WATER MONITORING WEBPAGE WITH FAUCET SIMULATOR FOR HOMEOWNERS AND WATER METER MANUFACTURERS

Kevin Daill, Nathan Huven, James Meaden, and Josiah Sanchez

California State University, Monterey Bay

CST 499, Computer Science Capstone

Dr. Eric Tao

Fall B 2022

EXECUTIVE SUMMARY

The Water Monitoring Webpage With Faucet Simulator provides a tool for use by anyone who has their own hardware. This differs from other monitoring software primarily designed to be used with hardware made exclusively by the same company. The Webpage will serve as one element of the water monitoring system. Users and hardware will communicate with the web page using a web application programming interface (API). The Webpage using an API makes no distinctions between hardware, allowing it to be universal from an implementation point of view. With this webpage, users will be able to view overall usage, set water monitoring alerts, and use the water faucet simulator. The simulator will assist with providing data in absence of hardware.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION/BACKGROUND

The proposed project is a water monitoring webpage with faucet simulator intended for open source use. The project aims to provide a free alternative to existing water monitoring software available. It will allow anyone interested in using the software the chance to do so without a purchase or requiring specific hardware. The end users are anyone with access to the internet through a computer. Users will simply go to the website and explore the different web pages. As each web page is viewed, the page is self explanatory with features such as creating a user account and filling out forms. In addition, a simulator is provided to view the response of the webpage, making hardware optional. If users are curious to design custom hardware to interface with the webpage, they can view the Application Programming Interface (API) web page for instructions on how to interact with the web site.

There are existing water monitoring systems that offer end-to-end solutions and include hardware and software. The purchase of the hardware is accompanied with the supporting software. Water monitoring hardware alone exists through different manufacturers as well. The hardware can be an analog or digital meter, with some meters having a feature of a pulse output. The pulse output can be interfaced with other custom electronics to sense the output and process it. For example, sense the pulse and transmit that data wirelessly to a web application. Water monitoring software alone is limited with some companies offering solutions tailored to a specific end user with a specific hardware in mind. There are fewer software options available with most being proprietary. The proposed project offers users an open source solution without the need for specific hardware.

FEASIBILITY DISCUSSION

ENVIRONMENTAL SCAN

There are companies with products that have software in the form of an app that provide water usage information; however, users are required to buy the accompanying hardware. The solutions they offer cover the three main components that make up a water monitoring system: water meter, network hardware, software application. Companies such as AlertLabs, PowerX, and BlueBot offer water monitoring software, but are used with their custom hardware. Even though these companies offered paid solutions they have demonstrated how effective water monitoring solutions can be. In case studies documented by Alert Labs Inc., they provide evidence as to how their solutions helped Minto Apartments detect water leaks which resulted in repairs and an estimated savings of $100,000 in water related expenses (2022).

Chetu offers developers to build the software made for specific customer requirements. AquaCUE offers software. The Water Monitoring Webpage seeks to provide a solution to the first component, the software application component, and is used as a stand alone product to be paired with a user provided network hardware and water meter.

AlertAQ by Alert Labs, is software that operates through cellular communication, users buy hardware that communicates with software through a cellular network. PowerX Technology and Bluebot are two similar solutions to AlertAQ regarding a hardware centered approach; however, networking is provided through wifi. The difference between these existing solutions and the Water Monitoring Webpage we intend to design is, it will be targeted to those who wish to bring their own hardware. There will be no hardware requirement, only an API requirement if users connect hardware to the webpage.

JUSTIFICATION

Many people are unable to afford software to monitor your household water usage. This software is usually only available with the purchase of expensive hardware who charge a premium for their own custom software solutions. With many of the water authorities in California mandating limited water usage it is becoming important for people of all income levels to have the ability to easily monitor their water usage in real time. By providing people access to open source water monitoring and water usage simulation software it is expected that more people will be able to increase the efficiency of their water usage. For those looking to invest in hardware this also gives them an opportunity to try similar software before they purchase their equipment.

ETHICAL CONSIDERATIONS

The project provides an interface (webpage) that monitors water usage habits of individuals, this raises privacy concerns. The initial capstone project is a proof of concept, and security will not be heavily implemented. After the capstone and if the project lives on, security will have to be properly included in the end-to-end design. If other contributors take over to assist with updating and maintaining the software, it is important to pass on the importance of security and privacy.

Another ethical concern is the impact to the water monitoring companies who provide the software with a hardware purchase. Since the capstone is designed to be open source, this project has the impact of encroaching into their market. External contributors should be made aware they could be competing in an industry with companies that depend on profits. These companies will likely have a number of employees whose livelihood depends on sales.

Underprivileged groups will not be negatively impacted by our capstone because the site offers an affordable solution to simulate water usage opposed to otherwise buyable software. The site should be able to run on almost any computer. Those who are unable to gain access to their own computer should be able to access the site from a browser at their local library.

The project will pose accessibility issues for those with disabilities. People who are blind will be unable to utilize our simulation page unless aided. Also, people who are unable to use their hands will not have a way to interact with our applications.

To eliminate or mitigate ethical concerns including accessibility or cost we have developed our capstone to be completely browser based to allow any user to simulate water-usage with data they choose cost free. Our webpage will aim to be user friendly and straightforward to give any community an opportunity to be better educated on water-usage related data. For accessibility concerns, they will not be addressed in our capstone given the time constraint.

LEGAL CONSIDERATIONS

When it comes to legal considerations, there are multiple items to be weary of. One of the most important legal considerations are existing patents. If other companies have similar software, we have to make sure that all of our methods used are our own. There is also the possibility of breaking contractual agreements and obligations between companies and counties. There is a potential liability concern from clients if our simulation is inaccurate upon release. Our target client could use more water, by following an inaccurate simulation, which would then increase the client’s water bill. Our team is dedicated to preventing this by providing accurate and reliable information to our clients. Our team does not want to violate or break any existing agreements There will be no profit made throughout the development of this project and it will strictly be for educational and research purposes.

LONG TERM

The long term life of the project will be heavily dependent on resources that are available. Currently the project is utilizing trial based platform services such as Heroku to deploy the web page and interact with an SQL database. A funding mechanism would need to be identified in order to continue to cover the costs of running the servers. This could be done via advertisements, licenses issued to water meter manufacturers, or incentive based programs run in partnership with water utilities.

Regarding the software, other enhancements would need to be made to increase the relevance of the web page. Providing push notifications being one example, where a user would not need to log in to get an alert. Other features include the ability to group meters together based on location. This would help property owners with associating usage with a particular area. Hardware is another area for enhancements, currently the team has a custom device used to test the functionality of the web page. Providing more in depth API documentation would be necessary to ensure hardware can interact with the system independent of vendor or manufacturer.

PLATFORM

Several technologies were utilized to create and host the different aspects of the application. GitHub was utilized as a code repository for managing the backend codebase. This allowed the team to collaborate on code while having the flexibility to update local versions of the application. The backend application was hosted on Heroku. The database was also hosted on Heroku utilizing a MySQL database called JawsDB. Heroku was selected due to its ability to host virtual servers using cloud technology. It also offered an affordable price point. This allowed us to have an always-on server without the need to maintain our own hardware. In addition, Heroku offers a deployment pipeline between GitHub and Heroku, which streamlined the time the team had to spend on redundant tasks.

Replit was utilized to both code and host the frontend application. Replit was chosen because the team had previous experience building websites on the platform. It offers collaboration features that allow team members to see in real-time the updates made by others. In addition, it is a free service and one that requires minimal effort to get a new project up and running. This allowed the team to save on both cost and time. The website was designed using Java Script and the REACT framework. REACT was chosen due to its ability to design modern looking websites with relative ease. Components from Material UI were also utilized in order to provide modern looking interfaces.

MAJOR FUNCTIONS

Major functions of this application are designed and meant to track water usage. Providing that data to the user for them to continue to make conscious decisions concerning their water usage. This is accomplished through the use of simulations for water meters that are attached to the user. The simulations allow users to set the flow rate of their meter (done in gallons per minute) and will update the amount of water used within our server. The user also can set alerts that will update them, only when they have met a specific threshold that they have set. This is a very simple overview of the applications major functions, with more detailed descriptions to follow.

Signup / Login

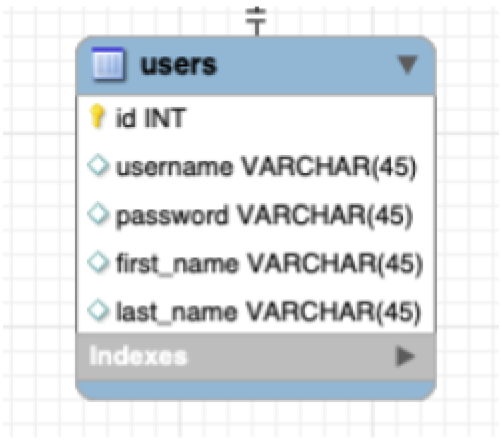


Fig 1.1 Database

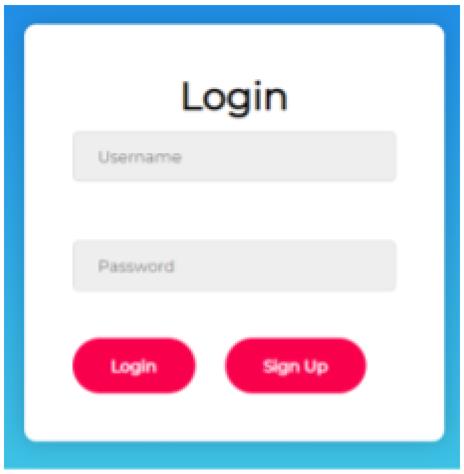


Fig 1.2 Login Component

Signup and login functionality is provided to ensure users have personalized access to the water monitoring services.

User Home Page

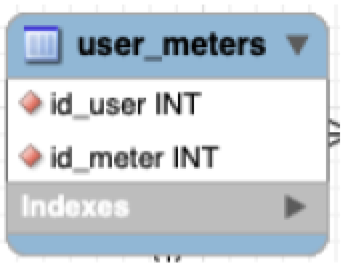


Fig 2.1 User Meters

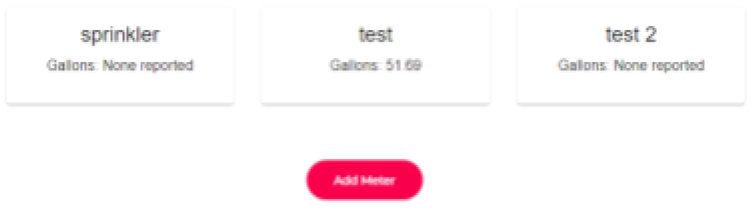
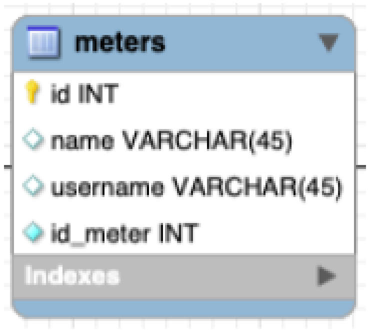


Fig 2.2 User Home Page

The user's home page allows the user to see the status of all current meters. It also provides access to the add meter functionality.

Add Meter



meters
id INT
name VARCHAR(45)
username VARCHAR(45)
id_meter INT
Indexes

Fig 3.1 Meters Table

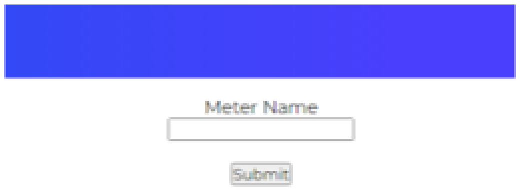
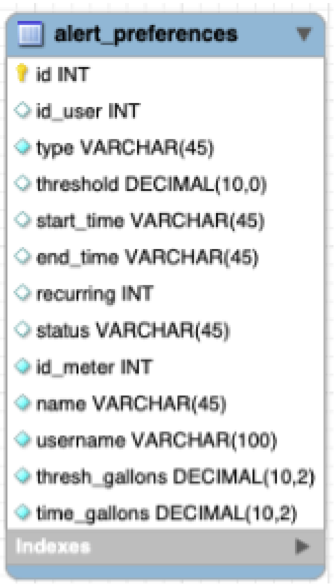


Fig 3.2 Add Meter Component

Add meter allows the user to link all of their meters to their account. Once a meter is added, the user is able to see its current status on the home page. The user may add any number of meters to their account.

Add Alert Preferences



alert_preferences
id INT
id_user INT
type VARCHAR(45)
threshold DECIMAL(10,0)
start_time VARCHAR(45)
end_time VARCHAR(45)
recurring INT
status VARCHAR(45)
id_meter INT
name VARCHAR(45)
username VARCHAR(100)
thresh_gallons DECIMAL(10,2)
time_gallons DECIMAL(10,2)
Indexes

Fig 4.1 Alert Preferences

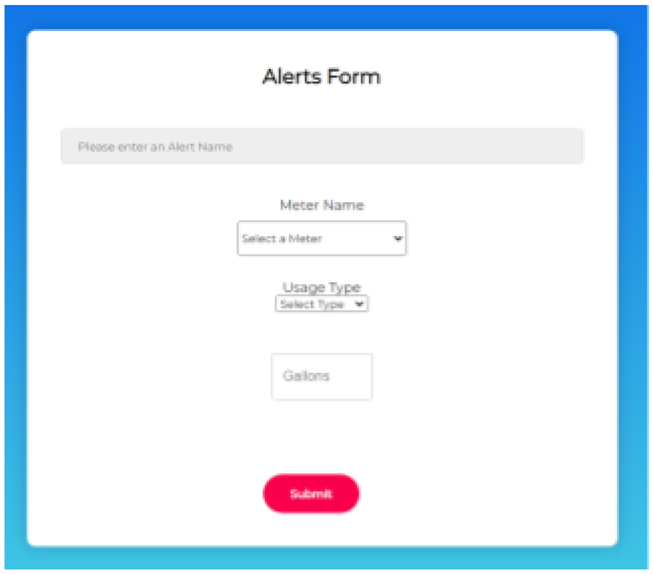
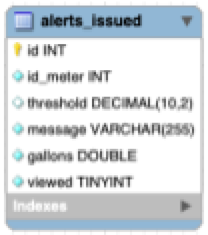


Fig 4.2 Alert Form Component

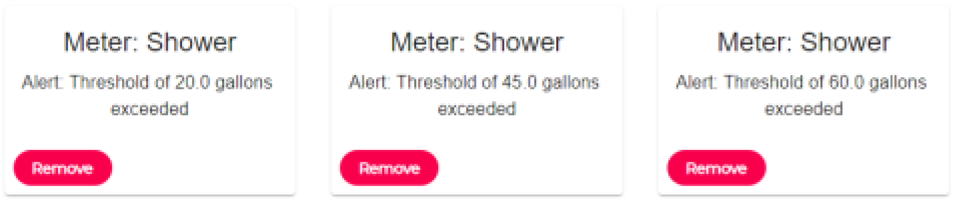
The add alerts functionality allows the user to define alerts they would like to receive for each of their meters. The user may define as many alerts as they like. Two types of alerts are offered. The user can set an overall threshold that will issue an alert once their meter has used a user specified number of gallons. In addition, the user can define time-based alerts, which will issue an alert if the user specified number of gallons has been utilized within the time-frame specified by the user.

View Alerts



alerts_issued
id INT
id_meter INT
threshold DECIMAL(10,2)
message VARCHAR(255)
gallons DOUBLE
viewed TINYINT
Indexes

Fig 5.1 Alerts Issued



Meter: Shower
Alert: Threshold of 20.0 gallons exceeded

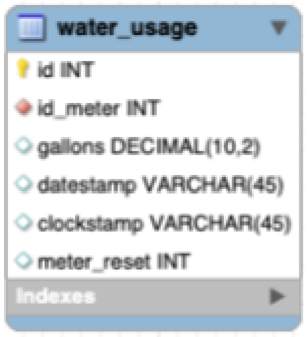
Meter: Shower
Alert: Threshold of 45.0 gallons exceeded

Meter: Shower
Alert: Threshold of 60.0 gallons exceeded

Fig 5.2 Alerts Page

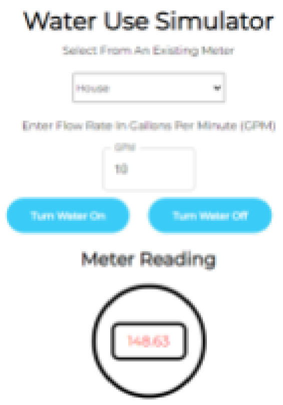
Upon clicking on the alerts button on the home-page you will be able to see which thresholds have been reached in the amount of gallons that have been consumed in the simulation.

Water Use Simulator



water_usage
id INT
id_meter INT
gallons DECIMAL(10,2)
datestamp VARCHAR(45)
clockstamp VARCHAR(45)
meter_reset INT
Indexes

Fig 6.1 Water Usage



Water Use Simulator
Select From An Existing Meter
House
Enter Flow Rate In Gallons Per Minute (GPM)
GPM
10
Meter Reading
148.63

Fig 6.2 Water Simulator

The water usage simulator acts as a virtual water faucet. It allows the user to define any of their existing meters to be attached to it. The user can turn the faucet on and off and specify the flow rate. The system will then update the selected meter with the number of gallons that have been utilized. This allows users to test out the functionality of the website without the need for physical hardware.

Backend Server

The backend server provides APIs for use by both physical meter hardware and the frontend website. These include endpoints capable of the following:

- Creating a new user
- Logging into the website
- Posting water usage data for a given meter
- Addition of a new meter to an existing user
- Addition of a new alert preference to an existing meter
- Retrieval of a user's existing meters and usage data
- Marking of an alert as viewed

USABILITY TEST PLAN

The usability test plan consisted of utilizing a focus group of testers who met our desired demographic. We used a focus group consisting of Brad, Jillian, Rosalie, Keith, and Sabrina. They are all adults with differing ages. We did not have anyone younger than twenty-two or older than sixty-five. They fit our target audience because they are adults who are familiar with using computers and were able to interact with our web page with ease.

Testers were each provided the same form based test scripts and were asked to complete the form while executing the provided scenarios. Scenarios were provided for each of the major components of the application. Testers were also encouraged to navigate the site after the specified scenarios had been completed. Testers were observed by a member of the team while performing testing.

After testing was completed, the team reviewed the details the testers provided in their forms. The team also discussed what was observed while the tester was interacting with the application. This detail was used to identify bugs and enhancements that were then prioritized into items that could be addressed prior to final submission of the project and those that would have to wait until after submission.

USABILITY TESTING/EVALUATION

Our usability testing included having each of us pick a tester who would follow a series of instructions while walking through our application. A user only needed a personal computer with internet to run the water meter simulations.

We used a focus group consisting of Brad, Jillian, Rosalie, Keith, and Sabrina.They are all adults with differing ages. We did not have anyone younger than twenty-two or older than sixty-five. They fit our target audience because they are adults who are familiar with using computers and were able to interact with our web page with ease. We would like to enable the target client/group to test the usability of our website simulation of the water monitoring tool. Each user will be able to create an account and sign in. A user will be able to navigate to their home page to view their added water meters. In addition visit several other tabs that will allow the adding of meters, view a simulation of a desired meter, and read more about our program with illustrations of how it works behind the scenes.

When setting up the focus-group tests, the one thing that we wanted to achieve was simplicity. This was accomplished by guiding the user through different scenarios via instructions in the form. This ranged from simple tasks from creating the account and signing in, to creating alerts and adding more meters for the user to manage. For the most part, responses to the website were well received with users finding most of the categories very simple and easy to understand. However, there were some bugs that were occurring for a few users that soured the taste of an otherwise well functioning site.

Appendix A was sent to testers, with feedback responses routed automatically once submitted. Appendix B were the test scenarios of the most importance to using the web site, and contained some of the most fundamental tasks such as creating an account and navigating through the different pages. Below are the results of feedback received from each of the five testers.

Keith found the overall navigation between pages was easy. Through the process of signing in and logging out, Keith experienced issues logging in with his credentials after signing up. While creating an account, it was recommended by Keith that we provide the option to view the password being typed in (eye). The adding of water meters was easy to understand and implement for Keith in addition to running the simulation. The sections of the webpage were not relevant to the average user who would not understand code as illustrated in the API page.

Rosalie was able to complete all of the assigned testing tasks while only encountering minimal issues. She did get stuck when she was asked to add a new alert preference. She indicated she was unable to determine whether or not her alert preference had been added. She also provided feedback that while using the water faucet simulator she was expecting triggered alerts to become visible without having to navigate away from the simulator. Also, she indicated an expected alert was not triggered upon initial use of the simulator.

Brad found the web site and tasks easy to perform. He did not provide any critique or recommendations to improve the site. It was a good overall user experience. He completed the use remotely without any assistance.

Jillian described the site as easy to use and offered some critique in areas that were not intuitive to her. She recommended adding a top navigation that allows a user to sign up. She also questioned why a user needs to log back in after creating a meter. She found adding an alert difficult and stated the site kept giving her an error when she entered less than 50. She was unable to complete this task. Although she found the task of turning the simulator on and off she recommended leaving the currently pushed button highlighted. The API did not interest her and she felt the text was difficult, most likely because it is written for programmers.

Sabrina was able to complete all of the tasks assigned to her, with a few bugs here and there. Some of the bugs that occurred were common threads amongst all of the testers in our focus-group. One of which being the "Signup" page not redirecting or giving verification to the user to proceed to a different page to login. Once she successfully logged in, she was able to easily create a meter and simulate the meter. Which is where she ran into a small issue where alerts were not loading. All in all, Sabrina found the site to be easy to navigate and all around simple for users. Although she may not have a coding background, she did find the API page interesting and asked how it operates.

TIMELINE/BUDGET

We were able to meet the milestones set in the proposal although some of them were not on schedule. In particular the implementation of the frontend was not completed until the week of usability testing. This was due to having to learn some new aspects of REACT that were unfamiliar to the team, such as automatic redirecting of pages. However, we had planned for the implementation of the frontend going longer and were able to absorb most of that effort into our fourth week.

For our budget, we had planned on not incurring any out of pocket expenses. In order to accomplish this we planned on using free resources such as GitHub, Replit, and Heroku to develop and host our application. One issue we encountered was Heroku changed its usage policies and was set to eliminate free accounts midway through our effort. However, we learned they replaced this with a credit based system they implemented in partnership with GitHub for students. We were able to secure credits and stick to our budget.

FINAL IMPLEMENTATION

The application consists of a frontend web application that users can interact with to manage their accounts and view their usage data and a backend server connected to a database that can receive data from meters and manage requests from the frontend application.

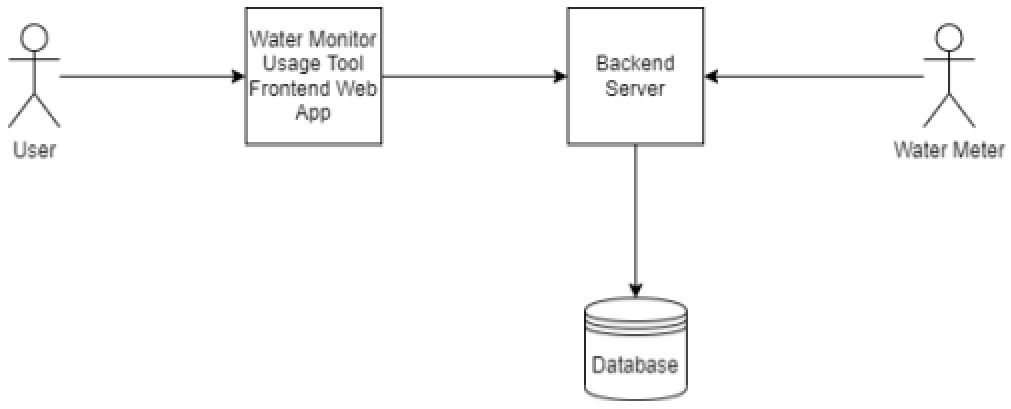
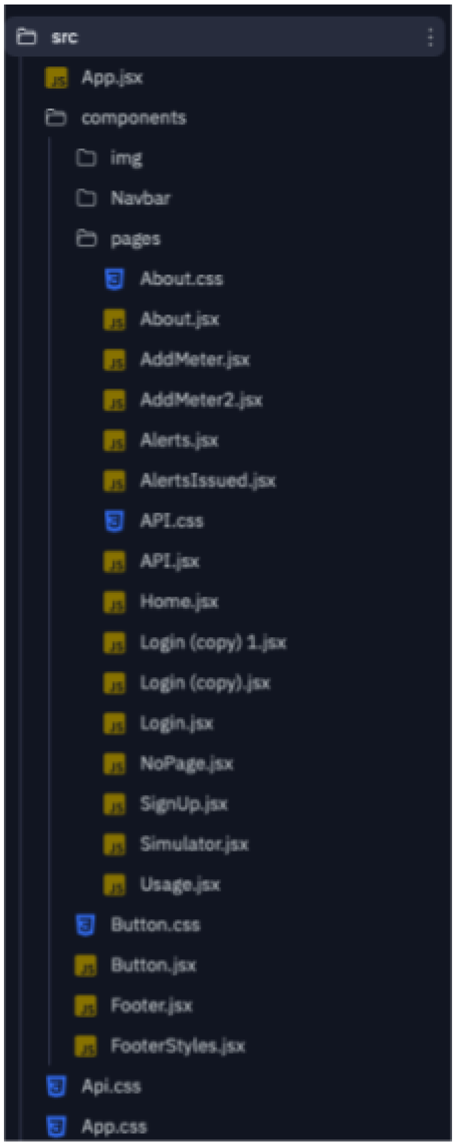


Fig 7.1 Application Structure

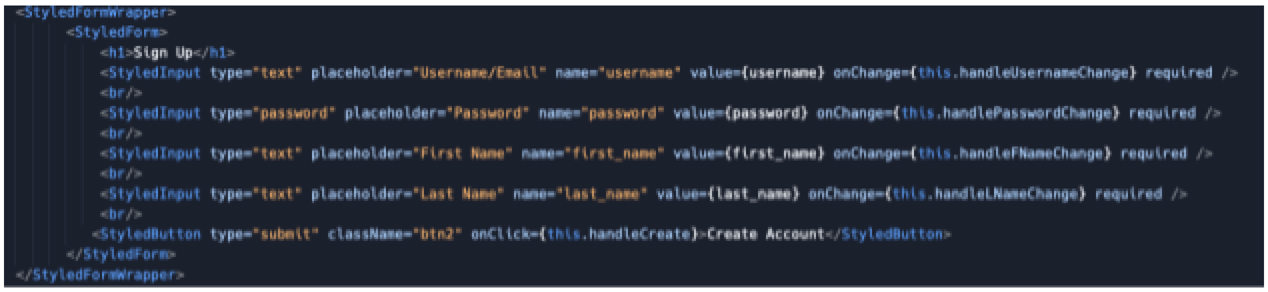
The front end was implemented using Replit to store all of the pages that can be viewed on the web page. Below you will see the overall folder structure on the water monitoring Replit and some of the code used to implement a particular feature.



```
src
  App.jsx
  components
    img
    Navbar
    pages
      About.css
      About.jsx
      AddMeter.jsx
      AddMeter2.jsx
      Alerts.jsx
      AlertsIssued.jsx
      API.css
      API.jsx
      Home.jsx
      Login (copy) 1.jsx
      Login (copy).jsx
      Login.jsx
      NoPage.jsx
      SignUp.jsx
      Simulator.jsx
      Usage.jsx
    Button.css
    Button.jsx
    Footer.jsx
    FooterStyles.jsx
  Api.css
  App.css
```

Fig 7.2 Frontend File Structure

The front end for the sign up page and code are displayed below. It consists of code to communicate with the backend Spring Boot server. The user enters into the text boxes and when the button is clicked a fetch call is executed with JSON data provided to the backend SpringBoot server which handles the request by adding the value to the database.



```
<StyledFormWrapper>
    <StyledForm>
        <h1>Sign Up</h1>
        <StyledInput type="text" placeholder="Username/Email" name="username" value={username} onChange={this.handleUsernameChange} required />
        <br/>
        <StyledInput type="password" placeholder="Password" name="password" value={password} onChange={this.handlePasswordChange} required />
        <br/>
        <StyledInput type="text" placeholder="First Name" name="first_name" value={first_name} onChange={this.handleFNameChange} required />
        <br/>
        <StyledInput type="text" placeholder="Last Name" name="last_name" value={last_name} onChange={this.handleLNameChange} required />
        <br/>
       <StyledButton type="submit" className="btn2" onClick={this.handleCreate}>Create Account</StyledButton>
    </StyledForm>
</StyledFormWrapper>
```

Fig 7.3 StyledForm for Sign Up

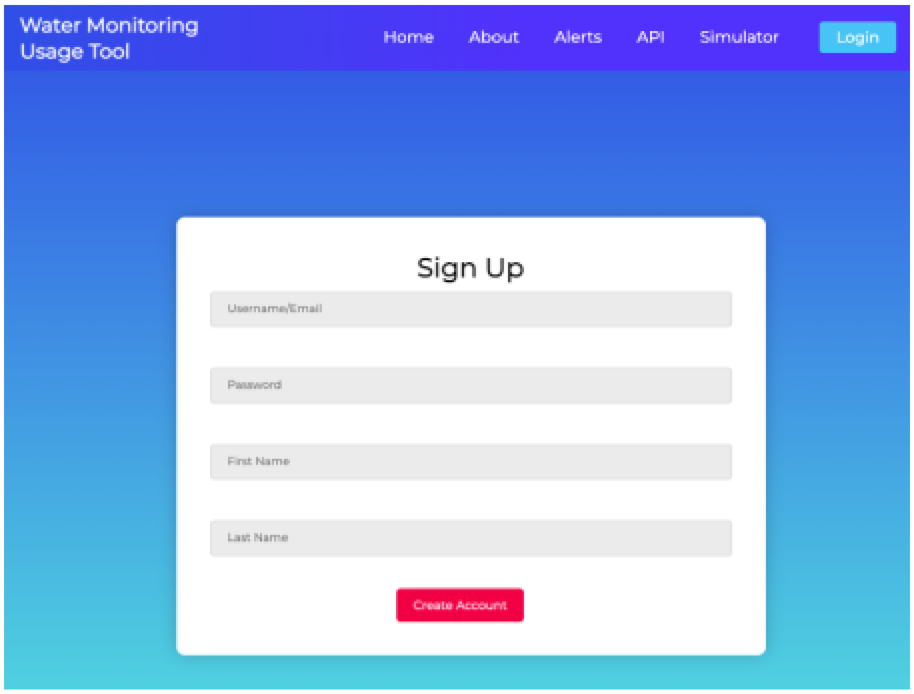
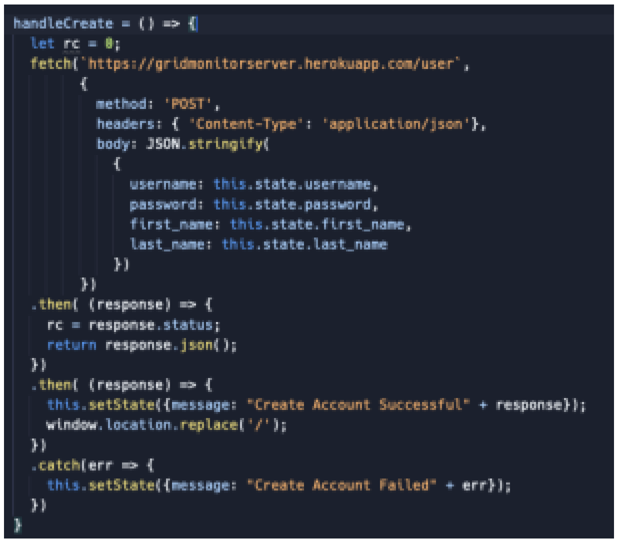
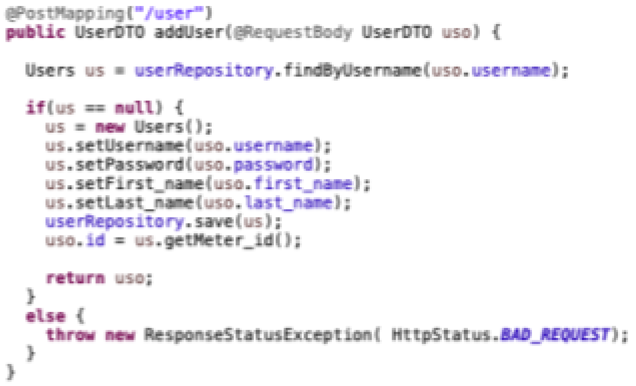


Fig 7.4 Sign Up Result



```
handleCreate = () => {
  let rc = 0;
  fetch(`https://gridmonitorserver.herokuapp.com/user`,
        {
          method: 'POST',
          headers: { 'Content-Type': 'application/json'},
          body: JSON.stringify(
            {
              username: this.state.username,
              password: this.state.password,
              first_name: this.state.first_name,
              last_name: this.state.last_name
            })
        })
  .then( (response) => {
    rc = response.status;
    return response.json();
  })
  .then( (response) => {
    this.setState({message: "Create Account Successful" + response});
    window.location.replace('/');
  })
  .catch(err => {
    this.setState({message: "Create Account Failed" + err});
  })
}
```

Fig 7.5 SignUp Handler

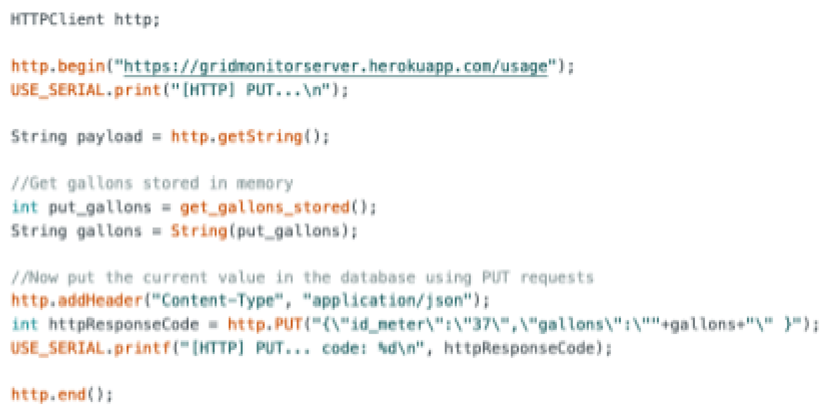


```
@PostMapping("/user")
public UserDTO addUser(@RequestBody UserDTO uso) {

  Users us = userRepository.findByUsername(uso.username);

  if(us == null) {
    us = new Users();
    us.setUsername(uso.username);
    us.setPassword(uso.password);
    us.setFirst_name(uso.first_name);
    us.setLast_name(uso.last_name);
    userRepository.save(us);
    uso.id = us.getMeter_id();

    return uso;
  }
  else {
    throw new ResponseStatusException( HttpStatus.BAD_REQUEST);
  }
}
```

Fig 7.6 User Creation on Backend

The hardware was implemented using an Espressif ESP-WROOM-32 microcontroller development (ESP32) board and Gredia ¾" Hall Effect water flow meter purchased from Amazon. The circuit board was designed using KiCad electronic design automation software. Other miscellaneous components purchased from Amazon include but are not limited to: junction box, socket connector, push button switch, micro usb cable, on/off switch, battery holder, and LED lights.

The water flow meter works with the ESP32 microcontroller to sense water flow and relay the usage back to the web page. This is accomplished by periodic pulses sent by the flow meter to the ESP32 microcontroller. The ESP32 processes the pulse and sends information to the web page in the form of a PUT request. Below is a piece of code stored on the ESP32 that is responsible for sending the PUT message and images of the hardware used to test the web page functionality.



```
HTTPClient http;

http.begin("https://gridmonitorserver.herokuapp.com/usage");
USE_SERIAL.print("[HTTP] PUT...\n");

String payload = http.getString();

//Get gallons stored in memory
int put_gallons = get_gallons_stored();
String gallons = String(put_gallons);

//Now put the current value in the database using PUT requests
http.addHeader("Content-Type", "application/json");
int httpResponseCode = http.PUT("{\"id_meter\":\"37\",\"gallons\":\""+gallons+"\" }");
USE_SERIAL.printf("[HTTP] PUT... code: %d\n", httpResponseCode);

http.end();
```

Fig 7.7 Hardware Backend Implementation

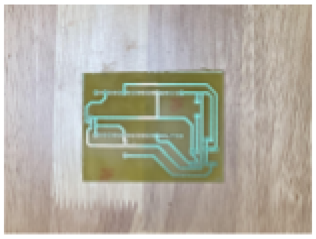


Fig 7.8 Circuit Board

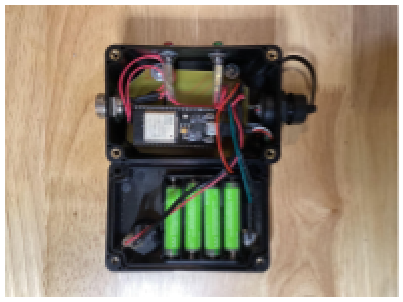


Fig 7.9 Hardware Device

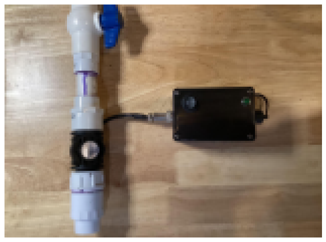
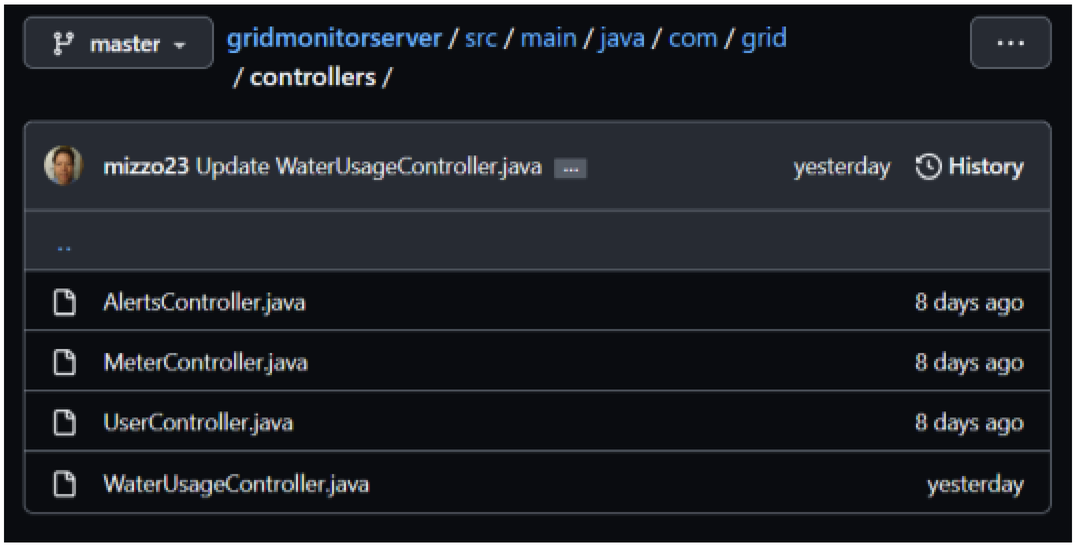


Fig 7.10 Hardware Attached to Meter

The backend was implemented using Java Spring Boot and a MySQL database. Application code is maintained on GitHub. Both the application code and the database run on a cloud based server hosted by Heroku.

The backend consists of multiple REST controllers. Each controller is focused on the different types of entities stored in the database. As seen below, we provide REST APIs to view, add, and update information related to users, meters, alerts, and water usage.



master

gridmonitorserver / src / main / java / com / grid / controllers /

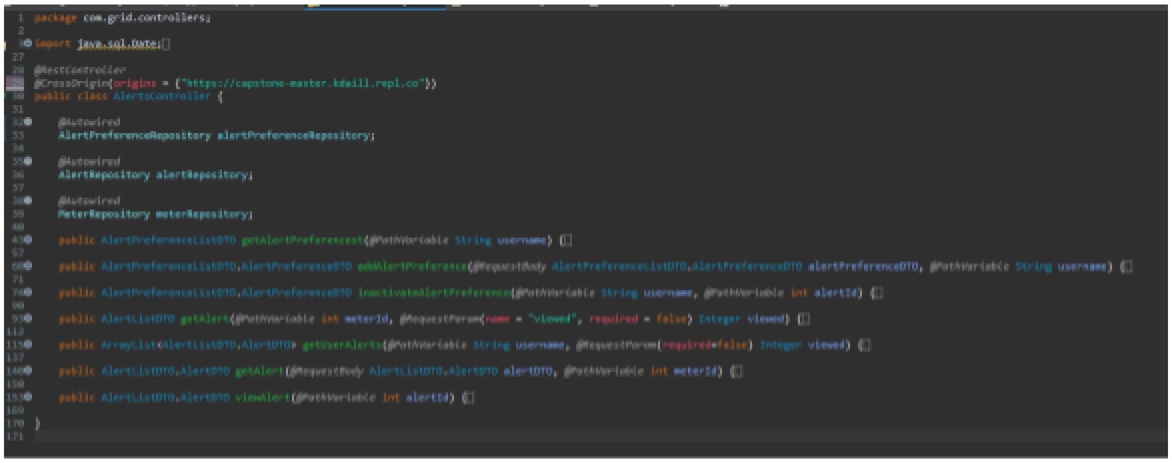
mizzo23 Update WaterUsageController.java · yesterday · History

..

AlertsController.java	8 days ago
MeterController.java	8 days ago
UserController.java	8 days ago
WaterUsageController.java	yesterday

Fig 7.11 Backend Controllers

Here is an example of the services offered by the Alerts Controller:



```java
package com.grid.controllers;

import java.sql.Date;[]

@RestController
@CrossOrigin(origins = {"https://capstone-master.kdaill.repl.co"})
public class AlertsController {

    @Autowired
    AlertPreferenceRepository alertPreferenceRepository;

    @Autowired
    AlertRepository alertRepository;

    @Autowired
    MeterRepository meterRepository;

    public AlertPreferenceListDTO getAlertPreferencest(@PathVariable String username) {[]

    public AlertPreferenceListDTO.AlertPreferenceDTO addAlertPreference(@RequestBody AlertPreferenceListDTO.AlertPreferenceDTO alertPreferenceDTO, @PathVariable String username) {[]

    public AlertPreferenceListDTO.AlertPreferenceDTO inactivateAlertPreference(@PathVariable String username, @PathVariable int alertId) {[]

    public AlertListDTO getAlert(@PathVariable int meterId, @RequestParam(name = "viewed", required = false) Integer viewed) {[]

    public ArrayList<AlertListDTO.AlertDTO> getUserAlerts(@PathVariable String username, @RequestParam(required=false) Integer viewed) {[]

    public AlertListDTO.AlertDTO getAlert(@RequestBody AlertListDTO.AlertDTO alertDTO, @PathVariable int meterId) {[]

    public AlertListDTO.AlertDTO viewAlert(@PathVariable int alertId) {[]

}
```

Fig 7.12 Alerts Controller

DISCUSSION

With any application, there are expected to be problems and bugs during every phase. This may come from the early stages with planning and designing, to the actual implementation of said designs. We also fell victim to these during our development.

Backend Struggles

Beginning with the backend server, the issues began with hosting and finding the right ways to allow our frontend to communicate with our database. The issue was encountered when we were initially setting up Heroku to become our backend host, using Jaws DB. This then led to connection issues that we encountered with our database. The issue that occurred was the max number of connections were being utilized at one time which we solved using connection pooling. Following the theme of database struggles, there were also instances of an out of sync database. We began noticing that the database became out of sync due to a possible delay from either Heroku or from our frontend host.

Frontend Struggles

Our frontend struggles were significantly lighter than our backend struggles. One of these being, building a better understanding of the file structure when using JSX syntax. Through research and studying documentation, we were able to resolve this issue to have a more solid understanding of the file structure. We also ran into issues with implementing a login method on the frontend in a way that would only show users their specific dashboard/portal. This was resolved through a new mapping that was included in the backend of the application to cross check the database with the information input by the user.

Rollout

The application rollout to the users was handled during user acceptance testing. Each tester was introduced to the application by one of the project team members. The team member explained the background and purpose of the application and then provided the user the test scenarios and a link to the application website. Testers were then asked to explore the application on their own and attempt to complete the test scripts. Based on user feedback, the users found the site to be intuitive and were able to complete all scenarios with little to no assistance.

Future Vision

In the future, the team envisions an extensive and user-friendly web application for all to use, regardless of background. Some of these user-friendly additions include:

- Providing enhancements to the web application to include push notifications.
- Providing filtering and grouping options for user meters based on geographic location.
- Provide instructions through video tutorials and walkthroughs.

Along with these, we envision a mobile application that allows users to access their dashboard from virtually anywhere. We would also like to provide programmers and developers, who are seeking to utilize our APIs for their own projects, a curated section giving them a more detailed resource to understand the backend services. Ultimately, the largest goal would be to find a more reliable hosting platform to better support and scale our product.

Reflection

Overall, the team dynamic that we have developed over the course of the program shined through during the course of this project. Like any project, this journey did not come without its fair share of bumps and bruises. However, we were able to overcome all obstacles through communication and teamwork and deliver a product we are all proud to present to our peers.

REFERENCES

Apartments Save $100,000 with AlertAQ^{TM} Water Intelligence. (2022). *Alert Labs Inc.* Retrieved from:

https://drive.google.com/file/d/1HMMEoPUg1KtQLXg3L_IMlNrA_w_EhYjh/view?usp=s

haring

GOV.CA. (2022, MAY24). *California Adopts More Aggressive Water Conservation Measures.*

Retrieved from:

https://www.gov.ca.gov/2022/05/24/california-adopts-more-aggressive-water-conservat

ion-measures/

APPENDIX A: GOOGLE FORMS

https://docs.google.com/forms/d/1Ki578wdGGIc2Sy_Fq6eUlb0jf0SQfTBOl3MRhQwMO0w/edit

APPENDIX B: TEST SCENARIOS

https://docs.google.com/spreadsheets/d/1MLr0g0bolUPErQs0_3sXTgrSUr5sdEgIJleXBs48ccw/e
dit?usp=sharing