SJSU SAN JOSÉ STATE UNIVERSITY

Department of Electrical Engineering
EE104 - Fall 2024

LAB 7 REPORT

Scorpio

TEAM MEMBERS

Alicia Cunningham

Brandyn Gabel

Sergio Reyes Silva

Contents

PURPOSES AND GOALS

The purpose of this lab was to practice creating applications with OpenAI. For example, it made a web interface allowing users to upload a file. This interface will also answer questions using information from the uploaded file.

HOW TO INSTALL THE PROGRAMS

We need to install Spyder(Anaconda 3), which was previously installed and used in lab 1. We will also use Open AI

- Anaconda
  - Click on the link below and click on the Free download button. Follow the prompts after.
    - https://www.anaconda.com/
  - Once the Anaconda Navigator has been installed. Download these four leading apps.
    - Spyder
- OpenAI

HOW TO RUN THE PROGRAMS

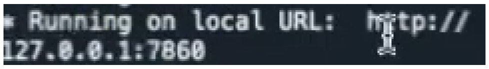
OPENAI to answer questions

1. Open anaconda

2. Open the given file GardioFileChatbot.py.

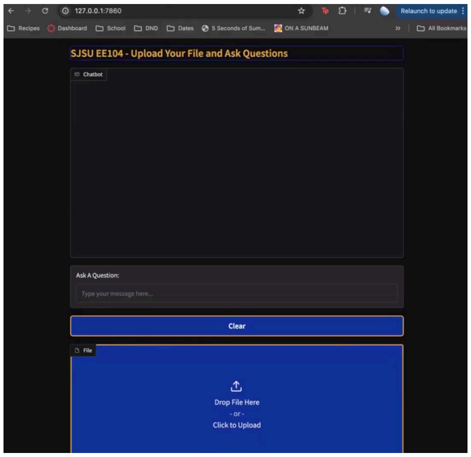
GradioFileChatbot.py

3. Run the Program and copy the local URL

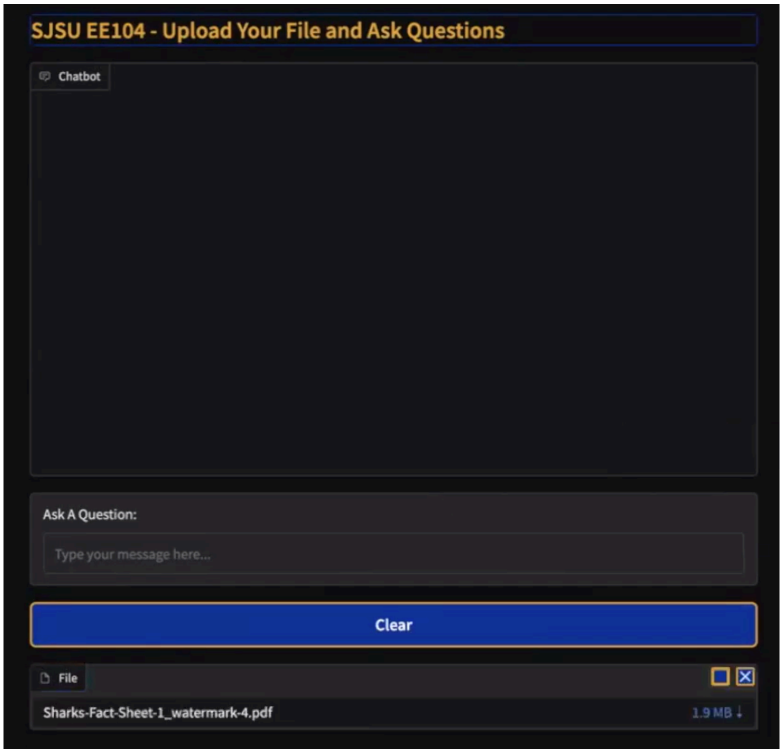


```
* Running on local URL:  http://
127.0.0.1:7860
```

4. Paste the URL to the browser.



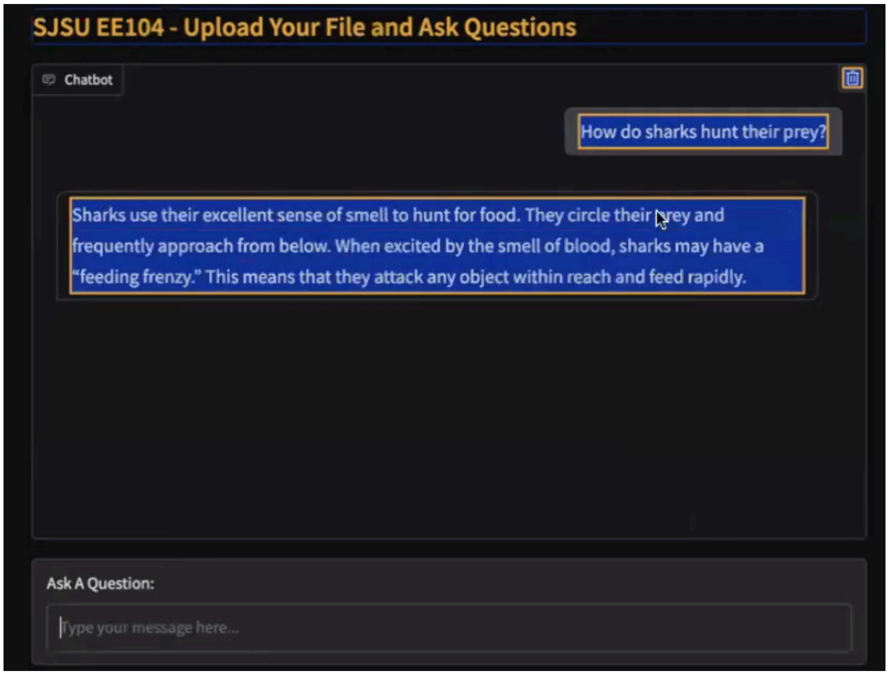
5. Upload file on the interface



SJSU EE104 - Upload Your File and Ask Questions

Sharks-Fact-Sheet-1_watermark-4.pdf

6. Type a question



SJSU EE104 - Upload Your File and Ask Questions

How do sharks hunt their prey?

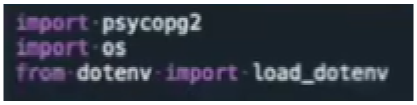
Sharks use their excellent sense of smell to hunt for food. They circle their prey and frequently approach from below. When excited by the smell of blood, sharks may have a "feeding frenzy." This means that they attack any object within reach and feed rapidly.

Build a Chatbot to answer questions.

1. Open anaconda

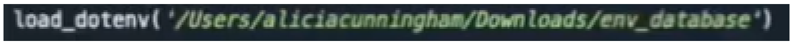
2. Create database file

3. Import libraries



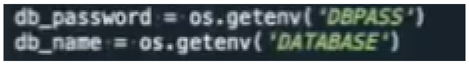
```
import psycopg2
import os
from dotenv import load_dotenv
```

4. Load environment variables from .env file



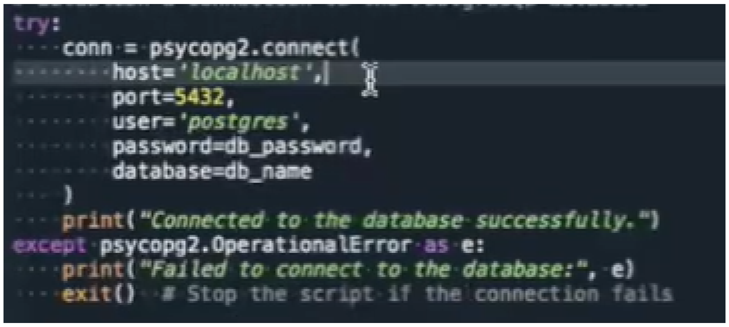
```
load_dotenv('/Users/aliciacunningham/Downloads/env_database')
```

5. Access environment variables



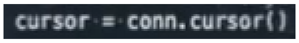
```
db_password = os.getenv('DBPASS')
db_name = os.getenv('DATABASE')
```

6. Establish a ceonnection to the PostgrenSQL database



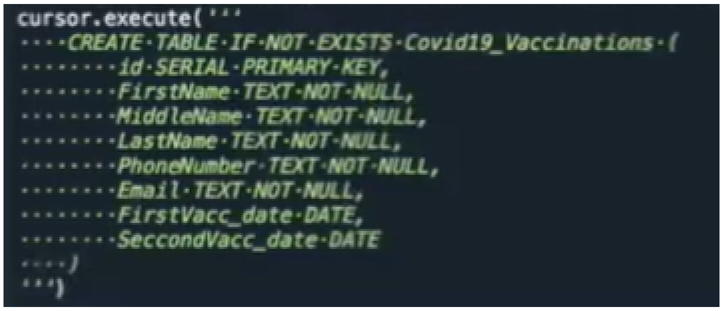
```
try:
    conn = psycopg2.connect(
        host='localhost',
        port=5432,
        user='postgres',
        password=db_password,
        database=db_name
    )
    print("Connected to the database successfully.")
except psycopg2.OperationalError as e:
    print("Failed to connect to the database:", e)
    exit()  # Stop the script if the connection fails
```

7. Create a cursor object to eseciute SQL commands\



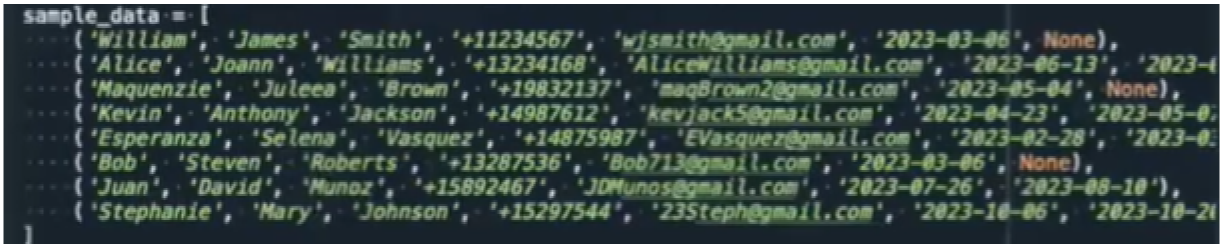
```
cursor = conn.cursor()
```

8. Create the task table



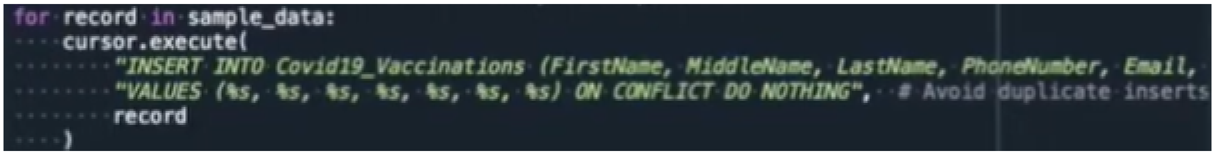
```
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Covid19_Vaccinations (
        id SERIAL PRIMARY KEY,
        FirstName TEXT NOT NULL,
        MiddleName TEXT NOT NULL,
        LastName TEXT NOT NULL,
        PhoneNumber TEXT NOT NULL,
        Email TEXT NOT NULL,
        FirstVacc_date DATE,
        SeccondVacc_date DATE
    )
''')
```

9. Insert sample data



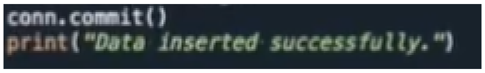
```
sample_data = [
    ('William', 'James', 'Smith', '+11234567', 'wjsmith@gmail.com', '2023-03-06', None),
    ('Alice', 'Joann', 'Williams', '+13234168', 'AliceWilliams@gmail.com', '2023-06-13', '2023-(
    ('Maquenzie', 'Juleea', 'Brown', '+19832137', 'maqBrown2@gmail.com', '2023-05-04', None),
    ('Kevin', 'Anthony', 'Jackson', '+14987612', 'kevjack5@gmail.com', '2023-04-23', '2023-05-0;
    ('Esperanza', 'Selena', 'Vasquez', '+14875987', 'EVasquez@gmail.com', '2023-02-28', '2023-0;
    ('Bob', 'Steven', 'Roberts', '+13287536', 'Bob713@gmail.com', '2023-03-06', None),
    ('Juan', 'David', 'Munoz', '+15892467', 'JDMunos@gmail.com', '2023-07-26', '2023-08-10'),
    ('Stephanie', 'Mary', 'Johnson', '+15297544', '23Steph@gmail.com', '2023-10-06', '2023-10-2(
]
```

10. Execute the insert command for entry



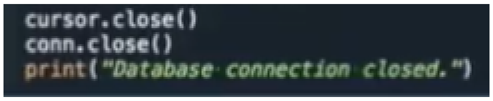
```
for record in sample_data:
    cursor.execute(
        "INSERT INTO Covid19_Vaccinations (FirstName, MiddleName, LastName, PhoneNumber, Email,
        "VALUES (%s, %s, %s, %s, %s, %s, %s) ON CONFLICT DO NOTHING",  # Avoid duplicate inserts
        record
    )
```

11. Commit the changes



```
conn.commit()
print("Data inserted successfully.")
```

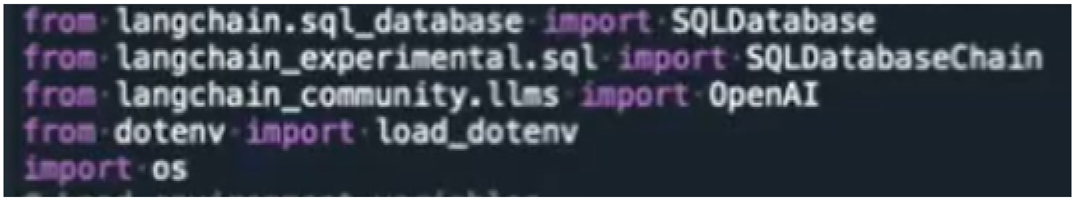
12. Close the connection



```
cursor.close()
conn.close()
print("Database connection closed.")
```

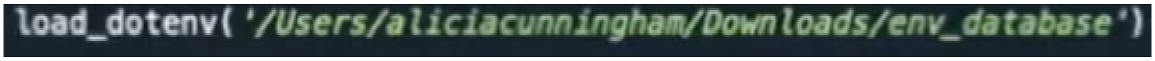
13. Create the APP file

14. Import libraries



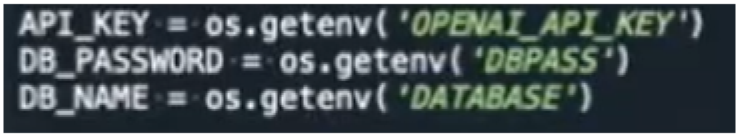
```
from langchain.sql_database import SQLDatabase
from langchain_experimental.sql import SQLDatabaseChain
from langchain_community.llms import OpenAI
from dotenv import load_dotenv
import os
```

15. Load environment variables



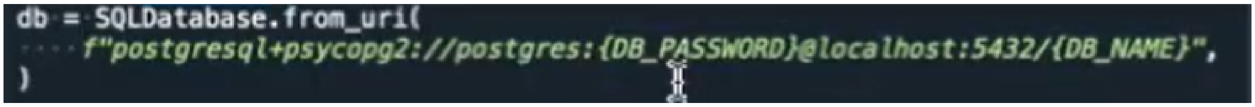
```
load_dotenv('/Users/aliciacunningham/Downloads/env_database')
```

16. Access environment variables



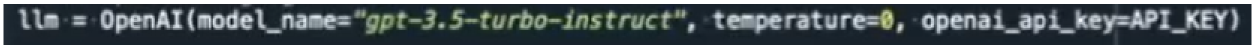
```
API_KEY = os.getenv('OPENAI_API_KEY')
DB_PASSWORD = os.getenv('DBPASS')
DB_NAME = os.getenv('DATABASE')
```

17. Setup database



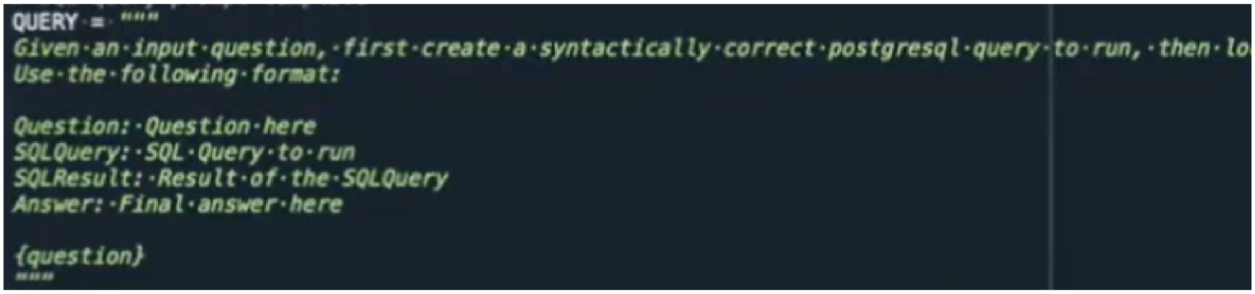
```
db = SQLDatabase.from_uri(
    f"postgresql+psycopg2://postgres:{DB_PASSWORD}@localhost:5432/{DB_NAME}",
)
```

18. Set up language model



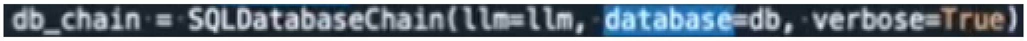
```
llm = OpenAI(model_name="gpt-3.5-turbo-instruct", temperature=0, openai_api_key=API_KEY)
```

19. SQL query prompt template



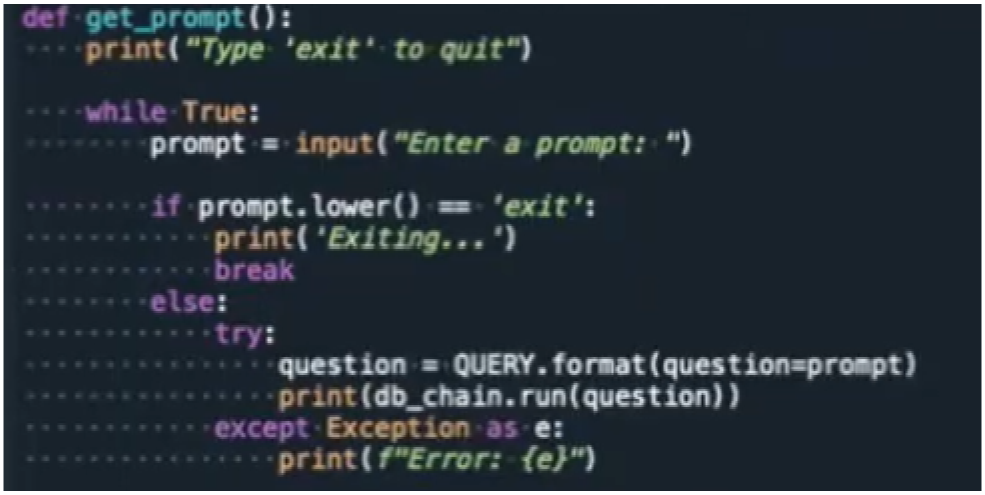
```
QUERY = """
Given an input question, first create a syntactically correct postgresql query to run, then lo
Use the following format:

Question: Question here
SQLQuery: SQL Query to run
SQLResult: Result of the SQLQuery
Answer: Final answer here

{question}
"""
```

20. Create the database chain



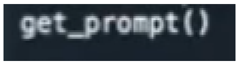
```
db_chain = SQLDatabaseChain(llm=llm, database=db, verbose=True)
```

21. Function to prompt user and execute queries



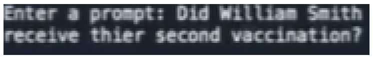
```
def get_prompt():
    print("Type 'exit' to quit")

    while True:
        prompt = input("Enter a prompt: ")

        if prompt.lower() == 'exit':
            print('Exiting...')
            break
        else:
            try:
                question = QUERY.format(question=prompt)
                print(db_chain.run(question))
            except Exception as e:
                print(f"Error: {e}")
```

22. Run the prompt



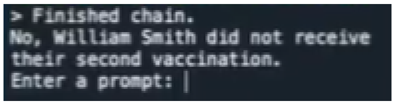
```
get_prompt()
```

23. Run program
24. Enter prompt



```
Enter a prompt: Did William Smith
receive thier second vaccination?
```

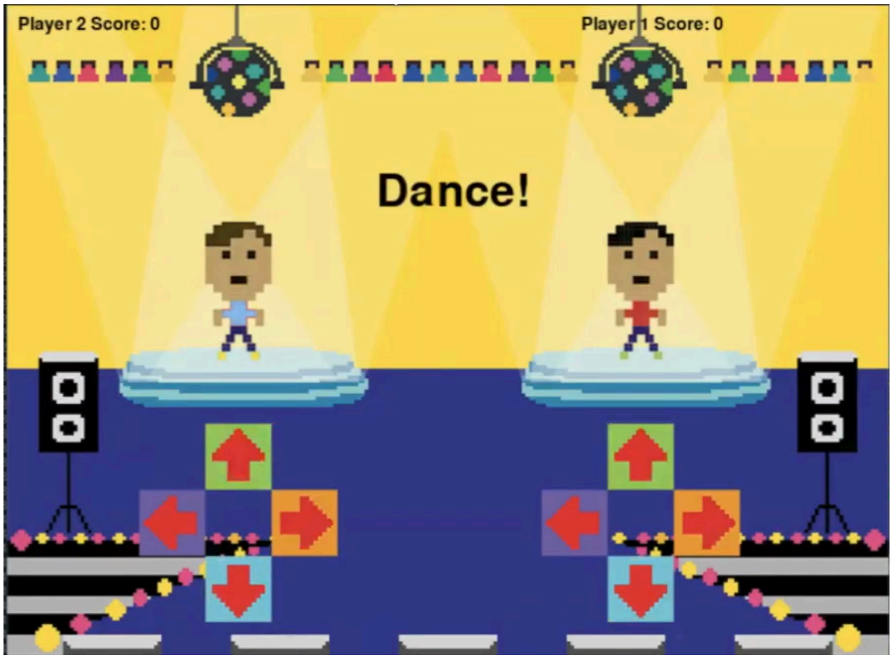
25. Output



```
> Finished chain.
No, William Smith did not receive
their second vaccination.
Enter a prompt: |
```

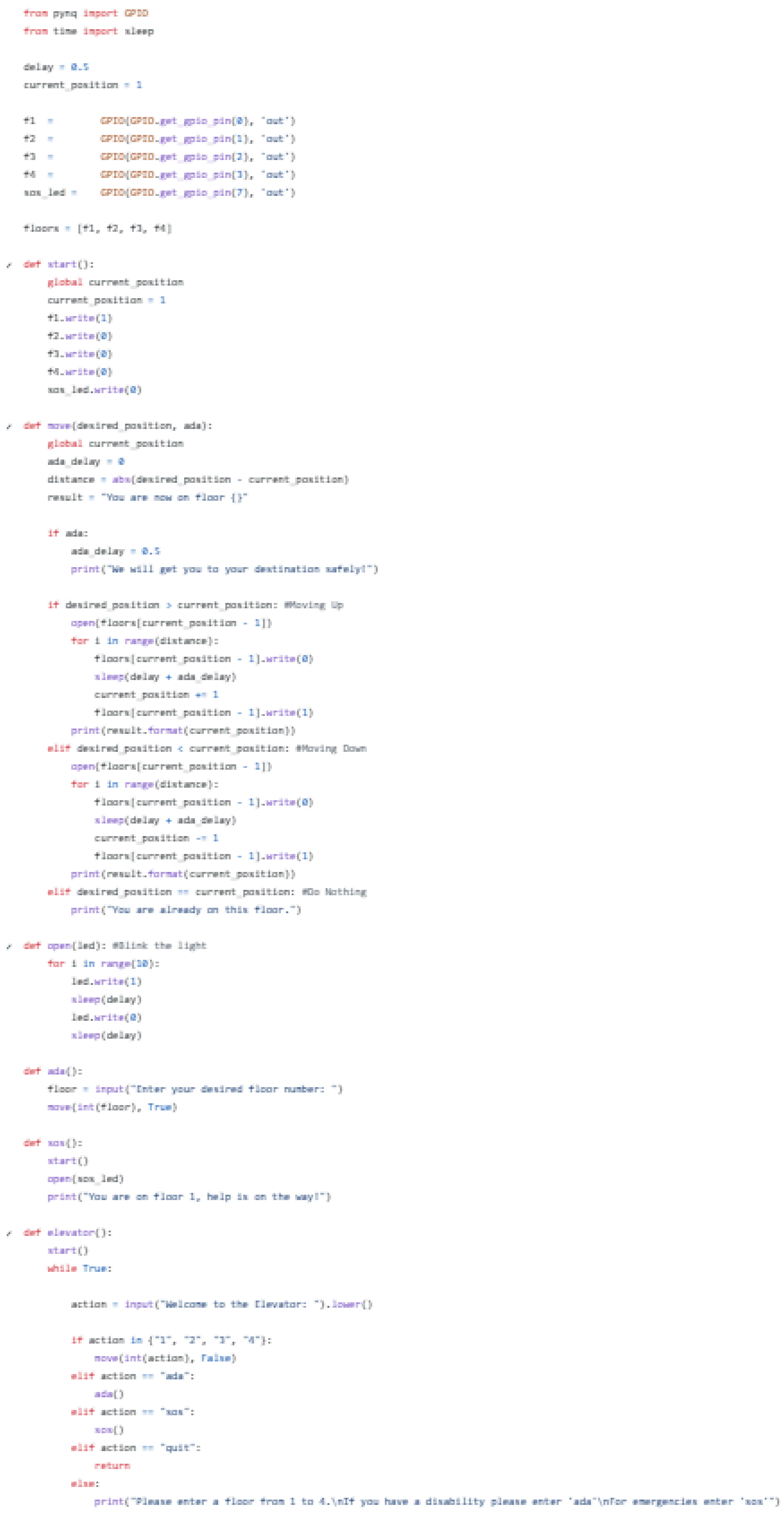
Game Development

1. Working dance challenge game



Github Link for Codes, Including Elevator Software:

https://github.com/Alicia-Cunningham/Lab7_EE104



```
from pynq import GPIO
from time import sleep

delay = 0.5
current_position = 1

f1  =       GPIO(GPIO.get_gpio_pin(0), 'out')
f2  =       GPIO(GPIO.get_gpio_pin(1), 'out')
f3  =       GPIO(GPIO.get_gpio_pin(2), 'out')
f4  =       GPIO(GPIO.get_gpio_pin(3), 'out')
sos_led =   GPIO(GPIO.get_gpio_pin(7), 'out')

floors = [f1, f2, f3, f4]

def start():
    global current_position
    current_position = 1
    f1.write(1)
    f2.write(0)
    f3.write(0)
    f4.write(0)
    sos_led.write(0)

def move(desired_position, ada):
    global current_position
    ada_delay = 0
    distance = abs(desired_position - current_position)
    result = "You are now on floor {}"

    if ada:
        ada_delay = 0.5
        print("We will get you to your destination safely!")

    if desired_position > current_position: #Moving Up
        open(floors[current_position - 1])
        for i in range(distance):
            floors[current_position - 1].write(0)
            sleep(delay + ada_delay)
            current_position += 1
            floors[current_position - 1].write(1)
        print(result.format(current_position))
    elif desired_position < current_position: #Moving Down
        open(floors[current_position - 1])
        for i in range(distance):
            floors[current_position - 1].write(0)
            sleep(delay + ada_delay)
            current_position -= 1
            floors[current_position - 1].write(1)
        print(result.format(current_position))
    elif desired_position == current_position: #Do Nothing
        print("You are already on this floor.")

def open(led): #Blink the light
    for i in range(10):
        led.write(1)
        sleep(delay)
        led.write(0)
        sleep(delay)

def ada():
    floor = input("Enter your desired floor number: ")
    move(int(floor), True)

def sos():
    start()
    open(sos_led)
    print("You are on floor 1, help is on the way!")

def elevator():
    start()
    while True:

        action = input("Welcome to the Elevator: ").lower()

        if action in {"1", "2", "3", "4"}:
            move(int(action), False)
        elif action == "ada":
            ada()
        elif action == "sos":
            sos()
        elif action == "quit":
            return
        else:
            print("Please enter a floor from 1 to 4.\nIf you have a disability please enter 'ada'\nFor emergencies enter 'sos'")
```

This code controls the KRIA board to operate like an elevator that services four floors. It also has an ADA mode where it moves slower for patients with disabilities, and an SOS mode to quickly return patients to the first floor.

The board controls five LEDs to represent the four floors and an SOS button.The person can interact with the board by typing into the command line what they wish to do. If all dependencies are on your computer, you can completely simulate the elevator behavior without needing LEDs

PROCESS & WORKFLOW

For this lab, we worked on it individually, and each team member completed the program. We met through Zoom to record our video of the program working. Lastly, we all worked on our lab report.

VIDEO RECORDINGS

Recording Title	URL	Notes
LAB7Software_Scorpio.mp4	https://drive.google.com/file/d/1xxH2BU8izrB6lcILRKASHpwfikhGY9Rg/view?usp=sharing	This video contains an LAB 7 video recording.

CONCLUSIONS

This further demonstrated the capabilities and functionality of Python and OpenAI. In this case, we could use Python code to create a web interface to upload a file and ask questions regarding the information in the file. We could also add more functionality to a game. We also gained more knowledge and experience on how to use Python. We also had the opportunity to use and gain more experience with the OpenAI platform.

REFERENCES

- https://platform.openai.com/docs/quickstart
- https://platform.openai.com/docs/api-reference/introduction