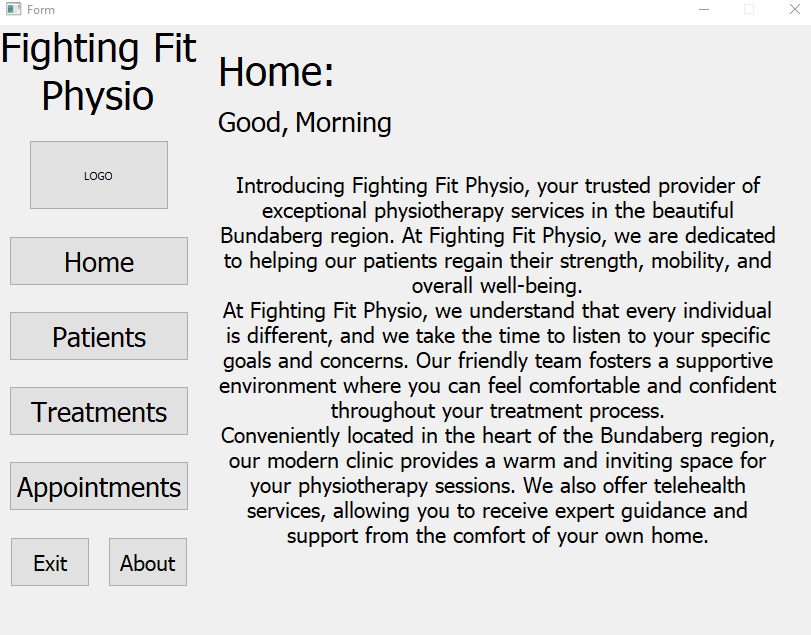
**Mind Map:**

|  |  |
| --- | --- |
| **Prescribed Criteria** | **Self-Determined Criteria** |
|  |  |

A screenshot of a computer

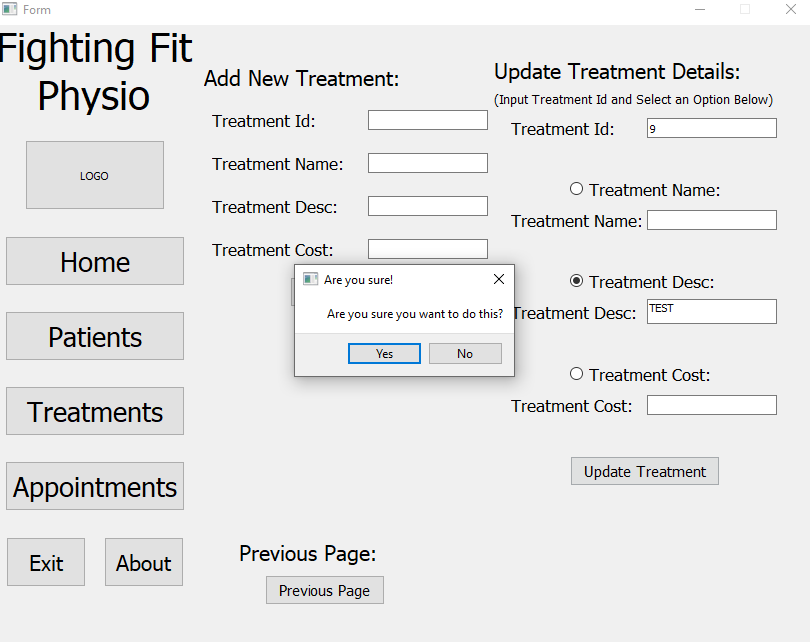
Description automatically generated

**A Welcome, home page, that outlines what Fighting Fit Physio is and what they hope to achieve, so all staff understand the company and its goals. The “Good, Morning” also changes depending on the time of day.**

**Searching a patients information using their “Patient Id”**

***Running Code:***

A screenshot of a computer

Description automatically generated

**Clicking the “Yes” button on the previous pop up box runs the function.**

**Updating a section of a treatment’s details. Where it then prompts the user for confirmation using an “are you sure you want to do this?” pop up box. This is also seen throughout the solution with similar functions. Therefore, addressing the useability principal safety.**

***Pseudocode:***

BEGIN

SET ds =DataStore() SET Student First Name = FirstName INPUT Treatment Description

SET Student Last Name = LastName INPUT Treatment Name

SET self.ui = QMainWindow() SET Patient Address = Address self.ds.Add\_Treatment(

SET Patients = self.ds.Matching\_Patient\_id() SET Patient Height = Height Treatment Id,

INPUT Patient Id SET Patient Weight = Weight Treatment Cost,

IF Patient Id NOT IN Patients THEN SET Patient Number of Treatments = Amountoftreatmentstaken Treatment Description,

INPUT Patient First Name self.ds.Add\_Patient(FirstName, Height, Weight, LastName, Amountoftreatmentstaken, Address) Treatment Name

INPUT Patient Last Name )

INPUT Patient Address SET Treatments = self.ds.Matching\_Treatment\_Id()

INPUT Patient Height INPUT Treatment Id

INPUT Patient Weight IF Treatment Id NOT IN Treatments THEN

INPUT Patient Number of Treatments INPUT Treatment Cost

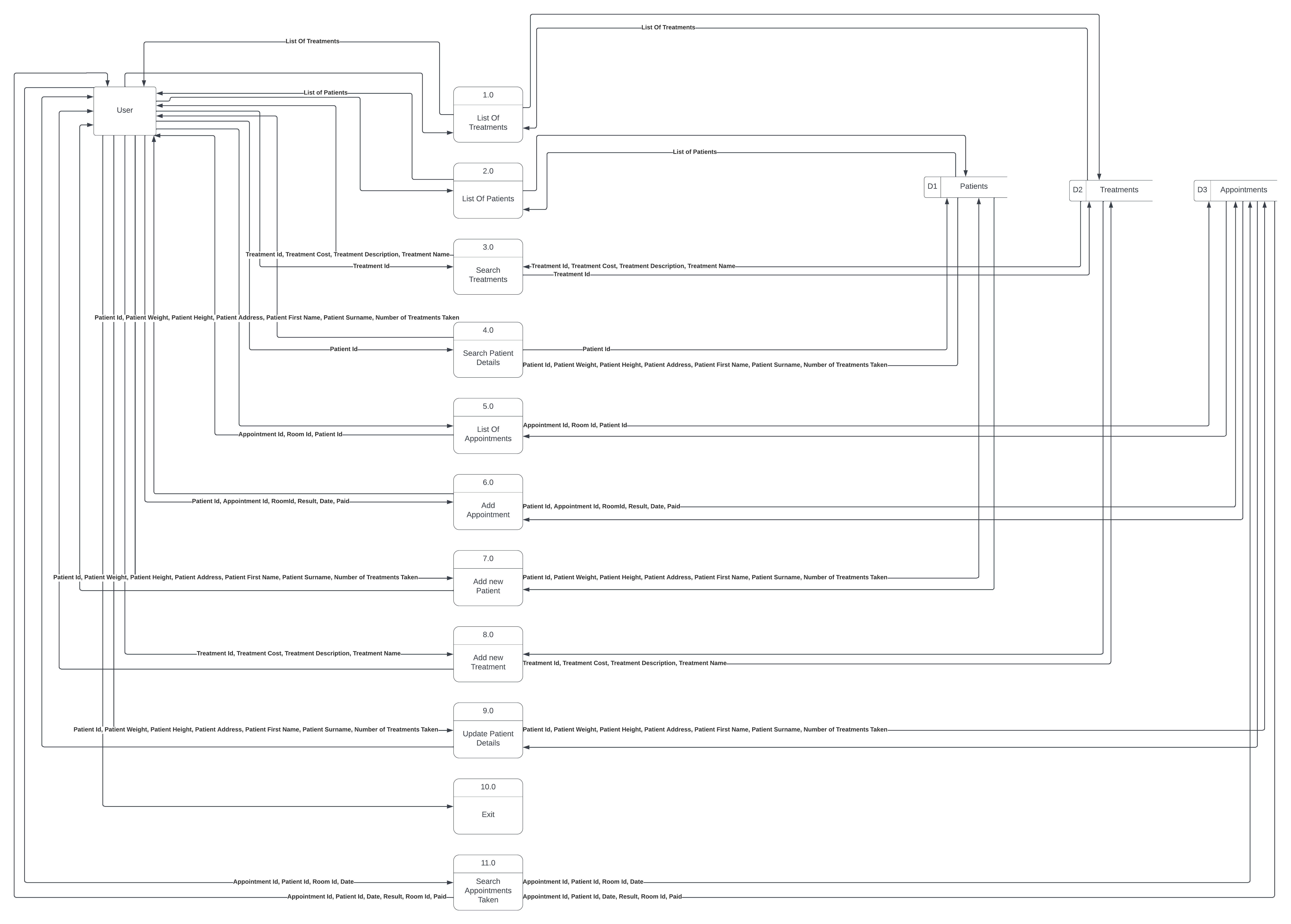
DISPLAY Error Pop Up Box SET Test Price = price

SET Patient Name = name ds.insert\_new\_test(code, name, desc, price)

SET Patient D.O.B = dob

SET Patient Height = height

SET Patient Weight = weight



**-DB Browser for SQLite**

DB Browser for SQLite is a visual open-source tool made for creating, designing, and editing database files that are compatible with SQLite. (*Edureka*, 23 Oct. 2019) DB Browser was used to manage and generate the data base in this solution.

**Tools and Languages:**

**Data Flow Diagram (DFD):**

**-Python**

Python is an object-oriented, strongly typed coding language, with easy-to-use syntax, perfect for first time learners of code. (Python.org, unknown) This language was chosen to type the solution as it offered a time effective, and easy to code solution when developing the prototype. Python also allowed the user to have a simple way to debug the solution compared to other coding languages such as C++ or C#. Further allowing the prototype to be built more time effectively.

**-Lucid Chart**

Lucid chart is an online tool that helps users sketch and develop flowchart diagrams, also providing designs to help creatives develop their ideas. (Lucid Chart, Unknown) Lucid Chart was used to generate the ERD for the solution. Lucid Chart was used for its simple, easy-to-use, and easy-to-navigate user interface. As well as this, Lucid Chart supplied the correct shapes needed to generate the ERD.

**-Adobe XD**

Adobe XD is an easy-to-use vector-based design platform that was built with performance at the top of mind. (What is Adobe used for, Oct 26, 2020) Adobe XD was used to generate the WireFrames for this solution and was chosen over many alternatives due to its accessibility, speed at which designs could be generated, its reliability, and the overall feel of the program didn’t feel clunky or bloated with unnecessary features.

**-QT Designer**

Qt Designer is a tool that allows easy creation of GUIs. (Graphical User Interface) Allowing users to drag and drop QWidget objects to build the GUI. (Leodanis Ramos, Unknown) This tool was selected over other GUI build tools due to its easy “plug and play” style used when developing the GUI. The tool is also heavily compatible with python allowing the rest of the solution easy to code as it aligned with the chosen coding language.

**-SQLite**

SQLite is the most used database engine across all programming. SQLite is a Language Library that implements fast, very-reliable, self-contained, SQL database engine. It was chosen for this solution when compared to other SQL Database Engines, such as MYSQL, due to its simplicity, its file format being stable, the fact that SQLite is built into all mobile devices and most computers, meaning it is well received and well documented which proves useful when encountering any errors. The source code is in the public domain and the developers are completely transparent and dedicated to their work. (SQLite Home Page, n.d.)

**-Freeplane**

Freeplane is a free and open-source software application that provides online tools for creating mind maps, concept mapping or information mapping. (Freeplane documentation, n.d.) Freeplane was chosen over other mind mapping tools such as Xmind, Mindmeister, or Diagrams.net due to the application being more optimized, more accessible, it runs on any operating system that includes Java, and Freeplane is much more efficient when using its mapping tools.

**-Microsoft Visual Studio Code**

Microsoft Visual Studio Code is a powerful developer tool, that combines the simplicity of a source code editor with useful and quality tools, such as IntelliSense code completion and debugging. The easy “edit-build-debug” cycle allows for less time managing you enviorment, and more time to produce code. (Microsoft, 3 Jan 2023) Vs Code was chosen to produce this solution for its easy-to-use debugging, allowing for a time efficient process to deploy the solution.

A screenshot of a medical application

Description automatically generatedA screenshot of a computer

Description automatically generated

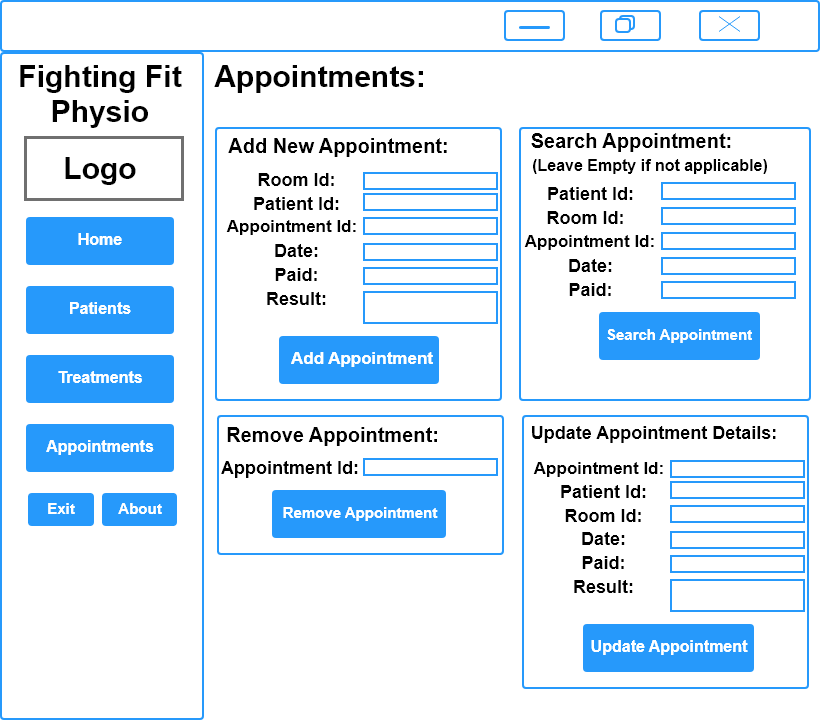
**Learnability**

The banner on the left side of the screen and its buttons address the useability principal **Learnability**, as the banner and its ability to change pages similar to several other commonly used websites, therefore being easy to learn as this way of site navigation should already be innate to the user.

**Annotated Wire Frames:**

**Effectiveness**

All functions throughout the solution address the useability principle, **effectiveness**, as it takes minimal user input to supply the user with the information they want. As you can see, only requiring just the Patients Id to search all of their information.

A screenshot of a computer

Description automatically generated

**Accessibility**

All Pieces of text and buttons present in the solution are written in bold and in an easy-to-read font size (18). Therefore, addressing the useability principle of **Accessibility**, as it allows users who struggle to read small text or see to use the solution more easily and freely.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Utility**

All functions present in the solution address the useability principle of **utility**, as they supply the user with all the possible functions the faculty will need to use. Such as, “Add New Treatment” and etc.

**IMPACTS:**

**SOCIAL**

**PERSONA**

**ECONOMICAL**

**CONCEPTUAL SCHEMA:**

Patients (

Id PK NOT NULL

First Name NOT NULL

Last Name NOT NULL

Height NOT NULL

Weight NOT NULL

Address NOT NULL

Number of Treatments Taken NOT NULL

)

Appointments (

Id PK NOT NULL

Result NOT NULL

Date NOT NULL

Room Id NOT NULL

Patient Id FK NOT NULL

)

Treatments (

Id PK NOT NULL

Treatment Name NOT NULL

Description NOT NULL

Cost NOT NULL

TreatmentAppointments (

Treatment Id FK NOT NULL

Appointment Id FK NOT NULL

)

**DATA DICTIONARY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Validation Rules | Example Data |  |
| Appointments |  |  |  |  |
| Id | INTEGER | Primary Key | 0-∞ |  |
| Date | INTEGER | NOT NULL | DD-MM-YYYY |  |
| Result | TEXT | NOT NULL | Description |  |
| Patient Id | INTEGER | Foreign Key | 0-∞ |  |
| Room Id | INTEGER | NOT NULL | 0-10 |  |
| Appointment Paid | TEXT | NOT NULL | Yes or No |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Patients |  |  |  |  |
| Id | INTEGER | Primary Key | 0-∞ |  |
| Height | INTEGER | NOT NULL | 0-∞ |  |
| Weight | INTEGER | NOT NULL | 0-∞ |  |
| Address | TEXT | NOT NULL |  |  |
| First Name | TEXT | NOT NULL |  |  |
| Last Name | TEXT | NOT NULL |  |  |
| Amount of Treatments taken | INTEGER | NOT NULL | 0-∞ |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Treatment |  |  |  |  |
| Id | INTEGER | Primary Key | 0-∞ |  |
| Cost | INTEGER | NOT NULL | 0-∞ |  |
| Description | TEXT | NOT NULL | Description |  |
| Treatment Name | TEXT | NOT NULL |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TreatmentAppointments |  |  |  |  |
| Appointment Id | INTEGER | Foreign Key | 0-∞ |  |
| Treatment Id | INTEGER | Foreign Key | 0-∞ |  |

The **Treatment** table in the database follow the normalization process. The process is as follows:

1. Each Record within a table is atomic in nature – The database table follows this law as all records cannot be broken into smaller parts and are separated into records that are different from others.
2. Each record within a table is fully dependent on the primary keys within that record to give overall meaning – The database records follow this law as they are all given meaning by the treatment id the primary key of the table.
3. Each record must not have any data dependencies other than the primary keys – The database table follows this law as it solely relies on the tables primary key.

(Docherty, in et al., Nelson Digital Solutions for QCE, Section 2.2, pg 140)

**Normal Forms:**

CREATE TABLE "Appointments" (

"Id" INTEGER NOT NULL UNIQUE,

"Date" INTEGER NOT NULL,

"Result" TEXT NOT NULL,

"PatientId" INTEGER,

"RoomId" INTEGER NOT NULL,

"Paid" TEXT NOT NULL,

PRIMARY KEY("Id" AUTOINCREMENT),

FOREIGN KEY("PatientId") REFERENCES "Patients"("Id")

);

CREATE TABLE "Patients" (

"Id" INTEGER NOT NULL UNIQUE,

"Height" INTEGER NOT NULL,

"Weight" INTEGER NOT NULL,

"Address" TEXT NOT NULL,

"FirstName" TEXT NOT NULL,

"LastName" TEXT NOT NULL,

"Amountoftreatmentstaken" INTEGER,

PRIMARY KEY("Id" AUTOINCREMENT)

);

CREATE TABLE "Treatment" (

"Id" INTEGER NOT NULL UNIQUE,

"Cost" INTEGER NOT NULL,

"Description" TEXT NOT NULL,

"TreatmentName" TEXT NOT NULL,

PRIMARY KEY("Id" AUTOINCREMENT)

);

**CONCREATE SCHEMA:**

The **Appointment** table in the database follow the normalization process. The process is as follows:

1. Each Record within a table is atomic in nature – The database table follows this law as all records cannot be broken into smaller parts and are separated into records that are different from others.
2. Each record within a table is fully dependent on the primary keys within that record to give overall meaning – The database records follow this law as they are all given meaning by the appointment id the primary key of the table.
3. Each record must not have any data dependencies other than the primary keys – The database table follows this law as it solely relies on the tables primary key.

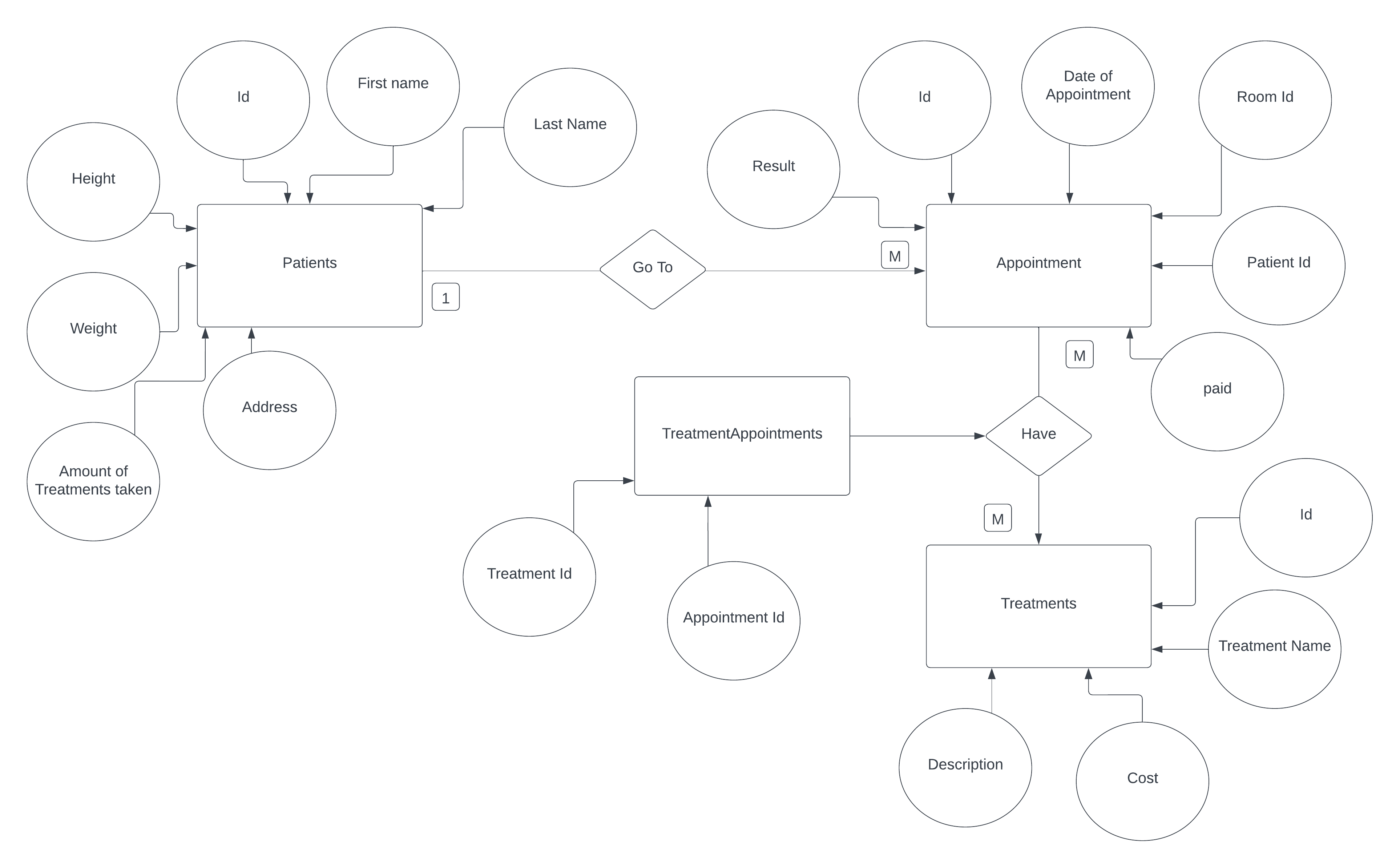
(Docherty, in et al., Nelson Digital Solutions for QCE, Section 2.2, pg 140)

The **patients** table in the database follow the normalization process. The process is as follows:

1. Each Record within a table is atomic in nature – The database table follows this law as all records cannot be broken into smaller parts and are separated into records that are different from others.
2. Each record within a table is fully dependent on the primary keys within that record to give overall meaning – The database records follow this law as they are all given meaning by the patient id the primary key of the table.
3. Each record must not have any data dependencies other than the primary keys – The database table follows this law as it solely relies on the tables primary key.

(Docherty, in et al., Nelson Digital Solutions for QCE, Section 2.2, pg 140)

**ERD:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria:** | **Type of Criteria:** | **Accomplished:** | **How Was it Accomplished:** | **Refinements** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

References:

*What is python? Executive summary*. (n.d.). Python.Org. Retrieved May 21, 2023, from <https://www.python.org/doc/essays/blurb/>

Qt Designer and Python: Build Your GUI Applications Faster. (n.d) Retrieved May 28, 2023, from <https://realpython.com/qt-designer-python/>

*Why visual studio code?* (n.d.). Retrieved May 21, 2023, from <https://code.visualstudio.com/docs/editor/whyvscode>

*Intelligent diagramming*. (n.d.). Lucidchart. Retrieved May 21, 2023, from <https://www.lucidchart.com>

Snr Digital Solutions Syllabus. (2019). Retrieved May 19, 2023, from <https://www.qcaa.qld.edu.au/downloads/senior-qce/syllabuses/snr_digital_solutions_19_syll.pdf>

*Sqlite home page*. (n.d.). Retrieved July 21, 2023, from <https://www.sqlite.org/index.html>

*Home—Freeplane documentation*. (n.d.). Retrieved July 21, 2023, from <https://docs.freeplane.org/>

What is Adobe XD and What is it used for? (n.d.) Retrieved July 18, 2023 from <https://www.adobe.com/products/xd/learn/get-started/what-is-adobe-xd-used-for.html>

“SQLite Browser | DB Browser for SQLite Explained.” *Edureka*, 23 Oct. 2019, https://www.edureka.co/blog/sqlite-browser/.

Evaluation and Refinements:

A picture containing black, darkness

Description automatically generated

**Annotated Code:**

A picture containing black, darkness

Description automatically generated

A picture containing black, darkness

Description automatically generated