**A One Pager for the Intelligent Automation Garage (IAG) portal**

**Alpha Mackie**

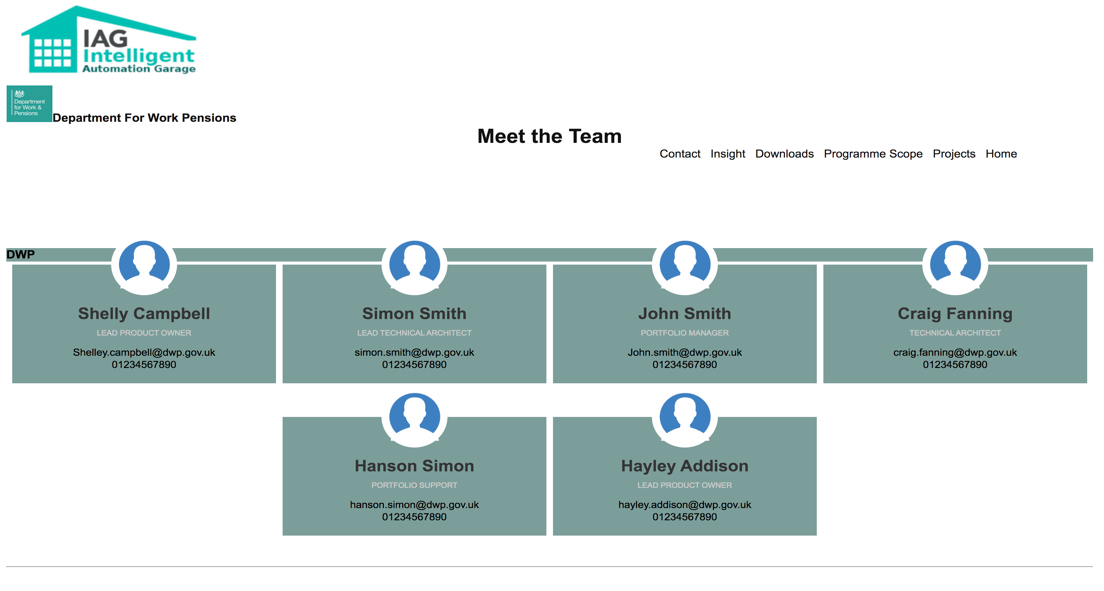
**Software Developer Level 4**

Introduction

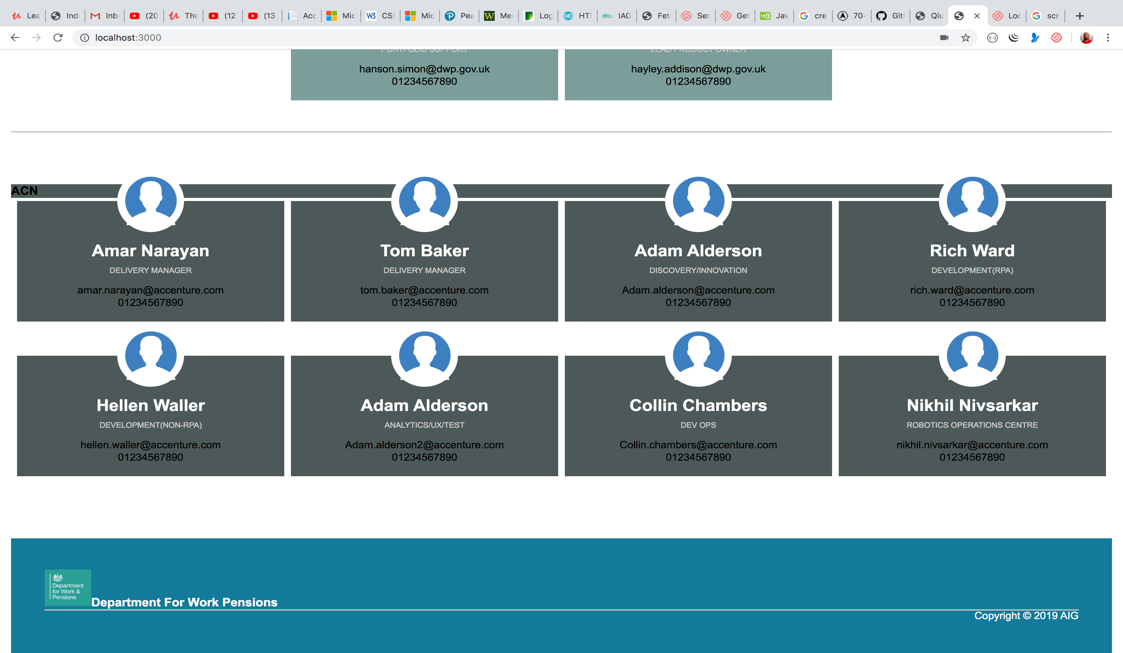
I was tasked with creating a webpage that displays members of staff in the IAG department. The webpage should pull the staff data from a database and display that data on the page dynamically. The page should display the Staff Name, Role, Email and Phone number. Our Delivery manager stated that this page might be integrated into the IAG portal at a later stage.

First step

Before I started this project, I had a discussion with one of my colleagues who created and usually maintains the IAG portal. He gave me some advice as to what technologies I needed to be familiar with in order to accomplish this task successfully. Therefore, I started off learning Nodejs and Express in order to create a server, Mongo DB and mongoose which he recommended for the database. This was a challenging process for me as I have never used these technologies before apart from Express. With his guidance I was pointed in the right direction to learn just what I needed in order to complete the project as per requirements. Please see a snapshot of the finished project below:



*Figure 1a: The Automation Intelligent Garage (IAG) staff page.*



*Figure 1b: The Automation Intelligent Garage (IAG) staff page.*

Development process

The development environment I used was Visual Studio Code (VS Code). I started off by implementing the Model, View and Controller (MVC) architecture. Even though this is just a one-page web project, it has enabled me to get a better understanding of this architectural pattern for when faced with a large and more complex application. I therefore created and named my folders as such. I also created a Public folder with a static file called style.css and an images folder that contained all the logos and images.

The User Interface (UI)

I created a simple HTML file and saved it with an ‘ejs’ extension to get the ‘ejs’ code to work in it. I structured the page such that the look and feel is like the other pages in the portal, very clear and not crowded with text. I used header tag for the main header for the page, a ‘nav’ tag for the navigation links but rendered them inactive as the page is a standalone page for now. The ‘div’ tags in the body have a class of wrapper for the contents in the body and a footer tag with a class of ‘main-footer’. The colour scheme for the header, body and footer, the font sizes, the position and size of the logos all look uniform, as instructed by my colleague that maintains the portal.My colleague advised that I made sure everything matches the pages of the IAG website, as the layout of the other pages already satisfy the standards for user experience, for example, the visually impaired user. In terms of visually impaired User Experience, I made sure that for images and logos, I applied the ‘alt tag’ with a description of the image or logo. I also contrasted the background and foreground colours, e.g. black text on a white background, white text on a dark background. I applied that in the external CSS file called ‘style.css’. Please see a sample images (figures 2, 4a and b) below:

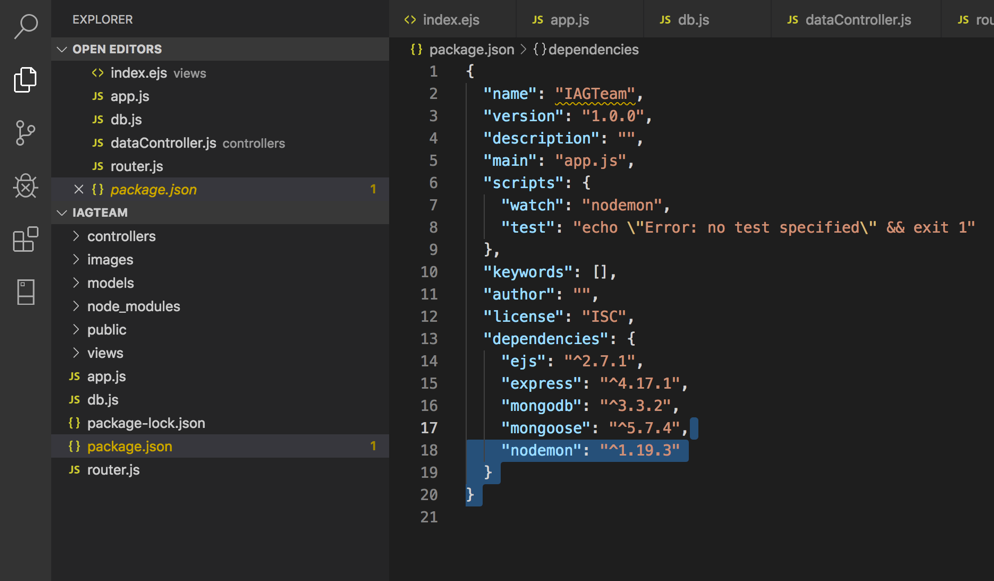
**A screenshot of a cell phone

Description automatically generated**

*Figure 2: Part of Css file showing some of the styling of the page.*

I initialised the project by using the ‘npm init ‘command on the terminal in VS code; this created a node\_modules folder containing all the sub folders and files I needed for the project, a package-lock.Json file and a very important file called package.json where all the dependencies and the script to run the server will reside. I then installed EJS, Express, Mongo db, Mongoose and Nodemon using the npm install command on the terminal in VS Code.

Please see image below:



*Figure 3: Package.json file showing all the dependencies.*

I then setup the server in the ‘app.js’ file and brought in all the packages I needed to get the server up and running by requiring them. I created the HTML file and named it a ‘.ejs ‘ file as it will contain the EJS templating language commands, it lets you generate HTML markup to display the data from the database dynamically and use a JavaScript forEach loop to loop through the data fields, rapped in EJS syntax. I also setup the view engine, use the EJS file as my view engine which is in the views folder and specified what port the server is to listen on, which is port 3000.

Please see the images below:

A screenshot of a computer

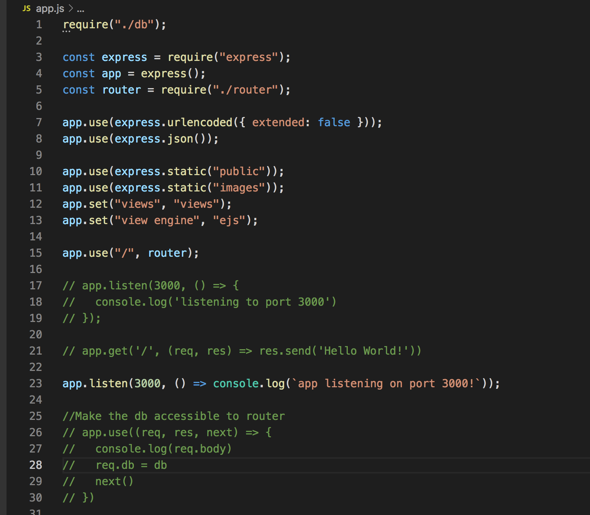
Description automatically generated

*Figure 4a: index.ejs file which contains the HTML markup.*

A screenshot of a cell phone

Description automatically generated

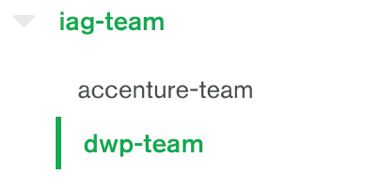
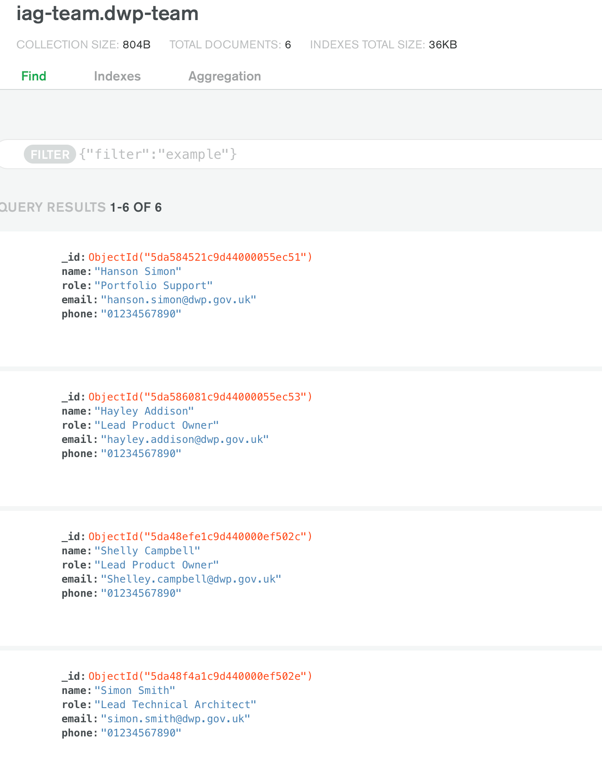
*Figure 4b: Index.ejs file which contains the HTML markup.*



*Figure 5: App.js file.*

My next step was to create the Database using a cloud database application called Mongo Atlas. I named the database the ‘iag-team’ which consisted of a collection and two documents. In Mongo db terms, a table is known as a collection and documents are analogous to rows in other forms of databases like SQLDB. I also created a file called router.js which has the route to the data in the Mongo db.

Please see image below:

*Figure 6: Mongo DB database displaying 2 collections and showing 4 documents in the Accenture-team collection*.

I then created two files for the database schemas called dwpteam.js and accentureteam.js respectively as shown below:



*Figure 7a: Database Schema*



*Figure 7b: Database schema.*

Project Challenges

The major challenge I had in this project was establishing a database connection to get data on the webpage. As seen on figure 9a, the commented-out code was the initial code I wrote but output the error as seen on figure 8a. I added the missing closing bracket and then got a new error that mongoose was not defined. I then declared mongoose and got another error stating that the connection string was not defined. After these attempts in solving the problem, I decided to read through the mongoose documentation and watched a tutorial on the topic. I then rewrote the code as seen on figure 9a which worked.

Text

Description automatically generated

*Figure 8a: Syntax error while trying to establish a database connection.*

Text

Description automatically generated

*Figure 8b: Mongoose not defined.*

Text

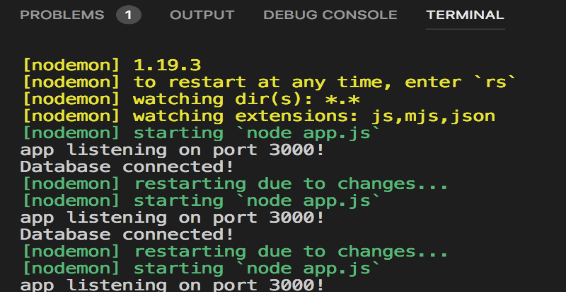
Description automatically generated

*Figure 8c: Connection string not defined.*

I created the ‘db.js’ file and required the mongoose package to setup the database connection string which I copied from mongo db Atlas when I was setting up the database. I then ran the ‘npm run start’ command to get the server and the database are up and running successfully as shown in the images below:

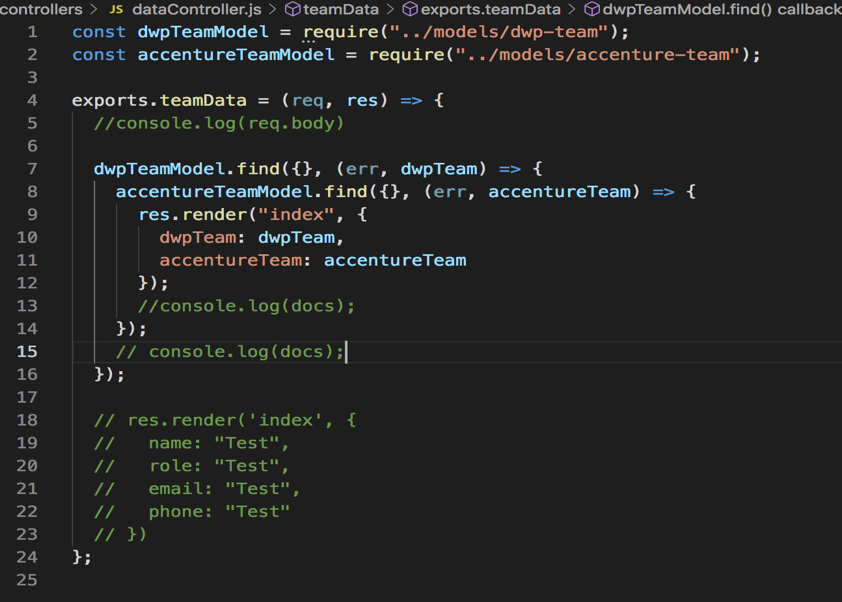


*Figure 9a:* *Database connection string.*

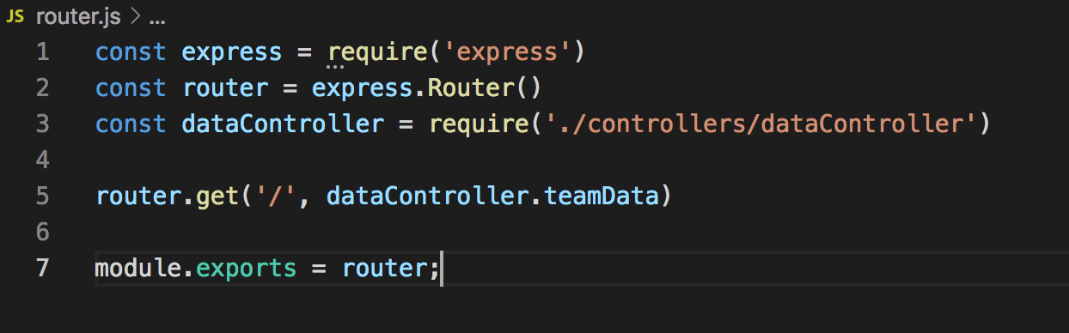


*Figure9b: A successful connection and message on Vs Code terminal.*

I created a dataController file, required the two database schema files within it that models the data on the database and rendered in the webpage, finally the router.js file that creates the route. Please see the two files below:



*Figure 9c: DataController file.*



*Figure 10: The Router.js files.*

Final step

After completing the project, I presented it to my senior developer who then did a manual test, similar to what I did after completion to ensure that it works, by loading the code on the vs code environment, typing on the command line ‘npm run watch’ to get the server and the database up and running. He then opened a browser and typed ‘localhost:3000’ on the address bar, which then displayed the full page. He was very satisfied that it worked, approved it and also presented it to the delivery manager who commented on what other features he would like to see on the page and stated that this could be done at a later stage when he makes the decision to integrate it into the portal. At the moment, besides the navigation menu on the top-right of the page which I disabled, there are no other interactive features on the page as it is more or less a static page. As seen above on figure 1a and b, the page displays all the data dynamically and if more staff data is added to the database it will be displayed on the page dynamically.

Conclusion

In this project I learnt a great deal in terms of the technologies that I have been exposed to such as JavaScript which is the programming language I used and applied the camel case naming convention for the variables, I used ‘const’ and ‘let’ declarative syntax; for example, in figure 9c, the variable ‘teamData’ is in camelCase. The ‘EJS’ templating language, which enabled me to write less and efficient code. Also, I used chrome developer tools to do my debugging and running of some of my function codes to make sure I get my desired outputs. The most challenging aspect of this project was getting the database connection to work as well as pulling the data from Mongo dB and rendering the webpage with the data. With the support of one my colleagues who is also a senior developer and has had some experience with NoSQL databases, I pair programmed with him to achieve the challenge of pulling the data from Mongo db. I also applied some problem-solving skills I learnt at Makers bootcamp where we were taught to read the error message; if we understand it, apply the fix. If not, a console log to try to get some sort of data output, apply break points or do a google search for similar problems and the solutions that was applied.

It would be a great satisfaction for me when this page is integrated into the IAG portal. However, just like some other mini projects in the IAG, the integration of this one has been put on hold, as stated by the delivery manager.