-

Abstract

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Team members

Steven Kennedy 40025827

Jonathan Leathem 40059090

Conor McAleavey 40047330

Chris McClune 40138365

Rebecca McKevitt 401xxxxx

Kevin O’Hare 15481042

Victoria Stacey 401xxxxx

**CSC 7051**

**Report**

Contents

**1.** **Peer marking** 3

**2.** **Describe and evaluate the development model used** 4

**3.** **Design (400 words max)** 4

**4.** **Implementation (500 words max)** 6

**4.1. User Interface (Java FX)** 6

**4.2. Database** 6

**4.3. Queue System** 6

**5.** **Testing (max 200 words)** 7

**6.** **Evaluation of Project and Product Produced (max 300 words)** 7

# **Peer marking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student Number** | **Student Email** | **Student Name** | **Section(s) Tested** | **Peer mark (whole numbers only)** |
| 40025827 | skennedy2001@qub.ac.uk | Steven Kennedy |  |  |
| 40059090 | jleathem03@qub.ac.uk | Jonathan Leathem |  |  |
| 40047330 | cmcaleavey02@qub.ac.uk | Conor McAleavey |  |  |
| 40138365 | cmcclune02@qub.ac.uk | Chris McClune |  |  |
| 401xxxxx | @qub.ac.uk | Rebecca McKevitt |  |  |
| 15481042 | kohare08@qub.ac.uk | Kevin O’Hare |  |  |
| 401xxxxx | @qub.ac.uk | Victoria Stacey |  |  |

# **Describe and evaluate the development model used (200 words max)**

The development of this product did not rigidly adhere to a single development model, rather, combined the benefits of a number of popular software development methodologies, including, but not limited to, Agile Development, eXtreem Programming (XP) and The Waterfall Model.

Due to the limited resources at our disposal for this project we initially decided to closely follow the Waterfall Model. At the outset, this provided our project with structure and a definite end date, which was key to our team completing the project before the submission deadline. Despite the obvious benefits of this model, we decided that using this development methodology in isolation was limiting to our project. This was because as the software progressed, the access to the client became more valuable – allowing for more interesting queries and an end-product that was hopefully even closer to the client’s expectations.

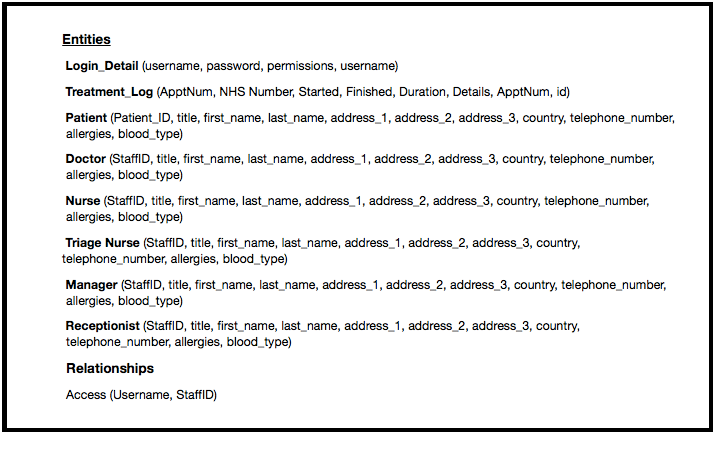
Although we did adopt a general Agile development practice for our project – towards the end of the software’s implementation we did lean towards the XP methodology (Beck 1996). This benefited us greatly because of the constantly changing approach to aspects of our system, were we found ourselves creating regular increments of the software, reviewing and improving the code as we went along.

# **Design (400 words max)**

* DB Design (you may include a DB Schema)
* Software System Design (you may include any UML class diagrams, high level system architecture etc.)
* User Interface Design

**Database Schema**

To reduce redundancy and minimise chances of loss of data, we decided to produce a schema to model our database from. Also some of the tables chosen in the schema are designed are for the propose of expanding the system. The reason for this is to create a system that is more flexible and that will be able to cope with future modifications. The entities and relationships in the database schema are as follows:

 **UML Diagram (Rebecca working on)**

**User Interface Design**

A user interface was created using Java FX and Scene Builder 2.0. The user interface has various pages which we have designed to be consistent in colors and theme. The layout of the pages is quite simple as we intended the system to be intuitive and uncomplicated for users.

The user interface consists of the following pages:

* + - * Login - To be accessed by doctors, nurses and receptionist. A login screen which the user will use to enter the credentials which are their staff number and password.
      * Queue System - To be accessed by the triage nurse. This screen has a drop down box which allows the user to modify the patients position in the queue by assigning them a new triage priority. There is another drop down box which allows the user to select which treatment room to assign to the patient.
      * Reception Layout - To be accessed by the receptionist. The screen allows the user to search for the details of patients who have entered the A&E department. There are four fields where patients details can be entered. These are first name, last name, postcode and NHS number. There is a table of search results of patients in the database whose details match those inputed. The patients information that is displayed is title, first name, last name, street, city, postcode and phone number. The user can select the patient from the list and add them to triage.
      * Treatment Room - To be accessed by doctors and nurses working in the treatment rooms. The first name, last name blood type and known allergies of the current patient being treated are displayed on the screen. The time the treatment began is also shown on the screen. The user has the option to extend the treatment time and there is a text box for the user to enter a summary of the treatment which they can save in the system.
      * Triage - To be accessed by the triage nurse. This screen shows a table including the patients NHS number, first name, surname, allergies and condition. There is a drop down box which is used to assign a triage category to the patient which can then be saved into the system.

# **Implementation (500 words max)**

*Describe your implementation of the main functional parts of the system. You may illustrate this with code snippets and screen shots of the UI.*

# **4.1.User Interface (UI)**

# Designed using JavaFX Scene Builder 2.0 – our UI plays a key role in the overall delivery of our system to the client. Therefore, it was important we created a friendly and easy to use UI. Scene Builder afforded us the opportunity to create such a design with and array of built-in features such as labels, buttons, and frames among other visually impressive utilities.

# We created our UI by firstly adding a new FXML file from within Eclipse – setting up the basic elements of the screen, ensuring to reference the corresponding Java Controller class in the project for each FXML file. We then opened this file in Scene Builder to add features and to make more precise layout edits to the file.

# Handle methods within the controlling class of the UI window are performed using button, which once clicked will call the methods or algorithms as needed. The function of the UI windows is to allow the user to enter data as depending on the window type, such as entering Patient names into the Reception screen (i.e. Reception Layout class), or to assign a triage category to a Patient. UI values are also used to pass across data added to one UI onto another UI in the form of strings.

# **4.2. Database**

xxxxxxxxxx

# **4.3. Queue System**

Contained within QueueController.java class, our queue provides some of the most functionally important aspects of our system. The queue is displayed using the Eclipse console queue data is performed within the console output using ‘System.out.print’ functions to display the information as refreshed from the runnable thread every four seconds using an infinite loop. All of the code needed to print out data, sort data, check if patients are in treatment rooms, etc, are kept within this main thread. Patient, TreatmentRoom, TreatmentRoomThread and PatientThread are all classes used by this queue controller class; ‘Queue.fxml’ is used to control aspects of the queue, e.g. extension of time, search functions, edit triage, open up treatment room windows for Doctors, etc.

Instances of new Patients are created and passed into the queue controller class from variables passed across using the ‘TriageController.class.’

As a group, we decided that for the current system having our queue displayed in the UI was

Linked lists as part of the Java Collections Framework are used to store patients in the queue and the Treatment Rooms as part of their own unique linked list. Patients within the queue are sorted using ‘Collections.sort,’ that is called for every refresh of the queue. Each Patient object and Treatment Room object have assigned their own threads called on creation of the unique object, were each created new thread is run in the background using a runnable thread in the queue controller class, to allow the main thread to run without interruption (i.e. a thread within a thread). Boolean variables are used to check whether Patients are in rooms, and the use of Treatment Room count threads to check if they were to be removed from the treatment room.

# **Testing (max 200 words)**

*Describe and evaluate the Testing strategy used throughout the project.*

Ideally as a team we would have liked to have JUint tested the entire project, but due to a technical road block with testing the fxml UI we were unable to. However, we continuously checked the functionality of the project for each new update to the code, using ‘ad hoc’ tests as necessary and reused test cases to ensure previous functioning code was unaffected by updates.

We functionally tested the database……

# **Evaluation of Project and Product Produced (max 300 words)**

*Does the product fulfil all the requirements, what additional features does it have… This section should provide a thorough and honest reflection on the process followed in the project and the results of that process.*

To help our team visualize the design and structure of the system we intended to implement we decided to create a database schema and UML diagram of the system. Then a database was able to be created based upon these designs. After the creation of the UML diagram several basic classes such patient and doctor were coded.

Then the focus shifted to the creation of the user interface system in Java FX. The team designed the several different screens to be used in the system.

The team’s then shifted back to coding the system were the main functionalities were implemented into the system and these were linked to the user interface. During this process the user interface underwent some major redesigns to compliment the new code. In hindsight it would likely have been a wiser decision to wait until the majority of the code was implemented and functioning before designing the user interface. Which probably would have been a better use of the group’s time.

The group then moved onto testing the system and removing and correcting defaults.

The first stage of our project involved the planning and design of our system. We conducted several meetings in which we examined the customer’s product description and system requirements. This allowed us to design a system around the customer – ensuring that the end product was as close to the customer’s expectations as would be reasonably expected. It was in this stage of the development we were able to plan and assign roles to each team member given an end goal insight.