**Section 4 Prototype**

**Aim of the prototype**

The main goal of this prototype is to establish a foundation for the final product. It will also provide the pharmacy with a preview of the final product’s appearance and functionality, whilst also allowing for feedback. The prototype will not be a fully functional program and the feedback from the pharmacy will be crucial for the solutions development, and I will involve and try to adhere to feedback in the post prototype stage.

Features I have included in my prototype:

* Creation of the database to store system data
* Adding patient details into the database – giving onscreen output
* Editing patient details into the database – allowing searching, updating, and deletion of data through SQL queries
* System responding to the user after completing a process – a message box should be displayed
* Creating a GUI to link to the database ensuring navigation to different processes and that processes are fully functional

I have decided to solely focus on patient data entry, updating, and deletion in my prototype as this functionality is very easy to replicate for the rest of the tables. The overall main function of this prototype is to ensure I am able to implement a GUI. As previously mentioned, I will be using Tkinter for the GUI.

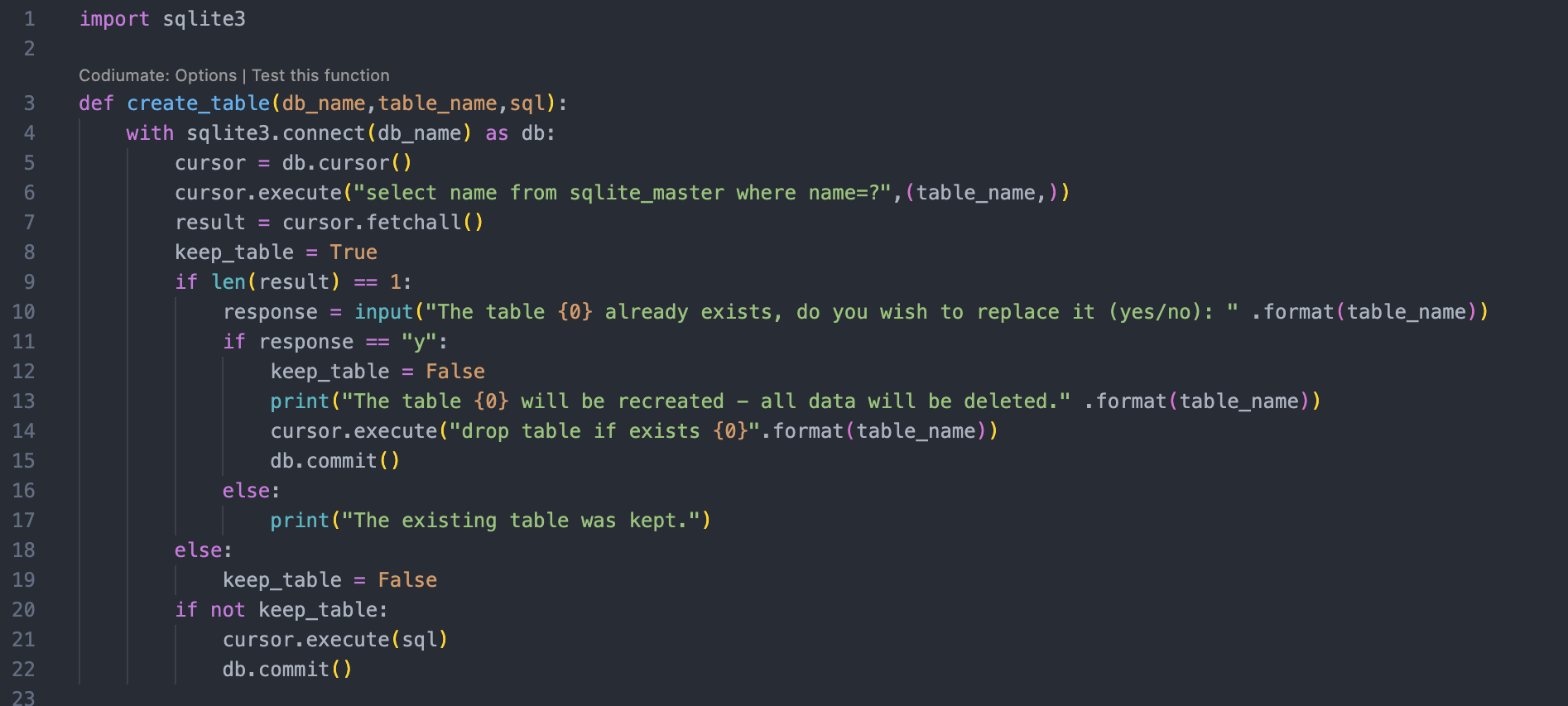
I will also not be using validation routines in the prototype as this would have been far too time consuming at this stage of development.

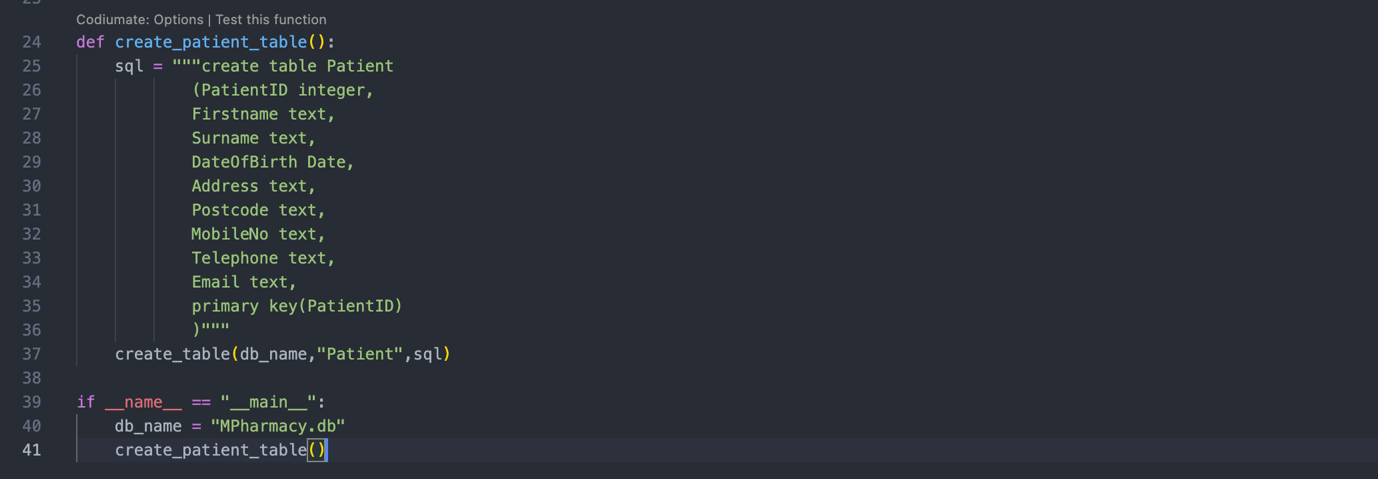
Additionally, the login system has also been left out as it is not vital to the core functionality of the overall system at this stage. Its impact becomes more relevant to the system in later stages of development and having to repeatedly log in would have been far too time consuming.

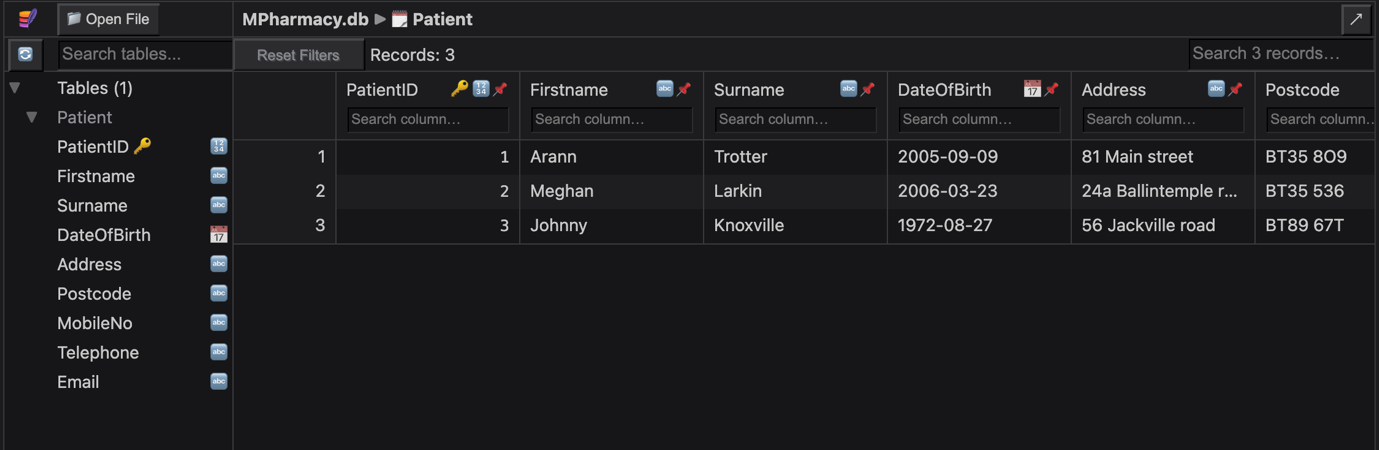
For my program I have used two python files for each window in the main program, one to hold classed and code for buttons and entry fields, the other for background processes such as validation routines. This helps the code become more efficient and easier to read for future developers of the system.

Creating the database

The create\_table function checks to see if the database table exists already within the DB file, and allows the user to create a new table or keep the existing table, in order to prevent crashing.



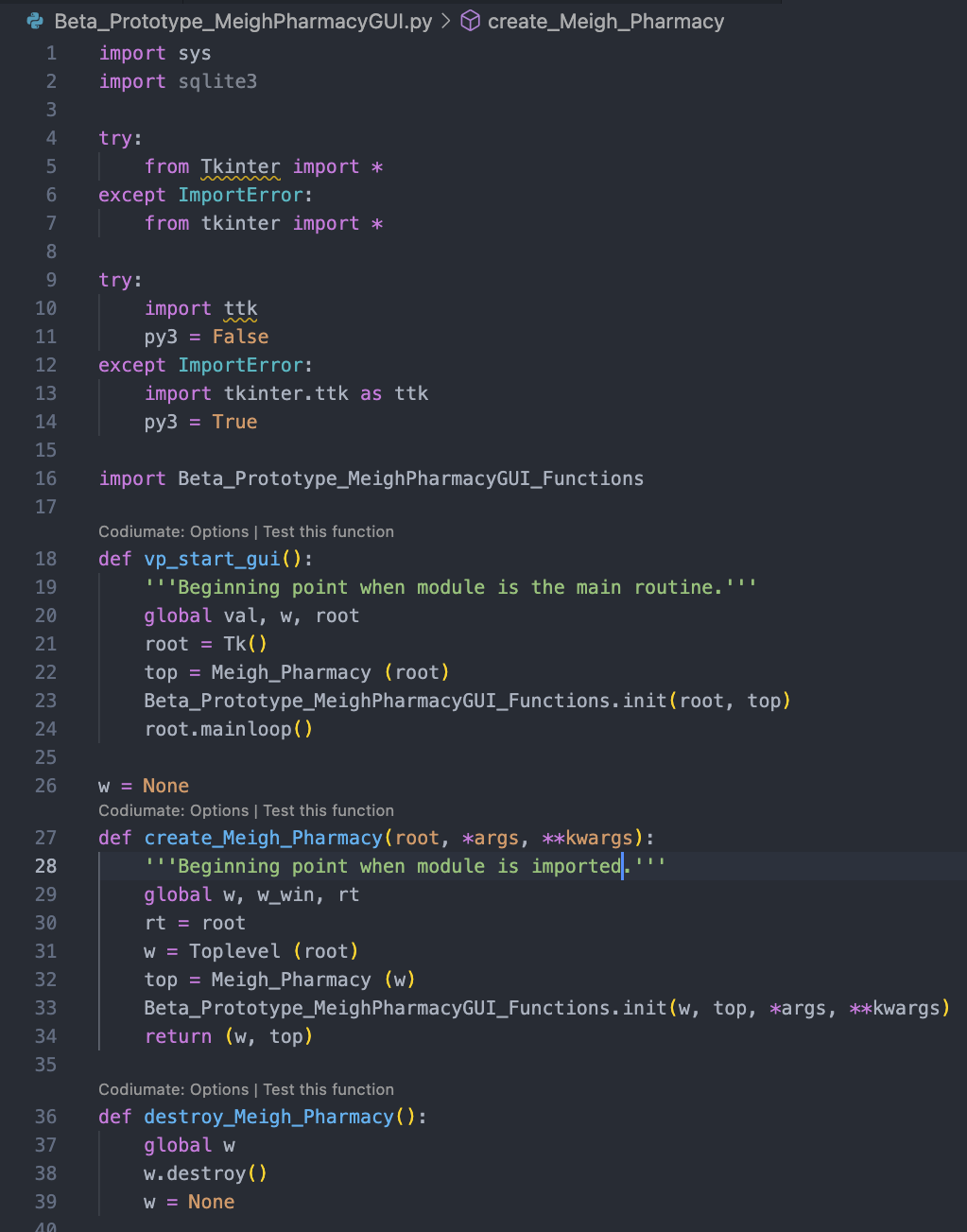


Below is the created database viewed using an online DB Viewer:

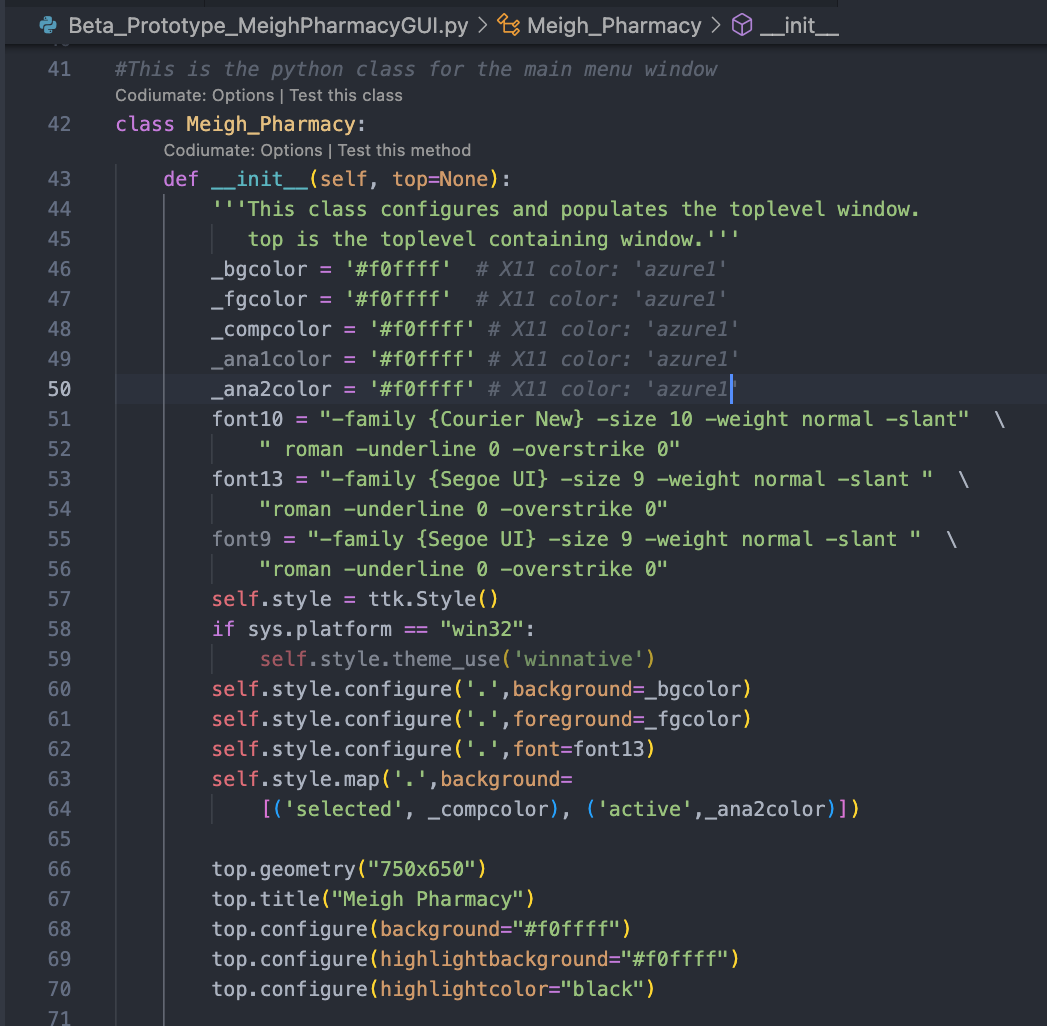
In the patient table PatientID is used as a primary key and can be used later in my main menu GUI to search for Patient details.

After creating the database, I then moved onto writing the code for my main menu GUI that will be able to add, search, edit, and delete values in the database.

The code below imports the necessary modules required for my GUI.

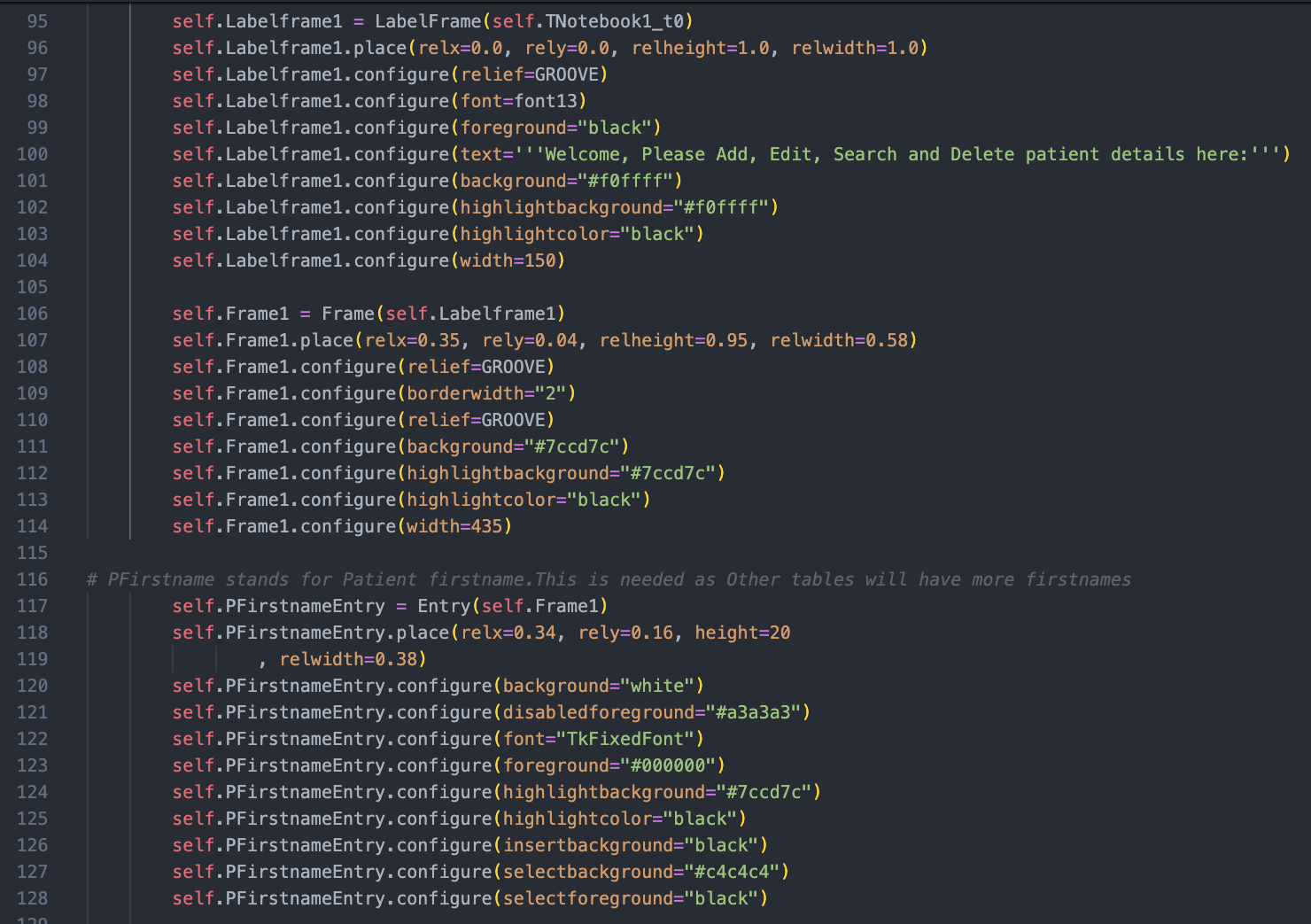


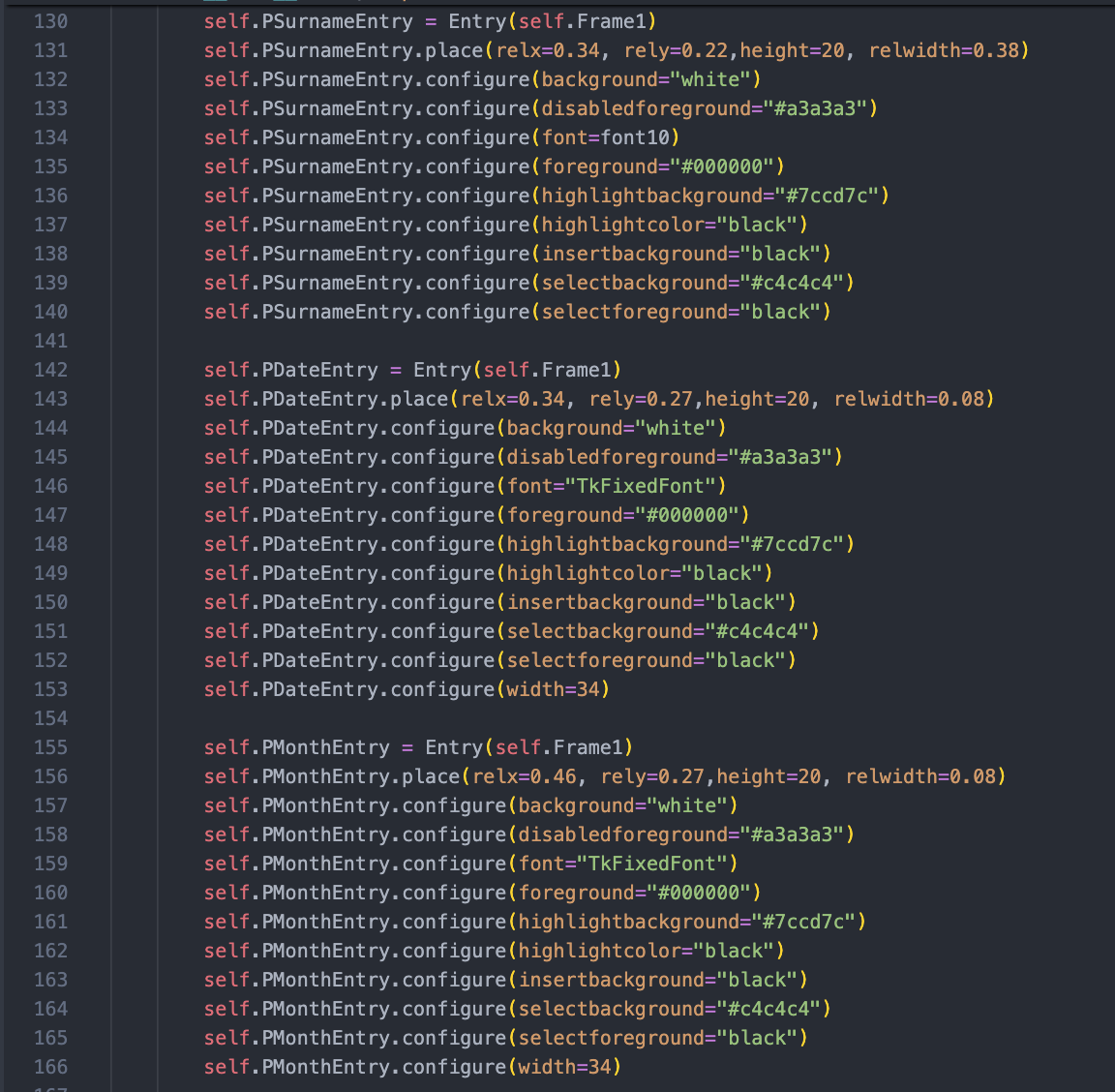
The code below shows that the main window is a python class, where in the code creates the basic components of the window such as the color, font and size of the window. The Colors also have their own variables as I felt this would make my code more efficient.

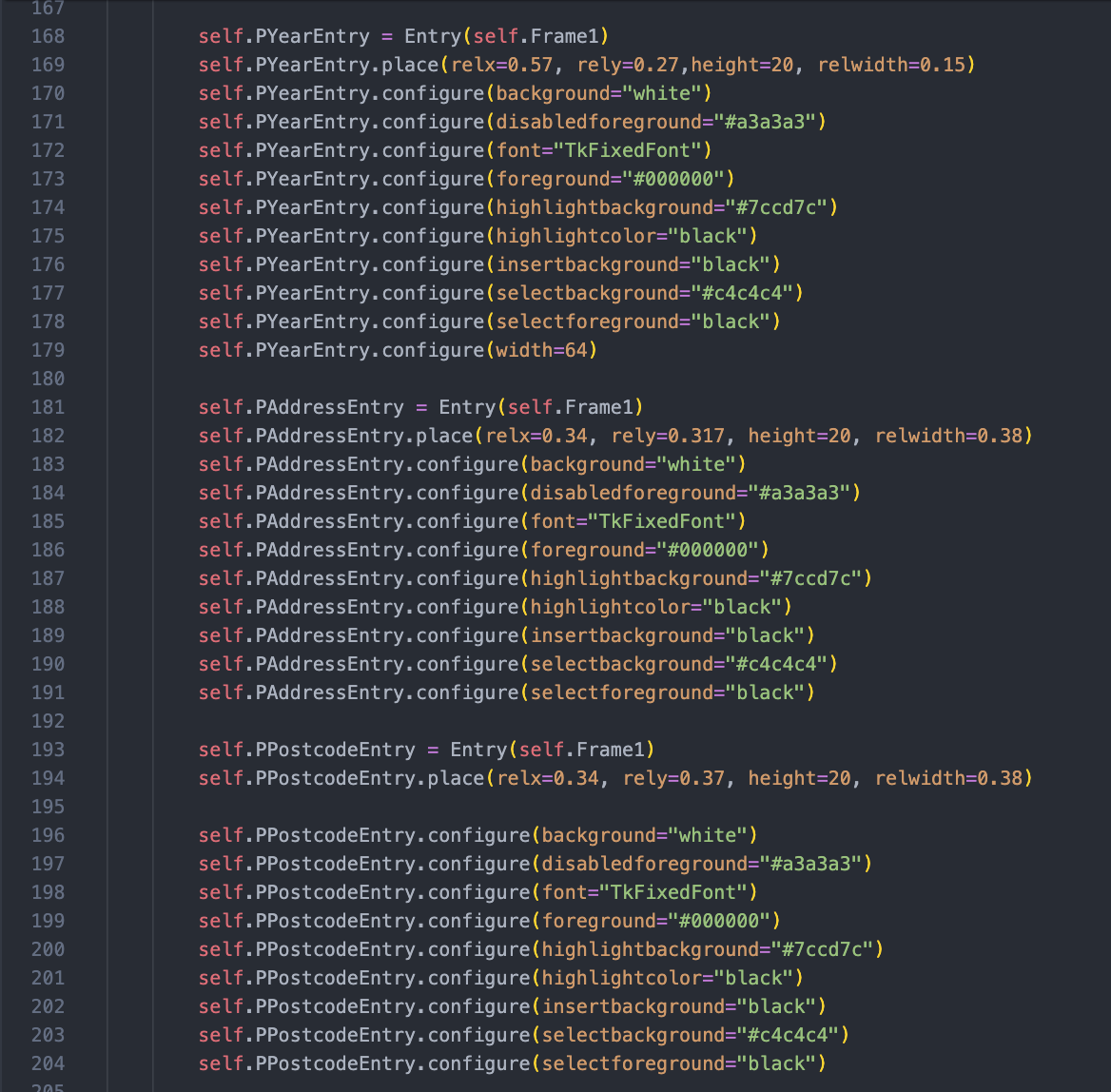


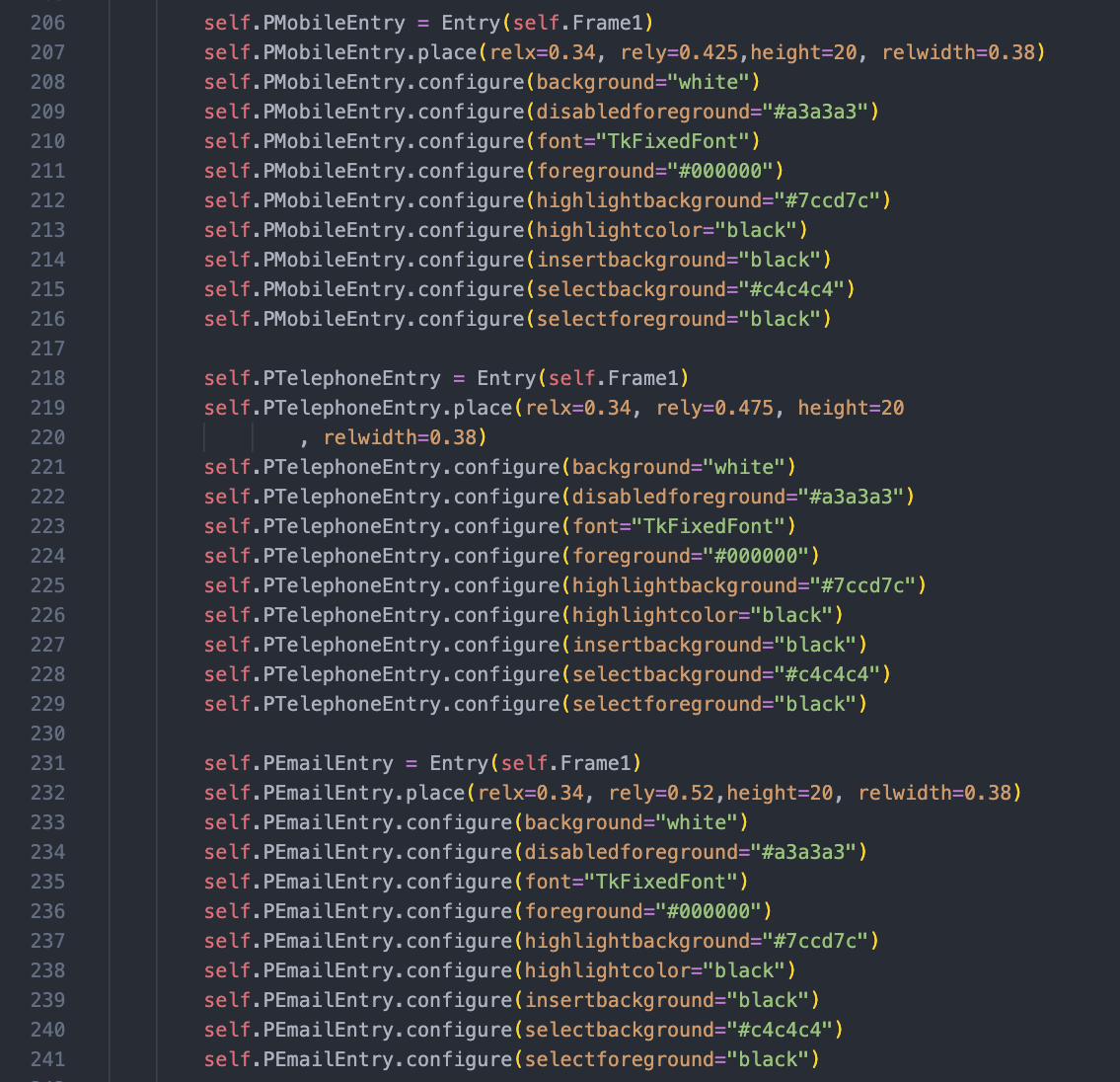
To create the aforementioned widget system, I had to code using ttk’s TNotebook widget. This helped me to create an intuitive widget system that would be extremely user friendly as each widget would contain its own frame for each table in the database. The use of this further helped with the efficiency of my code as it reduced the amount needed. The code I wrote is displayed below:

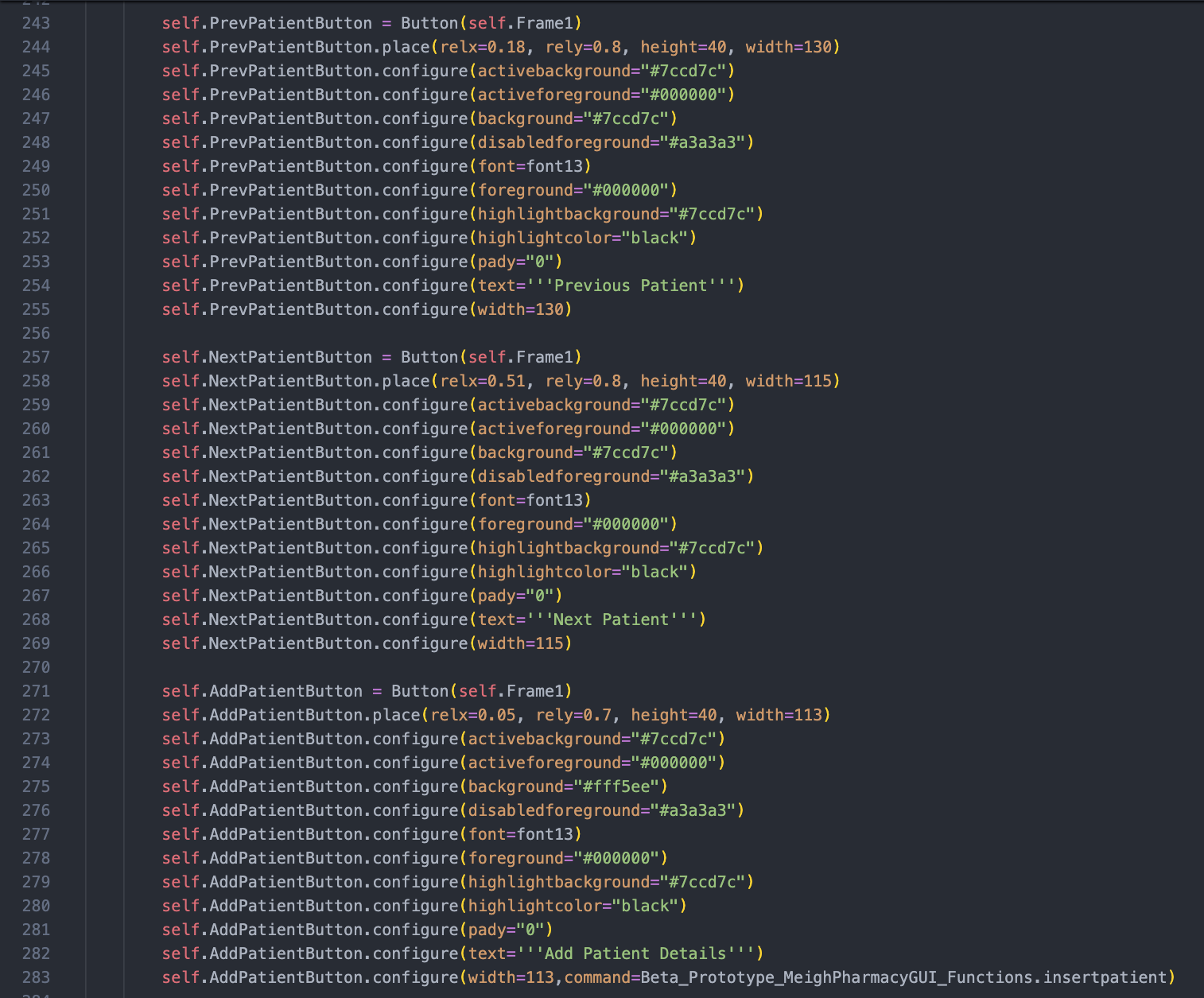


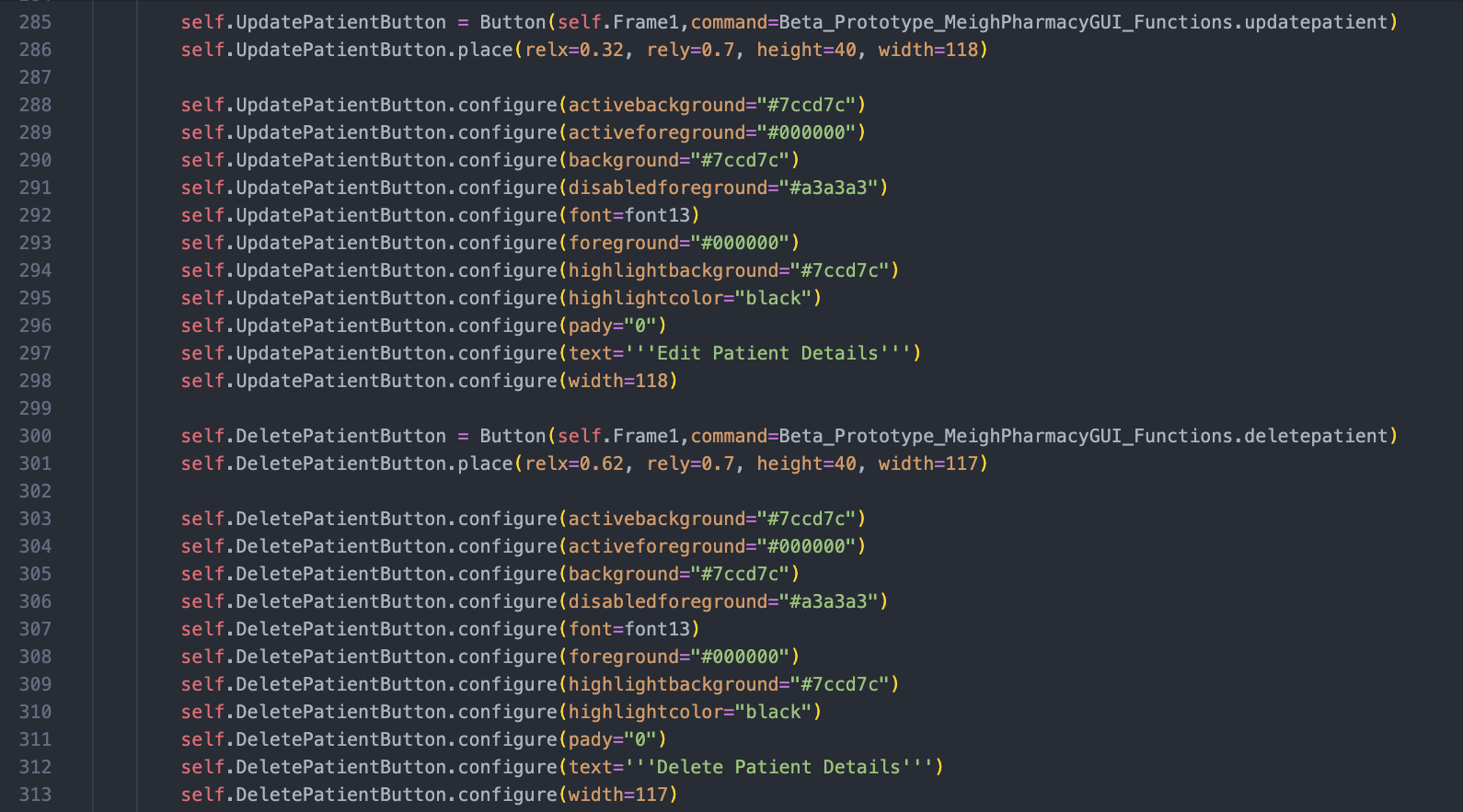


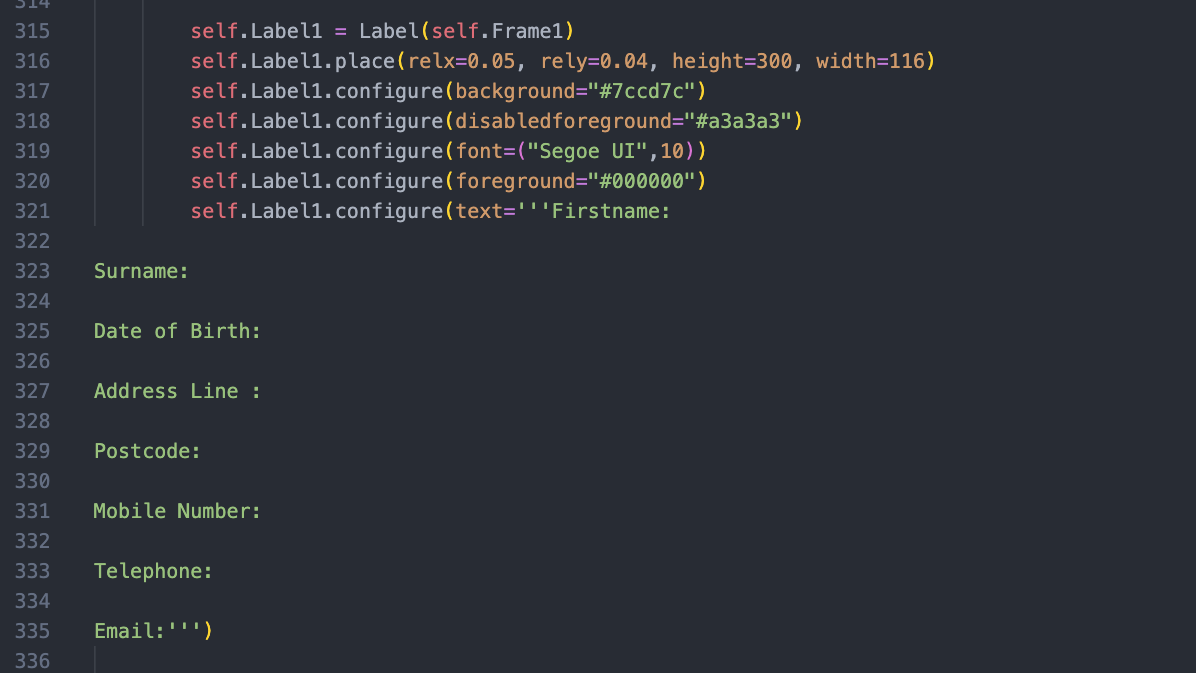










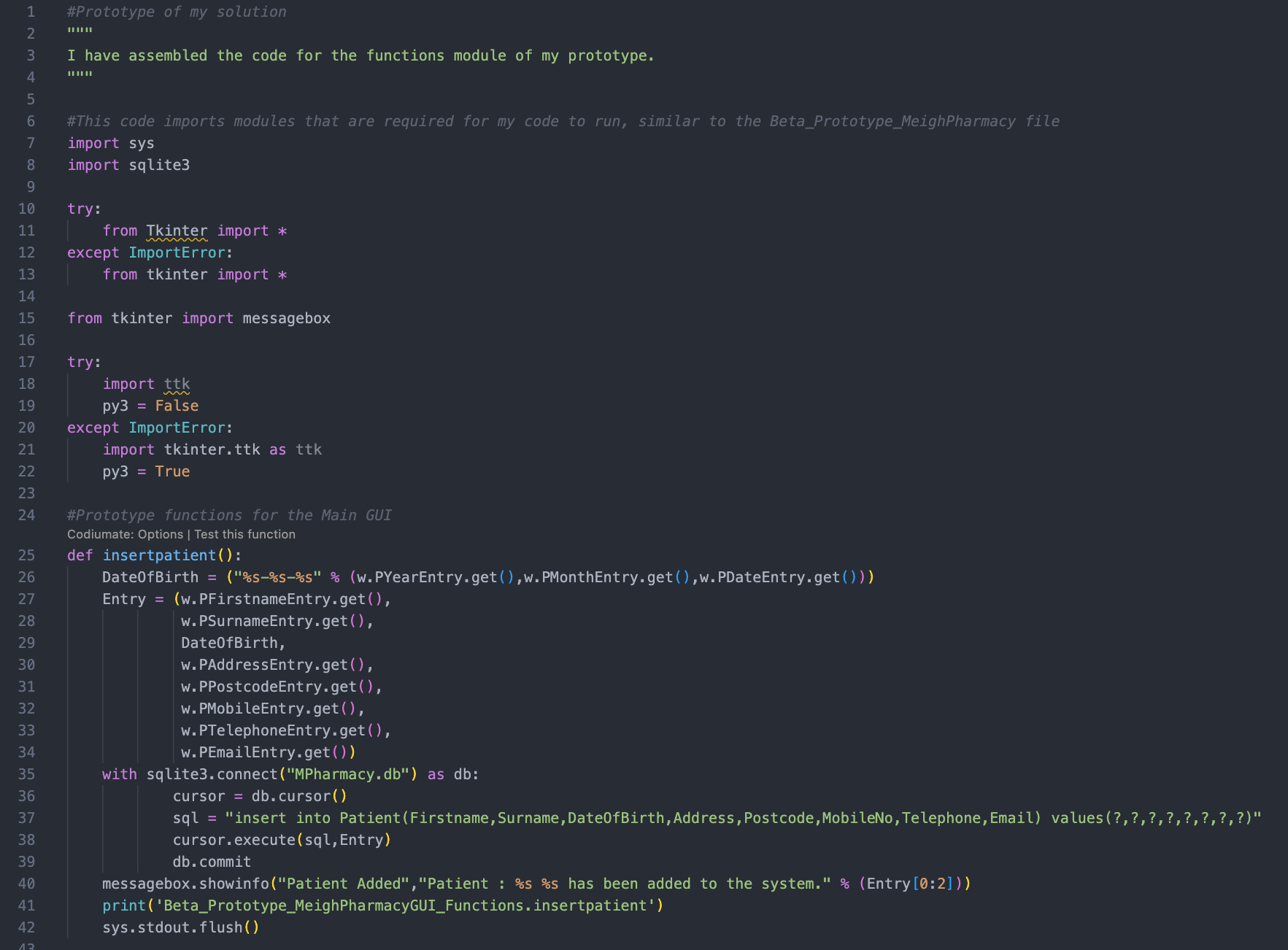




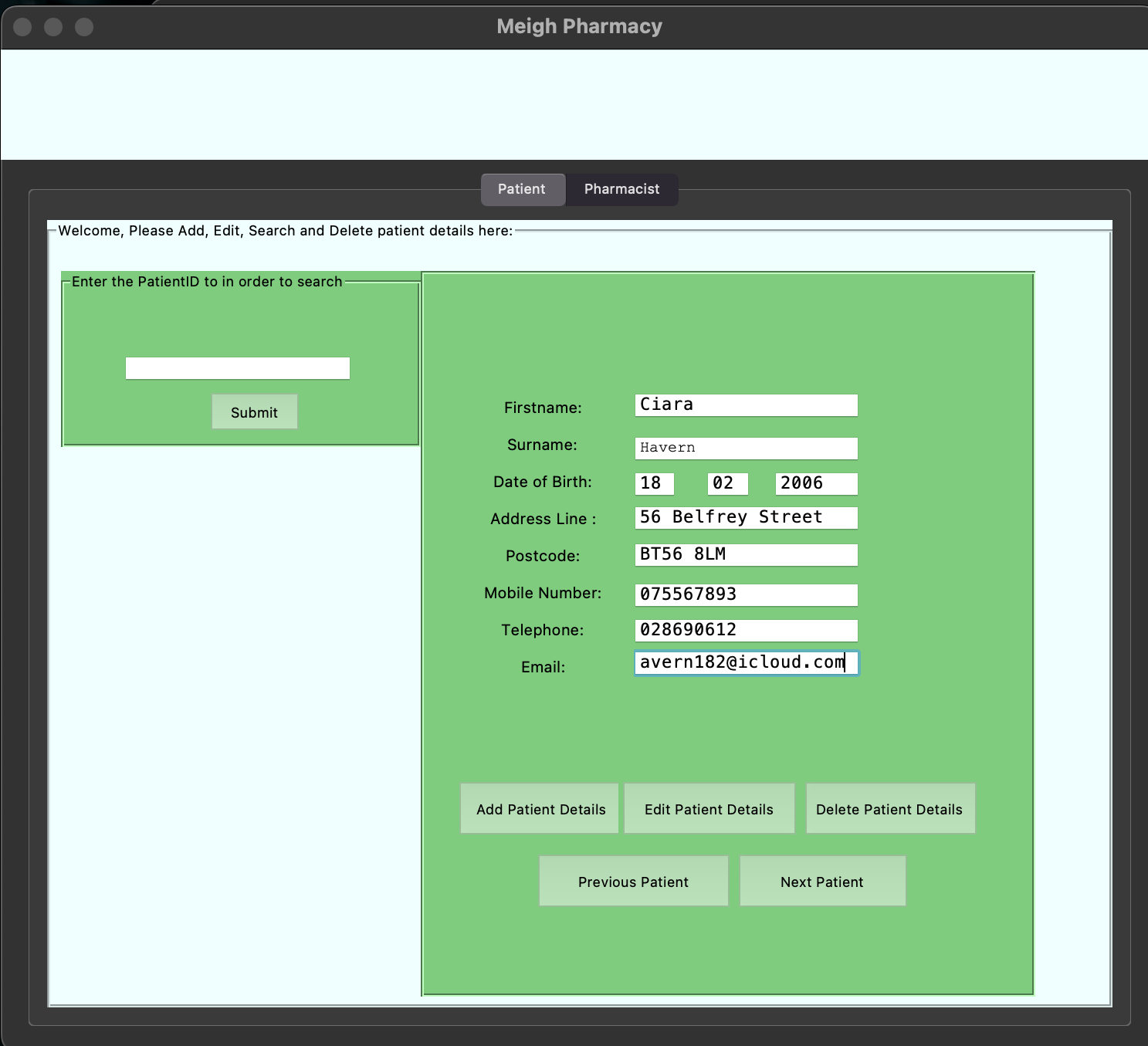
Below is the GUI created when I run the program:

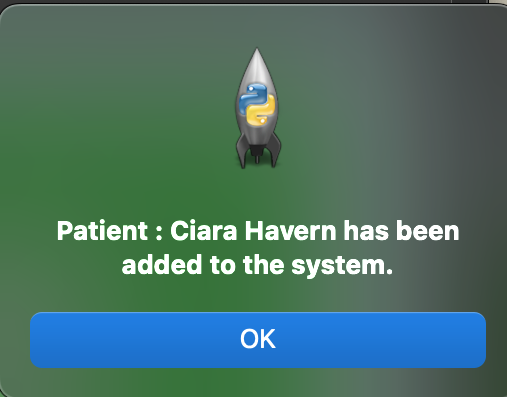
Currently, as you can see, I have created buttons to add, edit and delete patient details. I have also created buttons for previous record and next record, but these wont be functional in my prototype.

Next, I moved on to creating the functions python file to store the background functions of my Main GUI. It houses all of the main processes in functions that are defined

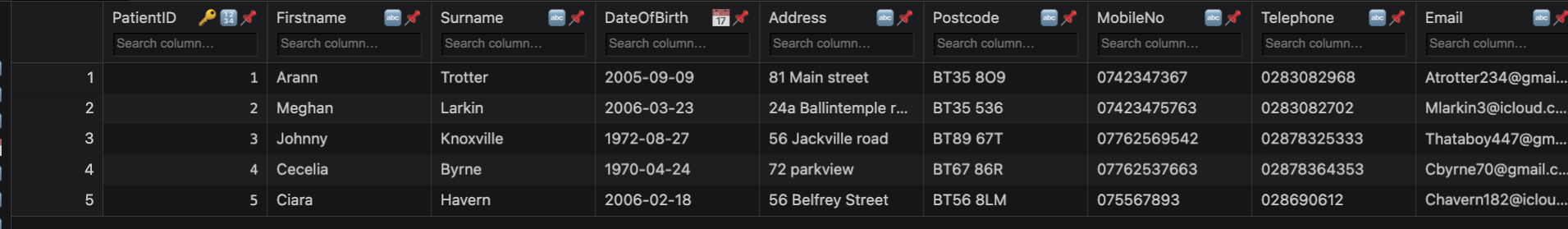


The function insertpatient() inserts data into the database, here is an example of it below :





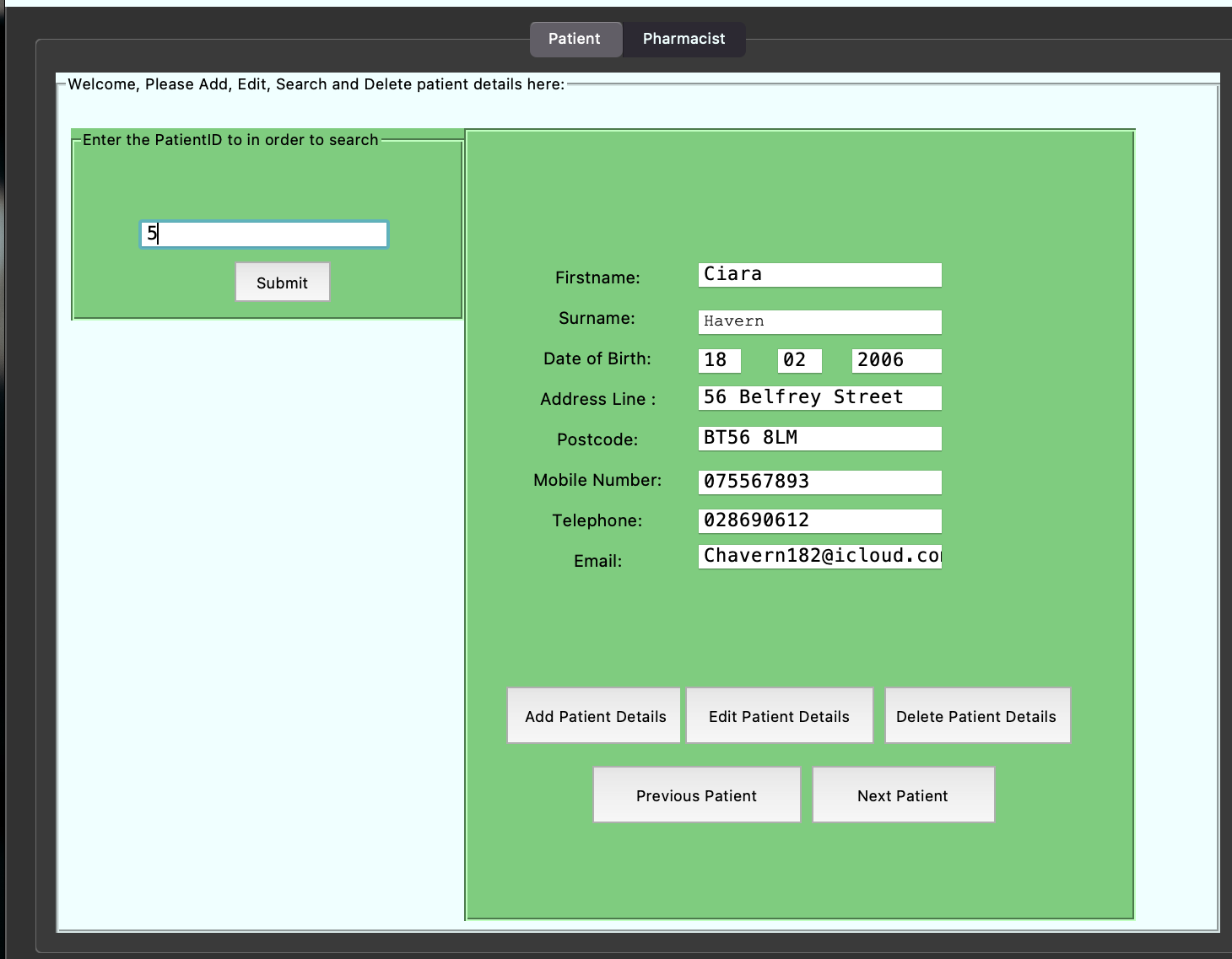
You can also see the GUI Response after we have added the patient details.

Here is the record in the database:

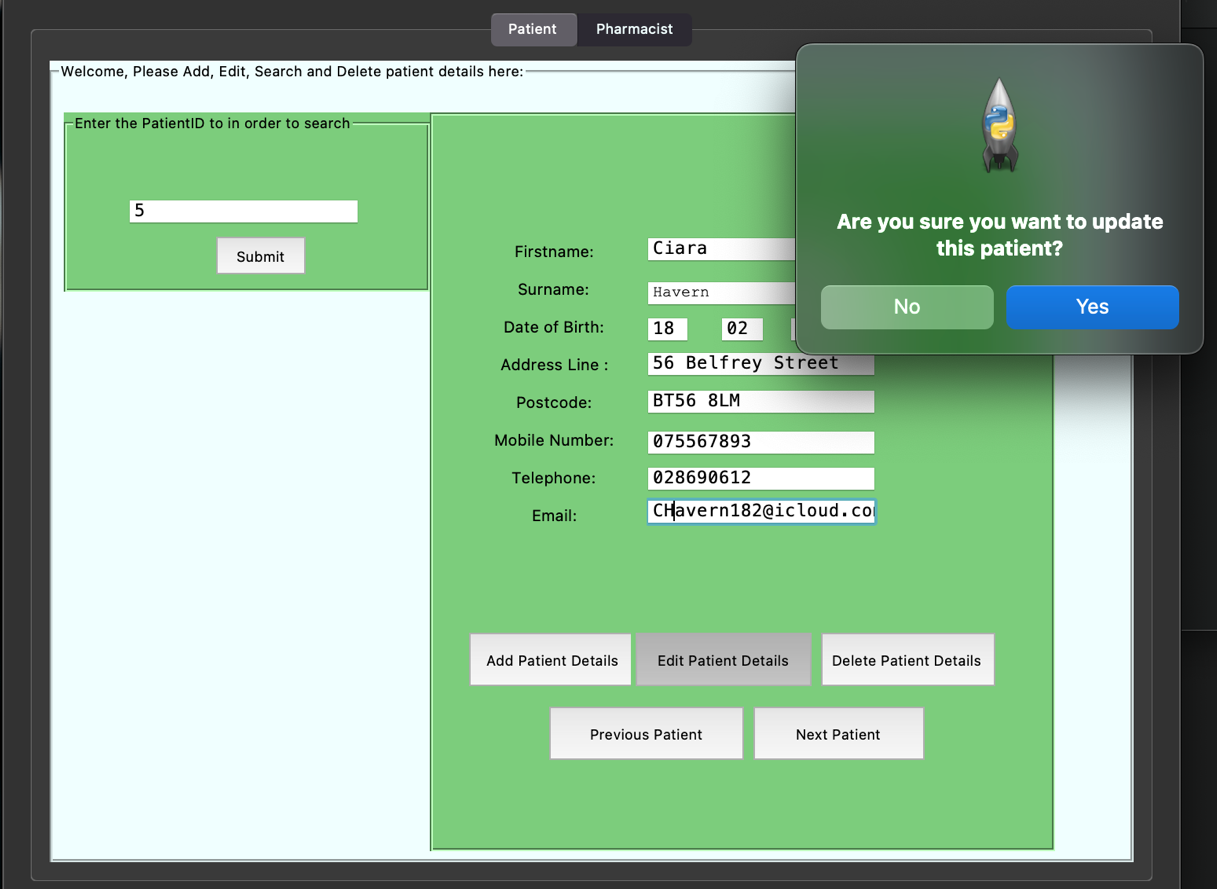
Here is the code for updating the patient record. It takes values in the entry fields and updates the record that has been searched.

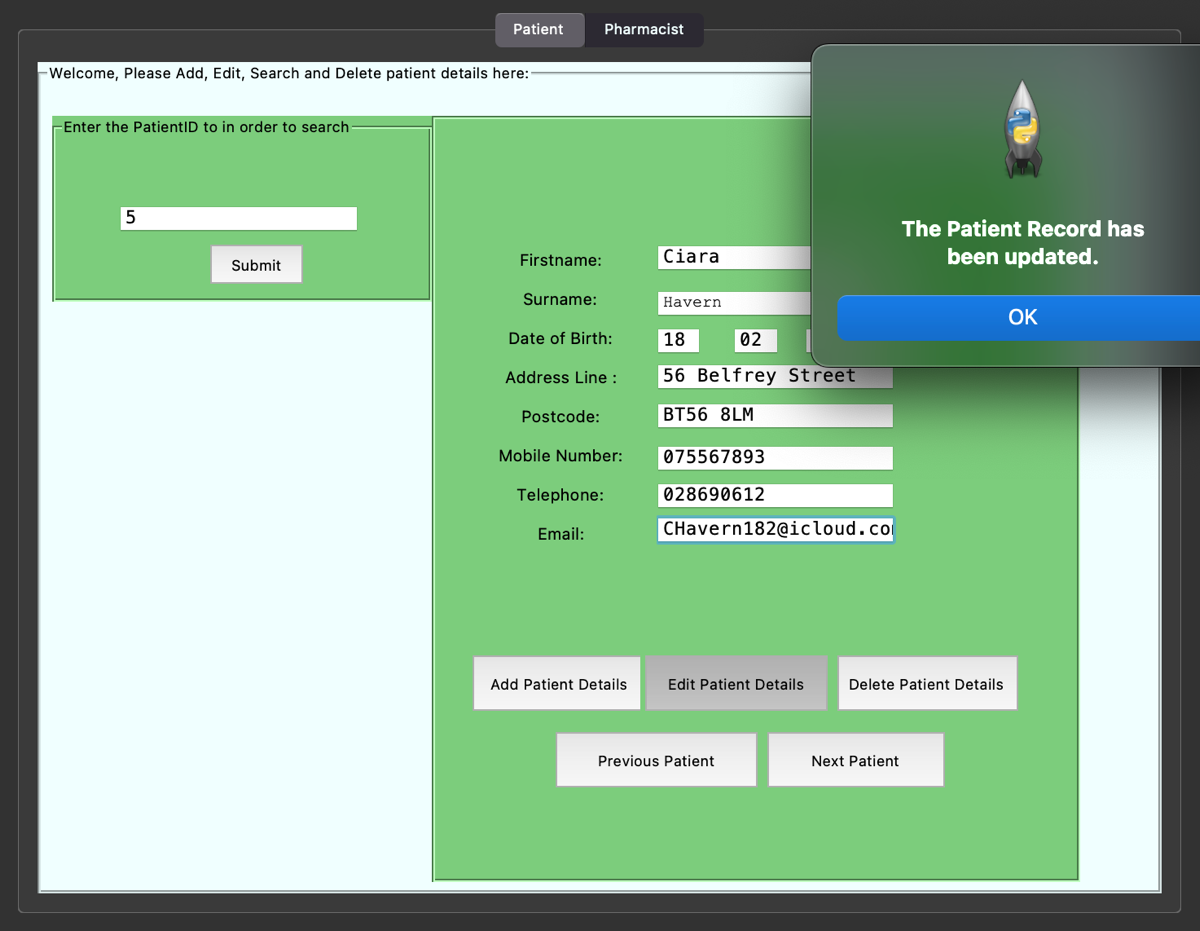


Here is an example of Updating a record:

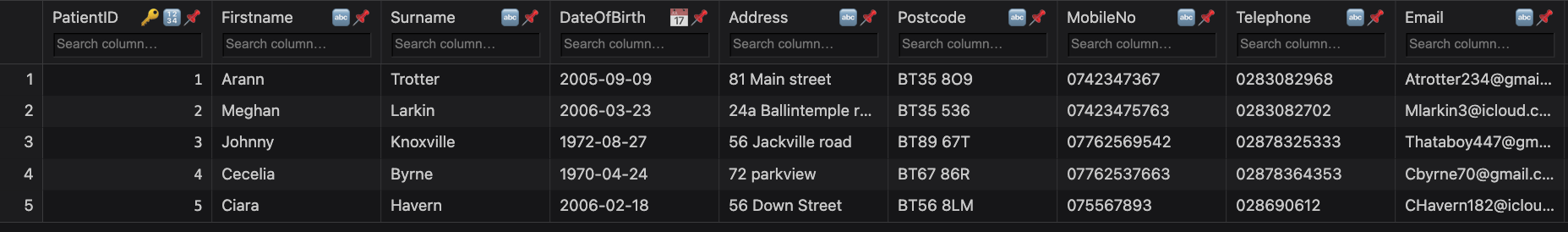


Patient with PatientID = 5 is searched for and then data enters the entry fields.



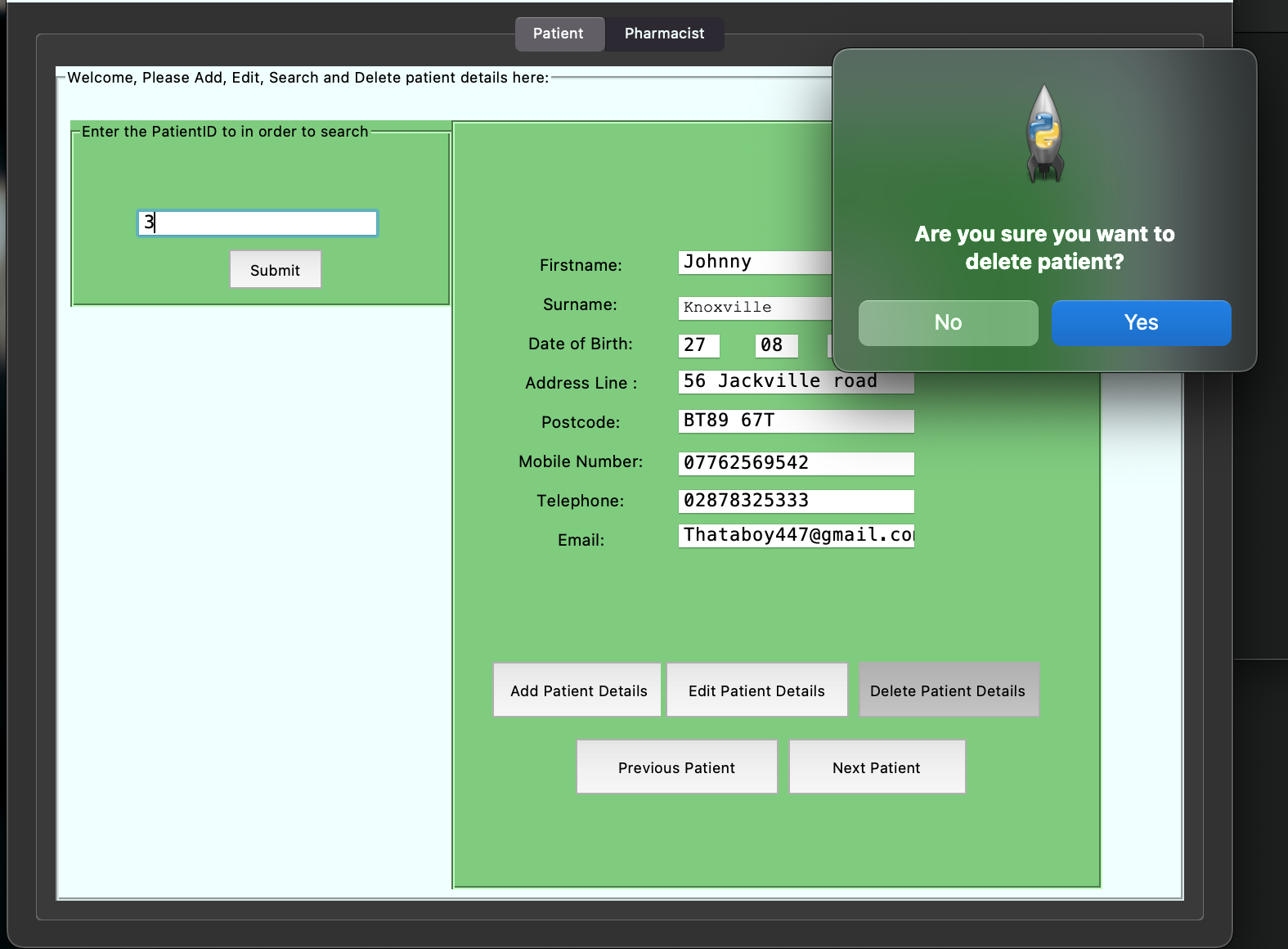
The system asks for conformation for editing the patient details.

Confirmation that the record has been edit.

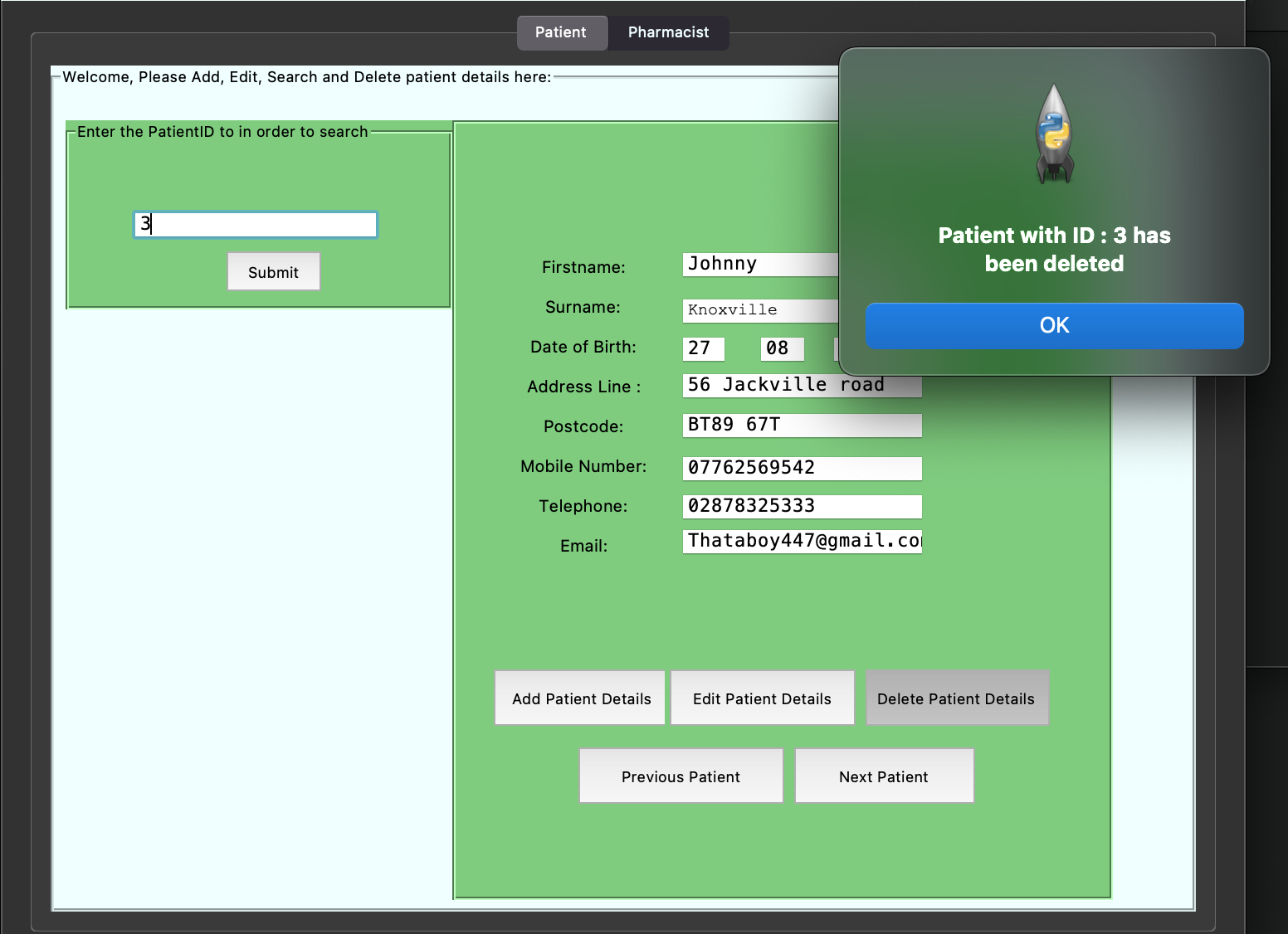


Above is the updated record with the new address in patient 5’s row.

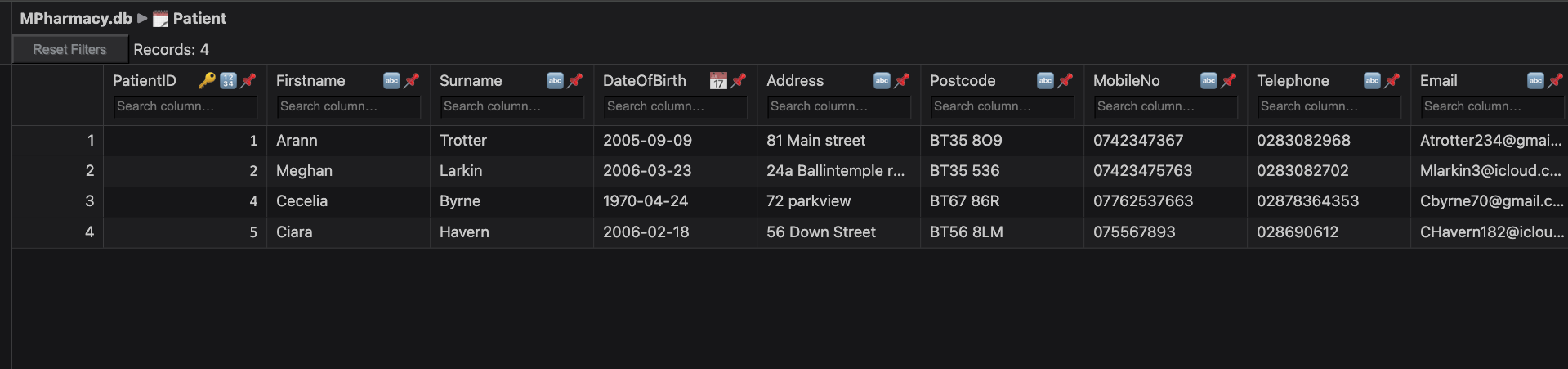
Below is the code for deleting patient details from the database:



Asking the user for confirmation that they want to delete the Patient.



--The system notifying the user that the certain patient has been deleted.

--The patient record is no longer in the Table below:

**Evaluation of my prototype**

**Strengths**

* I have completed my objectives in the functionality of the system including:
* Addition, Editing and Deletion of Patient Details
* Creation of MPharmacy.db file to store the patient details
* System responds to user actions
* Successfully linked the MPharmacy.db database to a GUI
* Successfully implemented the use of TNotebook widgets which make the system more efficient and user friendly
* Created a fully functional Main menu GUI
* Efficient use of code resources has reduced redundancy in my code and made the code itself overall more efficient

**Weaknesses**

* Some very important functions of the system were left out of the prototype including:
* No login in system, which is vital in adding security and safety to the system, if included in the prototype it would have been able to be tested, verified and refined for the final program
* Manipulation of other records, in that it would have made more sense to create all of the tables needed in the final design
* No printed outputs mean that making what is actually in the database impossible to view
* No access levels at this stage
* Lack of a user guide means that the system can become confusing, however this can be fixed with a simple readme file

**Refinement Recommendations**

* I Believe I should use my prototype and further develop it to create my final system as it contains the correct functionality that could be used as a foundation for my final solution.
* Create a Readme with user instructions