**NAME: AGBOLADE REBECCA**

**STUDENT ID: D3077427**

**Module Title:**

**Software for Digital Innovation**

**Title:**

**A case study related to weather is used to demonstrate the evaluation of software tools, safety concerns, and programming frameworks in the development of an application.**

Contents

[**Introduction** 3](#_Toc155740470)

[**Evaluation of Software Tools** 3](#_Toc155740471)

[**Structures of Data and Basics of Programming** 4](#_Toc155740472)

[**Reviewing Software Tools: Recognising Benefits and Drawbacks** 4](#_Toc155740473)

# **Introduction**

In this evaluation, I explored a tool interacting with the Open Meteo API, focusing on its functionality in gathering, analysing, and interpreting weather-related information. The application has two stages: an initial phase and an extended phase. I delved into the software tools utilized, their compatibility, constraints, security and risk implications, and their application to fundamental programming concepts and data structures.

# **Evaluation of Software Tools**

**1. SQLite for Database Management Suitability:**

The programme has selected SQLite as its Database Management System (DBMS), which makes sense given its simplicity to utilise seamless integration. The application's priorities align well with the optimal resource management offered by the sqlite3 library and environment controller implementation.

Limitations: SQLite does have limitations, though. Although it operates well for the current application, handling bigger data sets may present difficulties with scalability. Furthermore, it may not be able to handle progressively complicated tasks that require for more powerful database systems.

**2. Data Visualisation using Matplotlib:**

Relevance: A variety of charts, including multi-line charts, grouped bar charts, and scatterplots, was created with Matplotlib in this case study, which is used frequently for data visualisation. Its versatility makes it an appropriate match for the application by improving one's awareness of weather-related patterns. Matplotlib also works well with other Python modules, facilitating a cohesive and scalable development environment.

Restrictions: When interacting with complex visualisations, Matplotlib's learning curve may be steeper. Further tweaking may be necessary to attain the desired aesthetics because the pre-set appearances of some charts may not be suitable.

3. **Requests for Appropriate API Interaction:** The requests part of the package works well for minimising HTTP requests to the OpenMetEO API. Its ease of use renders it easier to retrieve weather-related data in JSON format, which enhances the application's overall effectiveness. The library's adept management of abnormalities and fixes also simplifies the difficult process of handling failures during API requests.

Restrictions: Requests do have certain restrictions, though. The application's performance may be hampered by its inadequacy to handle multiple in tandem queries at once, especially in situations where quick data retrieval is essential. Furthermore, managing the means of authentication of some APIs may provide obstacles, which reduces the versatility of the entire framework.

Implications for Security and Risk

Significant security and risk challenges arose during the development of the weather-related application, primarily in respect to the incorporation of other APIs like the OpenMetEO API. The protection of sensitive information, in particular API keys, which are vital to authenticating queries to the API, was one main worry. It became essential to develop suitable safety procedures, including using environment variables, to address This

The given code uses an API key, a private piece of data that needs to be protected, to access the OpenMetEO API. The code includes the practice of storing API keys in environment variables to reduce the danger of unauthorised access and potential security breaches. The above approach adheres to secure coding best practices and improves security at the same time.

# **Structures of Data and Basics of Programming**

1. Combining Clauses and Functions

The source code structure of the application incorporates basic programming concepts, using functions and clauses for a flexible and structured design. To improve code comprehension and sustainability, the insert\_daily\_weather\_entry function, for example, incorporates the procedure of putting weather data into the database.



**2. Python Data Structures**

The application makes optimal use of Python's robust data structures to manage a broad range of data kinds. Python's data structures can be used in practical applications, as seen using for loops, which make it simple to iterate over query results and API answers.

# **Reviewing Software Tools: Recognising Benefits and Drawbacks**

It's apparent from the conclusion of our investigation into the software tools—Requests, Matplotlib, and SQLite—that each has pros and cons of its own. A detailed grasp of these tools' features is demonstrated by their use in the application context.

It is critical to acknowledge the drawbacks of these technologies and clarify that software development is not a field where there is a universal answer. This knowledge encourages a methodical approach to tool selection, assuring that the instruments used are in accordance with the specifications and limitations of the project.

In conclusion, the application development procedure shows a careful assessment of software tools in addition to utilising fundamental programming ideas and data structures. This holistic approach establishes the structure to foster growth.

As we get to the end of our research into software tools for our weather-related application, Requests, Matplotlib, and SQLite each have certain inherent limits in addition to their many advantages. Every tool has a distinct function that adds to the functionality of the application. But it's important to recognise that there isn't a single tool that works for every situation, and that means knowing the minute details of each one.

Security precautions are crucial, particularly when managing sensitive data such as API keys. Not only is it best practice to safeguard these keys via environment variables, but it's also essential to maintaining the integrity of our application. This hands-on training has shown how secure coding techniques have real-world applications, which is a lesson that goes beyond the classroom setting. We've used basic programming ideas like functions and clauses, which greatly enhanced the code's readability and coherence. Python's modular data structures, as exemplified by my code samples, have shown to be quite useful when managing a variety of data kinds. These are the foundations of real-world-working code, not merely theoretical ideas from lectures.

To sum up, this exploration of software tools, security issues, and the foundations of programming has been enlightening. Making code function is not enough; it also needs to be done securely and effectively. I've grown to appreciate the delicate equilibrium that must be struck when choosing and utilising tools, being aware of their advantages and disadvantages, and accepting the constant learning that is a part of the ever-evolving area of application development as a student navigating this field. These interactions have benefits as the starting point for this project as well as the continuous process of adjusting and developing in the dynamic field of coding and programming.