Con-Sea-Erge Fish Feeder

DESIGN DOCUMENT

Team 25
ISU Accounting
Dr. Fila
Devin Milligan/Firmware
Ethan Peterson/Firmware
Drake Dodson/Backend
Ryan Hickok/Backend
Josh Van Drie/Frontend
Brian Tran/Frontend
Hunter Northern/Hardware
sdmay23-25@iastate.edu
https://sdmay23-25.sd.ece.iastate.edu

Revised: 12/2/2022

Executive Summary

Development Standards & Practices Used

- Model Views
- Agile
  - Weekly standup meetings
  - Iterative project management
  - Constant input from clients
- Version Control Software
  - Git
  - Allowed us to monitor the different versions of software being changed
  - CI/CD for automated software testing

Summary of Requirements

- Create a physical device that can dispense food into a tank of fish
- Program physical device to accept updates to a schedule to feed the fish as well as convey data about sensor data to backend/frontend
- Develop a backend that can dispatch updates between frontend and physical device as well as store user and tank data
- Develop a front facing application that users can login to and manage their individual tanks

Applicable Courses from Iowa State University Curriculum

Com S 309 - Software Development Practices

Com S 319 - User Interfaces

Com S 409 - Software Requirements

New Skills/Knowledge acquired that was not taught in courses

3D Printing

React Native

Firebase

Circuit board schematic design

Part selection for a embedded design

Wirelessly connected embedded systems

Table of Contents

1 Team

1.1 Team Members

Drake Dodson

Ryan Hickok

Devin Milligan

Hunter Northern

Ethan Peterson

Brian Tran

Joshua Van Drie

1.2 Required Skill Sets for Your Project

UI/UX Frontend Design

Firebase BaaS

Backend Design

Database Management

Hardware Design

Firmware Design

3D Printing / 3D Design

1.3 Skill Sets covered by the Team

UI/UX Frontend Design

Backend Design

Database Management

Hardware Design

Firmware Design

1.4 Project Management Style Adopted by the team

We will be using an agile approach to this project. The agile process will aid in our project development throughout the project in many ways including input from the clients themselves. We will be completing this project in small manageable phases, whilst getting input from the client on each small step that we pass. We are developing this project in such a way that it is a fluid process to jump between phases we are currently working on depending on the needs of the clients themselves. Once we are able to have tangible prototypes, our team will be putting these prototypes in the hands and eyes of the clients to follow the process of getting complete and informative input from the users themselves.

We are managing our Agile process through our "stand-up" meetings over discord team call every Sunday. The main goal of this meeting is to go over the generalities of what each team member worked on in the previous week. As well, we will be assigning week-long tasks to each individual in these meetings to ensure we are keeping on track and utilizing our teammates' skills to their fullest. During this meeting, we will also talk about any blockers that each team member might have as well.

We will be using Gitlab to track our progress as a team and to identify areas where we are falling behind. Issues will be assigned to team members when a task needs to be completed, and during the stand-up meetings, these issues will be re-evaluated if they need more time, people, or resources. This will help our Agile process move along as fluidly as possible.

1.5 Initial Project Management Roles

Brian - Frontend

Josh - Frontend

Hunter - Hardware Design / 3D Printing

Ryan - Backend / Firebase

Drake - Backend / Firebase

Devin - Firmware

Ethan - Firmware

2 Introduction

2.1 Problem Statement

The Accounting Services office has four permanent resident fish living in their office space that need to be monitored and fed around the clock. Since the employees in the Accounting office are not able to fulfill the needs of the fish on weekends, after work hours, and on vacations, this introduces the problem of how to continue to meet those needs while not currently in the office. This problem is an important issue because the fish are part of the Iowa State family and need to be well taken care of 24/7.

The solution we propose is to have a wired-electronic fish care device attached to the tank of the fish that will be able to be controlled remotely from an iPhone application. This device and application will allow the users to control the automatic fish feeding schedule from their iPhone, while also being able to monitor the pH and temperature level of the tank. This will be an adequate solution to this problem because it will allow for the fish to be monitored and fed around the clock while not having to be physically in the office.

2.2 Intended Users and Uses

Our product will be used by The Accounting Services office employees at Iowa State. The fish and employees of the Accounting Services office will benefit from the project. The employees are very passionate about their resident fish living in the office and want to be able to take care of them in the best way possible around the clock, even when not in the office. They will need a way to monitor the fish tanks' pH and temperature remotely to ensure that they are doing fine, as well as be able to control their feeding schedule, in the office and out of the office. They will benefit from this project due to the peace of mind knowing that their fish are being fed and monitored around the clock, allowing them to keep their residents longer and healthier.

2.3 Requirements & Constraints

Functional requirements

1. Device Requirements
    - 1.1. Sensing Technologies
        - 1.1.1. The system shall have the ability to read pH levels of the water
        - 1.1.2. The system shall have the ability to read temperature levels of the water
    - 1.2. Device Enclosure
        - 1.2.1. The device shall be 2.5' x 2.5' in x 4'
        - 1.2.2. The device shall hold a months worth of food
        - 1.2.3. The device shall be able to hold varying amounts of food
        - 1.2.4. When the device dispenses food the food shall make it into the water of the tank
        - 1.2.5. The device shall be mounted outside of the tank
    - 1.3. Device Power Supply
        - 1.3.1. The device shall be powered by an outlet with 5V power supply
    - 1.4. LED
        - 1.4.1. The device shall have an LED to display the status of the tank
        - 1.4.2. When the device is not connected to the internet, the LED shall display a yellow light
        - 1.4.3. When the feeding motor is jammed, the LED shall display a red light
        - 1.4.4. When the device is operating normally, the LED shall display a green light
2. Frontend Requirements
    - 2.1. Login
        - 2.1.1. The user shall have a username and password
        - 2.1.2. When the user wants to access tank information they shall login through using their username and password
    - 2.2. Notifications
        - 2.2.1. When the scheduled feeding takes place the user shall be notified with a notification
        - 2.2.2. When pH or temperature levels are high the user shall be notified with a notification
    - 2.3. Tank
        - 2.3.1. When the user adds a new tank they shall have the ability to give that tank a unique name
        - 2.3.2. While the device is connected to the frontend it shall display pH levels and temperature levels of the tanks

- 2.3.3. When the user clicks on the manual feeding option the device shall dispense food into the tank
  - 2.3.4. The user shall be able to set a schedule for when feedings should be done
  - 2.3.5. The user shall be able to set levels for high pH values and temperature values
  - 2.3.6. When a feeding takes place the user shall be able to view a timestamp of the time that food was dispensed
- 2.4. Mobile Device
  - 2.4.1. The frontend shall be written in React Native
  - 2.4.2. The frontend shall be accessible on iOS and android

3. Backend Requirements
   - 3.1. Frontend Connectivity/API
     - 3.1.1. The backend shall connect to the frontend app and receive updates about the status of tank schedules
     - 3.1.2. The backend shall verify login information with the frontend.
   - 3.2. Hardware Connectivity/API
     - 3.2.1. The backend shall connect to specific hardware applications
     - 3.2.2. While the backend is connected to the hardware it shall update the hardware parameters when changes are made from the frontend
   - 3.3. Backend Processes
     - 3.3.1. The backend shall store login information, connected devices, previous feedings and monitoring levels within a database
     - 3.3.2. The backend shall notify the frontend of any failures detected within the firmware device
4. Firmware Requirements
   - 4.1. Wireless/Backend Connectivity
     - 4.1.1. The device shall be able to connect wirelessly to the backend via Wi-Fi
     - 4.1.2. While the device is online it shall send information to the backend
     - 4.1.3. While the device is online it shall receive updates from the backend
   - 4.2. Feeder Control
     - 4.2.1. When the motor is jammed the device shall notify the backend
     - 4.2.2. The device shall move the motor to dispense food of an amount specified by the user
     - 4.2.3. When the device receives an update to the schedule the device shall update the onboard schedule to the new one
   - 4.3. Sensor Readings
     - 4.3.1. While the device is connected to the backend it shall send temperature and pH information
     - 4.3.2. When a set time is reached the device shall send an update to the backend of the new sensor readings
   - 4.4. Offline Use
     - 4.4.1. When the device is incapable of connecting to the backend it shall then enter offline mode
     - 4.4.2. The device shall store the current schedule in memory
     - 4.4.3. While the device is offline the device shall continue to use the stored schedule
     - 4.4.4. The device shall have an internal clock that can be used to keep track of time
     - 4.4.5. The device shall have a preset “doomsday” schedule that the device can revert to on software reset
     - 4.4.6. While offline the device shall update the LED to display that the device is offline

Nonfunctional requirements

1. The device shall be easy to fill
2. The device shall be easy to mount to the tank and will not cause any alterations to the tank
3. The frontend shall be easy to use and navigate
4. The frontend shall be accessible from most mobile devices
5. The backend shall be able to handle the amount of user data stored in the backend

3 Project Plan

3.1 Project Management/Tracking Procedures

We will be using an agile approach to this project. The agile process will aid in our project development throughout the project in many ways including input from the clients themselves. We will be completing this project in small manageable phases, whilst getting input from the client on each small step that we pass. We are developing this project in such a way that it is a fluid process to jump between phases we are currently working on depending on the needs of the clients themselves. Once we are able to have tangible prototypes, our team will be putting these prototypes in the hands and eyes of the clients to follow the process of getting complete and informative input from the users themselves.

We are managing our Agile process through our “stand-up” meetings over discord team call every Sunday. The main goal of this meeting is to go over the generalities of what each team member worked on in the previous week. As well, we will be assigning week-long tasks to each individual in these meetings to ensure we are keeping on track and utilizing our teammates' skills to their fullest. During this meeting, we will also talk about any blockers that each team member might have as well.

We will be using Gitlab to track our progress as a team and to identify areas where we are falling behind. Issues will be assigned to team members when a task needs to be completed, and during the stand-up meetings, these issues will be re-evaluated if they need more time, people, or resources. This will help our Agile process move along as fluidly as possible.

3.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Frontend

- Mockup Design
- Initial Frontend Development
    - Research useful libraries
    - Create React components
    - Test cases for usage
- Connect with Backend

        - Implement components with data from backend
        - Ensure data is displayed properly
    - Give demo to clients
        - Ensure that clients are able to use application with ease
        - Clients should be able to use application on their own
    - Publish to App store

Backend

- Backend Ideation
- Initial Backend development
    - Creating REST controllers
    - Setting up database
    - Creating dataflow
- Cloud deployment
    - Test google firebase
    - Evaluate it's effectiveness
- Full system integration
    - Verify that data flow works between backend, frontend and firmware

Hardware/Enclosure

- Enclosure Ideation
- Enclosure Prototyping and measurements
- Enclosure Testing
    - Test feeding apparatus
        - Test motor speed. May need adjustment in gear ratio for speed
    - Test fish tank mounting
    - Test food containment/input
    - Receive input on the device looks
- Enclosure Final Design
- Produce Final Design

Hardware/Chip-Down

- Research development boards for prototype
- Purchase development boards
- Configure a complete prototype device consisting of development boards
- Choose the final components for the device
- Design a custom PCB
- Produce PCB
- Assemble final product boards

Firmware

- Development board actions
    - MCU/Wifi Development Board
        - Connect to internet
        - Send mock requests to a service like POSTman for testing purposes
        - Send requests to our external backend service
        - Handle requests from the backend service

        - Be able to store information in non-volatile memory
            - Feeding Schedule and last fed time
        - Configure an RTOS timer to keep track of scheduled timings
        - Complete device states in accordance with device sensors and schedule
        - Identify state of Wifi connection and motor to display problems to LED
    - Temperature Development Board
        - Use the MCU development board to communicate with the temperature sensor development board
        - Send requests to configure the temperature sensor chip
        - Receive data from the temperature sensor
        - Test received data from the temperature sensor for accuracy
        - Send data from temperature sensor to MCU development board
    - pH Development Board
        - Use the MCU development board to communicate with the pH sensor development board
        - Send requests to configure the pH sensor chip
        - Receive data from the pH sensor
        - Test received data from the pH sensor for accuracy
        - Send data from the pH sensor to the MCU development board
    - Motor
        - Use the MCU development board to turn the motor on and off
    - LED
        - Use the MCU development board to turn the LED on and off
        - Use the MCU development board to turn the LED on any requested color
- Custom PCB
    - Confirm the device works as the development board prototype
    - Refine code that was necessary with development boards
    - Test device actions
        - Test device communication with the backend
        - Test device pH reading
        - Test device temperature reading
        - Test device motor actuation
        - Test device LED modes
        - Test device process actions

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone	***Metric***
Firmware Milestones	
Connection to backend	Able to receive/send a request to/from the backend service
Temperature Reading	Ability to receive data from the temperature sensor

	and convert this data into a recognizable and accurate temperature
pH Reading	Ability to receive data from the pH sensor and convert this data into a recognizable and accurate pH
Motor Running	Able to start and stop the motor on command from the MCU
LED Running	Able to turn the LED off and on. Also, able to turn the LED to any color requested by MCU
Task Scheduler Functional	This is measured by the ability to perform a task within the MCU based solely on a time delay handled by the MCU. (e.g. perform a temperature reading every 5 minutes)
Offline Feeding Scheduler Function	MCU is able to recognize that it is offline and still able to feed fish on schedule.
Hardware/Enclosure Milestones	
Prototype design	Prototype designed out with measurements of all the parts.
Prototype in 3D printing software	Prototype designed in the software needed to 3D print enclosure.
Final Prototype	Measured out prototype that fits the enclosure and all of our parts correctly.
Hardware/Chip-Down Milestones	
Frontend Milestones	
Create Mockup	Full design is complete with all application pages and components
Create Components	All components designed in the mockup are implemented but do not display live data
Connect to backend	Components and application are connected to backend and display live data from database
Test Components	Ensure that all components are behaving correctly and handle changes in data
Publish Application	Make app available for clients to download from the app store
Backend Milestones	

Get service on Firebase	The backend is running and accessible through firebase
Connection to the Firmware	The backend is able to send schedule updates to the firmware
Connection to the Frontend	The backend can send and receive updates from the frontend application
Connect to database	The backend is able to store data within a database and access that information
CI/CD Pipeline	The backend can be continuously deployed when changes are made
Data Storage and Analytics	The backend will store the readings from the sensors and store the information to view trends

3.4 Project Timeline/Schedule

Firmware Gantt chart

10/17	10/24	10/31	11/7	11/14	11/28	12/5	12/12
Waiting for parts to arrive and additional research	In Parallel Designing the enclosure parts to hold the food and attach to motor						
	Connection to backend						
		pH and temperature implementation and testing					
				Motor working and testing			
						Mode identify and LED use for this	

1/16	1/23	1/30	2/6	2/13	2/20	2/27	3/6	3/20	3/27	4/3	4/10	4/17	4/24	5/1	5/7
Task Scheduler function															
		Offline scheduler													
						Testing(T)									

Backend

10/17	10/24	10/31	11/7	11/14	11/28	12/5	12/12
Setting up Spring boot							
		Connection to Firmware					
		Connection to Frontend					
				Deploy to FireBase			
					Connect to Database		

1/16	1/23	1/30	2/6	2/13	2/20	2/27	3/6	3/20	3/27	4/3	4/10	4/17	4/24	5/1	5/7
CI/CD															
	Data Storage and analytics														
						Testing(T) / Feature Refinement									

Frontend

10/17	10/24	10/31	11/7	11/14	11/28	12/5	12/12
Figma Design							
		Develop login and main screen					
			Connect to backend				
				TDD, creating tests for components			

1/16	1/23	1/30	2/6	2/13	2/20	2/27	3/6	3/13	3/20	3/27	4/3	To end date	
Develop main screen with tank/fish info													
	Test backend connection with features												
		Frontend feed scheduler (making and viewing)											
TDD, tests for components and integration.													
						Code refinement, testing, and documentation							
		Figure out and deploy application for iOS											

3.5 Risks And Risk Management/Mitigation

Risks

Risk	Probability
Chip-down parts not being in stock by end of the project	0.25

Device not allowed to connect to Iowa State WiFi	0.1
PCB parts ship date too late	0.2
Google Firebase being difficult to work with	0.1
Cannot publish App to Apple App Store	0.2

3.6 Personnel Effort Requirements

Tasks	Time (Person-Hours)
Learn CAD and design the 2 different hardware designs for prototypes	21
Research Firebase and implement in design	8
Learn React-Native and state management for mobile applications	10
Learn Figma and create design	5

3.7 Other Resource Requirements

- IOS Device
- Google Firebase access
- Hardware device internet access
- Enclosure
    - Access to 3D Printer
    - 3D Printer Filament
    - Access to 3D modeling software
- Device Hardware
    - MCU/Wifi enabled chip
    - Temperature conversion module
    - K-Type Thermocouple
    - pH Probe
    - pH conversion module
    - 3.3v - 5v DC Motor
    - RGB LED

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	**Description**	**Examples**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

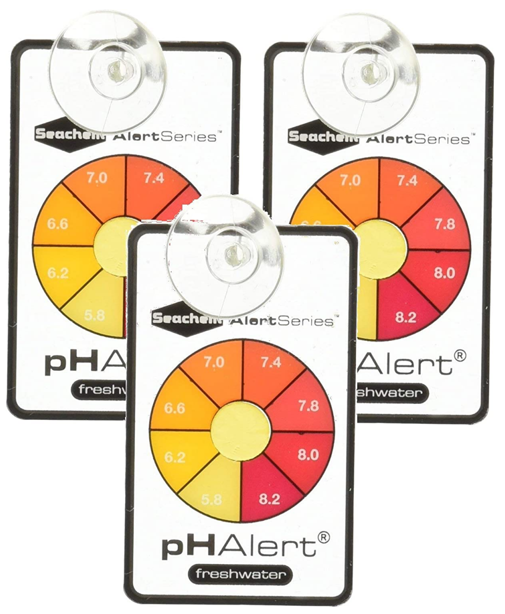
4.1.2 Prior Work/Solutions

FISHNOSH Automatic Fish Feeder for Aquarium

FISH NOSH NEW GENERATION 2022 12:39 7:00 16:00 22:00 360°	**Pros:** ● Works with multiple types of food ● Aesthetically pleasing ● Easy to use ● Fits most fish tanks **Cons:** ● No wireless connectivity to phone ● Battery Powered ● Food amount set manually by opening/closing the window ● Feeding schedule not fully customizable ● Does not include PH/ Temperature Sensor

Amazon Link	

Seachem pH Alert Devices

 Seachem AlertSeries 7.0 7.4 7.8 8.0 8.2 6.6 6.2 5.8 pHAlert® freshwater	**Pros:** ● Very simple to use ● Cheap **Cons:** ● Must be in the vicinity of the tank to know if PH is bad ● Only lasts 3-6 months ● Not customizable to different fish species ● Takes a lot of time to change (according to customer reviews) ● Can be inaccurate ● One time use, cannot be moved from tank to tank

4.1.3 Technical Complexity

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles

   The systems that we will have are as follows

   - React Frontend
     - Able to send REST calls to send and receive data
     - Written in React Native
   - Cloud-Based Backend
     - Sends calls to both the frontend and firmware
     - Stores data in a database
     - Deployed on Google Firebase
   - Firmware
     - Controls hardware sensors to properly obtain measurements and feed fish
     - Simple operating system to control hardware interface with backend
     - Scheduler to control device feeding and schedule backend updates
     - Non-volatile memory storage for many different device parameters and data
     - Connect to backend to send and receive requests through Wifi
     - Bluetooth connectivity for initial device setup
   - Hardware
     - pH Sensor
       - 0.1 pH Accuracy
     - Thermometer

        - 0.5 °F Accuracy
    - Motor
        - Gearing in order to achieve desired speed of the dispenser
    - Food enclosure
        - Able to store food and allow for easy dispensing
    - Tank Mounting Apparatus
        - Device is able to be secured to fish tank without requiring alterations to the tank itself
    - Custom Designed PCB
        - PCB is a custom design specifically made to fit in our device with the required parts

2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

    - User permissions in app prohibit monitoring/modification of unauthorized tanks
    - Data about fish tanks is stored and evaluated
    - The firmware and app are easy to connect to without advanced software knowledge
    - App will be user-friendly
    - Hardware will need to feed fish an accurate amount of food during each feeding
    - Backend, Frontend, and Firmware will follow good coding practices as well as IEEE standards

4.2 Design Exploration

4.2.1 Design Decisions

- Frontend Design Decision - Language to use for app development
    - As the user-facing application for our project, we needed to ensure that the frontend application of our project is easy to use, fast, and visually appealing. As an integral part of the project, we needed to decide on a language that would make it easy for us to develop the application as well as provide a good user experience for our clients.
- Backend Design Decision - Cloud service for app
    - There are a couple important considerations for cloud service providers and it is important that we look into one that will fit our use cases. For instance, we need to make sure that the service is always available as well as able to handle the amount of data that it needs to handle
- Firmware/Hardware Design Decision - Base Hardware for Device
    - Choosing the main processing power for our device left us with a couple different options of moving forward. We were able to choose between creating a solution based off an off-the-shelf Raspberry Pi design or developing our own custom chip-down design for our product. This is an important decision to make due to this being the main procession power for our device and this device will also be completing the backend to hardware connection. The chip-down design ends up being a more customizable and cheaper solution allowing us to add in more functionalities than we otherwise would have. While the raspberry pi would be much more intuitive to implement, we are confident that we have the knowledge to use the chip-down design. For these reasons, we opted for the chip-down design.
    - *Enclosure Design* - Choosing our differing material to use for our design to account for cooling and being able to be used by the water left us looking into differing plastics to 3D

print our enclosure. We also had to look into the 2 different designs of prototypes for the enclosure and see which fit our clients needs closer. Lastly, we had to make a decision on how the enclosures would hold the included hardware which led us to designing slots of each

4.2.2 Ideation

For the Frontend framework design decision, we came up with React Native, Swift, Flutter, Android Studio, and AWS Amplify for mobile development. We used a technique similar to rapid ideation to come up with various frameworks to develop and compare with. AWS Amplify is a part of the greater whole of the array of AWS services. The base framework allows us to develop everything from the backend to CI/CD and including the frontend; although it seems that the base application visuals and available libraries are left to be desired. In addition, there comes a cost factor on different payment tiers as well as a knowledge barrier.

Likewise, for Flutter, we are unfamiliar with the semantics of the frontend and CLI language for DART as well as knowledge on integration. Swift also had similar cons, where we are also unfamiliar with the language. Since we were aiming to develop an application for iOS devices. It seemed that it could have been a viable option if not for the complexity of implementation and non-understanding of the coding language/environment.

Thus leaves React Native to be considered as the last and primary option. React Native supports creating web applications for mobile devices which could also be deployed to a website with little to no changes. We are also familiar with the framework including Node.Js. React Native's has a wide range of libraries available for development which include being able to test and view the development application on devices through an external app.

4.2.3 Decision-Making and Trade-Off

Hardware Weighted Decision Matrix

	Cost	Part Availability	Ease of Implementation	Functionality Provided	Size	Total
Criterion Weight [0-1]	0.5	0.75	0.4	0.8	0.6	-
Chip Down Design [1-10]	5	7	4	10	9	22.75
Raspberry Pi [1-10]	4	7	8	7	5	19.05

Frontend Weighted Decision Matrix

	Ease of	External	Appearance	Knowledge	Porting to	Total

	Implementati on	Libraries Available	of Application	of Language	iPhone for Clients	
Criterion Weight [0-1]	0.5	0.5	0.6	0.8	0.5	-
Android Studio [0-10]	8	5	2	8	2	15.1
Swift [0-10]	5	7	9	2	10	18
React Native [0-10]	9	10	9	7	8	24.5
Flutter	7	6	6	2	7	15.2
AWS amplify	5	7	4	5	7	15.9

Backend Weighted Decision Matrix

	Ease of Implementation	Learning resources available	Scalability	Cost	Performance	Availability	Total
Criterion Weight [0-1]	0.20	0.15	0.10	0.25	0.15	0.15	-
Google Firebase [0-10]	8	7	5	8	7	7	7.25
AWS [0-10]	5	6	7	5	8	8	6.25
Microsoft Azure [0-10]	6	4	6	6	8	8	6.3

4.3 Proposed Design

4.3.1 Overview

Our proposed design consists of a wirelessly connectable device that is remotely accessible from anywhere with an internet connection through the use of an iPhone application. The main use of this device is to allow consistent monitoring of the temperature and pH levels of a fish tank, while also being able to set a feeding schedule in which the device will act out keeping the fish happy and fed at all times without any direct interaction with the fish tank.

There are multiple different components of this design that allow us to get the stated functionality. In order for the phone application to be able to connect to the tank device, we will be implementing a cloud-based backend service that will run out-of-sight to the user but will coordinate the connection between the phone application and the physical device on the fish tank itself. The phone application will communicate with the backend service through a cellular/Wi-Fi internet connection, which acts as a middleman between the feeding device and the phone app, sending any requests from the application to the physical device. This path also works in the reverse direction, allowing the monitoring information or device warning to be sent from the hardware device, over a Wi-Fi connection, to the phone application.

Context Diagrams

Diagram 1

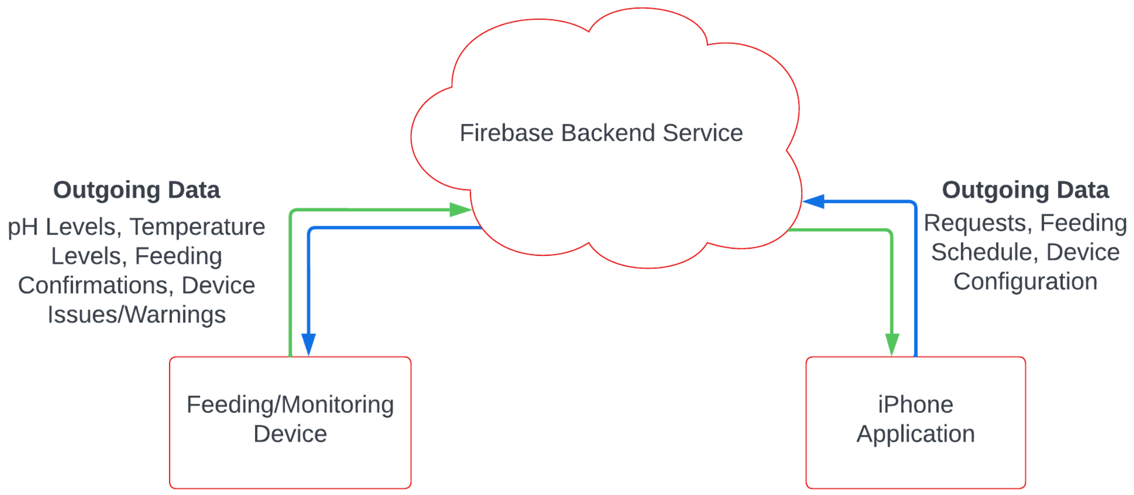
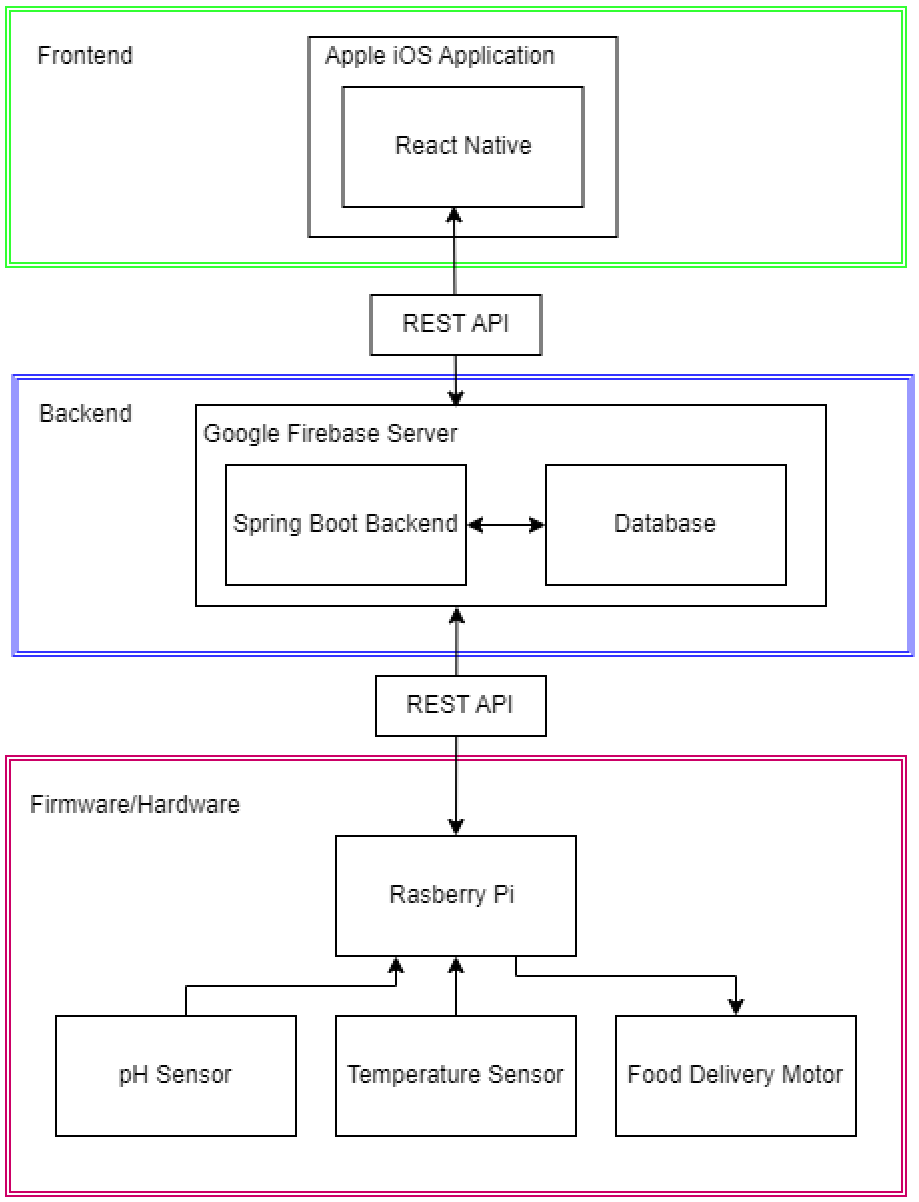

Firebase Backend Service
Outgoing Data
pH Levels, Temperature Levels, Feeding Confirmations, Device Issues/Warnings
Outgoing Data
Requests, Feeding Schedule, Device Configuration
Feeding/Monitoring Device
iPhone Application

Diagram 2



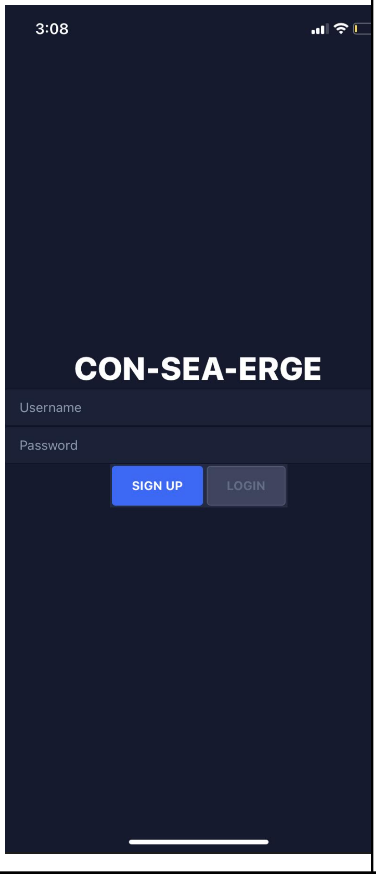
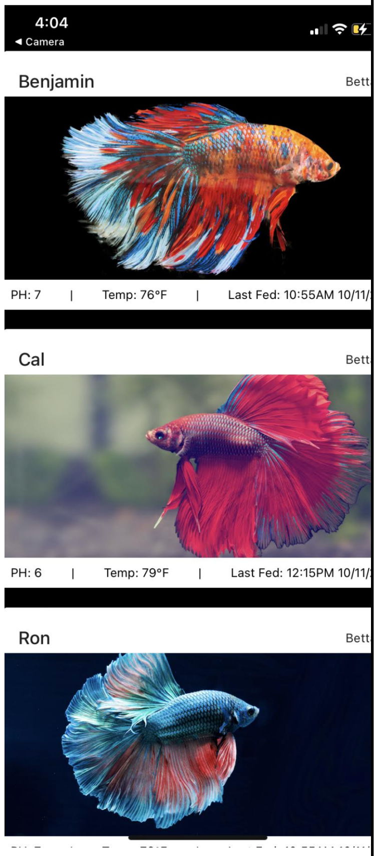
4.3.2 Detailed Design and Visual(s)

Frontend

The frontend of our application will be built using the react-native language along with the Expo framework to create and initialize the application. React-native is a supplementary version of React that allows for cross platform application development. Based in javascript, it makes it easy to develop good-looking user interfaces that can be ported to both ios and android, as well as websites. Expo is the supporting framework for react-native that provides a large set of tools and makes initialization of the application simple and easy to start.

Our application will be broken down into three main screens that will provide all of the information the user will need. On opening the application, the user will be prompted with a login screen to ensure that only our clients will have access to view their fishes' information. Once logged in, the main screen will show a list of cards of all of the users fish. These cards contain the name, species, and picture of the fish, and will provide quick information to display the current PH, temperature, and last fed time for each. This screen will allow the user to monitor all of the fishes at once to make sure there are no issues with the tank.

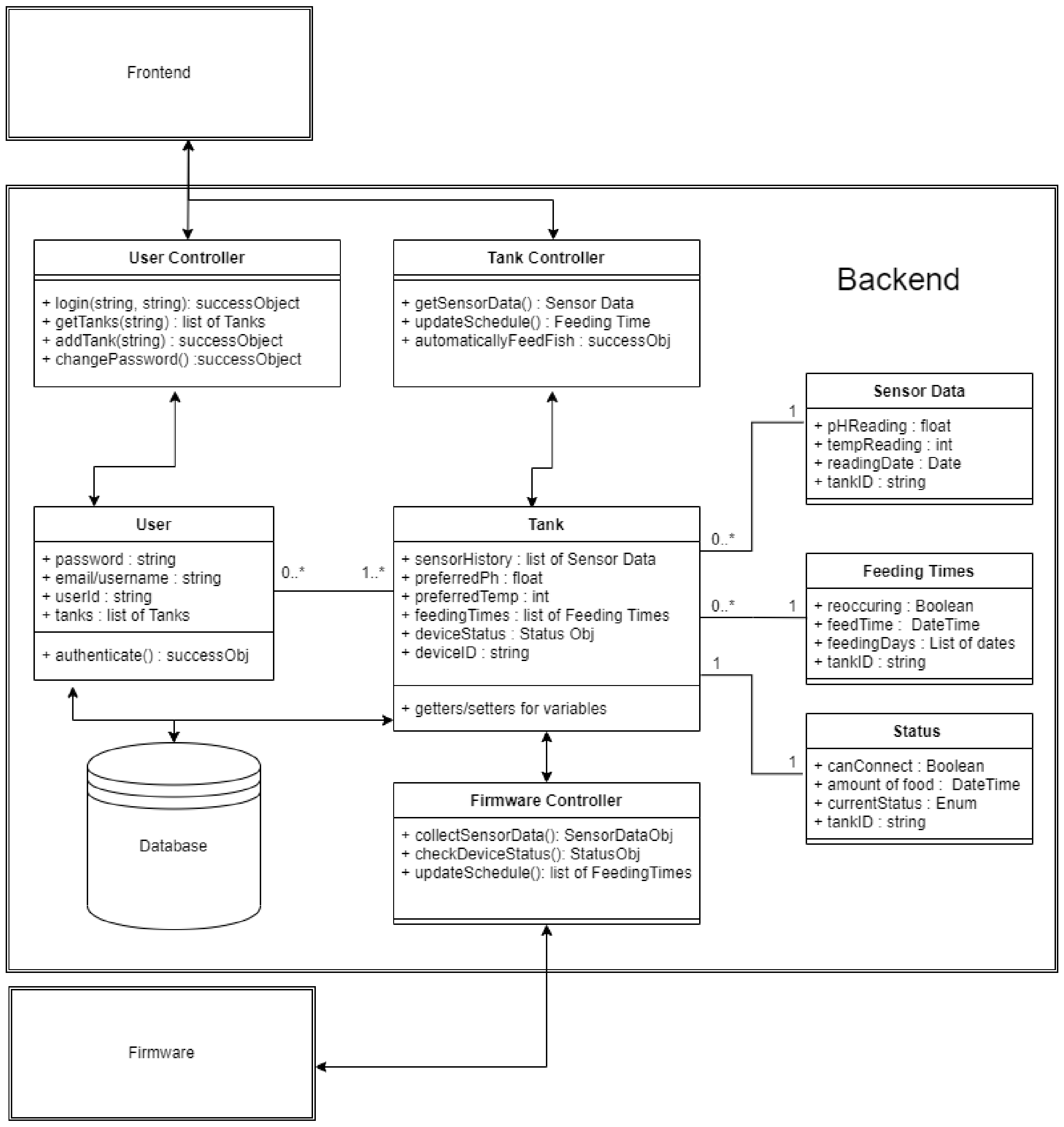
Clicking on a fish card will open a new screen dedicated to that specific fish. This page will display the PH and Temperature circular data components that are color coded on a scale from green to red to show whether or not the tanks are in good conditions for the fish. This page will also allow the user to manually feed the fish at any given time, adjust the feeding schedule, or edit the fishes' information.

Login	Fish Cards	Fish Components
 3:08 CON-SEA-ERGE Username Password SIGN UP LOGIN	 Benjamin Bett PH: 7 | Temp: 76°F | Last Fed: 10:55AM 10/11/ Cal Bett PH: 6 | Temp: 79°F | Last Fed: 12:15PM 10/11/ Ron Bett	

Backend

The Backend will be a Google Firebase Backend as a service and most code will be kept in the front-end. There will be two main class components with a User class and Tank class. The User class will handle all of the user data such as login and each user will have the ability to add and connect to new tanks. The tank objects will contain information pertaining to each feeder device which includes the history of the sensor data, a list of feeding times, and the status of the tank. Data from the User and Tank classes will be stored within a database

The system will have 3 front-facing controllers. Two for the frontend to interact with and one for the firmware to interact with. Each controller will have a form of authentication for the messages that get sent to them in order to prevent requests from being made that shouldn't be.

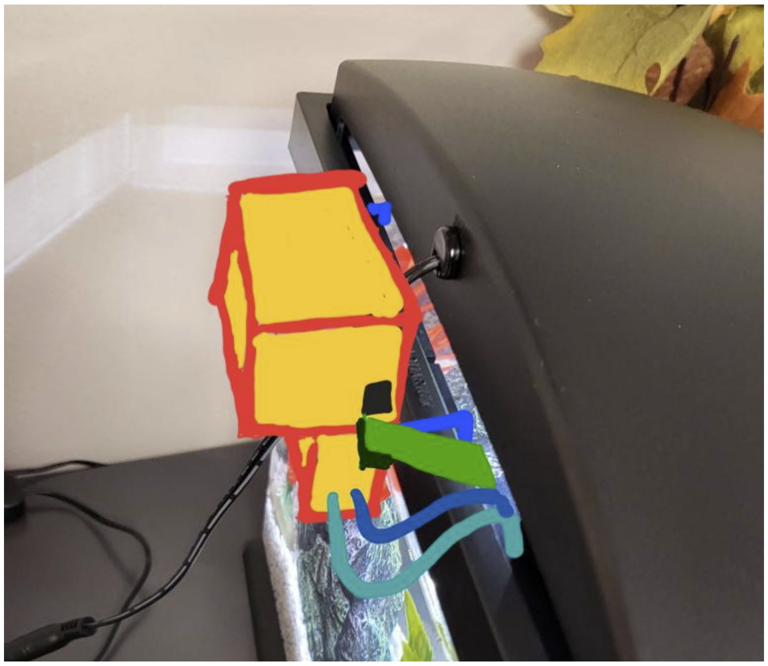


Frontend
Backend
User Controller
+ login(string, string): successObject
+ getTanks(string) : list of Tanks
+ addTank(string) : successObject
+ changePassword() :successObject
Tank Controller
+ getSensorData() : Sensor Data
+ updateSchedule() : Feeding Time
+ automaticallyFeedFish : successObj
Sensor Data
+ pHReading : float
+ tempReading : int
+ readingDate : Date
+ tankID : string
User
+ password : string
+ email/username : string
+ userId : string
+ tanks : list of Tanks
+ authenticate() : successObj
Tank
+ sensorHistory : list of Sensor Data
+ preferredPh : float
+ preferredTemp : int
+ feedingTimes : list of Feeding Times
+ deviceStatus : Status Obj
+ deviceID : string
+ getters/setters for variables
Feeding Times
+ reoccuring : Boolean
+ feedTime : DateTime
+ feedingDays : List of dates
+ tankID : string
Status
+ canConnect : Boolean
+ amount of food : DateTime
+ currentStatus : Enum
+ tankID : string
Firmware Controller
+ collectSensorData(): SensorDataObj
+ checkDeviceStatus(): StatusObj
+ updateSchedule(): list of FeedingTimes
Database
Firmware
0..*
1..*
1
0..*
0..*
1
1
1

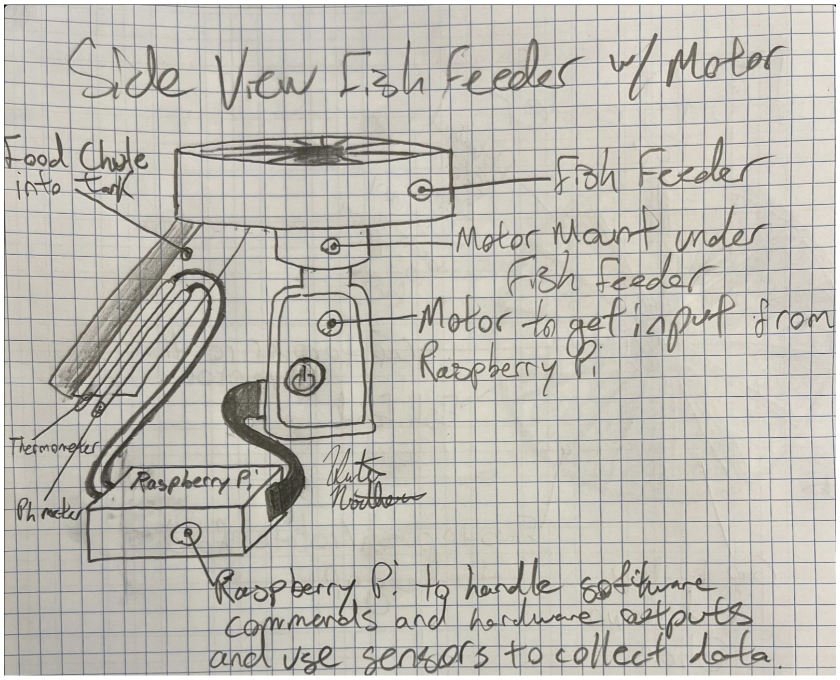
Hardware/Enclosure

The enclosure is based on our drawings and will be designed in Solid Works / CAD to then be 3-D printed to fit our components such as the probes, firmware, and food. We are basing our designs off of 2 different enclosures (A and B shown below). Enclosure A will use a mechanized corkscrew method to push food out of the enclosure down the chute into the fish tank allowing the number of rotations to be controlled and putting out a controlled amount of food while still holding and enclosing all of the components of our design. Enclosure B will use a sectional rotator to move a controlled amount of food around which then drops down the chute into the fish tank to feed the fish and keep all the components consolidated.

Enclosure A:



Enclosure B:


Side View Fish Feeder w/ Motor
Food Chute into tank
Fish Feeder
Motor Mount under Fish feeder
Motor to get input from Raspberry Pi
Thermometer
Ph meter
Raspberry Pi
Raspberry Pi to handle software commands and hardware outputs and use sensors to collect data.

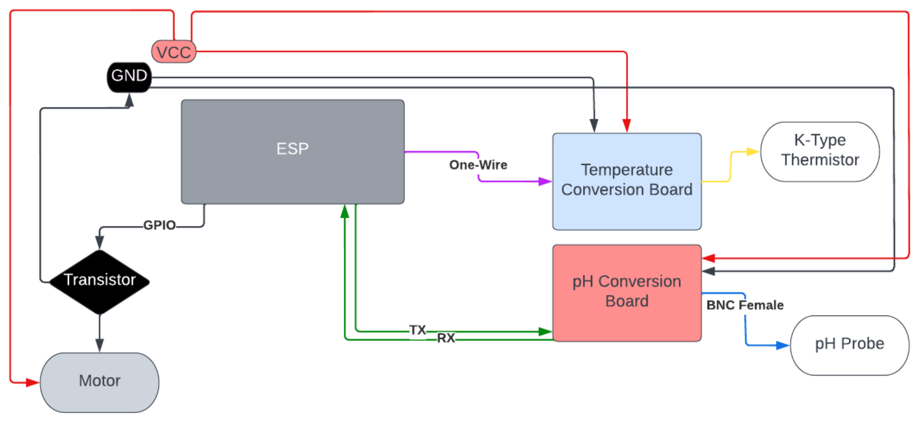
Hardware/Chip-Down

The PCB hardware that we are using in this project is based on a custom chip-down design. This allows our team to select very specific components for the functionalities that we require to be included in the hardware of the device itself.

We will be including the components listed below:

- pH Probe
    - Consumer-Grade pH probe
    - This type of probe was chosen because of its price compared to higher-end grade probes, as well as the accuracy of 0.01 pH is above our requirements.
- pH Conversion Board
    - Module: EZO-pH
    - This conversion board allows us to simply take readings from the pH probe and translate these readings into meaningful pH values quickly.
- Temperature Probe
    - K-Type thermocouple
    - K-Type thermocouples are less prone to oxidation and acidity than their counterpart of J-Type, which is important for a water application
- Temperature Conversion Board
    - Module: Adafruit-1727
    - This conversion board allows us to simply take readings from the thermocouple and translate these readings into meaningful temperatures quickly.
- Wi-Fi/BLE Enabled MCU
    - Module: ESP32-C3-MINI-1
    - This component will enable the device to have an internet connection through Wi-Fi while as well supplying the ability to connect to a phone over the BLE service.
- DC Motor
    - Compact, DC motor geared for torque
    - We do not need a quick motor as the dispensing of food has to be relatively slow in order to dispense the correct amount of food. As well, a motor geared for torque over speed will jam less frequently on foot pellets than compared to a speedier motor.

These hardware components will be connected in accordance with the following diagram:



This device will also feature an LED for warnings and direct information about device operation. (This is not pictured in the diagram above)

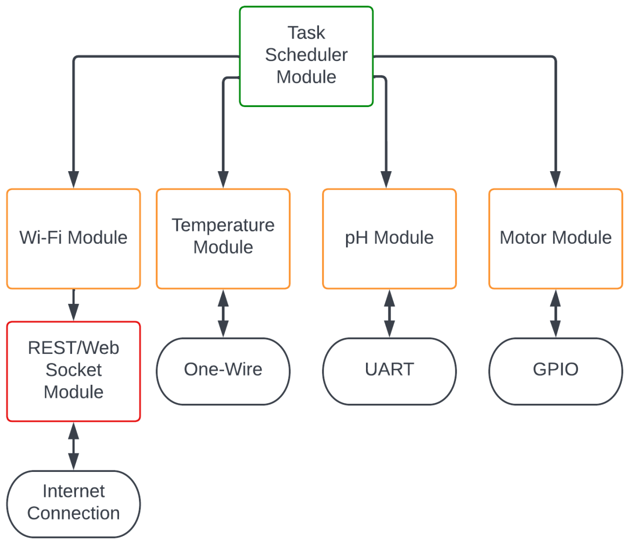
Firmware

The firmware will include a modular design to allow for easy portability of the code and allow for simple changes down the road, if necessary.

The main modules in the firmware are listed below:

- pH Module
    - Requests and receives pH information from the pH conversion board
    - Communication Protocol: UART
- Temperature Module
    - Requests and receives temperature information from the temperature conversion board
    - Communication Protocol: One-Wire
- Motor Module
    - Controls the actuation of the motor
    - Communication protocol: GPIO
- Wi-Fi Module
    - Allows ability to connect to internet
    - Receives/sends requests to the backend service
- REST/Web Socket Module
    - Configures and creates requests to the backend service in the specified formats
- Task Scheduler Module
    - Main control module that handles the timing of the device application and the initiation of device actions

The class hierarchy is displayed in the diagram below:



4.3.3 Functionality

Once our design is fully functional, the feeding mechanism will be able to feed fish in the tank on a specified schedule and report pH and temperature data from the tank to the app. The functional app will allow users to log in, view/set feeding schedules, and monitor pH and temperature levels of specific tanks.

A user would log into the mobile application and be able to view all the tanks that they monitor. They will be able to see the current pH and temperature with the current reading. Our device will take these measurements every 5 minutes then send that data to the backend and our backend will send that to the frontend to be displayed to the user. If the pH or temperature get to a dangerous level for the fish, a notification will be sent out to the user to notify them. A feeding schedule for each tank will be able to be set and viewed by the user. Whenever the schedule is updated, the backend will send that data to our device and it will save that to the non-volatile memory to be used from then on. If the internet connection disconnects on our device, the schedule will still be acted upon as it is still in the memory. Using the internal clock, we can do calculations to continue the schedule.

For the enclosure our product will have the functionality to consolidate all of our components and to push food into the fish tank in a controlled manner in regards to serving size and component structure.

4.3.4 Areas of Concern and Development

Our current design encompasses all the requirements and user needs outlined by our clients. They emphasized some key aspects that we took into account such as it being a mobile application, having the ability to set and view a feeding schedule when they are out of office and being able to monitor both pH and temperature remotely. As long as there aren't any immovable blockers, our design completes the project's requirements.

Concerns

Will our connection from frontend to backend and backend to firmware work correctly?

This was an initial concern for us going into this project. If we are not able to solve a problem, it would be detrimental to the success of our project. Over the course of implementing all the separate sections, we researched how to connect all three and it looks like we will be able to do it just fine using our experience in the field.

Will our development board be able to handle staying on schedule without WiFi connection? Will it stay on schedule after it reconnects?

Our clients made it clear that this should be a goal to have it stick to the schedule while being disconnected from the WiFi and also get back on schedule after reconnecting. Looking deeper into all the components on the Development board, we will be able to keep track of time using the internal clock and comparing that to the last time that the fish was fed. While this might be difficult to implement, we believe that we are capable of accomplishing this as the firmware team is very experienced in low-level programming.

Will our backend hinder the performance of the systems?

The main concern here is that the backend needs to be constantly active so that it can send out updates to the device from the mobile act. The backend works as a middle ground between the two front-facing devices and it is imperative that the backend doesn't hinder the processes of either the mobile app or the physical device. This means that if the backend is down you will be unable to view data from the firmware on the mobile app. We believe that we will be able to accomplish this by deploying the app onto a cloud service to make it constantly available for the users. Once this is done we can do testing to make sure that the data flow is fast enough and consistent enough to make the product a pleasant experience for the user.

How to roll out an application for iOS devices?

Currently we have thought of several ways to get the app to our clients. First thought would be to release the application to the App store but it is seemingly unlikely as the user base is quite small and limited with the App store having several restrictions to be published and required ongoing maintenance as well. Our second thought would be to directly install the application through a devtool but have to do extensive research to learn how to circumvent certain restrictions.

Questions for TAs, Clients, and Teachers

How fast would the client like changes to the device to go into effect?

If an iPhone application isn't possible, would a web application suffice? (More research and thought needed)

4.4 Technology Considerations

SpringBoot

SpringBoot is a fairly popular backend framework and it is used in a lot of places to create microservices. The framework is open source which means we do not need to pay to use the framework and it has many resources available for us to learn more about it. One of the main reasons we went with this framework over others was due to our team having a higher exposure to the language of Java in our courses as well as some of us having professional experience using Java Spring Boot. Other alternatives could be something like Ruby on Rails or .Net, however, these frameworks are written in other languages that the team is less familiar with.

Firebase

Firebase is one of the leading backend frameworks nowadays, and it pairs great with our app because of its lengthy list of features and generous free package. Firebase's free tier allows for a large number of authentications, database hits, and cloud uploads/downloads, so the fact that our app will require a relatively small number of each of these items makes the free tier a great option. In addition, one of the features of Firebase is that there is no need to set up or connect to an external server, as Google handles all of the hosting on its own. One of the aspects of Firebase that has proven to be challenging so far has been getting it connected to our app since we are using React Native for the frontend. This is particularly challenging because Google has simple setup procedures for frameworks such as Flutter, but there is a bit more to work through when connecting to a React Native app.

React

React is heavily used throughout the world. There are plenty of libraries, documentation and support regarding the framework. By using React, we can use the various libraries that are widely available via open-source. Primary reason that we chose React is that our team is quite familiar and comfortable with React using JavaScript or TypeScript. Alternatives include using AWS Amplify to create the application or Flutter but these tools/environments are unfamiliar territory to the team.

Chip-Down Design/Firmware

One of the main reasons that we went with a "chip-down" design instead of moving forward with an off the shelf solution for the PCB, such as a Raspberry Pi, was that the off the shelf solutions were not readily available and in stock. Deciding to go with a more customizable solution allows our product to be specifically tailored to the purpose that we are designing it for, and it also allows us to decrease the amount of unnecessary hardware in the device.

The selection of the ESP as the main MCU that will be running our design allowed us to easily know which language and IDE to be using since the ESP module code is written in C and uses an IDE tailored to that device. With a Raspberry Pi solution we would be able to use different programming languages, but being that C is one of the lowest-level full programming languages, this would allow our code to be as quick as possible allowing for the best performance and configurability.

The main trade-off with moving forward with a chip-down design is that it increases the project complexity. Raspberry Pi's are extremely easy to work with and have a lot of community support online if issues arise. That main benefit of the off the shelf solution allowed us to consider that as a viable option over the rest of the benefits of the chip-down design. After weighing the pros and cons of the two, it seemed going with a chip-down design would be better for the project as a whole.

Enclosure

The enclosure is only limited by Solid Works specifications and 3-D Printing capabilities but both technologies are heavily researched and time-tested so in terms of enclosure design we will be able to produce a product that meets our specifications.

4.5 Design Analysis

For the frontend, we have some simple components created such as components for visually representing pH levels and temperatures. We created a card component representing fish tanks and a login screen. Currently we have dummy data to show on the frontend while we work on connecting to the backend; waiting for the firmware to be able to provide data to the backend as well. We have not started testing components or APIs as of yet but plan to do so when we can run some test calls. We also plan to add visual unit and integration tests for our frontend components in the future. We also plan to implement a CI/CD pipeline to ensure our deployment is successful and that each consequent update would not break the application. For the backend, we have researched potential backend services and decided that we will use Google Firebase to host our backend. Our project is setup in Firebase, and we are currently working on integrating it to the frontend so that information can be passed and tested. The current issue that we are working through revolves around assigning a frontend type to the project so that connections can start to be made. As for the firmware, we have figured out how to send code to our Development board using the Arduino IDE using C as the programming language. We have been able to connect to the WiFi using the Wifi module. We are currently looking into handling REST api calls after that, using Postman to test that. Later, we will actually connect the board to Firebase once the Backend gets fully set up. As for the sensors, this week, one person of our firmware team will focus on that while the other person on the team will focus on the WiFi aspect. This week, we plan on working on getting the sensor code implemented where it will output both the temperature and pH level. Overall for the firmware aspect, we have not run into any issues as of now.

5 Testing

5.1 Unit Testing

Frontend

Static Analysis - Eslint has been set up in the frontend application. Eslint will enforce consistent code styling, warns about unused code, removes undefined variables, and more. Adding Eslint to our project makes the code more readable and sets up a baseline of standards, allowing consistency between developers to adhere to.

Type Checking - By default, Javascript is an untyped language, meaning that it is possible to pass incorrect object types to functions that cannot process them at runtime. We added type checking to our project to ensure that the construct we are passing to a function matches what the function was designed to accept. This prevents unexpected errors and behaviors when running the application.

Jest Unit Tests - The frontend code is verified by unit tests, which cover each component. Jest Unit Tests are designed to ensure that each part of a component functions properly.

Firmware

Unit Tests - We will have unit tests in place for the different functionalities that we have including pH, temperature and WiFi. For the WiFi aspect, we will test possible situations that could happen such the WiFi going out then coming back on and verifying that our code will detect the disconnection and will connect once it comes back up. For temperature and pH, we will test this using various liquids of differing temperature and pH. We will verify that the output that we get from our device is similar to what we expected.

5.2 Interface Testing

Firmware

The hardware on the device itself creates an interface, through an RGB LED, that will be utilized by the user to quickly notice warnings or issues happening on the device. These warnings will be displayed by utilizing different colors set on the LED for each warning. This interface will be conducted through many different tests as there are many warnings the LED will display. These tests will need to be manually actuated for these issues to occur and to be tested.

- Wifi Disconnection
- Connect the device to a wifi source and then remove the power from the wifi system.
- Confirm the LED changed to the correct warning color
- Motor Jammed
- Start the feeding process on the device and hold the motor, disallowing it from being able to spin correctly. This would simulate jamming the motor.
- Confirm the LED changed to the correct warning color
- Doomsday Mode
- Connect the device to a wifi source and then remove the power from the wifi system.
- Remove the power source from the device and then reinsert the power source back into the device to power it back up
- Confirm the LED changed to the correct warning color
- Normal Operation
- Setup device as expected. (i.e. connected to wifi and power)
- Let device run for an extended period of time

- Ensure the device LED continues to display the correct operational color throughout the whole testing process

<u>Frontend</u>

The frontend will display components that are interactable to adjust variables such as scheduling, visual notices for pH levels and temperature. There will also be warnings that pop up or skeleton loaders that should replace components while waiting for data. These types of tests require manual intervention/data manipulation to occur and would allow us to check all cases.

- Main screens
- Should be the main default fallback page and easily accessible
- Go through various navigation steps and testing
- Loading data
- Components should load skeleton components for items whilst waiting for data/compilation
- Should not break the user interface if there is either no data or faulty data
- Notifications
- Should show up based on conditions set by either user or system and show relative warning/notification
- Show up as non-invasive and easy to dismiss

5.3 Integration Testing

<u>Backend-Frontend Integration</u>

Using google firebase we can interface with the frontend directly. Testing for the frontend to the backend will require making integration tests within the React frontend that will make calls to the firebase backend that will be able to retrieve data and return it to the frontend. Tests will be successful if the Backend-Frontend round trip receives the correct data that was sent to the backend.

<u>Backend-Firmware Integration</u>

Initial integration tests will be performed using a tool called Postman. This tool will allow us to independently test the REST API calls to the backend whilst not utilizing the backend to ensure that issues are isolated solely to the firmware on the device. Once this testing is completed we will integrate the firmware and backend by directing all API calls from the device to the backend service. Tests will be written to make a specific call to the backend and wait for a response from the backend service to confirm that the expected result was returned.

5.4 System Testing

<u>Backend-Frontend Integration</u>

For backend to frontend connection we will leverage a couple integration tests outlined in section 5.3 to make sure that full run-throughs of data are being correctly fired off. We can utilize test databases and separate connections through Firebase to give an accurate representation of what the system may actually do while also keeping important user data separate.

<u>Backend-Firmware Integration</u>

The integration tests used for the backend-firmware will be a vital part of the system tests, as the device needs to be integrated correctly into the backend for the whole system to be functional. The integration test is described in section *5.3.*

Frontend Unit Tests

Frontend unit tests will tell us there are components that are affecting functionality. In addition, snapshot tests will also be performed, allowing us to check if the components are rendered correctly. With unit testing and snapshot tests, debugging would be easier and verifies that the system is working as expected. These tests are described in sections 5.1, 5.2, and 5.5.

Firmware Units Tests

Firmware unit tests are important for system testing because these tests follow the requirements of the project for features available within the device itself. These will need to be tested to ensure compliance with the requirements and that when all information is received by the device it is able to handle the actions correctly and without error. These tests are described in section *5.1.*

5.5 Regression Testing

CI/CD Pipeline - Our CI/CD pipeline in Gitlab is configured to run all of our test code when making a pull request. This will ensure that when we add new code to our project, what we have added does not break existing features. New features in our application should build upon old functionality without causing it to fail.

Frontend

Snapshot Tests - Snapshot tests take a screenshot of each screen of the application and compare it to a new screenshot when you push to gitlab. This ensures that when you make changes to the UI, what gets merged to the main branch is what you expect. If a snapshot differs from what it was previously, it will warn the developer and the developer can update the snapshot if the changes were expected otherwise they will know that they have created a bug.

Firmware

Regression testing for the firmware is important as a lot of changes can have adverse effects on other parts of the code itself due to memory constraints and register crossovers. The firmware of our device will benefit from the use of the CI/CD Pipeline to ensure that each new code version uploaded is in compliance with all the previous release tests.

5.6 Acceptance Testing

Frontend

E2E Testing - End to end testing verifies that our application runs as expected from the users perspective. This style of testing is done by building the release configuration of the application and allowing full control of the UI through the testing interface. We will be using the Detox package to do this kind of testing in our app. E2E tests will confirm that all functionality such as button clicks, user input, authentication and more are working properly.

Backend

Data storage and passing must be fully functional and thoroughly tested in order to ensure that the backend as a whole functions properly. Acceptance testing would include adding sample data into our databases and simulating transfers to both the frontend and the firmware. Once the data is able to be properly stored and accurately received by the frontend and firmware, our acceptance tests for the backend will be complete. The client would most likely not be too involved in this portion of testing since it involves a lot of behind-the-scenes work that is not constrained by any requirements.

Firmware

Acceptance testing of the physical device features will be performed with physical tests as the device itself interacts with the world. We will perform tests to ensure that the device complies with the requirements documents. PH testing will be done through use of pH solutions at three different set pH levels. Temperature testing will be done using a store-bought thermometer and varying temperatures of water to confirm that our device is getting the accuracy that was required. The motor will be tested thoroughly within the enclosure of the device to ensure that each section of food dispensed is in an accurate amount to what was set and is consistent with every dispense. LED will be tested as described in section *5.2*.

5.7 Results

Our results would be considered successful in implementation if all unit tests, integration tests, interface tests, system tests, regression tests, and acceptance tests are completed successfully. As well, these tests should ensure that they cover the entire scope of the project to guarantee that each module is being tested thoroughly.

The results of our tests are the overall integration and compatibility measures of our components being built into a singular product. With these measurements giving insight into how our client wants their product to function and fulfill their needs we are able to tweak and change our product based on our tests to further satisfy the aspect of the product our client focuses on the most.

From all of our testing throughout the different aspects of this document we have found ways to improve our product but overall that our product continues to bridge the gap between our clients problem and our solution.

6 Implementation

Front-End Software:

Beyond our design plan, we have several screens that allow the user to login the application and to view basic dummy information as we continue to connect the frontend, backend, and firmware together. There are several issues that we are working on such as dealing with certain permissions on iOS and ensuring that designs are cross-compatible across operating systems. The major features have yet to be connected with the other moving parts but the elements and components are still currently works in progress. There is still quite a bit of testing to be done to ensure that functionality is as expected. Moving forward the frontend will also need to have some of the finer details of UI/UX polished as well as add other features that are either quality of life or secondary tasks from the client.

Back-End Software:

As of right now the Google Firebase has been connected to the frontend and we have started implementing the database as well as authentication. From here we need to expand out the features that the backend has as well as getting a full round trip done for the data. We also need to work on interfacing with the physical device and how we plan to do schedule updates. We also need to iron out how our data structures are going to look which will be a discussion between the frontend and the backend team.

Firmware/Hardware:

Moving forward the firmware has many steps to be completed in order to finish the functionalities of the device. Moving into the next phase of this project we will have to implement the motor functionality into the firmware design in order to ensure that we are able to dispense food accurately. This requires finding a motor that will function in our design and with our hardware. As well, we will need to implement a task scheduler for our device to be able to handle incoming commands from the Firebase backend. This will allow us to act upon actions that are requested by the user from their phone application. Functionality to be updated after those items are done are just to ensure that our device functions properly and continuously no matter the conditions that come into play while under real-world testing.

Enclosure:

In implementing the enclosure we will continue resizing and shaping the enclosure as we finalize the components we intend to use. For now we have a base design in CAD of the different prototypes and will work on printing them with the motors and boards we have selected thus far.

7 Professional Responsibility

7.1 Areas of Responsibility

NSPE	**IEEE**
Work Competence	IEEE covers work competence in great detail and includes additional items such as the importance of criticism, treating people with respect, and improving competence. One of the major differences that I noticed is that IEEE focuses more on improving technical competence as opposed to ensuring that one has enough competence to complete a job.
Financial Responsibility	Financial responsibility is not heavily covered in IEEE's standards, but it does mention the fact that bribery should always be rejected. This is the main difference between the two, as NSPE covers this section in much greater detail.

Communication Honesty	IEEE’s standards also emphasize the importance of truthful communication with stakeholders. One difference here is that IEEE’s code also discusses unlawful conduct and bribery as items to prevent and avoid.
Health, Safety, Well-Being	IEEE emphasizes that health, safety, and well-being are of utmost importance and also includes the importance of privacy. One major difference is that IEEE notes that it is imperative to make endangering factors well known to the public as well as the environment.
Property Ownership	As mentioned above, one item that is brought up in IEEE’s standards is the importance of personal. This relates to property ownership because privacy refers to the way that one’s belongings and information are respectfully left alone. One difference in this section is that IEEE’s standards do not specifically reference the protection of ideas as NSPE does.
Sustainability	As mentioned above, IEEE’s standards highlight the importance of disclosing factors that may endanger the environment. However, unlike NSPE’s standards, IEEE lacks information about protecting natural resources.
Social Responsibility	Social responsibility in IEEE is very similar to NSPE, as both mention that the improvement of society is paramount. The major difference in IEEE’s code is that it puts a higher emphasis on utilizing technology as a means to benefit society.

7.2 Project Specific Professional Responsibility Areas

Work Competence	High - We have nearly completed a thorough planning process that has allowed us to compile the highest-quality blueprint possible for our project. In addition, we have already made strides in creating the frontend and getting components connected through the backend.
Financial Responsibility	Medium - In total, each device has cost us a little over $100, which is right around what we expected for a device like this.

Communication Honesty	High - We have made it a priority to communicate each aspect of our planning process with our stakeholders so that they are up to date with our proposed solution and can provide us with feedback as we work.
Health, Safety, Well-Being	High - We have taken into consideration the health and safety of the fish in the tanks that we will be adding our devices to, and we have made changes to our design to reflect this. For example, we plan on protecting our dev boards with an enclosure so that no water is able to harm the boards and the fish are able to stay safe.
Property Ownership	High - In planning our device, we initially threw around the idea of cutting holes in the lid of the fish tank. However, out of respect for the client's property we scrapped this idea. Our design is able to attach onto the existing tank without making any physical modifications.
Sustainability	Low - This area of responsibility does not apply to our project as much as others because there is not much of a way for our feeding device to affect the environment or natural resources.
Social Responsibility	Medium - Although this has not applied to our project on a large scale yet, we are benefitting the ISU accounting offices with our product. Our product will help our clients to save time and make feeding their fish and cleaning their tanks less of a hassle.

7.3 Most Applicable Professional Responsibility Area

Work Competence: Our team has done a great job in the area of work competence up to this point in the project because we have made it a priority to work toward producing the highest quality product possible. In addition, through the planning process, we have set deadlines for ourselves, communicated thoroughly with our stakeholders, and acted in a professional manner, which have all contributed to our success thus far. Specifically, we already have parts delivered, a functional frontend screen, a connected backend, and a wifi-connected board. These are just a few of the examples of how timely and efficient our team has been so far this semester, and I envision that continuing through the rest of the year.

8 Closing Material

8.1 Discussion

Our project, as a whole, is designed to fully meet the requirements of our users. We carefully crafted our requirements document, with our clients input, in order for our device to meet each and every need that they would like to see in the final product. We then took that requirements document and started designing our product ideas around the main purpose of ensuring that we are able to fulfill every requirement given to us by our clients. This allows our product to be a client-driven solution which will best meet their needs as well because throughout the entire design process we are getting input from the clients to ensure that our idea is matching their expectations.

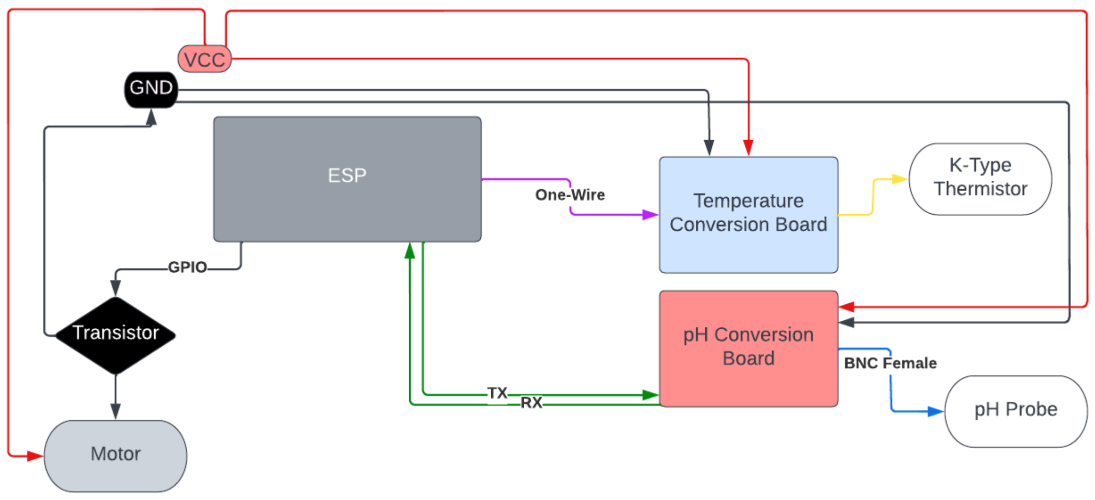
8.2 Conclusion

Throughout the first phase of this project, first semester, we have been able to complete many of our goals that we had created in the beginning. Our goals for the first phase of this project were to create a hardware device that is similar to a final production interaction, initiate a backend service that is able to communicate with both the frontend and the device, generate a preliminary design for an enclosure in CAD, and send a round-trip message from the device to the frontend application. We figured that these goals would put our product in a good place to pick it up in the second semester, as the completion of these goals shows the main functionality throughout each section of this project.

We have been able to complete all of those goals this semester effectively. Our team kept track of progress for these goals throughout the semester by having a “standup” meeting every Sunday to record each section’s progress towards the goals and to alter each person’s tasks accordingly so we were on track to complete this by phase end. At this point in the project we have a functional device that is able to communicate with the backend and frontend applications to perform the actions necessary. This functionality will be augmented greatly in our second phase activities.

8.4 Appendices

Device Connection Schematic



8.4.1 Team Contract

Team Members:

1) Devin Milligan__________________ 2) Ethan Peterson____________________

3) Drake Dodson	4) Brian Tran
5) Hunter Northern	6) Ryan Hickok
7) Joshua Van Drie	8)

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Sunday, 11am-12pm, virtual.
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. email/discord and in-class
3. Decision-making policy (e.g., consensus, majority vote):
    a. Majority vote on the decisions we make but to also compromise when faced with differences.
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Using minutes spreadsheet in folder to record the meeting minutes.
    b. Person in charge: Anyone in the meeting, default Brian

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. To be at every meeting unless noted otherwise to team members.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. To have a base level of work completed by each team meeting to provide updates regarding if help or clarification is needed.
3. Expected level of communication with other team members:
    a. To provide progress updates and notify team members of any problems or blockers that they come across or need help with.
4. Expected level of commitment to team decisions and tasks:
    a. State your position on decisions and discuss as well as contribute to team tasks with at least 1 or 2 things.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

    Brian - Frontend Lead Design
    Josh - Frontend Support/Testing
    Hunter - Hardware Lead Design
    Ryan - Backend Lead Design
    Drake - Backend Support/Testing
    Devin - Firmware Lead Design, Client Correspondent
    Ethan - Firmware Support/Testing

2. Strategies for supporting and guiding the work of all team members:
    a. Consistent communication of progress and properly delegating work in addition to providing assistance to tasks that require it.
3. Strategies for recognizing the contributions of all team members:
    a. Work overview at progress checkpoints and at team meetings.

Collaboration and Inclusion

Drake - SE, full stack development using React for frontend and Java for backend. I also have experience working with C/C++.

Brian - SE, front-end developer with experience in React, Type/Java-script, nodejs and with slight knowledge in java.

Hunter - Cpr E, a lot of embedded systems background with hardware structure design as well, knowledge in C, Java, Python, and a few other languages.

Ethan - CPRE, A lot of experience of writing test code in Java through internship. Low level programming experience from CPRE classes in C.

Devin - SE, A lot of experience in developing embedded devices, mostly firmware design, but experience in hardware design as well. Have programmed applications in C, C++, Java, and more languages.

Ryan - SE, experienced in Java and C, some experience working with frontend development, interested in working on backend

Josh - CPRE, experience in Java, React and C, looking to work mostly on the frontend to develop my skills. Also have experience with 3D design in autocad for the physical parts of the project.

1. Strategies for encouraging and supporting contributions and ideas from all team members:
    a. Being sure that no one gets outspoken and when we have ideas we share them with the team and ask those that aren't contributing as much for their perspective.
    b. Keep shared documents for brainstorming ideas
    c. Communicate with team members and ask for advice when we are having blockers.
2. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
    a. Talk about the problem, in a civil manner, to the team during our scheduled meetings, or other times confirmed with the team, or talk to our advisor to help address the issue if they are uncomfortable talking to the team members.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. To create a wonderful project to feed the fish
    b. To get exposure working on a team to create an end product
    c. To gain experience working with a 3rd-party sponsor/client
    d. Gain skills and experience in keeping good design documentation
2. Strategies for planning and assigning individual and team work:
    a. Delegate work evenly based on complexity, team roles, and time constraints.
    b. Delegation of work will be done during the Sunday meetings
3. Strategies for keeping on task:
    a. Consistent communication and progress check-ups.
    b. Progress check-ups will be done during the Sunday meetings

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. Discuss them during team meetings and attempt to resolve them as a group.
2. What will your team do if the infractions continue?
    a. Communicate with the team advisor/teacher about potential problems.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Devin Milligan____ DATE9/18/2022___________
2) Ethan Peterson____ DATE 9/18/2022__________
3) Drake Dodson____ DATE 9/18/2022__________
4) Brian Tran____ DATE 9/18/2022__________
5) Hunter Northern______________________________________ DATE 9/18/2022__________
6) Ryan Hickok____ DATE 9/18/2022____________
7) Joshua Van Drie_______________________________________ DATE 9/18/2022 ___________