

A Smart Aquatic Ecosystem

DESIGN DOCUMENT

sdmay26-21

Md Maruf Ahamed (Advisor/Client)

Carson Irving (Frontend/Backend/AI)

Max Purvis (Hardware)

Daniel Pender (Hardware)

Aden Koziol (AI)

Marina Mannan (Embedded Systems)

Griffin Urban (Frontend)

Ryan Holden (Backend/Frontend)

sdmay26-21@iastate.edu

sdmay26-21.sd.ece.iastate.edu

Revised: Dec. 7th, 2025

Executive Summary

The Smart Aquatic Ecosystem is an AI assisted smart aquarium that automates much of the work required to care for aquatic life, creating a healthier environment for the fish and reducing the responsibilities placed on the fish tank's owner.

Problem and Importance

Traditional aquarium maintenance requires frequent manual monitoring of water quality and fish health, which can lead to inconsistent care and increased stress for aquatic life. Our Smart Aquatic Ecosystem aims to utilize AI and automation to eliminate the need for constant manual adjustments, thereby creating a more stable environment for fish and other aquatic life.

Together, this system addresses all the user's needs by eliminating the need for constant manual care. It monitors water quality and fish health, performs automatic feedings and water replacement, and alerts the user to any urgent changes in the environment.

Design Requirements

The two most important design requirements are automated monitoring and controlling of the environment inside the fish tank. Ideal water conditions can be created by controlling measurable variables of the water quality, including pH, temperature, dissolved oxygen, and turbidity.

Design and Technologies Used

Our implementation will utilize smart sensors to track this data in real-time and visualize it on a web-based application for the user's benefit. Automated controls will regulate temperature, feeding, and water exchange/flow. Additionally, the system will track the fish population with a camera using an AI-driven computer vision model. This data will be used along with water quality patterns to model population trends and predict fish health issues. The web app will display records of every adjustment made to the environment and notify the user to recommend any necessary actions to ensure the continued health of the ecosystem.

Key Components and Progress so far

- Hardware
 - Water Quality Sensors: Turbidity Sensor, Temperature Sensor, pH Sensor, Dissolved Oxygen Sensor
 - Arduino UNO R4 Wifi
 - Fish Tank Heater
 - Feeder prototype with servo motor
- Software (Backend)
 - SQLite Database
 - .NET 9 Web API Application (REST API)
 - Nginx Web Server with Reverse Proxy
 - Certbot auto-renewing HTTPS Certificate

- Software (Frontend)
 - Web-based dashboard built using React, the dashboard displays:
 - Temperature, oxygen, water clarity, and pH are each shown on a line graph
 - Most recent image taken by the camera
 - Admin panel for manually testing database functionality
- AI
 - YOLOv8n AI vision model
 - Python script using TensorFlow
 - Fish_detection training data

Next Steps

- Purchase fish, lighting, and tank decorations
- 3D print the tank lid
- Set up the aquarium and begin training the AI model

Learning Summary

Development Standards & Practices Used

Software

- **Formatting:** camelCase and 4-space tabs for programming language
- **API:** .NET 9 WEB API for backend.
- **Documentation:** Descriptive Git commits and work on individual branches before merge approval

Hardware

- **Organized Wiring:** All wiring is routed cleanly to minimize clutter and maintain a clear, easy-to-follow layout.
- **Color Coordination:** Uniform colors are used for power, ground, and data wires to reduce wiring errors.
- **Proper Grounding:** A common ground for all components to ensure stable readings and reliable communication between them.
- **Documentation of Design Choices:** All component selections, circuit parameters, pinout, and expected current draw are fully documented.
- **Incremental Hardware testing:** Testing each sensor and component independently before adding to the primary circuit.
- **Modularity:** Hardware subsystems are designed to be modular, allowing components to be replaced or upgraded without requiring the entire system to be rewired.
- **Safety:** Waterproofing is applied to exposed circuit components, and power is supplied correctly to prevent damage and ensure safe operation.

Summary of Requirements

- AI computer vision fish population and growth analysis
- Automatic feeding mechanism using actuators
- Basic Aquarium Equipment (Tank, Filter, Air Pump)
- Advanced Aquarium Sensors (pH, turbidity, water quality, dissolved oxygen, temperature)
- Web-based application to monitor insights, control systems, and real-time visualization of data

Applicable Courses from Iowa State University Curriculum

SE

- 2280 - *Introduction to Data Structures*
- 3090 - *Software Development Practices*
- 3170 - *Introduction to Software Testing*
- 3190 - *Construction of User Interfaces*
- 3630 - *Introduction to Database Management*

CS

- 3110 - *Introduction to the Design and Analysis of Algorithms*

CPRE

- 2880 - *Embedded Systems I*

EE

- 1850 - *Introduction to Electrical Engineering and Problem Solving I*
- 2010 - *Electric Circuits*
- 2300 - *Electronic Circuits and Systems*
- 2850 - *Problem-Solving Methods and Tools for Electrical Engineering*

New Skills/Knowledge acquired that were not taught in courses

- Fishtank maintenance and proper installation/use
- Fish care and maintenance
- TailwindCSS
- Typescript
- Calibration procedures for pH and dissolved oxygen sensors
- CAD software

Table of Contents

1. Introduction	1
2. Requirements, Constraints, And Standards	2
3 Project Plan	5
3.1 Project Management/Tracking Procedures	5
3.2 Task Decomposition	5
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	6
3.4 Project Timeline/Schedule	7
3.5 Risks and Risk Management/Mitigation	7
3.6 Personnel Effort Requirements	9
3.7 Other Resource Requirements	10
4 Design	11
4.1 Design Context	11
4.1.1 Broader Context	11
4.1.2 Prior Work/Solutions	12
4.1.3 Technical Complexity	13
4.2 Design Exploration	13
4.2.1 Design Decisions	13
4.2.2 Ideation	13
4.2.3 Decision-Making and Trade-Off	14
4.3 Proposed Design	15
4.3.1 Overview	15
4.3.2 Detailed Design and Visual(s)	16
4.3.3 Functionality	18
4.3.4 Areas of Concern and Development	19
4.4 Technology Considerations	20
4.5 Design Analysis	22
5 Testing	23
5.1 Unit Testing	23
5.2 Interface Testing	24
5.3 Integration Testing	24
5.4 System Testing	24
5.5 Regression Testing	25
5.6 Acceptance Testing	25
5.7 Security Testing	25
5.8 User Testing	25
5.9 Results	26
6 Implementation	28
7 Ethics and Professional Responsibility	31
7.1 Areas of Professional Responsibility/Codes of Ethics	31
7.2 Four Principles	33

7.3 Virtues	34
8 Closing Material	37
8.1 Conclusion	37
8.2 Appendices	39
9 Team	40
9.6 Team Contract	41

List of Figures

- Figure 1 (page 7): Project Schedule
- Figure 2 (page 14): Software Components
- Figure 3 (page 15): Tank Lid Sketch
- Figure 4 (page 15): Automatic Feeder Design
- Figure 5 (page 16): Camera
- Figure 6 (page 16): Backend Code Structure
- Figure 7 (page 17): Arduino Circuit
- Figure 8 (page 24): Frontend Test Coverage
- Figure 9 (page 25): Backend Test Coverage
- Figure 10 (page 26): Arduino Sample Test Results
- Figure 11 (page 26): AI Camera Detection
- Figure 12 (page 27): Arduino Circuit
- Figure 13 (page 28): Assembled Arduino Circuit
- Figure 14 (page 28): Feeding System

1. Introduction

1.1. PROBLEM STATEMENT

Keeping an aquarium may sound simple, but it's actually a delicate process where small changes in water quality can have a significant impact on the fish's health. Most aquarium owners stick to a maintenance routine, but what happens when life gets in the way? Our project aims to address this issue with a smart, AI-powered aquarium management system that continuously monitors water conditions in real-time and responds automatically. By using sensors and intelligent controls, the system can regulate temperature, feeding, water flow, and even perform water changes when needed. There will also be artificial intelligence tracking the growth and population of the fish to provide more insights for the owners. This creates a healthier aquatic environment for the fish while reducing the workload for busy owners. The insights gathered will be easily accessible remotely via a web interface or application, providing peace of mind for owners who may be away from the tank.

1.2. INTENDED USERS

Who will use the product you create? Who benefits from or will be affected by the results of your project? List as many users or user groups as are relevant to your project (at least three). For each user or user group, (1) describe the user and their key characteristics (e.g., a persona), (2) identify their need(s) related to the project (e.g., a needs statement), and (3) discuss how they might benefit or derive value from the product you create. Justify how these benefits/this value connect to your overarching problem statement.

User Group 1: Casual Aquarium Owners

Description: Individuals or families who have a fish tank at home. They typically have limited knowledge of aquarium care and little time for regular maintenance.

Needs Statement: They need an easy and low-effort way to keep their fish alive and healthy without having to constantly monitor things such as water quality, feeding, and cleaning.

Benefits: The AI tank provides automation for feeding, monitors water quality, and tracks the health of the fish. This makes for a more accessible and less stressful experience for casual fish owners.

User Group 2: Commercial Users

Description: Commercial users, such as pet stores, need to manage several tanks with different species of fish. The staff at these stores may not always be experts, making consistent care across many tanks challenging.

Needs Statement: They require a scalable solution that ensures consistent water quality, reduces labor costs, and prolongs fish survival.

Benefits: AI automation will reduce manual monitoring and feeding, and maintain standardized care across tanks. This will save costs and increase reliability.

User Group 3: Enthusiasts and Hobbyists

Description: Experienced aquarium owners who might have more expensive fish and setups.

Needs Statement: They need accurate, detailed data and precise environmental controls to optimize their tank conditions, monitor fish health, and experiment with advanced setups.

Benefits: The AI system provides real-time feedback on parameters such as temperature, water quality, and pH. This will allow enthusiasts to achieve higher success rates with their many different fish, making advanced care more manageable.

User Group 4: Fish

Description: A Variety of species of fish that will be living inside the tank.

Needs Statement: They require precise temperature, water quality, oxygenation, and regular feeding to survive and remain comfortable.

Benefits: The hardware on board will automatically track all water requirements, ensuring fish safety and alerting owners to any abnormalities. Additionally, fish will be fed automatically, reducing the risk of not being fed due to forgetfulness.

2. Requirements, Constraints, And Standards

2.1. REQUIREMENTS & CONSTRAINTS

Functional requirements:

- Sensors:
 - pH sensor should read pH accurately within (+/-) 0.1 pH - **constraint**
 - Temperature sensor should read accurately within (+/-) 1°F - **constraint**
 - Dissolved oxygen sensor should read accurately within (+/-) 1 mg/L - **constraint**
- Feeding Mechanism:
 - Should hold up to a week's worth of fish food

User Experiential Requirements:

- Application:
 - Should have a response time of no greater than 1 second - **constraint**
 - Should display any sensor range outside of their standard set value
 - Should allow the user to schedule feeding intervals and times
 - Should monitor trends in sensor data and feeding patterns
 - Should display tank conditions, quality, and population statistics

Environmental Requirements:

- Electronics housing should be water-resistant
- Electronics should not interfere with water quality, dissolved oxygen, and other water metrics

Physical Requirements:

- Should fit inside the lid of the tank
- Must not restrict tank accessibility
- Components should be mounted securely

Economic Requirements:

- The final product must cost less than \$500 to construct - **constraint**
- Components should be chosen for long-term reliability to minimize replacement costs.

Aesthetic Requirements:

- There should be minimal visible wires, boards, or electronics to the user
- Should not massively increase the footprint of the fishtank

AI Requirements:

- The machine learning model should be able to decipher fish in the image with ~95% accuracy
- The time from the camera to the frontend should be under 200 ms

2.2. ENGINEERING STANDARDS

Standards Importance:

Engineering standards are important because they ensure safety, reliability, and consistency in design and production. They help engineers follow proven methods, ensure systems are compatible with one another, and maintain quality across various projects. Standards also make communication clearer between teams and help prevent errors or failures in engineering work.

ISO/IEC 42001:2023 - Information Technology - Artificial Intelligence - Management System

- **ISO/IEC 42001:2023:** This is an international standard that specifies requirements for establishing, implementing, and maintaining an artificial intelligence system. It is intended for entities that design or utilize AI-based products. It is essential because it is the world's first AI management system standard.

IEEE 830-1998: Recommended Practice for Software Requirements Specifications

- **IEEE 830-1998:** This standard provides structured guidelines for writing and organizing the requirements for a software system. It helps ensure that every functionality, such as water monitoring, feeding automation, and data logging, is clearly described and testable. The goal is to enhance communication among developers, engineers, and stakeholders, ensuring that everyone shares a common understanding of the system's requirements.

IEEE 1016-2009: Systems and Software Engineering — Software Design Descriptions

- **IEEE 1016-2009:** This standard outlines how to document a system's software design properly. It includes how to describe system architecture, interfaces between modules, data flows, and logic control. For the fish tank project, this means clearly defining how sensors, microcontrollers, and the AI module interact, for example, how temperature or turbidity data are processed and used to automatically adjust pumps or lights.

Project Relevance:

- **IEEE 830-1998** is relevant because it helps define precise software requirements for each automated task, such as feeding, lighting, and water control. This ensures the system functions as intended and reduces design errors.
- **IEEE 1016-2017** is important for documenting how sensors, controllers, and the AI system work together. It supports easier debugging and future updates.
- **ISO/IEC 42001:2023** is relevant because it promotes safe and responsible use of AI. It helps ensure the AI makes reliable decisions and operates ethically within the fish tank system.

Additional Standards:

- **IEEE 29119-2021:** Software and Systems Engineering - Software Testing - Test Processes
- **IEEE 16085-2020:** Systems and Software Engineering - Life Cycle Processes - Risk Management

Modifications:

- **IEEE 29119-2021:** To incorporate this standard into our project, we will outline a testing scope to ensure all components are adequately tested. For example, our testing scope could be sensors, AI population tracking, the feeding system, and water quality calculations.
- **IEEE 16085-2020:** To incorporate this standard into our project, we will create a list of risks to make sure that all are accounted for. For example, a risk could be a wrong turbidity sensor reading. To account for this, we will calibrate the sensor weekly, remove outliers, and calculate averages.

3 Project Plan

3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

We are using agile for our project management style. While we currently have a general idea of the project's goals, there is still considerable room for those goals to evolve and refine. It would be a mistake for us to lock ourselves into a design when we aren't 100% sure of all of the possibilities with the project. Being able to pivot and add new features or change our focus quickly is essential in our project, and agile is the best project management style to support these rapid changes.

We are tracking progress and assigning roles in Plaky, a project management tool similar to Trello. The entire team can see what each person is currently working on and get a general idea of their progress. If more detailed information is needed, we can also examine Git commits in GitLab to see exactly what changes each person is making. We chose to do this because not all aspects of our project can be tracked through git commits, as there are some non-programming problems that we will have to solve. These include automated fish feeding, water cleaning, and water temperature control. While there will likely be some code associated with solving these problems, additional work needs to be accomplished that cannot be tracked through Git.

3.2 TASK DECOMPOSITION

With the help of Plaky mentioned above, we can split the project into smaller tasks and subtasks. Currently, our team has organized a file-tree-like structure to split our functions:

1. Hardware
 - a. Arduino HTTP requests
 - b. Sensor setup
 - c. Feeding system design
 - d. Setup camera
 - e. Lid construction
2. Software
 - a. Frontend
 - i. landing page
 - ii. HTTP receive
 - b. Backend
 - i. Receiving HTTP
 - ii. Database integration
 - iii. Server hosting
 - c. AI
 - i. AI population Tracking
 - ii. Utilize a camera with an AI model

Each of these tasks can be broken down further into smaller subtasks to address individual problems. The linear timeline of completion of tasks is described in section 3.4

3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

To organize our progress, we have divided our tasks into three main categories: Hardware, Software, and AI. Each task within these categories has a particular milestone and metric/evaluation criteria we use to measure our project's progress.

Hardware:

1. Task 1: Arduino HTTP requests
 - a. Milestone: Arduino successfully sends GET/POST requests to the server
 - b. Evaluation Criteria: > 95% requests received successfully
2. Task 2: Sensor setup
 - a. Milestone: Temperature, turbidity, pH, and oxygen sensors calibrated and stable
 - b. Evaluation Criteria: Sensor readings within +/- 5% of reference values
3. Task 3: Feeding system design
 - a. Milestone: Servo motor dispenses food reliably on schedule
 - b. Evaluation Criteria: < 5% deviation from scheduled feed times over 1 week
4. Task 4: Camera setup
 - a. Milestone: Camera captures clear images at defined intervals
 - b. Evaluation Criteria: > 90% usable image rate over 100 captures
5. Task 5: Lid construction
 - a. Milestone: Lid securely houses sensors and camera with waterproofing
 - b. Evaluation Criteria: Passes leak and sensor stability tests

Software:

1. Task 1: Frontend landing page
 - a. Milestone: Responsive UI with tank status and manual controls
 - b. Evaluation Criteria: Loads in < 2s, compatible on mobile and desktop interfaces
2. Task 2: Frontend HTTP requests received
 - a. Milestone: Frontend receives and displays real-time sensor data
 - b. Evaluation Criteria: > 95% data sync accuracy
3. Task 3: Backend HTTP handling
 - a. Milestone: Server receives and parses Arduino data
 - b. Evaluation Criteria: > 95% parse success rate over 100 transmissions
4. Task 4: Backend database integration
 - a. Milestone: Sensor data stored and retrievable from the database
 - b. Evaluation Criteria: < 2s query latency, > 99% data integrity
5. Task 5: Backend server hosting
 - a. Milestone: Server hosted and accessible via local network
 - b. Evaluation Criteria: > 99% uptime

AI:

1. Task 1: AI population tracking
 - a. Milestone: AI model detects and counts fish in tank images
 - b. Evaluation Criteria: > 80% accuracy on annotated test set
2. Task 2: AI camera integration
 - a. Milestone: AI processes live camera feed for fish tracking
 - b. Evaluation Criteria: < 2s latency per frame, > 80% frame classification rate

3.4 PROJECT TIMELINE/SCHEDULE

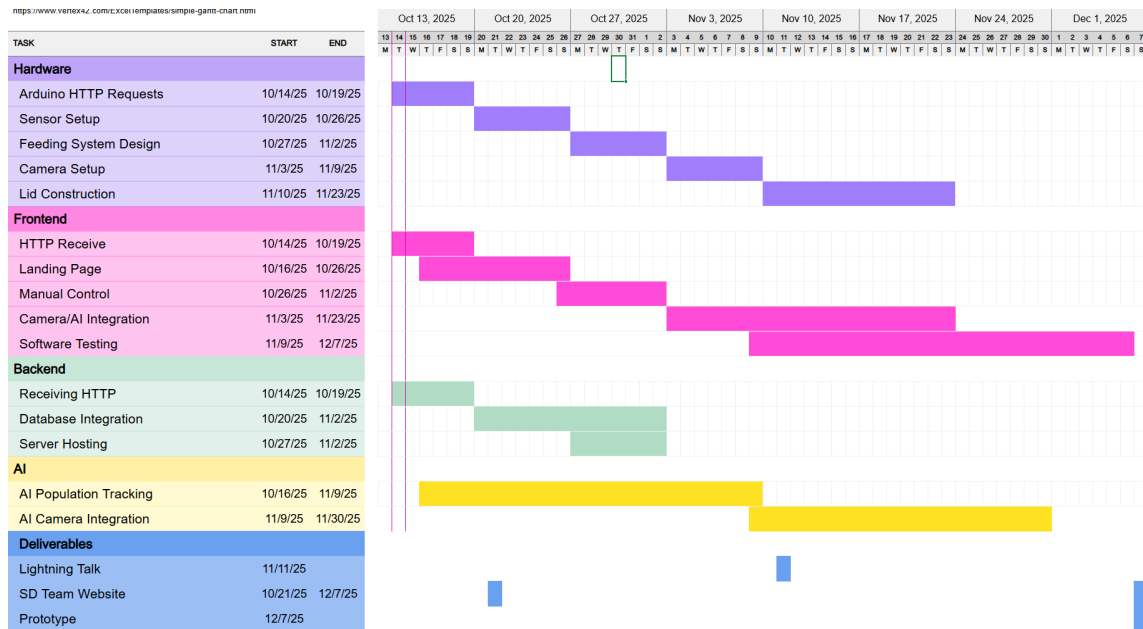


Figure 1: Project Schedule

Prototyping and testing will be ongoing throughout the project. Our project management style is agile, so this schedule will be updated as necessary to address the project's requirements.

3.5 RISKS AND RISK MANAGEMENT/MITIGATION

1. Arduino HTTP requests
 - a. Risks
 - i. Poor signal strength could cause the backend requests to the Arduino to fail ($P = .6$)
 - ii. Arduino may have limited memory, and the pictures taken by the camera might be too large ($P = .6$)
 - b. Mitigation Plan
 - i. Use retries so that it will try multiple times and make sure the WiFi connectivity is strong.
 - ii. We could switch out the Arduino for a more expensive Raspberry Pi that has more memory.
2. Sensor setup
 - a. Risks
 - i. Sensors may need calibration after a specific period of time ($P = .6$)
 - ii. Electrical noise could cause readings not to be accurate ($P = .6$)
 - b. Mitigation Plan
 - i. Implement a warning email that is sent to the user at regular intervals to recalibrate sensors.
 - ii. We could replace the inexpensive sensors we purchased with higher-grade ones.

3. Feeding system design
 - a. Risks
 - i. The servo motor may misalign ($P = .6$)
 - ii. Food could run out without the user noticing. ($P = .8$)
 - b. Mitigation Plan
 - i. Upgrade the servo motor to a higher-grade servo
 - ii. Add a weight sensor, or calculate how many times the fish can be fed with the container of food. Notify user when out.
4. Setup camera
 - a. Risks
 - i. The camera could capture blurry or dark images ($P = .7$)
 - ii. The camera may not be the most compatible with Arduino ($P = .2$)
 - b. Mitigation Plan
 - i. Test with multiple lighting setups
5. Lid construction
 - a. Risks
 - i. Failure to waterproof could lead to electrical damage ($P = .3$)
 - ii. Parts and wires could come loose ($P = .6$)
 - b. Mitigation Plan
 - i. Make 3D printed mounts to secure equipment properly
6. Landing page
 - a. Risks
 - i. UI may not look correct on mobile ($P = .9$)
 - ii. Load time might be longer than 2 seconds ($P = .6$)
 - b. Mitigation Plan
 - i. Use a responsive framework like React
 - ii. Optimize scripts or lessen picture quality
7. HTTP receive
 - a. Risks
 - i. Websocket implementation errors ($P = .2$)
8. Receiving HTTP
 - a. Risks
 - i. Overload if too many requests are being made ($P = .6$)
 - b. Mitigation Plan
 - i. Grab data from a database instead of sending a request to the Arduino. Arduino updates the database every x seconds.
9. Database integration
 - a. Risks
 - i. Slow queries as data piles up ($P = .6$)
 - b. Mitigation Plan
 - i. Use a lighter-weight database like SQLite
10. Server hosting
 - a. Risks
 - i. Server overheating ($P = .1$)

11. AI population Tracking
 - a. Risks
 - i. The model does not perform well due to a small dataset ($P = .7$)
 - ii. False positives like reflections or bubbles ($P = .7$)
 - b. Mitigation Plan
 - i. Use object detection libraries that exist and have been extensively trained.
 - ii. Use background subtraction and average out the fish count over multiple images.
12. Utilize a camera with an AI model
 - a. Risks
 - i. Latency > 2 seconds because of AI response time ($P = .9$)
 - b. Mitigation Plan
 - i. Use a machine learning model instead, which is much faster

3.6 PERSONNEL EFFORT REQUIREMENTS

Task	Explanation	Estimated Hours
Hardware Design & Assembly	Select and connect sensors (pH, turbidity, temperature, DO), actuators (heaters, pumps, feeders), and power supply circuits	60
AI & Computer Vision Development	Develop and train models for fish counting, growth tracking, and predictive analysis of water quality trends	100
Web Dashboard Development	Design and implement a web-based interface for monitoring and alerts	100
System Integration & Testing	Combine hardware, firmware, and software modules, test communication, reliability, and automation features.	50
Calibration & Optimization	Fine-tune sensors, control thresholds, and AI model accuracy under real conditions.	40
Server Management, Application Hosting, and REST API	Keep the server updated and maintain security rules. Configure the server to host backend and frontend applications. Retain the domain and HTTPS certificate.	100

3.7 OTHER RESOURCE REQUIREMENTS

Hardware Components:

- Arduino Uno
- Turbidity Sensor
- Temperature Sensor
- Dissolved Oxygen Sensor
- pH Sensor
- Servo Motor
- Feeding Mechanism
- Power Supplies

Software Components:

- Arduino IDE
- .NET Core Web API (Backend)
- React JS/ Tailwind (Frontend)
- SQLite (Database)

Fish Tank Components:

- Tank
- Air Pump
- Filter
- Water Heater

4 Design

4.1 DESIGN CONTEXT

4.1.1 Broader Context

Area	Description	Examples
------	-------------	----------

Public health, safety, and welfare	The Smart Aquatic Ecosystem ensures stable living conditions for fish and reduces risks associated with improper aquarium care. It additionally helps prevent dangerous water conditions and provides the user with real-time tracking of conditions.	Reduces the risk of fish not being fed due to missed feeding times. Enables peace of mind when away from home by issuing alerts and automatically feeding fish. Prevents unsafe water conditions that could harm fish.
Global, cultural, and social	The system respects cultural practices and ethical standards, aligning with engineering standards to ensure a quality product that can be operated by any user, regardless of cultural or social background.	Aligns with international standards of engineering, AI, user experience, and animal welfare. Enables accessible aquarium maintenance for users of all backgrounds by providing intuitive controls and visual indicators.
Environmental	The Smart Aquatic Ecosystem improves environmental stability by monitoring water quality and preventing harmful conditions within aquatic habitats. Additionally, it reduces waste by automatically feeding, detecting data trends over extended periods, and promoting responsible care practices.	Reducing water waste by identifying when water changes are needed. Preventing chemical imbalances that could harm fish Minimizing excess fish food waste keeps the ecosystem healthier. Supporting sustainable aquarium practices by tracking long-term data.
Economic	The Smart Aquatic Ecosystem offers economic benefits by reducing recurring operational and maintenance costs, minimizing fish loss, and extending the lifespan of aquarium equipment. It improves cost efficiency for users by automating manual tasks and preventing critical failures.	Lowering costs by avoiding fish loss due to unnoticed water quality changes. Reduces unnecessary purchases of chemicals, food, and testing supplies through efficient monitoring systems.

4.1.2 Prior Work/Solutions

The Dangbei Smart Fish Tank 1 Ultra is a product with a similar concept that has not yet been released to the market. It was announced at IFA 2025 in August, but it is currently unavailable for purchase by consumers. Dangbei's announcement for the Smart Fish Tank advertises AI-powered

feeding and water quality monitoring [1].

The main distinction that sets our design apart is our usage of an AI camera vision model. Both our system and the Dangbei system automate feeding and monitor water quality. Only our system uses an AI camera vision model to track the fish population and chart growth trends.

Pros of our design:

- AI-powered feeding schedules
- Automatic water quality monitoring
- An AI camera vision model to monitor fish population and health
- 20-gallon tank, easier for a broader market of consumers to fit in their homes

Pros of Dangbei Smart Fish Tank 1 Ultra:

- Full-Color Spectrum Ecological Light
- Dual-Circulation filtration system

There are no other similar products currently available. Older innovative aquarium systems offer live camera feeds that allow users to view their tank at any time, but none implement AI vision models to monitor the tank automatically [2].

There are other University students in the US looking at similar ideas. An engineering student at Brown University posted an idea to a discussion forum, proposing a “smart camera” to track fish activity and provide insight into aquarium care [3]. However, this person does not appear to be pursuing the idea any further or working on a prototype. Overall, our design would provide a unique introduction to the market, offering new features that are not available anywhere else.

References

Internet Newsgroup

- [1] T. Ricker, “AI fish tank,” The Verge, Aug. 29, 2025.
<https://www.theverge.com/news/767931/ai-fish-tank> (accessed Oct. 28, 2025).

Internet Newsgroup

- [2] “Smart Tank Technology is the Future of the Aquarium Hobby,” Rate My Fish Tank.
<https://www.ratemyfishtank.com/blog/smart-tank-technology-is-the-future-of-the-aquarium-hobby> (accessed Oct. 28, 2025).

General Internet Site

- [3] “Ideas about a ‘smart’ (AI-driven) aquarium monitoring camera,” Reef2Reef, Jul. 07, 2025.
<https://www.reef2reef.com/threads/ideas-about-a-smart-ai-driven-aquarium-monitoring-camera.119015/> (accessed Oct. 28, 2025).

4.1.3 Technical Complexity

1. The AI fish tank is multifaceted, comprising many subsystems that must work together to function correctly. The most complex system within this project is the AI. The AI is responsible for monitoring the fish, their growth, and population. To ensure the AI functions as intended, it requires collaboration between hardware and software. We needed

- to pair the AI with a server that the camera, through the Arduino, sends data to, so the AI can track what we need. In addition to the AI, we have many sensors. We are measuring the temperature, pH, dissolved oxygen, and turbidity of the fish tank. On the hardware side, we have calculated the current draw and devised a method for all sensors to transmit information to a website. This requires an entirely different skill set from the software side. To make this all function, we have a server set up to store the gathered information.
2. Our system utilizes computer vision to continuously monitor the fish, a feature not commonly found in any commercially available product. Our hardware collects data across multiple parameters. Many products offer some data for display, but few allow for the amount of information our system can display. We also maintain a data pipeline that's fed to a remote server and website for users to view. This enables more consistent timestamps, allowing users to make more informed decisions. Most available systems only offer numbers locally or through a simple app. This design provides automated long-term analysis.

4.2 DESIGN EXPLORATION

4.2.1 Design Decisions

1. AI / Computer Vision Design - We have decided to use a YOLOv8n model, which has already been trained on fish. We chose this because it is lightweight and will be able to be run on a server. We also decided to connect a camera to our Arduino, which will take pictures for a specified period and send them to a database.
2. Sensor & Hardware Design - We chose to use an Arduino for our main hardware component. We chose this because it allows us to easily connect numerous sensors and write a code loop that will collect data for a specified period. The Arduino has wifi capabilities, so we could easily send it to a server and insert the data into a database.
3. Backend Design - .NET 9 was chosen for our backend because of team familiarity. This speeds up our development time and allows us to solve problems more efficiently. It also is open source, cross compiles to multiple architectures, and is decently performant.
4. Database Design -

4.2.2 Ideation

We examined various models, as well as different approaches to running them and implementing object detection.

1. YOLOv8n model on cloud (chosen) - Deploy the script onto the cloud so that the object detection is done there. The Arduino just sends the image to the cloud.
2. YOLOv8n model on Arduino - The Arduino isn't that powerful, and constantly using the object detection script, as well as the automatic feeder and sensor reading, would be too much for the Arduino to handle.
3. Smaller YOLOv3 model on Arduino - Even with a smaller model, the Arduino is doing too much work, and the Arduino script wouldn't be as clean, and the whole system wouldn't be very modular.
4. EfficientDet-Lite0 on cloud - Deploy the script on the cloud. Use the EfficientDet-Lite0 model.

5. Train a new machine learning model and deploy it on the cloud. Deploy the script on the cloud. Train our own machine learning model on pictures of our own fish tank.

4.2.3 Decision-Making and Trade-Off

1. YOLOv8n model on cloud (chosen)
 - a. Pros: Clean, Best machine learning model
 - b. Cons: Must deploy on the cloud
2. YOLOv8n model on Arduino
 - a. Pros: One less cloud deployment
 - b. Cons: Too much happening on Arduino, Arduino is not very powerful, and less clean Arduino Script
3. YOLOv3 model on Arduino
 - a. Pros: Arduino can run it, one less cloud deployment
 - b. Cons: Less accurate model, too much happening on Arduino
4. EfficientDet-Lite0 on cloud
 - a. Pros: Clean
 - b. Cons: Must be deployed on the cloud, Slower and less accurate model
5. Train a new machine learning model and deploy it on the cloud
 - a. Pros: Clean, works better in theory
 - b. Cons: Spend lots of time training a new model, probably less accurate, must be deployed on the cloud

4.3 PROPOSED DESIGN

4.3.1 Overview

Technical Design

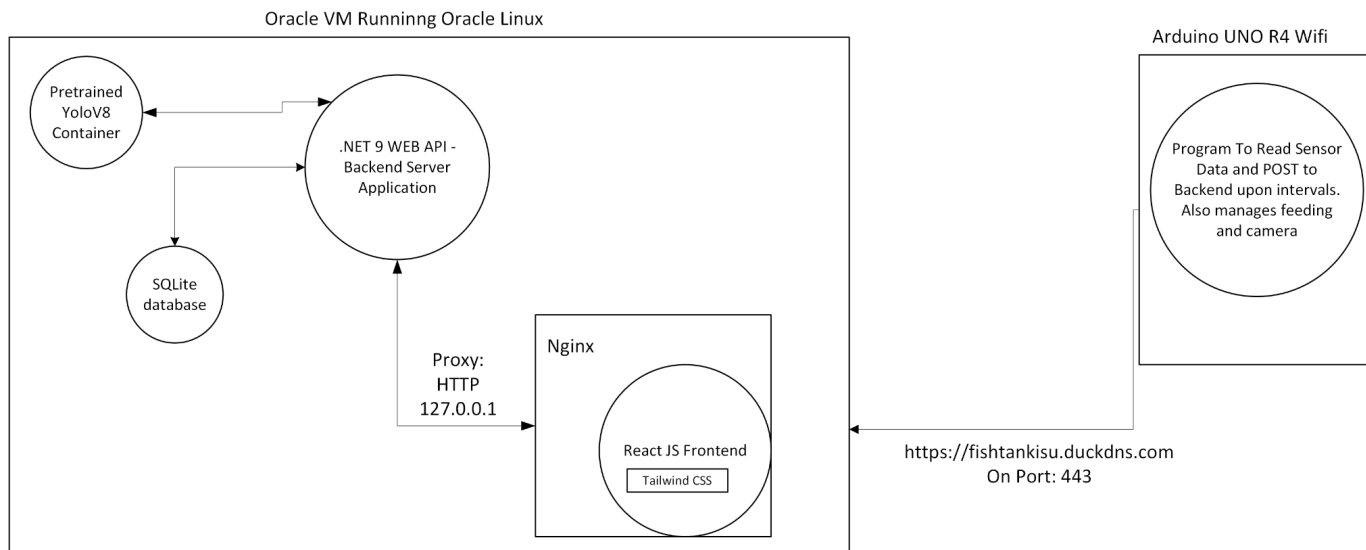


Figure 2: Software Components

Currently, we are hosting a .NET 9 WEB API backend application on an Oracle VM running Oracle Linux. We are using SQLite as our database, as it is a lightweight solution, and we were initially using a much less powerful virtual machine. React with Tailwind CSS is used for the frontend and is also hosted on the VM using Nginx. The pre-trained model will also run on the server in a Docker container. The Arduino communicates with the virtual machine over HTTP, posting images and sensor data to be stored in the database.

Physical Design

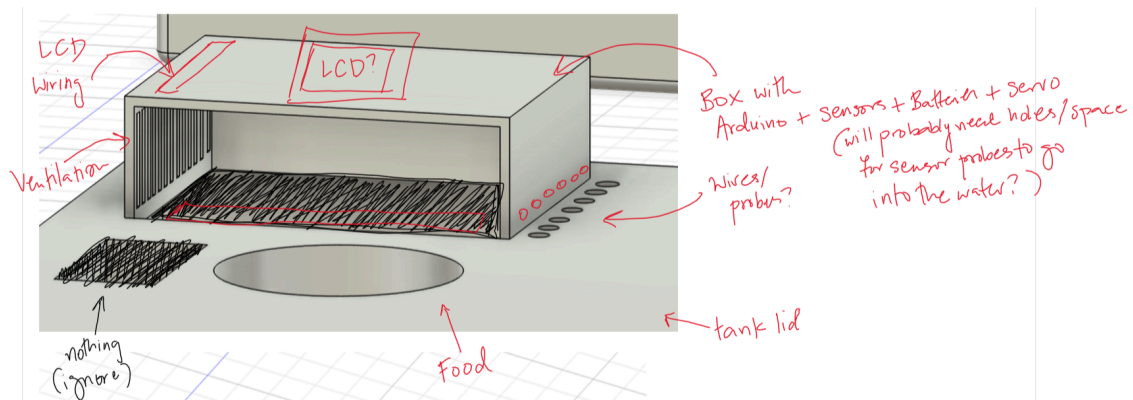


Figure 3: Tank Lid Sketch

We plan to design a lid for the fish tank that integrates with the Arduino and all our sensors. We want the design to be practical, so keeping the form factor as close to a standard fish tank as possible is one of our main focuses in the physical design.

4.3.2 Detailed Design and Visual(s)

Design For Fish Feeder

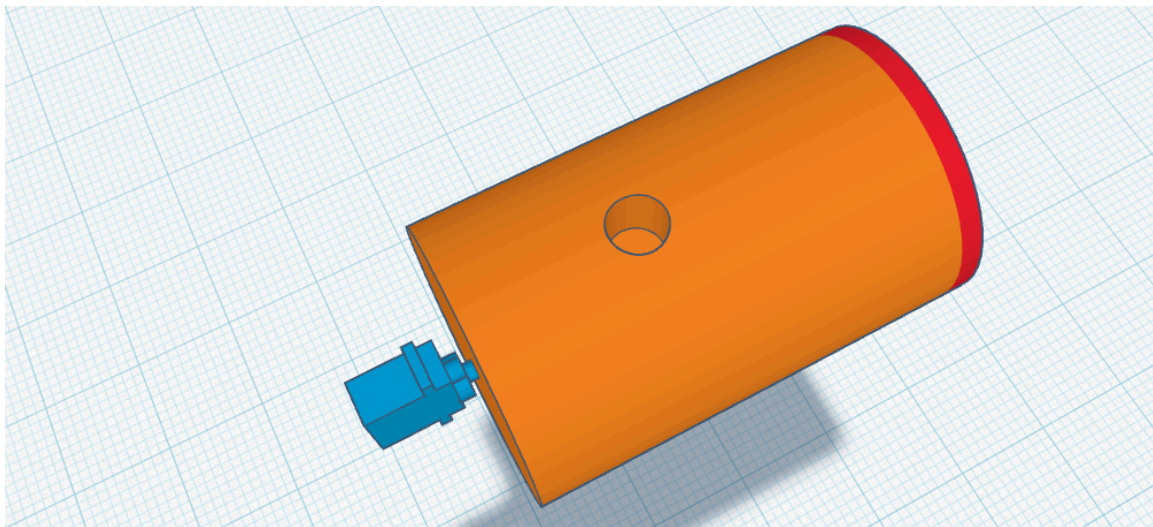


Figure 4: Automatic Feeder Design

The automatic fish feeder is a relatively simple design and will be controlled via a servo motor from the Arduino. In the image above, the blue part represents the servo motor attached to the end of the cylinder. The orange is the hollow cylinder that holds the fish food, and the red is the cap that you will take off to refill the food. The feeder works by spinning the entire cylinder, allowing the food to fall for a short period before clogging the smaller hole due to gravity. You will be able to test the functionality to see how much of your chosen food will dispense, and then adjust the number of rotations you would like to schedule at feeding time.

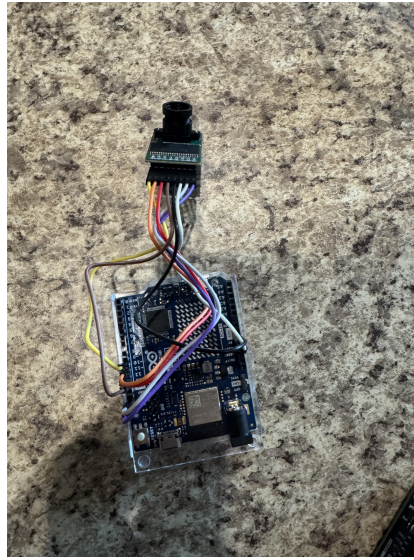


Figure 5: Camera

The camera will be placed against the side of the tank; however, we are still experimenting with different lenses to achieve the broadest possible field of view while still capturing usable images. The Arduino handles taking the pictures and streaming the data to the backend for storage in the database. The photos are stored in base64 strings for easy storage and use with the frontend. This is subject to change based on how we ingest the images into the AI.

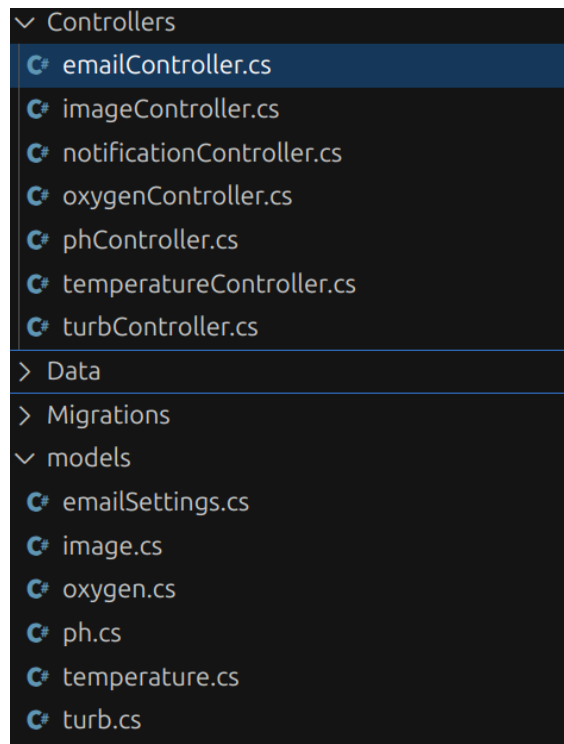


Figure 6: Backend Code Structure

Our backend is organized with models and controllers. Each model, when defined, can have its own table in our database. If any changes are made to the model in the future, auto-generated migrations can be applied to the SQLite table to update it accordingly. For database and API request handling, we utilize all first-party Microsoft packages. For email notifications, the MailKit package is planned to be used.

We are using environment variables to store credentials on the server where needed; these are automatically grabbed by .NET when appropriately named and used in the server program where required.

Arduino Circuit:

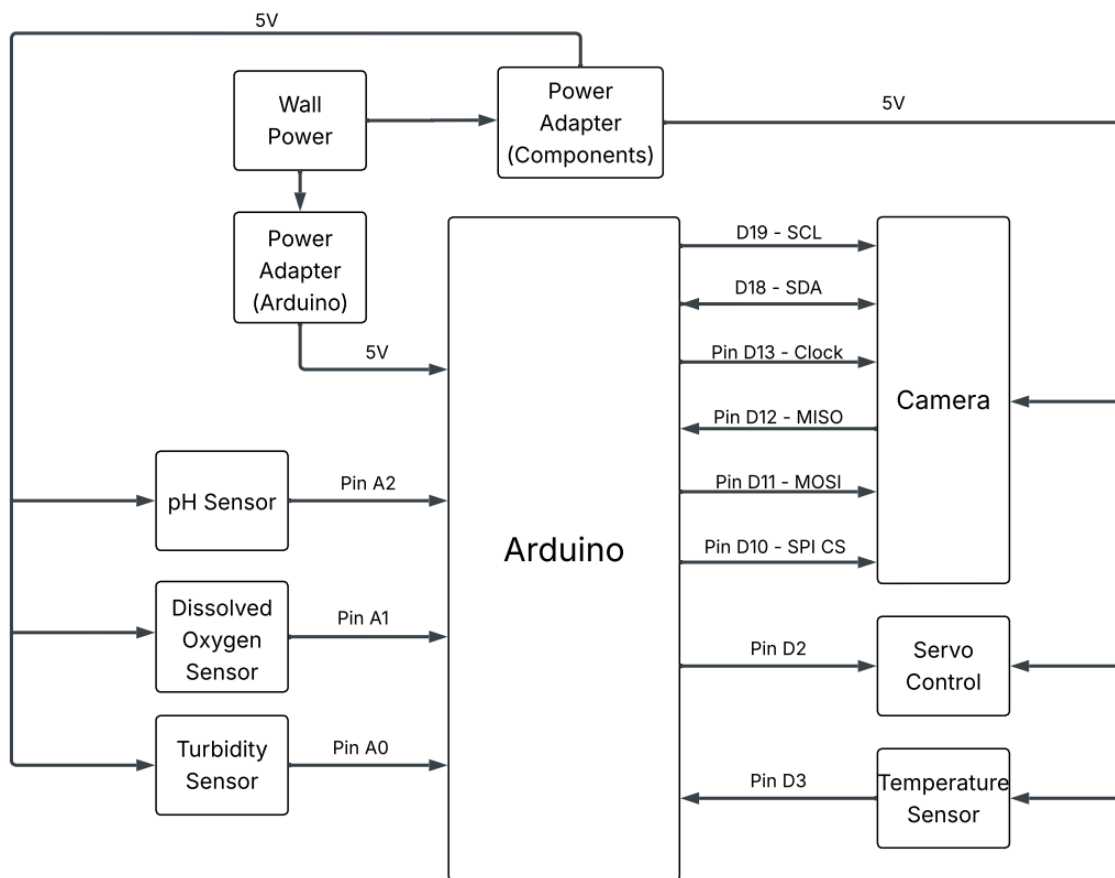


Figure 7: Arduino Circuit

The Arduino UNO R4 WiFi functions as the central controller for the entire system. All sensors interface directly with the Arduino through either digital or analog pins, allowing it to collect water-quality data, process inputs, and control output devices such as the feeder and camera. To ensure stable and reliable operation, the sensors, camera module, and servo motor are powered by a separate dedicated power adapter rather than drawing power from the Arduino itself. This design prevents current limitations that could interfere with sensor readings or communication. The Arduino is powered independently through its own barrel-jack supply.

4.3.3 Functionality

Our design is intended to assist aquarium owners by automating routine tank maintenance and providing real-time visibility into the tank's health. The system integrates several sensors (temperature, turbidity, dissolved oxygen, and pH) connected to an Arduino for environmental monitoring. As the aquarium operates in a home setting, users primarily interact with it through a web-based dashboard. This dashboard enables users to view current sensor readings, track trends over time, and receive notifications when conditions exceed a recommended range.

In its intended final form, the system will not only measure water conditions but also automatically react to them. For example, if the water temperature falls below a defined safe range, the controller will activate a heater to restore it to normal levels. Similarly, an automated feeding mechanism will dispense food according to a user-defined schedule. The goal is to minimize the user's manual involvement while ensuring stable and healthy conditions for the fish.

A typical user experience might look like this:

1. The user installs the aquarium system and connects it to their home network.
2. The aquarium continuously reads water parameters through its sensors.
3. The user checks the dashboard from a phone or computer to view conditions.
4. If any parameter is abnormal, the system sends an alert and (when applicable) automatically adjusts the environment.
5. The user can override settings or change schedules directly from the dashboard.

This functionality forms the basis of a smart aquarium that both informs the user and takes corrective action to maintain a healthy aquatic environment.

4.3.4 Areas of Concern and Development

The current design meets early requirements for data collection and user visibility, as our sensors are being calibrated and we have established successful communication between the Arduino and the backend. The dashboard displays incoming data, and we are currently integrating WebSocket-based notifications. These features begin to satisfy user needs for real-time monitoring and transparency.

However, the system is not yet capable of taking autonomous corrective actions. Our primary challenge moving forward is transitioning from data reporting to complete environmental control, while ensuring safety and reliability when actuators, such as heaters or pumps, operate without direct human intervention.

Our primary concerns moving forward include:

- Completing hardware calibration and validating stable sensor readings.
- Prototyping the automated feeding mechanism and enabling microcontroller-driven actuation.
- Clarifying the level of autonomy expected

Questions for clients, TAs, and faculty advisers include:

- How autonomous is the system expected to be? Should it fully self-correct environmental conditions, or only notify the user to take action?
- Are there recommended safety guidelines or constraints for automated feeding or temperature-regulation systems?

4.4 TECHNOLOGY CONSIDERATIONS

Technology/ Component	Purpose	Strengths	Weaknesses	Trade-off/ Alternative
Arduino UNO R4 WiFi	Central microcontroller for sensor reading, control logic, and communication	-Widely supported, easy to program, large community -Built-in WiFi -Multiple analog and digital I/O pins	-Limited processing power -Highly sensitive to current.	An alternative would be a Raspberry Pi for heavier data processing, but it comes with higher costs and complexity, and it only has digital pins.
pH Sensor	Measures water acidity/alkalinity	-Direct, real-time measurement -Analog output is easily read and interpreted by the Arduino	-Requires regular calibration -Fragile	There are digital pH probes with internal compensation for more stable readings, but they would be harder to connect with the Arduino and cost more.
Dissolved Oxygen Sensor	Measures oxygen saturation in water	-Essential for fish health monitoring -Provides real-time readings	-Requires an electrolyte solution in the probe to complete the circuit. -Fragile	There are optical dissolved oxygen sensors that are more accurate but far more expensive
Turbidity Sensor	Monitors water clarity	-Low cost and easy to integrate with Arduino	-Affected by ambient light -Little information available from the manufacturer.	There are optical backscatter probes that are more accurate but more costly and difficult to calibrate.
Temperature Sensor	Measures water temperature	-Accurate, digital output reduces noise -Can use multiple sensors on a single bus	-Slower response in rapidly changing temperatures.	Thermistors can offer a faster response but require more circuitry and are not as safe in water.

SG90 Servo Motor	Rotates a container to drop food into the tank	-Simple design -Easy for the user to replace food when needed	-Can't support very much horizontal weight -Draws a lot of current on startup	Stronger servo motors are available to support the weight of the food container; upgrading to one of these is a potential solution.
ArduCAM-Mini-5MP-Plus-OV5642 Camera Module	Takes pictures of the tank	-Compatible with Arduino -Supports still image capture	-The ArduCAM library needs to be updated to support the Arduino R4	Several alternatives exist, but are less compatible with Arduino, if at all.
.NET 9 WEB API	Handles Backend HTTP Requests / Database Management	-Familiarity -Ease of Use -Cross Platform -Open Source -Library Support	-Less performant than something like C or C++	An alternative would be Java, but our backend workers are more familiar with C# and .NET Core
SQLite	A database to store sensor data and images gathered from the Arduino	-Very performant -Great Support in .NET 9 -Easy to set up	-Requires Mitigations to refactor tables -Struggles with high concurrency	An alternative would be MySQL, but since we currently lack sufficient computing power with our VM, SQLite is the best choice, as it has all the features we need. It would need to be reconsidered in light of massive amounts of traffic.
React JS	JavaScript library for building our user interface	-Component-based design is modular and allows for easy scaling of our interface.	-View rendering can lead to performance issues if the code is not optimized	We could switch to Solid.js if we end up needing better performance, but for now, React fits our needs better

Tailwind CSS	CSS framework for designing the user interface	-Very customizable -Live changes to the user interface make styling faster	-Inline styling can create cluttered HTML files.	Bootstrap is a popular alternative that offers easy-to-use design templates, but we prefer the more detailed customization options of Tailwind.
--------------	--	---	--	---

4.5 DESIGN ANALYSIS

Backend:

The backend consists of a .NET 9 Web API application with models and controllers for each required object. This is paired with a SQLite database, which was chosen for its low resource usage. .NET 9 has also allowed us to accomplish everything an API needs to do. We are currently using Xunit.NET for unit testing, and it is working great to ensure future updates do not cause outages or downtime due to unnoticed errors. Additionally, we have utilized the GitLab CI/CD pipeline to automatically test cases upon pushing updates to the main branch. This has proven extremely helpful in ensuring that no issues reach the live build.

Hardware/ Electrical:

The hardware system successfully integrates sensing, imaging, and feeding to support automated aquarium monitoring. Water quality sensors take reliable measurements and pass them through the Arduino to the server. The camera captures images and sends them to the server, along with sensor data, to be used by the AI detection model. The feeding system uses a controlled servo motor to dispense food at programmed intervals.

5 Testing

5.1 UNIT TESTING

Backend:

The models are the objects that contain the sensor data to coordinate with the database. However, there are exceptions, such as Data Transfer Objects (DTO), which are used to combine post requests from the Arduino into a single request instead of posting each sensor reading individually. Every model is being unit tested with the following testing properties.

- That constructors/setters are functioning correctly.
- Those data types match the expected behavior.
- That each model can be instantiated safely.
- That the properties hold correct values after assignment.

The XUNIT library is used to create unit tests that will automatically run when changes are pushed to GitLab. A coverlet collector is used to collect code coverage information, and an open-source report generator tool is used to print this information into the CICD console. The most recent report can also be found stored on the server.

Frontend:

The frontend utilizes two libraries for unit testing:

- React Testing Library
- Vitest

The React testing library is essential for testing individual components on the dashboard, ensuring each is in the correct position and contains the proper attributes and labels. The Vitest library is used to create mock server calls that simulate data, allowing us to test whether all frontend helper functions are working correctly, whether information on the dashboard is displayed accurately, and whether any functions manipulating data display the correct result of the formulas used. The unit tests on the frontend are incorporated into the GitLab CICD pipeline, meaning that on every push to the main branch, all tests are run on the frontend to ensure a viable product. In the event of a test failure, the build fails and halts before launching on the live server.

Hardware:

Most hardware testing cannot be automated, so running the code on the Arduino and testing manually over long periods is often the best, or only practical, way to test. We have chemical solutions to calibrate the sensors. Once calibrated, we must monitor the results to ensure they are reasonable and that the sensors continue to function properly over an extended period.

5.2 INTERFACE TESTING

Our design consists of three main components that must work together: hardware, backend, and frontend. The hardware interface includes all the sensors and components attached to the fish tank. The backend software comprises the API methods and the database, while the frontend software is the user interface.

So far, we have tested each interface individually through unit testing and manually verified the data transfer between the three components. We have extensively tested the communication between the backend and frontend interfaces in two ways. One, by loading the most recent data from the database and displaying it to the user interface, ensuring that no data is lost in the process. The second

5.3 INTEGRATION TESTING

Our project has many individual portions. The most critical integration path is transferring data from the sensors to the backend. Still important but not as critical is displaying the data that was sourced to the front end. Our objective is to present data from the sensors clearly and concisely for the users. If we cannot get the sensors to communicate with our server, our product will not function. To test this system and ensure it works, we have wired all our sensors, submerged them in water for a prolonged period, and verified that the data is transmitted at the specified frequency. The data must also be accurate, as any misstep in the data could lead to poor decisions, such as overheating the fish tank. This would violate our requirement not to harm the fish. Additionally, to ensure the data received by the server is accurate, we have verified it against a screen displaying our raw sensor data values.

Backend Specific Integration Testing

Our backend controllers are being tested with a mock database to ensure that they handle information correctly and that the database responds as expected. This is also done with XUnit and is run each time a change is made to the main branch of our GitLab repository.

5.4 SYSTEM TESTING

Our system-level testing is straightforward to verify at a high level. We will have all sensors and the camera wired up and in the fishtank. Our front-end will then display all the compiled data along with a recent photo of the fish tank. The unit test should include the calibration of each sensor. Send functionality by the Arduino. Unit test for each backend component and front-end. Interface test for the frontend to ensure good functionality and a positive user experience. Integration should utilize all elements to create a cohesive system.

5.5 REGRESSION TESTING

The primary way we ensure new additions do not break existing functionality is through versioning. We have set up our GitLab with CI/CD pipelines. These pipelines include both backend unit tests for our API and frontend unit tests. This ensures that new code will not break old code.

The primary way we ensure the additions do not break existing functionality is by manually testing the project's previous functionality. We test the frontend acting as a user, the backend with Postman, and the camera detection by counting the fish.

5.6 ACCEPTANCE TESTING

We will demonstrate that the design requirements are being met by having test cases for each of the requirements. For example, we will run a test to take a picture of the fish tank and ensure the fish are detected. We will ensure that the feeding system provides the fish with a regular supply of food. We will ensure that the frontend buttons function as intended. Etc.

We will involve our client in the acceptance testing by walking them through some of the design requirements, such as checking the fish population against what the app indicates, verifying the accuracy of the water quality, and ensuring they receive alerts if the water quality is incorrect.

5.7 SECURITY TESTING

While we are now focused on functionality, security is something that we are most definitely considering. POST requests will be limited to the Arduino with a private API key, although the API key check has not been implemented. A test will be added to ensure that requests with the incorrect string are rejected and those with the correct string are accepted.

We are still working out the details of how we will implement user authentication and what that entails. However, once that is added, it will create a much greater need for security testing and for how we will handle authentication. This will also require isolation of information to prevent users from accessing each other's data.

There are also testing suites that we would like to set up on the frontend to check for XSS attacks and other data manipulation attacks. As the frontend gets more fleshed out, we will have more opportunities to implement this kind of testing.

5.8 USER TESTING

The team plans to involve real users by having them interact with the system via the web-based dashboard. Users will be tested on their ability to navigate through the interface and perform tasks such as monitoring the water quality and receiving alerts for abnormal conditions. Feedback will be collected through surveys, direct observation, and interviews to evaluate usability, clarity, and overall user experience. This input will help the team identify areas for improvement and ensure that the system meets user needs effectively and intuitively.

5.9 RESULTS

Frontend:

Currently, the frontend is at 84% coverage. This number should be in the 90s or above. However, we have yet to develop the tests to cover our WebSocket functionality. Despite this, that number should not be criticized; it is an important milestone that we have reached as a team. Most of the remaining frontend functionality has been thoroughly tested and reported on, yielding excellent results. Below are the results from a recent coverage test on the frontend. These results show near full completion of testing on several utility functions and the home dashboard page.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	77.22	49.43	65.21	84.44	
src	80	55.35	87.5	86.66	
App.css	0	0	0	0	
home.tsx	80	55.35	87.5	86.66	39-40,42,44,70-71
src/components	66.66	23.8	60	64.28	
LineChart.tsx	66.66	23.8	60	64.28	66-86
src/hooks	62.5	0	28.57	90.9	
websocket.tsx	62.5	0	28.57	90.9	12
src/utils	90	80	100	90	
fetch.ts	89.47	75	100	89.47	4,17
testUtils.tsx	100	100	100	100	
PASS Waiting for file changes...					

Figure 8: Frontend Test Coverage

Backend:

To the right are the results of the current backend coverage tests. Currently, it has 99.2% coverage, which is impressive; however, this level of coverage must be maintained as the application continues to grow. Implementing tests as you add more functionality is imperative, as failing to test while creating technical debt can lead to an exponential increase.

```
48 Assemblies: 1
49 Classes: 15
50 Files: 15
51 Line coverage: 99.2%
52 Covered lines: 277
53 Uncovered lines: 2
54 Coverable lines: 279
55 Total lines: 571
56 Branch coverage: 66.6% (12 of 18)
57 Covered branches: 12
58 Total branches: 18
59 Method coverage: 100% (60 of 60)
60 Full method coverage: 96.6% (58 of 60)
61 Covered methods: 60
62 Fully covered methods: 58
63 Total methods: 60
64 fishTankAPI 99.2%
65 FISHTANKAPI.Controllers.feedRequiredController 96.6%
66 FISHTANKAPI.Controllers.imageController 96.6%
67 FISHTANKAPI.Controllers.oxygenController 100%
68 FISHTANKAPI.Controllers.phController 100%
69 FISHTANKAPI.Controllers.postController 100%
70 FISHTANKAPI.Controllers.TemperatureController 100%
71 FISHTANKAPI.Controllers.TurbController 100%
72 FISHTANKAPI.DTO.postDTO 100%
73 FISHTANKAPI.models.emailSettings 100%
74 FISHTANKAPI.models.feedRequired 100%
75 FISHTANKAPI.models.Image 100%
76 FISHTANKAPI.models.Oxygen 100%
77 FISHTANKAPI.models.PH 100%
78 FISHTANKAPI.models.Temperature 100%
79 FISHTANKAPI.models.Turb 100%
```

Figure 9: Backend Test Coverage

Hardware -

Below is a screenshot of the Arduino serial monitor during an extended test of the calibrated sensors in a cup of water. The relevant parameters listed are pH, dissolved oxygen content, temperature, and turbidity level in the water. All values were within the expected range.

```
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 717 | TurbV: 3.50 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 716 | TurbV: 3.50 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 714 | TurbV: 3.49 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 712 | TurbV: 3.48 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 712 | TurbV: 3.48 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 716 | TurbV: 3.50 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 710 | TurbV: 3.47 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 713 | TurbV: 3.48 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 711 | TurbV: 3.48 | Turb: LOW
DO (mg/L) : 8.83 | pH: 7.00 | TempC: 21.12 | TempF: 70.03 | TurbADC: 715 | TurbV: 3.49 | Turb: LOW
```

Figure 10: Arduino Sample Test Results

6 Implementation

Describe any (preliminary) implementations of your design thus far. Support any general, descriptive text with relevant images. If your project involves inseparable activities between design and implementation, you can list them in either the Design section or this section.

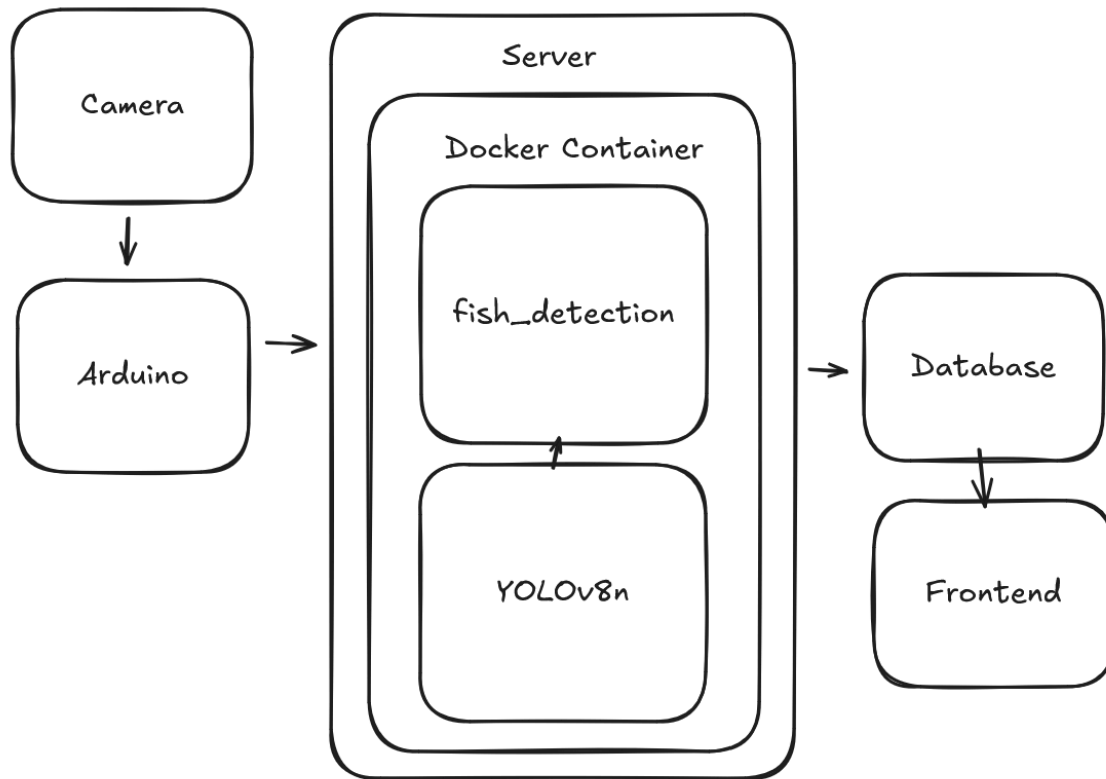


Figure 11: AI Camera Detection

A pipeline was created using TensorFlow and YOLOv8n to detect fish in the tank and display them on the frontend. The current implementation is:

- A camera that captures an image of the fish tank
- An Arduino that takes the image from the camera and sends it to an API endpoint
- A server that runs a Docker container with the TensorFlow scripts and the YOLOv8n model
- A database that is used to store the image and the fish count
- An area in the frontend that displays the image with the detected fish and a fish count

Circuit:

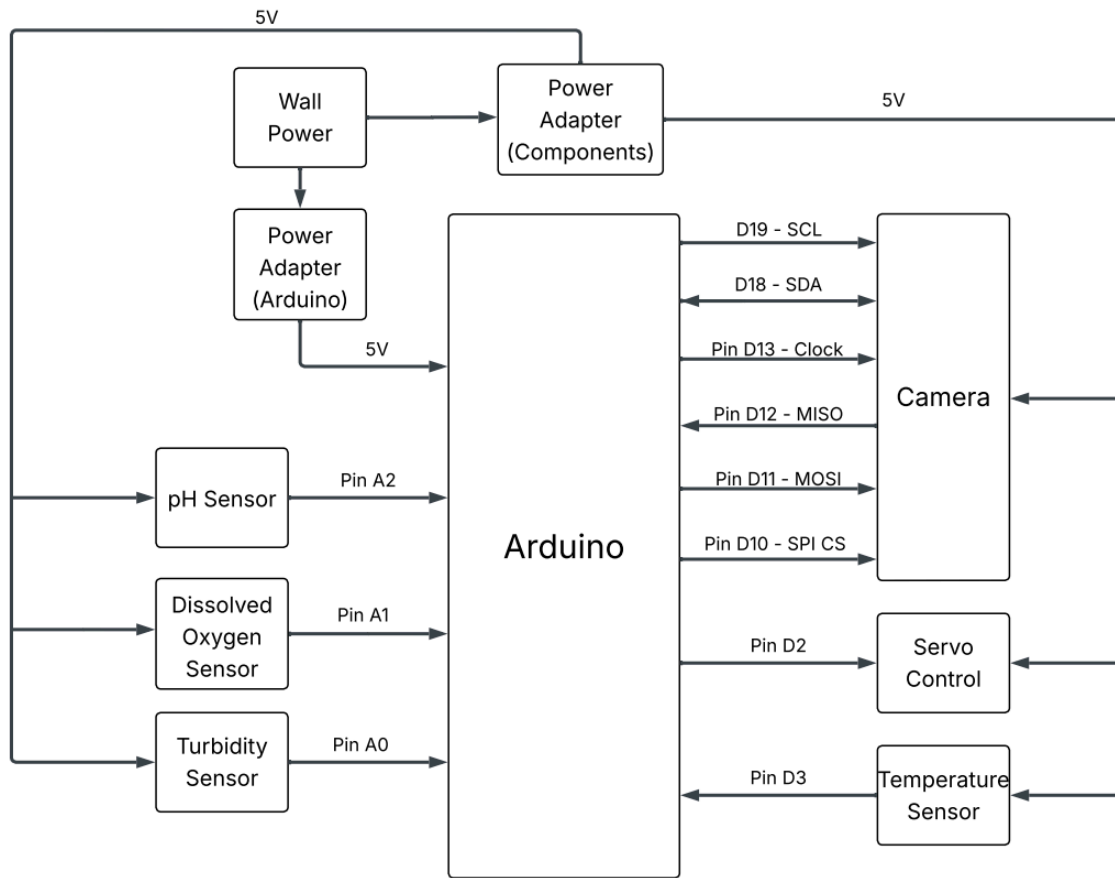


Figure 12: Arduino Circuit

The Arduino UNO R4 WiFi serves as the central controller for the system. It collects data from the various sensors, interfaces with the camera module, sends processed information to the server, and controls the automated feeding system. The current implementation includes the following:

- A dedicated power adapter that supplies the Arduino through its barrel jack input.
- A power adapter to power all of the sensors, camera, and servo motor for feeding.
- The pH sensor, dissolved oxygen sensor, and turbidity sensor are all connected to the Arduino's analog input pins. Each provides a measurable analog voltage proportional to its respective water quality parameter.
- The camera module, temperature sensor, and servo motor control line are connected to the Arduino's digital pins. SPI and I2C communication are used for the camera, while simple digital I/O is used for the servo and temperature sensor.

The fully assembled circuit is pictured below:

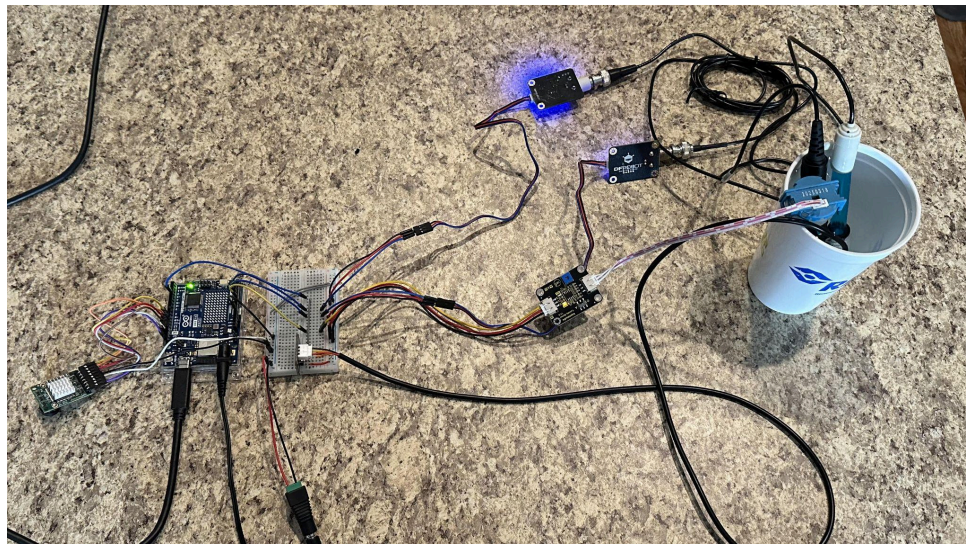


Figure 13: Assembled Arduino Circuit



Figure 14: Feeding System

The feeding system uses a servo motor to rotate a cylindrical container that holds the fish food. The container includes a hole on its top surface. During feeding, the servo rotates the container 180 degrees, positioning the hole downward and allowing food to dispense into the tank. After dispensing, the servo returns the container to its original position to prevent continuous feeding. The feeding schedule is controlled in software. A programmable delay or timer allows the user to specify the frequency at which the fish are fed.

7 Ethics and Professional Responsibility

7.1 AREAS OF PROFESSIONAL RESPONSIBILITY/CODES OF ETHICS

This discussion is with respect to the paper by J. McCormack and colleagues 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

Area of Responsibility	Definition	Relevant IEEE Code of Ethics Item	How Our Team Has Engaged With This Responsibility
Safety & Welfare of the Public	Ensuring that the system does not harm users, animals, the environment, or property	1. “To hold paramount the safety, health, and welfare of the public...”	We designed all electronics to be isolated from the tank water. Separate power supplies reduce electrical hazards, and the heater and sensors are tested to prevent unsafe conditions for the fish.
Understanding of Technology & Societal Implications	Promoting Knowledge about technology and how it affects society	2. “To improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies...”	We researched sensor calibration and water quality measurements before integrating them into our system. We documented system functions and how the technology interacts with the tank ecosystem
Avoiding Conflicts of Interest	Acting in ways that prevent bias or personal gain from affecting professional decisions	3. “To avoid real or perceived conflicts of interest whenever possible...”	Team decisions on component selection and design were made collaboratively and transparently, without favoritism or bias.
Honesty and Integrity	Being truthful about limitations, mistakes, and results	5. “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...”	When there are issues with our system and design, we document them openly instead of ignoring them, and corrected errors as they arise.

Competence	We work only within areas where we are qualified, and we continue to improve our skills.	6. “To maintain and improve our technical competence and to undertake technological tasks for others only if qualified...”	We studied datasheets, tutorials, and guides to ensure proper use of the sensors and camera. We consulted online resources when new challenges arose.
Respect for Colleagues	Treating team members fairly, valuing contributions, and maintaining professional communication	7. “To treat all persons fairly and with respect, and to not engage in discrimination...”	We divide tasks respectfully, listen to each other's ideas, and share workload evenly. Meetings are collaborative and constructive.
Responsibility to the Profession	Acting in ways that reflect well on engineering as a discipline and support colleagues in ethical conduct	10. “To support colleagues and co-workers in following this code of ethics...”	We follow design documentation standards, maintain logs of our work, and encourage each other to adhere to ethical practices, such as transparency in testing and reporting results

Our team is performing well in the area of safety and welfare of the public. Every design decision related to sensor placement, power distribution, and waterproofing has been made with safety as the primary consideration. For example, we intentionally separated the power supply for components from the Arduino's logic power to prevent electrical overloading and reduce noise that could lead to incorrect water-quality measurements. We also ensured that no part of the circuit, except for the sensor probes, is exposed to the water. This attention to detail demonstrates a strong understanding of engineering responsibility and commitment to protecting both users and the fish.

One area where our team could improve is Competence, particularly in documenting lessons learned and design choices. While we have been testing components effectively, our notes and design records are sometimes informal or scattered across multiple platforms. Moving forward, our team should commit to maintaining centralized, clear, and consistent documentation, such as shared wiring diagrams, sensor calibration logs, and test results. Establishing a standard format for documentation and updating it regularly could help ensure that our final product meets professional engineering standards.

7.2 FOUR PRINCIPLES

Broader Context	Beneficence	Nonmalificence	Respect for Autonomy	Justice
Public health, safety, and welfare	Protects fish by monitoring their environment and sending alerts under unsafe conditions	Prevents harm by alerting when water conditions are dangerous	Allows users to control the feeding schedule and water parameters according to their preferences	Ensures all fish in the tank receive equal monitoring and care
Global, cultural, and social	Promotes responsible pet ownership by encouraging careful monitoring of fish health	Minimal negative social impact, no cultural norms are harmed	Allows users from different backgrounds to customize settings for their tank environment	Provides equal access to information about fish welfare for all users, regardless of experience
Environmental	Conserves water by reducing the need for frequent water changes	Uses electronic components that consume energy and resources	Users can choose when to run water quality sensors or heaters, controlling environmental impact	Fairly distributes ecological costs. No particular group bears an excessive burden from energy use.
Economic	Lowers costs associated with fish tank maintenance (e.g., time, resources)	Potential minor cost from electricity usage and component wear.	Users can decide how often to feed the fish or run automated systems, thereby controlling their expenditure.	Makes high-quality monitoring affordable to a broader range of users, promoting equity and accessibility.

Important Broader Context-Principle Pair: Public health safety and welfare - Beneficence

One of the most critical aspects of our project is promoting the welfare of the fish by continuously monitoring water quality and alerting the user when unsafe conditions occur. This directly benefits the animals by preventing stress, illness, or death. To ensure this, we incorporate redundant sensing, regular calibration procedures, and clear alert mechanisms so that unsafe conditions are reliably detected and communicated to the user.

Lacking Broader Context-Principle Pair: Environmental - Nonmaleficence

One area in which our project is currently lacking is the intersection of economic context and the principle of justice. Because our system requires multiple sensors, a microcontroller, and a dedicated power setup, the total cost may limit accessibility for hobbyists or small-scale aquarium owners who would still benefit from improved monitoring and fish-care automation. This raises a fairness concern; the advantages of safer, more stable tank environments may only be accessible to users who can afford higher-end equipment. To address this issue, the team could work on developing a lower-cost version of the design, such as reducing the number of required sensors or utilizing more affordable alternatives.

7.3 VIRTUES

Three virtues important to our team include:

Respect: Being respectful towards one another builds trust and establishes a sense of community, fostering a welcoming environment for innovative ideas.

Communication: Clear communication is paramount, as it allows for timely progress and minimizes opportunities for misunderstandings.

Integrity: Integrity ensures that everyone is accountable for their contributions, maintains transparency in decision-making, and upholds honesty in both successes and setbacks.

As a team, we support these virtues by fostering an inclusive environment where all voices are heard, holding regular check-ins to ensure alignment, and documenting our work openly so that responsibilities and outcomes are clear to all members.

Individual Reflections of Virtues:

Ryan Holden

One virtue I displayed this semester was persistence. Persistence is important to me because you need it to push through any roadblocks you face on the way to your goal. I have shown persistence this semester by making progress on the project while dealing with responsibilities for other classes and in my personal life.

A virtue that I have not shown well this semester is communication. Communication is the most important aspect of a team project like this, because we have seven people who need to work together to create one cohesive design. Moving into next semester, I want to improve my communication habits by talking to every person on the team each week instead of only communicating with the few people that I am directly working with.

Carson Irving

One virtue I displayed throughout the semester was technical initiative. I addressed multiple parts of the project head-on and implemented key functionality to keep the project moving. I have learned and acted on multiple parts of the project, from the frontend functionality, backend design, server management, database management, web hosting, and programming the Arduino. I find it very enjoyable to be able to work on all parts of the project and learn a little bit of everything. It helps me be able to suggest better solutions to problems that I come across in the project, as I can have the perspective of the entire project.

One virtue that I have not demonstrated that I should have is better communication and project management skills. While I have been very proactive, I could have gotten more of the team involved in some choices and maybe created a more general understanding of the functioning of the project. Going forward, I will take a step back and try to get more opinions from the team on some of the problems I come across for an increased perspective and hopefully a better solution.

Aden Koziol

One virtue I displayed throughout the semester is problem-solving. This virtue is important to me because it is very important for engineers. Engineers face problems every day and need to come up with solutions. I demonstrated this by coming into the semester with minimal knowledge of machine learning algorithms and image processing. I researched and planned, then I eventually came up with a solution.

One virtue that I did not demonstrate is long-term planning. This virtue is important to me because I believe that all engineers need this skill. It is important to keep an eye on the long-term goal in order to stay on track. I feel that because I started the semester off with minimal knowledge, I had to take smaller steps and plan for the short term. In the coming semester, I will be sure to plan out long-term goals to keep myself on track.

Marina Mannan

One virtue I have demonstrated in my senior design work thus far is perseverance. This is important to me because complex projects inevitably involve challenges, and perseverance ensures that I continue learning and contributing even when tasks feel overwhelming. I showed perseverance by working with CAD software for the first time, despite the steep learning curve and initial setbacks. Nevertheless, through practice and seeking out tutorials, I was able to create a model that supported our project's goals.

A virtue that is important to me but that I have not demonstrated as strongly is communication. Communication matters because it makes individual contributions visible and minimizes misunderstandings. I recognize that I have not consistently shared updates or asked questions clearly. To improve, I plan to provide regular updates and become more involved in the group overall. This will strengthen collaboration and help integrate my efforts more effectively into the team's success.

Daniel Pender

A virtue that I have demonstrated throughout senior design so far is diligence. Projects often have faults that get missed due to time constraints or ignorance. I have made sure to double check the workings of the project at each step. Making sure that we have not gotten too far ahead with glaring oversight. Doing the little things right and ensuring that we're going step by step and not skipping something. Something as simple as making sure that the wires are connected correctly and that the arduino code is functioning properly. I often worked on calibrating the sensors we could and researching how to get a good baseline/functioning.

A virtue that I did not demonstrate as strongly this semester is time management. Often, multiple projects would stack up around the same time, and I would procrastinate a bit instead of starting tasks early. This sometimes led to unnecessary stress and rushed work. Moving forward, I plan to improve my time management by starting assignments earlier, spacing out my workload more evenly, and staying more consistent with my weekly planning so I can better support both my team and the project overall.

Max Purvis

One virtue that I have demonstrated in my senior design work thus far is patience. Patience is important because engineering projects rarely move at a perfectly steady pace. Maintaining patience helps keep the team atmosphere positive and prevents frustration from slowing down progress. I've demonstrated patience by adapting to delays in component shipments and having to find things to do while I wait. Instead of letting these delays discourage me, I used the time to refine documentation, review designs, research components, and prepare for when parts arrive.

One virtue that I have not demonstrated as strongly is Leadership. Leadership is important because taking initiative and helping guide the team can prevent bottlenecks and improve coordination, especially when multiple tasks overlap. To demonstrate leadership more effectively, I plan to take more initiative during planning discussions and support teammates by offering help or stepping in when progress slows.

Griffin Urban

A virtue I have demonstrated in my work on this project is leadership. Leadership is essential for banding team members together and ensuring that we maintain a positive attitude. I've demonstrated this by ensuring that team members are aligned on the work that needs to be done and by helping define areas of improvement as a group that we can pursue. I strive to ensure that every team member feels heard and has someone to turn to when concerns or questions arise.

A virtue that I have not demonstrated strongly is Perseverance. This semester has been extremely busy for me, so pushing myself to juggle various classes, projects, and clubs has been difficult. I am greatly looking forward to this upcoming spring semester, in which my load will be much lighter than it is currently. I plan to improve my perseverance by allocating more time to fine-tune the project, engaging in more discussions with the team and advisor, and creating long-term plans to support my growth, the team's growth, and the project's growth.

8 Closing Material

8.1 CONCLUSION

Work Completed:

- Basic frontend dashboard displaying the following information
 - Live data from sensors in the backend
 - Graphing of data over a time period
 - Estimated quality of the tank based on data
 - Imaging from the camera inside the tank
 - Admin dashboard for testing data values
- Backend
 - REST API handling information requests from the frontend and data collection from the Arduino
 - Unit testing for all models
 - Integration testing for controllers with Mock DB
- Server Management
 - Nginx Web Server with HTTPS Certificate and Reverse Proxy to securely communicate with the backend
 - Certbot to renew the HTTPS certificate automatically
 - Automatic Subdomain renewal with duckdns (<https://fishtankisu.duckdns.org>)
 - Traffic Security Rules and Firewall Configuration
- Hardware
 - All sensors are set up and sending measurements to the server
 - The camera is able to send images to the server
 - The fish feeding system is able to dispense food, though still in development
 - All circuit components are powered effectively and safely
- AI Population Tracking
 - Model has a 75.21% Accuracy Score
 - Implemented Docker container for ease of use
 - Fish in images are highlighted in green boxes

Goals:

- Hardware
 - Finalize Fish Feeder Design
 - Finish calibrating sensors and make sure measurements are accurate
 - Perform maintenance on the pH sensor probe
 - Implement feeding functionality by communicating with the backend
 - Finalize the lid design
 - Water Pump To Easily Exchange Water
 - Set up and mount circuitry in the lid

- Backend
 - Implement Authentication and Credential Handling
 - Implement Correct Data Isolation For Private Information
 - Implement running the AI container from a backend controller
 - Backend support for the account system and login
 - Integrate security testing, input validation, etc
 - Add logging for irregular events and notifications
 - Finish implementing Email Notifications
 - Integrate Website Notification over Websocket
- Frontend
 - Support mobile interfaces
 - Login Screen and account system
 - Feeding Schedule Controls
 - Settings page
 - History page to display past changes
 - Graphing page to display multiple graphs at the same time
 - Improved control over the time period on graphs

Plan of Action:

- We have recently planned to start meeting more regularly to better communicate where we are with our assigned portion
- We are going to plan further in the future to make sure there is always an attainable goal for each aspect of our project
- With our next semester of funding, we will be able to get fish in the tank and start confronting all of the present challenges

Constraints and Challenges:

- Parts: Several delays and shipping issues occurred when attempting to order parts. This caused us to begin assembly and testing of physical components later than we had anticipated.
- Some sensors are highly sensitive and require delicate maintenance to make sure they are working properly
- 3D printing the lid and component housing required several training sessions, the purchase of filament, and printer availability. These constraints delayed the printing process, and as a result, we were unable to print the lid as scheduled.

8.2 APPENDICES

Arduino Pinout:

Pin	Function
A0	Turbidity Sensor Data
A1	Dissolved Oxygen Sensor Data
A2	pH Sensor Data
A3	Open
A4	Open
A5	Open
D0	Open
D1	Open
D2	Servo Control
D3	Temperature Sensor Data
D4	Open
D5	Open
D6	Open
D7	Open
D8	Open
D9	Open
D10	ArduCAM SPI Chip Select
D11	ArduCAM SPI Master Output Slave Input
D12	ArduCAM SPI Master Input Slave Output
D13	ArduCAM SPI Serial Clock
D18/SDA	ArduCAM Serial Data
D19/SCL	ArduCAM Serial Clock

9 Team

9.1 TEAM MEMBERS

- 1) Carson Irving
- 2) Danny Pender
- 3) Max Purvis
- 4) Griffin Urban
- 5) Aden Koziol
- 6) Ryan Holden
- 7) Marina Mannan

9.2 REQUIRED SKILL SETS FOR YOUR PROJECT

Non-Technical

- General Fish Tank Knowledge
- User Experience

Technical

- SE
 - Frontend Design
 - Backend Design
 - Database Management
 - Machine Learning
- EE
 - Circuit Design
 - Arduino
 - Sensors
 - Data Collection
- CPRE
 - CAD
 - 3D Printing
- CYBE
 - Server Management

9.3 SKILL SETS COVERED BY THE TEAM

Non-Technical

- General Fish Tank Knowledge (Carson)
- User Experience (Marina, Aden, Carson)

Technical

- SE
 - Frontend Design (Aden, Carson, Ryan, Griffin)

- Backend Design (Aden, Carson, Ryan, Griffin)
 - Database Management (Aden, Carson, Ryan, Griffin)
 - Machine Learning (Aden)
- EE
 - Circuit Design (Max, Danny)
 - Arduino (Max, Danny, Carson)
 - Sensors (Max, Danny)
 - Data Collection (Max, Danny, Carson)
- CPRE
 - CAD (Marina)
 - 3D Printing (Marina)
- CYBE
 - Server Management (Carson)

9.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

The team used an Agile approach for project management. It was decided that everyone would be responsible for a part of the project, and everyone communicated to connect the parts. The team would take smaller steps and communicate frequently to make progress.

9.5 INITIAL PROJECT MANAGEMENT ROLES

Griffin Urban — Frontend Developer
 Daniel Pender — Hardware
 Aden Koziol — AI Population Tracking
 Ryan Holden — Backend/Frontend Developer
 Marina Mannan — Embedded Systems Engineer
 Carson Irving — Full-stack Developer/ Server Management
 Max Purvis — Hardware

9.6 Team Contract

Team Members:

- | | |
|------------------|------------------|
| 1) Carson Irving | 2) Danny Pender |
| 3) Max Purvis | 4) Griffin Urban |
| 5) Aden Koziol | 6) Ryan Holden |
| 7) Marina Mannan | |

Team Procedures

1. Weekly meetings will be held at 1:00 pm on Mondays over Discord.
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face): Face to face
3. Decision-making policy (e.g., consensus, majority vote): Majority vote

4. Procedures for record-keeping (i.e., who will keep meeting minutes and how they will be shared/archived): General meeting notes and discussion topics will be taken for future reference. It is essential that ideas are transcribed

Participation Expectations

1. Each member is expected to attend meetings and workshops on time and be prepared to participate.
2. Each member is responsible for completing their share of assignments on time.
3. Each member is expected to be reachable and able to communicate with other members.
4. Each member is expected to contribute and provide input on all team decisions.

Leadership

1. Each member will take the lead on engineering one of the individual components. Team organization and client interaction will have no set leader.
2. All team members will be accountable for supporting one another and providing guidance to others.
3. Team members will track others' progress through Git to ensure they are on track and making contributions to the project.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

Carson - I have a decent amount of prior experience taking care of a large fish tank, and I think it will allow me to better understand how to bridge the engineering efforts with the foundation of taking care of the fish. I also have working experience making applications in .NET which I can utilize to create our backend for our project.

Aden - I have some experience working with prompt engineering and machine learning for AI models. This will be of use when it comes to the camera.

Marina - I have previous experience working with sensors and data collection, which will be useful in monitoring the parameters of the fish tank (pH levels, temperature). Additionally, I am interested in the user experience and how data is conveyed to users in a clear and meaningful manner.

Max - I have experience in circuit design and testing. This will be useful to ensure the functionality of all electronic components.

Danny - I have experience using servos and basic circuit designs. In order to maintain a healthy fish tank, we are going to need to add micro nutrients and food to the tank. There will need to be some moving components that allow our AI and sensors to add the necessary nutrients.

Griffin - I have experience developing front-end software, so my focus for the project will be on user interactions with the system and how data is conveyed to the user.

Ryan - I have experience in both frontend and backend web development that will be useful for designing the user's web app and communicating real-time updates on the state of the aquarium.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

Giving all ideas a chance, and if someone is very passionate about a certain part of the project, letting them work on it will be beneficial to the entire team. We are allowed to make mistakes, as long as we correctly learn from them.

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?)
 - Introduce the issue to the team
 - Provide supporting evidence on why there is an issue and how changing will improve the progress/ overall quality of the project
 - The entire team will take a vote deciding the fate of the issue; the majority vote wins

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

EE:

- Gain a better understanding of the concepts involved in the interaction between software and electronics. Learn to work with team members of different skillsets

SE/CYBE/CPRE:

- Gain a better understanding of what it takes to train a Computer Vision model and the complexity of training a model.
- Learn how to bridge the gap between software and electronics/hardware.

Overall:

- Take great care of the fish!
- Learn to manage and work with a team on a larger scale and a longer-term project.

2. Strategies for planning and assigning individual and team work:

An issue tracking board so we can see the progress of parts of the project:

<https://fishproject-band.plaky.com/spaces/169731/boards/197021>

Git for tracking code progress and seeing contributions (Gitlab provided)

3. Strategies for keeping on task:

On the issue tracker, we have a field for a deadline that indicates when the work is expected to be completed. Obviously, this can be adjusted if it is not met, but it will give a good baseline and prevent procrastination.

Communicating often with the team to make sure everyone is staying on task and isn't stuck on something that someone else could help with

Consequences for Not Adhering to the Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

We will communicate with the team member and explain the issue we are having and what the team expects moving forward.

2. What will your team do if the infractions continue?

The team will contact the professor (s) and explain the issue, discussing possible resolutions.

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) Carson Irving | DATE 9/9/2025 |
| 2) Aden Koziol | DATE 9/9/2025 |
| 3) Max Purvis | DATE 9/9/2025 |
| 4) Griffin Urban | DATE 9/9/2025 |
| 5) Daniel Pender | DATE 9/9/2025 |
| 6) Ryan Holden | DATE 9/9/2025 |
| 7) Marina Mannan | DATE 9/9/2025 |