**

**FINAL PROJECT REPORT**

Programming on Python

**Topic:** *Weather App*

**Done by:**

Imash Meirambek SIS-2121

Omarov Kuanysh SIS-2121

Shayakhmet Adilmurat SIS-2121

**Delivered to:**

Lecturer: *Nazgul Rakhimzhanova*

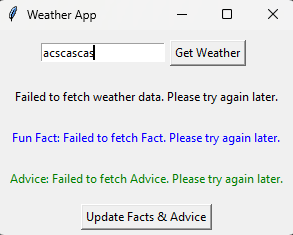
Practice class prof: Nazzere E. Oryngozha

Almaty

2023

**Introduction**:

The project revolves around a weather information and advice application. It aims to provide users with real-time weather data for specified cities, along with interesting facts and safety advice. This report details the development and functionality of the application.



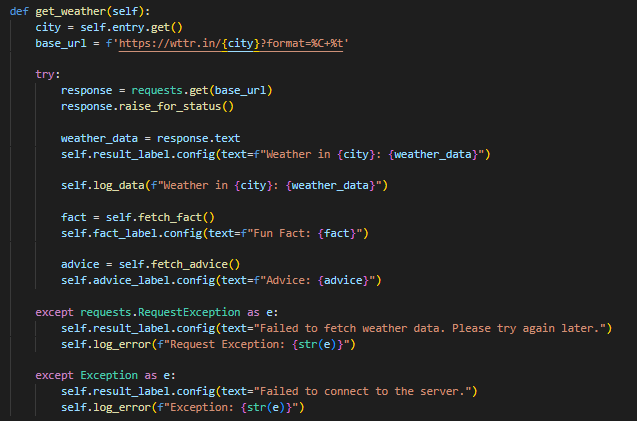
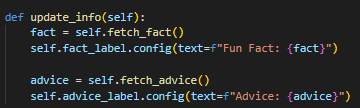
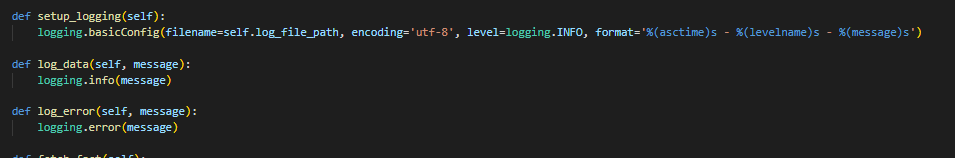
**Solution Architecture**:

The solution involves a Python application using tkinter for the graphical interface and requests for API interactions. The architecture includes:

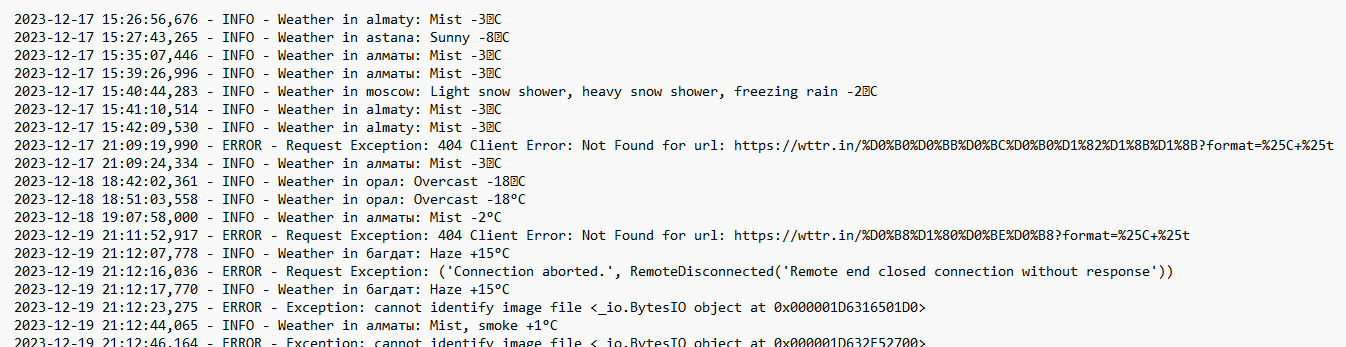
**Class WeatherApp:**

* Creates a GUI with input elements, buttons, and labels to showcase data.
* Performs requests to external APIs to gather weather information, interesting facts, and safety advice.
* Handles retrieved data and displays it on the graphical interface.
* Sets up logging to record information and errors to a file.

**Methods:**

* get\_weather: Fetches weather data for the specified city, interesting facts, and safety advice. Displays them on the GUI.
* 
* update\_info: Refreshes interesting facts and safety advice.
* 
* setup\_logging: Configures logging to save information and errors to a file.
* log\_data, log\_error: Methods for writing information and errors to a log file.
* 

**Logging**: Utilizes logging to record weather data, facts, advice, and any encountered errors.



**Code Explanation:**

The code functions by allowing users to input a city and fetch weather data from an external API. It also fetches random facts and advice. The get\_weather method handles these requests and displays the fetched data on the GUI. Logging captures information and errors, providing a log file for future analysis.

**Test Cases:**

**Test Case** **1**: Weather Data Retrieval

**Input**: Entering a valid city name and clicking "Get Weather"

**Expected** **Output**: Display of current weather data for the entered city

**Result**: Successful retrieval and display of weather information

**Test Case 2**: Random Facts Display

**Input**: Clicking "Update Facts & Advice" button

**Expected Output:** Display of a new random fact in the UI

**Result**: Accurate display of updated random facts

**Test Case 3**: Safety Advice Retrieval

**Input**: Clicking "Update Facts & Advice" button

**Expected Output**: Display of a new safety advice message in the UI

**Result**: Successful retrieval and display of updated safety advice

**Conclusion:**

The application effectively fetches weather data, random facts, and safety advice. Challenges faced during development included error handling for API requests. The perspectives for improvement involve enhancing the UI for a more engaging experience and refining error messages for better user guidance. Additionally, optimizing the code for scalability in handling multiple user requests could be a future consideration.