**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 21/2/2021 |

|  |  |
| --- | --- |
| **Prepared for:** | Dora Chua |

|  |  |
| --- | --- |
| **Class:** | 3A/41 |

|  |  |
| --- | --- |
| **Submitted by:** |  |

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1828290 | Bryon Kaan |
| 1828331 | Dextor Goh |

**IOT CA2**

**Step-by-step Tutorial**

**DIPLOMA IN BUSINESS INFORMATION TECHNOLOGY**

**DIPLOMA IN INFORMATION TECHNOLOGY**

**DIPLOMA IN INFOCOMM SECURITY MANAGEMENT**

**ST0324 Internet of Things (IOT)**

**2017/2018 Semester 1**

**Table of Contents**

[Section 1 Overview of project 3](#_Toc64817644)

[A. Where we have uploaded our tutorial 3](#_Toc64817645)

[B. What is the application about? 3](#_Toc64817646)

[C. How does the final RPI set-up looks like? 4](#_Toc64817647)

[D. How does the web or mobile application look like? 6](#_Toc64817648)

[E. System architecture of our system 9](#_Toc64817649)

[F. Evidence that we have met basic requirements 10](#_Toc64817650)

[G. Bonus features on top of basic requirements 11](#_Toc64817651)

[A. Quick-start guide (Readme first) 11](#_Toc64817652)

[Section 2 Hardware requirements 12](#_Toc64817653)

[Hardware checklist 12](#_Toc64817654)

[Hardware setup instructions 12](#_Toc64817655)

[Fritzing Diagram 13](#_Toc64817656)

[Section 3 Software Requirements 14](#_Toc64817657)

[Software checklist 14](#_Toc64817658)

[Software setup instructions 15](#_Toc64817659)

[Section 4 Source codes 15](#_Toc64817660)

[run.py 15](#_Toc64817661)

[app/\_\_init\_\_.py 16](#_Toc64817662)

[app/base/routes.py 16](#_Toc64817663)

[app/base/templates/login.html 20](#_Toc64817664)

[app/base/templates/register.html 22](#_Toc64817665)

[app/base/templates/layouts/base.html 24](#_Toc64817666)

[app/base/templates/layouts/base-fullscreen.html 27](#_Toc64817667)

[app/home/routes.py 29](#_Toc64817668)

[app/home/templates/camera.html 33](#_Toc64817669)

[app/home/templates/index.html 38](#_Toc64817670)

[app/home/templates/pump.html 43](#_Toc64817671)

[pubsub.py 47](#_Toc64817672)

[text\_rekognition.py 49](#_Toc64817673)

[checkbowl.py 50](#_Toc64817674)

[store\_sens.py 50](#_Toc64817675)

[Section 5 Task List 53](#_Toc64817676)

[Section 6 References 53](#_Toc64817677)

# Section 1 Overview of project

* 1. Where we have uploaded our tutorial

Fill up the Google form here to submit your links and then paste the links here of your Youtube and tutorial document here as well.

<http://bit.ly/1910s2iotca2>

|  |  |
| --- | --- |
| **Youtube** | https://youtu.be/uHTbP17fby4 |
| **Public tutorial link** | https://github.com/Zerolegacy/IotCA2 |

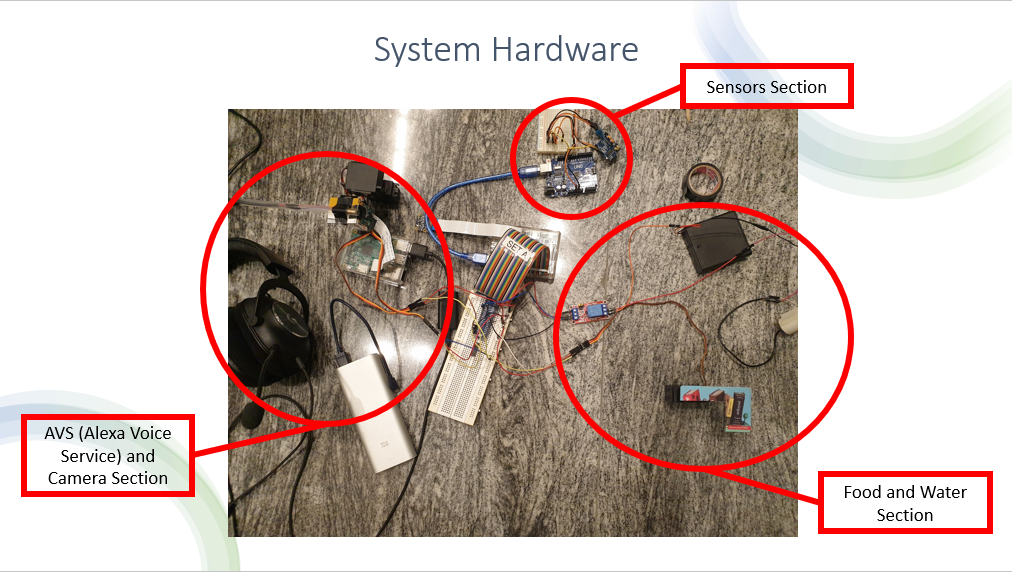
* 1. What is the application about?

Provide a brief description of your application here. Who is the target audience? How can your app help your target audience?

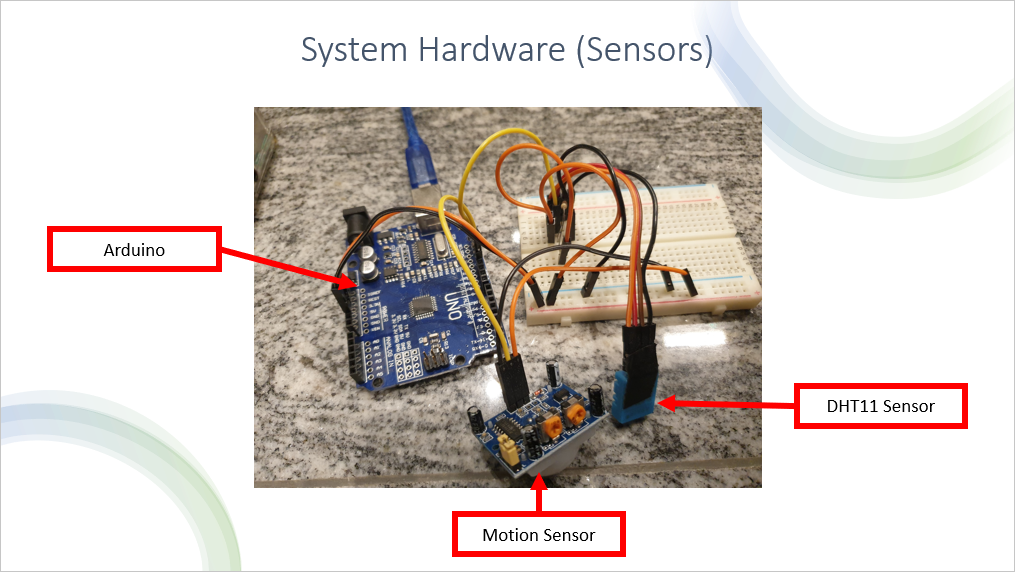
Our application is a remote pet feeder. Our target audience are any pet owners who want to be able to control the feeding of their pet and monitor their pets remotely. Our website is accessible from anywhere and from there, the user is able to watch the livestream, control the camera of livestream and look at other information such as temperature and motion. These features help the monitoring part of the target audience’s needs. The app is also able to control the mechanisms in the feeder to dispense food and refill water into the pet’s food and water bowl.

* 1. How does the final RPI set-up looks like?

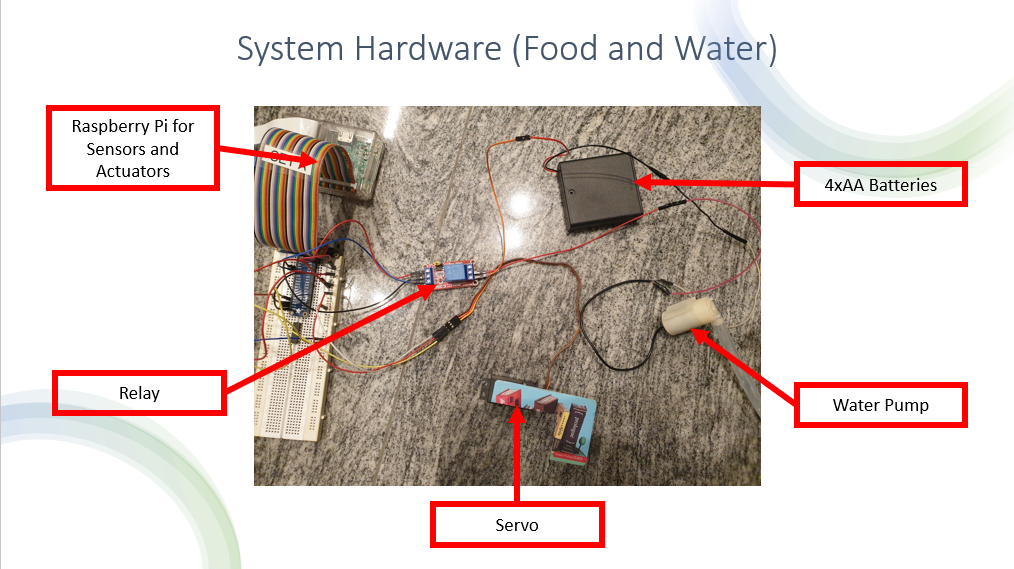
Provide a photo of your final RPI hardware set-up. You may want to mark-up (annotate or draw arrows) and refer to this in Section F for instance.



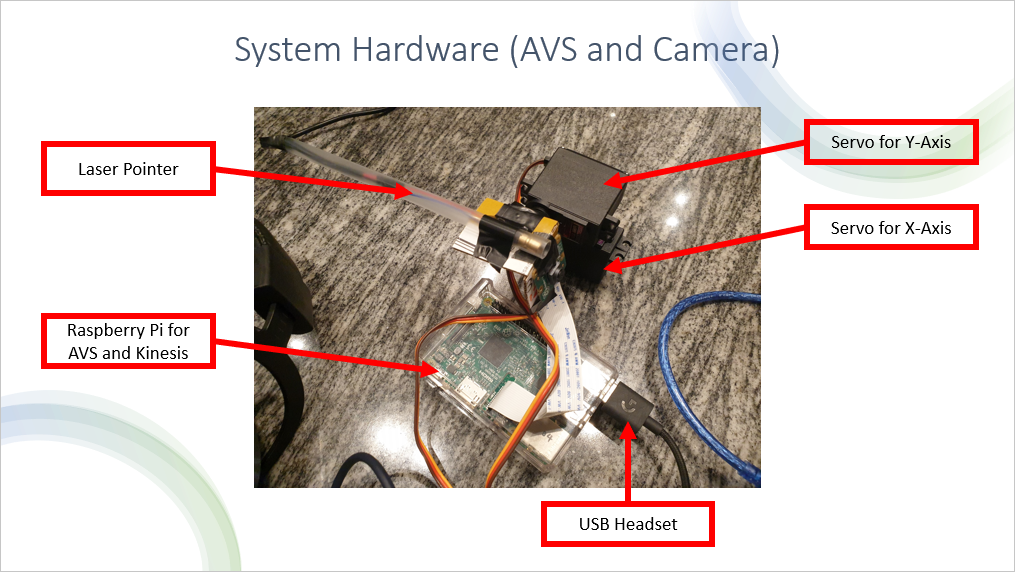
**Figure A: System Hardware**



**Figure B: System Hardware (Sensors)**

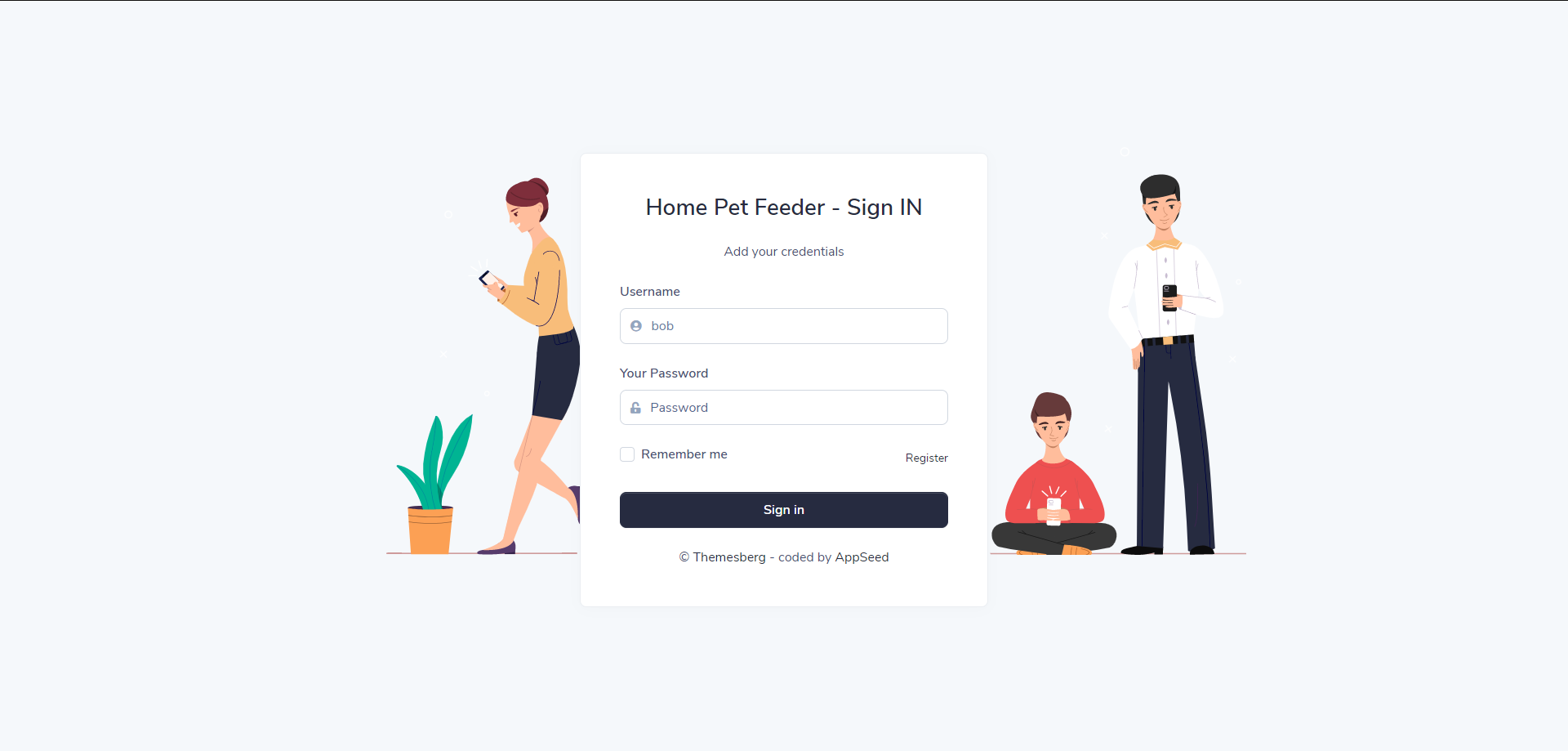


**Figure C: System Hardware (Food and Water)**

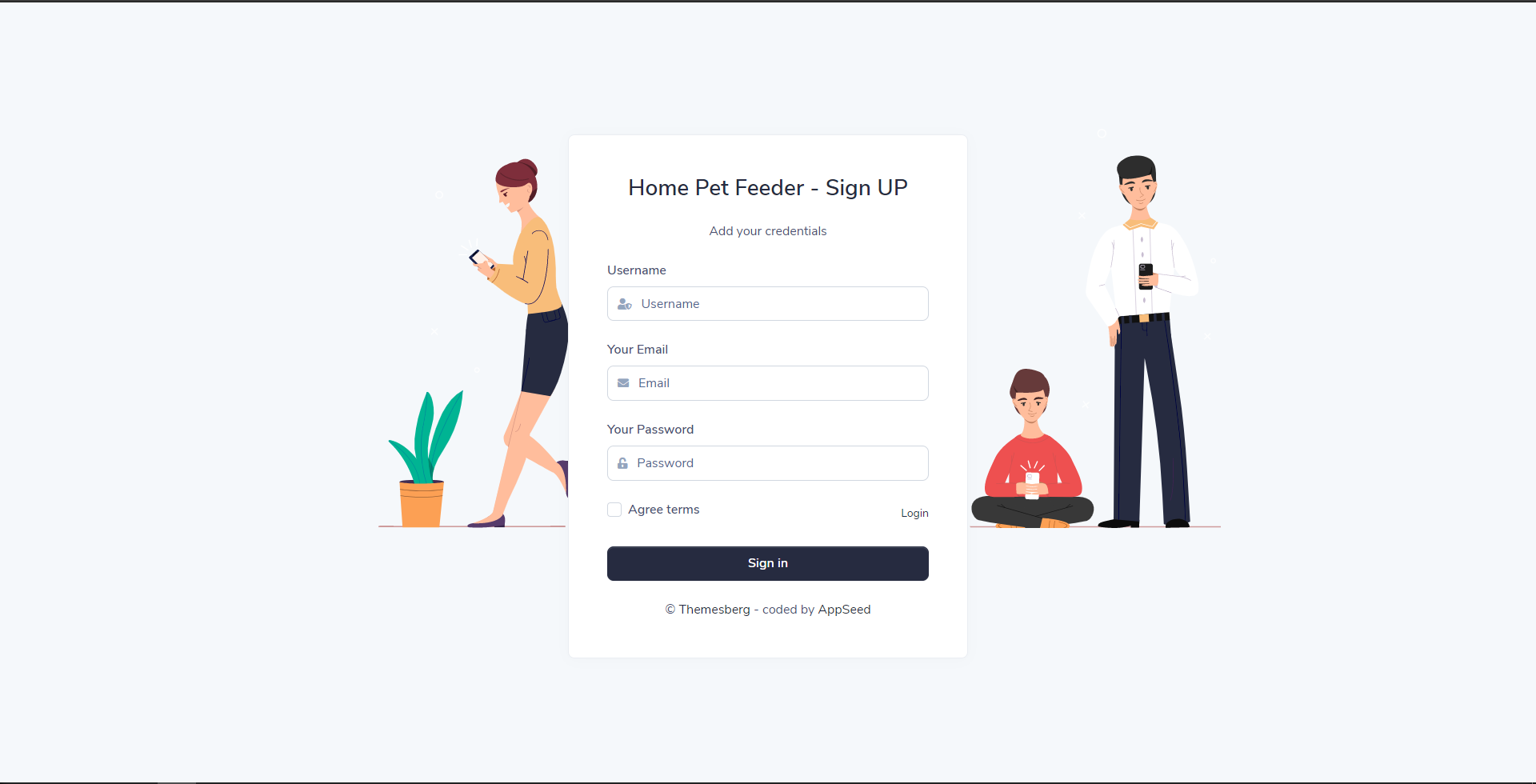
  
**Figure D: System Hardware (AVS and Camera)**

* 1. How does the web or mobile application look like?

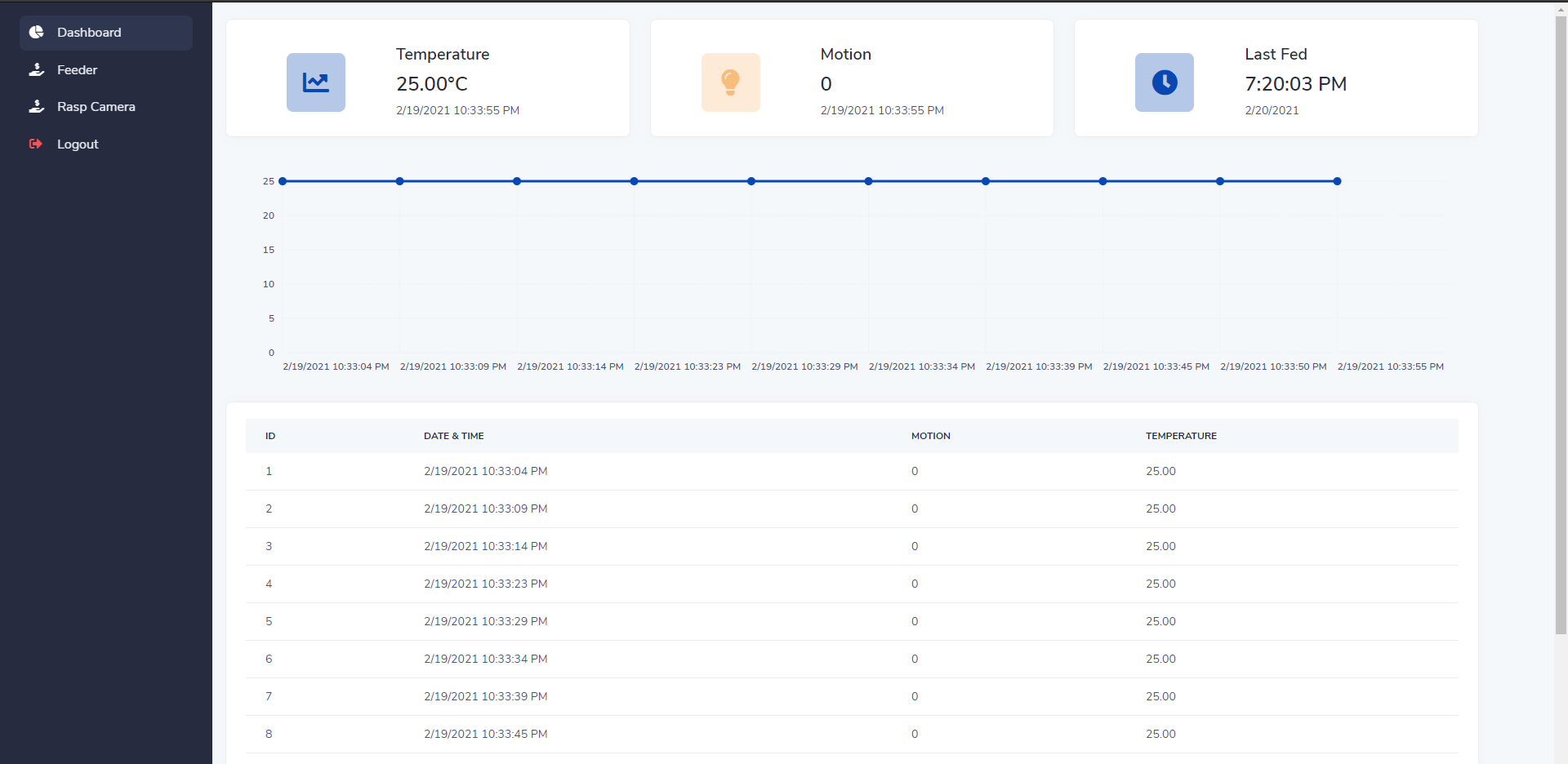
Provide at least one screenshot of your web app, and more if your web app consists of more than 1 page. Otherwise, I will assume your webapp only can show 1 page. Label your screenshots so that they may be referenced in Section F.



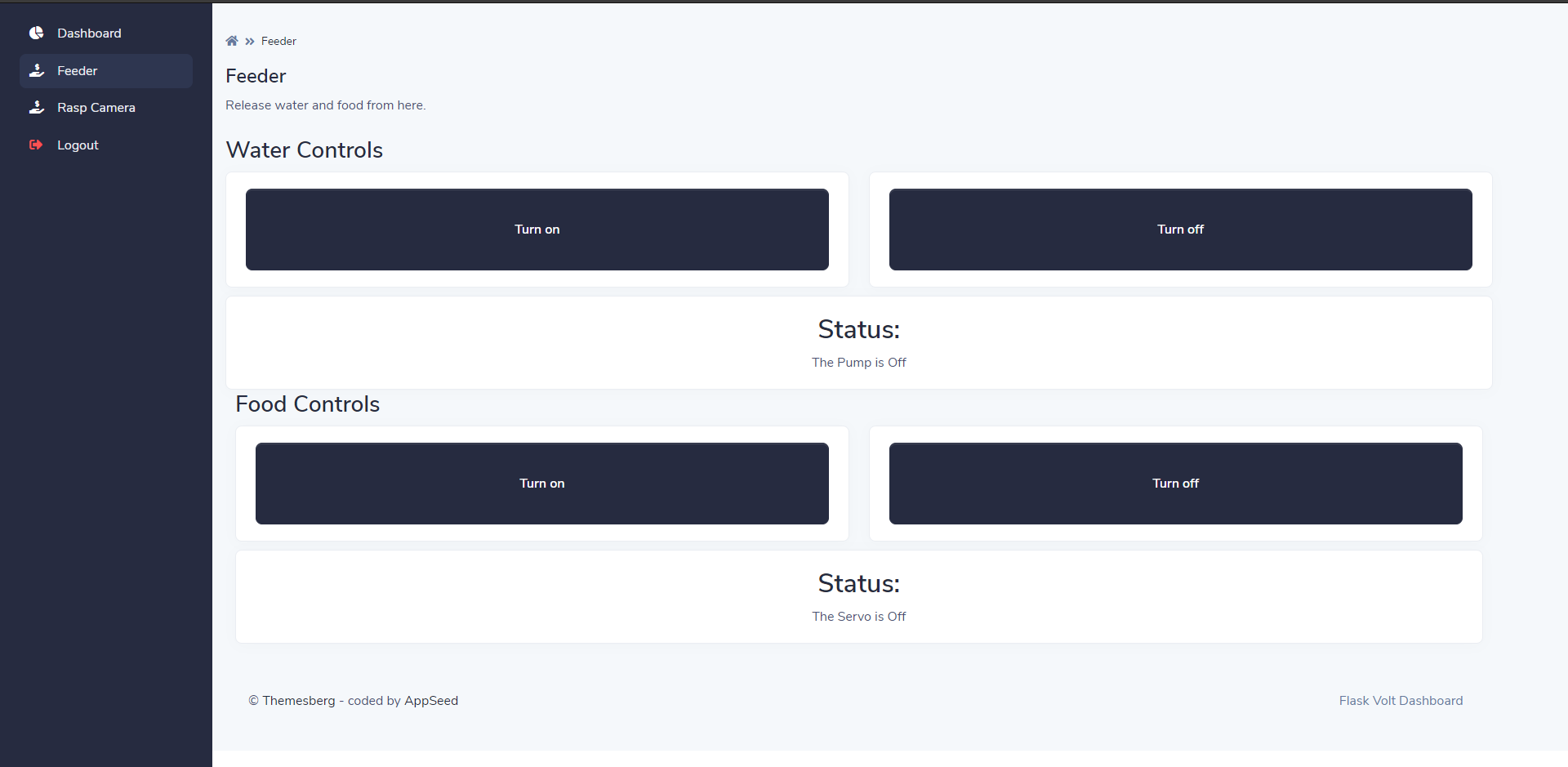
**Figure E: Login Page**



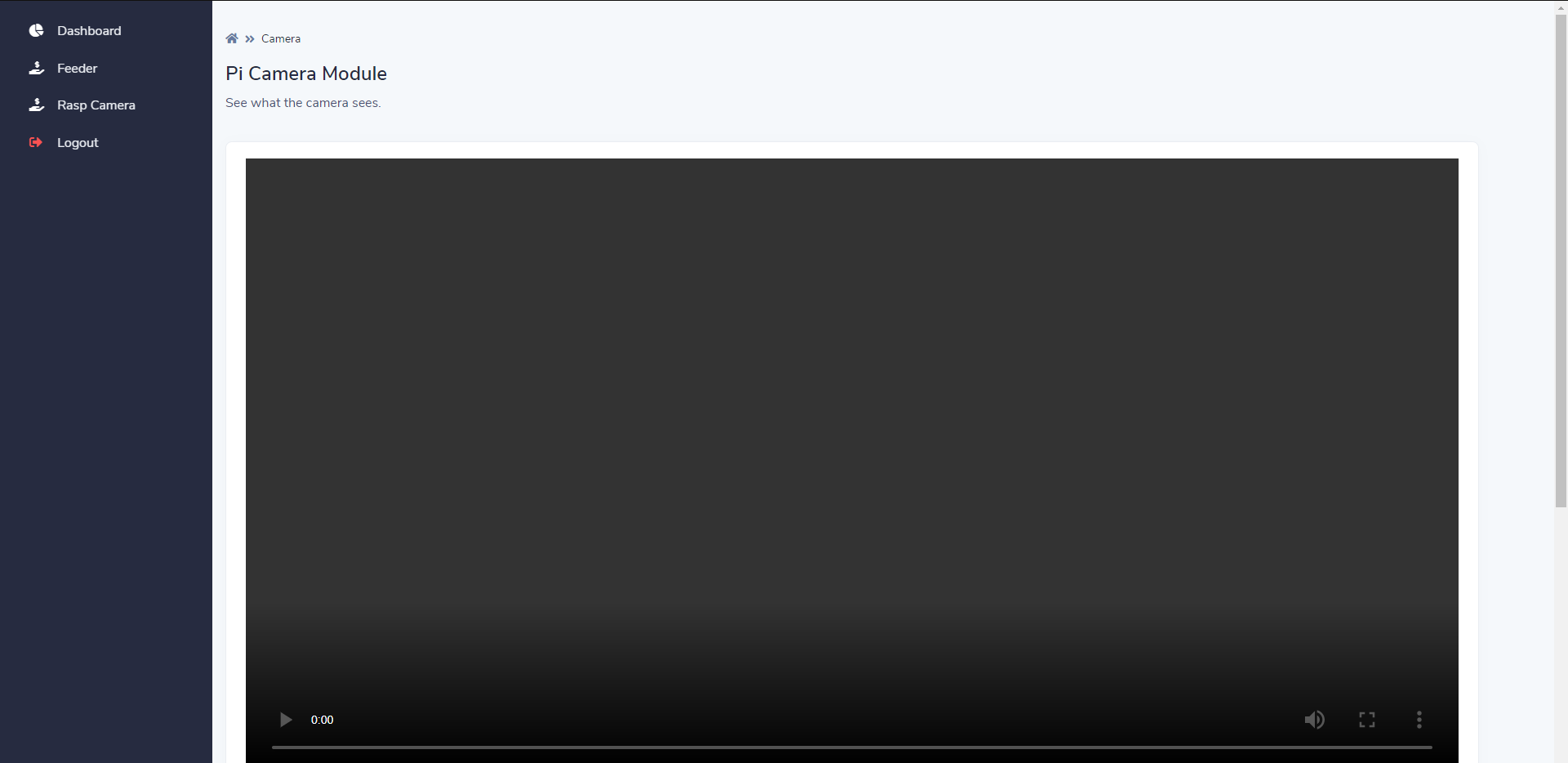
**Figure F: Sign Up Page**



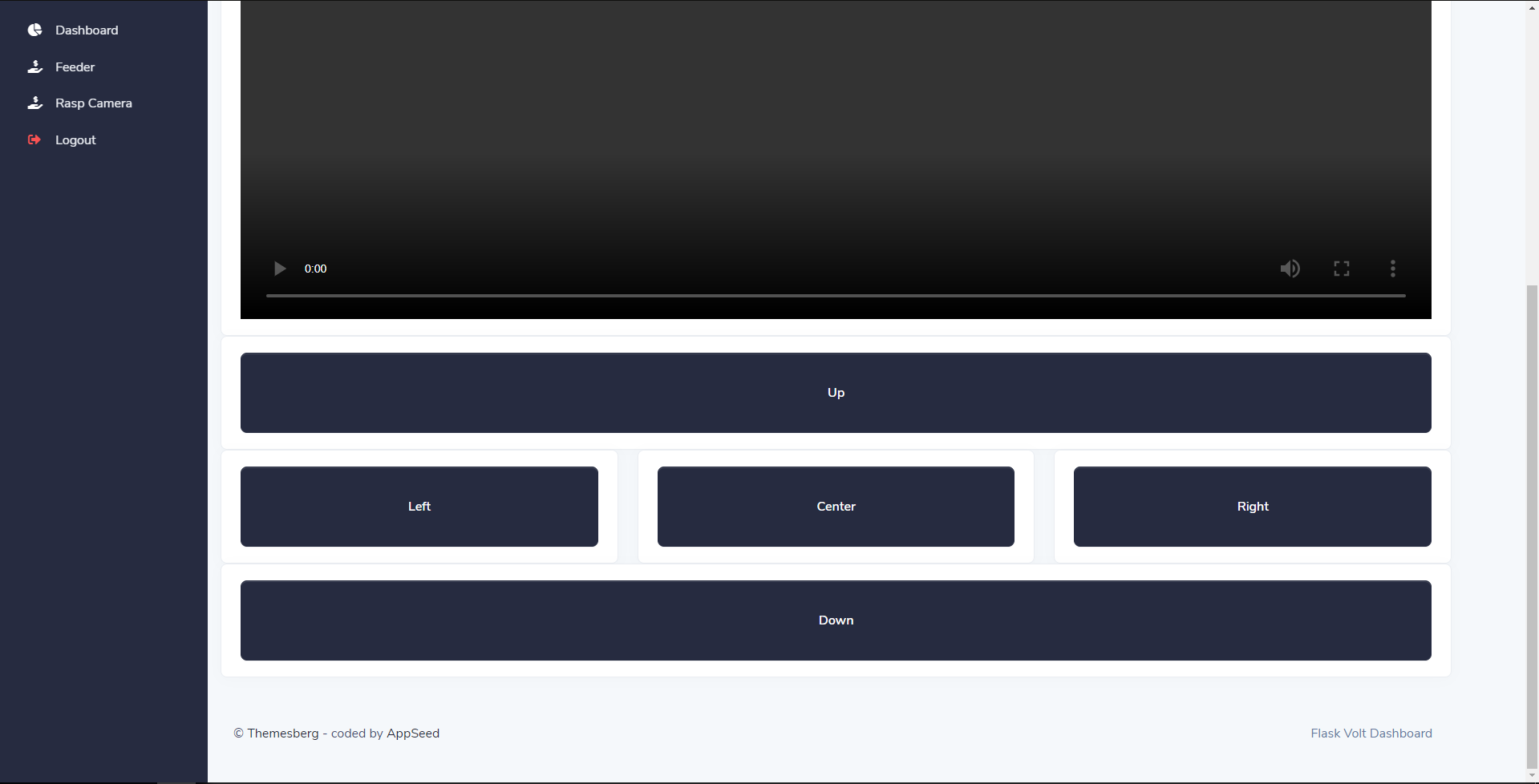
**Figure G: Dashboard Page**



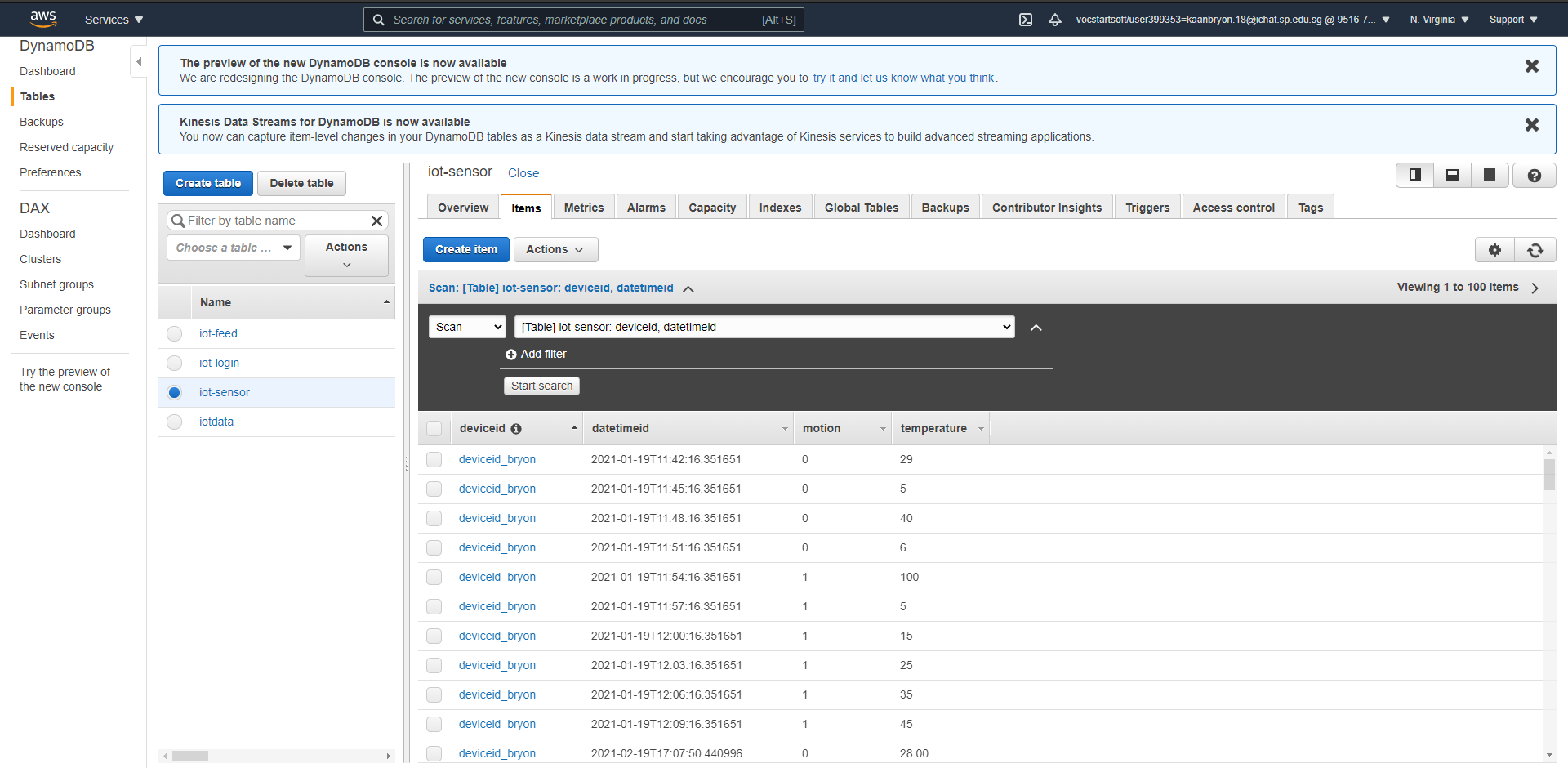
**Figure H: Feeder Page**



**Figure I: Web Stream Page (stream)**



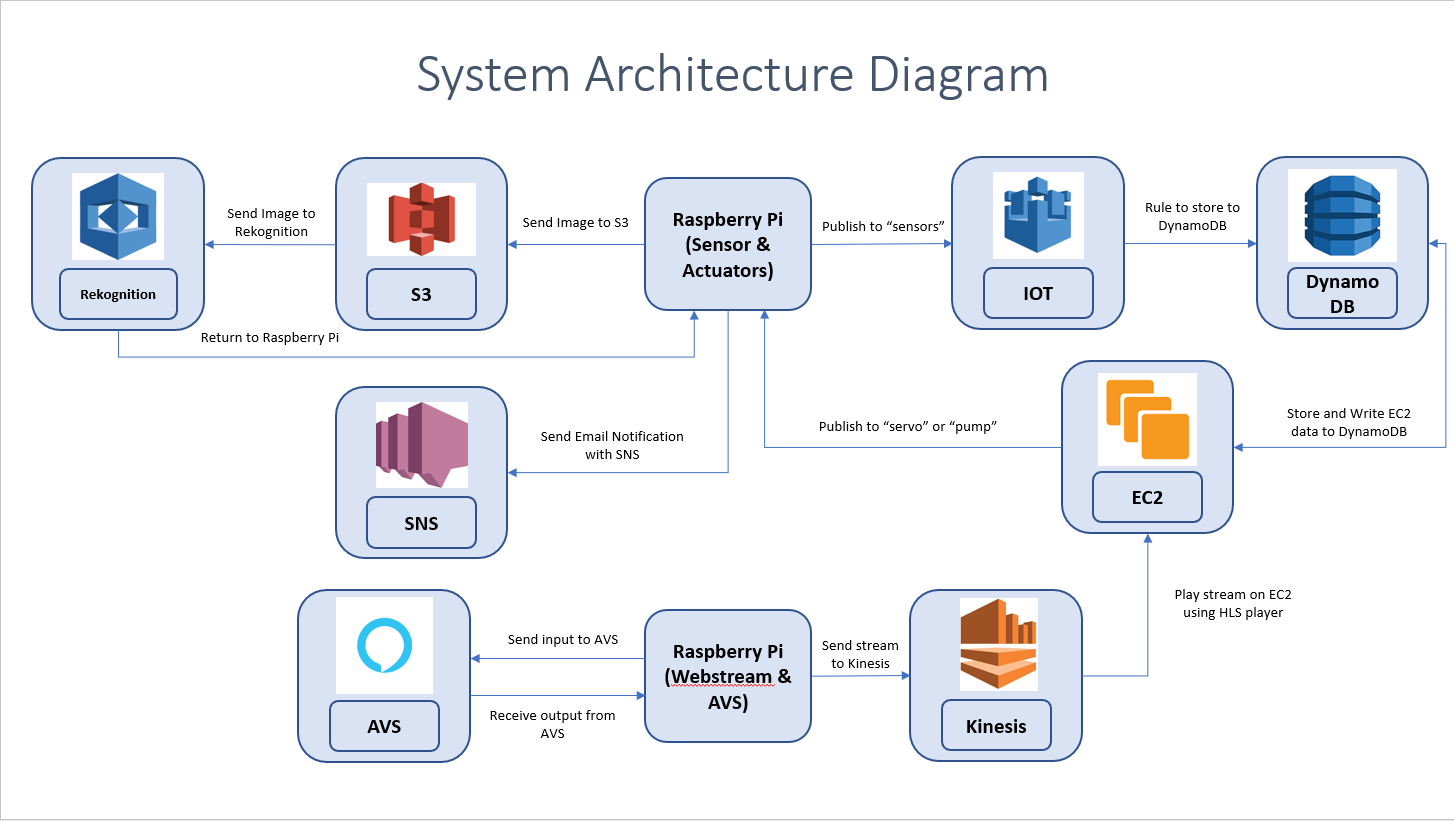
**Figure J: Web Stream Page (camera controls)**

****

**Figure K: Dynamodb**

* 1. System architecture of our system

Provide a hand-drawn or computer-drawn system architecture diagram please. Example given below.



* 1. Evidence that we have met basic requirements

Provide bullet list to describe how your group has met basic requirements

|  |  |
| --- | --- |
| Requirement | Evidence |
| Used three sensors | Used Servos, DHT11, Motion Sensor, Laser Diode, PiCam and Water Pump. See Figure A, B, C and D |
| Used MQTT | Our MQTT endpoint --> sensor, servo, servoX, servoY, pump, feed and bowlalert  Example of data sent through MQTT :  Sensor: DHT11 Temperature value, Motion Sensor value, Time and Deviceid.  Servo: Servo value  ServoX: Servo value  ServoY: Servo value  Pump: Pump on off value  Feed: Time  Bowlalert: Email message  See Source Codes. |
| Stored data in cloud | Stored login information, last fed time, DHT11 temperature and motion sensor values in DynamoDB in AWS. See Figure K |
| Used cloud service | Use AWS Rekognition, hosted web server on EC2, AWS SNS for email alerts, AWS S3 and Kinesis for Web Stream, AWS AVS for Alexa. AWS MQTT for publish and subscribing to topics. Dynamodb for storing data.  See Figure K and Source Codes. |
| Provide real-time sensor value / status | Show the real-time value and status of DHT11 Temperature, Motion Sensor, Servo and Water Pump. See Figure G and H. |
| Provide historical sensor value/ status | Show the historical value and status of DHT11 Temperature. See Figure G. |
| Control actuator | Placed button on webpage to control servo and water pump. See Figure H. |

* 1. Bonus features on top of basic requirements

Provide bullet list of the bonus features you have added on top of basic requirements

1. Log in system with secure password storage on dynamodb.
2. Live Web Stream using AWS Kinesis.
3. Integrated Alexa into RaspberryPi using AWS Alexa Voice Service.
4. Use of Amazon SNS for Email Notification
5. Use of Rekognition for text identification
   1. Quick-start guide (Readme first)

Give a few lines of basic instructions on how I need to run your app, e.g

1. First connect hardware as in Section 2
2. Install all dependencies as in Section 3
3. For Alexa and Live Stream, one of the RaspberryPis has to be Version 10 and the appropriate setting up is required.
4. Git clone <https://github.com/Zerolegacy/IotCA2>
5. Transfer the website folder to your ec2 instance
6. Install dependencies by running apt-get install -r requirements.txt
7. Update the aws credentials in **~/.aws/credentials** **in ec2 and both RaspberryPi**
8. Then run the run.py file for the web server. Run.py can be found in the website folder
9. Transfer the RaspberryPi folder to your RaspberryPi(non-Version 10).
10. Update **certificate.pem.crt**, **private.pem.key**, **public.pem.key**, **rootca.pem in website folder** with your own certs and details.
11. Update **certificate.pem.crt**, **private.pem.key**, **public.pem.key**, **rootca.pem** **in RaspberryPi folder** with your own certs and details.
12. Update your Endpoint API link inside **store\_sens.py**, **text\_rekognition.py** and **pubsub.py**.
13. Run python3 store\_sens.py to start getting values from the sensors to the pi.
14. Run python checkbowl.py to start monitor if the foodbowl is empty.
15. Run python pubsub.py to start listening to topics with MQTT for actuator commands.
16. Install Amazon Kinesis following <https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp>
17. Create a credential file with the format “CREDENTIALS <aws\_access\_key> <session to end> <aws\_secret\_access\_key> <aws\_session\_token>”
18. In the other RaspberryPi (Version 10), run the following command for livestream to start:

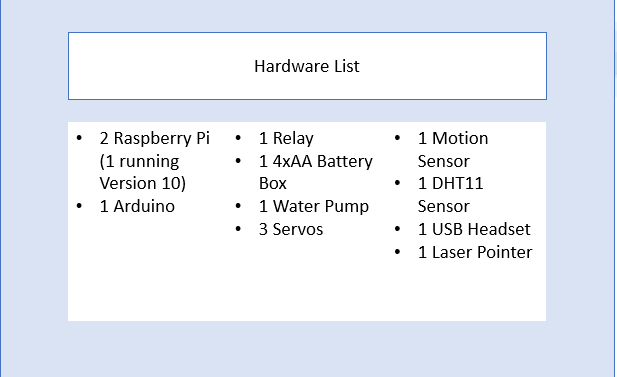
*gst-launch-1/0 v412src do-timestamp=TRUE device=/dev/video0 ! videoconvert ! video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! omxh264enc control-rate=1 target-bitrate=5120000 periodicity-idr=45 inline-header=FALSE ! h264parse ! Video/x-h264,stream-format=avc,alignment=au,width=640,height=480,framerate=30/1,profile=baseline ! Kvssink stream-name=”<stream name>” credential-path=”<credentials file path from step 16>” aws-region=”us-east-1*

1. Integrate Alexa using the docummentations from the following: <https://developer.amazon.com/en-US/docs/alexa/avs-device-sdk/raspberry-pi.html>
2. In the other RaspberryPi (Version 10), run the following command for Alexa to start:

*cd /home/pi/sdk-folder/sdk-build PA\_ALSA\_PLUGHW=1 ./SampleApp/src/SampleApp ./Integration/AlexaClientSDKConfig.json ../third-party/alexa-rpi/models*

# Section 2 Hardware requirements

Hardware checklist

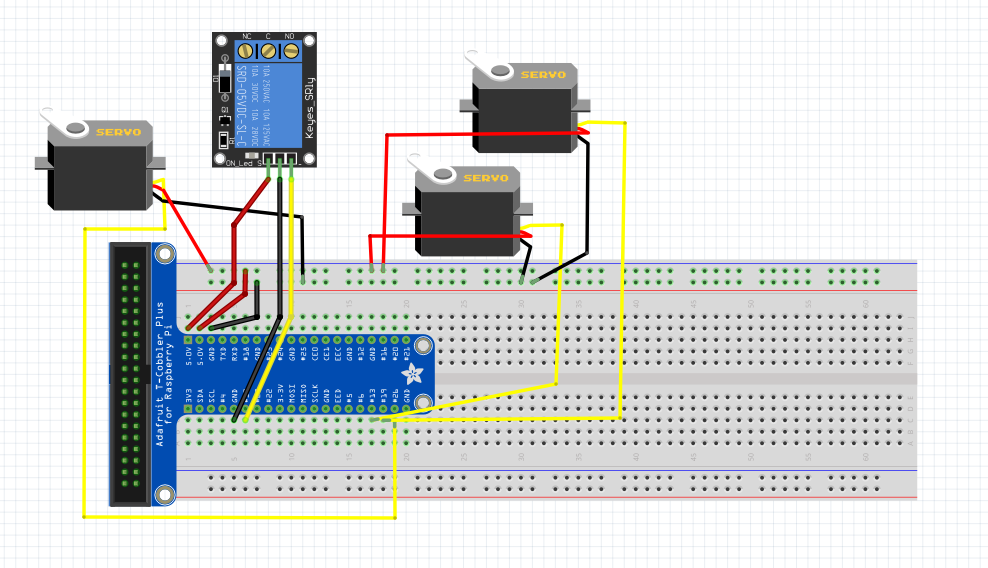


Hardware setup instructions

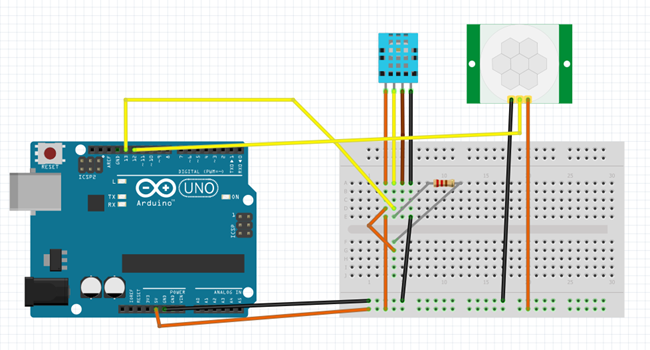
Describe any special setup instructions here

* + 1. One of the RaspberryPi has to be upgraded to Version 10.
    2. For controllable camera to work, a servo(ServoX) has to be connected to another servo(ServoY). Afterwards, the picam(live stream) and the laser diode has to be connected to them as well. See Figure D for how it is connected.
    3. Picam(text\_rekognition) has to be placed where it can see the foodbowl with the paper reading “EMPTY”.
    4. USB Headset connected to RaspberryPi running Version10 for Alexa.

Fritzing Diagram



**(Raspberry Pi)**



**(Arduino)**

# Section 3 Software Requirements

Software checklist

If your applications needs the user to install additional Python or other libraries, pleasse provide here. A simple one like this is sufficient.

Pip installed Libraries:

1. flask

2. flask\_login

3. flask\_migrate

4. flask\_wtf

5. flask\_session

6. flask\_mysqldb

7. email\_validator

8. python-decouple

9. gunicorn

10. numpy

11. botocore3

12. AWSIoTPythonSDK.MQTTLib

Apt-get installed libraries:

1. python-opencv

2. cmake

3. gcc

4. g++

5. libssl-dev

6. libcurl4-openssl-dev

7. liblog4cplus-dev

8. libgstreamer1.0-dev

9. libgstreamer-plugins-base1.0-dev

10. gstreamer1.0-plugins-base-apps

11. gstreamer1.0-plugins-bad

12. gstreamer1.0-plugins-good

13. gstreamer1.0-plugins-ugly

14. gstreamer1.0-tools

Arduino libraries:

1. Adafruit\_Unified\_Sensor

2. DHT\_sensor\_library

Software setup instructions

Describe any special setup instructions here, e.g some libraries you need to pip install or some API key you need to create/request etc

# Section 4 Source codes

All source codes, including Python, HTML files etc

### run.py

# -\*- encoding: utf-8 -\*-

"""

Copyright (c) 2019 - present AppSeed.us

"""

from flask\_migrate import Migrate

from os import environ

from sys import exit

from decouple import config

from config import config\_dict

from app import create\_app

# WARNING: Don't run with debug turned on in production!

DEBUG = config('DEBUG', default=True)

# The configuration

get\_config\_mode = 'Debug' if DEBUG else 'Production'

try:

# Load the configuration using the default values

app\_config = config\_dict[get\_config\_mode.capitalize()]

except KeyError:

exit('Error: Invalid <config\_mode>. Expected values [Debug, Production] ')

app = create\_app( app\_config )

Migrate(app)

if \_\_name\_\_ == "\_\_main\_\_":

app.run('0.0.0.0',8080, threaded=True, debug=False)

### app/\_\_init\_\_.py

# -\*- encoding: utf-8 -\*-

"""

Copyright (c) 2019 - present AppSeed.us

"""

from flask import Flask, url\_for, session

from flask\_session import Session

from flask\_login import LoginManager

from importlib import import\_module

from logging import basicConfig, DEBUG, getLogger, StreamHandler

from os import path

login\_manager = LoginManager()

def register\_extensions(app):

login\_manager.init\_app(app)

def register\_blueprints(app):

for module\_name in ('base', 'home'):

module = import\_module('app.{}.routes'.format(module\_name))

app.register\_blueprint(module.blueprint)

def create\_app(config):

app = Flask(\_\_name\_\_, static\_folder='base/static')

app.config.from\_object(config)

app.secret\_key = 'super secret key'

app.config['SESSION\_TYPE'] = 'filesystem'

register\_extensions(app)

register\_blueprints(app)

sess = Session()

sess.init\_app(app)

return app

### app/base/routes.py

# -\*- encoding: utf-8 -\*-

"""

Copyright (c) 2019 - present AppSeed.us

"""

from flask import jsonify, render\_template, redirect, request, url\_for

from flask\_login import (

current\_user,

login\_required,

login\_user,

logout\_user

)

from flask\_wtf import FlaskForm

from wtforms import TextField, PasswordField

from wtforms.validators import InputRequired, Email, DataRequired

from app.base import blueprint

import hashlib, binascii, os

from flask\_login import UserMixin

from app import login\_manager

import boto3

from boto3.dynamodb.conditions import Key, Attr

## login and registration

class LoginForm(FlaskForm):

username = TextField ('Username', id='username\_login' , validators=[DataRequired()])

password = PasswordField('Password', id='pwd\_login' , validators=[DataRequired()])

class CreateAccountForm(FlaskForm):

username = TextField('Username' , id='username\_create' , validators=[DataRequired()])

email = TextField('Email' , id='email\_create' , validators=[DataRequired(), Email()])

password = PasswordField('Password' , id='pwd\_create' , validators=[DataRequired()])

def hash\_pass( password ):

"""Hash a password for storing."""

salt = hashlib.sha256(os.urandom(60)).hexdigest().encode('ascii')

pwdhash = hashlib.pbkdf2\_hmac('sha512', password.encode('utf-8'),

salt, 100000)

pwdhash = binascii.hexlify(pwdhash)

return binascii.b2a\_base64((salt + pwdhash),newline=False) # return bytes

def verify\_pass(provided\_password, stored\_password):

stored\_password = stored\_password[2:-1]

"""Verify a stored password against one provided by user"""

stored\_password = binascii.a2b\_base64(stored\_password).decode('ascii')

salt = stored\_password[:64]

stored\_password = stored\_password[64:]

print(stored\_password)

pwdhash = hashlib.pbkdf2\_hmac('sha512',

provided\_password.encode('utf-8'),

salt.encode('ascii'),

100000)

pwdhash = binascii.hexlify(pwdhash).decode('ascii')

print(pwdhash)

return pwdhash == stored\_password

class User(UserMixin):

def \_\_init\_\_(self, \*\*kwargs):

for property, value in kwargs.items():

# depending on whether value is an iterable or not, we must

# unpack it's value (when \*\*kwargs is request.form, some values

# will be a 1-element list)

setattr(self, property, value)

def get(username):

dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

table = dynamodb.Table('iot-login')

response = table.query(KeyConditionExpression=Key('username').eq(username))

if response:

items = response['Items']

if items:

if username==items[0]['username']:

user = User(username=username,password=items[0]['password'])

if user:

return user

else:

return None

def get\_id(self):

return self.username

@login\_manager.user\_loader

def user\_loader(username):

return User.get(username=username)

@blueprint.route('/')

def route\_default():

return redirect(url\_for('base\_blueprint.login'))

## Login & Registration

@blueprint.route('/login', methods=['GET', 'POST'])

def login():

login\_form = LoginForm(request.form)

if 'login' in request.form:

# read form data

username = request.form['username']

password = request.form['password']

# Locate user

user = User.get(username)

# Check the password

if user and verify\_pass(password, user.password):

login\_user(user)

return redirect(url\_for('base\_blueprint.route\_default'))

# Something (user or pass) is not ok

return render\_template( 'login.html', msg='Wrong user or password', form=login\_form)

if not current\_user.is\_authenticated:

return render\_template( 'login.html',

form=login\_form)

return redirect(url\_for('home\_blueprint.index'))

@blueprint.route('/register', methods=['GET', 'POST'])

def register():

login\_form = LoginForm(request.form)

create\_account\_form = CreateAccountForm(request.form)

if 'register' in request.form:

username = request.form['username']

email = request.form['email' ]

password = request.form['password']

dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

table = dynamodb.Table('iot-login')

response = table.put\_item(

Item={

'username': username,

'email': email,

'password': str(hash\_pass(password))

})

return render\_template( 'register.html',

msg='User created please <a href="/login">login</a>',

success=True,

form=create\_account\_form)

else:

return render\_template( 'register.html', form=create\_account\_form)

@blueprint.route('/logout')

def logout():

logout\_user()

return redirect(url\_for('base\_blueprint.login'))

@blueprint.route('/shutdown')

def shutdown():

func = request.environ.get('werkzeug.server.shutdown')

if func is None:

raise RuntimeError('Not running with the Werkzeug Server')

func()

return 'Server shutting down...'

### app/base/templates/login.html

{% extends "layouts/base-fullscreen.html" %}

{% block title %} Login {% endblock %}

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

{% block body\_class %} bg-soft {% endblock body\_class %}

{% block content %}

<main>

<!-- Section -->

<section class="vh-lg-100 d-flex align-items-center">

<div class="container">

<div class="row justify-content-center form-bg-image" data-background="/static/assets/img/illustrations/signin.svg">

<div class="col-12 d-flex align-items-center justify-content-center">

<div class="signin-inner my-3 my-lg-0 bg-white shadow-soft border rounded border-light p-4 p-lg-5 w-100 fmxw-500">

<div class="text-center text-md-center mb-4 mt-md-0">

<h1 class="mb-0 h3">

<a target="\_blank"

href="https://appseed.us/admin-dashboards/flask-dashboard-volt">Home Pet Feeder</a> - Sign IN

</h1>

<br />

<p>

{% if msg %}

{{ msg | safe }}

{% else %}

Add your credentials

{% endif %}

</p>

</div>

<form method="post" action="" class="mt-4">

{{ form.hidden\_tag() }}

<!-- Form -->

<div class="form-group mb-4">

<label for="email">Username</label>

<div class="input-group">

<span class="input-group-text" id="basic-addon1"><span class="fas fa-user-circle"></span></span>

{{ form.username(placeholder="Username", class="form-control") }}

</div>

</div>

<!-- End of Form -->

<div class="form-group">

<!-- Form -->

<div class="form-group mb-4">

<label for="password">Your Password</label>

<div class="input-group">

<span class="input-group-text" id="basic-addon2"><span class="fas fa-unlock-alt"></span></span>

{{ form.password(placeholder="Password", class="form-control", type="password") }}

</div>

</div>

<!-- End of Form -->

<div class="d-flex justify-content-between align-items-center mb-4">

<div class="form-check">

<input class="form-check-input" type="checkbox" value="" id="defaultCheck5">

<label class="form-check-label" for="defaultCheck5">

Remember me

</label>

</div>

<div class="form-check">

<a href="{{ url\_for('base\_blueprint.register') }}" class="small text-right">Register</a>

</div>

</div>

</div>

<button type="submit" name="login" class="btn btn-block btn-primary">Sign in</button>

</form>

<div class="d-flex justify-content-center align-items-center mt-4">

<span class="font-weight-normal">

&copy; <a href="https://themesberg.com" target="\_blank">Themesberg</a>

- coded by <a target="\_blank" href="https://appseed.us">AppSeed</a>

</span>

</div>

</div>

</div>

</div>

</div>

</section>

</main>

{% endblock content %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}{% endblock javascripts %}

### app/base/templates/register.html

{% extends "layouts/base-fullscreen.html" %}

{% block title %} Register {% endblock %}

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

{% block body\_class %} bg-soft {% endblock body\_class %}

{% block content %}

<main>

<!-- Section -->

<section class="vh-lg-100 d-flex align-items-center">

<div class="container">

<div class="row justify-content-center form-bg-image" data-background="/static/assets/img/illustrations/signin.svg">

<div class="col-12 d-flex align-items-center justify-content-center">

<div class="signin-inner my-3 my-lg-0 bg-white shadow-soft border rounded border-light p-4 p-lg-5 w-100 fmxw-500">

<div class="text-center text-md-center mb-4 mt-md-0">

<h1 class="mb-0 h3">

<a target="\_blank"

href="https://appseed.us/admin-dashboards/flask-dashboard-volt">Home Pet Feeder</a> - Sign UP

</h1>

<br />

<p>

{% if msg %}

{{ msg | safe }}

{% else %}

Add your credentials

{% endif %}

</p>

</div>

<form method="post" action="">

{{ form.hidden\_tag() }}

<!-- Form -->

<div class="form-group mb-4">

<label for="email">Username</label>

<div class="input-group">

<span class="input-group-text" id="basic-addon3"><span class="fas fa-user-shield"></span></span>

{{ form.username(placeholder="Username", class="form-control") }}

</div>

</div>

<!-- End of Form -->

<!-- Form -->

<div class="form-group mb-4">

<label for="email">Your Email</label>

<div class="input-group">

<span class="input-group-text" id="basic-addon3"><span class="fas fa-envelope"></span></span>

{{ form.email(placeholder="Email", class="input form-control", type="email") }}

</div>

</div>

<!-- End of Form -->

<div class="form-group">

<!-- Form -->

<div class="form-group mb-4">

<label for="password">Your Password</label>

<div class="input-group">

<span class="input-group-text" id="basic-addon4"><span class="fas fa-unlock-alt"></span></span>

{{ form.password(placeholder="Password", class="form-control", type="password") }}

</div>

</div>

<!-- End of Form -->

<div class="d-flex justify-content-between align-items-center mb-4">

<div class="form-check">

<input class="form-check-input" type="checkbox" value="" id="defaultCheck5">

<label class="form-check-label" for="defaultCheck5">

Agree terms

</label>

</div>

<div class="form-check">

<a href="{{ url\_for('base\_blueprint.login') }}" class="small text-right">Login</a>

</div>

</div>

</div>

<button type="submit" name="register" class="btn btn-block btn-primary">Sign in</button>

</form>

<div class="d-flex justify-content-center align-items-center mt-4">

<span class="font-weight-normal">

&copy; <a href="https://themesberg.com" target="\_blank">Themesberg</a>

- coded by <a target="\_blank" href="https://appseed.us">AppSeed</a>

</span>

</div>

</div>

</div>

</div>

</div>

</section>

</main>

{% endblock content %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}{% endblock javascripts %}

### app/base/templates/layouts/base.html

<!--

=========================================================

\* Volt - Bootstrap 5 Admin Dashboard

=========================================================

\* Product Page: https://themesberg.com/product/admin-dashboard/volt-bootstrap-5-dashboard

\* Copyright 2020 Themesberg (https://www.themesberg.com)

\* Designed and coded by https://themesberg.com

=========================================================

\* The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. Please contact us to request a removal.

-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Primary Meta Tags -->

<title>

Home Pet Feeder - {% block title %}{% endblock %} | AppSeed

</title>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="title" content="Volt - Free Bootstrap 5 Dashboard">

<meta name="author" content="Themesberg">

<meta name="description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta name="keywords" content="bootstrap 5, bootstrap, bootstrap 5 admin dashboard, free bootstrap 5 dashboard, bootstrap 5 dashboard, bootstrap 5 charts, bootstrap 5 calendar, bootstrap 5 datepicker, bootstrap 5 tables, bootstrap 5 datatable, vanilla js datatable, themesberg, themesberg dashboard, themesberg admin dashboard" />

<link rel="canonical" href="https://themesberg.com/product/admin-dashboard/volt-bootstrap-5-dashboard">

<!-- Open Graph / Facebook -->

<meta property="og:type" content="website">

<meta property="og:url" content="https://demo.themesberg.com/volt">

<meta property="og:title" content="Volt - Free Bootstrap 5 Dashboard">

<meta property="og:description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta property="og:image" content="https://themesberg.s3.us-east-2.amazonaws.com/public/products/volt-bootstrap-5-dashboard/volt-bootstrap-5-dashboard-preview.jpg">

<!-- Twitter -->

<meta property="twitter:card" content="summary\_large\_image">

<meta property="twitter:url" content="https://demo.themesberg.com/volt">

<meta property="twitter:title" content="Volt - Free Bootstrap 5 Dashboard">

<meta property="twitter:description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta property="twitter:image" content="https://themesberg.s3.us-east-2.amazonaws.com/public/products/volt-bootstrap-5-dashboard/volt-bootstrap-5-dashboard-preview.jpg">

<!-- Favicon -->

<link rel="apple-touch-icon" sizes="120x120" href="/static/assets/img/favicon/apple-touch-icon.png">

<link rel="icon" type="image/png" sizes="32x32" href="/static/assets/img/favicon/favicon-32x32.png">

<link rel="icon" type="image/png" sizes="16x16" href="/static/assets/img/favicon/favicon-16x16.png">

<link rel="manifest" href="/static/assets/img/favicon/site.webmanifest">

<link rel="mask-icon" href="/static/assets/img/favicon/safari-pinned-tab.svg" color="#ffffff">

<meta name="msapplication-TileColor" content="#ffffff">

<meta name="theme-color" content="#ffffff">

<!-- Fontawesome -->

<link type="text/css" href="/static/assets/vendor/@fortawesome/fontawesome-free/css/all.min.css" rel="stylesheet">

<!-- Notyf -->

<link type="text/css" href="/static/assets/vendor/notyf/notyf.min.css" rel="stylesheet">

<!-- Volt CSS -->

<link type="text/css" href="/static/assets/css/volt.css" rel="stylesheet">

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

</head>

<body>

<nav class="navbar navbar-dark navbar-theme-primary px-4 col-12 d-md-none">

<a class="navbar-brand mr-lg-5" href="/">

<img class="navbar-brand-dark" src="/static/assets/img/brand/light.svg" alt="Volt logo" /> <img class="navbar-brand-light" src="/static/assets/img/brand/dark.svg" alt="Volt logo" />

</a>

<div class="d-flex align-items-center">

<button class="navbar-toggler d-md-none collapsed" type="button" data-toggle="collapse" data-target="#sidebarMenu" aria-controls="sidebarMenu" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

</div>

</nav>

<div class="container-fluid bg-soft">

<div class="row">

<div class="col-12">

{% include 'includes/sidebar.html' %}

{% block content %}{% endblock content %}

</div>

</div>

</div>

{% include 'includes/scripts.html' %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}{% endblock javascripts %}

</body>

</html>

### app/base/templates/layouts/base-fullscreen.html

<!--

=========================================================

\* Volt - Bootstrap 5 Admin Dashboard

=========================================================

\* Product Page: https://themesberg.com/product/admin-dashboard/volt-bootstrap-5-dashboard

\* Copyright 2020 Themesberg (https://www.themesberg.com)

\* Designed and coded by https://themesberg.com

=========================================================

\* The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. Please contact us to request a removal.

-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Primary Meta Tags -->

<title>

Home Pet Feeder - {% block title %}{% endblock %} | AppSeed

</title>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="title" content="Bryon's Fire Detector Dashboard">

<meta name="author" content="Themesberg">

<meta name="description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta name="keywords" content="bootstrap 5, bootstrap, bootstrap 5 admin dashboard, free bootstrap 5 dashboard, bootstrap 5 dashboard, bootstrap 5 charts, bootstrap 5 calendar, bootstrap 5 datepicker, bootstrap 5 tables, bootstrap 5 datatable, vanilla js datatable, themesberg, themesberg dashboard, themesberg admin dashboard" />

<link rel="canonical" href="https://themesberg.com/product/admin-dashboard/volt-bootstrap-5-dashboard">

<!-- Open Graph / Facebook -->

<meta property="og:type" content="website">

<meta property="og:url" content="https://demo.themesberg.com/volt">

<meta property="og:title" content="Volt - Free Bootstrap 5 Dashboard">

<meta property="og:description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta property="og:image" content="https://themesberg.s3.us-east-2.amazonaws.com/public/products/volt-bootstrap-5-dashboard/volt-bootstrap-5-dashboard-preview.jpg">

<!-- Twitter -->

<meta property="twitter:card" content="summary\_large\_image">

<meta property="twitter:url" content="https://demo.themesberg.com/volt">

<meta property="twitter:title" content="Volt - Free Bootstrap 5 Dashboard">

<meta property="twitter:description" content="Volt is a free and open source Bootstrap 5 Admin Dashboard featuring 11 example pages, 100 components and 3 plugins with Vanilla JS.">

<meta property="twitter:image" content="https://themesberg.s3.us-east-2.amazonaws.com/public/products/volt-bootstrap-5-dashboard/volt-bootstrap-5-dashboard-preview.jpg">

<!-- Favicon -->

<link rel="apple-touch-icon" sizes="120x120" href="/static/assets/img/favicon/apple-touch-icon.png">

<link rel="icon" type="image/png" sizes="32x32" href="/static/assets/img/favicon/favicon-32x32.png">

<link rel="icon" type="image/png" sizes="16x16" href="/static/assets/img/favicon/favicon-16x16.png">

<link rel="manifest" href="/static/assets/img/favicon/site.webmanifest">

<link rel="mask-icon" href="/static/assets/img/favicon/safari-pinned-tab.svg" color="#ffffff">

<meta name="msapplication-TileColor" content="#ffffff">

<meta name="theme-color" content="#ffffff">

<!-- Fontawesome -->

<link type="text/css" href="/static/assets/vendor/@fortawesome/fontawesome-free/css/all.min.css" rel="stylesheet">

<!-- Notyf -->

<link type="text/css" href="/static/assets/vendor/notyf/notyf.min.css" rel="stylesheet">

<!-- Volt CSS -->

<link type="text/css" href="/static/assets/css/volt.css" rel="stylesheet">

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

</head>

<body class="{% block body\_class %}{% endblock body\_class %}">

{% block content %}{% endblock content %}

{% include 'includes/scripts.html' %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}{% endblock javascripts %}

</body>

</html>

### app/home/routes.py

# -\*- encoding: utf-8 -\*-

"""

Copyright (c) 2019 - present AppSeed.us

"""

#Imports

from app.home import blueprint

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from flask import render\_template, redirect, url\_for, request, jsonify, Response

from flask\_login import login\_required, current\_user

from app import login\_manager

from jinja2 import TemplateNotFound

import json

import numpy

import datetime

from decimal import Decimal

import sys

import time

import threading

import boto3

from boto3.dynamodb.conditions import Key, Attr

#Define Variables

session = boto3.Session()

credentials = session.get\_credentials()

access\_key = credentials.access\_key

secret\_key = credentials.secret\_key

session\_token = credentials.token

host = "aw3nvob3lstxq-ats.iot.us-east-1.amazonaws.com"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("PubSub-p1828290")

my\_rpi.configureEndpoint(host, 8883)

my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing

my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz

my\_rpi.configureConnectDisconnectTimeout(20) # 10 sec

my\_rpi.configureMQTTOperationTimeout(20) # 5 sec

my\_rpi.connect()

#Define Classes

#Class for Dashboard

class GenericEncoder(json.JSONEncoder):

def default(self, obj):

if isinstance(obj, numpy.generic):

return numpy.asscalar(obj)

elif isinstance(obj, datetime.datetime):

return obj.strftime('%Y-%m-%d %H:%M:%S')

elif isinstance(obj, decimal.Decimal):

return float(obj)

else:

return json.JSONEncoder.default(self, obj)

#Define Functions

#Functions for converting data into json format

class GenericEncoder(json.JSONEncoder):

def default(self, obj):

if isinstance(obj, numpy.generic):

return numpy.asscalar(obj)

elif isinstance(obj, Decimal):

return str(obj)

elif isinstance(obj, datetime.datetime):

return obj.strftime('%Y-%m-%d %H:%M:%S')

elif isinstance(obj, Decimal):

return float(obj)

else:

return json.JSONEncoder.default(self, obj)

def data\_to\_json(data):

json\_data = json.dumps(data,cls=GenericEncoder)

return json\_data

#Function for retrieving data from dynamodb

def get\_data\_from\_dynamodb(n):

try:

dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

table = dynamodb.Table('iot-sensor')

response = table.query(

KeyConditionExpression=Key('deviceid').eq('deviceid\_bryon'),

ScanIndexForward=False

)

items = response['Items']

data = items[:n]

data\_reversed = data[::-1]

return data\_reversed

except:

import sys

print(sys.exc\_info()[0])

print(sys.exc\_info()[1])

#Define Blueprint Routes

@blueprint.route('/index')

@login\_required

def index():

datetimeid = datetime.datetime.now().isoformat()

return render\_template('index.html', segment='index', datetime=datetimeid)

@blueprint.route('/<template>')

@login\_required

def route\_template(template):

try:

if not template.endswith( '.html' ):

template += '.html'

# Detect the current page

segment = get\_segment( request )

# Serve the file (if exists) from app/templates/FILE.html

return render\_template( template, segment=segment )

except TemplateNotFound:

return render\_template('page-404.html'), 404

except:

return render\_template('page-500.html'), 500

# Helper - Extract current page name from request

def get\_segment( request ):

try:

segment = request.path.split('/')[-1]

if segment == '':

segment = 'index'

return segment

except:

return None

@blueprint.route("/api/getdata",methods = ['POST', 'GET'])

def apidata\_getdata():

if request.method == 'POST':

try:

data = {'chart\_data': data\_to\_json(get\_data\_from\_dynamodb(10)),

'title': "IOT Data"}

print(data)

return jsonify(data)

except:

print(sys.exc\_info()[0])

print(sys.exc\_info()[1])

return None

@blueprint.route("/writePump/<status>")

def checkPump(status):

out = ""

if status == "On":

out = "The Pump is On"

my\_rpi.publish("pump", status, 1)

else:

out = "The Pump is Off"

my\_rpi.publish("pump", status, 1)

return out

@blueprint.route("/writeServo/<status>")

def checkServo(status):

out = ""

if status == "On":

message = {}

out = "The Servo is On"

my\_rpi.publish("servo", "-1", 1)

now = datetime.datetime.now()

message['fed\_time'] = now.isoformat()

my\_rpi.publish("feed", json.dumps(message), 1)

else:

out = "The Servo is Off"

my\_rpi.publish("servo", "0", 1)

return out

@blueprint.route("/streamServo/<status>")

def moveServo(status):

out = ""

if status == "Up":

my\_rpi.publish("servoY", "1", 1)

elif status == "Left":

my\_rpi.publish("servoX", "1", 1)

elif status == "Center":

my\_rpi.publish("servoX", "0", 1)

my\_rpi.publish("servoY", "0", 1)

elif status == "Right":

my\_rpi.publish("servoX", "-1", 1)

elif status == "Down":

my\_rpi.publish("servoY", "-1", 1)

return out

@blueprint.route("/camera")

def cameraModule():

global access\_key,secret\_key,session\_token

print(access\_key,secret\_key,session\_token)

return render\_template("camera.html", akey=access\_key, skey=secret\_key, stoken=session\_token)

### app/home/templates/camera.html

{% extends "layouts/base.html" %}

{% block title %} Camera {% endblock %}

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}

<style>

#playerContainer,

.player {

width: 100%;

height: auto;

min-height: 400px;

background-color: black;

outline: none;

}

.vjs-big-play-button {

display: none !important;

}

</style>

{% endblock stylesheets %}

{% block content %}

<main class="content" onload=startStream()>

<div class="preloader bg-soft flex-column justify-content-center align-items-center">

<img class="loader-element animate\_\_animated animate\_\_jackInTheBox" src="/static/assets/img/brand/light.svg" height="60" alt="Volt logo">

</div>

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center py-4">

<div class="d-block mb-4 mb-md-0">

<nav aria-label="breadcrumb" class="d-none d-md-inline-block">

<ol class="breadcrumb breadcrumb-dark breadcrumb-transparent">

<li class="breadcrumb-item"><a href="index.html"><span class="fas fa-home"></span></a></li>

<li class="breadcrumb-item active" aria-current="page">Camera</li>

</ol>

</nav>

<h2 class="h4">Pi Camera Module</h2>

<p class="mb-0">See what the camera sees.</p>

</div>

</div>

<div class="row justify-content-md-center">

<div class="pump-status col-12">

<div class="card border-light shadow-sm">

<div class="card-body">

<div id="playerContainer">

<!-- HLS.js elements -->

<video id="hlsjs" class="player" controls autoplay></video>

<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>

</div>

</div>

</div>

</div>

</div>

<div class="row justify-content-md-center">

<div class="col-sm-12">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="up" class="btn btn-primary pump-button">Up</button>

</div>

</div>

</div>

<div class="col-sm-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="left" class="btn btn-primary pump-button">Left</button>

</div>

</div>

</div>

<div class="col-sm-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="center" class="btn btn-primary pump-button">Center</button>

</div>

</div>

</div>

<div class="col-sm-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="right" class="btn btn-primary pump-button">Right</button>

</div>

</div>

</div>

<div class="col-sm-12">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="down" class="btn btn-primary pump-button">Down</button>

</div>

</div>

</div>

</div>

{% include 'includes/footer.html' %}

</main>

{% endblock content %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.slim.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/aws-sdk/2.490.0/aws-sdk.min.js"></script>

<script>

function startStream() {

var protocol = "HLS";

var streamName = "test";

// Step 1: Configure SDK Clients

var options = {

//Change Keys here

accessKeyId: "{{ akey }}",

secretAccessKey: "{{ skey }}",

sessionToken: "{{ stoken }}" || undefined,

region: "us-east-1",

}

var kinesisVideo = new AWS.KinesisVideo(options);

var kinesisVideoArchivedContent = new AWS.KinesisVideoArchivedMedia(options);

// Step 2: Get a data endpoint for the stream

console.log('Fetching data endpoint');

kinesisVideo.getDataEndpoint({

StreamName: streamName,

APIName: protocol === 'DASH' ? "GET\_DASH\_STREAMING\_SESSION\_URL" : "GET\_HLS\_STREAMING\_SESSION\_URL"

}, function(err, response) {

if (err) { return console.error(err); }

console.log('Data endpoint: ' + response.DataEndpoint);

kinesisVideoArchivedContent.endpoint = new AWS.Endpoint(response.DataEndpoint);

// Step 3: Get a Streaming Session URL

var consoleInfo = 'Fetching ' + protocol + ' Streaming Session URL';

console.log(consoleInfo);

kinesisVideoArchivedContent.getHLSStreamingSessionURL({

StreamName: "test",

PlaybackMode: "LIVE",

HLSFragmentSelector: {

FragmentSelectorType: "SERVER\_TIMESTAMP",

},

ContainerFormat: "FRAGMENTED\_MP4",

DiscontinuityMode: "ALWAYS",

DisplayFragmentTimestamp: "NEVER"

}, function(err, response) {

if (err) { return console.error(err); }

console.log('HLS Streaming Session URL: ' + response.HLSStreamingSessionURL);

// Step 4: Give the URL to the video player.

var playerElement = $('#hlsjs');

playerElement.show();

var player = new Hls();

console.log('Created HLS.js Player');

player.loadSource(response.HLSStreamingSessionURL);

player.attachMedia(playerElement[0]);

console.log('Set player source');

player.on(Hls.Events.MANIFEST\_PARSED, function() {

video.play();

console.log('Starting playback');

});

});

});

};

</script>

<script>

$(document).ready(function() {

startStream();

});

</script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>

<script>

function turnUp(){

$.ajax({url: "/streamServo/Up",

success: function(result){

console.log("Up")

}

})

}

function turnLeft(){

$.ajax({url: "/streamServo/Left",

success: function(result){

console.log("Left")

}

})

}

function turnCenter(){

$.ajax({url: "/streamServo/Center",

success: function(result){

console.log("Center")

}

})

}

function turnRight(){

$.ajax({url: "/streamServo/Right",

success: function(result){

console.log("Right")

}

})

}

function turnDown(){

$.ajax({url: "/streamServo/Down",

success: function(result){

console.log("Down")

}

})

}

$(document).ready(function(){

$("#up").click(function(){

turnUp();

});

$("#left").click(function(){

turnLeft();

});

$("#center").click(function(){

turnCenter();

});

$("#right").click(function(){

turnRight();

});

$("#down").click(function(){

turnDown();

});

});

</script>

{% endblock javascripts %}

### app/home/templates/index.html

{% extends "layouts/base.html" %}

{% block title %} Dashboard {% endblock %}

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

<main class="content">

<div class="preloader bg-soft flex-column justify-content-center align-items-center">

<img class="loader-element animate\_\_animated animate\_\_jackInTheBox" src="/static/assets/img/brand/light.svg" height="60" alt="Volt logo">

</div>

<div class="row justify-content-md-center">

<div class="col-12 col-sm-6 col-xl-4 mb-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<div class="row d-block d-xl-flex align-items-center">

<div class="col-12 col-xl-5 text-xl-center mb-3 mb-xl-0 d-flex align-items-center justify-content-xl-center">

<div class="icon icon-shape icon-md icon-shape-blue rounded mr-4 mr-sm-0"><span class="fas fa-chart-line"></span></div>

</div>

<div class="col-12 col-xl-7 px-xl-0">

<div class="d-none d-sm-block">

<h2 class="h5">Temperature</h2>

<h4 class="mb-1" id="latest-temp">Loading...</h3>

</div>

<small id="latest-temp-time">Loading...</small>

</div>

</div>

</div>

</div>

</div>

<div class="col-12 col-sm-6 col-xl-4 mb-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<div class="row d-block d-xl-flex align-items-center">

<div class="col-12 col-xl-5 text-xl-center mb-3 mb-xl-0 d-flex align-items-center justify-content-xl-center">

<div class="icon icon-shape icon-md icon-shape-secondary rounded mr-4"><span class="fas fa-lightbulb"></span></div>

</div>

<div class="col-12 col-xl-7 px-xl-0">

<div class="d-none d-sm-block">

<h2 class="h5">Motion</h2>

<h4 class="mb-1" id="latest-motion">Loading...</h3>

</div>

<small id="latest-motion-time">Loading...</small>

</div>

</div>

</div>

</div>

</div>

<div class="col-12 col-sm-6 col-xl-4 mb-4">

<div class="card border-light shadow-sm">

<div class="card-body">

<div class="row d-block d-xl-flex align-items-center">

<div class="col-12 col-xl-5 text-xl-center mb-3 mb-xl-0 d-flex align-items-center justify-content-xl-center">

<div class="icon icon-shape icon-md icon-shape-blue rounded mr-4 mr-sm-0"><span class="fas fa-clock"></span></div>

</div>

<div class="col-12 col-xl-7 px-xl-0">

<div class="d-none d-sm-block">

<h2 class="h5">Last Fed</h2>

<h4 class="mb-1" id="last-fed-time">Loading...</h3>

</div>

<small id="last-fed-date">Loading...</small>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="line-chart" style="position: relative"></div>

<div class="card border-light shadow-sm mb-4">

<div class="card-body">

<div class="table-responsive">

<table class="table table-centered table-nowrap mb-0 rounded">

<thead class="thead-light">

<tr>

<th class="border-0">ID</th>

<th class="border-0">Date & Time</th>

<th class="border-0">Motion</th>

<th class="border-0">Temperature</th>

</tr>

</thead>

<tbody id="sensor-table">

<!-- Item -->

<!-- End of Item -->

</tbody>

</table>

</div>

</div>

</div>

{% include 'includes/footer.html' %}

</main>

{% endblock content %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}

<script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>

<script >

sensordata = null

count = 0

var fed\_time = "Loading"

var fed\_date = "Loading"

function getNewData() {

datetime\_arr = []

motion\_arr = []

temp\_arr = []

jQuery.ajax({

url: "/api/getdata",

type: 'POST',

success: function(ndata, textStatus, xhr) {

sensordata = JSON.parse(ndata.chart\_data)

} //end success

}); //end ajax

for (i in sensordata) {

datetime = sensordata[i].datetimeid;

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleDateString() + " " +jsdatetime.toLocaleTimeString();

temp = sensordata[i].temperature;

//console.log(temp)

motion = sensordata[i].motion;

motion\_arr.push(motion);

temp\_arr.push(temp);

datetime\_arr.push(jstime);

//console.log(i);

}

//console.log(temp\_arr)

new Chartist.Line('.line-chart', {

labels: [datetime\_arr[0], datetime\_arr[1], datetime\_arr[2], datetime\_arr[3], datetime\_arr[4], datetime\_arr[5], datetime\_arr[6],datetime\_arr[7], datetime\_arr[8], datetime\_arr[9]],

series: [

[temp\_arr[0], temp\_arr[1], temp\_arr[2], temp\_arr[3], temp\_arr[4], temp\_arr[5], temp\_arr[6], temp\_arr[7], temp\_arr[8], temp\_arr[9]]

]

}, {

fullWidth: false,

chartPadding: 30,

height: "300px",

plugins: [

Chartist.plugins.tooltip()

]

});

jsdatetime = new Date(Date.parse("{{ datetime }}"));

fed\_time = jsdatetime.toLocaleTimeString();

fed\_date = jsdatetime.toLocaleDateString();

if (count!=0) {

$("#latest-temp").html(temp\_arr[9]+"°C")

$("#latest-motion").html(motion\_arr[9])

$("#latest-motion-time").html(datetime\_arr[9])

$("#last-fed-time").html(fed\_time)

$("#last-fed-date").html(fed\_date)

$("#latest-temp-time").html(datetime\_arr[9])

} else {

$("#latest-motion").html("Loading...")

$("#latest-motion-time").html("Loading...")

$("#last-fed-time").html("Loading...")

$("#last-fed-date").html("Loading...")

$("#latest-temp-time").html("Loading...")

}

updateTable(sensordata)

count++

}

function updateTable(sensordata) {

var rows = "";

var counter = 1;

for (i in sensordata) {

var row = "<tr>";

row += "<td>"+counter+"</td>";

datetime = sensordata[i].datetimeid;

jsdatetime = new Date(Date.parse(datetime));

jstime = jsdatetime.toLocaleDateString() + " " +jsdatetime.toLocaleTimeString();

row += "<td>"+jstime+"</td>";

row += "<td>"+sensordata[i].motion+"</td>";

row += "<td>"+sensordata[i].temperature+"</td>";

rows += row;

counter++

}

$("#sensor-table").html(rows)

}

$(document).ready(function() {

setInterval(function() {

getNewData()

}, 3000);

});

</script>

{% endblock javascripts %}

<script src="@@path/vendor/chartist/dist/chartist.min.js"></script>

<script src="@@path/vendor/chartist-plugin-tooltips/dist/chartist-plugin-tooltip.min.js"></script>

### app/home/templates/pump.html

{% extends "layouts/base.html" %}

{% block title %} Feeder {% endblock %}

<!-- Specific Page CSS goes HERE -->

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

<main class="content">

<div class="preloader bg-soft flex-column justify-content-center align-items-center">

<img class="loader-element animate\_\_animated animate\_\_jackInTheBox" src="/static/assets/img/brand/light.svg" height="60" alt="Volt logo">

</div>

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center py-4">

<div class="d-block mb-4 mb-md-0">

<nav aria-label="breadcrumb" class="d-none d-md-inline-block">

<ol class="breadcrumb breadcrumb-dark breadcrumb-transparent">

<li class="breadcrumb-item"><a href="index.html"><span class="fas fa-home"></span></a></li>

<li class="breadcrumb-item active" aria-current="page">Feeder</li>

</ol>

</nav>

<h2 class="h4">Feeder</h2>

<p class="mb-0">Release water and food from here.</p>

</div>

</div>

<div class="row justify-content-md-center">

<div class="col-sm-12">

<h3>Water Controls</h3>

</div>

<div class="col-sm-6">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="b1" class="btn btn-primary pump-button">Turn on</button>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="b2" class="btn btn-primary pump-button">Turn off</button>

</div>

</div>

</div>

<div class="pump-status col-12">

<div class="card border-light shadow-sm">

<div class="card-body">

<h2>Status:</h2>

<span id="pump-status">The Pump is Off</span

</div>

</div>

</div>

</div>

<div class="row justify-content-md-center">

<div class="col-sm-12">

<h3>Food Controls</h3>

</div>

<div class="col-sm-6">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="b3" class="btn btn-primary pump-button">Turn on</button>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card border-light shadow-sm">

<div class="card-body">

<button id="b4" class="btn btn-primary pump-button">Turn off</button>

</div>

</div>

</div>

<div class="pump-status col-12">

<div class="card border-light shadow-sm">

<div class="card-body">

<h2>Status:</h2>

<span id="servo-status">The Servo is Off</span

</div>

</div>

</div>

</div>

{% include 'includes/footer.html' %}

</main>

{% endblock content %}

<!-- Specific Page JS goes HERE -->

{% block javascripts %}

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>

<script>

function turnPumpOn(){

$.ajax({url: "writePump/On",

success: function(result){

$("#pump-status").html(result);

}

})

}

function turnPumpOff(){

$.ajax({url: "writePump/Off",

success: function(result){

$("#pump-status").html(result);

}

})

}

function turnServoOn(){

$.ajax({url: "writeServo/On",

success: function(result){

$("#servo-status").html(result);

}

})

}

function turnServoOff(){

$.ajax({url: "writeServo/Off",

success: function(result){

$("#servo-status").html(result);

}

})

}

$(document).ready(function(){

$("#b1").click(function(){

console.log("water on");

turnPumpOn();

});

$("#b2").click(function(){

console.log("water off");

turnPumpOff();

});

$("#b3").click(function(){

console.log("food on");

turnServoOn();

});

$("#b4").click(function(){

console.log("food off");

turnServoOff();

});

});

</script>

{% endblock javascripts %}

### pubsub.py

# Import SDK packages

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from time import sleep

from gpiozero import MCP3008, Servo

import json

import random

import RPi.GPIO as GPIO

servo = Servo(26)

servox = Servo(19)

servoy = Servo(13)

RELAY = 17

GPIO.setmode(GPIO.BCM)

GPIO.setup(RELAY, GPIO.OUT)

GPIO.setwarnings(False)

# Custom MQTT message callback

def servocustomCallback(client, userdata, message):

print("Received a new message: ")

print(message.payload)

servo.value = int(message.payload)

sleep(1)

servo.value = None

print("from topic: ")

print(message.topic)

print("--------------\n\n")

def servoXcustomCallback(client, userdata, message):

print("Received a new message: ")

print(message.payload)

servox.value = int(message.payload)

sleep(1)

servox.value = None

print("from topic: ")

print(message.topic)

print("--------------\n\n")

def servoYcustomCallback(client, userdata, message):

print("Received a new message: ")

print(message.payload)

servoy.value = int(message.payload)

sleep(1)

servoy.value = None

print("from topic: ")

print(message.topic)

print("--------------\n\n")

def pumpcustomCallback(client, userdata, message):

print("Received a new message: ")

print(message.payload)

status = str(message.payload)

if "On" in status:

print("On")

GPIO.output(RELAY, GPIO.HIGH)

else:

GPIO.output(RELAY, GPIO.LOW)

print("from topic: ")

print(message.topic)

print("--------------\n\n")

host = "ADD IN ENDPOINT API"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("PubSub-p1828331")

my\_rpi.configureEndpoint(host, 8883)

my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing

my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz

my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec

my\_rpi.configureMQTTOperationTimeout(5) # 5 sec

# Connect and subscribe to AWS IoT

my\_rpi.connect()

my\_rpi.subscribe("servo", 1, servocustomCallback)

my\_rpi.subscribe("servoX", 1, servoXcustomCallback)

my\_rpi.subscribe("servoY", 1, servoYcustomCallback)

my\_rpi.subscribe("pump", 1, pumpcustomCallback)

sleep(2)

while True:

print("listening")

sleep(5)

sleep(5)

### text\_rekognition.py

import boto3

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

import botocore

from picamera import PiCamera

from time import sleep

host = "ADD IN ENDPOINT API"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("p1828331-PubSub")

my\_rpi.configureEndpoint(host, 8883)

my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing

my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz

my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec

my\_rpi.configureMQTTOperationTimeout(5) # 5 sec

my\_rpi.connect()

# Set the filename and bucket name

BUCKET = 'sp-p1828290-s3-bucket' # replace with your own unique bucket name

location = {'LocationConstraint': 'us-east-1'}

file\_path = "/home/pi/Desktop"

file\_name = "test.jpg"

def takePhoto(file\_path,file\_name):

with PiCamera() as camera:

#camera.resolution = (1024, 768)

full\_path = file\_path + "/" + file\_name

camera.capture(full\_path)

sleep(3)

def uploadToS3(file\_path,file\_name, bucket\_name,location):

s3 = boto3.resource('s3') # Create an S3 resource

exists = True

try:

s3.meta.client.head\_bucket(Bucket=bucket\_name)

except botocore.exceptions.ClientError as e:

error\_code = int(e.response['Error']['Code'])

if error\_code == 404:

exists = False

if exists == False:

s3.create\_bucket(Bucket=bucket\_name,CreateBucketConfiguration=location)

# Upload the file

full\_path = file\_path + "/" + file\_name

s3.Object(bucket\_name, file\_name).put(Body=open(full\_path, 'rb'))

print("File uploaded")

def detect\_text(bucket, key, region="us-east-1"):

rekognition = boto3.client("rekognition", region)

response = rekognition.detect\_text(

Image={

"S3Object": {

"Bucket": bucket,

"Name": key,

}

},

)

return response['TextDetections']

takePhoto(file\_path, file\_name)

uploadToS3(file\_path,file\_name, BUCKET,location)

highestconfidence = 0

best\_bet\_item = "Unknown"

for label in detect\_text(BUCKET, file\_name):

print("{DetectedText} - {Confidence}%".format(\*\*label))

if label["Confidence"] >= highestconfidence:

highestconfidence = label["Confidence"]

best\_bet\_item = label["DetectedText"]

if label["DetectedText"] == "EMPTY":

my\_rpi.publish("bowlalert","Attention! The food bowl is empty.",1)

if best\_bet\_item!= "Unknown":

print("This should be a {} with confidence {}".format(best\_bet\_item, highestconfidence))

### checkbowl.py

import time

import os

while True:

os.system("python text\_rekognition.py")

time.sleep(15)

### store\_sens.py

import serial

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from time import sleep

import sys

import datetime as datetime

import json

host = "ADD IN ENDPOINT API"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("p1828331-PubSub")

my\_rpi.configureEndpoint(host, 8883)

my\_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

my\_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing

my\_rpi.configureDrainingFrequency(2) # Draining: 2 Hz

my\_rpi.configureConnectDisconnectTimeout(10) # 10 sec

my\_rpi.configureMQTTOperationTimeout(5) # 5 sec

my\_rpi.connect()

try:

ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)

print("Successfully connected to database!")

update = True

while update:

try:

line = ser.readline()

sensorvalues = line.split()

tempvalue=sensorvalues[0]

humidvalue=sensorvalues[1]

motionvalue=sensorvalues[2]

print("Temperature value:", tempvalue)

print("Humidityvalue:", humidvalue)

print("Motion value:", motionvalue)

print("Wait 2 secs before getting next values..")

message = {}

message["deviceid"] = "deviceid\_bryon"

now = datetime.datetime.now()

message["datetimeid"] = now.isoformat()

message["temperature"] = tempvalue

message["motion"] = motionvalue

my\_rpi.publish("sensors", json.dumps(message), 1)

sleep(5)

except KeyboardInterrupt:

update = False

cursor.close()

cnx.close()

except:

print("Error while inserting data...")

print(sys.exc\_info()[0])

print(sys.exc\_info()[1])

except:

print(sys.exc\_info()[0])

print(sys.exc\_info()[1])

# Section 5 Task List

A table listing members names and the parts of the assignment they worked on

|  |  |  |  |
| --- | --- | --- | --- |
| Name of member | Part of project worked on | | Contribution percentage |
| Bryon  Dextor | Setup EC2 Webpage  Setup DynamoDB for Login & Sensor  Setup Kinesis Web Stream on EC2  Setup Hardware  Documentation  Setup MQTT for Sensors & Actuators  Setup IOT Core Rules  Setup SNS Email Notification  Setup Alexa Voice Services  Setup Text Recognition using Rekognition  Setup Hardware  Documentation  Setup MQTT for Sensors & Actuators | | 100%  100%  100%  50%  50%  50%  100%  100%  100%  100%  50%  50%  50% |
|  |  |  | |
|  |  |  | |

# Section 6 References

Developer.amazon.com. n.d. [online] Available at: <https://developer.amazon.com/en-US/docs/alexa/avs-device-sdk/raspberry-pi.html> [Accessed 17 February 2021].

Docs.aws.amazon.com. n.d. What Is Amazon Kinesis Video Streams? - Amazon Kinesis Video Streams. [online] Available at: <https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/what-is-kinesis-video.html> [Accessed 18 February 2021].

**-- End of CA2 Step-by-step tutorial --**