Group 2: Ameer Karas, Joshua Vazzoler, Katrina David, Yefan (Tony) Li and Chris Purkiss
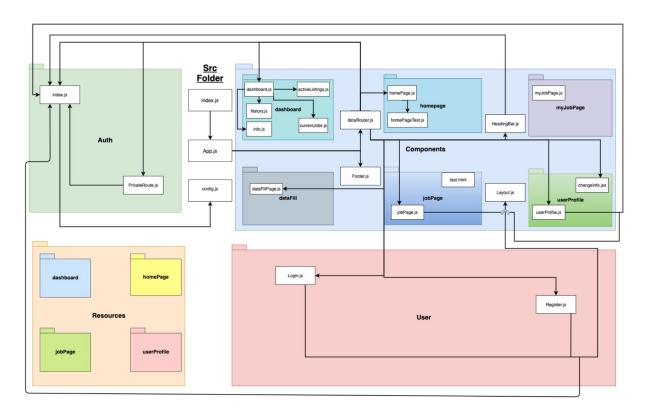
COMP4050 project design and overview

Package diagram:

Overview of the code (strengths and weaknesses): (300 words)

Weaknesses:
- No database
- Many folders with no useful files in them, bringing confusion
- Overall the code and files are very spread out making it confusing to understand what connects with what

Strengths:
- Each .js file is relatively short, keeping each file and its purpose clear.
- Each of the features has its own component/file, meaning the fixing or improving each feature is relatively easy
-

When doing an overview of the code base, we found several issues, first of which is the lack of a database at all in the code base. This lack of a database causes the profile and dashboard section of the website to be completely useless, with errors popping up and blocking access to those pages. One of the most important improvements we can make to this project is to implement a database, to fix this weakness of the current code base. As well as this, the code uses many, many different folders, many of them with no useful files in them. An example of this is that the Resources file has four files in it, and in each of those files is only a gitIgnore file, this makes the layout and interaction of the files more confusing

Group 2: Ameer Karas, Joshua Vazzoler, Katrina David, Yefan (Tony) Li and Chris Purkiss

than if the layout was more well planned. The final weakness found was that the overall spread of the files is very confusing, with the index.js file being imported by 7 other files, this makes its use confusing and lessens any distinct purpose it may have.

The code also presents some good strengths which can be used to improve upon and make our deliverable project better overall. First of these is the use of many, many components and classes, rather than just one large file. What this means is that the components and files can be given a clear purpose, for example; displaying the favours history or a user's profile information. A second strength that this code has is that many of the features have their own folder/file, for example, the dashboard has its own file, as does the user profile. What this means is that debugging and altering these features/components is much easier than it would otherwise be.

Feature implementation: (500 words)

Map and Search

Dashboard

User Profile

Request System

- o `New-Feature (Katrina with some help from Josh)`: Map and search - is a front-end feature group 2 would like to implement. For this feature, the intention is to have a map that works similarly to Google Maps and will be able to pinpoint locations of other members in the community who are in close proximity to you (the user) with favours that you (the user) are in need of. To execute this, we plan on referring to a Google API Client that we can integrate into the front-end

   **More detail, discuss implementation?

- o Front-end Feature (Tony): user profile → discuss feature with hidden contact details, for which user requests to see number and sends email out. Email authentication should be discussed since an email would be required.

- o `Additional Front-end Feature (Chris)`: Member dashboard → Displays the personal information about each of the users,

- o `Back-end Feature (Ameer with help from Josh) (Mern Stack - express, mongodb and node js)`: Creating the database and an API for the whole project.

Home Page → consist of map and search feature (New Feature, considered to be both front end and back end).

Dashboard → dashboard will include available jobs in the proximity of the user which he or she can do. Will also provide history,

Technology Stack

The technology stack we have chose to implement is the MERN stack. This includes:

- MongoDB for the database technology
- Express.js for the back-end technology
- React.js for the front-end technology
- Node.js for the package management technology.