“Backend was developed collaboratively, and I handled integration, database design, and API consumption.”

“Our backend is built using ASP.NET Core Web API and follows a layered architecture.”

1. Project Overview (Start Here)

“Smart Documentation Hub is a centralized document management system where users can securely upload, manage, version, and collaborate on documents.”

Goal:

- Secure storage
- Version control
- Collaboration
- Audit tracking

◇ 2. High-Level Architecture

“Our system follows a client–server architecture.”

Components:

- **Frontend** → User interaction
- **Backend (ASP.NET Core Web API)** → Business logic
- **Entity Framework Core** → ORM
- **SQL Server** → Database

👉 *Pause here. Let them absorb.*

◇ 3. Database & ER Diagram (Short Recap)

“The database is designed using an ER diagram with entities like User, Document, DocumentVersion, InlineComments, ActivityLog, and PasswordResetToken.”

Explain in one line:

- Relationships ensure data integrity
- Versioning and auditing are built-in

◇ 4. Backend Architecture (Very Important)

“The backend is structured using a layered architecture.”

Layers:

- **Controllers** → Handle HTTP requests
- **DTOs** → Data contract between frontend and backend
- **Services** → Business logic
- **EF Core (DbContext)** → Database operations

◇ 5. Role of DTOs (Say This Confidently)

“We use DTOs to avoid exposing database entities directly.”

DTOs, or Data Transfer Objects, are simple objects used to transfer data between the frontend and backend.

They contain only the data required for a specific API request or response

DTOs prevent exposing internal database structure and sensitive fields like passwords.

Why:

- Security (no passwords / internal IDs)
- Clean API responses
- Loose coupling between DB and UI

◇ 6. FULL API FLOW (This Is the Heart 💙)

🔁 Common API Flow

Say this slowly 👇

1. Frontend sends request using a DTO
2. Controller receives and validates DTO
3. DTO is mapped to Entity
4. Service layer processes business logic
5. EF Core translates LINQ to SQL
6. SQL Server stores or fetches data
7. Entity is mapped back to DTO
8. Response is sent to frontend

👉 This one flow explains **ALL APIs**.

◇ 7. Authentication Flow

“Authentication APIs handle user registration, login, and access control.”

Flow:

- User sends credentials via DTO

- Backend validates user
- Secure access is granted

Explain:

“Sensitive data is never returned to the client.”

◇ 8. Document Upload Flow

Say this confidently:

1. User uploads document from UI
2. DocumentUploadDTO is sent to API
3. Controller validates file
4. Document entity is created
5. Initial version is stored in DocumentVersion
6. Activity is logged
7. Success response returned

Explain line:

“We store metadata in the database and manage versions instead of overwriting documents.”

◇ 9. Document Versioning Flow

“Each update creates a new document version.”

Flow:

- New version number generated
- Version linked to document
- Old versions remain unchanged

👉 Shows **good design thinking**.

◇ 10. Inline Comment Flow

Explain like this:

1. User selects text range
2. CommentDTO sent with start and end index
3. Comment stored against document version
4. Other users can view comments

Say:

“This enables collaborative editing.”

◇ 11. Activity Logging Flow

“Every important action is tracked.”

Flow:

- Upload
- Edit
- Comment
- Delete

Explain:

“This helps in monitoring and auditing system usage.”

◇ 12. Password Reset Flow

“Password reset is handled securely using tokens.”

Flow:

- Token generated
- Expiry time set
- Token marked used after reset

◇ 13. EF Core & Migrations

Say this line clearly:

“Entity Framework Core Code-First approach is used.”

Flow:

- Models → Migrations
- Migrations → SQL tables
- Schema changes → Version controlled

◇ 14. Error Handling & Validation

Mention briefly:

- Input validation in controllers
- Meaningful HTTP status codes
- Safe failure responses

◇ 15. Final Conclusion (End Strong 💪)

Use this exact closing:

“Overall, the backend ensures secure data handling, clean API contracts using DTOs, proper version control, and scalable architecture using EF Core and SQL Server.”