Summery of the project

Technology Used: For this problem statement, I have used Flask, a micro web framework written in Python that is widely used to develop web applications. It is lightweight and modular, which makes it a perfect choice for building RESTful APIs.

Dependencies: I have used the following dependencies for building the APIs:

- Flask: For creating the web application
- JWT: For token-based authentication
- datetime: For generating expiry time of the token
- functools: For wrapping the decorator functions
- requests: For making HTTP requests to the OpenWeatherMap API

Thought Process: The first step was to create a plan for building the APIs. The problem statement required us to create a login API, logout API, and get weather information API. The weather information API was supposed to return data for 30 cities in JSON format and had to be paginated with an ideal page size of 10 items.

I started by creating a Flask web application and defining the routes for the APIs. For authentication, I decided to use token-based authentication using JWT. I created a decorator function that would wrap the weather information API and would authenticate the user using the token.

To fetch weather data for 30 cities, I used the OpenWeatherMap API. I made HTTP requests to the API for each city and appended the responses to a list. I then implemented pagination using Python's built-in list slicing functionality.

To ensure that the code followed proper REST framework guidelines, I used appropriate HTTP methods (GET for retrieving data) and status codes (200 for successful requests and 401 for unauthorized requests).

Trade-Offs: One trade-off I made was to hardcode the username and password for authentication instead of using a database to store user credentials. This was done to keep the code simple and to focus on the main problem statement.

Other Solutions: One alternative solution for authentication would be to use OAuth2.0, a popular protocol for secure authentication and authorization. Another solution could be to use a database to store user credentials and implement a registration API for users to create their accounts. Also, we can internally pass the access token to the 3rd Api to call.

Overall, I chose Flask because it is a lightweight and modular framework that makes it easy to build RESTful APIs. JWT was chosen for authentication because it provides a secure and scalable

way to implement token-based authentication. By using these technologies, I was able to create a solution that met all the requirements of the problem statement.