

Game Of Thrones

Muthupandian Mohan, Vincent Ibanez, Vinaya Chandiramani, and Shubhra Rajadhyaksha

Abstract—This final report illustrate the implementation and design of a bathroom reservation system as a project assignment for COEN 315, Web Architecture and Protocols. The goal of this project was to develop a product that can be viable in a business situation but also give us the opportunity to implement a system that involves hardware and software components that interact through the web. Components implemented by the team members are a microcontroller-based modules with input sensors and outputs, firmware code for the different modules, front-end PHP-based server, Backend Restful API web server and a user-facing web and mobile application.

Index Terms—Game Of Thrones

I. INTRODUCTION

A. Summary

Almost everyone has had an urgent need of using a wash-room in a public setting, especially when we are in middle of something very important. In these instances, one would have to individually go and check if any of the washroom stalls are free. In an office setting where there are much more employees compared to the number of stalls, a substantial amount of time could be wasted and could hampers productivity. If all the stalls in one particular washroom are full, an employee will have to either wait there till they get vacant, or go to other washrooms. And this process goes on and on if all the stalls are occupied. This process takes lot of time and energy and it is really frustrating.

After going through all these steps, if we manage to get a stall free, what is worse is that it is sometimes out of tissue papers, not clean or smelling weird. In that case we have to start the process of searching free stall again. What if we come up with the system which solves all these issues?

A bathroom management system resolves all these issues. With this, a user can easily find out which all stalls are free, check the restrooms that have most stalls open, with the comfort of seating inside your office cubicle or even reserve the stalls for sometime before you actually check in, check if the stall has tissue papers or if it needs refill, with auto odor control system in each stall. This system with feedback, which help the management to clean the stalls as soon as they get alert saying “ Stall needs cleaning” or “ Paper needs replacement” is what our project is about. This can be also used to detect other issues in future like plumbing issues or leakages, etc.

Bathroom custodians and facility managers will also find benefit in a management system since the data aggregated from this system can allow them to manage bathroom cleaning and

maintenance operations efficiently. This can effectively result in dollar savings from employee productivity and minimal bathroom downtime and it also greatly improves employee satisfaction.

II. PROJECT OBJECTIVE

A. The Purpose

The main purpose of this project is to develop a system that can solve the following problems:

- Check status of stalls (available or busy) from website or mobile device
- Have a fresh toilet with air freshener by just pressing a button on screen
- Know status of cleanliness and toilet papers beforehand
- Request maintenance to clean the stalls in advance by button click
- Reserve a stall
- Save time

B. Rational for Hardware and Software

In order for a reservation system to be effective and easy to use, it must be easily accessible to the end user. Smart phones and web applications has always been the most accessible device for many users. Almost everybody that has a smartphone, has access to a cellular internet connection. Therefore, implementing a reservation system through a web-based approach would be most ideal solution in these types of applications. This would mean that a reservation system running on the cloud server will need to handle data from the smartphones. In addition to this, the reservation system must also have a way to relay the information to the IoT devices and at the same time gather information from the sensor inputs. This implementation must have the ability to be scaled when many IoT devices are deployed in the field. This means that the IoT device must also be able to communicate to the cloud server through the internet infrastructure. For a bathroom reservation system to be effective, there must be established need for a queuing system. This is means that a typical application where a queuing system would be effective is an office or public environment setting where there are a high number of users compared to the number of bathroom stalls.

C. Related Products

We performed research on currently existing products on the market to assess the need in the market. We looked at several similar products ranging from a project prototype to a developing commercial product. Although most of it is still at its infancy, we looked at their current features.

TABLE I
PRODUCT COMPARISON

	App-based	Modular	Reservation System	Does not require proprietary hardware
Good2Go	x	?	x	
Hackster Toilet Management System	x		x	
Game of Thrones	x	x	x	x

The Hackster toilet management system is a simple reservation system built using a microcontroller with an ARM processor on the same device. It has an accessible web interface that acts as a web application that provides a simple reservation queue. Although it works, it is still lacking features that would be useful in an office environment setting.

Next we looked at developing commercial product, the Good2Go bathroom system. This solution certainly has more features since its developed by a larger company. Although the information provided in their website is limited, it can be derived from their product catalog that their solution uses an app-based reservation system that power on a fully automated bathroom solution. From wireless door locking mechanism, wireless soap dispensers and wireless faucets. It seems like this would be a good solution for an office environment. However, we believe that its strengths is also one of its drawbacks. A system of this scale would require proprietary door locking mechanisms, proprietary water valves as well as proprietary sensors and gadgets. To deploy something like this on large scale would mean that more capital up front would be required and integration may or may not be easy.

III. DESIGN

A. Our Solution

Our design is inspired by two key guidelines: simplicity and low-cost. With these, we developed the following objective for our design:

- Our solution would build on top of currently existing bathroom products
- Our solution would not require any proprietary hardware (i.e. new door locks, new toilet paper holders etc.)
- Our solution would be modular and low-cost.

In designing the IoTs that we would need, we ask ourselves the basic question of:

- What are the essential needs for a bathroom user that we need to capture and report?

A typical bathroom user look at a few things. First, the bathroom must be available for use, then it must be working, clean and more importantly it must be stocked with bathroom supplies. To provide these information, we developed the following IoTs:

Main Bathroom Door (BD) Unit

The main BD unit acts as the main brains of the system. Every stall under the Game of Thrones system would require its own BD unit. This unit is responsible for capture information regarding door lock and unlock status. It will also have a display that could display several important information such as bathroom reservation name, clean and maintenance status.

Additionally, the bathroom will have buttons that will allow the user to request bathroom cleaning as well as bathroom maintenance requests.

Toilet Paper (TP) Unit

The TP unit will provide information on the toilet paper level. This is separate from the main BD unit since in some bathroom stall setups, this is usually placed closer to the toilet. The TP unit will provide information if the toilet paper is running low. This information can then be used by the bathroom custodians to alert them to restock bathroom supplies. Additionally, it also has a TP request button that allows a user to override the TP sensor and to alert the custodian that a refill is needed immediately.

Air freshener (AF) unit

The AF unit will provide fragrance to the bathroom stall. This can be triggered automatically every after bathroom use or it can be triggered remotely if the user would like the bathroom to smell fresh upon their arrival. The AF unit is designed to accept a standard Glade cartridge refill so that no proprietary air freshener refill module has to be developed.

B. Design and System Architecture

One of the main features of the Game of Thrones system is to be able to reserve a bathroom stall through the mobile app. Due to this web integration is a essential part of the design. The IoTs mentioned in the previous section communicate to the IoT server through Wi-Fi by means of RESTful HTTP requests. The reservation system logic and data aggregation is done in the IoT server. The IoT server captures several bathroom information such as Door Open/Close Status and toilet paper level status and as such, it stores state variables for each of the bathroom variables. Between the user and the IoT server also lies the AWS server. The AWS server serves the front-end requests that parses the RESTful results into a more usable format. It is then relayed to the Web application through the browser or through the PhoneGap app. All of these systems rely on the internet infrastructure for communication.

The IoT server is designed to have 3 slots for queuing since anything more than 3 may prompt the user to simply reserve another stall. The algorithm used for the reservation is simply a First-in-First-out queue in which the first user who reserves the stall is also the first user to take turn in using the stall. After the first user leaves the stall, the second user fills in the first spot, then the 3rd user fills the second, then the 3rd slot becomes empty, free to be reserved by a new user. When the queue is empty then a user physically enters the stall, a place holder will be placed in the queue indicating that a current user is in the stall.

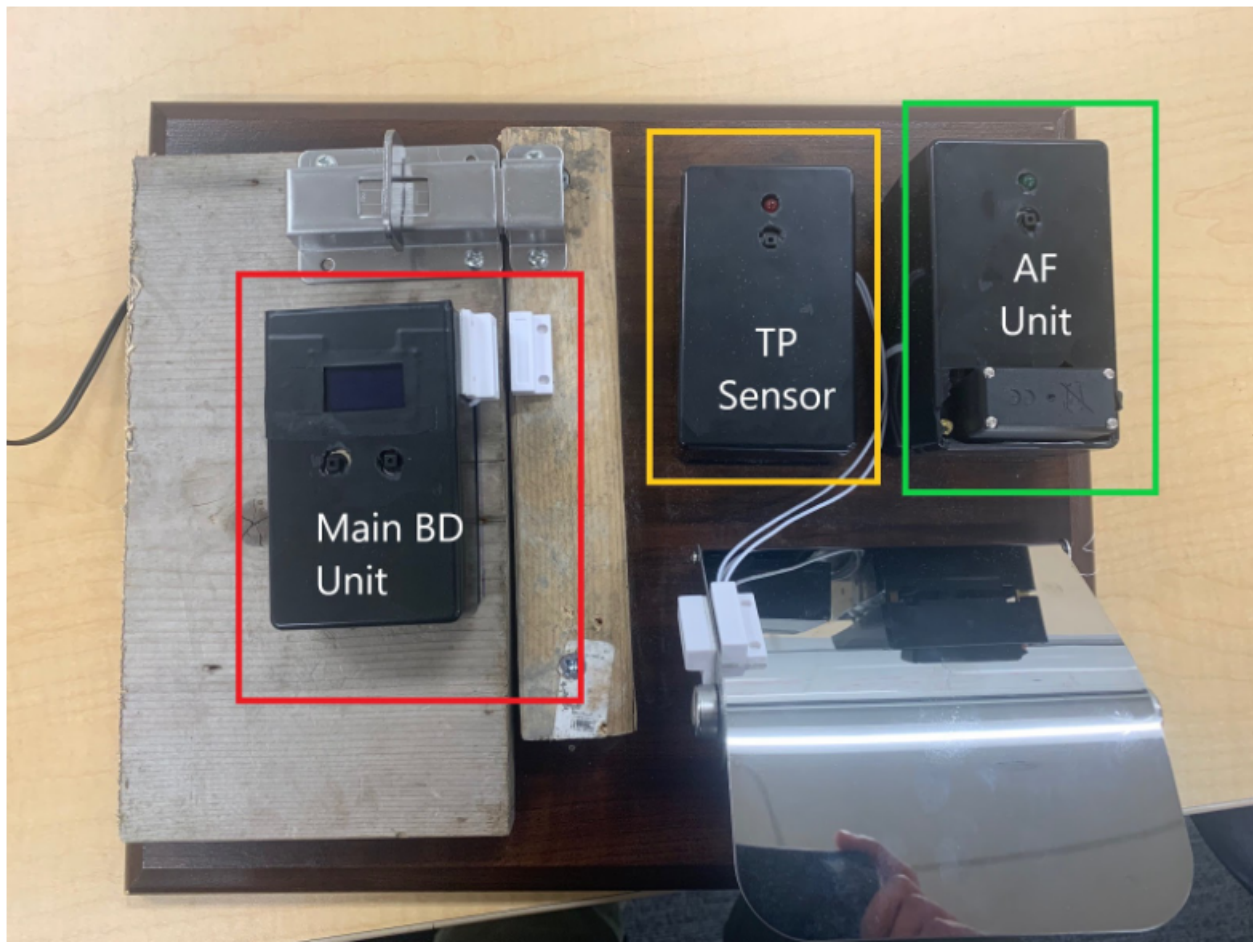


Fig. 1. IoT unit

Additionally, the main BD unit display output plays a large part in the queuing system. When a stall is reserved it displays the name of the user that the stall is currently reserved for. This helps people properly identify the stall that is reserved for them and to prevent a user that is not reserved to accidentally enter a stall that is reserved for others. This same display also displays information if a stall is occupied, open or out of order.

C. Hardware

The IoTs are designed with cost in mind. The HiLetGo ESP32 was used as the microcontroller since it works under Arduino development software which have large library support. It is also has several important features integrated such as a power regulator, I2C interface as well as a WiFi interface. Another reason is that these device are relatively inexpensive and breaking one or two of them wouldn't be too costly for development.

1) :

a) Display: The display selected for used was an OLED screen. This is chosen as it consumes low power, however it is not without its drawbacks. One issue with OLEDs is that the display image must be constantly moving to avoid screen burn in. To resolve this, a sideways scrolling routing was implemented when displaying text.

Although the OLED screens come in I2C and SPI interface variants, the one that we purchase was an I2C version. The performance of this interface was originally okay until we noticed that the I2C pins in the ESP32 shorts after an hour of usage. Later on, we traced this down to a missing pull-up resistors suggested on the I2C lines.

b) Sensor Selection: The sensors are need to capture the door lock/unlock status as well as the displacement on the toilet paper holder flap between the normal and low position. One of the options that was looked at is a linear resistor which could measure displacement. This could be attached to the shaft of the bathroom lock and calibrated to capture lock and unlock position of the lock shaft. However due to the mechanical nature of this setup was abandoned for a non-contact sensor option.

Next a non-contact sensor, the Reed-Hall magnetic sensor is evaluated. This sensor is similar to the sensors found in most home security systems that triggered when contact is broken on the window or on the door. Since this is non-contact by nature, this is much more durable and runs on very low power. Due to the two-state nature of the door lock/unlock status, this sensor fits the use case. Proper resistor termination was used in order to guarantee low current draw in powering this sensor.

c) Servo Motor Issues: The servo motor is used to push the Air freshener cartridge against the shaft that expels the

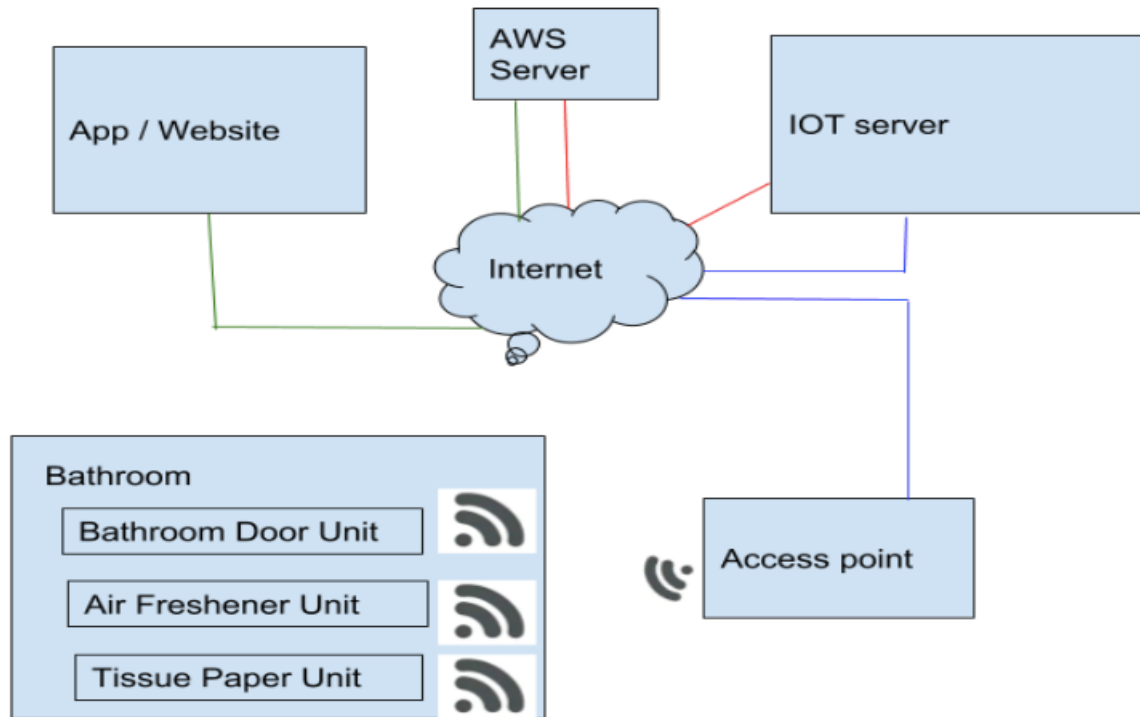


Fig. 2. System Architecture

air freshener on the top of the AF unit. Originally, the servo motor is powered directly from the regulator of the ESP32 chip. However, during testing, it was seen that the servo motor was stalling when it is activated. This is then realized due to the insufficient power supply coming from the ESP32 board that is not designed to handle the current draw from the motor. The address this peak surge in current a capacitor was added across the 5V supply lines to add a little bit of current on the single instance that the motor was activated.

d) *Platform*: The circuits were assembled on a proto-board then mounted on a black project box. A number of wires, jumpers, resistors, LEDs and button were used to connect components to one another.

e) *Firmware*: The main logic for the IoTs are fairly simple. Every time an event is detected, it generates an HTTP request to the IoT server so that status flags between the IoT server and the IoT are synced. An event is detected by constant polling, however some events are instantaneous in nature such as when the momentary button is pressed. To properly capture this, a Interrupt routing was added to capture the edge of the button press. A flag is then asserted. Then later on during the main loop this flag is checked and any post process is run to address the button press. A button debounce logic is also added to provide clean button presses and avoid re-triggers.

D. Software

Client side Scripting for designing the website:

- HTML5
- CSS

- BootStrap

Server Side Scripting:

- NodeJS - Used to get IoT updates, backbone of IoT Server
- PHP - used to send GET requests to IoT server
- JavaScript - Used for Document Object Manipulation

Mobile:

PhoneGap (desktop app)- For creating basic structure and development

PhoneGap for android app (<https://build.phonegap.com/>)

1) *Exposed API List*:

a) *Bathroom Door Unit APIs*: <http://www.vibanez.com:8071/mainbd/status>

- Main BD Status request

- Expected values:

- bd_rsrvd: 1/0
- bd_uname1:EMPTY
- bd_uname2:EMPTY
- bd_uname3:EMPTY
- bd_uname4:EMPTY
- bd_uname5:EMPTY
- bd_drclk: 1/0
- bd_clnrqst:1/0
- bd_mtnrqst:1/0

- Default response:

○ bd_rsrvd:0,bd_uname:USERNAME,bd_drclk:0,bd_clnrqst:0,bd_mtnrqst:0
[http://www.vibanez.com:8071/mainbd/request?bd_uname=](http://www.vibanez.com:8071/mainbd/request?bd_uname=USERNAME)

USERNAME

- Request reservation for user with name USERNAME

- Must be issue only if `bd_rsrvd = 0`.
- If successful, `bd_rsrvd` will change to 1, then `bd_uname` will change to USERNAME.

`http://www.vibanez.com:8071/mainbd/cnclrsrv`

- Cancel current reserved user in the stall.
- Must only be issued if `bd_rsrvd = 1`, and `bd_drck = 0`, meaning stall is reserved but user hasn't arrived.

`http://www.vibanez.com:8071/mainbd/drck`

- Set door lock flag in the BD unit

`http://www.vibanez.com:8071/mainbd/drunck`

- Reset door lock flag in the BD unit

`http://www.vibanez.com:8071/mainbd/clnrqst`

- Set clean request flag in the BD unit

`http://www.vibanez.com:8071/mainbd/clnclr`

- Reset clean request flag in the BD unit

`http://www.vibanez.com:8071/mainbd/mntcrqst`

- Set maintenance request flag in the BD unit

`http://www.vibanez.com:8071/mainbd/mntccclr`

- Reset maintenance request flag in the BD unit

b) *TP Sensor Unit APIs:* `http://www.vibanez.com:8071/tp/status`

- Status request
- Expected Values:

○ `tp_lvlok:1/0`

○ `tp_clnrqst:1/0`

- Default response:

○ `tp_lvlok:0,tp_rqst:0`

`http://www.vibanez.com:8071/tp/lvlokset`

- Set lvlok flag in the TP unit

`http://www.vibanez.com:8071/tp/lvlokreset`

- Reset lvlok flag in the TP unit

`http://www.vibanez.com:8071/tp/setrqst`

- Set tprqst flag in the TP unit

`http://www.vibanez.com:8071/tp/clrrqst`

- Reset TP request flag in the TP unit

c) *Air freshener Unit APIs:* `http://www.vibanez.com:8071/af/spray`

- Request Spray:

○ `sprayrqst:1`

`http://www.vibanez.com:8071/af/sprayStat`

- Current Status:

○ `sprayrqst:X`

`http://www.vibanez.com:8071/af/sprayClr`

- Clear Spray Flag:

○ `sprayrqst:0`

E. User Workflow Testing

The main functionalities of the app include:

- Reserving a Restroom
- Maintenance of the Restroom

From the main landing page, we navigate to 'Reserve a Restroom' which brings us to this screen, which gives us an overview of the status for each bathroom, and a user can request to get in a virtual queue.

On entering the user details, he user gets added to the queue (which can accommodate a maximum of 3 at a time), and can even request for different amenities (like requesting toilet

paper, spraying the air freshener, or requesting maintenance) while using the restroom. The request then gets logged in the maintenance view as well.

The maintenance page reflects how the management services views the user requests.

IV. RESULT

An employee submits their username to reserve a restroom and thus gets into a virtual queue. Post that, the user can request restroom maintenance at his/her convenience, which gets logged on the maintenance screen, and is subject to clearance by the maintenance team. Clean UI/UX and debouncing logic helps to declutter any retriggers or accidental requests.

V. CONCLUSION AND RECOMMENDATIONS

To summarise the app:

- It provides an easy and convenient platform to reserve a restroom.
- Virtual queuing saves time so an employee can approach the stall only when it's their turn.
- Employees can request restroom services at the click of a button.
- Keeping tabs on restroom maintenance is now easy and on cue.
- Janitorial service can clear requests once they have been tended to.

VI. LESSONS LEARNED

- When we were looking for project ideas, our discussions started with jotting down whatever problems we faced in our personal lives which we can take care of by building our product. Most of us had faced the toilet waiting problem, so we thought of sticking to this one. So the first point worth mentioning is : Always make a product which you would have loved to use.
- We developed a product which will help the users in positive way. It will save their time. So whenever we develop any product, we have to think about what users want and how can our product help in making someone's life easier. The UI/UX aspect of it was designed, keeping in mind the end users and how simple they need it to be.
- This project helped us to understand that front-end and back-end integration is the most difficult part and understanding common functionalities and each other's perspectives. Another major learning from this project is getting extreme clarity on the requirements of the project, especially on items that can be taken for granted. We thankfully managed to finish everything on time.
- Working with IOT/ hardware and web technologies together is not as easy as it looks. It took lot of sleepless night to come up with this product. But the learning will stay forever with us.

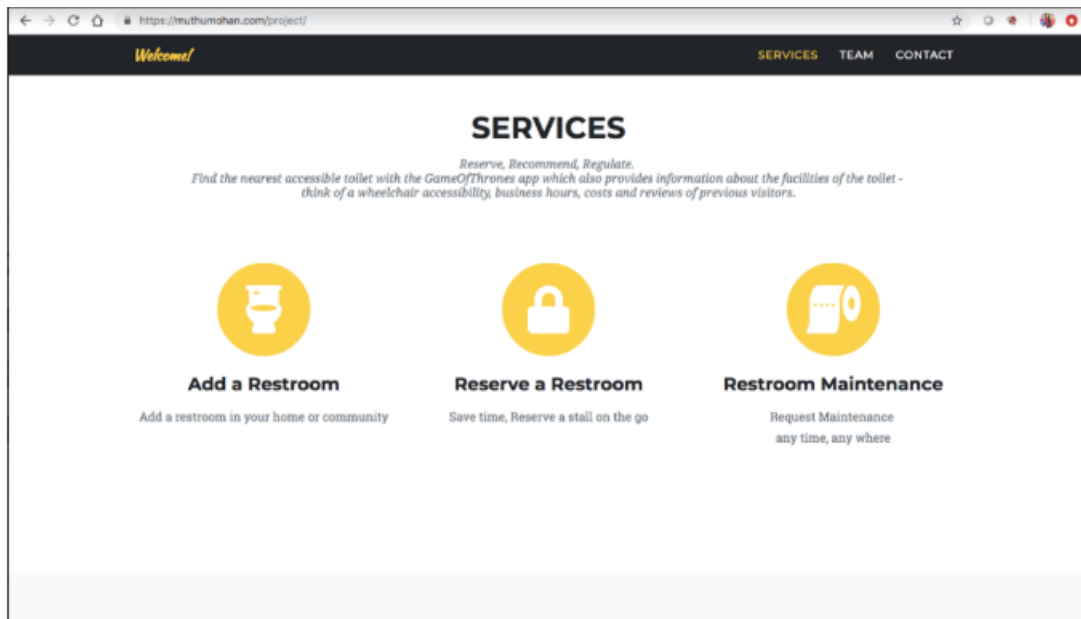


Fig. 3. Index Page

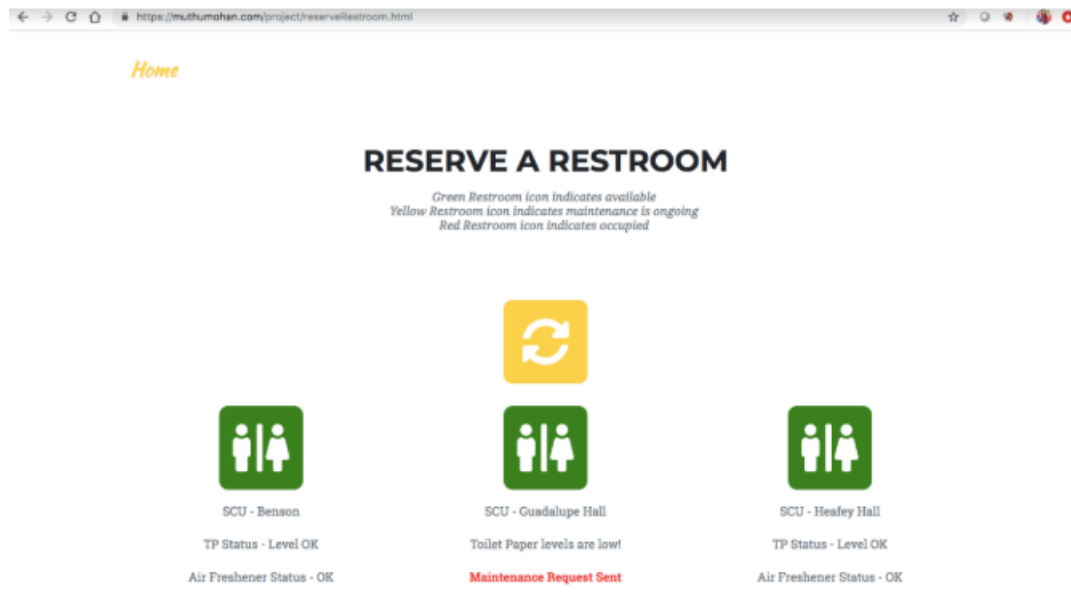


Fig. 4. Reserve a Restroom

A. Future Improvements

- Adding text notifications and an email service to notify when it is the user's turn
- Statistical Reporting for maintenance view to see how often cleaning is needed, how many toilet paper requests were served, and also monitoring employees' performance.
- We can add database for users and passwords for each one of them.
- RFIDs for each users that can be used after booking a stall.
- Add an expiry time to each stall booking, if the person who booked the stall does not show up for some prede-

fined time.




VII. REFERENCES



- How to setup interrupts: <https://techtutorialsx.com/2017/09/30/esp32-arduino-external-interrupts/>
- Debounce circuit: <https://www.arduino.cc/en/tutorial/debounce>
- OLED Drivers: <https://startingelectronics.org/tutorials/arduino/modules/OLED-128x64-I2C-display/https://learn.adafruit.com/monochrome-oled-breakouts/arduino-library-and-examples>
- I2C Setup: <https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/i2c.html>

ENTER YOUR DETAILS

UserName

ADD

SCU - Guadalupe Hall


Toilet Paper levels are low!

Toilet Paper request sent

1. Stall reserved For: Julio
2. Stall reserved For: Diana
3. Stall reserved For: empty

Fig. 5. Entering User Details

Home



SCU - Benson

Clean Status: All Good



Toilet Paper Levels: All Good

Maintenance Status: All Good

Clear TP Request

Clear Clean Request

Clear Maintenance Request

SCU - Guadalupe Hall

Clean Status: All Good


Toilet Paper Levels: Needs Replacement!

Maintenance Status: All Good

Clear TP Request

Clear Clean Request

Clear Maintenance Request



SCU - Heafey Hall

Clean Status: All Good

Toilet Paper Levels: All Good

Maintenance Status: All Good

Clear TP Request

Clear Clean Request

Clear Maintenance Request

Fig. 6. Maintenance

- Servo Driver: <https://randomnerdtutorials.com/esp32-servo-motor-web-server-arduino-ide/>
- Hackster Toilet Management System: <https://www.hackster.io/taifur/toilet-management-system-8e2786>
- Good2Go Toilet Management System: <https://www.good2go.global/aboutus>

VIII. APPENDIX

A. Product value link

<https://drive.google.com/open?id=1CHd-qagw-uIB4oKKObaetelcjtIWI8t->

B. Demo Video Link

<https://photos.app.goo.gl/CU4kvdcWikCpSy42A>

C. Hardware Inventory

HiLetgo ESP32

<https://www.amazon.com/HiLetgo-ESP-WROOM-32-Development-Microcontroller-Integrated/dp/B0718T232Z>

Diymall 128x64 OLED LCD

<https://www.amazon.com/gp/product/B00O2LLT30>

TowerPro MG995 Digital Servo

<https://www.amazon.com/REES52-MG995-TowerPro-Digital-Servo/dp/B0156059W2>

Reed Hall Sensor

<https://www.ebay.com/itm/5-12V-Door-Magnet-Relay-Module-Normally-Closed-Reed-Control-Switch-For-Arduino/401466037070>

D. Software Inventory

1) Backend:

- NodeJS - Used to get IoT updates, backbone of IoT Server
- PHP - used to send GET requests to IoT server
- avascript - Used for Document Object Manipulation

2) Web and Mobile:

- HTML5
- Bootstrap
- Javascript
- Phonegap

E. Execution Instructions

Start IoT server using nodeJS On the Amazon EC2 server
 run:Npm install expressNpm initNode backend_server.jsStart
 Front end PHP ServerSetup access point for the IoT devices.
 Configure the routerTurn on IoT devicesPlug in the power
 Open mobile applicationBook a bathroom stall through the application

F. Arduino Code Compilation

GoT Bathroom Demo Arduino Source Code

To compile, select device mode:

-DEVICE_BD Unit

-DEVICE_TP Unit

-DEVICE_AF Unit

Set selected device mode to 1, set others to 0.

Set wifi mode to WIFI_MODE_CLIENT = 1 for testing

Ensure that the client SSID and client password credentials are correct

Connect the ESP32 target board

Run the compile command

Repeat the step for every IoT