**1. Azure Resource Setup**

**1.1 Web App**

* **Name**: MyWebAppSp
* **Plan**: App Service Plan (Basic B1 or higher)
* **Runtime**: .NET 8
* **Region**: North Europe
* Deploy the web app using the Azure portal.

**1.2 SQL Database**

* **Name**: SprintEvaluationDB
* Configure tables:
  + **Users**: For user authentication (username, email, password).
  + **Evaluations**: For storing evaluation results (VideoUrl, Score, Feedback, Timestamp).

**1.3 Storage Account**

* **Name**: SprintStorageAccount
* Use Blob Storage for:
  + Uploading and storing videos.
  + Structuring folders by user ID for better organization.

**1.4 Azure Functions**

* **Purpose**: Handle video processing and pose detection logic.
* Write an Azure Function that:
  + Triggers when a video is uploaded to Blob Storage.
  + Calls the pose detection Python API for processing.
  + Stores evaluation results in the SQL Database.

**1.5 Azure OpenAI**

* Integrate GPT for generating detailed feedback.
* API Key and Endpoint: Configure in the Azure portal.

**2. Backend Code Integration**

**2.1 Video Processing API (Python)**

* Ensure the Flask server runs on Azure Container Apps or a local machine with proper endpoints.
* **Endpoint**: /process-video
* The VideoService in C# calls this endpoint for pose analysis.

**2.2 C# Web API**

* Implement your code in the web app backend:
  + Use the VideoController to handle requests.
  + Configure dependency injection for IVideoService in Program.cs.
* Add configuration for:
  + SQL Database connection string.
  + Python API endpoint.

**3. Database Integration**

**SQL Setup**

* Use Entity Framework Core for migrations:
  + Define models in AppDbContext (Evaluation).
  + Migrate to create tables.
* Add connection string in appsettings.json or local.settings.json (Azure Functions).

**4. Workflow Implementation**

**4.1 User Authentication**

* Use Azure SQL Database for storing user data.
* Add authentication middleware to the web app.
* Configure JWT tokens for secure API calls.

**4.2 Video Upload**

* Frontend uploads videos to Blob Storage using a secure SAS token.
* Azure Function triggers processing when a video is uploaded.

**4.3 Pose Detection**

* The Python API processes videos and returns a JSON response.
* Example response: