Capstone Project: Step-by-Step Tutorial

This guide will walk you through setting up and running the Capstone Sports App project, which consists of two separate repositories: the Flutter front-end and the ASP.NET backend. The goal is to make this process as straightforward as possible for users unfamiliar with development tools.

Prerequisites

Before proceeding, ensure you have the following tools installed:

- **Git**: For cloning repositories.
- **Visual Studio Code (VS Code)**: For Flutter development.
- **Visual Studio 2022 or later**: For ASP.NET development.
- **Flutter SDK**: For front-end development.
- **.NET SDK**: Version 8.0 or later.
- **MSSQL**: For the database connection.

> **Note**: It is better to install these tools beforehand, but if not, this guide will go over some steps to install them. For more precise steps or to access the original documentation, links to the relevant resources are provided.

Step 1: Setting Up MSSQL Server

1. **Download and Install MSSQL Server**:
    - Visit the MSSQL Server Download Page.
    - Download the installer suitable for your operating system (e.g., Developer edition for free usage).
    - Run the installer and follow the guided steps. Select the "Basic" installation option for a straightforward setup, or "Custom" if you need more control over the features.
2. **Set Up a New Database**:
    - Install and open **SQL Server Management Studio (SSMS)**.
    - Connect to your local server by choosing the default server name (or entering the one you configured during installation).
    - In the left pane, right-click on **Databases** and select **New Database**.
    - Enter a name for your database (e.g., `SportsAppDB`) in the dialog box and click **OK** to create the database.

3. **Tip**: If you're unsure about the server name during login, you can find it in the SQL Server Configuration Manager under "SQL Server Services".

Step 2: Setting Up Flutter for Windows Applications

1. **Install Flutter SDK**:
    - Visit the Flutter Installation Guide and download the appropriate installer for Windows.
    - Extract the downloaded file to a location of your choice (e.g., `C:\flutter`).
    - Important: When choosing a location to extract the Flutter SDK, avoid directories that require administrative permissions, such as `C:\Program Files`. Using these locations can lead to permission issues when running Flutter commands. Instead, select a location like `C:\flutter` or a directory in your user folder to ensure smooth operation.
    - Add the `bin` folder of the Flutter SDK to your system's PATH environment variable:

        - Search for "Environment Variables" in the Windows Start menu.
        - Under "System Variables," find the `Path` variable and click **Edit**.
        - Click **New** and add the path to the `bin` folder of your Flutter SDK (e.g., `C:\flutter\bin`).
        - Click **OK** to save and close the windows.
2. **Verify Installation**:
    - Open Command Prompt and run:

      flutter doctor
    - This command checks your environment and displays the status of your Flutter installation.
    - Address any issues displayed in the output, such as missing dependencies or tools (e.g., Android SDK or Visual Studio).
3. **Install Additional Tools for Windows Development**:
    - Flutter requires Visual Studio for building Windows apps. Ensure that the following workloads are installed:
        - **Desktop Development with C++**
        - **Universal Windows Platform (UWP) Development**

- If not already installed, open the Visual Studio Installer and add these workloads.

4. **Create a New Flutter Project**:
   - Open **Visual Studio Code**.
   - Ensure the following extensions are installed:
     - **Flutter** (Dart Code)
     - **Dart** (Dart Code)
   - Press `Ctrl + Shift + P` (or `Cmd + Shift + P` on Mac) and type `Flutter: New Project`.
   - Select a folder to save your project and name it.
   - Once created, open the project folder in VS Code.
5. **Clone the Flutter Repository**:
   - Open the terminal in VS Code and run:

     git clone https://github.com/ChristianSaenz/SportsApp-Front-end-.git
   - Navigate to the cloned repository folder.
6. **Install Dependencies**:
   - Open the `pubspec.yaml` file in VS Code.
   - Save the file (`Ctrl + S` or `Cmd + S`), which will automatically download the required dependencies.

7. **Important**: Ensure the versions of the packages specified in `pubspec.yaml` match the ones required by the project. Using newer or older versions of packages may lead to compatibility issues. Stick to the version numbers defined in the file unless updating is explicitly necessary.
8. **Run the Application as a Windows App**:
    - In the terminal, run:

      flutter run -d windows
    - Ensure you are running it as a Windows application, as the browser will not work due to CORS restrictions.

Note: If you encounter issues, run `flutter doctor` again to check for missing dependencies.

Step 3: Setting Up Visual Studio and the Back-End (ASP.NET)

1. **Install Visual Studio**:
    - Download Visual Studio 2022 or later from the official website.
    - During installation, ensure the following workloads are selected:
        - **ASP.NET and web development**

        - **.NET desktop development**
2. **Clone the Backend Repository**:
    - Open **Visual Studio**.
    - Click **Clone a Repository** and enter:

      https://github.com/ChristianSaenz/SportsApp-Backend-.git
3. **Open the Project**:
    - Once cloned, open the solution file in Visual Studio.
4. **Install Dependencies**:
    - Dependencies should restore automatically. If not, navigate to **Tools > NuGet Package Manager > Manage NuGet Packages for Solution** and install any missing packages.
5. **Important**: Make sure the versions of the dependencies match those specified in the project. Mismatched versions can lead to runtime errors or unexpected behavior.
6. **Configure the Database**:
    - Open the **Query** folder in the cloned backend repository.
    - Use the `create_db.txt` file to create tables in your database:
        - Copy the SQL commands from the file.
        - In SSMS, select your database and click **New Query**.
        - Paste the commands and execute them.

- Repeat the process with the data population queries.

7. **Set the Connection String**:
   - Locate the `appsettings.Development.json` file.
   - Replace the connection string with your database details:

User Authentication Example:

"ConnectionStrings": {

"DefaultConnection": "Server=localhost;Database=SportsAppDB;User Id=root;Password=your_password;"

- }

Windows Authentication Example:

"ConnectionStrings": {

"DefaultConnection": "Server=localhost\SQLEXPRESS;Database=SportsAppDB;Trusted_Connection=True;TrustServerCertificate=True;"

- }

8. **Run the Backend**:
   - Click **Run** in Visual Studio to build the project.

- If successful, Swagger should open in your default browser, displaying API documentation. Note the port number (e.g., `https://localhost:5001`).

Step 4: Connecting the Front-End to the Back-End

1. **Update the API Endpoints**:
   - In the Flutter project, navigate to the `ApiHandler` file in the `handlers` folder.
   - Replace the port numbers in the endpoints with the port number from your backend Swagger URL.
2. **Run the Application**:
   - Restart the Flutter application to apply the changes.

Troubleshooting Tips

- **Flutter Issues**: Run `flutter doctor` to identify missing dependencies.

- **Database Errors**: Verify the connection string and ensure MSSQL Server is running.
- **Build Errors**: Double-check that all dependencies are installed in Visual Studio.
