**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | **Cross-platform Development** |
| **Assessment Title:** | **Analysis of Information Systems** |
| **Lecturer Name:** | **David Gonzalez** |
| **Student Full Name:** | **Alejo Santos, Telmuun Dunia, Luis Ramirez** |
| **Student Number:** | **2023197, 2023306, 2023169** |
| **Assessment Due Date:** | **TBC** |
| **Date of Submission:** |  |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Contents

[Introduction. 3](#_Toc198555125)

[Weather app Alutech. 4](#_Toc198555126)

[Requirements. 4](#_Toc198555127)

[1. Installation steps. 5](#_Toc198555128)

[2. Usage. 5](#_Toc198555129)

[3. Apis used. 6](#_Toc198555130)

[Challenges Encountered and Solutions Implemented. 6](#_Toc198555131)

[Reflections on the Development Process. 7](#_Toc198555132)

[Potential Future Improvements. 7](#_Toc198555133)

**Technical report.**

# Introduction.

This project is a modern, responsive, and user-friendly weather application developed using the React framework. Designed with simplicity and efficiency in mind, the app offers a clean and intuitive interface for users to check real-time weather conditions in any city around the world.

The core functionality includes fetching live weather data from a public API, displaying key information such as temperature, weather conditions, humidity, wind speed, and a 5-day forecast. Users can search for any location, with results updating dynamically based on their input.

To enhance the overall user experience, the app also includes features such as location-based weather detection, animated weather icons, and customizable theme options (e.g., light/dark mode). Altogether, this project showcases a balance between functionality and design, demonstrating how modern front-end technologies like React can be used to build practical and visually engaging applications.

# Weather app Alutech.

This weather app was designed to be a simple and user-friendly application.

## Requirements.

The basic requirements to run this project are the same as every React project, for example:

Operating System:

* Windows
* macOS
* Linux

Node.js

* v14 or higher

Dependencies

These are typical dependencies required to run the app

* React – Front-end framework (react, react-dom, react-scripts)
* Axios – For making HTTP requests to the weather API
* Weather API – Example: OpenWeatherMap API key (you’ll need to register and get your own key)
* dotenv (optional) – For managing API keys securely

Once you have completed all these requirements, you are ready to download the application from GitHub or follow these steps.

## Installation steps.

1. Clone repository:

<https://github.com/TelmuunDn/weather-app-ca2-aoe.git>

1. Install dependencies:

npm install

1. install Expo CLI(if not installed)

npm install -g expo-cli

1. Install required Expo packages

npx expo install expo-location expo-font expo-linear-gradient

1. Install additional packages

npm install @react-native-async-storage/async-storage

npm install lodash

npm install react-native-vector-icons

1. Start the Expo app

npx expo start –tunnel

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Open the QR code in the Expo Go app on your phone or run it on an emulator.

## Usage.

Here is a brief explanation of some features that you will find inside this application:

1. The application is built in tabs: the Home tab, Forecast, and Location.
2. Every tab contains different information.
3. On the Home tab, you will see the current temperature in your location, and you can change it between Fahrenheit and Celsius.
4. Forecast tab, you will see the next five days' forecast with the date, max and low temperature, and the probability of rain.
5. On the Location tab, you will find a search bar to check the temperature in the place that you desire.

## Apis used.

 IPAPI (for location by IP)

* Fetches current city/country by user's public IP
* Fallback to this method when GPS is unavailable

🌦 Meteomatics API

* Primary source for weather data
* Requires API key (base64 encoded username:password)
* Set your credentials directly in the Authorization header in the request

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Authorization: "Basic " + btoa("your\_username:your\_password")

## Challenges Encountered and Solutions Implemented.

Integrating a Weather API

Solution Implemented:

* Researched and selected a suitable weather API (e.g., OpenWeatherMap, WeatherAPI).
* Registered for an API key and handled authentication.

Handling User Input (City Search)

Solution Implemented:

* Created a search bar with controlled components (useState) for city input.
* On form submission, the app triggered a new fetch for the specified city’s weather data.

Displaying Weather Data Nicely

Solution Implemented:

* Parsed JSON responses and formatted dates, temperatures, and icons.
* Used conditionals to show different icons or backgrounds based on weather type.

## Reflections on the Development Process.

Learning how React works and applying it to a real-world weather app project was a great learning experience. It was rewarding to see how everything we studied — like components, hooks, and API calls — came together to build something useful.

Using GitHub for version control made it easier to keep track of changes and work as a team. We also had regular discussions about how to improve the app and fix bugs, which helped us stay organized and move forward smoothly.

We gained hands-on experience with building a full React app, using external APIs (like a weather API), and managing state with hooks like useState and useEffect. It also helped us better understand error handling, async data fetching, and how to make an interface that updates based on real-time data.

One thing we could have done differently was planning the app’s structure more clearly from the beginning. We didn’t have a full idea of what the final product should look like, so we had to go back and change things a few times. With better early planning, we could have avoided some of the refactoring and feature changes later on.

# Potential Future Improvements.

Add Dynamic Backgrounds or Icons

Change the background or weather icons based on conditions (e.g., sunny, rainy, snow).

Why? Makes the app more visually engaging and informative.

Improve Mobile Responsiveness

Make sure the layout looks clean and functions smoothly on all screen sizes.

Add Timezone and Local Time Display

Show the local time of the searched city alongside weather data.

Support for Multiple Languages

Add a language selection feature using i18n libraries (like react-i18next).

Weather Alerts or Notifications

Show warnings for extreme weather like storms, heat waves, etc.

Save Favorite Locations

Let users bookmark cities they frequently check.