## eSoftware Engineering 2 Report

Subject Number: 14013304-3

Year 2022 Second Semester

Umm Al-Qura University

**Clinic Management System (Icare)**

Umm Al-Qura University, KSA

Faculty of Computer and Information Systems

Dept. of Computer Science

2022

This Report Is Part of The Software Engineering 2 Subject of The Department of Computer Science at Umm Al-Qura University

**Group Name:** Abysswalkers

Logo

Description automatically generated**Group Logo:**

**Group Members:**

1. Khalil Ibrahim Alsulaimani (Group Leader) ID:439007816 Group: 1
2. Saher Samir Alhazmi ID:439000763 Group:
3. Abdulaziz Othman Ghazi ID:439004604 Group:3

Dept. of Computer Science Internet: http://uqu.edu.sa

Faculty of Computer and Information Systems Phone: +966 25501000

Umm Al-Qura University

Kingdom of Saudi Arabia

ACKNOWLDGMENTS

We would like to thank our great Dr for overseeing this subject, guiding us, and teaching us how to make a report for our future and we wish him the best because he gave us the best he can.

ABSTRACT

Now a days applications are integrated into our daily life for both consumers and businesses therefore any business of any size needs up to date software that can do the core functions of a clinic system such as adding patients ,editing ,searching , booking appointments and has a hierarchy for the system users such receptionist , managers, and owners to maintain information integrity which is necessary to get consumers trust which will result in better profits. There are many systems that provide that for large-scale hospitals and clinics however for their scale that results in a very expense information systems that are rigid and nonflexible with customizability and its features furthermore these large-scale applications demand high spec systems to run on which is not an optimal option for small to medium businesses .That’s where Icare comes in as An Information System for a small to medium sized clinic that will solve the large-scale application problems by being a lightweight desktop application that will work on low to medium spec systems which will provide the core functionality any small to medium clinic would need moreover won’t be bigger than what the user needs i.e., they pay for what they need and won’t have any unnecessary features there paying for .This will be achieved by using tools such as , java for the backend with javaFX for frontend interfaces and MySQL for the database all these three tools are great for light weight desktop applications. In conclusion we aim to fill a gap in the market where cheaper smaller apps will work for clients that can be easily customized to their needs that work on low spec systems.

**Table Of Contents**

[Chapter 1 Introduction ..................................................................................](#chapter1)..................................7

[1.1. Purpose Of The Project..................................................................................](#purpose_of_the_project)...........................8

[1.2 Purpose Of This Document..............................................................................](#purpose_of_the_document)..........................8

[1.3 Overview Of This Document ..........................................................................](#overview_of_the_document)..........................8

[Chapter 2 Requirements Analysis ...............................................................................](#chapter2)....................9

[2. Requirements Analysis ...........................................................................................](#system_analysis)..................10

[2.1 Project Scope .................................................................................................](#project_scope).............10

[2.2 System Requirements ...................................................................................](#system_requirments)...............10

[2.2.1 Functional Requirements ..........................................................................](#functional_requirments)...10

[2.2.1.1 Login Screen …………………………………………………….10](#login_screen)

[2.2.1.2 Owner Screen](#owner_screen) ……………………………………………………13

[2.2.1.3 Manger Screen](#mannger_screen)…………………………………………………...25

[2.2.1.4 Receptionist Screen](#receptioist_screen)………………………………………………33

[2.2.3 Non-Functional Requirements ...................................................................](#non_functional_requirments)..40

[2.2.3.1 Look And Feel Requirements.......................................................](#Look_and_feel_requirements).40

[2.2.3.2 Usability Requirements .................................................................40](#Usability_requirements)

[2.2.3.3 Security Requirements ..............................................................](#Security_requirements)....40

[2.2.3.4 Performance Requirements ..........................................................](#Performance_requirements).40

[Chapter 3 Design Considerations .......................................................................................](#chapter3)...........41

[3. Design Considerations ..............................................................................](#Design_Considerations)................................42

3.1 Design Constraints ......................................................................................................42

[3.1.1 Software Environment .................................................................................42](#Software_Environment)

[3.1.2 End-User Characteristics .............................................................................42](#End_User_Characteristics)

[3.2.2 Project Management Strategies ....................................................................43](#Project_Management_Strategies)

[3.2.3 Development Method ...................................................................................43](#dev_method)

3.2.3.1 Analysis…………………………………………………………..44

3.2.3.2 Design……………………………………………………………44

3.2.3.3 Implementation…………………………………………………..44

3.2.3.4 Testing……………………………………………………………44

[3.2.4 Gannet Chart……………………………………………………………….](#gannet_Chart)44

[Chapter 4 System Design .............................................................................................................45](#chapter4)

1. [System Design ..................................................................................................................46](#System_Design)

[4.1.1 Architectural Pattern ................................................................................................46](#Architectural_patterns)

[4.1.2 The Model-View-Controller Pattern ....................................................................... 46](#The_Model_View_Controller_Pattern)

[4.1.3 The Component And Link Diagram ………………………………………………47](#The_Component_and_Link_Diagram)

[4.1.4 Class Diagram……………………………………………………………………...48](#over_all_class_digram)

[4.1.4.1 Overall Class Diagram……………………………………………..…….48](#over_all_class_digram)

[4.1.4.2 DoctorQuries Class Diagram………………………………………..…...49](#doctor_quries_class_digram)

[4.1.5 Software Architecture Diagram……………………………………………………50](#Software_Architecture_Diagram)

[4.1.6 Layered Architecture Diagram……………………………………………….……50](#Layered_Architecture_Diagram)

[Chapter 5 Implementation .............................................................................................](#chapter5)................51

1. [Implementation ...........................................................................................................](#implemnation)......52

[5.1 Main Method….……………………………………………………………………](#mainMethod)52

[5.2 Login Method..……………………………………………………………………..](#loginMethod)53

[5.3 Database Connection………………………………………………………………](#DBConnect)54

[5.4 Logger Class……………………………………………………………………….](#LoggerClass)54

[5.5 Objects and Queries Classes……………………………………………………….](#ObjectAndQuriesClasses)55

[5.5.1 The Object Class………………………………………………………....](#theObjectClass)54

[5.5.2 The Query Class……………………………………………………….](#TheQuriesClass)....56

[5.6 Testing function……………………………………………………………………57](#testingFunctions)

[Chapter 6 Validation](#chapter6) ................................................................................................................58

6. [Validation](#Validation) .........................................................................................................................59

6.1 Interface testing and database testing .......................................................................59

[Receptionist](#Receptionist)……………………………………………………………………………………………59

[6.1.2 Manager](#Manager)…………………………………………………………………………………………………60

6.2 [Juint Testing](#JuintTesting)……………………………………………………………………………………………………………………61

[6.2.1 set up](#setup)…………………………………………………………………………………………………………63 6.2.2 [Testing Functionality](#testingFunctions) ........................................................................................64

[6.2.3 test results](#testresults)………………………………………………………………………………………………………65

[Reference: ............................................................................................................](#refrences)

[CHAPTER 1 INTRODUCTION](#content)

* 1. **[Purpose of the project](#content)**

The purpose of this project is to provide an application for clinics of small to medium size that will provide the core necessary functions the establishment would need without any over the board capabilities that won’t be utilized furthermore providing just the needed functions will also cut down on cost of development therefore making a cheaper costing application that will sell for less and when needed can be upgraded easily.

**[1.2 Purpose Of This Document](#content)**

This document is intended to provide all the software engineering details of the Icare system from the conceptual and technical aspects of the application.

**[1.3 Overview Of This Document](#content)**

This report divided into six chapters. Chapter 1 introduces the main idea of the project, describes the purpose of this project/document, reviews of existing systems, and describes similar systems to our existing system and background about machine learning and natural language processing.

Chapter 2, system analysis chapter gives a detailed textual description of the system and depicts the types of users of the system, identifies functional and non-functional requirements, covers several UML diagrams, including Use-Case Diagram, Data-Flow Diagram and Context Diagram and provides proposed/alternative solutions.

Chapter 3, design considerations, specifies the design constraints of hardware and software environment, and defines end user characteristics, Also, explains architectural strategies and focus on the algorithm to be used, methodology of development and architectural pattern.

Chapter 4, system design, describes system architecture and program flow and detailed component description.

Chapter 5, this chapter describes the implementation of Icare system. It describes the tools and way to implement the system.

Chapter 6, This chapter describes the validation of Icare system using Juint unit testing Framework.

[CHAPTER 2 SYSTEM ANALYSIS](#content)

[**2.1** **System Analysis**](#content)

This chapter presents the analysis of the Icare system. It starts by describing the project scope, then discusses the types of users who are going to deal with the system, it lists functional and nonfunctional requirements of the system and finally context level, data flow and use case diagrams.

[**2.2** **Project Scope**](#content)

The scope of the project is to design a desktop application that can provide the core functions in a lightweight and cost-effective way for small to medium size clinics.

[**2.3** **System Requirements**](#content)

This Section will provide all the requirements for the project from the functional to the non-functional.

Firstly, what are differences between Functional and Non-functional requirements :

* Functional requirements explain how the system must work, while
* Non-functional requirements explain how the system should perform.

[**2.4** **Functional Requirements**](#content)

This section will contain all the functional requirements for all the users of the application.

* + - 1. **[Login Screen:](#content)**

Graphical user interface, application

Description automatically generatedThe login will be the first interface the user interacts with when the application will start, there will be two inputs username and password , and two buttons login and clear.

* + 1. **Functional Requirements :**
       1. **Req001:**

User clicks sign without entering any information

**Req001a:** Error text will appear at top of screen

**Req001b:** Error text will be in red

**Req001c:** error message will be “No inputs given please enter username and password”

* + - 1. **Req002:**

There will be three users at the launch of the application:

1.Owner(admin)

2.Mannger

3.Receptionest

Each user will have a unique ID for login where moreover each of them will have a different prefix of the username and the ID must be longer than 3 therefore it will be prefix which is 3 letters and one number minimum.

**Req002a :**

Owner username must start with “OWN” in capital or small then followed by users unique ID.

**Req002b :**

Manager username must start with “MAN” capital or small followed by the users unique ID.

**Req002c:**

Receptionist username must start with “REP” capital or small followed by the users unique ID.

* + - 1. **Req003** :

User gave incorrect username or username

**Req003a :** Do not enter the system

**Req003b:** Error message will be displayed at top of screen

**Req003c:** Error message will be displayed at top of screen

**Req003d :** Display invalid error message “Invalid Email or Password Please Try again”

* + - 1. **Req004:**

Password must be longer then 3

* + - 1. **Req005:**

Username and password must not be empty

* + - 1. **Req006:**

Clear Button will empty contents of text boxes

* + 1. **Login Graphical user interface (GUI) :**
       1. **Req007:**

Interface on start up

* + - * 1. **Req007a** : Label “Sign in” visible: True
        2. **Req007b** : Label “error Message” visible: False
        3. **Req007c :** Label “Username” visible: True
        4. **Req007d :** Textbox “Enter Username” visible: True
        5. **Req007e :** Label “Password” visible: True
        6. **Req007f :** Textbox “Enter Password” visible: True
        7. **Req007g :** Button “Sign In” visible: True
        8. **Req007h :** Button “Clear” visible: True
  1. Graphical user interface, application

     Description automatically generated[**Owner Screen**](#content)**:**
     1. **Description :**

This is the screen which will appear after user signs in successfully as an Owner , will be the main hub for all their work as it leads to all the other screens that have all there permitted actions.

* + 1. **Functional Requirements:**
    2. **Req008:** welcome message + user’s full name
    3. **Req009:** Show number of patient on the system
    4. **Reg0010:** Shownumber of doctors on the system
    5. **Reg011:** Show number of employees on the system
    6. **Reg012:** Pressing "log out" button at any time should close the page and log the time the user logged out and open the log in page
    7. **Reg013:** Pressing "Refresh stats" to connection with database
    8. **Reg014:**Menu tap edit account
    9. **Reg015:**Menu tap doctors
    10. **Reg016:**Menu tap employees
    11. **Owner Graphical user interface (GUI) :**
    12. **Reg017:**

1. Label “welcome massage + username” visible: True
2. Label “number of patience” visible: True
3. Label “number of patients in the system” visible: True
4. Label “number of doctors” visible: True
5. Label “number of doctors in the system” visible: True
6. Label “number of employees” visible: True
7. Label “number of employees in the system” visible: True
8. Button “log out” visible: True
9. Button “refresh stats” visible: True
10. Menu tap “edit account” visible: True
11. Menu tap “doctors” visible: True
12. Menu tap “employees” visible: True
    * 1. Graphical user interface, website

         Description automatically generated**Change name screen :**
      2. **Reg018:** No fields can be empty
      3. **Reg018a:** if detected any field is empty when add button is pressed

display error message “Empty TextField Detected”

* + 1. **Req019 :**Owner username must start with “OWN” in capital or small then followed by users unique ID
    2. **Reg020:** Full name must not contain any numbers
    3. **Reg021a:** if detected number in Full name field when add button pressed display error message “Full-name Cannot Contain Number”
    4. **Reg022 :** Pressing "Apply change" to save changes to database
    5. **Change name Graphical user interface (GUI) :**

**Reg023:**

1. Menu tap “change name” visible: True
2. Menu tap “change username” visible: True
3. Menu tap “change password” visible: True
4. Label “Enter username” visible: True
5. textbox visible: True
6. Label “Error message” visible: False
7. Label “Enter new full name” visible: True
8. textbox visible: True
9. Button “apply change” visible: True
   * 1. Graphical user interface, website

        Description automatically generated**Change username screen:**
     2. **Reg024:** No fields can be empty
     3. **Reg024a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”
     4. **Req025 :**Owner username must start with “OWN” in capital or small then followed by users unique ID
     5. **Reg026 :** Pressing "Apply change" to save in database
     6. **Change username Graphical user interface (GUI) :**

**Reg027:**

1. Menu tap “change name” visible: True
2. Menu tap “change username” visible: True
3. Menu tap “change password” visible: True
4. Label “Enter old username” visible: True
5. textbox visible: True
6. Label “Error message” visible: False
7. Label “Enter new username” visible: True
8. textbox visible: True
9. Button “apply change” visible: True
   * 1. Graphical user interface, website

        Description automatically generated**Change password screen:**
     2. **Reg028:** No fields can be empty
     3. **Reg028a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”
     4. **Req029 :**Owner username must start with “OWN” in capital or small then followed by users unique ID
     5. **Reg030:** Password must be longer then 3
     6. **Reg031:** Pressing "Apply change" to save in database
     7. **Change password Graphical user interface (GUI) :**

**Reg032:**

1. Menu tap “change name” visible: True
2. Menu tap “change username” visible: True
3. Menu tap “change password” visible: True
4. Label “Enter username” visible: True
5. textbox visible: True
6. Label “Error message” visible: False
7. Label “Enter new password” visible: True
8. textbox visible: True
9. Button “apply change” visible: True
   * 1. Graphical user interface

        Description automatically generated**Add manager screen:**
     2. **Reg032:** No fields can be empty

**Reg032a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”

* + 1. **Reg033:** Full name must not contain any numbers

**Reg033a:** if detected number in Full name field when add button is pressed display error message “Full-name Cannot Contain Number”

* + 1. **Reg034:** Manager username must start with “MAN” capital or small followed by the users unique ID
    2. **Reg035:** Password must be longer then 3
    3. **Reg036:** ID must be smaller than 2147483646

**Req036a:** if detected ID is bigger than 2147483646 when add button is pressed display error message “ID Is Too Big”

* + 1. **Req037:** If Manager added successfully display message “Manager Added Successfully” in blue

**Req037a:** If Manager was not added for any reason display error message “Manager Wasn’t Added Successfully” in red

* + 1. **Reg038:** Clear Button will empty contents of text boxes
    2. **Reg039:** Menu tap add manager
    3. **Reg040:** Menu tap edit manager username
    4. **Reg041:** Menu tap edit manager password
    5. **Reg042:** Menu tap delete manager
    6. **Add manager Graphical user interface (GUI) :**

**Reg043:**

1. Menu tap “add manager” visible: True
2. Menu tap “edit manager username” visible: True
3. Menu tap “edit manager password” visible: True
4. Menu tap “delete manager” visible: True
5. Label “full name” visible: True
6. textbox visible: True
7. Label “Error message” visible: False
8. Label “username” visible: True
9. textbox visible: True
10. Label “password” visible: True
11. textbox visible: True
12. Label “ID” visible: True
13. textbox visible: True
14. Button “add” visible: True
15. Button “clear” visible: True
    * 1. Graphical user interface, website

         Description automatically generated**Edit manager username screen:**
      2. **Reg044:** No fields can be empty

**Reg044a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”

* + 1. **Reg045:** Manager username must start with “MAN” capital or small followed by the users unique ID
    2. **Reg046:** Pressing "Apply change" to save in database
    3. **Edit manager username screen Graphical user interface (GUI) :**

**Reg047:**

1. Menu tap “add manager” visible: True
2. Menu tap “edit manager username” visible: True
3. Menu tap “edit manager password” visible: True
4. Menu tap “delete manager” visible: True
5. Label “enter manager old username” visible: True
6. textbox visible: True
7. Label “Error message” visible: False
8. Label “enter new username” visible: True
9. textbox visible: True
10. Button “apply change” visible: True
    * 1. Graphical user interface, website

         Description automatically generated**Edit manager password screen:**
      2. **Reg048:** No fields can be empty

**Reg048a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”

* + 1. **Reg049:** Manager username must start with “MAN” capital or small followed by the users unique ID
    2. **Reg050:** Password must be longer then 3
    3. **Reg051:** Pressing "Apply change" to save in database
    4. **Edit manager password Graphical user interface (GUI) :**

**Reg052:**

1. Menu tap “add manager” visible: True
2. Menu tap “edit manager username” visible: True
3. Menu tap “edit manager password” visible: True
4. Menu tap “delete manager” visible: True
5. Label “enter manager username” visible: True
6. textbox visible: True
7. Label “Error message” visible: False
8. Label “Enter new password” visible: True
9. textbox visible: True
10. Button “apply change” visible: True
    * 1. Graphical user interface

         Description automatically generated**Delete manager screen:**
      2. **Reg053:** No fields can be empty

**Reg053a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”

* + 1. **Reg054:** Manager username must start with “MAN” capital or small followed by the users unique ID
    2. **Reg055:** Pressing "Apply change" to save in database
    3. **Delete manager Graphical user interface (GUI) :**

**Reg056:**

1. Menu tap “add manager” visible: True
2. Menu tap “edit manager username” visible: True
3. Menu tap “edit manager password” visible: True
4. Menu tap “delete manager” visible: True
5. Label “enter manager username to delete” visible: True
6. textbox visible: True
7. Label “Error message” visible: False
8. Button “apply change” visible: True
   1. Graphical user interface, application

      Description automatically generated[**Manger Screen**](#content)**:**
      1. Description :

This is the screen which will appear after user signs in successfully as a manager , will be the main hub for all their work as it leads to all the other screens that have all there permitted actions.

* + 1. **Functional Requirements:**

**Graphical user interface, application

Description automatically generated**

* + 1. **Req057:**Show welcome in the name of the manager
    2. **Req058:**On the first line Show number of doctors in the system
    3. **Req059:**On the second Show number of patients in the system
    4. **Req060**:On the Third line show number of Receptionist in the system
    5. **Req061**: Pressing "log out" button at any time should close the page and log the time the user logged out and open the log in page
    6. **Req062:**Pressing "Refresh stats" to connection with database
    7. **Req063:** Menu tab will be displayed at the top bar to navigate to edit doctor screen
    8. **Req064**: Menu tab will be displayed at the top bar to navigate to edit Receptionist screen
    9. **Manger main screen – Graphical User Interface**

**Req065:**

1. Label "number of doctor" visible: True.
2. - label " number of doctor in system " visible :true
3. Label " number of patients " visible: True.
4. - Label " number of patients in system " visible: True.
5. - Label " number of Receptionist " visible: True.
6. -Lebel " number of Receptionist in system" " visible: True.
7. - Button "log out" enabled: True.
8. - Button "Refresh stats" enabled: True.
   * 1. **Graphical user interface, website

        Description automatically generated Doctor – Graphical Interface:**
     2. **Req066:** In the first line enter id doctor
     3. **Req067:** Enter full name of doctor
     4. **Req068:** Enter field
     5. **Req069:** Pressing "clear " to clear the data
     6. **Req070:** Pressing "add" to add the doctors
     7. **Req071:** Menu tab will be displayed at the top bar to navigate to add doctor screen
     8. **Req072:** Menu tab will be displayed at the top bar to navigate to Delete doctor screen

**Req0073:**

1. **-** Label "Enter ID" visible: True.
2. - Textbox " Enter ID "", enabled :True, visible :true
3. - Label " Enter Full Name " visible: True.
4. - Textbox " Enter Full Name " ", enabled :True , visible: True.
5. - Label " Enter field " visible: True.
6. Textbox" Enter field " ", enabled :True, visible: True.
7. - Button "clear" enabled: True.
8. - Button "Add" enabled: True.
   * 1. **Req074:**Enter Id of doctor to delete the Appointment Booking
     2. **Req075:** Pressing "Delete " to delete the doctor
     3. **Graphical user interface, application

        Description automatically generated Delete Doctor – Graphical Interface**

**Req076:**

1. - Label " Enter Id of doctor to delete " visible: True.
2. - Textbox " Enter Id of doctor to delete "", enabled :True, visible :true
3. - Button "Delete" enabled: True.

**Graphical user interface, application

Description automatically generated**

* + 1. **Req077:** In the first line enter full name of Receptionist
    2. **Req078:** Enter the username
    3. **Req079:** Enter Password
    4. **Req080:** Enter ID to add Receptionist
    5. **Req081:** Pressing "Add " to Add the Receptionist
    6. **Req082:** Pressing "Clear " to Clear the Data
    7. **Req083:** Menu tab will be displayed at the top bar to navigate to Add Receptionist screen
    8. **Req084:** Menu tab will be displayed at the top bar to navigate to Edit Receptionist username screen
    9. **Req085:** Menu tab will be displayed at the top bar to navigate to Edit Receptionist password screen
    10. **Req086:** Menu tab will be displayed at the top bar to navigate to Delete Receptionist screen
    11. **Receptionist – Graphical Interface**

**Req087:**

1. **-** Label "Full name" visible: True.
2. - Textbox " Full name "", enabled :True, visible :true
3. - Label " Username " visible: True.
4. - Textbox " Username " ", enabled :True , visible: True.
5. - Label " Password" visible: True.
6. Textbox" Password " ", enabled :True, visible: True.
7. - Button "clear" enabled: True.
8. - Button "Add" enabled: True.

**Graphical user interface, website

Description automatically generated**

* + 1. **Req088:** In the first Enter manger old username
    2. **Req089:** In the second line Enter New Username
    3. **Req090**: Pressing "Apply change" to save changes to database
    4. **Graphical user interface, website

       Description automatically generatedEdit Receptionist – Graphical Interface**

**Req091:**

1. **-** Label " Enter manger old username " visible: True.
2. - Textbox " Enter manger old username "", enabled :True, visible :true
3. - Label " Enter New username " visible: True.
4. - Textbox " Enter New username " ", enabled :True , visible: True.
5. - Button "Apply change" enabled: True.
   * 1. **Req092**: In the first Enter Receptionist username
     2. **Req093:** In the second line Enter New Password
     3. **Req094:** Pressing "Apply change" to save changes to database
     4. **Graphical user interface, text, application

        Description automatically generated Delete Receptionist – Graphical Interface**

**Req095:**

1. **-** Label " Enter Receptionist username " visible: True.
2. - Textbox " Enter Receptionist username "", enabled :True, visible :true
3. - Label " Enter New Password " visible: True.
4. - Textbox " Enter New Password " ", enabled :True , visible: True.
5. - Button "Apply change" enabled: True
   * 1. **Req096:** Enter Receptionist Username
     2. **Req097:** Pressing "Apply change" to save changes to database
     3. **Receptionist – Graphical Interface**

**Req098:**

1. - Label " Receptionist Username " visible: True.
2. - Textbox " Receptionist Username "", enabled :True, visible :true
3. - Button "Apply change" enabled: True.
   1. **Graphical user interface

      Description automatically generated**[**Receptionist** **Screen** :](#content)
      1. **Description:**

This is the screen which will appear after user signs in successfully as a receptionist, will be the main hub for all their work as it leads to all the other screens that have all there permitted actions.

* + 1. **Functional Requirements:**
       1. **Re099:** No fields can be empty

**Re099a:** if detected any field is empty when add button is pressed display error message “Empty TextField Detected”

* + - 1. **Re100:** Full name must not contain any Numbers

**Re100a:** if detected number in Full name field when add button is pressed display error message “Full-name Cannot Contain Number”

* + - 1. **Re101:** Number must contain exactly 10 numbers

**Re101a:** if detected number of numbers not equal to 10 when add button is pressed display error message “Number Must contain 10 numbers given number is + (number of numbers given by user)”

**Re101b:** Number must start with 05

**Re101c:** if detected number does not start with 05 when add button is pressed display error message “Number Must Start with 05”

**Re101d:** number must be smaller than 2147483646

**Re101e:** if detected number is bigger than 2147483646 when add button is pressed display error message “Number Is Too Big”

* + - 1. **Req102:**ID must be smaller than 2147483646

**Req102a:** if detected ID is bigger than 2147483646 when add button is pressed display error message “ID Is Too Big”

* + - 1. **Req103:** If patient added successfully display message “Patient Added Successfully” in blue

**Req103a:** if patient was not added for any reason display error message “Patient Wasn’t Added Successfully” in red

* + - 1. **Req104:** Combo box will contain all the available doctors in the clinic which the patient will be assigned to
      2. **Req105:** logout button will return user to login screen and log the time in which they logged out
      3. **Req106:** Refresh button would refresh the database so if any changes happened user can see them
      4. **Re107:** Menu tab will be displayed on top bar to allow access to doctor print option which will print to a file all the information about the doctors
      5. **Re108:** Menu tab will be displayed on top bar to allow access printing todays Appointments

**Re108a**:If no Appointments available on the same day Error message “No Appointments Have Been Booked Today”

**Re108b**: If appoints booked on the day a file will be saved in the Prints directory and opened on the user’s desktop.

* + 1. **Receptionist Graphical user interface (GUI) :**
       1. **Req109:**

Interface on start up

* + - * 1. Tab “Main Screen” visible : True
        2. Tab “Search for Patient” Visible: True
        3. : Label “ID” visible: True
        4. :Textbox ”Enter Patient ID” visible: True
        5. **:** Label “Error message” visible: False
        6. **:** Label “Full Name” visible: True
        7. **:** Textbox “Enter Patient Name” visible: True
        8. **:** Label “Number” visible: True
        9. **:**Textbox “Enter Patient Phone Number” visible: True
        10. **:** Label “Age” visible: True
        11. **:** Textbox “Enter Patient Age” visible: True
        12. : label “Choose Doctor” visible: True
        13. **:** Combo Box visible : True
        14. **:** Button “Log Out” visible: True
        15. **:** Button “Refresh” visible: True
    1. **Search screen :**

Graphical user interface, application

Description automatically generatedThis screen can be accessed by a tab on the top of the user’s interface which will display the following screen that will allow the user to search for a patient by their ID.

* + - 1. **Re110 :** TextField must not be empty

**Re110a:** if TextField detected empty display error message “ Empty TextField Detected”

**Re110b:** ID must be valid i.e., a patient must have that ID

**Re110c:** if given an invalid ID display error message “Patient Wasn’t Found”

**Re110d:** if ID is valid display message “ Patient Found” in blue and display that patient’s information in the name, age, and number

* + 1. **Login Graphical user interface (GUI) :**
       1. **Req111:**

Interface when user selects Search for Patient Tab

1. Label "Enter Patiant ID:” Visable :True
2. Textbox Visable :True
3. Button "Search " Visible :True
4. Label “Error Message” Visible: True
5. Label ”name” Visible:True
6. Label ”Age ” Visible :True
7. Label ”Number ” Visible :True
8. Label “Users Name” Visible: False
9. Label “Users Age” Visible: False
10. Label “Users Number” Visible: False
    1. **Graphical user interface, website

       Description automatically generatedBook Appointment Screen:**
       1. **Description :**

This screen will be accessed through the first screen via menu tab which will open the above interface

* + 1. **Functional requirements :**
    2. **Req112:** The Patient ID will follow same Requirements as Req033
    3. **Req113:** Combo box will display all the doctors’ names to book appointment with
    4. **Req114:** Date picker will allow the user to select the date for the appointment
    5. **Req115:** Spinner will allow the user to select the time for the appointment

**Req115a:** the hours will only display work hours options (8-16) in the 24 hours system

* + 1. **Req116:** If doctor booked on that time and date no other appointments can be booked on top

**Req116a:** if user attempts to book an already booked appointment display error message “Appointment Already Booked With The Same Doctor And Time”

* + 1. **Req117:** User can book up to maximum 3 appointments per day

**Req117a:** If user already booked 3 appointments and receptionist tries to book on more on that same date display error message “Appointment Limit Of 3 A Day Reached”

* + 1. **Book Appointment Graphical user interface (GUI) :**
    2. **Req118: Screen after user clicks book appointment menu tab**

1. Label “Error Message” Visible =False
2. Label “Patient ID” Visible =True
3. Text Box “Patient ID”
4. Label “Doctor” Visible =True
5. Combo box for doctors
6. Label “Date” Visible =True
7. Date Picker Visible =True
8. Label “Time(Hours)” Visible =True
9. Spinner to choose time Visible =True
10. Button “Make Appointment “ Visible =True

* 1. **[Non-functional Requirements](#content)**

Non-functional requirements are specifying the system’s quality attributes and its criteria regarding look and feel, usability, security, performance requirements. These requirements and others have carefully described in the following points:

* + 1. **[Look and feel requirements](#content) :**
* The Application should give clear and straightforward feedback to the user specially when performing CRUD operations error messages should point at the exact problem furthermore if operation successful show appropriate message.
  + 1. [**Usability requirements :**](#content)
* The application should be easy to use and have clear interfaces to make sure the user does not get confused or frustrated.
* The application should have an easy learning curve to save resources when it comes to training new users.
  + 1. **Security requirements :**
* Users should not have the ability to see interfaces they do not have permission for example receptionist should not have the ability to complete the operations that the manger has the rights to.
* The application should not permit error causing inputs to be added to the databases and keep the integrity of the system high.
  + 1. [**Performance requirements :**](#content)
* The application should be stable and fast as not much high computation is needed therefore user will expect fast operations and results.

[CHAPTER 3 DESIGN CONSIDERATIONS](#content)

3. [**Design Considerations :**](#content)

This chapter presents the design constraints of Icare system, starting by defining the design constraints of software, then describing end user characteristics, discussing the methodology to develop Icare system and the architectural pattern.

[**3.1** **Design Constraints**](#content)

**[3.1.1 Software Environment](#content)**

To implement our system, we used the following software environments:

* **JavaFX:**

JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. It is a collaborative effort by many individuals and companies with the goal of producing a modern, efficient, and fully featured toolkit for developing rich client applications.

* **MySQL:**

MySQL relational database system used to organize data into one or more data tables in which data my be related to each other using relations.

* **Junit:**

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

* **Java:**

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today’s digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well**.**

**[3.1.2 End-User Characteristics](#content)**

The targeted audience of this system are clinic office personnel such as the owner, manger, and receptionist as they will provide services for the doctors and patients.

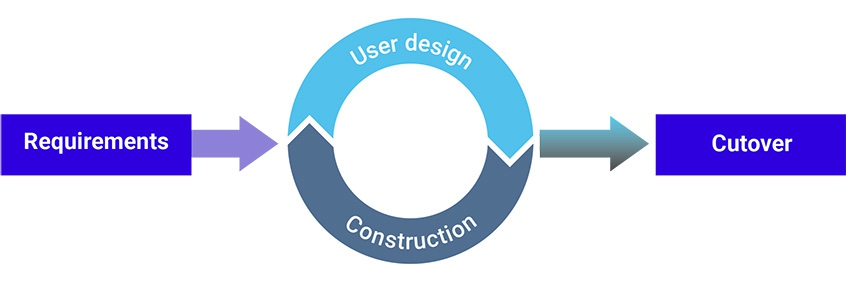
**[3.2 Project Management Strategies :](#content)**

In this project we used the rapid application development methodology :

Rapid application development (RAD) is a condensed development process that produces a high-quality system with low investment costs. Scott Stiner, CEO, and president of UM Technologies, said in Forbes, “This RAD process allows our developers to quickly adjust to shifting requirements in a fast-paced and constantly changing market.” The ability to quickly adjust is what allows such a low investment cost.

The rapid application development method contains four phases: requirements planning, user design, construction, and cutover. The user design and construction phases repeat until the user confirms that the product meets all requirements.

We used this strategy as it is most effective for projects with a well-defined business objective and a clearly defined user group, but which are not computationally complex. RAD is especially useful for small to medium projects that are time sensitive.



**[3.2.3 Development Method :](#content)**

Our development stages where as follows :

* + - **Analysis**: In this phase, we must specify the requirements of the proposed system such as:
* The key features and functionalities.
* Tools and dataset that is help us to build the system.
  + - **Design**: In this phase, we:
* Draw all the diagrams that meet the requirements.
* Design the database and relations
  + - **Implementation:** In implementation phase, we followed a sequence of steps such as:
* Develop top down from the highest privilege user (the owner) till the least privileged user
* Adding the login features with the logs used to track sign ins and outs
  + - **Testing:**
      * After each user is implemented all their functionality is tested before moving onto the next user.
      * After testing each user, we test the pervious users as well to make sure no problems have affected them.

**[3.2.4 Gannet Chart :](#content)**

A picture containing timeline

Description automatically generatedA Gantt chart is a project management tool assisting in the planning and scheduling of projects of all sizes, although they are particularly useful for simplifying complex projects. Project management timelines and tasks are converted into a horizontal bar chart, showing start and end dates, as well as dependencies, scheduling, and deadlines, including how much of the task is completed per stage and who is the task owner. This is useful to keep tasks on track when there is a large team and multiple stakeholders when the scope changes.

[CHAPTER 4 SYSTEM DESIGN](#content)

**[4](#content)**[.](#content) **[System Design](#content)**

This chapter presents is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

UML : Unified Modeling language is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system.

**[4.1.1 Architectural patterns :](#content)**

architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. Architectural patterns are similar to software design pattern but have a broader scope.

**[4.1.2 The Model-View-Controller Pattern :](#content)**

Diagram

Description automatically generatedThe Model View Controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

**[4.1.3 The Component and Link Diagram :](#content)**

Diagram

Description automatically generatedA link component is a reusable link definition that can be used to describe how various values returned by one operation can be used as input for other operations. In this way, links provide a known relationship and traversal mechanism between the operations.

**[4.1.4 Class Diagram :](#content)**

A class diagram is a type of diagram and part of a unified modeling language (UML) that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes.

**Diagram

Description automatically generated4.1.4.1** [**Overall structure class diagram:**](#content)

**[4.1.4.2 DoctorQuries class diagram:](#content)**

A piece of paper with writing

Description automatically generated with low confidence

**[4.1.5 Software Architecture Diagram :](#content)**

Diagram

Description automatically generatedSoftware architecture is the defining and structuring of a solution that meets technical and operational requirements. Software architecture optimizes attributes involving a series of decisions, such as security, performance, and manageability. These decisions ultimately impact application quality, maintenance, performance, and overall success.

**[4.1.6 Layered Architecture Diagram :](#content)**

Layered architectures are said to be the most common and widely used architectural framework in software development. It is also known as an n-tier architecture and describes an architectural pattern composed of several separate horizontal layers that function together as a single unit of software.

Timeline

Description automatically generated

[CHAPTER 5 IMPLEMENTATION](#content)

**5.** **[Implantation:](#content)**

In this chapter we will describe the way we implemented the Icare system , from the start up to how we coded some of the main functionality like logging and singing in for the different users.

**5.1** [**Main Method:**](#content)

The main method Inherits from the Application class to provide the start up to the application, where using the FXMLLoeader we can provide the fxml file we want to show at the start .

The fxml files contain the visual elements and settings whereas there is another class tied to each fxml that is called a controller that’s how this project is laid out.

Text

Description automatically generated

**5.2** [**Login Method:**](#content)

The login method is inside the login Controller class and when the user clicks the login button the method is called, when its called it reads the prefix of the username and compares it to the user prefixes we specified in the **Req002a** and if the prefix matches the specified ones the username and password will be verified if they are incorrect the user will be denied accesses and prompted with a message , if correct the username and time of access will be logged furthermore the system will check if this is the users first login as that will open a extra window so the user has to change the default username they where assigned .

Text

Description automatically generated

**5.3** [**Database Connection:**](#content)

To connect to the MySQL database, we will be making a separate class that will connect to the database , we made it in one class to remove repeating the code in every class that connects to the database therefore making the code cleaner and more efficient.

Text

Description automatically generated

To connect the database, we used the MySQL JDBC Driver file to make the code work.

**5.4** [**Logger Class:**](#content)

The logger class logs all sign ins/outs to achieve that in a well-organized way we initialized three variables that will be used to name the variables based on the time and its format and the last variable is used for obtaining the time which is saved inside the file next to who made the logged action:

Text

Description automatically generated

**5.5** [**Objects and Queries Classes:**](#content)

Every entity in the project will have at least two classes an object class and objectQuries class .

**[5.5.1 The Object class:](#content)**

The object class is your standard OOP Object class containing the main variables of that object and the corresponding constructor and setter and getter classes.

Text

Description automatically generated

**5.5.2** [**The ObjectQuries classes:**](#content)

The objectQuries classes will have a connection variable that will allow it to connect to the database, after successful connection the class has several methods that use Prepared statements that have been set as variables but missing the needed information from the user which will be given by the user at run time to execute the query.

Text

Description automatically generated

After initializing all the prepared statements, we can use them in the functions of the class, in the functions you will see the use of Observable lists they are used for array lists that will change in runtime in javaFX system with a combo box element :

Text

Description automatically generated

**5.6** [**Testing function:**](#content)

Each ObjectQuries class will contain these two methods the setAutoCommitFalse and Rollback method, there used in the testing class to make sure the test case data is not saved in the database to keep the integrity of the data :

Text

Description automatically generated

CHAPTER 6 VALIDATION

**6.** **Validation**

This chapter describes the testing of the Icare system and how the data is stored in the database.

**6.1 Interface testing and database testing :**

The following section will show some of the screens we tested and some of the data present in the database and how its saved in its respected table.

**6.1.1 Receptionist :**

In this interface we add the information of Patient in system, shown in Figure 1 below and show the updated database in Figure 2.

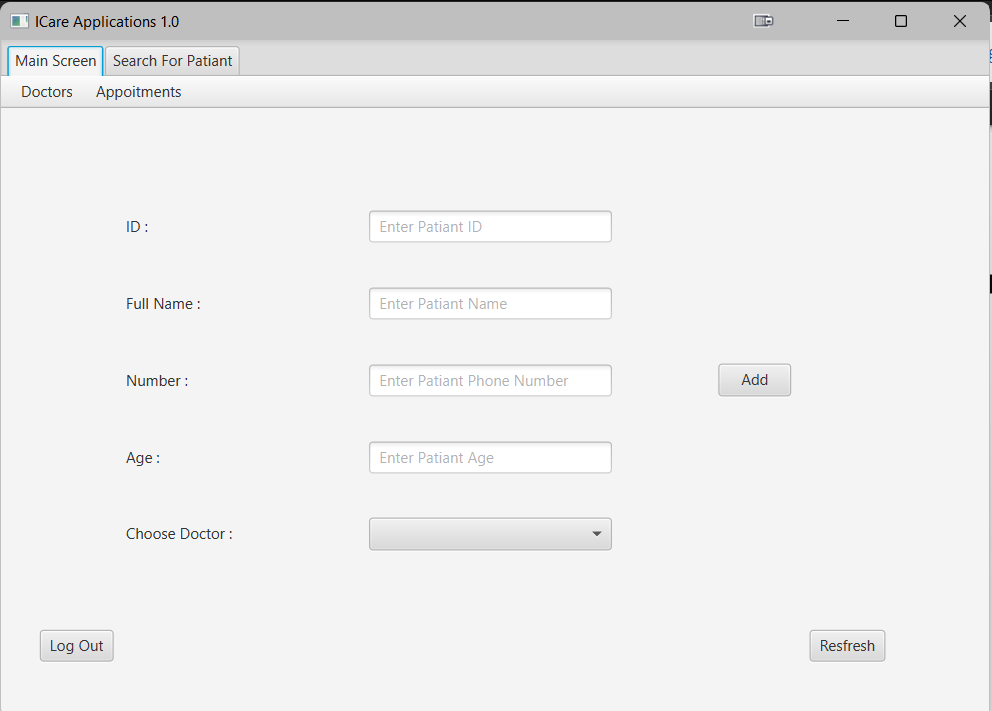


Figure 1: receptionist form interface

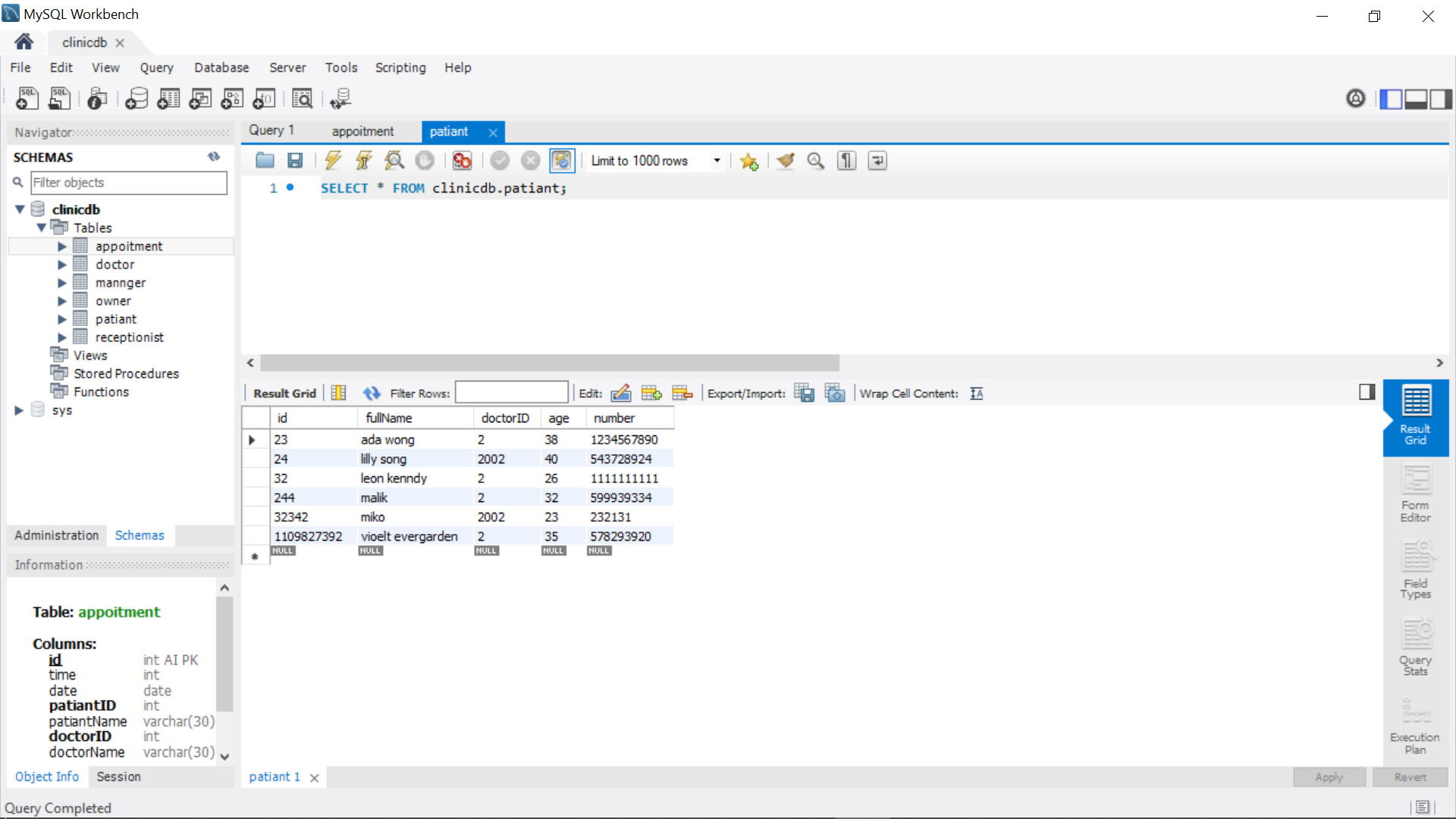


Figure 2: Database for insert Patient information

**6.1.2 Manager :**

Sign in page: this interface allows for manger to sign in to his/her account and check the information validation

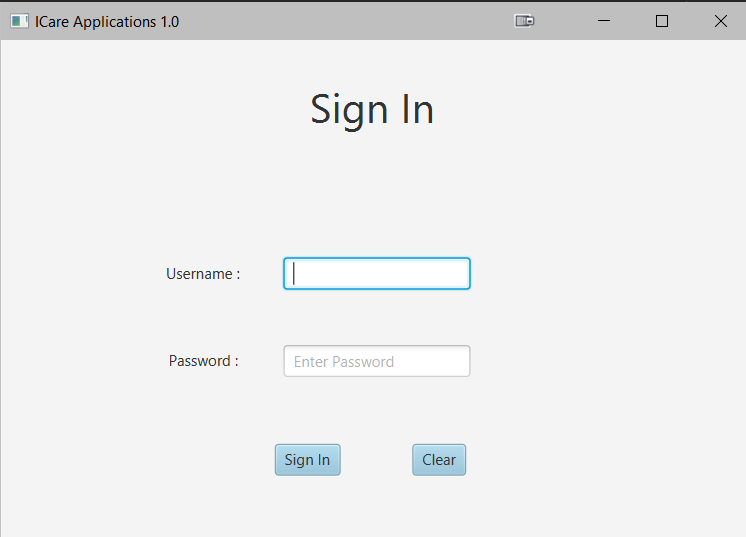


Figure 3: manager sign in Clinic.

• **Edit profile page**: this interface allows for manger to update his/her information , as shown in Figures 4

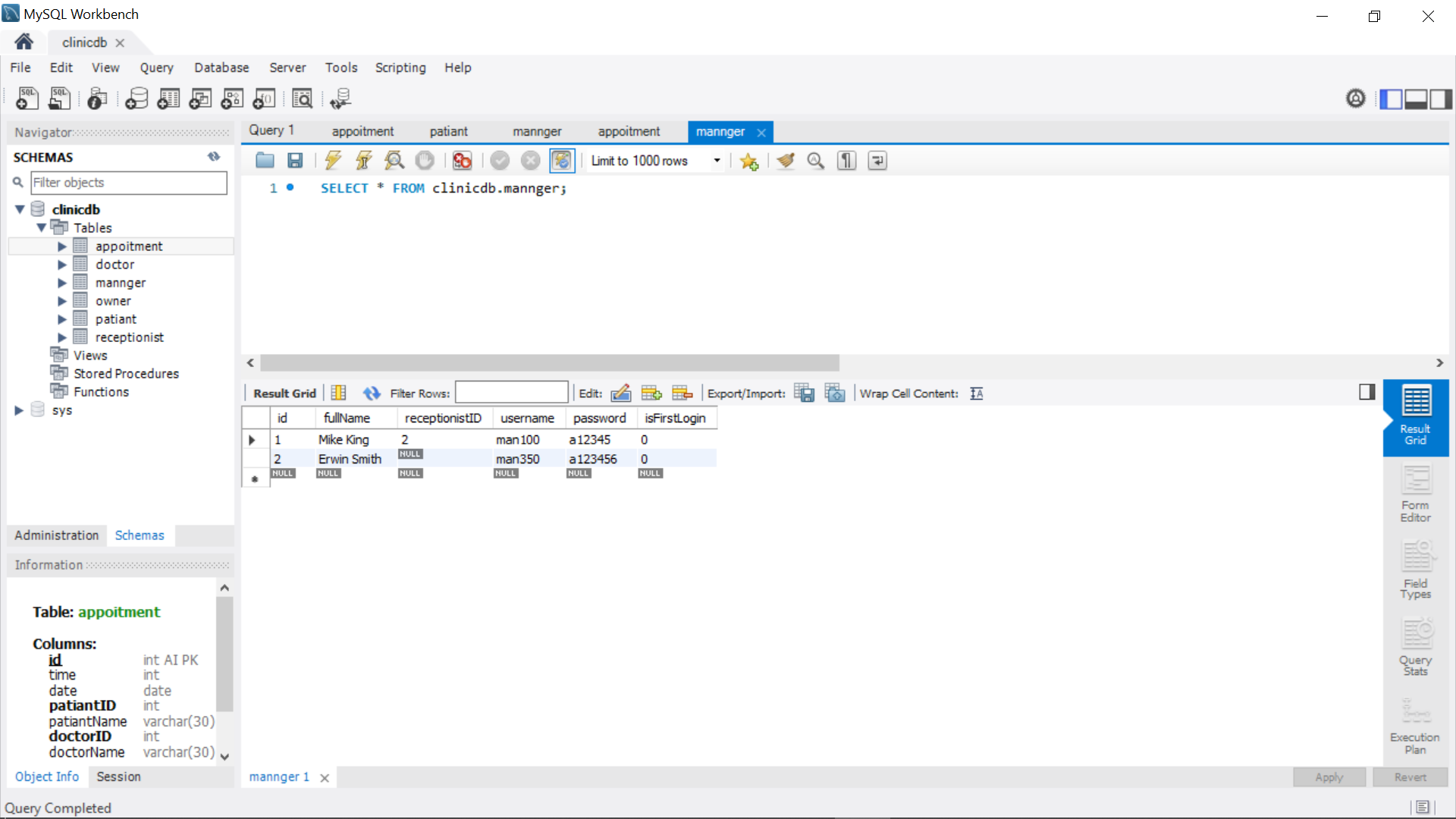
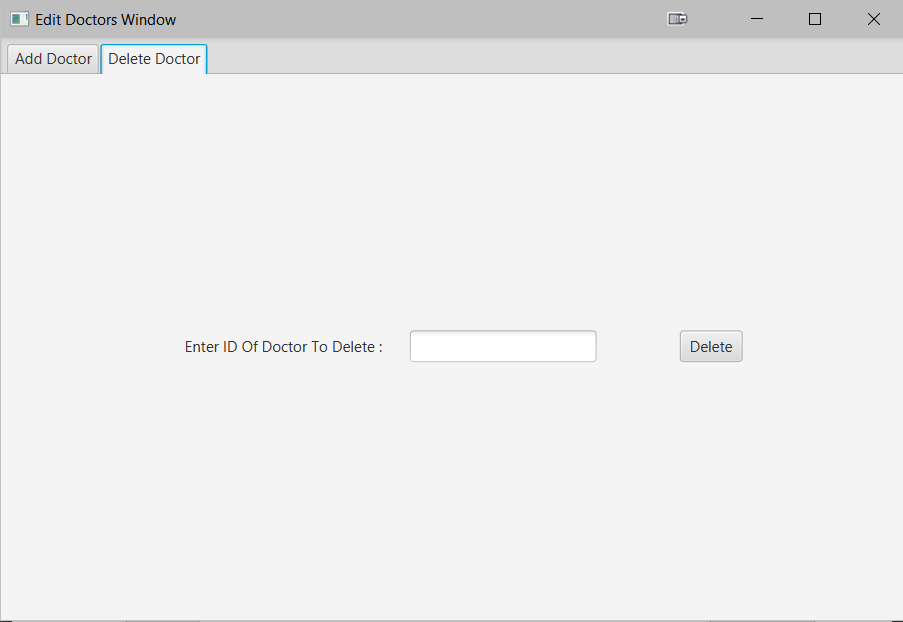


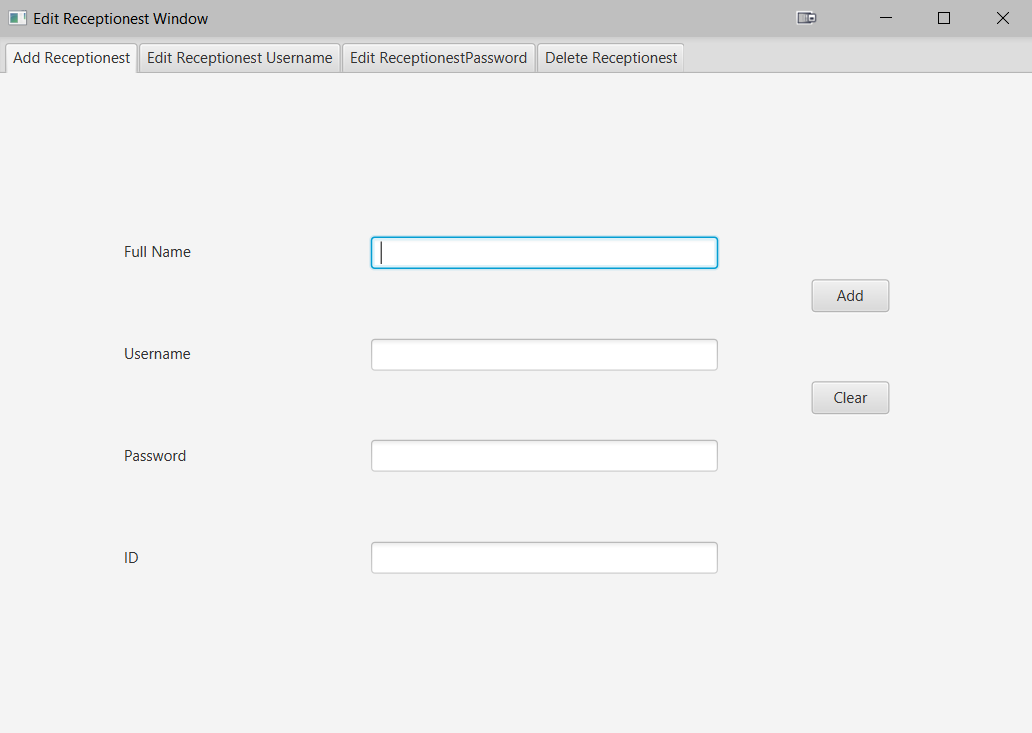
Figure 4: Database edit manager’s profile

• **Delete Doctor** : this interface allows for manger to delete Doctor



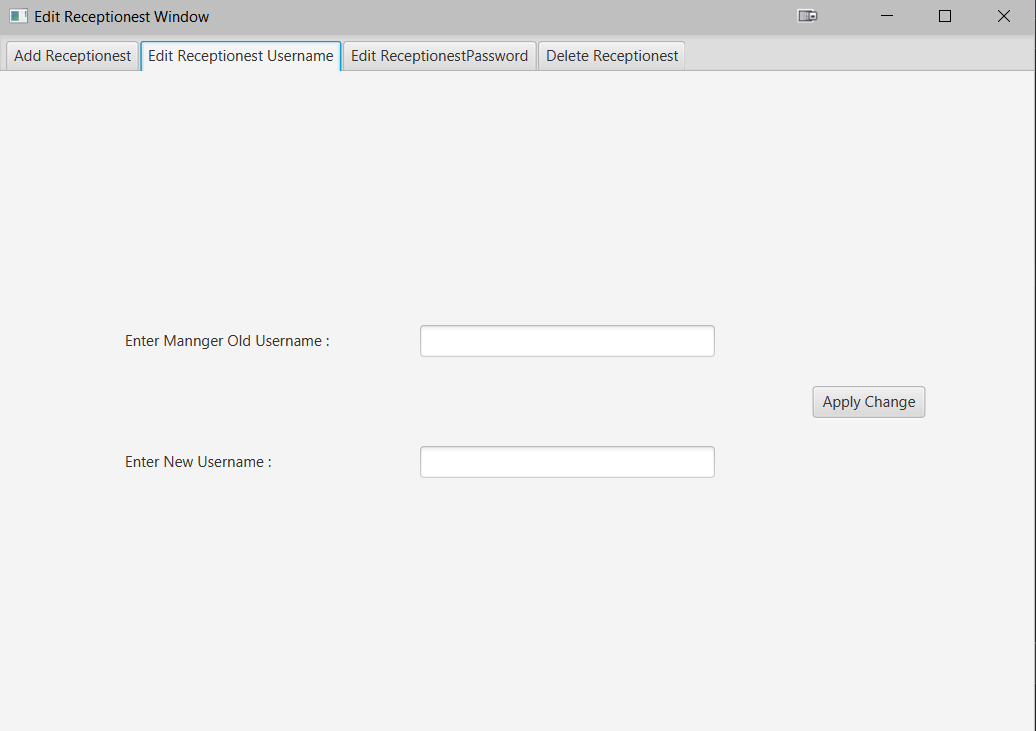
**• Add receptionist**

In this interface we add the information of receptionist in system



**• Edit receptionist**

in this interface we edit the name of receptionist



**6.2 Juint Testing :**

Here we will show some of the test code used to test the system using the Junit Framework.

**6.2.1 set up :**

In the figure below we can see in the teardown and setup we put some code to make sure the test data doesn’t get saved in the database and that’s done by setting autoCommitFalse that will prevent the data from being saved in the database after the test is done we will rollback and that’s essentially recovering the database to its pretest state therefore ensuring the integrity of the database data .



**[6.2.2 Testing Functionality](#content)** [:](#content)

The following code snippets will show some of the tests run and the result of the tests.

The tests follow the same idea as the tests are automatically generated by the IDE however they are made with a fail tag to ensure the developers configure the tests apriority so they can show the proper test results.

The tests firstly print out the name of the test then the test takes in some dummy data that’s used to test that particular situation of use of the test whether it be test nonacceptable data or acceptable data.

Text

Description automatically generated After taking the dummy data we input the expected outcome of the test and how the test should compare them equals, true or false etc. After that the method the test is made to test is called and the returned output is compared to the expected output.

**6.2.3 test results :**

In the final build of the system these where the results of the tests of this class :

Graphical user interface, application

Description automatically generated

[**References :**](#content)

<https://enkonix.com/blog/functional-requirements-vs-non-functional/>

<https://openjfx.io/>

<https://en.wikipedia.org/wiki/JUnit>

<https://en.wikipedia.org/wiki/MySQL>

<https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/>

<https://medium.com/the-andela-way/system-design-in-software-development-f360ce6fcbb9>

<https://www.techopedia.com/definition/3243/unified-modeling-language-uml>

<https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

<https://www.geeksforgeeks.org/mvc-design-pattern/>

<https://www.ibm.com/docs/en/api-connect/10_reserved_instance?topic=api-defining-link-components>

<https://www.techopedia.com/definition/16466/class-diagram>

<https://www.techopedia.com/definition/24596/software-architecture>

<https://www.baeldung.com/cs/layered-architecture>

--end of report--