Design and Implementation

Topic: Weather Restful APIs
Design:
The application will be a spring-boot application (based on requirement). The application will be concerned with providing weather reports based on country and city. Thus, this would be a simple HTTP rest service which would have one endpoint (more can be added later on) and will be utilizing OpenWeatherMap api to fetch weather data and return it to the user.

- **Core Behavior**: Application will have one endpoint that takes a query parameter that includes city and country name and returns weather description of that location to the user.
- **Security**: Application will have its own API keys and it will only serve requests that provide valid API key. The keys will be simple string values and there will be only 5 valid API keys supported by the application. Each api key is rate limited and can be used only 5 times in one hour.
- **Storage**: For every city & country based location requested, we will be storing weather description in in-memory database.
- **Weather Data**: For every city & country location requested, if the weather description is already stored in H2 DB and it is not expired, we will return it. If it expires, we will fetch fresh weather data from OpenWeatherMap and update our H2 database followed by returning it to the user.
- **Error handling:** Invalid API keys, missing api keys, api key's rate limit exceeded, weather data not found, will be the main exceptions handled by the application.

Implementation:
1. The application will be distributed into layers: Controller, Service, Persistence (Repository & Model) using Spring-Web.

- **Controller layer**: This would only have one controller primarily; WeatherController. This will have one endpoint "/weather?q=city_name,country_name"
- **Service layer**: This will perform the core logic including api key validation(ApiKeyValidationService), rate limit checks(RateLimiterService) and fetch weather data(WeatherService).
- **Persistence layer**: This will use SpringJPA to store Weather data in the H2 database. We will have a WeatherRepository and a Weather Entity that stores weather descriptions.

2. For maintainability, we will be using interfaces for Service layers.
3. Rate limiting will be performed by RateLimiterService which will store 5 supported API Keys in a concurrent hash map and update their usage timings each time they are used. If an api key is used for 5 times in the last hour, we will throw an exception which then would reject the user request with appropriate response. Exceptions will be managed globally using @ControllerAdvice.
4. OpenWeatherMap URL and api key will be taken from application.properties and ENV variables.
5. Time window as well as total number of calls allowed in that window for rate limiting will also be taken from application.properties.