NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & IT

SSCS (WITH SPECIALIZATION IN DATASCIENCE)

CT-159 DATA STRUCTURES ALGORITHMS

PROJECT TITLE: WEATHER FORECASTING USING DATA STRUCTURES

Group Members:
M.Hamza DT-048
Afifa Siddique DT-003
Dania Karrar DT-007
Maryam Ashraf DT-050

Submitted to:
S.Faiza Naseem

1. Introduction:

The "MDAH Weather App" is a Python-based weather application that allows users to retrieve real-time weather information for various cities. The application features a graphical user interface (GUI) built using the Tkinter library. Users can input a city name, and the application fetches weather data from the OpenWeatherMap API, displaying details such as weather conditions, temperature, and pressure.

2. Tools and Techniques Used:

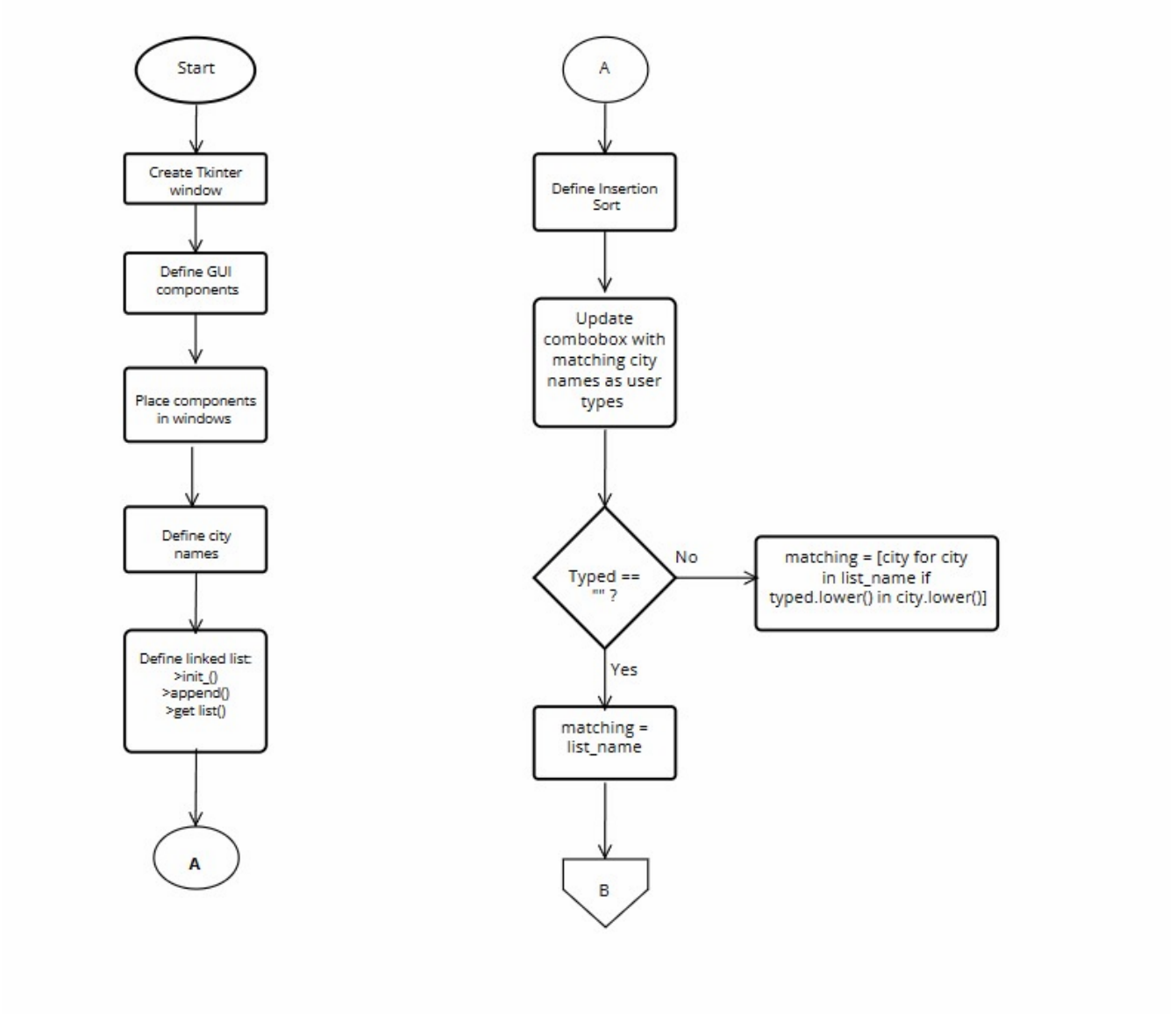
Tools:

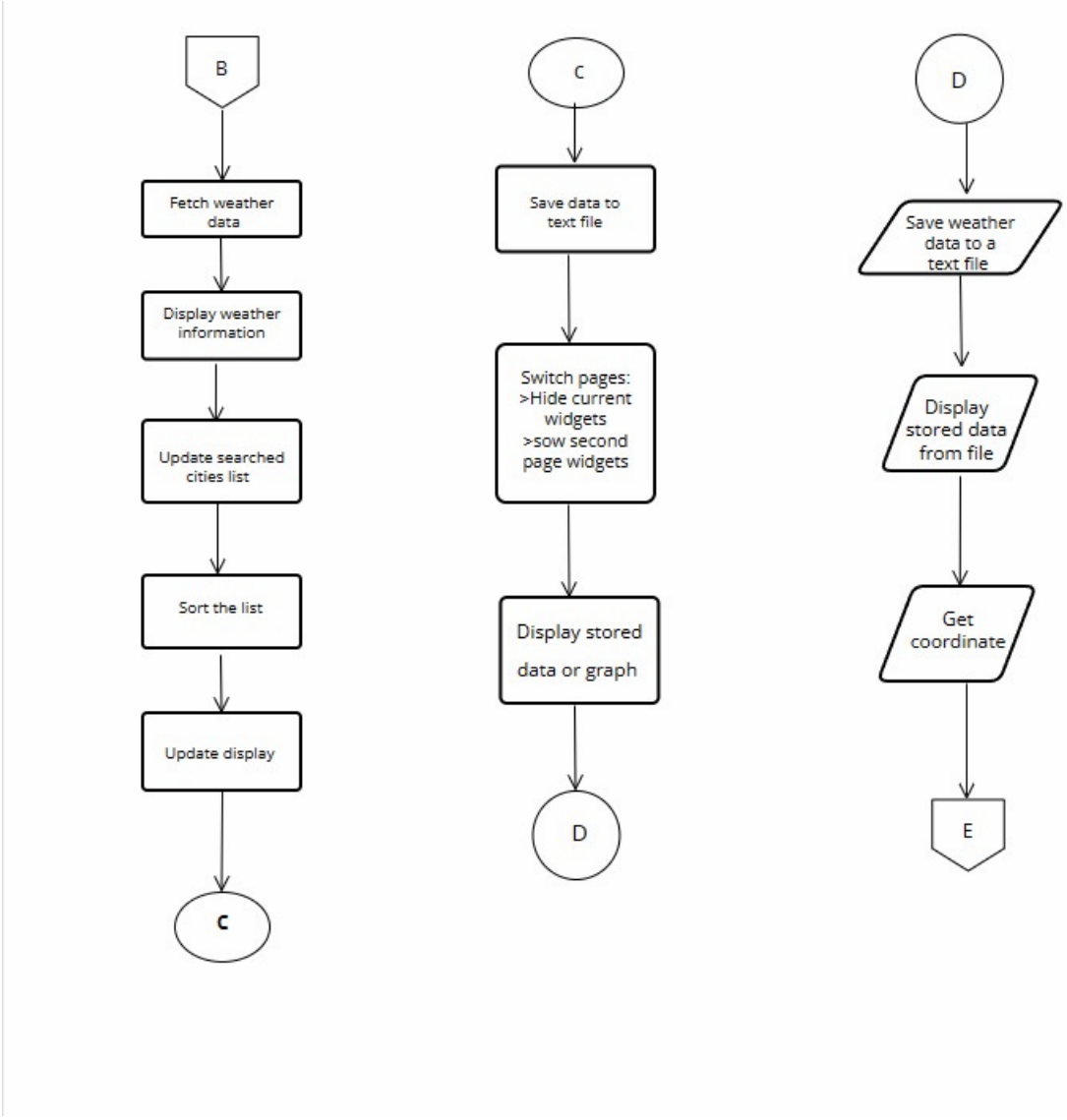
- Python: The primary programming language used for development.
- Tkinter: A Python GUI toolkit used for creating the application's graphical interface.
- Requests: A library for making HTTP requests, utilized for fetching weather data from the OpenWeatherMap API.
- JSON: Used for handling and parsing data obtained from the API.
- CSV: Utilized for reading city coordinates from a CSV file.
- Datetime: Employed for obtaining the current date and time.

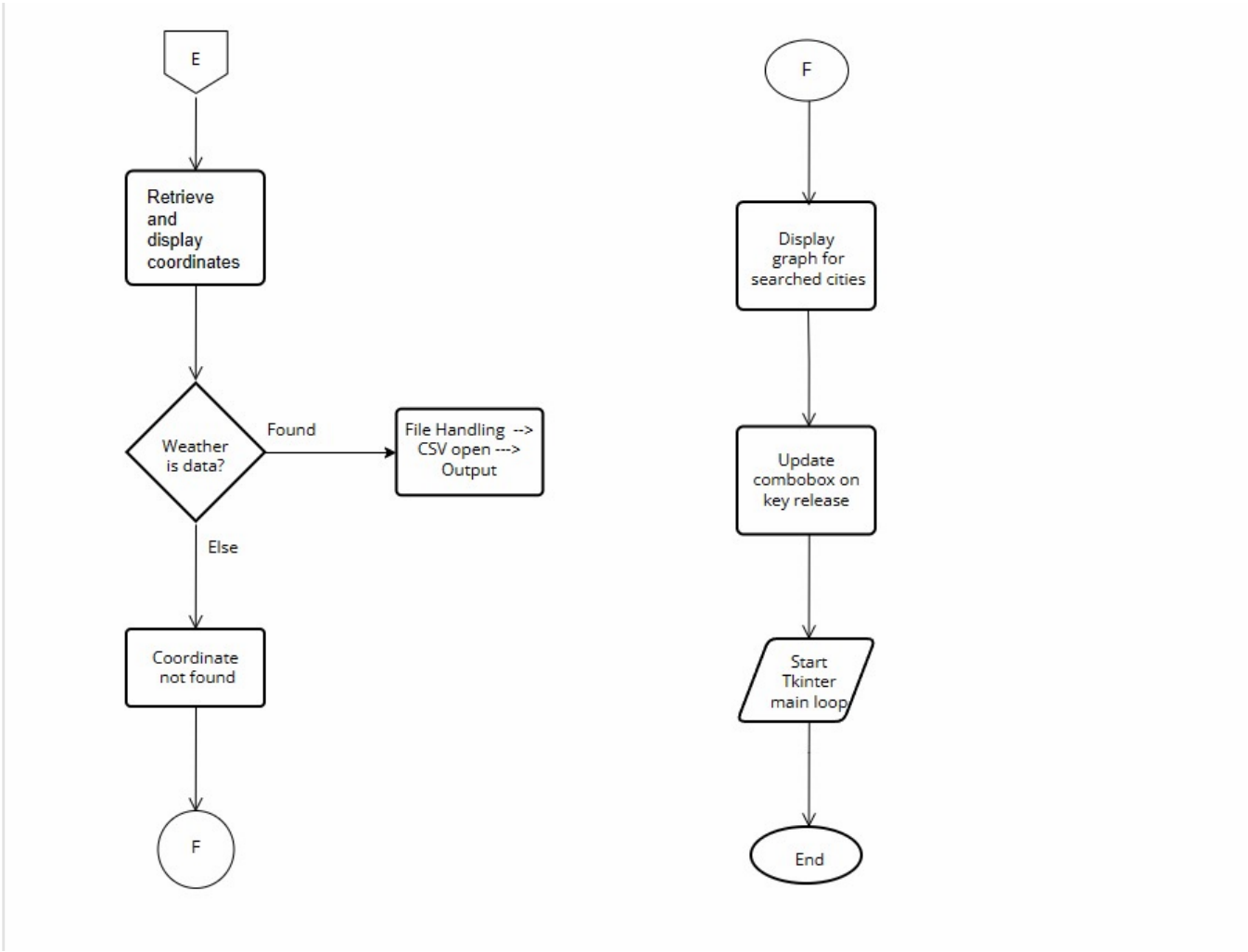
Techniques:

- Linked List: Implemented a linked list data structure to store and manage a list of searched cities.
- Stacks: To switch pages, push and pop opeartions have been used.
- Insertion Sort: Used for sorting the list of searched cities alphabetically.
- Linear Search: For searching the city name.
- File Handling: Created functions to save weather data to a text file and read/display content from it.
- Canvas and Scrollbar: Incorporated a canvas widget with a scrollbar to display stored data with scrolling capability.
- Graphical Visualization: Developed a graphical representation of temperature-pressure relations for searched cities.
- Combobox: Implemented a dropdown list for city selection using the Tkinter Combobox widget.
- API Integration: Fetched weather data from the OpenWeatherMap API using HTTP requests.
- Event Handling: Employed event handling to update the Combobox dynamically as the user types.
- City Coordinates CSV File: A dataset containing latitude and longitude coordinates for various cities. This data is utilized for displaying the coordinates of a selected city in the application.

2. Flow Diagram:




B
Fetch weather data
Display weather information
Update searched cities list
Sort the list
Update display
C
C
Save data to text file
Switch pages:
>Hide current widgets
>sow second page widgets
Display stored data or graph
D
D
Save weather data to a text file
Display stored data from file
Get coordinate
E


E
Retrieve and display coordinates
Weather is data?
Found
File Handling --> CSV open ---> Output
Else
Coordinate not found
F
F
Display graph for searched cities
Update combobox on key release
Start Tkinter main loop
End

4. Main Algorithms Implemented in the Project:

- **Insertion Sort Algorithm:**

Description: Used to sort the list of searched cities alphabetically.

Implementation: The insertion sort algorithm iterates through the list, comparing and rearranging elements to maintain the sorted order.

- **Linear Search Algorithm:**

Description: Used to search city names

Implementation: The algorithm updates the combobox suggestions by filtering the list of city names based on a case-insensitive substring match with the user's input. It then dynamically displays the matching cities in the combobox and searching co-ordinates from the csv files and then saving the co-ordinates in the text file

- **Fetch Weather Data Algorithm:**

Description: Retrieves weather data from the OpenWeatherMap API for a given city.

Implementation: Utilizes the Requests library to make an HTTP request to the API, providing the city name and API key. The obtained data is then processed and returned.

- **Get Coordinates Algorithm:**

Description: Retrieves latitude and longitude coordinates for a given city from a CSV file.

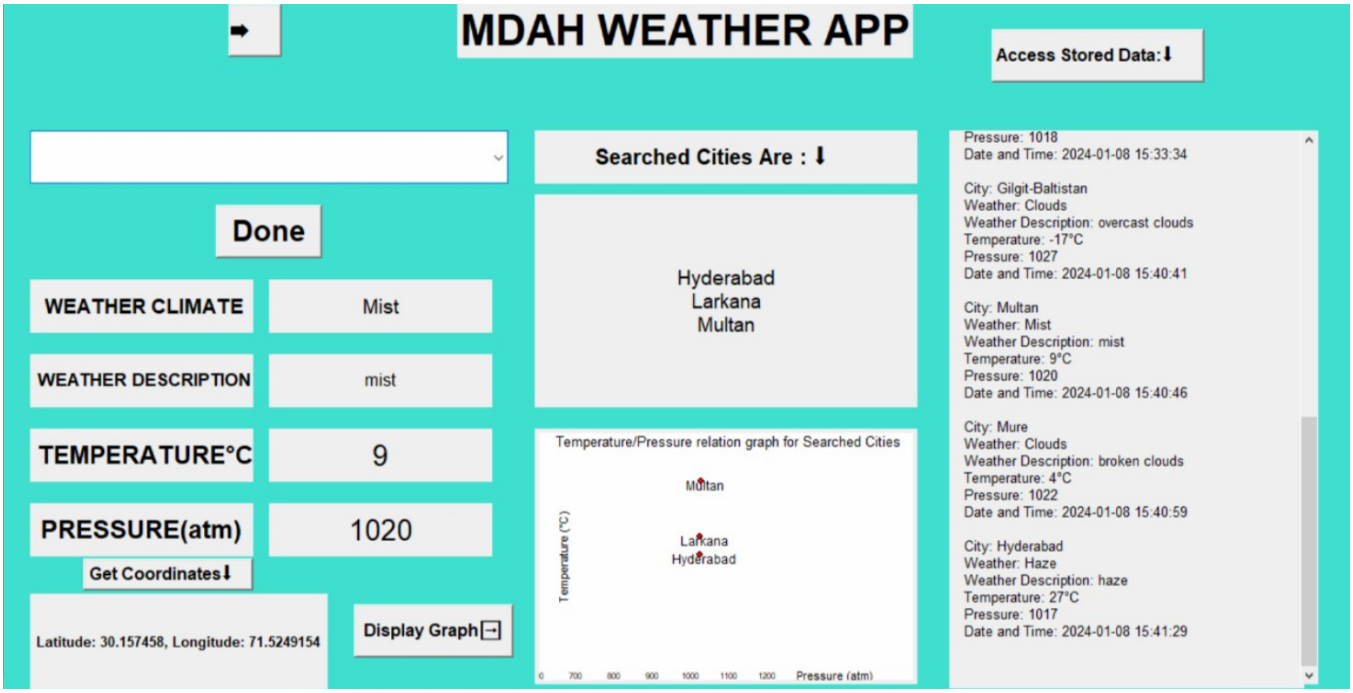
Implementation: Reads the city coordinates from a CSV file using the CSV library, searching for a match based on the city name.

- **Display Graph Algorithm:**

Description: Generates a graphical representation of the temperature-pressure relations for searched cities.

Implementation: Uses the Canvas widget to draw circles at specific coordinates based on temperature and pressure values. City names are displayed next to the corresponding points, creating a scatter plot.

5. OUTPUT:



Conclusion:

In conclusion, this weather application demonstrates the integration of GUI development, API usage, and basic data handling. The project provides a user-friendly interface for fetching and visualizing weather data, along with features for data persistence and graphical representation. The implementation highlights the use of various programming concepts and techniques to create a comprehensive weather application.