Yes — React + C# + SQL is a **very solid and recommended stack** for the kind of app you're describing. Here's why it works well and how the pieces fit together:

**🧱 Stack Breakdown for Your Character Profile App**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Technology** | **Role** |
| **Frontend** | **React** | Build a dynamic, responsive UI for creating and viewing character profiles |
| **Backend** | **C# (ASP.NET Core)** | Handle API requests, business logic, and data validation |
| **Database** | **SQL Server or SQLite** | Store character data persistently (names, traits, backstories, etc.) |

**✅ Why This Stack Is a Great Fit**

* **React** gives you a fast, component-based UI — perfect for forms, lists, and profile views.
* **C# with ASP.NET Core** is powerful, scalable, and integrates easily with SQL databases.
* **SQL** gives you structured storage and querying — ideal for filtering characters by traits, tags, or timelines.
* You get **strong typing** and **tooling support** across the stack (especially with Visual Studio and VS Code).
* It’s a **real-world architecture** — great for your portfolio and transferable to enterprise work.

**💡 Project Feature Ideas**

* Create/edit/delete character profiles with fields like name, age, species, backstory, traits
* Tag characters by genre, alignment, or universe
* Search and filter by attributes (e.g., “all elves from the fantasy genre”)
* Add image uploads or markdown bios
* Save favorites or group characters into story arcs

**🧪 Bonus: Local Dev Setup**

* Use **SQLite** for quick local development (no server setup)
* Use **Entity Framework Core** for easy C# ↔ SQL mapping
* Use **Axios or Fetch** in React to call your C# API endpoints

If you’d like, I can help you scaffold the project structure, set up the API routes, or design the database schema. Want to start with the backend or frontend first?