University of Dublin

TRINITY COLLEGE

Visualisation of the Data Set needed to predict survival time of Patients with Motor Neuron Disease.

Arthur Sasunts
B.A.(Mod.) Computer Science
Final Year Project April 2020
Supervisor: Ms. Gaye Stephens

School of Computer Science and Statistics
O'Reilly Institute, Trinity College, Dublin 2, Ireland

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Signed: _____ Date: _____

Abstract

Motor Neuron Disease(MND) is a disease that progressively attacks the human body's motor neurons or nerves. Currently there is no treatment for this disease. It is a rare disease with heterogeneity of cause and progression. This heterogeneity results in many revisions of diagnostic instruments, scoring systems and relevant data items. As a rare disease, research is part of the care process and therefore the care pathway is also under constant revision and includes clinical trials.

The ADAPT centre in Trinity College Dublin together with the FutureNeuro centre are currently creating a 'Motor neuron disease patient data platform' (MND-PDP) and are working with researchers across Europe who have recently created a model based on 8 data items for predicting a survival endpoint for patients with ALS (Amyotrophic lateral sclerosis), a type of MND.

As part of this platform, the ADAPT centre require a web application that displays a particular section of individual patients' MND register records to facilitate communication of their understanding of the prediction model. The users of the application are the register manager and the MND PDP project team. The application will not be used in a clinical setting.

This project presents an analysis of motor neuron disease, the prediction model and the relevant data set. Thereafter it details the design and implementation of a web application which displays excerpts from the individual patients' MND register records along with an overview of the prediction model. The application is populated with synthetic patient data which was provided by the project team and based on the format of the Irish Motor Neuron Disease population Register. The web application was validated by clinical, register and technical professionals.

Acknowledgements

Firstly I would like to say a huge thank you to my supervisor Ms. Gaye Stephens for her amazing support and guidance throughout this project.

I would also like to thank my friends and family for their support over the last 4 years, especially my brother Vahe. He has always been by my side and helped me any chance he got and for that I am very grateful.

Table of Contents

List of figures

List of tables

Abbreviations

MND	Motor neuron disease.
ALS	Amyotrophic lateral sclerosis.
ALSFRS-R	ALS Functional Rating Scale – Revised.
FVC	Forced vital capacity.
SNIP	Sniff nasal inspiratory pressure.
BiPAP	Bilevel Positive Airway Pressure.
COVID-19	Coronavirus.
PDP	Patient data platform.
PM	Prediction Model.
API	Application programming interface.
DOM	Document Object Model.
JSX	JavaScript XML.
ES6	ECMAScript.
HTML	Hypertext Markup Language.
UI	User interface.
UX	User experience.
UC	Use case.
SPA	Single page application.

Terminology

React	A JavaScript library used for creating user interfaces.
Firebase	A web services platform used for authentication, data storage and hosting.
Semantic UI	A front end development framework which uses Hypertext Markup Language (HTML).
Node.js	An open source environment that allows users to execute and run JavaScript code outside a web browser.
Adobe XD	An application used to design web and mobile applications.
Rare disease	A disease that affects a very small portion of the population.
Motor Neuron Disease	A rare disease that progressively attacks the human body's motor neurons or nerves.
ALSFRS-R	A rating scale designed to quantify the progression of ALS as rated by the patient.
C9orf72 repeat expansion	A hexanucleotide repeat expansions in the chromosome 9 open reading frame 72 gene.
Gastrostomy	An opening into the stomach from the abdominal wall, made surgically for the introduction of food.
Tracheostomy	An incision in the windpipe made to relieve an obstruction to breathing.
Palliative care	Care for the terminally ill and their families, especially that provided by an organised health service.
Population based register	A population based register contains records for people diagnosed with a specific type of disease who reside within a defined geographic region.
Referral based register	A referral based register contains records for people diagnosed with a specific type of disease who have been referred to a hospital or facility. Referral based registers only holds a subset of the full population with the disease.
El escorial	A criteria used for the diagnosis of ALS (definite vs probable or possible ALS).
Dyspnoea	Difficult or laboured breathing.
Orthopnoea	Shortness of breath that occurs when lying flat.

1. Introduction

This project involves the presentation of the data set needed to predict survival time of patients with Motor Neuron Disease by producing a visual dashboard in a web application. The project team comprised ADAPT and FutureNeuro researchers involved in the Motor Neuron Disease Patient Data Platform project. The web application is created using the React framework along with Firebase web services. The data used is synthetic and was provided by the project team. It mimics the data that is stored it in Irish ALS/MND register.

This project provides an analysis of the process taken, the technologies used and the research carried out. It will then discuss the design by exploring the planning that was carried out, a prototype design of the patient dashboard, the functional and non-functional requirements, the use case, the architectural design and the final user interface design. Thereafter the implementation of the application will be examined by studying each page and component used. To close, the evaluation and conclusion will be discussed.

1.1 Motivation

Interest in the healthcare domain and the opportunity to develop a web application were key motivators for this project. Participation in an internet applications module during my BA in Computer Science studies sparked an interest. The basic building blocks on the creation of small scale web applications using the Vue.js framework and how to pull data from Application Programming Interfaces (APIs) and databases to display on a user interface were covered during the module. This project provided an opportunity to improve my web development skills and overall understanding of web applications.

The healthcare domain was also of interest to me as it is rewarding creating a project that could potentially be of benefit for the user as well as the patient. This project provided a context in which to develop both technical and healthcare knowledge. Having limited knowledge about MND, a whole new territory to research and understand was opened up. An understanding of patient information collection and transfer was gained. Information about patients in the national MND clinic is currently collected using paper. This information is brought to Trinity College Dublin to be manually entered into the Irish ALS/MND register. The only way the register staff can view the information of individual patients is by finding their files and

potentially flicking through many pages or by using excel style pages which provide a view on the register. With the web application, developed as part of this project, this process is simplified by searching, subject to data protection and security considerations, a patients name or identification to reveal the relative information that is displayed on a dashboard. The idea for the web application provided motivation to demonstrate skills as well as helping the MND register manager and providing understanding of the prediction model to current and future MND-PDP researchers.

Having close friends whose relatives are diagnosed with MND and friends who are nurses that may have to support patients with MND was also a big motivator for this project. It is rewarding knowing that my project can be used in the real world and may contribute to improvements in healthcare information systems design as part of the ADAPT/FutureNeuro MND PDP project.

1.2 Objectives

The objectives of this project are listed here and expanded on below:

- Get a good understanding about MND and the prediction model.
- Create a database and store the sample data.
- Learn the React framework.
- Create a prototype of the design for the patient dashboard.
- Display the data on the application obtained from the database.
- Connect the application with the Irish ALS/MND Register.

By carrying out in depth research about MND from online resources and a selection of complex papers[1][2][3][4][10][11][14][16][17] provided by the project team, my understanding about the disease and the prediction model increased in an incremental fashion which helped me with the development of the web application.

Researching and comparing the wide variety of web service platforms available allowed me to choose the appropriate platform that consisted of all the features that were required for this project, in this case, Google's Firebase platform was the platform of choice. It provides a real-time database where the sample data is stored.

With my current knowledge of Vue.js, making web applications was already familiar to me. However, my decision to use React was mainly because it is a more widely used platform that provides an easier framework to implement and manage,

accompanied with many online resources. In my opinion, React is easier to use because the user can divide the code into separate components and therefore can manage their project more efficiently.

Designing a prototype using Adobe XD helped me create a vision on which the web application could be based.

One aspect of this project was to connect the application with the Irish ALS/MND register. However, with the current situation regarding COVID-19, this opportunity was not available. Some testing was performed to simulate the connection to the register.

The overall objectives for this project was to communicate understanding of the data set needed to predict survival time of patients with MND and to provide the MND register manager with a view of the register data. Design wise, the objective was to create a simple but useful web application that is easy for the users to navigate through, as well as being aesthetically pleasing.

1.3 Achievements

The achievements of this project include:

- A fully functional web application.
- Validation of the content displayed.
- The usability of the application.

The main achievement was the completion of the web application. It was created by following the requirements of the PDP manager who has also validated the application. Since the application was completed, it has already been used successfully in a number of project meetings. This project will also feed into the Core Information Modelling task of the MND PDP project, especially into a masters computer science student's project.

1.4 Report structure

2. Background discusses the process implemented for the creation of the project, explains the technologies used and analyses the research carried out.

3. Design describes the thought process that went into planning the design of the web application, displays an early prototype, discusses the project requirements, explains the UI architecture and presents the final design.

4. Implementation explores how the web application was created by discussing the page routing, Firebase's integration and the different pages and components.

5. Evaluation discusses the final results, testing and considerations made and the challenges encountered throughout the project.

6. Conclusion & Future work explains the future work and outcomes of the project.

2. Background

This chapter explains the process that was carried out for the creation of this project. It portrays the different technologies used that binds this project together, focusing on the React framework, the Firebase web services platform and the styling of the application. The reasons for choosing these technologies will be presented. Later, the chapter will present the research carried out in the healthcare domain targeting rare diseases, MND, ALS, the Irish ALS/MND register, the prediction model and the data set upon which the prediction model is based. This data set is comprised of a mixture of simple, complex, static and dynamic data and it is this data that is visualised using the web application.

2.1 Process

Figure 2.1: Process overview of the project. [1]

Throughout the whole project a specific process was followed which can be imagined from the diagram above (`Figure 2.1`: Process overview of the project.). Firstly research using journal articles and from project meetings was carried out to gain a good understanding about MND, the prediction model and the relevant data set. Then the design process began where the overview of the application was discussed in detail with my supervisor, a prototype of the patient dashboard was created, the functional and non-functional requirements of the application considered, a use case diagram was drawn up and the architectural design was researched. After the design was created, development of the web application began. When the first version of the application was complete, it was presented to the project team during a meeting where valuable feedback was obtained and used in the next iteration of the application. This process was repeated multiple times over meetings until the project team were happy with the content displayed on the application. During the final presentation, the PDP manager confirmed that everything was correct and validated the application. The final process was

[1] https://www.flaticon.com/analysis, https://www.flaticon.com/sketchbook, https://www.flaticon.com/feedback, https://www.flaticon.com/coding, https://www.flaticon.com/checklist, https://www.flaticon.com/right-arrow, https://www.flaticon.com/recycle

evaluation. During this process the results of the project were examined, testing and considerations were carried out and the challenges encountered were discussed.

2.2 Technologies Used

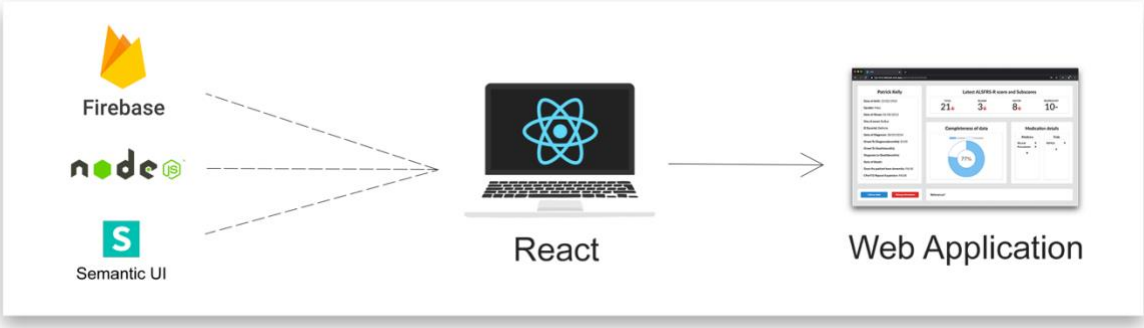

Figure 2.2: Diagram of technologies used. [2]

2.2.1 React

The React framework is a JavaScript library used for creating user interfaces. It was created by Facebook and is a widely used framework. React is very versatile as it allows users to create web applications as well as native mobile applications with the React-native platform which works on both android and iPhone devices. Reacts main purpose is to render data to the DOM (Document Object Model) however with the help of many additional libraries, the user can create complex and unique applications. For example, the React Router library allows you to connect the different pages of the application together and navigate to them easily[5]. Managing and maintaining your application is made easier with React. It allows the user to split up the different pages/sections of the application into separate components which provides a cleaner work space without having all the code jumbled into one file. Components can be created as functional or class-based components. Functional components are declared with a function that then returns some JSX (JavaScript XML) [5]. `Figure 2.3`: Example of a functional component in React. below is an example of a functional component.

```
const Greeting = (props) => <div>Hello, {props.name}!</div>
```

Figure 2.3: Example of a functional component in React. [3]

[2] https://www.pikpng.com/ , https://www.freecodecamp.org , https://reactjs.org/

[3] https://en.wikipedia.org/wiki/React_(web_framework) - Functional_components

Class-based components are declared using ES6 (ECMAScript) classes[5]. They are also known as stateful components as they can hold values in their state which can be used anywhere throughout the component and can also be sent to different components as "props". The class-based component in `Figure 2.4`: Example of a class-based component in React. has a state that holds the variable 'color'. This variable is sent as a prop to the component '<ChildComponent>' which will allow that component to receive and use the variable. The examples in `Figure 2.3`: Example of a functional component in React. and `Figure 2.4`: Example of a class-based component in React. are similar to the code available in the appendix.

```
class ParentComponent extends React.Component {
  state = { color: 'green' };
  render() {
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Figure 2.4: Example of a class-based component in React. [4]

The reason for choosing the React framework is because it is the most used JavaScript framework.[6] From experience, there are plenty of resources online to help with the project development, managing an application is clearer by using components and the addition of libraries gives the user freedom to create an application with their specifications.

2.2.2 Firebase

Firebase is Google's web services platform. It provides many useful features which were essential for my project. Some of the features that it administers are authentication, database storage, web-hosting, cloud functions, among many others. The aim of this application is for it to be used with real patient data from the Irish ALS/MND Register. By not having constant access to this database, for confidential reasons, a sample file with synthetic patient information was provided for me to work with. Firebase provided the framework to use this data within the application. With

[4] https://en.wikipedia.org/wiki/React_(web_framework) - Class-based_components

the helpful documentation on the Firebase website, implementing it's features with React was made possible. Importing the sample file into the real-time database was achievable as well as pulling data from the database straight to the application. Firebase's Authentication feature added a level of security which only grants access to authenticated users. With user management, users can be added by email and password, or removed from the system entirely. In addition, publishing the application was possible using Firebase's hosting feature. This was advantageous as the need to use an external hosting service was not necessary. Firebase also allows users to create their own custom domain and all of this can be done with no cost at all. One of the main reasons for choosing Firebase over, for example, Amazon Web Services was because it is free to use and since the application is not large-scale, the usage limit will not be exceeded. Firebase also provides a much easier platform to use (based on previous experience using Amazon Web Services). Setting up a project is straightforward, the dashboard is clear and easy to navigate and the documentation is sufficient.

2.2.3 Semantic UI

Semantic UI is a front end development framework which uses Hypertext Markup Language (HTML) and helps creates aesthetic designs. It provides over 50 user interface (UI) components and over 3000 theming variables[7]. Semantic UI is also officially integrated with the React framework, called Semantic UI React and makes the usage of the library easier. Features can be directly inputted as react components and therefore the resulting code is immensely clean. With a great variety of customisation options, users can design their application to their pleasing. Using the Semantic UI React library throughout the web application gave it the aesthetic look it currently obtains. It's features and components such as Containers, Grids, Buttons, Modals, Tables, to name a few, helped style and shape the application in the exact way that was planned. The reason for using Semantic UI instead of Bootstrap, another very popular front end development framework was mainly because of the React integration. It allowed for simple integration with the project code and provided a flat, materialistic UI design which was preferred.

2.2.4 Node.js

Node.js is an open source environment that allows users to execute and run JavaScript code outside a web browser, i.e. your computer. With this environment, access to local files are available, access to databases can be achieved, communication to and from http requests is possible, among many other useful

features. Node.js also allows users to install and use packages in their applications. It allowed me to combine all the required modules together, e.g. Firebase, and got the web application fully functioning. The reason for using Node.js is because of previous experience with other projects. These experiences were generally good and provided simple project package management.

2.2.5 Adobe XD

Adobe XD is an application used to design web and mobile applications. It offers many tools and provides the user a canvas to create a working prototype of their desired application. This tool helped me create the patient dashboard design imagined in my head which was then implemented using React and Semantic UI. The reason for using Adobe XD is because it was created specifically to design web and mobile UIs and it is also a free to use application.

2.3 Research

This section will discuss the research carried out before starting development on the web application. As mentioned before the topics reviewed are rare diseases, MND, ALS, the Irish ALS/MND register, the prediction model and the data set upon which the prediction model is based. This research was relevant for this project as it provided a good understand about MND, the prediction model and the corresponding data set used, which in turn allowed me to create the web application and communicate understanding of the data set needed to predict survival time of patients with MND. Carrying out interdisciplinary research was not easy by any means. While comfortable in my field of study with creating the web application, it was important to completely understand the content that was being produced. A whole new world of terminology was introduced which took some time to get used to. The provided papers by the project team were difficult to grasp but were essential in the development of the application. This was achieved by thoroughly reading the papers in depth and through the meetings with the project team. The research section of the project took up a considerable amount of time but was necessary. This interdisciplinary work has increased my ability of becoming a 'T-shaped' researcher. T-shaped researchers are described by being "able to cultivate both their own discipline, and to look beyond it". [8]

2.3.1 Rare Disease

A rare disease is defined as a disease that affects a very small portion of the population and in Europe, when it affects 1 in 2000 people. Characteristics of rare diseases are:

- Often chronic, progressive, degenerative, and life-threatening.
- They cause high level of pain and suffering.
- There is no existing effective cure.
- Most rare diseases are genetic and may be present in a person's entire life even if symptoms are not shown straight away.[9]

Rare diseases affect around 6% of the population. There are at least 300,000 individuals diagnosed with a rare disease in Ireland. The diagnosis of rare diseases are often delayed for many years due to their rarity and lack of expertise.[10] Scoring systems for rare diseases are often created to quantify the progression for these diseases such as the ALS functional rating scale.

2.3.2 Motor Neuron Disease (MND)

Motor Neuron Disease falls under the rare disease category. There are many types of MNDs and they affect the body's motor neurons. The motor neurons are cells that are located in the central nervous system and control voluntary muscle activity such as walking, speaking, breathing and swallowing. It is a progressive disease and as time goes on, patients with MND start to lose their ability of these essential muscle activities. The types of MNDs include:

- Amyotrophic Lateral Sclerosis (ALS).
- Progressive bulbar palsy.
- Primary lateral sclerosis.
- Progressive muscular atrophy.
- Kennedy's disease. [11]

Usually there are no concrete tests that tell you if a patient is diagnosed with any MNDs. In most cases, diagnosis is given with a physical examination followed by a neurological exam carried out by specialists in the MND clinics. These neurological examinations rule out other diseases which the patient may have or measure muscle activity. Since MNDs are progressive and have no cure, the only treatment there is to help the patients is to slow down the degeneration. Such treatments include drugs/medicine such as Riluzole, exercise, nutrition management or non-invasive ventilation/assisted ventilation[11] to name a few. However once the disease reaches a certain point, the only option is palliative care. Patients with MND often attend clinics where they get assessed by specialists who also record their information which is later entered into the Irish ALS/MND register. MNDs usually affect people above the age of 50 but can also affect younger people. They can be passed down genetically but in most cases they occur sporadically and symptoms can be shown at any age[12].

2.3.3 Amyotrophic Lateral Sclerosis (ALS)

Note: the following information has been extracted from the 'Motor Neuron Disease Research Patient Data Platform Background Document for Sprint Zero with ADAPT D-Lab'.[11]

Amyotrophic Lateral Sclerosis (ALS) is a type of MND and is a progressive, debilitating, neurodegenerative disease with a life expectancy of three to five years. 55% of patients affected by ALS are males and 45% are females. The average age of diagnosis for males is roughly 61.8 years and for females, around 64.0 years. The site of onset can vary with 60% occurring in the spinal region, 29% in the bulbar area, 5%

in the thoracic/respiratory system, 5% by cognitive/behaviour and 1% mixed in the spinal and bulbar regions. Patients with ALS often deal with physical degeneration such as muscle weakness and stiffness, change in speed and change in swallowing. 40 to 60% of patients with ALS also have mild to moderate cognitive and/or behavioural deficits. 80% of ALS patients have received the medication Riluzole, a medication used to treat ALS which can slow down the rate of degeneration and can increase a patient's survival by up to three months.[13] The ALS Functional Rating Scale – Revised (ALSFRS-R) is used to quantify the progression of the ALS which is discussed in detail in 2.3.6 ALS Functional Rating Scale – Revised (ALSFRS-R).

2.3.4 The Irish ALS/MND Register

The Irish ALS/MND register contains information on almost 2,553 patients, recorded in November 2018. It was set up in 1993 and is located in the National Centre for Neuroscience in Trinity College Dublin. The registers main purpose is to generate new knowledge around MND, identify patients for clinical trials and help towards finding a cure or better treatment for MND. Currently the data is received by a researcher using pen and paper in the MND clinics. It is taken from the patient's chart and consent is always agreed by the patient before any information is used. This data is then transferred to Trinity College where it is manually inputted into the register. It is a population based register. Some of the register data is used in the prediction model.[11]

2.3.5 The Prediction Model (PM)

Currently there are no models that accurately predict the course of ALS and it's outcomes. 14 specialised ALS centres across Europe collaborated with the aim of developing and validating a model to predict survival without tracheostomy and non-invasive ventilation for more than 23 hours per day for individual patients with ALS. Their work was published by The Lancet Neurology on March 26th 2018. These ALS centres are located in Utrecht, Netherlands; Dublin, Ireland; Torino, Italy; London, Sheffield, Oxford, UK; Lisbon, Portugal; Hannover, Jena, Germany; St Gallen, Switzerland; Tours, Limoges, France; Leuven, Belgium. The prediction model was created using data that was collected from the largest ALS dataset in Europe between Jan 1st, 1992 and Sept 22nd, 2016 which included data from 11,475 patients. This data was collected from 14 ALS centres in the cities mentioned above. The centres in Ireland, Italy and The Netherlands use a population based register while the rest of the centres use a referral based register. The data consisted of clinical, cognitive and genetic information about the patients, totalling a number of

16 characteristics. However the model uses eight candidate variables for its prediction. The eight candidate variables were selected by using a backward elimination procedure with bootstrapping.[14] These variables are listed and explained in `Table 2.1`: Table of the eight candidate data items used in the prediction model. below.

Data item	Input type	Data Type	Dynamic/Static	Description
Bulbar versus non bulbar onset	String	Clinical	Static	Whether site of the disease is bulbar or spinal.
Age at onset	Integer	Clinical	Static	Age of the patient when the disease began.
Definite versus probable or possible ALS	String	Clinical	Dynamic	The El Escorial criteria used for the diagnosis of ALS.
Diagnostic delay	Integer	Clinical	Static	Time from onset of weakness or bulbar symptoms to diagnosis.
Forced vital capacity (FVC)	Integer	Clinical	Dynamic	The amount of air that can be forcibly exhaled from your lungs after taking the deepest breath possible, as measured by spirometry.[15]
Progression rate	Integer	Clinical	Dynamic	Defined by the slope on the ALSFRS-R.
Frontotemporal dementia	Boolean	Cognitive	Dynamic	Frontotemporal dementia according to the Neary or Rascovsky criteria.
Presence of C9orf72 repeat expansion	Boolean	Genetic	Static	A hexanucleotide repeat expansions in the chromosome 9 open reading frame 72 gene.

Table 2.1: Table of the eight candidate data items used in the prediction model.

Note: FVC is used to measure respiratory function and was used by all specialised ALS centres except for the centre in Dublin who used sniff nasal inspiratory pressure (SNIP), which is known to be correlated with FVC.[14]

Based on the 8 data items described in the table above, the prediction model returns the median predicted survival time for a patient in months. The prediction model is implemented in R.

There are five groups associated with the median survival times:

- Very short: 17.7 months
- Short: 25.3 months
- Intermediate: 23.3 months
- Long: 43.7 months
- Very long: 91 months

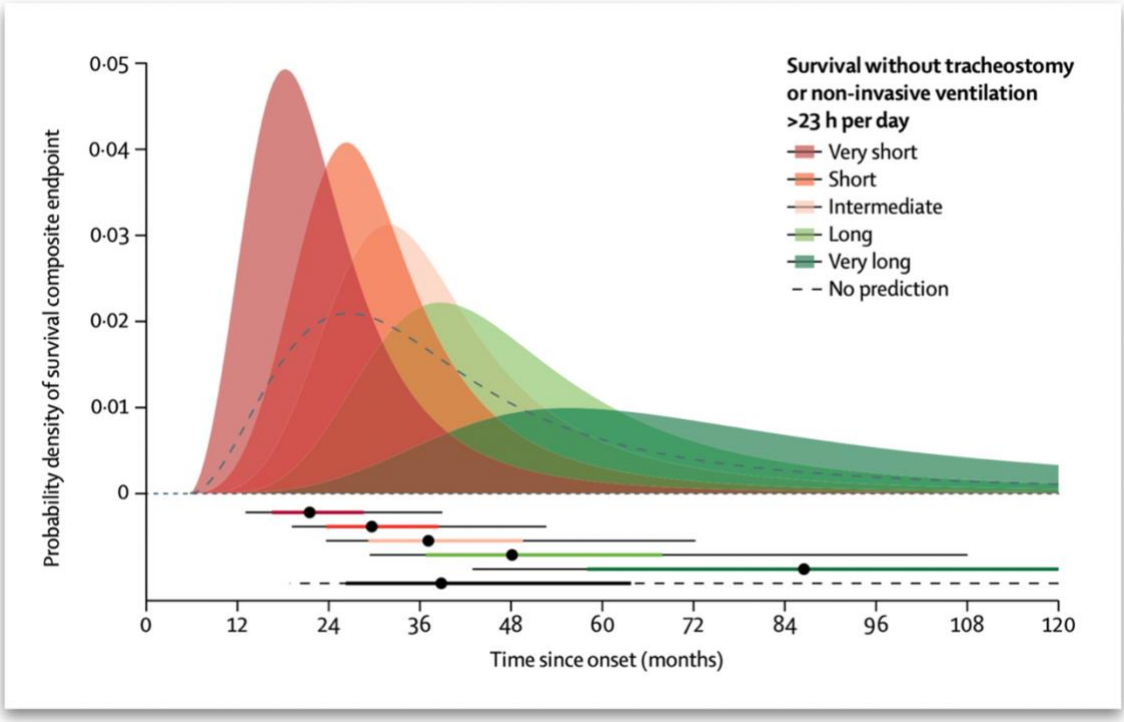

Figure 2.5: Graph of probability densities corresponding to times of survival endpoint.[5]

The graph above (`Figure 2.5`: Graph of probability densities corresponding to times of survival endpoint.) illustrates the probability density of the survival endpoint for the five groups. The probability density is not evenly distributed over time. The largest number of patients are predicted to pass away at the point where the curve is the highest. For example, if a patient falls under the very short category, their survival is predicted to end at around 17 months, however there is a small chance that the patient will survive for longer, which is reflected by the downward slope at the right hand side of the curve. These prediction curves provide guidance for discussing prognosis with patients.

[5] https://www.thelancet.com/journals/laneur/article/PIIS1474-4422(18)30089-9

The method used to create the PM were as follows; the 16 patient characteristics were assessed as potential predictors of a composite survival outcome. Backward elimination with bootstrapping was used to select these predictors. Data from the 14 ALS centres were combined using the internal–external cross-validation framework.[14]

2.3.6 ALS Functional Rating Scale – Revised (ALSFRS-R)

The scoring systems for rare diseases can be frequently revised to provide a more accurate result. The ALS functional rating scale presented on the web application is the current version in use, the ALSFRS-R, where the 'R' at the end stands for revised. It should be noted that there is work progressing to divide the scale into three subscales (bulbar, motor and respiratory) and to use the values from the subscales to further inform clinicians of a patient's functional rating. The subscales are also presented on the web application. The ALSFRS-R is a rating scale designed to quantify the progression of ALS as rated by the patient. It consists of 12 questions about the patient's daily life that are graded by a score of 0 to 4, with 4 being 'normal' and 0 indication loss of a function, for example, speech. Normally the patient would answer the questions, however, if they are unable to communicate, the patient's caregiver will answer for them. The 12 functions that are rated for each patient are presented in `Table 2.2`: Table of the ALSFRS-R bulbar functions., `Table 2.3`: Table of the ALSFRS-R motor functions. and `Table 2.4`: Table of the ALSFRS-R respiratory functions. below.[16]

Function	Subdomain	Scoring system
1. Speech	Bulbar	**4:** Normal speech process. **3:** Detectable speech disturbance. **2:** Intelligible with repeating. **1:** Speech combined with non-vocal communication. **0:** Loss of useful speech.
2. Salivation	Bulbar	**4:** Normal. **3:** Slight but definite excess of saliva in mouth; may have night time drooling. **2:** Moderately excessive saliva; may have minimal drooling. **1:** Marked excess of saliva with some drooling. **0:** Marked drooling.
3. Swallowing	Bulbar	**4:** Normal eating habits. **3:** Early eating problems – occasional choking. **2:** Dietary consistency changes. **1:** Needs supplemental tube feeding. **0:** NPO (nothing by mouth).

Table 2.2: Table of the ALSFRS-R bulbar functions.

Function	Subdomain	Scoring system
4. Handwriting	Motor	**4:** Normal. **3:** Slow or sloppy: all words are legible. **2:** Not all words are legible. **1:** No words are legible, but can still grip pen. **0:** Unable to grip pen.
5a. Cutting food and handling utensils without gastrostomy	Motor	**4:** Normal. **3:** Somewhat slow and clumsy, but no help needed. **2:** Can cut most foods, although slow and clumsy; some help needed. **1:** Food must be cut by someone, but can still feed slowly. **0:** Needs to be fed.
5b. Cutting food and handling utensils with gastrostomy	Motor	**4:** Normal. **3:** Clumsy, but able to perform all manipulations independently. **2:** Some help needed with closures and fasteners. **1:** Provides minimal assistance to caregiver. **0:** Unable to perform any aspect of task.
6. Dressing and hygiene	Motor	**4:** Normal function. **3:** Independent; Can complete self-care with effort or decreased efficiency. **2:** Intermittent assistance or substitute methods. **1:** Needs attendant for self-care. **0:** Total dependence.
7. Turning in bed and adjusting bed clothes	Motor	**4:** Normal function. **3:** Somewhat slow and clumsy, but no help needed. **2:** Can turn alone, or adjust sheets, but with great difficulty. **1:** Can initiate, but not turn or adjust sheets alone. **0:** Helpless.
8. Walking	Motor	**4:** Normal. **3:** Early ambulation difficulties. **2:** Walks with assistance. **1:** Non-ambulatory functional movement only. **0:** No purposeful leg movement.
9. Climbing stairs	Motor	**4:** Normal. **3:** Slow. **2:** Mild unsteadiness or fatigue. **1:** Needs assistance. **0:** Cannot do.

Table 2.3: Table of the ALSFRS-R motor functions.

Function	Subdomain	Scoring system
10. Dyspnoea	Respiratory	**4:** None. **3:** Occurs when walking. **2:** Occurs with one or more of the following: eating, bathing, dressing. **1:** Occurs at rest: difficulty breathing when either sitting or lying. **0:** Significant difficulty: considering using mechanical respiratory support.
11. Orthopnoea	Respiratory	**4:** None. **3:** Some difficulty sleeping at night due to shortness of breath, does not routinely use more than two pillows. **2:** Needs extra pillows in order to sleep (more than two). **1:** Can only sleep sitting up. **0:** Unable to sleep without mechanical assistance.
12. Respiratory insufficiency	Respiratory	**4:** None. **3:** Intermittent use of BiPAP. **2:** Continuous use of BiPAP during the night. **1:** Continuous use of BiPAP during day. **0:** Invasive mechanical ventilation by intubation or tracheostomy.

Table 2.4: Table of the ALSFRS-R respiratory functions.

As mentioned above, these questions can be categorized into three main subgroups and are used to calculate the bulbar, motor and respiratory scores. The bulbar subgroup contains the total score of questions 1-3, the motor subgroup contains the total score of questions 4-9 and the respiratory subgroup contains the total score of questions 10-12. A total ALSFRS-R score of all the questions is also calculated. These subscores are then used to calculate the ALSFRS-R slopes. The total ALSFRS-R slope is calculated with the following formula, slope=(48 - ALSFRS-R Total / time from onset). The bulbar slope is calculated by, slope=(12 - Bulbar-score / time from onset). The motor slope is calculated by, slope=(24 - Motor-score / time from onset) and the respiratory slope is calculated by, slope=(12 - Respiratory-score / time from onset).[16] "The ALSFRS-R slope is well established as a prognostic indicator in ALS". [17] Example results of these slopes can be seen in `Figure 2.6`: Graph of longitudinal total ALSFRS-R and ALSFRS-R subscores. below.

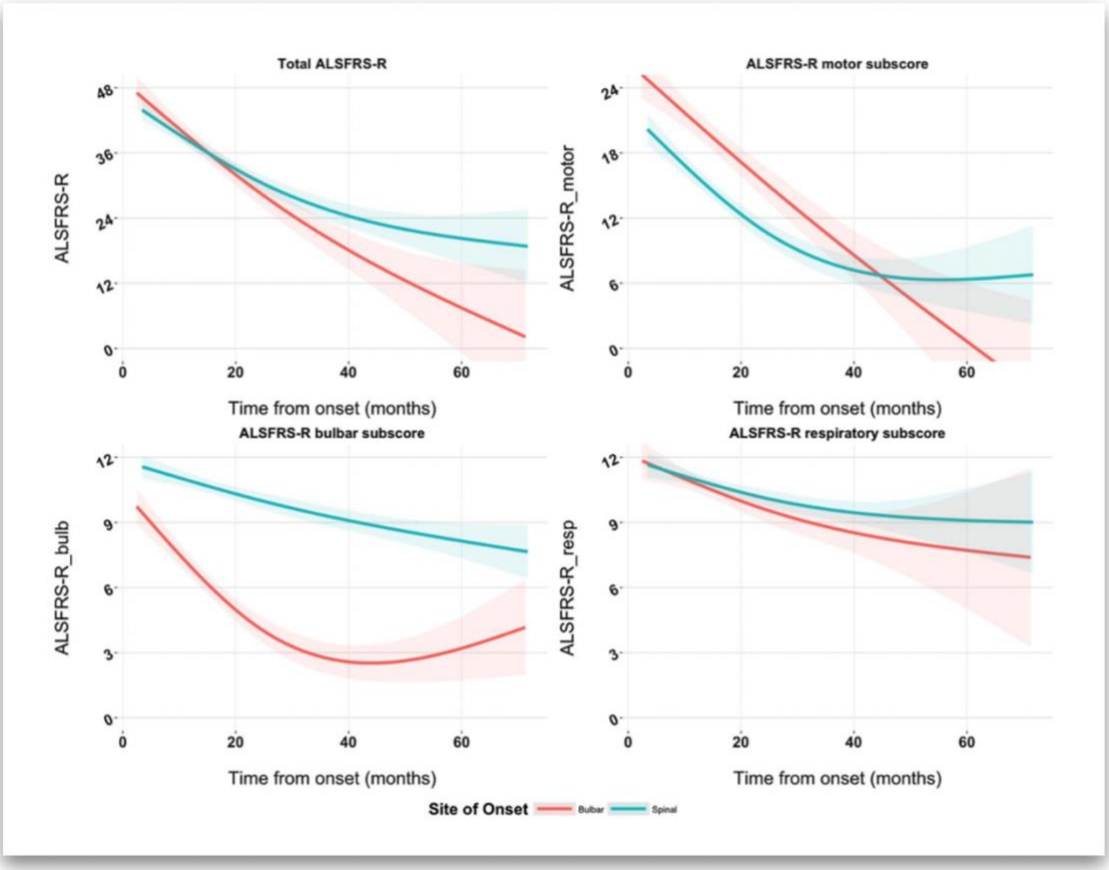

Figure 2.6: Graph of longitudinal total ALSFRS-R and ALSFRS-R subscores.[6]

The ALSFRS and its revised version are similar with a two main differences. The unrevised version consists of ten questions that are identical to the first ten questions in the revised version, which contains two extra questions about respiration. This leaves the unrevised version with a total score of 40 compared to the revised version with 48. All of the specialised ALS centres mentioned in `2.3.5 The` Prediction Model (PM) used the ALSFRS-R score except the centre in Oxford which used the ALSFRS score. In order to compare the scores, they transformed the ALSFRS score to the ALSFRS-R score by multiplying by 1.2.[14]

6 https://jnnp.bmj.com/content/88/5/381

3. Design

This chapter describes the thought process that went into planning the design of the web application, displays an early prototype, discusses the project requirements, explains the UI architecture and presents the final design. MND is a rare disease with heterogeneity of cause and progression. This heterogeneity results in many revisions of diagnostic instruments, scoring systems and relevant data items. Based on the reading of the provided papers and the research carried out, it is clear to see that the data and information will change over time. The Irish ALS/MND register will grow larger, the ALSFRS-R may be adjusted and It is expected that the dataset for the prediction model will evolve and change as understanding of the disease progresses. Therefore, the web application presented in this project is designed with these requirements in mind. It has the necessary framework accompanied by the tools and features needed to accommodate these potential changes.

3.1 Planning

Before delving into the creation of the web application, meetings with my supervisor and with the ADAPT project team concerning various aspects about the web application were carried out. The aspects discussed included:

- UI design
- UX design
- Data requirements
- Data protection
- Information architecture

These discussions were very informative and allowed me brainstorm ideas on how to implement the web application. As development of the project began, more ideas and feedback were received from the project team by reviewing the early stages of the design. This allowed me to make adjustments and add more content to the application as requested.

3.2 Prototype

Designing a prototype is essential in any stage of a project. Adobe XD provided a platform for me to bring my ideas to life. A simplistic, flat, materialistic style was the main objective for the overall design of the application. Most successful applications in today's world such as Twitter, Instagram and Facebook follow this concept which drew some inspiration.

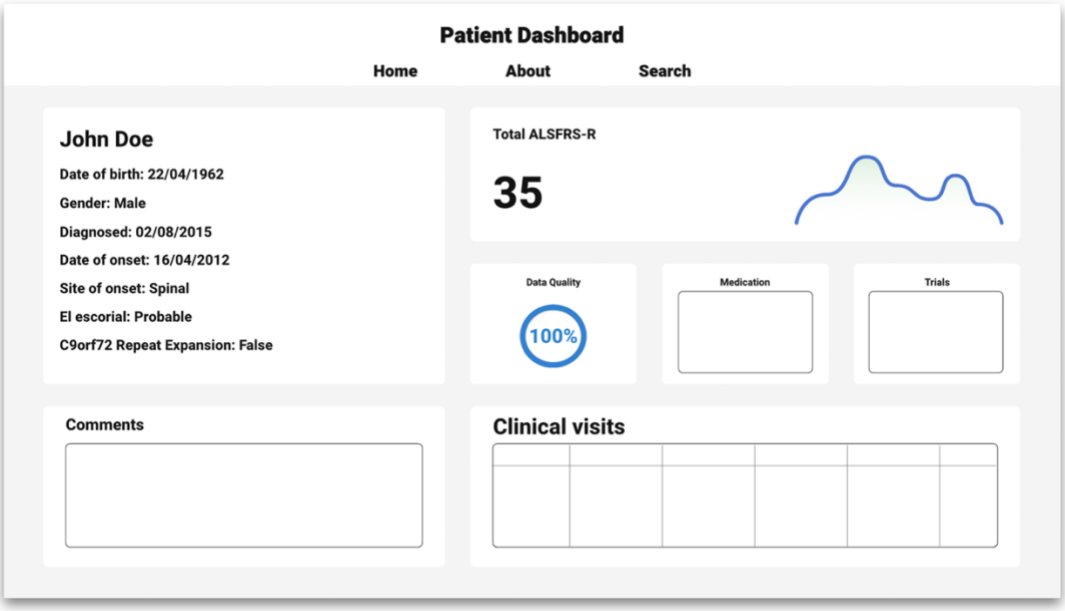

Figure 3.1: Prototype design of the patient dashboard made using Adobe XD.

The flat, materialistic design can be seen in `Figure 3.1`: Prototype design of the patient dashboard made using Adobe XD. above. It includes a simple navigation bar at the top of the page with the title in the centre and the page names underneath. The colours of the dashboard are consistent and minimal. Each section (white block) displays relevant information about each patient. Inspiration was also taken from the 'ALS Population overview' dashboard (`Figure 3.2`: Prototype design of the 'ALS Population Overview'.) presented in the 'Motor Neuron Disease Research Patient Data Platform Background Document for Sprint Zero with ADAPT D-Lab'.[11]

Figure 3.2: Prototype design of the 'ALS Population Overview'.

3.4 Requirements

This application was designed with the main purpose of providing the user a simple overview dashboard of an individual patients' MND register record. As well as that, to display information about the prediction model and terminology used.

3.4.1 Functional requirements

The functional requirements for this application as agreed with the project team are as follows:

- Login/logout with authenticated user emails.
- Present a welcome page explaining the use for the application.
- Display relevant information about the prediction model.
- Provide explanation for the terminology used.
- Receive and list all the patients from the database.
- Search for any patient by entering their name in the input box.
- Order the patients by ALSFRS-R score, alphabetically or by clinical visits.
- Display a patient's MND register record on a dashboard.

These functional requirements provided a foundation for the design of the application. It laid out the number of different pages needed and what information to be displayed.

3.4.2 Non-functional requirements

The non-functional requirements for the application were as follows:

- **Usability**
  The design should be simple and clear. Each page should follow the same design pattern, be minimal but display all the relevant information to the user.

- **Performance**
  The web application should be easy to navigate by providing the pages in a navigation bar at all times. It should receive the data from the database as quick as possible to display on the screen.

- **Security**
  The application should provide an authenticated login system so that only authenticated users can enter as it obtains confidential information about patients. It will not display any data if the user is not authenticated.

3.5 Use case

This section analyses the web application's use case diagram and textual description.

3.5.1 Use case diagram

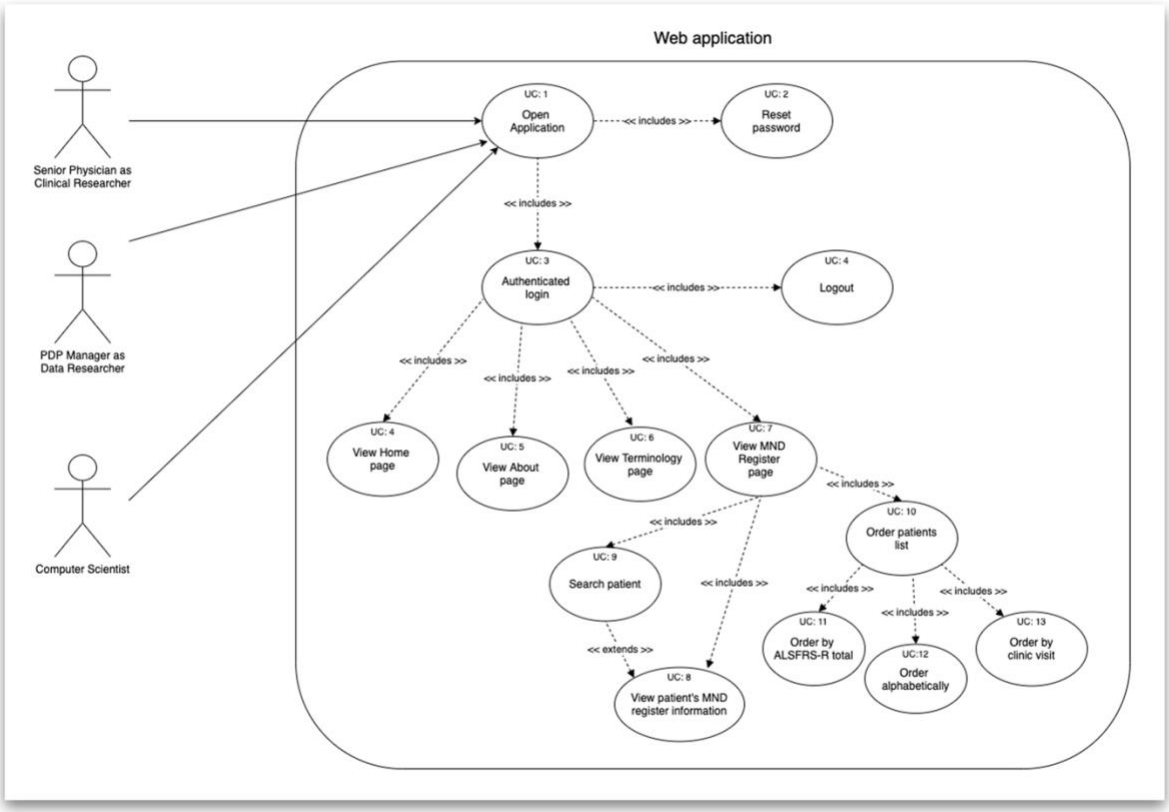

Figure 3.3: Use case diagram for the web application.

The diagram above (`Figure 3.3`: Use case diagram for the web application.) describes the use case for the web application. The three actors are the Senior Physician, the PDP manager and computer scientists. In order to use the application, the user must open the application, where they will be presented with the Login page and sign in using an authenticated email and password, which is set up in the Firebase database. The user can also reset their password if forgotten on the Login page. After successful login, the user can view all the available pages or choose to logout. These use cases all possess the <<includes>> tag because without logging in, they cannot be accessed. From the MND Register page, the user can either view a patient's MND register information by clicking on the patient card or by searching for a specific patient. The user can also order the patients list by the ALSFRS-R total, alphabetically or by clinic visits. Again these use cases contain the <<includes>> tag however the use case 'View patient's MND register records' also contains the <<extends>> tag as it is not necessary to search for a patient to view their information.

3.5.2 Use case textual description

Use case name	Web Application.
Actors	Senior Physician, PDP manager, computer scientists.
Preconditions	The user must have an authenticated email and password.
Trigger	This use case is initiated when the user enters the application.
Description	This use case describes the event of a user interacting with the web application. The user must first sign in. When signed in, the user can navigate through the Home, About, Terminology and MND Register pages. In the MND Register page, the user can search for a patient, order the patient list by ALSFRS-R score, alphabetically or by clinic visits. The user can view a patient's MND register record by clicking on the patient card. The user can choose to log out at any time by pressing on the 'Logout' button on the navigation bar.
Normal scenario	The user signs in successfully using their authenticated email and password and navigates the application freely.
Error scenario	The user fails to sign in as they forgot their password and must reset it by pressing on the 'Forgot password?' button on the Login page.
Conclusion	The use case concludes when the user logs out of the application.

Table 3.1: Use case textual description.

3.6 Architectural design

This section describes the architectural design pattern and UI heuristics used to create the web application.

3.6.1 Model View Controller (MVC)

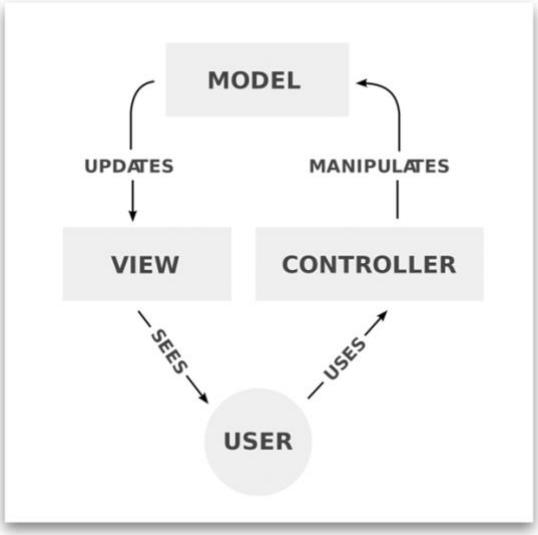

Figure 3.4: Model View Controller architecture. [7]

The architectural design of this project follows the Model View Controller (MVC) protocol (`Figure 3.4`: Model View Controller architecture.). "MVC is an architectural design pattern that separates the user interface from the application. It organises interactive systems into a collection of interaction objects made up of Model-View-Controller triples where the Model is any object (application), the view is an object which provides a visual representation of a model (output) and the Controller is an object which handles input actions, sending messages to the view or model, as appropriate (input)".[18] An example is when a user is logging into the application. In this case, the model is the login form and the view is the Login page. When the user enters their details and logs in, the controller listens if the authentication state has changed or not. On successful login, the state is confirmed and the controller updates the view to display the home page and vice a versa when the user logs out of the application. Another example of MVC is in the MND Register page. When the page is loaded, a loading indicator will appear which indicates that the controller is listening to Firebase to receive the data about the patients. When the data is received, the controller will update the view to list the patients across the screen. This design process is enforced throughout the application.

[7] https://en.wikipedia.org/wiki/Model–view–controller - /media/File:MVC-Process.svg

3.6.2 Nielson's Heuristics

This application was also designed while keeping Jakob Nielson's heuristics in mind. There are ten heuristics and they were created for finding usability problems in UI designs. The heuristics are as follows:[19]

1. **Visibility of system status**

   "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time."

   The application follows this heuristic by displaying appropriate error messages upon login (wrong email/password) and loading indicators when data is being fetched from the database.

2. **Match between system and the real world**

   "The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order."

   The application uses terms and phrases related to the MND healthcare domain to explain the prediction model.

3. **User control and freedom**

   "Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo."

   With a dominant navigation bar at the top of the application, the user can navigate back to any section if they leave accidentally. When the logout button is pressed, a confirmation will pop up. When an action is triggered accidentally, for example the clinical visits button, there is a clear, red 'x' icon so the user can exit.

4. Consistency and standards

"Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions."

This application follows a consistent design throughout all sections. Same style components are used on each page with the same colour scheme which provides a consistent flow when navigating through the application.

5. Error prevention

"Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action."

When the user logs out, a pop up will appear to confirm whether the user wishes to log out which prevents accidental log outs. If the user hovers over the 'x' symbol beside an item inputted in the medicine or trials sections, a popup will appear asking the users if they are sure they want to delete the item.

6. Recognition rather than recall

"Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate."

The Home page describes clearly, the purpose for the application and what each section carries out. The About page explains the prediction model and the terminology page describes the terms and phrases used. The user can view these pages at any time without the need to remember all the information.

7. Flexibility and efficiency of use

   "Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions."

   Using the search bar and ordering system can drastically improve the efficiency of viewing a patient's MND register record.

8. Aesthetic and minimalist design

   "Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility."

   The overall design of the application is consistent and minimalistic. Only relevant information is displayed and is short and concise.

9. Help users recognise, diagnose and recover from errors

   "Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution"

   Clear error messages are displayed if the user enters the wrong email or password upon login or if there are too many unsuccessful login attempts.

10. Help and documentation

    "Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large."

    The Home page describes how to navigate through the application the About page provides a detailed explanation of the prediction model, the Terminology page describes the terms and phrases used and information popups explain the scores of each ALSFRS-R value in the Clinical visit tab.

Using Nielson's heuristics helped locate any issues with the design of the application. With constant testing carried out and through user feedback for each iteration of the design, the application was improved by following these heuristics. The examples mentioned above can be reviewed in the User interface design section 3.7.

3.7 User interface design

This section presents the up to date version of the web application, displaying the user interface for each page.

3.7.1 Login page

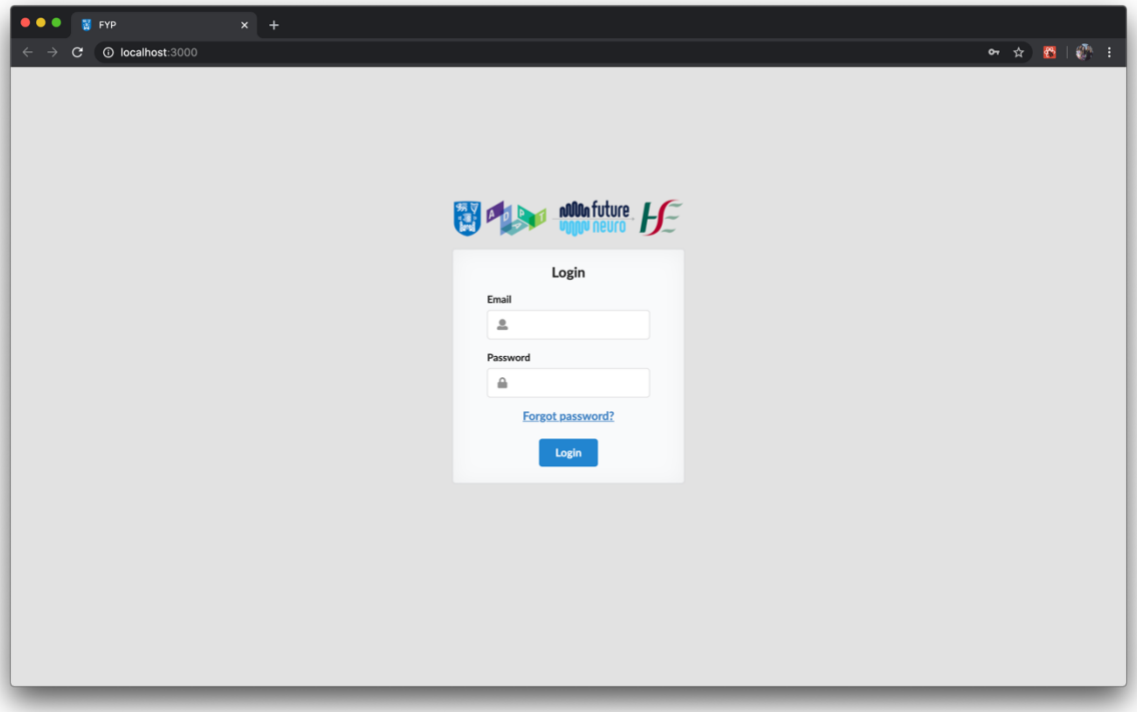

Figure 3.5: Login page user interface (UC 1).

The login page (`Figure 3.5`: Login page user interface (UC 1).) has a very simple UI design. It contains a login form where the user is prompted to enter their email and password and logged in when the login button is pressed (UC 3). Above the form, all the logos of the associated parties of the project team are displayed. Above the login button is a link for the user to reset their password in case it is forgotten (UC 2). When the link is pressed, a popup will appear which will ask the user to enter their email and will send a password reset email if the user exists in the database, otherwise an error message will appear. If the user exists, a green confirmation message will be displayed (Figure 3.6). Error messages will also be displayed below the form if the password is incorrect, the email is not recognised or too many unsuccessful login attempts were tried (`Figure 3.6`: Login error(red) and confirmation(green) messages.).

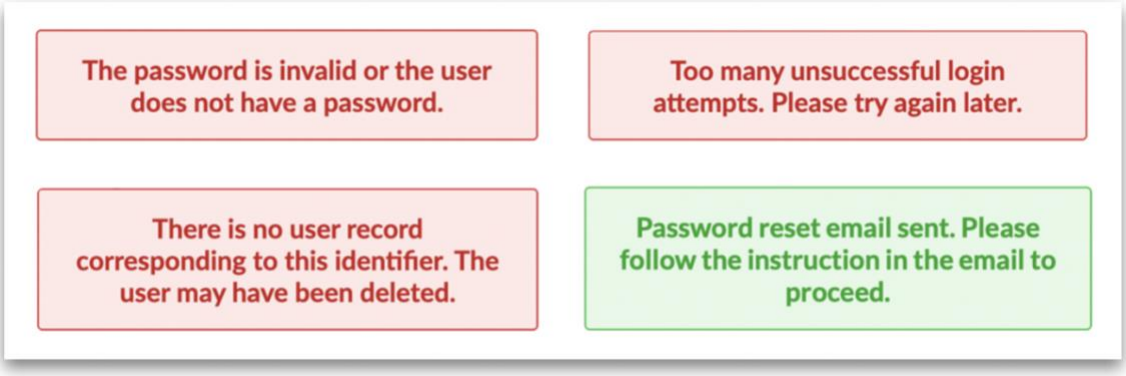

Figure 3.6: Login error(red) and confirmation(green) messages.

The application will also prevent an intruder from trying to access the application. If the same email is constantly used to try reset the password, all requests will be blocked and an error message will appear (`Figure 3.7`: Password reset error message after too many emails sent.).

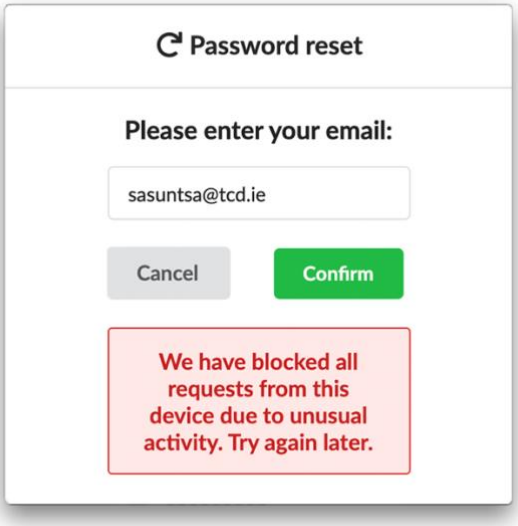

Figure 3.7: Password reset error message after too many emails sent.

3.7.2 Home page

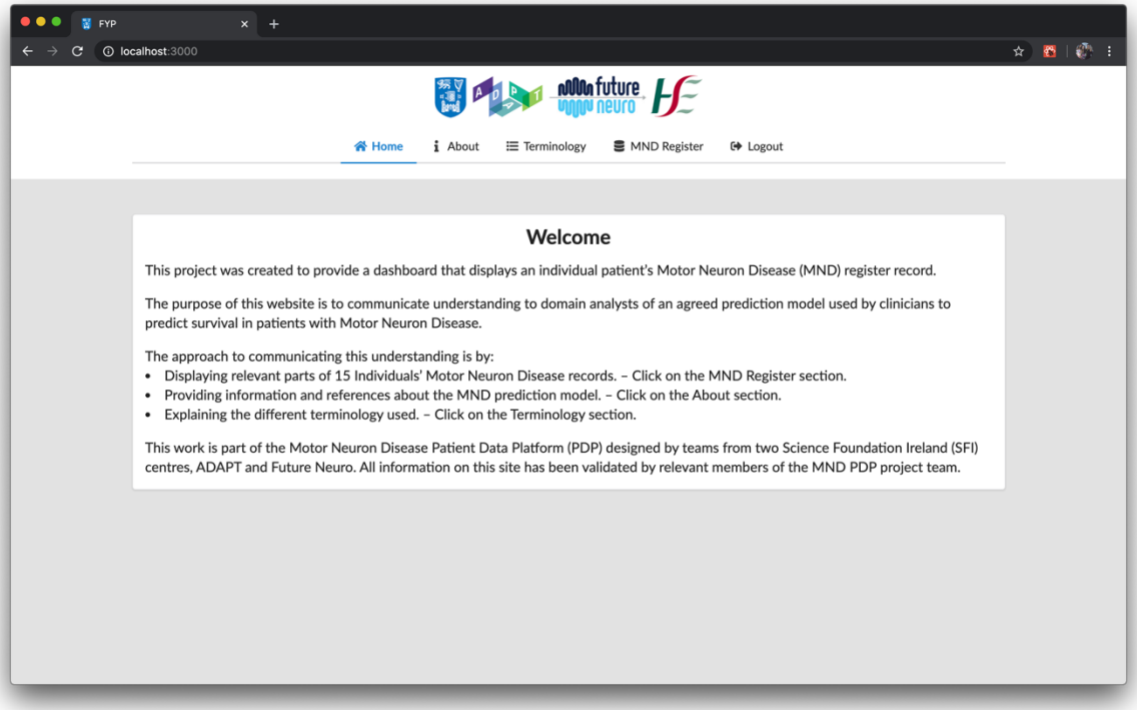

Figure 3.8: Home page user interface (UC 5).

After the user logs in, they directed to the Home page (`Figure 3.8`: Home page user interface (UC 5).) where they are welcomed by information about the application. This section explains the background of the application, it's purpose and what each page will provide. The navigation bar is located at the top of the page and is present on each page. The user can navigate to any section of the application easily by clicking on the page names on the navigation bar. The user can also logout at any time by pressing the logout button (UC 4) where they will be prompted to logout or stay on the current page (`Figure 3.9`: Logout prompt (UC 4).).

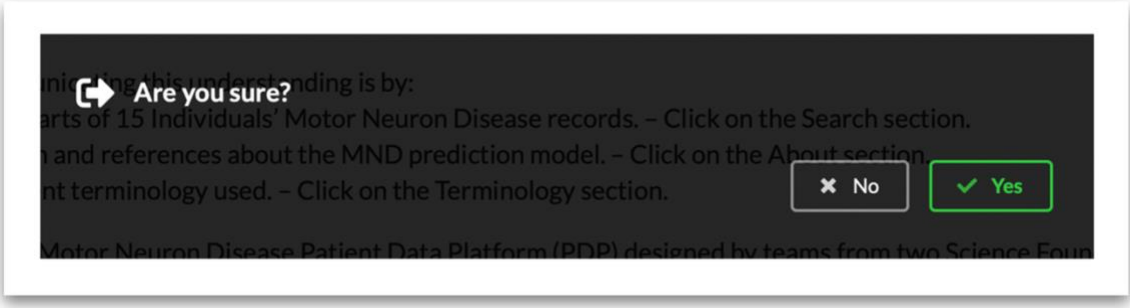

Figure 3.9: Logout prompt (UC 4).

3.7.3 About page

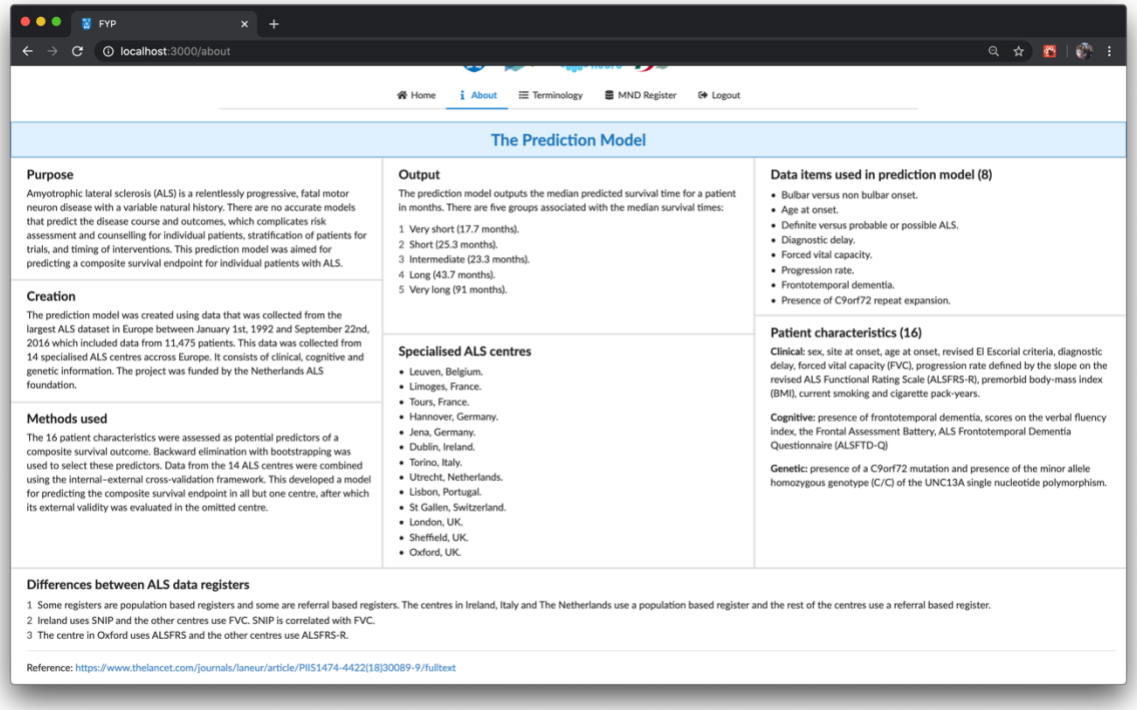

Figure 3.10: About page user interface (UC 6).

The About page (`Figure 3.10`: About page user interface (UC 6).) displays a descriptive infographic about the prediction model. This page provides the user quick access to the important details about the prediction model if needed. The sections covered are:

- The purpose of the PM.
- The output of the PM.
- The data items used in the PM.
- The creation of the PM.
- The methods used for the PM.
- The specialised ALS centres involved.
- The patient characteristics analysed.
- The differences between the ALS data registers used in each centre.

Each section is presented in a ‘Segment’ block. The page is divided into three rows. The first row displays the blue PM title. The second row contains the segments from ‘Purpose’ to ‘Patient characteristics’ which are split into three columns and the last row contains the ‘Differences between ALS data registers’ segment. The link to the published document about the prediction model is also referenced at the bottom of the page.

3.7.4 Terminology page

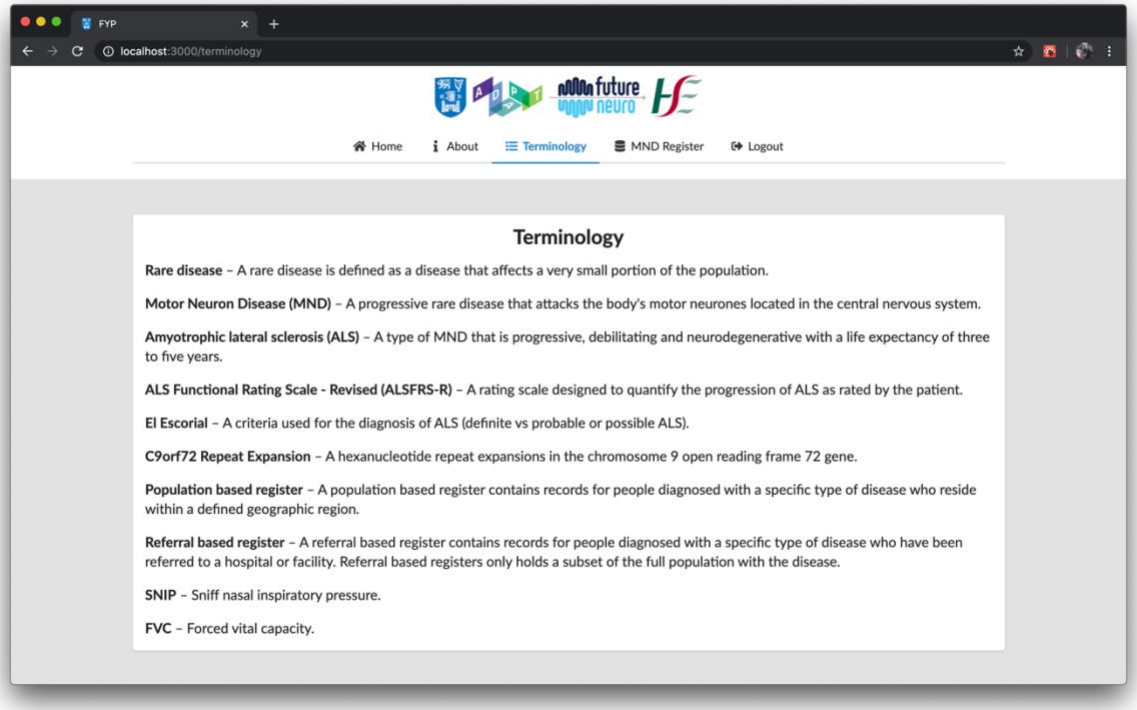

Figure 3.11: Terminology page user interface (UC 7).

The Terminology page (`Figure 3.11`: Terminology page user interface (UC 7).) is similar to the Home page. It contains one segment block in the centre of the page which describes the different terminology used throughout the application. This may be useful for the user if they are unsure about some of the words or phrases used.

3.7.5 MND Register page

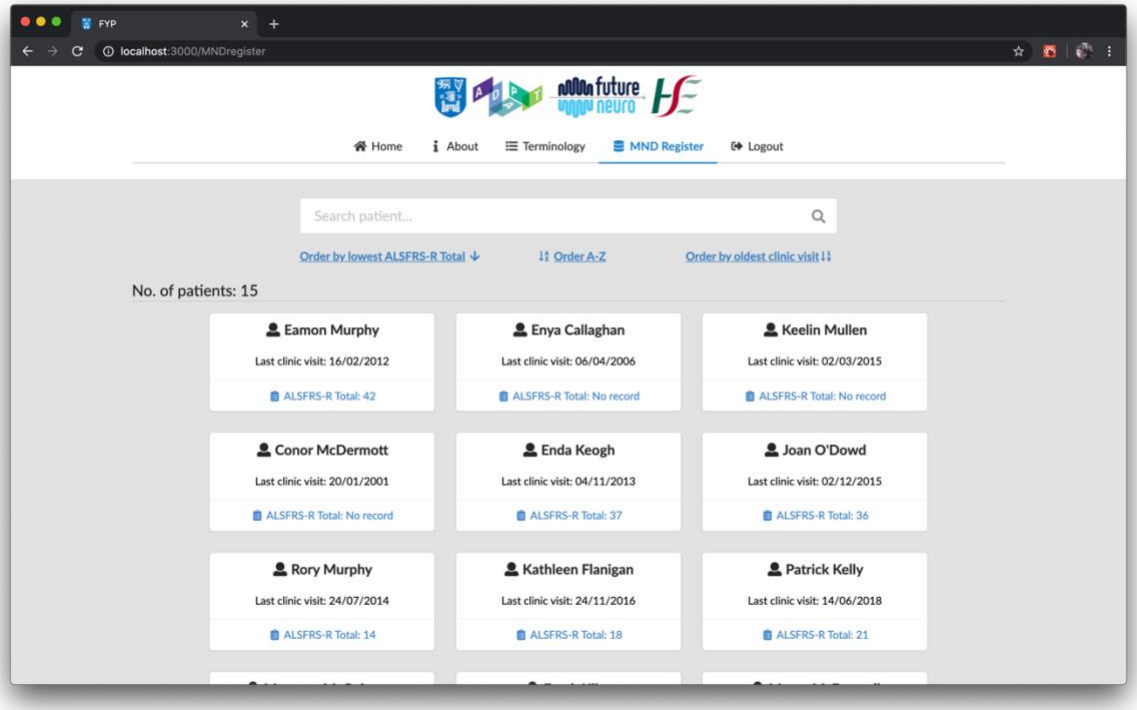

Figure 3.12: MND Register page user interface (UC 8).

The MND Register page (`Figure 3.12`: MND Register page user interface (UC 8).) is where the user will find the patients' MND register records. An input box is located at the top of the page and is used to filter through the patients by entering a patient's first or last name (UC 10). Below the input box, are buttons the user can press to order the patient list by the lowest/highest ALSFRS-R score (UC 12), alphabetically from A-Z or Z-A (UC 13) and by the oldest/latest clinic visit (UC 14). Below these buttons to the left, is an indicator letting the user know how many patients are listed in the MND register. All the patients from the database are listed using a Grid and List system. The patients are first received from the Firebase database and then displayed onto the page. As the page is fetching the data, a loading indicator (`Figure 3.13`: Loading indicator.) will appear before displaying the patient cards. The patients are ordered as they appear in the database, however the user can order them how they like using the order buttons. The patients are listed in cards. Each card contains the patient's name, last clinic visit and total ALSFRS-R score. The options for ordering the patients and the information displayed on the cards was as specified by the project team.

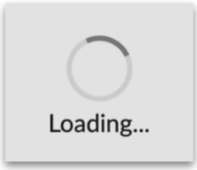

Figure 3.13: Loading indicator.

3.7.6 Patient dashboard

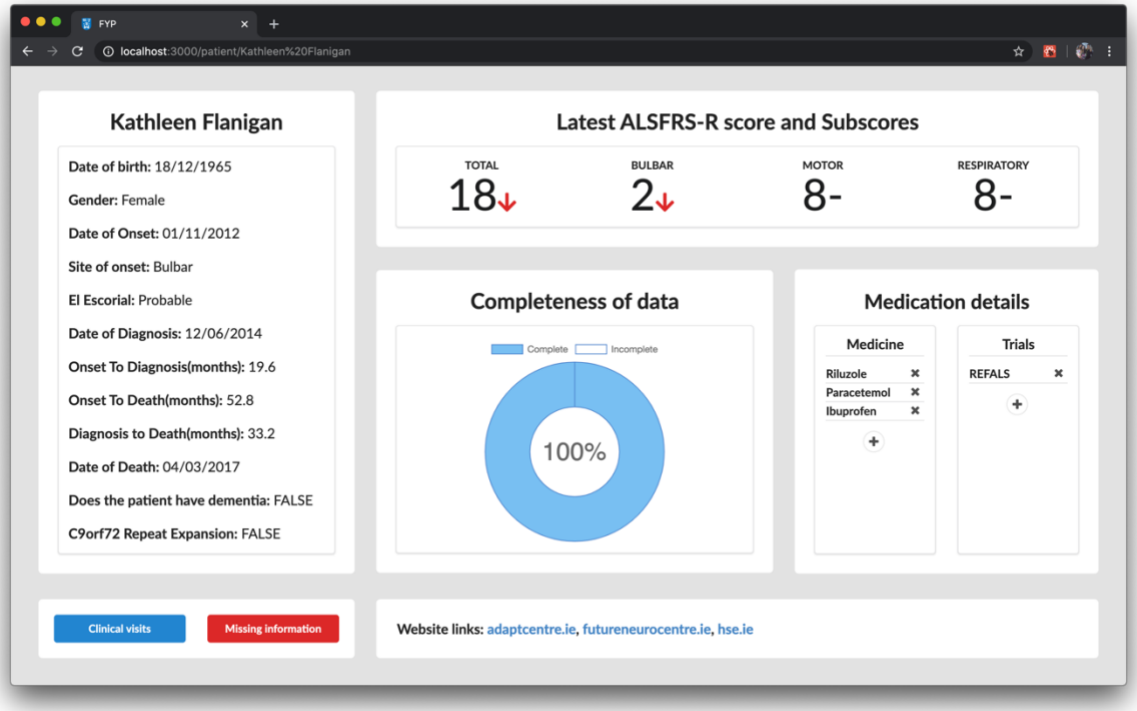

Figure 3.14: Patient dashboard user interface (UC 9).

The Patient Dashboard UI (`Figure 3.14`: Patient dashboard user interface (UC 9).) is where most of the background work occurs. The user is greeted with the above screen when they click on a patient card with the corresponding patient information being displayed (UC 9). The page is divided into two rows. The first row contains the four main segments which is divided into two columns. The fist column displays the patient's demographic data. The second column is further divided into two rows with the second row being split into two columns. It displays the latest ALSFRS-R score and subscores in the first row and the completeness of the data represented as a doughnut chart and the medication details in the second row. The medication details segment contains two boxes where the user can enter any medication or trials the patient is on by pressing the '+' symbol or remove them by pressing the 'x' symbol. If the cursor is hovered over the 'x' symbol beside an inputted item, a popup will appear asking the user if they are sure they want to delete the item (`Figure 3.19`: Confirmation popup.). The second row is divided into two columns, the first being

the segments that contains the 'Clinical Visits' and 'Missing Information' buttons and the second being the references segment containing the website links for ADAPT, FutureNeuro and HSE. When the 'Clinical Visits' button is pressed, a modal will appear containing three tabs.

The first tab displays the clinical visits table (`Figure 3.15`: Patient clinical visits table.) which represents the data relating to the patient's clinical visits, if it was recorded and added to the database.

Beaumont Hospital visit	Clinical visit date	SNIP(cm H2O) Occluded Method	ALSFRS-R												
			Total/48	1	2	3	4	5a	5b	6	7	8	9	10	11	12
✓	14/06/18															
✓	24/08/17															
✓	20/04/17															
✓	15/12/16															
✓	15/09/16		21↓	0	1	2	2	1		1	2	2	0	4	4	2
✓	28/04/16		24↓	0	1	3	3	2		2	1	2	0	4	4	2
✓	21/01/16		26↑	1	2	3	3	2		2	1	2	0	4	4	2
✓	22/10/15		24↓	1	0	3	3	2		2	1	2	0	4	4	2
✗	15/06/15															
✓	11/06/15		29↓	1	1	3	3	2		2	2	2	1	4	4	4
✓	26/03/15		30↓	1	1	3	3	3		2	2	2	1	4	4	4
✗	02/03/15		31↓	1	1	3	3	3		2	3	2	1	4	4	4
✓	18/12/14		33↑	3	1	3	3	3		3	3	2	1	4	3	4
✓	25/09/14		27↓	2	1	1	3		3	2	2	2	1	3	3	4
✓	17/07/14															
✓	19/06/14		33	3	1	3	3	4		2	3	2	1	4	3	4

Figure 3.15: Patient clinical visits table.

The second tab displays the graph of the total ALSFRS-R score and the graphs of the bulbar, motor and respiratory ALSFRS-R subscores relating to the day of the clinical visit (`Figure 3.16`: Graphs of total ALSFRS-R and ALSFRS-R sub scores.). Each score will appear with the value and date when hovered over.

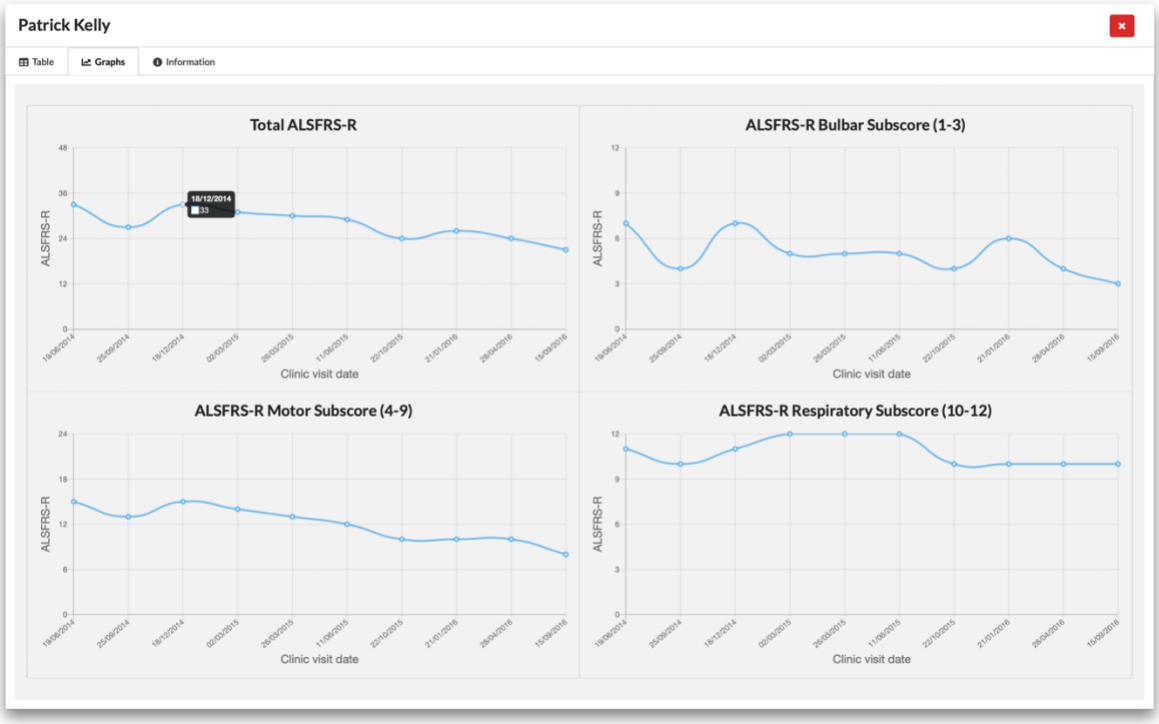

Figure 3.16: Graphs of total ALSFRS-R and ALSFRS-R sub scores.

The final tab displays information about ALSFRS-R questions (`Figure 3.17`: Information about ALSFRS-R questions.). A pop up will appear explaining the values of the scores 0-4 for each question when the cursor is hovered over the information icon.

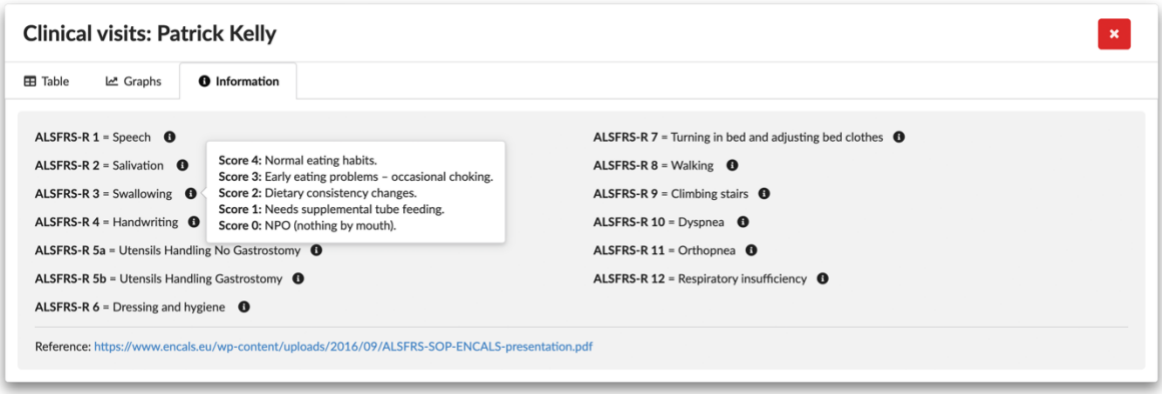

Figure 3.17: Information about ALSFRS-R questions.

When the 'Missing Information' button is pressed, another modal will open displaying the information that is missing from the patient's demographic data if it exists, otherwise it will state that "All information has been recorded." (`Figure 3.18`: Modal of missing information.). A disclaimer is presented at the bottom of the modal if missing information exists stating that 'If missing information is related to death

(e.g. 'Date of Death'), the patient may still be alive.', letting the user know that this information is not missing.

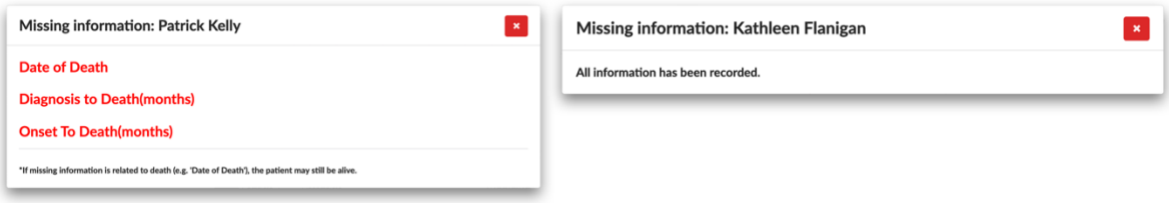

Figure 3.18: Modal of missing information.

Figure 3.19: Confirmation popup.

4. Implementation

This chapter explores how the web application was created by discussing the page routing, Firebase's integration and the different pages and components. The web application was developed using the React framework. The react project was created by running the command line 'npx create-react-app <name of app>' in Terminal, which creates a template React folder to base the project on. Npx is a package runner tool that comes with Node Package Manger (NPM). Authenticated login/logout, database storage and website hosting is provided by Firebase. The application is styled using components imported from Semantic UI such as buttons, tables, modals, popups and forms, to mention a few and charts that are imported from Chart.js. This application is a single page application (SPA) which renders the different components using React Router.

4.1 Routing

The application routes through the pages using React Router and is carried out in the 'App.js' file (see `1. App.js` in appendix). React Router renders components to be displayed based on the route being used in the url.[20] It connects the path with the component. For example, the AboutPage component is connected to the path '/about'. So when the user clicks on the About page, the url path will change to '/about' and the component will be rendered. The 'index.js' file (see `2. index.js` in appendix) uses ReactDOM.render() to render the <App> component which in turn will render the rest of the components with React Router as appropriate. In order for React Router to work, the <App> component is wrapped inside the <BrowserRouter> element. The application will check if the user is authenticated with Firebase, using the onAuthStateChanged() API. When the application is opened, the user is not authenticated and therefore will render the Login page. When the user signs in, the authentication state is confirmed and the home page will render. From here, they can navigate to any page by pressing the buttons on the navigation bar which are links to the url paths of the pages. All of the components are placed inside the <Route> tag which will render the components based on the url paths clicked. The application renders the current url path which is received from this.props.location.pathname and when the page is refreshed, it will stay on the same page. This pathname is received by using the withRouter() function which keeps a history of all the routes. If the application is left idle for 30 minutes, the user will be logged out of Firebase and therefore will no longer be authenticated. The Login page will then be rendered again.

4.1 Firebase

Firebase is implemented into the application through the 'Firebase.js' file (see `3. Firebase.js` in appendix). Every Firebase project contains its own configuration which is used to initialise Firebase into the code. Once initialised, access to the Firebase databases and authenticated users is available. Three Firebase features are used in this project:

Authentication

Firebase provides their own Authentication feature which allows you to authenticate users to an application. Users can be added manually from the Firebase console and each contain their own unique user identification. A number of different sign in methods are provided such as, email/password, Facebook, Twitter, and Google account to name a few. The method used in this application is email/password. When the user is signing in, the application will check if they are an authenticated user with the signInWithEmailAndPassword() API (see `4.1 login(e)` in appendix). If the email and password matches with a user in the Firebase authenticated users list, the user is authenticated and will enter the application, otherwise an error message will appear. If the application is left idle for 30 minutes, the user will be logged out automatically.

Databases

Firebase provides two types of cloud-based databases, the Realtime Database and Cloud Firestore. The Realtime Database stores data in a JSON tree and allows users to import JSON files directly, whereas the Cloud Firestore stores data in documents that are arranged in collections.[21] Both versions of databases are used in this application.

The sample data received from the project team was originally an excel file. This file was converted into JSON format and imported into the Realtime Database from where the data is read. The Realtime Database JSON tree format can be viewed in `Figure 4.1`: Firebase Realtime Database JSON tree format. below.

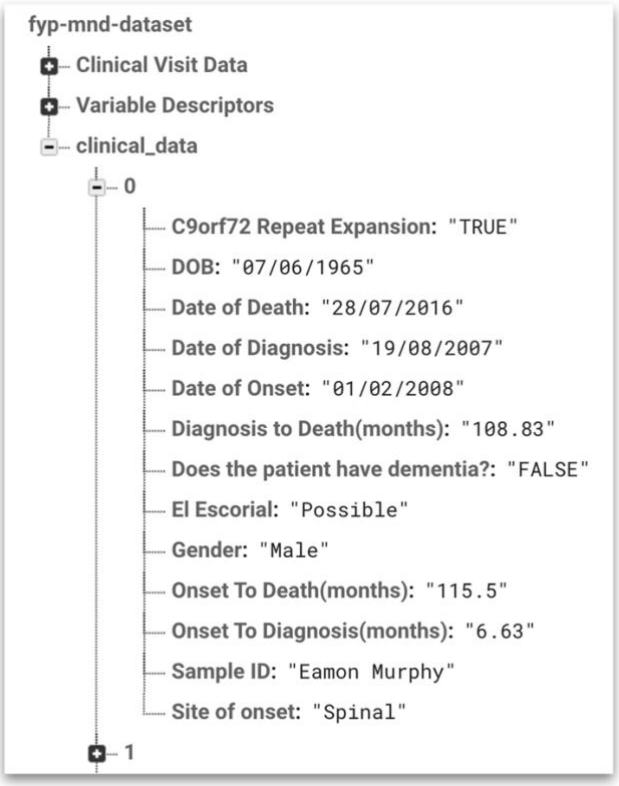

Figure 4.1: Firebase Realtime Database JSON tree format.

The Cloud Firestore is used to store each patients medication details (medicine and trials) which are inputted by the user from the Patient dashboard page. It is stored in the 'medication' collection where each patient will have their own document with their medication details, if entered by the user. The Firestore database format can be viewed in `Figure 4.2`: Firebase Firestore format. below.

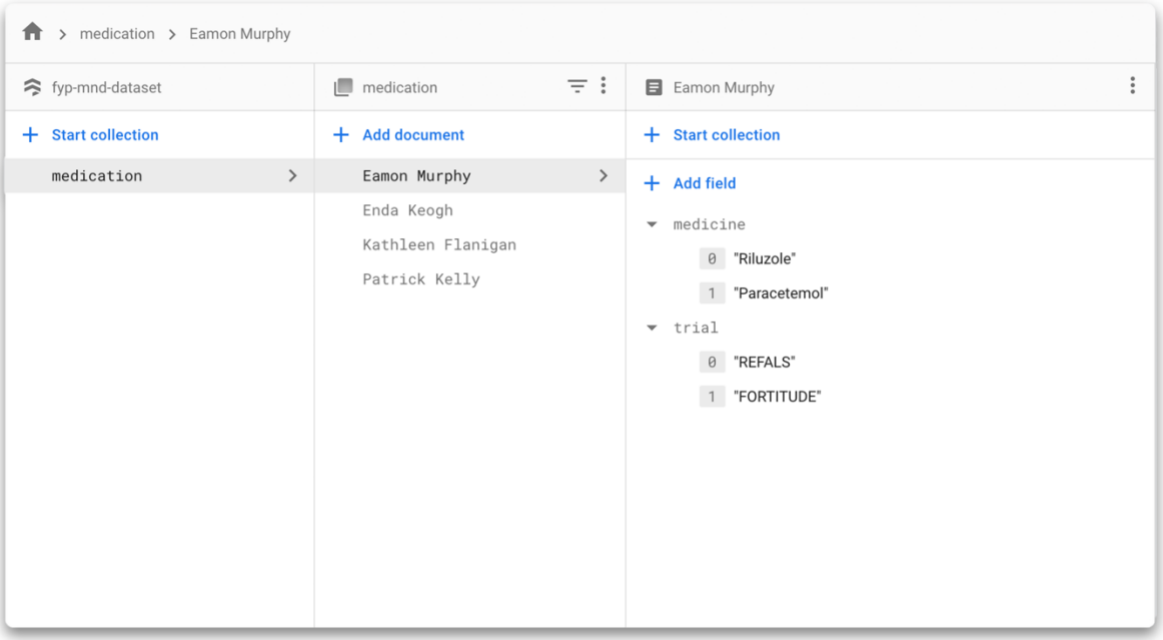

Figure 4.2: Firebase Firestore format.

Hosting

Firebase also allows users to host their websites free of charge. It provides a default domain name, however the user can later change it. For this project the domain name is 'fyp-mnd-dataset.web.app'. Hosting the web application was very useful, especially during the COVID-19 situation as it allowed me to share it with the project team, where they could use it themselves and return feedback.

4.1 Pages

This application consists of six main pages which are explained below. The navigation bar is rendered at the top of every page. These pages set up the dimensions and style using the Grid system that is imported from Semantic UI and render the different components.

4.1.1 Login Page

This page renders the login form and carries out the authenticated login check. It is written in the 'LoginPage.js' file. The login form takes two inputs from the user (email and password) and when the login button is pressed, the login() function is called (see `4.1 login(e)` in appendix). This function checks whether the user's details exist in the Firebase Authentication users list and whether they are correct, otherwise appropriate error messages will be displayed as seen in `Figure 3.6`: Login error(red) and confirmation(green) messages. and `Figure 3.7`: Password reset error message after too many emails sent. in `3.7.1 Login` page. This check is carried out by calling the Firebase API, singInWithEmailAndPassword(email, password) which takes the values from the login form inputs. If there is an error, the appropriate error message will be set and made visible on the screen. The login form also allows the user to reset their password if it is forgotten by pressing on the 'Forgot password?' button. When this button is pressed a modal will appear that contains an input box for the user to enter their email. When an email is entered and the confirm button is pressed, the sendResetEmail() function will be called (see `4.2 sendResetEmail(email)` in appendix). This function calls the Firebase API, sendPasswordResetEmail(email) which takes the value inputted in the input box. Again, if there is an error, the appropriate error message will be set and made visible on the screen, otherwise a confirmation message will be displayed and an email will be sent to the user.

4.1.2 Home Page

The Home page renders a segment which is inside a container that displays the welcome information to the user. It is written in the 'HomePage.js' file. It is a basic page that does not contain any functions.

4.1.3 About Page

The About page renders the infographic which provides an explanation of the prediction model as discussed in `3.7.3 About` page. It is written in the 'AboutPage.js' file. This page is divided into three rows, with the second row containing three columns. This is achieved by using the <Grid> component. There are eight sections where the text is placed inside components. Both components are imported from Semantic UI. Again, like the Home page, this page does not contain any functions.

4.1.4 Terminology Page

Like the Home page, the Terminology page renders a segment inside a container that displays the terminology used throughout the application. It is written in the 'TerminologyPage.js' file. This page does not contain any functions.

4.1.5 MND Register Page

The MND Register page pulls the data from Firebase and renders a list of the patients onto the screen. It is written in the 'MNDregister.js' file. Firstly this page will render everything above the divider (search box, order buttons, the number of patients) and a loading indicator underneath. Once the data has been fetched, it will render the list of patients.

When the page is loaded, it will fetch the patients' demographic data stored in the 'patients' global array and the clinical visits data stored in the 'clinical_visits' global variable which is carried out in the componentDidMount() function (see `5.1 componentDidMount(` in appendix). The patients are rendered in a list using the map() function and are presented in a <Card> component, imported from Semantic UI. When the user searches for a specific patient, the 'patients' array is filtered using the filter() function and the result(s) is stored in a new array called 'filteredPatients' (see `5.2 filteredPatients` in appendix). This array is used when rendering the list of patients. If the search box is empty, all patients will be listed and if a value is entered, it will filter the array based on the entered value and list the corresponding patients.

Since the patients' demographic and clinical visits information are stored in two separate arrays, the lowest ALSFRS-R score and last clinic visit must be obtained from the 'clincal_visits' array. This is carried out in the the getClinicalInfo() function (see `5.3 getClinicalInfo(patient, type)` in appendix). This function stores each patient's total ALSFRS-R score and clinic visit date in two separate temporary arrays (ALSFRSR, Date) by matching the patients name from both the 'patients' and 'clincal_visits' global arrays. It takes two parameters, 'patient' which is the current patient being mapped and 'type' to determine which value to return. If the type is 'date', the function will sort out the dates and return the latest clinic visit date recorded. If the type is 'ALSFRSR', it will return the lowest ALFRS-R total score. These values are then displayed on the patient cards.

The patient list is ordered in the sortBy() function (see `5.4 sortBy(type, sortFrom)` in appendix). This function takes two parameters, 'type' to determine which button is pressed and 'sortFrom' to determine how to sort (e.g. lowest or highest ALSFRS-R score). If the 'Order A-Z' button is pressed, the sortBy('alphabetical', 'start') is called. This function will then sort out the patients by last name from A to Z and will change to button to render 'Order Z-A'. Since the 'patients' array does not contain the ALSFRS-R total score and clinic visits data, when the order by ALSFRS-R total score or order by clinic visit date is pressed, it will add the latest ALSFRS-R score and clinic visit to each patient's object in the array, which is obtained from the 'clinical_visits' array. The patient list will be ordered similarly as mentioned above corresponding to which button is pressed. The ALSFRS-R total score and clinic visit date is then removed from the patient object as it will interfere with the completeness of data in the Patient Dashboard by adding an extra value.

4.1.6 Patient Dashboard

The Patient Dashboard page renders the dashboard with the patients demographic and clinical visits information. It is written in the 'PatientDashboard.js' file. The page is divided into two rows. The first row is divided into two columns with the first being the demographic information segment where the <DemographicInfo> component is rendered. The second column contains the latest ALSFRS-R scores and subscores, the completeness of data and medication segments where the <ALSFRSRscores>, <Completeness> and <Medication> components are rendered. This column is further divided into two rows with the second row divining into two columns. The next row from the overall grid contains the segment with the 'Clinical visits' and 'Missing information' buttons in one column, where the <ClinicalVisits> and

<MissingInfo> components are rendered and the references segment in another column. These components will be discussed in more detail in `4.2 Components`. The shape of the dashboard is achieved by using the Semantic UI Grid system (see `6.1 Structure of dashboard` in appendix).

This page receives the patient's demographic and clinical visits data as props from the MND Register page when clicked on a specific patient and will store this data in the local storage of the browser. To resolve an error on a refresh, it will first check if this data is stored in local storage otherwise it will receive it as a prop (see `6.2 Data check` in appendix). Now when the page is refreshed, it will receive the data from the local storage, without any issues and when the user exits out of the patient dashboard, the local storage will be cleared so the same patient information will not be displayed on another patient's dashboard.

The Patient Dashboard page will also order the clinical visit data by the latest date, which is displayed in the clinical visits table. This is carried out in the orderByLatestDate() function (see `6.3 orderByLatestDate()` in appendix). This function will filter through the 'clincal_visit_data' array and store only the selected patient's data into the 'single_clinic_data' global array. This array is then sorted by combining the dates as one number, reversing them and comparing the largest number from the lowest. For example, 13/05/2011 and 22/03/2015 becomes 11025031 and 51023022. The comparison is made and the most recent date (largest number) is added to the top of the array. The Patient Dashboard carries out two other functions, getMissingInfo() and getCompletness() where the results are sent to the <MissingInfo> and <Completeness> components. These functions will be discussed in `sections 4.2.3` Completeness `and 4.2.9` Missing information.

4.2 Components

Components provide a great way to clean up code and have an ordered project structure. Instead of having all of the code in one file, it is possible to separate the code chunks into different components, each of which execute a certain function. The 'Patient Dashboard' page is where most of the technical code is written, therefore it is divided up into nine components. Each component receives the relevant data as props from their parent components. For example, the 'Patient Dashboard' page will send the patient's demographic data to the 'Demographic Information' component, or the patient's clinical visits data to the 'ALSFRS-R Scores' component.

4.2.1 Demographic Information

The <DemographicInfo> component renders the patient's demographic information. It is written in the 'DemographicInformation.js' file. It is a simple component that displays the patient's name as the header and the corresponding details underneath

in a component. The <DemographicInfo> component can be viewed in Figure 3.14: Patient dashboard user interface (UC 9)..

4.2.2 ALSFRS-R Scores

The <ALSFRSRscores> component renders the patient's latest ALSFRS-R total score and ALSFRS-R Bulbar, Motor and Respiratory subscores in a horizontal component. It is written in the 'ALSFRSRscores.js' file. There are four functions that are used to determine whether the patient's scores are increasing, decreasing or the same as the previous recorded score (see 7. ALSFRSRscores.js in appendix). The following functions are listed below:

- 7.1 getTotalScore().
- 7.2 getBulbarScore().
- 7.3 getMotorScore().
- 7.4 getRespiratoryScore().

The getTotalScore() function receives the total scores from the patient's clinical visits data and checks if the latest score is higher, lower or the same as the previous score. If higher, it will display the score with a green arrow, if lower, it will display the score with a red arrow and if it's the same, it will display the score with a dash line. The getBulbarScore(), getMotorScore() and getRespiratoryScore() functions get the relevant scores, adds them up and then carries out the same comparison as the above function. For example the getBulbarScore() function will take the ALSFRS-R scores 1,2 and 3 and add them up for each recorded clinical visit and then carry out the comparison. If the patient does not have any clinical visit data recorded, a dash line will be displayed under each score. The <ALSFRSRscores> component can be viewed in Figure 3.14: Patient dashboard user interface (UC 9)..

4.2.3 Completeness

The <Completeness> component renders the completeness of the patient's demographic information as Doughnut chart. It is written in the 'Completeness.js' file. The chart is imported from the Chart.js library and the data is calculated from the getCompleteness() function (see 6.4 getCompleteness() in appendix). This function is calculated in the Patient Dashboard page and is then sent to the component as a prop. The function obtains the number of values missing from the demographic data and divides it by the total amount to get the completeness value. The <Completeness> component can be viewed in Figure 3.14: Patient dashboard user interface (UC 9)..

4.2.4 Medication

The <Medication> component renders the two boxes where the user can input any medicine the patient is taking or any trials they are on. It is written in the 'Medication.js' file. The two boxes contain a circular '+' symbol where the user can press to activate an input box. Once an item is entered, it will be listed with an 'x' symbol beside it. When the cursor is hovered over this symbol, a popup will appear asking the user if they are sure they want to delete it and they can confirm by pressing on the symbol, which will remove the item from the list. The <Popup> component imported from Semantic UI is used. The items that are inputted are stored in the Firebase Firestore database. This component has four main functions:

getItems()

This function receives the items from Firebase and stores them into a local array (medicine_list or trial_list) which is used to display the items (see `8.1 getItems(type)` in appendix). It takes the parameter 'type', which is used to determine if the medicine or trials are being received. Each patient has their own 'document' in the database. When the function calls the Firebase API to receive the data, it checks if the patient document exists, otherwise it will set the local array as empty. If it does exist, it will check if there is any data inputted to the document. If so, it will pull the items and store it in the local array, otherwise it will set the array as empty, as the document may exist but be empty. These check prevents any errors from occurring.

displayItems()

This function displays the items in a list (see `8.2 displayItems(type)` in appendix). Again, it takes the parameter 'type', to determine which values to display. It renders the items, using the map() function which goes through an array and displays each value. The 'x' symbol is also rendered beside each item.

addItem()

This function will take a newly inputted item and add it to the database as well as the local array to be rendered (see `8.3 addItem(e, inputType)` in appendix). Since the local array is nested, all the items are first pushed into a single temporary array, as Firebase does not accept nested arrays. This temporary array is then used to update the corresponding patient's document in Firebase, adding the newly inputted item.

deleteItem()

This function will delete an item from the Firebase Database and pull the data again (see

8.4 deleteItem(type) in appendix). It takes the item name as a parameter. Like the previous function, it will add all the items to a temporary array but leave out the item to be deleted by checking if the names are the same. This array is then used update the corresponding patient's document in Firebase, removing the item that was deleted.

The <Medication> component can be viewed in Figure 3.14: Patient dashboard user interface (UC 9)..

4.2.5 Clinical Visits

The <ClinicalVisits> component renders the blue 'Clinical visits' button. It is written in the 'ClinicalVisits.js' file. This component renders a modal which opens when the button is pressed and displays the Table, Graphs and Information tabs. In the first tab, the <Table> component is rendered, in the Graphs tab, the <LineChart> component is rendered where the four ALSFRS-R score graphs are displayed and in the Information tab, the <InfoPopup> component is rendered where the ALSFRS-R questions are explained. The <ClinicalVisits> component can be viewed in Figure 3.14: Patient dashboard user interface (UC 9)..

4.2.6 Table

The <Table> component renders the clinical visits table for each patient. It is written in the 'Table.js' file. One of the main aspects the table provides is checking if the total ALSFRS-R score is higher, lower or the same as the previous score. In order to accomplish this, the data was split into two arrays and rendered separately. If the rows of the table begin with undefined/empty values, these rows are stored in the 'undef' array and the rows with data are stored in the 'data' array. These arrays are then rendered separately in the same table. This was necessary because if the table started with an empty row, the total scores could not be checked correctly. This occurs in the getUndefined(), getData(), printUndefined() and printData() functions which are explained below.

getUndefined()

This function receives all the empty rows of the table and stores them into the 'undef' global array, i.e. if the table begins with empty rows (see 9.1 getUndefined() in appendix). Firstly it checks if the first total value is undefined and if so, it will go through a loop and store the row data into the array and will break if the total value is not undefined. Otherwise the function will not run. This way all of the empty rows at the start of the array are received.

getData()

This function will skip the empty rows at the beginning of the table and store the rows including and after the first row that contains data into the 'data' global variable (see `9.2 getData()` in appendix). Firstly it checks if the first total value is undefined, otherwise it will store all of the rows into the array. If it is undefined, it will check how many rows from the beginning are empty and skips them. If a row is not empty, it will store that row and the following rows into the global array.

printUndefined() & printData()

The printUndefined() and printData() functions print the two arrays to make up the clinical visits table (see `9.3 printUndefined()` & `9.4 printData()` in appendix). Firstly they call the getUndefined() and getData() functions to receive the table rows. The map() function is then used to render each row with its corresponding data. First the 'undef' array will be printed and then the 'data' array will be printed, combining them together to make one table.

The total scores are then checked with the checkTotal() and checkIfPrevUndefined() functions. The checkTotal() function will take a total value and compare it with the previous value to check if its higher, lower or the same (see `9.5 checkTotal()` in appendix). If higher, it will render the value with a green arrow going up, if lower, it will render the value with an arrow going down and if the same, it will render the value with a dash line. First it checks if the value is undefined and will skip it if so. Then it checks if the score is the last value in the table where it will render just the value with no symbol. The function will assigned the previous value to the 'prev' global variable for comparison. If the previous value is undefined, the checkIfPrevUndefined() function is called where it will skip all of the undefined values and assign the next real value to the global variable (see `9.6 checkIfNextUndefined()` in appendix). This is done recursively checking if the value is undefined and assigning the next value to it until it finds a real value. Now the function will carry out the comparisons checking if 'total' is equal, less than, or greater than the 'prev' value rendering the correct symbol beside the total value. The <Table> component can be viewed in `Figure 3.15`: Patient clinical visits table..

4.2.7 Line Chart

The <LineChart> component renders the line charts for the ALSFRS-R score and subscores. It is written in the LineChart.js file. Like the Doughnut chart, the line chart is also imported from the Chart.js library. This component contains similar functions

as the <ALSFRS-Rscores> component (see 10. LineChart.js: 10.1 getTotalScores(), 10.2 getBulbarScores(), 10.3 getMotorScores(), 10.4 getRespiratoryScores() in appendix). These functions carry out similar actions. They check if a value is not undefined and add it to the 'scores' global array along with its corresponding clinical visit date which is added to the visitDate' global array. The 'scores' array is used as the Y axis values and the visitDate array is used as the X axis values.

The getBulbarScores(), getMotorScores() and getRespiratoryScores() functions first add up the total value of their corresponding ALSFRS-R scores and then add it to the 'scores' array. For example the getBulbarScores() function will add up the ALSFRS-R scores 1,2 and 3 and then add it to the 'scores' array. The component also receives a 'type' prop from the <ClincalVisits> component so it will have the correct Y axis max value and step size and also calls the correct function to get the scores (see 10. LineChart.js in appendix). The <LineChart> component can be viewed in Figure 3.16: Graphs of total ALSFRS-R and ALSFRS-R sub scores..

4.2.8 Information Popup

The <InfoPopup> component renders the information circle icon that displays the correct information about the ALSFRS-R values when hovered over. It is written in the 'InfoPopup.js' file. The component receives a 'number' prop identifying the ALSFRS-R value. It checks which number is received and returns the correct information in a popup. For example, if the number is 1, it will return the information about ALSFRS-R 1, which is 'speech'. The <InfoPopup> component can be viewed in Figure 3.17: Information about ALSFRS-R questions..

4.2.9 Missing information

The <MissingInfo> component renders the red 'Missing information' button. It is written in the 'MissingInfo.js' file. This component renders a modal which lists the values that are missing from the patient's demographic information if there exists any, otherwise it will display a message stating that 'All information has been recorded'. A disclaimer at the bottom is also noted relating to the values that correspond with death stating that the patient may still be alive (the information is not actually missing). The missing information is obtained from the getMissingInfo() function which is run in the Patient Dashboard page and sent as a prop to the component (see 6.5 getMissingInfo() in appendix). This function checks if each value

is null/undefined and adds it to the ‘missing_info ‘global array if so. The <MissingInfo> component can be viewed in Figure 3.18: Modal of missing information..

4.2.10 Navigation Bar

The <NavBar> component renders the navigation bar which is at the top of each page. It is written in the ‘NavBar.js’ file. It renders the logos at the top and a <Menu> component underneath, which is imported from Semantic UI. When each page is pressed, it becomes the active item on the menu and is highlighted as blue. The component stores the active item in the local storage so that on a refresh, the active item will remain the same for the corresponding page. When the log out button is pressed, a modal will open prompting the user to log out. If the user logs out, the local storage will be cleared and the firebase.auth().signout() API will be called. The user will then be signed out of Firebase and the Login page will be rendered.

5. Evaluation

Throughout this project many aspects of the web application and the communication of the prediction model were discussed in detail. The application was under an iterative design process which proceeded to develop to make sure that the content being displayed was precise. This chapter will discuss the final results, testing and considerations made and the challenges encountered throughout the project.

5.1 Results

With many versions of the web application developed, which were presented to the project team, improvements were constantly being made as requested which led to the final design as seen in 3.7 User interface design. The web application was provided to the team where they were free to use and test it. All of the sections and content displayed has been validated by the PDP manager.

5.2 Testing & considerations

One of the objectives of this project was to connect the Irish ALS/MND register to the application. This was planned to be carried out in the National Centre for Neuroscience in Trinity College Dublin, where the data is kept however with the current situation regarding COVID-19, this objective is no longer a possibility. The Irish ALS/MND register holds roughly 2,553 patients, recorded in November 2018. As it was not possible to test the application with the register, the synthetic data file that was provided by the project team was filled with 2,621 patients using dummy data as seen in Figure 5.1: MND register page listing 2,621 patients. below. Next to the number of patients, a timer was included to display the time it takes to load the patients. The timer begins when the page is loaded and stops when the data is fetched. This test demonstrates that if connected to the Irish ALS/MND register, the application will perform with reasonable load times. As we can see, it takes roughly 2.2 seconds to fetch the data of 2,621 patients.

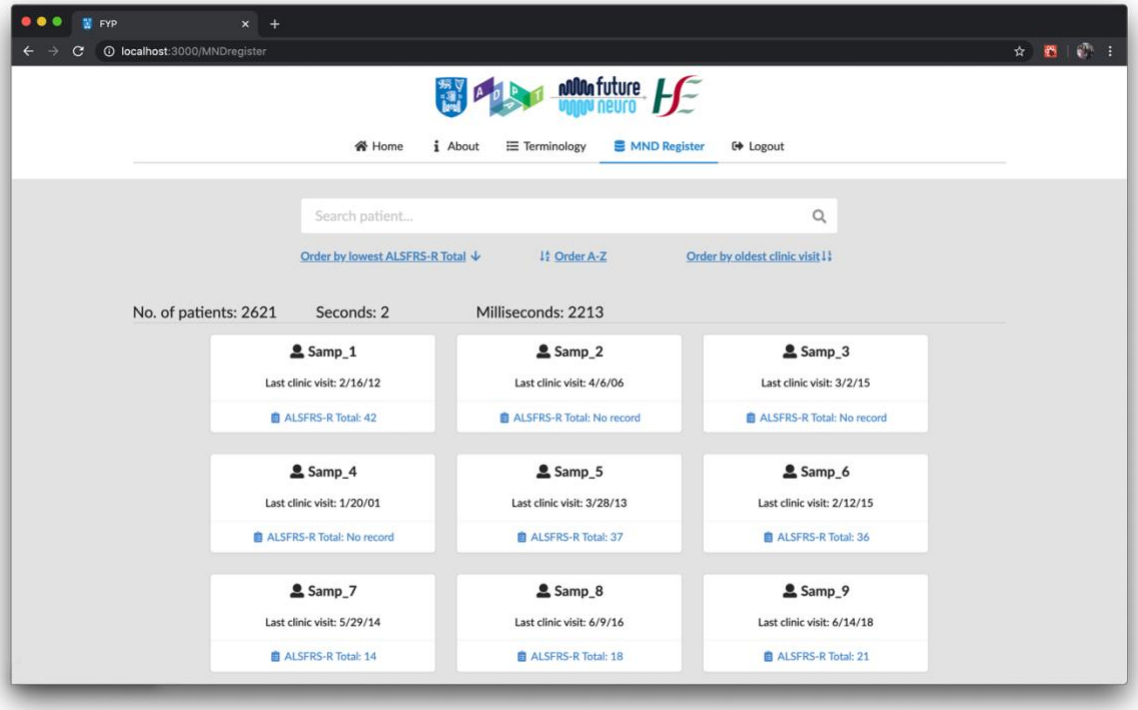

Figure 5.1: MND register page listing 2,621 patients.

A consideration that was taken into account was supressing sensitive information on the patient demographic column in the patient dashboard. This information being, ‘onset to death’, ‘diagnosis to death’ and ‘date of death’. This would be beneficial because if a patient was shown their dashboard, they wouldn’t have to see this sensitive information which may upset them. A solution to implement this would be to add a button at the top right of the box which will trigger a blur effect to turn on or off over this sensitive information. This feature can be implemented if requested however the web application was built by the specifications of the PDP manager and project clinician. This feature can be seen in `Figure 5.2`: Patient dashboard with sensitive information blurred out. below.

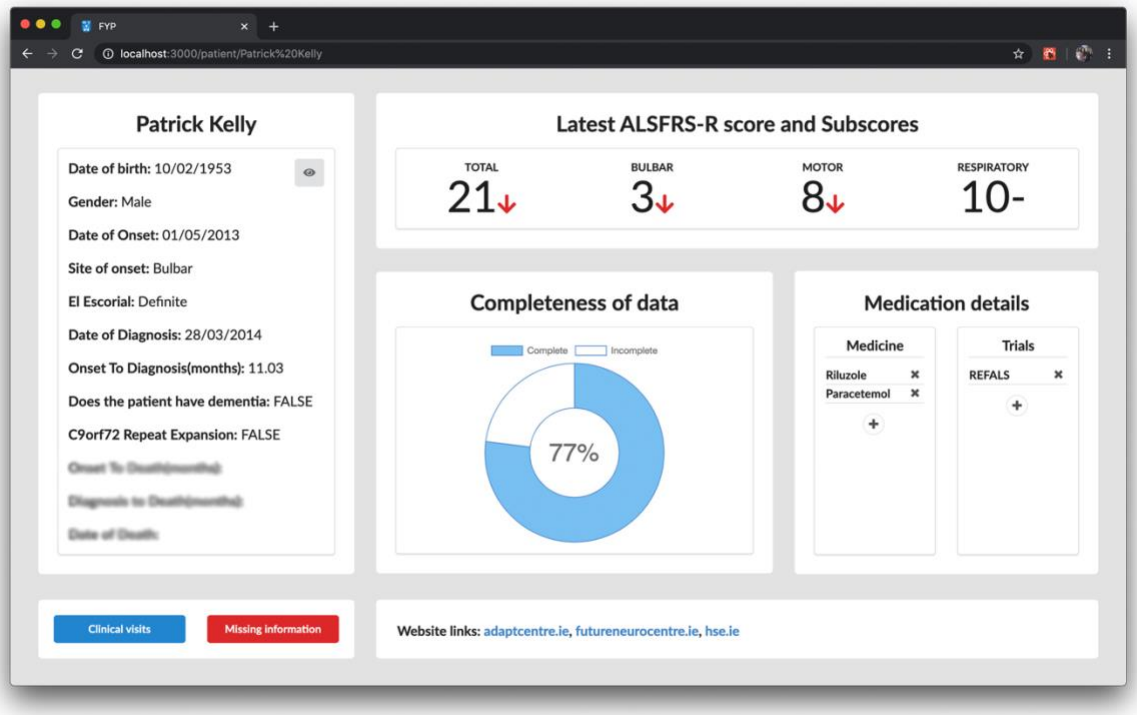

Figure 5.2: Patient dashboard with sensitive information blurred out.

The ALS centres in the Netherlands conducted an online survey, asking 242 Dutch ALS patients if they would like to see their prediction model results.[22] From the results (`Figure 5.3`: Online survey to assess preferences of patients regarding individualized prediction of survival.) of this survey, it is clear to see that more than half of the patients agree. Therefore keeping this consideration into future additions of the web application would be a good idea.

Questions & responses

1. If it were possible to estimate your survival reliably, would you like to know?
   - Yes: 84 (65·6%)
   - No: 25 (19·5%)*
   - I don't know: 19 (14·8%)

2. Would you like to know exactly, for example, expressed in the following terms: you have a chance of dying after 1 year, 2 years, 3 years, etc., or would you prefer to know your survival estimated as a rough duration, e.g. short survival, intermediate survival, long survival?
   - Exactly: 55 (52·9%)
   - Roughly: 40 (38·5%)
   - No preference: 9 (8·7%)

3. When would you like to discuss your predicted life expectancy with your doctor?
   - During the first consultation (diagnosis): 30 (28·8%)
   - During the second consultation: 22 (21·2%)
   - At another appointment: 36 (34·6%)
   - No preference: 16 (15·4%)

4. Would you like to be able to refer to personalized information about your life expectancy yourself, for example by using a website?
   - Yes: 71 (68·9%)
   - No: 32 (31·1%)

Figure 5.3: Online survey to assess preferences of patients regarding individualized prediction of survival.[8]

5.3 Challenges

Throughout this project, many challenges were encountered. Challenges regarding both the healthcare and technology domain. These challenges include:

Understanding the healthcare aspect of the project

Upon beginning this project, many papers about MND, ALSFRS-R and the prediction model were provided. These papers were not easy to read as they contained terminology that was completely new to me from a different study field. However by reading the papers thoroughly and by attending meetings with the project team, my knowledge of the project was quick to grow. Having a good understanding of the healthcare aspect of the project was crucial for communicating the data set needed for the prediction model on the web application. By overcoming this challenge, a new skill has been acquired in terms of interdisciplinary. This new skill can be very important in the future when working on projects, similar to this one, that involve two or more disciplines. It also allowed me to improve as a 'T-shaped' researcher as mentioned before.

Learning the React framework

Learning the React framework was challenging without having any previous experience. This was a big decision to make at the start of the project, one that turned out to become very beneficial. By following tutorials and documentation from online recourses, my understanding of React grew quickly within a couple of weeks. By continuously developing the web application and implementing new features, my knowledge grew even more to the point of being comfortable coding in React. This new skill is very valuable in the web development world and one that can be used in the future.

Implementing Firebase

Firebase, while being very useful in this project, was difficult to implement. It took a considerable amount of time however with trial and error and with sufficient documentation it was achieved. Firebase serves an important part in this project and working with databases, authentication and hosting are also great skills to acquire. Again these skills can be valuable in the workforce.

8 https://www.thelancet.com/journals/laneur/article/PIIS1474-4422(18)30089-9/fulltext

Debugging

No product is ever perfect upon completion, there almost always come with some bugs or errors. During the development of this web application, many bugs were encountered. A lot of these bugs were very challenging to fix and some seemed like they could not be fixed no matter what approach was taken. The routing of the pages and page refreshes were one of the more difficult aspects to get right. However, with extensive debugging and multiple techniques tried, all of the bugs on the web application were resolved.

6. Conclusion & Future work

Motor Neuron Disease is a rare disease that progressively attacks the human body's motor neurons or nerves with no current treatment. This project was created as part of the ADAPT and FutureNeuro MND Patient Data Platform project. The main objective was to develop a web application which displays a visualisation of the data set needed to predict survival time of Patients with Motor Neuron Disease. This report examined an analysis of motor neuron disease, the prediction model and the relevant data set. Thereafter it deconstructed the design and implementation of the web application. The project as a whole was successful with the web application being fully complete as specified by the PDP manager. The only objective that was not possible to complete was to connect the application to the Irish ALS/MND register. An alternative task was put in place to simulate the connection to the register. This chapter discusses the future work and outcomes of the project.

6.1 Future work

All of the objectives for this project that were possible to complete were accomplished. However this project is not finished yet. This section will look at the future work left for the project.

6.1.1 Connection with the Irish ALS/MND Register

As mentioned above, it was not possible to connect the application to the Irish ALS/MND Register, due to the COVID-19 outbreak. The data is kept in the National Centre for Neuroscience in Trinity College Dublin, where this connection would have been conducted. Due to data protection, it is not permitted to use this data outside of Trinity College. This is the next main objective for the project and is a significant one.

6.1.2 Adding functionality

As researchers learn more about motor neuron disease, new information will be studied, scoring systems will be revised and the prediction model may be changed. This web application is built to adapt to these changes which can be implemented quickly. Therefore more functionality can be added if requested by the PDP project team.

6.2 Representing the data set for interoperability

Now that an understanding of the data set required for the prediction model has been achieved a next step could be to represent this data for interoperability. As mentioned in 2.3.5 The Prediction Model (PM), there are 14 European centres involved in creating the prediction model. These centres intend to share and aggregate the data from their various registers. This sharing and aggregation requires robust information models. Some candidate modelling representations and technologies to consider are HL7's Fast Healthcare Interoperability Resource (FHIR)[23], OpenEHR[24] (based on ISO 13606[25]) and Resource Definition Framework (RDF).[26]

6.2 Outcomes

Overall the project was successful and very well received by the users. The web application was created with the purpose of visualising the data set needed to predict survival time of Patients with Motor Neuron Disease. The application has been validated by clinical, register and technical professionals.

This project will also feed into the Core Information Modelling task of the MND PDP project, especially into a masters computer science student's project.

The web application has already been used successfully in a number of project meetings.

Appendix

The appendix presents the important functions used in the components as discussed in

4. Implementation of the report. The full source code can be view at this link:
https://github.com/Arthur9817/FYP-React-Project/tree/master/fyp-react

The web application can be viewed at this link:
https://fyp-mnd-dataset.web.app/

Email: test@gmail.com
Password: test123

Note: Use the email and password above to log into the web application. These credentials will expire in two months.

1. App.js

```
1.  import React, { Component } from "react";
2.  import { Route, Redirect, withRouter } from "react-router-dom";
3.  import firebase from "./Firebase";
4.  import LoginPage from "./pages/LoginPage";
5.  import HomePage from "./pages/HomePage";
6.  import AboutPage from "./pages/AboutPage";
7.  import TerminologyPage from "./pages/TerminologyPage";
8.  import MNDregister from "./pages/MNDregister";
9.  import PatientDashboard from "./pages/PatientDashboard";
10. import NavBar from "./components/NavBar";
11.
12. class App extends Component {
13.     constructor(props) {
14.         super(props);
15.
16.         this.authListener = this.authListener.bind(this);
17.
18.         this.state = {
19.             authUser: null,
20.         };
21.     }
22.
23.     UNSAFE_componentWillMount() {
24.         localStorage.getItem("authUser") &&
25.             this.setState({
26.                 authUser: JSON.parse(localStorage.getItem("authUser")),
27.             });
28.     }
29.
30.     UNSAFE_componentWillUpdate(nextProps, nextState) {
31.         localStorage.setItem("authUser", JSON.stringify(nextState.authUser));
32.     }
33.
34.     componentDidMount() {
35.         this.authListener();
36.     }
```

```
37.
38.     authListener() {
39.         firebase.auth().onAuthStateChanged((authUser) => {
40.             if (authUser) {
41.                 this.setState({ authUser });
42.                 this.timeout = setTimeout(() => {
43.                     this.setState({ authUser: null });
44.                     firebase.auth().signOut();
45.                 }, 1800000);
46.             } else {
47.                 this.setState({ authUser: null });
48.                 localStorage.clear();
49.             }
50.         });
51.     }
52.
53.     render() {
54.         return (
55.             <div>
56.                 {this.state.authUser ? (
57.                     <Route>
58.                         <Redirect
59.                             to={{
60.                                 pathname: this.props.location.pathname,
61.                             }}
62.                         />
63.                         <NavBar />
64.                         <Route path="/" exact component={HomePage} />
65.                         <Route path="/about" component={AboutPage} />
66.                         <Route path="/terminology" component={TerminologyPage} />
67.                         <Route path="/MNDregister" component={MNDregister} />
68.                         <Route path="/patient" component={PatientDashboard} />
69.                     </Route>
70.                 ) : (
71.                     <Route>
72.                         <Redirect
73.                             to={{
74.                                 pathname: "/",
75.                             }}
76.                         />
77.                         <Route path="/" exact component={LoginPage} />
78.                     </Route>
79.                 )}
80.             </div>
81.         );
82.     }
83. }
84.
85. export default withRouter(App);
```

2. index.js

```
1.  import React from "react";
2.  import ReactDOM from "react-dom";
3.  import { BrowserRouter } from "react-router-dom";
4.  import "semantic-ui-css/semantic.min.css";
```

```
import App from "./App";
import * as serviceWorker from "./serviceWorker";

ReactDOM.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>,
    document.getElementById("root")
);

serviceWorker.unregister();
```

3. Firebase.js

```
import firebase from "firebase/app";
import "firebase/auth";
import "firebase/database";
import "firebase/firestore";

// Your web app's Firebase configuration
const firebaseConfig = {
    apiKey: "******",
    authDomain: "******",
    databaseURL: "******",
    projectId: "******",
    storageBucket: "******",
    messagingSenderId: "******",
    appId: "******",
    measurementId: "******",
};

// Initialize Firebase
firebase.initializeApp(firebaseConfig);

export default firebase;
```

4. LoginPage.js

4.1 login(e)

```
async login(e) {
    e.preventDefault();

    try {
        await firebase
            .auth()
            .signInWithEmailAndPassword(this.state.email, this.state.password);
    } catch (error) {
        this.setState({
            message: error.message,
            messageColor: "red",
            hideMessage: false,
        });
    }
}
```

4.2 sendResetEmail(email)

```
async sendResetEmail(email) {
    try {
        await firebase.auth().sendPasswordResetEmail(email);
        this.setState({
            message:
                "Password reset email sent. Please follow the instruction in the email to proceed.",
            messageColor: "green",
            hideMessage: false,
            openModal: false,
            hideResetMessage: true,
            passwordReset: "",
        });
        this.timeout = setTimeout(() => {
            this.setState({ hideMessage: true });
        }, 5000);
    } catch (error) {
        this.setState({
            message2: error.message,
            messageColor: "red",
            hideResetMessage: false,
        });
    }
}
```

5. MNDregister.js

5.1 componentDidMount()

```
componentDidMount() {
    firebase
        .database()
        .ref("clinical_data")
        .on("value", (snapshot) => {
            let tmp = [];
            snapshot.forEach((snap) => {
                tmp.push(snap.val());
            });
            this.setState({ patients: tmp });
        });

    firebase
        .database()
        .ref("Clinical Visit Data")
        .on("value", (snapshot) => {
            let clinic_data = [];
            snapshot.forEach((snap) => {
                clinic_data.push(snap.val());
            });
            this.setState({ clinic_visit: clinic_data, isLoading: false });
        });
}
```

5.2 filteredPatients

```
let filteredPatients = this.state.patients.filter((patient) => {
    return (
        patient["Sample ID"]
            .toLowerCase()
            .indexOf(this.state.search.toLowerCase()) !== -1
    );
});
```

5.3 getClinicalInfo(patient, type)

```
getClinicalInfo(patient, type) {
    let ALSFRSR = [];
    let date = [];
    for (let i = 0; i < this.state.clinic_visit.length; i++) {
        if (this.state.clinic_visit[i].undefined === patient["Sample ID"]) {
            date.push(this.state.clinic_visit[i]["Clinical visit date"]);
            if (this.state.clinic_visit[i]["Total"] !== undefined) {
                ALSFRSR.push(this.state.clinic_visit[i]["Total"]);
            }
        }
    }

    if (type === "date") {
        date.sort(function (a, b) {
```

```
            var aa = a.split("/").reverse().join(),
                bb = b.split("/").reverse().join();
            return aa > bb ? -1 : aa < bb ? 1 : 0;
        });
        return date[0];
    } else {
        if (ALSFRSR.length > 0) {
            return Math.min.apply(Math, ALSFRSR);
        } else {
            return "No record";
        }
    }
}
```

5.4 sortBy(type, sortFrom)

```
sortBy(type, sortFrom) {
    if (type === "alphabetical") {
        if (sortFrom === "start") {
            this.setState({ alphabetStart: false });
            this.state.patients.sort(function (a, b) {
                var splitA = a["Sample ID"].split(" ");
                var splitB = b["Sample ID"].split(" ");
                var lastA = splitA[splitA.length - 1];
                var lastB = splitB[splitB.length - 1];

                if (lastA < lastB) return -1;
                if (lastA > lastB) return 1;
                return 0;
            });
        } else {
            this.setState({ alphabetStart: true });
            this.state.patients.sort(function (a, b) {
                var splitA = a["Sample ID"].split(" ");
                var splitB = b["Sample ID"].split(" ");
                var lastA = splitA[splitA.length - 1];
                var lastB = splitB[splitB.length - 1];

                if (lastA > lastB) return -1;
                if (lastA < lastB) return 1;
                return 0;
            });
        }
    } else {
        for (let j = 0; j < this.state.patients.length; j++) {
            for (let i = 0; i < this.state.clinic_visit.length; i++) {
                if (
                    this.state.clinic_visit[i].undefined ===
                    this.state.patients[j]["Sample ID"]
                ) {
                    if (this.state.clinic_visit[i]["Total"] !== undefined) {
                        this.state.patients[j]["Total"] = parseInt(
                            this.state.clinic_visit[i]["Total"]
                        );
                    }
                    this.state.patients[j]["Visit"] = this.state.clinic_visit[i][
```

```
                        "Clinical visit date"
                    ];
                }
            }
        }

        if (type === "visit") {
            if (sortFrom === "oldest") {
                this.setState({ lastVisit: false });
                this.state.patients.sort(function (a, b) {
                    var aa = a["Visit"].split("/").reverse().join(),
                        bb = b["Visit"].split("/").reverse().join();
                    return aa < bb ? -1 : aa > bb ? 1 : 0;
                });
            } else {
                this.setState({ lastVisit: true });

                this.state.patients.sort(function (a, b) {
                    var aa = a["Visit"].split("/").reverse().join(),
                        bb = b["Visit"].split("/").reverse().join();
                    return aa > bb ? -1 : aa < bb ? 1 : 0;
                });
            }
        } else if (type === "ALSFRS-R") {
            let tmp1 = [];
            let tmp2 = [];

            for (var i = 0; i < this.state.patients.length; i++) {
                if (this.state.patients[i]["Total"] === undefined) {
                    tmp1.push(this.state.patients[i]);
                } else {
                    tmp2.push(this.state.patients[i]);
                }
            }

            if (sortFrom === "lowest") {
                this.setState({ isLowest: false });
                tmp2.sort((a, b) => a.Total - b.Total);
            } else {
                this.setState({ isLowest: true });
                tmp2.sort((a, b) => b.Total - a.Total);
            }

            tmp2 = tmp2.concat(tmp1);

            this.setState({ patients: tmp2 });
        }
        this.state.patients.forEach(function (v) {
            delete v.Visit;
            delete v.Total;
        });
    }
}
```

6. PatientDashboard.js

6.1 Structure of dashboard

```
<Container fluid>
    <Container style={{ width: "97%" }}>
        <Grid centered columns={2} style={{ margin: 0 }}>
            <Grid.Row>
                <Grid.Column width={5} style={{ minWidth: 350 }}>
                    <DemographicInfo data={this.state.patient_data} />
                </Grid.Column>

                <Grid.Column width={11} style={{ minWidth: 350 }}>
                    <Grid.Column>
                        <ALSFRSRscores data={this.state.single_clinic_data} />
                    </Grid.Column>

                    <Grid columns={2}>
                        <Grid.Column width={9}>
                            <Completeness data={this.state.completness} />
                        </Grid.Column>

                        <Grid.Column width={7} style={{ minWidth: 350 }}>
                            <Medication data={this.state.patient_data} />
                        </Grid.Column>
                    </Grid>
                </Grid.Column>
            </Grid.Row>
        </Grid>
    </Container>

    <Container style={{ width: "97%" }} fluid>
        <Grid centered columns={2} style={{ margin: 0 }}>
            <Grid.Column width={5} style={{ minWidth: 350 }}>
                <Container className="button-box">
                    <Grid columns={2}>
                        <Grid.Column>
                            <ClinicalVisits
                                id={this.state.patient_data["Sample ID"]}
                                data={this.state.single_clinic_data}
                            />
                        </Grid.Column>

                        <Grid.Column>
                            <MissingInfo
                                id={this.state.patient_data["Sample ID"]}
                                data={this.state.missing_info}
                                state={this.state.isMissing}
                            />
                        </Grid.Column>
                    </Grid>
                </Container>
            </Grid.Column>
            <Grid.Column width={11} style={{ minWidth: 350 }}>
                <Container className="reference-box">
                    <h3>References*</h3>
```

```
                </Container>
            </Grid.Column>
        </Grid>
    </Container>
</Container>
```

6.2 Data check

```
patient_data: (this.state = JSON.parse(localStorage.getItem("patient"))
    ? JSON.parse(localStorage.getItem("patient"))
    : props.location.data.user),
clinic_visit_data: (this.state = JSON.parse(
    localStorage.getItem("clinic"))
    ? JSON.parse(localStorage.getItem("clinic"))
    : props.location.data.clinic)
```

6.3 orderByLatestDate()

```
orderByLatestDate() {
    for (let i = 0; i < this.state.clinic_visit_data.length; i++) {
        if (
            this.state.clinic_visit_data[i].undefined ===
            this.state.patient_data["Sample ID"]
        ) {
            this.state.single_clinic_data.push(this.state.clinic_visit_data[i]);
        }
    }
    this.state.single_clinic_data.sort(function (a, b) {
        var aa = a["Clinical visit date"].split("/").reverse().join(),
            bb = b["Clinical visit date"].split("/").reverse().join();
        return aa > bb ? -1 : aa < bb ? 1 : 0;
    });
}
```

6.4 getCompleteness()

```
getCompleteness() {
    this.state.completness = Number(
        ((Object.entries(this.state.patient_data).length -
            this.state.missing_info.length) /
            Object.entries(this.state.patient_data).length) *
            100
    ).toFixed(1);
}
```

6.5 getMissingInfo()

```
async getMissingInfo() {
    await Object.entries(this.state.patient_data).forEach((element) => {
        if (element[1] === null || element[1] === "") {
            this.state.missing_info.push(element[0]);
        }
    });
```

```
7.
8.      if (this.state.missing_info && this.state.missing_info.length > 0) {
9.          this.setState({ isMissing: true });
10.     }
11. }
```

7. ALSFRSRscores.js

7.1 getTotalScore()

```
1.  getTotalScore() {
2.      let tmp = [];
3.      for (let i = 0; i < this.state.clinic_data.length; i++) {
4.          if (this.state.clinic_data[i]["Total"] !== undefined) {
5.              tmp.push(this.state.clinic_data[i]["Total"]);
6.          }
7.      }
8.
9.      if (tmp.length > 0) {
10.         if (tmp[0] < tmp[1]) {
11.             return (
12.                 <div>
13.                     {tmp[0]}
14.                     <Icon size="tiny" name="arrow down" color="red" />
15.                 </div>
16.             );
17.         } else if (tmp[0] > tmp[1]) {
18.             return (
19.                 <div>
20.                     {tmp[0]}
21.                     <Icon size="tiny" name="arrow up" color="green" />
22.                 </div>
23.             );
24.         } else if (tmp[0] === tmp[1]) {
25.             return <div>{tmp[0]}-</div>;
26.         } else return tmp[0];
27.     } else {
28.         return "-";
29.     }
30. }
```

7.2 getBulbarScore()

```
1.  getBulbarScore() {
2.      let latest = [];
3.      for (let i = 0; i < this.state.clinic_data.length; i++) {
4.          let tmp = [];
5.          if (this.state.clinic_data[i]["Total"] === undefined) {
6.          } else {
7.              if (this.state.clinic_data[i]["ALSFRS 1 Speech"] !== undefined) {
8.                  tmp.push(this.state.clinic_data[i]["ALSFRS 1 Speech"]);
9.              }
10.             if (this.state.clinic_data[i]["ALSFRS 2 Salivation"] !== undefined) {
```

```
                tmp.push(this.state.clinic_data[i]["ALSFRS 2 Salivation"]);
            }
            if (this.state.clinic_data[i]["ALSFRS 3 Swallowing"] !== undefined) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 3 Swallowing"]);
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            latest.push(total);
        }
    }

    if (latest.length > 0) {
        if (latest[0] < latest[1]) {
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow down" color="red" />
                </div>
            );
        } else if (latest[0] > latest[1]) {
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow up" color="green" />
                </div>
            );
        } else if (latest[0] === latest[1]) {
            return <div>{latest[0]}-</div>;
        } else return latest[0];
    } else {
        return "-";
    }
}
```

7.3 getMotorScore()

```
getMotorScore() {
    let latest = [];
    for (let i = 0; i < this.state.clinic_data.length; i++) {
        let tmp = [];
        if (this.state.clinic_data[i]["Total"] === undefined) {
        } else {
            if (this.state.clinic_data[i]["ALSFRS 4 Handwriting"] !== undefined) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 4 Handwriting"]);
            }
            if (
                this.state.clinic_data[i][
                    "ALSFRS 5a Utensils Handling No Gastrostomy"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.clinic_data[i][
```

```
                        "ALSFRS 5a Utensils Handling No Gastrostomy"
                    ]
                );
            }
            if (
                this.state.clinic_data[i][
                    "ALSFRS 5b Utensils Handling Gastrostomy"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.clinic_data[i]["ALSFRS 5b Utensils Handling Gastrostomy"
]
                );
            }
            if (
                this.state.clinic_data[i]["ALSFRS 6 Dressing and hygiene"] !==
                undefined
            ) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 6 Dressing and hygiene"]);
            }
            if (
                this.state.clinic_data[i][
                    "ALSFRS 7 Turning in bed and adjusting bed clothes"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.clinic_data[i][
                        "ALSFRS 7 Turning in bed and adjusting bed clothes"
                    ]
                );
            }
            if (this.state.clinic_data[i]["ALSFRS 8 Walking"] !== undefined) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 8 Walking"]);
            }
            if (
                this.state.clinic_data[i]["ALSFRS 9 Climbing stairs"] !== undefined
            ) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 9 Climbing stairs"]);
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            latest.push(total);
        }
    }

    if (latest.length > 0) {
        if (latest[0] < latest[1]) {
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow down" color="red" />
                </div>
            );
```

```
        } else if (latest[0] > latest[1]) {
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow up" color="green" />
                </div>
            );
        } else if (latest[0] === latest[1]) {
            return <div>{latest[0]}-</div>;
        } else return latest[0];
    } else {
        return "-";
    }
}
```

7.4 getRespiratoryScore()

```
getRespiratoryScore() {
    let latest = [];
    for (let i = 0; i < this.state.clinic_data.length; i++) {
        let tmp = [];
        if (this.state.clinic_data[i]["Total"] === undefined) {
        } else {
            if (this.state.clinic_data[i]["ALSFRS 10 Dyspnea"] !== undefined) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 10 Dyspnea"]);
            }
            if (this.state.clinic_data[i]["ALSFRS 11 Orthopnea"] !== undefined) {
                tmp.push(this.state.clinic_data[i]["ALSFRS 11 Orthopnea"]);
            }
            if (
                this.state.clinic_data[i]["ALSFRS 12 Respiratory insufficiency"] !==
                undefined
            ) {
                tmp.push(
                    this.state.clinic_data[i]["ALSFRS 12 Respiratory insufficiency"]
                );
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            latest.push(total);
        }
    }

    if (latest.length > 0) {
        if (latest[0] < latest[1]) {
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow down" color="red" />
                </div>
            );
        } else if (latest[0] > latest[1]) {
```

```
            return (
                <div>
                    {latest[0]}
                    <Icon size="tiny" name="arrow up" color="green" />
                </div>
            );
        } else if (latest[0] === latest[1]) {
            return <div>{latest[0]}-</div>;
        } else return latest[0];
    } else {
        return "-";
    }
}
```

8. Medication.js

8.1 getItems(type)

```
getItems(type) {
    let tmp = [];

    if (type === "medicine") {
        firebase
            .firestore()
            .collection("medication")
            .doc(this.state.patient_data["Sample ID"])
            .get()
            .then((doc) => {
                if (doc.exists) {
                    if (
                        doc.data().medicine !== null ||
                        doc.data().medicine !== undefined
                    ) {
                        tmp.push(doc.data().medicine);

                        this.setState({ medicine_list: tmp });
                    } else {
                        tmp.push([]);
                        this.setState({ medicine_list: tmp });
                    }
                } else {
                    tmp.push([]);
                    this.setState({ medicine_list: tmp });
                }
            });
    } else if (type === "trial") {
        firebase
            .firestore()
            .collection("medication")
            .doc(this.state.patient_data["Sample ID"])
            .get()
            .then((doc) => {
                if (doc.exists) {
                    if (doc.data().trial !== null || doc.data().trial !== undefined) {
```

```
                        tmp.push(doc.data().trial);
                        this.setState({ trial_list: tmp, isLoading: false });
                    } else {
                        tmp.push([]);
                        this.setState({ trial_list: tmp, isLoading: false });
                    }
                } else {
                    tmp.push([]);
                    this.setState({ trial_list: tmp, isLoading: false });
                }
            });
    }
}
```

8.2 displayItems(type)

```
displayItems(type) {
    var list = [];
    var itemType = "";

    if (type === "medicine") {
        list = this.state.medicine_list[0];
        itemType = "medicine";
    } else if (type === "trial") {
        list = this.state.trial_list[0];
        itemType = "trial";
    }

    if (list === undefined) {
        return;
    } else {
        return list.map((item, index) => (
            <Container style={{ textAlign: "left" }} key={index}>
                <List divided>
                    <List.Item>
                        <List.Content>
                            <b style={{ fontSize: 15 }}>{item}</b>

                            <Popup
                                inverted
                                trigger={
                                    <Icon
                                        style={{ float: "right" }}
                                        link
                                        name="x"
                                        onClick={() => {
                                            this.deleteItem(item, itemType);
                                        }}
                                    />
                                }
                                position="bottom right"
                            >
                                Are you sure you want to delete the {itemType}: '{item}
'?
                            </Popup>
                        </List.Content>
```

```
                    </List.Item>
                    <List.Item />
                </List>
            </Container>
        ));
    }
}
```

8.3 addItem(e, inputType)

```
addItem(e, inputType) {
    e.preventDefault();

    if (inputType === "medicine") {
        const newItem = this.state.medicine_input;
        this.state.medicine_list[0].push(newItem);

        var tmp = [];
        this.state.medicine_list[0].forEach((medicine) => {
            tmp.push(medicine);
        });

        if (newItem !== "") {
            firebase
                .firestore()
                .collection("medication")
                .doc(this.state.patient_data["Sample ID"])
                .set(
                    {
                        medicine: tmp,
                    },
                    { merge: true }
                );
        }
    } else if (inputType === "trial") {
        const newItem = this.state.trial_input;
        this.state.trial_list[0].push(newItem);

        var tmp2 = [];
        this.state.trial_list[0].forEach((trial) => {
            tmp2.push(trial);
        });

        if (newItem !== "") {
            firebase
                .firestore()
                .collection("medication")
                .doc(this.state.patient_data["Sample ID"])
                .set(
                    {
                        trial: tmp2,
                    },
                    { merge: true }
                );
        }
    }
```

```
47.
48.     this.setState({
49.         toggleMedicineInput: false,
50.         toggleTrialInput: false,
51.         trial_input: "",
52.         medicine_input: "",
53.     });
54. }
```

8.4 deleteItem(type)

```
1.  async deleteItem(item, inputType) {
2.      let tmp = [];
3.      if (inputType === "medicine") {
4.          this.state.medicine_list[0].forEach((medicine) => {
5.              if (medicine !== item) {
6.                  tmp.push(medicine);
7.              }
8.          });
9.          await firebase
10.             .firestore()
11.             .collection("medication")
12.             .doc(this.state.patient_data["Sample ID"])
13.             .set(
14.                 {
15.                     medicine: tmp,
16.                 },
17.                 { merge: true }
18.             );
19.         this.getItems("medicine");
20.     } else if (inputType === "trial") {
21.         this.state.trial_list[0].forEach((trial) => {
22.             if (trial !== item) {
23.                 tmp.push(trial);
24.             }
25.         });
26.         firebase
27.             .firestore()
28.             .collection("medication")
29.             .doc(this.state.patient_data["Sample ID"])
30.             .set(
31.                 {
32.                     trial: tmp,
33.                 },
34.                 { merge: true }
35.             );
36.         this.getItems("trial");
37.     }
38.     this.setState({ openConfirm: false });
39. }
```

9. Table.js

9.1 getUndefined()

```
getUndefined() {
    if (this.props.data[0].Total === undefined) {
        for (let i = 0; i < this.props.data.length; i++) {
            if (this.props.data[i].Total === undefined) {
                this.state.undef.push(this.props.data[i]);
            } else {
                break;
            }
        }
    }
}
```

9.2 getData()

```
getData() {
    this.state.data = [];

    let count = 0;
    if (this.props.data[0].Total === undefined) {
        for (let i = 0; i < this.props.data.length; i++) {
            if (this.props.data[i].Total === undefined) {
                count++;
            } else {
                for (let i = 0; i < this.props.data.length - count; i++) {
                    this.state.data.push(this.props.data[i + count]);
                }
                break;
            }
        }
    } else {
        for (let i = 0; i < this.props.data.length; i++) {
            this.state.data.push(this.props.data[i]);
        }
    }
}
```

9.3 printUndefined()

```
printUndefined() {
    this.getUndefined();
    return this.state.undef.map((visit, index) => (
        <Table.Row
            style={{
                fontSize: 16,
                backgroundColor: "#f2f2f2",
                fontWeight: "bold",
            }}
            key={index}
        >
            <Table.Cell>
```

```
                {this.check_beumont_visit(visit["Beaumont Hospital visit"])}
            </Table.Cell>
            <Table.Cell>{visit["Clinical visit date"]}</Table.Cell>
            <Table.Cell>{visit["SNIP(cm H2O)_Occluded Method"]}</Table.Cell>
            <Table.Cell>{visit["Total"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 1 Speech"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 2 Salivation"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 3 Swallowing"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 4 Handwriting"]}</Table.Cell>
            <Table.Cell>
                {visit["ALSFRS 5a Utensils Handling No Gastrostomy"]}
            </Table.Cell>
            <Table.Cell>
                {visit["ALSFRS 5b Utensils Handling Gastrostomy"]}
            </Table.Cell>
            <Table.Cell>{visit["ALSFRS 6 Dressing and hygiene"]}</Table.Cell>
            <Table.Cell>
                {visit["ALSFRS 7 Turning in bed and adjusting bed clothes"]}
            </Table.Cell>
            <Table.Cell>{visit["ALSFRS 8 Walking"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 9 Climbing stairs"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 10 Dyspnea"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 11 Orthopnea"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 12 Respiratory insufficiency"]}</Table.Cell>
        </Table.Row>
    ));
}
```

9.4 printData()

```
printData() {
    this.getData();
    return this.state.data.map((visit, index) => (
        <Table.Row
            style={{
                fontSize: 16,
                backgroundColor: "#f2f2f2",
                fontWeight: "bold",
                borderTopColor: "red !important",
            }}
            key={index}
        >
            <Table.Cell style={{ borderTop: "1 !important" }}>
                {this.check_beumont_visit(visit["Beaumont Hospital visit"])}
            </Table.Cell>
            <Table.Cell>{visit["Clinical visit date"]}</Table.Cell>
            <Table.Cell>{visit["SNIP(cm H2O)_Occluded Method"]}</Table.Cell>
            <Table.Cell>{this.check_total(visit["Total"])}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 1 Speech"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 2 Salivation"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 3 Swallowing"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 4 Handwriting"]}</Table.Cell>
            <Table.Cell>
                {visit["ALSFRS 5a Utensils Handling No Gastrostomy"]}
            </Table.Cell>
            <Table.Cell>
```

```
                {visit["ALSFRS 5b Utensils Handling Gastrostomy"]}
            </Table.Cell>
            <Table.Cell>{visit["ALSFRS 6 Dressing and hygiene"]}</Table.Cell>
            <Table.Cell>
                {visit["ALSFRS 7 Turning in bed and adjusting bed clothes"]}
            </Table.Cell>
            <Table.Cell>{visit["ALSFRS 8 Walking"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 9 Climbing stairs"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 10 Dyspnea"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 11 Orthopnea"]}</Table.Cell>
            <Table.Cell>{visit["ALSFRS 12 Respiratory insufficiency"]}</Table.Cell>
        </Table.Row>
    ));
}
```

9.5 checkTotal()

```
 check_total(total) {
    if (total === undefined) {
        return;
    }
    if (this.state.index === this.state.data.length) {
        return total;
    }

    if (this.state.index < this.state.data.length) {
        this.state.prev = this.state.data[this.state.index].Total;
    }

    if (this.state.prev === undefined) {
        this.checkIfPrevUndefined();
    }

    if (total === this.state.prev) {
        this.state.index = this.state.index + 1;
        return (
            <div>
                {total} <Icon name="minus" />
            </div>
        );
    } else if (total < this.state.prev) {
        this.state.index = this.state.index + 1;

        return (
            <div>
                {total}
                <Icon name="arrow down" color="red" />
            </div>
        );
    } else if (total > this.state.prev) {
        this.state.index = this.state.index + 1;

        return (
            <div>
                {total}
                <Icon name="arrow up" color="green" />
```

```
            </div>
        );
    } else {
        return total;
    }
}
```

9.6 checkIfNextUndefined()

```
checkIfPrevUndefined() {
    if (this.state.prev === undefined) {
        if (this.state.index === this.state.data.length - 1) {
            return;
        }
        this.state.index = this.state.index + 1;
        this.state.prev = this.state.data[this.state.index].Total;
        this.checkIfPrevUndefined();
    } else {
        return;
    }
}
```

10. LineChart.js

```
class LineChart extends Component {
    constructor(props) {
        super(props);

        this.state = {
            visit_data: this.props.data,
            scores: [],
            visitDate: [],
            totalMax: null,
            totalStepSize: null,
        };

        if (this.props.type === "total") {
            this.state.totalMax = 48;
            this.state.totalStepSize = 12;
            this.getTotalScores();
        } else if (this.props.type === "motor") {
            this.state.totalMax = 24;
            this.state.totalStepSize = 6;
            this.getMotorScores();
        } else if (this.props.type === "bulbar") {
            this.state.totalMax = 12;
            this.state.totalStepSize = 3;
            this.getBulbarScores();
        } else if (this.props.type === "respiratory") {
            this.state.totalMax = 12;
            this.state.totalStepSize = 3;
            this.getRespiratoryScores();
        }
    }
```

10.1 getTotalScores()

```
getTotalScores() {
    for (let i = 0; i < this.state.visit_data.length; i++) {
        if (this.state.visit_data[i].Total !== undefined) {
            this.state.scores.push(this.state.visit_data[i].Total);
            this.state.visitDate.push(
                this.state.visit_data[i]["Clinical visit date"]
            );
        }
    }
}
```

10.2 getBulbarScores()

```
getBulbarScores() {
    for (let i = 0; i < this.state.visit_data.length; i++) {
        let tmp = [];
        if (this.state.visit_data[i]["Total"] === undefined) {
        } else {
            if (this.state.visit_data[i]["ALSFRS 1 Speech"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 1 Speech"]);
            }
            if (this.state.visit_data[i]["ALSFRS 2 Salivation"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 2 Salivation"]);
            }
            if (this.state.visit_data[i]["ALSFRS 3 Swallowing"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 3 Swallowing"]);
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            this.state.scores.push(total);

            this.state.visitDate.push(
                this.state.visit_data[i]["Clinical visit date"]
            );
        }
    }
}
```

10.3 getMotorScores()

```
getMotorScores() {
    for (let i = 0; i < this.state.visit_data.length; i++) {
        let tmp = [];
        if (this.state.visit_data[i]["Total"] === undefined) {
        } else {
            if (this.state.visit_data[i]["ALSFRS 4 Handwriting"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 4 Handwriting"]);
            }
```

```
            if (
                this.state.visit_data[i][
                    "ALSFRS 5a Utensils Handling No Gastrostomy"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.visit_data[i][
                        "ALSFRS 5a Utensils Handling No Gastrostomy"
                    ]
                );
            }
            if (
                this.state.visit_data[i][
                    "ALSFRS 5b Utensils Handling Gastrostomy"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.visit_data[i]["ALSFRS 5b Utensils Handling Gastrostomy"]

                );
            }
            if (
                this.state.visit_data[i]["ALSFRS 6 Dressing and hygiene"] !==
                undefined
            ) {
                tmp.push(this.state.visit_data[i]["ALSFRS 6 Dressing and hygiene"]);
            }
            if (
                this.state.visit_data[i][
                    "ALSFRS 7 Turning in bed and adjusting bed clothes"
                ] !== undefined
            ) {
                tmp.push(
                    this.state.visit_data[i][
                        "ALSFRS 7 Turning in bed and adjusting bed clothes"
                    ]
                );
            }
            if (this.state.visit_data[i]["ALSFRS 8 Walking"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 8 Walking"]);
            }
            if (
                this.state.visit_data[i]["ALSFRS 9 Climbing stairs"] !== undefined
            ) {
                tmp.push(this.state.visit_data[i]["ALSFRS 9 Climbing stairs"]);
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            this.state.scores.push(total);

            this.state.visitDate.push(
                this.state.visit_data[i]["Clinical visit date"]
            );
```

```
        }
    }
}
```

10.4 getRespiratoryScores()

```
getRespiratoryScores() {
    for (let i = 0; i < this.state.visit_data.length; i++) {
        let tmp = [];
        if (this.state.visit_data[i]["Total"] === undefined) {
        } else {
            if (this.state.visit_data[i]["ALSFRS 10 Dyspnea"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 10 Dyspnea"]);
            }
            if (this.state.visit_data[i]["ALSFRS 11 Orthopnea"] !== undefined) {
                tmp.push(this.state.visit_data[i]["ALSFRS 11 Orthopnea"]);
            }
            if (
                this.state.visit_data[i]["ALSFRS 12 Respiratory insufficiency"] !==
                undefined
            ) {
                tmp.push(
                    this.state.visit_data[i]["ALSFRS 12 Respiratory insufficiency"]
                );
            }

            let total = 0;
            for (let i = 0; i < tmp.length; i++) {
                total = total + parseInt(tmp[i]);
            }

            this.state.scores.push(total);

            this.state.visitDate.push(
                this.state.visit_data[i]["Clinical visit date"]
            );
        }
    }
}
```

References

Citation style: APA 6th Edition

[1] Criteria For the Diagnosis of ALS. (n.d.). Retrieved from http://www.alsa.org/assets/pdfs/fyi/criteria_for_diagnosis.pdf

[2] Franchignoni, F., Mandrioli, J., Giordano, A., Ferro, S., & ERRALS Group. (2015, April 27). A further Rasch study confirms that ALSFRS-R does not conform to fundamental measurement requirements. Retrieved from https://www.tandfonline.com/doi/full/10.3109/21678421.2015.1026829

[3] Ludolph, A., Drory, V., Hardiman, O., Nakano, I., Ravits, J., Robberecht, W., & Shefner, J. (2015, June 29). A revision of the El Escorial criteria - 2015. Retrieved April 26, 2020, from http://www.wfnals.org/downloads/A revision of the El Escorial criteria 2015.pdf

[4] Cedarbaum, J. M., Stambler, N., Malta, E., Fuller, C., Hilt, D., Thurmond, B., & Nakanishi, A. (1999, October 31). The ALSFRS-R: a revised ALS functional rating scale that incorporates assessments of respiratory function. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0022510X99002105

[5] React (web framework). (2020, April 9). Retrieved from https://en.wikipedia.org/wiki/React_(web_framework)

[6] Martin, S. (2019, November 8). The Top JavaScript Frameworks For Front-End Development in 2020. Retrieved from https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

[7] Semantic UI. (n.d.). Retrieved from https://semantic-ui.com/

[8] Brown, R., Deletic, A., & Wong, T. (2015, September 16). Interdisciplinarity: How to catalyse collaboration. Retrieved from https://www.nature.com/news/interdisciplinarity-how-to-catalyse-collaboration-1.18343

[9] What is a rare disease? (2007, April 15). Retrieved from https://www.eurordis.org/sites/default/files/publications/Fact_Sheet_RD.pdf

[10] Daly, A., Dunne, C., Clerkin, C., Dennedy, M. C., Lambert, D., Little, M., … Wall, D. (2019, July). Model of Care for Rare Diseases. Retrieved from https://www.hse.ie/eng/about/who/cspd/ncps/rare-diseases/resources/model-of-care-for-rare-diseases.pdf

[11] Stephens, G., Impey, S., & Berry, D. (2019, October 1). Motor Neuron Disease Research Patient Data Platform Background Document for Sprint Zero with ADAPT D-Lab. Retrieved from project team.

[12] Facts about MND. (n.d.). Retrieved from https://imnda.ie/about-mnd/more-facts-about-motor-neurone-disease/

[13] Riluzole. (2020, April 15). Retrieved from https://en.wikipedia.org/wiki/Riluzole

[14] Westeneng, H.-J., Debray, T. P. A., Visser, A. E., van Eijk, R. P. A., Rooney, J. P. K., Calvo, A., … van den Berg, L. H. (2018, March 26). Prognosis for patients with amyotrophic lateral sclerosis: development and validation of a personalised prediction model. Retrieved from https://www.thelancet.com/journals/laneur/article/PIIS1474-4422(18)30089-9/fulltext

[15] Leader, D. (2020, January 29). What Is Forced Vital Capacity (FVC)? Retrieved from https://www.verywellhealth.com/forced-expiratory-capacity-measurement-914900

[16] Al-Chalabi, A. (2015, June). ALSFRS-R Training. Retrieved from https://www.encals.eu/wp-content/uploads/2016/09/ALSFRS-SOP-ENCALS-presentation.pdf

[17] Rooney, J., Burke, T., Vajda, A., Heverin, M., & Hardiman, O. (2017, May 4). What does the ALSFRS-R really measure? A longitudinal and survival analysis of functional dimension subscores in amyotrophic lateral sclerosis. Retrieved from https://jnnp.bmj.com/content/88/5/381

[18] Doherty, G. (2019). CS4051 Human Factors - User Interface Architectures. Retrieved from Trinity College Dublin.

[19] Liyanage, E. (2016, October 2). 10 Usability heuristics explained. Retrieved from https://medium.com/@erangatl/10-usability-heuristics-explained-caa5903faba2

[20] Ziroll, B. (2019, November 11). Learn React Router in 5 Minutes - A Beginner's Tutorial. Retrieved from https://www.freecodecamp.org/news/react-router-in-5-minutes/

[21] Sharma, A. (2018, September 29). Realtime Database vs. Cloud Firestore - Which Database is Suitable for your Mobile App. Retrieved from https://medium.com/datadriveninvestor/realtime-database-vs-cloud-firestore-which-database-is-suitable-for-your-mobile-app-87e11b56f50f

[22] Westeneng, H.-J., Debray, T. P. A., Visser, A. E., van Eijk, R. P. A., Rooney, J. P. K., Calvo, A., … van den Berg, L. H. (2018, March 26). Supplementary appendix. Retrieved from https://www.thelancet.com/journals/laneur/article/PIIS1474-4422(18)30089-9/fulltext

[23] Health Level Seven International. (n.d.). Retrieved from https://www.hl7.org/

[24] openEHR Clinical Knowledge Manager. (n.d.). Retrieved from https://www.openehr.org/ckm/

[25] ISO 13606-1:2019. (2019, June 7). Retrieved from https://www.iso.org/standard/67868.html

[26] RDF. (2014, February 25). Retrieved from https://www.w3.org/RDF/