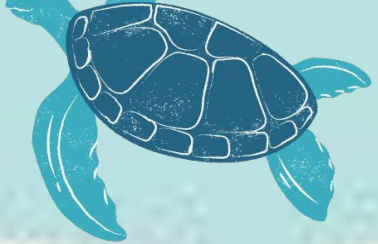
Web Application for Aqualab Sensor Monitoring and Analysis - Milestone 6

Ruth Garcia, Haley Hamilton, Greg Thompson

Milestone 6 Overview:

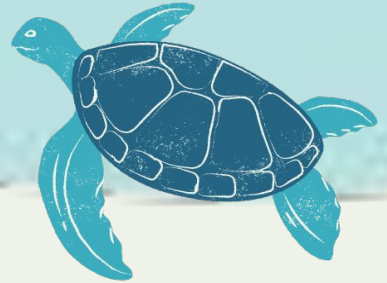


- Implement, test, and demo **final UI additions/styling**
  - Included adding final tweaks to the frontend including clearer ranges, the appearance of charts, CSV file downloading format, and the calculated data relationships.

- Implement, test, and demo **user roles and permissions**
  - Implementing JWT tokens and flask decorator functions to associate a client with a user and restrict/grant access to features based on their user role.

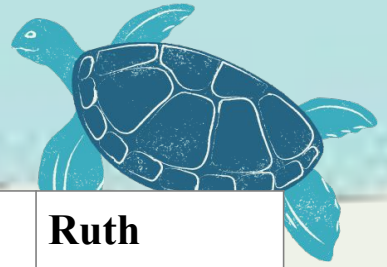
- **Final system integration** and error handling
  - Implemented stop run button, change range and change frequency features, updates after testing with live sensor and testing different “program recovery after shutdown” scenarios

Milestone 6 Overview:



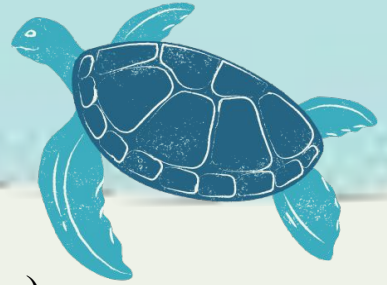
- Implement, test, and demo of **the entire system**
  - Tested system with lab sensor multiple times, everything is functional.
- **Conduct evaluation and analyze results**
  - Tested 10 different features and UI pages cases with 7 lab volunteers and three different roles
- **Create user/developer manual**
- **Create demo video**
- Make app more accessible remotely and Create a User Logging Feature
  - Did not have enough time to complete this, future plans to help lab team/next project iteration are in the works!

Milestone 6 Progress Matrix:



Task	Greg	Haley	Ruth
Implement, test, and demo **final UI additions/styling**	0%	80%	20%
Implement, test, and demo **user roles and permissions**	0%	70%	30%
Final system integration and error handling	60%	40%	0%
Implement, test, and demo **of the entire system**	30%	50%	20%
Conduct evaluation and analyze results	33%	33%	33%
Create user/developer manual	80%	20%	20%
Create demo video	0%	80%	20%
Make app more accessible remotely	-	-	-
Create a User Logging Feature	-	-	-

Final System Integration

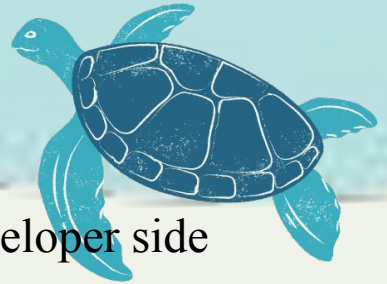


Includes:

- Full implementation of final features (change range, change frequency)
- Implementing a stop run button
- Testing and making changes for “Program recovery after shutdown” scenarios
- Made changes to ensure proper connection to sensors

- Client explained that 1 water sensor would read both dissolved oxygen(DO) and carbon dioxide (CO_2) and both we need to be monitored.
- The needed user interface, backend, and database changes were made to accommodate this

User Manual



- Lengthy document to explain the system from both user side and developer side

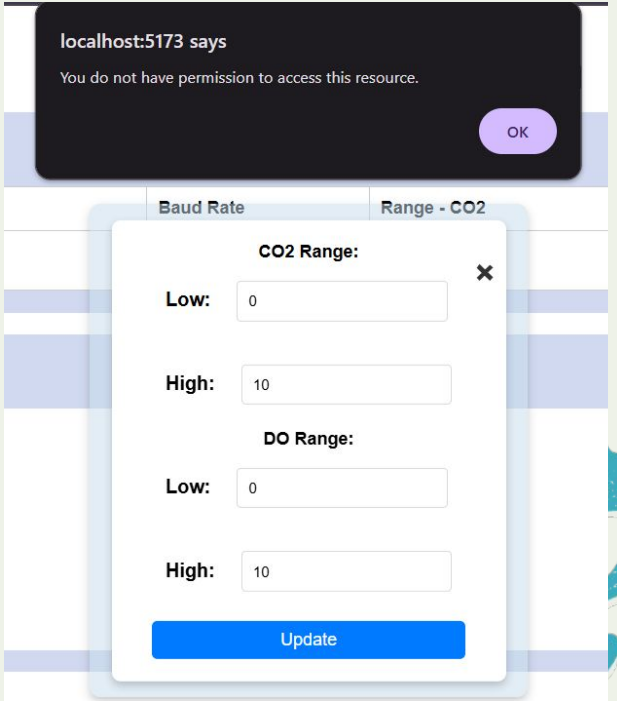
User: instructions for Mongodb and installation
Instructions to configure sensors
Instructions for all the tools and features

Developer: Explanation on each file and function of each system component

User Roles/Permissions

- Implemented JWT token creation at successful login
- Token contains user id and role
- <u>Basic functionality:</u>
  - User tries to complete action
  - Client token sent to the backend with the action request
  - Role is verified can action completed / receives alert

```
# This route updates a high/low range values for a sensor in
@self.app.route("/change_range/<id>", methods=["PATCH"])
@require_role(["admin"])
def change_range(id):
    sensor_id = {"_id": ObjectId(id)}  # Correctly format the
    existing_sensor = sensor_collection.find_one(sensor_id) #
    if not existing_sensor:
        return jsonify({"message": "Sensor not found"}), 404
```

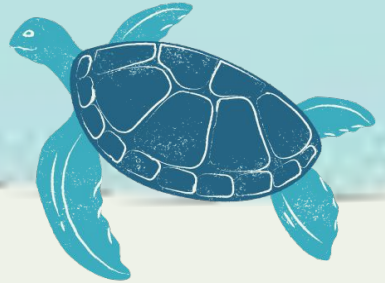


Test/Demo Entire System:

Tested the system hooked up with the sensor:

- Made changes to ensure proper sensor connection
- Let system run/played with sensor water to monitor data
- Tested program recovery scenarios (unplugged sensor, sensor reboot, etc…)

Full System Demo:

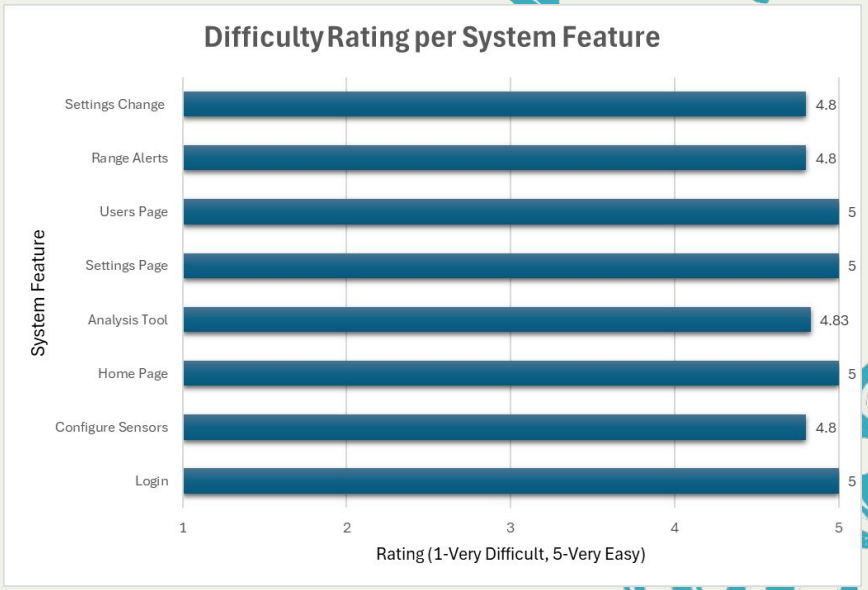


Insert link here

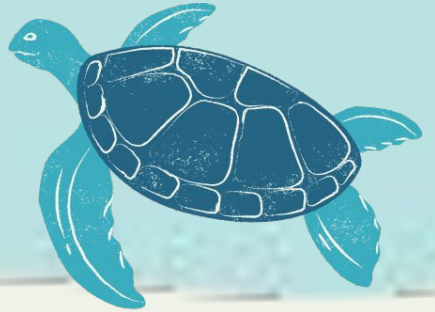
UI and User Acceptance Testing/Evaluation:

- We did not get to user logging
  - Will implemented by a future team

- User Acceptance was completed
  - 2 admin
  - 4 observers
  - 10 different testing scenarios
- Analysis?
  - Overall: Client very satisfied!
  - Our Change Range button should be ‘louder’
  - Easy to navigate and quick to understand UI
  - Alerting system on screen could be more obvious



Lessons Learned?

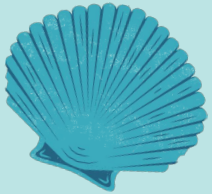


Importance of agile development process:

- Difficult to align schedules
- Easy to lose sight of tasks that were/needed to be completed for the milestone
- Originally used JIRA, but it was difficult to keep updated and became extra work
- Important to remember how helpful weekly scrum meetings and daily stand ups can be.

Importance of planning:

- difficult to plan a complex system with a lot of features upfront
- Would have benefited the project to spend more time in this area
- Ex: main program backend and architecture of the React frontend
- Moments we lost sight of some intended features/functionality and were not implemented in the easiest/scalable/best way



Questions?