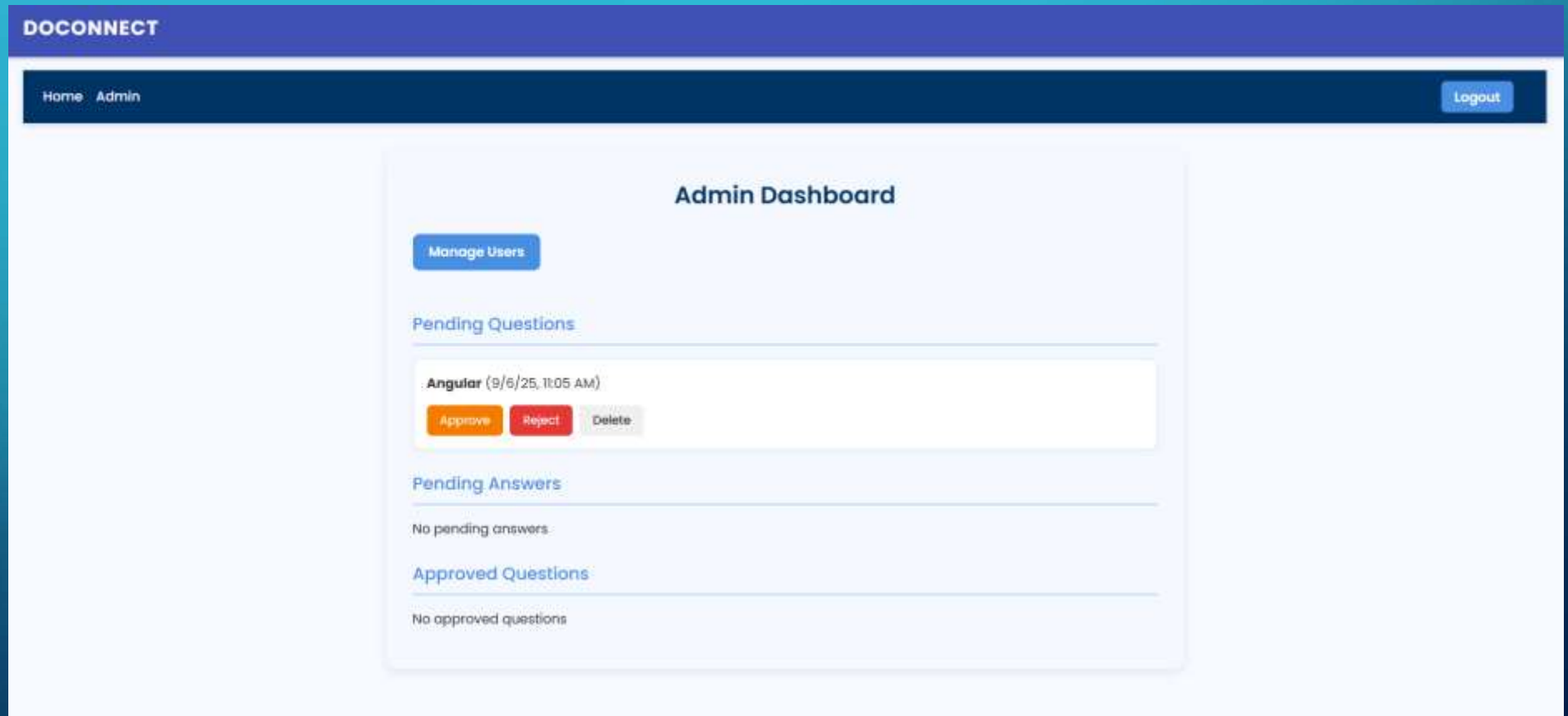
Password

Login

ADMIN DASHBOARD PAGE



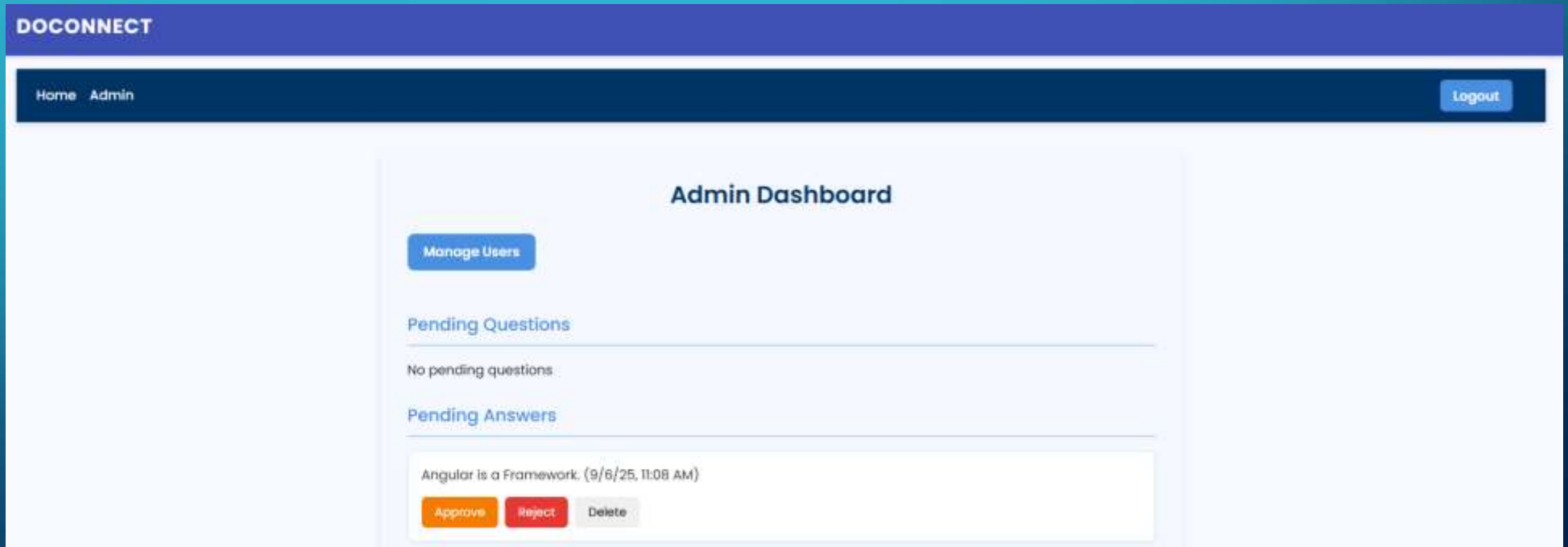
ADMIN ACCEPT/REJECT QUESTION



PROJECT SCREENSHOTS

DOCONNECT

ADMIN ACCEPT/REJECT ANSWER



The screenshot displays the DOCONNECT Admin Dashboard. At the top, there is a purple header with the 'DOCONNECT' logo. Below this is a dark blue navigation bar containing 'Home', 'Admin', and a 'Logout' button. The main content area is titled 'Admin Dashboard' and features a 'Manage Users' button. Under the 'Pending Questions' section, it states 'No pending questions'. The 'Pending Answers' section shows a single entry: 'Angular is a Framework. (9/8/25, 11:08 AM)'. This entry has three buttons: 'Approve' (orange), 'Reject' (red), and 'Delete' (grey).

DOCONNECT

Home Admin Logout

Admin Dashboard

Manage Users

Pending Questions

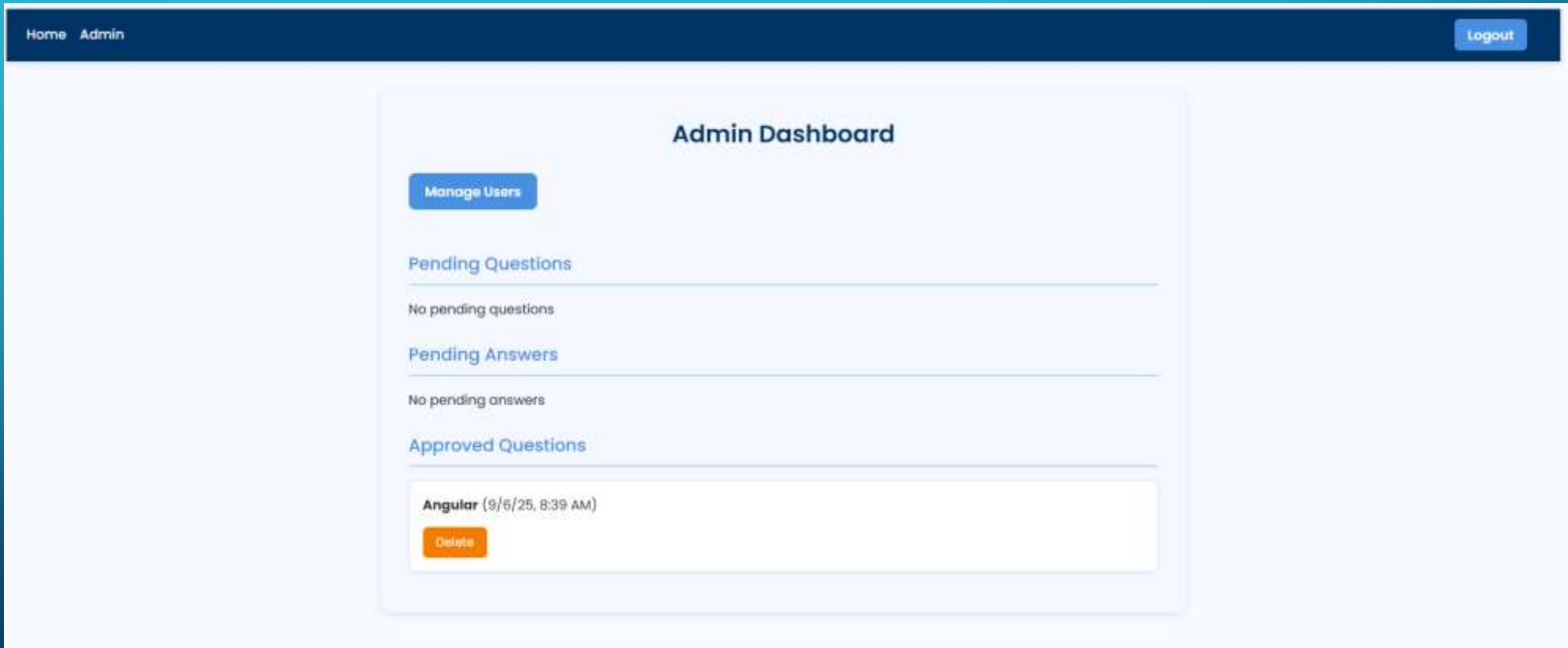
No pending questions.

Pending Answers

Angular is a Framework. (9/8/25, 11:08 AM)

Approve Reject Delete

ADMIN DELETE QUESTION



PROJECT SCREENSHOTS

DOCONNECT

ADMIN ADD USER

DOCONNECT

[Home](#) [Admin](#) [Logout](#)

User Management

Add New User

[Add User](#)

All Users

ID	Username	Email	Role	Actions
1	admin	admin@test.com	1	Edit Delete
2	User1	user1@gmail.com	0	Edit Delete
4	testuser	test@test.com	0	Edit Delete

ADMIN MODIFY USER

[Home](#) [Admin](#) [Logout](#)

User Management

Add New User

[Add User](#)

All Users

ID	Username	Email	Role	Actions
1	admin	admin@test.com	1	Edit Delete
2	User1	user1@gmail.com	0	Edit Delete

Edit User

[Save](#) [Cancel](#)

PROJECT SCREENSHOTS

DOCONNECT

ADMIN DELETE USER

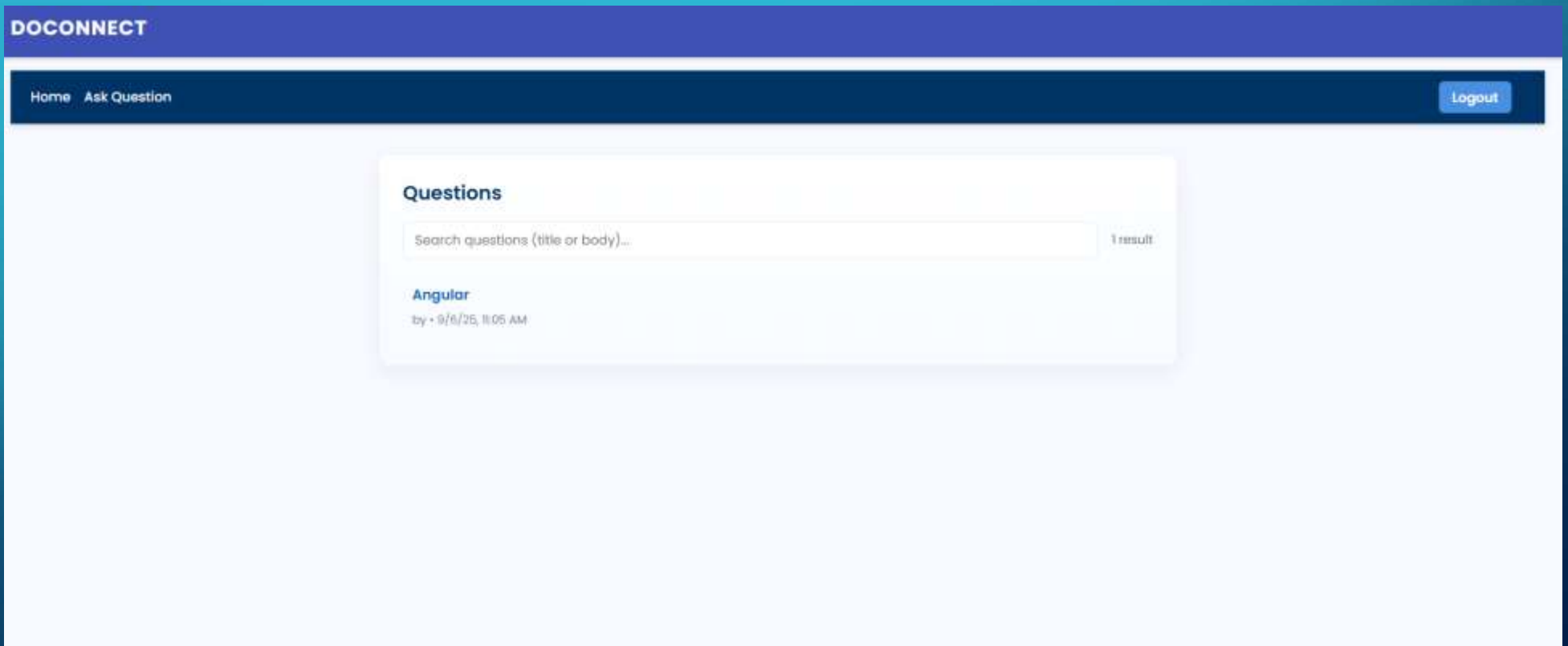
The screenshot displays the Admin Dashboard interface. At the top, a navigation bar includes links for 'Questions', 'Ask Question', 'Admin Dashboard', and 'Logout'. Below this, a secondary navigation bar shows 'Home' and 'Admin'. A modal dialog box is open in the center, titled 'localhost:4200 says', with the message 'Are you sure you want to delete this user?' and 'OK' and 'Cancel' buttons. The main content area is titled 'User Management'. It features an 'Add New User' section with input fields for 'Username', 'Email', and 'Password', and an 'Add User' button. Below this is a table titled 'All Users' with columns for 'ID', 'Username', 'Email', 'Role', and 'Actions'. The table contains two rows of user data.

ID	Username	Email	Role	Actions
1	admin	admin@test.com	1	Edit Delete
2	user1	user1@gmail.com	0	Edit Delete

PROJECT SCREENSHOTS

DOCONNECT

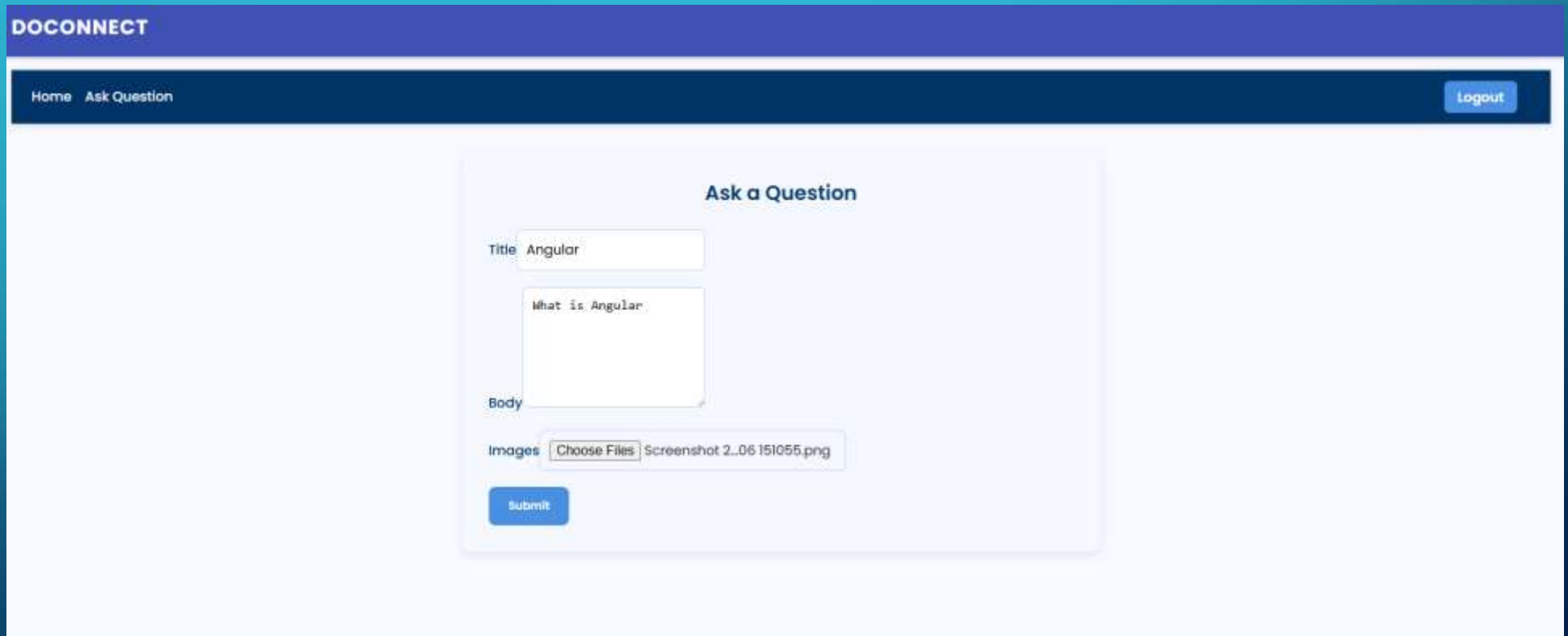
USER DASHBOARD



PROJECT SCREENSHOTS

DOCONNECT

USER ASK QUESTION



The screenshot displays the 'DOCONNECT' web application interface. At the top, a purple header bar contains the 'DOCONNECT' logo. Below this, a dark blue navigation bar features links for 'Home' and 'Ask Question', along with a 'Logout' button on the right. The main content area is white and features a central 'Ask a Question' form. This form includes a 'Title' field with the text 'Angular', a 'Body' text area containing 'What is Angular', and an 'Images' section with a 'Choose Files' button and a file name 'Screenshot 2...06 151055.png'. A blue 'Submit' button is positioned at the bottom of the form.

DOCONNECT

Home Ask Question Logout

Ask a Question

Title Angular

Body What is Angular

Images Choose Files Screenshot 2...06 151055.png

Submit

PROJECT SCREENSHOTS

DOCONNECT

USER ADD ANSWER

DOCONNECT

[Home](#) [Ask Question](#) [Logout](#)

Write an Answer

Answer

Angular is a Framework.

Images

Choose Files

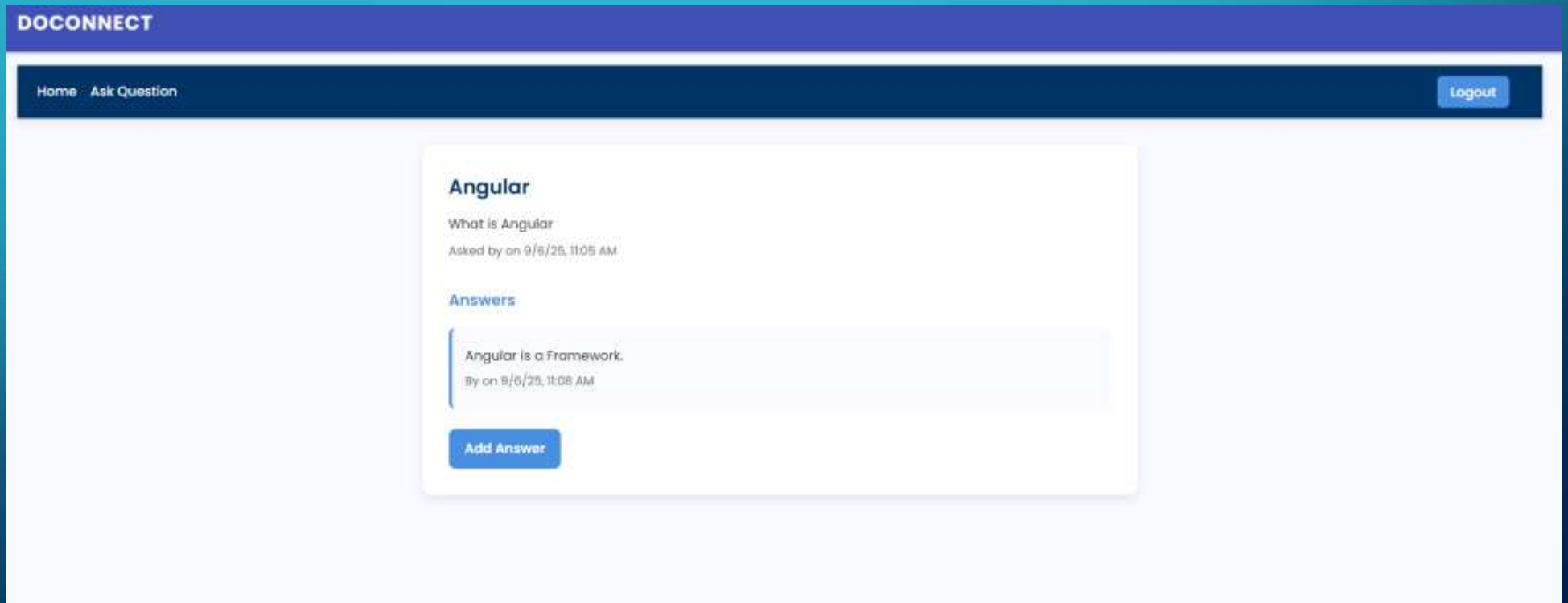
No file chosen

Submit

PROJECT SCREENSHOTS

DOCONNECT

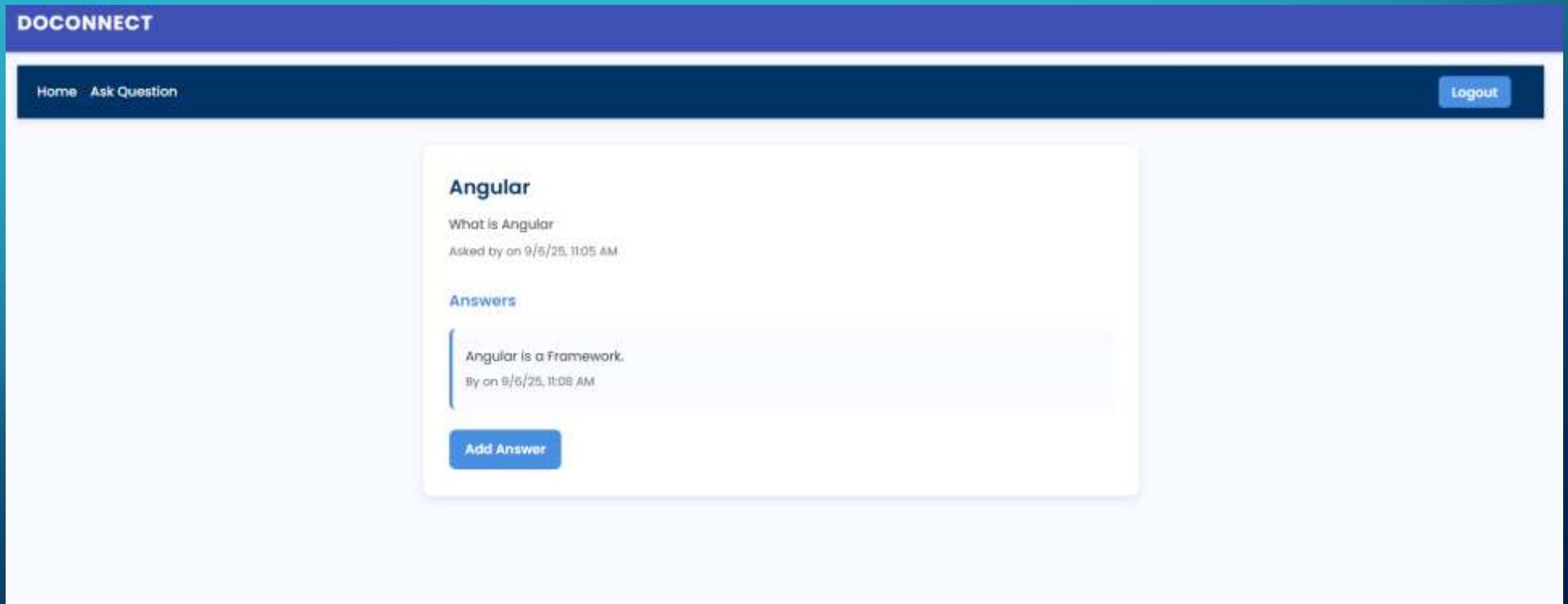
USER VIEW QUESTION



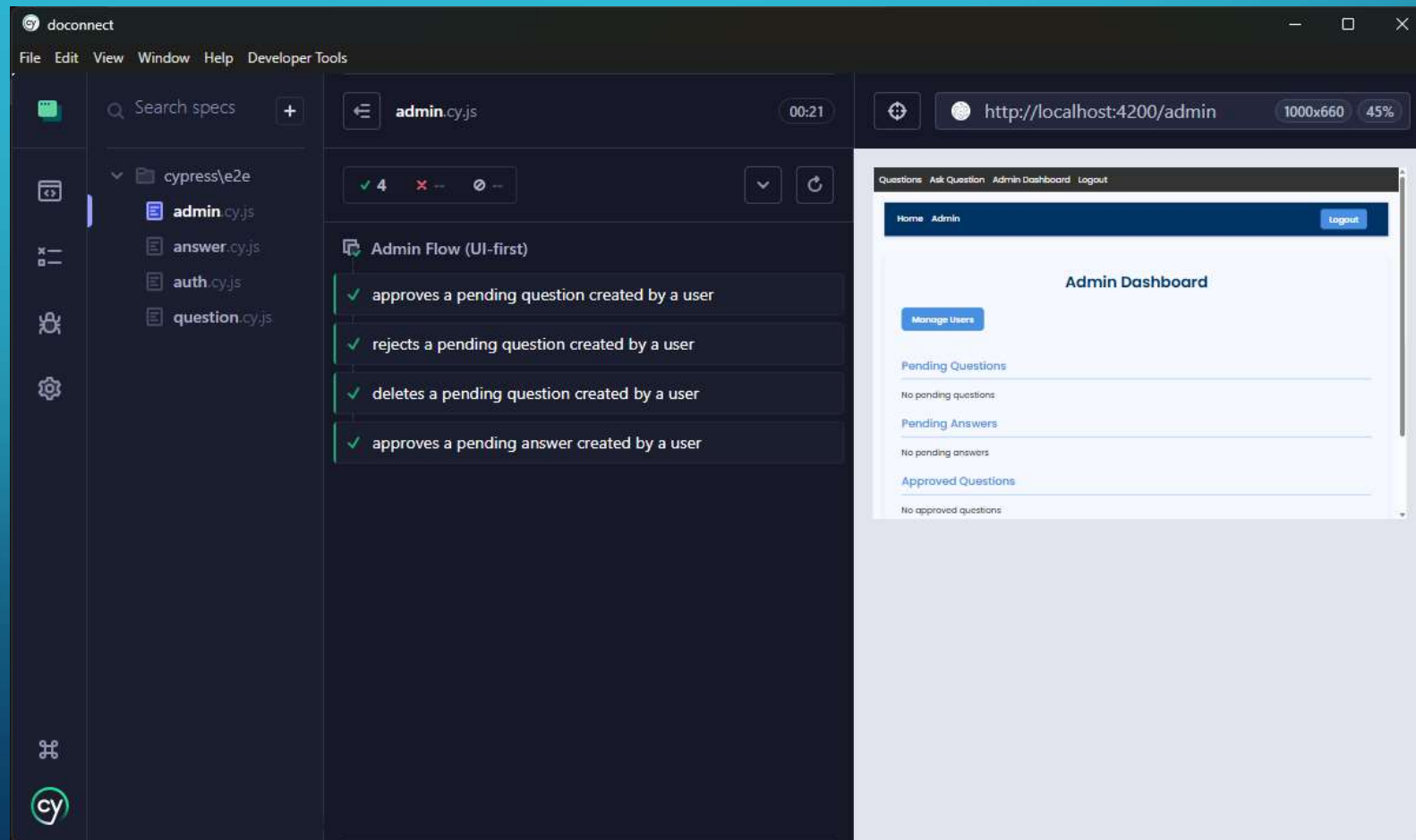
PROJECT SCREENSHOTS

DOCONNECT

USER VIEW QUESTION



TESTING - FRONTEND



TESTING - BACKEND

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  QUERY RESULTS

SELECT "q"."Id", "q"."Body", "q"."CreatedAt", "q"."Status", "q"."Title", "q"."UserId"
FROM "Questions" AS "q"
WHERE "q"."Id" = @__p_0
LIMIT 1
[xUnit.net 00:00:15.61] Finished: DoConnect.Api.Tests
DoConnect.Api.Tests test succeeded (20.9s)

Test summary: total: 4, failed: 0, succeeded: 4, skipped: 0, duration: 20.9s
Build succeeded in 35.1s

Workload updates are available. Run `dotnet workload list` for more information.

murug@SARAVANAN MINGW64 /d/DoConnect (main)
$
```

PROJECT SCREENSHOTS

DOCONNECT

GITHUB REPOSITORY

The screenshot displays the GitHub repository page for 'DoConnect_CapeStone'. The repository is owned by 'SACAS-BSC-CS-III' and is currently on the 'main' branch. The page shows a list of files and their commit history:

File	Commit Message	Commit Time
DoConnect.Api	First Commit	5 days ago
doconnect	First Commit	5 days ago
DoConnect.sln	First Commit	5 days ago
README.md	Create README.md	5 days ago
schema.sql	Added Schema File outside the API Folder	5 days ago

The README section is titled 'DoConnect Project' and lists the following features:

- User Registration/Login (JWT)
- Admin CRUD for users/questions/answers
- Ask/Answer Questions with approval workflow
- Angular + .NET 8 + SQL Server

The 'About' section on the right indicates that there is no description, website, or topics provided. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows a bar chart of the repository's language composition:

Language	Percentage
TypeScript	37.9%
C#	38.0%
SCSS	13.3%
HTML	7.8%
TSQL	4.5%
JavaScript	0.7%

The 'Suggested workflows' section is also visible on the right side of the page.

DOCONNECT

THANKYOU