Software Design Specification

Water Weather Station

WESTERN WASHINGTON UNIVERSITY
WINTER 2023

Version 5.4

Client
ERIK FRETHEIM

Team
KAITLYN RICE
GARRETT KING
EMMA GEARY

Table of Contents

0. Glossary

BLE : Bluetooth Low Energy
DNS : Domain Name Server
CSV : Comma Separated Values
BCNF : Boyce-Codd Normal Form
SoC : System-on-Chip

1. Introduction

1.1. Purpose

This document will provide the required specifications for the Weather Water Station project. It will explain the interaction between devices and the function of every device.

1.2. Document Conventions

The acronym WWXS stands for Water Weather Stations. The term Buoy will reference the hardware unit that houses the processor and sensors. The term Device Interface will reference the mobile device software application used on a smart device. The term Web Interface will reference the Front-end website that runs on a web server.

1.3. Intended Audience and Reading Suggestions

This document is written with the intent of being reviewed by future WWXS Developers and WWXS Project Managers. This document is not designed to be a user guide and is only meant to be used to evaluate the completion of the software.

1.4. Project Scope

1.4.1 Goals

This project is on-going and has already been worked on by multiple teams. Our team will implement:

- A more appealing and responsive web design which will provide a better user experience and comprehension of collected data
- Web server functionality to handle app requests accordingly. This includes restructuring of the database and creating endpoints for app connections.
- A display feature on the app to show the data retrieved from the buoy. This will allow users to view and analyze data at the location. The app design will be cohesive with the website.
- On the app we will implement a buoy dashboard screen and location history screen, and rework the flow of the app to be less linear.
- Functionality within the app to connect to devices either by scanning for nearby devices, or by attempting to connect to a specific device.

1.4.2 Stretch Goals

As an ongoing project, subsequent releases will likely include features for:

- Ease of use and access
- Improving power consumption of the device
- Improving efficiency of the apps
- Stronger data security measures.
- Update interface of website
- Implementation of encrypted buoy passwords
- Redesign app to match the design of the website

1.5. References

There is documentation available for the Buoy, Device Interface, and Web Interface, on the sharepoint site:https://wwu2.sharepoint.com/sites/WWXS. Figure 1 is available from "Secure Software Development" (1st. ed.) by Marie Deschene and Erik Fretheim.

2. Overall Description

The WWXS will offer a unique way of keeping students engaged beyond the end of their camp. The buoy will allow for physical interaction with a device that creates useful data. With a higher level of user interaction Users will be more likely to retain the skills they have learned at the Sea Discovery Camp.

2.1. Product Perspective

The User should use this device to collect data about Water Weather conditions and practice Cyber-Security skills they have been taught. Users will hopefully see the value in the information they learned at the SEA Discovery Center by interacting with the WWXS. Researchers can use the WWXS to collect data about water conditions around the Pacific Northwest. This data is free and easily accessible though the web interface.

2.2. Product Features

The WWXS product will log sensor data, like the temperature and turbidity of the water that the Buoy is placed in, and allow for the user to access this data by transferring it beforehand from the Buoy to the Device Interface, then storing the data in the Web Interface. The Web Interface will allow the user to view the sensor data collectively and give the user the option to export the data to use elsewhere.

2.3. User Classes and Characteristics

Middle school students who've participated in the cybersecurity camp at the SEA Discovery Center are the primary user base for the software. Their interaction with the software is integral in the collection of data from the buoy.
Researchers are users that will only interact with the data collected by the students. Their main role is as the consumers of the information offered by our product.

2.4. Operating Environment

The buoy is designed to be used in aquatic environments where the appropriate sensors are submerged in no less than 2 meters of water. While the buoy can work in any body of water that meets this condition, bodies of water in close proximity to the SEA Discovery Center will likely be most common.

The Device Interface will be used outdoors near the buoy as it receives data from the Buoy and uploads it to the web server. The students and researchers will be accessing the data on the Web Interface through a web browser on a computer or mobile device.

2.5. Design and Implementation Constraints

The buoy must cost less than $50 to source. The device must adhere to Bluetooth Low-Energy (BLE) standards and subsequent radio transmission standards for communicating with the Device Interface.

2.6. Assumptions and Dependencies

The Device Interface software is running on a smart device such as a phone or tablet. The smart device is assumed to have basic internet access and ability to connect to bluetooth in order to upload data from the Buoy to the web server. The web server is running on a Unix-based operating system.

3. Features

3.1. Web Interface

3.1.1. Description

The web interface will be a Graphical User Interface accessible with any modern internet browser (i.e. Edge, Safari, Chrome, Firefox). It will display the information collected by the Buoys and allow for the downloading of that information by researchers. It will also allow for the sharing of keys to connect to the buoys from one user to another given permission.

3.1.2. Priority

This is a primary feature of the WWXS project. It is a high priority task, as the Web Interface is the portal to access the information collected by the Buoys. The Web Interface must retrieve data from the database and display it.

3.1.3. Stimulus and Response

The user visits a URL and will be greeted with a landing page.

3.1.4. Functional Requirements

The landing page should take less than 200ms to load not including DNS Requests. The entire site should be easy to navigate with a consistent menu bar and easily identifiable icons. It should allow the user to view and sort weather data by: source Buoy, date, and user.

3.2. User Authentication

3.2.1. Description

The Web Interface will have user accounts for the use of allowing access to Buoys. Each user should be able to be assigned a key or made the owner of it so they can share access to a Buoy.

3.2.2. Priority

This is a medium-priority task as it affects the primary user base but doesn't subtract from the current state of the project.

3.2.3. Stimulus and Response

The user will input some form of identification and a secret password to authenticate a web session.

3.2.4. Functional Requirements

The user authentication process should have a reasonable amount of security requirements. Such as minimum password requirements, password hashing, and encryption. Users accounts should also only be created by an authorized user. User authentication is implemented through a third-party such as Firebase.

3.3. Database

3.3.1. Description

The Firebase database will store user accounts and group passwords, while raw buoy data will be sent directly to the Web Interface from the app using a web API. If needed, a Firestore database will store buoy data. See Appendix 8.1 Database Scheme.

3.3.2. Priority

This is a high-priority task because user authentication is required to access a given buoy.

3.3.3. Stimulus and Response

The app will send formatted requests to the Firebase database to grant user access to a buoy, and to retrieve the users list of authorized buoys with associated data. If needed, The Web Interface will send formatted requests to the Firestore database for data and the database will respond with the information in the request.

3.3.4. Functional Requirements

The Firebase database will store user accounts and group passwords. If needed, a database will be constructed using Firestore free plan to store buoy data and cut costs, due to the fact that there should not be an excessive amount of data being stored.

3.4. Exporting Data

3.4.1. Description

The Web Interface will allow the download of raw buoy data in a commonly used file format such as CSV. This should be able to be sorted, then downloaded so a user can download data from: a single Buoy, a certain User, or over a specified period of time.

3.4.2. Priority

This is a medium-priority low-priority feature as the app and website are the two parts of this project that require the most attention at this time and their function requires the buoy to export data accurately.

3.4.3. Stimulus and Response

A user specifies the data they want. Then a file is downloaded from the Web Interface.

3.4.4. Functional Requirements

The compilation of the data should either:

A. Take less than 300ms to start downloading. or B. Show a progress bar or status indicator to provide the user with feedback that the data is being compiled before download.

3.5. Multi-user access to Buoy

3.5.1. Description

The Buoy will allow for different users with smart devices to establish connection and have access to a singular Buoy and its data. The Buoy currently allows for just a single smart device to establish and maintain exclusive rights to connect and access the Buoy. No

other smart device can establish connectivity with the Buoy once the first smart device has established connection with the Buoy. The scope of our responsibilities for implementing multi-user access is to modify the app to show the user their list of authorized buoys with associated data including buoy passwords, and to rework the bluetooth scanning logic to allow the connection to a specific device. In order for multi-user access to be fully functional, encrypted buoy passwords will also need to be implemented to check if the app's password matches the buoy password. This is a stretch goal for our group and may need to be implemented by the new 491 group.

3.5.2. Priority

This is a medium priority feature as it has been requested since some users may try to connect to a Buoy that is restricted to one specific smart device, yet it could be the same user that is attempting to connect.

3.5.3. Stimulus and Response

A user with specific privileges and keys accesses a specific Buoy and establishes a connection to it if they have the correct privileges. A user can set access to different Buoys through the Web Interface, which will update the user's set of accessible Buoys.

3.5.4. Functional Requirements

The user account shall have designated access given to the Buoy through the keys and group permissions set through the Web Interface. The user account on the Device Interface shall have a list of keys and access privileges for a certain set of Buoys. The Buoy shall have a specific key that is used to check the against with the keys given by the user on the Device Interface. The software shall fetch the list of Buoys the user can connect to from the Web Interface to the Device Interface.

3.6. Optimizing Power Usage of Buoy

3.6.1. Description

Measuring and testing changes with the sleep timer and the elapsed time to capture data from the sensors, and testing which values and optimizing the code can help increase the battery life. The Buoy's battery life can allow for the Buoy to capture data for long periods of time without needing to replace the batteries and missing moments when data could be captured.

3.6.2. Priority

This is a low priority task. The SoC we are currently using has features to use a low power state in between times of capturing data, so there is very little power drain from the battery.

3.6.3. Stimulus and Response

When the Buoy is powered on, when it is capturing data from the sensors, and when it is in a low power state in between capturing data, the amount of power necessary to run these tasks changes and uses more battery power.

3.6.4. Functional Requirements

The Buoy shall capture data when it wakes from the low power state to going back to low power state within 2000ms and shall transfer the logged data from the last 24 hours within 1000ms from the Buoy to the Device Interface. The Buoy should last at least 2 weeks before the battery power is depleted.

3.7. “Power on” LED for Buoy

3.7.1. Description

The buoy, though always on once the battery is connected, will have an LED light indicator of battery life. This will give users confidence that the buoy is functioning.

3.7.2. Priority

This is low-priority as it does not affect the functionality of the buoy and serves as assurance to users that the buoy is on and running.

3.7.3. Stimulus and Response

Once the buoy is given power, it remains on. The LED will light up to indicate this.

3.7.4. Functional Requirements

The LED on the buoy will light up when the buoy is given power. As the power drains, the LED light will dim before turning off once power is depleted at about 2 weeks.

3.8. More Sensors for Buoy

3.8.1. Description

The buoy currently has sensors implemented for insolation, salinity, turbidity, and temperature. A dissolved oxygen sensor has been a requested addition for the sensor lineup for the buoy.

3.8.2. Priority

This is a medium-priority task as the organization of the buoy takes precedence. This task cannot be effectively implemented until the buoy setup has been organized. Due to the new timeline, the task of implementing the dissolved oxygen sensor has been redistributed to the latest project group iteration. However, creating the space necessary for adding the additional sensor is within our scope and is part of the buoy organization task.

3.8.3. Stimulus and Response

When the buoy is powered on, it will begin gathering data from the various sensors attached to it and storing it to a database for 30 days.

3.8.4. Functional Requirements

The buoy will be placed in water for no more than 30 days. The sensors will collect data from the surface of the water as well as 2 meters below the surface of the water and store it in a database for users to collect via bluetooth connection.

3.9. Button for Bluetooth

3.9.1. Description

The buoy utilizes an ESP32 to run the software. Currently, the bluetooth is on 24/7 as it searches for a device to connect to in order to transfer collected data. However, this drains the batteries within a couple of hours. We will implement a button that, when pressed, will search for a bluetooth connection. Otherwise, the device will be in sleep mode.

3.9.2. Priority

This is a high priority task. Currently, the buoy cannot function under the necessary conditions without something implemented to lower the power consumption. This will be integral to ensuring the battery lasts for the full 30 days.

3.9.3. Stimulus and Response

When the button is pressed, the bluetooth signal from the buoy will initiate for 20 seconds, allowing a user with a smartphone to connect with the app and collect data. After 20 seconds, the buoy will return to sleep mode and the bluetooth signal will end.

3.9.4. Functional Requirements

The buoy will be placed in water for no more than 30 days. The button will turn on the bluetooth signal for a limited amount of time before shutting off to preserve battery life.

3.10. Buoy Assembly

3.10.1. Description

The buoy is currently working but the assembly is messy and confusing. We shall build a buoy prototype that accommodates the current sensors but also future-proofs the buoy by already including the necessary wires for additional sensors.

3.10.2. Priority

This is a high-priority task as the organization of the buoy takes precedence. More sensors cannot be added comfortably before this task is complete. Additionally, the comprehension of future groups relies on an organized state.

3.10.3. Stimulus and Response

When the buoy is powered on, it will begin gathering data from the various sensors attached to it and storing it to a database for 30 days.

3.10.4. Functional Requirements

The buoy will be placed in water for no more than 30 days. The sensors will collect data from the surface of the water as well as 2 meters below the surface of the water and store it in a database for users to collect via bluetooth connection.

3.11. App-Server Communication

3.11.1. Description

The current state of flow of the app is very linear. The user can log in, scan, connect, and send data. This will be reworked to bring the user to a buoy dashboard screen upon login. This screen will show the user their list of authorized buoys, with the option to view location history and associated data of each buoy. The list of authorized buoys, and all associated data, will be gathered upon landing at the buoy dashboard screen using http requests from the app to API endpoints set up on the web server. See Appendix 8.2: App Storyboard with Buoy Dashboard Screen and Individual Scan.

3.11.2. Priority

This is a high-priority task, because future functionality of users being able to connect to a specific buoy will require the user to see their list of buoys.

3.11.3. Functional Requirements

In order to retrieve data successfully, the API endpoint to retrieve data from must also be implemented.

4. External Interface Requirements

4.1. User Interfaces

4.1.1. Device Interface

The WWXS will include a reasonably intuitive user interface with the purpose of allowing the user to create a secure connection to the Buoy and retrieve data. Then allow for the uploading of that Data to the database.

4.1.2. Web Interface

The Web Interface will display a visually appealing data vs. time display and options for downloading the data. It will also include a way of accessing a user's account information and giving permissions to use keys for specific buoys.

4.2. Hardware Interfaces

4.2.1. Buoy

The Buoy does not currently have any hardware interface features. The unit is always-on once the battery is connected.

4.3. Software Interfaces

4.3.1. Database

The database will be a standard database interface with query language specific to its implementation. It should have the basic ability to add, remove, and update information contained within the Database.

4.3.2. Web API

The Web API will include functions that allow the Web Interface to access the database and send buoy data to the app via http requests. It should only allow the features specific to the project to prevent mis-use of the database by the public.

4.4. Communications Interfaces

4.4.1. Buoy Bluetooth API

The Buoy will communicate with the smart device utilizing the Bluetooth Low-Energy (BLE) protocol. The Device Interface will send commands to the Buoy and the Buoy will process and run the command requests.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

The buoy shall process and complete requests given to it by the device interface within a reasonable amount of time. The buoy shall return the latest data log within 2000ms from the time the data is requested. As for the buoy battery performance and efficiency, the project currently does not have any set figures on how long the battery should last, but since the buoy uses replaceable AA-size batteries, we will aim for a run-time of 2 weeks on a single pair of batteries before they are depleted of power. The Web Interface should load the homepage in less than 500ms under a reasonable wired network condition.

5.2. Safety Requirements

We do not expect any possible safety issues in the operation of the WWXS. Our device includes no physical movement and uses a pre existing form factor of a water bottle that can be reasonably handled by any individual.

5.3. Security Requirements

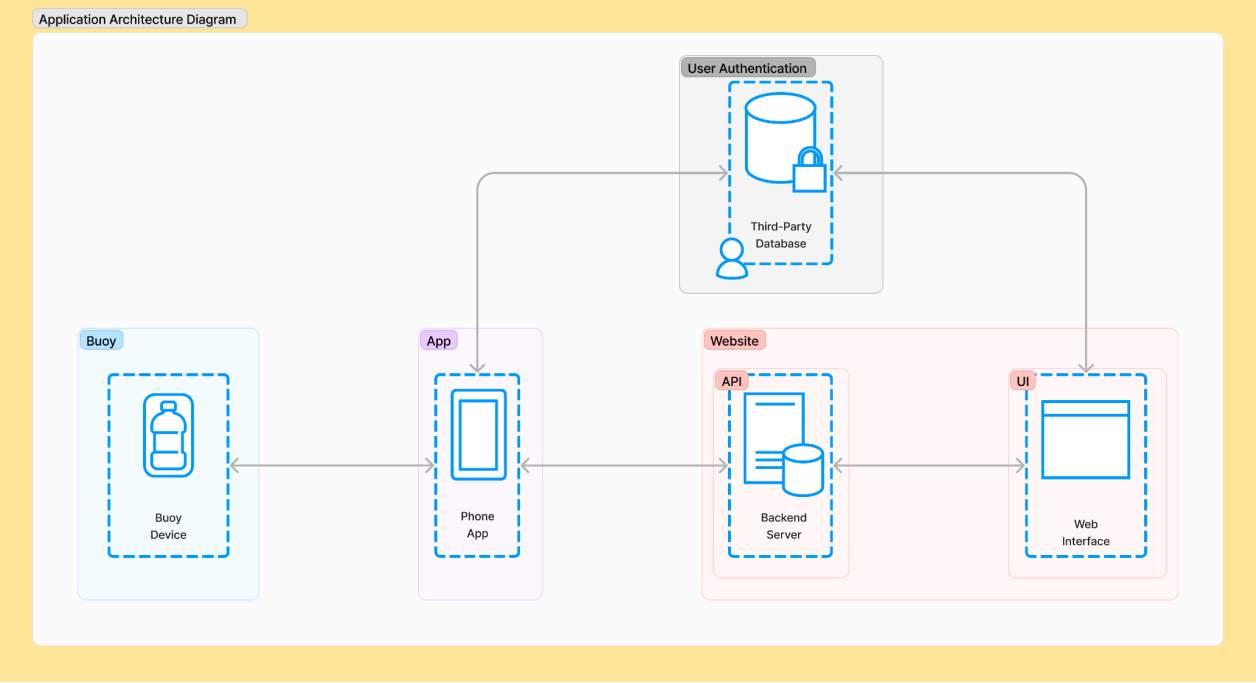
The Database will not be communicated with directly from any user interface without the use of a secure API. User information should be minimal and passwords should at least be hashed and not stored in any un-encrypted form. We recommend the use of SSL Certificates such that the Web Interface uses HTTPS but this may be beyond the scope of the project as it can be cost-prohibitive and the user data contains no personal information beyond a username and password.

5.4. Software Quality Attributes

- The Web Interface should allow users to view data in less than 2 interactions.

- The Web Interface should handle all HTTP errors with a visually descriptive error page.
- The Device Interface should display useful information on its connection status with both the Buoy and Web Interface so it is both engaging and easy to diagnose issues.
- The Device Interface should be able to receive and parse data from older versions of the Buoy code along with new versions of the Buoy code
- The Device Interface should be able to receive and parse data from Buoys with newly added sensors along with Buoys that have less sensors.
- The Device Interface should show a visual warning message if the data received from the Buoy is incomplete or corrupted.
- The Buoy should wake from the low-power sleep state, read and log all the sensor data, and go back to the low-power sleep state under 2000 ms.

6. Analysis Models



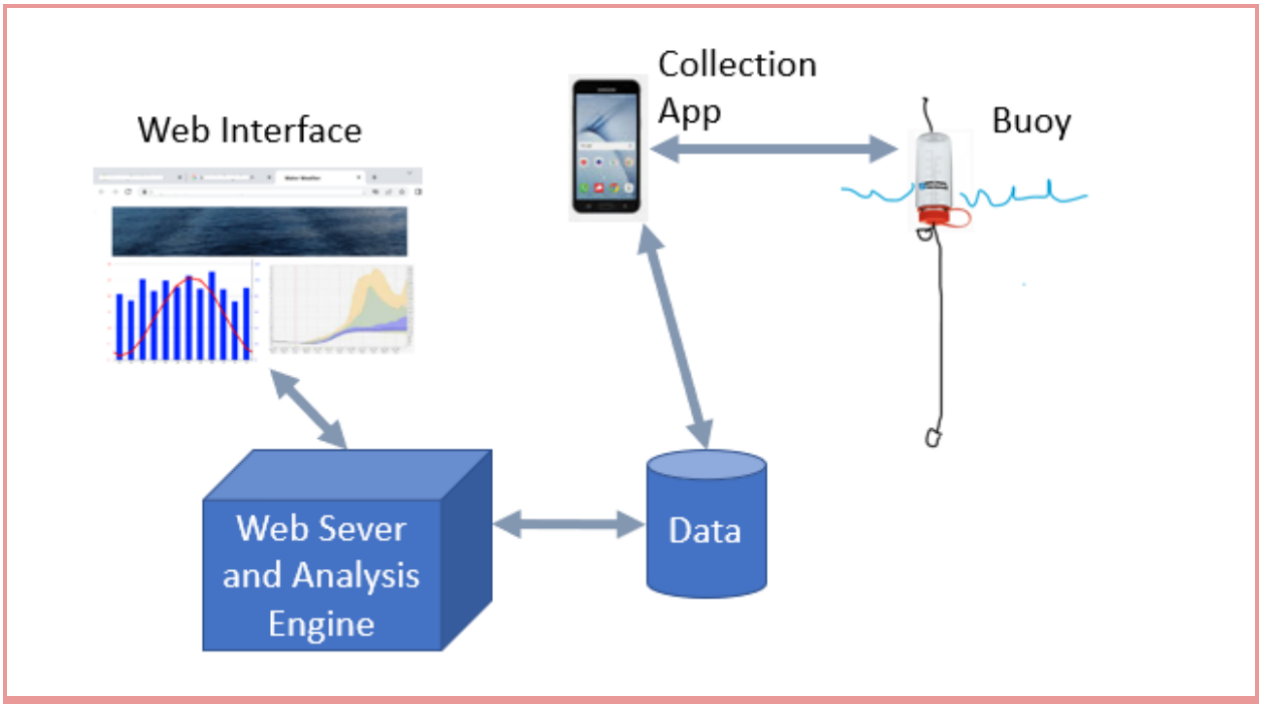


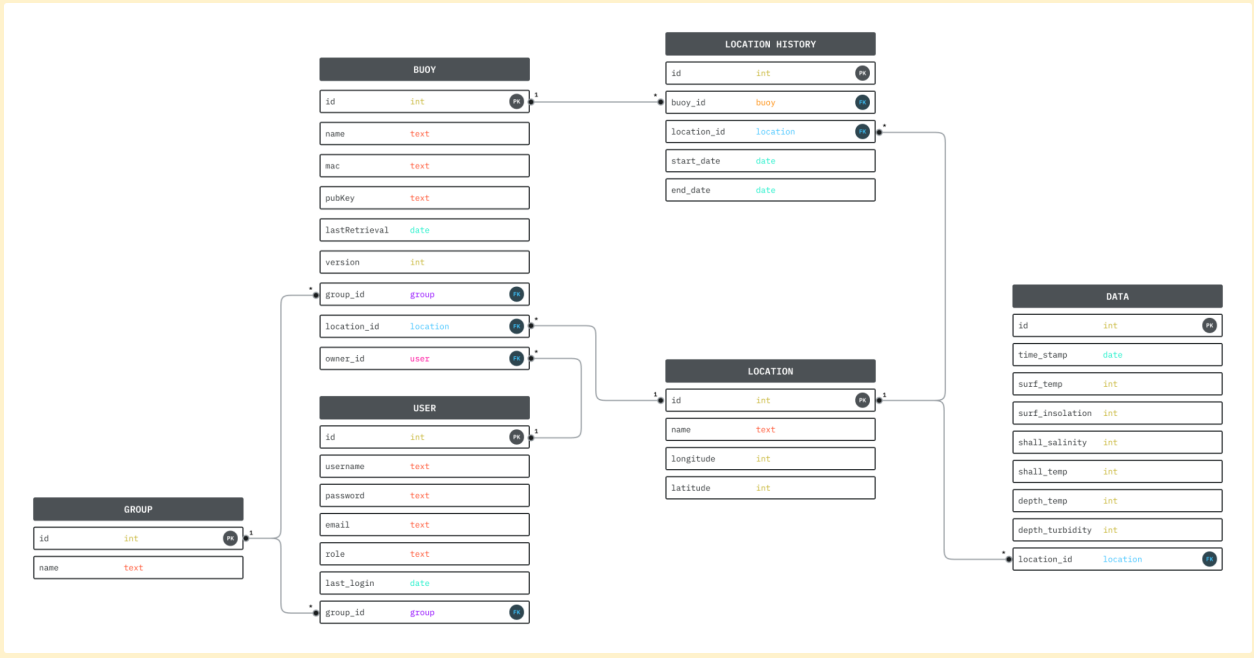
Figure 1: Application Architecture of the Water Data Collection System

7. Issue List

None currently.

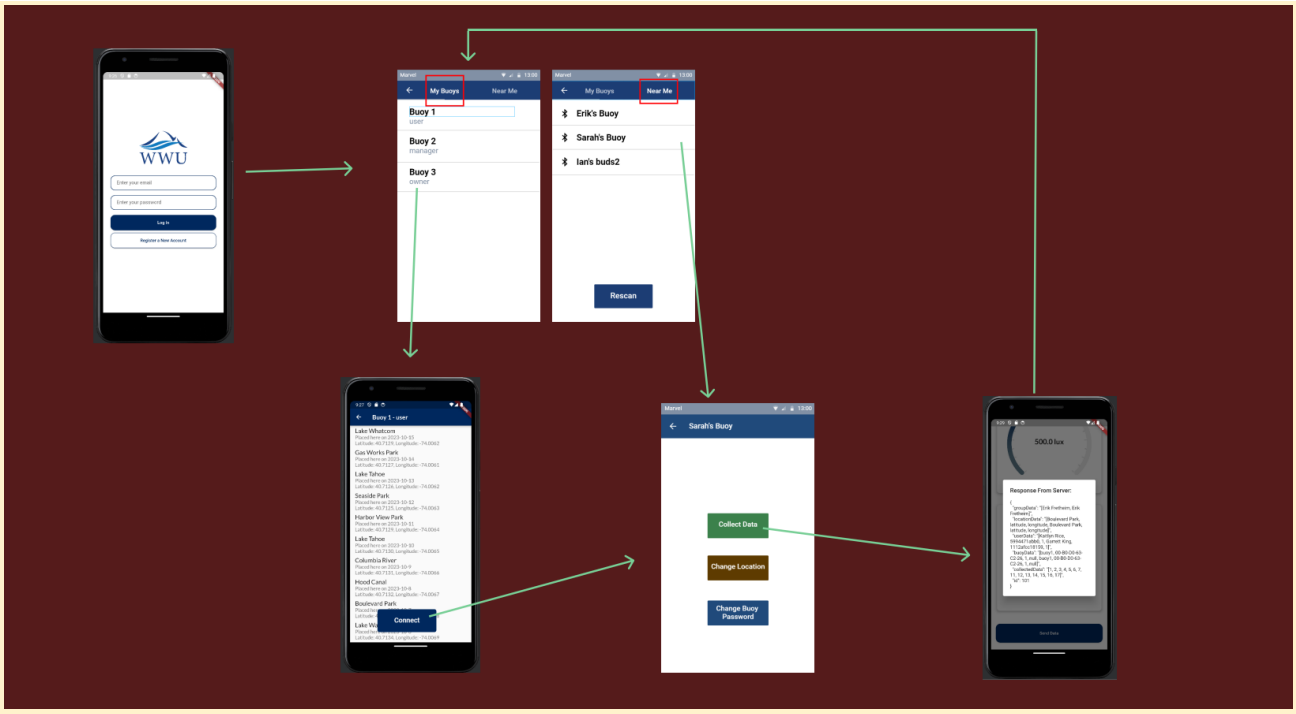
8. Appendix

8.1. Database Schema



Appendix 8.1: Database Schema for the Water Data Collection System

8.2. App Storyboard



Appendix 8.2: App Storyboard with Buoy Dashboard Screen and Individual Scan

9. Schedule

Finals week | Short week

Week of:	Tasks:
3/28/2023	Get established, plan meetings for the quarter
4/3/2023	Garrett and Emma finish Flutter modules
4/10/2023	Exporting Data between buoy/app, app/site, multi-user access
4/17/2023	Exporting Data between buoy/app, app/site, multi-user access
4/24/2023	Exporting Data between buoy/app, app/site, multi-user access
5/1/2023	MVP mostly finished, check with sponsor
5/8/2023	Add/fix anything sponsor request, request sponsor approval. Make presentation
5/15/2023	Practice presentation. Present on Friday 5/19

5/22/2023	Polish MVP
5/29/2023	MVP delivered, sponsor sign-off due
6/5/2023	Finals week
9/27/2023	Get established, plan meetings for the quarter Implement more sensors for buoy, Data display on app, Data interpretation on website
10/2/2023	Implement more sensors for buoy, multi-user access,, Data interpretation on website
10/9/2023	Implement more sensors for buoy, LED light, multi-user access, Data display on app, Data interpretation on website
10/16/2023	Implement more sensors for buoy, LED light, Data display on app, Beautifying site upload website to server
10/23/2023	Implement more sensors for buoy, LED light, Data display on app, Beautifying site upload website to server and fix database
10/30/2023	Project mostly done, check with sponsor
11/6/2023	Add/fix anything that sponsor requests, request sponsor approval
11/13/2023	Work on presentation
11/20/2023	Tie up loose ends
11/27/2023	Practice presentation
12/6/2023	Presentations
12/11/2023	Final product delivered

Version History

Ver.	Date	Team Member	Change
1.0	2/3/2023	Kaitlyn Rice	Formatted Document
1.1	2/3/2023	Emma Geary	Added information, made some edits

1.2	2/3/2023	Garrett King	Reviewed Document, made some edits
	2/7/2023	Emma Geary	Made edits according to feedback
2.0	2/24/2023	Emma Geary	Made edits according to feedback from V1
2.1	3/9/2023	Emma Geary	Made edits based on sponsor feedback.
“	“	Emma Geary	Added schedule outline
3.0	3/10/2023	All	Added schedule details for next two quarters
3.1	4/20/2023	Emma Geary	Added Section 3.9 for new requirements. Moved some of the schedule around for Spring quarter to accommodate presentation dates.
3.2	4/21/2023	Garrett King	Updated Database features to more accurately represent what we have, as well as what we may implement.
4.0	5/17/2023	Emma Geary	Edited wording on the “additional sensors” requirement that is more specific to the kind of sensor we have been requested to add. Added a requirement for better buoy organization.
5.0	10/18/2023	Kaitlyn Rice	Update priority of website functionality and replace analysis model figure 1
5.1	10/18/2023	Kaitlyn Rice	Add Appendix & database diagram
5.2	10/23/2023	Emma Geary	Updated wording in section 3.8
5.3	10/24/2023	Garrett King	Updated all app related documentation
5.4	10/24/2023	Kaitlyn Rice	Updated software interface for web api