Weather Adventures
Software Design Specification

Melodie Collins, Joey Le, Peter Nelson, Angela Pelky, Alexa Roskowski - 03/12/2023

Table of Contents

1. SDS Revision History

Date	Author	Description
02/15/2023	MC	Created the SDS document.
02/19/2023	MC, AR	Turned in Basic summary of SDS.
02/23/2023	MC, AP, AR	Initial Version Finished.
02/24/2023	JL, PN	Edited and Peer Reviewed.
02/26/2023	All	Turned In Second Version for Initial Project Plan Assignment on Canvas.
03/12/2023	All	Turned In Final Version for Project 2 Final Project Assignment on Canvas.

2. System Overview

Many times, especially in Oregon, your outdoor adventures get interrupted from the weather. As such, we decided to make a desktop application called Weather Adventures to help solve this problem. The services provided by the system Weather Adventures include allowing a user to click on any area in Lane County or Eugene and to get more detailed information about the weather conditions in that specified area. The system will also allow a user to filter for certain weather conditions and get activity recommendations based on these conditions.

The system is divided into four main components including a desktop user interface, a main Weather Adventure module, an activity recommender module, and a Weather API. The Weather API will collect different information from Oregon weather stations, such as temperature and precipitation. The weather information will be useful to allow users to make well-informed decisions regarding outdoor areas to explore. The Activity Recommender module will take the mapped data from the main module and, using pre-specified conditions according to different weather circumstances, will store activity recommendations for the user. Finally, it will sort these activities based on distance from the grid coordinates the user clicked and it will return the nearest 3 activities. The Main Weather Adventure module will serve as a key module for communication between backend data logic parsing and the frontend desktop user interface. The desktop user interface module will display all data onto a roadmap of either Lane County or the city of Eugene that a user will be able to scroll over, zoom in, zoom out, select a location for more weather information, and select weather

filters. Furthermore, the user will be able to choose to get activity recommendations based on their location, or locations recommended for their desired activity.

3. Software Architecture

Weather Adventures has four main components as seen in Figure 3.1:

1. Desktop User Interface - An image window that displays a roadmap and possible user inputs including: weather filters, a location selector, an activity recommender, and a locations recommender.
2. Weather Adventure Main - Controls and manages the system.
3. Activity Recommender - Module that takes mapped data and stores activity recommendations using pre-specified conditions according to different weather circumstances.
4. Weather API - Module that collects different information from Oregon weather stations.

We decided to implement this program as a desktop application instead of a mobile app because of our time limit and our group's limited knowledge on programming mobile apps. This will allow us to have enough time to concentrate on more intricate details and features of the program while keeping the system accessible, easy-to-use, and easy-to-learn.

The system uses an API because we decided it is more important to spend our time implementing the analysis on the implementations of the weather instead of worrying about making a map or collecting up to date weather information.

The grid logic is used to differentiate areas on the map, so we will have more in depth information on each specific spot.

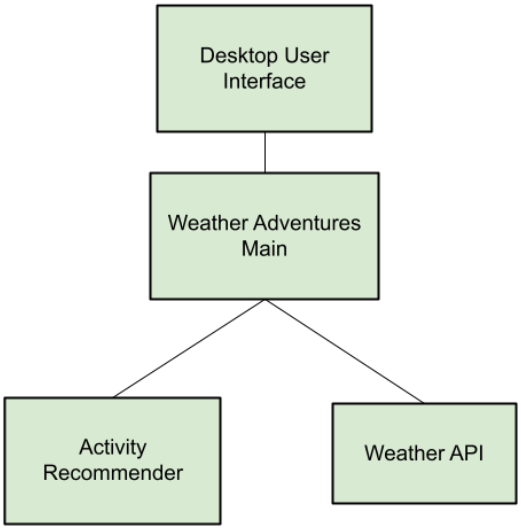

Figure 3.1: System Architecture of Weather Adventures

4. Software Modules

4.1. Desktop User Interface

a. **Module's role and primary function:** The Desktop User Interface module is an image window that displays a roadmap. The image window has clear and intuitive buttons, input fields, and filters for the user to interact with.

b. **Interface to other modules:** The roadmap is displayed with clear markings for different locations and landmarks. This module communicates with the Weather Adventures Main module.

c. A state diagram can be seen in Figure 4.1

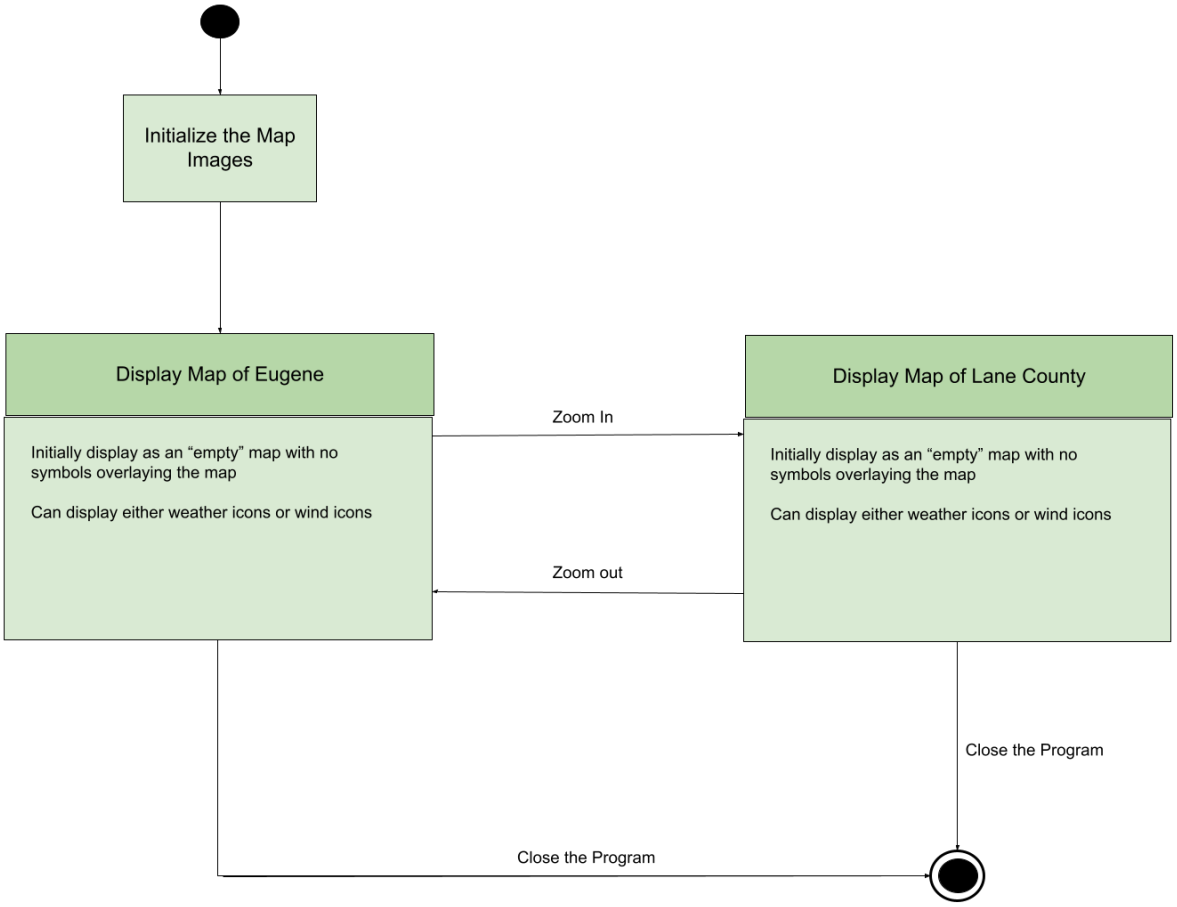

Figure 4.1: State Diagram of the Desktop User Interface Module

e. **Design rationale:** When the user opens the application a default map of Eugene will be displayed. The user will have to choose to either zoom in/out of the image, or to click on a specific spot on the grid. Based on these choices, either a new map image will be displayed or a pop-up of activity recommendations and weather data will be displayed. The user should be able to do either of these two actions until they want to close the application. Therefore, there is a branch added in the graph where the user either continues using the application and loops back to choosing between zooming in/out and clicking on the grid, or closes the application.

4.2. Weather Adventures Main

a. **Module’s role and primary function:** The Weather Adventures Main module controls and manages the system. It has clear indicators for system status, errors, and user feedback.
b. **Interface to other modules:** The main control panel has access to all the modules and functionalities of the system. It has clear indicators for system status, errors, and user feedback.
c. Please reference Figure 6.1 for Weather Adventures Main dynamic, sequence UML diagram.

4.3. Activity Recommender

a. **Module’s role and primary function:** The Activity Recommender module takes mapped data and stores activity recommendations using clear criteria and algorithms for matching weather conditions with activities. The module has a database to store and retrieve different activity recommendations.

b. **Interface to other modules:** It communicates with the Weather Adventures Main module, which stores data received from the Weather API module to a specific grid location.
c. An object diagram of the Activity Recommender can be seen in Figure 4.2 describing the structure of an Activity Object

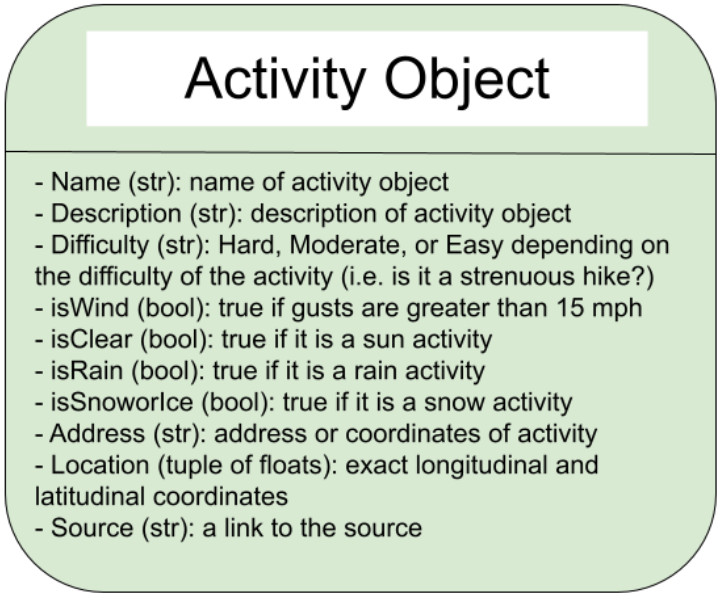

Figure 4.2: A static, object-oriented database UML model to describe an activity object

d. An activity diagram of the Activity Recommender module can be seen in Figure 4.3.

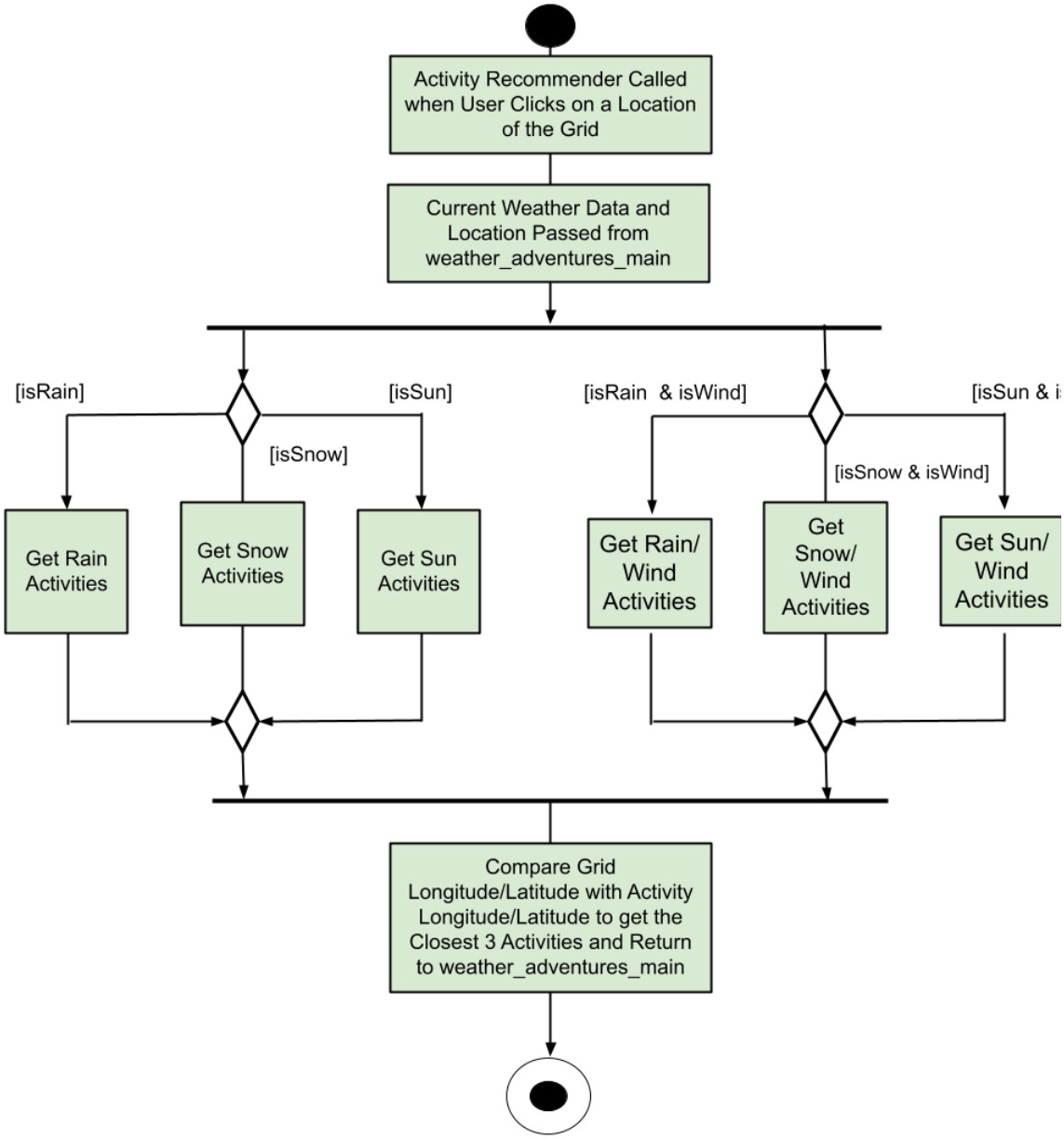

Figure 4.3: A dynamic, activity UML diagram to describe the activity recommender

e. **Design rationale:** When a user clicks on a location on the map, our system functionality will recommend activities based on current weather conditions. Hence, we will need to call the Weather API module to get the current weather data. Then, depending on the data received, we will base our activity recommendations based on whether it is windy, raining, snowing, or sunny. Lastly, we will return the nearest 3 recommended activities.

4.4. Weather API

a. **Module's role and primary function:** The Weather API module collects different information from Oregon weather stations.

b. **Interface to other modules:** The module has a clear API for accessing and processing weather data from different sources. It has a reliable and accurate data pipeline for updating the weather information in real-time. This module interacts with the Weather Adventures Main module.

c. A use case diagram of the Weather API module can be seen in Figure 4.4.

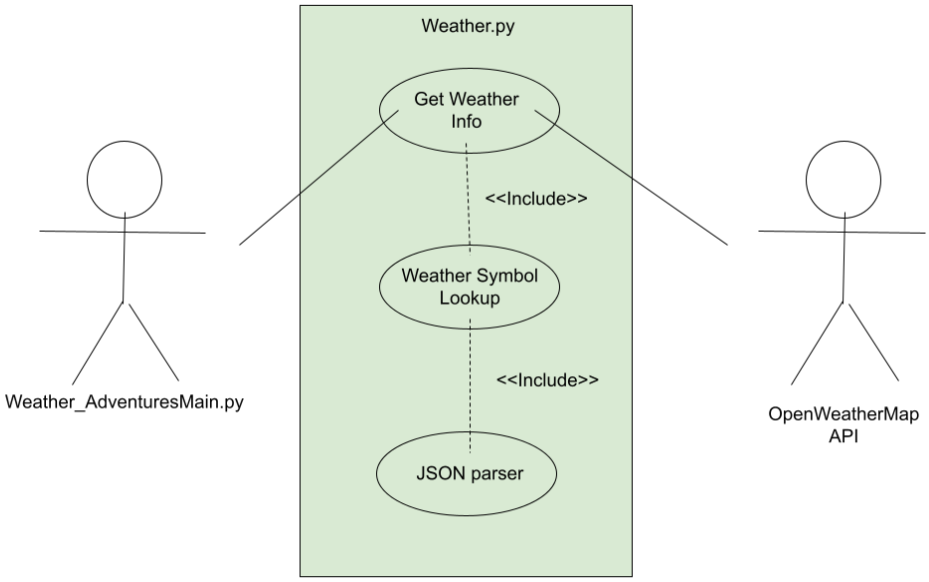

Figure 4.4: Use Case Diagram of the Weather API module

d. A sequence diagram of the Weather API module can be seen in Figure 4.5.

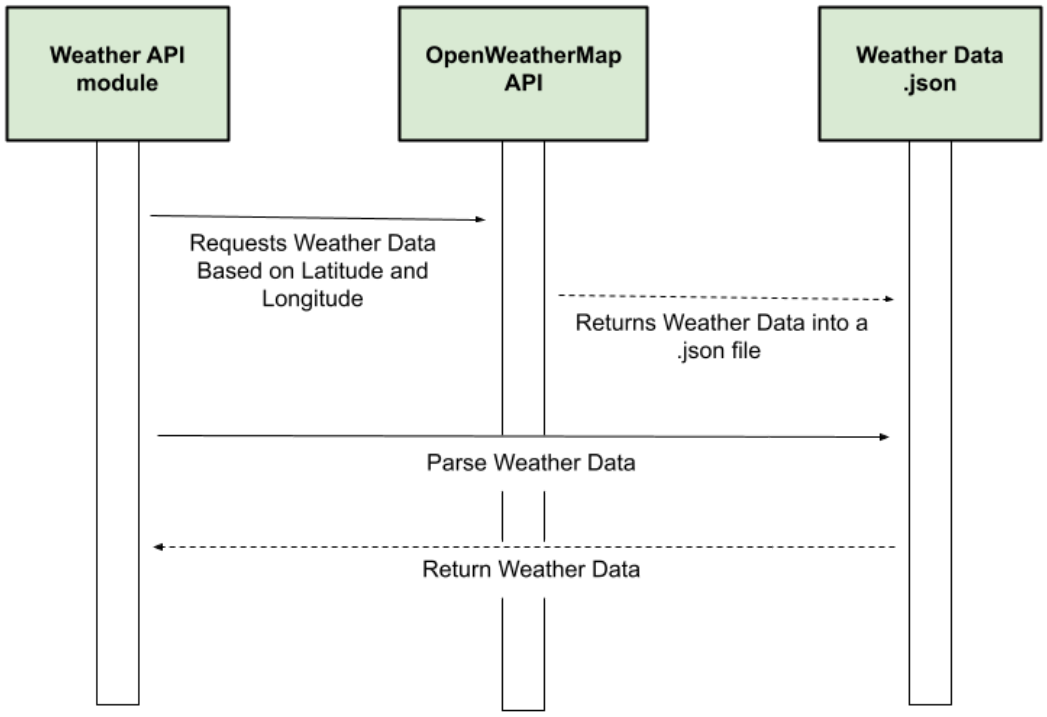

Figure 4.5: Sequence Diagram of the Weather API module

e. **Design rationale:** In order to work with the weather data received from the OpenWeatherMap API, we need to parse the .json file that it returns. The weather API module will then hold this parsed weather data for other modules to pull from.

5. Alternate Designs

While planning how to go about making the Weather Adventure application originally, we were not going to have the Weather Adventure Main module, nor Desktop User Interface as seen in Figure 5.1. Without the Weather Main Module, we found that the original Weather Adventure User Interface was doing too many different actions, so we decided to divide it up into Weather Main Module and Desktop User Interface, as in Figure 3.1.

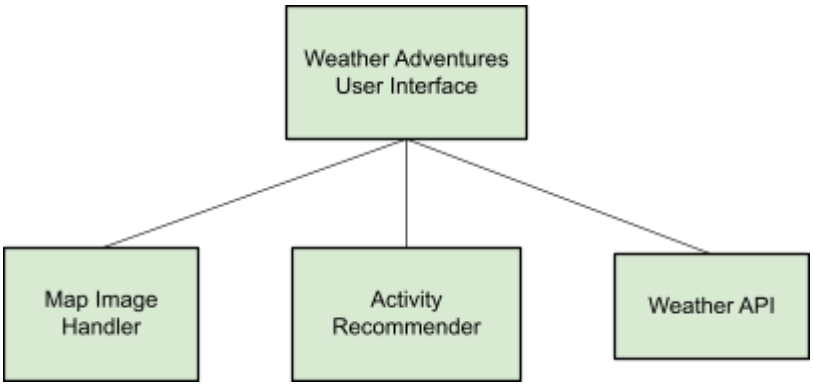

Figure 5.1: Version 1 of System Architecture for Weather Adventures

In our second version of the System Architecture, which can be seen in Figure 5.2, we were going to have the Activity Recommender interact between the other modules,and we were still thinking of using an API for the maps. We quickly found that the calls to the Weather API took a bit of time, so we wanted to limit the amount of times we did that. As such when we initialize the map we store all the necessary weather information to be able to reference later. We also deleted a module called Grid Logic, since the GUI was able to make buttons with overlaid images, so another module was unnecessary.

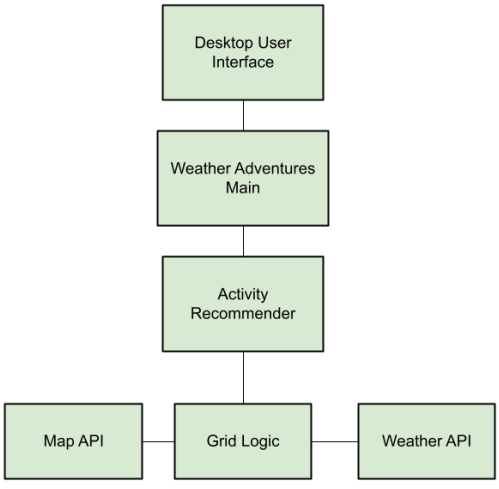

Figure 5.2: Version 2 of System Architecture for Weather Adventures

6. Dynamic Models of Operational Scenarios

Our application has one main use case, when the user opens the app looking for activity recommendations based on the local weather for the area. Here the application will create a map and overlay the map with a grid of symbolic representations of the current weather in the area as soon as the user opens the application. Then, if the user wants more information or an activity recommendation, they can click the area and we will return a pop-up with the desired information. This information is also reflected in the sequence diagram of Figure 6.1.

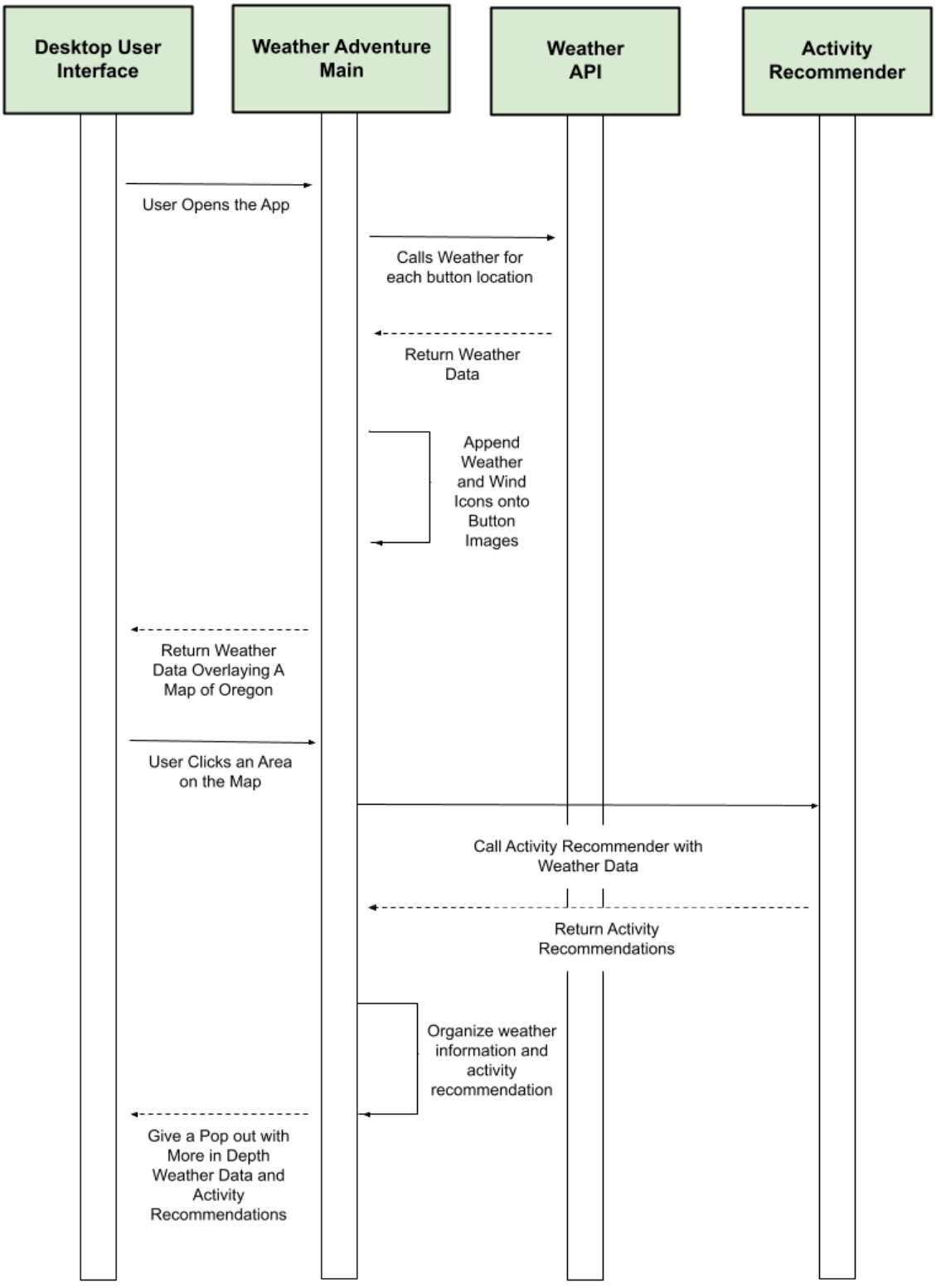

Figure 6.1: Sequence Diagram of Normal Use Case

7. Acknowledgements

This template builds on a template produced by Professor Anthony Hornof in CS 422 Winter Term 2023.

The diagrams follow UML diagram syntax provided by the tenth and global edition of *Software Engineering* by Ian Sommerville.