

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SCHOOL OF SCIENCE AND TECHNOLOGY**

**FINAL ASSESSMENT FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; YEAR 2**

**ACADEMIC SESSION 2021; SEMESTER 3**

**PRG2104: OBJECT ORIENTED PROGRAMMING**

**Project DEADLINE: Week 14**

INSTRUCTIONS TO CANDIDATES

* This assignment will contribute 50% to your final grade.
* This is an individual assignment.

**IMPORTANT**



**Lecturer’s Remark** (Use additional sheet if required)

I Raveesh Shibchurn (Name) 19053156 std. ID received the assignment and read the comments R.S / 21/07/2021 (Signature/date)

**Academic Honesty Acknowledgement**

“I Raveesh Shibchurn (student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment.”

R.S / 21/07/2021 (Student’s signature / Date)

Contents

[Introduction 3](#_Toc77796523)

[Project implementation 3](#_Toc77796524)

[Features Explanation 4](#_Toc77796525)

[UML Diagram 20](#_Toc77796526)

[Personal reflection 21](#_Toc77796527)

[System Analysis 22](#_Toc77796528)

# Introduction

The medical industry is evolving with years; thus, keeping track of data of patients is critical for future reference or even for further analysis. With this advent of technology in the medical sector, +Care is developed to enhance the collection of data of patients be it in a hospital be it in a clinic. The simplistic approach of +Care promises a good user experience.

The user of +Care can easily add a new patient in the system, view a patient’s details and also delete patient record. More features implemented in the application are discussed in the features section.

# Project implementation

Tools used to implement the project:

* VS code to write the coding of +Care.
* Scenebuilder to design +Care’s frontend interface.

+Care was implemented using the model view controller approach. The model being the patient class which is used throughout the coding. The views, designed using scenebuilder, are controlled by the controllers which execute the functions and handle the operations of the application.

# Features Explanation

For the UI, a simplistic design approach was implemented for the +Care app, where the user will gain access to a diverse set of features that will help them to input and view the details of patients. The app’s simplistic design includes a standardized theme throughout the pages as well as a menu bar at the top giving them more options or insight about +Care. The app’s ease of access makes it a powerful tool that anyone within the medical sector can take advantage of to manage and analyse their patients’ data.

A blue colour has been chosen for the main parts of the application based on the psychology of colours. This colour has been chosen because psychologically, it is said to promote knowledge and tranquillity, to generate a positive and trustworthy vibe, and in general to create secure environment.

1. +Care overview

Diagram

Description automatically generated

Figure 1 – Welcome page

When we click on get started, we will be directed to the +Care’s patient overview page.

Figure 1 shows the Welcome page which greets the doctor using the application. This page was styled using Cascading Style Sheets. The get started button’s function is coded in the WelcomeController.scala. The file, More options, and Help are from the RootLayout. Their respective functions are explained below.

Graphical user interface, application

Description automatically generatedFigure 2 – Patient Overview page

Search bar used to filter the patients in the tableview. (Functionality displayed below)

Used to remove a selected patient from tableview.

Used to edit a selected patient’s detail.

Used to add a new patient (explained below)

A tableview was implemented to display all the patientData (observable buffer)

The patient overview page displays all the main features of the +care. This page is the very essence of the application, and all the features will be explained in detail below.

Graphical user interface

Description automatically generated

Figure 3: clicked a patient in the tableview

The above page portrays what the overview display when the user clicks on a specific patient in the tableview. When this explained action is executed, the patient details will be displayed.

1. Add new patient Features

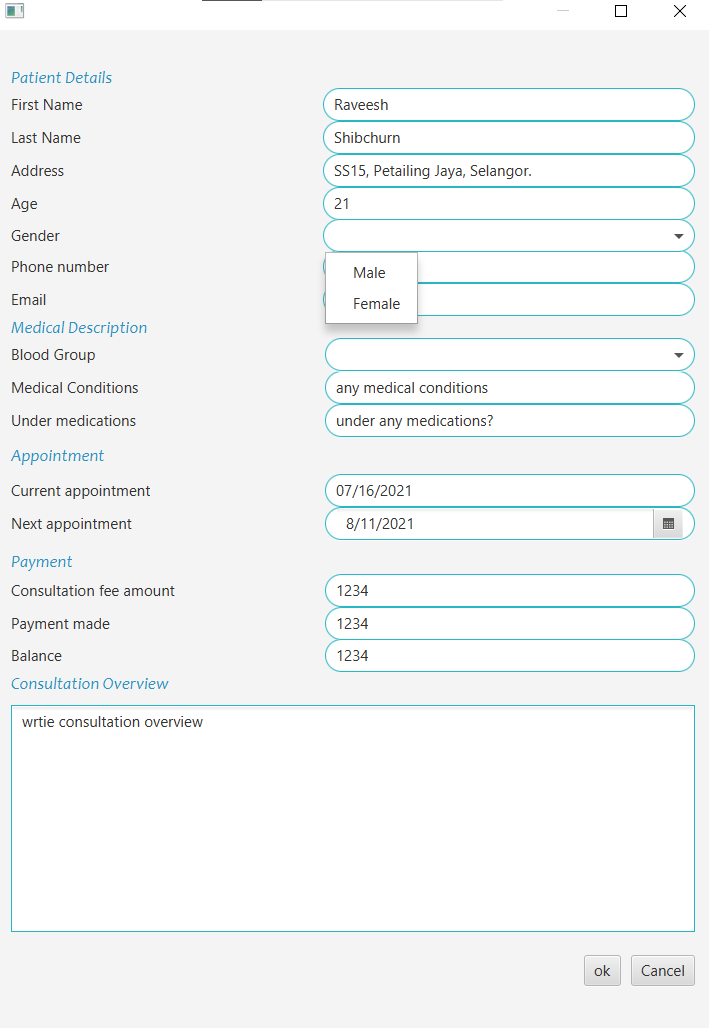
Graphical user interface, application

Description automatically generated

Figure 4: New button

When we click on the new button in the patient overview, a dialog is displayed. Therefore, the user just needs to fill in the details of the patient in the form and click on OK to confirm the transaction; thus, the new patient is added in the system. However, if the user clicks on cancel, no new patient will be added. The following diagrams will show the step to add a new patient.

Text field to insert the patient’s first name



Choice box to input the gender

Text field to insert the patient’s age

Text field to insert the patient’s address

Text field to insert the patient’s last name

Figure 5: The Patient edit dialog

Graphical user interface, application

Description automatically generated

Choice box to select which blood group to input

Text field to input phone number

Text field to input email

Figure 6: The Patient edit dialog

Calendar

Description automatically generated

A date picker to select the next appointment date

Text field that automatically input the current date for the user

Text field to input what medications the patient is taking

Text field to input the medical conditions

Figure 7: The Patient edit dialog

Graphical user interface, application

Description automatically generated

Text area to write the consultation overview. The user can write as much as he wants and details the entire consultation for future reference.

Text field to input the balance that the patient needs to pay

Text field to input the payment did

Text field to input the consultation fee

Figure 8: The Patient edit dialog

Figures 5,6,7 and 8 show patient edit dialog in which the user can input the patient’s details, medical history, appointment dates, payment status and consultation details. Moreover, all the fields have a respective data type.

The following details each field type:

* First name is a StringProperty
* Last name is a StringProperty
* Address is a StringProperty
* Age is an ObjectProperty of type integer
* Phone number is an ObjectProperty of type integer
* Gender is a StringProperty
* Email is a StringProperty
* Address is a StringProperty
* Blood group is a StringProperty
* Medical condition is a StringProperty
* Under medication is a StringProperty
* Current appointment is an ObjectProperty of type LocalDate
* Next appointment is an ObjectProperty of type LocalDate
* Consultation fee amount is an ObjectProperty of type integer
* Payment is an ObjectProperty of type integer
* Balance is an ObjectProperty of type integer
* Consultation overview is a StringProperty

Choice boxes are used to select the gender and blood group to help the user to input the correct information and to hinder the user from making mistakes. Furthermore, the current appointment date’s field will automatically display the current date the user is adding the patient’s details. However, they can edit the current date if they are adding an entry of a patient from a previous date. Moreover, I implemented a date picker to allow the user to easily assign the next appointment date without making mistakes.

Specifying the field hinders the user from making errors. There is error checking for all fields except for blood group because sometime the patients may not know their blood group; thus, the system allows the user to keep this field empty.

For Instance, in the above diagram, I have purposedly write the phone number as +23057170909 which is not of integer type. Thus, an error message will be seen to warn the user to change the text to integer. The alert message is popped up with sound effect. Figure 9 depicts the alert:

Graphical user interface, application

Description automatically generated

Figure 9: alert message

Graphical user interface, application

Description automatically generated

Figure 10: Corrected phone number

In figure 10, we can notice that all the fields are filled correctly. The user can just click on the ok button to add the patient. Thus, a patient object is created and added to the observable buffer of type patient. The patient first name and last name will be displayed in the tableview, and the patient’s details will appear on the right-hand side split pane, as shown in figure 11. When the user successfully adds a new patient, a sound effect is played.

Graphical user interface, text, application, email

Description automatically generated

Figure 11: New patient added

1. Edit patient’s details

The user can edit the information of any patients at any time. The user just needs to select the patient and the patient edit dialog will be displayed with all the details of the patient. The user just needs to update the data and clicks on the ok button to confirm the changes. However, if the user clicks on cancel, no changes will be made. Furthermore, if the user does not select any patient and clicks on the edit button; an alert will be displayed to advise the user to select a patient in the tableview.

The figure below shows what happened when the user selected patient Raveesh and clicked on edit button.

Graphical user interface, application

Description automatically generated

Figure 12: Edit button action

1. Remove patient

When the user selects a patient from the tableview and click on the remove button, the selected patient object is deleted from the patientData observable buffer and removed from the tableview.

Graphical user interface, application, email

Description automatically generated

Figure 13: Selected patient to be removed

Graphical user interface, application

Description automatically generated

Figure 14: Patient removed

1. Search feature

The search feature allows the user to search for a specific patient by typing the name in the search bar. While typing the name, the tableview will be filtered and sorted to display the patient related to the search entry. Figure 13 depicts this scenario.

Graphical user interface, text, application

Description automatically generated

Search bar made using a text field.

Figure 15: Search feature

1. Reset feature

When the user clicks on more option in the menu bar, there is a reset option. This feature allows the user to delete all the patients in the table view and this step is irreversible. Thus, an alert is displayed to ask the user’s confirmation to proceed to reset (figure 14). When the user clicks on ok, the system will clear all the data in the patientData observable buffer and the tableview will be empty (figure 15)

Graphical user interface, text, application, email

Description automatically generated

Figure 16: Reset alert

Graphical user interface, application

Description automatically generated

Figure 17: Reset executed

1. About

When the user clicks on help on the menu bar, there is a menu dropdown consisting of about section. When the user clicks on the about, a small description on +Care is displayed.

Graphical user interface, application

Description automatically generated

Figure 18: About

1. Save file

When the user clicks on the file section in the menu, there is a drop down having the save option. Thus, if the user clicks on save, a save dialog is displayed. The implementation to save the file is not completely done.

Graphical user interface, application

Description automatically generated

Figure 19: save

# UML Diagram

Diagram

Description automatically generated

# Personal reflection

Object oriented programming concepts were implementing throughout the development of +Care. Firstly, +Care consists of a model class that is the patient class. Thus, when a user creates a new patient record in +Care, it becomes an instance of the patient class. This approach promotes efficiency in coding since we can reuse this model throughout the coding in other classes to implement more features related to patient.

Furthermore, there are the implementations of controllers which handle the functionalities of the view pages. These pages are where the user interactions occur, for example in the patient overview page the user can use function like new, edit or delete. We also designed the FXML files in accordance with the system requirements and also to display and accommodate each patient objects.

Problem encountered

Implementing the search function was a very lengthy process since I had to grasp both the JavaFX and ScalaFX API in order to understand the online resources to implement the search function in +Care. Moreover, the implementation of a database raises various errors in my code which consequently made me drop this idea. Furthermore, at the beginning, understanding the various features of the scenebuilder was an ordeal but when I grasped it, the flow of designing the scenes were seamless and fun.

# System Analysis

Strengths of +Care

* +Care’s features are all functional and the UI of the system is simplistic and easy to understand. During the design implementation, the layout of all the pages was kept consistent in terms of colour.
* +Care permits easy reversal of actions. For instance, the user can cancel when editing a patient’s details.
* The application keeps the users in control since they are the one that decide what to do once they start the application.
* The application reduces short-term memory load since the system is easy to understand and no extra reading or knowledge is required to grasp +Care features. +Care was implemented with known features and technology.
* +Care meets all the system requirements – add new patient, delete patient record, and edit patients’ details. Moreover, we implemented a search features to assist the user when searching for a specific patient. There is even a reset feature to delete all the patient records.

Weaknesses of +Care

* +Care shows only the save dialog when you select save option in the menu bar. Due to some difficulty in coding the save function, the system cannot be saved in a proper format.
* +Care does not allow the user to highlight a patient’s record as an important record in the table. Thus, all the patients in the system cannot be differentiated in the tableview.
* +Care does not have a database. The implementation of a database will enhance the system performance and efficiency.