COMP0004 Coursework

For this coursework, my website implements features that allow it to read csv files for the data of patients, and handles it without having the website crash or terminate if the file doesn't exist. This was done by returning a dataframe with one data, which has the ID which mentions that there is an error, and that the file path does not exist. The website also has the ability to view all of the patient ID's, and they have the ability to click on each of these patient ID's, where it will bring them to a page which has all of the details regarding that patient.

It also has the ability to search for patients by entering a keyword that the user wants to search for. It will return a list of the details of all the patients which contains the keyword. It also has the ability to get data of the oldest person, as well as filtering the patients by their gender. Both of these functions will also return a list of the details of all the patients which matches the requirements.

For the more challenging requirements, I have also added the ability for the website to add a new patient, edit existing patient data, as well as deleting them entirely. Everytime one of these functions is used, a new csv file is made with the file path "data/patients101.csv". I have also added the ability to save the current dataframe to JSON on the web page. When this is called, it creates a new file with the file path "patients101.json". Finally, it also has the ability to show a bar graph of the age of patients, from 0 to 130. If the patient's age is outside of the range, it will be placed at the nearest age group in the range.

For the design and programming process, I developed my code where each class is responsible to do one specific thing for the webpage. I believe that all of my classes are appropriate as they all serve one purpose, and they have easy to understand names.

Almost all the webpages use a specific servlet and a jsp page to display data. However some very simple pages use basic html as their jobs are only to send POST and GET requests to other servlets. This means that the elements of the classes are closely related and contribute to one purpose in the webpage. It also allows the code to have less dependency on one another, as making a change in a page will not alter the behavior of all the other pages.

The classes that contain the functions needed by the servlets are contained in the model class. This class contains functions which allows the webpage to give all of its functionalities. It also contains a dataframe, which itself contains an arraylist of columns. This dataframe is used to contain the details of all the patients. The functions needed to write and read CSV and write JSON files are separated to

reduce the size of the model class, but they are inherited by the model class. This allows for testing to be simpler, as it is easier to spot where the error occurs.

Finally, the ModelFactory class creates a new model class if it doesn't already exist. It inherits the ability to read CSV files from the DataLoader class. When it creates a new model class, it populates the empty dataframe of the model by reading the data from "data/patients100.csv". If a model class already exist, then it returns the existing model. This is called on every single servlet, as it ensures that all of it is working with the same model.

I believe that I have used a cohesive OO design practice. I made a good use of abstraction for the model class, as it inherits from the dataframe class, which itself also inherits from the column class. This allows for a higher level of abstraction, as each of these classes have a clear purpose for their overall purpose to store and perform functions on the patient's data.

Each one of the JSP and servlet classes are also cohesive with a very clear and focused purpose. This is because each JSP file are used to display a specific webpage, and all of the functions needed to be performed are called by the servlet, where the functions were inherited from the model class. The servlet sets the value of an attribute, and the JSP file retrieves it, and uses it to display processed data on the page. This means that there is minimal dependencies between the different web pages as the only classes interacting with each other is the JSP with its servlet.

I believe that the overall quality of my work is very good as I completed all of the requirements, including the challenge requirements. I have also included some error handling which prevents the webpage from crashing. My work also shown good OO design practices. This includes having each class handle one task, abstraction, encapsulation, as well as having little dependencies between the classes.