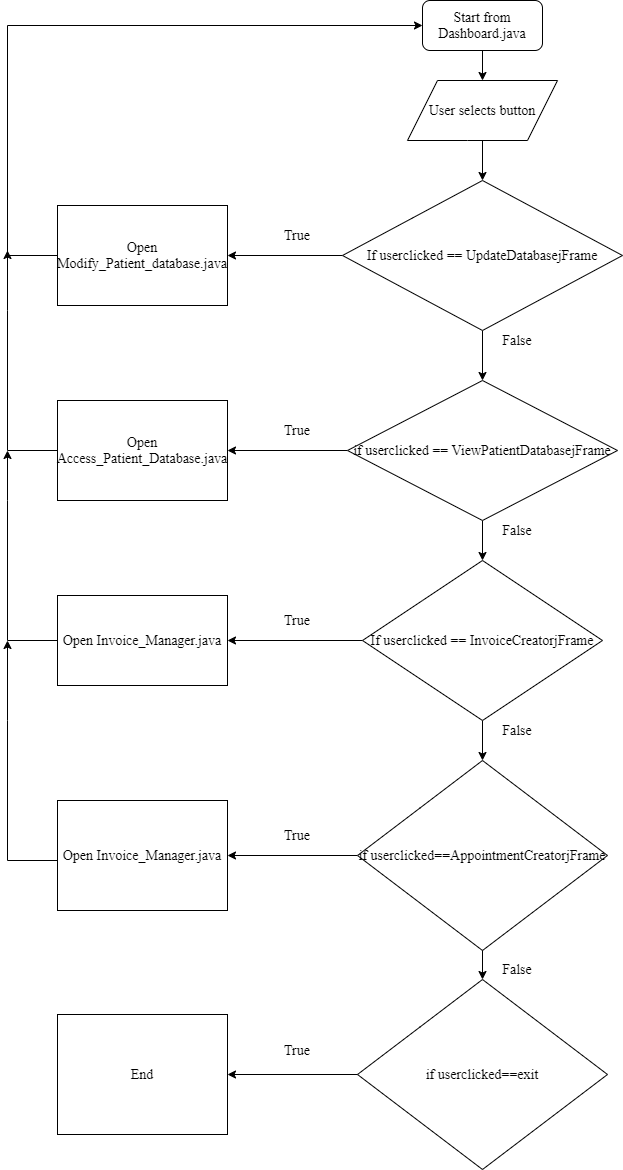
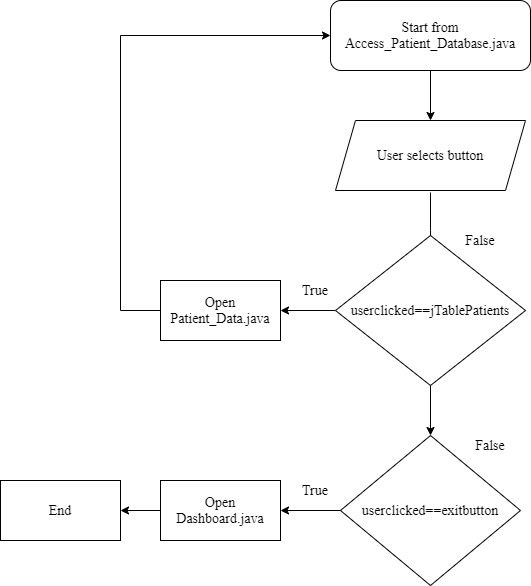
### **(Figure 1.)** Starting the programme

**(Figure 2.)** Dashboard navigation



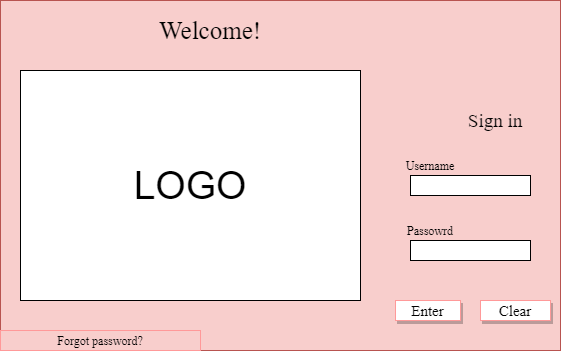
### **(Figure 3.)** Individual patient view



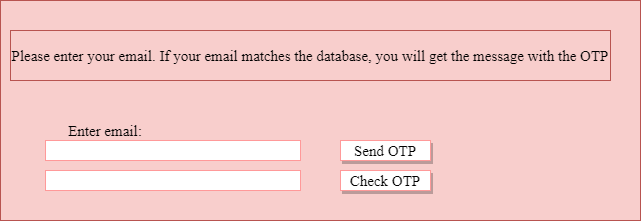
### **User Interface Design**

The following is the program's user interface. The color palette and the logo on the login page were both reviewed and accepted by the client and are a close match of the actual design. Every input field is a jTextField (blank white rectangles), which were incorporated with jButton (white shadowed rectangles) in order to manipulate data from jTextField. In addition, jTable was used to represent data from the MySQL database. In order to avoid clashes and inconsistencies of values which have a small range (Doctor, Sex, etc.), jComboBox and jCheckbox were used.

(**Figure 4.**) Login screen



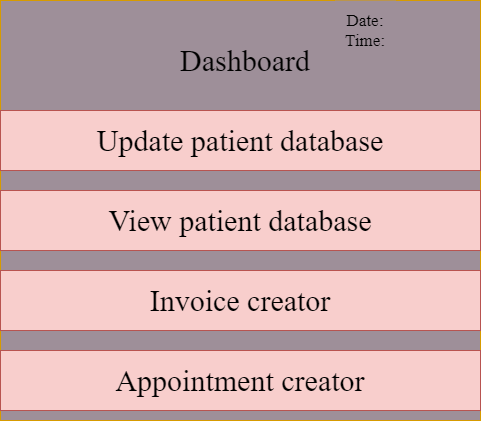
(Figure 5.) OTP request panel



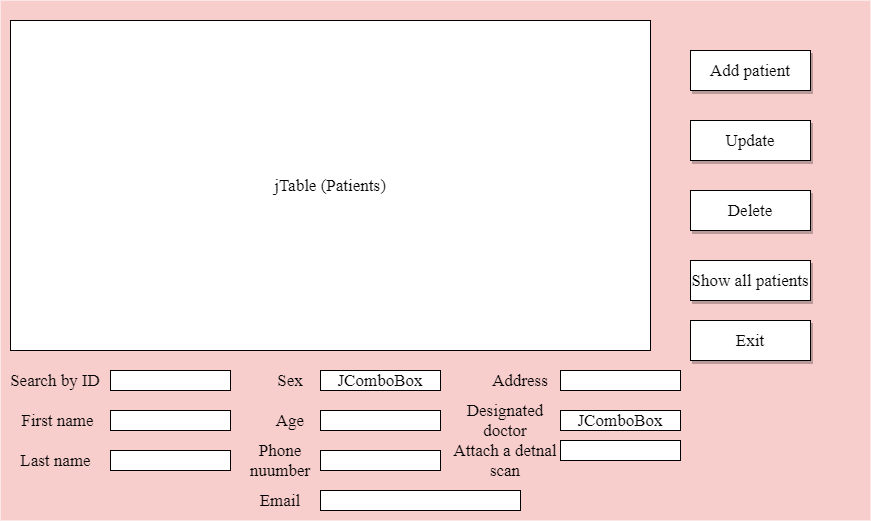
(Figure 6.) Admin data update page



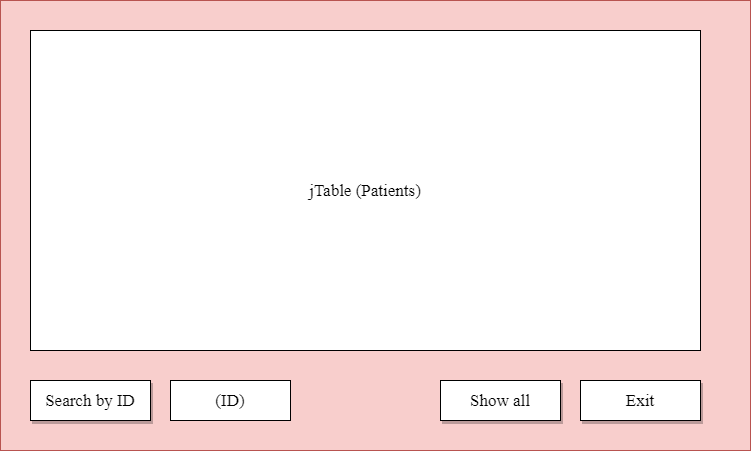
(Figure 7.) Non-edit access to the database



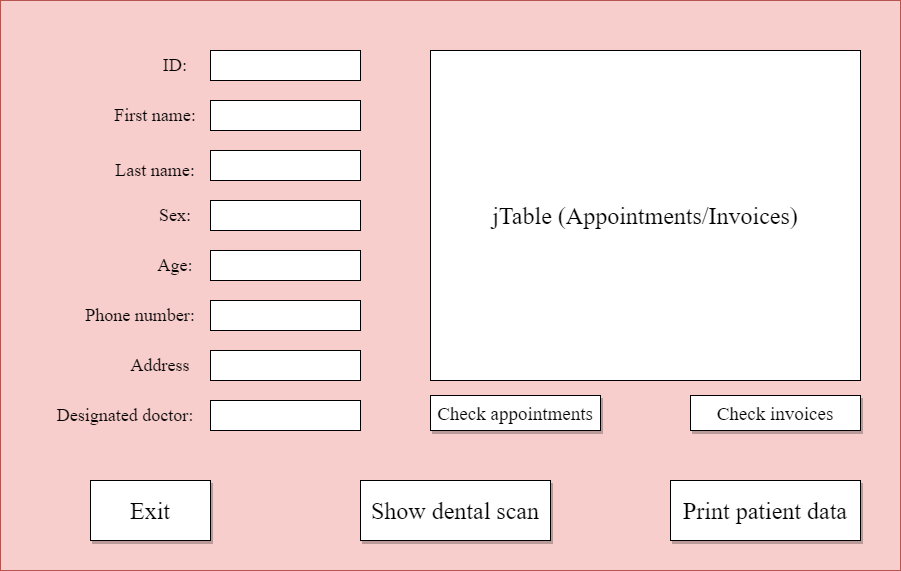
(Figure 8.) Individual user data



(Figure 9.) Invoice manager (Create invoice panel)



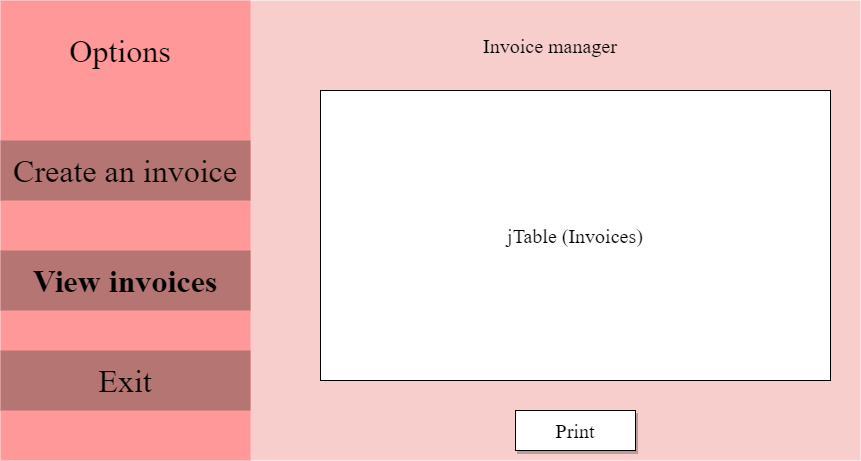
(Figure 10.) Invoice manager (View invoices panel)



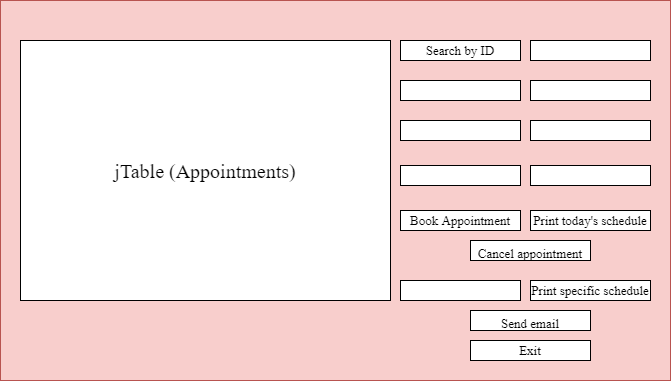
(Figure 11.) Appointment maker

## 

(Figure 12.) Invoice Manager

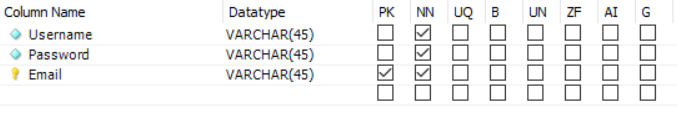


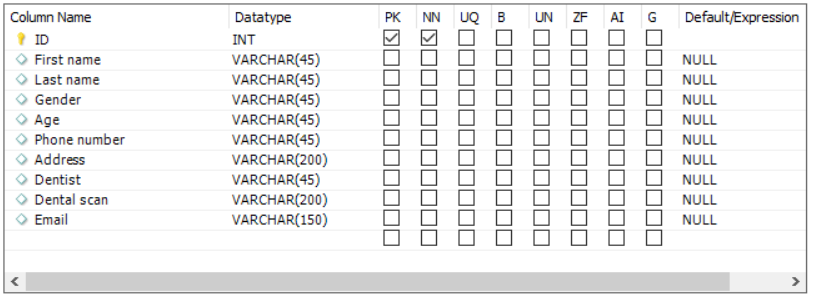
## (Figure 13.) Appointment creator



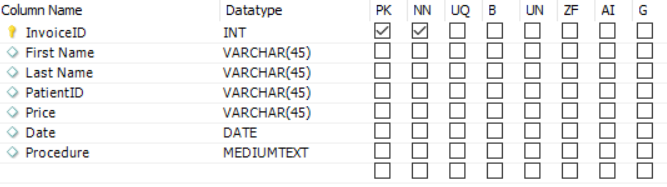
### Database tables

(Figure 12.) User table

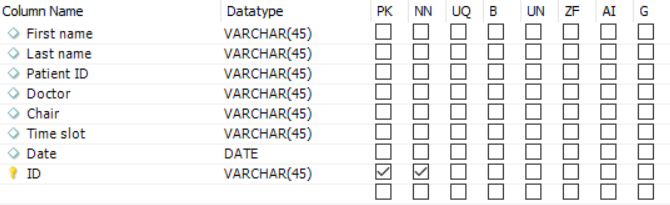


(Figure 13.) Patient table

(Figure 15.) Invoice table

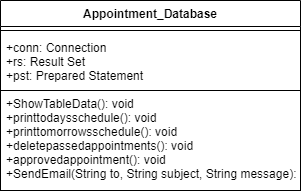
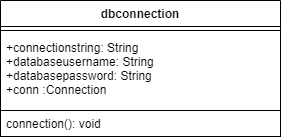
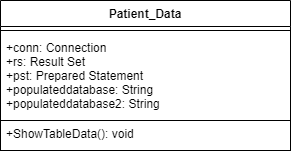
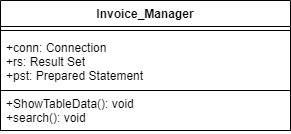
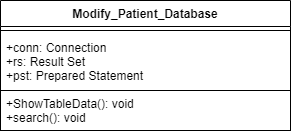
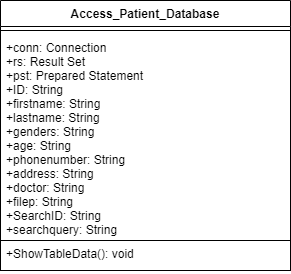
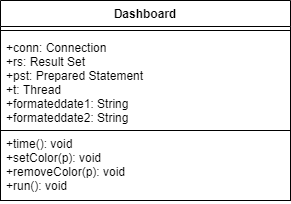
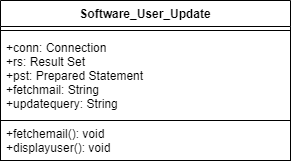
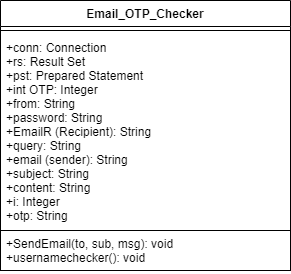
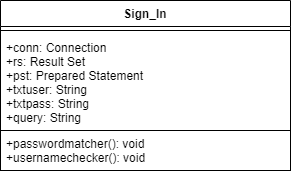


(Figure 16.) Appointment table



All of the column data types have been shown.

### **UML Diagrams**



Word count: 92

**Testing plan**

|  |  |  |
| --- | --- | --- |
| Testing type | Test | Example |
| GUI Test | All of the GUI components run and display successfully | Class Modify\_Database runs and displays all of the GUI components |
| GUI Navigation | The user can easily navigate between jFrames. | Class Dashboard is easily navigable and all of the classes are accessible on both ends (to and from) |
| Confirming that the authentication is accurate | The software starts with the login page. Once the user enters the login credentials, the programme should check whether the information matches. Appropriate responses should also be given |  |
| Ensuring that the OTP system is working | In case the user wants to alter login credentials, the database checks whether the inserted email matches the database. If a match occurs, it should send an 6-digit random code to the user's email. | The class Email\_OTP\_Checker successfully communicates with the email server, allowing for an OTP security system |
| Database connection | Testing the communication of the program with the MySQL Database | Class dbconnection returns connection to respective Classes, enabling access to the database |
| SQL data insertion | All of the values from jTextfield, jComboBox and jCheckbox should be entered and stored in the MySQL Database | Class Modify\_Database successfully inserts and data |
| SQL data deletion | User is able to remove data from the MySQL Database directly from the program | Appointment class gives the option to the user to cancel any appointments. |
| SQL data modification | The user is able to edit specific rows and data entries in case of an error or update. | Modify patient database displays selected row in appropriate jTextfields which the user can then update or modify |
| Individual user profiles | The user is able to access individual user profiles | Patient data class displays individual user information along with upcoming appointments and past invoices. |
| Invoice creator | Invoices are capable of being created from a template with predetermined prices. | Invoice manager class successfully creates invoices from a template, which could then be printed. |
| Invoice storage/access | The invoices should be stored in the database and should be accessible. | Invoice manager stores invoices along with the option of accessing and printing them. |
| Appointment clashes | User is able to successfully create an appointment and is able to prevent any clashes | The Appointments Class ensures that no two identical appointments can be made, prompting the user to change the appointment parameters if the programme finds anything clashing |
| Appointment chronology | Appointment schedule should be displayed in chronological order | Data in jTable from Appointment class is represented chronologically, both in dates and time |
| Printing data from MySQL database | Data extracted from MySQL is able to be printed | ViewOnly Class gives the user the ability to print Patient Data |
| Printing tables from MySQL database | jTables already populated from MySQL are printable | Receipt class gives the ability to print the receipt table. |
| Creating PDFs of both the data and the tables from MySQL | Data and jTables extracted from MySQL are able to be transformed into PDFs | Appointment class gives the ability to create a PDF of the schedule of an appointment in any given future date. |
| Sending reminder emails to patients regarding appointments | The ser is able to extract emails from database and send data to those emails directly from the app | The user can send reminder emails about schedule appointments |