**Tech Stack Overview**

* **Backend:** C# with .NET Web API
* **Frontend:** React with Vite
* **Database:** MongoDB (NoSQL for flexibility in CV structure)

**Project Plan**

**1. Backend (C# + .NET)**

* Use **ASP.NET Core Web API** to create RESTful endpoints.
* Implement **JWT authentication** for user management.
* Use **MongoDB.Driver** to interact with the database.
* Define models for **User**, **CV**, and **Templates**.
* Implement **CRUD APIs** for managing CVs.

**2. Frontend (React + Vite)**

* Use **React Router** for navigation.
* Use **Tailwind CSS** or another UI library for styling.
* Implement **Redux or Zustand** for state management.
* Build a **drag-and-drop CV editor** using libraries like react-beautiful-dnd.
* Integrate with the backend using **Axios or Fetch API**.

**3. Database (MongoDB)**

* Store user accounts with hashed passwords.
* Store CVs in JSON format for flexibility.
* Store predefined CV templates.

**Key Features**

✅ User authentication (Register/Login)  
✅ Create, edit, and delete CVs  
✅ Use pre-defined templates  
✅ Export CV as PDF  
✅ Share CV via link  
✅ Drag-and-drop interface

# Apps Configuration:

## Create a New .NET Web API Project

dotnet new webapi -n CVBuilderAPI

## Install Required NuGet Packages

dotnet add package Microsoft.AspNetCore.Mvc.Core

dotnet add package MongoDB.Driver

dotnet add package Swashbuckle.AspNetCore

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

## 1.3 Configure MongoDB Connection



## 2.1 Create a New Vite React App

## npm create vite@latest cv-builder-frontend --template react-ts

## cd cv-builder-frontend

## npm install

## 2.2 Install Dependencies

npm install tailwindcss postcss autoprefixer

npx tailwindcss init -p

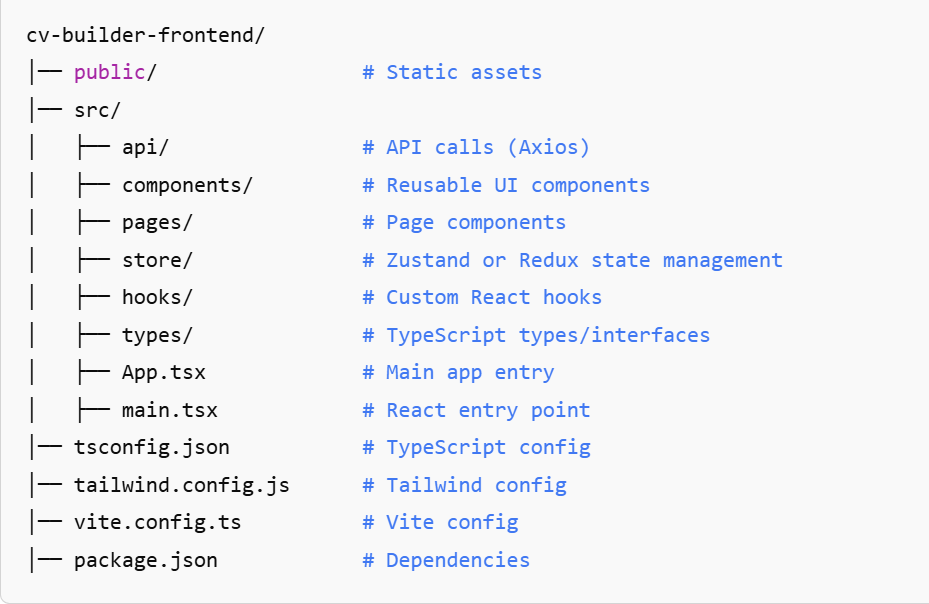
npm install react-router-dom

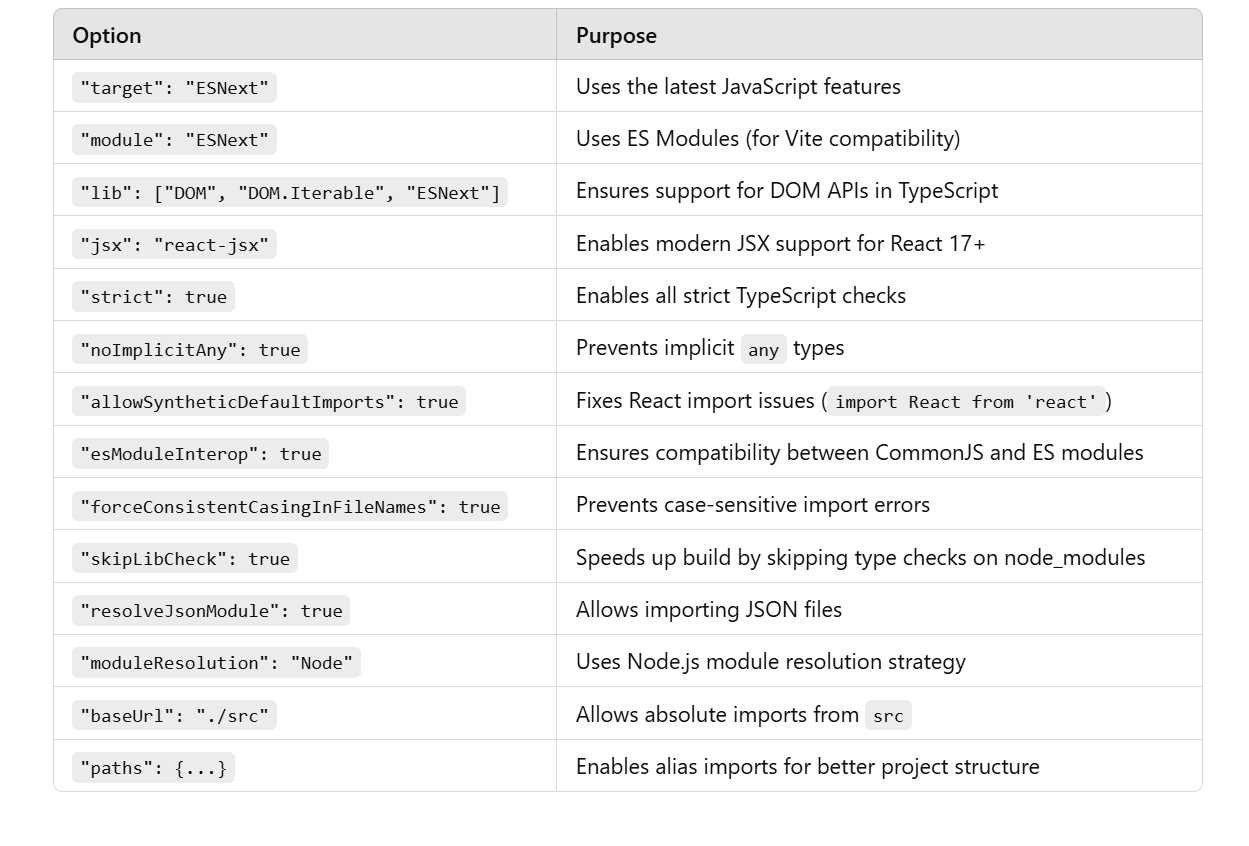
npm install axios

npm install zustand

npm install @reduxjs/toolkit react-redux

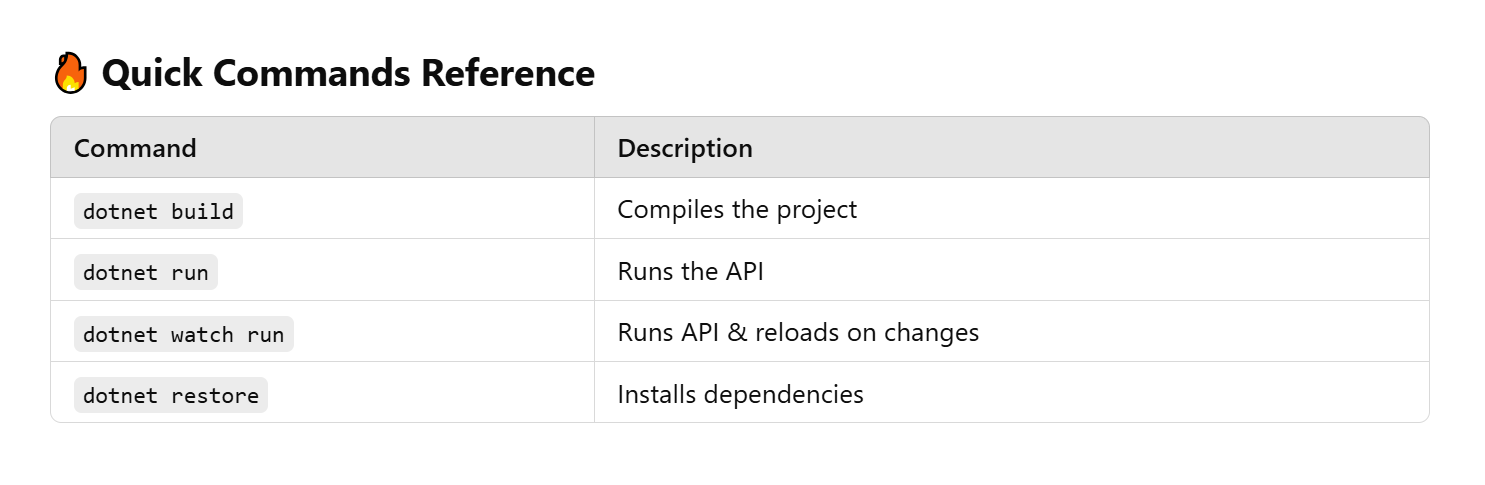
npm install @mui/material @emotion/react @emotion/styled





## 3.1 Run MongoDB using Docker

docker run -d -p 27017:27017 --name mongo-container mongo



**Next Steps**

* **Implement ICvService and CvService** for handling CV data
* **Create MongoDB models** for storing CVs
* **Build controllers** to expose API endpoints