Feeding Dashboard

# Aims and Objectives

The main aim of this project is to create an application, with the primary goal of identifying which patients need referring to a dietitian. The Critical Care Unit (CCU) is a branch of the hospital which is underfunded: prioritising those most in need is key. Using Electron as a base framework, combined with Next.js and React.js for the rendering, it is possible to create an application which will run efficiently on all operating systems.

The critical path, detailed in the Project Planning phase, will be continually monitored to prevent any potential delays. This involves identifying key milestones in the project. Said milestones will be used as an easy way to measure how much of the project has been completed: ensuring the project will be completed within the planned timeframes. A risk assessment will be taken to ensure that if issues do arise, there is at least one fallback option that can help to deliver the project on time. This is a time sensitive project which has a definitive end point, finishing behind schedule is not an option: hence the severity of maintaining the critical path. Without this software being delivered on time; patients are at risk of going untreated, and the CCU is forced to use more of its budget to spread themselves thinner. Proactive measures will be taken once if the critical path suffers deviation. As an example, the project will be planned for a 10% leniency in duration. This ensures that if a large, unforeseen issue is discovered, it can be met with accordance without disrupting the deadline. A simple solution for these issues can be in the form of focus-group sessions, with emphasis on diagnosing specific problems, to switching the internal structure allowing for the best use of resources. Following this, the project should have no issue with the due date of 17th April.

Creating an application, capable of running on multiple operating systems, is the first major step. With their limited budget, it is unknown how much can be spent on proprietary licences. Allowing for access to multiple operating systems gives the CCU freedom to expand, or constrict, their technology budget according to their needs, not the needs of the software. Using the same hardware, Virtual Machines will allow for collection of metric data on different operating systems. Examples of metric data collected include Memory Usage and speed at which the Machine Learning algorithm returns a value. The chosen approach for this objective: using a mixture of Electron and Next.js. This will free the project team, letting them focus on the project itself, instead of the different versions traditionally needed. While it would be possible to develop this as a Web Application, given the confidentiality of the CCU, it is important to note the security implications. Restricting this to a desktop application allows entire separation from the internet, widely restricting the possibility of a data breach. The use of Electron as a Multi-Operating-System framework means that there is not much consideration needed to add extra platforms. This boost in efficiency helps to deliver the project on time because code errors are no longer unique to the OS, but project instead.

Given the CCU is dealing with too many patients for them to manage without assistance, it is crucial that the application is capable of accurately predicting those who would benefit from seeing a dietitian. This will be accomplished by allowing users to upload new csv files, containing patient data; then analysing it with the bespoke Machine Learning algorithm. Patients who meet the criteria will be automatically flagged for referral, upon review. It is vital that this algorithm is validated as having a high accuracy, there is no sense delivering an extra feature which will just further complicate the CCU worker. Accuracy can be clearly measured by determining the ratio of perfectly categorised patients to those who were mischaracterised. This objective will be achieved by executing Python code within the Electron application, then outputting the returned python data to graphs for easy viewing.

Creating and exporting graphs and forms of patient data is necessary for continued operations. It allows the workers in the CCU to easily access the data one time, then continue carrying it throughout the day for easy references. Examples of graphs and forms generated include individualised patient overviews, a full list of referrals for the dietitian, and a full list of patients currently admitted. These graphs and forms will also have 3 quantifiable metrics, visual aesthetics, accessibility, and exportability, their importance is paramount. Without the styling having careful consideration, there is a risk of it being too complicated: left unused. To ensure that these are met, different design options will be presented to a small focus group, independent to the development group. This focus group will consist of people from a variety of age brackets and technological ability, ensuring fair accessibility for all users. Options should also be included for different themes, with classic examples being Light and Dark mode, including high contrast mode and options for text size. As previously mentioned, the CCU is spread quite thin. Giving them the ability to export these generated reports would mean they are able to spend less time running back and forth to a computer, and more time spent with the patients who need them. To ensure that this doesn’t cause the projects time constraint to dwindle, the designs shown to the focus group will take place before any development begins. This allows for any revisions to be made before concrete steps have been taken.

# Requirements

## Functional

A key functionality of the proposed healthcare management system is the ability to display a detailed list of all patients within the system. There should also be options allowing users to filter the results and display who is most in need of seeing a specialised dietitian. This feature is designed to help streamline the patient management of the Critical Care Unit, allowing for better delegation of resources.

Without the ability to analyse the patients’ records, however, the application serves no more purpose than a standard patient record viewing tool. This is where the Machine Learning algorithm takes physiological measurements of patients within the Critical Care Unit, flagging them if they should be referred to a specialist. This step needs to happen before the user can filter results, otherwise no new records would be processed for referral.

* Present individual patient data
* Ability to upload new patient data as a CSV
* Generate exportable graphs and reports regarding various patient data
* Run on different operating systems.

## Non-Functional

Produce an efficient design which can run on minimal hardware.

Design a solution which is aesthetically pleasing and intuitive.

There should be an encrypted pass key used to access the software

Allow for different styling options for accessibility