Software Requirements Specification

for

Weather App

Version 1.1 approved

Prepared by Alexandros Papadopoulos, 2654

Aristotle University of Thessaloniki

6/2/2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Glossary additions	6/2/2022	Added glossary	1.1

1. Introduction

1.1 Purpose

Weather app is meant to show you the weather broadcast for the same and the following day about a city you choose. It's meant to be fast and easy to use, bypassing any unnecessary steps that may delay the user's experience.

1.2 Document Conventions

This document uses the following conventions

Admin	A person that has higher level access, who maintains the site and its records
End user	A person using the site to check weather
Weather API	The *OpenWeatherMap* service which provides us with the weather broadcast information
Alternative API	Service to be used when Weather API is unavailable, TBD

1.3 Intended Audience and Reading Suggestions

This project is a prototype of "Weather App". It is useful for the analyst(s) and programmer(s) that will write and subsequently implement the requirements stated on this file, as well as to people that may happen to join the team midway development.

1.4 Product Scope

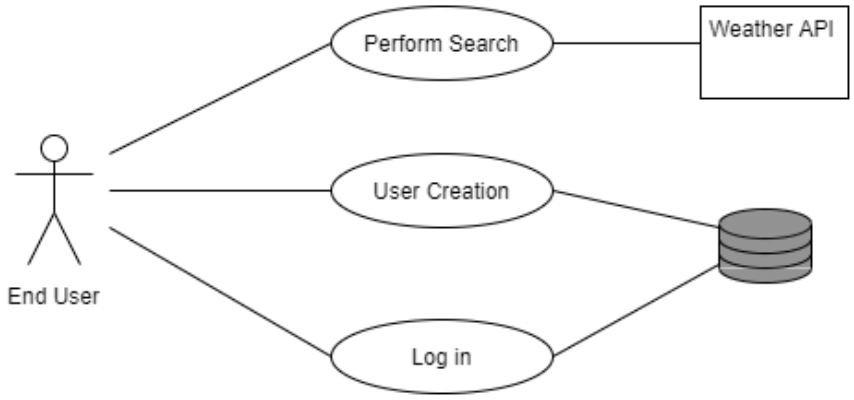
The purpose of the Weather App project is to create a free, easy, and fast to use tool to check details about the weather broadcast. The project is based on a document database that holds user records along with their saved cities.

1.5 References

- https://github.com/dipankardas011/Weather-App#readme

2. Overall Description

Below are two diagrams showing the main use case for the project



*Functions and characteristics marked with * will come in later of waves of deployment after the basic structure and functionality of the app has been created and deployed*

2.1 Product Perspective

- User
  It includes only necessary information on the logged in user like email and username

- City
  Cities can be saved for faster access from a logged in user

2.2 Product Functions

- Weather check
  After choosing a city the end user can see details regarding the weather broadcast for the same and the following day

- User Creation (*)
  The user should be able to create an account on the site, providing only necessary details for their account.

- Daily Report (*)
  The end user can choose to receive daily report about the weather of specific cities in email or push notification form

- Urgent Notification (*)

The end user can opt-in for notifications about sudden changes to the weather broadcast of the cities they choose, in push notification form

2.3 User Classes and Characteristics

The system will support two types of user privileges, End User, and Admin.
End users have access to main functions, and Admins will have access to main and admin functions.

The End User should be able to do the following:
- Check the weather broadcast for an inputted city (necessary)
- Check the weather broadcast from a saved city (necessary) (*)
- Create account (important) (*)
- Opt-in for daily reports (important) (*)
- Opt-in for urgent notifications (important) (*)

The Admin user should be able to do the following:
- View all users and their details (necessary)
- Resolve tickets (important)

2.4 Operating Environment

- Web app, runs on browser
    - Chromium based (Chrome, Opera, Edge)
    - Firefox
- Database: Mongo (*)

2.5 Design and Implementation Constraints

- Browser apps can burn through hardware capabilities fast. The app needs to be lightweight.

2.6 User Documentation

For end users a simple how to and FAQ will be provided on the website containing information on how to use the site.

2.7 Assumptions and Dependencies

- **Weather API:** To fetch all the necessary data to display we will use the weather API of OpenWeatherMap

- **DevOps**: Azure will be used to containerize the app

- **Hosting**: Heroku will be used to host the app

3. External Interface Requirements

3.1 User Interfaces

- Front end software (JavaScript)
- External calls for weather and city data

3.2 Hardware Interfaces

- Windows
- Any modern browser, chromium based (Opera, Edge, Chrome), Mozilla etc.

3.3 Software Interfaces

Operating system	Any that supports modern browsers, like Windows, macOS, all well-known LINUX distributions
Database	MongoDB for holding the user's saved data like user information and saved cities
Hosting &	The app is hosted in Heroku and uses docker for shipping and deploying
Integration	We are using the Fetch API that is built-in for javascript to get the data necessary for the broadcasts (weather and city info)

3.4 Communications Interfaces

This project supports all types of modern web browsers. We are using HTML, CSS, and JavaScript to load and render data.

4. System Features

4.1 Check a weather broadcast by searching

4.1.1 Description and Priority

Choosing a city and displaying the weather details for it. It is the main and most important feature of the city

4.1.2 Stimulus/Response Sequences

- **Search for a city that exists:** The results of the search show up as a dropdown menu with extra details (e.g. Country) so the end user can select which one they want
- **Search for a city that doesn't exist:** A dropdown message will notify the end user that their search yielded no results

4.1.3 Functional Requirements

REQ-1: Weather API or alternative API is not offline

4.2 Check a weather broadcast by choosing from the list

4.2.1 Description and Priority

Selecting a city from a list and displaying the weather details for it. It is the main and most important feature of the city

4.2.2 Stimulus/Response Sequences

- **Click a city from the list:** The weather broadcast for the city shows up

4.2.3 Functional Requirements

REQ-1: Weather API or alternative API is not offline
REQ-2: User is logged in

4.3 Create a user

4.3.1 Description and Priority

Clicking on "Create user" should prompt the user to a form to input their data to create a user within the site.

4.3.2 Stimulus/Response Sequences

- **Click "create user" button:** The necessary form shows up so the user can input their information
- **User inputs the data and clicks "Save":** A screen shows up indicating the user to check their inbox for a confirmation email
- **User confirms their account creation:** A user record is created and saved into the database

4.3.3 Functional Requirements

Invalid text input: error message

4.4 Log in

4.1.1 Description and Priority

Clicking on “Login” should prompt the user to a form to input their data to log in to the site

4.1.2 Stimulus/Response Sequences

- **Click “User login” button:** The necessary form shows up so the user can input their information
- **User inputs the data and clicks “login”:** If data is correct the user is redirected to the front page with ‘logged in’ status. If the data is incorrect an error message is displayed and redirection occurs.

4.1.3 Functional Requirements

Invalid text input: error message

4.5 Save a city to user profile

4.1.1 Description and Priority

After login, the user can save a city for faster future access.

4.1.2 Stimulus/Response Sequences

- **[After searching a city] the user clicks the ‘save city’ button next to the broadcast data:** The city is saved to the user’s record and is accessible from the front page.

4.1.3 Functional Requirements

REQ-1: Weather API or alternative API is not offline
REQ-2: User is logged in

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Fetching weather data:** This should be as fast as possible and should consider that outside systems may be unavailable or slow. Circumstances under which data is not delivered should be kept to as less as possible.

- **User system:** Users should be able to create an account and login without having to perform many steps. The user creation system and the user login system should be lightweight and effective.

5.2 Safety Requirements

If the host of the weather data is unavailable or offline a secondary one should be ready to be used as a replacement for as long as is needed.

5.3 Security Requirements

Since hosting is being done on Heroku most of the security needs are being met by them, without extra steps needing to be taken by the development team to meet security standards.
Some basic steps should still be taken as to not expose users' information, which would be covered by providing a working login system.

5.4 Software Quality Attributes

- **Availability:** Weather data should be always available for users to check
- **Correctness:** The fetched data should be mapped to the requested city

5.5 Business Rules

Admins should generally be able to perform any action necessary to maintain the site. This includes but is not limited to: viewing user info for service needs, viewing analytics based on user's searches, privileges on actions like banning users or restoring locked accounts.

End users should generally be able to perform a search, create a user and login. This means interaction with the end product as opposed to admin functions.

6. Other Requirements

To Be Determined List

1. *Which weather service API will be used when the main one is down, slow or generally unavailable.*