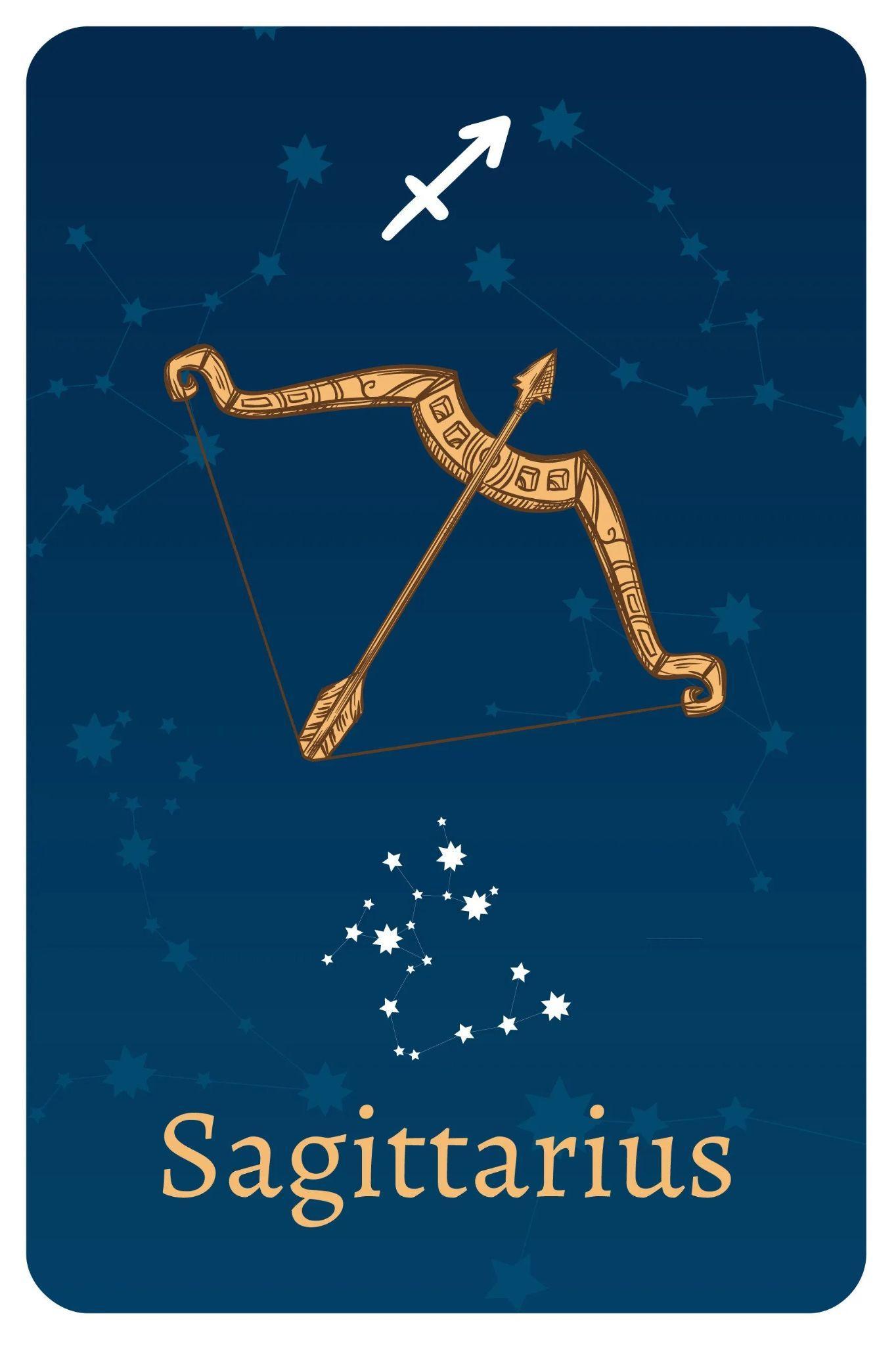
****

TEAM MEMBERS

Shivansh Shukla

Nathan Lee

Edwin Kulakkattolikel

Contents

[PURPOSES AND GOALS 3](#_heading=h.gjdgxs)

[HOW TO INSTALL THE PROGRAMS 4](#_heading=h.30j0zll)

[DEPENDENCIES 4](#_heading=h.1fob9te)

[HOW TO RUN THE PROGRAMS 4](#_heading=h.3znysh7)

[DESIGN ARCHITECTURE 5](#_heading=h.2et92p0)

[PROCESS & WORKFLOW](#_heading=h.tyjcwt) 6

[TEST DATA](#_heading=h.3dy6vkm) 7

[VIDEO RECORDINGS](#_heading=h.1t3h5sf) 9

[CONCLUSIONS](#_heading=h.4d34og8) 9

[REFERENCES](#_heading=h.2s8eyo1) 9

# PURPOSES AND GOALS

This lab aims to advance students' software development proficiency by focusing on OpenAI applications. The objectives include implementing a web scraper to download files, using OpenAI for responsive chatbots, and establishing a client-server environment for intranet file queries. Grading follows Python Programming criteria, and references support the process. Deliverables, including a narrated lab report, are submitted on GitHub. Use cases range from web scraping and client-server solutions to OpenAI-driven Q&A and automated testing with Selenium. Data science tasks involve risk factor identification in loan datasets and time series-based trend prediction using COVID-19 data. The assignment concludes with a creative Dance Challenge game, fostering broader Python application interest. The overarching goal is to enhance technical skills and ignite curiosity in the diverse applications of Python programming.

1. **Client-Server Solution**:
   1. Provide detailed instructions for using the client-server solution.
2. **Web Scraper Solution**:
   1. Offer clear instructions for using the web scraper solution.
3. **OpenAI File Q&A**:
   1. Implement an OpenAI chatbot using information from stored files.
   2. Follow instructions from chatbot-custom-data\_Instructions.txt.
4. **OpenAI Web Crawl Q&A**:
   1. Develop an OpenAI chatbot using information from web-scraped files.
   2. Follow instructions from web-qa\_Instructions.txt.
5. **Test Automation using Selenium**:
   1. Utilize Selenium to generate Python scripts for website testing.
   2. Select a website for automated testing from the provided list.
6. **Data Science – Risk Factor Identification**:
   1. Add a new column to a loan dataset, indicating recommendations based on risk levels.
   2. Choose a loan dataset from the provided sources.
7. **Data Science - Trend Prediction using Time Series Method**:
   1. Use Time Series method for predicting outcomes based on historical information.
   2. Apply the method to Santa Clara County COVID-19 data or CDC COVID-19 data.
8. **Game Development – Dance Challenge:**
   1. Design a Dance Challenge game for two players.
   2. Allow for changing the music in the game.

# HOW TO INSTALL THE PROGRAMS

## DEPENDENCIES

**OpenAI File Q&A:** pip install the following:

llama\_index, gradio, plotly, scipy, scikit-learn, flask, langchain, openai, pandas, numpy, glob2, datetime

Python version 3.10.11

**Open AI Web Crawl Q&A:** pip install the following:

llama\_index, gradio, plotly, scipy, scikit-learn, flask, langchain, openai, pandas, numpy, glob2, datetime

tiktoken, python-dotenv

Python version 3.10.11

**Data Science – Risk Factor Identification**: pip install pandas

**Data Science - Trend Prediction using Time Series Method**: pip install numpy, pandas, matplotlib

**Selenium Test:** Install Selenium IDE Chrome extension, download ChromeDriver and add it to PATH. Install Python, upgrade pip, and install selenium, pytest, and python-dotenv. Use npm to install Selenium Side Runner globally. Modify the Python script for the ChromeDriver path. In terminal, install Python packages. Run Selenium Side Runner with selenium-side-runner <ProjectName>.side in terminal, adjusting paths as needed.

**Dance Challenge:** Install Python and Pygame Zero with pip install pgzero. Copy the code to a .py file, like dance\_game.py. In the terminal, navigate to the script's directory and run pgzrun dance\_game.py. Ensure necessary images and sounds are available.

# HOW TO RUN THE PROGRAMS

**OpenAI Chatbot:** In order to run the program, the user needs to run embeddings.py file on a preferred IDE, and then got to localhost:8001 to access the chatbot website. Once entered, the user can ask the chatbot a question about the site.

**OpenAI File Q&A:** In order to run the program, the user needs to first pip install the libraries, and have python 3.10.11. Once installed, open up PowerShell, then enter the following commands:

cd C:\webcrawl-q-and-a

env\Scripts\activate

python web-qa.py

Once entered, the user can now start asking questions about football after csv files and the txt files are generated.

**Data Science – Risk Factor Identification**: download loan data csv, run code

**Data Science - Trend Prediction using Time Series Method**: download covid data csv, adjust dates, run code

**Selenium Test:** To run the Selenium test scripts, first, ensure you have Python installed on your system. Install the required packages using the command ‘pip install selenium pytest python-dotenv.’ Download the Chrome WebDriver from ChromeDriver Downloads and save it to a location on your computer. Create a .env file to store sensitive information like passwords. Update the path to the Chrome WebDriver in your Python script. Open a PowerShell or command prompt window in the script directory and execute the script using pytest <your\_script\_name>.py. Review the test results in the console. Ensure the website is accessible during script execution.

**Dance Challenge:** To play the game, ensure you have Python and Pygame Zero installed. Save the provided code as a .py file (e.g., dance\_game.py). Open a terminal in the file's directory and run the game using the command pgzrun dance\_game.py. Follow on-screen instructions; typically, use arrow keys to mimic dance moves. The game continues until a mistake is made, displaying scores and a "Game Over" message. Close the game window to exit. Ensure required assets are in the same directory or adjust file paths in the code.

# DESIGN ARCHITECTURE

**OpenAI Chatbot and Webcrawl:** Hardware: CPU and computer as a whole. Software: Uses IDE to run python programs with the provided OpenAI API Key generated by OpenAI, and uses the internet to communicate with OpenAI API to run the artificial intelligence. It also accesses all the libraries that were installed. Localhost is used to communicate with the html and css code so that the user can communicate with the AI Chatbot.

**Data Science – Risk Factor Identification**: Hardware: CPU Software: Pandas library, code, csv file

**Data Science - Trend Prediction using Time Series Method**: Hardware: CPU Software: Pandas library, code, csv file

**Selenium Test:** On the hardware side, a personal computer serves as the primary environment for the Selenium IDE and associated tools. Software components include browser extensions, specifically Selenium IDE as a Chrome extension, and the Selenium WebDriver (utilizing ChromeDriver). Python is the chosen programming language, with additional libraries, namely selenium, pytest, and python-dotenv, enhancing test script functionality. Configuration details, such as passwords and license codes, are stored in a separate .env file. The test development process involves recording tests with Selenium IDE, exporting them to Python scripts, and executing them using the selenium-side-runner command.

**Dance Challenge:** The dance game's architecture revolves around the Pygame Zero framework, providing a modular structure for game logic, user input handling, and graphics. Players interact with the game using keys, triggering dance moves displayed through actor animations. The code manages game states, score updates, and the initiation of new dance sequences. The hardware interaction includes user input devices for key presses, graphical output to display devices, and audio output for background music.

# PROCESS & WORKFLOW

**OpenAI Chatbot:** First the user needs to run embeddings.py. This file gets the contents from chatbot\_docs folder and converts it to csv file for openai. Then open chatbot.py, and then webapp.py. Once ran, the API is loaded up and the user then has to go to localhost:8001 to access the chatbot AI.

**OpenAI Web Crawl Q&A:** First execute web-qa.py file. Then the program will then create csv files of the data of the website. Then, the user can ask questions about information from the website through the terminal.

**Data Science – Risk Factor Identification**: Loan approver updates information into excelsheet. Loan approver runs code. Loan approver updates status of loan based on results

**Data Science - Trend Prediction using Time Series Method**: Data analyst loads covid data set csv. Runs code. Analyses the predicted data.

**Selenium Test:** The automated test development process with Selenium IDE involves downloading software, installing Python, creating supporting files, recording a test, exporting it to Python, and running it using Selenium Side Runner. Users set up Selenium IDE, record interactions with a website, export the test to Python, and execute it with Selenium Side Runner. Results are reviewed for pass/fail conditions, allowing for quick adjustments to the script or test plan. This streamlined process ensures efficient and repeatable automated testing, enhancing overall testing capabilities and reliability for web applications.

**Dance Challenge:** The game initializes with library setup and variable initialization, rendering dancers and arrows based on player input. It involves displaying and generating dance sequences, scoring, and transitioning to new moves. Players follow on-screen instructions, score based on correctness, and may reach a game over state. The end displays final scores, stops music, and offers a restart or exit option. This streamlined process ensures an engaging dance game experience.

# TEST DATA

| Test Name | Output |
| --- | --- |
| OpenAI Test |  |
| OpenAI Chatbot |  |
| Webcrawl CSV generating files |  |
| OpenAI API using web crawl to answer questions |  |
| Dance Challenge |  |
| Selenium Test |  |

# VIDEO RECORDINGS

| **Recording Title** | **URL** | **Notes** |
| --- | --- | --- |
| Lab 8: OpenAI Chat Bot | <https://youtu.be/eHcCQiOrm7U> | Demonstration of Chat Bot |
| Lab 8: OpenAI WebCrawling | <https://youtu.be/B84kUnqidGI> | Demonstration of Web Crawl |
| Lab8-DataScience | <https://youtu.be/pReesxBUi0w> | Data Science: Loan Risk Assessment and Covid Prediction Demo Video |
| Lab 8: Selenium Test | <https://youtu.be/ggXI_SGW4Q4> | Login Test using selenium demo |
| Lab 8: Dance Challenge | [EE104 - Lab 8: Dance Challenge](https://youtu.be/wS4xw64qBeU) | Dance challenge game demo |

# CONCLUSIONS

In conclusion, this lab was designed with the overarching goal of advancing students' software development proficiency through a diverse range of Python applications, ranging from web scraping and client-server solutions to OpenAI-driven Q&A and automated testing. The team successfully implemented solutions for each specified task, including the development of a client-server environment, web scraper, OpenAI chatbots, data science tasks, Selenium test automation, and a creative Dance Challenge game. Each component was meticulously crafted to meet Python programming criteria, utilizing relevant libraries and frameworks. The lab not only enhanced technical skills but also sparked curiosity in the myriad applications of Python programming.

The key lesson learned from this lab is the importance of clear and comprehensive documentation. While the team achieved success in implementing the specified solutions, the process could be further streamlined and improved with more detailed documentation. Clearer instructions, comments within the code, and additional references would contribute to a more user-friendly experience for future developers and users of these programs.

# REFERENCES

* What Is Agile Reporting? (Definition and How To Complete) <https://www.indeed.com/career-advice/career-development/agile-reporting>
* Test Report <https://strongqa.com/qa-portal/testing-docs-templates/test-report>