Criterion C: Development

List of Complexities:

1. Libraries
2. Adding data to database
3. Updating data in database
4. Deleting data in database
5. Searching for specific data in database
6. Use of object-oriented design and classes
7. Use of 2D-arrays
8. Images
9. Use of multiple threads
10. Use of nested loops
11. Use of nested control functions
12. SQL commands used
13. Validation

Libraries:

```
### Importing Modules ###
from tkinter import *
from tkinter import ttk
from tkcalendar import *
from PIL import Image, ImageTk, ImageGrab
import datetime
import os
import sqlite3
from tkinter.filedialog import askopenfilename
from threading import Thread
import win32gui
```

Many libraries have been used to complete the project which have all helped to enhance the program's functionality. The libraries were useful in reducing the time spent on the front-end of the project (GUI) and helped to focus more on the back end of the project (functionality).

Purpose of Libraries Used:

Library Name	**Purpose of Library**
from tkinter import *	Imports tkinter classes for GUI
from tkinter import ttk	Imports a specific class of tkinter module for GUI
from tkcalendar import *	Allows the use of calendar
from PIL import Image, ImageTk, ImageGrab	Allows the use of images – get, update, and save – in the user's laptop
import datetime	Adds time and date to function

import os	Allows Python to interact with the operating systems and the files
import sqlite3	Allows the connection of the sqlite3 database with Python
from tkinter.filedialog import askopenfilename	Allows the user to find filenames on their laptop and get the file's location
from threading import Thread	Allows for multiprocessing and parallel running of programs
import win32gui	Allows access to Window's Component Object Model (COM) and can control some applications

Adding data to database:

```
def addUserData():
    global userFocus, userInfo

    saveUserImage()
    with open("./Images/user.png",'rb') as file:
        userBlob = file.read()

    # Connect to database
    connection = sqlite3.connect('restuarantDatabase.db')

    # Create a cursor
    cursor = connection.cursor()

    cursor.execute("""SELECT * FROM user""")
    userInfo = cursor.fetchall()

    genderCombobox['state'] = "normal"
    positionCombobox['state'] = "normal"
    if(userFocus == "customer"):
        connection.execute("""INSERT INTO user (
        id,position,first,last,gender,birthDay,birthMonth,birthYear,address,username,password,email,salary,joinDay,joinMonth,joinYear,bonus,image)
        VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)""", (len(userInfo)+1,'customer',firstNameEntry.get(),lastNameEntry.get(),genderCombobox['values'][genderCombobox.current()],
        DOBLabell.cget("text").split("/")[0],DOBLabell.cget("text").split("/")[1],DOBLabell.cget("text").split("/")[2],addressEntry.get(),userEntry.get(),passEntry.get(),emailEntry.get(),
        'N.A.','N.A.','N.A.','N.A.','N.A.',userBlob)
                           )
    else:
        connection.execute("""INSERT INTO user (
        id,position,first,last,gender,birthDay,birthMonth,birthYear,address,username,password,email,salary,joinDay,joinMonth,joinYear,bonus,image)
        VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)""", (len(userInfo)+1,positionCombobox['values'][positionCombobox.current()].lower(),firstNameEntry.get(),lastNameEntry.get(),
        genderCombobox['values'][genderCombobox.current()],DOBLabell.cget("text").split("/")[0],DOBLabell.cget("text").split("/")[1],DOBLabell.cget("text").split("/")[2],addressEntry.get(),
        userEntry.get(),passEntry.get(),emailEntry.get(),salaryEntry.get(),joinDateLabell.cget("text").split("/")[0],joinDateLabell.cget("text").split("/")[1],
        joinDateLabell.cget("text").split("/")[2],bonusEntry.get(),'N.A.')#userPhotol)
                           )
    genderCombobox['state'] = "readonly"
    positionCombobox['state'] = "readonly"
    emptyArrayText(entryNames)

    global userPhoto
    userPhoto = ImageTk.PhotoImage(Image.open(parentPath+"\\Program\\Images\\noUser.jpg").resize((200, 300), Image. ANTIALIAS))
    photoCanvas.itemconfigure(userImage, image=userPhoto)

    # Commit our command
    connection.commit()

    # Close our cursor
    cursor.close()
```

General Overview:

The addUserData() function opens the sqlite3 database. It then gets the information of a new user from the User Details Frame and adds it to the database.

Working of Code:

The connection connects the sqlite3 database to Python. The cursor allows complex logical operations by going through the table row by row. The userInfo converts row by row database into a 2D array. The gender and position combobox become editable when the state attribute is set to normal. If the user being focused is a customer, the salary, joinDay, joinMonth, joinYear and bonus fields get added as 'N.A.'. Otherwise, these fields are also included from the userDetails frame. The gender and position combobox are then converted

back to being uneditable when the state attribute is set to read-only. The connection to the database is lost when the connection is committed, and the cursor is closed.

Image:

Old Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	397.76	BLOB
2	2	manager	Robyn	Mitchell	Female	10	2	1921	Carrefour, Rosewood Road 784, El Paso, GEO 6005	NdX5w1	ic8dwMhVTvrp	Robyn_Mitchell8176@bulaffy.com	3102.96	19	7	1890	682.38	BLOB
3	3	waiter	Oliver	Tailor	Male	8	3	1859	Telekom, Buttonwood Way 7365, Lakewood, CRI 6616	FPLZP1	zgFdPb7dc0GF	Oliver_Tailor7151@vetan.org	4845.4	19	8	1933	631.97	BLOB
4	4	cook	Barney	Eastwood	Male	1	9	1872	Mars, Garfield Rue 129, Boston, COD 1207	lCfQql	izm2AAXxqhni	Barney_Eastwood3616@nickia.com	1771.08	30	4	1957	234.93	BLOB
5	5	customer	Rufus	Evans	Male	7	6	1942	Team Guard SRL, Anns Crossroad 3055, Los Angeles, ZWE 8266	m5VwhE	Mjcq5cWFdfhs	Rufus_Evans1822@atink.com	6302.34	27	1	2007	694.11	BLOB
6	6	admin	Teagan	Morris	Female	17	5	1843	UPC, Becklow Walk 7008, Reno, BOL 6722	L1gIRC	R208MmAplvz4	Teagan_Morris8718@acrit.org	5332.19	23	11	1825	577.08	BLOB
7	7	manager	Javier	Crawford	Male	28	7	2001	Team Guard SRL, Longman Grove 9816, San Bernardino, KAZ 1081	173vAY	iBr75rKnC4yz	Javier_Crawford4450@mafthy.com	4836.0	10	6	1869	489.29	BLOB
8	8	waiter	Alexa	Ingram	Female	25	9	1944	Comcast, Clavell Way 9521, Chicago, NGA 7888	SnWv1o	acfGjv2AocrK	Alexa_Ingram8111@dionrab.com	4367.0	21	11	1812	786.7	BLOB
9	9	cook	Miriam	Thomas	Female	19	7	1893	CarMax, Emily Rue 585, Columbus, GTM 5207	pI1jS8	YsF74niohc9W	Miriam_Thomas293@mafthy.com	9936.56	8	8	1892	682.99	BLOB
10	10	customer	Cedrick	Todd	Male	11	8	1909	CarMax, Lexington Way 4916, Long Beach, KEN 0413	OQbzG3	LKmBfQn0r0Z1	Cedrick_Todd6418@fulss.net	7080.01	12	2	1802	44.62	BLOB
11	11	admin	Tom	Watt	Male	21	11	1858	It Smart Group, Thrale Rue 9719, Reno, OMN 6743	K2IpN3	VzDRAGrAH8h8	Tom_Watt7481@cispeto.com	2145.18	20	9	1921	278.71	BLOB
12	12	manager	Martha	Truscott	Female	20	1	1801	ENEL, Carolina Pass 2425, Lincoln, MCO 5763	xC8N9O	XI5LMh3bmfJM	Martha_Truscott6250@typill.biz	2030.31	21	6	1953	558.93	BLOB
13	13	waiter	Nate	Fisher	Male	25	3	1977	Global Print, Champion Crossroad 8032, Huntsville, ARM 2547	bN5OdW	eiaS2pqH48EX	Nate_Fisher2830@bulaffy.com	8282.39	11	1	1992	588.81	BLOB
14	14	cook	Gina	Townend	Female	17	11	1826	ExxonMobil, Western Drive 8976, Lancaster, MOZ 6785	3UnED1	2Y3kykn8sUf	Gina_Townend8565@acrit.org	5424.82	10	9	1807	992.39	BLOB
15	15	customer	George	Thorpe	Male	17	3	1972	Team Guard SRL, Bendall Grove 3142, Fullerton, VUT 2886	PBkJEC	0CDdtKVf9BY2	George_Thorpe4209@joiniaa.com	1450.96	20	7	1868	149.87	BLOB
16	16	admin	Courtney	Bennett	Female	21	6	1906	21st Century Fox, Bayberry Pass 151, Lisbon, LTU 8283	bedoRW	905KW82Yf0er	Courtney_Bennett2189@hourpy.biz	4121.19	27	11	1987	468.93	BLOB
17	17	manager	Sofia	Tanner	Female	16	5	1849	Team Guard SRL, Tiptree Route 955, Arlington, PNG 3417	OGGEPl	g0ybYjGNNyGv	Sofia_Tanner7729@extex.org	3807.19	25	7	1835	412.54	BLOB
18	18	waiter	Moira	Farmer	Female	12	2	1874	Apple Inc., Marina Drive 8206, Lancaster, CAF 1744	jtLgCR	e40DpS70TdGf	Moira_Farmer1527@naiker.biz	4641.39	20	8	2022	487.84	BLOB
19	19	cook	Penny	Tyrrell	Female	21	9	1924	ExxonMobil, Enford Alley 6535, Lincoln, SVK 1415	ntpGLF	gibsXGorLly	Penny_Tyrrell9742@gompie.com	6705.83	12	3	1941	853.57	BLOB
20	20	customer	Megan	Long	Female	26	3	1890	Amazon.com, Aylward Boulevard 8309, Louisville, CPV 3871	Sgi5xV	aeUgfnxsa752	Megan_Long4617@bauros.biz	4172.98	22	2	1963	710.05	BLOB
21	21	admin	Raquel	Lambert	Female	30	10	1803	ExxonMobil, Cleveland Pass 8895, Escondido, EST 5344	heHwTx	XNF5BR3WbILR	Raquel_Lambert6776@twace.org	7197.91	18	9	1933	465.01	BLOB
22	22	manager	Ryan	Preston	Male	4	11	1905	Coca-Cola Company, Gatonby Way 9930, Milano, CYM 3461	TsUym4	YM0xGxr0AhBu	Ryan_Preston8728@nickia.com	4007.18	22	8	1953	24.98	BLOB
23	23	waiter	Anabel	Hammond	Female	30	8	1920	Telekom, Boldero Vale 3589, Salem, ARG 6885	ljAxJ9	pIl6oG9IQDbt	Anabel_Hammond7347@sveldo.biz	8703.75	7	2	1957	915.57	BLOB
24	24	cook	Rosemary	Upsdell	Female	30	4	1975	CarMax, Argyle Pass 8711, Tulsa, CAN 0016	AP0w0V	0HOaWrvr37wE	Rosemary_Upsdell9924@bretoux.com	9980.75	14	4	2002	348.51	BLOB
25	25	customer	Cassidy	Clarke	Female	7	11	2002	Erickson, Fawn Crossroad 4080, Tallahassee, ZMB 0854	7G1uQx	f3f0j9j2Jbpe	Cassidy_Clarke3762@nickia.com	4049.86	3	5	1954	7.59	BLOB

Updated Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	500.0	BLOB
2	2	manager	Robyn	Mitchell	Female	10	2	1921	Carrefour, Rosewood Road 784, El Paso, GEO 6005	NdX5w1	ic8dwMhVTvrp	Robyn_Mitchell8176@bulaffy.com	3102.96	19	7	1890	682.38	BLOB
3	3	waiter	Oliver	Tailor	Male	8	3	1859	Telekom, Buttonwood Way 7365, Lakewood, CRI 6616	FPLZP1	zgFdPb7dc0GF	Oliver_Tailor7151@vetan.org	4845.4	19	8	1933	631.97	BLOB
4	4	cook	Barney	Eastwood	Male	1	9	1872	Mars, Garfield Rue 129, Boston, COD 1207	lCfQql	izm2AAXxqhni	Barney_Eastwood3616@nickia.com	1771.08	30	4	1957	234.93	BLOB
5	5	customer	Rufus	Evans	Male	7	6	1942	Team Guard SRL, Anns Crossroad 3055, Los Angeles, ZWE 8266	m5VwhE	Mjcq5cWFdfhs	Rufus_Evans1822@atink.com	6302.34	27	1	2007	694.11	BLOB
6	6	admin	Teagan	Morris	Female	17	5	1843	UPC, Becklow Walk 7008, Reno, BOL 6722	LIgIRC	R208MmAplvz4	Teagan_Morris8718@acrit.org	5332.19	23	11	1825	577.08	BLOB
7	7	manager	Javier	Crawford	Male	28	7	2001	Team Guard SRL, Longman Grove 9816, San Bernardino, KAZ 1081	173vAY	iBr75rKnC4yz	Javier_Crawford4450@mafthy.com	4836.0	10	6	1869	489.29	BLOB
8	8	waiter	Alexa	Ingram	Female	25	9	1944	Comcast, Clavell Way 9521, Chicago, NGA 7888	SnWv1o	acfGjv2AocrK	Alexa_Ingram8111@dionrab.com	4367.0	21	11	1812	786.7	BLOB
9	9	cook	Miriam	Thomas	Female	19	7	1893	CarMax, Emily Rue 585, Columbus, GTM 5207	pI1jS8	YsF74niohc9W	Miriam_Thomas293@mafthy.com	9936.56	8	8	1892	682.99	BLOB
10	10	customer	Cedrick	Todd	Male	11	8	1909	CarMax, Lexington Way 4916, Long Beach, KEN 0413	OQbzG3	LKm8fQn0r0Z1	Cedrick_Todd6418@fulss.net	7080.01	12	2	1802	44.62	BLOB
11	11	admin	Tom	Watt	Male	21	11	1858	It Smart Group, Thrale Rue 9719, Reno, OMN 6743	K2IpN3	VzDRAGrAH8h8	Tom_Watt7481@cispeto.com	2145.18	20	9	1921	278.71	BLOB
12	12	manager	Martha	Truscott	Female	20	1	1801	ENEL, Carolina Pass 2425, Lincoln, MCO 5763	xC8N9O	Xl5LMh3bmfIM	Martha_Truscott6250@typill.biz	2030.31	21	6	1953	558.93	BLOB
13	13	waiter	Nate	Fisher	Male	25	3	1977	Global Print, Champion Crossroad 8032, Huntsville, ARM 2547	bN5OdW	eiaS2pqH48EX	Nate_Fisher2830@bulaffy.com	8282.39	11	1	1992	588.81	BLOB
14	14	cook	Gina	Townend	Female	17	11	1826	ExxonMobil, Western Drive 8976, Lancaster, MOZ 6785	3UnED1	2Y3kyKn8sUf	Gina_Townend8565@acrit.org	5424.82	10	9	1807	992.39	BLOB
15	15	customer	George	Thorpe	Male	17	3	1972	Team Guard SRL, Bendall Grove 3142, Fullerton, VUT 2886	PBkJEC	0CDdtKVf9BY2	George_Thorpe4209@joiniaa.com	1450.96	20	7	1868	149.87	BLOB
16	16	admin	Courtney	Bennett	Female	21	6	1906	21st Century Fox, Bayberry Pass 151, Lisbon, LTU 8283	bedoRW	905KWB2Yf0er	Courtney_Bennett2189@hourpy.biz	4121.19	27	11	1987	468.93	BLOB
17	17	manager	Sofia	Tanner	Female	16	5	1849	Team Guard SRL, Tiptree Route 955, Arlington, PNG 3417	OGGEPl	g0ybYjGNNyGv	Sofia_Tanner7729@extex.org	3807.19	25	7	1835	412.54	BLOB
18	18	waiter	Moira	Farmer	Female	12	2	1874	Apple Inc., Marina Drive 8206, Lancaster, CAF 1744	jtLgCR	e40DpS70TdGf	Moira_Farmer1527@naiker.biz	4641.39	20	8	2022	487.84	BLOB
19	19	cook	Penny	Tyrrell	Female	21	9	1924	ExxonMobil, Enford Alley 6535, Lincoln, SVK 1415	ntpGLF	gibsXGorLly	Penny_Tyrrell9742@gompie.com	6705.83	12	3	1941	853.57	BLOB
20	20	customer	Megan	Long	Female	26	3	1890	Amazon.com, Aylward Boulevard 8309, Louisville, CPV 3871	Sgi5xV	aeUgfnxsa752	Megan_Long4617@bauros.biz	4172.98	22	2	1963	710.05	BLOB
21	21	admin	Raquel	Lambert	Female	30	10	1803	ExxonMobil, Cleveland Pass 8895, Escondido, EST 5344	heHwTx	XNFSBR3WblLR	Raquel_Lambert6776@twace.org	7197.91	18	9	1933	465.01	BLOB
22	22	manager	Ryan	Preston	Male	4	11	1905	Coca-Cola Company, Gatonby Way 9930, Milano, CYM 3461	TsUym4	YM0xGxr0Ah8u	Ryan_Preston8728@nickia.com	4007.18	22	8	1953	24.98	BLOB
23	23	waiter	Anabel	Hammond	Female	30	8	1920	Telekom, Boldero Vale 3589, Salem, ARG 6885	lJAxJ9	pIl6oG9IQDbt	Anabel_Hammond7347@sveldo.biz	8703.75	7	2	1957	915.57	BLOB
24	24	cook	Rosemary	Upsdell	Female	30	4	1975	CarMax, Argyle Pass 8711, Tulsa, CAN 0016	AP0w0V	0HOaWrvr37wE	Rosemary_Upsdell9924@bretoux.com	9980.75	14	4	2002	348.51	BLOB
25	25	customer	Cassidy	Clarke	Female	7	11	2002	Erickson, Fawn Crossroad 4080, Tallahassee, ZMB 0854	7G1uQx	f3f0j9j2Jbpe	Cassidy_Clarke3762@nickia.com	4049.86	3	5	1954	7.59	BLOB
26	26	admin	Barney	Addley	Male	29	8	1981	Leadertech Consulting, Glenwood Alley 3063, Minneapolis, BVT 8278	gl5JFB	aEPMjSxUQbZS	Barney_Addley527@famism.biz	3767.53	20	10	1984	161.0	BLOB

The record that is added has been highlighted

Updating data to database:

```
def updateUserData():
    global userInfo

    # Connect to database
    connection = sqlite3.connect('restuarantDatabase.db')

    # Create a cursor
    cursor = connection.cursor()

    cursor.execute("""SELECT * FROM user""")
    userInfo = cursor.fetchall()
    for users in range(len(userInfo)):
        userNumber = 'user'+str(users+1)
        if(globals()[userNumber].id == int(listbox.get(listbox.curselection()).split(")")[0])):
            genderCombobox['state'] = "normal"
            connection.execute('''UPDATE user SET first = ?, last = ?, gender = ?, birthDay = ?, birthMonth = ?, birthYear = ?, address = ?, username = ?, password = ?, email = ?, salary = ?
            genderCombobox['state'] = "readonly"
        del globals()[userNumber]

    # Commit our command
    connection.commit()

    # Close our cursor
    cursor.close()

    # Close our connection
    connection.close()

    connection = sqlite3.connect('restuarantDatabase.db')
    cursor = connection.cursor()
    cursor.execute("""SELECT * FROM user""")
    userInfo = cursor.fetchall()
    for users in range(len(userInfo)):
        userNumber = 'user'+str(users+1)
        globals()[userNumber] = Users(userInfo[users][0],userInfo[users][1],userInfo[users][2],userInfo[users][3],userInfo[users][4],userInfo[users][5],userInfo[users][6],userInfo[users][7],
    userList()
    connection.commit()
    cursor.close()
    connection.close()
```

General Overview:

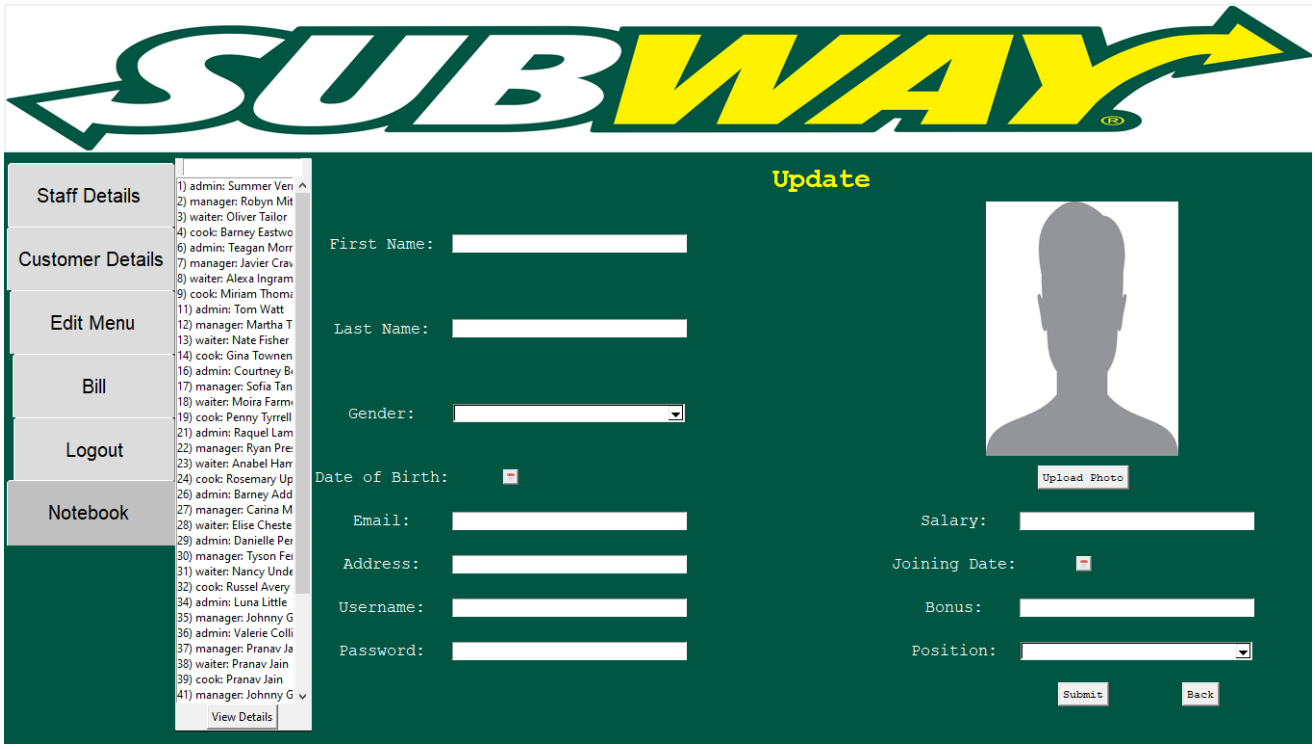
The updateUserData() function opens the sqlite3 database. It then gets the all the information of the user from the User Details Frame and updates the database at the user's unique id number.

Working of Code:

The connection connects the sqlite3 database to Python. The cursor allows complex logical operations by going through the table row by row. The userInfo converts row by row database into a 2D array. The id in

each row of userInfo is checked with the id shown in listbox selection. The gender combobox become editable when the state attribute is set to normal. The details of the user is updated with the new details in the userDetails frame. The gender combobox is then converted back to being uneditable when the state attribute is set to read-only. The connection to the database is lost when the connection is committed, and the cursor is closed. The connection is then again established to update the userInfo with the updated information and the user listbox is updated with the new information. The connection is then committed, and the cursor is closed.

Image:



Old Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	397.76	BLOB

Updated Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	500.0	BLOB

The updated quantity is highlighted

Deleting data in database:

```
def deleteUserData():
    global userInfo

    # Connect to database
    connection = sqlite3.connect('restuarantDatabase.db')

    # Create a cursor
    cursor = connection.cursor()

    cursor.execute("""SELECT * FROM user""")
    userInfo = cursor.fetchall()
    for users in range(len(userInfo)):
        userNumber = 'user'+str(users+1)
        if(globals()[userNumber].username == userEntry.get()):
            genderCombobox['state'] = "normal"
            deleteQuery = "DELETE from user where id = " + str(globals()[userNumber].id)
            connection.execute(deleteQuery)
            genderCombobox['state'] = "readonly"
            emptyArrayText(entryNames)

            for index in range(users+1, len(userInfo)):
                userNumber = 'user'+str(index+1)
                connection.execute('''UPDATE user SET id = ? WHERE id = ?''', ((int(globals()[userNumber].id) - 1), (int(globals()[userNumber].id))))
                globals()[userNumber].id = int(globals()[userNumber].id) - 1


    # Commit our command
    connection.commit()

    # Close our cursor
    cursor.close()

    # Close our connection
    connection.close()

    connection = sqlite3.connect('restuarantDatabase.db')
    cursor = connection.cursor()
    cursor.execute("""SELECT * FROM user""")
    userInfo = cursor.fetchall()
    for users in range(len(userInfo)):
        userNumber = 'user'+str(users+1)
        globals()[userNumber] = Users(userInfo[users][0],userInfo[users][1],userInfo[users][2],userInfo[users][3],userInfo[users][4],userInfo[users][5],userInfo[users][6],userInfo[users][7]
    userList()
    connection.commit()
    cursor.close()
    connection.close()
```

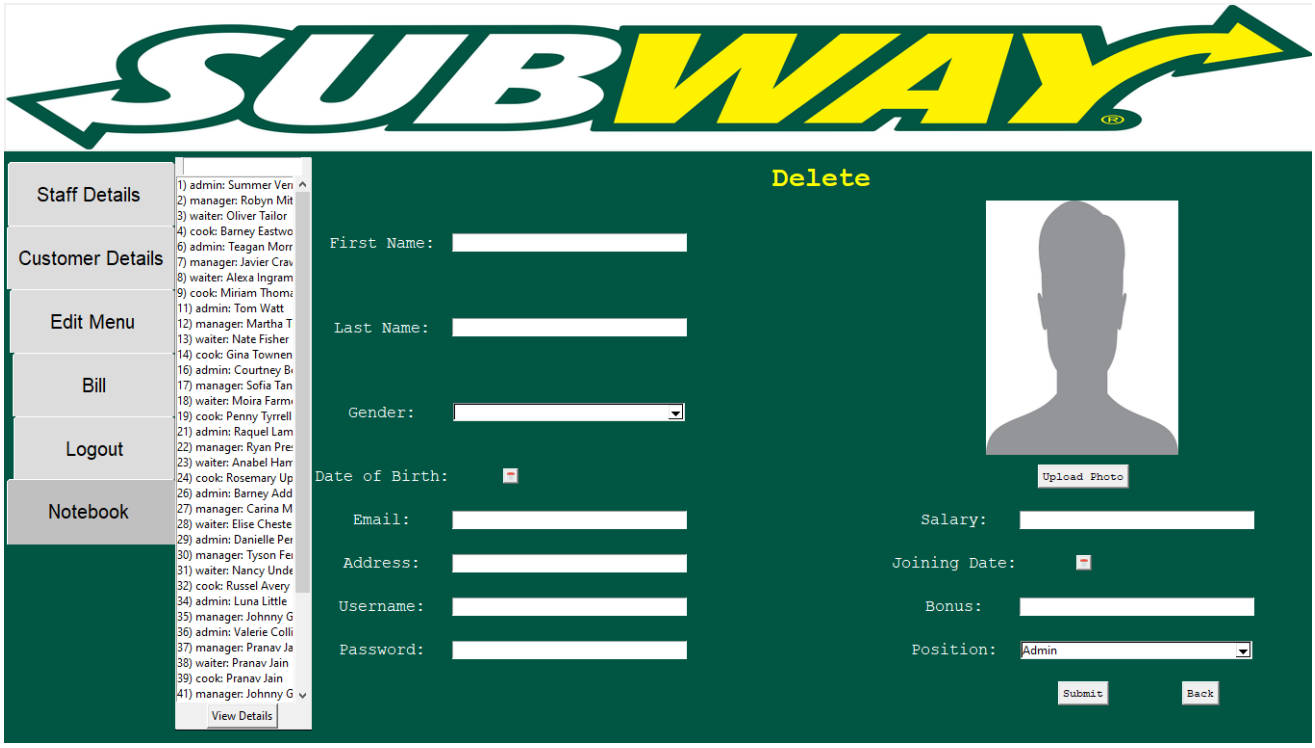
General Overview:

The deleteUserData() function opens the sqlite3 database. It then finds the user id in the database by comparing the usernames and deletes the user. It then regenerates the rest of the users, after the deleted user, by giving them a new id which is the (old id – 1). It also updates the userInfo variable after the user has been deleted.

Working of Code:

The connection connects the sqlite3 database to Python. The cursor allows complex logical operations by going through the table row by row. The userInfo converts row by row database into a 2D array. The username in each row of userInfo is checked with the username shown in username entry. The gender combobox become editable when the state attribute is set to normal. The user is deleted at the specific id. The gender combobox is then converted back to being uneditable when the state attribute is set to read-only. The entry fields are then emptied. The users after the deleted id are then updated with a new id which is (old id -1). The connection to the database is lost when the connection is committed, and the cursor is closed. The connection is then again established to update the userInfo with the updated information and the user listbox is updated with the new information. The connection is then committed, and the cursor is closed.

Image:



Old Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	500.0	BLOB
2	2	manager	Robyn	Mitchell	Female	10	2	1921	Carrefour, Rosewood Road 784, El Paso, GEO 6005	NdX5w1	ic8dwMhVTvrp	Robyn_Mitchell8176@bulaffy.com	3102.96	19	7	1890	682.38	BLOB
3	3	waiter	Oliver	Tailor	Male	8	3	1859	Telekom, Buttonwood Way 7365, Lakewood, CRI 6616	FPLZP1	zgFdPb7dc0GF	Oliver_Tailor7151@vetan.org	4845.4	19	8	1933	631.97	BLOB
4	4	cook	Barney	Eastwood	Male	1	9	1872	Mars, Garfield Rue 129, Boston, COD 1207	lCfQql	izm2AAXxqhni	Barney_Eastwood3616@nickia.com	1771.08	30	4	1957	234.93	BLOB
5	5	customer	Rufus	Evans	Male	7	6	1942	Team Guard SRL, Anns Crossroad 3055, Los Angeles, ZWE 8266	mSVwhE	MjcqScWFdfhs	Rufus_Evans1822@atink.com	6302.34	27	1	2007	694.11	BLOB
6	6	admin	Teagan	Morris	Female	17	5	1843	UPC, Becklow Walk 7008, Reno, BOL 6722	LIgIRC	R208MmAplvz4	Teagan_Morris8718@acrit.org	5332.19	23	11	1825	577.08	BLOB
7	7	manager	Javier	Crawford	Male	28	7	2001	Team Guard SRL, Longman Grove 9816, San Bernardino, KAZ 1081	173vAY	iBr7SrKnC4yz	Javier_Crawford4450@mafthy.com	4836.0	10	6	1869	489.29	BLOB
8	8	waiter	Alexa	Ingram	Female	25	9	1944	Comcast, Clavell Way 9521, Chicago, NGA 7888	SnWv1o	acfGjv2AocrK	Alexa_Ingram8111@dionrab.com	4367.0	21	11	1812	786.7	BLOB
9	9	cook	Miriam	Thomas	Female	19	7	1893	CarMax, Emily Rue 585, Columbus, GTM 5207	pI1jS8	YsF74niohc9W	Miriam_Thomas293@mafthy.com	9936.56	8	8	1892	682.99	BLOB
10	10	customer	Cedrick	Todd	Male	11	8	1909	CarMax, Lexington Way 4916, Long Beach, KEN 0413	OQbzG3	LKm8fQn0r0Z1	Cedrick_Todd6418@fulss.net	7080.01	12	2	1802	44.62	BLOB
11	11	admin	Tom	Watt	Male	21	11	1858	It Smart Group, Thrale Rue 9719, Reno, OMN 6743	K2IpN3	VzDRAGrAH8h8	Tom_Watt7481@cispeto.com	2145.18	20	9	1921	278.71	BLOB
12	12	manager	Martha	Truscott	Female	20	1	1801	ENEL, Carolina Pass 2425, Lincoln, MCO 5763	xC8N9O	Xl5LMh3bmfIM	Martha_Truscott6250@typill.biz	2030.31	21	6	1953	558.93	BLOB
13	13	waiter	Nate	Fisher	Male	25	3	1977	Global Print, Champion Crossroad 8032, Huntsville, ARM 2547	bN5OdW	eiaS2pqH48EX	Nate_Fisher2830@bulaffy.com	8282.39	11	1	1992	588.81	BLOB
14	14	cook	Gina	Townend	Female	17	11	1826	ExxonMobil, Western Drive 8976, Lancaster, MOZ 6785	3UnED1	2Y3kykn8sUf	Gina_Townend8565@acrit.org	5424.82	10	9	1807	992.39	BLOB
15	15	customer	George	Thorpe	Male	17	3	1972	Team Guard SRL, Bendall Grove 3142, Fullerton, VUT 2886	PBkJEC	0CDdtKVf9BY2	George_Thorpe4209@joiniaa.com	1450.96	20	7	1868	149.87	BLOB
16	16	admin	Courtney	Bennett	Female	21	6	1906	21st Century Fox, Bayberry Pass 151, Lisbon, LTU 8283	bedoRW	905KW82Yf0er	Courtney_Bennett2189@hourpy.biz	4121.19	27	11	1987	468.93	BLOB
17	17	manager	Sofia	Tanner	Female	16	5	1849	Team Guard SRL, Tiptree Route 955, Arlington, PNG 3417	OGGEPl	g0ybYjGNNyGv	Sofia_Tanner7729@extex.org	3807.19	25	7	1835	412.54	BLOB
18	18	waiter	Moira	Farmer	Female	12	2	1874	Apple Inc., Marina Drive 8206, Lancaster, CAF 1744	jtLgCR	e40DpS70TdGf	Moira_Farmer1527@naiker.biz	4641.39	20	8	2022	487.84	BLOB
19	19	cook	Penny	Tyrrell	Female	21	9	1924	ExxonMobil, Enford Alley 6535, Lincoln, SVK 1415	ntpGLF	gibsXGorLly	Penny_Tyrrell9742@gompie.com	6705.83	12	3	1941	853.57	BLOB
20	20	customer	Megan	Long	Female	26	3	1890	Amazon.com, Aylward Boulevard 8309, Louisville, CPV 3871	Sgi5xV	aeUgfnxsa752	Megan_Long4617@bauros.biz	4172.98	22	2	1963	710.05	BLOB
21	21	admin	Raquel	Lambert	Female	30	10	1803	ExxonMobil, Cleveland Pass 8895, Escondido, EST 5344	heHwTx	XNFSBR3WblLR	Raquel_Lambert6776@twace.org	7197.91	18	9	1933	465.01	BLOB
22	22	manager	Ryan	Preston	Male	4	11	1905	Coca-Cola Company, Gatonby Way 9930, Milano, CYM 3461	TsUym4	YM0xGxr0Ah8u	Ryan_Preston8728@nickia.com	4007.18	22	8	1953	24.98	BLOB
23	23	waiter	Anabel	Hammond	Female	30	8	1920	Telekom, Boldero Vale 3589, Salem, ARG 6885	jjAxJ9	pJl6oG9IQDbt	Anabel_Hammond7347@sveldo.biz	8703.75	7	2	1957	915.57	BLOB
24	24	cook	Rosemary	Upsdell	Female	30	4	1975	CarMax, Argyle Pass 8711, Tulsa, CAN 0016	AP0w0V	0HOaWrvr37wE	Rosemary_Upsdell9924@bretoux.com	9980.75	14	4	2002	348.51	BLOB
25	25	customer	Cassidy	Clarke	Female	7	11	2002	Erickson, Fawn Crossroad 4080, Tallahassee, ZMB 0854	7G1uQx	f3f0j9j2Jbpe	Cassidy_Clarke3762@nickia.com	4049.86	3	5	1954	7.59	BLOB
26	26	admin	Barney	Addley	Male	29	8	1981	Leadertech Consulting, Glenwood Alley 3063, Minneapolis, BVT 8278	gl5JFB	aEPMjSxUQbZS	Barney_Addley527@famism.biz	3767.53	20	10	1984	161.0	BLOB

The record that is going to be deleted is highlighted

Updated Database:

	id	position	first	last	gender	birthDay	birthMonth	birthYear	address	username	password	email	salary	joinDay	joinMonth	joinYear	bonus	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	admin	Summer	Vernon	Female	4	10	1908	It Smart Group, Chelsea Manor Grove 9251, San Antonio, LSO 4143	7R7XWO	LolyYQ3r1sWQ	Summer_Vernon7360@dionrab.com	5262.19	13	5	2009	397.76	BLOB
2	2	manager	Robyn	Mitchell	Female	10	2	1921	Carrefour, Rosewood Road 784, El Paso, GEO 6005	NdX5w1	ic8dwMhVTvrp	Robyn_Mitchell8176@bulaffy.com	3102.96	19	7	1890	682.38	BLOB
3	3	waiter	Oliver	Tailor	Male	8	3	1859	Telekom, Buttonwood Way 7365, Lakewood, CRI 6616	FPLZP1	zgFdPb7dc0GF	Oliver_Tailor7151@vetan.org	4845.4	19	8	1933	631.97	BLOB
4	4	cook	Barney	Eastwood	Male	1	9	1872	Mars, Garfield Rue 129, Boston, COD 1207	lCfQql	izm2AAXxqhni	Barney_Eastwood3616@nickia.com	1771.08	30	4	1957	234.93	BLOB
5	5	customer	Rufus	Evans	Male	7	6	1942	Team Guard SRL, Anns Crossroad 3055, Los Angeles, ZWE 8266	m5VwhE	MjcqScWFdfhs	Rufus_Evans1822@atink.com	6302.34	27	1	2007	694.11	BLOB
6	6	admin	Teagan	Morris	Female	17	5	1843	UPC, Becklow Walk 7008, Reno, BOL 6722	L1gIRC	R208MmAplvz4	Teagan_Morris8718@acrit.org	5332.19	23	11	1825	577.08	BLOB
7	7	manager	Javier	Crawford	Male	28	7	2001	Team Guard SRL, Longman Grove 9816, San Bernardino, KAZ 1081	173vAY	iBr75rKnC4yz	Javier_Crawford4450@mafthy.com	4836.0	10	6	1869	489.29	BLOB
8	8	waiter	Alexa	Ingram	Female	25	9	1944	Comcast, Clavell Way 9521, Chicago, NGA 7888	SnWv1o	acfGjv2AocrK	Alexa_Ingram8111@dionrab.com	4367.0	21	11	1812	786.7	BLOB
9	9	cook	Miriam	Thomas	Female	19	7	1893	CarMax, Emily Rue 585, Columbus, GTM 5207	pI1jS8	YsF74niohc9W	Miriam_Thomas293@mafthy.com	9936.56	8	8	1892	682.99	BLOB
10	10	customer	Cedrick	Todd	Male	11	8	1909	CarMax, Lexington Way 4916, Long Beach, KEN 0413	OQbzG3	LKmBfQn0r0Z1	Cedrick_Todd6418@fulss.net	7080.01	12	2	1802	44.62	BLOB
11	11	admin	Tom	Watt	Male	21	11	1858	It Smart Group, Thrale Rue 9719, Reno, OMN 6743	K2IpN3	VzDRAGrAH8h8	Tom_Watt7481@cispeto.com	2145.18	20	9	1921	278.71	BLOB
12	12	manager	Martha	Truscott	Female	20	1	1801	ENEL, Carolina Pass 2425, Lincoln, MCO 5763	xC8N9O	XI5LMh3bmfIM	Martha_Truscott6250@typill.biz	2030.31	21	6	1953	558.93	BLOB
13	13	waiter	Nate	Fisher	Male	25	3	1977	Global Print, Champion Crossroad 8032, Huntsville, ARM 2547	bN5OdW	eiaS2pqH4BEX	Nate_Fisher2830@bulaffy.com	8282.39	11	1	1992	588.81	BLOB
14	14	cook	Gina	Townend	Female	17	11	1826	ExxonMobil, Western Drive 8976, Lancaster, MOZ 6785	3UnED1	2Y3kykn8sUf	Gina_Townend8565@acrit.org	5424.82	10	9	1807	992.39	BLOB
15	15	customer	George	Thorpe	Male	17	3	1972	Team Guard SRL, Bendall Grove 3142, Fullerton, VUT 2886	PBkJEC	0CDdtKVf9BY2	George_Thorpe4209@joiniaa.com	1450.96	20	7	1868	149.87	BLOB
16	16	admin	Courtney	Bennett	Female	21	6	1906	21st Century Fox, Bayberry Pass 151, Lisbon, LTU 8283	bedoRW	905KWB2Yf0er	Courtney_Bennett2189@hourpy.biz	4121.19	27	11	1987	468.93	BLOB
17	17	manager	Sofia	Tanner	Female	16	5	1849	Team Guard SRL, Tiptree Route 955, Arlington, PNG 3417	OGGEPl	g0ybYjGNNyGv	Sofia_Tanner7729@extex.org	3807.19	25	7	1835	412.54	BLOB
18	18	waiter	Moira	Farmer	Female	12	2	1874	Apple Inc., Marina Drive 8206, Lancaster, CAF 1744	jtLgCR	e40DpS70TdGf	Moira_Farmer1527@naiker.biz	4641.39	20	8	2022	487.84	BLOB
19	19	cook	Penny	Tyrrell	Female	21	9	1924	ExxonMobil, Enford Alley 6535, Lincoln, SVK 1415	ntpGLF	gibsXGorLly	Penny_Tyrrell9742@gompie.com	6705.83	12	3	1941	853.57	BLOB
20	20	customer	Megan	Long	Female	26	3	1890	Amazon.com, Aylward Boulevard 8309, Louisville, CPV 3871	Sgi5xV	aeUgfnxsa752	Megan_Long4617@bauros.biz	4172.98	22	2	1963	710.05	BLOB
21	21	admin	Raquel	Lambert	Female	30	10	1803	ExxonMobil, Cleveland Pass 8895, Escondido, EST 5344	heHwTx	XNFSBR3WblLR	Raquel_Lambert6776@twace.org	7197.91	18	9	1933	465.01	BLOB
22	22	manager	Ryan	Preston	Male	4	11	1905	Coca-Cola Company, Gatonby Way 9930, Milano, CYM 3461	TsUym4	YM0xGxr0Ah8u	Ryan_Preston8728@nickia.com	4007.18	22	8	1953	24.98	BLOB
23	23	waiter	Anabel	Hammond	Female	30	8	1920	Telekom, Boldero Vale 3589, Salem, ARG 6885	JjAxJ9	pIl6oG9IQDbt	Anabel_Hammond7347@sveldo.biz	8703.75	7	2	1957	915.57	BLOB
24	24	cook	Rosemary	Upsdell	Female	30	4	1975	CarMax, Argyle Pass 8711, Tulsa, CAN 0016	AP0w0V	0HOaWrvr37wE	Rosemary_Upsdell9924@bretoux.com	9980.75	14	4	2002	348.51	BLOB
25	25	customer	Cassidy	Clarke	Female	7	11	2002	Erickson, Fawn Crossroad 4080, Tallahassee, ZMB 0854	7G1uQx	f3f0j9j2Jbpe	Cassidy_Clarke3762@nickia.com	4049.86	3	5	1954	7.59	BLOB

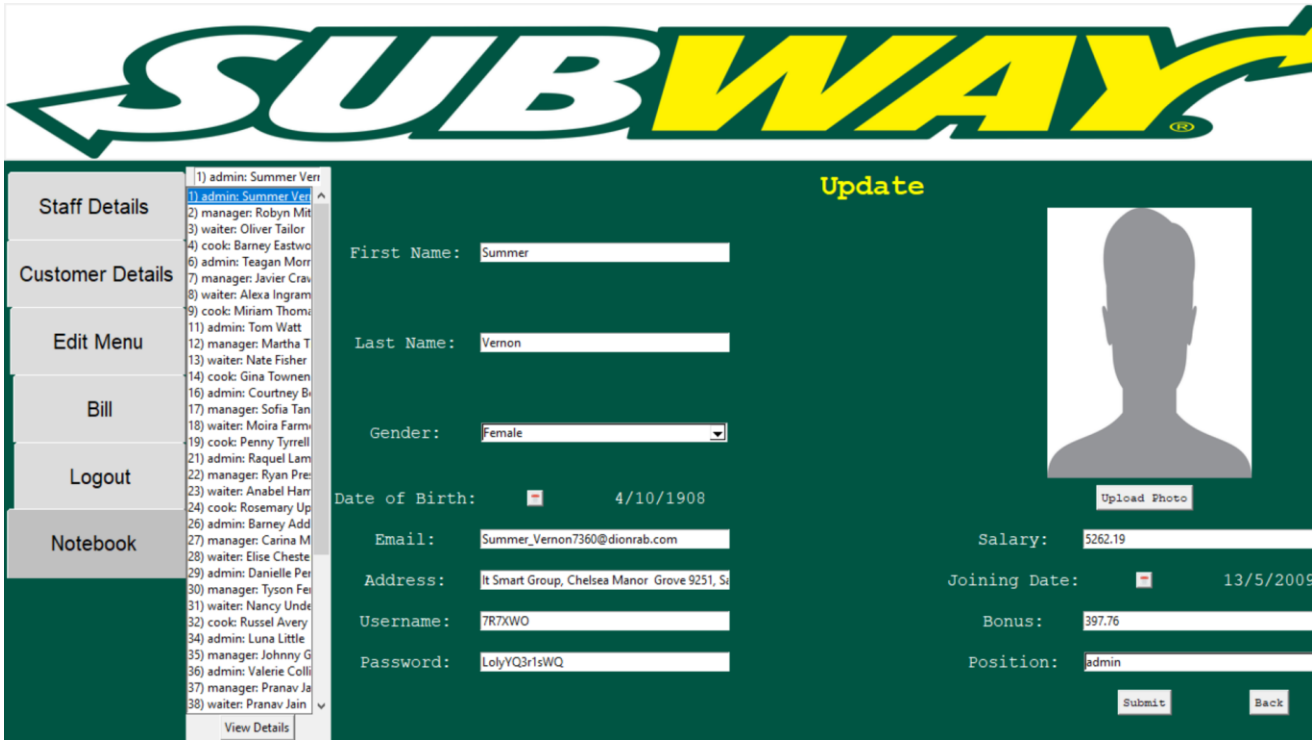
Searching for specific data in database:

```
def userList():
    ### Searches through database data to get data with specific characteristics ###
    itemList = []
    global userFocus, userInfo
    for i in range(len(userInfo)):
        user_no = "user"+str(i+1)
        if(userFocus == 'customer'):
            if(str(globals()[user_no].position) == 'customer'):
                itemList.append(str(globals()[user_no].id)+") "+str(globals()[user_no].position)+": "+str(globals()[user_no].first)+" "+str(globals()[user_no].last))
        else:
            if not(str(globals()[user_no].position) == 'customer'):
                itemList.append(str(globals()[user_no].id)+") "+str(globals()[user_no].position)+": "+str(globals()[user_no].first)+" "+str(globals()[user_no].last))
    updateListbox(itemList)
    return
```

General Overview:

The userList() function uses the data that was extracted from the database and searches through the whole user data for users who are customers or not customers according to the userFocus when this function is running. It goes through all the records in the userInfo and compares the position of the user with the user being focused on. If they position matches, the user is added to the user listbox.

Image:



Use of object-oriented design and classes:

```
class Users():
    def __init__(self, id, position, first, last, gender, birthDay, birthMonth, birthYear, address, username, password, email, salary, joinDay, joinMonth, joinYear, bonus):
        self.id = id
        self.position = position
        self.first = first
        self.last = last
        self.gender = gender
        self.DOB = str(birthDay) + "/" + str(birthMonth) + "/" + str(birthYear)
        self.address = address
        self.username = username
        self.password = password
        self.email = email
        self.salary = salary
        self.joinDate = str(joinDay) + "/" + str(joinMonth) + "/" + str(joinYear)
        self.bonus = bonus
# Connect to database
connection = sqlite3.connect('./restuarantDatabase.db')


# Create a cursor
userCursor = connection.cursor()



# Create a Table (Datatypes: NULL, INTEGER, REAL, TEXT, BLOB)
userCursor.execute("""CREATE TABLE IF NOT EXISTS user(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    position TEXT,
    first TEXT,
    last TEXT,
    gender TEXT,
    birthDay INTEGER,
    birthMonth INTEGER,
    birthYear INTEGER,
    address TEXT,
    username TEXT,
    password TEXT,
    email TEXT,
    salary REAL,
    joinDay INTEGER,
    joinMonth INTEGER,
    joinYear INTEGER,
    bonus REAL,
    image BLOB
    )
""")
```

```
# Query the database
userCursor.execute("""SELECT * FROM user""")


userInfo = userCursor.fetchall()
for users in range(len(userInfo)):
    userNumber = 'user'+str(users+1)
    vars()[userNumber] = Users(userInfo[users][0],userInfo[users][1],userInfo[users][2],userInfo[users][3],userInfo[users][4],userInfo[users][5],userInfo[users][6],userInfo[users][7]
```

General Overview:

The Users() class helped to simplify the programming by giving the data from the user table a name to call the data instead of using their indexes by objectifying each record in the table. Each record inherit the properties from the Users() class.

Working of Code:

The Users() class sets what each field in the database shoud be called instead of using the indexes. The connection connects the sqlite3 database to Python. The cursor allows complex logical operations by going through the table row by row. The SQL command 'CREATE TABLE IF NOT EXISTS' creates an empty table if the table is not found with the fields listed below with their datatypes. The userInfo converts row by row database into a 2D array. It goes through the whole user data and creates a variable instance for each user.

Use of 2D-arrays:

```
userInfo = userCursor.fetchall()
for users in range(len(userInfo)):
    userNumber = 'user'+str(users+1)
    vars()[userNumber] = Users(userInfo[users][0],userInfo[users][1],userInfo[users][2],userInfo[users][3],userInfo[users][4],userInfo[users][5],userInfo[users][6],userInfo[users][7]
```

General Overview:

The userInfo variable collates all the information from the database in the form of a 2D array. This is however making the code difficult to manage/edit as remembering the field name for each index would be very difficult. Thus, by converting it into an object, it is easier to refer that specific object.

Images:

```
def photoUpload():
    global userImage
    global userPhoto1
    userPhotoPath = askopenfilename(filetypes=(("png files", "*.png"), ("jpg files", "*.jpg"), ("jpeg files", "*.jpeg")))
    userPhoto1 = ImageTk.PhotoImage(Image.open(userPhotoPath).resize((200, 300), Image.ANTIALIAS))
    photoCanvas.itemconfigure(userImage, image=userPhoto1)
    photoCanvas.update()
    return
```

General Overview:

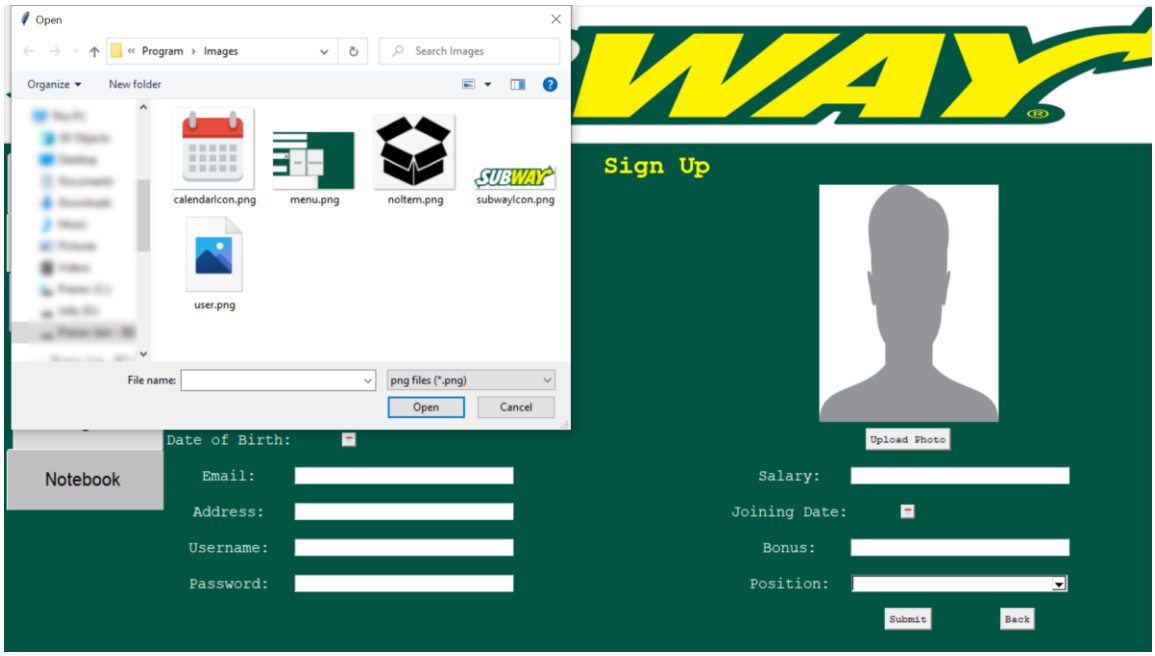
The photoUpload function asks the user to choose a photo from their file explorer. The images are usually of variable sizes and thus the photo gets resized and then shown to the user.

Working of code:

The userPhotoPath variable asks the user to select a photo that has an extension of ".png", ".jpg" or ".jpeg" and gets the file path to that photo. Then, the userPhoto1 variable opens the image and resizes that image in

200 x 300 pixel. Lastly, this image is attached to the photoCanvas and photoCanvas is updated to show the image to the user.

Image:



Use of multiple threads:

```
### Running a second program in parallel to the first ###
Thread(target = continuouslyCheckVariables).start()


### Starting Login Screen ###
loginFrame.place(x=475, y=350)
```

General Overview:

The threading allowed for parallel running of two programs. The thread ran the continuouslyCheckVariables() function while the program continued with the login page. This allowed for a continuous background check of userFocus and notebook tabs and how that would impact various frames while the program continued.

Use of nested loops:

```
def deleteMenuData():
    global itemInfo

    # Connect to database
    connection = sqlite3.connect('restuarantDatabase.db')

    # Create a cursor
    cursor = connection.cursor()

    cursor.execute("""SELECT * FROM menu""")
    itemInfo = cursor.fetchall()
    for items in range(len(itemInfo)):
        itemNumber = 'item'+str(items+1)
        if(globals()[itemNumber].item == itemNameEntry.get()):
            deleteQuery = "DELETE from menu where id = " + str(globals()[itemNumber].id)
            connection.execute(deleteQuery)
            emptyArrayText(entryNames)

            for index in range(items+1, len(itemInfo)):
                itemNumber = 'item'+str(index+1)
                connection.execute('''UPDATE menu SET id = ? WHERE id = ?''', ((int(globals()[itemNumber].id) - 1), (int(globals()[itemNumber].id))))
                globals()[itemNumber].id = int(globals()[itemNumber].id) - 1
        else:
            del globals()[itemNumber]

    # Commit our command
    connection.commit()

    # Close our cursor
    cursor.close()

    # Close our connection
    connection.close()

    connection = sqlite3.connect('restuarantDatabase.db')
    cursor = connection.cursor()
    cursor.execute("""SELECT * FROM menu""")
    itemInfo = cursor.fetchall()
    for items in range(len(itemInfo)):
        itemNumber = 'item'+str(items+1)
        globals()[itemNumber] = Items(itemInfo[items][0],itemInfo[items][1],itemInfo[items][2],itemInfo[items][3],itemInfo[items][4])
    menuList()
    connection.commit()
    cursor.close()
    connection.close()
```

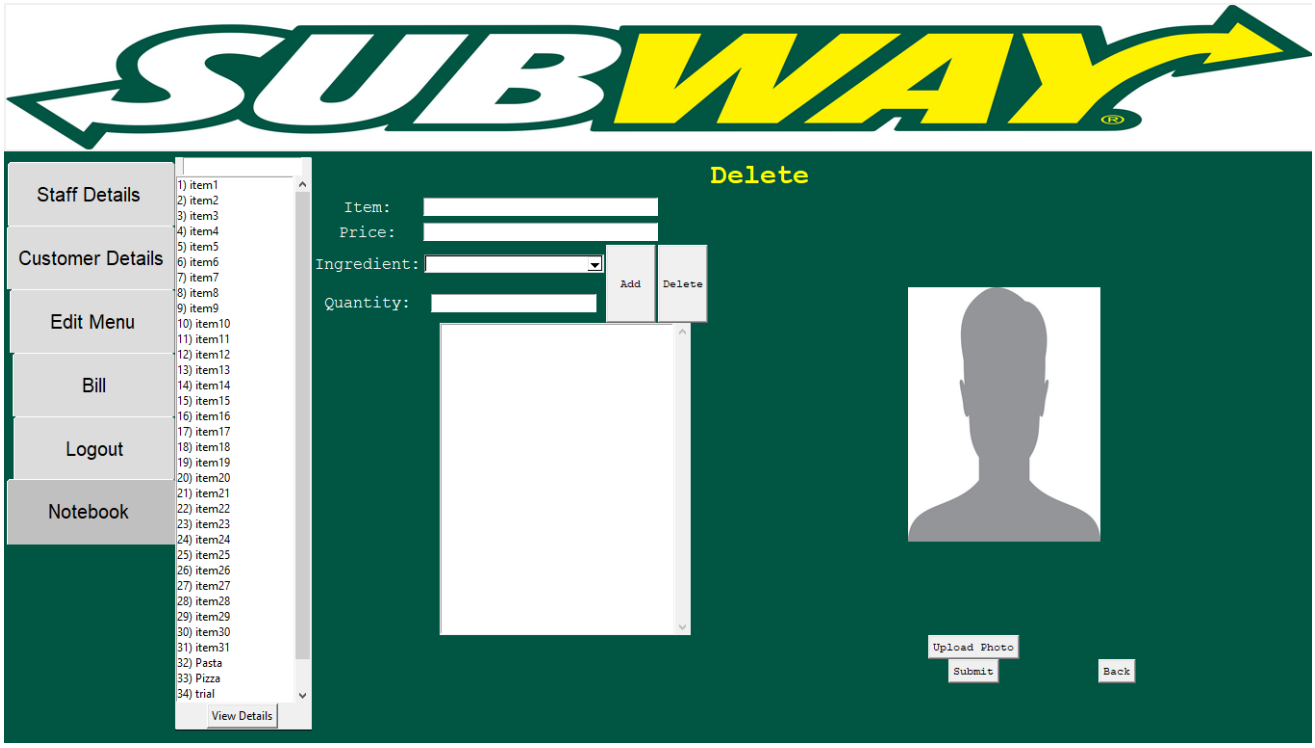
General Overview:

In the deleteMenuData(), the nested loop allowed to update the ids after the deleted item to be reduced by 1. The outer for loop found the position of the item to be deleted while the inner for loop went through the rest of the items and reduced their id by 1.

Working of Code:

The connection connects the sqlite3 database to Python. The cursor allows complex logical operations by going through the table row by row. The itemInfo converts row by row database into a 2D array. The item in each row of userInfo is checked with the item shown in item entry. The item is deleted at the specific id. The entry fields are then emptied. The items after the deleted id are then looped through and updated with a new id which is (old id -1). The connection to the database is lost when the connection is committed, and the cursor is closed. The connection is then again established to update the itemInfo with the updated information and the menu listbox is updated with the new information. The connection is then committed, and the cursor is closed.

Image:



Use of nested control functions:

```
def itemDetails():
    global detailForm
    if menuValidation():
        if(detailForm == 'add'):
            addMenuData()
        elif(detailForm == 'update'):
            updateMenuData()
        elif(detailForm == 'delete'):
            deleteMenuData()
    return
```

General Overview:

The itemDetails() function uses nested control functions like menuValidation() to check whether the information from the Menu Details frame is valid and thus can be added to/updated to /deleted from the database. It also includes other nested control functions like addMenuData(), updateMenuData(), deleteMenuData() which edit and control the data in the database.

SQL commands used:

```
deleteQuery = "DELETE from menu where id = " + str(globals()[itemNumber].id)

connection.execute('''UPDATE menu SET id = ? WHERE id = ?''', ((int(globals()[itemNumber].id) - 1), (int(globals()[itemNumber].id))))
connection.execute("""INSERT INTO menu (
                    id,item,price,ingredients,ingredientsQuantity,image) VALUES (?,?,?,?,?,?)""",
                   (len(itemInfo)+1,itemNameEntry.get(),itemPriceEntry.get(),ingredients,ingredientsQuantity,'N.A.')
                   )
```

General Overview:

SQL commands helped to interact with the database by allowing the data within the database to be edited. This could be by adding new records, updating previous records, or deleting the existing records.

Validation:

```
def userValidation():
    global userFocus
    if(len(firstNameEntry.get()) == 0):
        messagebox.showerror('Error', 'Empty First name')
        return 0
    else:
        for character in firstNameEntry.get():
            if not character.isalpha():
                messagebox.showerror('Error', 'Incorrect First name')
                return 0

    if(len(lastNameEntry.get()) == 0):
        messagebox.showerror('Error', 'Empty Last name')
        return 0
    else:
        for character in lastNameEntry.get():
            if not character.isalpha():
                messagebox.showerror('Error', 'Incorrect Last name')
                return 0

    genderCombobox['state'] = "normal"
    if(genderCombobox.current() == -1):
        genderCombobox['state'] = "readonly"
        messagebox.showerror('Error', 'Empty Gender')
        return 0
    genderCombobox['state'] = "readonly"

    if(len(DOBLabell.cget("text")) == 0):
        messagebox.showerror('Error', 'Empty Date of Birth')
        return 0

    for character in ['@','.']:
        if(len(emailEntry.get()) == 0):
            messagebox.showerror('Error', 'Empty email')
            return 0
        elif not character in emailEntry.get():
            messagebox.showerror('Error', 'Incorrect email')
            return 0

    if(userFocus != 'customer'):
        if(len(salaryEntry.get()) == 0):
            messagebox.showerror('Error', 'Empty Salary')
            return 0
        else:
            salaryInstance = salaryEntry.get().replace('.','',1)
            for character in salaryInstance:
                if not character.isdigit():
                    messagebox.showerror('Error', 'Incorrect Salary')
                    return 0

        if(len(joinDateLabell.cget("text")) == 0):
            messagebox.showerror('Error', 'Empty Join Date')
            return 0

        if(len(bonusEntry.get()) == 0):
            messagebox.askquestion('Question', 'Empty Bonus')
            return 0
        else:
            bonusInstance = bonusEntry.get().replace('.','',1)
            for character in bonusInstance:
                if not character.isdigit():
                    messagebox.showerror('Error', 'Incorrect Bonus')
                    return 0

        positionCombobox['state'] = "normal"
        if(positionCombobox.current() == -1):
            positionCombobox['state'] = "readonly"
            messagebox.showerror('Error', 'Empty Position')
            return 0
        positionCombobox['state'] = "readonly"

    return 1
```

```
    if(len(addressEntry.get()) == 0):
        messagebox.showerror('Error', 'Empty Address')
        return 0
    elif(len(addressEntry.get()) >= 100):
        messagebox.showerror('Error', 'Address is too long')
        return 0

    if(len(userEntry.get()) == 0):
        messagebox.showerror('Error', 'Empty Username')
        return 0
    elif(len(passEntry.get()) == 0):
        messagebox.showerror('Error', 'Empty Password')
        return 0
    else:
        connection = sqlite3.connect('restuarantDatabase.db')
        cursor = connection.cursor()
        cursor.execute("""SELECT * FROM user""")
        userInfo = cursor.fetchall()
        for users in range(len(userInfo)):
            userNumber = 'user'+str(users+1)
            if(globals()[userNumber].username == userEntry.get()):
                messagebox.showerror('Error', 'Incorrect Username')
                return 0
            elif(globals()[userNumber].password == passEntry.get()):
                messagebox.showerror('Error', 'Incorrect Password')
                return 0
        connection.commit()
        cursor.close()
        connection.close()
```

General Overview:

The userValidation() function checks whether all the human entered data is actually plausible and fit the criteria set by the program. This makes sure that any errors can be fixed here before getting added/updated/deleted from the database.

Image:

Bibliography

Anderson, J. (2019, March 25). *An Intro to Threading in Python*. Retrieved from realpython.com: https://realpython.com/intro-to-python-threading/

Codemy. (2019, March 27). *Open Files Dialog Box - Python Tkinter GUI Tutorial #15*. Retrieved from youtube.com: https://www.youtube.com/watch?v=Aim_7fC-inw

Codemy. (2020, May 22). *Create A Date Picker Calendar - Python Tkinter GUI Tutorial #72*. Retrieved from youtube.com: https://www.youtube.com/watch?v=fqfy-3IoVvs

freeCodeCamp. (2020, May 12). *SQLite Databases With Python - Full Course*. Retrieved from youtube.com: https://www.youtube.com/watch?v=byHcYRpMgI4

Tech With Tim. (2020, March 29). *Python Object Oriented Programming (OOP) - For Beginners*. Retrieved from youtube.com: https://www.youtube.com/watch?v=JeznW_7DlB0

Toad, T. (2022, February 4). *change the style of notebook tkinter*. Retrieved from codegrepper.com: https://www.codegrepper.com/code-examples/whatever/change+the+style+of+notebook+tkinter