**Temperature Automation Project Report**

**Course:** Low-Code Development  
**Student:** Maryam Musawi  
**Date:** Aug 20, 2025

**1. Introduction**

For this assignment, I created an automated system that collects live temperature data for different cities and updates an Excel spreadsheet automatically. The main goal was to replace manual temperature checking with a reliable automation that could run on its own.

**2. How It Works**

I built this using Python instead of UiPath because I work on a Mac and needed something that would work cross-platform. Here's what my automation does:

1. **Reads** city names from an Excel sheet (Column A)
2. **Fetches** current temperature and weather conditions from OpenWeatherMap API
3. **Updates** the Excel file with new data (Columns B and C)
4. **Runs automatically** every day at 8 AM (or can be run manually)

I chose to use an API instead of scraping Google because it's more reliable and doesn't break when websites change their design.

**3. Technical Stuff**

**What I Used:**

* **Python** with requests and openpyxl libraries
* **OpenWeatherMap API** (free tier)
* **Excel** for input/output
* **macOS** for development

**The Main Parts:**

The script reads each city name, calls the weather API, gets back the temperature and weather description, then writes everything to the Excel file. It also handles errors gracefully - if a city isn't found or the API is down, it marks those as "Error" instead of crashing.

**4. Challenges I Faced**

**Mac Compatibility:** Honstely, I tried to use UiPath but realized it doesn't work well on Mac. I switched to Python which was much better for cross-platform use.

**API Setup:** Getting the API key and figuring out how to make the requests was tricky at first, but once I got the first city working, the rest came together.

**Scheduling:** Making it run automatically at specific times was harder than I expected on Mac, so I implemented a simple time-checking system that works well.

**5. What Worked Well**

The automation is really reliable now. It can handle:

* Multiple cities at once
* API errors without crashing
* Different weather conditions
* Automatic Excel updates

I tested it with several cities and it worked perfectly every time. The Excel output looks clean and professional.

**6. What I Learned**

This project taught me that sometimes the simplest solution is the best one. Instead of using complicated low-code platforms, a well-written Python script can be just as effective and more portable.

I also learned how important error handling is in automation - things will go wrong (API limits, network issues), and good code should handle those gracefully.

As the sole contributor to this project, I handled all roles:

\*\*Business Analyst: \*\* Defined the automation requirements and objectives

\*\*Developer: \*\* Built the entire Python automation system

\*\*Quality Analyst: \*\* Tested and validated all functionality

**7. Conclusion**

This project successfully automates temperature data collection in a way that's reliable and efficient. It demonstrates practical low-code development skills using Python for real-world automation tasks.

Submitted by: Maryam Musawi

The code is available on GitHub at: https://github.com/maryammusawi

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.