



MY HEALTH SOLUTION

Hamza ijaz



Table of Contents

| | |
|---|---|
| OVERVIEW | 2 |
| GOAL | 2 |
| Required Software | 2 |
| How to set up Database? | 2 |
| How to set up Backend (FunctionApp)? | 3 |
| How to set up Frontend (react project)? | 3 |
| Backend Project structure? | 4 |
| Structure of Frontend | 6 |
| What extra have I done? | 9 |
| Improvement for Future | 9 |

OVERVIEW

MyHealthSolution is an end-to-end fully functioning project to store patients in a database table, and to display a list of patients on a website. The whole product consists of 3 parts:

1. Front-end website (ReactJS)
2. Back-end microservice (C# ASP.NET | Azure Function)
3. Database (MS SQL)

GOAL

To have a functional patient management system for a hospital

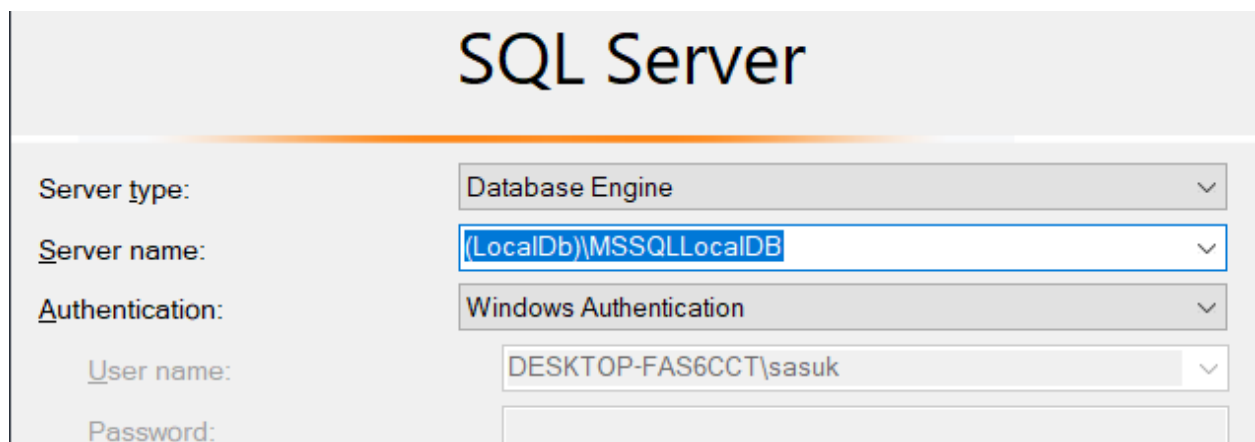
Required Software

In order to set up, you need to have following software:

1. Visual Studio or a similar IDE (or VS Code)
2. MS SQL Server Management Studio

How to set up Database?

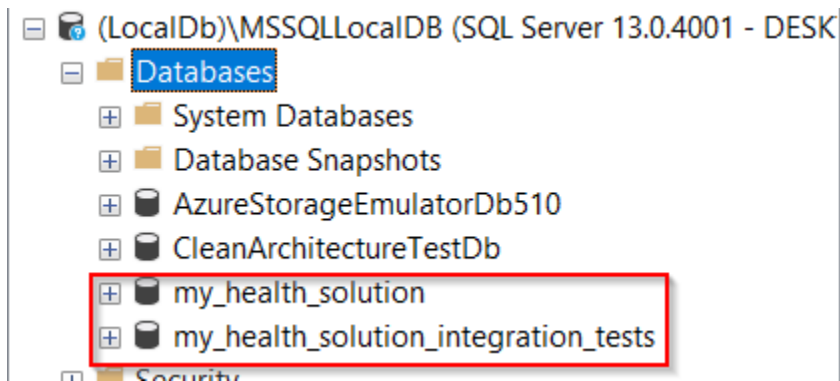
In order to set up local database, set servername to “(LocalDb)\MSSQLLocalDB”



The screenshot shows the 'SQL Server' configuration window. The 'Server type' is set to 'Database Engine'. The 'Server name' is set to '(LocalDb)\MSSQLLocalDB'. The 'Authentication' is set to 'Windows Authentication'. The 'User name' is set to 'DESKTOP-FAS6CCT\sasuk'. The 'Password' field is empty.

Create a new database named: my_health_solution

Create another database for integration tests: my_health_solution_integration_tests

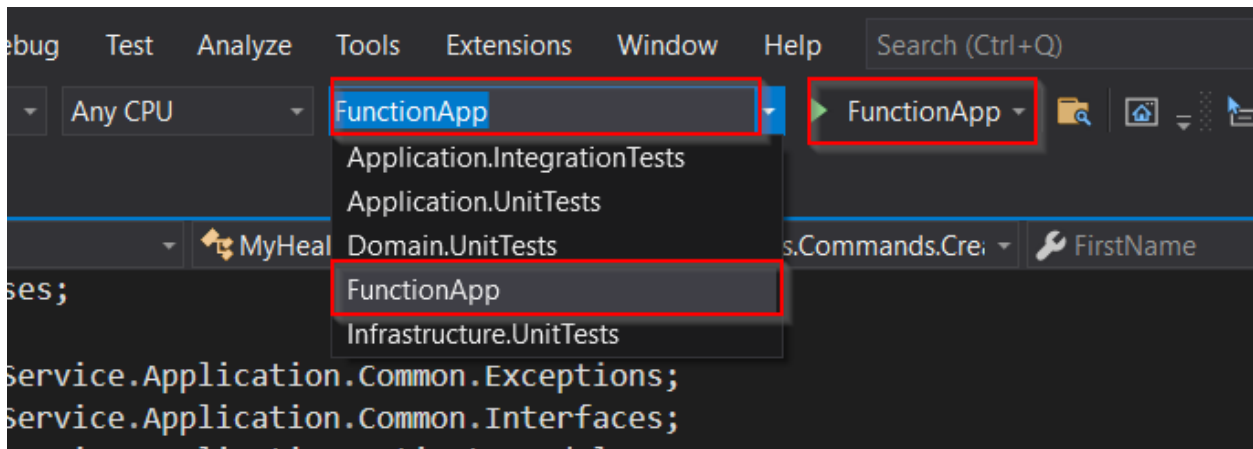


Create a new query on both databases. Paste from following file:
\\sql\\schema\\002_CreatePatientTable.sql

This will create a new table in the database.

How to set up Backend (FunctionApp)?

When you load the solution in Visual Studio, must select FunctionApp project from top.



When you will run it, Visual Studio Debug Console will start hosting.

How to set up Frontend (react project)?

You must have node installed on your machine. If you don't have, install from here
<https://nodejs.org/en/download/>

Once you have the code in your PC, open command line as admin and navigate to
\\healthsolution\\healthsolution

Then run: npm start

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.870]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd /d D:\hamza\healthsolution\healthsolution
D:\hamza\healthsolution\healthsolution>npm start
```

This should start the development server. The website will be opened automatically on your browser. If it does not open, type in browser: **http://localhost:3000**

```
Administrator: Windows PowerShell
Compiled successfully!

You can now view healthsolution in the browser.

Local:      http://localhost:3000
On Your Network: http://192.168.1.103:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

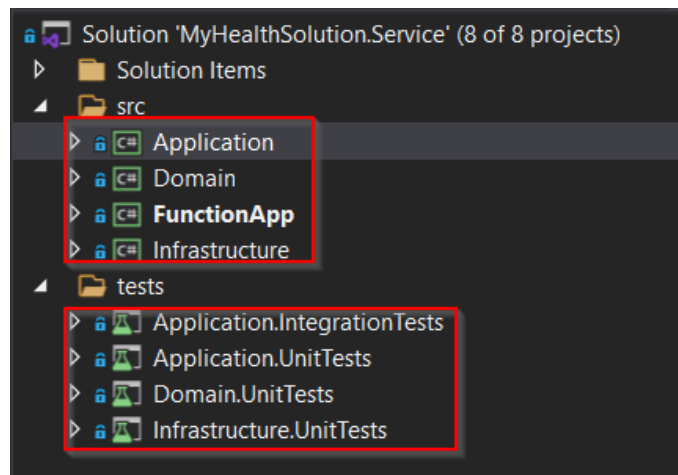
You will be able to see any build errors in Command Line.

Hopefully you will just need to install: `npm install --save react-router`

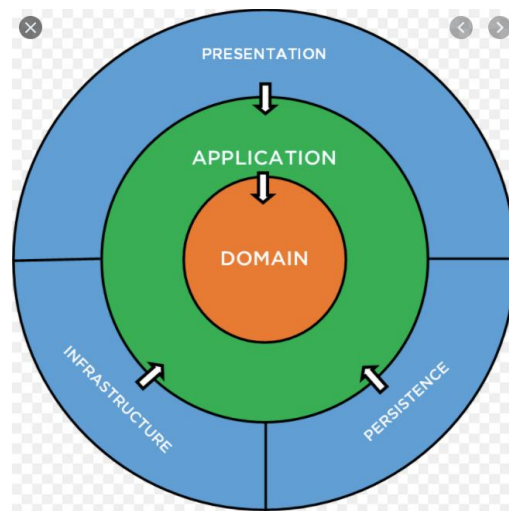
`npm install --save react-router-dom`

Backend Project structure?

The solution contains 4 projects and 4 test projects for them.



You might have heard of “Clean Architecture”. That’s what is used here.



There are few layers in the solution:

1. Domain (entities)
2. Application layer (contains all logic)
3. Presentation Layer (in our case, FunctionApp)
4. Infrastructure and persistence (helper services in our case)

Whenever a call is made to the code, the endpoint which is triggered is in FunctionApp.

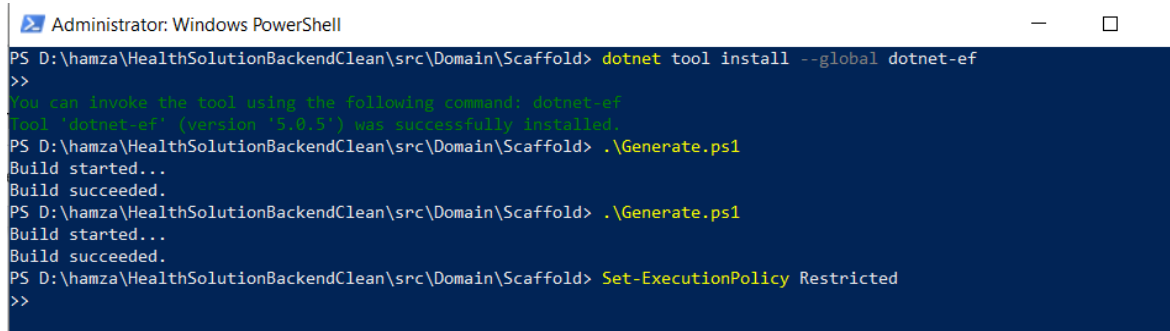
FunctionApp then passes that request to the application layer through MediatR. Application layer then calls the database to get/put required data, and returns its response to the function app.

Application layer also does the validation on requests.

EntityFramework is used in this project. I took “Database First” approach, then scaffolded to generate Domain Entities. In order to do that, I had to do the following:

I opened powershell as admin. Then:

1. Went to directory: \src\Domain\Scaffold
2. Ran this to enable running scripts on my system: Set-ExecutionPolicy RemoteSigned
3. Ran: Install-Module sqlserver
4. Then: dotnet tool install --global dotnet-ef
5. Then: **.\Generate.ps1** (this generates entities and automatically puts those into \src\Domain\Entities)
6. After running script, you may disable scripts again: Set-ExecutionPolicy Restricted

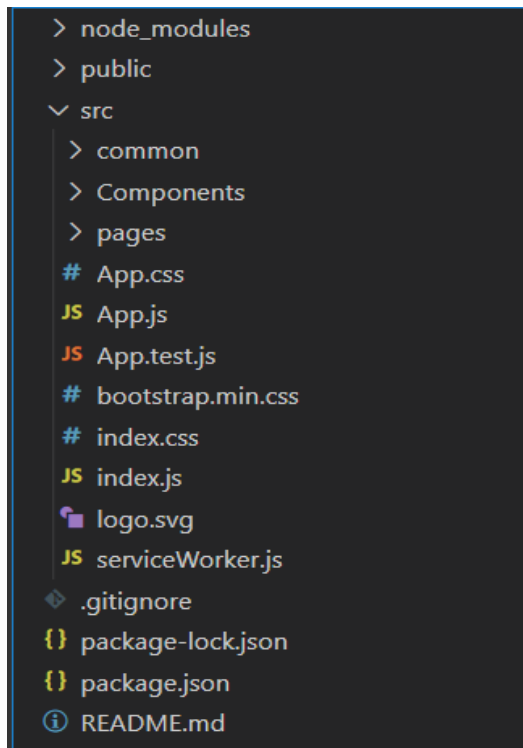


```
Administrator: Windows PowerShell
PS D:\hamza\HealthSolutionBackendClean\src\Domain\Scaffold> dotnet tool install --global dotnet-ef
>>
You can invoke the tool using the following command: dotnet-ef
Tool 'dotnet-ef' (version '5.0.5') was successfully installed.
PS D:\hamza\HealthSolutionBackendClean\src\Domain\Scaffold> .\Generate.ps1
Build started...
Build succeeded.
PS D:\hamza\HealthSolutionBackendClean\src\Domain\Scaffold> .\Generate.ps1
Build started...
Build succeeded.
PS D:\hamza\HealthSolutionBackendClean\src\Domain\Scaffold> Set-ExecutionPolicy Restricted
>>
```

Tests have also been written. Just set up the integration tests database as specified above, and run all the tests.

Structure of Frontend

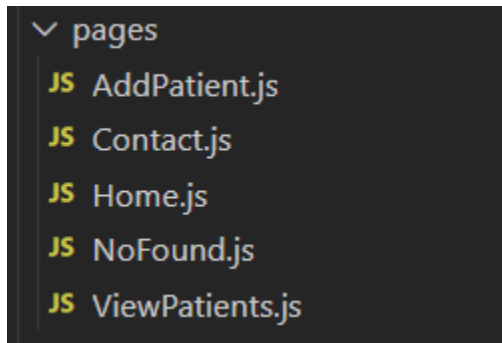
Frontend is a create-react-app. It has following structure:



```
> node_modules
> public
v src
  > common
  > Components
  > pages
  # App.css
  JS App.js
  JS App.test.js
  # bootstrap.min.css
  # index.css
  JS index.js
  logo.svg
  JS serviceWorker.js
.gitignore
package-lock.json
package.json
README.md
```

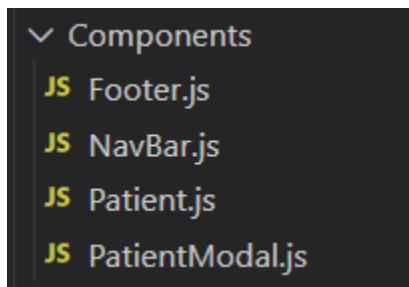
It is a single page application (SPA), but consists of 5 virtual pages:

1. Home
2. AddPatient
3. ViewPatient
4. Contact
5. NotFound



There are 4 components used:

1. NavBar
2. Footer
3. Patient
4. PatientModal



React-router-dom is used to navigation between pages

HTTP calls are made through axios. Currently the calls are being hardcoded to baseURL <http://localhost:7071/api> because this is for our functionApp. But this can be made generic in future.

New patients can be added in AddPatient page, and the stored ones can be viewed in ViewPatients page. There is a form for adding new a new patient. It also does captcha verification. For viewing the stored patients, clicking the “View Patient” makes a GET call to our API and get a list of stored patients, and displays few details of that patient.

John Snow

Date of Birth: 1991-02-02

[click to get details](#)

Mike Hussey

Date of Birth: 1985-05-01

[click to get details](#)

Clicking on a specific patient opens a new modal to show more details of that patient.

First Name: John

Last Name: Snow

Date of Birth: 1991-02-02

Gender: male

Health Cover Type: bupa

Policy Number: 123456

Street Address: 300 Collins St

Suburb: Melbourne

Post code: 3000

Close

Apart from this, there is a nav bar and a sticky footer.

What extra have I done?

1. I have created **integration tests and unit tests**.
2. I have implemented **Captcha validation** from frontend and backend.

Improvement for Future

There is always room for improvement. For this product, there can be many improvements made in future. For example:

1. Currently I have created on **GET Patient, POST Patient, and PUT Patient functions** due to lack of time. However, in reality, we should also have **DELETE patient** function to remove a required patient. Moreover, PUT Patient (Update Patient) is currently only in backend, and not implemented in front end due to time limitations
2. We should have the ability to record any allergies, medical conditions, or disabilities a patient has. This can be easily done by adding few more fields in Database table, backend, and frontend form.
3. I have intentionally left room for more client-side validation.
4. I have done server-side validation but there is also some room for that. For example, the input fields should be of length which database allows. E.g., Postcode should not be more than 10 characters. It should only be numerical. I have skipped such detailed validation intentionally.
5. The front end can have username/password validation in future
6. If I had some extra time, I could have done what is mentioned above and much more.