Retirement home management system

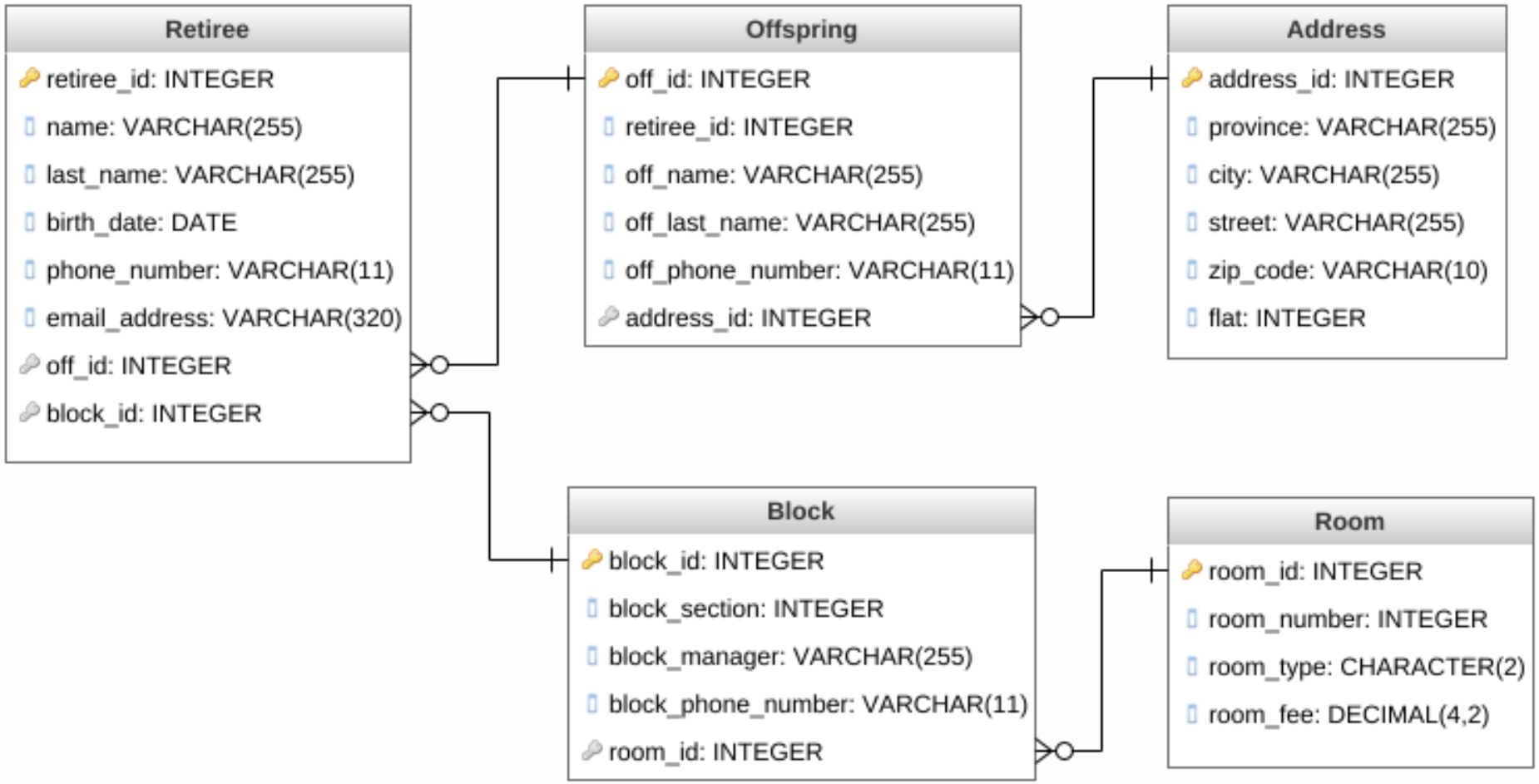
Python and SQL: intro / SQL platforms

Mohammad Saeed Pourjafar

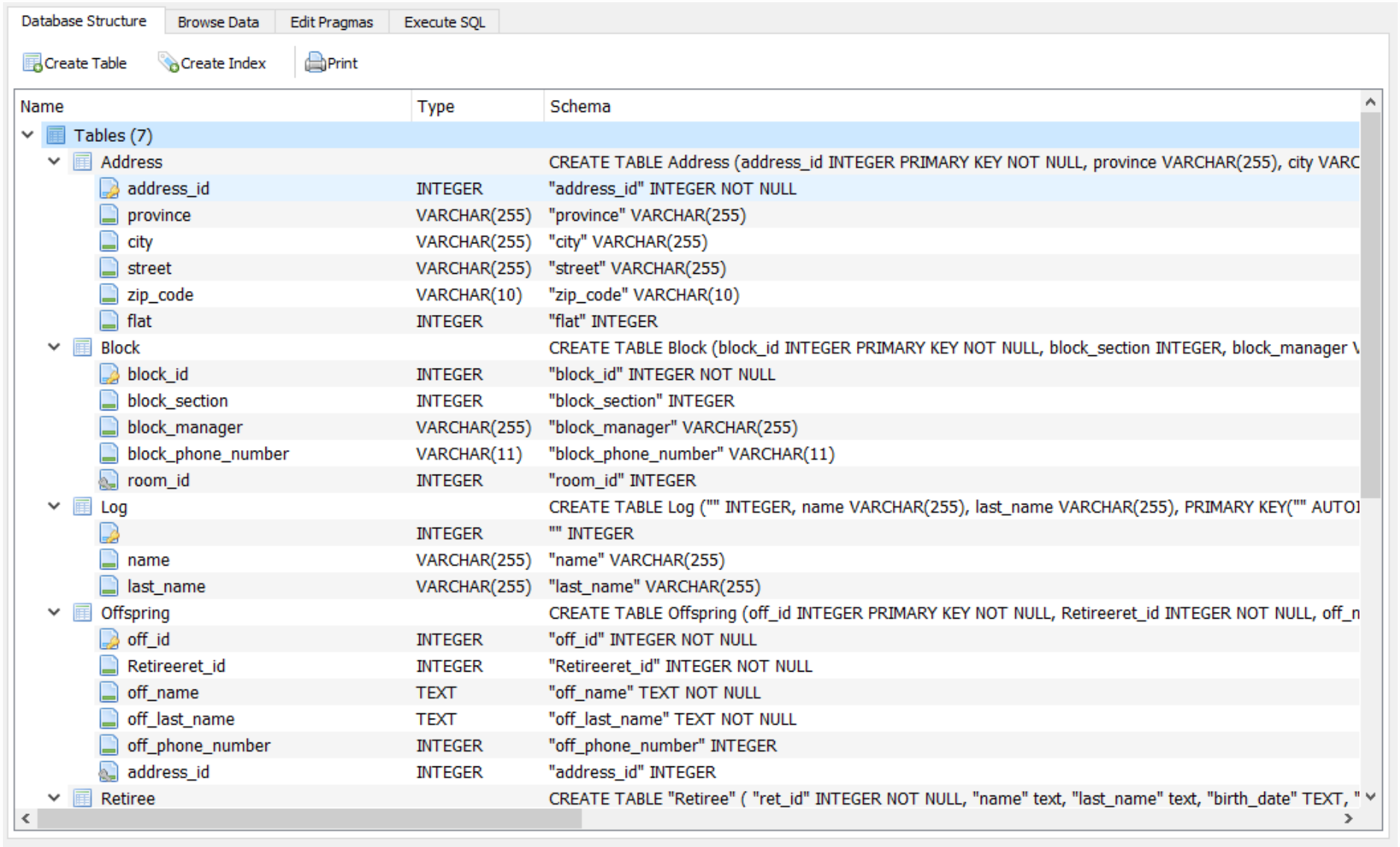
Introduction

The retirement home management system is a Graphical User Interface (GUI) for windows operating system which can be used to add, store and delete a retiree form the database. Also the administrator will be able to show the status quo of the retirees and update their information accordingly.

Database scheme



Database scheme



User manual

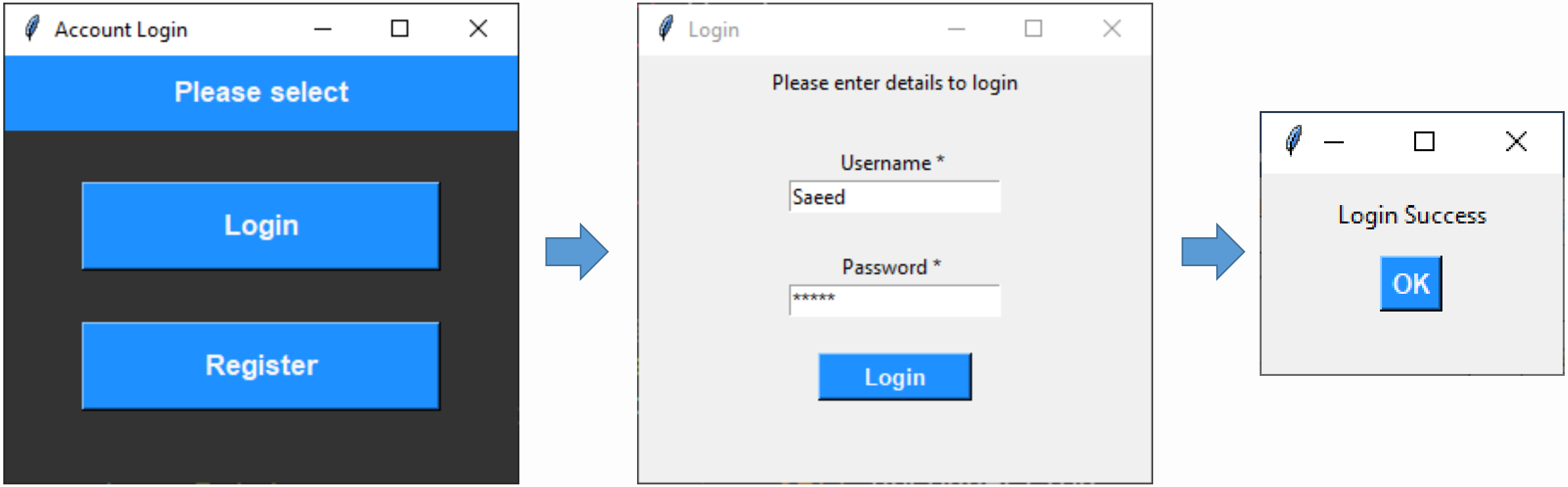
The application will be mainly used in retirement homes. Also, it can be extended and modified to any sort of application which deal with the different set of data from different segments e.g. day-care centers.

User manual

First for running the application you can either run the Retiree.ipynb file from Jupyter Notebook or you can open the file Retiree.py with Sublime text editor, hit *Ctrl+B* to run it more quickly (recommended). Next, you will have the homepage of the retirement management system which you can then interact with by the graphical user interface and each button that is associated with the specific function and will be discussed in the upcoming section.

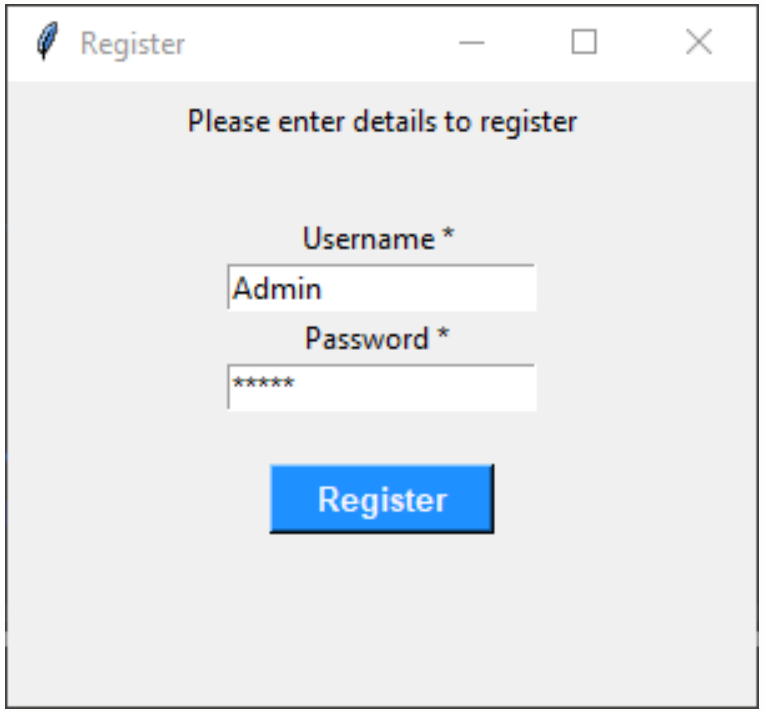
Screenshots

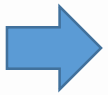
Login

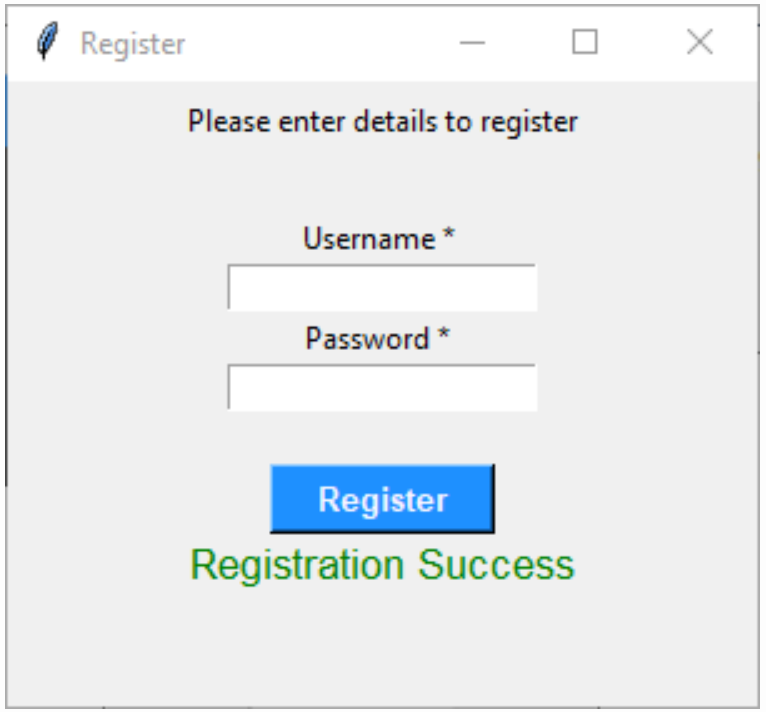


Screenshots

Register

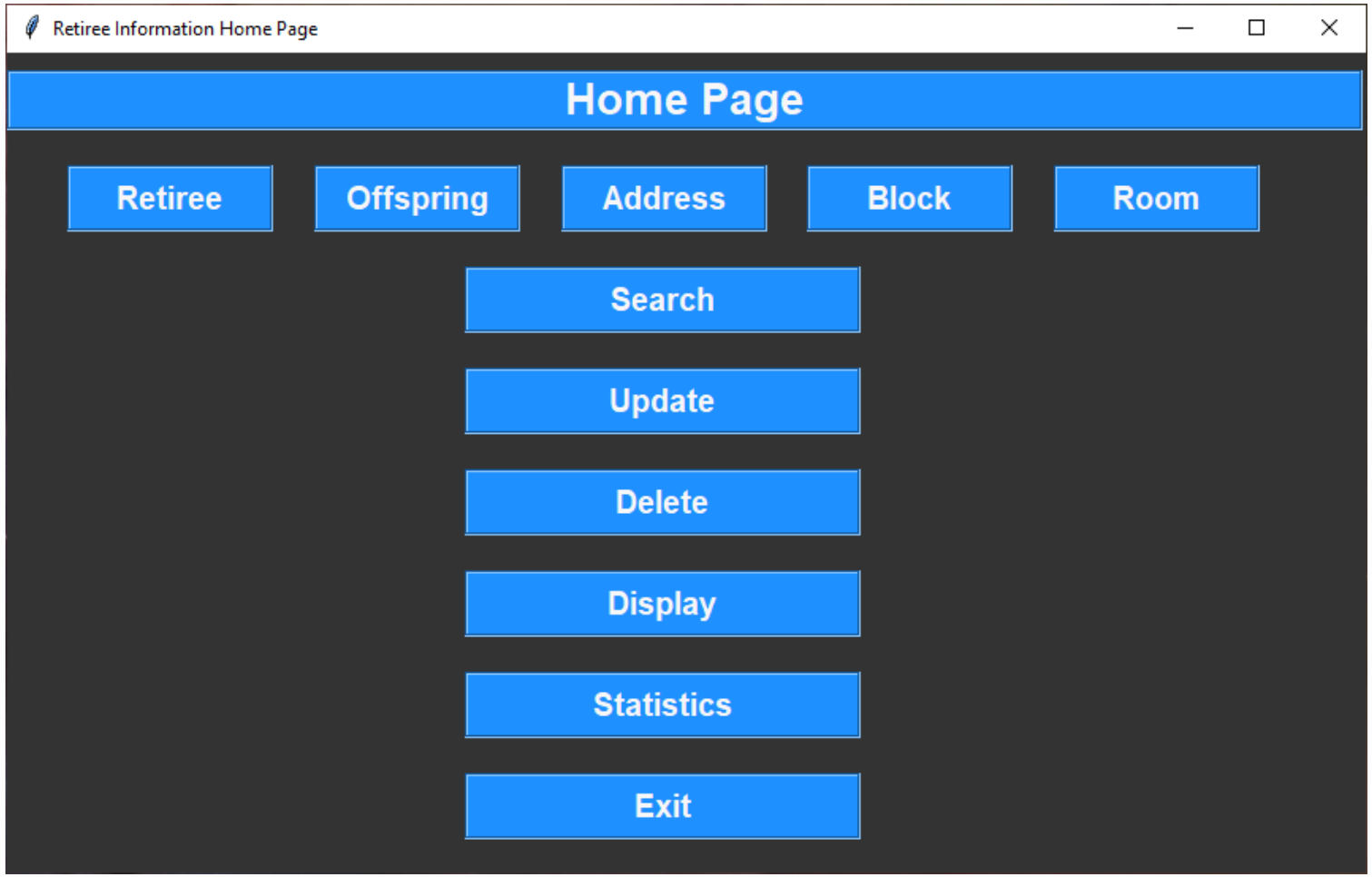






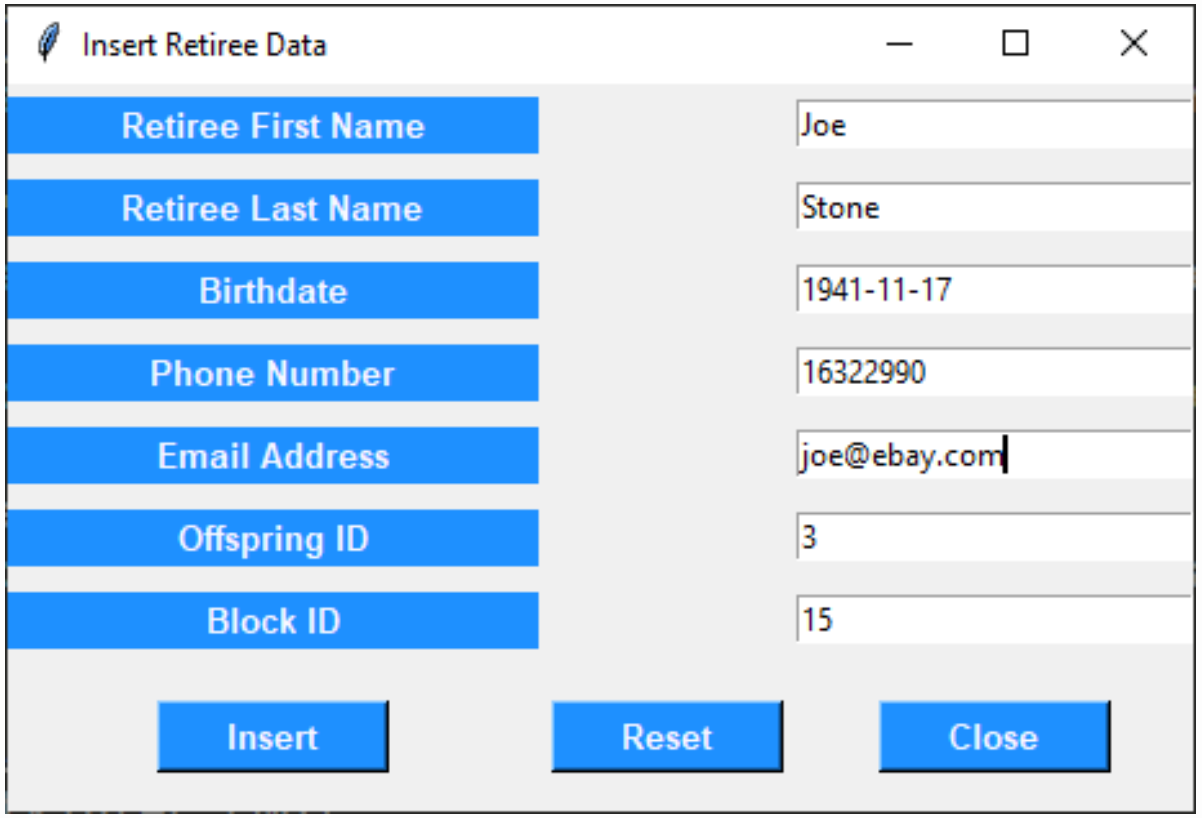
Screenshots

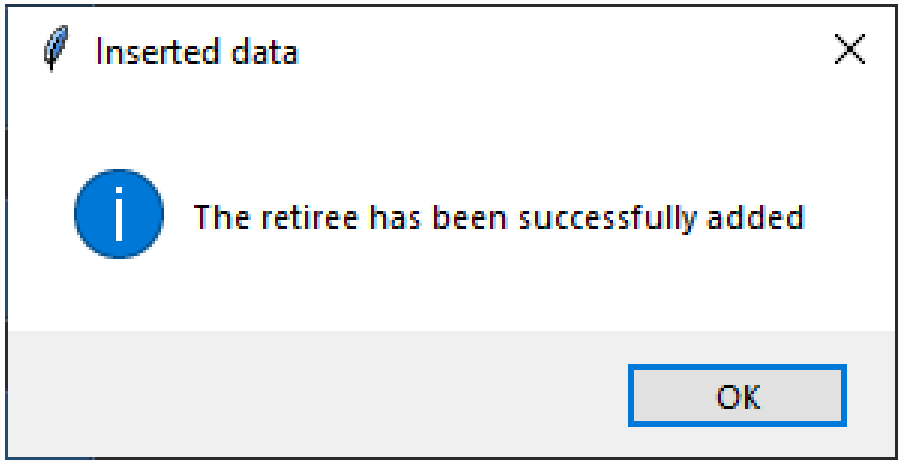
Homepage: The front page of the application



Screenshots

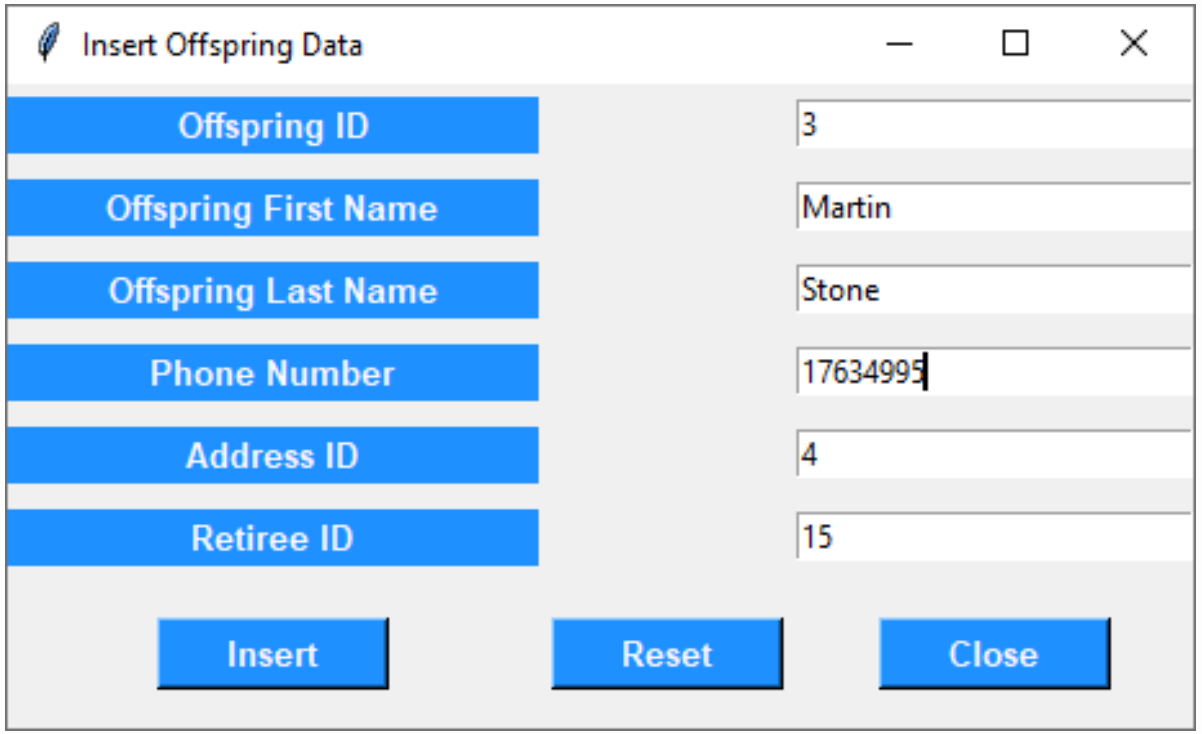
- **Inserts:** They are designed in a separate buttons and therefore windows in order to give the user the ability to enter the information from top to bottom (Retiree to Room) or from bottom to top (Room to Retiree)

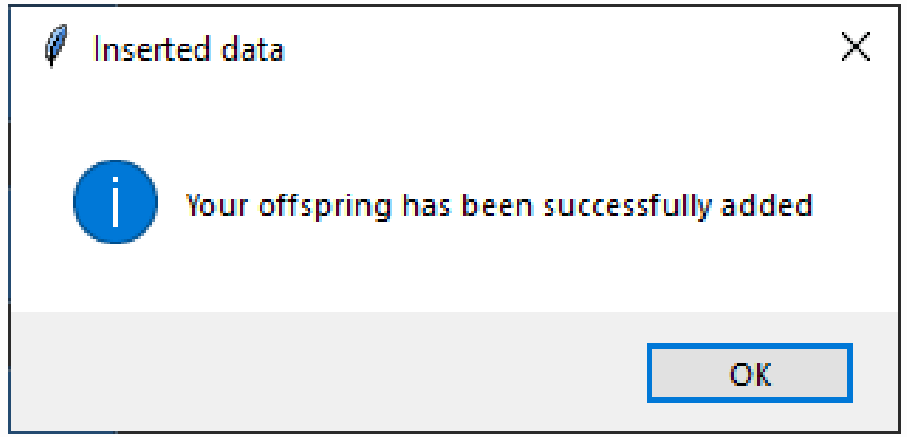




Screenshots

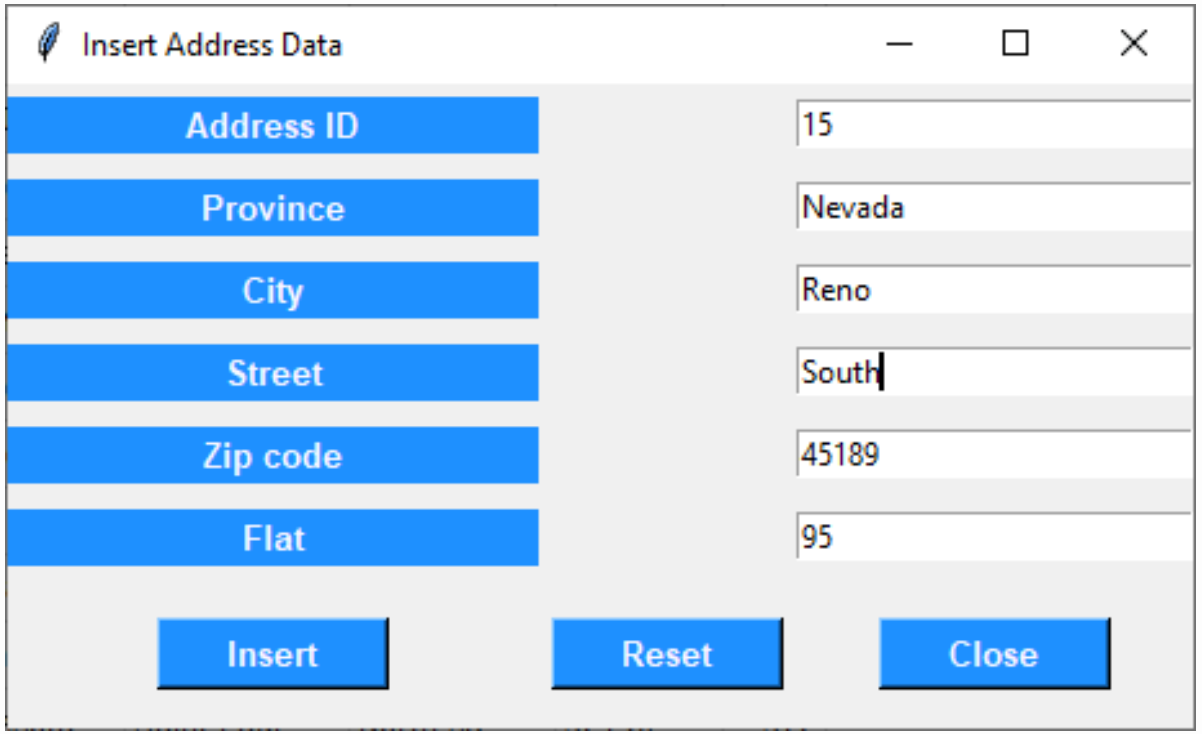
- **Inserts:** They are designed in a separate buttons and therefore windows in order to give the user the ability to enter the information from top to bottom (Retiree to Room) or from bottom to top (Room to Retiree)

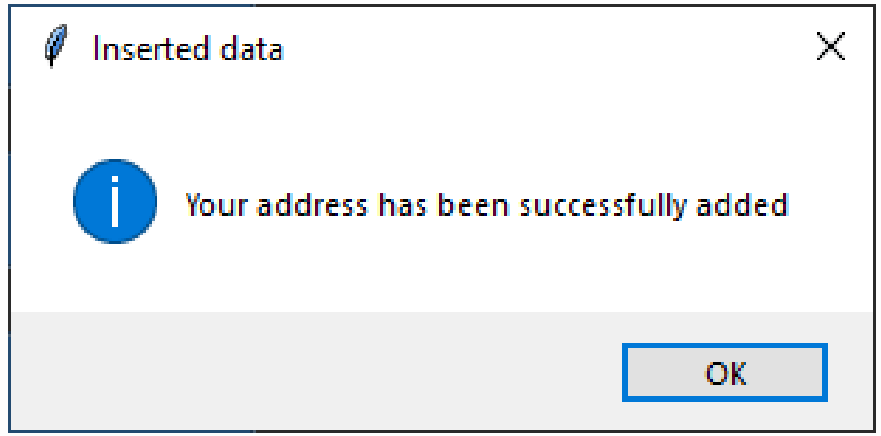




Screenshots

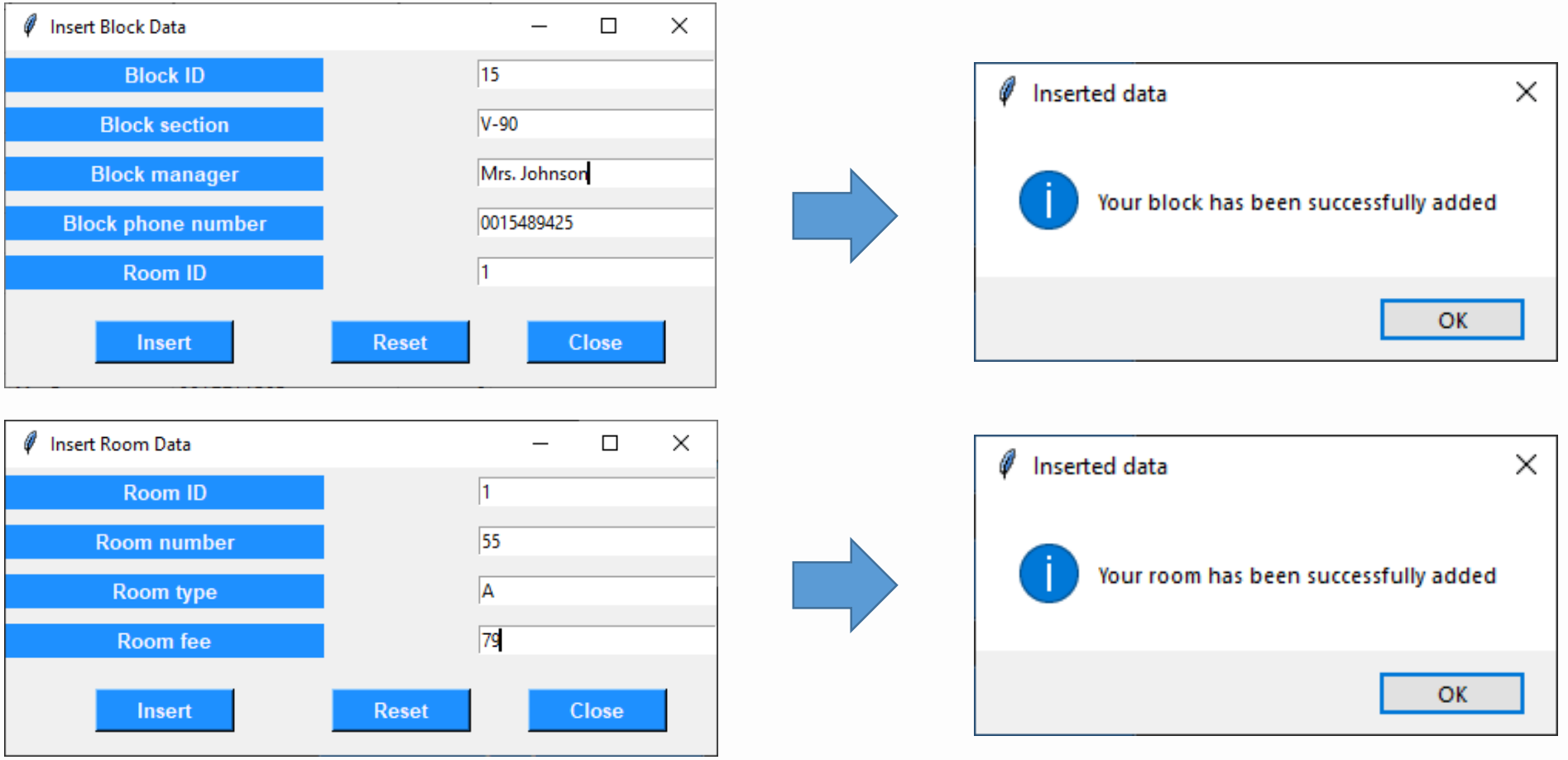
- **Inserts:** They are designed in a separate buttons and therefore windows in order to give the user the ability to enter the information from top to bottom (Retiree to Room) or from bottom to top (Room to Retiree)





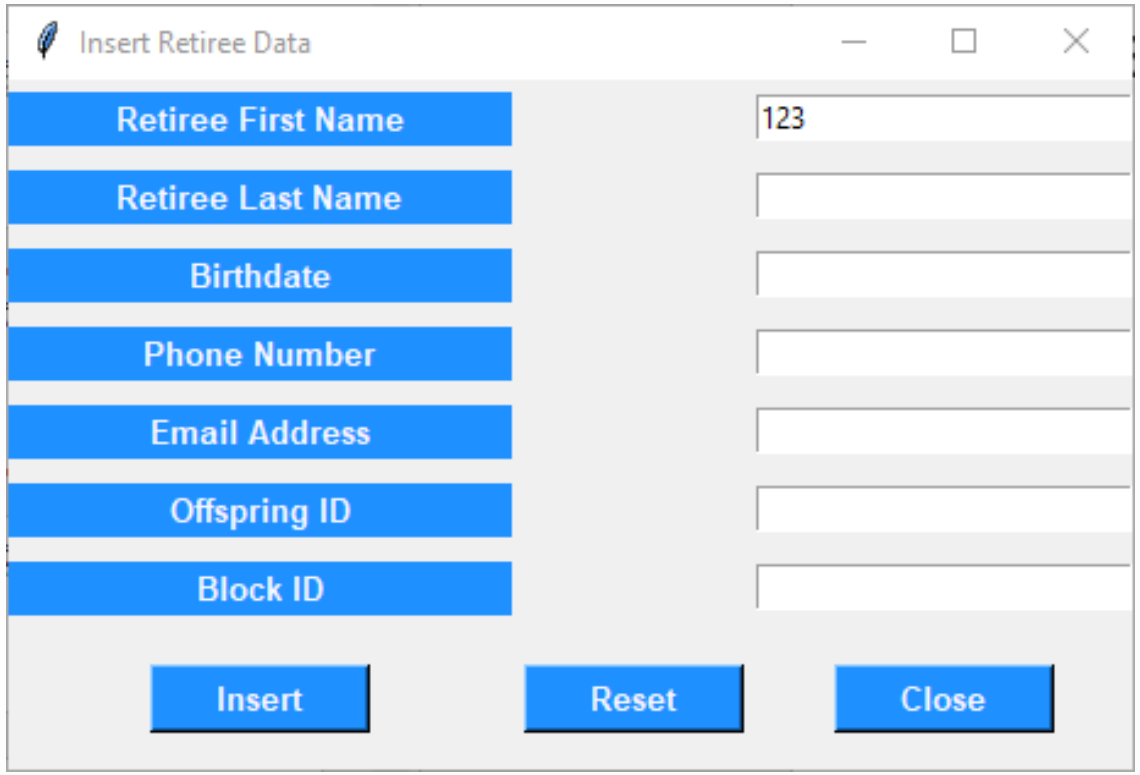
Screenshots

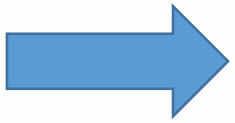
- **Inserts:** They are designed in a separate buttons and therefore windows in order to give the user the ability to enter the information from top to bottom (Retiree to Room) or from bottom to top (Room to Retiree)

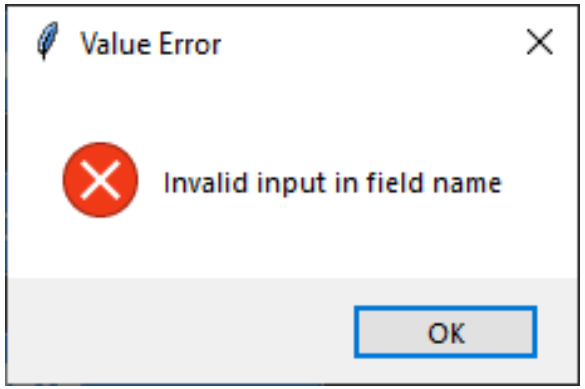


Screenshots

- **Inserts:** Will raise value error for wrong inputs

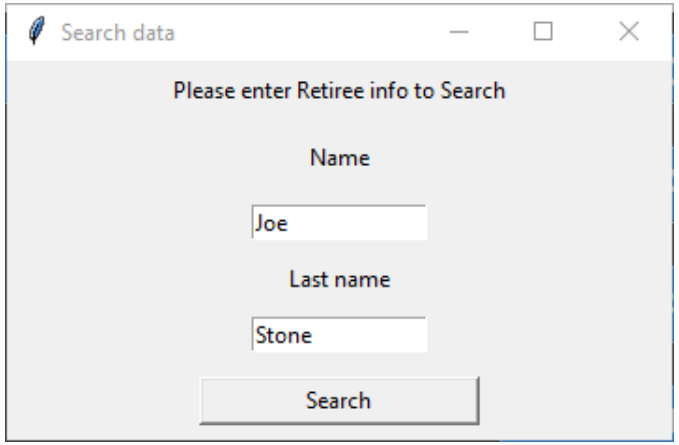


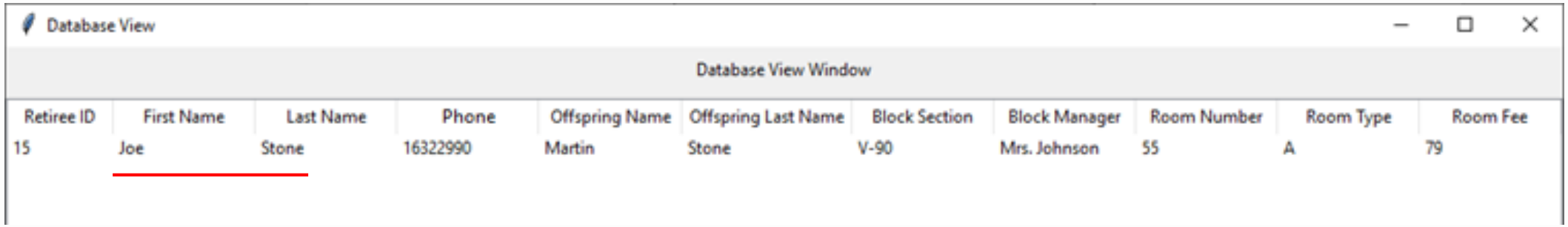




Screenshots

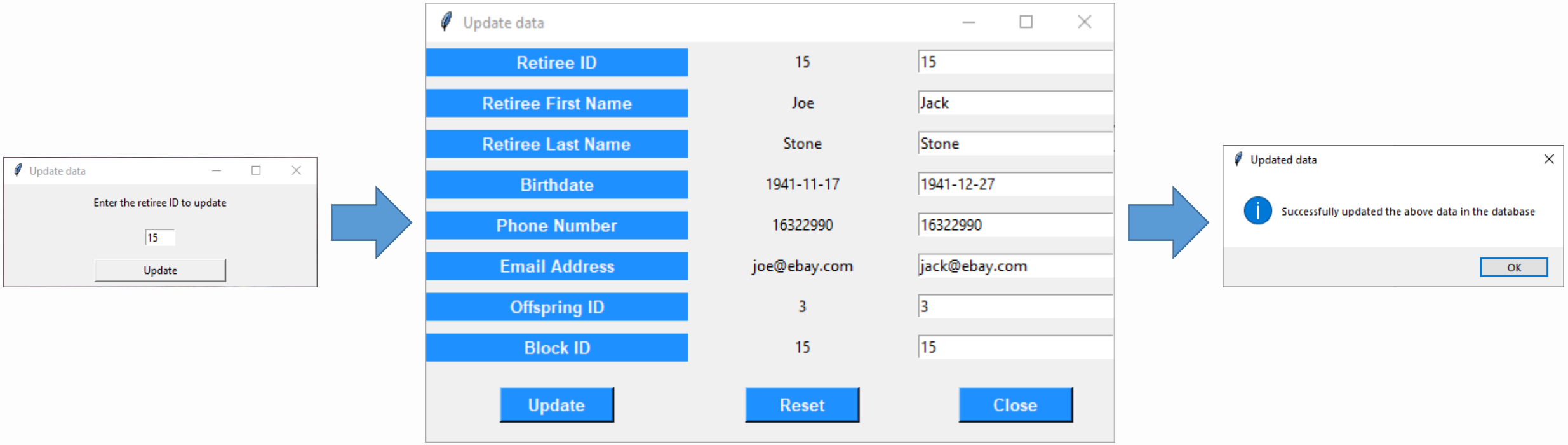
Search: By clicking on this button, we will be able to search the retirees based on their name and last name





Screenshots

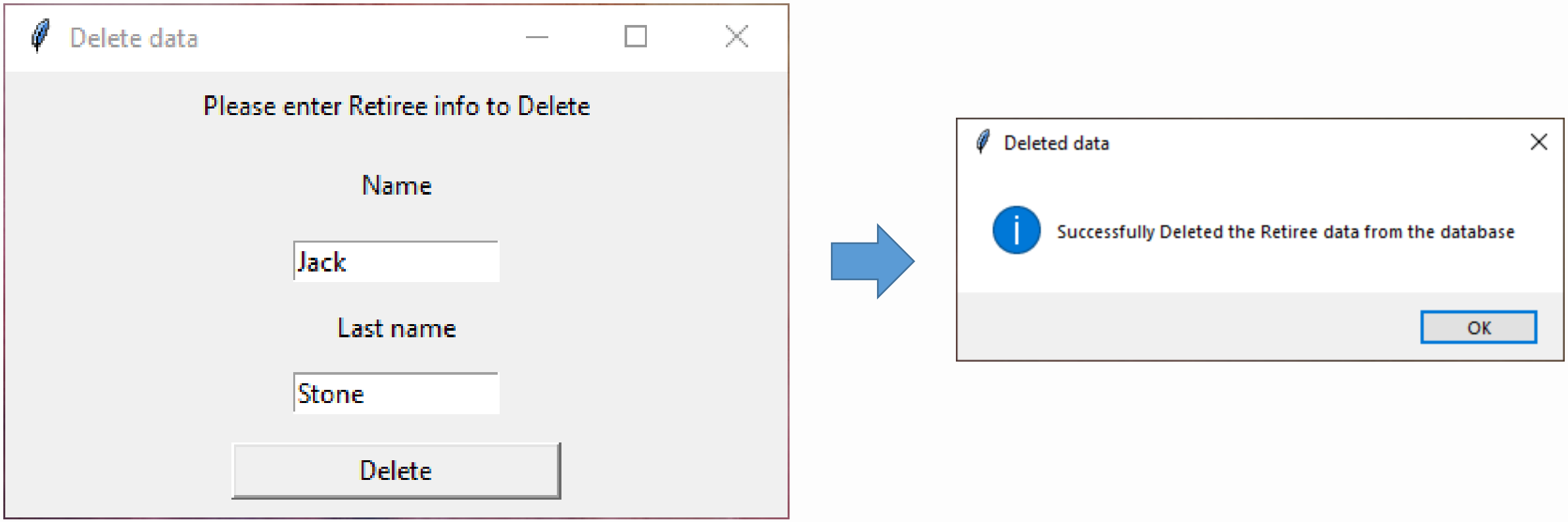
Update: Will update the information in Retiree table according to their ID. Previous values also shown in order to make it easier to compare the old and the new value.



14	15	Jack	Stone	1941-12-27	16322990	jack@ebay.com	3	15

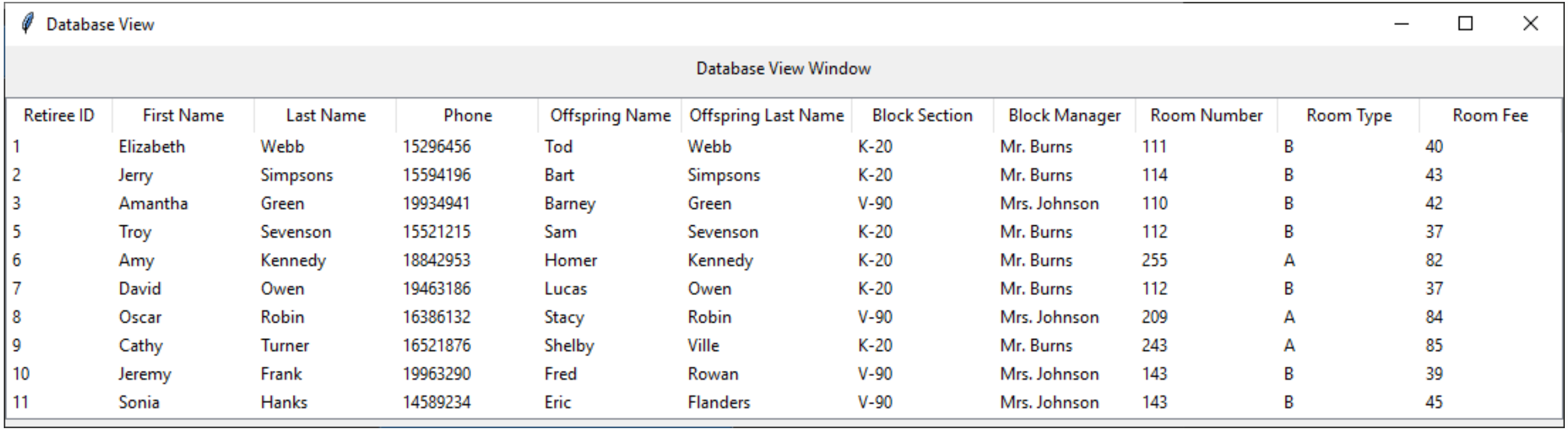
Screenshots

Delete: Will delete a retirees from Retiree table based on their name and surname



Screenshots

Display: Will display the scrollable information about retiree ID, name, last name, phone number, offspring name, offspring last name, block section, block manager, room number, room type and room fee.

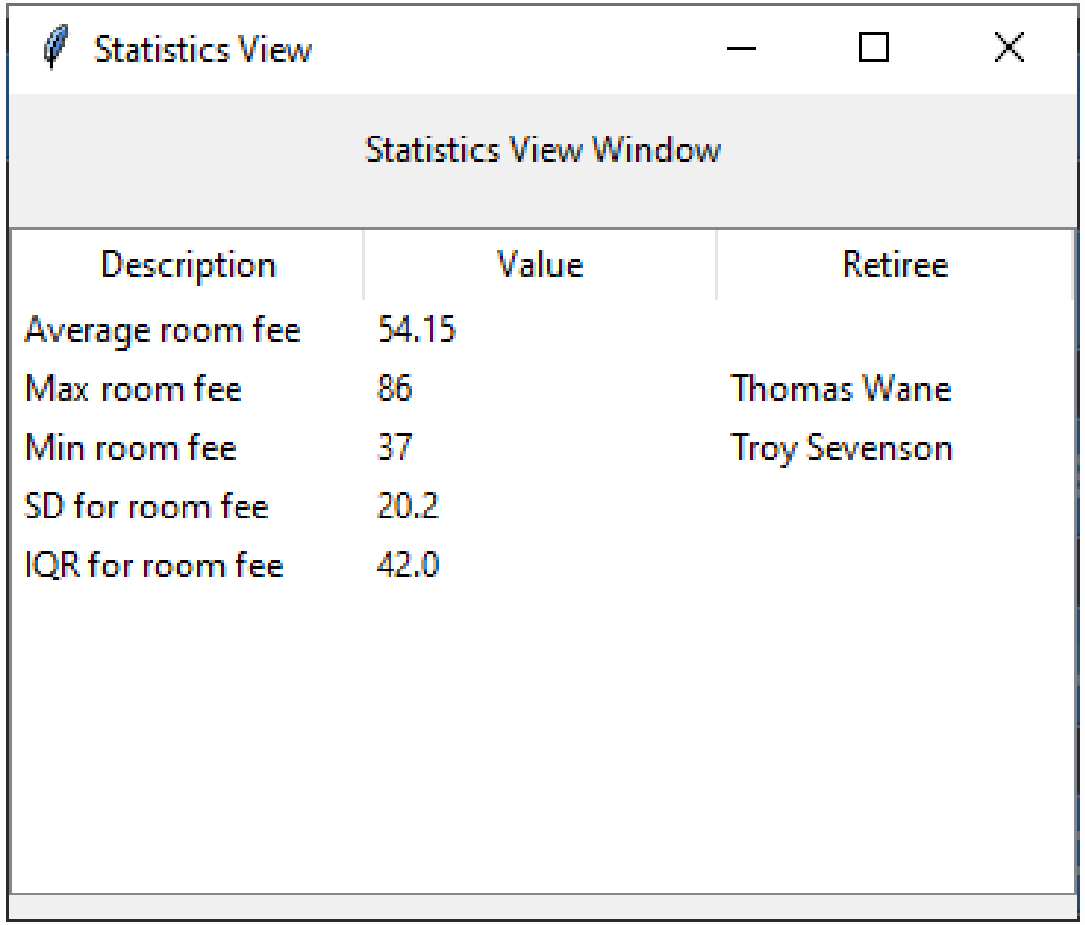


Retiree ID	First Name	Last Name	Phone	Offspring Name	Offspring Last Name	Block Section	Block Manager	Room Number	Room Type	Room Fee
1	Elizabeth	Webb	15296456	Tod	Webb	K-20	Mr. Burns	111	B	40
2	Jerry	Simpsons	15594196	Bart	Simpsons	K-20	Mr. Burns	114	B	43
3	Amantha	Green	19934941	Barney	Green	V-90	Mrs. Johnson	110	B	42
5	Troy	Sevenson	15521215	Sam	Sevenson	K-20	Mr. Burns	112	B	37
6	Amy	Kennedy	18842953	Homer	Kennedy	K-20	Mr. Burns	255	A	82
7	David	Owen	19463186	Lucas	Owen	K-20	Mr. Burns	112	B	37
8	Oscar	Robin	16386132	Stacy	Robin	V-90	Mrs. Johnson	209	A	84
9	Cathy	Turner	16521876	Shelby	Ville	K-20	Mr. Burns	243	A	85
10	Jeremy	Frank	19963290	Fred	Rowan	V-90	Mrs. Johnson	143	B	39
11	Sonia	Hanks	14589234	Eric	Flanders	V-90	Mrs. Johnson	143	B	45

Screenshots

Statistics: Will display the statistics from the database based on the room fee and return the following values:

- Average room fee
- Max/Min room fee: Which retiree paid the maximum/minimum price for his/her?
- Standard deviation for the room fee
- Inter quartile range for the room fee



Screenshots

SQL queries

Creating tables such as Retiree

```
self.dbCursor.execute("""CREATE TABLE IF NOT EXISTS Retiree (ret_id INTEGER PRIMARY KEY AUTOINCREMENT,
                         name TEXT NOT NULL,
                         last_name TEXT NOT NULL,
                         birth_date TEXT,
                         phone_number INTEGER,
                         email_address TEXT,
                         off_id INTEGER,
                         block_id INTEGER,
                         FOREIGN KEY (off_id)
                         REFERENCES Offspring (off_id),
                         FOREIGN KEY (block_id)
                         REFERENCES Block (block_id))""")
```

Screenshots

SQL queries

Creating indices

```
self.dbCursor.execute("""CREATE INDEX IF NOT EXISTS idx_Retiree_id ON Retiree (ret_id)""")
self.dbCursor.execute("""CREATE INDEX IF NOT EXISTS idx_name_lastname ON Retiree (name,last_name)""")
```

Search function with SQL query based on inner join

```
def Search(self, name, last_name):
    self.dbCursor.execute("""SELECT ret_id,name,last_name,phone_number,off_name,
        off_last_name,block_section,block_manager,room_number,room_type,room_fee FROM Retiree
        INNER JOIN Offspring ON Retiree.off_id = Offspring.off_id
        INNER JOIN Block ON Retiree.block_id = Block.block_id
        INNER JOIN Room ON Block.room_id = Room.room_id WHERE name = ? AND last_name = ?""",
        (name,last_name))
    searchResults = self.dbCursor.fetchall()
    return searchResults
```

Screenshots

SQL queries

Statistics function SQL query for retrieving the value of room fee based on three tables

```
def statistics(self):
    self.dbCursor.execute("""SELECT Room.room_fee FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return records
```

Screenshots

SQL queries

Max and Min query for room fee

```
def maxfee(self):
    self.dbCursor.execute("""SELECT name,last_name,MAX(Room.room_fee) FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return (records[0][0],records[0][1])

def minfee(self):
    self.dbCursor.execute("""SELECT name,last_name,MIN(Room.room_fee) FROM Retiree
        INNER JOIN Block ON Block.block_id = Retiree.block_id
        INNER JOIN Room ON Room.room_id = Block.room_id""")
    records = self.dbCursor.fetchall()
    return (records[0][0],records[0][1])
```

Screenshots

SQL queries

Creating a view named *v_mostData* for the most important fields of database

```
self.dbCursor.execute("""CREATE VIEW IF NOT EXISTS v_mostData AS SELECT ret_id,
    name,last_name,phone_number,off_name, off_last_name,block_section,block_manager,
    room_number,room_type,room_fee FROM Retiree
    INNER JOIN Offspring ON Retiree.off_id = Offspring.off_id
    INNER JOIN Block ON Retiree.block_id = Block.block_id
    INNER JOIN Room ON Block.room_id = Room.room_id UNION ALL SELECT * FROM Address""")
```

External packages

For this project, the Python's de facto standard GUI interface **Tkinter** is used which is a toolkit for developing applications in different operating systems also the **numpy** package is used for some statistical inference such as standard deviation and inter quartile range for the room fee.