Project Architecture

1. **MEAN stack** has been used to develop the application.
2. Server code is in the **/api** folder
3. Client(angular) code is in the **/client** folder
4. **app.js** is the entry point of the application

Setup

Requirements
1. Node.js
2. Mongodb

Setup the Server
1. Open the project root folder in the terminal and do **npm install**
2. Then run **npm start**

Setup the Client
1. Open the /client folder in the terminal and do **npm install**
2. Then run **npm start**

Setup the Database
1. Create a mongodb database called **assignment** on the local machine

Finally, open the application at **http://localhost:4200**

Pending points

Some points I could not manage to finish due to shortage of time. These are mentioned below -

1. Editing(changing) the 2 images from the profile page and changing the default image option. However, the other fields can be edited.
2. Allowing multiple skills to be entered into multiple text boxes.
3. Handling duplicate images even if the names are different. For this, I would have compared the md5 hash of the 2 images.
4. Finally, I could have also done some more unit testing, code optimization and refactoring.

Some extra features implemented

1. JWT based authentication
2. Auth Guard in angular

Explanation of the caching feature

I wanted to cache responses to GET requests in the in memory cache. However, as I have used JWT, the information(token) to differentiate each GET request is send in the header and not part of the URL(like /users/1). So I implemented this in a different manner.

If you hit this URL – localhost:3000/cache-test/1
(where 1 is the parameter that changes), the server sends out a "Hello from the server" text response. First time, I have added some delay to simulate fetching without cache. Next time onwards, for the next 10 seconds the response is cached. After 10 seconds, the cache expires and if this URL is rqeuested again, the cache will not be used.

Another GET request to be cached can be like localhost:3000/cache-test/2 and so on

Also, there is a limit that only 3 items can be set in the cache. If a 4th item wants to be cached, it must wait for some of the earlier items to expire.

So items can be added and removed from the cache, the cache has a size limit(3 items) and it can become invalidated(after 10 seconds). The cache is also in-memory and in the same process as the node server