**Overview**

This project implements an API and a frontend application as per the assessment requirements. The solution consists of:

1. **Backend**: A .NET 8 Web API using Clean Architecture to fetch data from Azure Blob Storage.
2. **Frontend**: A React application using Material-UI to display hierarchical data.

**Technologies Used**

* **Backend**: .NET 8, C#, Azure SDK, ASP.NET Core Web API
* **Frontend**: React, Material-UI, Axios
* **Infrastructure**: Azure Blob Storage

**Setup and Installation**

**Backend (.NET 8 API)**

1. Clone the repository.
2. Navigate to the backend project folder (PropertyAPI).
3. If you are using Visual Studio, just build and run the project, then go to step #8. If using VS Code, follow steps 4 to 7.
4. Install dependencies:
5. dotnet restore
6. Run the application:
7. dotnet run
8. The API should be accessible at http://localhost: 7174/api/properties.

**Frontend (React Application)**

1. Navigate to the frontend project folder (PropertyApp\react-app).
2. Install dependencies:
3. npm install
4. Run the application:
5. npm start
6. Open http://localhost:3000 in a browser to view the UI.

**API Details**

**Endpoint: Retrieve Properties**

* **URL**: /api/properties
* **Method**: GET
* **Response**: Returns structured JSON data of properties and nested spaces with rent roll.

**Assumptions**

* The SAS token provided has read access to the blob.
* The JSON structure remains consistent with the provided example.
* Error handling is implemented for missing/invalid data.

**Error Handling**

* Handles missing blob data gracefully.
* Returns appropriate status codes for errors.
* Logs exceptions for debugging.

**Conclusion**

This project successfully meets all the assessment criteria and follows best coding practices in both backend and frontend implementations.