**Software Requirements and Design Document**

**For**

**Group 8**

Version 1.0

**Authors**:

Marc Boustani

Carlos Saucedo

Carson Walker

Bryan Dean

Tomas Munoz-Moxey

# Overview (5 points)

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

We are creating a website for users to be able to find a spectrum of nature related areas such as bike trails, parks, nature trails and more. The users who interact with our system will be able to review and upload new areas by creating an account for the site. Additionally, the website will be able to user a person’s location and match them to nearby area from a database of locations we are currently in the progress of creating that users will be able to add to.

# Functional Requirements (10 points)

*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 what). 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. High Priority – The system shall properly handle user accounts, such as registering new users and logging in existing users.
2. High Priority – Display trail locations that are within a certain radius specified by the user, additionally meaning we need to be able to retrieve a user’s location.
3. High Priority – Users should be able to add locations to the database
4. Medium Priority – Displaying a user’s profile information, and giving options to enable Two-Factor Authentication (2FA).
5. Medium Priority – Adding user social stories for each location and giving logged-in users the ability to create stories for locations they visit.
6. Medium Priority – Prohibiting unregistered users from adding locations, social stories, or any other piece of data to the application, but still allowing them to view such things.
7. Low Priority – Designing the website in dark mode for aesthetic purposes.
8. High Priority – Giving users the ability to add reviews to the location
9. High Priority – Giving users the ability to add images to a location.
10. High Priority- Giving users the ability to view current weather for a given location
11. High Priority- Giving users the ability to upload and view social stories.
12. Medium Priority- Sending out ***stylized*** confirmation emails to registered users.

# Non-functional Requirements (10 points)

*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. Encrypting user passwords and not storing in plain text.
2. Limiting user interactivity if they do not have an account.
3. Ensuring web application is reliable in its load times.
4. Ensure web application is not hindering the performance of a user’s system.
5. Ensuring software is able to be easily maintained.
6. Email verification links should be sent in a timely manner (within 5 mins).

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases 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.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

# Class Diagram and/or Sequence Diagrams (15 points)

*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* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all 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 functions 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.*

# Operating Environment (5 points)

*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 project will run on a windows server with ASP.NET. Needs windows 10 installed as well as Microsoft SQL server. Program will run on the blazor architecture framework.

# Assumptions and Dependencies (5 points)

*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.*

Assumptions/Dependencies: we are using asp.net Identity to handle user accounts, login, registration, password encryption, and authorization. Through using this service we do not need a user class. We are using browserinterop to determine the users location. And mssql in visual studio to handle the database. Using openweather API to access weather information from the given location coordinates. We also rely on the MSSQL database to cache weather information so as not to be rate limited by openweather. We rely on sendgrid to send out the confirmation emails using the provided email from the user.