

AQUA WATCHER



Water Quality Monitoring and Usage Monitoring System

3rd year unified project (E/16/049, E/16/134, E/16/388)

Design Manual

Contents

1.0 Introduction	4
1.1 Why is this product?.....	4
1.1.1 Problem	4
1.1.2 Solution	4
1.2 Our solution- Water Quality Monitoring and usage monitoring System.....	4
1.2.1 Why?	4
1.3 System Overview.....	5
2.0 Software Deployment	6
2.1 Web Server	6
2.1.1 Overview	6
2.1.2 Roles in the system.....	6
2.1.3 Functionalities of the Web Server.....	7
2.2 Database Deployment	14
2.2.1 Overview.....	14
2.2.2 Steps.....	14
2.2.3 Collections in the Database	19
2.2.4 Security	22
2.2.5 Reliability and Scalability	22
2.3 Website.....	23
2.3.1 Overview	23
2.3.2 Customer	23
2.3.3 SuperAdmin View	25
2.3.4 Admin	26
2.4 Mobile Application	27
2.4.1 Overview.....	27
2.4.2 Notifications and Alerts.....	27

2.4.3 Security of Mobile Application	28
2.4.4 Configuration of the Mobile Application	28
2.4.5 Additional Resource Documentation for the Mobile Application	31
2.5 Embedded Software	31
2.5.1 Overview.....	31
2.5.2 Functionality : ESP 12E Chip(Node MCU)	32
2.5.3 Functionality : ATmega 328p Chip.....	35
2.6 Amazon Web Services cloud deployment.....	37
2.6.1 Overview.....	37
2.6.2 Steps.....	37
2.6.3 Scalability & Reliability	46
2.6.4 Security	46
3.0 Hardware Development	47
3.1 Required Components	47
3.1.1 Sensors	47
3.1.2 Tasks of other components	48
3.2 Designs and Diagrams.....	49
3.2.1 Designs for 3D printing.....	49
3.2.2 Circuit Diagrams	52
3.2.3 PCB and Schematic Designs.....	53
4.0 Security, Reliability and Scalability.....	54
4.1 Security.....	54
4.2 Reliability.....	55
4.3 Scalability	56
4.0 Resources.....	56

1.0 Introduction

1.1 Why is this Product?

1.1.1 Problem

Domestic Water wastage is one of the main problems faced by the people. This happens especially when people fill their water tanks and forget to shut the valve after the tank becomes full. And also people have no idea of the purity of the water which they are consuming. So we should find the solutions to reduce these problems.

1.1.2 Solution

Our solution to the above problems are a system that monitors the water level of the water tanks and at the same time monitors the quality of the water which fills the water tanks in a user friendly way.

1.2 Our solution- Water Quality Monitoring and usage monitoring System

1.2.1 Why?

In the modern world everything is done by using the latest technologies. So especially In Sri Lanka there is not any current method that monitors the water level and water quality of water tanks in a user friendly way.

So our system provides a way to monitor these things in a user-friendly way. By using simple android app and website users can easily monitor the water quality and water level of the tanks.

1.3 System Overview

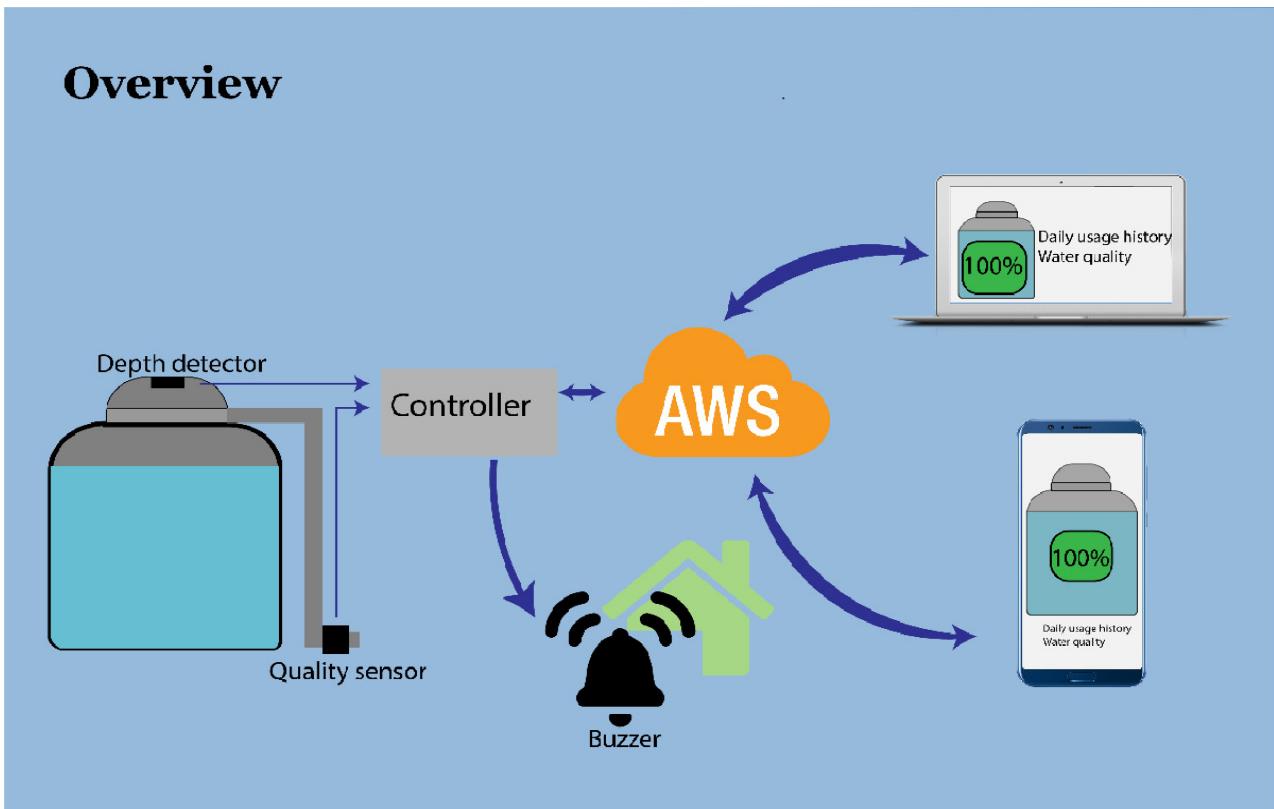


Figure 01

In above figure it can be seen that we use depth detector to get the water level of the tank and we use sensors to measure the quality of the water which fills the tank. Turbidity and TDS sensors are used to measure the water quality. And the measurements of these sensors are sent to AWS server wirelessly and provide those data to our website and mobile app. Users can monitor the data by using the website and also using the mobile app. If the Water level is in required level or if the water quality is not good then the user gets notification from the app and also it is indicated in the website. So any kind of user get notified doesn't matter if he/she uses a website or mobile app.

2.0 Software Deployment

2.1 Web Server

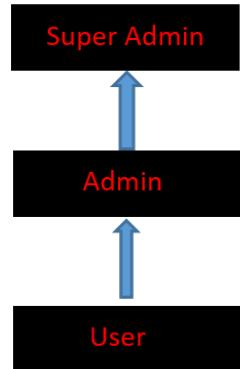
2.1.1 Overview

- Web server is developed using node.js with MongoDB cloud database deployment.
- MVC architecture is used here, therefore each section of the server is well isolated.
- The server is using http protocol and implemented as a stateless application. It supports POST, GET, PUT, DELETE requests. Hence it is designed to do create, read, update and delete (CRUD) operations.
- The web server is handling user requests with the collaboration of the database. Web server exchanges data between clients and database in JSON format.
- The server employs an internal router to route client requests to required endpoints.
- Apart from this, the server has many Security features, Controllers, Middleware and employs various object model types as well.
- The server uses asynchronous methods to improve efficiency.
- We have deployed 2 servers parallelly in ports 3000, 3001. All the requests will be going via port 80 of the main computer and handled using an NGINX server.

2.1.2 Roles in the system

- There are 3 main roles in our system. Each client with any role must use a unique email address to register on our system. They are listed with their functionalities below,
 1. User (Customer) – This is our customer. He can register devices to our service. By using either our website or mobile app, he can monitor those devices and receive notifications, warnings, and emails.

2. Admin (Technician) – This is our system administrator. He can monitor our system to find customers with water quality issues. Also, he can issue notices to customers as well. Responsibility to add, delete, modify user details are falls to admins.
3. Super Admin (Supervisor) – They are supposed to provide or remove access to the Admins into the system. They don't have access to interact with Customers.



Hierarchy of Roles

2.1.3 Functionalities of the Web Server

1. Login clients

- During a login operation, the client will send email and password in a JSON object such as {email : "user mail",password: "password"}. The server will cross-reference the data with the database and verify it.
- If the data is wrong, the server will send a response sending a JSON object of errors with a response code of 400.
- If the data matches those of a registered account, then it will create a jwt token with the account id, which is unique to each account, and set a reasonable expiration time. Also, the account name and role are acquired by the database and stored in a JSON object. Then response is sent to the client with that JSON object as body, jwt token as a header, and with 200 response code.

2. Authorization middleware

- Each client request will go through authorization middleware. Since a jwt token created with the unique account id is in request headers, the server verifies and

decodes the token to identify the user and role. Then grant the request if the request is within the scope of the client's scope. Otherwise redirected to a warning page.

- Also if the jwt token is expired in the client requests, the client is logged out. Therefore redirected to the login page.
- Each account in the database has a boolean value “isverified”. If this value is false, the account is registered but not verified. Therefore the user will be redirected to verification steps.

3. Sending data to clients

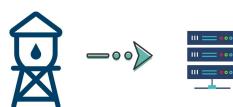
- A client sends GET requests to various endpoints in the server. All the responses are sent in JSON.

Ex:-

```
{tds: '110', turbidity: '2', level: '894', usage: '100', issue: false, _id: 6016ea5f7f441b019c88268f, email: 'thusharaweerasundara@gmail.com', area: 'kiribathkumbura', no: '3', created_at: 2021-01-31T17:35:27.907Z, _at: 2021-05-23T16:14:21.937Z, __v: 0}
```

4. Receiving Data from Devices

- During the registration process, each device will be assigned a device id corresponding to the user's email and device number. Therefore each device will have a unique endpoint. Readings from the devices will come as PUT requests to each endpoint in JSON format bodies with sensor readings. Data will be verified and then stored in the database if it passes the verification. Then 200 response codes will be sent, otherwise, 400 is sent.



5. Security Features

- Password hashing by “bcrypt” library to prevent user information compromises.
- Email verification to filter bot accounts.
- Authorization middleware to limit requests to corresponding role's scope.

- Setting burst limits using “ddos” library and set “rate limit headers” to prevent DDoS attacks. This blocks the IP addresses which send a burst of requests at the same time and sends 429 responses. Those IP addresses will be blocked for 1 hour.
- Hiding Server information to prevent specially designed attacks to the server.
- Using jwt tokens for client recognition using the “jsonwebtoken” library.

6. Sending Emails

- “nodemailer” library is used to handle communication via emails. Email addresses will be taken from the database. During registration, verification emails will be sent. Also, Admins can send notices as emails to Customers as well.

To allow emails via apps,

1. Go to this URL: <https://myaccount.google.com/lesssecureapps>
2. Log In to your designated email account.
3. Switch on “Allow less secure apps: ON”.

[← Less secure app access](#)

Some apps and devices use less secure sign-in technology, which makes your account vulnerable. You can turn off access for these apps, which we recommend, or turn it on if you want to use them despite the risks. Google will automatically turn this setting OFF if it's not being used. [Learn more](#)

[Allow less secure apps: ON](#)



7. Error handling

- In designing the server, user inputs are not to be trusted. They might cause harm to the system if not validated and checked for errors. Also, the data that comes from the devices could be erroneous as well. Therefore the error handling mechanism is integrated into the server.
- When outside data is received, before processing and storing they are validated and checked for errors. In case of no issues, data will be processed and stored accordingly. Otherwise, data will be ignored and the corresponding error message will be sent back.
- When the user inputs are received, a controller method called “handleErrors” will validate the data and generate error messages if there are any.

- When the data from devices are received data size, data type, data integrity, and whether the data value is within the numerical range is checked and stored using a controller method called “readings_put”.

8. Requests of each role

Role	Endpoint	Type	Purpose
All	/login	GET / POST	To login to account
All	/logout	GET	To logout
Super Admin	/superView	GET	Home page of Super Admin
Super Admin	/removeAdmin/:id	DELETE	To remove an Admin
Super Admin	/adminSignup	GET / POST	To register an Admin
Admin	/adminView	GET	Home page of Admin
Admin	/announce	GET	To send notices
Admin	/customers	GET	To get customer details by User name or Area
Admin	/remove/:id	DELETE	To remove a customer
Admin	/update	PUT	To update customer details
Admin	/report	GET	To get a system report
Admin	/name/:id	GET	To get customer details by User name
Admin	/area/:id	GET	To get customer details by Area
User	/home	GET	Home page of User

User	/tanks	GET	Get tank details
User	/statistics	GET	Get usage statistics
Device	/readings/:id	PUT	Update new readings
All	/restricted	GET(Redirect)	Redirect to here on out of scope requests from a role
All	/notfound	GET	On invalid requests

9. Controllers and functionalities

Controller Method	Functionality
handleErrors	Validate user inputs on login/signup
createToken	Create jwt tokens
notice_post	Send notices to customers
name_get	Get user information by user name
area_get	Get user information by user area
removeAdmin_delete	Remove admins
readings_put	Validate and update data for a tank
user_put	Validate and update user data
remove_delete	Remove a customer
userStatistics_get	To get usage statistics
report_get	To get system reports
userData_get	To get customer's tank data
userReg_post	Register customers and their tanks

adminSignup_post	Signup new admin
signup_post	Signup new customer
login_post	Handle login requests
logout_get	Handle logout requests
confirmation_Post	Handle email account confirmation

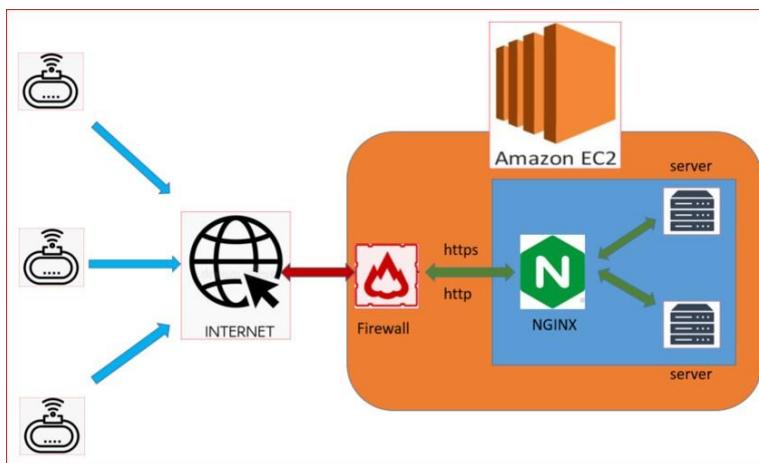
9. External libraries

Library	Purpose
Express	To overall deployment
Mongoose	Connect and control MongoDB database
Fs	To read key.pem and certificate.pem files
https	To implement https security features using .pem files
ddos	To limit request bursts
crypto	To create random hex values as tokens for account confirmation

nodemailer	To send emails
jsonwebtoken	To create jwt tokens
express-rate-limit	To limit the number of requests within a given window.
validator	To validate user inputs.

10. Reliability and Scalability

- This web server is expected to support a large number of clients. Therefore workload could be high on a single server
- Nginx server is configured to support 2 servers parallelly.
- Servers will receive requests in a round robin manner.
- Even if a single server is down or busy, requests will be served by the next server in line.
- This method can be scaled for many number of servers.
- You would have to increase the available CPU and RAM capacities for this.
- Since requests will be handled even when some servers are down or busy, system is highly reliable.
- Nginx configurations are in our github repo.



2.2 Database Deployment

2.2.1 Overview

- Database deployment is done using MongoDB cloud-based database.
- Mongoose external library was used to deploy with Node.js server.
- Data validation and error checking methods are included.
- Outside access to the database is limited.
- Data backup is enabled.

2.2.2 Steps

- 1. Create a MongoDB account by using a unique email.**
- 2. Login to your account**
- 3. Select Projects from the sidebar.**

The screenshot shows the 'UOP' interface with the 'Projects' tab selected in the sidebar. The main area displays a table with one project entry: 'Project 0' (1 Cluster, 2 Users, 0 Teams, 0 Alerts). A 'New Project' button is located in the top right corner of the main area. The sidebar also includes links for 'Alerts', 'Activity Feed', 'Settings', 'Access Manager', 'Billing', and 'Support'.

- 4. Select the New Project option and Name your project.**

The screenshot shows the 'Create a Project' form. At the top, there are tabs for 'Name Your Project' and 'Add Members'. Below the tabs is a note: 'Project names have to be unique within the organization (and other restrictions)'. A 'Project Name' input field is present. At the bottom right are 'Cancel' and 'Next' buttons.

5. Add Members and Set Permissions. You can give access to your colleagues here.

✓ Name Your Project > Add Members

Add Members and Set Permissions

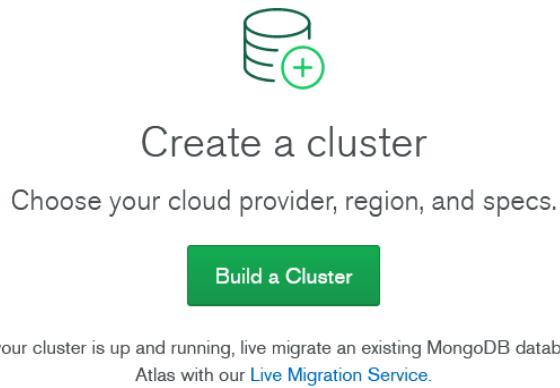
e16049@eng.pdn.ac.lk

Give your members access permissions below.

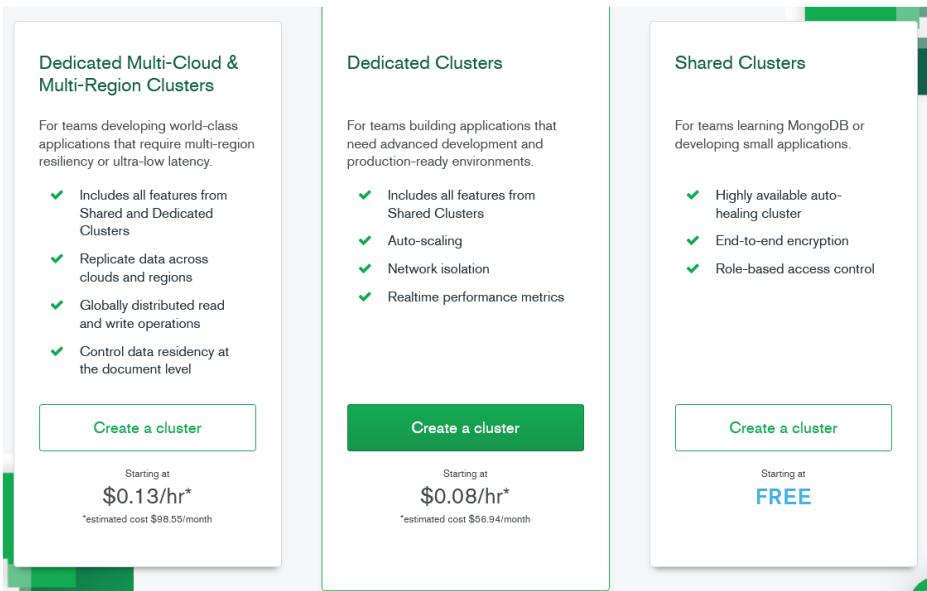
e16388@zohomail.com (you) Project Owner

Cancel ← Go Back Create Project

- 6. Then click Create Project.
7. Then click Build a Cluster.**



- 8. Select your package.**



9. Select provider and region. Here you should select aws as the provider.

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ▾

aws **Google Cloud** **Azure**

★ Recommended region ⓘ

NORTH AMERICA	ASIA	EUROPE
N. Virginia (us-east-1) ★	Singapore (ap-southeast-1) ★	Frankfurt (eu-central-1) ★
Oregon (us-west-2) ★	Mumbai (ap-south-1)	Ireland (eu-west-1) ★

10. Select RAM, Storage, and CPU preferences.
11. Turn on Backup if you are using a paid package.

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) Encrypted

Base hourly rate is for a MongoDB replica set with 3 data bearing servers.

Shared Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
M0 Sandbox	Shared	512 MB	Shared	Free forever
M0 clusters are best for getting started, and are not suitable for production environments.				
500 max connections Low network performance 100 max databases 500 max collections				
M2	Shared	2 GB	Shared	\$9 / MONTH
M5	Shared	5 GB	Shared	\$25 / MONTH

Additional Settings

MongoDB 4.4, No Backup

Turn on Backup (M2 and up)

See Backup Solutions for Paid Clusters (M2+)

- 12. Then click on Create Cluster and wait until it is created.**
- 13. Then click on Connect.**
- 14. Then add the IP addresses you want to have connections with this cluster. You should add your IP address here and. Then click Add IP Address.**
- 15. Also add a user who should have permission to access this cluster and assign a password. Then click Create Database User.**

① Add a connection IP address

IP Address	Description (Optional)
61.245.170.76	An optional comment describing this entry
<input type="button" value="Cancel"/> <input type="button" value="Add IP Address"/>	

② Create a Database User

This first user will have **atlasAdmin** permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username	Password	<input type="button" value="Autogenerate Secure Password"/>
user	*****	<input type="button" value="SHOW"/>
<input type="button" value="Create Database User"/>		

- 16. Now any unauthorized access will be blocked.**

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.

The screenshot shows a list of connection methods:

- Connect with the mongo shell**: Interact with your cluster using MongoDB's interactive Javascript interface.
- Connect your application**: Connect your application to your cluster using MongoDB's native drivers.
- Connect using MongoDB Compass**: Explore, modify, and visualize your data with MongoDB's GUI.

17. Then choose a connection Method. Since you are trying to connect this database with an application. Select Connect your application method.

① Select your driver and version

DRIVER	VERSION
Node.js	3.6 or later

② Add your connection string into your application code

Include full driver code example

```
const MongoClient = require('mongodb').MongoClient;
const uri = "mongodb+srv://user:<password>@cluster0.hkdmt.mongodb.net
/myFirstDatabase?retryWrites=true&w=majority";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology:
true });
client.connect(err => {
  const collection = client.db("test").collection("devices");
  // perform actions on the collection object
  client.close();
});
```

18. Select your Driver as Node.js and your Node.js version. Tick the Include full driver code example. Then copy the driver code and include it in your app. Replace <password> with the password for the user user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.

19. Now your database deployment is complete.

2.2.3 Collections in the Database

- Collections are defined as models using Node.js. Make sure their names begin with lowercase letters. Mongoose library is used for this.

customers :

Attribute	Type	Required	Maximum Length	Validation Method
firstname	String	true	20	isAlpha
lastname	String	true	20	isAlpha
contact	String	true	10	isInt
address	String	true	100	-
area	String	true	100	isAlpha
tanks	String	true	3	isInt
email(unique)	String	true	50	isEmail

report:

Attribute	Type	Required	Maximum Length	Validation Method
email(unique)	String	true	predefined	isEmail
no	String	true	predefined	isInt
monthlyUsage	String array	true	predefined	Not required

tank:

Attribute	Type	Required	Maximum Length	Validation Method
email(unique)	String	True	predefined	isEmail
area	String	True	predefined	Predefined
no	String	True	predefined	isInt
tds	String	True	predefined	isInt
turbidity	String	True	predefined	isInt
level	String	True	predefined	isInt
usage	String	True	predefined	isInt
issue	Boolean	True	predefined	Not required
created_at	Date	True	predefined	Not required
updated_at	Date	true	predefined	Not required

On new updates from devices,

- “issue” value will be calculated depending on water quality sensor readings.
- “updated_at” value will be updated to current time.

token:

Attribute	Type	Required	Maximum Length	Validation Method
_userId	Mongoose object id	true	Automatically set	Not required

token	String	true	Automatically set	Not required
createdAt	Date	true	Automatically set	Not required

user:

Attribute	Type	Required	Maximum Length	Validation Method
email(unique)	String	true	50	isEmail
password	String	true	Not required	Not required
role	String	true	predefined	roleValidator
isVerified	Boolean	true	predefined	Not required

- “roleValidator” method validates the client of a user belongs to, either user or admin or super admin.
- Also, this model has a login method to cross-reference email and password during login.
- “password” will be hashed using “bcrypt” library before storing.

- All these methods are Imported from validator library

isAlpha – check all the characters are alphabetic characters or not.

isInt – check whether numerical value is an integer or not.

isEmail- check if the given string is an email or not.

- When the application is started to execute, it will connect to the database. Then it will create required models in the collection as required.

2.2.4 Security

- Only collections “customer” and “tanks” store data from the outside. During customer registration, “customer” collection stores the customer inputted data. Also, “tanks” collection stores the data from the devices. Therefore when it comes to these 2 collections, inputs are thoroughly validated. If the validation fails, data will be dropped and the required warning message will be sent back. Validation happens using controller methods in the server.
- Outside access into the cluster is limited to selected IP addresses and users.(step 14 and 15)

2.2.5 Reliability and Scalability

- MongoDB automatically take snapshots of the data periodically in order to keep backups. (Step 11)
- Therefore in case of a database failure, it will be restored to reliability.
- MongoDB sharding allows us to scale our database. We need to enable Shard your cluster in our cluster and select the number of replicas we need.

Steps

1. Go to advanced settings in your cluster.
2. Enable Shard your cluster.
3. Set number of shards.

Advanced Settings

Shard your cluster (M30 and up)

YES

Sharding supports high throughput and large datasets, and can be increased as data requirements grow. Sharded clusters cannot be converted to replica sets.

2 Shards

Looking for more than 50 shards? [Contact MongoDB](#)

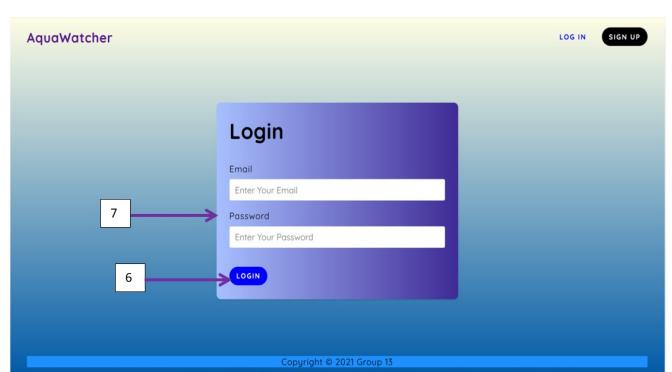
2.3 Website

2.3.1 Overview

- The aquawatcher website is built with html,css and php. It has role based access
- which is mentioned in [section 2.1.2](#).
- Dynamic web pages are used for efficiency.
- Different styles are used for different roles.
- Styles are in the public folder and views are in the views folder of our repo.

2.3.2 Customer

Customer is the user who uses the services and the product. After the user Brought the equipment , user has to create an account
www.aquawatcher.ml/signup

 <p>Water Quality and Usage Monitor</p> <p>User Sign Up!</p> <p>Email Enter Your Email</p> <p>Password Enter Your Password</p> <p>SIGN UP</p> <p>LOG IN</p> <p>SIGN UP</p> <p>Copyright © 2021 Group 15</p>	<ol style="list-style-type: none"> 1. Details required for signing up a user 2. Switch to login screen 3. Switch to signup screen 4. Signup after fill the details in form(4)
 <p>Water Quality and Usage Monitor</p> <p>Welcome, thusharaweerasingh@gmail.com LOG OUT</p> <p>User Details</p> <p>First Name</p> <p>Last Name</p> <p>Contact NO</p> <p>Address</p> <p>Area</p> <p>Copyright © 2021 Group 15</p>	<ol style="list-style-type: none"> 5. Details required from the user to complete the registration process.
 <p>AquaWatcher</p> <p>Login</p> <p>Email Enter Your Email</p> <p>Password Enter Your Password</p> <p>LOGIN</p> <p>LOG IN</p> <p>Copyright © 2021 Group 15</p>	<ol style="list-style-type: none"> 6. Login button 7. Details required to log in to the system

User Dashboard Screenshot:

- 8. Welcome, thusharaweerasundara@gmail.com LOG OUT
- 9. Product By GROUP 13
- 10. YOUR DEVICES
- 11. Total Usage of Today: 236 L
- 12. USAGE STATISTICS

8. The website link of the product
9. Link to the user guide
10. Link to contact system admins
11. Watch the user's devices
12. Access usage information

User Dashboard Screenshot:

- 13. Monthly Usage

13. Monthly usage graph

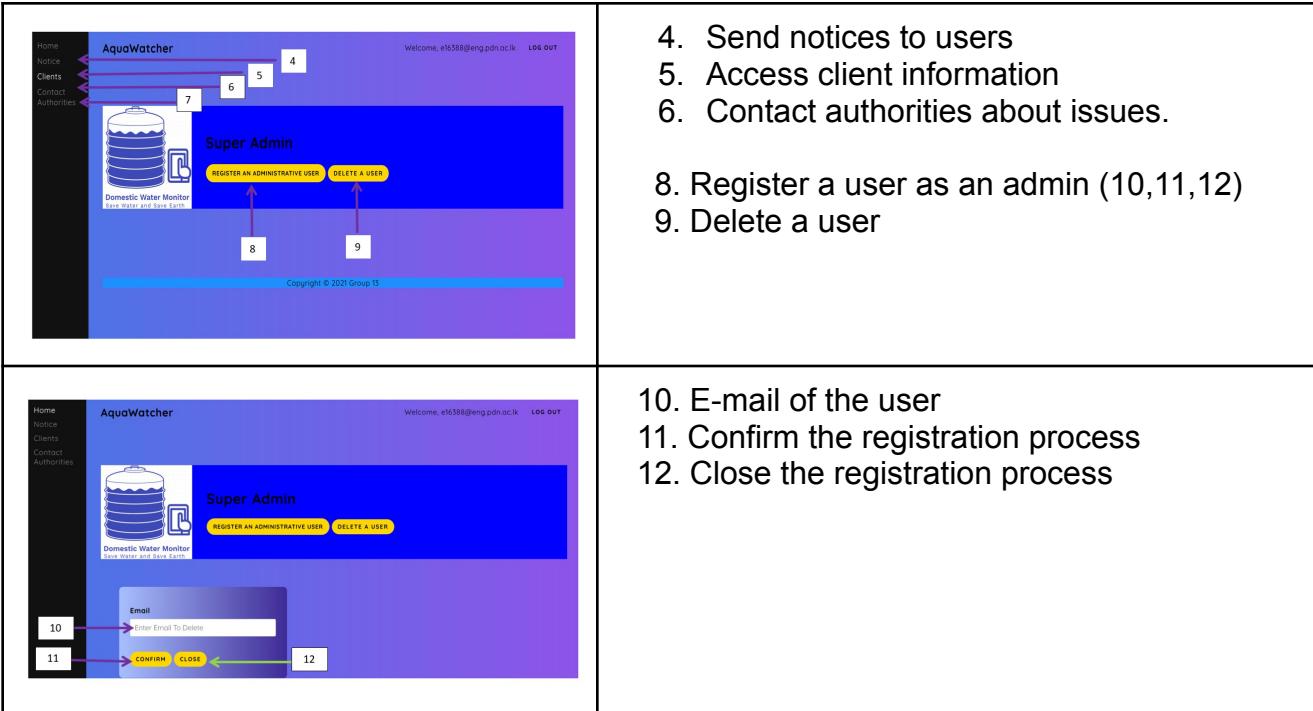
2.3.3 Super Admin View

The super Admin is the highest role in this model. Super admin can register a User as an admin or delete a user / admin. Also super admin can send notifications to the user , contact authorities about issues , take a system report and change the information of the user.

Super Admin Signup Screenshot:

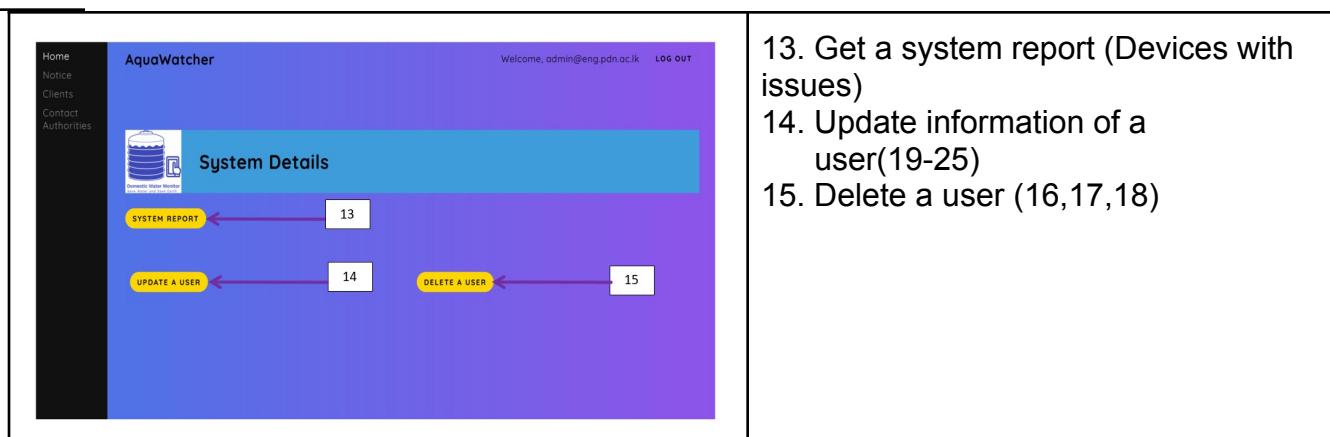
- 2. Enter Email of New User
- 3. Enter Password of New User
- 1. Admin Role: Super Admin
- Signup

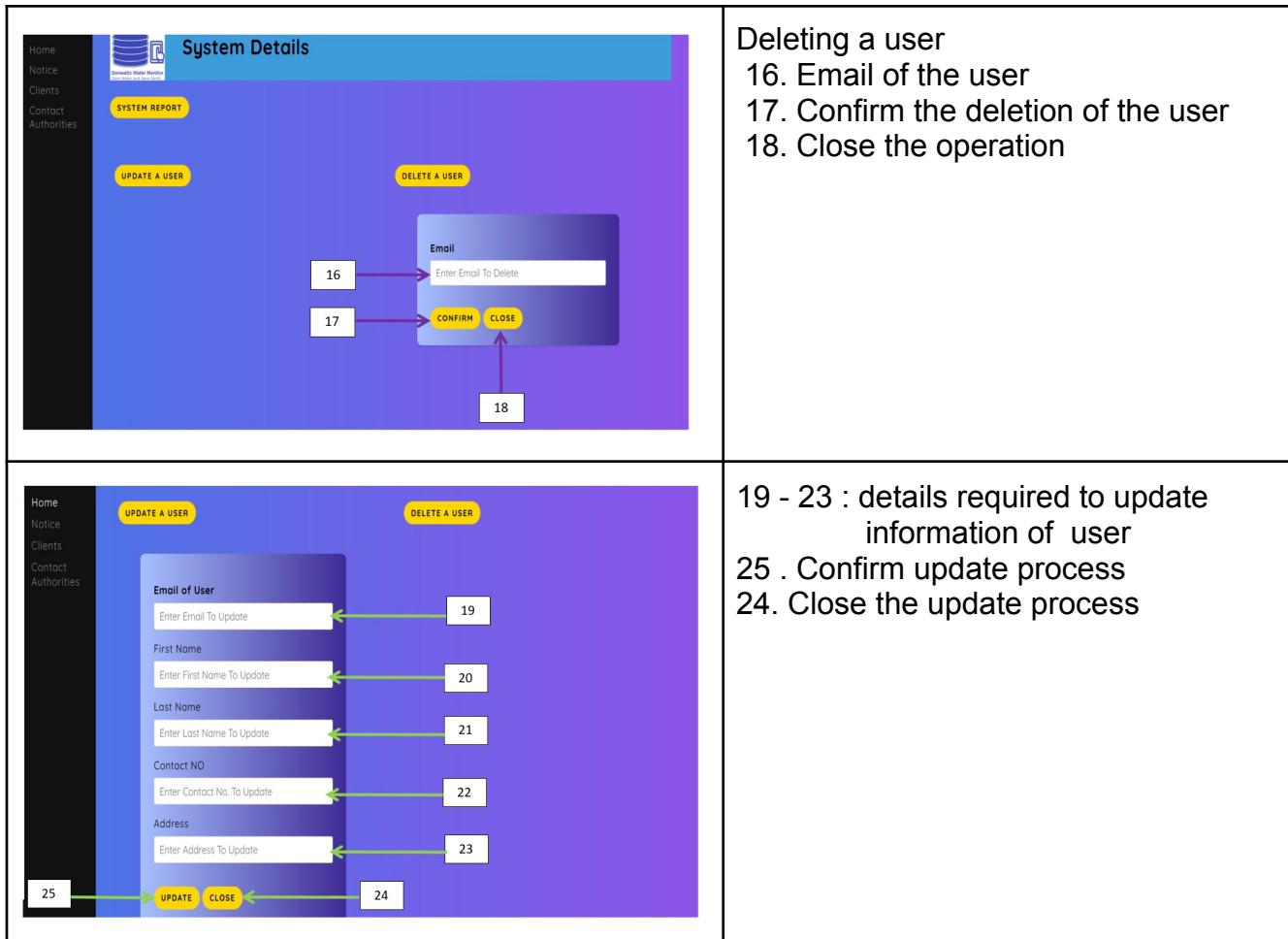
1. Select admin role from here. Super admin or admin
2. Information required for signup
3. Signup button



2.3.4 Admin

Admin have the clearance to do everything that a super admin can except update a user to admin.





2.4 Mobile Application

2.4.1 Overview

- Mobile application is developed using flutter sdk and dart language.
- Used mainly to access the information on the current status of the users tanks and daily usage.
- Notifications and alerts are sent to users when necessary.
- Email and password entered by the user is validated within the app before sending it to the server.

2.4.2 Notifications and Alerts

- There are two kinds of notifications. Notifications about water contaminations and notifications regarding the news such as watercuts.
- When logging in, a post request is sent to /login endpoint of the server and the receiving response includes the user details and within that the area of the user.
- When the user logged in to the app , it will initiate a firebase instance and subscribe to a firebase topic named after the area of the user. Ex: kandy
- Then system admins are able to send particular news for users within an area through firebase cloud messaging. Since firebase topic messaging guarantees to deliver the message at least once, users will get the alert.
- For alerts about contaminations , the mobile app updates its data from the server for every 12 seconds by sending a get request to /tanks endpoint with the included jwt cookie in headers. If either one of the tanks has an issue, the mobile application will generate a notification and change its status to *contamination detected!* Mode.

2.4.3 Security of Mobile Application

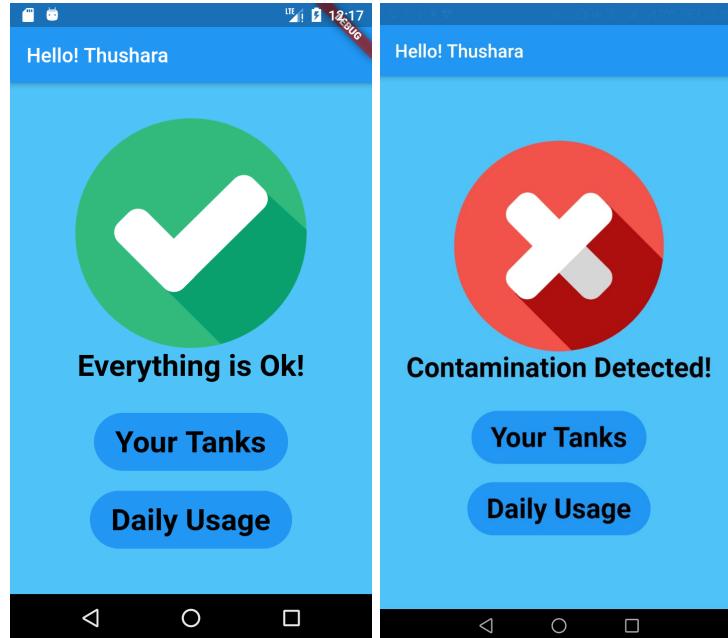
- For the mobile application , there is a login function and after that login function the user will keep the connection with the server through a jwt cookie.
- That cookie will be stored in a shared_preference instance. It is a common library to store non trivial data in mobile applications. Everytime the application makes a request to the server, the cookie will be retrieved from the instance to make the request and it will be updated after the response.
- Since the mobile app uses the shared preferences, even if we close the app , it will store the cookie until the user logs in again.

2.4.4 Configuration of the mobile application

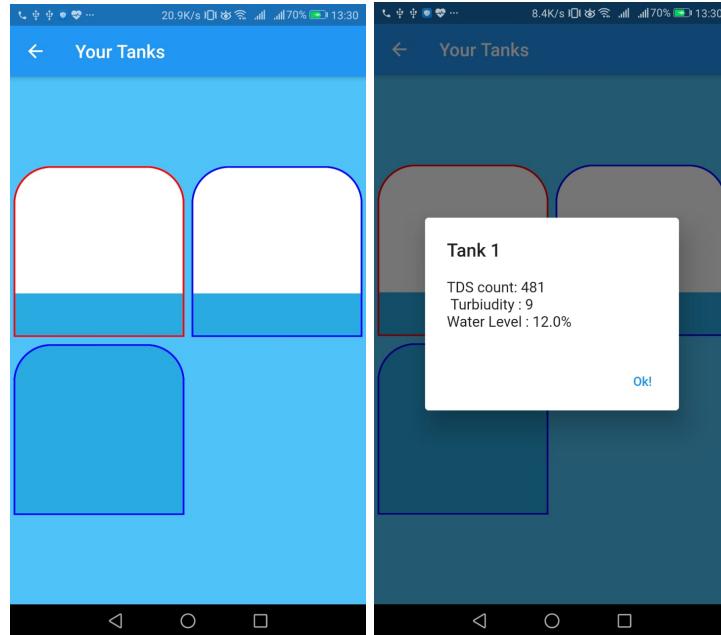
- Go to the playstore (android version is not developed yet) and download the aqua watcher application.
- After installing finishes , the user should login by giving the email and password which he provided earlier on his aquawatcher account.



- Then the user will enter into the home screen. Where the overall status of the tanks will be displayed. It will give users a rough idea about the status.



- Users can access the tank information using your tanks button. In the tanks screen, by touching a tank will pop up a message indicating the details of that particular tank. A series of thumbnails are used to represent various water levels and status of the tank. If there is a red outline around a tank , which means the tank is contaminated. The water level is indicated by the blue portion of the thumbnail.



- Users can access daily usage using the daily usage button in the home screen. In further developed versions the usage will be represented by graphs and will show daily and monthly usages.



2.4.5 Additional resource documentations for the mobile application

_____ This contains the documentations of the packages which are used in the mobile application development.

- Flutter documentation <https://flutter.dev/docs>
- Firebase Documentation
<https://firebase.google.com/docs/cloud-messaging>
- Local_Notifications.dart
https://pub.dev/packages/flutter_local_notifications
- http/http.dart <https://pub.dev/packages/http>
- shared_preferences https://pub.dev/packages/shared_preferences

2.5 Embedded Software

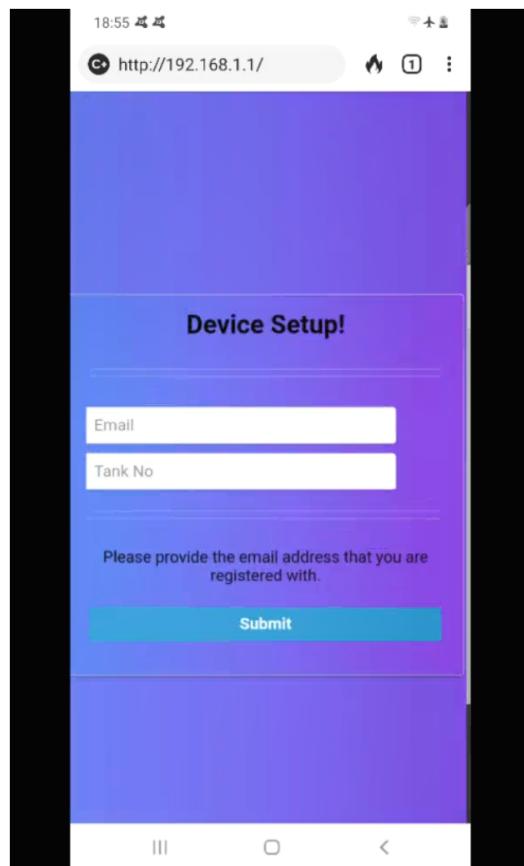
2.5.1 Overview

- 2 embedded modules are used in this product. They consist,
 1. ATMEGA328P – To get sensor readings
 2. ESP 12E (Node MCU) – For communication with the internet
- Programming is done using the Arduino language.
- Input validation/comparison done by ATMEGA328P chip.
- Device configuration details are stored in ESP 12E chip installed module.
- Communication between the chips is done by serial communication.
- Sensor readings taken by the ATMEGA328P chip are transmitted to ESP 12E chip.
- ESP 12E sends the data to the web server.

2.5.2 Functionality : ESP 12E chip (Node MCU)

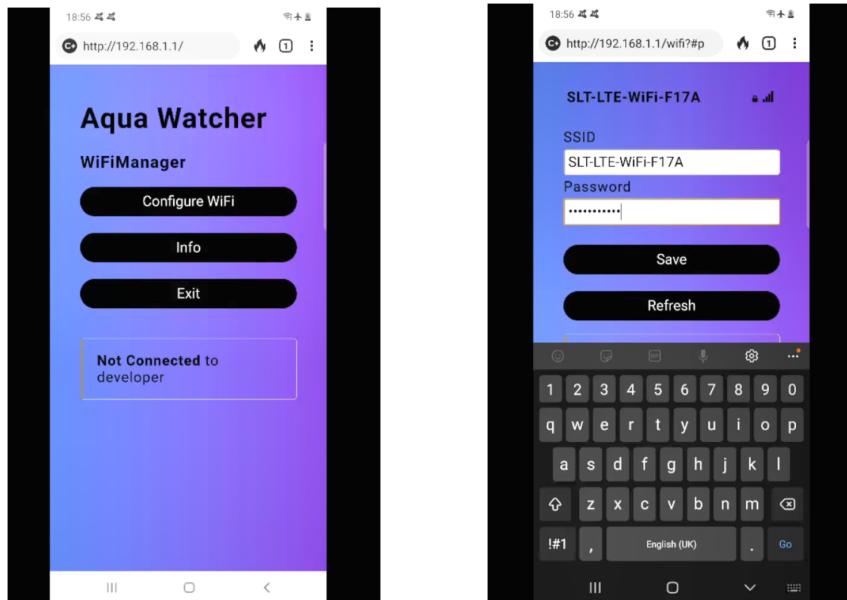
1. Device initialization process

- This module holds the device configuration data such as user name and device id. Upon powering up the device, if the configuration data is not initialized it will start a web server on port 80. You can connect to it via WiFi and go to network configuration. Else, you can go to URL: www.aquawatcher.com.
- “ESP8266HTTPClient”, “DNSServer” and “ESP8266WebServer” libraries were used. After entering User Email and device id, they will be stored in EEPROM and the server will be closed. To reset the values press the reset button.
- configDevice(),setValues(),serveReq() functions are handling this.



- Then the device will begin another web server on port 80 to get WiFi credentials of your router.
- Follow the same steps as before to enter the WiFi credentials. They will be stored in EEPROM and the server will be closed.
- “ESP8266HTTPClient”, “WiFiManager” and “ESP8266WebServer” libraries were used.

- “Configure WiFi” option allow you to go to the next window and store credentials.



- Required CSS styles and html documents are stored as a character array called “webpage”.
- JavaScript is used for send requests from web page to the device server
- Input data is sent in JSON format.
- XML Http Requests are used to make requests.
- Upon receiving 200 response code from the device, the UI will be closed.
- Upon receiving correct data from the web page, servers will be closed. Otherwise the server will send an error message and wait for correct data.
- DNS server will run on port 53.
- Via the DNS server, www.aqwsetup.com URL will be mapped to the web server of the device.
- You need to enter the IP address of your aws instance.
- After the initialization, that IP address will be combined with device id and a unique endpoint address will be generated to send data as put requests.

2. Connect to the web server

- After the above values are set, generated unique URL will be used to make a http connection to the server. “ESP8266HTTPClient” library will be used for this.
- Device will keep the connection as long as WiFi is available. Otherwise the device will try to connect to the WiFi network it was connected to before.

3. Serial communication

- Sensor readings from ATMEGA328P chips are received via serial communication pins.
- Since the serial monitor is used for print values, extra serial pins are required. "SoftwareSerial" library is used to define extra transmit and receive pins.
- When receiving data, the device stops all the other operations until the end of receiving. This is done by checking the availability of serial communication pins.
- Incoming messages will be read until the new line character.

4. Send Put requests

- Data from serial communication is already formatted in JSON.
- Data will be stored in a request body.
- Request headers will be set to "Content-Type" as "application/json".
- Then a put request will be made to the established connection.
- Upon receiving the status code it is checked whether the request is successful or not by status code.
- This is done in a loop.

5. Libraries used

Library	Purpose
SoftwareSerial	To serial communication with other module
ESP8266HTTPClient	To establish a connection with web server
ESP8266WiFi	Configure WiFi connection
DNSServer	To map www.aqwsetup.com to local web server
ESP8266WebServer	To setup a web server locally
WiFiManager	To store WiFi information

6. Functions used

Function	Purpose
setValues	Configuration of device id
serveReq	Serve local server requests
configDevice	Setup DNS server
connect	Store WiFi credentials
setup	Initialization of device
loop	Communication with web server in a loop

2.5.3 Functionality : ATMEGA 328P chip

1. Device Initialization

- After the device is installed and turned on, it will start making readings.
- It will take the depth of the tank and dimensions will be decided.
- Pin modes will be set on initialization.
- Flow Meter will make interrupts on rising edges. When 10L of water is flown, it will make an interrupt.

Analog In	External Interrupt	Analog Out
TDS sensor	Flow Meter	Speaker

Turbidity sensor		
UltraSonic sensor		

2. Device Operation

- When the water is coming in, quality readings will be made and sent to the server as well.
- There is a delay of 75ms between each reading.
- For each value, 10 readings will be taken and the average will be calculated for accuracy.
- If there is a water quality or level issue, an alarm will go off.
- If turbidity > 5 or TDS value > 200 or water level is over 90% or under 10% there will be warnings.
- During interrupts made by flow meters, water usage will be increased by 10L.
- Each 15 mins, usage data will be sent out.
- Operation occurs inside the loop function. There is a 10s delay for each iteration.

3. Serial Communication

- After the readings are taken, they are formatted to JSON.
- This is done by string concatenation.
- Then transmitted via default rx pin.

4. Functions Used

Function	Purpose
warn	Sound alarm
addL	Increase usage on interrupts

getTDS	Get TDS value
getTURB	Get turbidity
getLevel	Get water level
setup	Device initialization
loop	Continuous operation

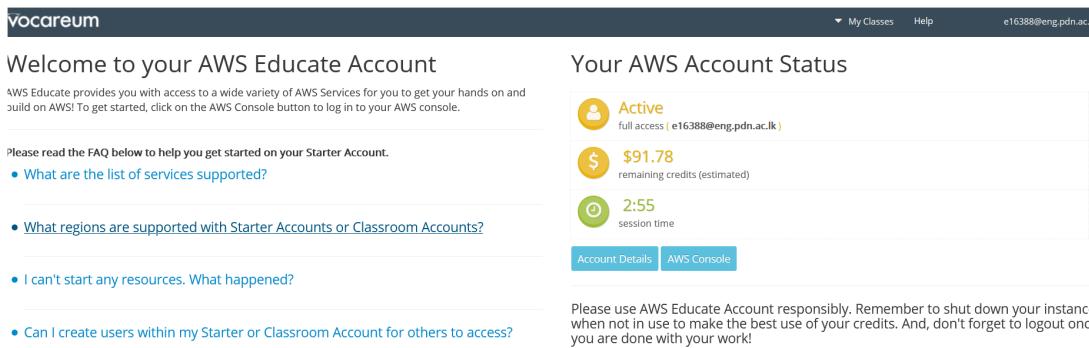
2.6 Amazon Web Services cloud deployment

2.6.1 Overview

- Web server is deployed on aws ec2 instance.
- Sever can be accessed via port 80.
- Owners can ssh to the instance via port 22.
- Client requests are handled by the server in ec2 instance.
- Route 53 service is used to add a domain name for our deployment.

2.6.2 Steps

1. Login to your AWS account.



The screenshot shows the AWS Educate Account dashboard. At the top, it says "Welcome to your AWS Educate Account". Below that, there's a section titled "Your AWS Account Status" with three metrics: "Active" (full access), "\$91.78" (remaining credits), and "2:55" (session time). At the bottom, there's a link to "Account Details" and another to "AWS Console". To the left, there's a sidebar with links like "What are the list of services supported?", "What regions are supported with Starter Accounts or Classroom Accounts?", "I can't start any resources. What happened?", and "Can I create users within my Starter or Classroom Account for others to access?".

2. Select AWS Console.

AWS Management Console

The screenshot shows the AWS Management Console homepage. On the left, there's a sidebar titled "AWS services" with options like "Recently visited services" and "All services". Below it, a section titled "Build a solution" offers three options: "Launch a virtual machine" (With EC2, 2-3 minutes), "Build a web app" (With Elastic Beanstalk, 6 minutes), and "Build using virtual servers" (With Lightsail, 1-2 minutes). To the right, there's a "Stay connected to your AWS resources on-the-go" section featuring the AWS Console Mobile App, and an "Explore AWS" section with links to "AWS Application Migration Service" and "S3 Object Lambda".

3. Select All services -> EC2.

This screenshot shows the "All services" dropdown expanded. Under the "Compute" category, "EC2" is highlighted with a yellow box. Other options include Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application Repository, AWS Outposts, EC2 Image Builder, and AWS App Runner. The rest of the page shows a grid of services under categories like Management & Governance, Security, Identity, & Compliance, and others.

4. Select Launch Instance -> Launch Instance.

This screenshot shows the "Launch instance" page in the AWS EC2 console. It includes sections for "Launch instance", "Scheduled events", and "Migrate a server". The "Launch instance" section has a prominent orange "Launch instance" button. The "Service health" section shows "This service is operating normally". The "Zones" table lists availability zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, and us-east-1f, each associated with a zone ID.

5. Choose an Amazon Machine Image (AMI). Ideally select a Linux based Machine Image.

Step 1: Choose an Amazon Machine Image (AMI)

6. Choose an Instance Type

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance	IPv6 Support
i2	i2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
i2	i2.micro	1	1	EBS only	-	Low to Moderate	Yes
i2	i2.small	1	2	EBS only	-	Low to Moderate	Yes
i2	i2.medium	2	4	EBS only	-	Low to Moderate	Yes
i2	i2.large	2	8	EBS only	-	Low to Moderate	Yes
i2	i2.xlarge	4	16	EBS only	-	Moderate	Yes
i2	i2.2xlarge	8	32	EBS only	-	Moderate	Yes
i3	i3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel | Previous | **Review and Launch** | Next: Configure Instance Details

7. Configure Instance Details

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of it.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot Instances	
Network	vpc-7d6aa300 (default)	<input type="button"/> Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	<input type="button"/> Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	Open	
Domain join directory	No directory	<input type="button"/> Create new directory
IAM role	None	<input type="button"/> Create new IAM role
Shutdown behavior	Stop	
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	

8. Add Storage as your requirement

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0a52a8f51496c3782	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

9. Configure Security Group.

1. Create a new Security Group.
2. Click Add Rule -> add new Custom TCP rule on port 80 and source as given.
3. Now unauthorized access is restricted according to these rules.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name:
Description: launch-wizard-1 created 2021-05-28T11:18:25.040+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP F...	TCP	80	Anywhere 0.0.0.0/0	e.g. SSH for Admin Desktop

[Add Rule](#)

10. Review Instance Launch. Review your selections and launch the instance.

Step 7: Review Instance Launch

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
i2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups

Security group name	Description
TestSecGroup	launch-wizard-1 created 2021-05-28T11:18:25.040+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	

Instance Details

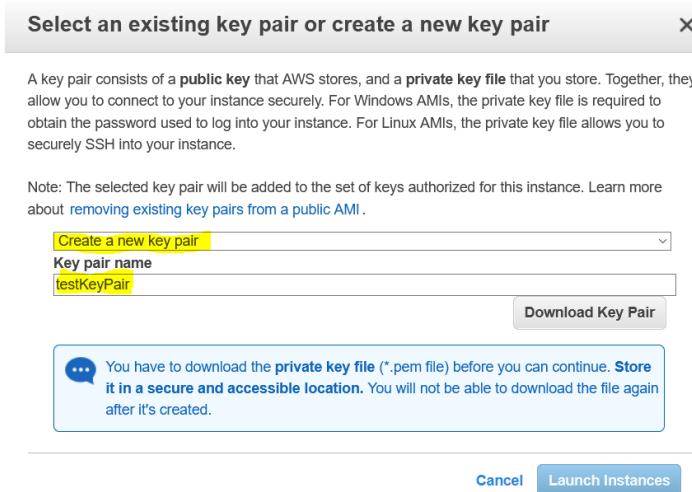
Storage

Tags

[Edit instance type](#) [Edit security groups](#) [Edit instance details](#) [Edit storage](#) [Edit tags](#)

[Cancel](#) [Previous](#) [Launch](#)

11. Create a new Key Pair to ssh into your instance. Then download it. Never lose this file.



12. Now your instance is launched. To view your instance. Click on View Instance.

Launch Status

Your instances are now launching
The following instance launches have been initiated: i-08d50b4409548a9da [View launch log](#)

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click [View Instances](#) to monitor your instances' status. Once your instances are in the **running** state, you can [connect](#) to them from the Instances screen. [Find out](#) how to connect to your instances.

Here are some helpful resources to get you started

- [How to connect to your Linux instance](#)
- [Amazon EC2: User Guide](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View instances](#)

Feedback English (US) ▾ © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

13. Select your instance

aws Services ▾ [Alt+S] [\[Alt+S\]](#) [vocstartsoft/user840816=e16388@eng.pdn](#)

New EC2 Experience [Learn more](#)

EC2 Dashboard [New](#)

Events

Tags

Limits

Instances [New](#)

Instances [New](#)

Instance Types

Launch Templates

Spot Requests

Savings Plans

Instances (3) [Info](#)

Name	Instance ID	Instance state	Instance type
-	i-08d50b4409548a9da	Running	t2.micro
-	i-0614d9df5ff174224	Stopped	t2.micro
-	i-02cd824340b692855	Stopped	t2.micro

14. Select Connect.

EC2 > Instances > i-08d50b4409548a9da

Instance summary for i-08d50b4409548a9da [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-08d50b4409548a9da	100.25.45.3 open address	172.31.48.151
Instance state	Public IPv4 DNS	Private IPv4 DNS
Running	ec2-100-25-45-3.compute-1.amazonaws.com open address	ip-172-31-48-151.ec2.internal
Instance type	Elastic IP addresses	VPC ID
t2.micro	-	vpc-7d6aa300
AWS Compute Optimizer finding	IAM Role	Subnet ID
User: arn:aws:sts::20560255782:assumed-role/vocstartsoft/user840816=e16388@eng.pdn.ac.lk is	-	subnet-571bdb66

15. Copy the highlighted command -> open a terminal in the folder with key pair->run the command->Type “yes”.

Connect to instance [Info](#)

Connect to your instance i-08d50b4409548a9da using any of these options

EC2 Instance Connect Session Manager **SSH client** EC2 Serial Console

Instance ID: i-08d50b4409548a9da

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is testKeyPair.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 testKeyPair.pem`
4. Connect to your instance using its Public DNS:
 `ec2-100-25-45-3.compute-1.amazonaws.com`

Example:
 `ssh -i "testKeyPair.pem" ubuntu@ec2-100-25-45-3.compute-1.amazonaws.com`

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

```
c:\Users\Thushara\Desktop>aws>ssh -i "group13.pem" ubuntu@ec2-3-90-163-133.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-163-133.compute-1.amazonaws.com (3.90.163.133)' can't be established.
EDSA key fingerprint is SHA256:MaIn2zukeICECx/vJBrV8H3BrijfcIauuafR0RAleAs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-90-163-133.compute-1.amazonaws.com,3.90.163.133' (EDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1048-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

72 packages can be updated.
17 updates are security updates.

Last login: Mon May 24 14:11:49 2021 from 212.104.237.48
ubuntu@ip-172-31-22-72:~$
```

16. Run following commands.

- a. sudo apt update
- b. sudo apt install git
- c. Git clone git clone -b master --single-branch
<https://github.com/cepdnaclk/e16-3yp-water-quality-monitoring-and-usage-monitoring-system.git>
- d. sudo apt install nginx
- e. sudo apt install node
- f. sudo apt install npm
- g. npm install pm2
- h. Cd e16-3yp-water-quality-monitoring-and-usage-monitoring-system
- i. npm install

17. Nginx configuration and server startup. Follow these steps.

- a. Run command cd..
- b. cd /etc/nginx/conf
- c. Replace the file nginx.conf file with the nginx.conf file found on our repo.
- d. Move back to
e16-3yp-water-quality-monitoring-and-usage-monitoring-system directory
- e. Run command sudo service nginx restart
- f. Run command sudo pm2 start app1.js
- g. Run command sudo pm2 start app2.js
- h. Pm2 table will look like this now.

id	name	namespace	version	mode	pid	uptime	�	status	cpu	mem	user	watching
1	app1	default	1.0.0	fork	14891	66s	0	online	0%	64.9mb	root	disabled
2	app2	default	1.0.0	fork	15626	59s	0	online	0%	65.7mb	root	disabled

18. Setting a domain name.

Run command cd..

- a. Go to www.freenom.com
- b. Check availability for your domain name.
- c. If it is available, select-> add it to the cart->purchase.
- d. Provide billing information.



Sorry, testdomain.ml is not available.

testdomain .tk	<input checked="" type="checkbox"/> Not available
testdomain .ga	<input checked="" type="checkbox"/> Not available
testdomain .cf	<input checked="" type="checkbox"/> Not available
testdomain .gq	<input checked="" type="checkbox"/> Not available

Cost Price Domains cannot be found cheaper elsewhere!			
MAX. CHARACTER	31	<input type="checkbox"/> COUNTRIES/LOCATIONS ONLY	
testdomain .com	COST PRICE	USD 8. ³⁸	<input type="button" value="Select"/>
testdomain .net	COST PRICE	USD 6. ⁷¹	<input type="button" value="Select"/>

- e. In the below example I have selected www.aquawatche.ml
- f. From AWS console, select services->Route 53

The screenshot shows the AWS Route 53 Dashboard. It includes sections for DNS management (1 Hosted zone), Traffic management (Create policy), and Availability monitoring (Create health check). Below this, there is a 'Register domain' section where users can enter a domain name and check its availability.

- g. Select Hosted Zone->Create hosted zone

The screenshot shows the AWS Route 53 Hosted zones page. It displays a table of existing hosted zones, including one named 'aquawatcher.ml'. A prominent orange 'Create hosted zone' button is visible at the top right of the page.

- h. Set purchased domain name and set type as a Public Hosted Zone and click Create Hosted Zone.

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name [Info](#)
This is the name of the domain that you want to route traffic for.

Valid characters: a-z, 0-9, !*# \$ % & '() * +, - / : ; < = > ? @ [\] ^ _ ` { } . ~

Description - optional [Info](#)
This value lets you distinguish hosted zones that have the same name.

The description can have up to 256 characters. 0/256

Type [Info](#)
The type indicates whether you want to route traffic on the internet or in an Amazon VPC.
 Public hosted zone.
A public hosted zone determines how traffic is routed on the internet.
 Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

Tags [Info](#)
Apply tags to hosted zones to help organize and identify them.

No tags associated with the resource.

You can add up to 50 more tags.

[Cancel](#) [Create hosted zone](#)

i. Select your hosted zone.

Hosted zones (1)									
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.									
<input type="button" value="C"/> View details Edit Delete Create hosted zone							< 1 > 		
Domain name	Type	Created by	Record count	Description	Hosted zone ID				
aquawatcher.ml	Public	Route 53	4	-	Z0502777KC3U9W3GEUBJ				

j. Select your hosted zone.

- k. Click on create record and create 2 records for [aquawatcher.ml](#) and [www.aquawatcher.ml](#). Value/Route traffic should be the public IP address of your aws instance. Also copy the DNS server list.

	Record name	Type	Routin...	Differ...	Value/Route traffic to
<input type="checkbox"/>	www.aquawatcher.ml	A	Simple	-	52.91.3.187
<input type="checkbox"/>	aquawatcher.ml	A	Simple	-	52.91.3.187
<input type="checkbox"/>	aquawatcher.ml	SOA	Simple	-	ns-1813.awsdns-34.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
<input type="checkbox"/>	aquawatcher.ml	NS	Simple	-	ns-1813.awsdns-34.co.uk. ns-741.awsdns-28.net. ns-1227.awsdns-25.org. ns-398.awsdns-49.com.

- l. In freenom site, go to services-> My Domains->[aquawatcher.ml](#)->DNS MANAGEMENT->Edit Nameservers. Paste the name servers in here

The screenshot shows the 'Nameservers' section of the Freenom DNS management interface. At the top, there are tabs for 'Information', 'Upgrade', 'Management Tools', and 'Manage Freenom DNS'. Below the tabs, it says 'Managing aquawatcher.ml'. Under the 'Nameservers' heading, there's a note: 'You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.' There are two radio button options: 'Use default nameservers (Freenom Nameservers)' (unchecked) and 'Use custom nameservers (enter below)' (checked). Below these options are five input fields labeled 'Nameserver 1' through 'Nameserver 5', each containing a specific IP address. At the bottom of the section is a blue 'Change Nameservers' button.

m. Wait around 10 mins and your domain name will be registered.

19. Now your web server is deployed.

20. You can enter the website using the domain name you purchased.

2.6.3 Scalability and Reliability

Check section **2.1.3 chapter 10.**

2.6.4 Security

- In step 9, security groups have been added. The connections to the instance can be made by only allowed ports
- The Nginx server monitors the communication. In the config file, admin access is restricted to internal users. Admin access is blocked from outside. Outsiders receive 403 status on trying to admin access.
- Ufw policies can be set to allow only required connections. Others will be denied.

Steps,

Run following commands

1. sudo apt install ufw
2. sudo ufw allow from "your ip"/24 to any port 22
3. sudo ufw allow 80/tcp
4. sudo ufw allow 'Nginx HTTP'

3.0 Hardware Development

3.1 Required Components

Item	Quantity
Turbidity Sensor	1
TDS Sensor	1
HC-SR04 Ultrasonic Sensor Module	1
Water flow meter	1
NodeMcu V3 Lua WIFI Module	1
Atmega328P chip	1
SP0008 speaker	1

Additional Components : DC-DC step down Buck converter, Li Battery Charger, push switches, solar panel

3.1.1 Sensors

- Water quality sensors**

Turbidity Sensor - measure the color of the water range: 1-200 NTU Max relative error: 3.09%

TDS Sensors - measure the how much substances are dissolved in the water range: 0 to 50,000 mg/L Accuracy: (+-)0.5%

- Depth sensor**

Ultrasonic Sensor - Reliable hardware for measuring short distances with a sufficient accuracy

3.1.2 Tasks of other components

NodeMcu: Provides the connectivity between the device and server. low cost which reduce the price of the product and low energy consumption which is essential to a long battery life of the device

Atmega328P chip: Getting the required measured readings from the sensors and send to the server via NodeMcu wireless connection

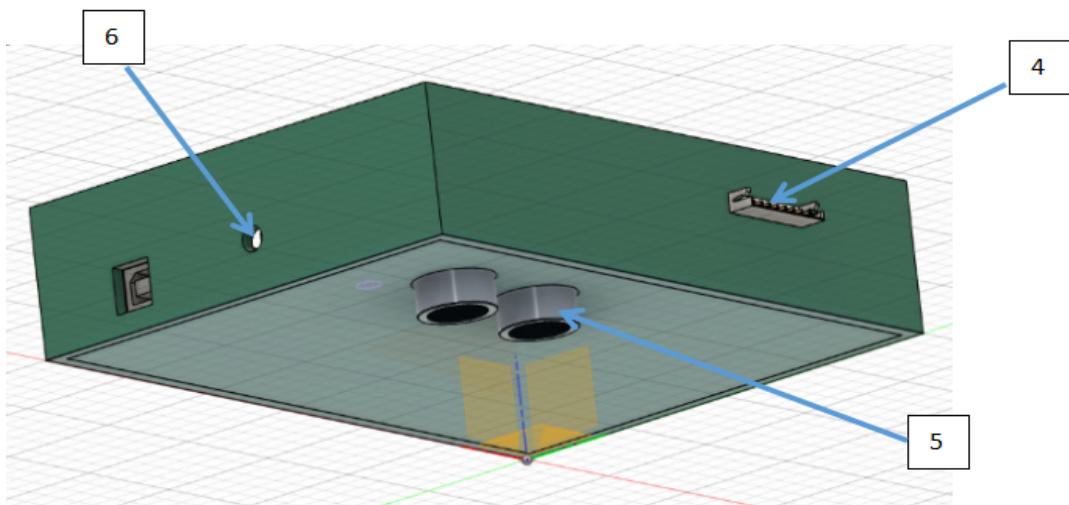
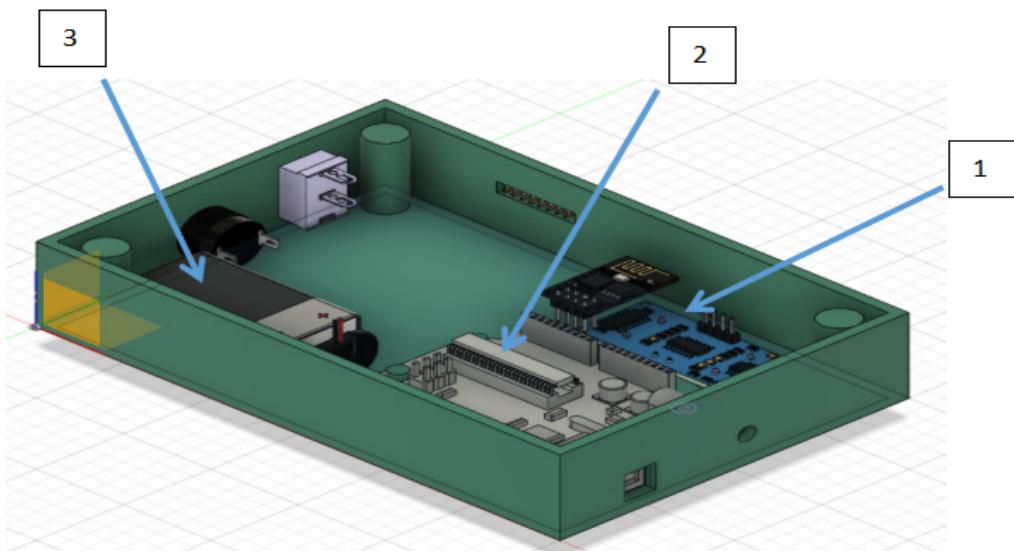
Speakers: Use to notify the user when contamination detected even during a power failure

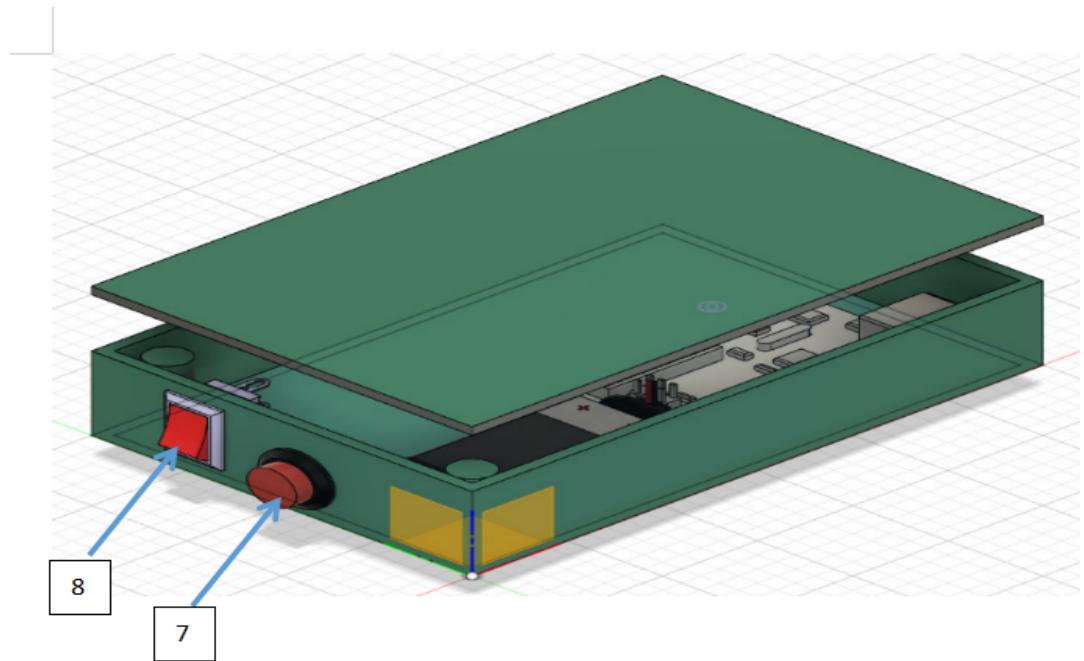
Solar panel: Charges the battery which powers the system

3.2 Designs and Diagrams

3.2.1 Designs for 3D printing

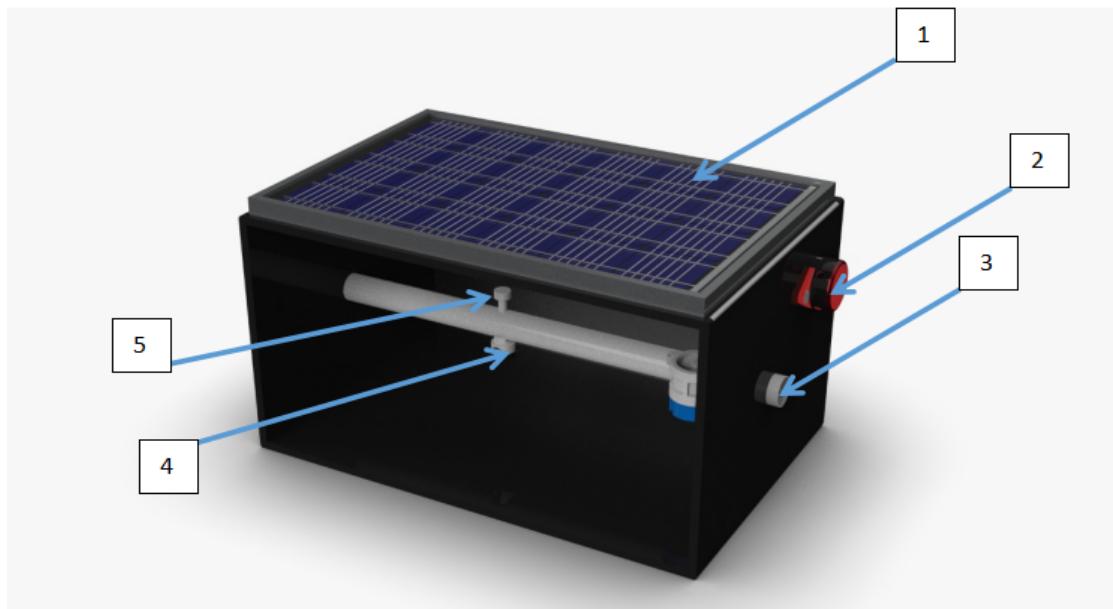
- Water Level Monitoring Module





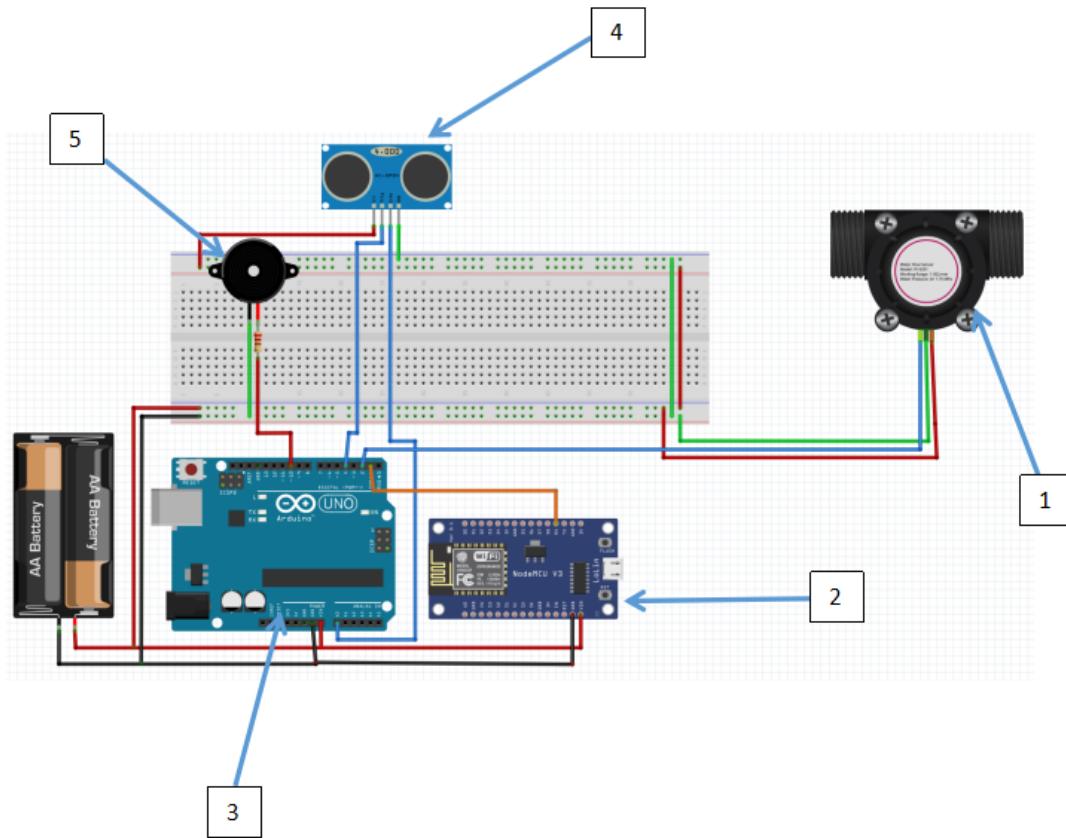
- 1 - NodeMcu
- 2 - Atmega328p Chip
- 3 - Rechargeable battery(Lithium Iron)
- 4 - Port for uploading code to atmega chip
- 5 - Ultrasonic sensor
- 6 - Inlet for wires
- 7 - Reset Button
- 8 - Power Button

- **Water Quality Monitoring Module**



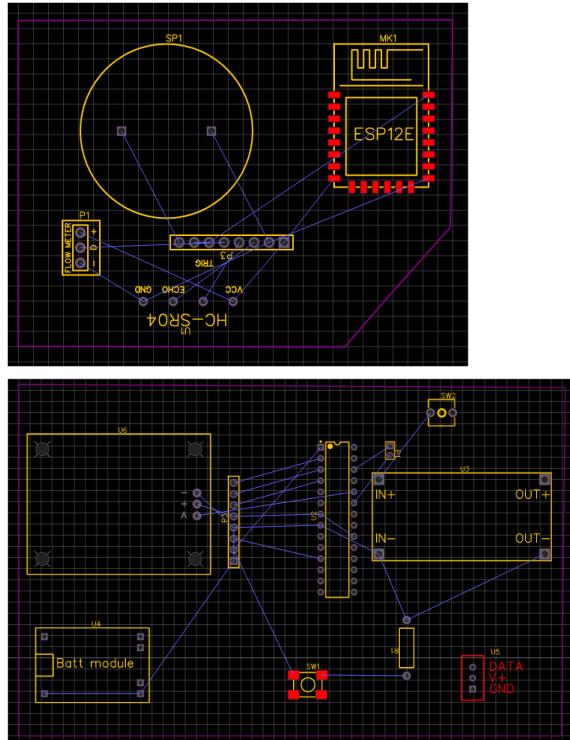
- 1 - Solar Panel
- 2 - Alarm
- 3 - Connect with pipe
- 4 - Turbidity sensor
- 5 - TDS sensor

3.2.2 Circuit Diagrams



3.2.3 PCB & Schematic Designs

- PCB Designs**



To reduce the complexity, circuit is divided to 2 parts for fabrication.
Has 2 layers.

UltraSonic Sensor - U1

Water Level Monitoring.ATMEGA328P chip - U2

Microcontroller chip.DC-DC step down Buck converter - U3

Voltage Controller.Li Battery Charger - U4

Battery Charging module.Turbidity Sensor - U5

Turbidity Measurement of Water.TDS Sensor - U6

TDS Measurement of Water.Flow Meter - P1

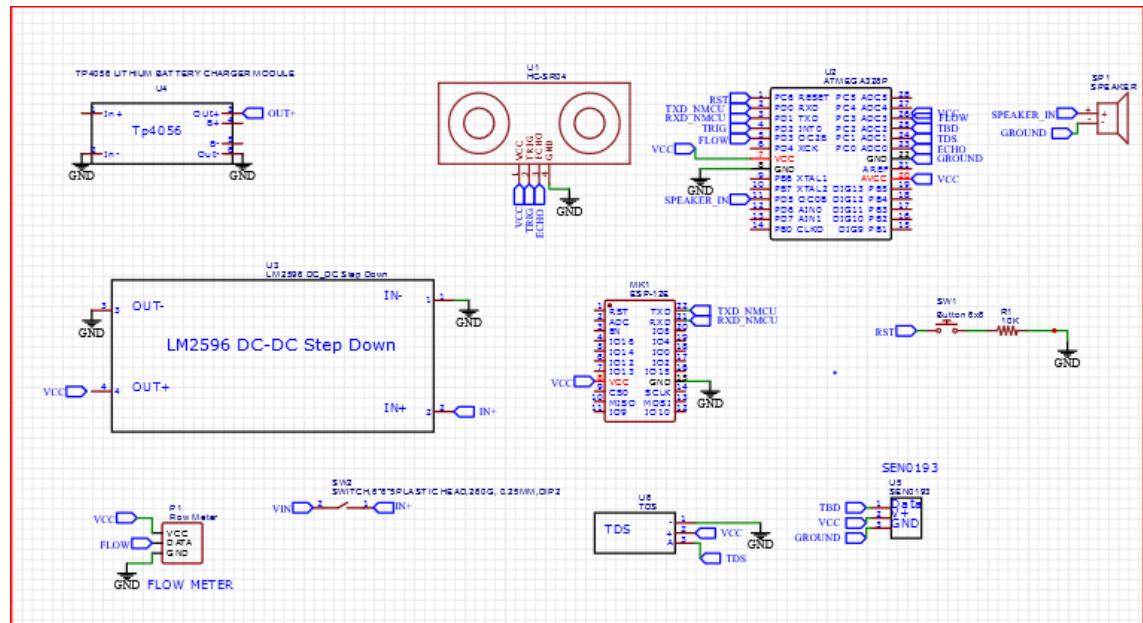
Water Usage Monitor.NODEMCU - MK1

Internet Connection Module.Speaker - SP1

Warning System.Reset Button - SW1

Power Button - SW2

- **Schematic Design**



4.0 Security, Reliability and Scalability

4.1 Security

- **Hardware Aspects**

- ❖ Water tight design
- ❖ Non toxic material for fabrication
- ❖ Data validation

When consider this due to the water tight design life time of the components will be very high. Since non toxic materials are used, there will not be any kind of problems with the consumption of the water.

Metallic parts which contact the water should be a less active but high electric conductive metal such as Copper. Therefore corrosion is avoided.

● **Software Aspects**

- ❖ Email verification on signup
- ❖ Hashing passwords using blowfish cipher
- ❖ Authorization middleware
- ❖ Setting the request rate limits

According to above taken security measures, DOS attacks are prevented and also risk of penetrating attacks are reduced. The reliability of the system is improved by using the above security measures.
(Refer Security sections in Software Deployment)

4.2 Reliability

- Solar powered rechargeable battery
- Ability to alert the user by using a buzzer even with no wifi
- Can work under extreme weather conditions
- Accurate sensor readings

According to above methods hardware reliability is increased. User doesn't want worry about power failures because user get notified by using buzzer and also during extreme weather conditions user get correct notifications. Since the sensor readings are accurate, users never gets false alarms.

4.3 Scalability

- Single app can monitor multiple tanks
- Easy to add new devices
- Can set up on any type of tank

Using our mobile app it easy to monitor multiple tanks just only clicking the

icons. And also by using the web site new devices can be added easily. Even there are different kind of water tanks in use, our system can implement on any kind of a water tank which makes any customer affordable for our service.

5.0 Resources

- All the designs and codes can be found on our github repository.
 - <https://github.com/cepdnaclk/e16-3yp-water-quality-monitoring-and-usage-monitoring-system>