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

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

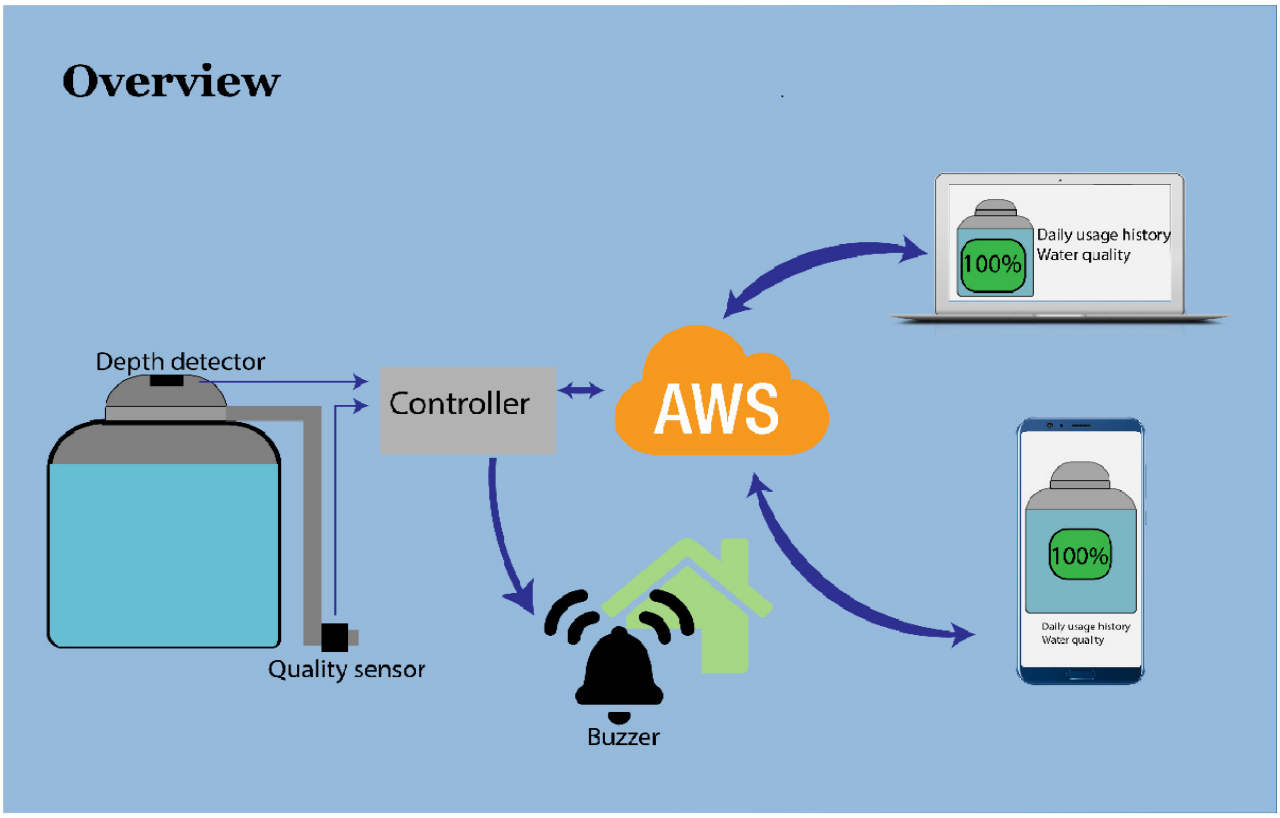


Figure 01

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

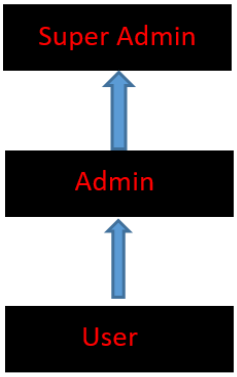
2.1.1 Overview

- 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.



Hierachy 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.

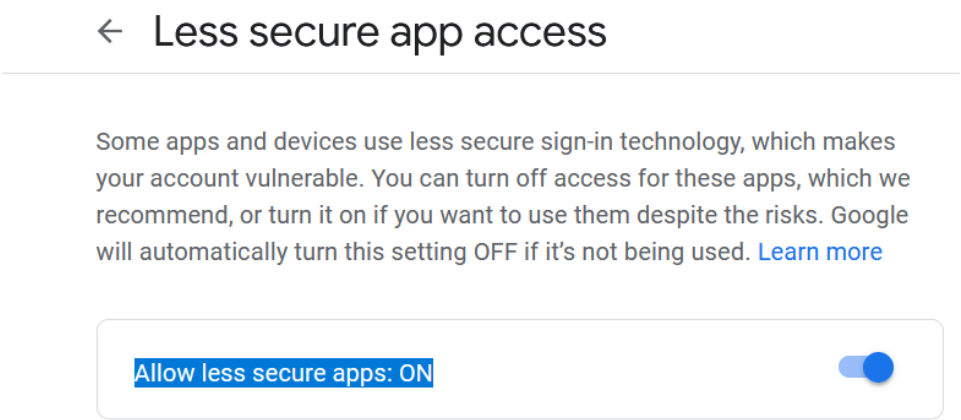
- Setting burst limits using “ddos” library and set “rate limit headers” to prevent DDoS attacks. This blocks the IP addresses which send a burst of requests at the same time and sends 429 responses. Those IP addresses will be blocked for 1 hour.
- Hiding Server information to prevent specially designed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



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

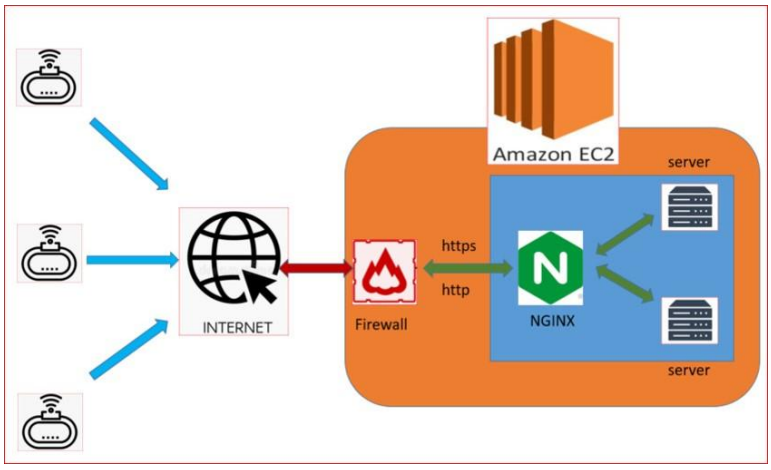
9. External libraries

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.
validatior	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.



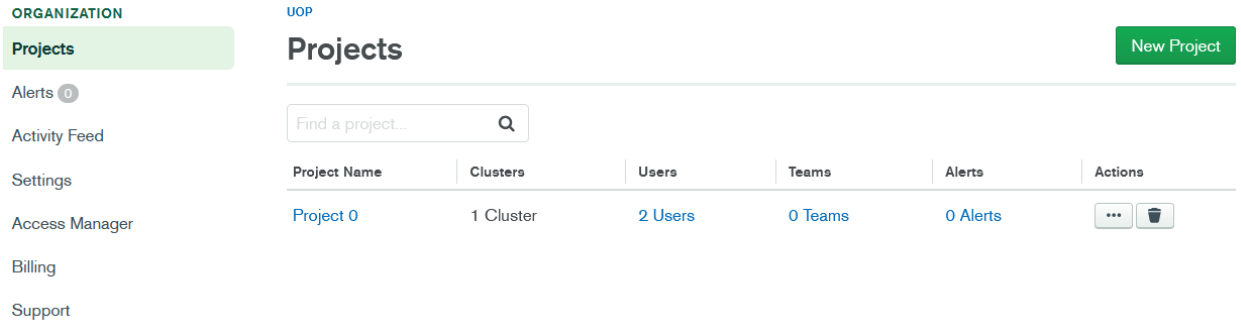
2.2 Database Deployment

2.2.1 Overview

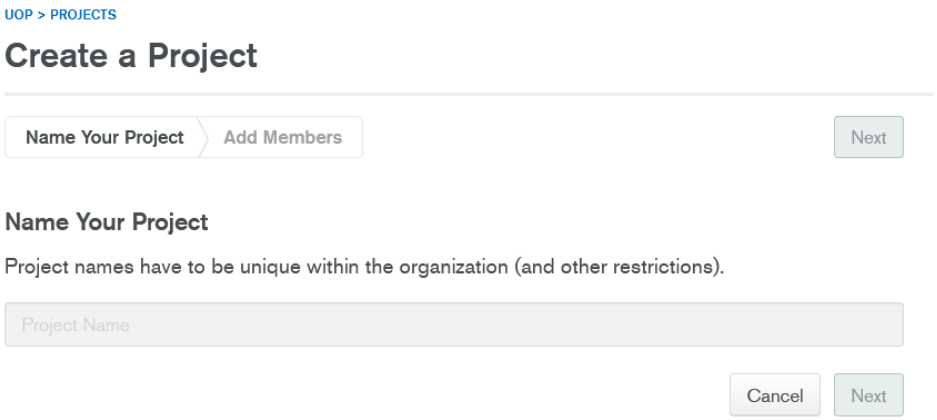
- Database deployment is done using MongoDB cloud-based database.
- Mongoose external library was used to deploy with Node.js server.
- Data validation and error checking methods are included.
- Outside access to the database is limited.
- Data backup is enabled.

2.2.2 Steps

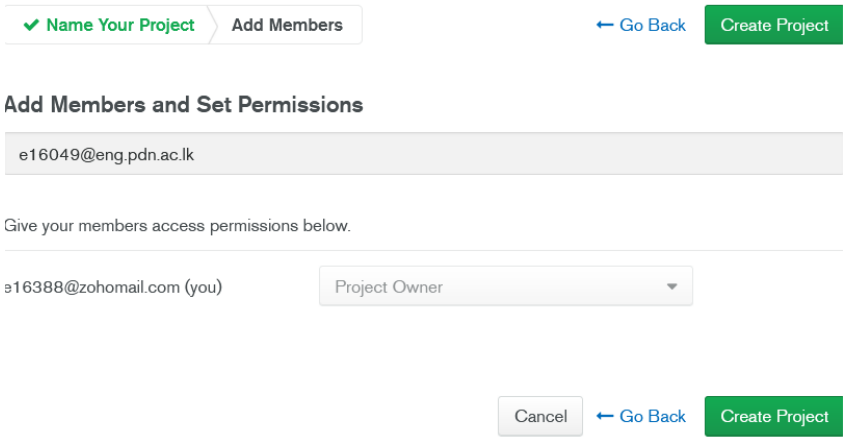
1. **Create a MongoDB account by using a unique email.**
2. **Login to your account**
3. **Select Projects from the sidebar.**



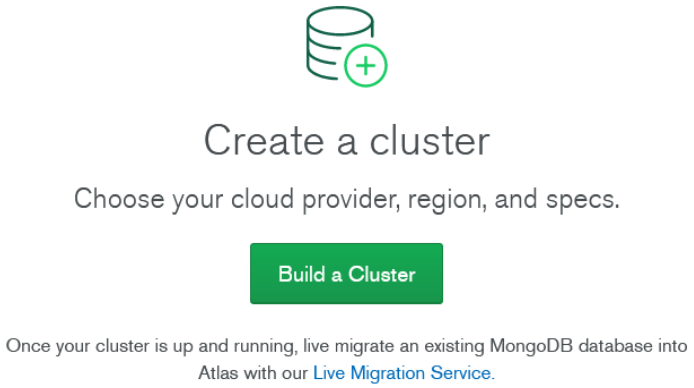
4. **Select the New Project option and Name your project.**



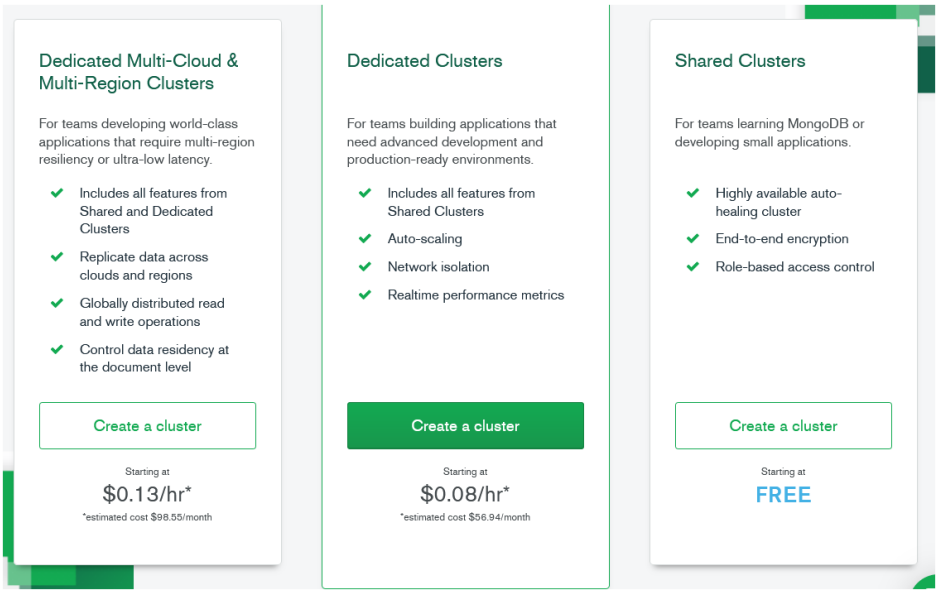
5. **Add Members and Set Permissions. You can give access to your colleagues here.**



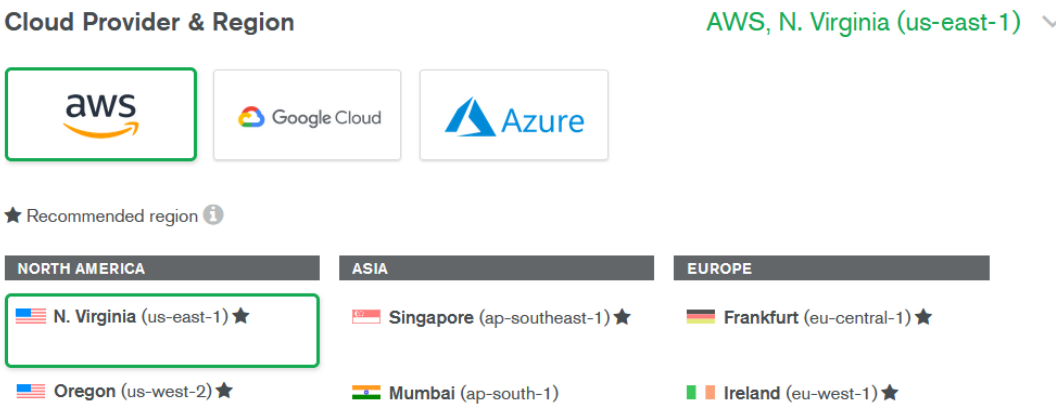
6. **Then click Create Project.**
7. **Then click Build a Cluster.**



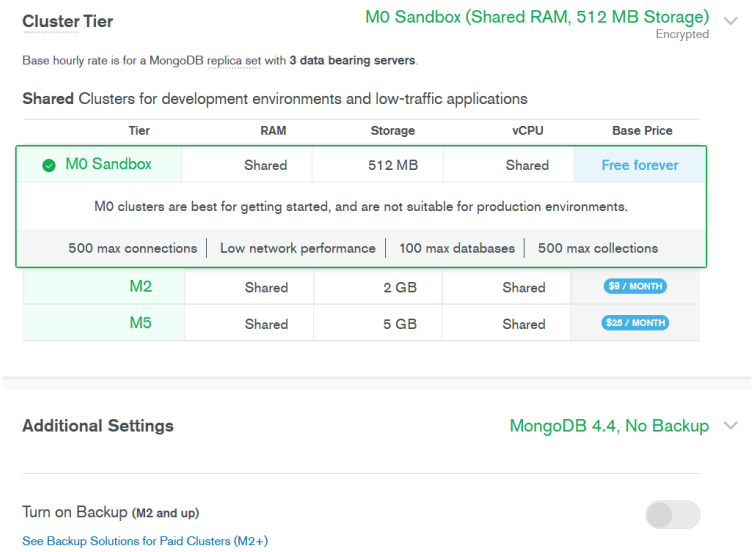
8. **Select your package.**



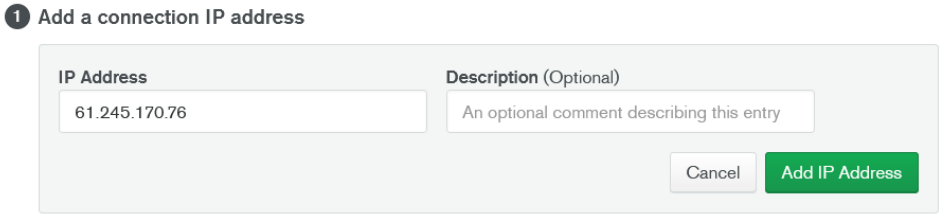
9. **Select provide and region. Here you should select aws as the provider.**

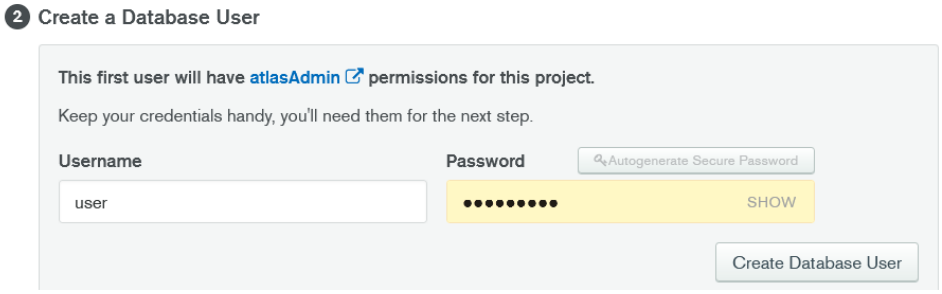


10. **Select RAM, Storage, and CPU preferences.**
11. **Turn on Backup if you are using a paid package.**

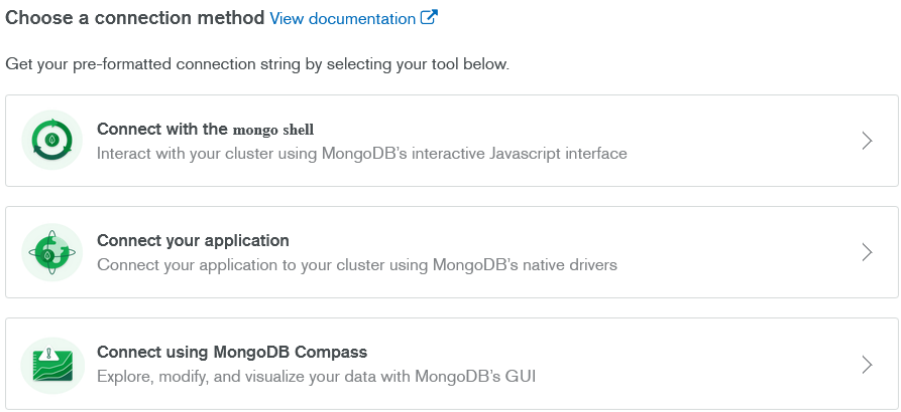


12. **Then click on Create Cluster and wait until it is created.**
13. **Then click on Connect.**
14. **Then add the IP addresses you want to have connections with this cluster. You should add your IP address here and. Then click Add IP Address.**
15. **Also add a user who should have permission to access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

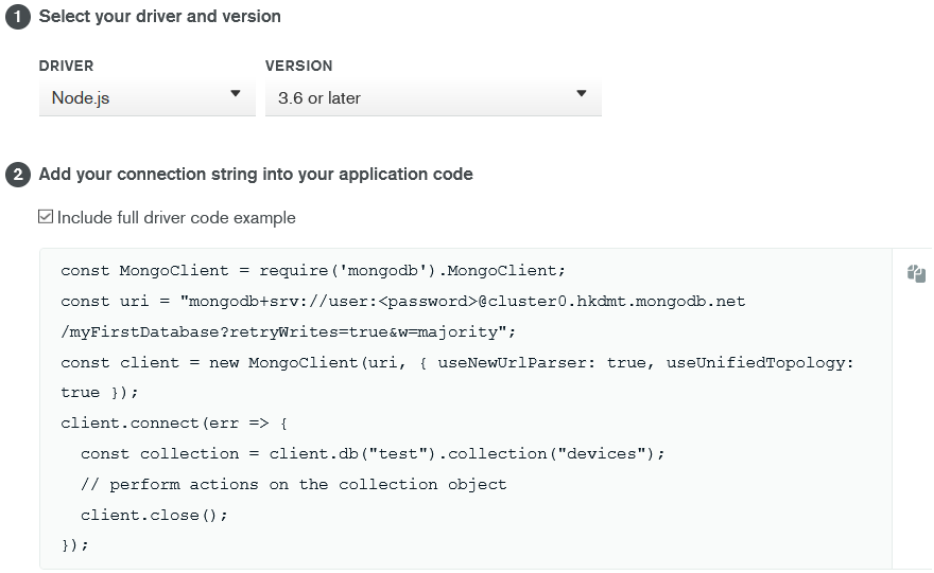




16. **Now any unauthorized access will be blocked.**



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



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.

cutomers :

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

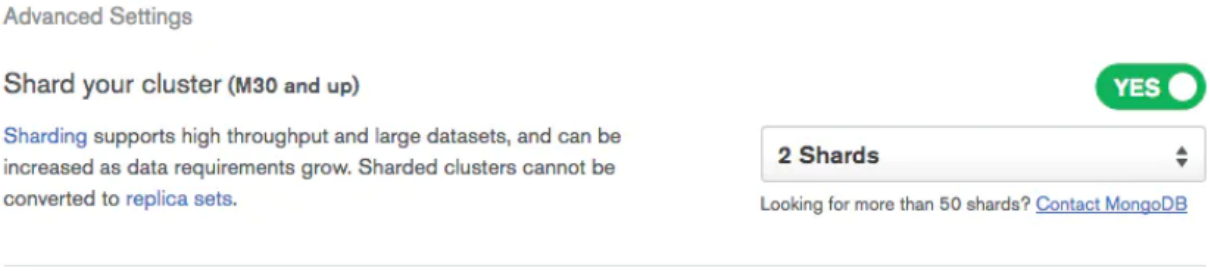
- Only collections “customer” and “tanks” store data from the outside. During customer registration, “customer” collection stores the customer inputted data. Also, “tanks” collection stores the data from the devices. Therefore when it comes to these 2 collections, inputs are thoroughly validated. If 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.



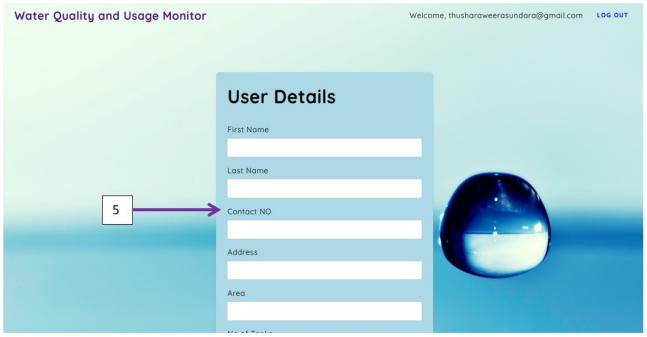
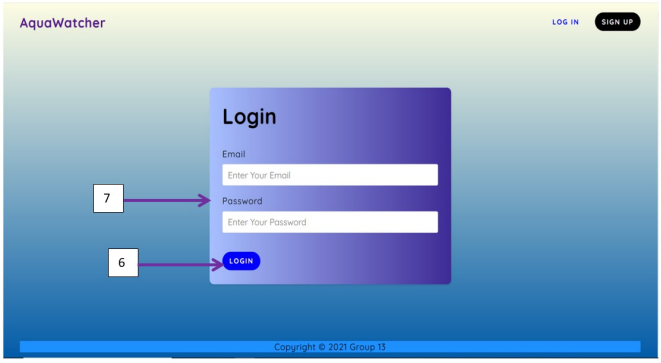
2.3 Website

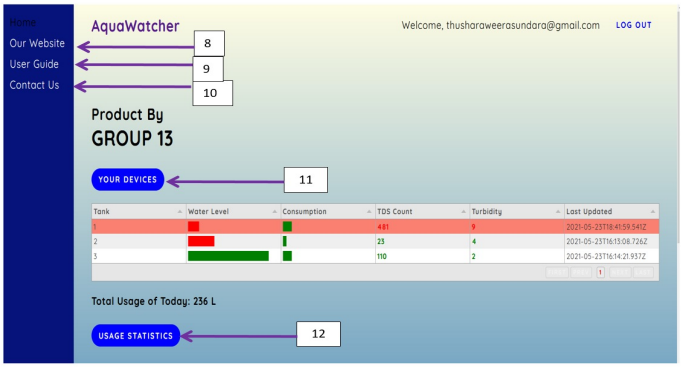
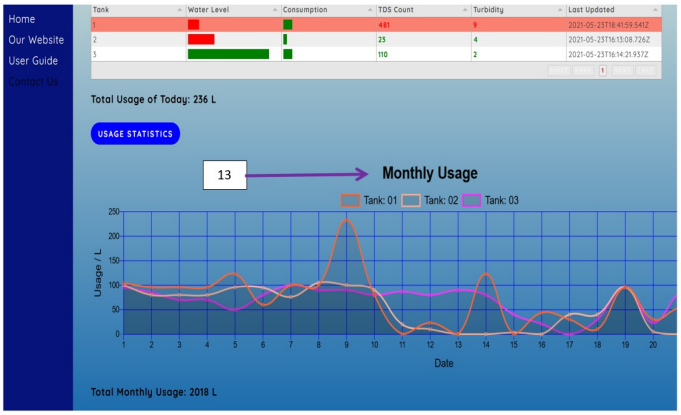
2.3.1 Overview

- 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

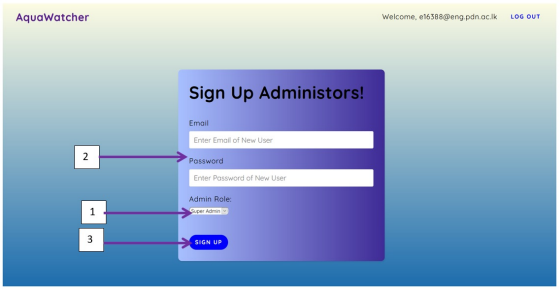
Customer is the user who uses the services and the product. After the user Brought the equipment , user has to create an account www.aquawatcher.ml/signup

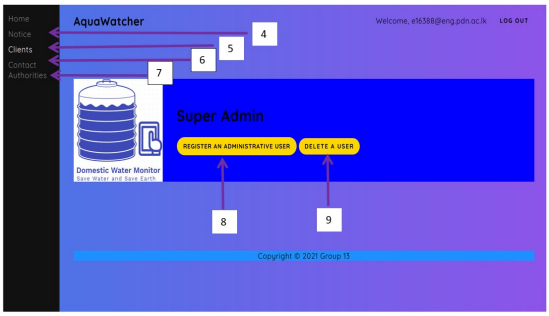
	1. Signup after fill the details in form(4) 2. Switch to login screen 3. Switch to signup screen 4. Details required for signing up a user
	5. Details required from the user to complete the registration process.
	6. Login button 7. Details required to log in to the system

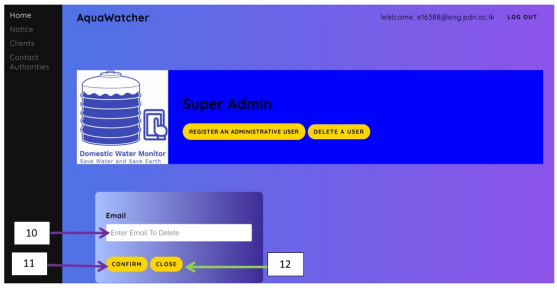
<table>
<tr>
<td> 
</td>
<td>8. The website link of the product
9. Link to the user guide
10. Link to contact system admins
11. Watch the user’s devices
12. Access usage information</td>
</tr>
<tr>
<td> 
</td>
<td>13. Monthly usage graph</td>
</tr>
</table>

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.

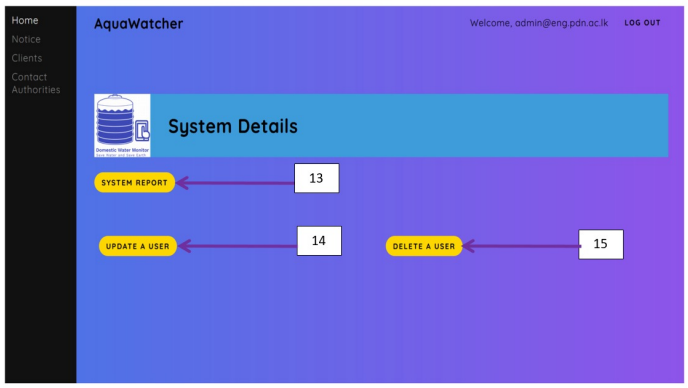
<table>
<tr>
<td> 
</td>
<td>1. Select admin role from here. Super admin or admin
2. Information required for signup
3. Signup button</td>
</tr>
</table>

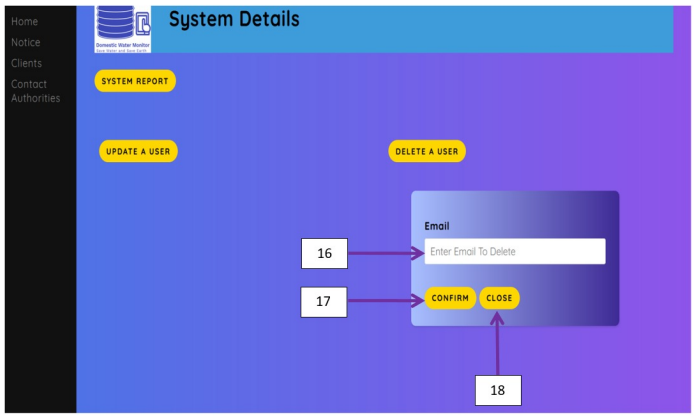
<table>
<tr>
<td> 
</td>
<td>4. Send notices to users
5. Access client information
6. Contact authorities about issues.

8. Register a user as an admin (10,11,12)
9. Delete a user</td>
</tr>
<tr>
<td> 
</td>
<td>10. E-mail of the user
11. Confirm the registration process
12. Close the registration process</td>
</tr>
</table>

2.3.4 Admin

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

<table>
<tr>
<td> 
</td>
<td>13. Get a system report (Devices with issues)
14. Update information of a user(19-25)
15. Delete a user (16,17,18)</td>
</tr>
</table>

	Deleting a user 16. Email of the user 17. Confirm the deletion of the user 18. Close the operation
	19 - 23 : details required to update information of user 25 . Confirm update process 24. Close the update process

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

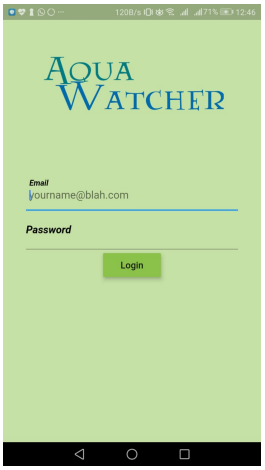
- There are two kinds of notifications. Notifications about water contaminations and notifications regarding the news such as watercuts.
- When logging in, a post request is sent to /login endpoint of the server and the receiving response includes the user details and within that the area of the user.
- 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

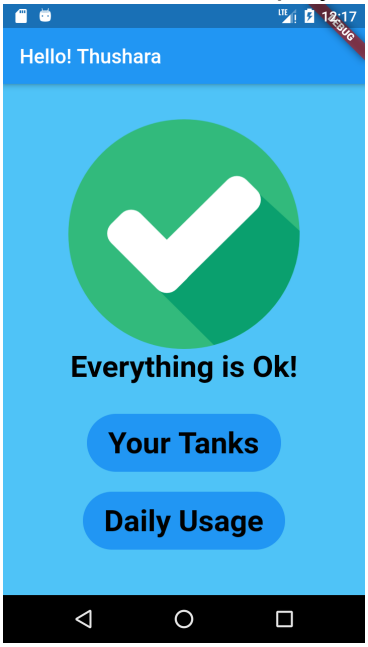
- For the mobile application , there is a login function and after that login function the user will keep the connection with the server through a jwt cookie.
- That cookie will be stored in a shared_preference instance. It is a common library to store non trivial data in mobile applications. Everytime 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.

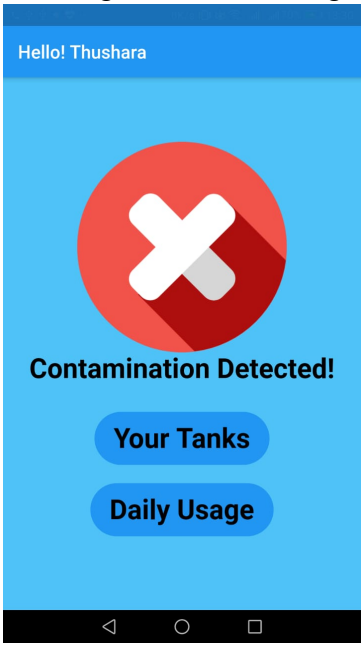
2.4.4 Configuration of the mobile application

- 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.

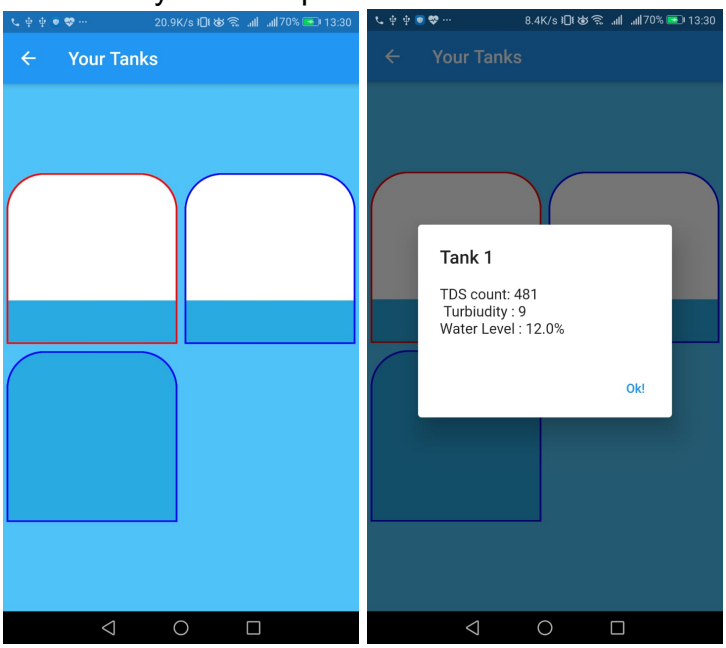


- Then the user will enter into the home screen. Where the overall status of the tanks will be displayed. It will give users a rough idea about the status.

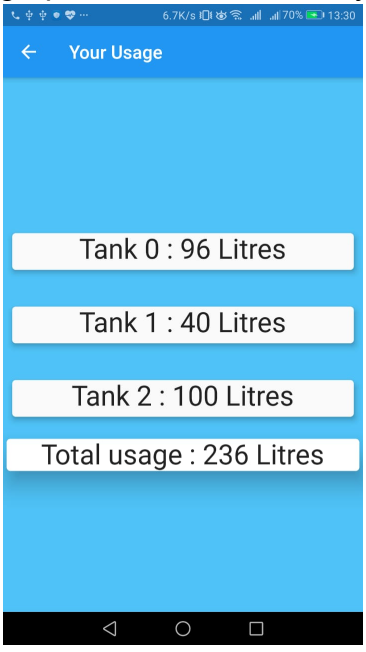




- Users can access the tank information using your tanks button. In the tanks screen,by touching a tank will pop up a message indicating the details of that particular tank. A series of thumbnails are used to represent various water levels and status of the tank. If there is a red outline around a tank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

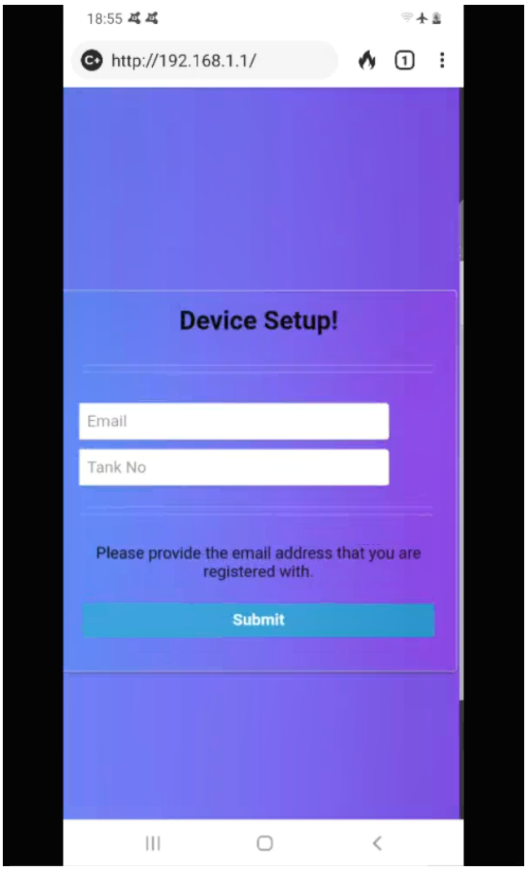
2.5.1 Overview

- 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)

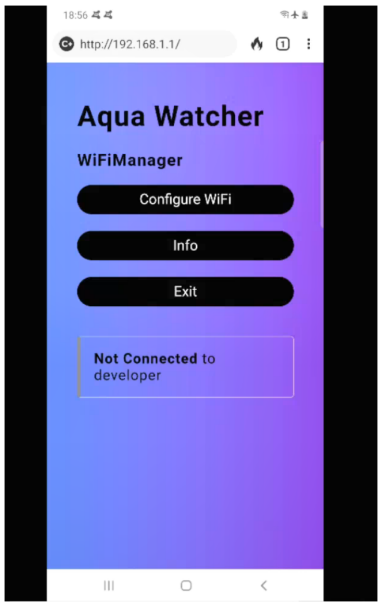
1. Device initialization process

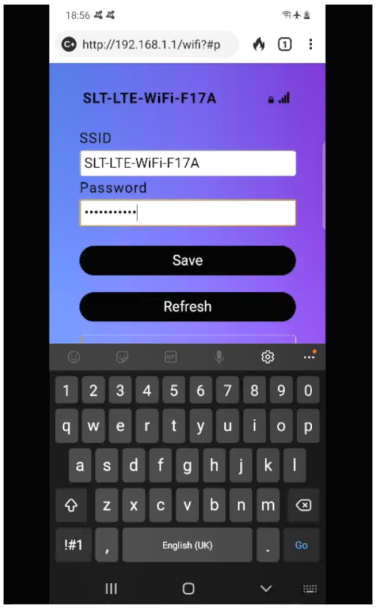
- This module holds the device configuration data such as user name and device id. Upon powering up the device, if the configuration data is not initialized it will start a web server on port 80. You can connect to it via WiFi and go to network configuration. Else, you can go to URL: www.aquawatcher.com.
- “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 accuaracy.
- 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 inturrupts 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

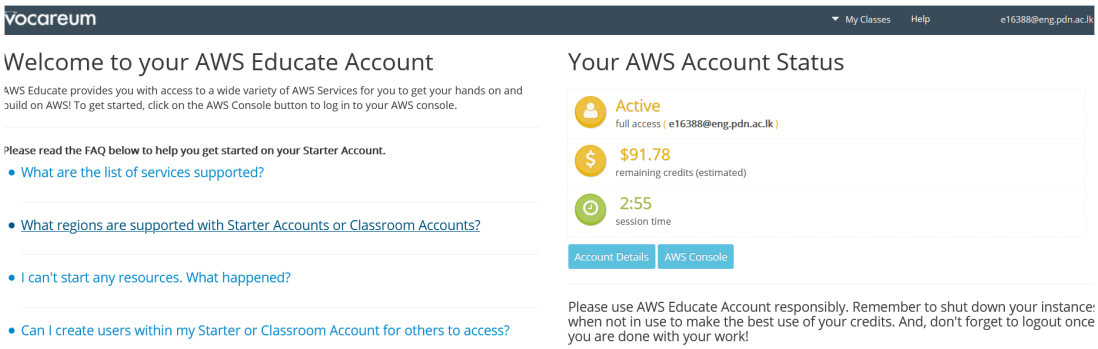
2.6 Amazon Web Services cloud deployment

2.6.1 Overview

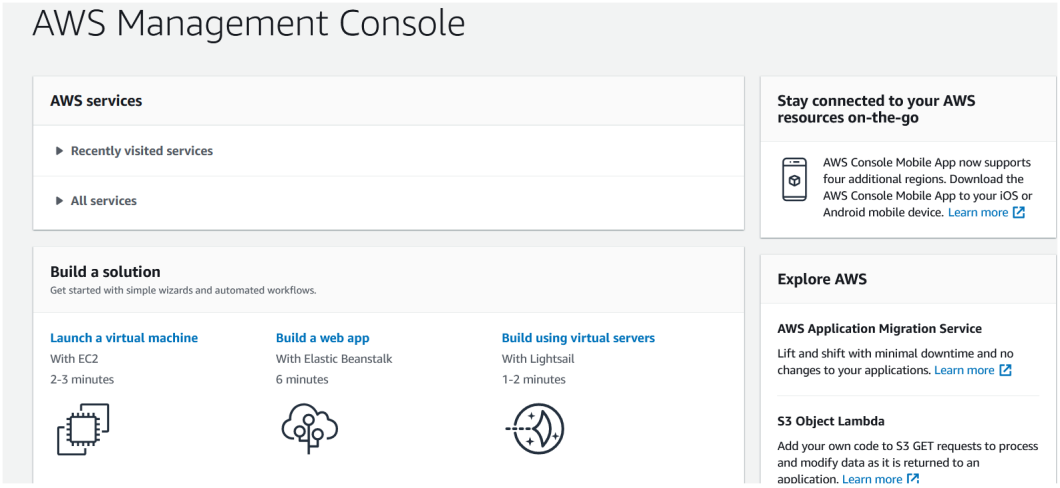
- Web server is deployed on aws ec2 instance.
- Sever can be accessed via port 80.
- Owners can ssh to the instance via port 22.
- Client requests are handled by the server in ec2 instance.
- Route 53 service is used to add a domain name for our deployment.

2.6.2 Steps

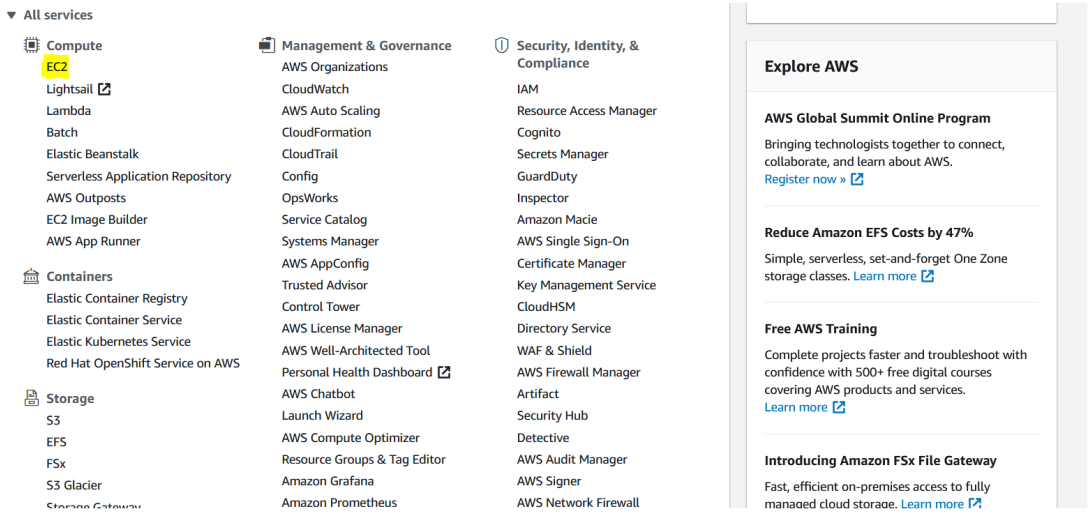
1. **Login to your AWS account.**



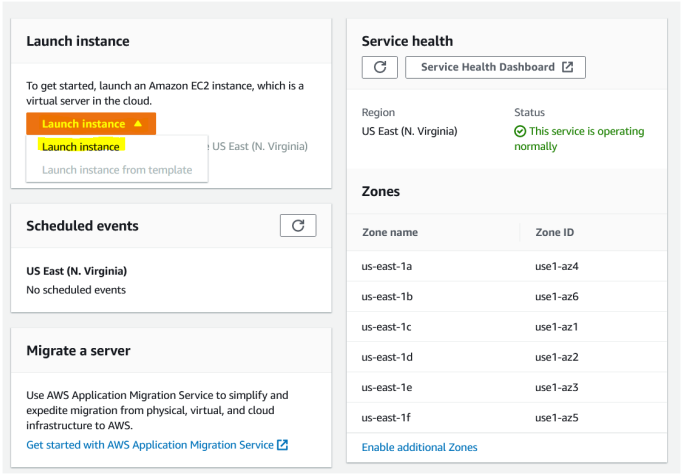
2. **Select AWS Console.**



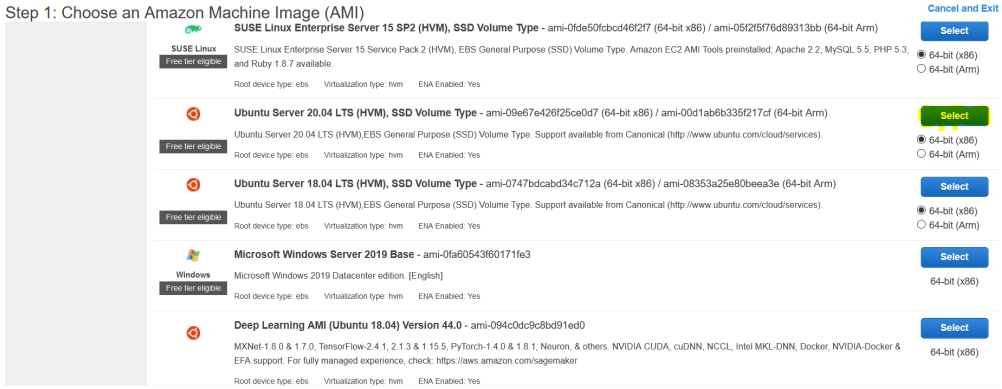
3. Select All services -> EC2.



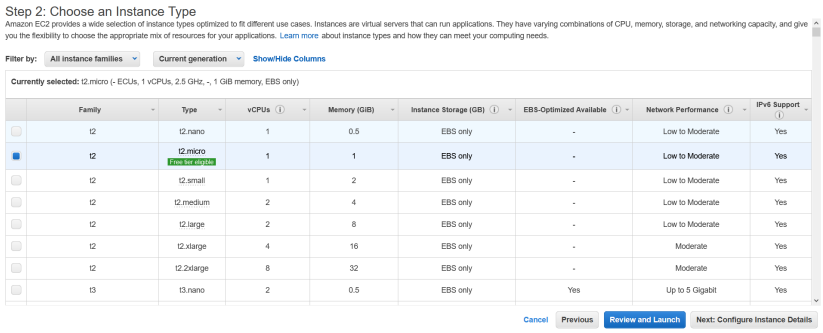
4. Select Launch Instance -> Launch Instance.



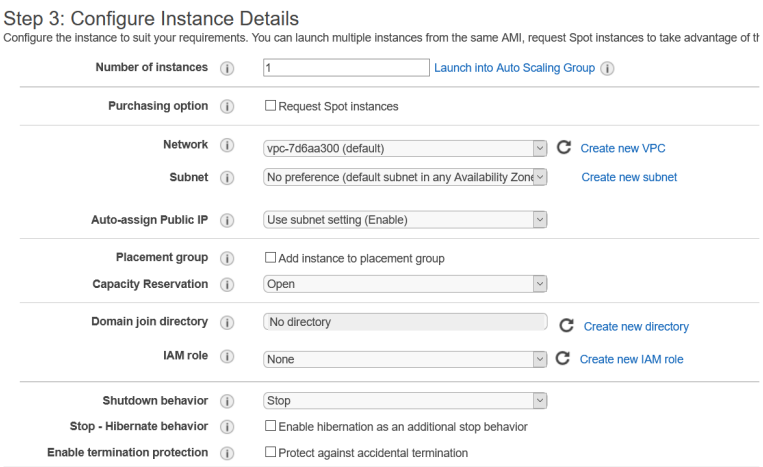
5. Choose an Amazon Machine Image (AMI). Ideally select a Linux based Machine Image.



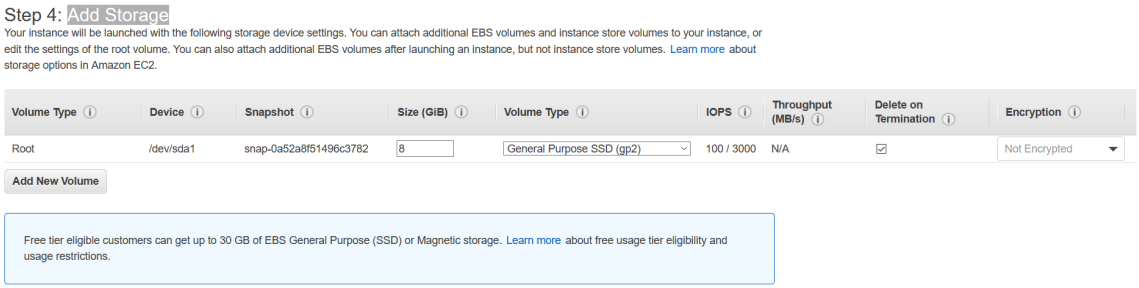
6. Choose an Instance Type



7. Configure Instance Details

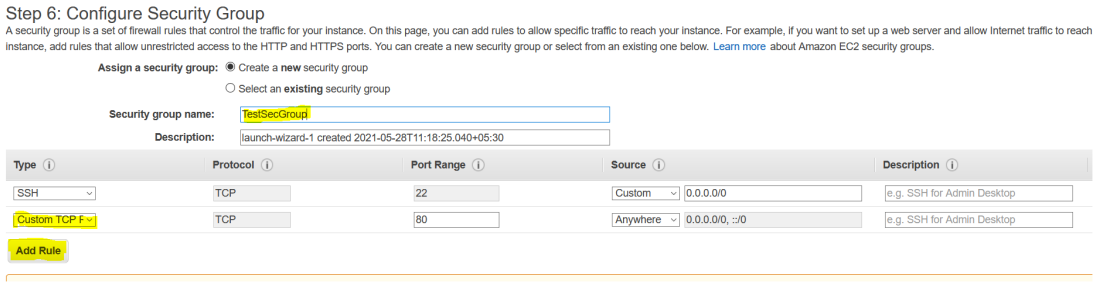


8. Add Storage as your requirement

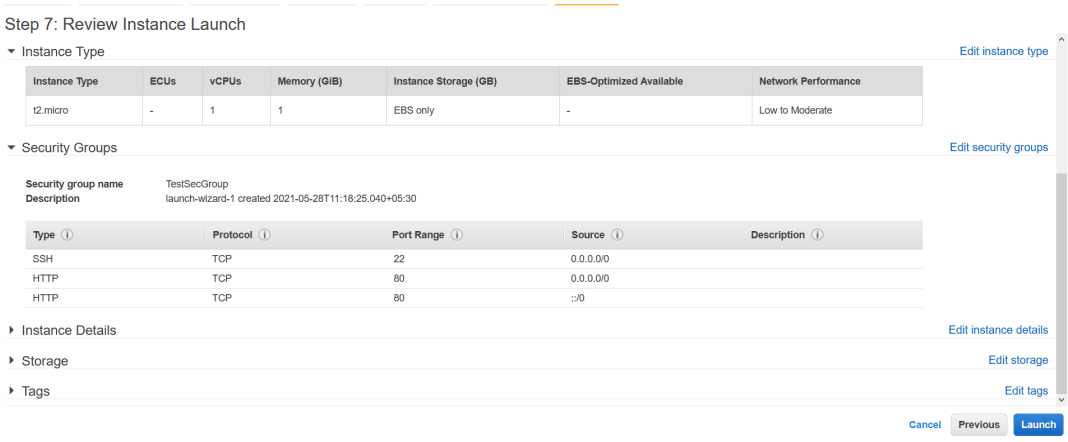


9. Configure Security Group.

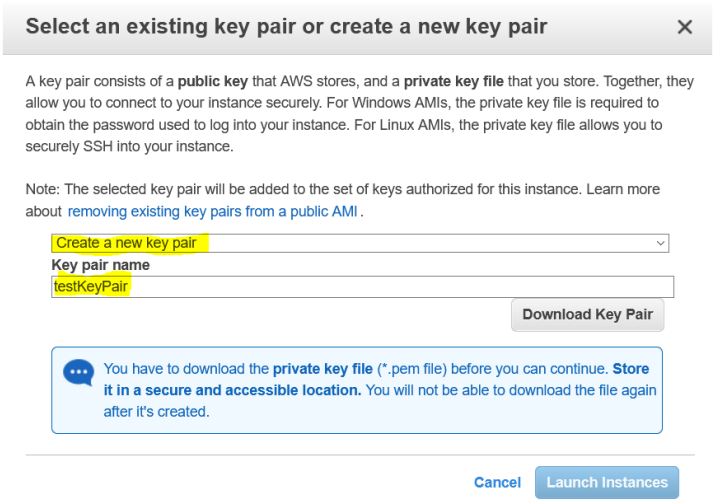
1. Create a new Security Group.
2. Click Add Rule -> add new Custom TCP rule on port 80 and source as given.
3. Now unauthorized access is restricted according to these rules.



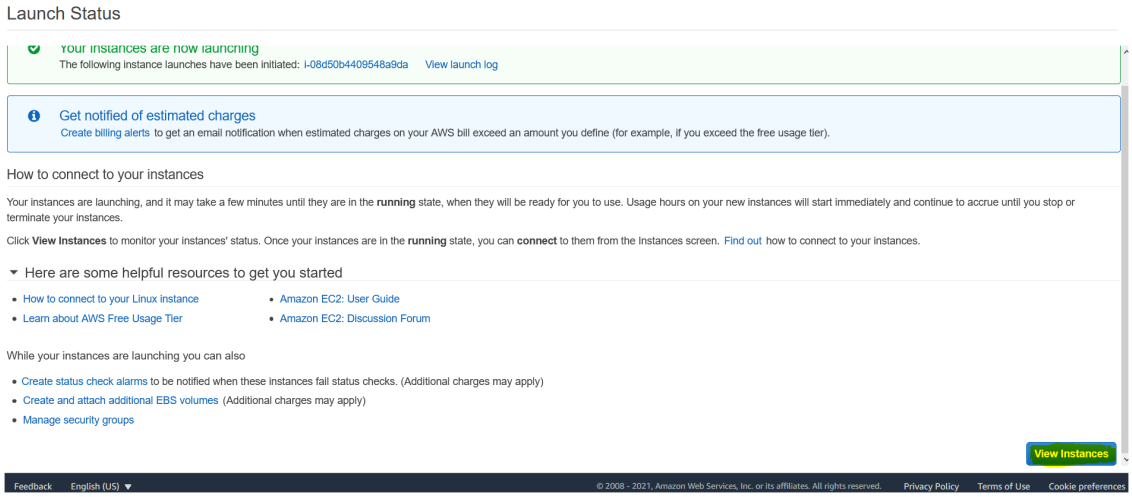
10. Review Instance Launch. Review your selections and launch the instance.



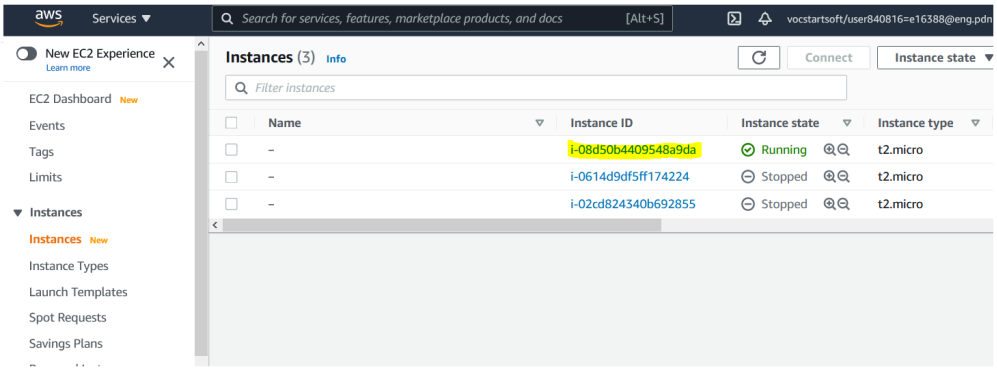
11. Create a new Key Pair to ssh into your instance. Then download it. Never lose this file.



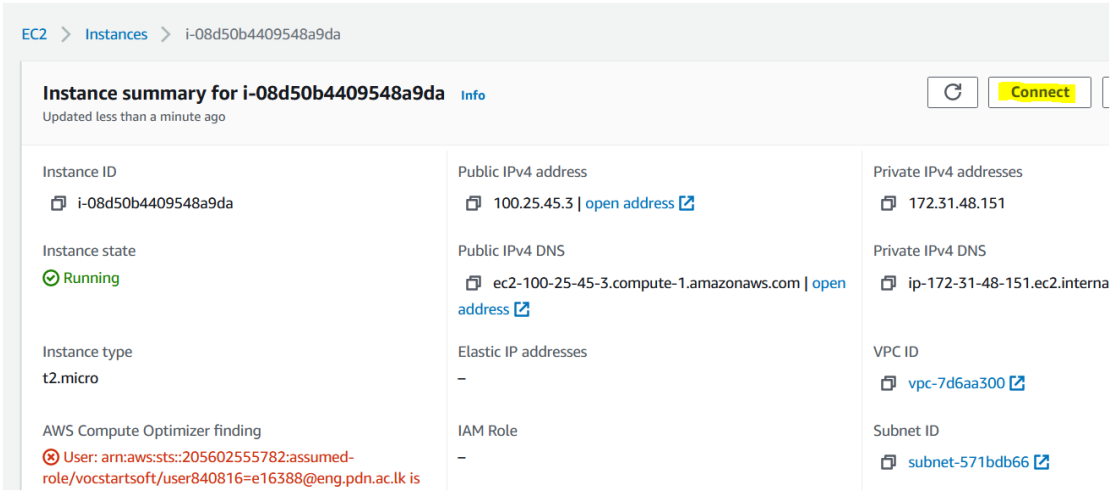
12. Now your instance is launched. To view your instance. Click on View Instance.



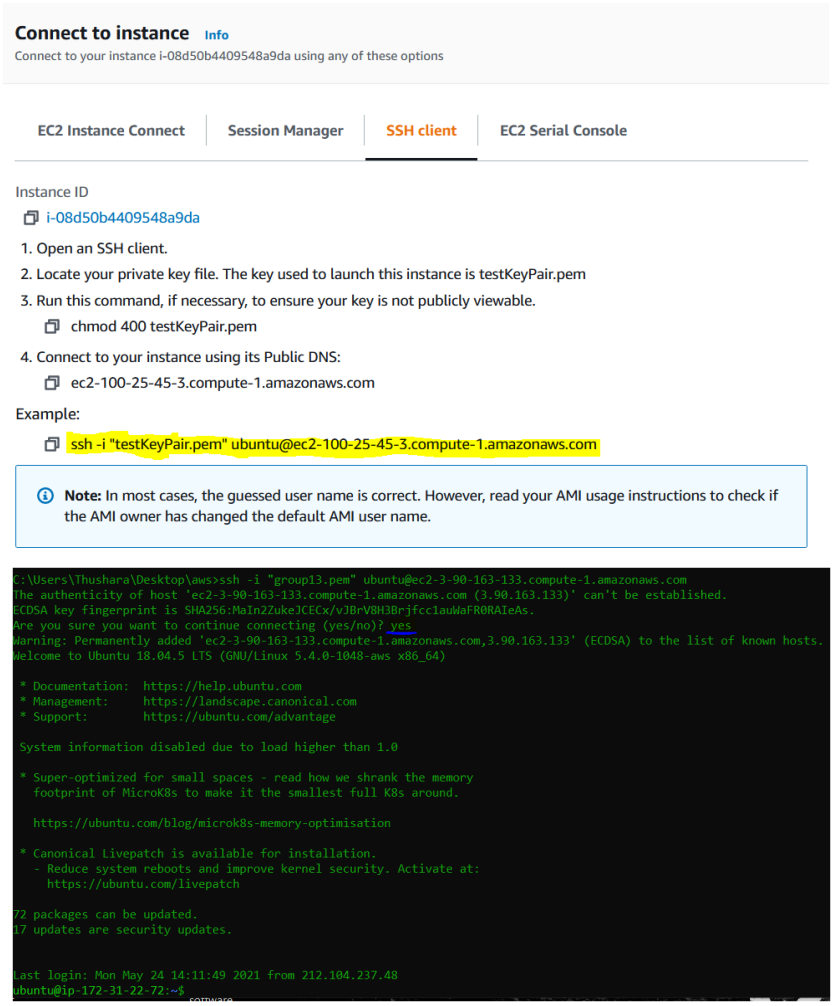
13. Select your instance



14. Select Connect.

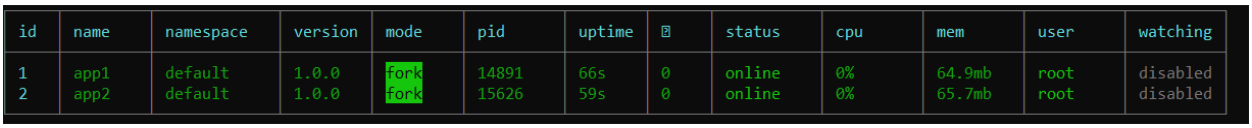


15. Copy the highlighted command -> open a terminal in the folder with key pair->run the command->Type “yes”.



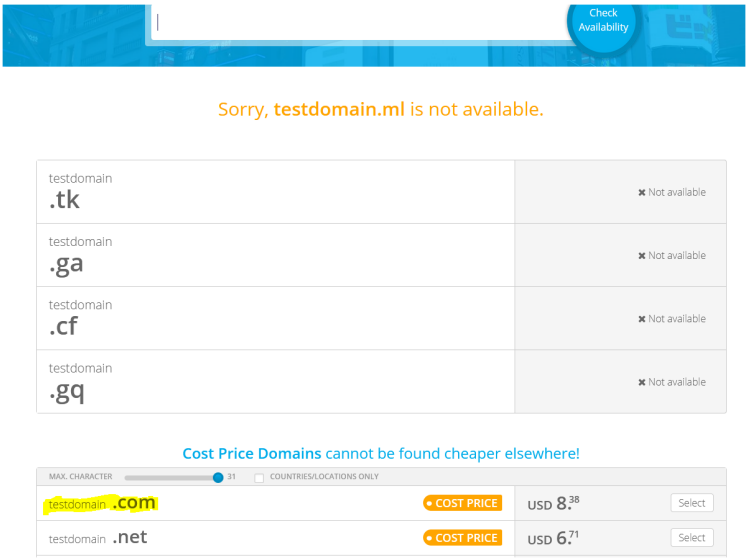
16. **Run following commands.**
    a. **sudo apt update**
    b. **sudo apt install git**
    c. **Git clone git clone -b master --single-branch https://github.com/cepdnaclk/e16-3yp-water-quality-monitoring-and-usage-monitoring-system.git**
    d. **sudo apt install nginx**
    e. **sudo apt install 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.**

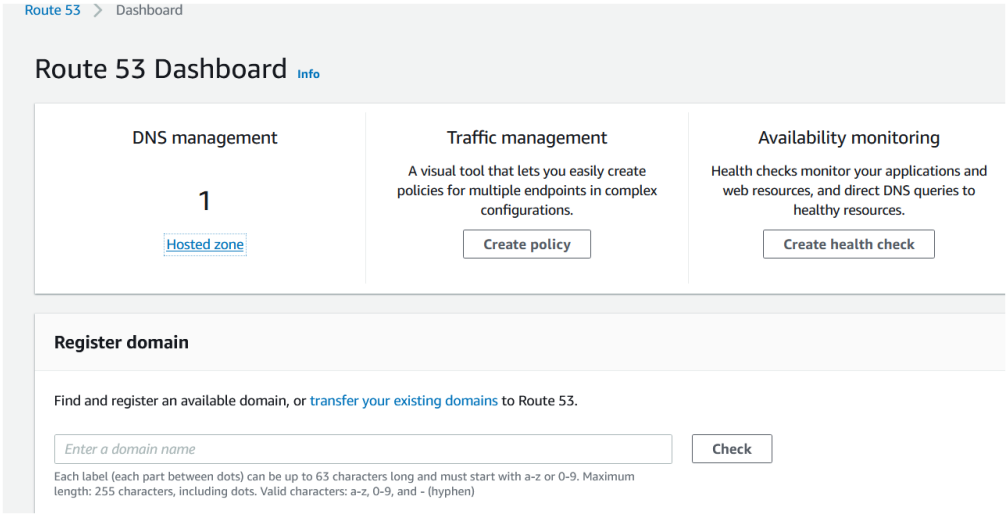


id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
1	app1	default	1.0.0	fork	14891	66s	0	online	0%	64.9mb	root	disabled
2	app2	default	1.0.0	fork	15626	59s	0	online	0%	65.7mb	root	disabled

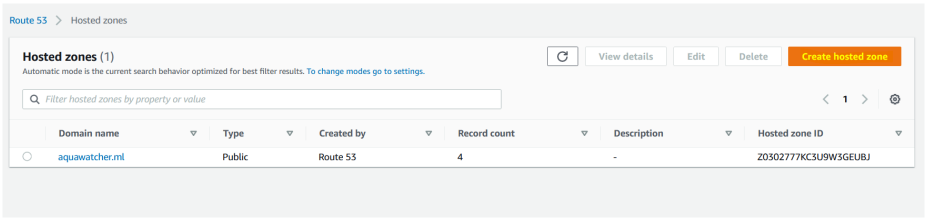
18. **Setting a domain name.**
    Run command cd..
    a. **Go to www.freenom.com**
    b. **Check availability for your domain name.**
    c. **If it is available, select-> add it to the cart->purchase.**
    d. **Provide billing information.**



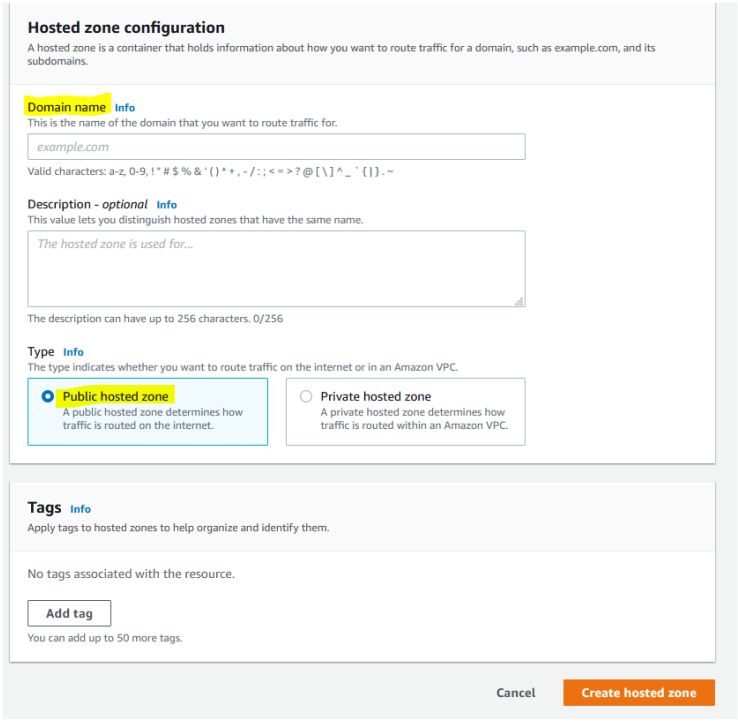
e. **In the below example I have selected www.aquawatche.ml**
f. **From AWS console, select services->Route 53**



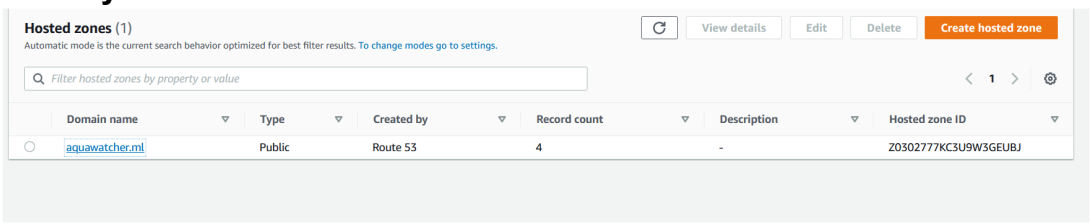
g. **Select Hosted Zone->Create hosted zone**



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



i. **Select your hosted zone.**

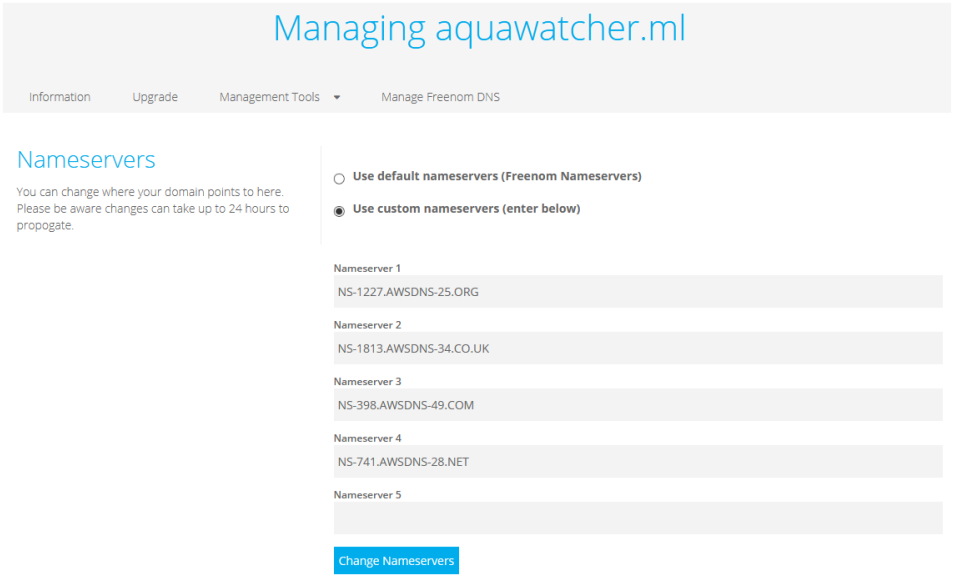


j. **Select your hosted zone.**

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
www.aquawatcher.ml	A	Simple	-	52.91.3.187
aquawatcher.ml	A	Simple	-	52.91.3.187
aquawatcher.ml	SOA	Simple	-	ns-1813.awsdns-34.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
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**



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

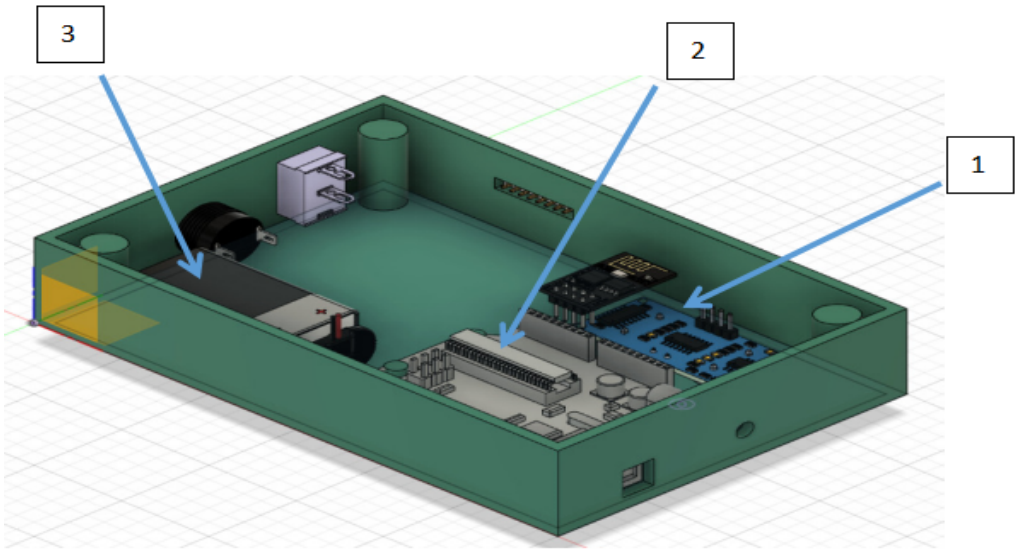
Speakers: Use to notify the user when contamination detected even during a power failure

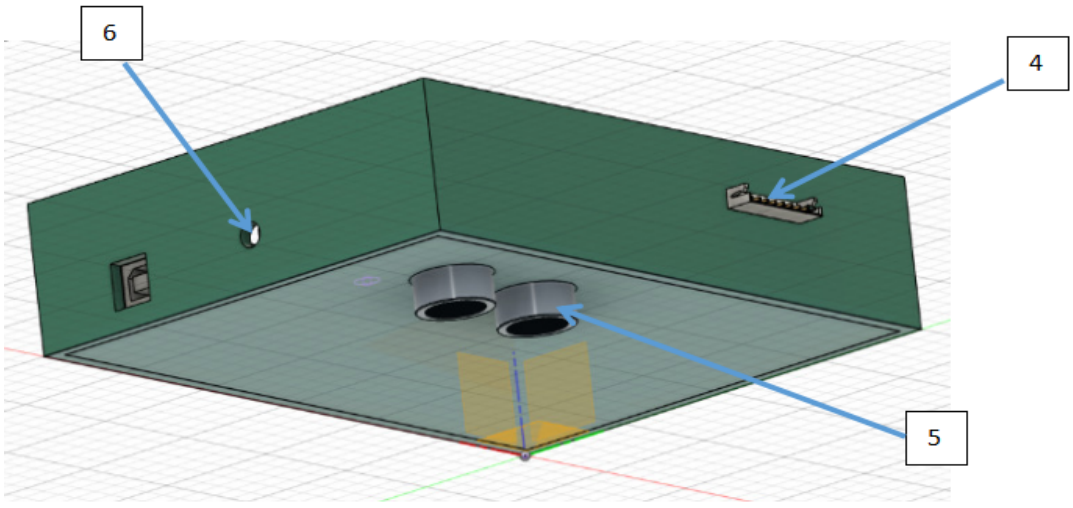
Solar panel: Charges the battery which powers the system

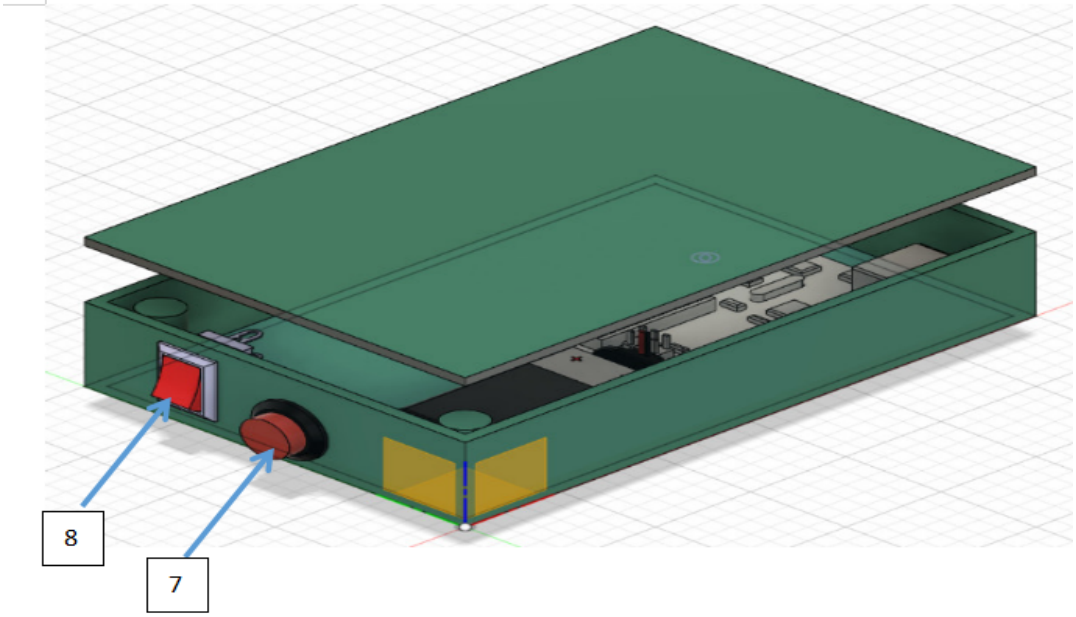
3.2 Designs and Diagrams

3.2.1 Designs for 3D printing

- **Water Level Monitoring Module**

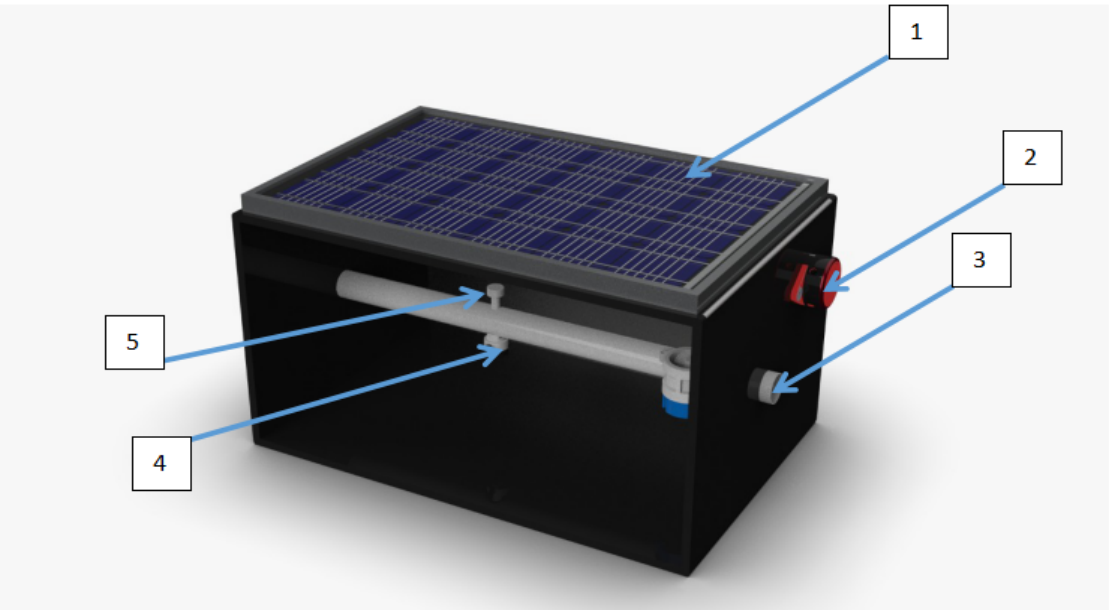






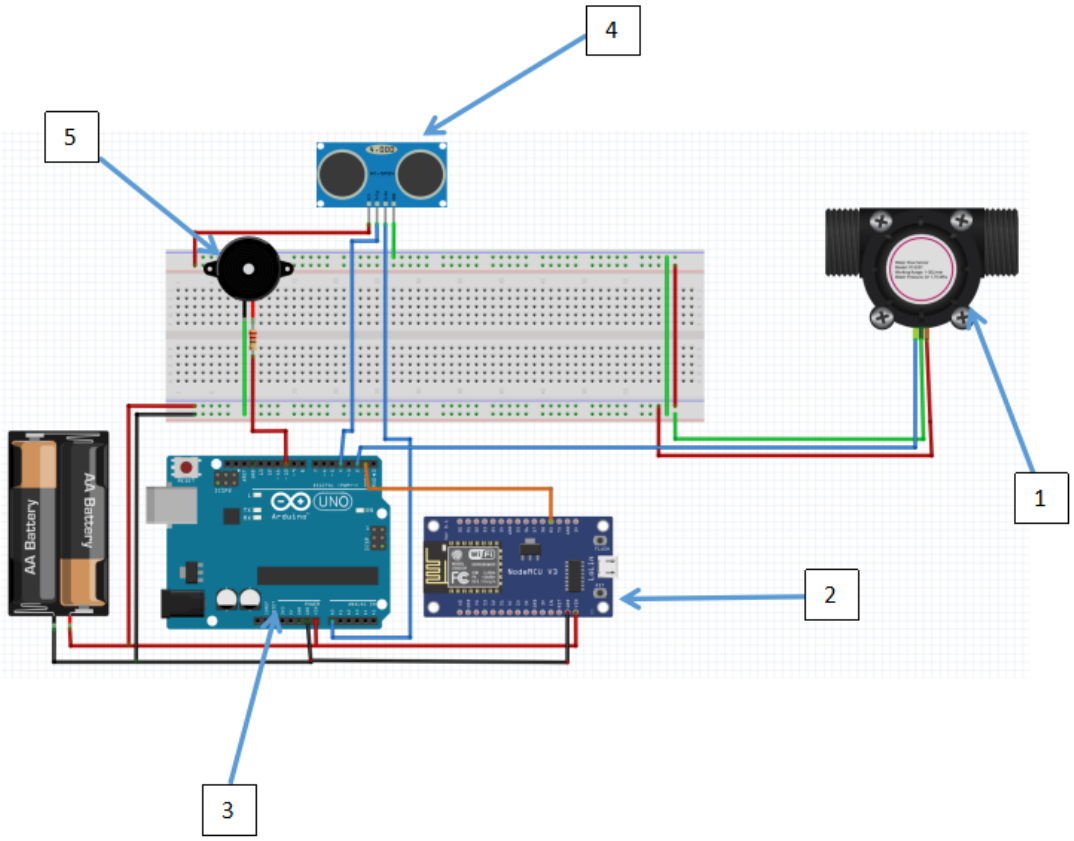
1 - NodeMcu
2 - Atmega328p Chip
3 - Rechargeable battery(Lithium Iron)
4 - Port for uploading code to atmega chip
5 - Ultrasonic sensor
6 - Inlet for wires
7 - Reset Button
8 - Power Button

- **Water Quality Monitoring Module**



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

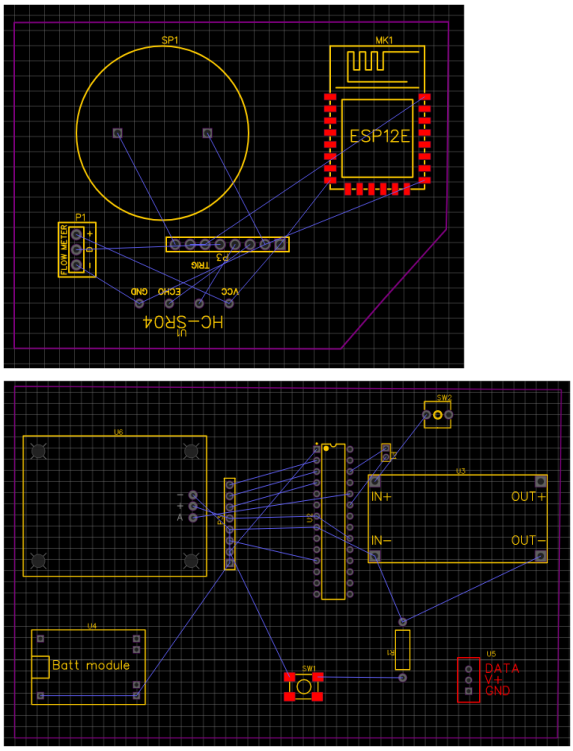
3.2.2 Circuit Diagrams



1 - flow meter
2 - NodeMcu
3 - Arduino Board
4 - Ultrasonic Sensor
5 - Buzzer

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

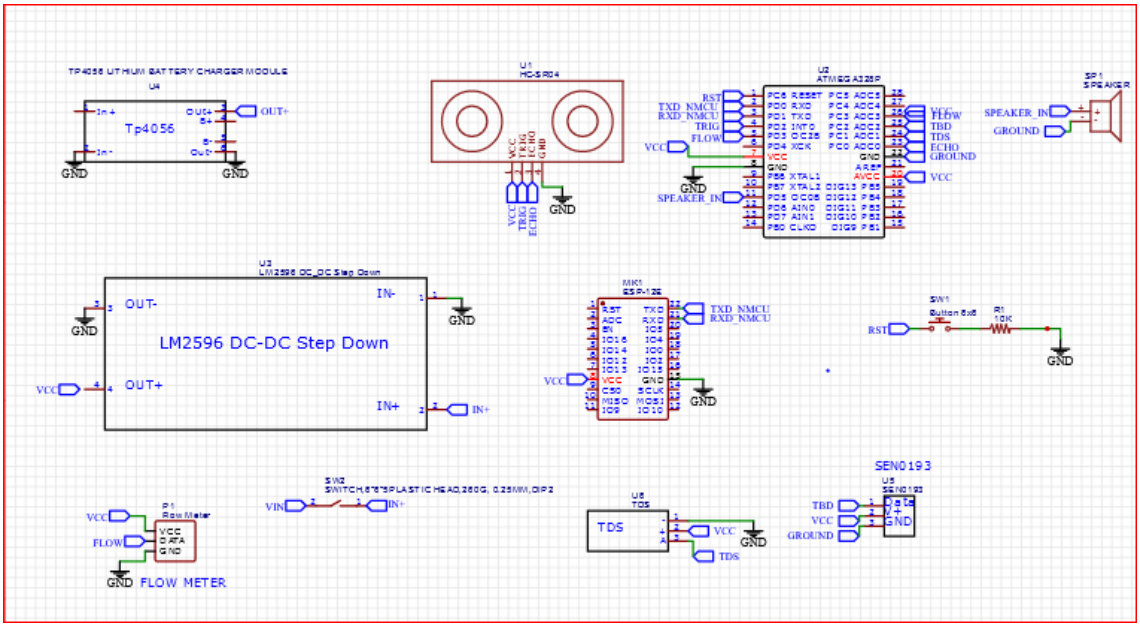
Water Usage Monitor.NODEMCU - MK1

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