Report

Contents

[Introduction: 3](#_Toc162430409)

[Aims and objectives: 3](#_Toc162430410)

[Objectives: 3](#_Toc162430411)

[Literature Review: 4](#_Toc162430412)

[Tools and methodologies: 4](#_Toc162430413)

[Current and new methods and technologies: 5](#_Toc162430414)

[Related Systems: 5](#_Toc162430415)

[MIMIC-II 5](#_Toc162430416)

[NHS - SystmOne 5](#_Toc162430417)

[Requirements: 8](#_Toc162430418)

[Project Planning & Team Roles: 9](#_Toc162430419)

[Key Tasks 9](#_Toc162430420)

[Show patients admitted to the CCU. 9](#_Toc162430421)

[Make an intuitive GUI using Tkinter. 9](#_Toc162430422)

[Write code that generates reports and graphs based on patient data. 9](#_Toc162430423)

[Implement algorithms that evaluate patient data. 9](#_Toc162430424)

[Gantt Chart 9](#_Toc162430425)

[Team Roles 10](#_Toc162430426)

[Implementation & Testing: 11](#_Toc162430427)

[Design 11](#_Toc162430428)

[Use case diagram. 12](#_Toc162430429)

[Class diagram. 13](#_Toc162430430)

[Sequence diagram. 15](#_Toc162430431)

[Testing 16](#_Toc162430432)

[Test Plan 16](#_Toc162430433)

[Test Log 17](#_Toc162430434)

[Evaluation & Lessons Learned: 18](#_Toc162430435)

[References: 19](#_Toc162430436)

[Appendices: 21](#_Toc162430437)

# Introduction:

Patients are admitted into the Critical Care Unit (CCU) in a hospital and many of the could benefit from seeing a dietitian. This may not be possible due to the high volume of patients and minimal number of resources provided. Due to a high volume of patients, it is easy to neglect other patients and some patients do not receive the care they deserve. To handle this problem, we are working on a project to create a system that prioritises patients based on their physiological measurements.

The aim of this project is to create a system that makes this possible and easier, to prioritise and ensure patients who need to see a dietitian can see one. Patients’ conditions are monitored, and data is collected, using the collected data the system decided which patient is to be flagged (flagged patients need to see a dietitian). This system displays a list of all the patients in the CCU, and filters the patients based on different categories (physiological measurements). This system is accessible to the CCU staff, they can add, edit and update patient details on the system. To help understand the patient’s condition the system generates reports on the patient, this reported is generated is an overview of the patient’s data and is displayed using tables and graphs. To ensure patient’s data is safe, there is limited access to the system, and only those authorised can access the system and the data stored in it.

Many patients would benefit from this system as those who need urgent care will be prioritised and receive the care they need, the patient’s condition is continuously monitored and updated in the system. With the use of this system, improvements and changes in patients can be monitored, and if a patient's condition does change the staff will be notified.

# Aims and objectives:

This project aims to develop a hospital dashboard which takes in patient data and flags the patients who are most in need of seeing a dietitian.

## Objectives:

1. Research similar projects so we can see what we can do differently.
2. Assign specific roles to different team members to distribute the workload.
3. Design and produce a test plan which will be figuring out the technical requirements and evaluating the use cases.
4. Implement and test the project to ensure functionality and reliability.
5. Evaluate which requirements we have met and or missed.

# Literature Review:

There is currently a need for the critical care unit to start prioritising patients who need to see a dietitian because there are insufficient resources for every patient to see a dietitian and the patients who need to see a dietitian the most may currently miss out. This is because it is very difficult for the critical care unit staff to efficiently prioritize patients.

So, developing a feeding dashboard which will flag the patients who need to see the dietitian, will aid the staff significantly, optimising and increasing healthcare resources due to less time needed to prioritize patients this will make sure that patients with a greater need get the help required.

## Tools and methodologies:

The tool we will use to plan this project is Astah UML. Astah UML is a program which allows for the creation of UML diagrams like the use case diagram. Using UML helps to define the scope of the project abstracting it into easily digestible sections. (Fernández-Sáez, Chaudron and Genero, n.d.). We have chosen Astah UML it’s written in Java so it will run on any device, allowing team members on a Mac to contribute to creating UML diagrams.

The tool we will use to manage this project is Gitlab - a version control software which stores a project in a repository. Using Gitlab will allow multiple team members to simultaneously work on the project at the same time while avoiding conflicts because changes to the same file are merged (Perez-Riverol *et al.*, 2016). This will make collaboration easier between team members and make developing the project much easier.

The software development methodology we are going to use to manage this project is a modified version of Scrum, which is an agile software development methodology. Using an agile software development methodology allows for easier collaboration between team members and stakeholders (Karrenbauer, Wiesche and Krcmar, 2019), which means that if the requirements change during the development life cycle we can adapt. We are using a modified version of Scrum because the Scrum methodology includes daily meetings called ‘daily Scrum’ (Schwaber and Sutherland, 2020), which we will modify into 2 weekly meetings instead. This will help the team collaborate while easily fitting it in with our timetables. A similar agile methodology we could have used is Xtreme Programming (Beck, K. and Andres, C. (2004)). This methodology is a more intensive process, with vigorous testing and revaluation taking place at every stage of development. While this would’ve provided the benefit of continuous bug checking to make robust code, this would not have been feasible to implement, as the group is not able to meet often enough to have the required amount of reflection and discussion. In contrast, Scrum still functions well on a less frequent basis.

## Current and new methods and technologies:

One technology we will use to implement this project is the Python programming language. Python is a cross-platform, multi-paradigm programming language (‘Python (programming language) ~ Information Technology ~ 2420 ~ kelas-karyawan-bali.kurikulum.org’, n.d.) meaning that it supports procedural and object-oriented programming. Therefore, choosing Python for programming this project will allow for the program to run on any platform without having to write multiple versions, reducing the time to develop. Furthermore, Python’s support for object-oriented programming will allow for the encapsulation of multiple complex datatypes, improving maintainability.

Another technology that we will use to implement this project is Tkinter. Tkinter is a library that comes with Python, which is used to create graphical user interfaces (Moore, 2018). Using Tkinter means that fewer extra libraries will need to be installed to run the program, making it simpler to deploy. Furthermore, being able to create a Graphical User Interface will make the program easier to use, meaning that less training will be required.

## Related Systems:

### MIMIC-II

The Multiparameter Intelligent Monitoring in Intensive Care II (Saeed, M *et al.,* 2011) was a system developed by academics to produce diagnostic and therapeutic data from a large population of adult Critical Care Unit patients. Patients in a CCU had data values associated with them, such as medication, test results etc. These values were used to create a public-access database for use in various medical research.

MIMIC-II’s functionality very closely resembles the nutrition dashboard that we are looking to develop. We are also looking to use many metrics of patient data, but rather than simply develop a database, we are looking to create an interactive GUI. Therefore, the likes of reports and summaries don’t have to be outsourced to another application, as we plan to have these functionalities built into our system. We can do this as the data we are going to use is already available to us – much of the purpose of MIMIC-II was the initial data collection, having to synchronise data from many different databases throughout the healthcare system.

### NHS - SystmOne

The UK’s National Health Service has a vast amount of data equating to over 80 million patient records (NHS, 2024). Many efforts have been made in recent years to digitalise the handling of their records, to maintain accuracy and consistency, as well as produce reports and summaries which can be used to aid future research into healthcare.

The NHS’ data is stored in a Personal Demographic Service (NHS, 2024). This acts as a bulk database for every patient on record. However, this does not contain any form of user interface for data handling. The PDS is instead interacted with from several solutions. For example, a hospital receptionist using a *Patient Administration System*, and a citizen using the NHS app will both be transferring data to and from the PDS.

GP surgeries need to often interact with the PDS. To do this, they will have a standardised system in their network. The three most used are *EMIS Web, SystmOne* (TPP, 2024)and *Vision* (GP Training Support, 2016)*.* These products bear a visual resemblance to the dashboard which we will be looking to develop.

#### SystmOne

SystmOne facilitates access to both patients and medical staff. Patients can use the service to register or change their details, as well as being able to order prescriptions.

The staff can use the service to view patient records and add any updates about recent appointments or changes to medication. There is also plenty of functionality to produce reports on the data:

* Clinical Reports
  + Reports created for local use, using whatever data is required.
  + i.e. A report on one GP surgery’s new admissions during February.
* National Reports
  + Reports intended to be viewed by many entities nationally.
  + Will uphold national standards of report structure.
* Strategic Reporting
  + Bulk data extract for use in another application, or to be sent to another organisation.

These features allow medical staff to produce any required reports quickly and effectively. This is especially true when compared to before digitalisation, where the NHS would have to collate data from however many surgeries/hospitals throughout the UK.



Fig 1.0 – A snapshot of the Clinical Record Viewer in SystmOne (Adewunmi, 2014).

In Fig 1.0, we can what the SystmOne application looks like while running on a staff computer. The user is presented with a Graphical User Interface, which has many features that we can use as inspiration for our dashboard:

* Side menu containing different functions.
* Main window which changes depending on the side menu selection.
* Ribbon of options displayed across the top of the screen.
  + Remain consistent regardless of function selection.
* Effective use of contrasting colours for easy visibility.

# Requirements:

The requirements of the system are quintessential to our project. This involves both identifying the requirements and fulfilling them in the implementation.

Functional requirements are those that have been directly requested or mentioned in the specification. These requirements allow the system to operate.

Non-functional requirements are ones that would improve the system beyond what is required of it. For example, improving the user experience and performance would greatly benefit the system’s utility, but is not completely necessary to enable the system to function.

Generally, it makes sense to prioritise the functional requirements. However, non-functional are still highly important, and have a great impact on a client’s satisfaction with their product.

|  |  |
| --- | --- |
| Functional Requirements | Non-functional Requirements |
| Provide a graphical user interface to the user. | The graphical user interface should be nice to look at and easy to use and inclusive in design. |
| Generate a list of patients in the Critical care unit. | Limit access to certain individuals depending on their user privilege. |
| Filter patients based on the category e.g., Body mass index | Create and delete user accounts. |
| Flag patients who need to see a dietitian depending on the data provided. | The ability to run on multiple different platforms (windows, Mac and Linux). |
| Review individual patient data and allow the user to edit it. | Good performance when doing more complex tasks. |
| Generate reports on individual patients e.g., bar and pie charts. |  |

# Project Planning & Team Roles:

## Key Tasks

### Show patients admitted to the CCU.

This task involves one of the team members writing Python code to open and read a CSV (comma-separated values) file and display its contents. Ideally, the code allows the user to choose a specific file they want to read and display as opposed to the file being hardcoded. Also, the contents of the file should be displayed in a Tkinter window in the form of a table. Realistically one person should be able to complete this task on their own in a short timeframe as this task does not have any complex requirements.

### Make an intuitive GUI using Tkinter.

This task involves the creation of a User Interface which would be used to navigate between the different functions of our software. We’ll be using the tkinter framework that comes pre-packaged with Python. This will likely be one of the harder tasks and as such more members of the team would have to be allocated to it and it might take longer to complete.

### Write code that generates reports and graphs based on patient data.

The objective of this task is to write code that interprets the data of individual patients and generates reports based on said data. It should also be able to look at the data of multiple patients and generate reports on them as well. The reports could be either in a table or graphically displayed in the form of multiple graphs. The task itself won’t be the most difficult in all likelihood and as such it won’t have that many resources dedicated to it. Perhaps two people should be able to complete it on time.

### Implement algorithms that evaluate patient data.

This task aims to make a functional implementation of the algorithms provided in the case study. These algorithms would take in the data of individual patients and if they deem it necessary, the patient would get flagged to see a dietitian. This task could pose some difficulties and as such more resources would have to be allocated.

## Gantt Chart

During the project we will be following a Gantt Chart to monitor our progress. A Gantt Chart allows us to visualise different details of tasks and their completion, including the time allocated, deadlines and any simultaneous tasks that may take place at the same time.

The first version of the Gantt Chart is shown below, and uses the previously stated key tasks in the Project Planning section, as well as some additional objectives. The chart may be modified during our project if deadlines or objectives change, and this will be evidenced in the report.

A graph with orange and white lines

Description automatically generated with medium confidence

Fig 2.0 – Gannt Chart 1st version.

## Team Roles

As we are following a scrum methodology, all members of the project have the same level of responsibility. This means that no one type of task is delegated to specific people. We all have equal responsibility for both the development of the system and for producing the documentation. We are also sharing the role of Team Coordinator week-by-week. The coordinator is responsible for holding weekly meetings with the group, to log the progress through the project.

# Implementation & Testing:

## Design

We first produced designs for the system. To do this, we made multiple UML diagrams using AstahUML. The diagrams produced were:

* Use case diagram.
  + This represents the users of the system, and which procedures they will be completing. It takes a high level view, only describing the procedures with minimal detail. This provides an early indicator of the size and complexity of the system.
* Class diagram.
  + This represents any classes that we will be using in the implementation. This is particularly important in OOP projects, which often have many class objects. This provides assistance in the programming of the system.
* Sequence diagram.
  + These diagrams represent a specific case of user(s) interacting with the system. It takes an example from the use case diagram, and shows interaction with a class using the methods described in the class diagram.

### Use case diagram.

A diagram of a diagram

Description automatically generated with medium confidence

Fig 3.0 – Here we can see that the use case diagram is very simple. This is because our system should only have one type of user. These diagrams become much more complicated when multiple users have access to the system.

### Class diagram.

A screenshot of a computer

Description automatically generated

Fig 3.1 – Again, our class diagram is relatively simplistic. This is because, even following OOP procedures, the system only requires one class. However, each patient does need a large number of methods to interact with the large number of attributes.

It is worth noting that we have a non-functional requirement of having user accounts in the system. On implementation, this would require a new class, but would have less attributes and methods.

### Sequence diagram.

A close-up of a document

Description automatically generated

Fig 3.2 – In this sequence diagram, we are representing the process of a user retrieving all of a patient’s data. We can see all the getters from the patient class being used.

## Testing

Before implementation began, we designed a test plan for the system. The test plan features different procedures that we should expect the system to be able to complete. These procedures are linked directly to the system requirements.

An accompanying test log is to be completed after implementation. This involves working through each test case and logging the outcome. If the test fails and requires changes, this is also mentioned in the test log.

### Test Plan

|  |  |  |
| --- | --- | --- |
| Test no. | Test | Expected Result |
| 1 | Can the system load patient data? | The system should load patient data from a user-allocated CSV file. |
| 2 | Is patient data presented to the actor? | A table should be generated using the supplied data from the CSV file, showing patients and all of their corresponding data. |
| 3 | Can patient data be organised? | Actor should be able to sort and filter patients by metrics e.g. show patients w/ BMI > 30. |
| 4 | Does the system flag a patient if they meet the right criteria? | The system should automatically recognise if a patient’s data suggests that they are due an appointment with a dietician. |
| 5 | Can a patient’s data be edited? | An actor should be able to select a patient and modify their data in the system. |
| 6 | Can reports be generated on patient data? | The actor should be able to generate summative reports on the patient data. This may include visual charts and/or tables of values. |
| 7 | Is the system protected by a login process? | All of the system’s functionalities should be inaccessible until a valid username and password are used to login. |
| 8 | Can the actor add an account to the system? | The actor should be able to add a new account to the system which could be used for future log-ins. |
| 9 | Does the system perform suitably? | System should have fast response times to user interaction. |

### Test Log

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Test Data | Pass/Fail | Notes/Changes |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |

# Evaluation & Lessons Learned:

# References:

Adewunmi, A. (2014) Medway Maritime Hospital Case Study. A Critical Appraisal of the Summary Care Record (Scr) Scheme in England [online]., p. 5. [Accessed 12 February 2024].

Beck, K. and Andres, C. (2004) Extreme Programming Explained: Embrace Change. 2nd ed. : Addison-wesley Professional.

Fernández-Sáez, A., Chaudron, M. and Genero, M. (no date) *Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department* [online]. Available from: <https://ceur-ws.org/Vol-1078/paper4.pdf>.

GP Training Support (2016) Computer systems in general practice. Available from: <https://gptraining.info/computer-systems-practice/#:~:text=The%20main%20systems%20are%20EMIS%20Web%2C%20SystmOne%2C%20and%20Vision>. [Accessed 11/02/2024].

Karrenbauer, J., Wiesche, M. and Krcmar, H. (2019) Understanding the Benefits of Agile Software Development in Regulated Environments. *Wirtschaftsinformatik 2019 Proceedings* [online]. Available from: <https://aisel.aisnet.org/wi2019/track07/papers/5/>.

Moore, A.D. (2018) *Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter* *Google Books* [online]. Packt Publishing Ltd. Available from: https://books.google.co.uk/books?hl=en&lr=&id=2kBbDwAAQBAJ&oi=fnd&pg=PP1&dq=tkInter&ots=NgppXWINnl&sig=1G9Rr3C3mGPDv757\_ZZdg1cuk60&redir\_esc=y#v=onepage&q=tkInter&f=false [Accessed 12 February 2024].

NHS (2024) Personal Demographics Service. Available from: https://digital.nhs.uk/services/personal-demographics-service [Accessed 11/02/2024].

Perez-Riverol, Y. *et al.* (2016) Ten Simple Rules for Taking Advantage of Git and GitHub. Markel, S., ed. *PLOS Computational Biology* [online]. 12 (7), p. e1004947. [Accessed 1 July 2020].

‘Python (programming language) ~ Information Technology ~ 2420 ~ kelas-karyawan-bali.kurikulum.org’ (no date) *kelas-karyawan-bali.kurikulum.org* [online]. Available from: <http://kelas-karyawan-bali.kurikulum.org/IT/en/2420-2301/Python_3721_kelas-karyawan-bali-kurikulumngetesumum.html>.

Saeed, M., Villarroel, M., Reisner, A.T., Clifford, G., Lehman, L.-W., Moody, G., Heldt, T., Kyaw, T.H., Moody, B., Mark, R.G.: Multiparameter Intelligent Monitoring in Intensive Care II: A public-access intensive care unit database\*: Critical Care Medicine. 39, 952–960 (2011).

Schwaber, K. and Sutherland, J. (2020) *Scrum Guide | Scrum Guides* *Scrumguides.org*. November 2020 [online]. Available from: <https://scrumguides.org/scrum-guide.html>.

TPP (2024) SystmOne. Available from: https://tpp-uk.com/products/ [Accessed 11/02/2024].

# Appendices:

A screenshot of a computer

Description automatically generated

A1.1 – This is a Gantt Chart revision as of 13/03/24. Some of the deadlines were extended as implementation took longer than expected.

A screenshot of a computer

Description automatically generated

A1.2 – This is a second Gantt Chart revision as of 27/03/24. Again, some deadlines were extended as implementation took longer than expected.