**Air Foil Coordinates**   
Project

**Student:** Zeyad Hassan Abdel Azeem  
**Serial\_no:** 10643

**Group:** A1

Email: zeyadhassan960@gmail.com

GitHub: [ZeyadHassan41 (Zeyad Hassan) (github.com)](https://github.com/ZeyadHassan41)

LinkedIn: [Zeyad Hassan | LinkedIn](https://www.linkedin.com/in/zeyad-hassan-907b7a244/)

**asd**

**Content**

1. **Introduction**
2. **Discuss The Problem**
3. **Project Structure**
4. **Data scraper**
5. **Search Engine**
6. **Data Export**
7. **Implementation and GUI**
8. **Introduction**

examining the Airfoil Coordinates Visualization Tool offers insights into various aspects of software development, including web scraping, user interface design, data visualization, and algorithm implementation.

1. **Discuss The Problem**

When we look at the website ( [UIUC Airfoil Data Site (illinois.edu)](https://m-selig.ae.illinois.edu/ads/coord_database.html) ) there no a quick method helps us to find the air foil we need.

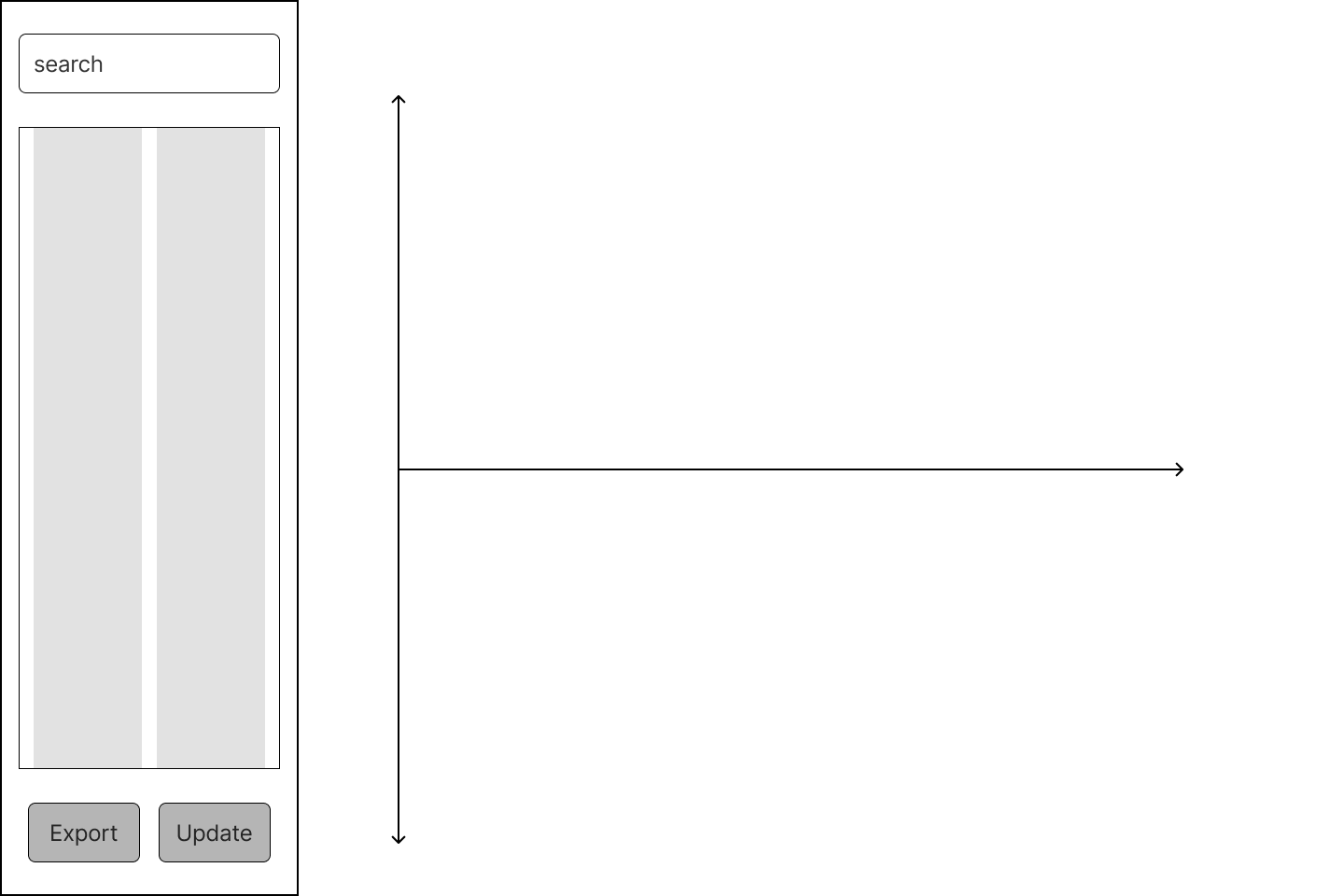
The fastest and most optimized solution for this problem is create **Search Engine** that helps the user to find the airfoil coordinates and gives a suggestion if the user entered a wrong airfoil name.

And create a **Graphical User Interface (GUI)** which will make dealing with the program easier for non-specialists.

Adding **plot coordinates chart** will make dealing with the graph more efficient in the future.

1. **Project Structure**

**UI Design**

****

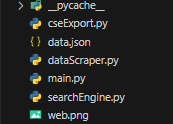
As we see in the above figure. The app consists of:

1. search bar
2. **List section Right list for x points and lift for y**, and **suggestions** will be displayed here if the search input not accurate.
3. **Control Button** lift button is export data (airfoil, x and y cords) in a csv file.
4. **Plot canvas** for present the diagram.

**Code Structure**

Every implementation in our app will be built in sperate File or another word object (OPP).

Python has a lot of libraries that will help us.

Every implementation will be explained in its section

* dataScraper.py

To get the data, we will need to scrape the web site ( [UIUC Airfoil Data Site (illinois.edu)](https://m-selig.ae.illinois.edu/ads/coord_database.html) ) beautifulsoup4 library is a good as it not very complex and has the basic method we needed.

* searchEngine.py

Our will just search in the JSON file which produce by the Data Scrape. The Search Engine will display suggestions if the input is not accurate. These suggestions are based on every character user Entered.

* csvEsport.py

As we see in the UI Design our data will be displayed in two lists x and y. So, to be easy for the user to access the airfoil data he has searched he can save it in csv file which can use in excel, solid works, etc...

* main.py (Interface)

main.py is file which will control all previous implementations how it will work in a pretty GUI.

1. **Data Scraper**

This Python script defines a class `AirfoilDataScraper` to scrape airfoil coordinate data from a specified URL and save it into a JSON file. Here’s a detailed description of the code:

**1. Imports:**

- `requests`: To send HTTP requests and handle responses.

- `BeautifulSoup` from `bs4`: To parse HTML content.

- `json`: To handle JSON file operations.

**2. Constants:**

- `URL\_START`: A base URL for the airfoil coordinates database.

**3. AirfoilDataScraper Class:**

- \*\*Initialization (`\_\_init\_\_` method)\*\*:

- `self.url`: Stores the URL to scrape.

- `self.data\_links`: Initializes an empty list to store the links to the airfoil coordinate files.

- `self.airfoil\_data`: Initializes an empty dictionary to store the scraped airfoil data.

- \*\*Data Scraping (`scrape\_data` method)\*\*:

- Sends a GET request to `self.url` and checks if the response is successful.

- Parses the HTML content of the response using BeautifulSoup.

- Finds all anchor (`<a>`) tags and extracts the URLs that start with "coord/".

- For each valid URL:

- Sends a GET request to fetch the airfoil coordinate file.

- Extracts the airfoil code from the URL.

- Parses the content of the coordinate file, extracting `x` and `y` coordinates from lines that contain exactly two numbers.

- Stores these coordinates in `self.airfoil\_data` using the airfoil code as the key.

- \*\*Save Data (`save\_data\_as\_json` method)\*\*:

- Takes a filename as an argument.

- Writes the `self.airfoil\_data` dictionary to a JSON file with the specified filename, formatted for readability with an indentation of 4 spaces.

**4. Example Usage:**

- Create an instance of `AirfoilDataScraper` with the desired URL.

- Call the `scrape\_data` method to collect the airfoil coordinates.

- Call the `save\_data\_as\_json` method to save the collected data into a JSON file.

```python

# Example Usage:

# scraper = AirfoilDataScraper("https://m-selig.ae.illinois.edu/ads/coord\_database.html")

# scraper.scrape\_data()

# scraper.save\_data\_as\_json("airfoil\_data.json")

```

This script automates the process of collecting airfoil coordinate data from a web page, parsing it, and saving it in a structured JSON format, making it easier to analyze and utilize the data in other applications.

1. **Search Engine**

This Python script defines a class `Search` that provides functionality to search for words or codes in a dictionary. Here's a breakdown of the code:

**1. Imports:**

- `json`: This module is imported to handle JSON data.

**2. Search Class:**

- \*\*Initialization (`\_\_init\_\_` method)\*\*:

- `self.dict`: Initializes with a dictionary passed as input.

- `self.code\_ser`: Initializes with a default value of "none" for the word/code being searched.

- `self.suggestions\_lst`: Initializes an empty list to store suggestions.

- \*\*Search Helper (`search\_helper` method)\*\*:

- Checks if the input code exists in the dictionary.

- If the code does not exist, it iterates through prefixes of the input code to find suggestions.

- Starts with the full input code and gradually shortens the prefix until a match is found.

- For each prefix, it iterates through the dictionary keys, checking if any key starts with the current prefix.

- If a match is found, it adds the matched key to the suggestions list.

- If the code exists in the dictionary, it returns the corresponding value.

**3. Example Usage:**

- Instantiate the `Search` class with a dictionary.

- Call the `search\_helper` method to search for a word/code and get suggestions or the corresponding value.

1. **Data Export**

This Python script defines a class `Export` for exporting airfoil coordinate data to a CSV file using the `filedialog` module from the Tkinter library and the `csv` module. Here's a breakdown of its components and functionality:

**Libraries:**

- \*\*from tkinter import filedialog\*\*: Imports the `filedialog` module from the Tkinter library, which allows the user to select a directory for saving the CSV file.

- \*\*import csv\*\*: Imports the `csv` module for handling CSV file operations.

**Export Class**:

- \*\*Initialization (`\_\_init\_\_` method)\*\*:

- Initializes the Export object with the airfoil code (`code`), X coordinates (`x\_coords`), and Y coordinates (`y\_coords`).

- Opens a file dialog for the user to select the directory where the CSV file will be saved.

- Stores the selected directory and other attributes.

- \*\*Export Data (`export\_data` method)\*\*:

- Checks if a directory was selected by the user.

- If a directory was selected:

- Constructs the file name for the CSV file based on the airfoil code and selected directory.

- Opens the file in write mode and creates a CSV writer object.

- Writes the airfoil code as the first row, followed by the column headers "x" and "y".

- Writes the X and Y coordinates to the CSV file row by row using the `writerows` method and the `zip` function to combine the X and Y coordinate lists.

- Prints the file name to the console for confirmation.

- If no directory was selected, the method does nothing.

This class provides a convenient way to export airfoil coordinate data to a CSV file, allowing users to specify the output directory and file name. It ensures that the exported CSV file contains the necessary headers and data format for easy interpretation and analysis.

1. **Implementation and GUI**

This Python script is a graphical user interface (GUI) application for visualizing and exporting airfoil coordinate data. Here's a breakdown of its components and functionality:

**Libraries and Modules**:

- \*\*matplotlib.pyplot\*\*: Used for plotting and visualizing airfoil coordinates.

- \*\*tkinter\*\*: Provides the GUI toolkit for creating the application's interface.

- \*\*matplotlib.backends.backend\_tkagg.FigureCanvasTkAgg\*\*: Allows embedding matplotlib figures within Tkinter applications.

- \*\*json\*\*: Used for handling JSON data.

- \*\*dataScraper\*\*: A custom module for scraping airfoil data from a specified URL.

- \*\*searchEngine\*\*: A custom module for searching airfoil data.

- \*\*cseExport\*\*: A custom module for exporting airfoil data.

**Airfoil Database URL:**

- \*\*URL\*\*: Specifies the URL of the airfoil database to scrape data from.

**GUI Standard:**

- \*\*root\_size\*\*: Sets the size of the Tkinter window.

- \*\*root\_bg\*\*: Sets the background color of the Tkinter window.

- \*\*font\*\*: Defines the font style and size for text elements.

- \*\*button\_bg\*\*, \*\*input\_bg\*\*, \*\*font\_color\*\*: Define colors for buttons, input boxes, and text.

- \*\*xpadding\*\*, \*\*ypadding\*\*: Define padding values for layout.

**Loading Data:**

- Loads airfoil coordinate data from a JSON file named 'data.json'.

**Tkinter Window Setup:**

- Creates a Tkinter window with specified attributes such as title, size, and background color.

**Plot Canvas Setup**:

- Initializes a matplotlib figure and canvas for plotting airfoil coordinates.

**Search Box and Search Button:**

- Provides a text entry box for users to input airfoil codes.

- Implements a search function to find and display corresponding airfoil coordinates.

- Displays suggestions for airfoil codes as the user types.

- Binds the Enter key to trigger the search function.

**X and Y Coordinate Lists:**

- Displays the X and Y coordinates of the selected airfoil.

- Updates the plot canvas with the selected airfoil coordinates.

**Export Button:**

- Exports the selected airfoil coordinates to a CSV file when clicked.

**Update Button:**

- Triggers a data scraper to update the airfoil coordinate data from the specified URL.

**Main Loop:**

- Starts the Tkinter event loop to run the application.

This script provides a user-friendly interface for visualizing and exporting airfoil coordinate data, along with options to update the data from an external source.