

DoConnect — Ankit Dwivedi

Full-Stack Web Application (Angular + ASP.NET Core)

Name: Ankit Dwivedi

Batch: Batch 1 Asp .Net Cohort

Instructor: Parth Shukla

Frontend: Angular • Backend: ASP.NET Core Web API • Database: SQL Server • Tests: xUnit +
Jasmine/Karma

Table of Contents

Front matter (roman numerals)

- *Title Page*
- *Declaration / Certificate*
- *Acknowledgement*
- *Abstract*
- *Keywords*
- *List of Figures*
- *List of Tables*
- *List of Abbreviations*

Chapters (numbered)

1. *Introduction*
 - 1.1 *Problem Statement*
 - 1.2 *Objectives*
 - 1.3 *Scope & Assumptions*
 - 1.4 *Stakeholders & Roles*
 - 1.5 *Success Criteria*
2. *Requirements & Analysis*
 - 2.1 *Functional Requirements*
 - 2.2 *Non-Functional Requirements (Security, Performance, Availability, Usability)*
 - 2.3 *Use-Case Diagram & Descriptions*
 - 2.4 *Constraints & Risks*

3. *Architecture & Design*
 - 3.1 *Technology Stack*
 - 3.2 *High-Level System Architecture (Diagram)*
 - 3.3 *Module/Layered Architecture*
 - 3.4 *Security Architecture (JWT, CORS, input validation)*
 - 3.5 *Design Decisions & Trade-offs*
4. *Database Design*
 - 4.1 *ER Diagram*
 - 4.2 *Schema & Relationships (Tables, Keys, Indexes)*
 - 4.3 *Sample/Seed Data (if any)*
 - 4.4 *Query & Performance Considerations*
5. *Backend (ASP.NET Core Web API)*
 - 5.1 *Project Structure*
 - 5.2 *Controllers & Routes (Overview)*
 - 5.3 *Endpoint Reference (Tables)*
 - 5.4 *Request/Response Models & Validation*
 - 5.5 *Error Handling & ProblemDetails*
6. *Frontend (Angular)*
 - 6.1 *Information Architecture (Routes)*
 - 6.2 *Component/Service Structure*
 - 6.3 *State, Interceptors & Guards*
 - 6.4 *UI/UX & Accessibility Guidelines*
7. *Key Feature Flows (Project-specific)*
 - *DoConnect: Asking Questions, Answering with Images, Admin Moderation (Approve/Reject + delete images on reject), Search/Tags*
8. *File & Media Handling*
 - 8.1 *Upload Pipeline (IFormFile, sanitization)*
 - 8.2 *Storage Layout (wwwroot/uploads)*
 - 8.3 *Serving via /uploads/* and Caching*
9. *Testing*
 - 9.1 *Backend Unit Tests (xUnit + Moq)*
 - 9.2 *Frontend Tests (Jasmine/Karma)*
 - 9.3 *Integration/Postman Suite & Test Data*
 - 9.4 *Coverage & Results*

10. Deployment & DevOps

10.1 Environments & Configuration (appsettings, environment.ts)

10.2 Build/Release Pipeline

10.3 Hosting & Reverse Proxy

10.4 Logging/Monitoring & Secrets Management

11. Results & Performance

11.1 Screens/Flows Working

11.2 Basic Metrics (load times, DB indices impact)

11.3 Limitations

12. Challenges & Mitigations

13. Future Enhancements

14. Conclusion

15. References

Appendices

A. Postman Collection (Endpoints & Tests)

B. Detailed API Contracts (full request/response schemas)

C. Extra Screenshots & Evidence

1. Project Overview

DoConnect.Api is a full-stack platform built with Angular and ASP.NET Core Web API. It implements secure authentication, domain-specific CRUD flows, and media handling where applicable. The UI is responsive, with guards and interceptors, and the backend follows REST standards with validation and proper error responses.

1.1 Objectives

- User authentication and role-based authorization (JWT).
- Clean, responsive Angular UI with route guards and interceptors.
- RESTful API with proper validation, pagination and error handling.
- Relational database modeled with EF Core (indexes, foreign keys).
- Automated tests across risky services/controllers; Postman collection for E2E flows.

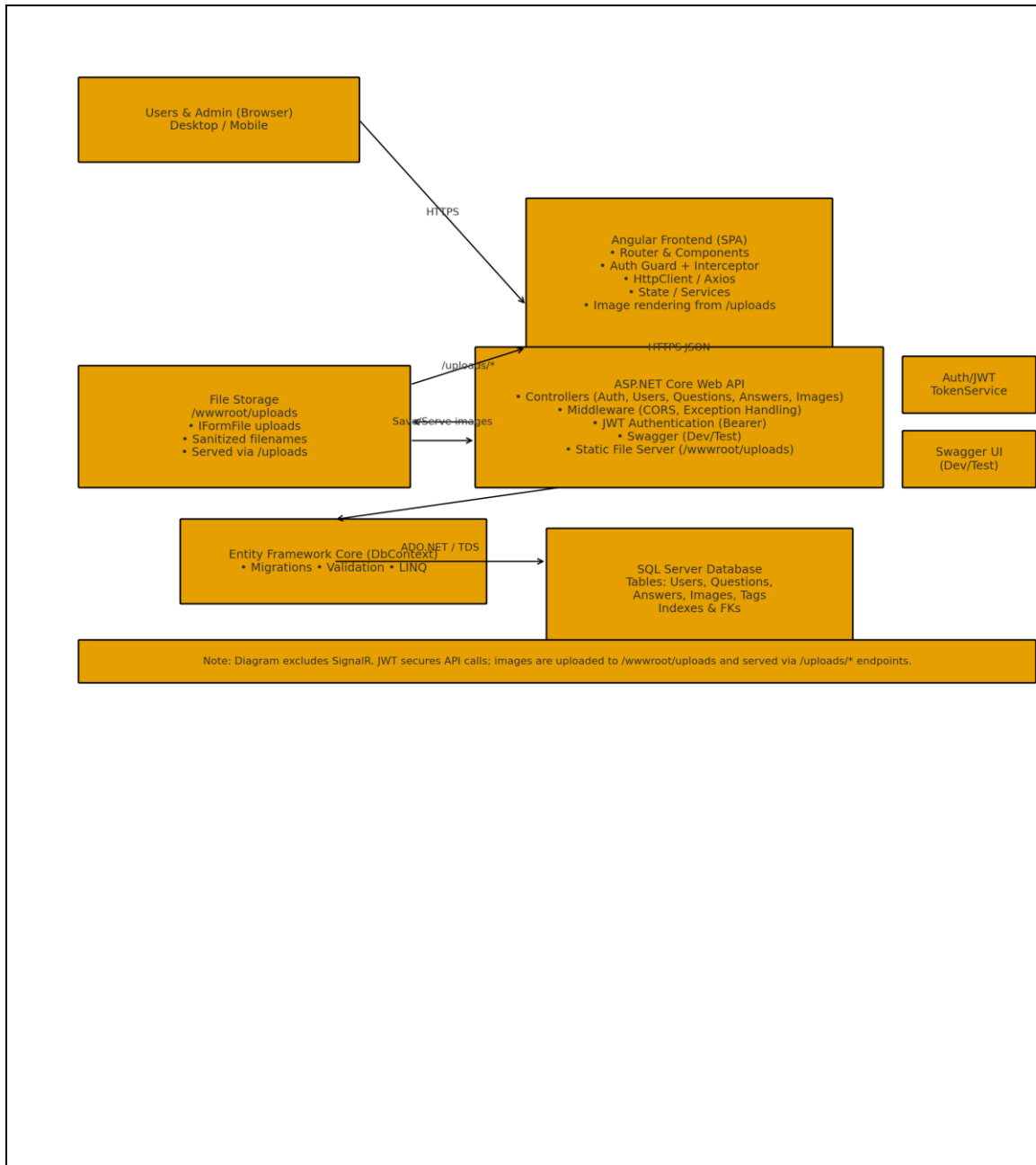
2. Architecture

2.1 Technology Stack

- Frontend: Angular (Router, Components, Forms, HttpClient/Interceptor).
- Backend: ASP.NET Core Web API (.NET 9), Entity Framework Core (Code-First).
- Database: SQL Server (dev SQLite optional).
- Auth: JWT (Bearer).
- Static file server: /wwwroot/uploads (if image uploads used).
- Testing: xUnit + Moq (backend); Jasmine/Karma (frontend).

2.2 System Diagram (Placeholder)

System Architecture Diagram



3. REST API (Generated from Controllers)

HTTP	Endpoint	Action	Notes
HTTPGET	api/admin/answers/pending	GetPendingAnswers	admin
HTTPPOST	api/admin/answers/{id:guid}/approve	ApproveAnswer	admin
HTTPPOST	api/admin/answers/{id:guid}/reject	RejectAnswer	admin
HTTPPOST	api/admin/questions	CreateQuestionAsAdmin	admin
HTTPGET	api/admin/questions/pending	GetPendingQuestions	admin
HTTPDELETE	api/admin/questions/{id:guid}	DeleteQuestion	admin
HTTPPOST	api/admin/questions/{id:guid}/approve	ApproveQuestion	admin
HTTPPOST	api/admin/questions/{id:guid}/reject	RejectQuestion	admin
HTTPPOST	api/admin/questions/{questionId:guid}/answers	PostAnswerAsAdmin	admin
HTTPGET	api/admin/users	List	adminusers
HTTPPOST	api/admin/users	Create	adminusers
HTTPDELETE	api/admin/users/{id:guid}	Delete	adminusers
HTTPGET	api/admin/users/{id:guid}	Get	adminusers
HTTPPUT	api/admin/users/{id:guid}	Update	adminusers
HTTPPOST	api/ai/ask	Ask	ai
HTTPPOST	api/auth/login	Login	auth
HTTPGET	api/auth/me	Me	auth
HTTPPOST	api/auth/register	Register	auth
HTTPGET	api/questions	List	questions
HTTPPOST	api/questions	Create	questions
HTTPGET	api/questions/{id:guid}	GetById	questions
HTTPGET	api/questions/{questionId:guid}/answers	List	answers
HTTPPOST	api/questions/{questionId:guid}/answers	Create	answers
HTTPPUT	api/users/me	UpdateMe	usersme

3.1 Auth & Security

- JWT issued on successful login; include in Authorization header as Bearer token.
- Protect admin-only routes with policies or role checks.
- Global exception handler and CORS configured for the Angular origin.

4. Data Model

4.1 Entities

Answer

Property	Type
Id	Guid
Text	string
Status	ApproveStatus
QuestionId	Guid
Question	Question
UserId	Guid
User	User
CreatedAt	DateTime
Images	ICollection<ImageFile>

ImageFile

Property	Type
Id	Guid
Path	string
UploadedAt	DateTime
QuestionId	Guid?
Question	Question?
AnswerId	Guid?
Answer	Answer?

Question

Property	Type
Id	Guid
Title	string
Text	string
Status	ApproveStatus
UserId	Guid
User	User
CreatedAt	DateTime
Answers	ICollection<Answer>
Images	ICollection<ImageFile>

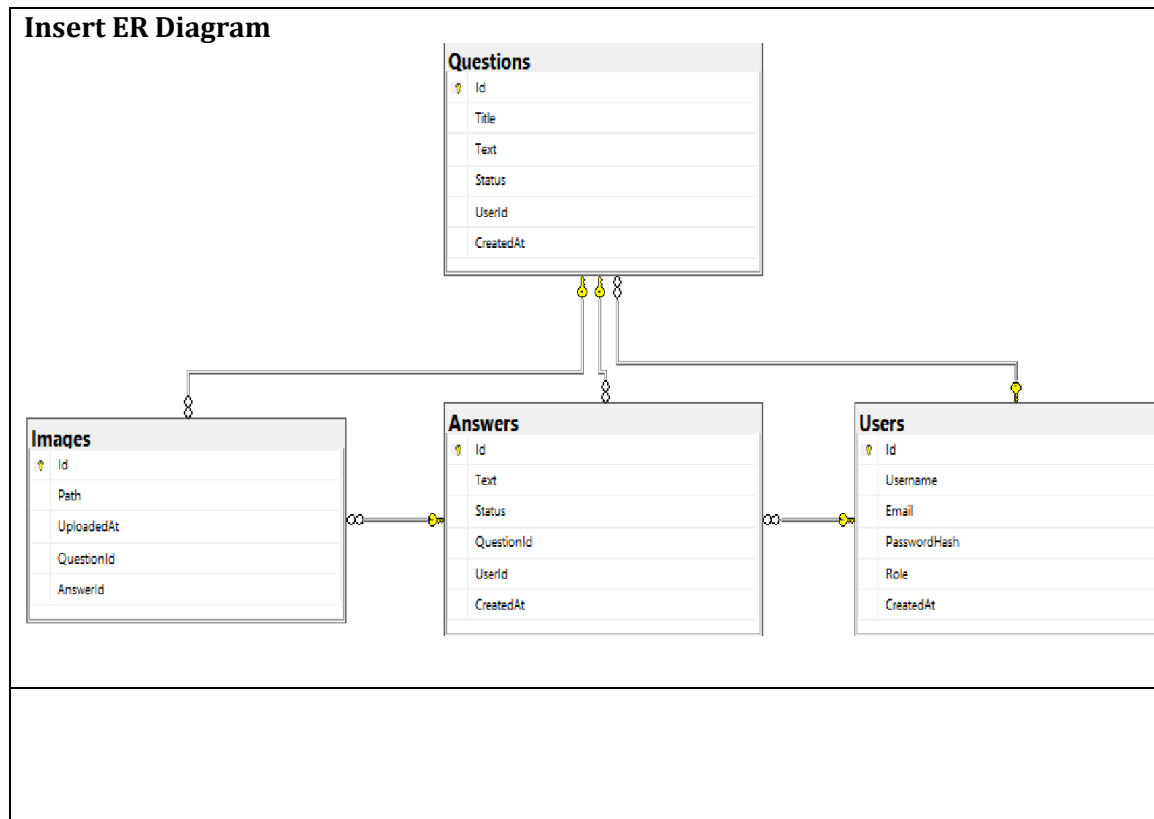
User

Property	Type
Id	Guid
Username	string
Email	string
PasswordHash	string
Role	RoleType
CreatedAt	DateTime
Questions	ICollection<Question>
Answers	ICollection<Answer>

4.2 Relationships & Indexes

- `e.HasIndex(u => u.Username).IsUnique();`
- `e.HasIndex(u => u.Email).IsUnique();`
- `e.HasOne(q => q.User)`
- `.WithMany(u => u.Questions)`
- `.HasForeignKey(q => q.UserId)`
- `e.HasIndex(x => x.UserId);`
- `e.HasIndex(x => x.CreatedAt);`
- `e.HasOne(a => a.User)`
- `.WithMany(u => u.Answers)`
- `.HasForeignKey(a => a.UserId)`
- `e.HasOne(a => a.Question)`
- `.WithMany(q => q.Answers)`
- `.HasForeignKey(a => a.QuestionId)`
- `e.HasIndex(x => new { x.QuestionId, x.UserId });`
- `e.HasIndex(x => x.CreatedAt);`
- `e.HasOne(i => i.Question)`

4.3 ER Diagram (Placeholder)



5. Frontend

- Auth: LoginComponent, RegisterComponent, Guards (AuthGuard, AdminGuard).
- Questions: AskComponent, QuestionListComponent, QuestionDetailComponent (with image gallery).
- Answers: AnswerFormComponent, AnswerListComponent (approve/reject badges).
- Tags & Search: TagChipComponent, SearchBarComponent, FiltersSidebar.
- Profile: ProfileViewComponent, ProfileEditComponent (optional avatar upload).
- Admin: AdminDashboard, UsersAdminComponent, PostsModerationComponent, ReportsComponent.
- Shared: Navbar (responsive with user dropdown), Snackbar/Toast, FileUploadComponent.

6. Validation & Error Handling

- DTO validation (Required/MaxLength/Range) for all inputs.
- Consistent 400/401/403/404 responses with ProblemDetails payloads.
- File upload limits and MIME checks where applicable.
- Global exception handler with structured logs.

7. Testing

- xUnit + Moq for service/controller units (auth, critical CRUD flows).
- Angular component tests for forms, guards, and service integration.
- Postman collection for end-to-end scenarios.

8. Deployment & DevOps

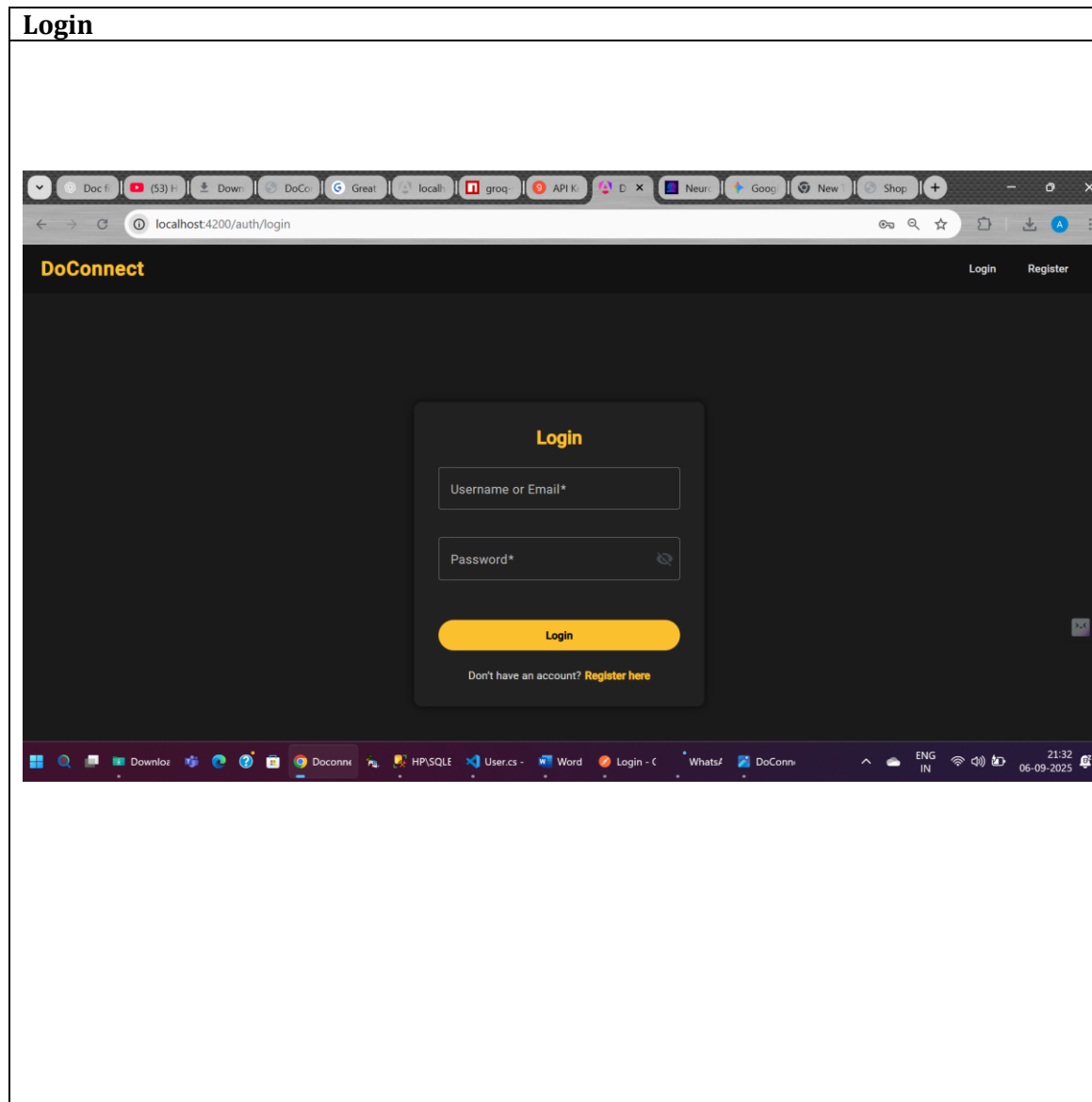
- Configuration via appsettings.* and environment.ts files.
- CI: restore → build → run tests → publish artifacts.
- Reverse proxy or static host for Angular; API on IIS/Azure/Docker.
- Log aggregation and monitoring (Serilog, Application Insights).

9. Accessibility & UX

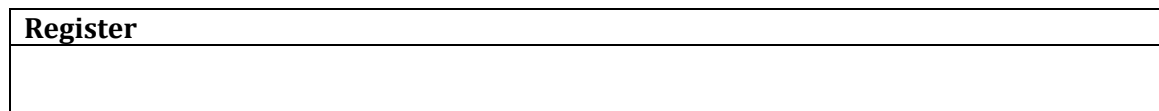
- Keyboard navigation, ARIA labels, color contrast compliance.
- Responsive navbar with user dropdown (profile/logout).
- Meaningful validation and error messages.

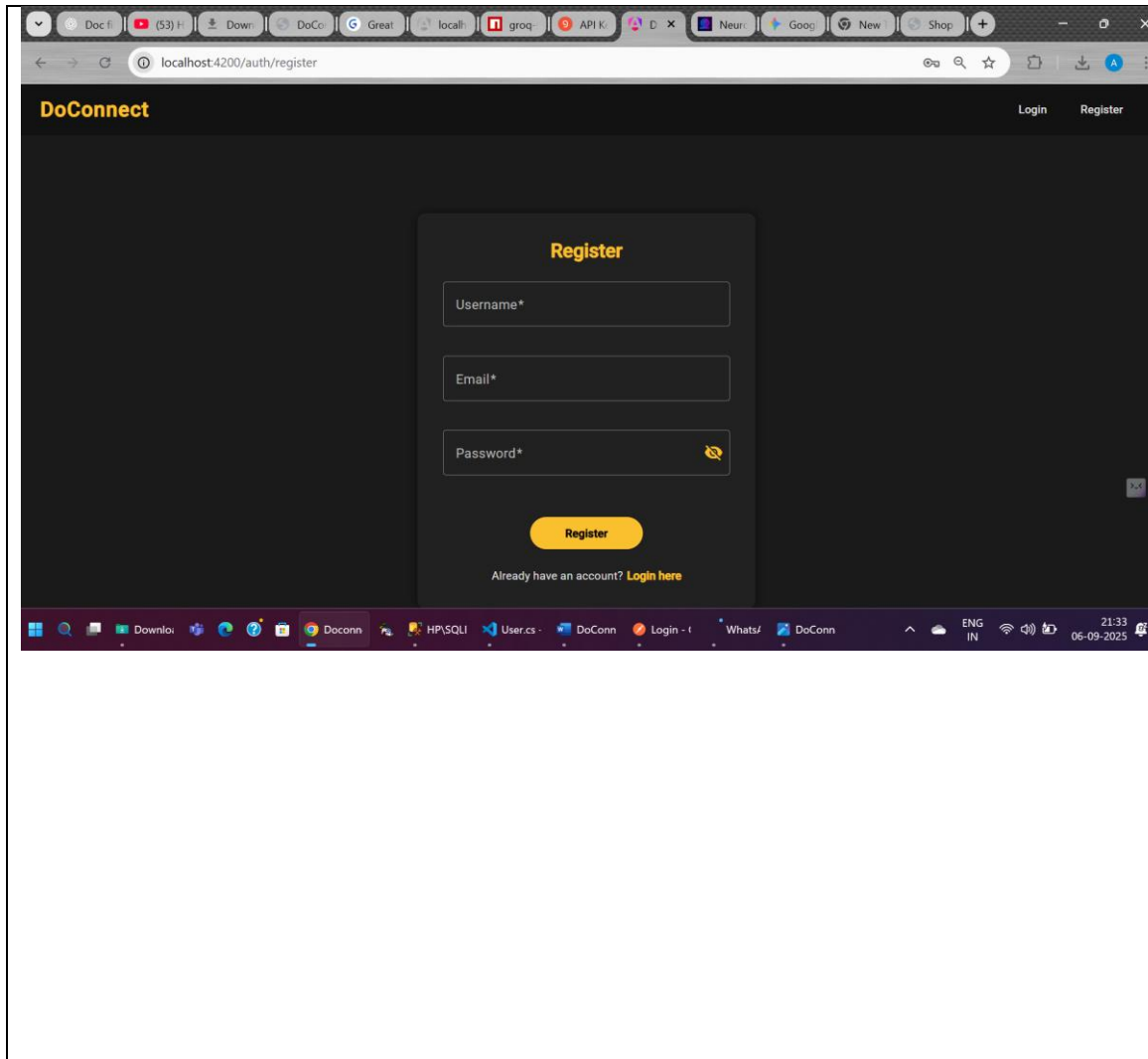
10. Implementation Screenshots

10.1 Login



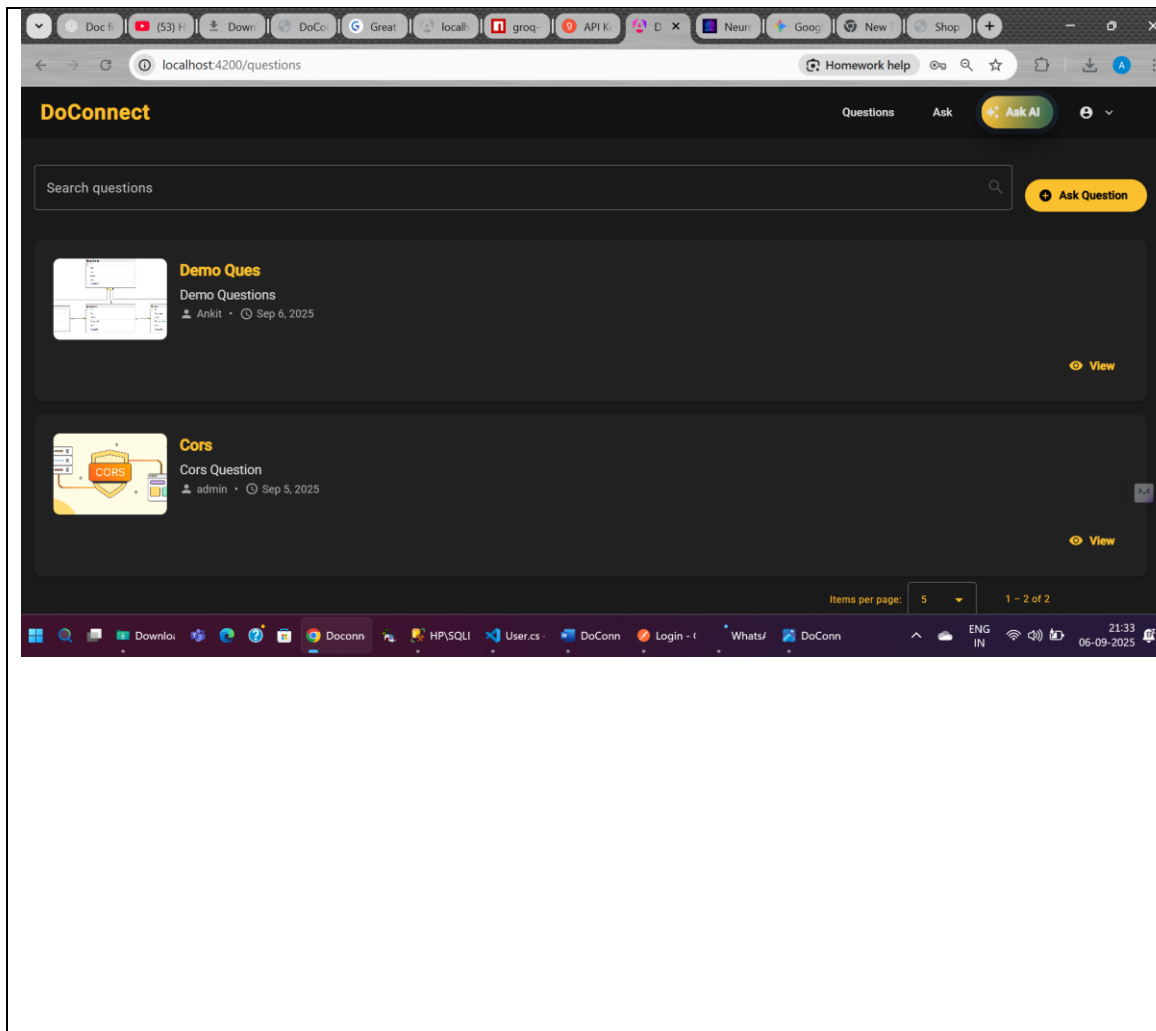
10.2 Register





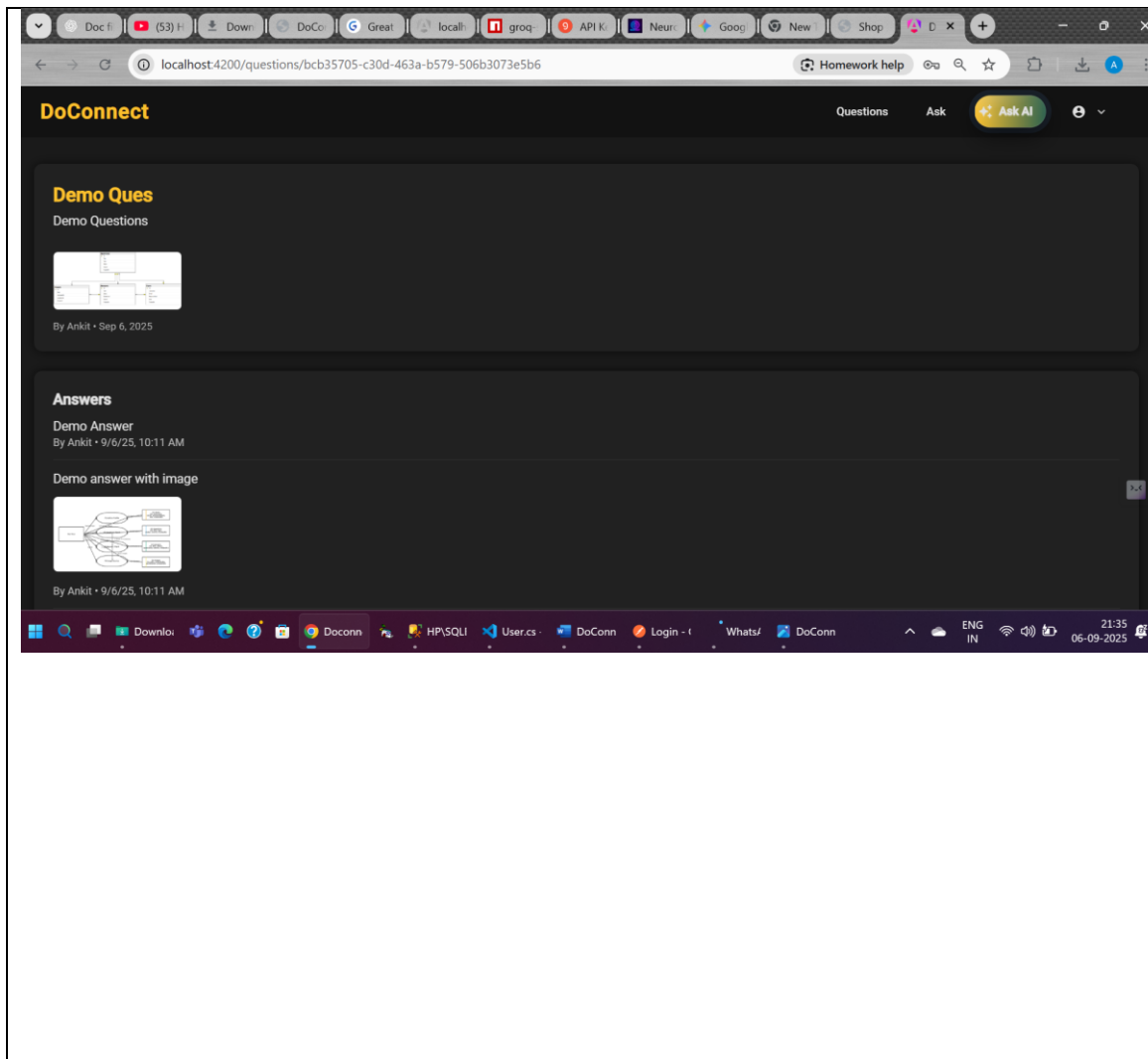
10.3 Landing / Dashboard

Landing / Dashboard



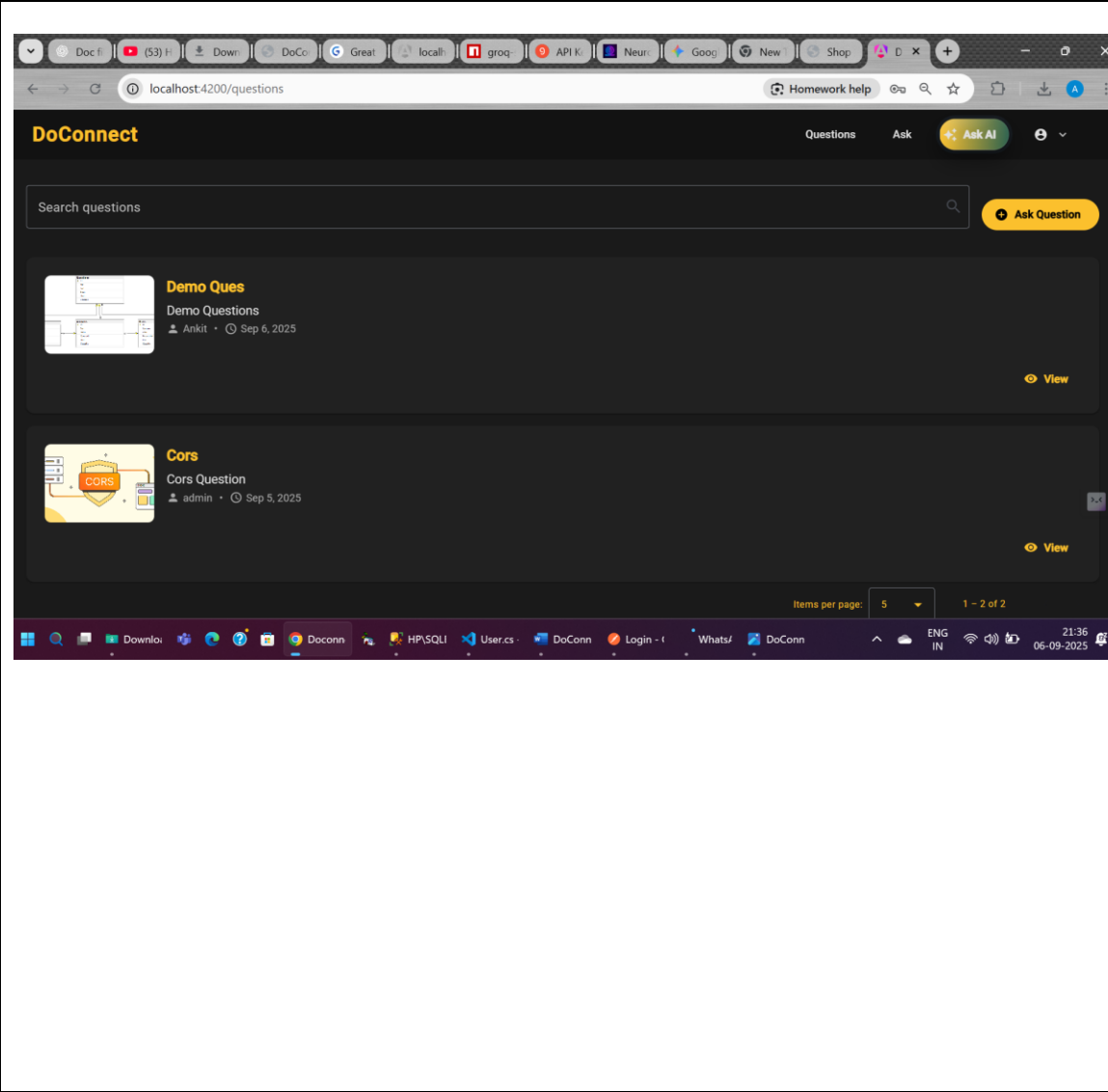
10.4 Primary CRUD Screen

Primary CRUD Screen



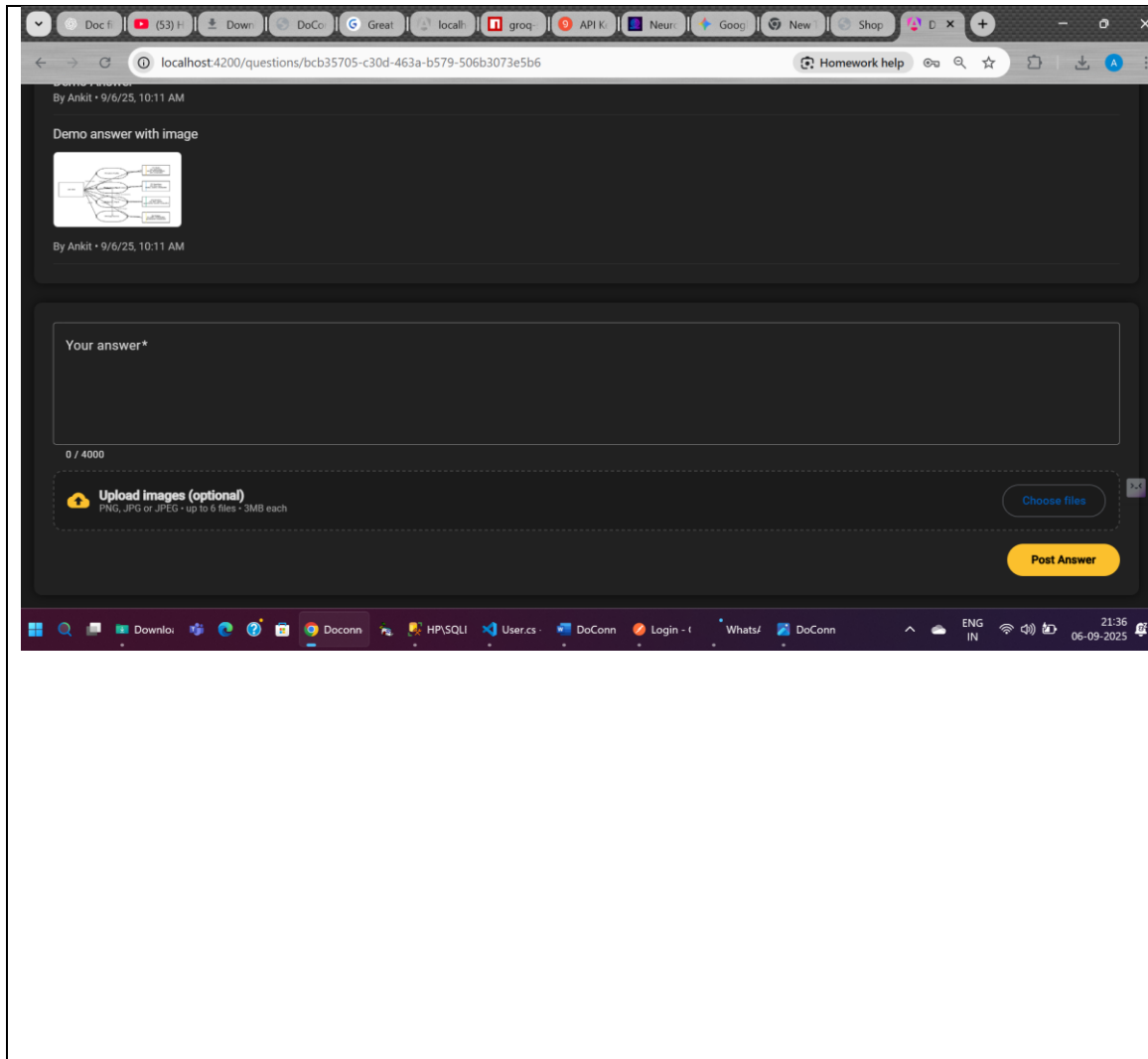
10.5 List + Pagination/Filters

List + Pagination/Filters



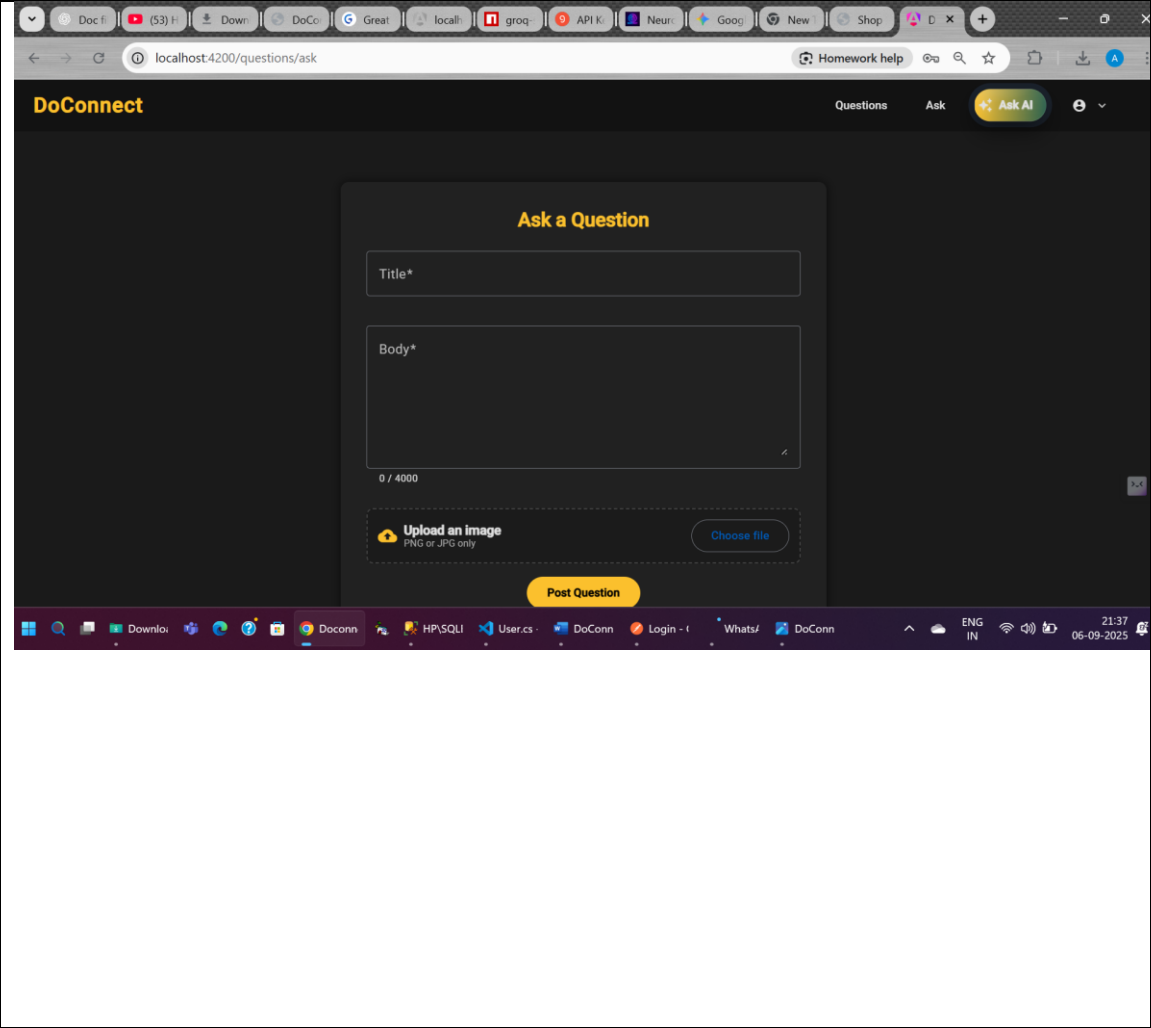
10.6 Detail View

Detail View



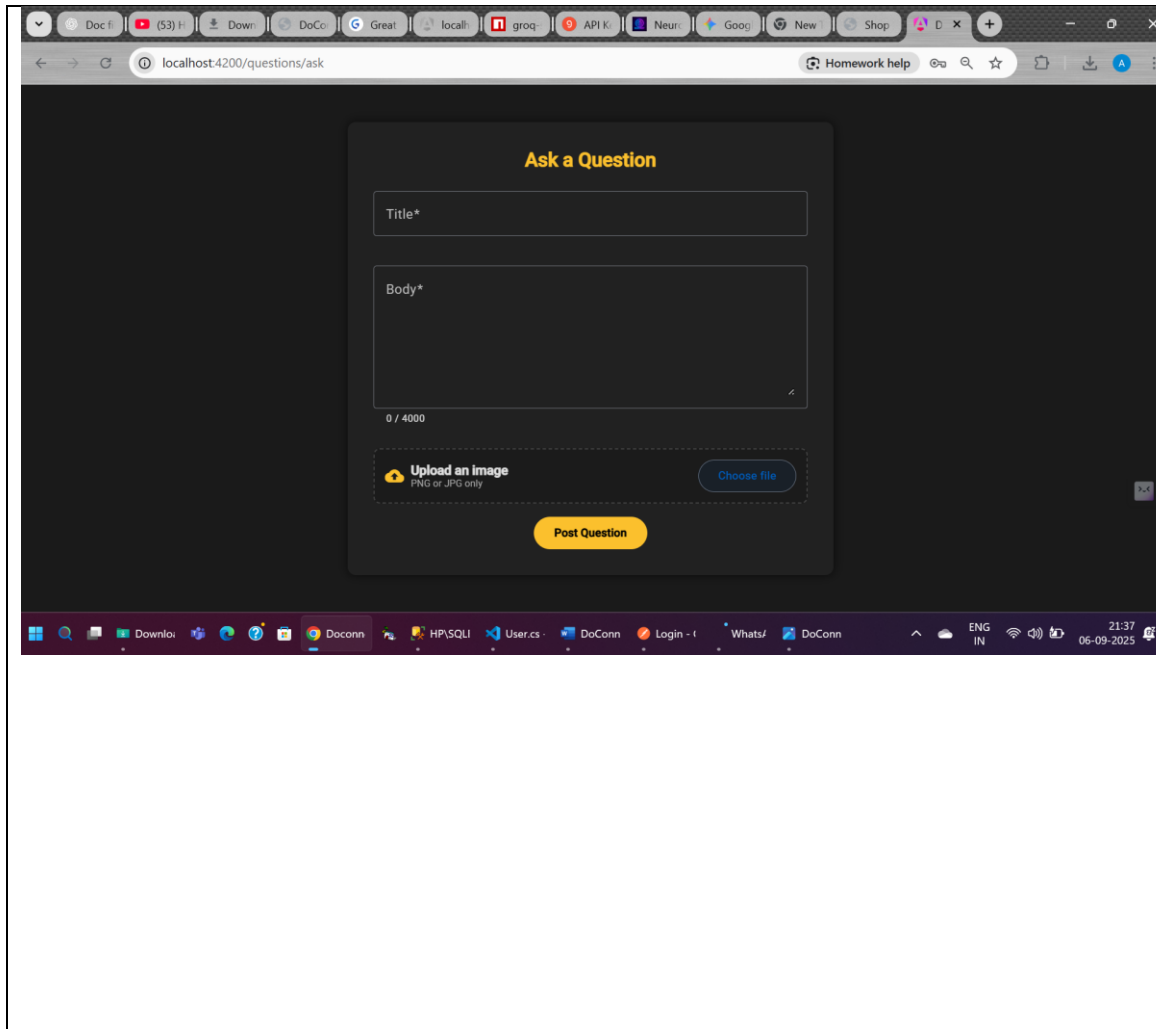
10.7 Create/Edit Form

Create/Edit Form



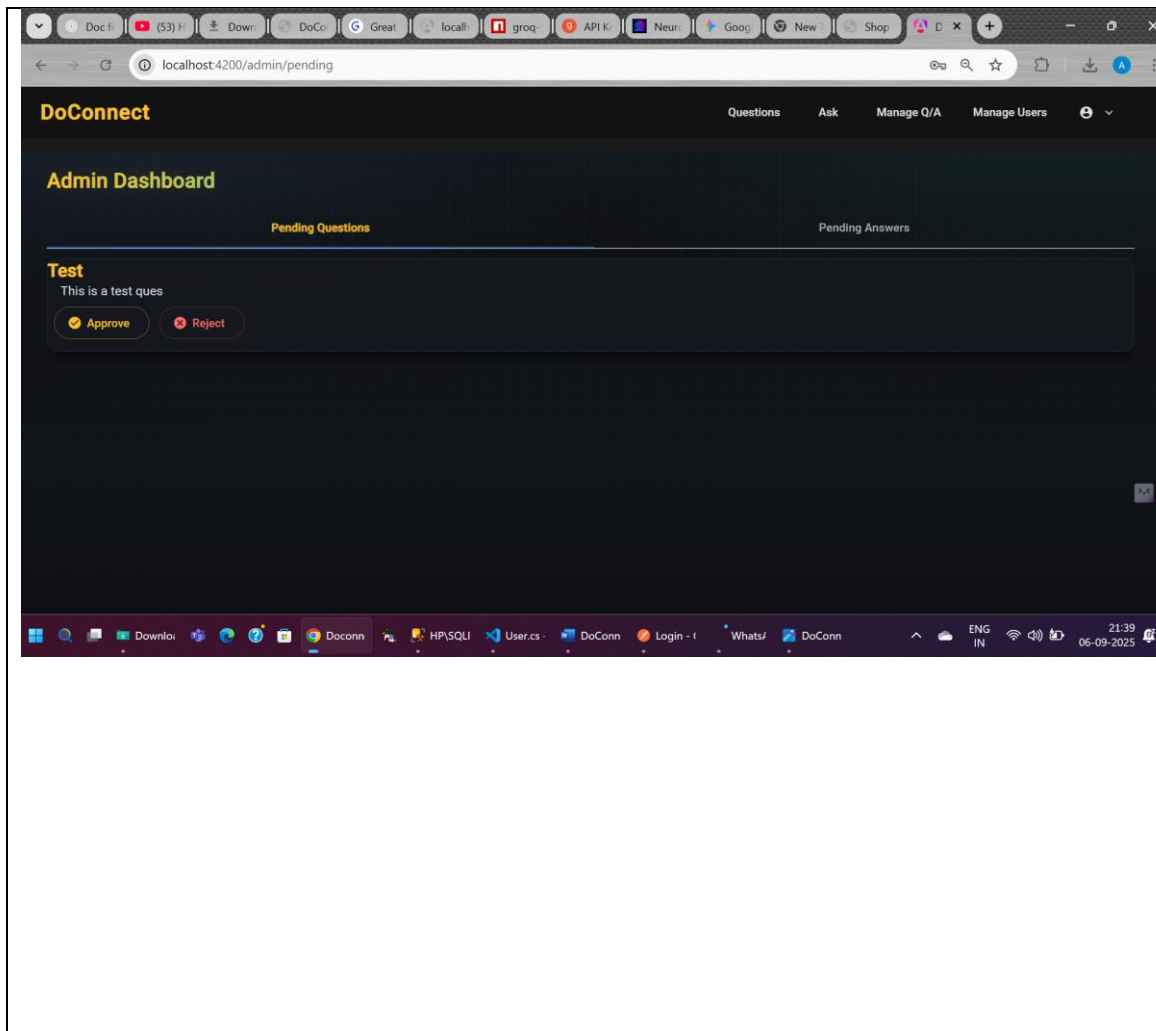
10.8 File Upload

File Upload



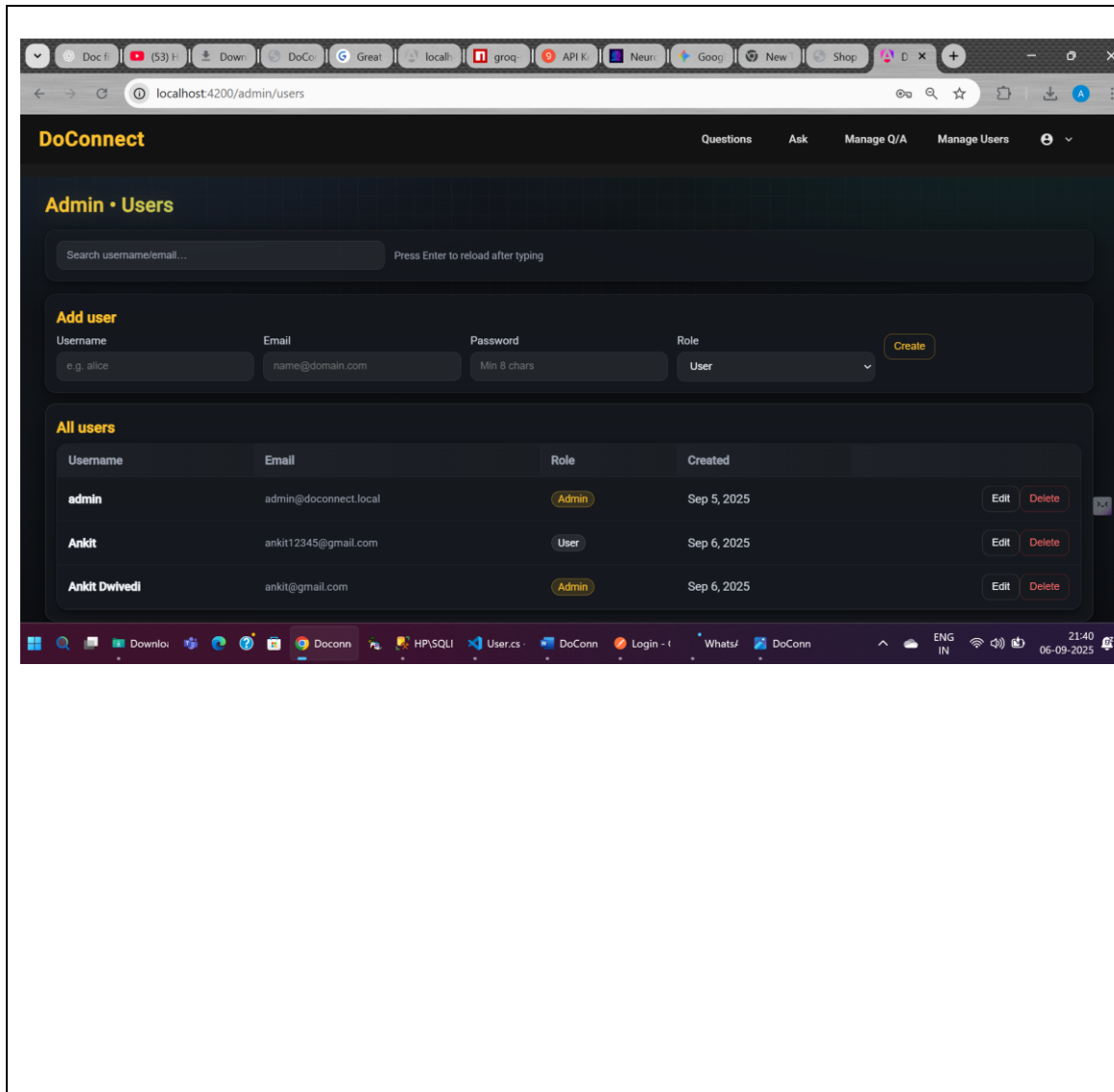
10.9 Admin/Reports

Admin/Reports



10.10 Manage Users (Admin)

Manage Users (Admin)



11. Submission Checklist

- DoConnect.Api_Final_Report_Ankit_Dwivedi.docx/pdf (this file).
- Backend source ZIP; Frontend source ZIP.
- Database schema SQL; seed data (if any).
- Postman collection + environment JSON.
- Screenshots embedded in Section 10.
- Private Git repository link (archived).

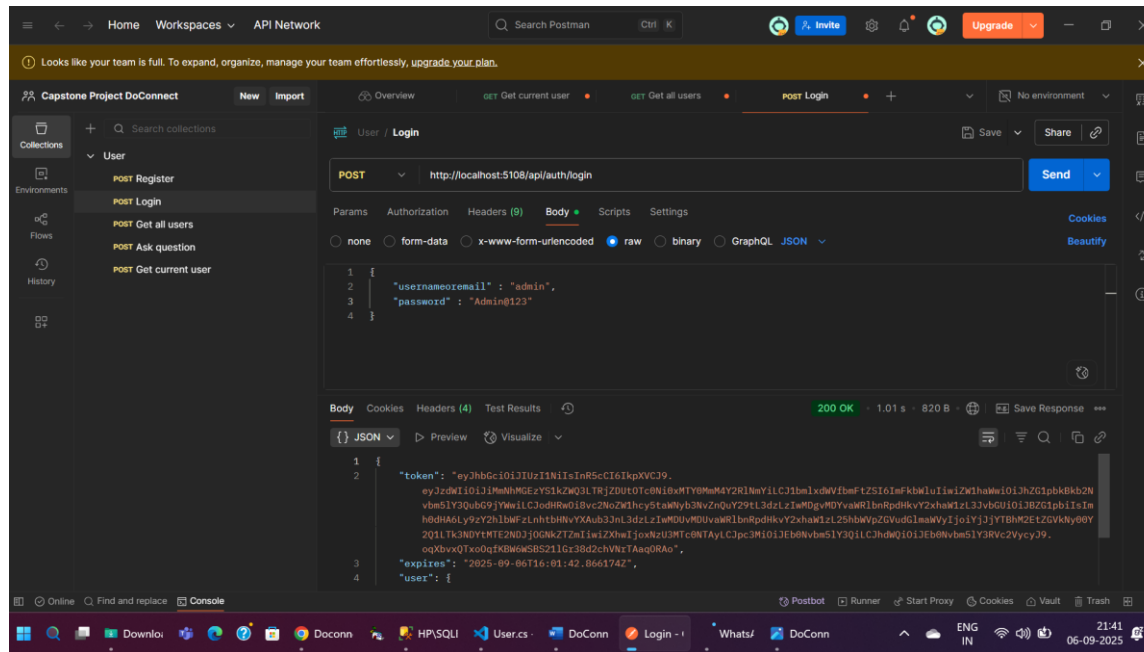
12. Conclusion

DoConnect.Api integrates a secure ASP.NET Core backend with an Angular frontend to deliver a robust, testable, and maintainable solution. With proper validation, relational data modeling, and CI-ready structure, it meets the academic evaluation standards and is ready for further extension.

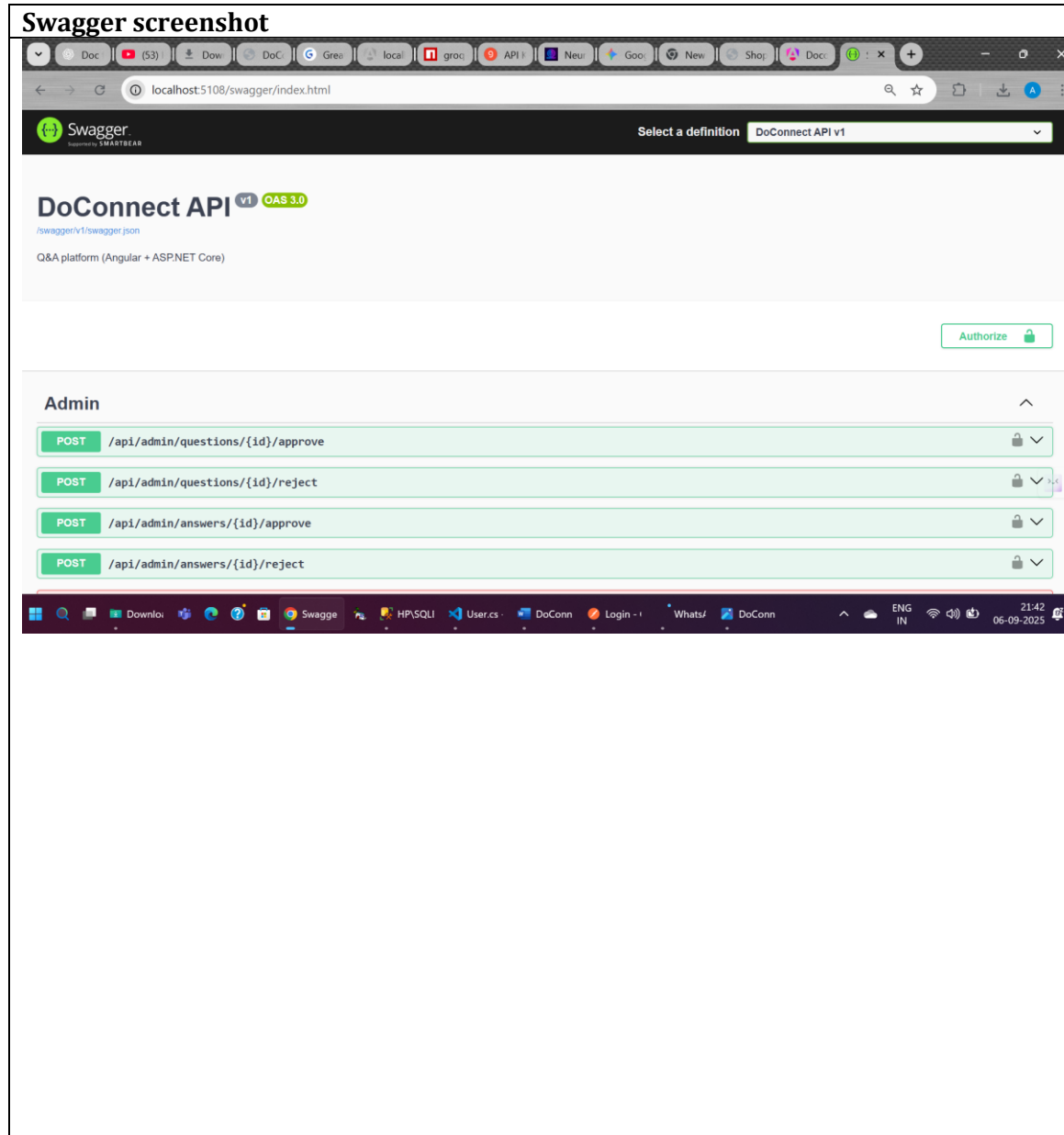
13. References

- Microsoft Docs — ASP.NET Core, EF Core
- Angular Docs — angular.io
- OWASP — Authentication & Input Validation

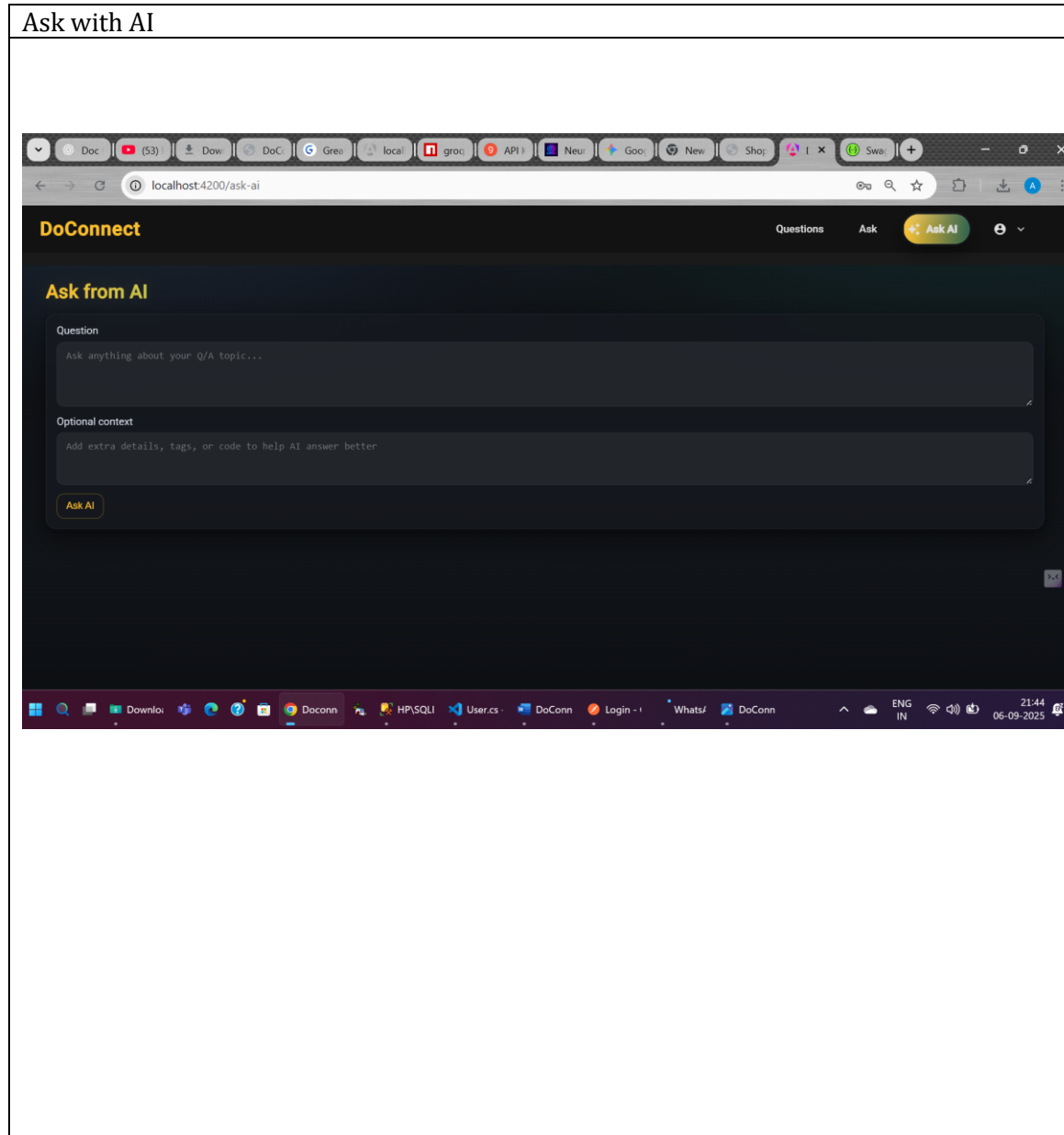
A.1 Postman Screenshot

Postman screenshot

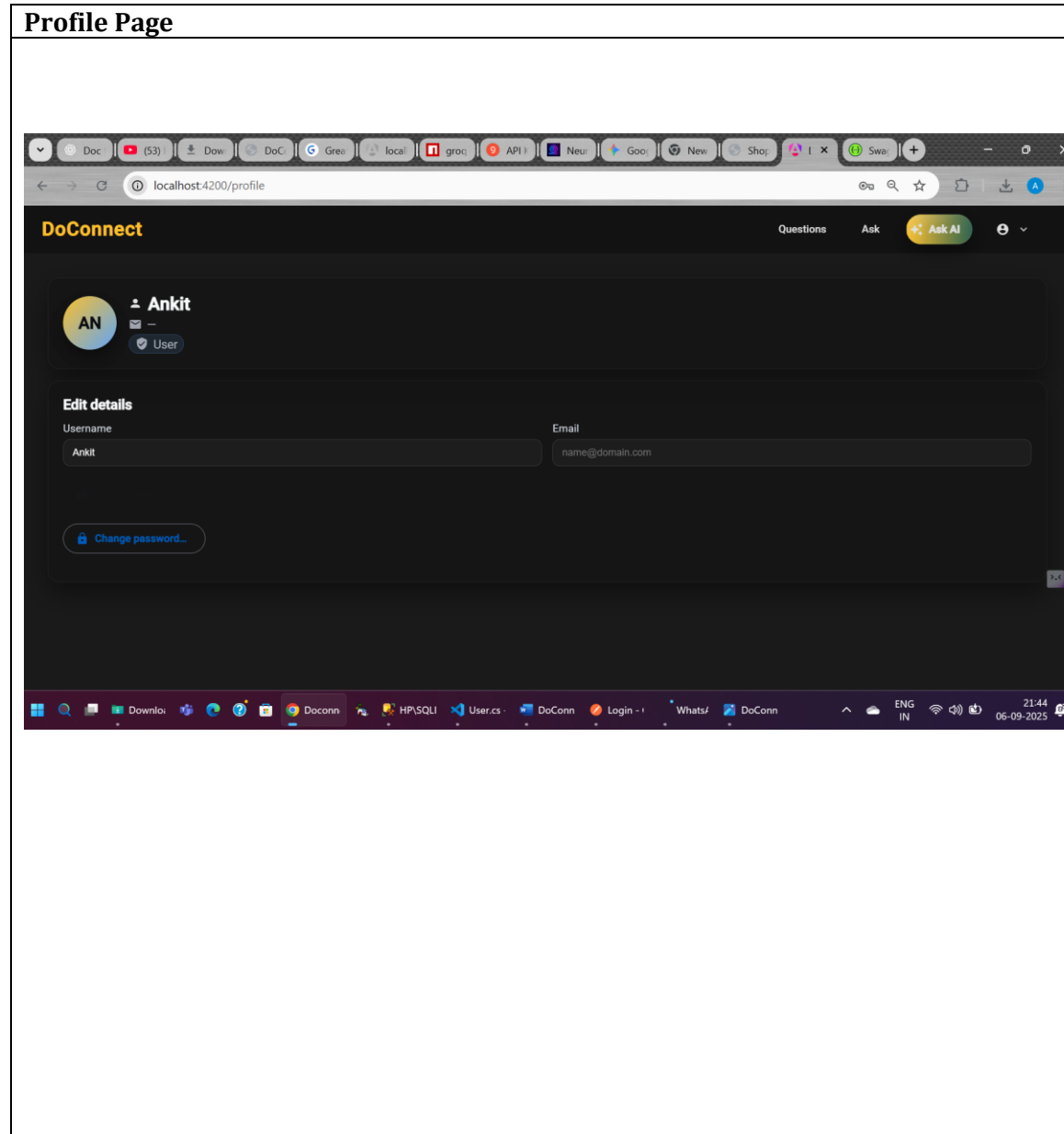
A.2 Swagger



A.3 Ask with AI

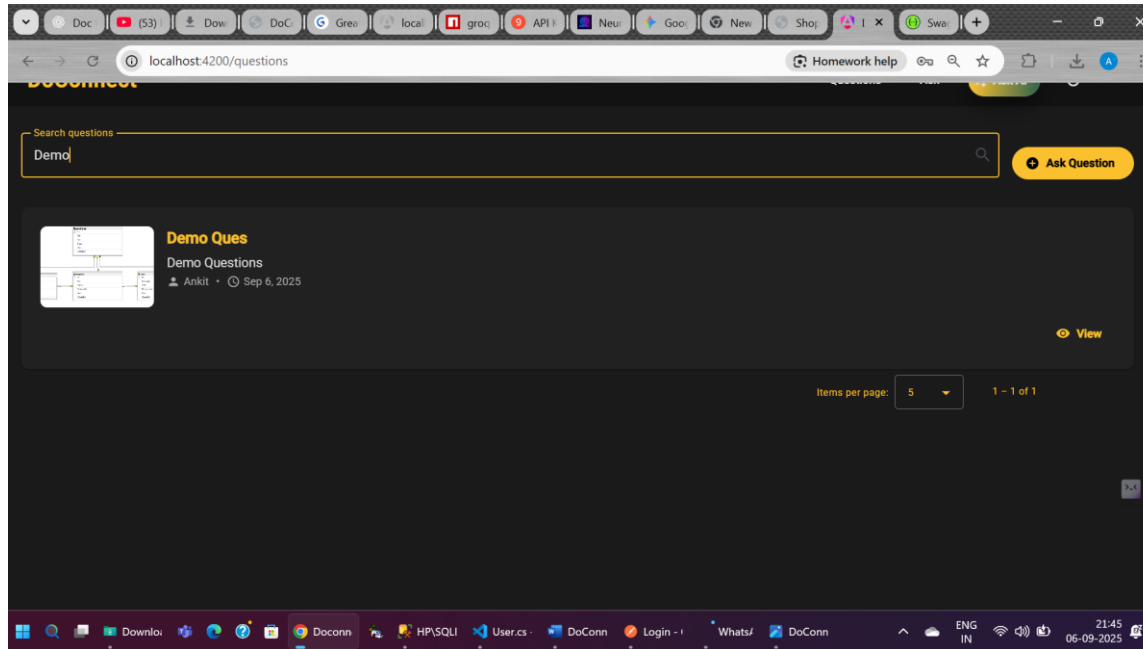


A.4 Profile Page



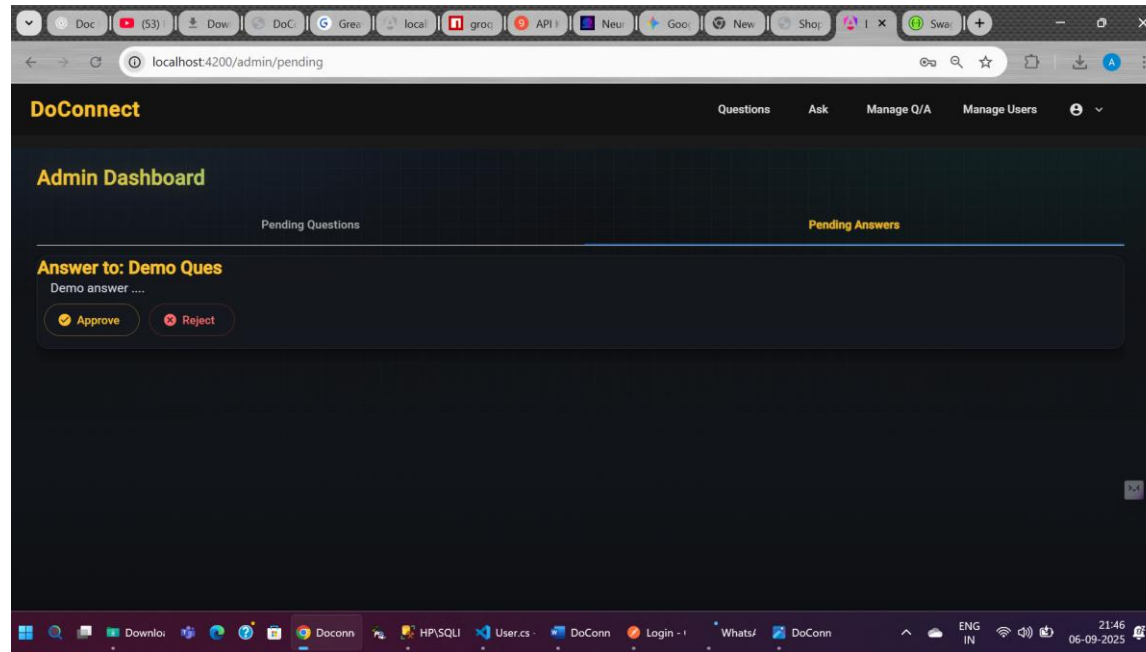
A.5 Search functionalities

Search functionalities



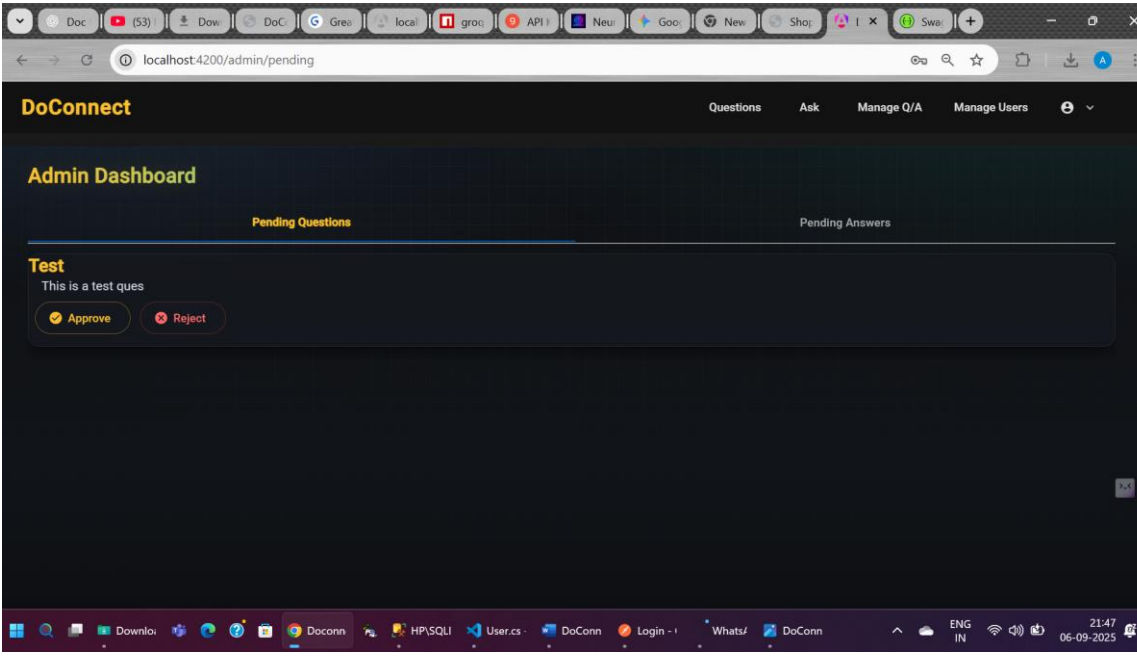
A.6 Manage pending answer by admin

Pending Answer



A.7 Manage Pending question by Admin

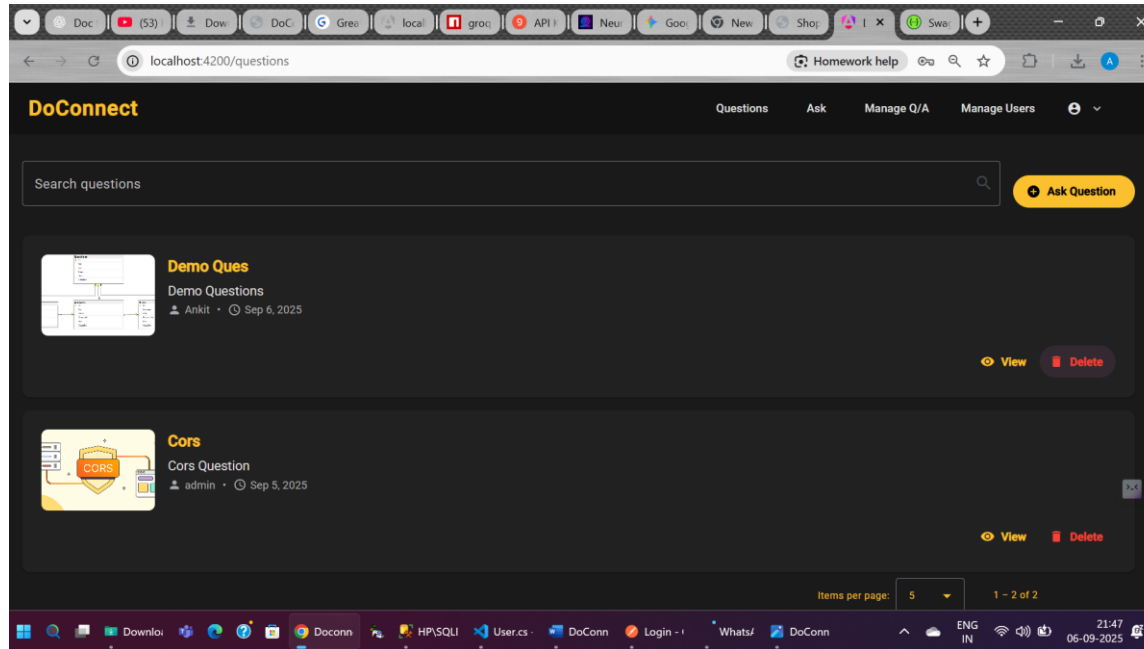
Pending Question



The screenshot shows a web browser window with the URL `localhost:4200/admin/pending`. The page title is "DoConnect". The navigation bar includes links for "Questions", "Ask", "Manage Q/A", and "Manage Users". The main content area is titled "Admin Dashboard" and features a tabbed interface with "Pending Questions" and "Pending Answers". Under the "Pending Questions" tab, there is a "Test" section with the text "This is a test ques" and two buttons: "Approve" (with a green checkmark icon) and "Reject" (with a red X icon). The browser's taskbar at the bottom shows various open applications, including "Download", "Doconn", "HPSQL", "User.cs", "DoConn", "Login", "Whats/", and "DoConn". The system clock indicates the time is 21:47 on 06-09-2025.

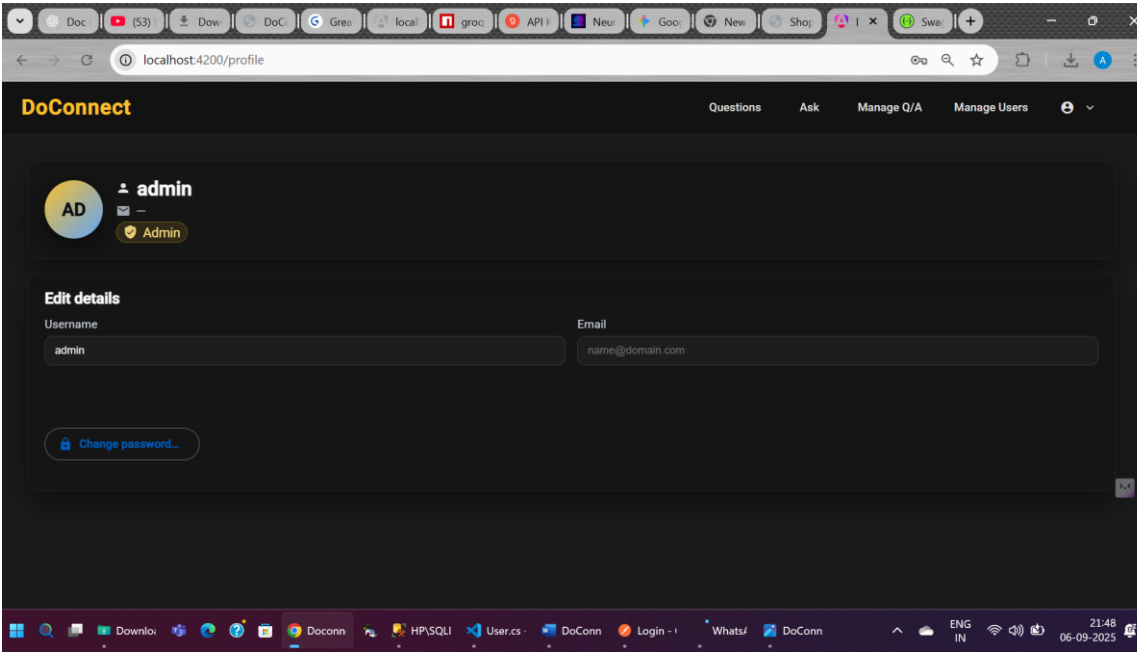
A.8 Delete Question functionality only for admin

Delete button



A.9 Admin Profile

Admin Profile



The screenshot displays the 'Admin Profile' page of the DoConnect application. The page is viewed in a web browser at the URL 'localhost:4200/profile'. The application's header includes the 'DoConnect' logo and navigation links for 'Questions', 'Ask', 'Manage Q/A', and 'Manage Users'. The profile section shows the user 'admin' with a profile picture 'AD' and a role 'Admin'. Below this is an 'Edit details' section with input fields for 'Username' (admin) and 'Email' (name@domain.com). A 'Change password...' button is also visible. The page is part of a web application running on localhost:4200.

A.10 Manage User Page By Admin

Insert Evidence 15

Manage User

localhost:4200/admin/users

DoConnect

QuestionsAskManage Q/AManage Users

Admin • Users

Search username/email...

Press Enter to reload after typing

Add user

Username

e.g. alice

Email

name@domain.com

Password

Min 8 chars

Role

User

Create

All users

Username	Email	Role	Created	
admin	admin@doconnect.local	Admin	Sep 5, 2025	EditDelete
Ankit	ankit12345@gmail.com	User	Sep 6, 2025	EditDelete
Ankit Dwivedi	ankit@gmail.com	Admin	Sep 6, 2025	EditDelete

Download

Doconn

HP/SQL

User.cs

DoConn

Login

Whats/

DoConn

21:48

06-09-2025