Software Requirements and Design Document

For

Group 14

Version 1.0

Authors:
Conner F
Ryan R
James H
Davion M
DJ S

1. Overview

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

Gustbuddy will be a functional web application that will deliver a myriad of weather information to the user which can be shared through social media. We are proposing to develop the web app with a variety of APIs and custom algorithms to refine and personalize weather data for users accessing the app.

The system aims to deliver relevant, personalized, location-based weather information to users. Florida specific data related to weather and hurricane forecasts, hurricane data and statistics, and

2. Functional Requirements

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just <u>what</u>). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

1. Day/Week/2 Week weather forecast (HIGH)
2. On-this day past weather report breakdown (HIGH)
3. Florida Hurricane Damage Analysis Breakdown (HIGH)
4. Hurricanes in Florida Analysis breakdown
5. User log in/notifications (MEDIUM)
6. Florida Focus (MEDIUM)
7. Social media sharing implementation (LOW)
8. Hurricane Probability (LOW)

3. Non-functional Requirements

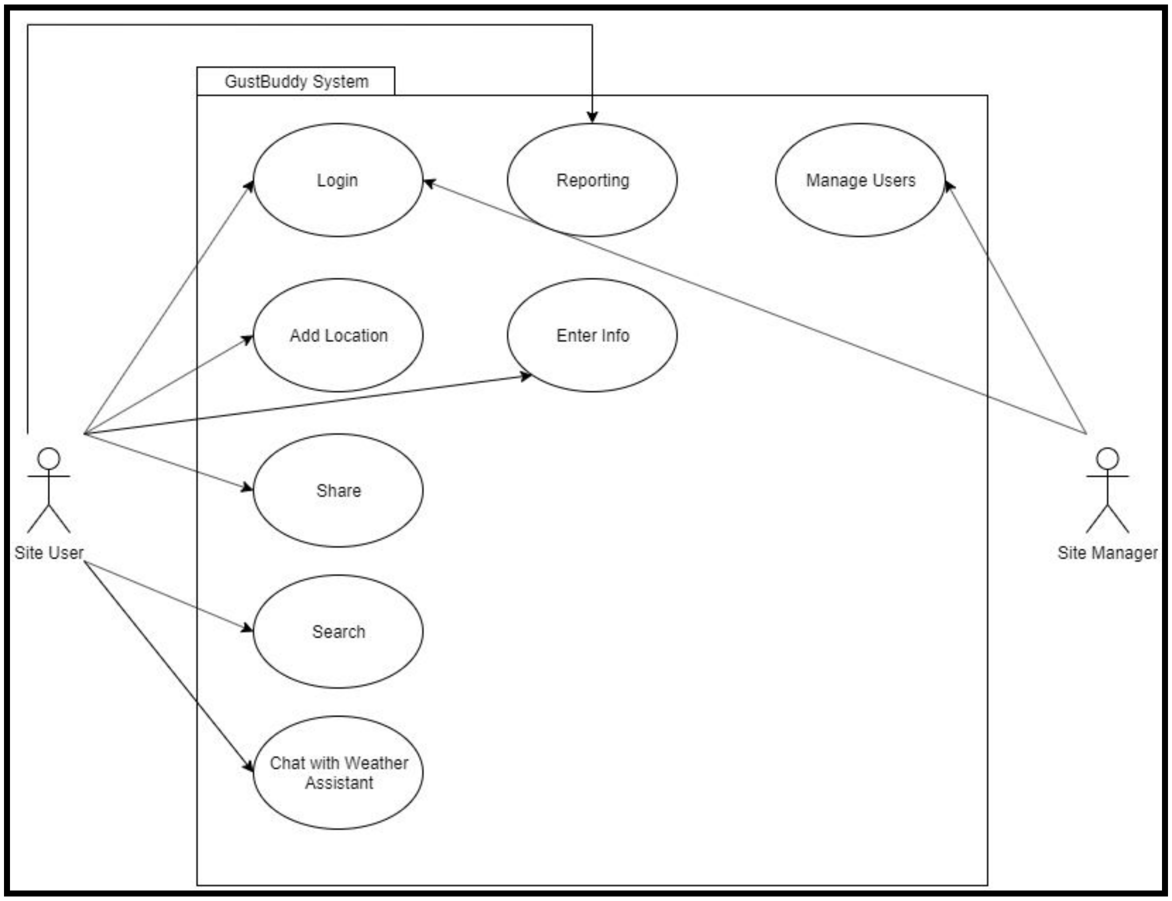
*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

1. Securing user's usernames and keeping their passwords anonymous
2. Writing efficient code
3. Attention to minimize performance lag
4. Consistent weather updates while site is open

4. Use Case Diagram

*This section presents the **use case diagram** for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

Actors: User, Administrators



5. Class Diagram and/or Sequence Diagrams

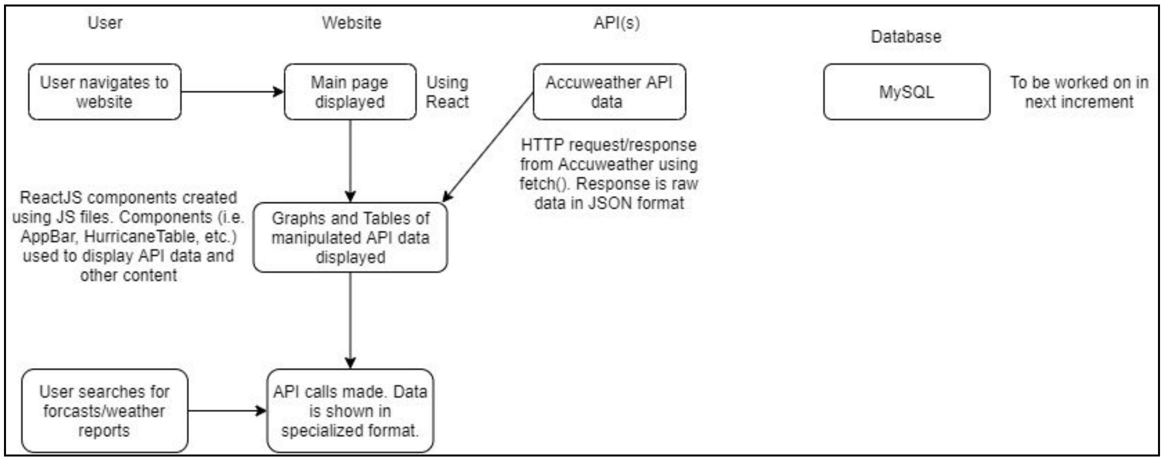
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **<u>not</u> Object Oriented** (i.e., you **<u>do not</u>** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for <u>all</u> the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the <u>functions</u> in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*


User
Website
API(s)
Database
User navigates to website
Main page displayed
Using React
Accuweather API data
MySQL
To be worked on in next increment
HTTP request/response from Accuweather using fetch(). Response is raw data in JSON format
ReactJS components created using JS files. Components (i.e. AppBar, HurricaneTable, etc.) used to display API data and other content
Graphs and Tables of manipulated API data displayed
User searches for forcasts/weather reports
API calls made. Data is shown in specialized format.

6. Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

The software currently operates as a locally hosted web application. The goal is for the web application to be published and be accessed from any remote host with an internet connection.

Hardware platform: Any
Operating system: Any

7. Assumptions and Dependencies

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

We plan to use React (A JavaScript library for building user interfaces), Mongo and/or SQL for our database. We will utilize a variety of APIs for various social media platforms (Facebook, Twitter, etc) as well as for incoming weather data (NOAA, etc). The final project will also need to be hosted on a web server for access outside of development.