Design Manual

Smart Vending Machine

Group 1

28th May, 2021

Bragadeeshan S	**E/16/055**
Karikaran V	**E/16/172**
Girishikan S	**E/16/115**

CONTENT

The Introduction

The smart vending machine idea was created because of the things we considered the problems with traditional vending such as Having to pay for the products with cash most of the time.Easy to hack Traditional vending machine.Prices and Expiry dates are not checked by the Traditional vending machine.As the solution we came to a conclusion of the smart vending machine.

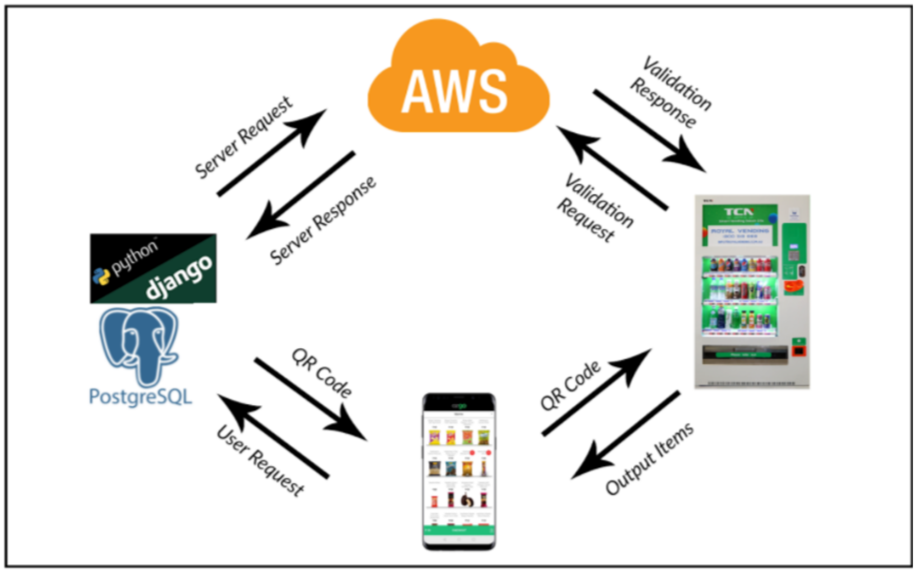
The Main advantages of smart vending. It has the (Gender,Age,generation wise) analysis , 24 hours Distribution and vending services , Transaction Database services. Which can be used to check the performance of a product in a specific market.Prices and expiry dates are real time because it is connected to the cloud.

The web application acts as an online store and Prints a QR code on the screen which is used to buy items from the smart vending machine.This Vending machine is power efficient, easy to use and Fast. The machine can connect to home Wifis and any kind of internet connection.The Vending machine is powered by Raspberry Pi 3 Model B which uses a Quad Core 1.2GHz Broadcom BCM2837 64bit Processor.

Why a Smart Vending Machine

If we consider the context of Covid 19 Quarantines and People needing day today use items.We can come to the conclusion that We need unmanned stores which do not need to be paid with cash all the time.Now a days everyone has an electronic card or two If we also consider that . We can clearly see the use of a smart vending machine in the future. The vending machine can always help you with all kinds of items. In the near future there will be no stores for small items such as toothpastes ,soaps and other items. That’s why we wanted to make our trend Vending Machine which can be used to deliver products by using the minimum effort.

Smart Vending Machine Overview



The smart vending machine works like the diagram shown above the user accesses the mobile application or web application when in the vicinity of the vending machine.The user goes through the application and selects and pays for the items he needs the application that was made using Python Django and as the database PostgreSQL.Then the app makes a HTTP request to the server on the cloud which is EC2 Instance which runs a linux ubuntu OS and also Gunicorn wsgi server is there to internally handle the code and the database is also PostgreSQL.After the request Server sends a HTTP response and with the details the application creates a QR code which is kind of a key which can be used to buy items from the vending machine.Then the user goes to the machine and shows the QR code to the Vending machine and the vending machine sends a validation HTTP request to the cloud server and if the QR code is Valid the server sends a Validation response and the item is outputted from the vending machine.

In a nutshell we used

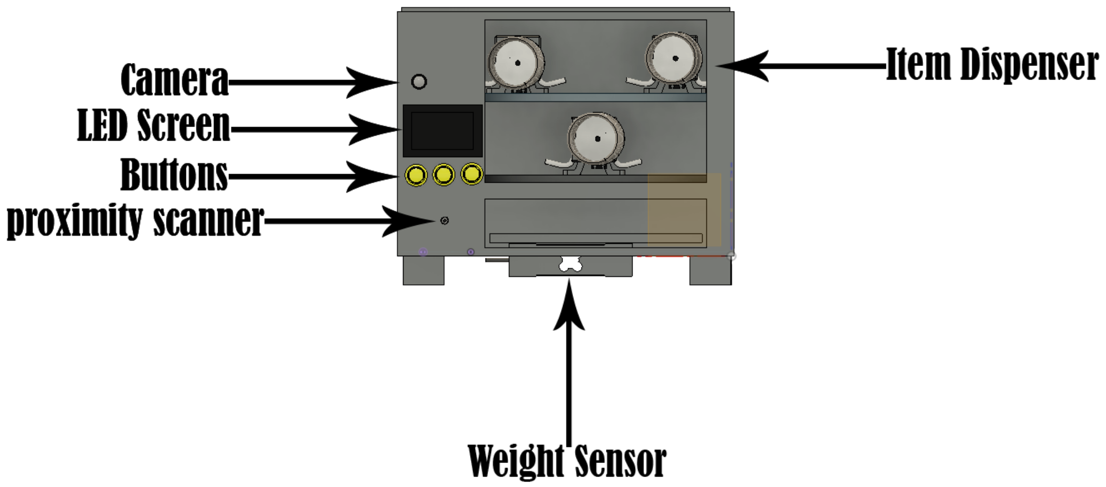
Web Application - Python Django

RestAPI -DjangoREST

Cloud Server - AWS EC2 Instance with linux Ubuntu OS

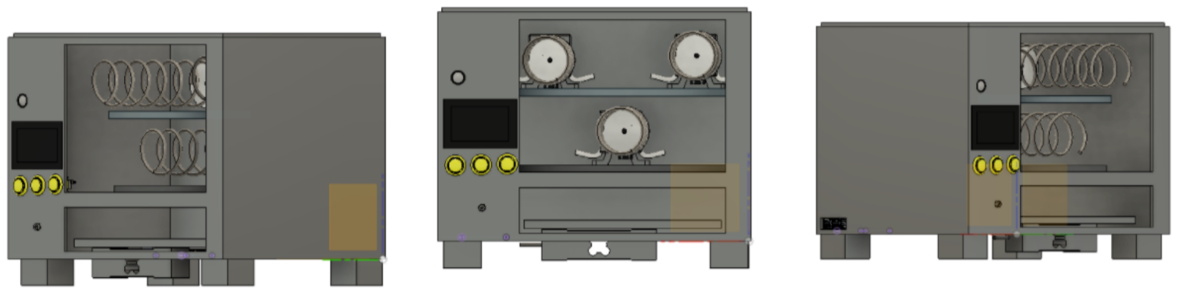
Requests- HTTPs

Design Overview



The overall Design Looks like the model figure above.The vending machine is 60 cm in height, 80 cm in width and 80 cm in length. The vending machine can hold up to 3 different average sized items (chocolate bars,mixture packets ,Sodatins) 4 in each under 300g.

3D CAD Design



The 3D cad designs were made to make sure we can have the Laser cuttings according to the dimensions. And to have an idea about the vending machine and how it is going to be there.These 3D designs were made using AutoDesk Fusion .The dimensions were specified as before.

Software Deployment

Web Application

Overview

Python Django is used to develop the Web application of the Smart Vending Machine.Main Objectives of the Web Application are,

- Interaction with the user
- Ordering the items available in the vending machine
- Updating details to the Cloud
- Managing the Items,Customers and the orders
- Providing sufficient resources to buy Items

The library Prerequisite

- **CacheControl** 0.12.6 : CacheControl is a port of the caching algorithms in httplib2 for use with requests session object.It was written because httplib2's better support for caching is often mitigated by its lack of thread safety. The same is true of requests in terms of caching.

- **Chardet** 4.0.0 : The Universal Character Encoding Detector for decoding the encrypted code

- **Dj-database-url** 0.5.0   : This simple Django utility allows you to utilize the 12factor inspired DATABASE_URL environment variable to configure your Django application.

- **Django** 3.1.5

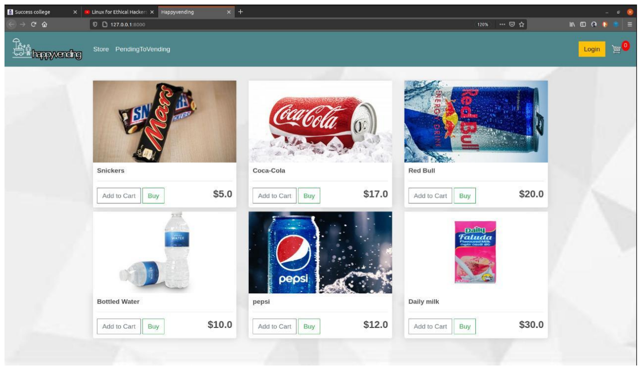
- **Django-filter** 2.4.0 : Django-filter is a reusable Django application allowing users to declaratively add dynamic QuerySet filtering from URL parameters.

- **Django-storages** 1.11.1 : django-storages is a collection of custom storage backends for Django.

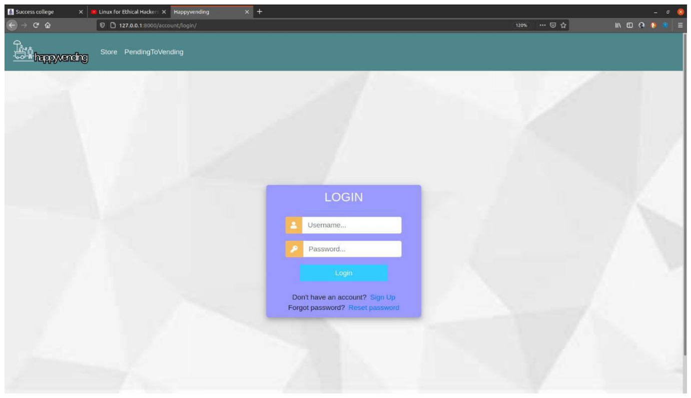
- **Django-widget-tweaks** 1.4.8 : Tweak the form field rendering in templates, not in python-level form definitions. Altering CSS classes and HTML attributes is supported.That should be enough for designers to customize field presentation (using CSS and unobtrusive javascript) without touching python code.

- **Djangorestframework** 3.12.2   : Django REST framework is a powerful and flexible toolkit for building Web APIs.
- **Djangorestframework-simplejwt** 4.6.0 : Simple JWT is a JSON Web Token authentication plugin for the Django REST Framework.
- **Gunicorn** 20.0.4 : Gunicorn ‘Green Unicorn’ is a Python WSGI HTTP Server for UNIX.
- **Html5lib** 1.0.1 : html5lib is a pure-python library for parsing HTML. It is designed to conform to the WHATWG HTML specification, as is implemented by all major web browsers.
- **Jmespath** 0.10.0 : JMESPath allows you to declaratively specify how to extract elements from a JSON document.
- **Msgpack** 0.6.2 : MessagePack is an efficient binary serialization format. It lets you exchange data among multiple languages like JSON. But it's faster and smaller
- **Opencv-python** 4.5.1.48   : for qrcode image reading
- **Pillow** 8.1.0   : The Python Imaging Library adds image processing capabilities to your Python interpreter.
- **PyJWT** 2.0.1 : JSON Web Token implementation in Python its support for django rest framework-simple jwt
- **Python-dateutil** 2.8.1   : The dateutil module provides powerful extensions to the standard datetime module, available in Python.
- **Pytz** 2020.5   : its used for reading the qrcode image
- **Qrcode** 6.1   : Pure python QR Code generator
- **Requests** 2.25.1   : Requests is a simple, yet elegant HTTP library. its used for Api call interaction
- **S3transfer** 0.3.4   : S3transfer is a Python library for managing Amazon S3 transfers.its help to connect aws s3 bucket
- **Sqlparse** 0.4.1 : sqlparse is a non-validating SQL parser for Python. It provides support for parsing, splitting and formatting SQL statements.
- **Whitenoise** 5.2.0 : Radically simplified static file serving for WSGI applications

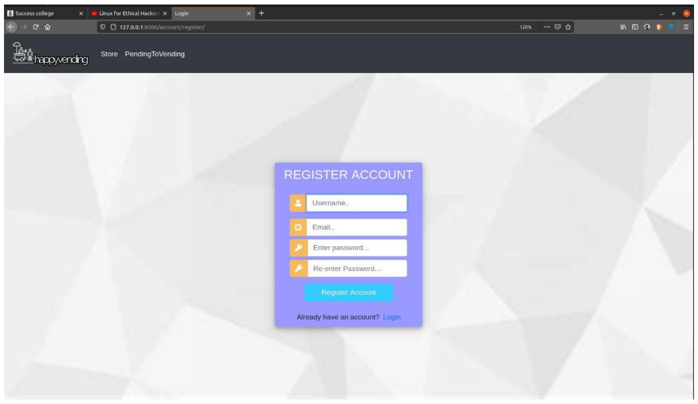
Web Application UI



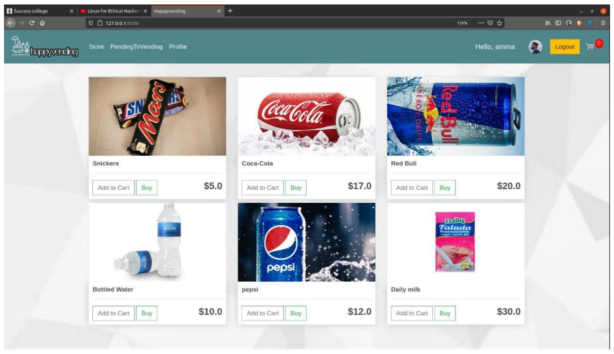
1. The Website gets you to this Home page which Can be used to directly buy items or you can sign in and buy things If you sign in the company can give you discounts or other options.And also from the Homepage you can go to the cart which has the items you selected and the total amount.Other than that you can also go to the tab pending orders and use the QR codes to get the paid items to dispense.



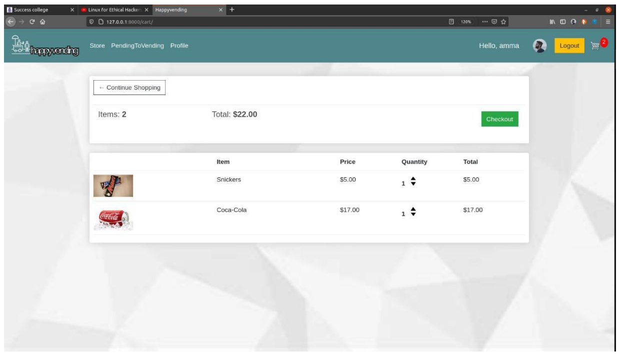
2. This Page is the page you get if you want to Login.You can type in your login credentials and get into the account where you can see your previous orders and etcs. Or If you don't have an account you can go to the Signup option and Sign up for a new account.There is an option to help you reset your password as well if you forget.



3. This is the SignUp page where you can Sign up for a new Account.



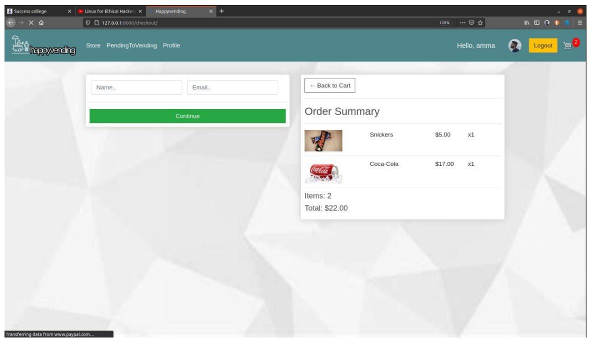
4. After Signing in you will have a page similar to the home page but you can see your account and you can make changes to your account when you click on your profile picture.And also you will have an Profile Button on your webpage where you can go and change your account settings.



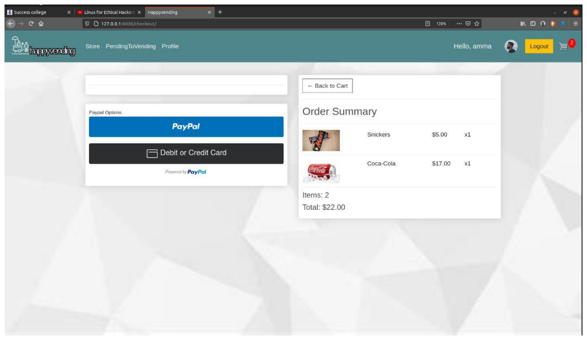
Items: 2 Total: $22.00

	Item	Price	Quantity	Total
	Snickers	$5.00	1	$5.00
	Coca-Cola	$17.00	1	$17.00

5. After you added the items your Cart the cart looks like this and you can edit the items that you are going to add here also.If you click check out it will take to a page where your can pay for the Items.You can use continue shopping to go back and add more items to the cart.

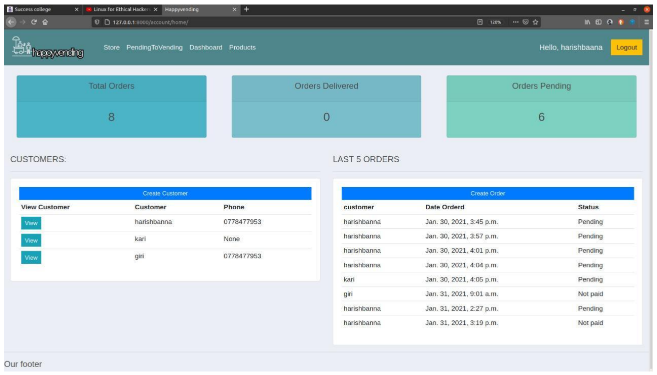


6. When you go to the checkout page you can see that it will ask for a name and an email. It is just for the invoice so you can use it for refunds and other proceeds.



7. After giving the Email and the name You can choose the way to pay (Paypal / Debit Card /Credit card).After choosing the payment method you will be redirected to a dialog box which is going popup and you can give your details there and pay for the items and you will receive the items.

If you are an Admin



Total Orders	Orders Delivered	Orders Pending
8	0	6

CUSTOMERS:

Create Customer

View Customer	Customer	Phone
View	harishbanna	0778477953
View	kari	None
View	giri	0778477953

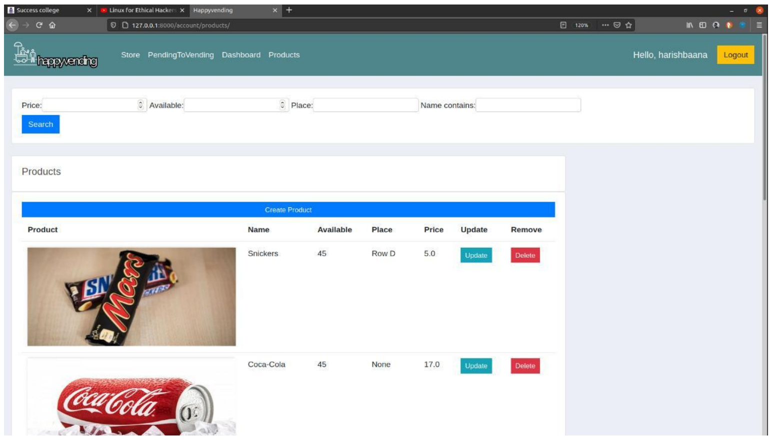
LAST 5 ORDERS

Create Order

customer	Date Orderd	Status
harishbanna	Jan. 30, 2021, 3:45 p.m.	Pending
harishbanna	Jan. 30, 2021, 3:57 p.m.	Pending
harishbanna	Jan. 30, 2021, 4:01 p.m.	Pending
harishbanna	Jan. 30, 2021, 4:04 p.m.	Pending
kari	Jan. 30, 2021, 4:05 p.m.	Pending
giri	Jan. 31, 2021, 9:01 a.m.	Not paid
harishbanna	Jan. 31, 2021, 2:27 p.m.	Pending
harishbanna	Jan. 31, 2021, 3:19 p.m.	Not paid

Our footer

1. If you Login with the admin Credentials you can see all the transactions that have happened with the vending machine over the time. And you can check who has bought the items and their characteristics according to their accounts.And also you can check the pending transactions.

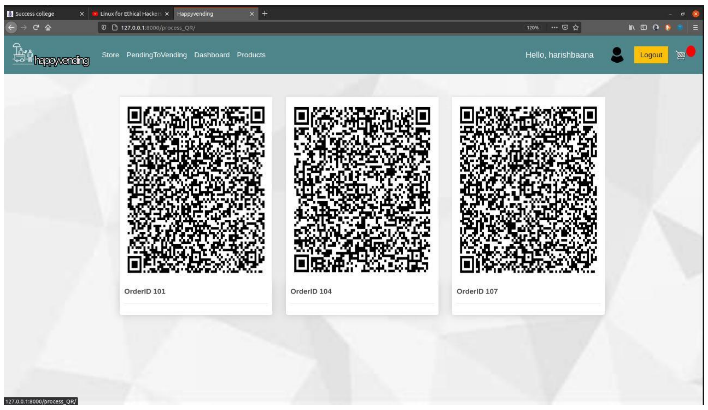


Products

Create Product

Product	Name	Available	Place	Price	Update	Remove
	Snickers	45	Row D	5.0	Update	Delete
	Coca-Cola	45	None	17.0	Update	Delete

2. If you Login as an Admin You can also change the number of products in the vending machine.If you see any miscalculations. And also you can search for items according to their places and the prices and also the names.



3. The other page you can check are the QR codes of the customers who are going to buy the items.

Backend

Backend tasks

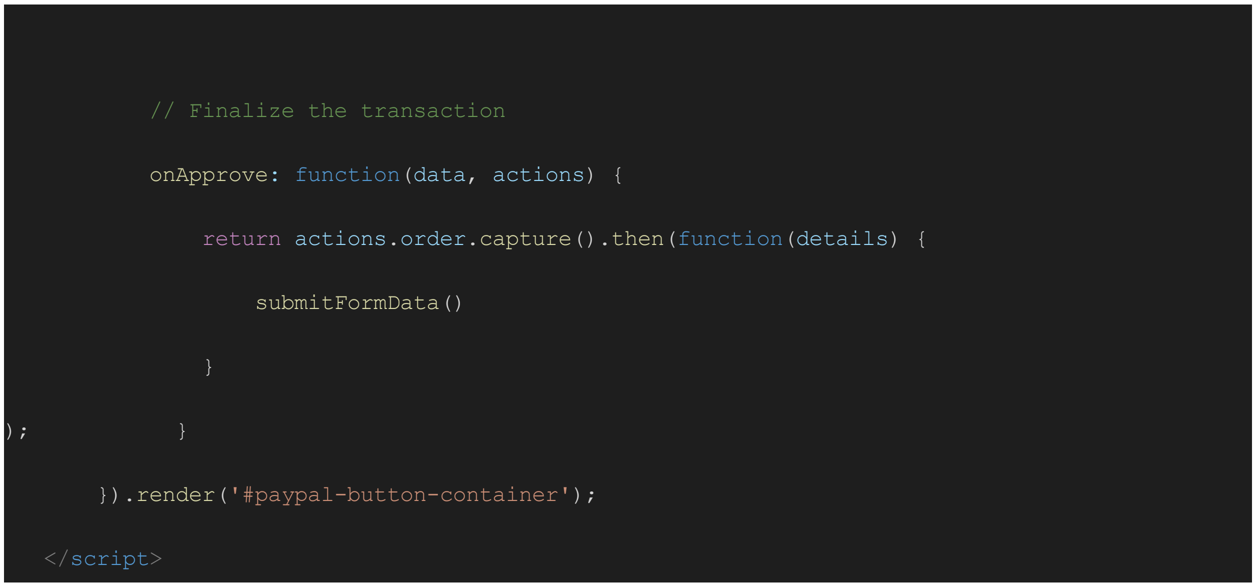
User Registration (User /Admin/Companies)

```
@unauthenticated_user
def registerPage(request):

   form = CreateUserForm()
   if request.method == 'POST':
       form = CreateUserForm(request.POST)
       if form.is_valid():
           user = form.save()
           username = form.cleaned_data.get('username')



           group = Group.objects.get(name='customer')
           user.groups.add(group)

           Customer.objects.create(user=user,name=user.username)

           messages.success(request, 'Account was created for ' + username)

           return redirect('login')
   context = {'form':form}
   return render(request, 'account/register.html', context)
```

Where users are classified into who they are according to their Credential. These Roles can determine the access given to a selected user.

Payment Handling

```
    <script
src="https://www.paypal.com/sdk/js?client-id=AYb_sqnf4YfwkdwcBUYcwSW1CkU6bxUwFf3VIYFDhltrMZptZ
IdexhYQa9WYypqUJ5BsBYAnmmmYpYRb&currency=USD"></script>
   <script>
       var total = '{{order.get_cart_total|floatformat:2}}';
       // Render the PayPal button into #paypal-button-container
       paypal.Buttons({

           style: {
               color:  'blue',
               shape:  'rect',
           },

           // Set up the transaction
           createOrder: function(data, actions) {
               return actions.order.create({
                   purchase_units: [{
                       amount: {
                           value: parseFloat(total).toFixed(2)
                       }
                   }]
               });
           },
```



```
          // Finalize the transaction
          onApprove: function(data, actions) {
              return actions.order.capture().then(function(details) {
                  submitFormData()
              }
);          }
      }).render('#paypal-button-container');
   </script>
```

Payment Handling is done through PayPal Which is a secured Payment Gateway

Add /Modify/Delete Items

The admin can Edit any information about the items available

Transactions

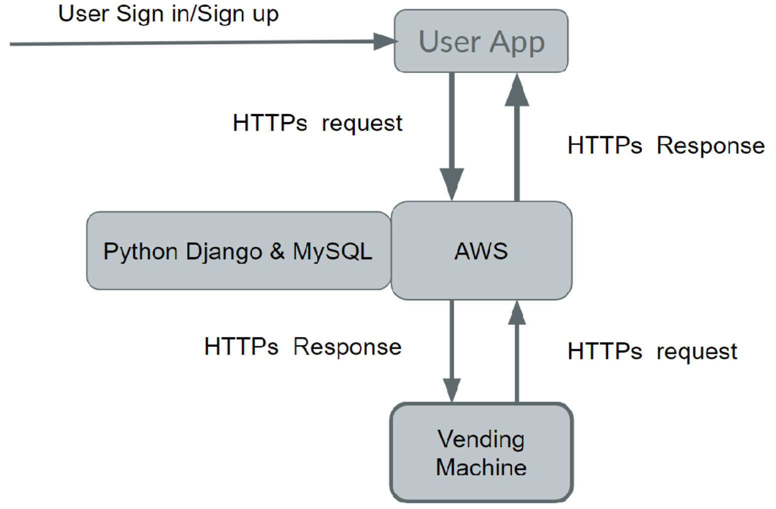
Every Transaction Done can be Seen by an admin .The User can also see the previous Transactions done by him.

Validation

```
def validate(qrData,serverData):

   valid=False

   serverIterm={}

   for iterm in serverData['OrderItems']:

       serverIterm[iterm['name']]=iterm['quantity']

   if((qrData['transactionComplete']==serverData["complete"])

   and (qrData['transactionstatus']==serverData["status"])

   and (float(qrData['transaction_id'])==float(serverData["transaction_id"]))

   and (qrData['transactionstatus']==serverData["status"])

   and (int(qrData['amountTopay'])==int(serverData["paidAmount"]))

   and (serverIterm==qrData['orderProductes'])

   and (qrData['customerName']==serverData["customer"])):

     valid=True

   print(serverIterm)

   return valid
```

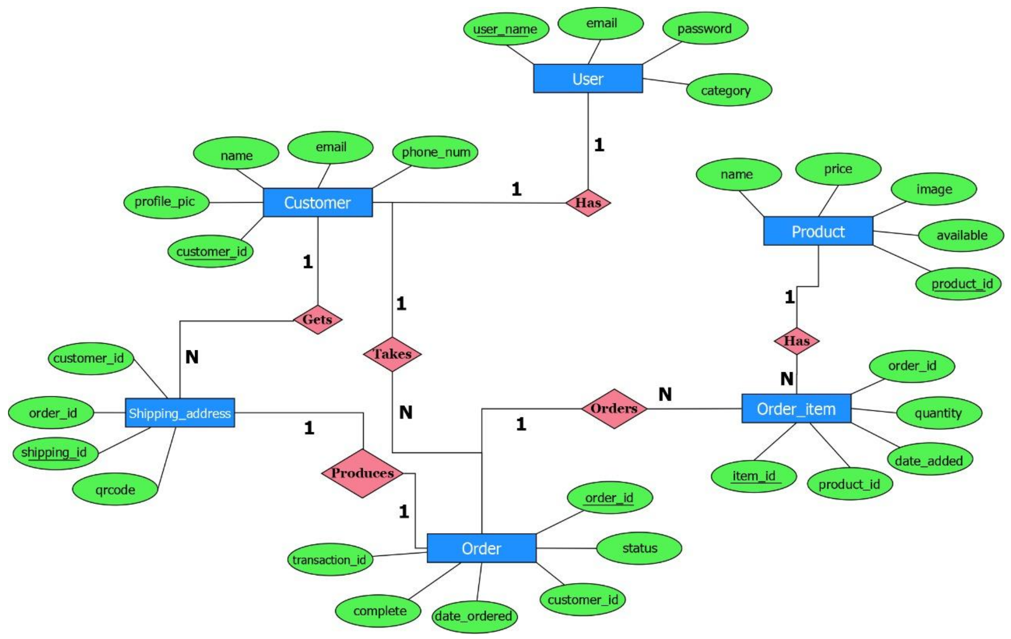
QR code is used for more security and also Django Rest API is used for validity

DataPath of the system



The flow of data is projected above normally a user signs in or signs up with the user app after that user app sends the server data through HTTPs requests and also the response from the server also is through HTTPs responses similarly the API in the vending machine also sends the request as HTTPs and the response is through HTTPs.

ER Diagram



The Main entities are User,Customer , Product, Order, Order Item And QR Code.Every Entity has unique primary Key. And if we go in to details Customer Makes an order or multiple orders so the relationship between Customer and the Order is one to many.Customer can get multiple QR codes so there will be a one to many relationship.likewise one order can have multiple order items that is also a one to many relationship.For Order to QR Address the relationship is 1 to 1. For products and Order Items it's 1 to many.

Cloud Server

- AWS EC2 Instance NGINX Server with Linux Ubuntu Server
- Gunicorn wsgi server
- Database is PostgreSQL
- Database is on Amazon RDS

EC2 Instance is used as our main server for the vending machine and the database which is a PostgreSQL is in the AMAZON RDS .If there are lots of requests with the help of EC2 instance load balancer make sure the requests are efficient to be read. The AWS server is connected and it manages the database on the Amazon RDS.

Security

- Role Based access Control (django.contrib.auth.decorators) is there to make sure every customer service provider will be able to define their roles such as User ,Admin and Super User

- Hashed Passwords   (django.contrib.auth.forms) to make sure the Password is more secure for each and every account.

- CSRF token is used to keep the Content safe.This make sures the Cross site request forgery will not happen to our sites

- For Secured Payment transaction Paypal is used because it is widely popular and very secure payment gate inside the application

- Ratelimit.   (from ratelimit.decorators) Is there to make sure the site does not crash from all the request overloads. And it makes sures the number of API calls are limited.

Reliability

- Email Authentication is there to make sure the email that is given by the user is his/hers and it exists.
- Reset forgot Password to make sure if any of the user forgets his/her credentials they can always get it back using their email.
- Paypal Payments are widely used and there to make sure to increase the reliability of the payment gateway.

Scalability

- Multiple users can access the server at the same time
- Pagination for the items available in the vending machine to make sure the web application Stays Efficient
- PostgreSQL can Easily Manage if the data gets into big data.

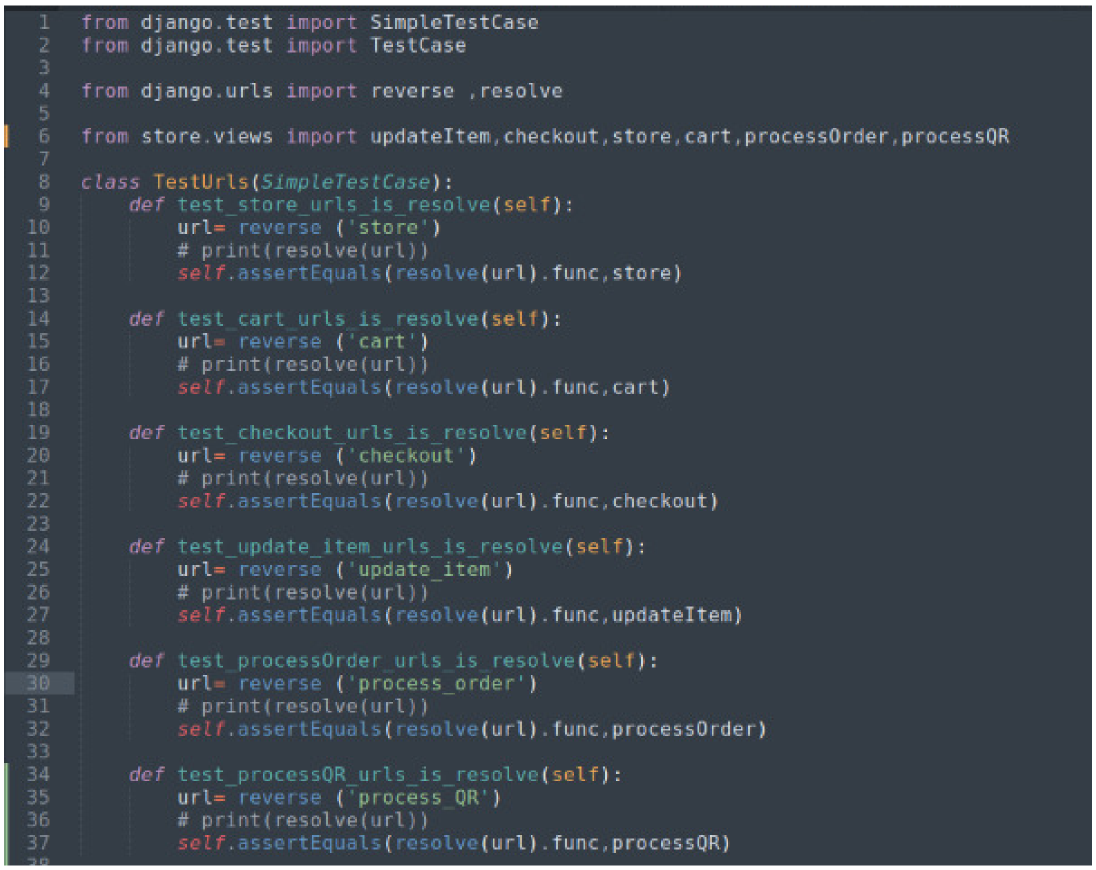
RestAPI

To authenticate we are using JSON web token because it securely transfers information between software and Hardware as an JSON file.It has 2 tokens one is access token which expires within 5 minutes .

We are using Rate limit to make sure the hardware/Web Application does not get overloaded by API calls to the cloud and it is only 60 API calls per hour.It ensure the safety of the systems.We can do GET,PUT,POST,DELETE in the APIs that are available in the system.

Software testing

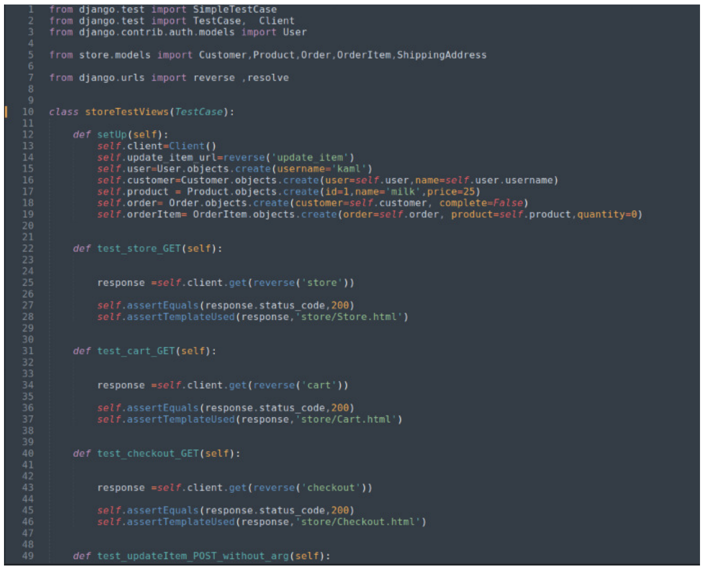
URL Unit Testing



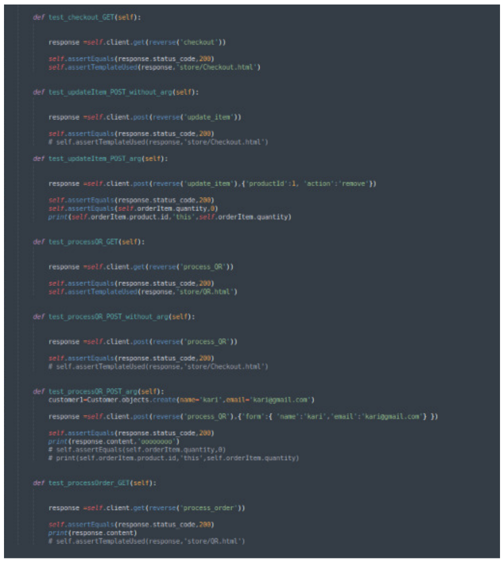
```python
from django.test import SimpleTestCase
from django.test import TestCase

from django.urls import reverse ,resolve

from store.views import updateItem,checkout,store,cart,processOrder,processQR

class TestUrls(SimpleTestCase):
    def test_store_urls_is_resolve(self):
        url= reverse ('store')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,store)

    def test_cart_urls_is_resolve(self):
        url= reverse ('cart')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,cart)

    def test_checkout_urls_is_resolve(self):
        url= reverse ('checkout')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,checkout)

    def test_update_item_urls_is_resolve(self):
        url= reverse ('update_item')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,updateItem)

    def test_processOrder_urls_is_resolve(self):
        url= reverse ('process_order')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,processOrder)

    def test_processQR_urls_is_resolve(self):
        url= reverse ('process_QR')
        # print(resolve(url))
        self.assertEquals(resolve(url).func,processQR)
```

We are checking to make sure every Url gets the reverse Match.By using the dummy http request. To make sure the path of the url is right and requests are coming back as intended.This test will make sure the communication inside/outside the web application can happen without any glitch.

POST/GET Request unit Testing



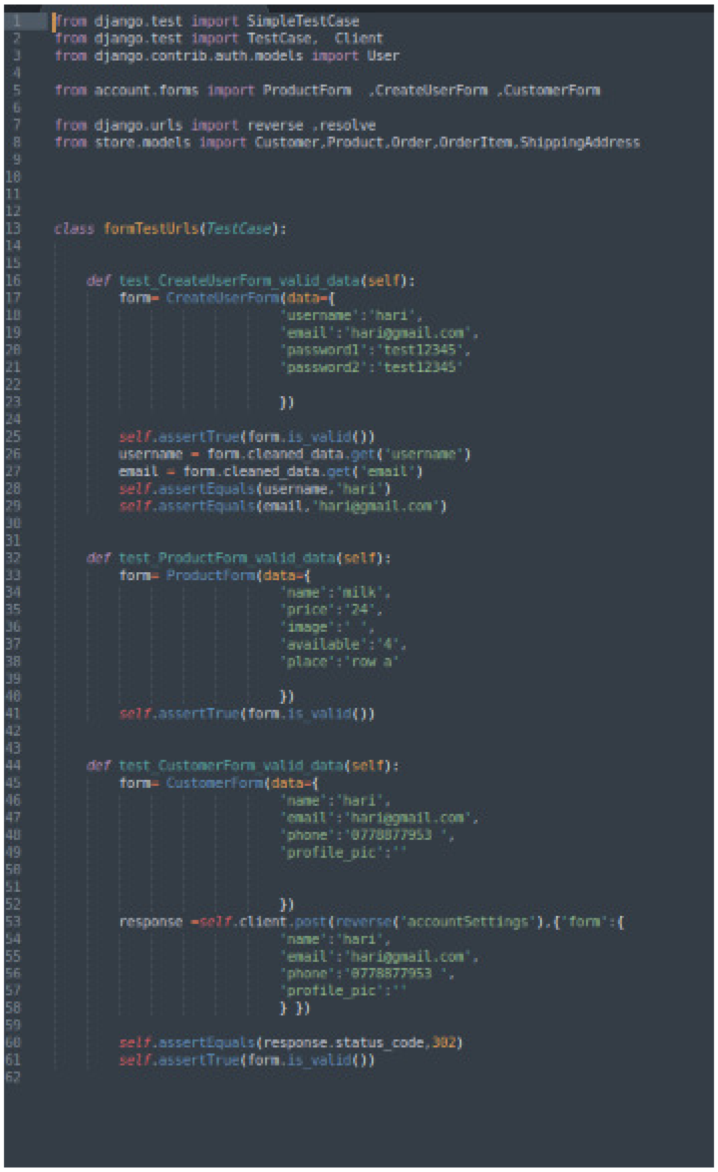
```
from django.test import SimpleTestCase
from django.test import TestCase,  Client
from django.contrib.auth.models import User

from store.models import Customer,Product,Order,OrderItem,ShippingAddress

from django.urls import reverse ,resolve


class storeTestViews(TestCase):

    def setUp(self):
        self.client=Client()
        self.update_item_url=reverse('update_item')
        self.user=User.objects.create(username='kaml')
        self.customer=Customer.objects.create(user=self.user,name=self.user.username)
        self.product = Product.objects.create(id=1,name='milk',price=25)
        self.order= Order.objects.create(customer=self.customer, complete=False)
        self.orderItem= OrderItem.objects.create(order=self.order, product=self.product,quantity=0)


    def test_store_GET(self):


        response =self.client.get(reverse('store'))

        self.assertEquals(response.status_code,200)
        self.assertTemplateUsed(response,'store/Store.html')


    def test_cart_GET(self):


        response =self.client.get(reverse('cart'))

        self.assertEquals(response.status_code,200)
        self.assertTemplateUsed(response,'store/Cart.html')


    def test_checkout_GET(self):


        response =self.client.get(reverse('checkout'))

        self.assertEquals(response.status_code,200)
        self.assertTemplateUsed(response,'store/Checkout.html')


    def test_updateItem_POST_without_arg(self):
```



```
    def test_checkout_GET(self):


        response =self.client.get(reverse('checkout'))

        self.assertEquals(response.status_code,200)
        self.assertTemplateUsed(response,'store/Checkout.html')


    def test_updateItem_POST_without_arg(self):


        response =self.client.post(reverse('update_item'))

        self.assertEquals(response.status_code,200)
        # self.assertTemplateUsed(response,'store/Checkout.html')

    def test_updateItem_POST_arg(self):


        response =self.client.post(reverse('update_item'),{'productId':1, 'action':'remove'})

        self.assertEquals(response.status_code,200)
        self.assertEquals(self.orderItem.quantity,0)
        print(self.orderItem.product.id,'this',self.orderItem.quantity)


    def test_processOR_GET(self):


        response =self.client.get(reverse('process_OR'))

        self.assertEquals(response.status_code,200)
        self.assertTemplateUsed(response,'store/OR.html')


    def test_processOR_POST_without_arg(self):


        response =self.client.post(reverse('process_OR'))

        self.assertEquals(response.status_code,200)
        # self.assertTemplateUsed(response,'store/Checkout.html')


    def test_processOR_POST_arg(self):
        customer1=Customer.objects.create(name='kari',email='kari@gmail.com')

        response =self.client.post(reverse('process_OR'),{'form':{ 'name':'kari','email':'kari@gmail.com'} })

        self.assertEquals(response.status_code,200)
        print(response.content,'oooooooo')
        # self.assertEquals(self.orderItem.quantity,0)
        # print(self.orderItem.product.id,'this',self.orderItem.quantity)


    def test_processOrder_GET(self):


        response =self.client.get(reverse('process_order'))

        self.assertEquals(response.status_code,200)
        print(response.content)
        # self.assertTemplateUsed(response,'store/OR.html')
```

CRUD operations are Checked using artificial AJAX API calls for that we are using Hard coded Json data.We are sending in dummy data to check the reliability of the authendication.To make sure the request are done right and no one can access the system with dummy data that can be used to hack the system.

Form Validation Testing

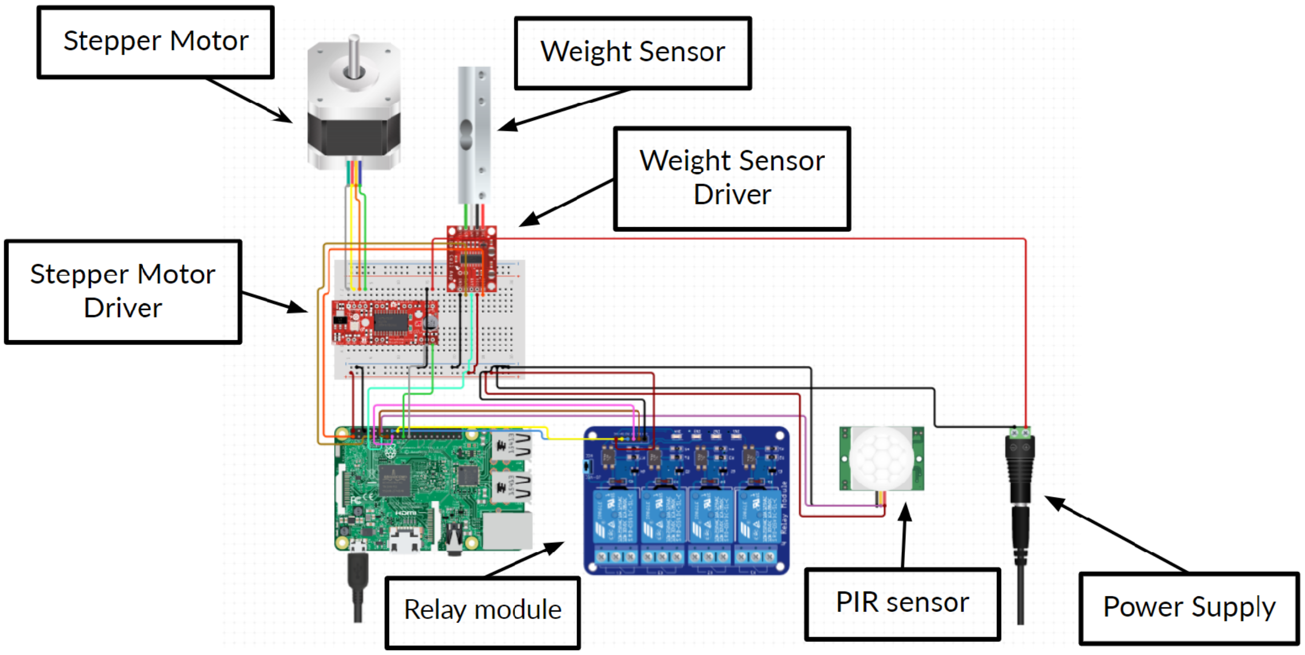


```python
from django.test import SimpleTestCase
from django.test import TestCase,  Client
from django.contrib.auth.models import User

from account.forms import ProductForm  ,CreateUserForm ,CustomerForm

from django.urls import reverse ,resolve
from store.models import Customer,Product,Order,OrderItem,ShippingAddress



class formTestUrls(TestCase):


    def test_CreateUserForm_valid_data(self):
        form= CreateUserForm(data={
                                    'username':'hari',
                                    'email':'hari@gmail.com',
                                    'password1':'test12345',
                                    'password2':'test12345'

                                    })

        self.assertTrue(form.is_valid())
        username = form.cleaned_data.get('username')
        email = form.cleaned_data.get('email')
        self.assertEquals(username,'hari')
        self.assertEquals(email,'hari@gmail.com')


    def test_ProductForm_valid_data(self):
        form= ProductForm(data={
                                    'name':'milk',
                                    'price':'24',
                                    'image':' ',
                                    'available':'4',
                                    'place':'row a'

                                    })
        self.assertTrue(form.is_valid())


    def test_CustomerForm_valid_data(self):
        form= CustomerForm(data={
                                    'name':'hari',
                                    'email':'hari@gmail.com',
                                    'phone':'0778877953 ',
                                    'profile_pic':''


                                    })
        response =self.client.post(reverse('accountSettings'),{'form':{
                                    'name':'hari',
                                    'email':'hari@gmail.com',
                                    'phone':'0778877953 ',
                                    'profile_pic':''
                                    } })

        self.assertEquals(response.status_code,302)
        self.assertTrue(form.is_valid())
```

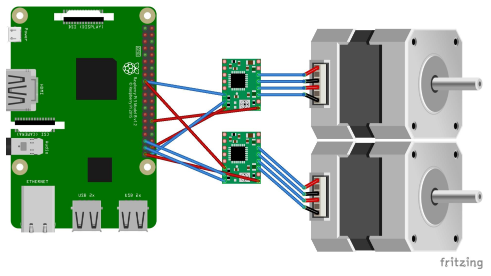
Authentication form is checked.We are creating a new user and database and checking the form. Post request is sent and check if it is updated. New products are created in example to check where all of them are being updated. To check if the forms are working fine and the authentication processes are working fine.

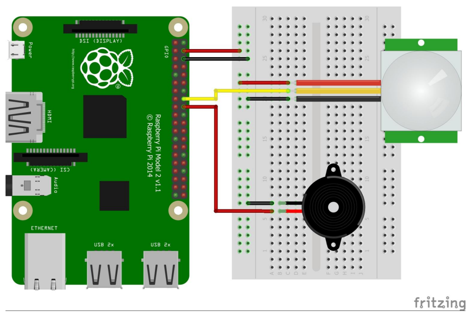
Hardware Overview

Circuit Diagram



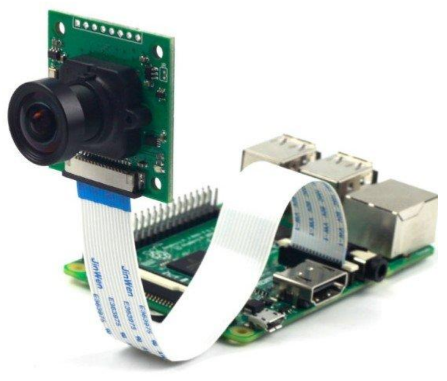
This is the physical interpretation of the circuit diagram Which shows how the components are connected.

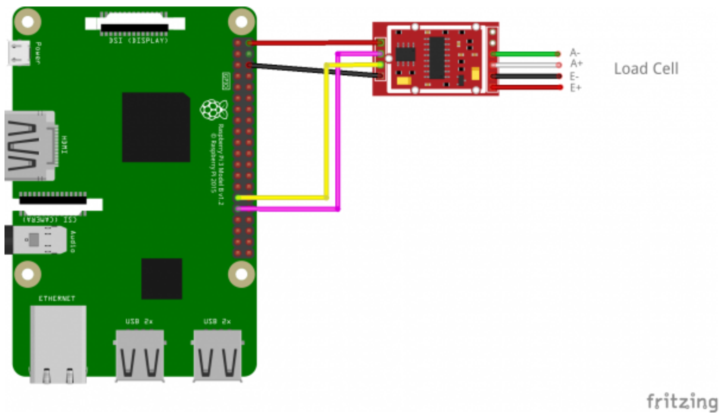




Stepper motor and stepper motor driver

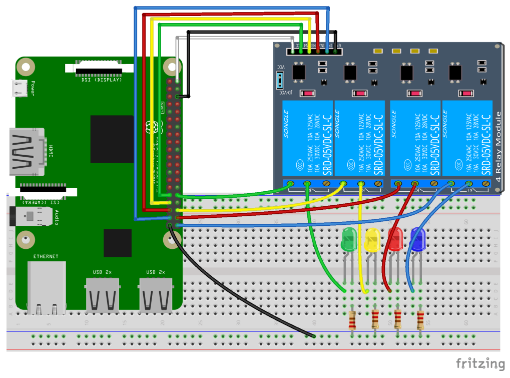
PIR sensor





Camera Module

Weight Sensor



Relay module

Raspberry Pi 3

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz 1GB RAM
- BCM43143 WiFi on board Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO , 4 x USB 2 ports 4 pole
- Stereo output and Composite video port Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps) Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position

Stepper Motor

- Motor Type: Bipolar Stepper
- Step Angle: 1.8 deg.
- Holding Torque: 40N.cm (56oz.in)
- Rated Current/phase: 1.7A
- Phase Resistance: 1.5Ohm±10%
- Insulation Resistance: 100MΩ¸ Min, 500VDC
- Insulation Strength: 500VAC for one minute

Stepper motor driver

- A stepper motor provides a constant holding torque without the need for the motor to be powered.Steppers provide precise positioning and repeatability of movement since good stepper motors have an accuracy of 3 – 5% of a step and this error is non-cumulative from one step to the next.
- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance

Camera Module V2 for Raspberry Pi

- 5 megapixel native resolution sensor-capable of 2592 x 1944 pixel static images.
- Supports 1080p30, 720p60 and 640x480p60/90 video.
- Camera is supported in the latest version of Raspbian, Raspberry Pi's preferred operating system.

Relay Module

- High-sensitivity (250 mW) and High-capacity (16 A) versions
- Rated voltage 12 V DC
- Rated current 20.8 mA
- Coil resistance 576 Ω
- Must operate voltage 75% max. of the rated voltage
- Must release voltage 10% min. of the rated voltage
- Max. voltage 180% of rated voltage (at 23°C)
- Power consumption Approx. 250 mW

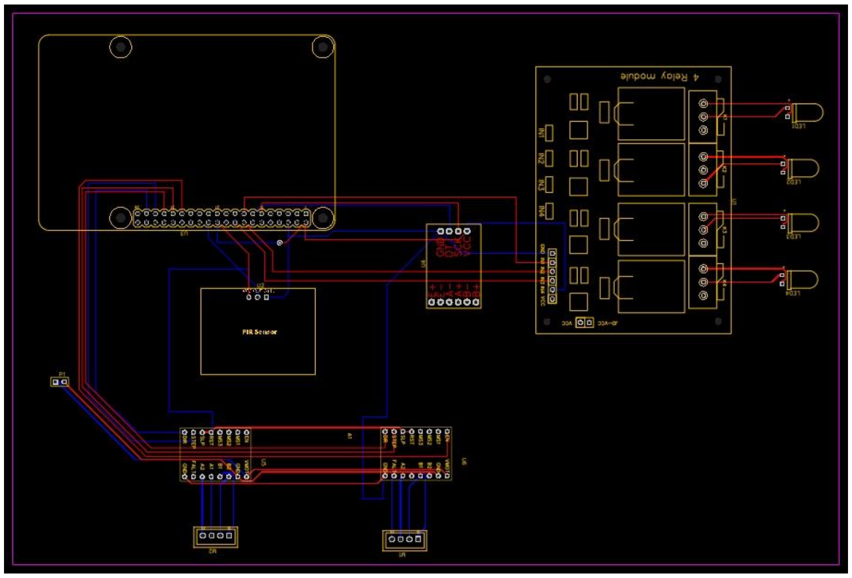
Weight Sensor

- Differential input voltage: ±40mV (Full-scale differential input voltage is ± 40mV)
- Data accuracy: 24 bit (24 bit A / D converter chip.)
- Refresh frequency: 10/80 Hz.
- Operating Voltage: 2.7V to 5V DC.
- Operating current: < 10 mA.
- Size: 24x16mm.

PIR sensor

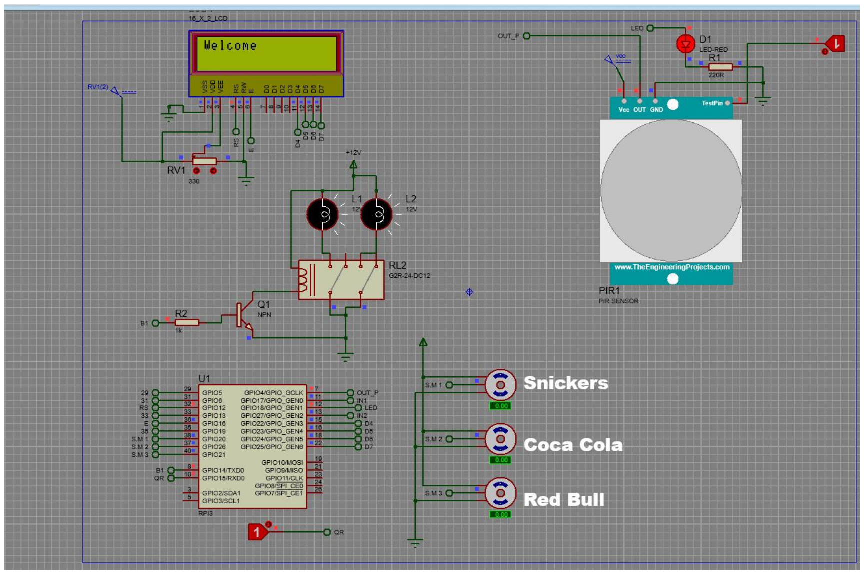
- Input voltage: DC 4.5~20V
- Static current: 50uA
- Output signal: 0,3V (Output high when motion detected)
- Sentry angle: 110 degree
- Sentry distance: max 7 m
- 120 degree detection angle
- Low power consumption in idle mode only 50uA and 65mA in fully active mode.

PCB Design



The EasyEDA app was used to create the PCB design . here we used all the basic libraries available to create this PCB design. It is double sided printing and we are using PCB printing method to get the PCB.For the pcb components we use raspberry pi 3 module and for the stepper motor we use bipolar stepper motor , for the stepper motor carrier we use DRV8825 motor driver carrier and we use HX711 weight sensor and we use relay module JD-VCC and pir sensor library.

Schematic View



Hardware Testing

PIR Testing

When the PIR detects motion, the output pin will go "high" to 3.3V and light up the LED.When you have the breadboard wired up, insert batteries and wait 30-60 seconds for the PIR to 'stabilize'. During that time the LED may blink a little. Wait until the LED is off and then move around in front of it, waving a hand, etc, to see the LED light up.

The Adafruit PIR has a trimpot on the back for adjusting sensitivity. You can adjust this if your PIR is too sensitive or not sensitive enough - clockwise makes it more sensitive.There are two 'timeouts' associated with the PIR sensor. One is the "Tx" timeout: how long the LED is lit after it detects movement - this is easy to adjust on Adafruit PIR's because there's a potentiometer.

Raspberry pi Testing

This is the main thing we have to do before the hardware assembles. Raspberry pi 3 is the main processor of our project. So we can do some of the testing and calibration for better working.

1. **Checking the power cable and HDMI cable**

Checking these according to the raspberry pi 3 handbook using multi meter and check is it okay or not.

2. **Checking the LEDs**

When a Raspberry Pi boots, one or more LEDs will activate. One is red, indicating power (PWR); the other is green, and indicates activity (ACT). (There is also a trio of green Raspberry Pi LED lights indicating the Ethernet status, if connected.) So, what do these LEDs indicate? Well, there's the normal status, which is both PWR and ACT LEDs activated. ACT flashes during SD card activity. Therefore, if there's no green light on your Raspberry Pi, there's a problem with the SD card.

Meanwhile, PWR blinks when power drops below 4.65V. As such, if the Raspberry Pi's red light doesn't light up, there's no power.

If only the red PWR LED is active, and there is no flashing, then the Pi is receiving power, but there is no readable boot instruction on the SD card (if present). On a Raspberry Pi 2, ACT and PWR LEDs lit up means the same.

When booting from an SD card, the Raspberry Pi's green ACT light should have an irregular blink. However, it can blink in a more regulated manner to indicate a problem:

- 3 flashes: start.elf not found
- 4 flashes: start.elf cannot launch, so it's probably corrupted. Alternatively, the card is not correctly inserted, or the card slot is not working.
- 7 flashes: kernel.img not found
- 8 flashes: SDRAM not recognized. In this case, your SDRAM is probably damaged, or the bootcode.bin or start.elf is unreadable.

3. **Is the power adapter suitable?**

Power issues can cause a Raspberry Pi to fail. It might switch off or hang when running, or it might simply fail to boot at all. To read the SD card accurately, a stable power supply unit (PSU) is required. To ensure your PSU is good enough, check that it meets the specification of your Raspberry Pi model. Similarly, confirm that the micro-USB from the PSU to the Pi is up to scratch. A lot of people use smartphone chargers to power their Raspberry Pi. This usually isn't the best idea; a dedicated, suitable PSU is the preferred approach.

4. **Have you installed the operating system?**

Check on the OS install or not if not check the correct version and install it properly.

5. **Is the microSD card reliable?**

A working Raspberry Pi will rely on a good quality SD card for booting and running the OS. If the SD card isn't working, then your Raspberry Pi will be erratic, or simply fail to boot. Start by checking the card works. You can do this by powering down the Pi and inserting the SD card into your PC. Use a reliable flash drive formatting tool, and attempt to reformat.

When setting up a new Raspberry Pi OS, always format the SD card prior to writing the image. This means using a reliable card reader/writer, as well as suitable media. Look for media with a high write speed, too, and superior error checking, to ensure a fast, efficient Raspberry Pi.

Budget

Raspberry Pi 3	1	7000	7000
Stepper Motor	2	1750	3500
Stepper motor driver	2	800	1600
Camera Module V2 for Raspberry Pi	1	1800	1800
Relay Module 12v 4 channel	1	450	450
HX711 Weight Sensor	1	650	650
HX711 Weight Sensor driver	1	190	190
PIR sensor	1	350	350
LED Strips	3	250	250
12V 2A DC Power Supply	1	300	300
Misc wire, 1ft 22-24 AWG	1	300	300
5.1 x2.2 mm DC panel mount jack-1	1	100	100
Fiber Glass	1	2000	2000
			18490