****

**INFORMATION AND COMMUNICATIONS UNIVERSITY**

**SCHOOL OF ENGINEERING**

**WEATHER APPLICATION DOCUMENTATION**

NAME: Mubita Haabasimbi Darious

SIN: 2402239941

PROGRAM: BSc information security and computer forensics

COURSE: Interactive web development

EMAIL: mubitadarious@gmail.com

PHONE NUMBER: 0961053660

GITHUB REPOSITORY: https://github.com/MUbita-darious/weather-app

APP URL: http://darioushaabasimbimubita.icu/

LECTURER: Mr Lameck Nsama

DUE DATE: 10th December 2024

Table of Contents

[**INTRODUCTION** 3](#_Toc183795585)

[**FUNCTIONAL REQUIREMENTS** 5](#_Toc183795586)

[**TECHNICAL ARCHITECTURE** 7](#_Toc183795587)

[**FRONTEND DOCUMENTATION** 11](#_Toc183795588)

[**BACKEND DOCUMENTATION** 14](#_Toc183795589)

[**AUTHENTICATION & AUTHORIZATION** 21](#_Toc183795590)

[**SECURITY & PERFORMANCE** 21](#_Toc183795591)

[**TESTING & QUALITY ASSURANCE** 22](#_Toc183795592)

[**MAINTENANCE & UPDATES** 22](#_Toc183795593)

[**CHALLENGES** 23](#_Toc183795594)

# **INTRODUCTION**

**Project Overview**

A weather application is a software tool or mobile app that provides users with weather-related information for specific locations. These applications gather data from weather stations, satellites, and other meteorological sources to deliver up-to-date details on current conditions, forecasts, and other weather-related factors. Weather apps are designed to help users make informed decisions based on weather patterns, whether they’re planning outdoor activities, traveling, or simply staying prepared for the day ahead.

**Target Audience**

This weather application is developed to provide real-time weather updates to users in Solwezi Province. However, the general public; pupils, students, employees, individuals seeking weather information for daily planning, travelers, outdoor enthusiasts can use this weather application.

**Technologies Used**

1. Front-end
2. HTML
3. CSS
4. Bootstrap
5. Fontawesome
6. Javascript
7. JQuery
8. AJAX
9. Backend
10. Javascript
11. JQuery
12. OpenWeatherMap API
13. Geolocation API
14. AJAX

c) Application Programming Interfaces Integrated

1. OpenWeatherMap API
2. Geolocation API
3. AJAX

d) Libraries And Frameworks

1. Flexbox
2. Bootstrap
3. Fontawesome

**Stakeholders**

1. Developer – Darious Haabasimbi Mubita
2. Residents of Solwezi Province
3. Lecturer

# **FUNCTIONAL REQUIREMENTS**

1. **Current Weather Data:**
   1. Temperature: Displays the current temperature in Celsius or Fahrenheit.
   2. Weather Conditions: Shows the type of weather (e.g., sunny, cloudy, rainy, snowing).
   3. Wind Speed and Direction: Displays current wind speed and its direction.
   4. Humidity: Shows the percentage of humidity in the air.
   5. Precipitation: Indicates any rainfall or snow, including intensity.
   6. UV Index: Provides the current UV index to help users take necessary precautions.
2. **Weather Forecast:**
   1. Hourly Forecast: Shows the temperature, weather conditions, wind speed, and other metrics for the next 12 to 24 hours.
   2. 7-Day Forecast: Displays daily weather predictions including temperature, weather conditions, and chance of precipitation.
   3. Air Quality Index (AQI): Displays the air quality, particularly for users in urban areas or places with pollution concerns.
3. **Location-based Services:**
   1. Current Location: The app automatically detects the user’s current location and shows the weather based on GPS data.
   2. Multiple Locations: Users can add multiple locations (e.g., home, work, travel destination) and view the weather for each location.
4. **User Preferences:**
   1. Unit Preferences: Users can toggle between Celsius/Fahrenheit for temperature and km/h/mph for wind speed.
   2. Language Support: No multi - language support
5. **Search Functionality**
   1. Current user location weather information
   2. Cross city weather information
   3. Province weather information
   4. International weather information

# **TECHNICAL ARCHITECTURE**

**Site Architecture:**

The site architecture displays a high-level architecture diagram for the weather application, which shows how all the different components interact in the application.

FRONT-END

MIDDLEWARE

(JavaScript, AJAX)

BACK-END

**Frontend Architecture**

The frontend architecture of an application is the structure and design of the client-side part of the application, everything that the user interacts with directly. It includes the organization of code, how components and features are structured, how data flows within the app, and how the user interface (UI) is built and maintained. Good frontend architecture helps ensure that the app is scalable, maintainable, and efficient.

BANNER SECTION

SECTION

SECTION

SECTION

**Backend Architecture**

The backend architecture of an app is crucial for its performance, scalability, security, and maintainability. It consists of components like servers, databases, APIs, business logic, and more. The choice of architecture monolithic, microservices, or serverless depends on the size and complexity of the app. A well-structured backend ensures that data flows smoothly, users receive a seamless experience, and the app can scale effectively to meet future demands.

OPEN WEATHER MAP DATABASE

AND

SERVER

AUTHENTICATION

EXTERNAL

APIs

7 DAY FORECAST SUBSCRIPTION

ONE CALL 3.0 SUBSCRIPTION

**OPEN WEATHER API**

GEOLOCATION API

CLIENT APP MIDDLEWARE

EXTERNAL

APIs

LIVE COORDINATES

**Open Weather Map Architecture**

EXTERNAL SERVICES

EXTERNAL APIs

ALL SUBSCRIPTIONS

7 DAY FORECAST SUBSCRIPTION

ONE CALL 3.0 SUBSCRIPTION

OAUTH 3.0

POST ROUTES

GET ROUTES

**OPEN WEATHER API**

# **FRONTEND DOCUMENTATION**

**HTML Conventions**

* Unique Id are assigned to HTML elements that have to be individually in CSS
* Class names are assigned to HTML elements that may share CSS rules
* Attributes - HTML element attributes are assigned in lowercase

**CSS Conventions**

* Class names are assigned using lowercase characters and hyphens
* Curly brackets are always introduces below id and class names
* Generic CSS rules are written and re-used

**Interactive Elements**

* Buttons and other elements that implement interactive behavior, the functionality is implemented in JQuery
* JQuery is used to push dynamic content from APIs

**Section Spacing**

* Each section has padding top and padding bottom of 10% or more
* Centered Content

**Headings**

* They are in uppercase
* They are bold or in a strong tag

**Fonts**

* Consistent font family and style is used consistently through the application

**Images, Background-Images And Logos**

* They are in png or jpeg formats
* They compressed for efficient loading

**Frontend Architectiure**

BANNER SECTION

SECTION

SECTION

SECTION

# **BACKEND DOCUMENTATION**

The backend of this this weather application uses a monolithic architecture, implemented using RESTFUL Application Programming Interfaces listed below;

**APIs Implemented**

1. **Open Weather Map API**

OpenWeatherMap is an online service, owned by OpenWeather Ltd, that provides global [weather](https://en.wikipedia.org/wiki/Weather) data via [API](https://en.wikipedia.org/wiki/API), including current weather data, [forecasts](https://en.wikipedia.org/wiki/Weather_forecasting), [nowcasts](https://en.wikipedia.org/wiki/Nowcasting_(meteorology)), and historical weather data. The company provides a minute-by-minute [hyperlocal](https://en.wikipedia.org/wiki/Hyperlocal) [precipitation](https://en.wikipedia.org/wiki/Precipitation) forecast. The [convolutional](https://en.wikipedia.org/wiki/Convolution) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) model uses meteorological broadcast services and data from [airport](https://en.wikipedia.org/wiki/Airport) [weather stations](https://en.wikipedia.org/wiki/Weather_station), [on-ground radar stations](https://en.wikipedia.org/wiki/Weather_radar), [weather satellites](https://en.wikipedia.org/wiki/Weather_satellite), [remote sensing](https://en.wikipedia.org/wiki/Remote_sensing) satellites, [METAR](https://en.wikipedia.org/wiki/METAR), and [automated weather stations](https://en.wikipedia.org/wiki/Automatic_weather_station).

1. **Geo-Location API**

With the Geolocation API, you can use geospatial data from cell towers and WiFi nodes to get the location of a device that does not have native geolocation or GPS.

**API Services**

1. **Open Weather Map**
2. One Call API 3.0

The subscription plan [**"One Call by Call"**](https://openweathermap.org/price) and gives access to the various data for any coordinates**. W**[**ith One Call API 3.0**](https://openweathermap.org/api/one-call-3) 1,000 API calls per day for free! It’s [Pay as you call](https://openweathermap.org/price).

1. Current Weather Data API

Access current weather data for any location

We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations

JSON, XML, and HTML formats

Included in both free and paid subscriptions

1. 7 Days Weather Forecast
2. **Geolocation**
3. Current latitude and longitude finder

**API Routes And Endpoints**

**Search Route**

**https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon={lon}&exclude={part}&appid=**[**{API key}**](https://home.openweathermap.org/api_keys)

Example Of API Response

**{**

**"lat":33.44,**

**"lon":-94.04,**

**"timezone":"America/Chicago",**

**"timezone\_offset":-18000,**

**"current":{**

**"dt":1684929490,**

**"sunrise":1684926645,**

**"sunset":1684977332,**

**"temp":292.55,**

**"feels\_like":292.87,**

**"pressure":1014,**

**"humidity":89,**

**"dew\_point":290.69,**

**"uvi":0.16,**

**"clouds":53,**

**"visibility":10000,**

**"wind\_speed":3.13,**

**"wind\_deg":93,**

**"wind\_gust":6.71,**

**"weather":[**

**{**

**"id":803,**

**"main":"Clouds",**

**"description":"broken clouds",**

**"icon":"04d"**

**}**

**]**

**},**

**"minutely":[**

**{**

**"dt":1684929540,**

**"precipitation":0**

**},**

**...**

**],**

**"hourly":[**

**{**

**"dt":1684926000,**

**"temp":292.01,**

**"feels\_like":292.33,**

**"pressure":1014,**

**"humidity":91,**

**"dew\_point":290.51,**

**"uvi":0,**

**"clouds":54,**

**"visibility":10000,**

**"wind\_speed":2.58,**

**"wind\_deg":86,**

**"wind\_gust":5.88,**

**"weather":[**

**{**

**"id":803,**

**"main":"Clouds",**

**"description":"broken clouds",**

**"icon":"04n"**

**}**

**],**

**"pop":0.15**

**},**

**...**

**],**

**"daily":[**

**{**

**"dt":1684951200,**

**"sunrise":1684926645,**

**"sunset":1684977332,**

**"moonrise":1684941060,**

**"moonset":1684905480,**

**"moon\_phase":0.16,**

**"summary":"Expect a day of partly cloudy with rain",**

**"temp":{**

**"day":299.03,**

**"min":290.69,**

**"max":300.35,**

**"night":291.45,**

**"eve":297.51,**

**"morn":292.55**

**},**

**"feels\_like":{**

**"day":299.21,**

**"night":291.37,**

**"eve":297.86,**

**"morn":292.87**

**},**

**"pressure":1016,**

**"humidity":59,**

**"dew\_point":290.48,**

**"wind\_speed":3.98,**

**"wind\_deg":76,**

**"wind\_gust":8.92,**

**"weather":[**

**{**

**"id":500,**

**"main":"Rain",**

**"description":"light rain",**

**"icon":"10d"**

**}**

**],**

**"clouds":92,**

**"pop":0.47,**

**"rain":0.15,**

**"uvi":9.23**

**},**

**...**

**],**

**"alerts": [**

**{**

**"sender\_name": "NWS Philadelphia - Mount Holly (New Jersey, Delaware, Southeastern Pennsylvania)",**

**"event": "Small Craft Advisory",**

**"start": 1684952747,**

**"end": 1684988747,**

**"description": "...SMALL CRAFT ADVISORY REMAINS IN EFFECT FROM 5 PM THIS\nAFTERNOON TO 3 AM EST FRIDAY...\n\* WHAT...North winds 15 to 20 kt with gusts up to 25 kt and seas\n3 to 5 ft expected.\n\* WHERE...Coastal waters from Little Egg Inlet to Great Egg\nInlet NJ out 20 nm, Coastal waters from Great Egg Inlet to\nCape May NJ out 20 nm and Coastal waters from Manasquan Inlet\nto Little Egg Inlet NJ out 20 nm.\n\* WHEN...From 5 PM this afternoon to 3 AM EST Friday.\n\* IMPACTS...Conditions will be hazardous to small craft.",**

**"tags": [**

**]**

**},**

**...**

**]**

**Current Weather Data Route**

**https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid=**[**{API key}**](https://home.openweathermap.org/api_keys)

**Response example**

**{**

**"coord": {**

**"lon": 7.367,**

**"lat": 45.133**

**},**

**"weather": [**

**{**

**"id": 501,**

**"main": "Rain",**

**"description": "moderate rain",**

**"icon": "10d"**

**}**

**],**

**"base": "stations",**

**"main": {**

**"temp": 284.2,**

**"feels\_like": 282.93,**

**"temp\_min": 283.06,**

**"temp\_max": 286.82,**

**"pressure": 1021,**

**"humidity": 60,**

**"sea\_level": 1021,**

**"grnd\_level": 910**

**},**

**"visibility": 10000,**

**"wind": {**

**"speed": 4.09,**

**"deg": 121,**

**"gust": 3.47**

**},**

**"rain": {**

**"1h": 2.73**

**},**

**"clouds": {**

**"all": 83**

**},**

**"dt": 1726660758,**

**"sys": {**

**"type": 1,**

**"id": 6736,**

**"country": "IT",**

**"sunrise": 1726636384,**

**"sunset": 1726680975**

**},**

**"timezone": 7200,**

**"id": 3165523,**

**"name": "Province of Turin",**

**"cod": 200**

**}**

# **AUTHENTICATION & AUTHORIZATION**

The Open Weather Map API uses the following for authentication;

1. Registration And Login System
2. Email Authentication
3. OAuth
4. API Encryption
5. API Key Authentication Per Request

# **SECURITY & PERFORMANCE**

SSL, encryption, input validation, CSRF protection are implemented by the client application or the API for security.

For performance optimization, caching strategies, image optimization, lazy loading are implemented by the API.

# **TESTING & QUALITY ASSURANCE**

The following tests were conducted to ascertain the quality, efficiency and zero-errors in the software product.

1. Unit testing
2. Integrated testing
3. Function testing
4. Acceptance testing

In-house and external tests were conducted, both online and offline.

# **MAINTENANCE & UPDATES**

Regular maintenance and updates are supposed to be made on a weekly basis. The API integrated into the weather application provide real-time data, thus, errors may occur due to the following;

1. Changes to the API keys by the API provider
2. Changes to API response JSON
3. Change in naming standards
4. API may temporarily stop working
5. Network failure
6. Unhandled errors flagged by the API
7. Patching security vulnerabilities
8. Updating dependencies

# **CHALLENGES**

1. Network failure - there has been continued network failure while developing the application, which led to authentication and authorization errors, failed requests and responses to and from the API
2. Finding the API - there are plenty of weather APIs on the market, therefore, identifying a well functional API with all the features needed for the application was a challenge. However, I tested a few and finally decided to use openWeatherMap as it was convenient to utilize and it had all the features required in one place.
3. Reading through documentation - the openWeatherAPI has different packages tailored for different purposes, in most cases with similar features. Thus, the documentation is vast and has a lot of information to read through. But with consistency, I managed to find the correct information and made a successful integration.
4. API Key errors - API keys in the first place were failing to synchronize with the API. But after a while they synched.
5. Error handling - there are many errors to catch and handle, thus, I wrote handlers for the necessary errors
6. Identifying subscriptions and API routes - Out of many subscriptions and routes on the API. I tested and tried most, and obtained the useful ones.
7. Deployment - network failure, but i managed to submit the project to both github and the school servers