1. **Data preparation and understanding**1.4 Explanation of code preparation & understanding

**Code Preparation**

***Step 1: Import the datasets.***

Import the libraries likely needed for later coding, including pathlib, pandas, and matplotlib. Next, use pathlib to define the path to the location of the data files. Then, utilize a self-defined function called 'create\_dataframe\_xlsx(xlsx\_file, sheet\_name)' to import four sheets from an Excel file named 'access-public-open-space-ward.xls' separately as dataframes. Additionally, make use of functions from the Pandas library, such as 'df.shape,' 'df.columns,' and 'df.info()' to create a function named 'print\_df\_information(Dataframe)' for analysing the basic information contained in those dataframes.

***Step 2: Merge the DataFrames***

Based on the information provided in the first sheet of the Excel file and the data source website (not included in the code), I learned that the ward boundaries of three wards in London (Hackney, Tower Hamlets, and Kensington and Chelsea) have changed since 2014 compared to 2013.

Regarding the dataframes, I noticed that there are four dataframes in pairs. Two of them contain the same type of data, with the only difference being that one contains all the wards in 2013 with corresponding data, while the other includes only the three changed wards in 2014. To maintain consistency, I decided to replace the data for the three wards in 2013 with the relatively latest ward data from 2014, leaving the rest unchanged as they remained the same in 2014.

To achieve this, I used the 'print()' function to identify the indices of the data that needed to be replaced. Then, I implemented a series of for-loops to remove that data. I also used a self-defined function called 'dataframe\_insert()' to insert the replacement data into the appropriate position, specifically before the Borough section. The key point here was to ensure that the column names of the two dataframes matched.

***Step 3: Missing Value check***

After merging the datasets, a null value check can be applied to each dataframe. While there are some NaN values, it's important to note that they do not significantly impact the overall data quality. Each dataframe's purpose and the reason for any NaN values are explained in the comments.

***Step 4: Save DataFrame as CSV file***

Use the self-defined function 'save\_CSVfile()' to save the two dataframes and provide them with appropriate names.

**Data Exploration**

**For Open Space Wards**

***Step 1: Draw a scatter graph of % of open space that has access***

First, a scatter graph of All Open Space vs. Open Space with access can be plotted, which the rate of x-axis and y-axis is % of open space that has access.

***Step 2: Draw a scatter graph of % Open Space with access***

Then, another scatter graph of Total area of ward vs. Open Space with access can be plotted, which the rate of x-axis and y-axis is % of open space that has access.

***Step 3: Draw a histogram graph of % open space***

hist() can be utilized to represent the distribution of data.

***Step 4: Draw a bar chart of Ward name vs. % open space for all the ward data***

dp.plot.bar can be used to plot a bar graph of Ward name vs. % open space for all the ward data, the labels of x-axis are overlapping, but still can represent the general trend.

**For Access to Open Space Wards**

***Step 5: Draw a bar chart of Ward name vs. Percentage of households with access to: Open Space of the first 50 data***

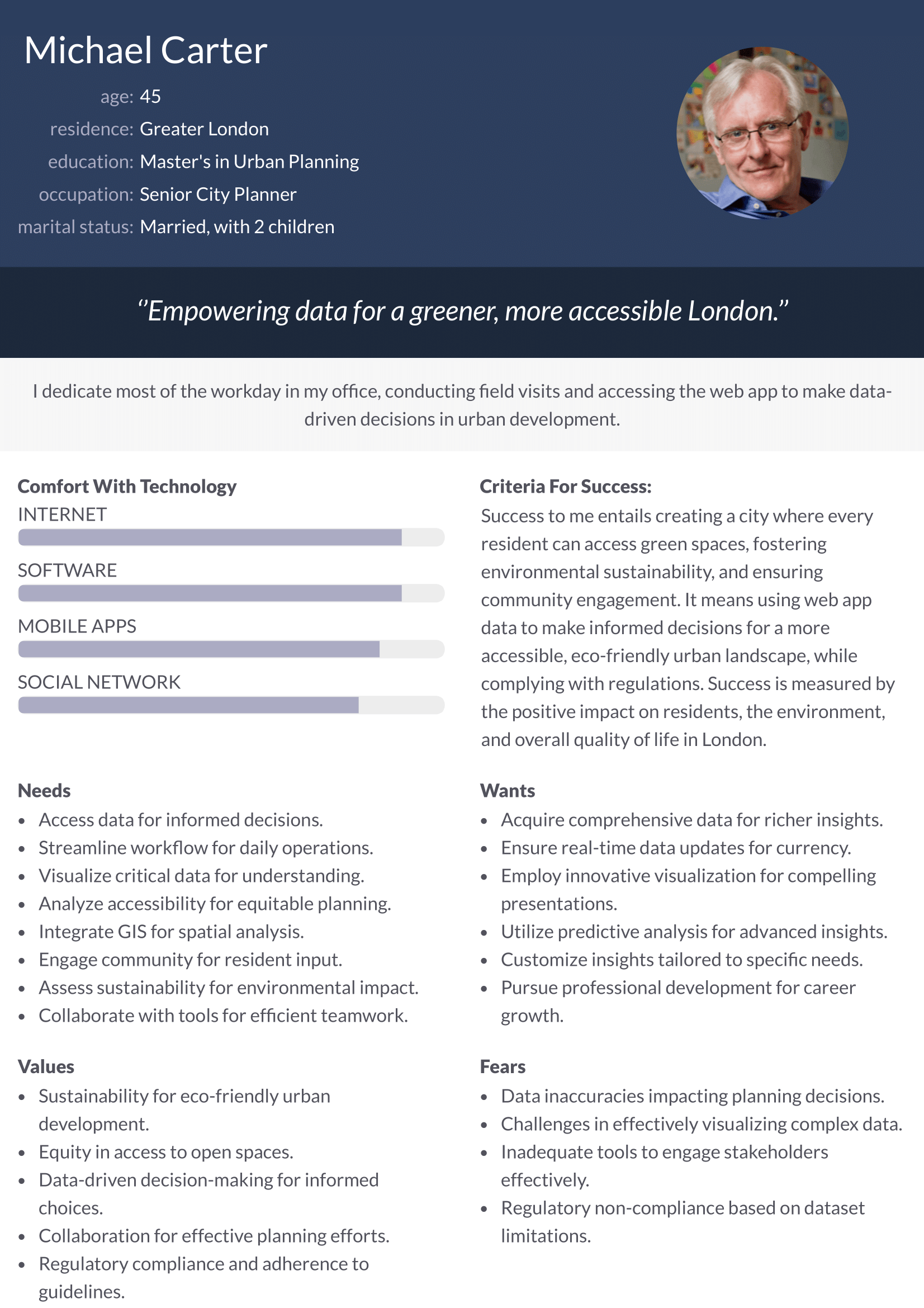
Another bar chart can be plotted to represent the data of Ward name vs. Percentage of households with access to: Open Space for only first 50 data, which makes the x-axis no overlapping, but only contains a quarter of the whole ward dataset.

# Product/project definition

## 2.2 Product overview

A REST API and a web application (data visualization app) will be developed using well-prepared datasets on Access to Public Open Space and Nature by Ward. The REST API will provide software developers with access to the dataset for their applications, while the web app will present the data visually and graphically to users.

## 2.3 Persona



# Tools & techniques

## 3.1 Source code control

<https://github.com/ucl-comp0035/comp0035-cwi-Fortis036.git>

**Weekly Record**

**Week 1**:

What I did in the last week:

I previewed the slides for this week's LECTURE last week and got a preliminary idea of what this semester's COURSEWORK will be from the COURSEWORK BREIFT and decided that I will be doing the COURSEWORK for both classes on an individual basis.

What I plan to do next week:

I'm going to prepare for writing the persona next week when I get my assigned database by browsing through the data to understand what it contains and the people who might need it (preparation for writing persona).

**Week 2**:

What I did in the last week:

Through the tutorial, I learnt about GitHub, created my own GitHub account, got to know the basics of GitHub, and was able to connect to the IDE in order to push, pull, and commit locally.

What I plan to do next week:

To understand the data further, to create a word document to write a brief of the data.

**Week 3**:

What I did in the last week:

Through the tutorial, I managed to create a virtual environment for the coursework. In additionally, I finished the initial draft of a persona.

What I plan to do next week:

I will use the persona draft to fill a persona template and finalize it. Try to start the coding part.

**Week 4**:

What I did in the last week:

Finished the product overview, and did some changes to the persona draft.

What I plan to do next week:

To finalize the coding part and fix unsolved issue of coding.

**Week 5:**

What I did in the last week:

Finished the coding part.

## Use of AI

AI is not used in this coursework.