EUROPEAN UNIVERSITY OF LEFKE
Faculty of Engineering
Department of Computer Engineering

COMP337
DATABASE MANAGEMENT SYSTEMS

Lab Work No. 7 Documentation

Prepared by Seward Richard Mupereri (20140175)

Submitted to Dr. Ferhun Yorgancıoğlu

Introduction

This program is a database management tool for students and instructors. It allows users to view, add, delete, and update student and instructor information in a SQLite3 database. The program is written in Python and has a GUI interface built with tkinter, and consists of three main screens:

- Student Management: This screen allows users to view a list of students, add a new student, delete a student, and update student information.

- Instructor Management: This screen allows users to view a list of instructors, add a new instructor, delete an instructor, and update instructor information.

- Advisor Management: This screen allows users to view a list of instructors and the students they advise, and add or remove students from an instructor's advisee list.

To use the program, users must first login through a window that displays information about the program and its developers. Additional features may also be available.

The following files are used in the program:

1. **interface.py**

```
import sqlite3
import queries
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import *
from tkinter import ttk
```

2. **queries.py**
3. **database.db**

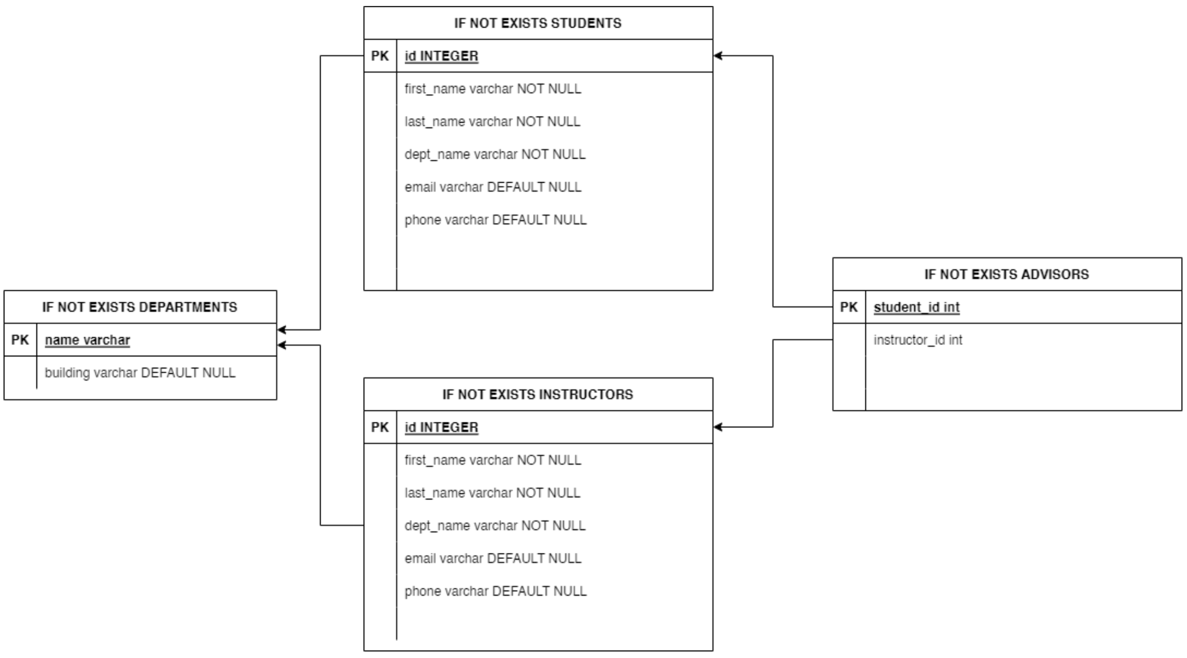
Database

The program uses a SQLite3 database to store and manage student and instructor information. The database has four tables: DEPARTMENTS, STUDENTS, INSTRUCTORS, and ADVISORS.
There are the queries to create the database tables and populate it with data for testing:

```
CREATE TABLE IF NOT EXISTS DEPARTMENTS (
  name varchar(50),
  building varchar(50) DEFAULT NULL,
  PRIMARY KEY(name)
);

CREATE TABLE IF NOT EXISTS STUDENTS (
  id INTEGER,
  first_name varchar(50) NOT NULL,
  last_name varchar(50) NOT NULL,
  dept_name varchar(50) NOT NULL,
  email varchar(50) DEFAULT NULL,
  phone varchar(50) DEFAULT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (dept_name) REFERENCES DEPARTMENTS (name) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS INSTRUCTORS (
  id INTEGER,
  first_name varchar(50) NOT NULL,
  last_name varchar(50) NOT NULL,
  dept_name varchar(50) NOT NULL,
  email varchar(50) DEFAULT NULL,
  phone varchar(50) DEFAULT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (dept_name) REFERENCES DEPARTMENTS (name) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS ADVISORS (
```

```sql
  instructor_id int(8),
  student_id int(8),
  PRIMARY KEY (student_id),
  FOREIGN KEY (instructor_id) REFERENCES INSTRUCTORS (id) ON DELETE CASCADE,
  FOREIGN KEY (student_id) REFERENCES STUDENTS (id) ON DELETE CASCADE
);

-- Insert data into DEPARTMENTS table
INSERT INTO DEPARTMENTS (name, building) VALUES
  ('Computer Science', 'Science Building'),
  ('Biology', 'Science Building'),
  ('Physics', 'Science Building'),
  ('Mathematics', 'Science Building'),
  ('English', 'Arts Building');

-- Insert data into STUDENTS table
INSERT INTO STUDENTS (id, first_name, last_name, dept_name, email, phone) VALUES
  ('20220001', 'John', 'Doe', 'Computer Science', 'john.doe@example.com', '555-555-1111'),
  ('20220002', 'Jane', 'Doe', 'Computer Science', 'jane.doe@example.com', '555-555-2222'),
  ('20220003', 'Bob', 'Smith', 'Biology', 'bob.smith@example.com', '555-555-3333'),
  ('20220004', 'Alice', 'Smith', 'Biology', 'alice.smith@example.com', '555-555-4444'),
  ('20220005', 'Tom', 'Johnson', 'Physics', 'tom.johnson@example.com', '555-555-5555'),
  ('20220006', 'Sue', 'Johnson', 'Physics', 'sue.johnson@example.com', '555-555-6666'),
  ('20220007', 'Mike', 'Williams', 'Mathematics', 'mike.williams@example.com', '555-555-7777'),
  ('20220008', 'Sarah', 'Williams', 'Mathematics', 'sarah.williams@example.com', '555-555-8888'),
  ('20220009', 'Chris', 'Jones', 'English', 'chris.jones@example.com', '555-555-9999'),
  ('20220100', 'Emma', 'Jones', 'English', 'emma.jones@example.com', '555-555-0000');

-- Insert data into INSTRUCTORS table
INSERT INTO INSTRUCTORS (id, first_name, last_name, dept_name, email, phone) VALUES
  ('9090001', 'Professor', 'X', 'Computer Science', 'professor.x@example.com', '555-555-1111'),
  ('9090002', 'Professor', 'Y', 'Biology', 'professor.y@example.com', '555-555-2222'),
  ('9090003', 'Professor', 'Z', 'Physics', 'professor.z@example.com', '555-555-3333'),
  ('9090004', 'Professor', 'A', 'Mathematics', 'professor.a@example.com', '555-555-4444'),
  ('9090005', 'Professor', 'B', 'English', 'professor.b@example.com', '555-555-5555');

-- Insert data into ADVISORS table
INSERT INTO ADVISORS (instructor_id, student_id) VALUES
  ('9090001', '20220001'),
  ('9090001', '20220002'),
  ('9090002', '20220003'),
  ('9090002', '20220004'),
  ('9090003', '20220005'),
  ('9090003', '20220006');
```

Visual Schema


IF NOT EXISTS STUDENTS
PK
id INTEGER
first_name varchar NOT NULL
last_name varchar NOT NULL
dept_name varchar NOT NULL
email varchar DEFAULT NULL
phone varchar DEFAULT NULL
IF NOT EXISTS DEPARTMENTS
PK
name varchar
building varchar DEFAULT NULL
IF NOT EXISTS ADVISORS
PK
student_id int
instructor_id int
IF NOT EXISTS INSTRUCTORS
PK
id INTEGER
first_name varchar NOT NULL
last_name varchar NOT NULL
dept_name varchar NOT NULL
email varchar DEFAULT NULL
phone varchar DEFAULT NULL

DEPARTMENTS Table

The DEPARTMENTS table contains information about the various departments in the school. It has the following fields:

- name (varchar(50)): The name of the department. This field is the primary key, which means it is a unique identifier for each department.
- building (varchar(50)): The building where the department is located. This field is optional and may be NULL.

STUDENTS Table

The STUDENTS table contains information about the students in the school. It has the following fields:

- id (INTEGER): The ID number of the student. This field is the primary key, which means it is a unique identifier for each student.
- first_name (varchar(50)): The first name of the student. This field is required and cannot be NULL.
- last_name (varchar(50)): The last name of the student. This field is required and cannot be NULL.
- dept_name (varchar(50)): The name of the department where the student is enrolled. This field is required and cannot be NULL. It has a foreign key constraint that references the name field in the DEPARTMENTS table. This means that the dept_name field in the STUDENTS table must contain a department name that exists in the DEPARTMENTS table.
- email (varchar(50)): The email address of the student. This field is optional and may be NULL.
- phone (varchar(50)): The phone number of the student. This field is optional and may be NULL.

INSTRUCTORS Table

The INSTRUCTORS table contains information about the instructors in the school. It has the same fields as the STUDENTS table, with the same foreign key constraint on the dept_name field.

ADVISORS Table

The ADVISORS table contains information about the students advised by each instructor. It has the following fields:

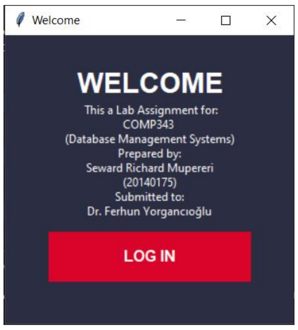
- instructor_id (int(8)): The ID number of the instructor. This field has a foreign key constraint that references the id field in the INSTRUCTORS table. This means that the instructor_id field in the ADVISORS table must contain an ID that exists in the INSTRUCTORS table.
- student_id (int(8)): The ID number of the student. This field has a foreign key constraint that references the id field in the STUDENTS table. This means that the student_id field in the ADVISORS table must contain an ID that exists in the STUDENTS table.

The database also has several sample data records inserted into the DEPARTMENTS, STUDENTS, and INSTRUCTORS tables. These records can be used for testing and demonstration purposes.

The program uses various SQL queries and operations to interact with the database, including SELECT, INSERT, UPDATE, and DELETE. These queries and operations are used to implement the various database management functions available in the program.

First Window: Program Information

The first window that appears when the program is launched is the welcome window. It displays a welcome message, some information about the program and its creator, and a "LOG IN" button.



This a Lab Assignment for:
COMP343
(Database Management Systems)
Prepared by:
Seward Richard Mupereri
(20140175)
Submitted to:
Dr. Ferhun Yorgancıoğlu

The welcome window has the following attributes:

- Size: 300x300 pixels
- Position: Centered on the screen
- Title: "Welcome"
- Background color: "#2B2D42" (dark blue)

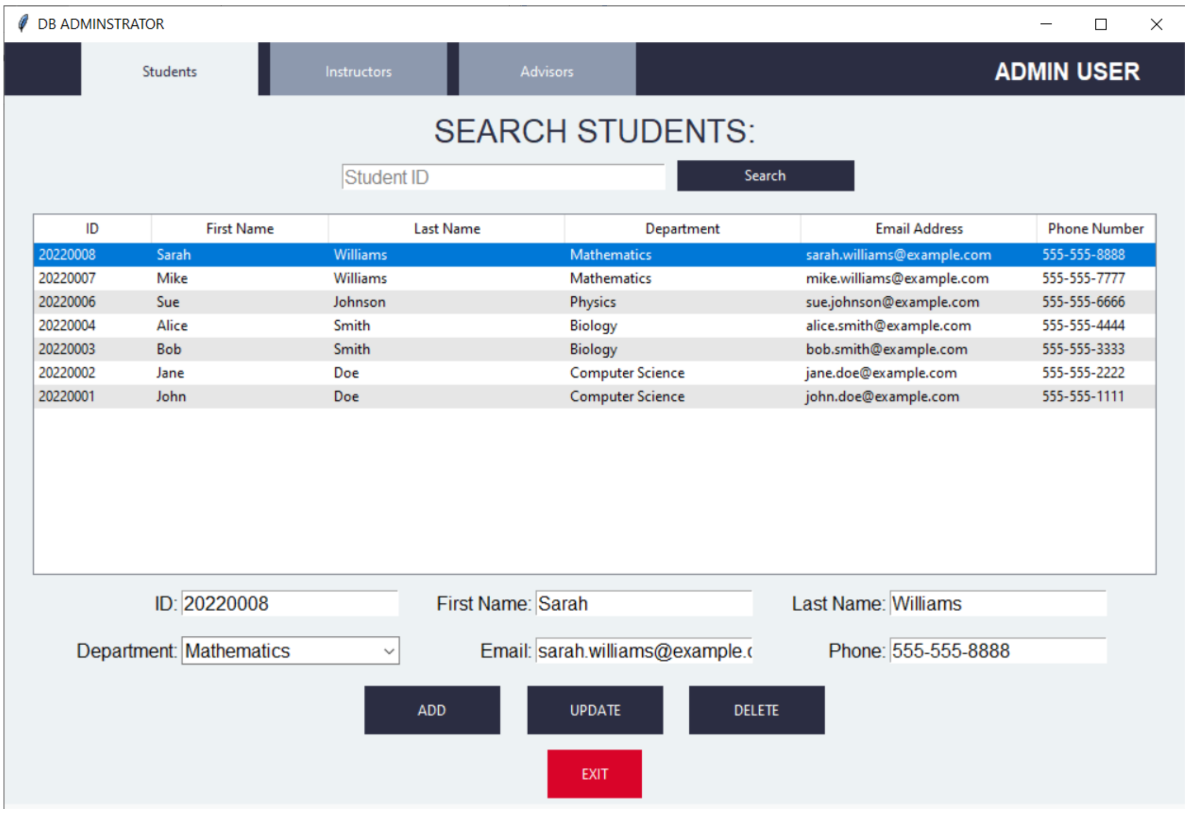
The welcome message is displayed in a label widget at the top of the window, using the "WELCOME" text and a large, bold font.

The information about the program and its creator is displayed in a label widget in the center of the window, using white text on a dark blue background.

The "LOG IN" button is displayed at the bottom of the window, using the text "LOG IN" and a red background color. When clicked, it will call the "admin_window" function and pass the root window as an argument.

The mainloop method is called at the end of the function to start the event loop and keep the window open until it is closed.

Screen 1: Student Management



ID	First Name	Last Name	Department	Email Address	Phone Number
20220008	Sarah	Williams	Mathematics	sarah.williams@example.com	555-555-8888
20220007	Mike	Williams	Mathematics	mike.williams@example.com	555-555-7777
20220006	Sue	Johnson	Physics	sue.johnson@example.com	555-555-6666
20220004	Alice	Smith	Biology	alice.smith@example.com	555-555-4444
20220003	Bob	Smith	Biology	bob.smith@example.com	555-555-3333
20220002	Jane	Doe	Computer Science	jane.doe@example.com	555-555-2222
20220001	John	Doe	Computer Science	john.doe@example.com	555-555-1111

ID: 20220008
First Name: Sarah
Last Name: Williams
Department: Mathematics
Email: sarah.williams@example.c
Phone: 555-555-8888

Description: The student management screen allows the administrator to view, search, add, update, and delete student records. The student records are displayed in a Treeview widget, which has columns for the student ID, first name, last name, department, email address, and phone number.

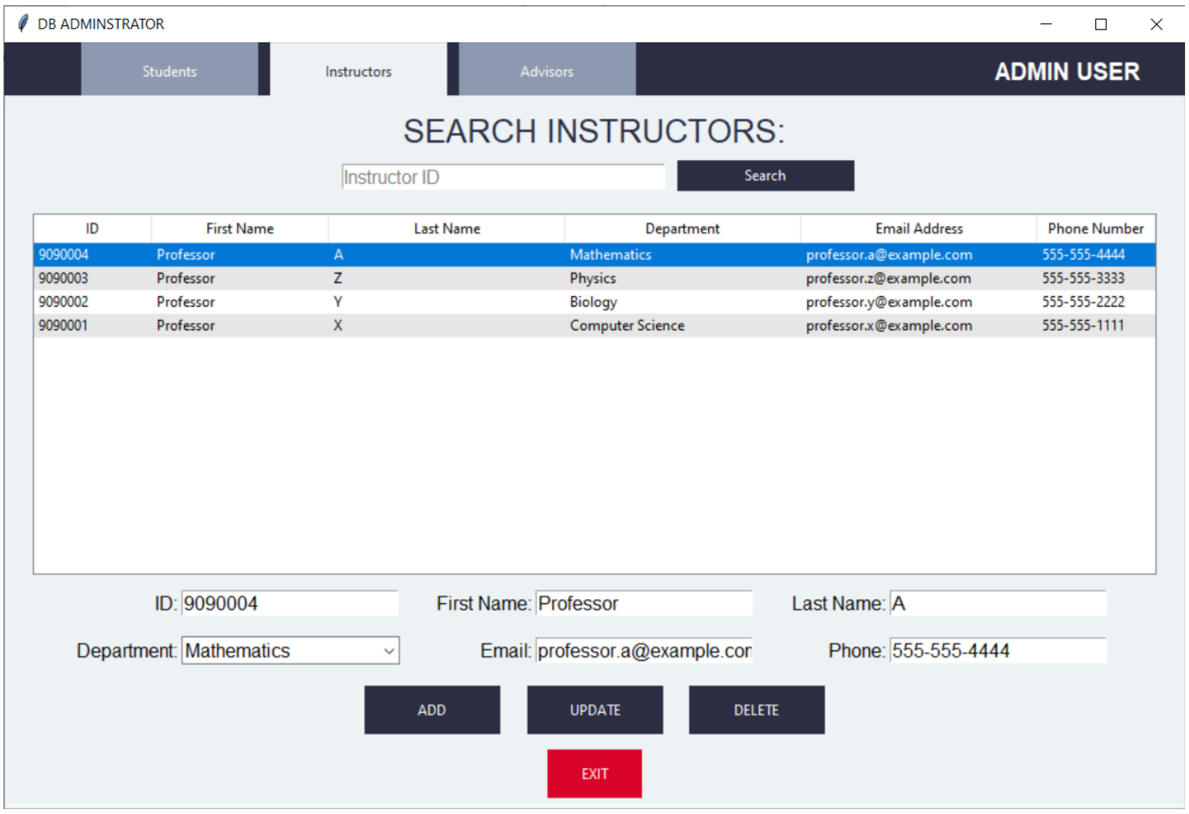
Functionality:

- The administrator can view all student records by clicking the "Students" button at the top of the screen. The records are displayed in the Treeview widget and can be scrolled through using the scrollbar.
- The administrator can search for a specific student record by entering the student ID in the search bar and clicking the "Search" button. If the student record is found, it will be highlighted in the Treeview widget.
- The administrator can add a new student record by filling out the form at the bottom of the screen and clicking the "Add" button. The form consists of fields for the student ID, first name, last name, department, email address, and phone number.
- The administrator can update an existing student record by selecting the record in the Treeview widget, modifying the desired fields in the form at the bottom of the screen, and clicking the "Update" button.
- The administrator can delete an existing student record by selecting the record in the Treeview widget and clicking the "Delete" button. A confirmation window will be displayed to confirm the deletion.

Implementation details:

- The Treeview widget is created in the **student_function** function, which is called when the administrator clicks the "Students" button. The widget is populated with data from the **STUDENTS** table in the database by executing a SELECT query and adding the returned data to the Treeview widget.
- The search functionality is implemented in the **search_student** function, which is called when the administrator clicks the "Search" button. This function executes a SELECT query with the entered student ID as a parameter, and highlights the matching student record in the Treeview widget if it is found.
- The add, update, and delete functionality is implemented in the **add_student**, **update_student**, and **delete_student** functions, respectively. These functions execute the appropriate INSERT, UPDATE, or DELETE query with the values from the form at the bottom of the screen as parameters.

Screen 2: Instructor Management



ID	First Name	Last Name	Department	Email Address	Phone Number
9090004	Professor	A	Mathematics	professor.a@example.com	555-555-4444
9090003	Professor	Z	Physics	professor.z@example.com	555-555-3333
9090002	Professor	Y	Biology	professor.y@example.com	555-555-2222
9090001	Professor	X	Computer Science	professor.x@example.com	555-555-1111

ID: 9090004
First Name: Professor
Last Name: A
Department: Mathematics
Email: professor.a@example.cor
Phone: 555-555-4444

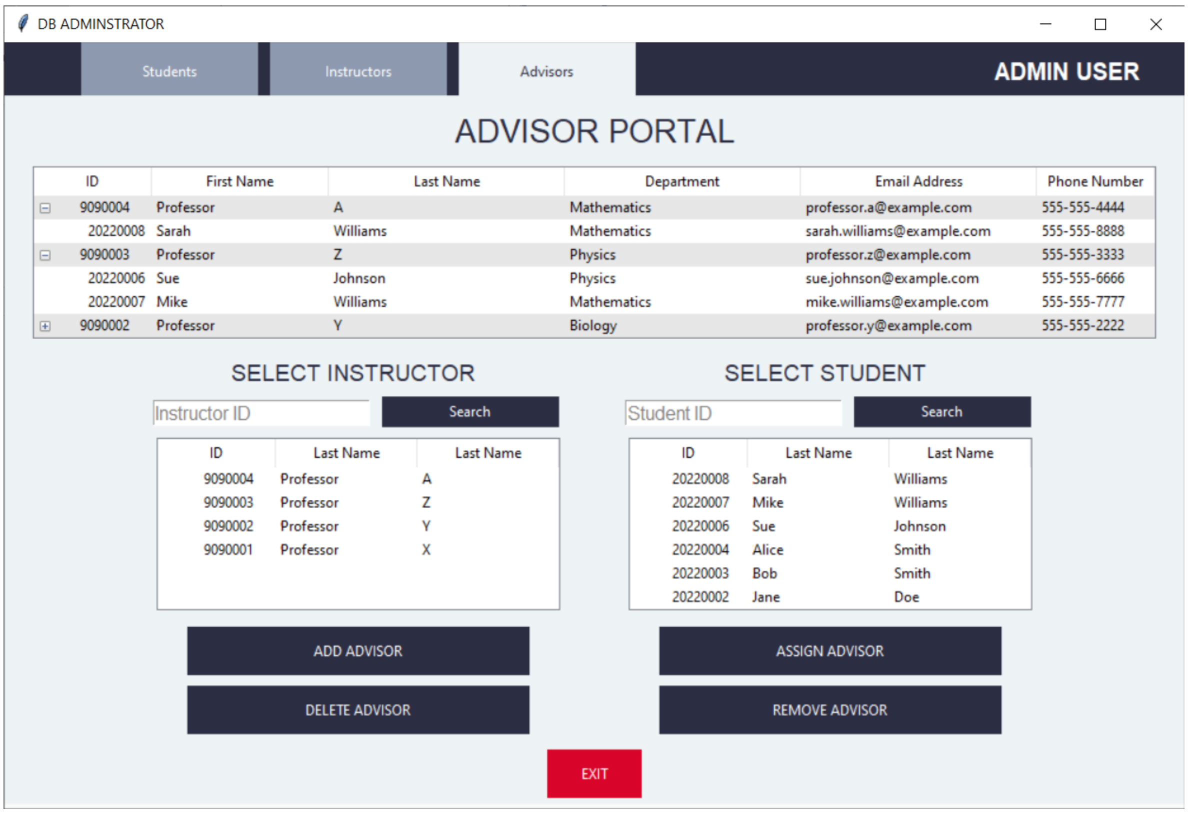
Functionality:

- Display a list of all instructors in a table
- Provide a search function to find a specific instructor by ID
- Allow the user to add, update, or delete an instructor
- Display a message indicating the success or failure of add, update, or delete actions

Implementation:

- To display a list of all instructors in a table, a **ttk.Treeview** widget is used. The widget is placed in a **Frame** widget in the main window and is populated with data from the INSTRUCTORS table in the database. Columns for the ID, first name, last name, department, email, and phone number of each instructor are included in the Treeview.
- A search function is implemented using an **Entry** widget and a **Button** widget. The user can enter an instructor ID in the Entry widget and press the Button to search for the instructor. A SELECT query is executed using the entered ID as a parameter, and the resulting data is used to highlight the matching row in the Treeview.
- To allow the user to add, update, or delete an instructor, a series of Entry widgets and Button widgets are provided. The user can enter the relevant information in the Entry widgets and press the corresponding Button to initiate the add, update, or delete action. An INSERT, UPDATE, or DELETE query is executed using the entered data as parameters, and a message is displayed indicating the success or failure of the action.

Screen 3: Advisor Management



	ID	First Name	Last Name	Department	Email Address	Phone Number
⊟	9090004	Professor	A	Mathematics	professor.a@example.com	555-555-4444
	20220008	Sarah	Williams	Mathematics	sarah.williams@example.com	555-555-8888
⊟	9090003	Professor	Z	Physics	professor.z@example.com	555-555-3333
	20220006	Sue	Johnson	Physics	sue.johnson@example.com	555-555-6666
	20220007	Mike	Williams	Mathematics	mike.williams@example.com	555-555-7777
⊞	9090002	Professor	Y	Biology	professor.y@example.com	555-555-2222

ID	Last Name	Last Name
9090004	Professor	A
9090003	Professor	Z
9090002	Professor	Y
9090001	Professor	X

ID	Last Name	Last Name
20220008	Sarah	Williams
20220007	Mike	Williams
20220006	Sue	Johnson
20220004	Alice	Smith
20220003	Bob	Smith
20220002	Jane	Doe

Functionality:

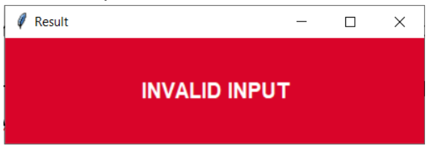
- The Advisor Management screen allows the user to view the Advisors and their respective students in a treeview widget.
- The user can expand each Advisor to view their students, which are nested beneath them in the treeview.
- The user can search for an Advisor by their ID using the search bar at the top of the screen. This will highlight the matching Advisor in the treeview.
- The user can also add, update, and delete Advisors using the buttons at the bottom of the screen.

Implementation Details:

- The treeview widget is created and populated with data from the INSTRUCTORS and STUDENTS tables in the database.
- The search function executes a SELECT query to retrieve the matching Advisor data based on the ID entered in the search bar.
- The add, update, and delete functions execute INSERT, UPDATE, and DELETE queries on the INSTRUCTORS table, respectively.
- The treeview is updated with the new data after each of these actions using the **insert**, **item**, and **delete** methods of the treeview widget.

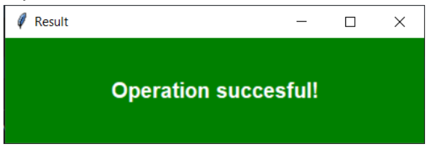
Extra Features:
When a user attempts to run an operation without the correct values, an Invalid Input windows pops if. If the correct values are entered an Operation Successful window opens.

Invalid Input Window



INVALID INPUT

Operation Successful Window



Operation succesful!

Conclusion

To conclude, this program provides a user-friendly interface for managing and interacting with the data stored in a database. The three screens of the program, namely Student Management, Instructor Management, and Advisor Management, allow an administrator to view, search, add, update, and delete records in the respective tables of the database.
The program makes use of various Tkinter widgets and functions to create the graphical user interface, including frames, labels, buttons, treeviews, and entry widgets. The program also utilizes SQLite3 to connect to the database and execute various SQL queries for data manipulation and retrieval.
Overall, this program serves as a useful tool for managing and organizing data in a database. It demonstrates the potential for using Tkinter and SQLite3 to create a functional and intuitive program for interacting with a database.