COSC 310 Software Engineering

MILESTONE 2
Project Description & Requirements

Due February 16, 2024

gitGoblins (L01)

Eric Harrison
Ian Steyn
Joshua Ward
Chase Winslow

Links:
[GitHub Repo]
[GitHub Dashboard]

PROJECT DESCRIPTION

- **Project Type:** IoT Sensor/Monitor Dashboard
- **Dataset:** Rain in Australia

We are developing a web-based dashboard for Australian weather data. The dataset we are using contains meteorological sensor data from hundreds of weather stations all over Australia, measured between 2007 and 2017. It includes data for rainfall, sunshine, temperature, wind, clouds, humidity and pressure.

This is a student project built for the purpose of learning, and as such our application only “pretends” the data is live, by allowing users to select a date and location, and then giving them “current” information for their selected location. This information is displayed in various interactive forms: a general weather summary, a map with graphic overlays, and two graphs which compare past and current data. The app also predicts whether it will rain the next day based on current weather conditions.

To access the app, users must register with our login system. They can then also receive optional email alerts about severe weather conditions in their location.

USER REQUIREMENTS **(what will users be able to do with the system?)**

- **Use Dashboard.** Users can access different parts of the system from a dashboard which contains the three visualization modules, the summary, and the location search bar.
- **Select Location.** Users can use the map or search bar to select any location in Australia, to then view weather data from the closest available weather station.
- **See Data Summary.** Users can access a summary of the most recent weather data for the selected location.
    - **See Rain Forecast.** The summary includes a prediction of whether it will rain the next day.
- **Visualize Data.** Users can visualize current and past weather data.
    - **Map.** Users can choose from different graphic overlays on an interactive map.
    - **Graph 1: Compare Past Data.** Users can select a single location to see a graph of its weather data over time.
    - **Graph 2: Compare Current Data.** Users can select multiple locations to see a graph comparing the current weather data of those locations.
- **Filter Data.** For the daily summary and all three visualizations, users can choose which types of weather data they want to see (e.g. rain, humidity, or wind).
- **Login & Register.** Users must register with a valid email address to be able to log in to the system.
- **Receive Alerts.** Users can sign up to receive email alerts warning them of current severe weather conditions in their location.

SYSTEM REQUIREMENTS

Functional Requirements (what will the system do to support users?)

- **Dashboard.** Acts as a home/start page for users. Provides access to all other parts of the system: data summary, visualizations, and location search.
- **Location Selection.** There are two location selection interfaces: the search bar, or clicking on a point on the map. Data is provided for the closest weather station to the selected location.
- **Date Selection.** Since we are not actually using live IoT data, the system must provide a way for the user to select the "current" date. The system pretends that the data is live.
- **Data Summary.** This contains an easy-to-read overview of the most current available weather data for the selected location, including temperature, rainfall, humidity, wind direction & speed, pressure and cloud cover.
  - **Rain forecast.** The system runs a machine learning algorithm on past weather data to predict whether it will rain the next day.
- **Visualizations.**
  - **Map.** The system provides functionality to zoom in and out, select locations, and choose from different graphic overlays on the interactive map. The overlays are icons attached to each location, with some attribute (colour, number, shape etc.) that tells you something about the data.
  - **Graph 1: Compare Past Data.** See user requirements.
  - **Graph 2: Compare Current Data.** See user requirements.
- **Filtering Data.** See user requirements.
- **Login & Registration System.** The system provides a way for users to register a profile with their email. This profile must be used to access the system.
- **Alert System.** Email notifications regarding current severe weather conditions or system issues are sent to users who have opted in to the alert system.
- **Database.** The system will get data from the dataset specified in our description, but this will be implemented using a database.
- **Administrator login.** There is a special administrator login. An administrator can:
  - Manually modify faulty data.
  - Manually send notifications through the alert system, regarding regional system issues.

Non-functional Requirements (Constraints)

Product Requirements

- **Web App.** The system must be a web-application.
- **Flexible Interface.** The user interface must be flexible (i.e. it must be usable on different devices and window sizes).
- **Good Performance.** Search results should be returned within a maximum of 3 seconds.

Organizational Requirements

- **Programming Tools.** The system must be developed using Python and Python-adjacent libraries and APIs. This is because TA/instructor support is available for these tools.
- **Development Tools.** The system must be developed with appropriate reliance on the following team-based tools: Git, Github, and Google Drive.
- **Due Dates.** The various components of the system must be completed by the milestones specified in COSC 310, with the final due date being April 12.

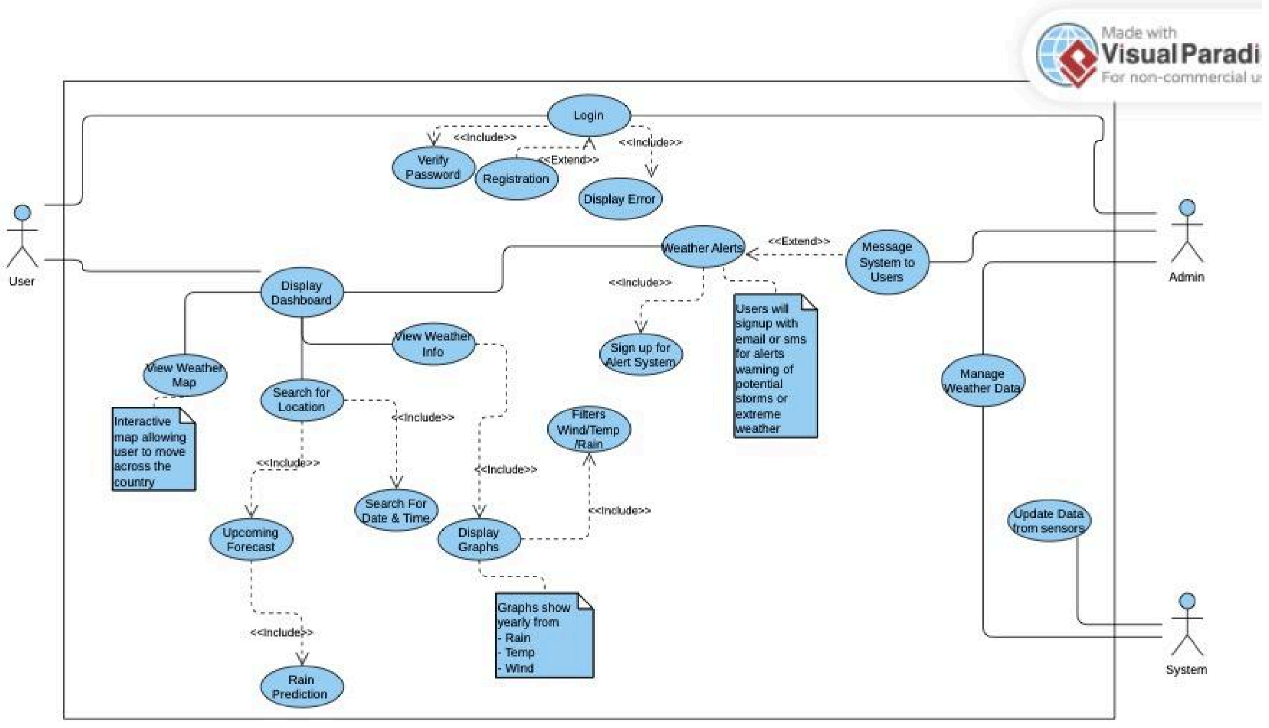
Miscellaneous

- **User Information.** The system will not collect any user information, except for their email for the notification system.
- **Data limitations.** The system is limited to Australian weather data from 2007-2017.

Domain Requirements (Standards)

- **Accessibility.** The system will implement at least basic web accessibility features, as specified in WCAG2 (Web Content Accessibility Guidelines). Most relevant:
    - 1. Perceivable. Especially:
        - 1.1 Text alternatives.
        - 1.4 Distinguishable.
    - 2.4 Navigable.

USE CASE DIAGRAM


User
Admin
System
Login
<<Include>>
Verify Password
Registration
<<Extend>>
<<Include>>
Display Error
Weather Alerts
<<Extend>>
Message System to Users
<<Include>>
Sign up for Alert System
Users will signup with email or sms for alerts warning of potential storms or extreme weather
Display Dashboard
View Weather Info
View Weather Map
Interactive map allowing user to move across the country
Search for Location
<<Include>>
<<Include>>
Search For Date & Time
Upcoming Forecast
<<Include>>
Rain Prediction
<<Include>>
Display Graphs
Graphs show yearly from
- Rain
- Temp
- Wind
<<Include>>
Filters Wind/Temp /Rain
Manage Weather Data
Update Data from sensors