**Phase II: Connecting Backend and Database to Frontend**

Phase II connects the ASP.NET Core Web API backend with the WordPress frontend (Build in Phase I only) to enable dynamic content management. It includes setting up a database, building an admin interface for uploading content, configuring media storage, and linking the frontend using JavaScript to fetch and display data. Local testing ensures the system works smoothly before deployment.

**Step 1: Set Up ASP.NET Core Backend**

1. Create a new ASP.NET Core Web API project.
2. Define models like Product, Carousel Item, Repository Item, etc.
3. Create API controllers to manage content (CRUD operations).
4. Configure CORS to allow your WordPress frontend to communicate with the API.
5. Ensure HTTPS is enabled during development for secure local API access.

**Step 2: Create SQL Database**

1. Choose a database system (e.g., SQL Server, SQLite, or PostgreSQL).
2. Design tables to store data for products, solutions, publications, videos, etc.
3. Link your ASP.NET Core project to the database using Entity Framework Core.
4. Use code-first or database-first approach based on your team’s preference.

**Step 3: Build Admin Panel (Optional)**

Create a simple admin interface (using any of the following approaches) for managing backend content:

**Option 1: Simple HTML Admin Page**

**How it works:**

1. Create admin user that will upload all data from backend, add relevant pages to upload front end material.
2. Build a plain HTML form with inputs (e.g., title, file upload).
3. Use JavaScript (Fetch API or axios) to send form data to ASP.NET Core Web API.
4. Handle uploads (images/videos), store metadata in the database.  
   Add security layers later — user authentication, file access control, and admin roles.
5. Simple, fast, and gives full backend control — ideal for developers.

**Step 4: Enable File and Media Upload**

1. Configure backend to accept images and video file uploads.
2. Store uploaded files in a server folder (e.g., /content/) or use cloud storage (optional).
3. Save file URLs and associated metadata (e.g., title, description, tags) in the database.
4. Ensure file size and type validation in the backend for security.

**Step 5: Link WordPress Frontend to Backend**

1. Use Elementor’s HTML or Code widget to add custom JavaScript that:
2. Fetches dynamic content (images, video, product details) from your backend API.
3. Displays it in the frontend using HTML structure (e.g., cards, sliders, video players).
4. You can also use AJAX in WordPress or REST API plugins if needed.

**Step 6: Test the Integration**

1. Check that WordPress receives and displays the correct data from the backend.
2. Validate that new content added via the admin panel:
3. Appears correctly on the frontend
4. Can be filtered or searched
5. Loads video and images without issues

**Local Testing Flow (Without Deployment)**

Component Tool/Platform

Frontend Local WordPress (LocalWP / XAMPP / WAMP)

Backend ASP.NET Core Web API (dotnet run in Visual Studio/VS Code)

Database SQL Server Express / SQLite (Local instance)

Integration JavaScript (Fetch API) or WordPress Plugin (e.g.,

Secure CORS configuration in ASP.NET Core

Communication (AllowAnyOrigin or localhost origin)