1. (5 pts) Provide the problem requirements and the conceptual model in UML for your project. You can reuse the ones made in Project 1.

Project Proposal

Problem: People rely on weather forecasts to plan their day, but it's a hassle to check multiple sources for accurate updates. So, we need an application that will provide the current weather conditions and forecasts for your location, making it simple for everyone.

Objective: The goal is to create an app that uses a database to gather weather information from different places. This app will give users the current weather and forecasts for their location.

Potential Tables:

WeatherData
- weather_id (PK)
- location_id (FK)
- temperature
- humidity
- wind_speed
- precipitation
- weather_description (examples: cloudy, sunny)
- timestamp (the time and date of data retrieval)

Location
- location_id (PK)
- city
- country
- latitude
- longitude

User
- user_id (PK)
- username
- email
- password

UserLocation
- user_location_id (PK)
- user_id (FK)
- location_id (FK)

Alerts
- alert_id (PK)

location_id (FK)
alert_type (examples: severe weather, storm)
alert_description
timestamp (the time and date of alert)

Business Requirements:

Problem Statement: The current process of obtaining accurate weather forecasts involves checking multiple sources, which can be difficult and time-consuming for users. To address this issue, we aim to develop an application that provides users with current weather conditions and forecasts for their location, simplifying the weather-checking process.

Nouns: Orange
Verbs: Green

Requirements:

1. The system should gather weather data from various sources.
2. Users should be able to access the application to view the weather info.
3. The application should display the current weather conditions and forecasts.
4. Users should be able to view the hourly weather forecasts.
5. Users should be able to see the weather forecast for the entire day.
6. The system should store weather data including temperature, humidity, wind speed, precipitation, and weather description.
7. The system should store information regarding the location such as the city, country, latitude, and longitude.
8. Users should be required to register and login to access personalized weather info using their email and will have a password to login.
9. The system should provide alerts for severe weather conditions or storms.
10. Users should receive notifications for alerts relevant to their chosen locations.
11. The system should timestamp weather data retrieval and alerts.
12. The application should be user-friendly and intuitive for easy navigation.

Nouns:

- Weather
- Forecast
- Location
- User
- Alert
- Temperature
- Humidity

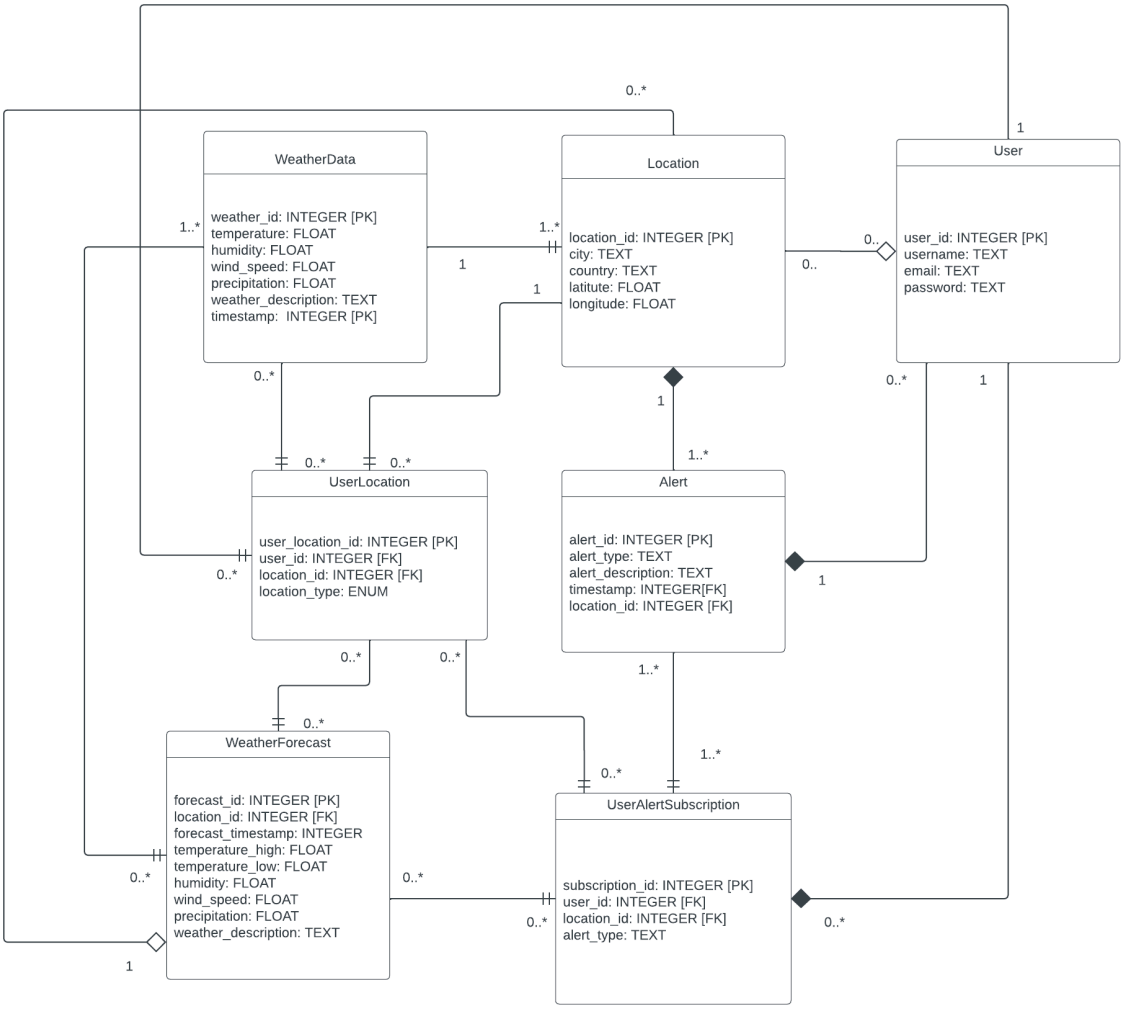
- Wind Speed
- Precipitation
- Weather Description
- City
- Country
- Latitude
- Longitude
- Username
- Email
- Password

Verbs:

- Gather
- Access
- Display
- View
- Register
- Login
- Store
- Provide
- Receive
- Timestamp

Conceptual Model (UML Class Diagram):

Lucid Charts Link:
https://lucid.app/lucidchart/023503e9-e397-4b48-b51f-287d15f42d79/edit?viewport_loc=-988%2C-501%2C3485%2C1896%2C0_0&invitationId=inv_77f3ec81-4c30-45ab-a5b9-6c79821a8d09



Functionalities that I selected to be used as an in-memory key-value storage

Regarding the Key-Value In-Memory Database implementation for this Weather application, I am planning to include the following functionalities:

- **Hourly Weather Forecast Storage**: I would like to store the hourly weather forecasts for the different locations.
- **Alerts Storage:** I would like to store alerts for severe weather conditions of storms.
- **User Preferences Storage**: I would like to store the user preferences such as preferred locations and alert settings.
- **Current Weather Data Storage**: I would like to store the current weather data for the different locations.

- Essentially, I would like to focus on efficiently storing and retrieving the weather data, the user preferences, and alerts using keys and values, with the keys representing the identifiers such as the location IDs or user IDs, and the values representing the data structures containing the weather information, user preferences, or alerts.

Step 2:

Describe the Redis data structures that you are going to use to implement the functionalities you described in the previous point.

1. Hourly Weather Forecast Storage: **I would like to use Hashes as the Redis Data Structure.**
   - What I am trying to expand on is how I can store hourly weather forecasts for different locations using hash objects. Each hash key can contain the location ID and the timestamp for the specific hour. The hash fields would represent the attributes such as temperature, humidity, wind speed, precipitation, and weather description.
2. Alerts Storage: **I would like to use Lists as the Redis Data Structure**
   - Here, what I am trying to expand more on are the alerts for severe weather conditions or storms that can be stored as lists in Redis. In this case, I would make each list represent the alerts for a specific location. The list key can contain the location ID, and each list item can contain details of the alert, such as the alert type, description, and timestamp.
3. User Preferences Storage: **I would like to use Hashes and Sets as the Redis Data Structures.**
   - Since I am trying to use both Hashes and Sets as the Redis Data Structures, regarding my usage of hashes, I would like to make it store the user profiles, where each hash key would represent a user ID, and the hash fields would store the user attributes like username, email, and more.
   - Regarding my usage of sets, I would like to make it store the user preferences such as preferred locations and alert settings. Each set key would contain the user ID, and the set members would represent the preferred location IDs or the subscribed alert types.
4. Current Weather Data Storage: **I would like to use Hashes as the Redis Data Structure.**
   - This will be a little similar to hourly weather forecasts, where the current weather data for different locations would also be stored as hash objects. Each hash key

would contain the location ID, and the hash fields would represent attributes such as temperature, humidity, wind speed, precipitation, weather description, and timestamp.

Step 3:

The redis commands that you would use to interact with your specific Redis structures. Describe the commands for all your use cases. Make sure to include all CRUD operations

1. **Hourly Weather Forecast Storage**:
   - **Initialize**: There is no specific initialization that I'm adding over here because I would like to dynamically create hash objects for each hourly forecast.
   - CRUD Operations:
     1. If I would like to **set** the hourly weather forecast for location:
        - HSET weather_forecast:{location_id}:{timestamp} temperature {temperature_value} humidity {humidity_value} wind_speed {wind_speed_value} precipitation {precipitation_value} weather_description {weather_description}

     2. If I would like to **get** hourly weather forecast for location:
        - HGETALL weather_forecast:{location_id}:{timestamp}

     3. If I would like to **update** hourly weather forecast for location:
        - HSET weather_forecast:{location_id}:{timestamp} temperature {new_temperature_value}

     4. If I would like to **delete** hourly weather forecast for location:
        - DEL weather_forecast:{location_id}:{timestamp}

2. **Alerts Storage**
   - **Initialize**: There is no specific initialization that I will be adding here because I would like to dynamically create lists for each location's alert.
   - CRUD Operations:
     1. If I would like to **add** an alert for a location:
        - LPUSH alerts:{location_id} "{alert_details}"

2. If I would like to **get** the alerts for a specific location:
    - LRANGE alerts:{location_id} 0 -1

3. If I would like to **remove** an alert for the specific location:
    - LREM alerts:{location_id} 0 "{alert_to_remove}"

3. **User Preferences Storage**
- **Initialize**: There is no specific initialization that I will be adding here because I would like to dynamically create hash objects and sets for each user.
- CRUD Operations:
    1. If I would like to **set** User Profile:
        - HMSET user_profile:{user_id} username "{username}" email "{email}" password "{password}"

    2. If I would like to **get** User Profile:
        - HGETALL user_profile:{user_id}

    3. If I would like to **add** preferred location for user:
        - SADD user_preferences:locations:{user_id} {location_id}

    4. If I would like to **add** to Alert type for a user:
        - SADD user_preferences:alerts:{user_id} "{alert_type}"

    5. If I would like to **get** subscribed alert types for a user:
        - SMEMBERS user_preferences:alerts:{user_id}

4. **Current Weather Data Storage:**
- **Initialize**: There is no specific initialization that I will be adding here because I would like to dynamically create hash objects for each location's current weather data.
- CRUD Operations
    1. If I would like to **set** current weather data for a location:
        - HSET current_weather:{location_id} temperature {temperature_value} humidity {humidity_value} wind_speed {wind_speed_value} precipitation {precipitation_value} weather_description "{weather_description}"

    2. If I would like to **get** current weather for a location:
        - HGETALL current_weather:{location_id}

3. If I would like to **update** current weather for a location:
    - HSET current_weather:{location_id} temperature {new_temperature_value}

4. If I would like to **delete** current weather data for a location:
    - DEL current_weather:{location_id}