Feederal Reserve

Design Document

sdmay23-43
Client: Campus Organizations Accounting Office
Advisor: Dr. Nick Fila

Team Member - Role
Jack Croghan – Backend Software
Sarah Degen – Circuit design/note taking
Melanie Fuhrmann – Frondend app dev/client contact
Bella Leicht – Security
Adan Maher – Security/organization
Brandon Mauss – Component design
Nathan Paskach – Firmware design

sdmay23-43@iastate.edu
https://sdmay23-43.sd.ece.iastate.edu

Revised: 4/29/2023

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

Summary of Requirements

- Dispense three pellets
- Measure pH within 0.1
- Measure temperature within 1℉
- Connected to an app
- Unobtrusive design

Applicable Courses from Iowa State University Curriculum

The following is a list of all Iowa State University courses whose contents were applicable to our project.

- EE: 201, 230
- CPRE: 230, 231, 288, 388
- COMS: 309, 363
- ENGL: 314

New Skills/Knowledge acquired that was not taught in courses

- 3D modeling
- HTML coding
- Flutter
- iOS apps
- Dart

Table of Contents

1 Team

1.1 Team Members

Jack Croghan

Sarah Degen

Melanie Fuhrmann

Bella Leicht

Adan Maher

Brandon Mauss

Nathan Paskach

1.2 Required Skill Sets for Your Project

Application

- App development
- Database knowledge
- Security knowledge

Hardware

- Circuit design
- Embedded systems knowledge
- Physics - for the feeding mechanism
- 3D modeling

1.3 Skill Sets covered by the Team

Adan: Pentesting, Physical Security, Network Security, Backend Development

Sarah: Circuit design

Nathan: Digital circuit design, Embedded systems, Soldering

Melanie: Front end app development, Gitlab, Embedded systems

Jack: Backend development, Software design

Bella: System pentesting, Network security, Hardware security

Brandon: Circuit design, Component design, 3D modeling

1.4 Project Management Style Adopted by the team

We are using a democratic project management style for both the subteams and the overall team. Most of the time decisions are made by coming to a consensus among those in the group or sub-group. When a consensus cannot be reached, we will reach out to the other sub-group or our advisor to assist with the decision or allow us to see another avenue that works better than the others we could not decide against.

1.5 Initial Project Management Roles

Software

- Jack - backend creation
- Melanie - frontend creation
- Adan - security, organization
- Bella - security

Hardware
- Nathan - firmware design
- Brandon - component design
- Sarah - circuit design, meeting notes

2 Introduction

2.1 Problem Statement

Campus Organizations Accounting and fish enthusiasts have problems monitoring the wellbeing of their fish over weekends and long breaks when access to the tanks is limited. While not in the office, the fish either can't be fed properly or have to be fed using slow "dissolving" fish food tablets that are over-kill for just a weekend. Most automatic fish-feeders release too much food at once, which is not ideal for fish tanks with only one (betta) fish. We will create a device that can deliver small amounts of fish food on a set schedule using an auger system. Also, it will measure the pH and temperature of the water. This will all be controlled by a secure web and iPhone application.

2.2 Intended Users and Uses

Oliver the Office Worker
- Wants a good work-life balance, wants their office fish to be well fed and cared for during weekends and vacations
- Needs a mechanism to feed and manage fish remotely so that they can enjoy being away from the office but know their "co-workers" are well taken care of
- They will be able to remotely manage the fish's feeding schedule, as well as check in on the general status of the fish tank

Francis the Fish Enthusiast
- Might forget to feed fish once in a while
- Not necessarily tech literate
- Would be able to continue to feed fish while on vacation and maintain their quality of care

Pete the Pet Store Owner (with aquariums)
- Doesn't want to spend time each day feeding all the fish tanks by hand
- Doesn't want to accidentally miss feeding one of them
- Would allow for regulated feeding of all fish so that none are forgotten and therefore starve

2.3 Requirements & Constraints

Functional
- Dispense up to 3 pellets of fish food per day and no more than 5 pellets/day
- Must be able to feed automatically on a schedule, manually, or on-demand via app request

- Must be able to store at least 7 days worth of food (~50 pellets)
- Must measure temperature within 1℉
- Must measure pH within 0.1
- Scheduled feeding must be able to be delayed by app user
- Visual indication/pop-up to indicate if a fish has been fed in a day or not.
- Must be able to indicate that fish has been fed by hand in the app
- Must not be able to be accessed by unauthorized users
- Powered via outlet plug in
- User login to secure app

Physical
- Attach to the top of a 1.5 gallon fish tank
- Size limit:
- Fits on the back of fish tank lid (8x8in) or in the back gap of the tank
- As small as possible so as not to distract from the fish

User Experience
- Must have a secure web app to display status of the fish tank, such as temperature, PH, and last feed
- App must be able to support connecting to multiple devices and displaying their information
- Aesthetically pleasing and intuitive design of app with themes being:
- Purple
- Red and gold
- Easy to refill
- Easy to remove and re-insert for tank cleaning
- Sleek, unobtrusive design
- iOS and Web applications
- Leave an access port/hole for manual feeding in design
- Notifications/indications in app for if the feeding failed or if the device is out of food
- Device should not be in front of the tank

Stretch goals:
- Variable amounts of food dispensed
- Salinity sensor
- Livestream of fish
- User log-in to be able to use ISU ID to login
- Fish pun name
- Backup battery in case of power outage

2.4 Engineering Standards

- 802.11 ac WiFi - for connecting to web app
- I2C, SPI, UART - for connecting to sensors
- IEEE 1621-2004 - Power control of consumer products
- NISTIR 8259A - Cyber Security Baseline

3 Project Plan

3.1 Project Management/Tracking Procedures

Our major tasks will follow a waterfall management style because there are certain tasks we need to complete before we can start the next. An example of this is designing the feeding mechanism before wiring it into the control board along with the sensors. Inside each major task we will use an agile management style so that we can rework our designs as necessary until we have the result we want. Each part will not be perfect on the first try, so this allows us to make small changes that lead to a more efficient product before moving on to the next task.

We have been implementing a mixture of waterfall and agile. The mechanism team and the app team are working independently in a waterfall fashion until we reach the point in which the components are ready to connect. Within our teams, we are working in an agile style, where we hook up and discuss where we're at and if things need to be adjusted as we go.

We will use a spreadsheet (team) and Trello (app) to track the tasks, who should complete it, and when it needs to be completed.

3.2 Task Decomposition

App

- Research frameworks and SDKs (what we are researching)
- Design UI
- Create backend and data storage
- Create front end
- Connect frontend to backend
- Connect app with mechanism

Mechanism

- Brainstorm prototype
- Explore parts for prototype
- Order parts
- Feeding mechanism rapid prototyping
- Basic workings
- Working with power
- Develop firmware for ESP32
- Test sensors
- Build working food dispensing mechanism
- Assemble minimum viable product
- Redesign based on client feedback
- Final design testing

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Mechanism:

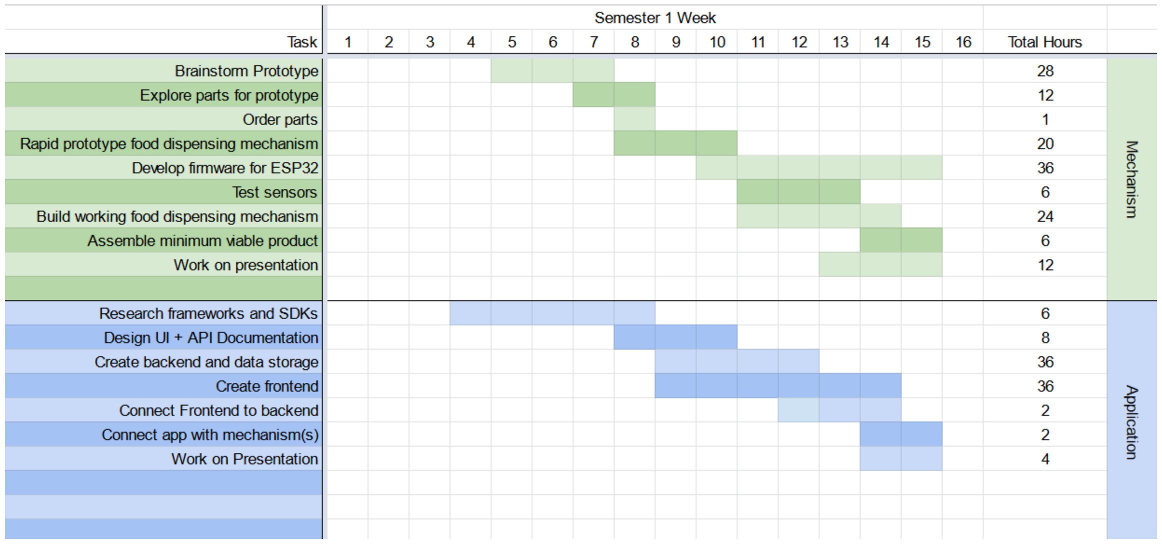
- Food dispenser prototype within food dispensing margins/tolerance
  - Week 8-10 - 5 pellets
  - Week 11-14 - 3 pellets
- Prototype circuit with sensors and microcontroller
  - Get readings
  - Within 1 °F
  - pH within 0.1
- Minimum viable prototype by week 14

App:

- Functional Backend
  - Sends/receives requests to/from mechanism
  - Stores data for user use
- Functional Frontend
  - User gives at least an 8/10 score for usability and design
  - Fish feeding manually/automatic
  - Scheduling available to user

3.4 Project Timeline/Schedule

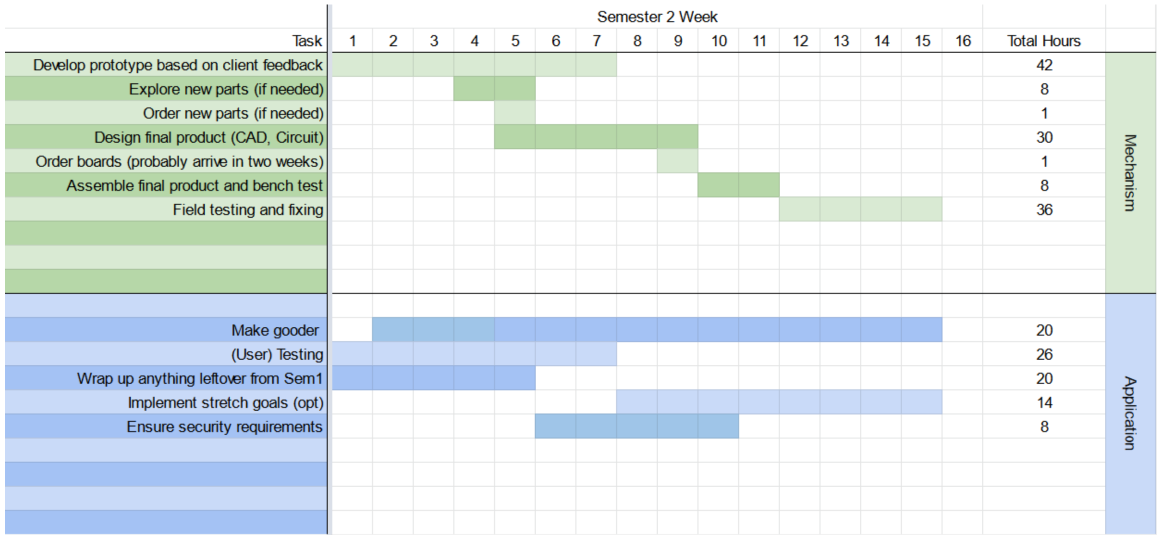
Semester 1:



	Semester 1 Week																	
Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total Hours	
Brainstorm Prototype					🟩	🟩	🟩										28	Mechanism
Explore parts for prototype							🟩	🟩									12	Mechanism
Order parts								🟩									1	Mechanism
Rapid prototype food dispensing mechanism								🟩	🟩	🟩							20	Mechanism
Develop firmware for ESP32										🟩	🟩	🟩	🟩	🟩	🟩		36	Mechanism
Test sensors											🟩	🟩	🟩				6	Mechanism
Build working food dispensing mechanism											🟩	🟩	🟩	🟩			24	Mechanism
Assemble minimum viable product														🟩	🟩		6	Mechanism
Work on presentation													🟩	🟩	🟩		12	Mechanism
Research frameworks and SDKs				🟦	🟦	🟦	🟦	🟦									6	Application
Design UI + API Documentation								🟦	🟦	🟦							8	Application
Create backend and data storage									🟦	🟦	🟦	🟦					36	Application
Create frontend									🟦	🟦	🟦	🟦	🟦	🟦			36	Application
Connect Frontend to backend												🟦	🟦	🟦			2	Application
Connect app with mechanism(s)														🟦	🟦		2	Application
Work on Presentation														🟦	🟦		4	Application

Figure 1: Semester 1 Gantt chart

Semester 2:



	Semester 2 Week																	
Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total Hours	
Develop prototype based on client feedback	🟩	🟩	🟩	🟩	🟩	🟩	🟩										42	Mechanism
Explore new parts (if needed)				🟩	🟩												8	
Order new parts (if needed)					🟩												1	
Design final product (CAD, Circuit)					🟩	🟩	🟩	🟩	🟩								30	
Order boards (probably arrive in two weeks)									🟩								1	
Assemble final product and bench test										🟩	🟩						8	
Field testing and fixing												🟩	🟩	🟩	🟩		36	
																		Application
Make gooder		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		20	
(User) Testing	🟦	🟦	🟦	🟦	🟦	🟦	🟦										26	
Wrap up anything leftover from Sem1	🟦	🟦	🟦	🟦	🟦												20	
Implement stretch goals (opt)								🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		14	
Ensure security requirements						🟦	🟦	🟦	🟦	🟦							8	

Figure 2: Semester 2 Gantt chart

3.5 Risks And Risk Management/Mitigation

Risk is a part of everything we do; Table 1 gives examples of the risks possible in this project, how likely they are to occur, what might happen if it occurs, and how we can mitigate the high likelihood risks.

Risk	Probability	Consequences	Mitigation	Notes
A part does not come within the expected timeframe	0.2	prototyping is delayed	N/A	
We fry a part without a replacement	0.05	need to order a new part, production potentially delayed	N/A	Depending on the part, like basic circuit components, we could get one from the ETG
Have trouble connecting software and hardware	0.9	a lot more work time to determine the problem and refactoring	Adhere to API between devices as strictly as possible	
Repeated rejections of UI Design	0.05	Delays development of frontend	N/A	Will require closer work with client if occurs
Flutter no longer supported	negligible	Have to switch to another platform of development	N/A	
Feeding mechanism falls outside of tolerance	0.3	will need to err on the side of less food to make sure we don't overfeed	N/A	Test, test, test
Electronic components getting wet during semi-weekly cleaning	0.5	Damage to electronic components, failure of device	build a case for long term, put in ziploc bag for prototype testing	

Table 1: Risks and risk management

3.6 Personnel Effort Requirements

Table 2 breaks down the number of hours we expect each portion of the project to take with an explanation of why.

Task	Person Hours	Explanation
Prototyping Device Brainstorming Rapid prototyping Firmware Sensors Working food dispenser Minimum viable product Changes from client feedback	174 40 20 36 6 24 6 42	Lots of time breadboarding, wiring, soldering, programming, and then doing it all again.
Testing Device Client feedback Final bench testing	50 42 8	There are a lot of components that need individual testing, and then testing the full system once it's assembled
App Planning	14	A lot of the planning takes place alongside one another so it shouldn't be terrible in total time.
FrontEnd Development	36	Will take lots of time to build pages and to interconnect everything. May take some refactoring if the client is unhappy with the design or feel.
BackEnd Development	36	Should get set up early but might take a bit to get stable and perfectly done.
Connecting device to app	2	Should be a short task
Presentation	16	Will take collaboration from all subgroups to plan, create, finalize, and execute.

Table 2: Projected Project hours

3.7 Other Resource Requirements

- Cooperation from ISU Solution Center for Active Directory use
- Server usage
- Cooperation from ISU IT to let the ESP32 onto the ISU network

4 Design

4.1 Design Context

4.1.1 Broader Context

The current fish feeders on the market dispense too much food for a single fish, which can make small tanks dirty. We are designing a smaller feeder for the Campus Organizations Accounting department. This will affect both the office workers and students who visit the office. Shown below in Table 3 are the considerations related to the project.

Section	**Considerations**
Public health, safety, and welfare	The most important thing that we have to consider when it comes to the safety, health, and welfare of those influenced by our project by far is for the overall welfare of the fish. This, in turn, creates an environment that those caring for the fish can be comfortable and feel secure in.
Global, cultural, and social	Our global cultural and social significance is fairly negligible. There aren't many ethical considerations that need to be given inside of the mainstream purview. The main impact would be for the wellness of fish to be more of a focus by individuals engaging with the product.
Environmental	Our design will use electricity from the wall at all times, but will save on gas from driving to the office to check on the fish. However, it prevents a more frequent changing of water in the tanks which will in turn save potable water.
Economic	The device will have an electricity cost associated with it, whether it is being actively used or not since it is taking pH and temperature reading often and always plugged into the wall. Our rapid prototyping supports tech businesses through the purchase of components.

Table 3: Broader context breakdown

4.1.2 Prior Work/Solutions

At the time of receiving this project, the client had described their process for trying to solve this problem before, and discussed the shortcomings of things on the market with respect to what they wanted for their tank. The main issue with on the market products was how large the volumes of food they fed were. Due to this, we haven't been comparing our feeder with any one product on the market currently, but more so what our client had noticed about alternative options and what they wanted or didn't want.

Pros of our design	Cons of our design
● Small, precise amount of food dispensed ● Fits on small fish tanks ● Senses water temperature and pH ● Connected to phone/web app for easy access and schedule changes ● Less wasteful, as it doesn't use batteries ● Stores multiple feedings worth of food ● Won't muck up water like tablets	● Depends on server connectivity ● Relies on building electricity ● Does not include a protein skimmer ● Can only work with one kind of feed (pellets)

Table 4: Design contrast

Information about current feeders on the market was found at:
https://aquariumstoredepot.com/blogs/news/best-automatic-fish-feeder

4.1.3 Technical Complexity

- Mechanism
  - Motor with encoder - for dispensing food and detecting if there has been a jam
  - Temperature sensor - for sensing temperature (a client requirement)
  - pH sensor - for sensing pH (a client requirement)
- App
  - Client-server relationship - The app needs to be able to connect and communicate with the mechanism, so therefore we need a server as well as clients in order for this communication to occur.
  - iOS and Web app frontends - the client needs a way to interact with the mechanism and to know the status of all the sensors. We decided to use iOS and Web apps for the client to be able to access the app from their phone or their computers depending on which is easier for them at any given time.
  - Backend with a database - a server is needed to communicate with the client and make higher-level decisions for the mechanism. A database can be used to store historical data for the temp and pH of the tanks.
- Integration
  - Connect hardware and software with server connection
  - Send and receive requests from both app and mechanism

The current industry standard for fish feeding typically only works for larger fish and/or aquariums. Our system works for individual smaller fish in a compact environment – dispensing a small volume of food as necessary.

4.2 Design Exploration

4.2.1 Design Decisions

- User Authentication: Okta SSO - for ease of sign-in by clients and keeping out non-admins if we choose to add a livestream option for student viewers.
- Feeder design: Gumball/Drum - small amount of food dispensing is the whole reason for the project. We wanted to be picky about precision and size to best suit the client's needs and wishes.
- App platform: iOS and Web - whether we implement it for iOS or not completely changes what tools we can use and the overall look/feel of our app's frontend. iOS has their own design requirements/guidelines that we'll need to follow. We'd also like to implement it for the web, so the users can easily use the app on their phone or on their computer, whichever is easiest. This means to support the two, we need a SDK to manage both, which is why we chose Flutter.

4.2.2 Ideation

We identified potential options by brainstorming ideas and sketching them out on the whiteboard. We then asked questions of the person who gave the option if something was not understood.

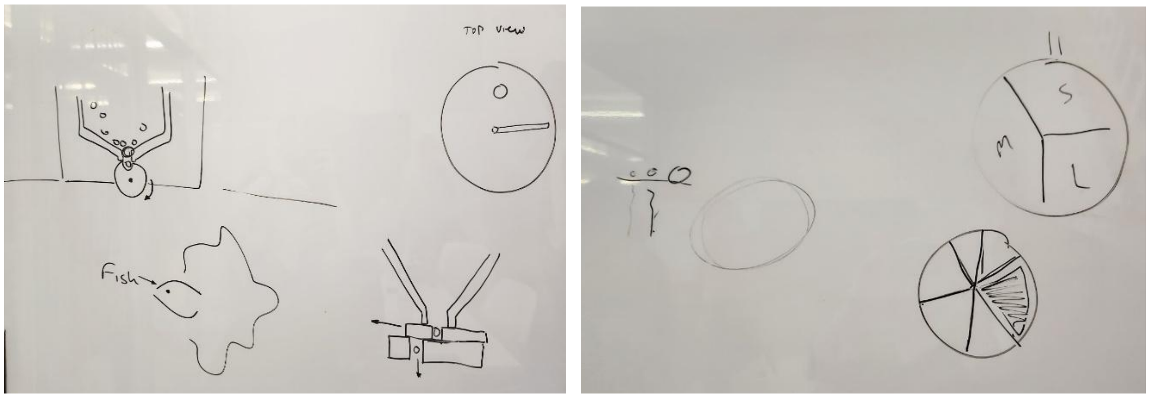


Figure 3: Initial design ideas

Feeder Designs:

- Coin slot – sorting pellets based on size to give a precise number of food pellets. Sorting works similar to how a coin sorter works, in that there are increasingly large holes on a track that food can fall into.
- Screw rotator - have the height between the levels on the screw be the size of 1.5 pellets, turn the screw a small amount of time to dispense the food
- Cereal dispenser - like the screw rotator but less accurate due to larger partitions
- Gumball machine - a scoop is used to move a group of pellets up higher than their resting position. At the apex of the scoop, there is a hole that's the same size as a pellet with the desired volume of food. This allows the desired volume of food to be delivered to the fish.
- Drum with food crevice- A vertical drum that catches one pellet from a hopper at a time as it rotates. Similar to the cereal dispenser, but only grabs one pellet at a time.
- Weekly pill tray- small compartments are individually loaded with desired amount of food by the user, and a slot of food is released for each feeding. Could be built into a wheel so the wheel simply rotates to line up the next day's food with the chute.

4.2.3 Decision-Making and Trade-Off

We decided to test the Gumball and Drum designs because:

- The weekly pill tray method would require too much of the users, and we'd like the user to have as little maintenance required as possible. Also, it would limit the user to using the feeder for only a week (7 days) max.
- The cereal dispenser will be too inaccurate.
- The coin slot won't be used because we determined the volume of food is more important than the number of pellets.
- The screw rotator accuracy depends highly on the amount the motor spins, which can change depending on the power provided to the motor and how well the motor responds. It also has a high probability of getting jammed by food dust.

Rather than locking into one design before testing, we've narrowed down the most likely options – gumball and drum designs – to experiment with them firsthand. Once we have working models of both we can do some more in-depth testing to make a more educated final decision based on evidence.

4.3 Proposed Design

This section refers to the design created in 491, section 6 covers the final design.

4.3.1 Overview

The automatic fish feeder consists of a food hopper, dispensing mechanism, temperature sensor, pH sensor, iPhone and Web app, database, and a microcontroller that connects the app and database to the dispensing mechanism and sensors.

Food pellets go into a hopper which feeds into a dispensing mechanism. The feeding mechanism's motor rotates to select a single pellet of food from the hopper and delivers it to the fish. To get more than one pellet of food, the mechanism repeats this action multiple times. The action will also be repeated if the mechanism does not detect that a pellet entered the trough. The automatic feeding times and amounts can be configured with an iPhone or Web app. Also available in the app are the water temperature and pH readings read by the microcontroller.

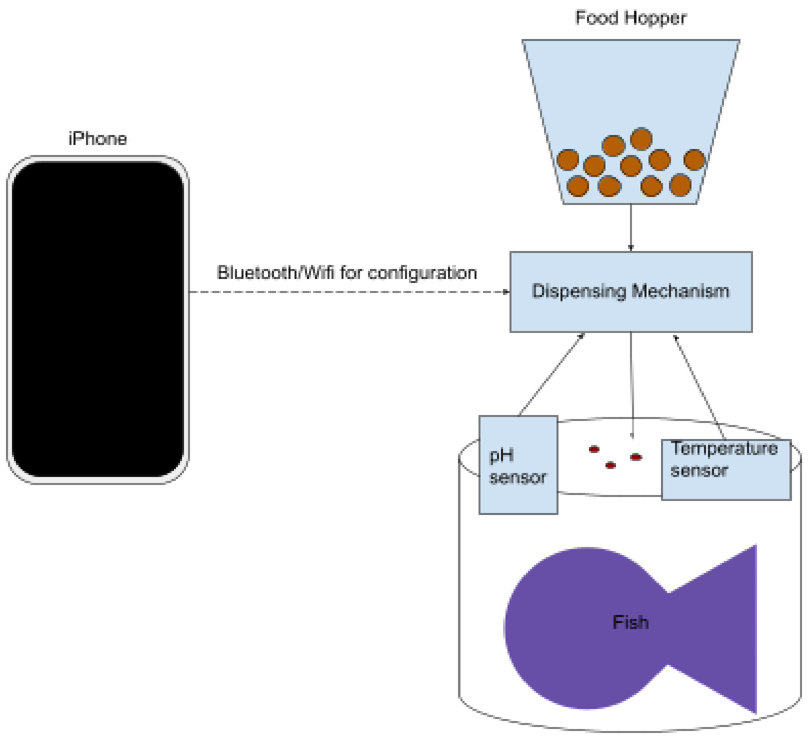


Figure 4: High level design

4.3.2 Detailed Design and Visual(s)

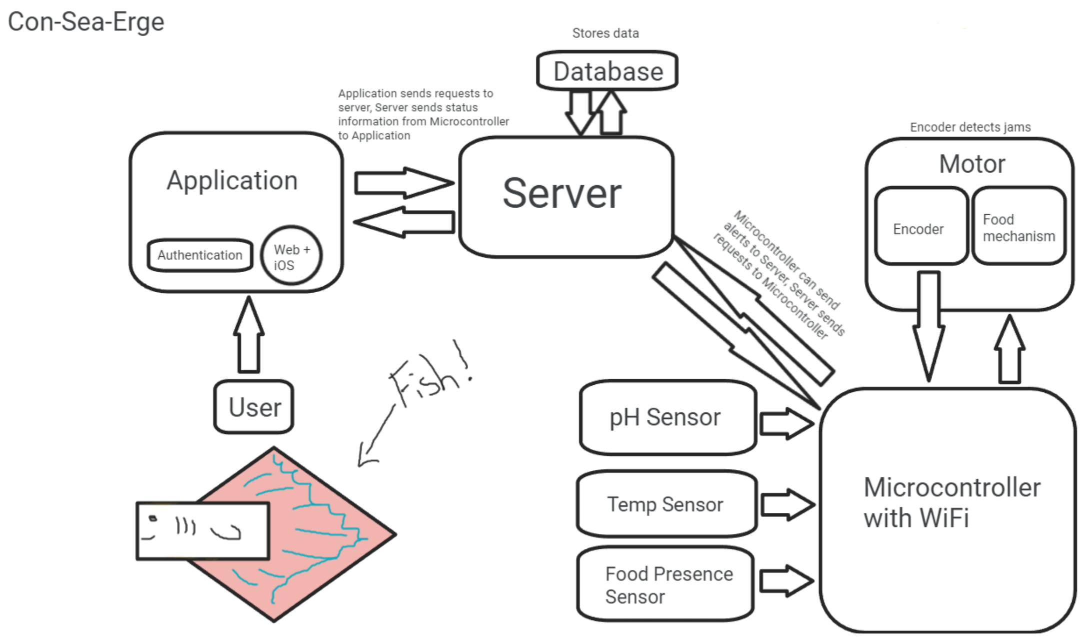

Con-Sea-Erge
Stores data
Database
Application sends requests to server, Server sends status information from Microcontroller to Application
Application
Authentication
Web + iOS
Server
Encoder detects jams
Motor
Encoder
Food mechanism
Microcontroller can send alerts to Server, Server sends requests to Microcontroller
User
Fish!
pH Sensor
Temp Sensor
Food Presence Sensor
Microcontroller with WiFi

Figure 5: Detailed block diagram of design

Our team can primarily be broken up into two sub-groups, working on the two main components of our device: software and hardware/firmware.

The application will be available on Web and iOS platforms as we will be implementing it in Flutter. This app will be the primary interface between our product and the user, so we will be heavily focusing on our visual design and giving our users as many options as they want/need. We will be displaying all the sensor data collected by the mechanism, then giving options for schedule use/creation so the user can set when they'd like the automatic feeder to run. We will also give them options to inform the system that they fed the fish by hand, as well as telling the feeder to run on user request. We will be implementing alerts so that we can inform the user if the water is out of pH range or temperature range, if the mechanism fails in any way, or if the feeder is out of food. For the UI, we will have a home/dashboard page where the quick details about the fish tanks can be viewed, such as the pH, temperature, and if it's been fed so far that day. We may also include the option to hit a button on the dashboard for each device to feed their fish. There will be a schedule tab, where the user can see all created schedules, toggle which schedule should be running if any, and create a new schedule. When "Create a new schedule" is clicked, it will take the user to a new page, where they can select which days, if any, they would like the device to dispense on and what time they'd like it to occur during the day as well as name the schedule. This new schedule will then be displayed in the schedule tab/page. Tentatively, we plan to add a settings page where the user may be able to change the color scheme of the app to their liking. We are also tentatively planning to store previous status data, so the user would be able to open a page and view the pH and temperatures over the past 50 days for each tank. This way if the user is interested in patterns with water quality, they can do so.

From the user end, accessing the device's setting/specifications will be a little bit more involved than a typical web-app requires. As a result of hosting the app on Iowa State's servers, the user will be required to either be on campus or connect to the Iowa State VPN for access. Past this requirement, user authentication is required to protect the security of both the fish and the user data; while the possibility of this hasn't been confirmed, we are hoping to implement Okta SSO to do that with the help of the Identity Team manipulating Active Directory roles.

The feeding mechanism will consist of a food hopper, a motor for actuating the dispenser, an encoder for sensing jams, and a food detection device using an LED and a photocell to confirm the presence of a food pellet. This is controlled by an embedded WiFi and Bluetooth enabled microcontroller. Automatic feeding times and amounts are configurable through the phone app. Also connected to the microcontroller are a temperature sensor and a pH sensor which will monitor the status of the fish tank's water. The microcontroller will put these readings up on the server to be scanned for anomalous values in order to alert the user.

4.3.3 Functionality

This design is anticipated to operate at the user level. However, it is possible to be developed further into a commercial product just as other fish feeders, that is beyond the scope of our project. At the user level, anyone with access to the devices and the application would be able to feed the fish either by hand or the machine, monitor the water conditions to decide when it might be necessary to change, and will alert the user on when the water conditions are no longer safe for the fish.

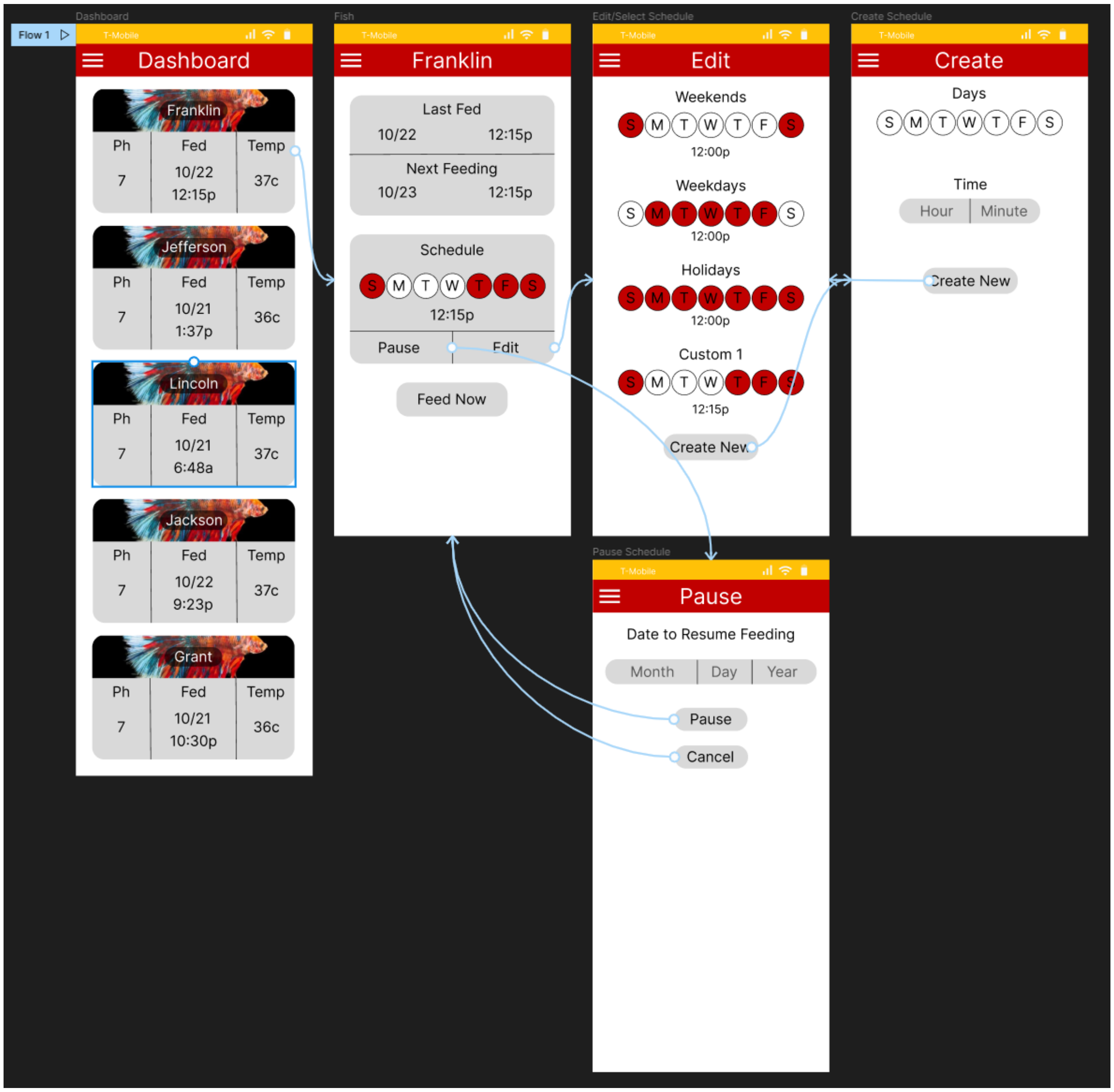


Figure 6: Possible app pages

After the user sets the schedule as shown above, the database and the microcontroller take over. When the scheduled time comes to feed the fish, the dispensing mechanism will rotate to remove one pellet from the hopper and drop it into the tank. This rotation step will continue until the necessary number of pellets have been dropped into the tank. If no pellet enters the dispensing area from the hopper, as registered by the LED and photocell, the mechanism will rotate again a max of two times to attempt to get another pellet. If all three of these rotations come up empty, the user will receive a notification through the app that the hopper is out of food. The user will also receive a notification if the motor in the dispensing mechanism encounters a jam which is read by the motor’s encoder.

The temperature and pH sensors will take readings every few minutes and the microcontroller will send them to the backend. These readings will be visible for each tank in the app.

4.3.4 Areas of Concern and Development

As far as we can tell, this should perfectly fit the needs of the user. Our greatest concern is accuracy of the feeding mechanism which has yet to be tested. We need it to not overfeed the fish, so we will be testing a couple designs and troubleshooting them throughout the process to try to get our mechanism to be as accurate and reliable as possible.

One concern we have is the user's connection to the devices while at home. They would need to be connected through the VPN if they want to use the Web-based application. We know that it is possible to have the iOS app connect, but are questioning how many issues this could cause. To address this concern, we will continue to research and then test many times to determine what needs to change.

4.4 Technology Considerations

The technologies implemented within our design are the pH, temperature, and feeding sensors. The pH sensor is an off the shelf pH sensor which connects to a microcontroller over a certain protocol. It is a known-working design, but it is also the most expensive piece of the design. There are very few alternatives, though. For the temperature sensor, we will use either a thermocouple with a control circuit or a thermistor in a resistor ladder. The thermocouple would be abstracted to another peripheral through a communication protocol, but it is much more expensive and probably less waterproof. The thermistor is very cheap and waterproof, but will require characterization and calibration before it can be used. Through testing of the circuit, we decided to use the thermistor even though it would be less accurate. Since the requirement for temperature was within ± 1℉, we determined that feasibility of integrating with the circuit was more important than accuracy since both would be accurate at that division.

4.5 Design Analysis

The hamburger menu was too difficult to implement therefore we decided to use buttons at the beginning of the page. We also determined that creating an iOS app was not going to work with the time we had left. The dispensing mechanism worked for the most part, we had some issues with the funnel jamming when two small pieces or one of the largest ones entered the funnel's tube.

5 Testing

Our testing plan will start with separated tests for the software and hardware portions. We will test each component first to make sure everything responds as we are expecting and then test the combination of the parts into each subsystem. Once each subsystem passes its test, we will combine them all together and run system tests to ensure the entire design works as we want it to. We will then give the system to the Campus Organization Accounting Office for acceptance testing. The feedback we get will then determine our next steps and tests. All of our testing will be done throughout the project to ensure we are meeting project requirements.

5.1 Unit Testing

We will be testing the software using unit testing. We will use postman to mock up a server in order to test the frontend separately from the backend, and vice versa for the backend to test independently of the front end.

Frontend: We will test many smaller components in the frontend individually, with both unit tests, UI testing, and postman mocked server tests. With postman's mocked server, we will test all of the following to answer two questions: Does it send the request to the server in the correct format and when requesting data from the server and receiving it, does it display it correctly? We will attempt to both send information to the server from the frontend, and then try to send data from the mocked server to the app. We will also be able to test whether the app displays what we expect it to if a connection to the server is not made, so unit testing will be a great time to check if the frontend of the application properly displays some of the error messages we will be implementing.

- Adding schedules
- Deleting schedules
- Switching between schedules
- Naming or renaming schedules
- Adding pictures for a tank
- Naming/renaming the tank
- Multiple devices displaying correctly
- pH displays correctly
- Temperature displays correctly
- Last fed time displays correctly
- Changing the color scheme is saved by the server properly
- Color scheme is set at app opening by saved color scheme setting

For UI testing:

- Ensure all buttons navigate properly- Does each button navigate to the correct page
- Color scheme changes in settings correctly and displays that scheme in all screens for that user.
    - Does closing and reopening the app reset the color scheme
    - Does navigating to a different screen in the app reset the color scheme
- Buttons and displays are arranged as intended in both iOS and Web displays
    - Run iOS and Web apps to ensure a cohesive look and feel in both.

For security test:

- Require user authentication
- Ensure TLS layer is implemented

Backend: The components being tested will be separated by packages in the file hierarchy. This includes checking auth functions, functions with the devices, and functions for editing/creating schedules. It is also important to test database connections and ensure data is being stored and retrieved correctly.

Hardware: The hardware units to be tested are the feeding mechanism and each of the sensors (motor encoder, food presence, pH, temperature). We will use the ESP32 with test-specific firmware to test these components by themselves. We will be looking at the values returned by the sensors to determine if they are near the expected values.

To test the temperature sensor, we will use a digital thermometer to read the temperature of a cup of water and compare it to the thermistor reading. The thermistor reading will then be calibrated by graphing the digital thermometer's reading against the thermistor's and using a best fit line to determine the calibration equation.

The pH sensor will be tested and calibrated using buffer solutions with known pH. We will start with the neutral buffer and use the adjustment knobs on the pH sensor to ensure this reading is 7. While we were testing the sensor with this method, we realized the adjustment knobs could only drop the reading to 7.9. Therefore an equation is necessary to drop it the rest of the way. Two more buffer solutions of pH 4 and 10 were used to create two more points for the best fit line. After calibration, two sanity checks will be done using lime juice and baking soda dissolved in water since these have different pH values than the buffer solutions, of about 2 and 9 respectively.

The food presence sensor reading is an analog value, but we will be comparing the voltage read when there is a pellet to no pellet to calibrate the levels we expect. The actual values do not matter, just the difference between them.

The motor encoder will be tested by running the motor without any contact with the food to have a base reading and then creating a jam to determine the output this creates. This will also test the hardware interfaces between the ESP32 and the peripherals.

Testing for the case will include visiting the client and placing the latest print in the lid of their spare tank. We will need to ensure that the holes for each of the sensors and LEDs do not conflict with the lip or bar of the tank lid. This time will also be used to get the client's feedback about the shape of the case. The case will also be drop tested at different heights to ensure the durability of the design.

5.2 Interface Testing

We will test the interface between the device and the app. We can test that the device can receive feeding time updates from the app by monitoring its debug outputs. Also, we can test that the device status makes it to the app by observing the values in the app and if they reflect the current state of the device. The conditions in which the device is in will be determined before the test is run, specifically the pH and temperature of the water. We will also test this connection by sending the request/command to feed and witnessing the device dispense real time.

5.3 Integration Testing

One integration is connecting the motor and encoder with the food dispenser and food presence sensor and testing if that subsystem reliably detects when a piece of food has been successfully dispensed. We will test if the dispenser can reliably transport one piece of food at a time via visual confirmation, if the food presence sensor detects it correctly every time, and if we can tell if the food mechanism is stuck from the reading of the motor encoder's output. The latter two tests will be similar to those in section 5.1 Unit Testing subsection Hardware.

Another is connecting all of the sensors to the ESP32 and testing that all are reporting correct values when polled. These tests are the same as those described in section 5.1 Unit Testing subsection Hardware.

Also, we will test how well all of the components fit inside of the case. To do this we will attach the pH sensor circuitry and thermistor leads to the main board, the dispensing bar to the motor, and place the combinations and LEDs in their predetermined places as explained in section 4.3.2. Layout decisions will need to be made if the parts do not fit or if they are difficult to put in.

A fourth is connecting the frontend and backend of the app, to ensure that all settings and data is passed correctly. In order to test the connection we will run the app with frontend and backend connected, then we will run the tests from the frontend, likely with similar tests or the exact same tests used with postman's mocked server. We can fill the backend with various information to make sure the expected information is displayed. We will also test that the information changed/saved in the front end is properly saved in the backend and that the information is retained and displayed upon the app closing and reopening.

Finally, we will connect the app and the feeding mechanism and repeat the previous test with real values from the device.

5.4 System Testing

When the app and the mechanism are fully connected, we can begin system testing. In order to test the full system, we will need to test that:

- When the "feed" option is pressed on the frontend, the mechanism dispenses the correct amount of food via visual confirmation
- When the feeder is empty, the frontend displays a error notification
- Waterproof testing of final hardware housing before putting the hardware in
- Error reporting on frontend of:
    - Food depleted
        - Using a full, partially full, and empty hopper
    - Dispenser jammed
        - Either by purposefully getting a piece of food stuck or applying pressure to the feeding mechanism's rotating portion to simulate a jam
    - pH out of range
        - Using lemon juice or baking soda dissolved in water
    - pH spike
        - By moving the pH sensor from water to lemon juice or baking soda water and back to the water
    - Temperature out of range
        - By placing the temperature probe in an environment that is hotter or colder than is safe for betta fish
    - Temperature spike
        - By moving the temperature probe from the correct temperature to an drastically different temperature environment and back to the original temperature
    - Maybe power loss if we can get a big enough hold-up capacitor
- When a schedule change is made, the device changes its dispensing timing to reflect the change
- If the user indicates that they want a scheduled feeding delay to feed the fish by hand, then the device changes its dispensing timing to reflect the change.

5.5 Regression Testing

Our team plan is each time an update is made to the application, the previously written tests will be automatically run by a gitlab CI runner.

When the sensors are connected to the final board, their outputs will be retested to ensure the output values are within the tolerances. This is needed due to the variability of resistance between the protoboard and final boards.

5.6 Acceptance Testing

Acceptance testing will be continuous throughout our project. As we progress through the project and continue to develop our product, our clients will use each prototype. When the newest prototype is developed, we will question the client on any problems they incurred with the feeding mechanisms, the temperature and pH sensors, the application and any of its features. Then, when we begin developing the next prototype, we will attempt to include any suggestions they might have for better usability. We have already completed some acceptance testing, as we have shown the client our various UI designs to receive feedback. This along with previous discussions, has helped guide our project and where we need to go with it, and special functionality the users would like.

5.7 Security Testing

Following the InfoSec guideline for information security, CIA, there are three things that must be tested and ensured in this project: confidentiality, integrity, and availability. On top of this, security testing is vastly different depending on our sub-group and will be broken down as follows:

Confidentiality	**Software**	On the software side, confidentiality is the ability for activity done by the client and our device to be private. We can ensure this by using transport layer security (TLS) in our HTTP communications, ensuring device settings are only available for reading by administrators of the device, and similar best practices. This can be tested using network sniffing techniques.
	Hardware	Luckily, there is no communication between the hardware side of the device and the user – there should be no need to ensure confidentiality in this case beyond the baseline design.
Integrity	**Software**	To ensure that communications to the device are received as intended by the administrator, we will use the TLS layer as stated in the confidentiality section. On top of this, we can have certain requirements of a user profile before allowing them access. We can use an offensive security penetration test to ensure that our measures are successful.
	Hardware	In this case, we recommend that the client/user lock the door when they're stepping away from the device – so as to ensure that there is no physical tampering of the device by visitors to the office. We can secure for accidental or unskilled tampering by light percussive testing, in cases such as a curious child or unknowing adult.

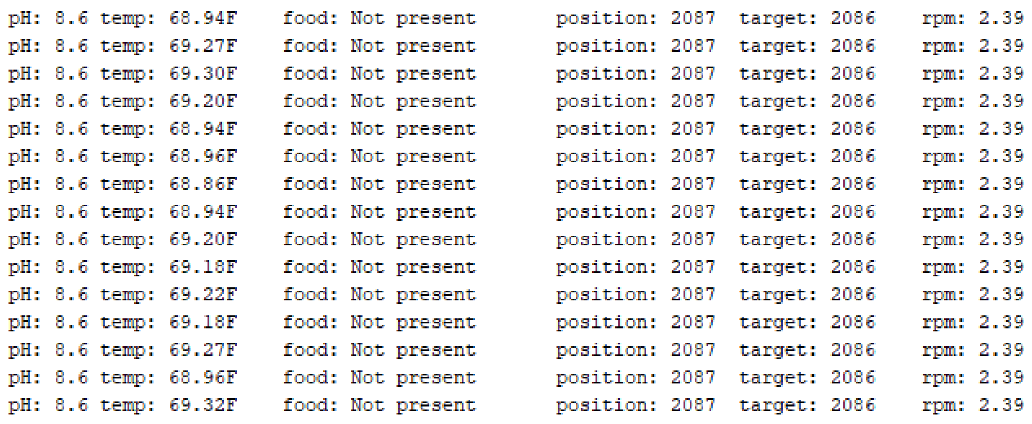
Availability	**Software**	While this is definitely going to be implemented and tested more at the software developing level, there are certain measures that we can take to add consistency to the device's communications. We'll be able to measure any changes and test their efficacy by network manipulation – creating low bandwidth scenarios, weak DOS attacks, etc.
	Hardware	Hardware availability is going to be similar to the testing done for hardware integrity – in that, more than likely, there is not going to be a case where users are maliciously tampering with the physical device. Testing will be done to ensure that minor accidents and curious prodding won't affect the device's functionality.

Table 5: Security testing breakdown

5.8 Results

5.8.1 Mechanism Testing

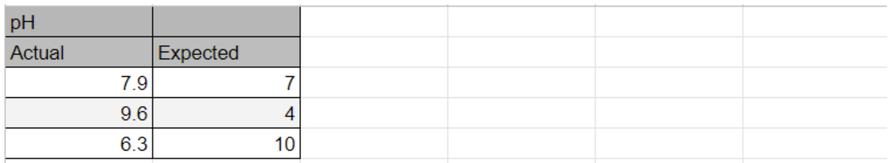
The initial testing of the circuit returns values for the temperature, pH, and food presence sensors. The motor also turns when voltage is applied to it. Calibrated values for the pH and temperature sensor are shown in Figure 7. The readings after calibration fall within required tolerances.



```
pH: 8.6 temp: 68.94F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.27F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.30F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.20F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 68.94F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 68.96F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 68.86F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 68.94F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.20F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.18F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.22F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.18F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.27F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 68.96F    food: Not present       position: 2087  target: 2086    rpm: 2.39
pH: 8.6 temp: 69.32F    food: Not present       position: 2087  target: 2086    rpm: 2.39
```

Figure 7: Feederal Reserve debug outputs after calibration

Figures 8 and 9 are how we determined the equation used to calibrate the sensors. The expected values for the pH readings came from buffer solutions with specific pH levels. Since there was no documentation for how the pH sensor was supposed to be connected, we determined that we interpreted the sensor values backwards. Therefore, basic solutions were read as acidic and vice versa. We chose to use a second order function to describe the calibration because the first order function was great for neutral solutions but did not give values within our tolerance of 0.1 for the acidic and basic solutions.



pH	
Actual	Expected
7.9	7
9.6	4
6.3	10

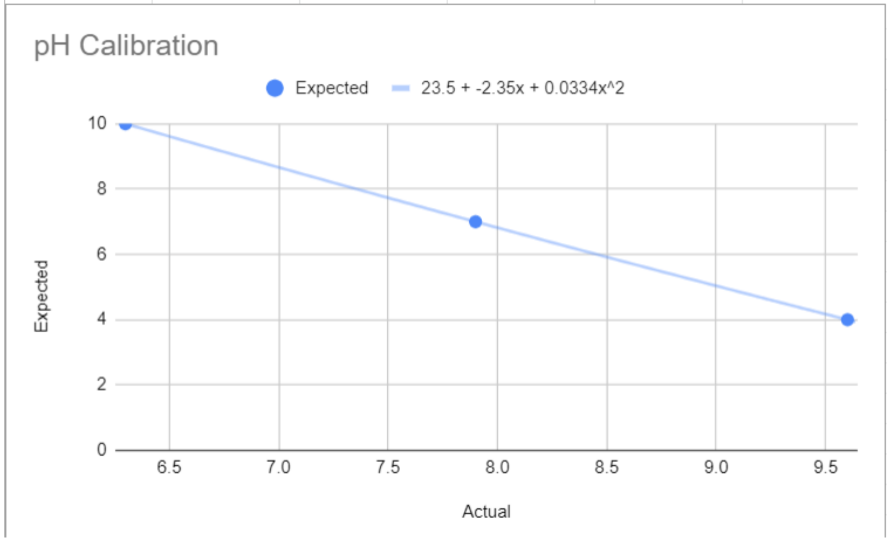
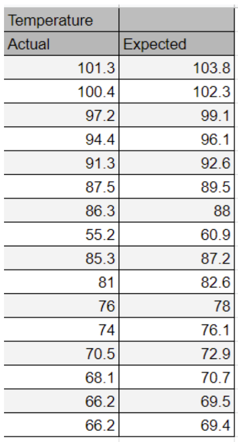


Figure 8: Feederal Reserve pH sensor vs pH buffer solutions

To determine the expected values for the temperature calibration, we used a digital meat thermometer that could read to the tenths place. Almost all of the sensor's measurements were within 2℉ of the digital thermometer. Even though the requirement for the temperature sensor is ±1℉, we read to the tenths place to limit the error in our equation.



Temperature	
Actual	Expected
101.3	103.8
100.4	102.3
97.2	99.1
94.4	96.1
91.3	92.6
87.5	89.5
86.3	88
55.2	60.9
85.3	87.2
81	82.6
76	78
74	76.1
70.5	72.9
68.1	70.7
66.2	69.5
66.2	69.4

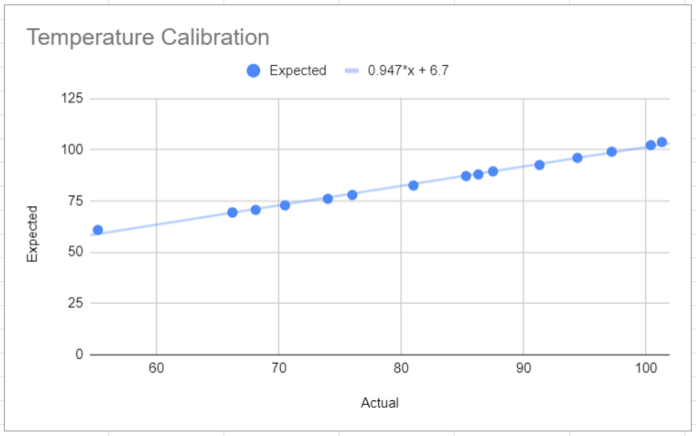


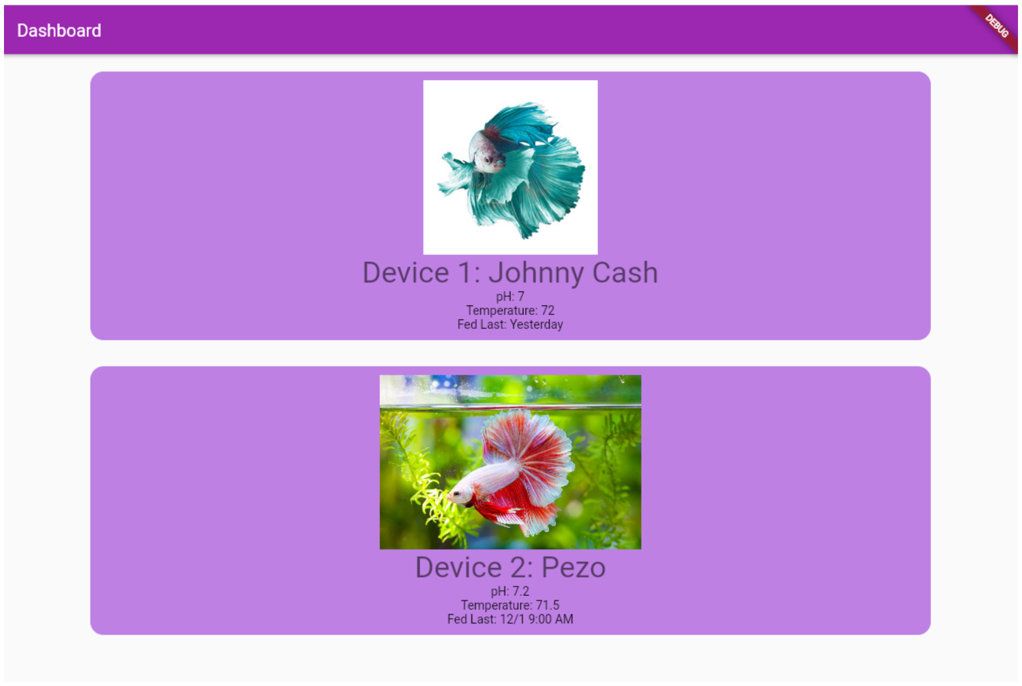
Figure 9: Feederal Reserve temp sensor. vs. digital thermometer

In the initial testing of the feeding mechanism, without any connection to the motor, the pellets were getting stuck on the lip created by the photoresistor's hole. To solve this issue, we chamfered the front edge of the photoresistor's hole and the food transport bar's hole to allow for a more gradual transition. When we connected the motor to the carrying bar, the larger pellets were transferred to the tank without any issues, but two smaller pellets often became jammed when they came out of the hopper together. The solution to this problem was to make the diameter of the hole in the carrying bar large enough to hold two of the small pellets. Also, we decreased the height of the storage section of the hopper and increased the drop distance to account for this.

5.8.2 Application Testing

The backend is at a point where every major feature has been created aside from the inclusion of websockets for the device connections. All other routes from the API documentation have been implemented and work for expected use cases. More work will need to be put in to ensure that everything is stable but as of right now it is functional. The job queue and ability to execute functions on a specified day at a specified time is also complete. The jobs are stored in a database so if the server ever loses lower or needs to restart, everything loads back in correctly and can continue to be used.

Frontend home page/dashboard has been created and is being implemented. There is still progress to be made in the page to make it more appealing to the client, but the results so far have been good.



Device 1: Johnny Cash
pH: 7
Temperature: 72
Fed Last: Yesterday

Device 2: Pezo
pH: 7.2
Temperature: 71.5
Fed Last: 12/1 9:00 AM

Figure 10: Current Dashboard view on Web

6 Implementation

Since the end of the first semester, many changes have been made to the device and the app. Reasons for the device changes are covered in the testing section. Changes made to the app were done due to client feedback and challenges associated with creating an iOS app.

The feeding mechanism consists of a food hopper, a motor for actuating the dispenser, an encoder for sensing jams, and a food detection device using an LED and a photocell to confirm the presence of a food pellet. The 3D model of the feeding mechanism case and food dispensing bar is shown in Figure 11.

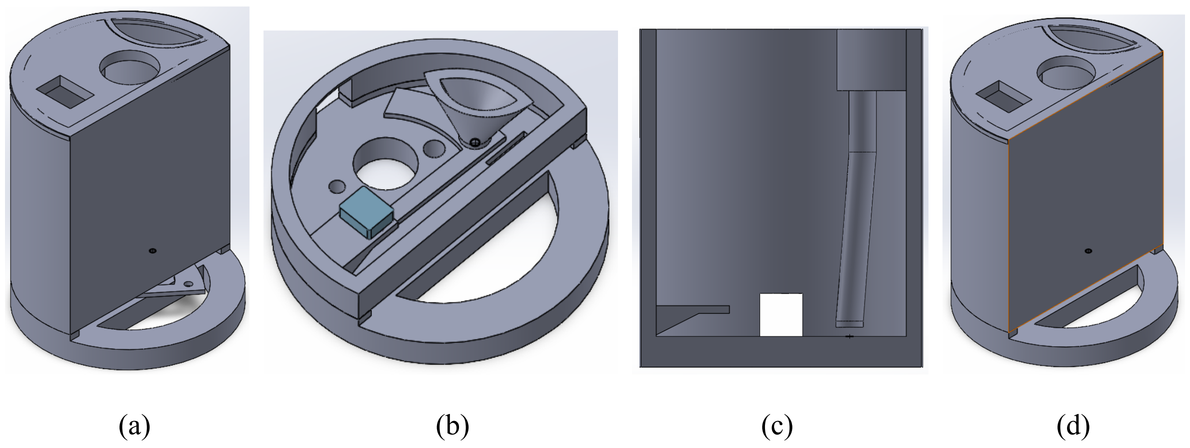


Figure 11: 3D model of feeding mechanism, (a) feeding position (b) retrieving food/not in use position from above (c) front view with front side removed (d) outside view of case

The large hole in the top center of Figure 11.a houses the upper portion of the pH probe's casing. The medium sized holes hold the LEDs that light up the tank. These LEDs are wired separately from the main feeding mechanism circuit and are controlled by a switch on the lid of the case, opposite of the hopper. A motor is screwed to the jut out on the left side of the case. Just below the hopper in Figure 11.b, a divot can be seen, this houses the photoresistor and the LED for testing if food was dispensed is attached to the side of the hopper directly above the photoresistor. Figure 12 shows the inside of the feeding mechanism with the sensors installed.

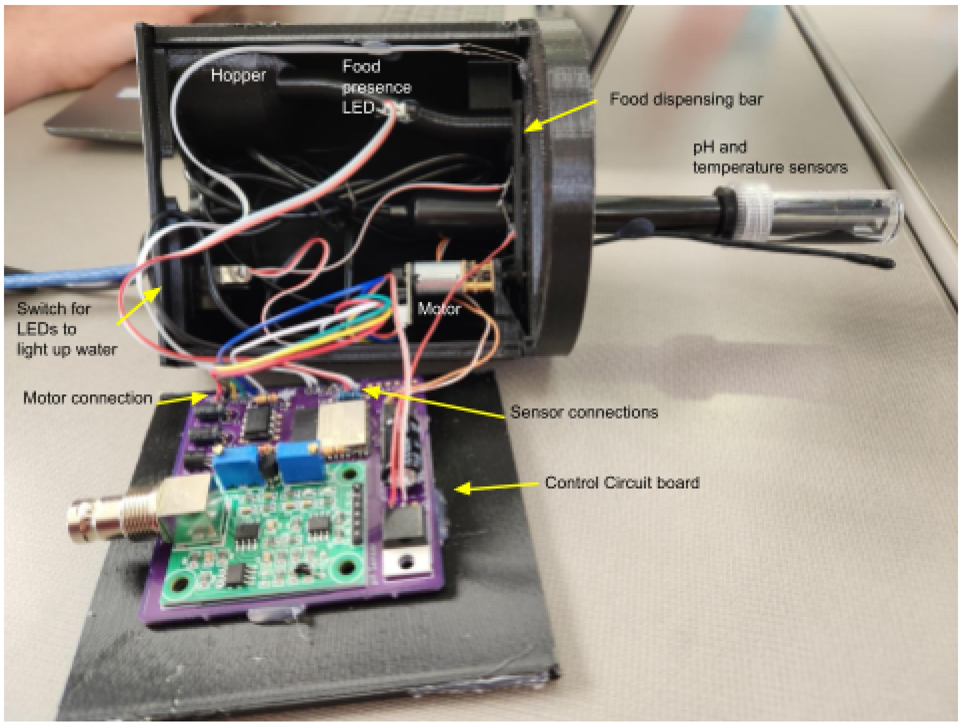


Figure 12: Inside of the feeding mechanism showing the board and connections

This system is controlled by an embedded WiFi and Bluetooth enabled microcontroller. Automatic feeding times and amounts are configurable through the phone app. Also connected to the microcontroller, as shown in Figure 13, are temperature and pH sensors which will monitor the status of the fish tank's water. The microcontroller will put these readings on the server to be scanned for anomalous values in order to alert the user.

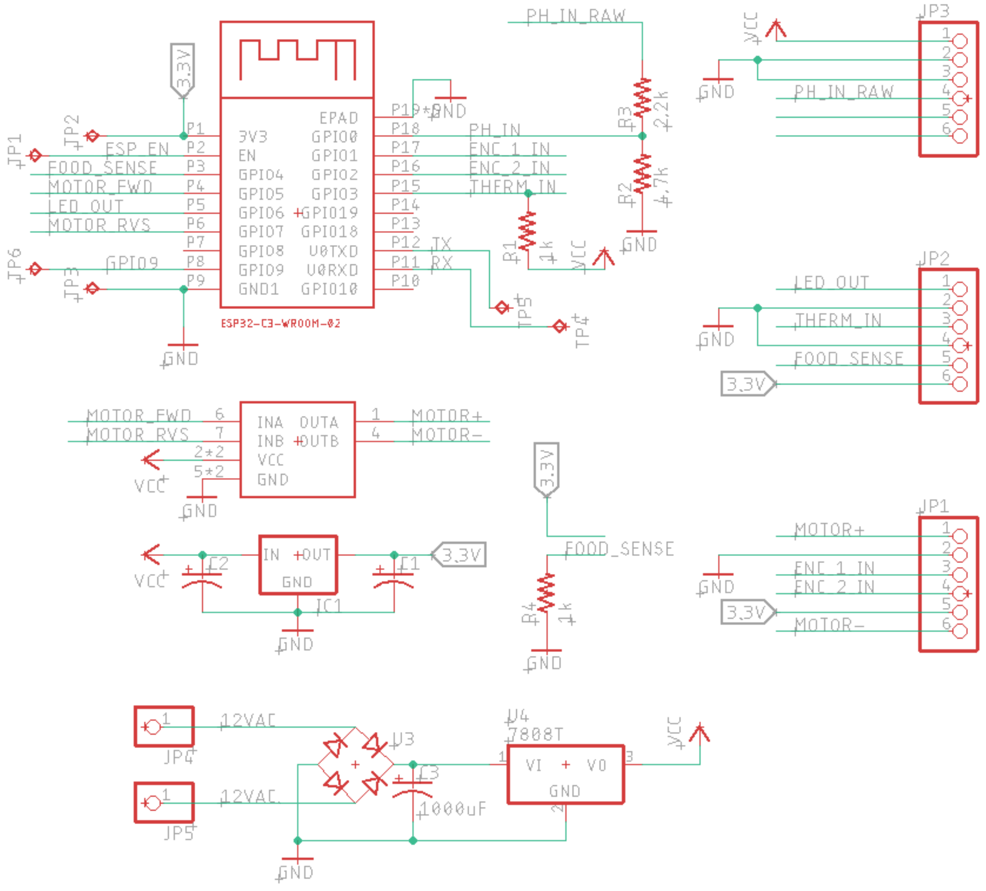
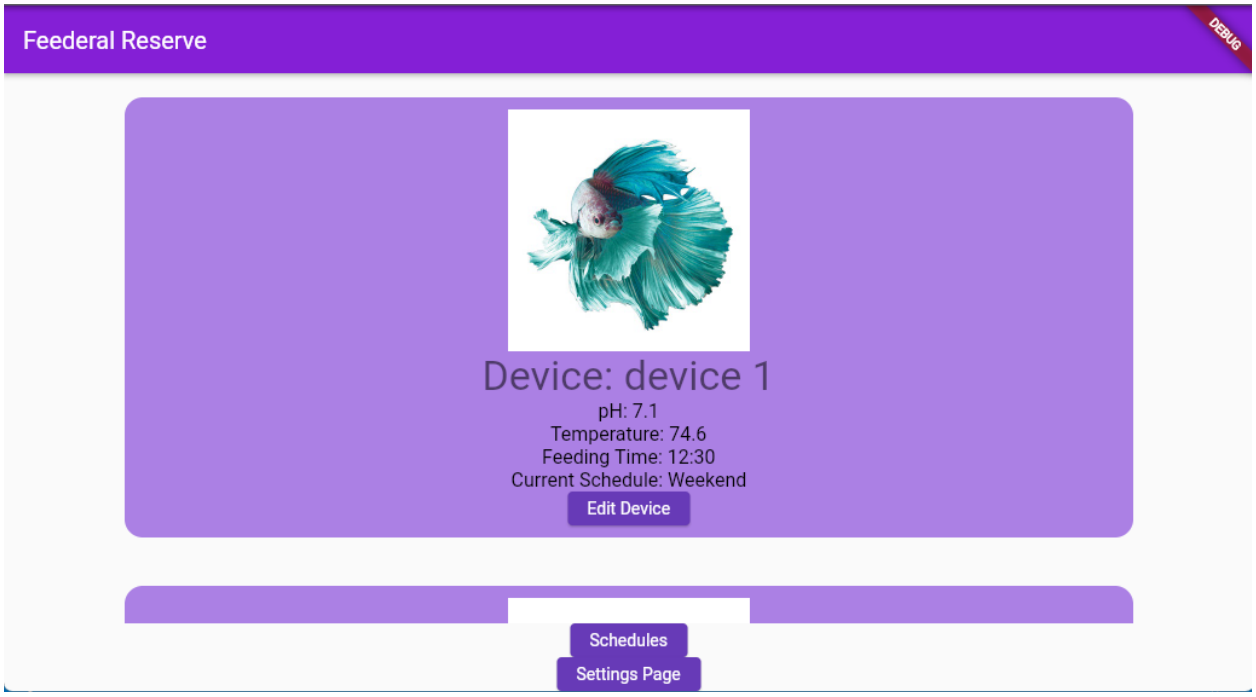


Figure 13: Pin diagram of control and measurement circuit

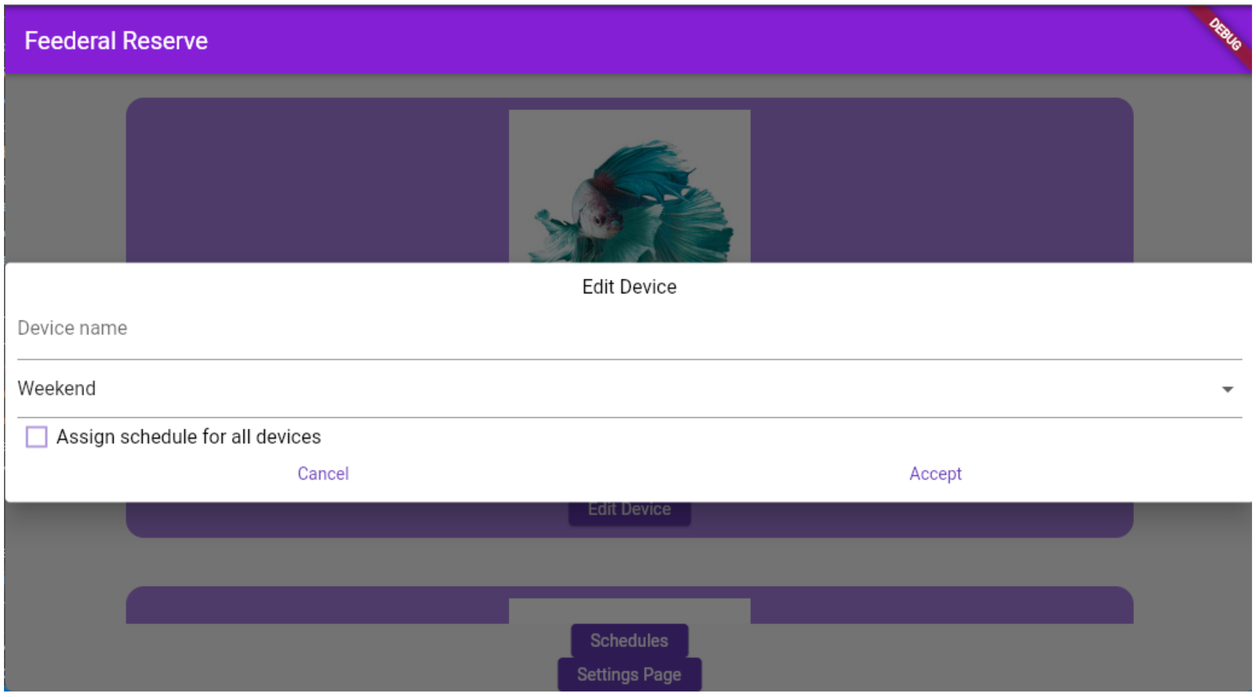
The application associated with the device contains a web-based application, with a proper frontend hosted on GitHub Pages, a backend, and a server hosted on a Raspberry Pi. Within the actual application, there are three distinct pages. The first of which is a dashboard consisting of the different devices that are connected to the account signed in along with the tank conditions observed such as pH, temperature, their feeding schedule, and the time the fish are scheduled to be fed. An example of depiction of a dashboard is shown in Figure 14.



pH: 7.1
Temperature: 74.6
Feeding Time: 12:30
Current Schedule: Weekend

Figure 14: Dashboard page with a sample device and current status

If the user selects the “Edit device” button below their respective fish, the user is able to select which schedule is being utilized to feed their fish.



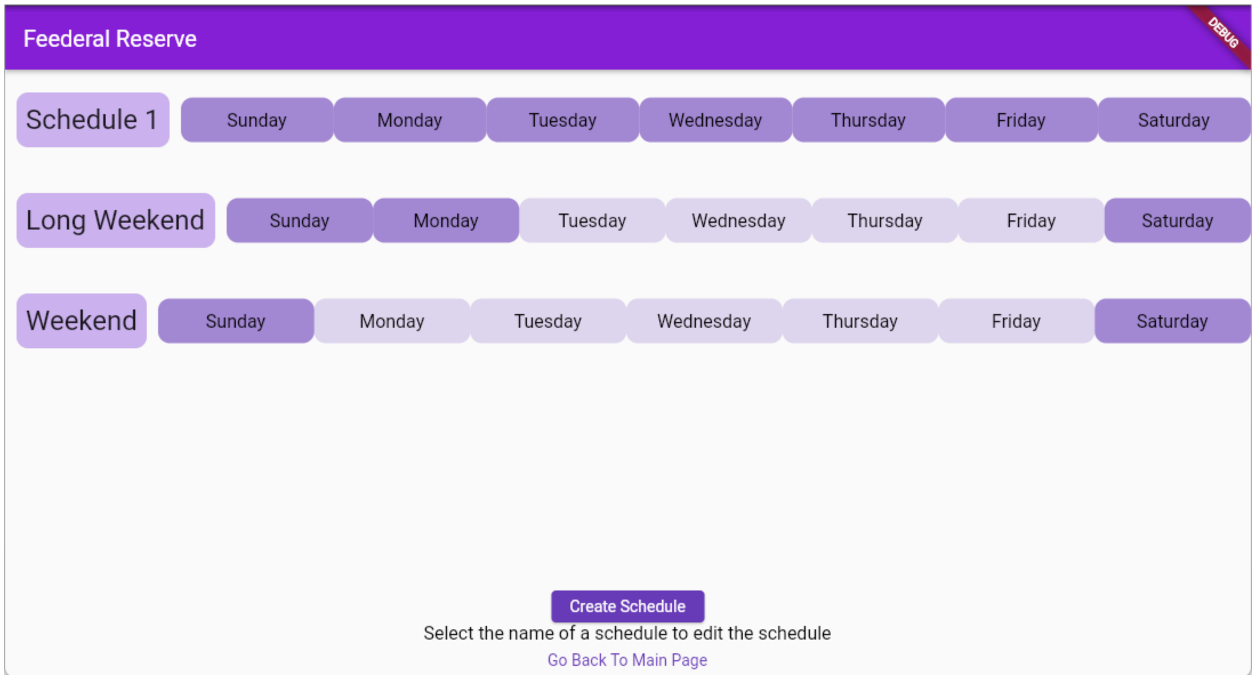
Device name

Weekend

☐ Assign schedule for all devices

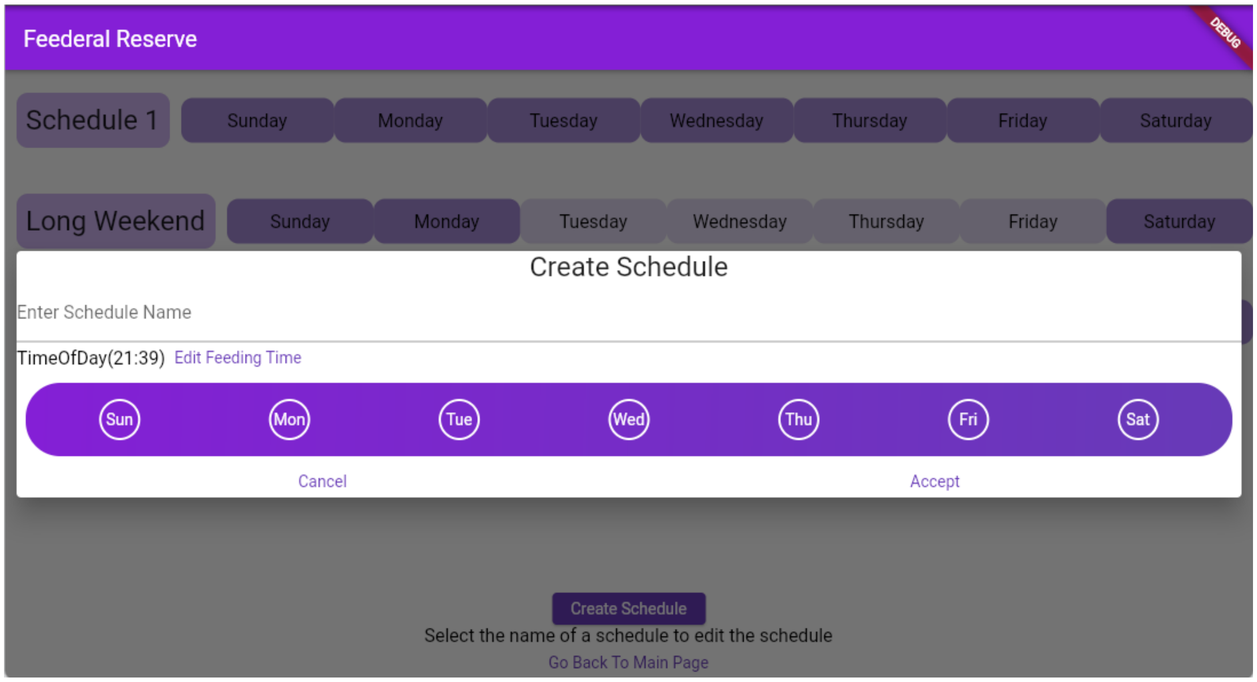
Figure 15: Edit Device Dialog brought up by hitting ‘Edit Device’ Button

The next page associated with the web application is the scheduling page. On this page, the user can view a schedule based on the ones they have previously created or create new schedules, including schedule names, times of feeding, and days of automated feeding, as shown in Figure 16. Of the schedules already created, the user will have the option to utilize their schedule as is or they have the option to edit it, an example is exhibited below in Figure 17. When selecting days, the highlighted days are the selected days, while the "empty" bubbles are unselected.



Select the name of a schedule to edit the schedule

Figure 16: Main schedule page with sample schedules already added



TimeOfDay(21:39)

Select the name of a schedule to edit the schedule

Figure 17: Example of creating a new schedule

The last page of the application is the settings page as seen in Figure 18 below. Within this page, the user has the option between two color themes. The first of which is the typical Iowa State University cardinal red and gold. The second color scheme is dedicated to the clients who adore the color purple. As the purple has been shown above, the red and gold color palette can be seen below in Figure 19.

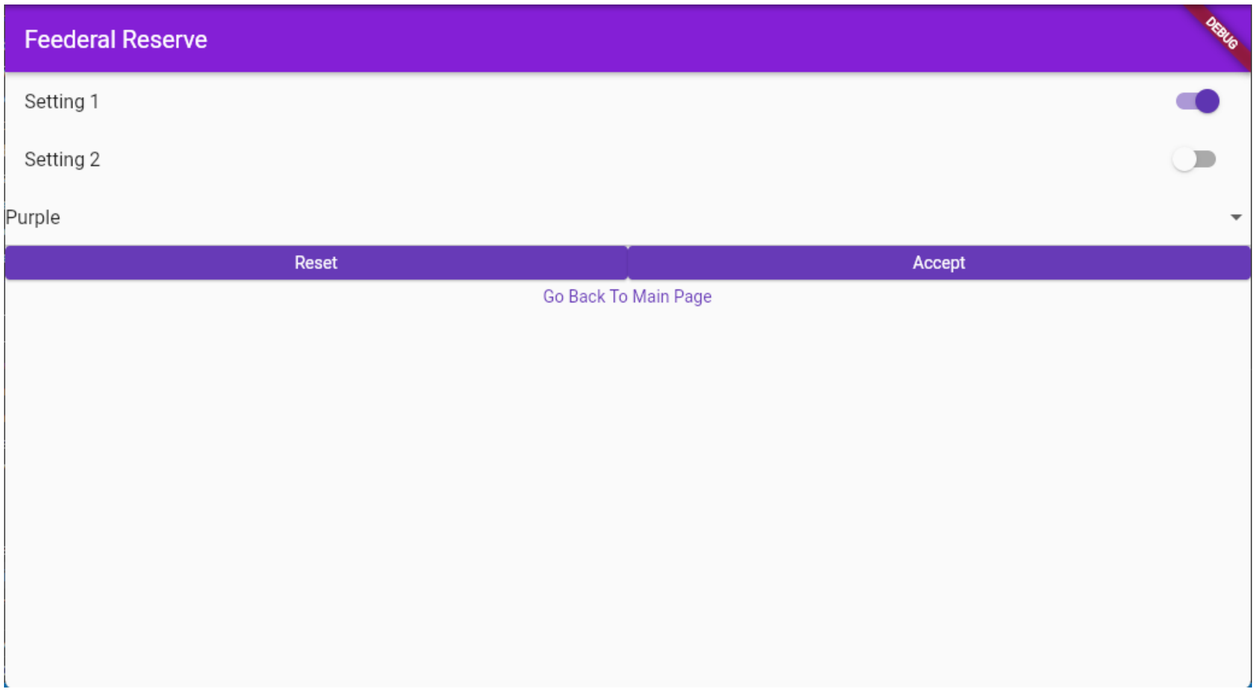


Figure 18: Display of Settings Page

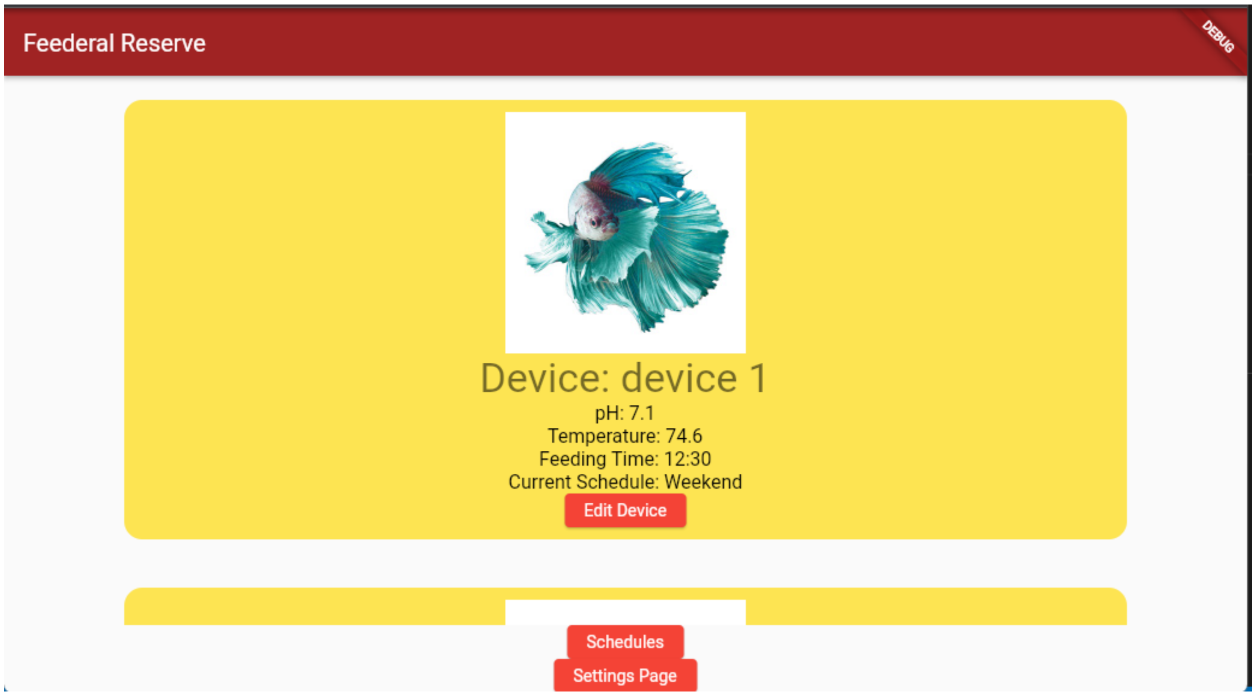


Figure 19: Display of alternate cardinal red and gold color theme

Lastly, the connection between the application and the server is initiated when the frontend makes http post and get requests using JSON to get information from the server. Both fetchAllDevices and fetchAllSchedules are called in the init_state functions for both the Dashboard and Schedules page. Whenever the user creates a new schedule (in the Create Schedule dialog) or changes which schedule a device is using (in the Edit Device dialog), it makes a post request and sends the newly updated information to the server. As it is currently implemented, the page will need to be reloaded for the new data from the server to be displayed.

7 Professional Responsibility

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", International Journal of Engineering Education Vol. 28, No. 2, pp. 416–424, 2012.

7.1 Areas of Responsibility

Table 5 compares IEEE standards to the areas of responsibility found in the before stated document.

Area of responsibility	IEEE standards that fit	In our own words	How the two differ
Work competence	to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations	Improve skills, but do not take on tasks that you cannot do without admitting it.	NSPE includes high quality work
Financial responsibility	to avoid unlawful conduct in professional activities, and to reject bribery in all its forms	Do not take bribes or break the law.	NSPE only has has deliver reasonable costs
Communication honesty	to avoid injuring others, their property, reputation, or employment by false or malicious actions, rumors or any other verbal or physical abuses	Do not make false claims about work or other people.	NSPE includes stakeholders
Health, safety, well-being	to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	Do not let your work be able to hurt others. And if it could, let it be known right away.	NSPE and IEEE are ont much different for this one
Property ownership	to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit	Take ownership of your work/mistakes and give credit where credit is due.	NSPE has nothing on taking criticism

	properly the contributions of others		
Sustainability	to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	Do not create a design that knowingly destroys the planet.	NSPE has a whole section for sustainability, where IEEE just slips it in with other ideas
Social Responsibility	to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression; to not engage in harassment of any kind, including sexual harassment or bullying behavior; to support colleagues and co-workers in following this code of ethics, to strive to ensure the code is upheld, and to not retaliate against individuals reporting a violation.	Be a decent human and treat everyone with the respect they deserve, not the amount you think they should have.	IEEE dives deeper into social responsibility

Table 5: IEEE ethics code compared to the seven areas of professional responsibility

7.2 Project Specific Professional Responsibility Areas

Work Competence is highly applicable to the project because giving anything less than our best would not be nice to Cara or Jamie. They are expecting a good product, and we need to make sure that happens. Currently we are performing at a medium level because each section (frontend, backend, and mechanism) are slowly coming together. We underestimated the time it takes to do each portion, but also built in some cushion to our timeline.

Financial Responsibility is applicable because even though the parts we are using are not too expensive, if we are not careful with prototyping the costs can add up. The team has a high level of performance here because we are reusing parts from the prototype to create the final design.

Communication Honesty is applicable because the only way we can create a great product is to communicate with the team, clients, and advisor. Our level of performance is medium because we are communicating every other week with the client and every week with our advisor, but we are not always clear on how the meeting will take place. We are working on this by making sure to schedule the meeting at least two weeks in advance to ensure we can meet in person. Another reason for a medium rating is because we tell our client like it is. We were originally supposed to have an iOS app, but due to time constraints we decided to only have a web app. We were very open to our client on why we needed to drop one of their original wishes.

Health, Safety, Well-Being is applicable because we want to keep the office's fish healthy and safe to ensure that the office's environment is still what our clients are used to. The team's performance level is currently medium because we have not been able to test the device in the office, but we are keeping in mind how well our design blends into the office's atmosphere.

Property Ownership is very applicable because as we develop this fish feeder, we must remember that our client expects our best work and will work with this product after we graduate. Our current performance is at a high level because we are passionate about the project and want to create the best product that we can so our client can continue using the feeder for years to come. We would not feel content with the final design if we only gave half of what we are able.

Sustainability is not as applicable to this project because the device's power consumption is low. But, we are trying to build the final feeders with parts from the prototypes so we waste the least possible number of parts. We think that our performance level is currently high because we only had to get one more major part for the final design - a temperature sensor. We originally looked at two different temperature sensors to see which one we thought was better, so we needed another one so both of the devices have the same layout.

Social Responsibility is applicable because this automatic fish feeder will benefit our clients for years to come. They will not have to worry about their fish over weekends or longer breaks and can spend that time and energy on other activities. We are currently at a medium performance level because we have received approval for the case design, but have yet to give the office a product to test or keep.

7.3 Most Applicable Professional Responsibility Area

The most applicable professional responsibility area for this project is Property Ownership because this is not a product the team will use for years, but our clients will. When someone makes something for themself they have pride in whatever they ended up with because they built it. But when someone is developing a product for someone else, they need to take ownership of their work and do the best they can so the client is also proud of the product.

8 Closing Material

8.1 Discussion

The final design meets the requirements of dispensing three pellets at a time, storing at least a week of food, measuring pH within 0.1pH and temperature within 1℉, controlled through a web based app, and the ability to set a feeding schedule. The requirements we were not able to implement are manually feed by a button in the app, send alerts, and the ability to change the color scheme.

8.2 Conclusion

Our goal was to have a feeding mechanism that can measure the pH and temperature of the water and be controlled by a web app that had the ability to schedule feedings, change color settings, and feed directly from the app. We started by drawing out designs on paper and then moved to designing in Flutter and SolidWorks for the app and feeding mechanism respectively. Time was our biggest constraint. There was so much we wanted to do, but we did not realize how long each task would take. To achieve the goals in a future design iteration, a set schedule should be used with each task enumerated by importance.

8.3 References

AquariumStoreDepot, The 7 Best Automatic Fish Feeders - 2023 Review, 2023.

8.4 Appendices

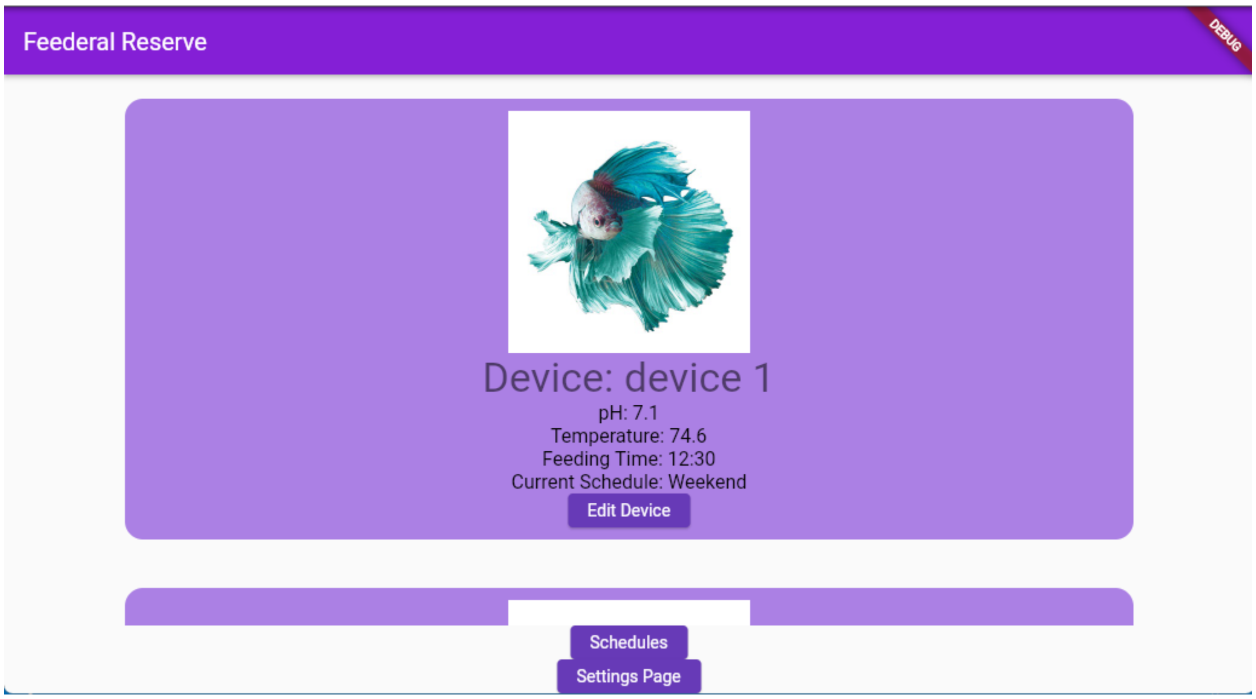
Appendix I - User/Operation Manual

Feederal Reserve Automatic Fish Feeder User Manual

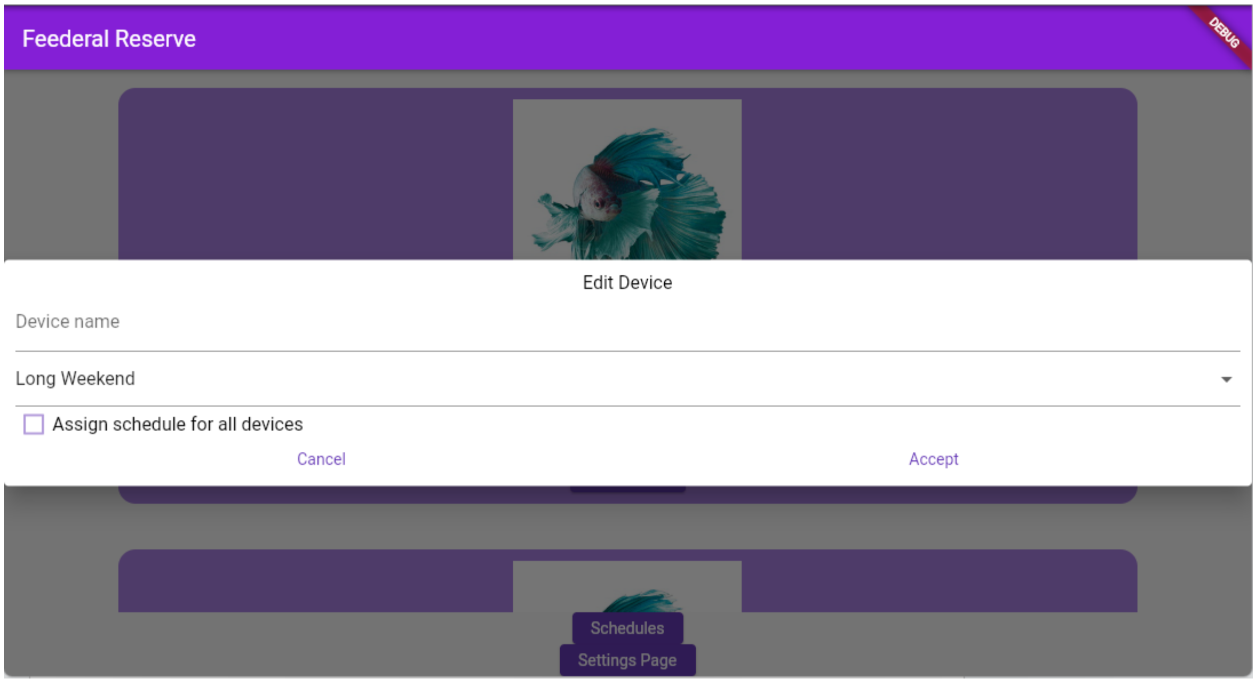
Navigating the App

Dashboard

On the dashboard, you can view all the devices on your fish tanks. This will show you the current status for the tank's pH, temperature, and schedule. Once each device tile there is an "Edit Device" button. When this is clicked it will pop up a menu where you can change the device name and the schedule for the device. You can also apply the same schedule to all other devices via this menu.

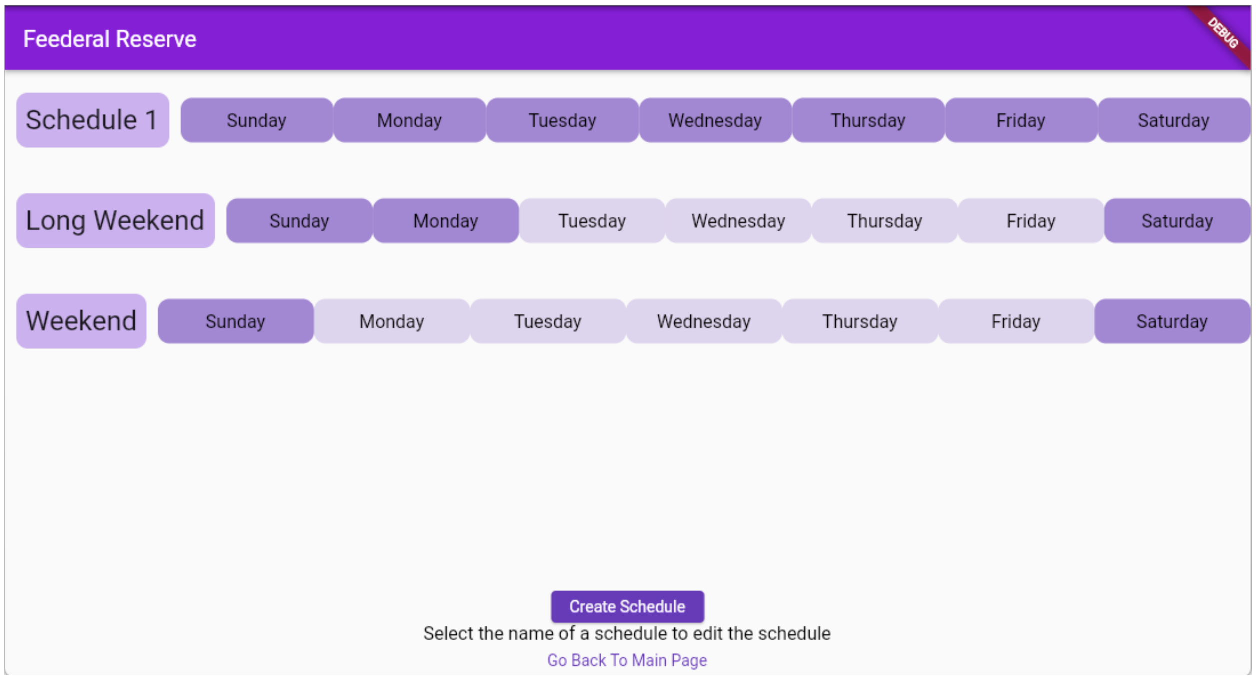


Device: device 1
pH: 7.1
Temperature: 74.6
Feeding Time: 12:30
Current Schedule: Weekend



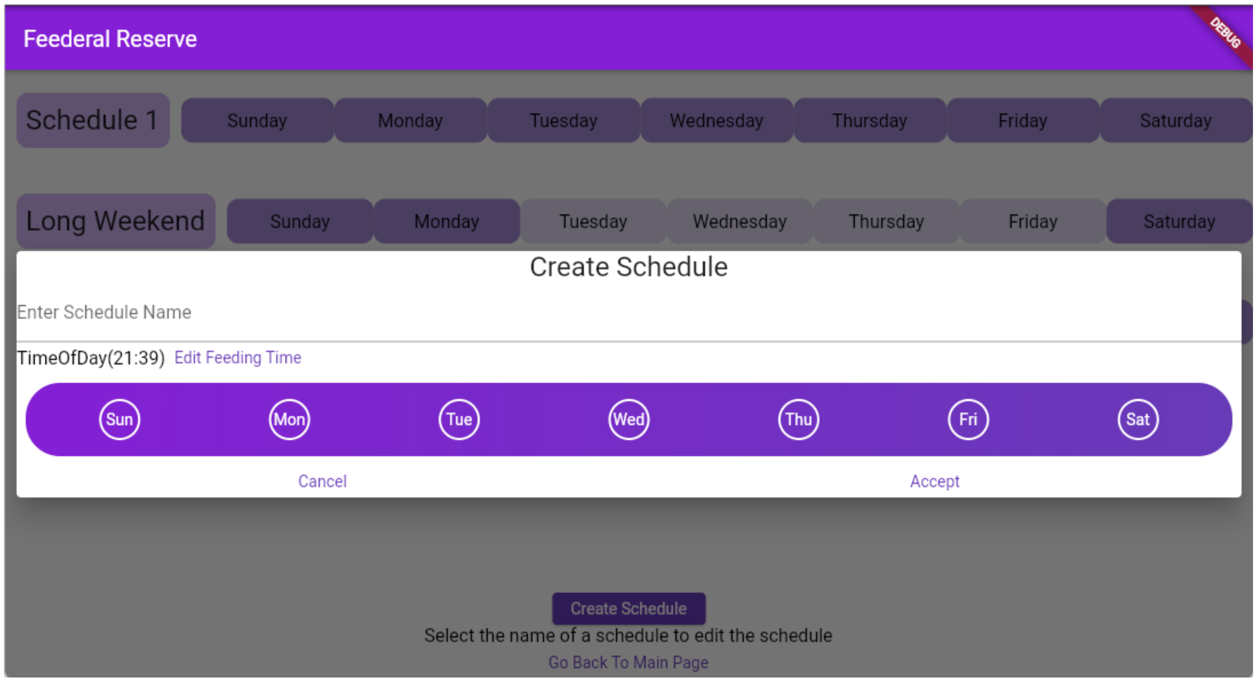
Schedules

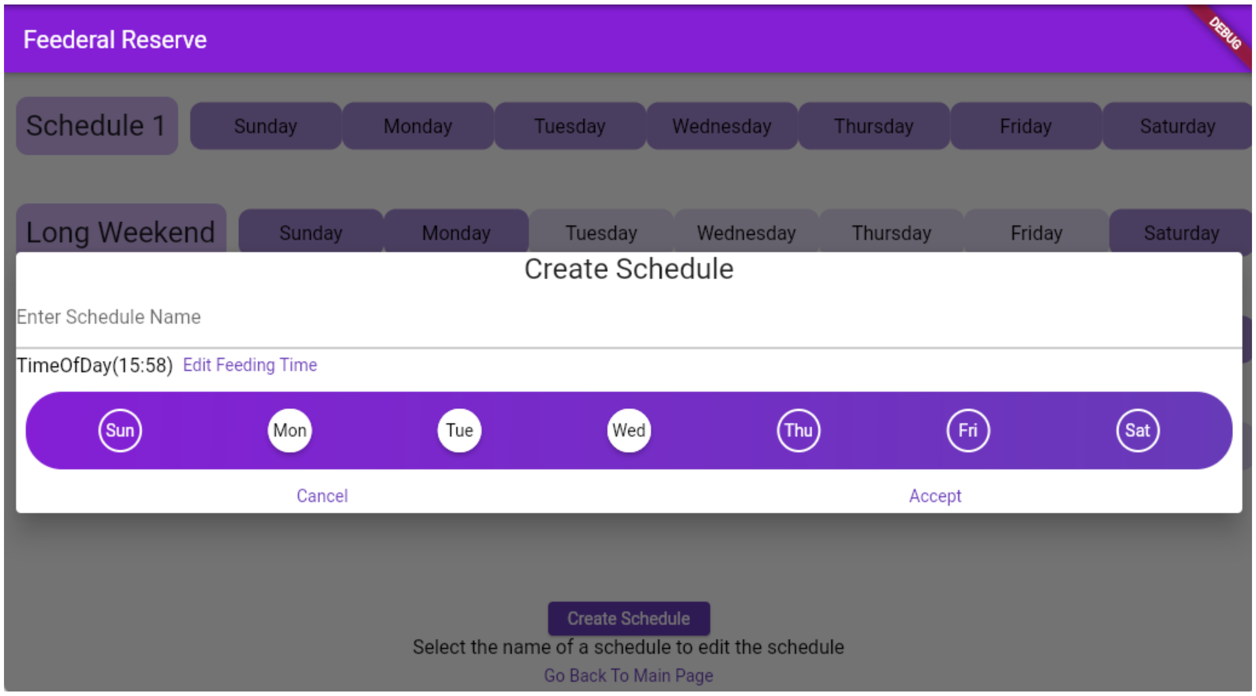
This page shows all schedules that have been created with the feeding days being the darker color of purple and the lighter colored days being manual feeding days. From here, you can create and edit your schedules. For the creation or modification of schedules, the colors will be inverse from what was previously stated. Therefore, the day of the week selected to have a scheduled feed will be a lighter color, filled circle and the manual feeding days will be the darker color, unfilled circle.



Schedule 1	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Long Weekend	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Weekend	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday

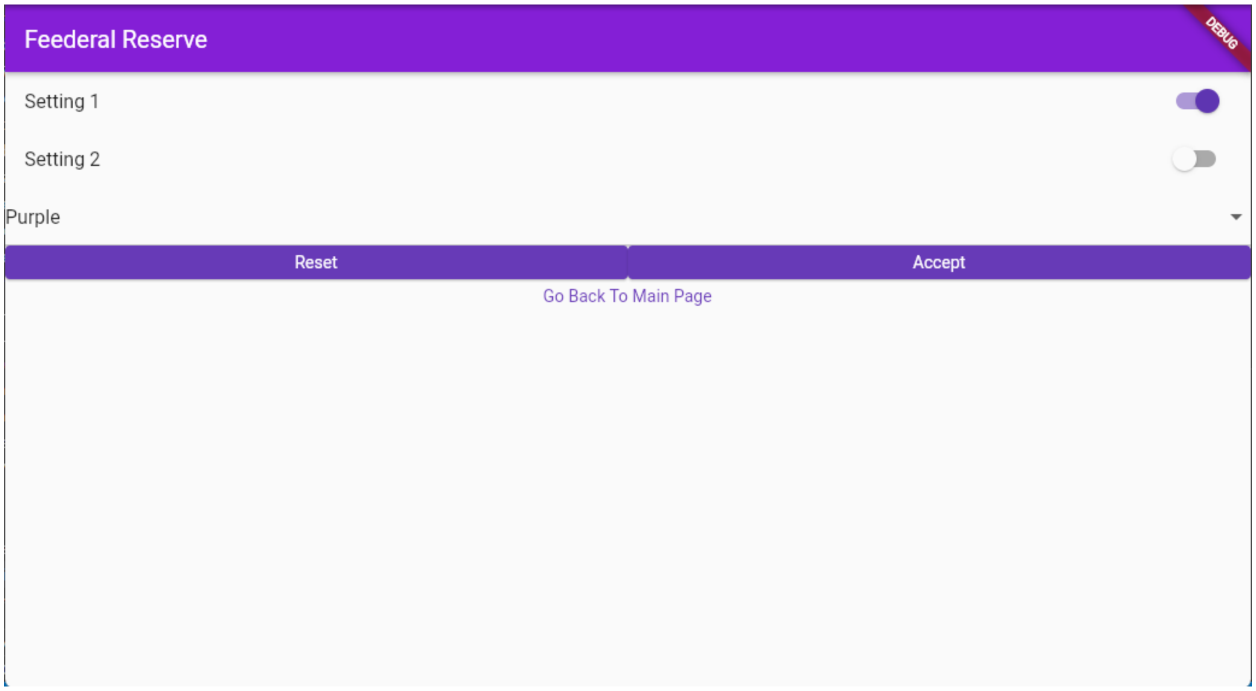
Select the name of a schedule to edit the schedule





Settings

We have a settings page implemented, but currently does not make permanent changes in the server/app, so therefore it is a placeholder for future development.



Setting 1

Setting 2

Purple

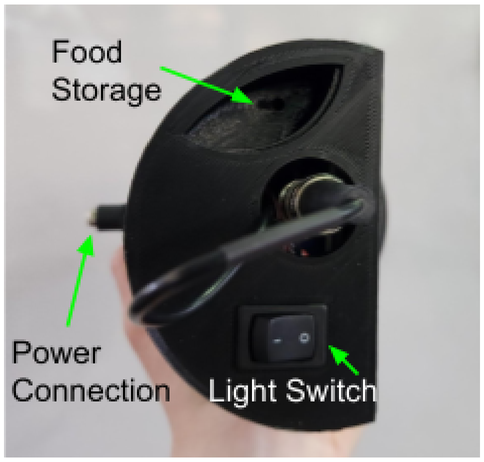
Reset

Accept

Feeding Device

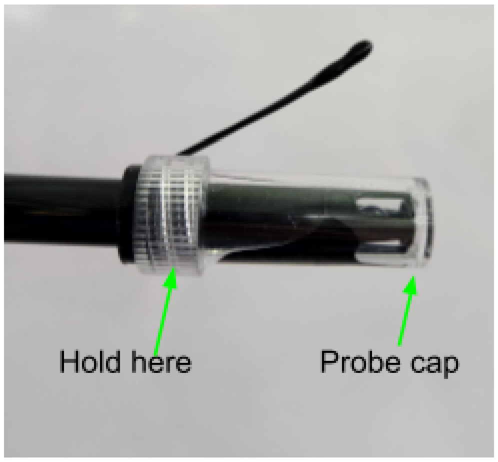
Using the Device

The device should be placed on top of the fish tank and will replace the current LED lid of the tank. The device will need to be plugged into the power outlet using the same power adapter as the original lid. The two objects protruding from the bottom of the device will both need to be submerged into the tank in order to get an accurate reading. The switch at the top is used to turn on the LEDs for lighting up the tank. The food pellets need to be loaded into the top portion of the device, on the opposite side from the switch, and this will feed the pellets down into the device.



Tank Cleaning and the Device

When cleaning the tank, the pH sensor must not be allowed to dry out. Keep the end submerged in water at all times. The cap that came with the pH sensor can be used to keep the end moist. Fill the cap about one third of the way with water and twist the cap on while holding the clear plastic piece that it twists into. The cap will not screw in well if you hold the black casing and could possibly dislodge the probe from the case.



Appendix II - Team Contract

Team Name ____Con-sea-erge____sdmay23-43______

Team Members:

1) _Adan Maher____________________ 2) _Sarah Degen___________________
3) _Nathan Paskach_________________ 4) _Melanie Fuhrmann______________
5) _Jack Croghan___________________ 6) _Bella Leicht___________________
7) _Brandon Mauss_________________

Team Procedures

1. **Day, time, and location (face-to-face or virtual) for regular team meetings:**

Meeting with Advisor:

Semester 1: Fri 9:00-9:30 AM, Weekly, virtual

Semester 2: Mon 1:15-2:15 PM, Weekly, f2f (Dr. Fila's Office)

Meeting with Client:

Semester 1: First Thurs 2:30-3:30 PM Monthly, f2f if possible (COA Office)

Semester 2: Mon 1:15-2:15 PM, Every other week, f2f (COA Office)

Meeting with Team:

Semester 1: Mon 8:30-9:30 AM, Weekly, f2f (library)

Semester 2: Mon after client/advisor meetings, f2f (TLA)

2. **Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):**

Snapchat/Discord

Face-to-face meetings

3. **Decision-making policy (e.g., consensus, majority vote):**

Consensus, and if unable to be reached, bring it up to the team to discuss/debate, then to the advisor if consensus is still unable to be reached.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Separate documents/sections for notes and questions for client, advisor, and team.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

Attendance is expected unless outside conflicts occur, then let the team know ahead of time (at least 2 hrs)

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadline

Complete tasks unless blockers arise, then communicate those, explore delegation.

3. Expected level of communication with other team members:

Try to answer questions in a timely manner, at least within a day.

4. Expected level of commitment to team decisions and tasks:

Everyone contributes to the team to give the fish the best end product possible

Leadership

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

Even ownership between team members as much as possible

Adan: Organization

Sarah: Meeting note taking, circuit design

Nathan: Firmware design

Melanie: Client interaction/ Frontend app dev

Jack: Software Component Design

Bella: Aquatic care

Brandon: Component Design

2. Strategies for supporting and guiding the work of all team members:

Agile/Scrum style task assignment. Component teams can help support each other on technical aspects

3. Strategies for recognizing the contributions of all team members:
At team meetings, explain what we have been working on for the previous week and what you want to work on for the upcoming week

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
Adan: Pentesting, Physical Security, Network Security, Backend Development
Sarah: Circuit design
Nathan: Digital circuit design, embedded systems, soldering
Melanie: Front end app development, Gitlab, embedded systems
Jack: Backend development, Software Design
Bella: System pentesting, network security, hardware security
Brandon: Circuit Design, Component Design
2. Strategies for encouraging and supporting contributions and ideas from all team members:
Listen to each other and give constructive feedback. Discuss professionally
3. 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?)
Discuss in a calm, professional way. Be open to constructive criticism and work together to build an open environment.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
Try and get a minimally viable product (very ugly but technically working prototype) to prepare for creating a more hardened product next semester
2. Strategies for planning and assigning individual and team work:
Work will be discussed at team meetings and potentially within component teams for more specific tasks.
3. Strategies for keeping on task:
Have a specific goal laid out at the beginning of the meeting, and someone to bring it back in focus.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
Talk to the person who has breached the contract and find out why it happened, and collectively come up with a way to alleviate the issue.
2. What will your team do if the infractions continue?
Bring up the issue to the advisor.
****

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) __Adan Maher______________________________________ DATE ___09/22/22________
2) __Sarah Degen______________________________________ DATE ___09/22/22_______
3) __Nathan Paskach___________________________________ DATE ____09/22/22_______
4) __Bella Leicht ______________________________________ DATE ____09/22/22_______
5) __Melanie Fuhrmann _________________________________ DATE ____09/22/22_______
6) __Jack Croghan______________________________________ DATE ____ 09/22/22______
7) __Brandon Mauss____________________________________ DATE _____09/22/22______