UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU | ing GROUP

4CS017 – Internet Software Architecture

Prototype 2

Student Name: Arjabi Shrestha

Student ID: 2408286

Table of Contents

1. Introduction

After building the foundation in the first prototype, this second iteration of the weather app has improved on the applications functionality with the addition of database. This iteration has improved by solving some of the weaknesses from the initial prototype by building a more robust application with the use of databases and PHP. With the use of backend technologies data management has greatly improved along with a more dynamic user experience.

2. Strengths

In addition to the advantages of the first prototype, this iteration introduces server-side caching. When a user searches a city's weather data, the application stores the data in the database, retrieves it displays the 7 days of data of that city. This makes it so that excess use of the OpenWeatherMap can be avoided.

The weather data from the API in the first prototype was fetched in JavaScript. This prototype mitigates this by fetching all the data through PHP which ensures that the data fetched is secure and reduces dependency from the API as well as reducing complexity on the client side. This increases the versatility and effectiveness of the application as it doesn't have to solely reply on the API to fetch data. If the API's server is down, or if in offline scenarios, the application can at least still show the data that have been previously entered.

Along with displaying 7 days of data, the application can also show real time weather data of the entered data. The data searched by the user is also fetched by the API, stores it in the database and displays the information.

3. Weaknesses

The weaknesses in the free plan of the OpenWeatherMap API still stand, i.e., restrictions on the number of requests within a specific timeframe and data updating only once every hour.

Any issues with the database, such as connection failures, could lead to loss of data or can prevent displaying the data.

When data is entered by the user it then stores it into the database. This leads redundant data storage.

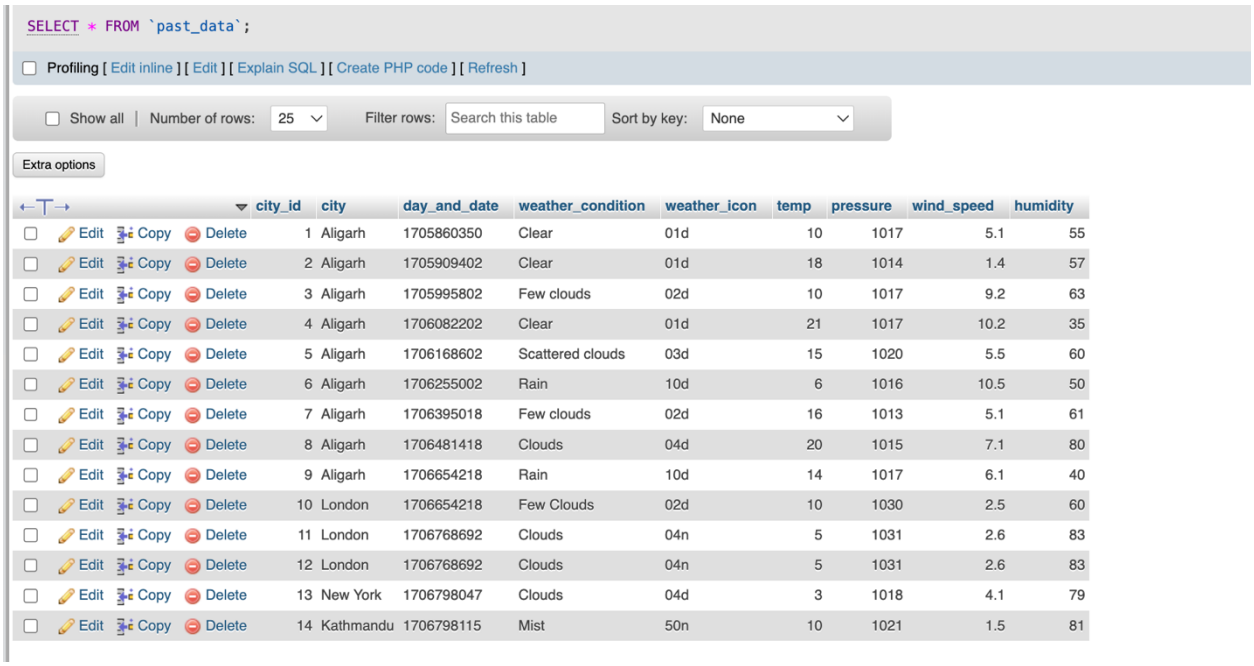
As more and more weather information is entered in the database, the volume of data increases with multiple cities and frequent entries. This could lead to increased database load and storage requirements. The application doesn’t provide a way to delete past data which are no longer of use.

Two PHP files had to be created to show 7 days of data and the latest data which makes it more complex and sometimes redundant data are also fetched.

Screenshot of MySQL table:

Database name: Weather_App

Table name: past_data



```
SELECT * FROM `past_data`;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	city_id	city	day_and_date	weather_condition	weather_icon	temp	pressure	wind_speed	humidity
Edit Copy Delete	1	Aligarh	1705860350	Clear	01d	10	1017	5.1	55
Edit Copy Delete	2	Aligarh	1705909402	Clear	01d	18	1014	1.4	57
Edit Copy Delete	3	Aligarh	1705995802	Few clouds	02d	10	1017	9.2	63
Edit Copy Delete	4	Aligarh	1706082202	Clear	01d	21	1017	10.2	35
Edit Copy Delete	5	Aligarh	1706168602	Scattered clouds	03d	15	1020	5.5	60
Edit Copy Delete	6	Aligarh	1706255002	Rain	10d	6	1016	10.5	50
Edit Copy Delete	7	Aligarh	1706395018	Few clouds	02d	16	1013	5.1	61
Edit Copy Delete	8	Aligarh	1706481418	Clouds	04d	20	1015	7.1	80
Edit Copy Delete	9	Aligarh	1706654218	Rain	10d	14	1017	6.1	40
Edit Copy Delete	10	London	1706654218	Few Clouds	02d	10	1030	2.5	60
Edit Copy Delete	11	London	1706768692	Clouds	04n	5	1031	2.6	83
Edit Copy Delete	12	London	1706768692	Clouds	04n	5	1031	2.6	83
Edit Copy Delete	13	New York	1706798047	Clouds	04d	3	1018	4.1	79
Edit Copy Delete	14	Kathmandu	1706798115	Mist	50n	10	1021	1.5	81