**Weather Dashboard Project Report.**

**Introduction**

Python software projects open a possibility of students implementing concepts related to the understanding of programming in the real world. Python is a versatile programming language that enables one to create real-world applications using external sources of data and user interfaces. Weather is a type of data that is very crucial and common as people and businesses rely on it when organising daily activities, agriculture, travelling and health. In this project, the researcher aims to create a Python weather dashboard to retrieve real-time and predicted weather information using an external weather application programming interface (API). The dashboard is a straightforward and yet efficient means of the user to have access to weather data in a graphical user interface (GUI).

The project objective is to develop a useful and convenient tool that illustrates the object-oriented programming (OOP), file management, exception management, and API exploration. The version control applied in the system is also with GitHub on project files management. Under the project, designing this dashboard will serve as a useful experience in creating a full software system that integrates various components of the Python ecosystem, including Tkinter, which has been used as part of the GUI, along with JSON manipulation, file storage, and external APIs to obtain real-time weather conditions (GeeksforGeeks, 2025a; Real Python, n.d.).

**Project Objectives**

The project has the following major goals:

* To create and develop a weather dashboard based on Python 3.
* To include Weather API to get real-time information.
* To develop a GUI using Tkinter that gives the weather information in a visually appealing manner.
* To enable users to save and manage favourite cities to be handled by use of JSON files.
* In order to use exception handling of API calls, file errors, and invalid input.
* To use version control in Git and GitHub to control development.

These aims will satisfy the requirements of the course and will result into a project demonstration that will depict the required technical concepts but will also give a practical product.

**Requirements and Tools of Systems.**

The programming language applied in the project is Python 3.x. The functionality is supported by a number of Python libraries. It is possible to request the API with the help of the requests library and in the case of failed connexions or wrongful replies the exception management techniques are applied (Requests Authors, n.d.). The libraries of json store and read favourite cities in a local JSON file (Python Software Foundation, 2025). The GUI library is used tkinter due to the fact that it is lightweight, default bundles with Python, and it is easy to learn among university students (GeeksforGeeks, 2025b).

In the case of the weather data, the project employs a free and useful service, WeatherAPi which can be used to retrieve the current weather and projections (Weatherapi.com, 2025). To process weather icons one utilises Pillow library, which enables one to add images to the Tkinter interface (Pillow, n.d.).

The system also has certain prerequisites on what a basic computer with Python installed and that the computer is connected to the internet in order to access weather data. Its application is on windows, Linux, and Mac. To do the version control, Git is employed to store changes, and the project is pushed to GitHub where it is stored as backup and collaboratively (GitHub Docs, 2025).

**System Design**

The project is a client-side application that is not that complicated. The user is interacted with the GUI where s/he enters or selects a city. This query is sent to the Weather API service which responds through a JSON query. This reply is inputted and shown in the Tkinter window. City favourites can also be saved locally by the user in a JSON file.

The software is of object-oriented nature. The primary one WeatherApp is the one dealing with the GUI and assembling other parts. The WeatherAPI class has Weather API requests and response processing. FileHandler class is used to take care of reading and writing of the files of the pinned cities. This division of roles simplifies this code and allows this code to be maintained (GeeksforGeeks, 2025c).

The interface is created to showcase all features on a single window. The upper section comprises the city input section, a search button and a pinned city drop down menu. The intermediate part displays the present weather conditions, such as temperature, condition, humidity, and speed of wind. There are also icons since a user finds it easier to use. A unit toggle switch changes to Celsius or Fahrenheit. Its bottom section shows a 5-day forecast where the cards are placed at horizontal orientation. The cards display the day, weather icon and maximum and minimum temperatures (Real Python, n.d.).

The flow of data will start with user typing in or choosing a city. A request based on API is sent and theJSON response is decoded. Real weather and forecast values are read and updated on the GUI. In case the user opts to save the city, the file handler is used to save it in the JSON file. These favourites are auto loaded every time the programme is restarted.

**Implementation**

The application is based on WeatherAPI having an API key to request weather details. The GET request is uttered with the parameters like city name, units and forecast days. The fields on the JSON response include location.name, location.country, current. temp c, forecast.forecast day, and so on. These are the values that are processed and represented in the Gui (WeatherAPI.com, 2025).

The file management is performed with the application of JSON. Pinned cities are saved in the list format in a file named favourites.json i.e. {"favourites": ["New York", "Berlin"]}. On pinning an urban area, it is appended to the file. When the application starts, the file is read and the drop down menu is filled with such values. This indicates the utilisation of the built Python JSON library in the storage of data (GeeksforGeeks, 2025d).

The application deals with various errors. In the event that the user inputs an invalid city, he/she is presented with an error message. In the event that the API request has failed due to network problems, then the programme creates a message that there is no internet connexion. The try-except block is also used to handle file handling errors, e.g. missing or corrupted files. Exception handling enhances the stability of the programme (Requests Authors, n.d.).

The layout of the GUI was done in Tkinter. The interface was built with frames, labels, and buttons as well as dropdown menus. Pillow library is used to manipulate images, whereas Tkinter grid system is used to organise a row of forecast cards (Pillow, n.d.; GeeksforGeeks, 2025e).

The application consists of:

* Automatic IP-based IP-based lookups.
* Reflecting the present weather condition of temperature, condition, humidity, and the speed of the wind.
* Icons for visual clarity.
* A 5-day forecast in card format.
* Unit switching between Celsius and Fahrenheit.
* Tabbing of favourite cities with dropdown menu.
* Exception management of all fundamental functions.
* Testing and Results

Valid inputs presented results in an appropriate manner. Also, invalid inputs like random text displayed an unambiguous error message. The network error handling was a result of disconnection of the internet. Refixed cities were reloaded and restarted, and they worked with files correctly.

The forecast cards presented the right details of each day. The unit switch was well synchronised between Celsius and the Fahrenheit. On balance, the tests revealed that the programme was adequate to all project requirements and functioned well.

**Difficulties and Solutions.**

One of the issues was how to make sure that pinned cities were properly placed in the JSON file. Initially, the file was not a list of data but rather a dictionary-like. This has been resolved through proper formatting of the file handling code (Python Software Foundation, 2025). The other problem was that of ensuring the GUI was attractive and at the same time very simple. This was done through the use of frames and icons, and not so elaborate design.

The location detection was also to be integrated meaning more API calls back with wrong defaults were sometimes given. This was enhanced through the request parametering. Lastly, the handling of errors was also tricky and the use of try except enabled all most of the main errors to be handled successfully.

**Version Control with GitHub**

In this project, Git was utilised as a means of version control. All the development phases were done, including the first API connexion, GUI configuration, and file management addition. These commits were sent to GitHub. GitHub was a free resource that offered a copy of the project, history of the changes, and any chance to share the work with other people. Collaborative software development is a variant to which version control is a critical skill, and this experience with GitHub was worthwhile in this project (GitHub Docs, 2025).

**Future Enhancements**

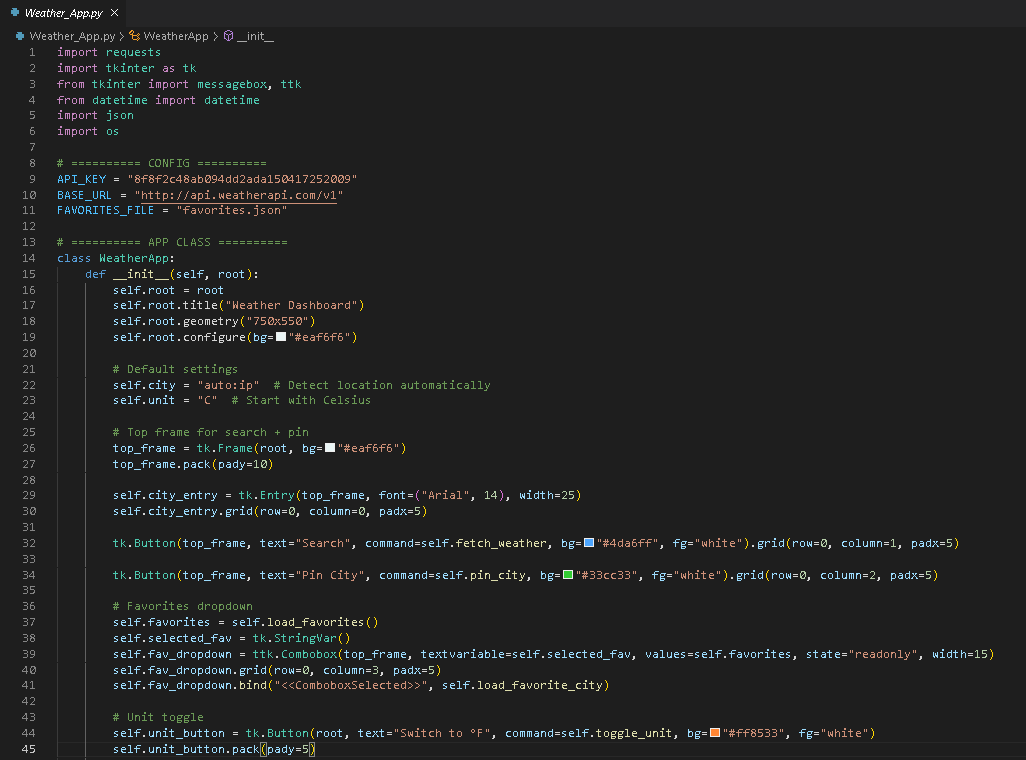
There are a number of additions that can be made in future versions. Additional weather information can give more information including the sunset and sunrise time, the quality of air, and the UV index. Several favourite cities would be allowed in a panel rather than a drop down. The mobile-sized windows should have a more appropriate responsive design. Lastly, the support of alternative weather APIs might be considered to have redundancy and result comparison.

**Conclusion**

In this project, the Python-based weather dashboard was implemented successfully, and several concepts of programming were integrated into a work system. It provided object-oriented programming, file manipulation, exception, API combination and GUI construction. The app is handy, easy to use and offers real time weather which is helpful in day to day life.

The project provided practical experience in developing software that communicates with external data and stores and manipulates local files and visualising data. GitHub was used as a version control, which provided the extra level of professionalism. On the whole, the project accomplished its goals and became a helpful learning process of real software creation.

**Screenshots of Code and App**



A screen shot of a computer program

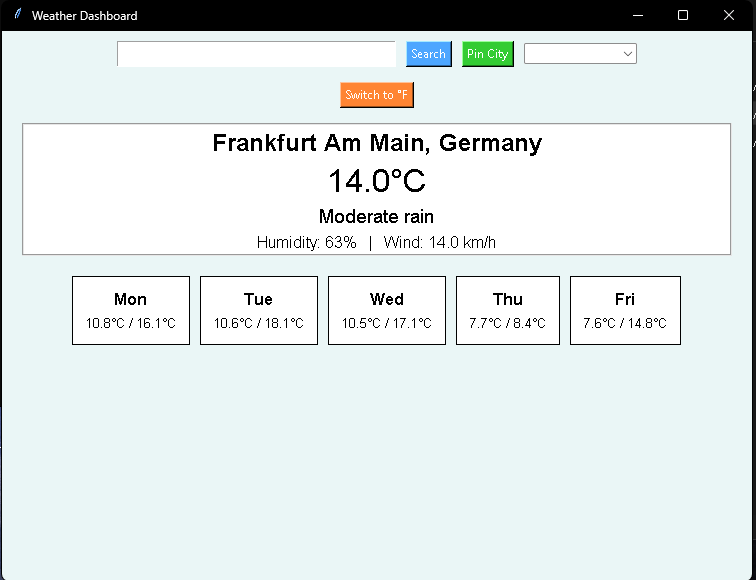
AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.



**References**

GeeksforGeeks (2025a) *Read JSON file using Python*. Available at: [https://www.geeksforgeeks.org/read-json-file-using-python/](https://www.geeksforgeeks.org/read-json-file-using-python/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

GeeksforGeeks (2025b) *Python Tkinter Tutorial*. Available at: [https://www.geeksforgeeks.org/python/python-tkinter-tutorial/](https://www.geeksforgeeks.org/python/python-tkinter-tutorial/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

GeeksforGeeks (2025c) *Working With JSON Data in Python*. Available at: [https://www.geeksforgeeks.org/python/working-with-json-data-in-python/](https://www.geeksforgeeks.org/python/working-with-json-data-in-python/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

GeeksforGeeks (2025d) *Reading Images With Python Tkinter*. Available at: [https://www.geeksforgeeks.org/reading-images-with-python-tkinter/](https://www.geeksforgeeks.org/reading-images-with-python-tkinter/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

GeeksforGeeks (2025e) *Python Tkinter Tutorial*. Available at: [https://www.geeksforgeeks.org/python/python-tkinter-tutorial/](https://www.geeksforgeeks.org/python/python-tkinter-tutorial/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

GitHub Docs (2025) *GitHub Docs*. Available at: https://docs.github.com/ (Accessed: 20 September 2025).

Pillow (n.d.) *ImageTk Module — Pillow Documentation*. Available at: https://pillow.readthedocs.io/en/stable/reference/ImageTk.html (Accessed: 20 September 2025).

Python Software Foundation (2025) *The Python Standard Library — json*. In: Python Documentation. Available at: https://docs.python.org/3/library/json.html (Accessed: 20 September 2025).

Real Python (n.d.) *Building a Python GUI Application With Tkinter*. Available at: [https://realpython.com/courses/building-gui-application-tkinter/](https://realpython.com/courses/building-gui-application-tkinter/?utm_source=chatgpt.com) (Accessed: 20 September 2025).

Requests Authors (n.d.) *Requests exceptions — Requests API Documentation*. Available at: https://docs.python-requests.org/en/latest/ (Accessed: 20 September 2025).

WeatherAPI.com (2025) *WeatherAPI Documentation*. Available at: [https://www.weatherapi.com/docs/](https://www.weatherapi.com/docs/?utm_source=chatgpt.com) (Accessed: 20 September 2025).