Coding Competency Task - Solution

# Task

This task is to produce a simple user registration and login page that will return a JSON Web Token (JWT) for authentication with the API for further requests and persist the data in the SQL database. The user registration page should only require:

* Username / Email
* Password

The purposes of proving understanding of concepts, software principals and frameworks involved. You should demonstrate the following:

* Best practices, possibly with comments on why.
* SOLID principals.
* RESTful APIs.

# Technologies Used

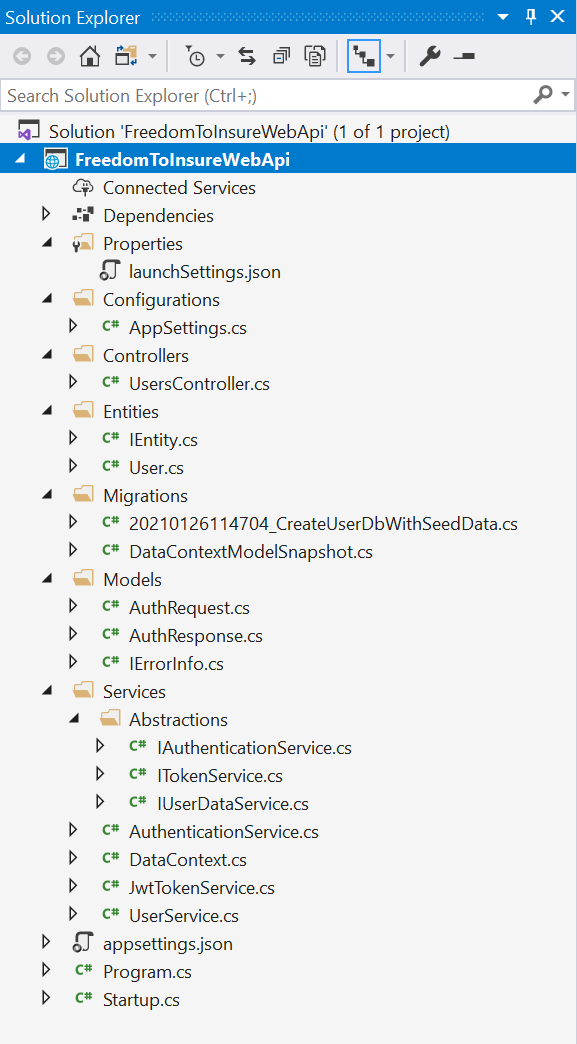
* Visual Studio 2019.
* Visual Studio Code
* SQL Server 2019.
* .NET Core 3.1
* ReactJs
* Material UI
* Entity Framework 3.1.0

The test should be completed with Visual Studio 2019 and Visual Studio Code. You can download both of these tools for free from <https://visualstudio.microsoft.com/downloads/>. SQL Server development edition can be used for free too and is an equivalent of the enterprise edition, this can be downloaded from <https://www.microsoft.com/en-gb/sql-server/sql-server-downloads>.

# FreedomToInsurceWebAPI – Back End

MS.NET Core web API project is created to develop the User web api service. The project structure (snapshot below) is self explanatory.The api has following endpoints/routes to demonstrate the JWT (JSON Web Token) authentication :

* **/users/authentic –** This public route accepts the http POST request containing “username” and “password” in the body. If the authentication is successful then it returns the user details and newly generated JWT authentication token in the response header.
* **/users/all** – Secure route that accepts http GET request and returns all the users in the database if the request header contains a valid JWT token. If the token is expired or invalid then 401 unauthorized response is returned.
* **/users/{id}** – Secure route that accepts the http GET request and returns the user with the specified id.

. 

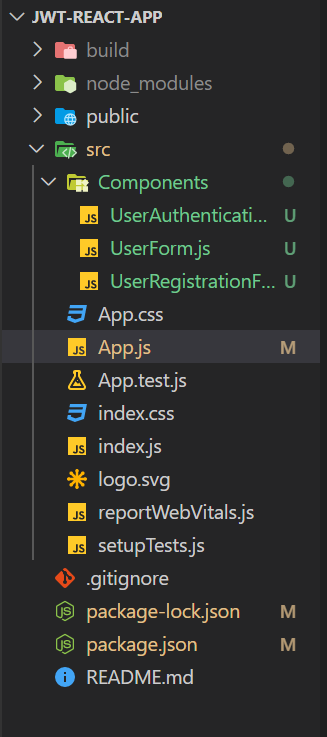
# Compilation & Execution – Back End

Steps to compile and run the web API locally are as follows:

* Download/clone the project from github.
* Open the solution file ***FreedomToInsureWebApi.sln*** in VS 2019.
* Change the db connection string in the ***appsettings.json*** file and point to the appropriate database for entity framework’s migrations to work properly.
* Compile the solution so that it downloads the desired NuGetPackages and other dependencies.
* Run the command = ***dotnet run*** from command prompt in the project folder (where the FreedomToInsureWebApi.csproj file is located). You should the message “***Now listening on http://localhost:6060***”.

# JWT-React-App – Front End

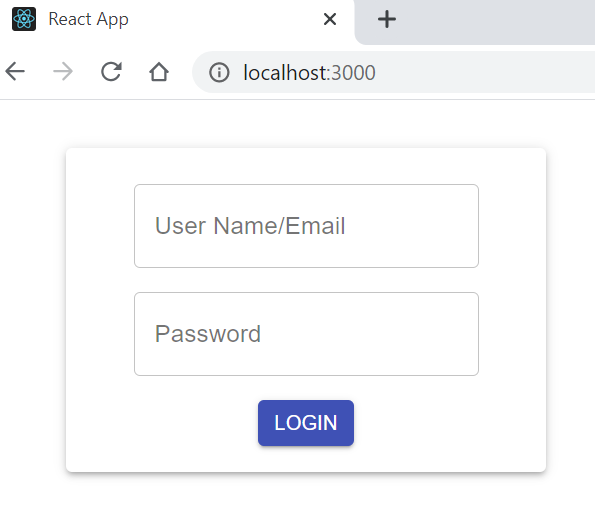
The front end is developed in ReactJs and Visual Studio Code. React uses component-based approach and React Hooks. The self-explanatory project structure us shown below:



# Compilation & Execution – React App Front End

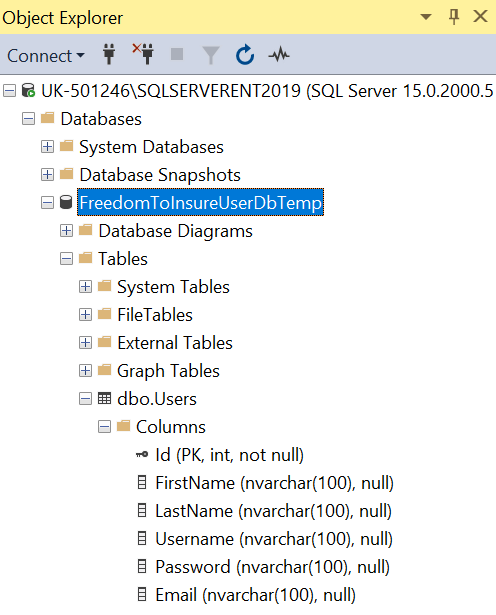
Steps to compile and run the web API locally are as follows:

* Download/clone the project from github.
* Download/install **CORS** browser plugin and enable it by clicking on the “C” icon in the browser. This plugin is required only if the API access is blocked by Cross Origin Resource Sharing (CORS) policy. All details about the CORS issue and it’s fix and source of download are mentioned at this URL = <https://chrome.google.com/webstore/detail/allow-cors-access-control/lhobafahddgcelffkeicbaginigeejlf?hl=en>
* Open the folder ***jwt-react-app*** in Visual Studio Code.
* Open the terminal and run **npm install** command.
* After the installation is finished, run **npm start**  command.
* After successfully running both commands, open the web browser (if it is not automatically opened by the **npm start** command) and paste URL = [***http://localhost:3000***](http://localhost:3000)and press “Enter”. It display the login page from where only 2 input fields as shown below. The default credentials to start the flow of the application with JWT token are
  + User Name = **admin**
  + Password = **admin**



# Database Details

In order to keep the back end API layer simple, entity framework migration approach is used. It creates the desired database ***FreedomToInsureUserDB*** in the SQL server. This database have only 1 table called ***User*** with the default admin user details (see data). The snapshot is shown below:



# Application Flow – Front/Back End

The flow of the application is as follows:

1. Login using the default credentials i.e. username = ***admin*** , password = ***admin***
2. React app will send the http POST request to our web API end point [**http://localhost:6060/Users/authenticate**](http://localhost:6060/Users/authenticate)**.**
3. On success, It will return the authenticated user details along with the JWT token which is stored in local storage so that it can be used in further requests securely.
4. If the user is authenticated, the next page should be registration page. Enter details for new user and click “Register” button. The app will send the http POST request with the existing JWT token (acquired when user was authenticated) to the web API end point [**http://localhost:6060/Users/register**](http://localhost:6060/Users/register)**.** On success, it will return the newly created user details.

At the moment due to shortage of time and for simplicity, the popups and user data grids are not implemented in React UI. The results can seen in the console windows of browser developer tools windows. Example snapshot is shown below:



PostMan API testing tool can also be used to see the results of the web API