

# **RISHAB'S APP**

**UIT1511 – Software Design Lab**

**A PROJECT REPORT**

**Submitted by**

**Tarun Prasad-205002112**

**Thanseer Hishak-205002113**

**G Udaykiron Chowdary- 205002114**

**Umarani -205002115**

**SSN COLLEGE OF ENGINEERING,**

**KALAVAKKAM**

**AUGUST 2022**

SOFTWARE ENGINEERING LAB

# Rishab's App

## Food ordering system

---



### Members

Thanseer Hishak  
Tarun Prasad  
G Udaykiron Chowdary  
Umarani K

# **RISHAB'S APP**

## **1. Abstract of the project**

### **1.1 Problem Description:**

These days the students are living a busy life. Thus they don't have time to stand in long queues. Rishab is a drink shop receives lot of orders everyday but students and teachers have to wait in long queues to order their drinks. Thus our project aims to solve this by providing an app to order their food which helps the students/teachers to skip long queues.

### **1.2 Motivation:**

The project's motivation is to design a software application to solve the waiting time in queues. This application helps reduce the wait time to order making it easier for the students/teachers.

### **1.3 Objective:**

The main objective of the software is to create a web-app that can handle the order requests and generate the bill, enabling users to order drinks online..

### **1.4 Client Description:**

Rishab is a juice shop that has lots of drinks and food items to sell. It is located in SSN College of Engineering thus often busy/crowded during breaks. This software will provide a hassle free travelling experience.

## **2.INTRODUCTION:**

### **2.1 Need for the software :**

Rishab accepts orders only through direct mode.

We aim to reduce the wait time by making the ordering process online using our software.

### **2.2 Project Management:**

It is required by the student to have internet access in order to access our facilities.

### **2.3 Deliverables :**

Fully working web-app that can accept orders and generate the bill and provide tokens.

### **3. REQUIREMENTS ENGINEERING:**

#### **3.1 Client Details:**

The client of our project will not be a single entity, but a community or a campus. Our project will be useful for institutions like SSN CE, where students and faculty visit the shop and generate orders for food frequently.

#### **3.2 List of Functional Modules:**

1. Login module
2. Catalogue module
3. Order summary module
4. Dashboard module
5. Feedbackmodule

## Login :

The customer and admin have to login to the system by giving name and password . if the customer is new user then he or she has to register to the website by entering the otp that is send to the user mail.

## Catalogue :

The products are displayed to the user based on categories like juice ,hot drinks and snacks these are categories and displayed to the customer in the catalogue page.

## Order :

In the order page the customer will order the snacks or drinks based on the displayed catalogue . Here the customer can also delete their orders and confirm orders.after finished the ordering process it will show the feedback form so that the customer can give their feedback about the services.

## Dashboard :

This module is used by the admin to process the pastorder and current order. The admin can also do the confirm paid order confirm deliveredorder and can also cancel the order. The admin will also view the feedback that was given by the customer.

## Implementation :

| Sprint# | Epic  | User Story # | Requirement / User Story   | Essential or Desirable | Description of the Requirement   | Remarks |
|---------|-------|--------------|--|------------------------|--|---------|
| 1       | Login | #01          | The details of the user must be recorded in the database   | Essential              | New user must provide the details like his email id, phone number, username and password   |         |
|         |       | #02          | Login page must contain two text fields for entering username and password and a button for submission | Essential              | User must be able to fill necessary details for login in the required fields. User must be able to login using his username and password |         |
|         |       | #03          | User profile must be displayed to the user once he clicks on it.                                       |                        | User once logged in must be able to view his profile. The user profile must contain the details of                                       |         |

|   |           |     |  |           |   |  |
|---|-----------|-----|--|-----------|---|--|
|   |           |     |  |           | the user like his/her email id,phone number and his username  |  |
|   | Catalogue | #04 | There must be a picture for the particular food product and description about the item and its price | Essential | User must be able to view all the items available in the store.User must be aware about the available items in the store. |  |
|   |           |     |  |           |   |  |
| 2 | Order     | #05 | The items which the user orders are recorded and stored as an order list                             | Essential | When the user sees an item available,h e must be able to order that item and it should be added to the order list         |  |
|   |           | #06 | The name of the customer along with names of the items they  | Essential | The user must be able to view his order list inorder to verify his orders.  |  |



|   |                     |     |  |           |  |  |
|---|---------------------|-----|--|-----------|--|--|
|   |                     |     | ordered along with its quantity must be displayed  |           |  |  |
| 3 | Bill generation     | #07 | The total amount for all the items must be calculated and must be displayed                              | Essential | User must be able to view the bill which contains the prices for the items that he ordered.    |  |
|   | order ID generation | #08 | The order id for the particular order must be generated for the user                                     | Essential | Admin must be verify whether the user has paid the bill and generate the token id for the user |  |
| 4 | Thank you           | #09 | The thank you mail must be generated and should be sent to the user along with the items he/she ordered. | Desirable | User must receive a thank you mail after he has order the items                                |  |

## Test cases :

| TC id | RS  | Test case description                                    | Testcase input   | Expected output    | result(pass/fail) |
|-------|-----|--|--|--------------------|-------------------|
| 1     | #01 | Enter correct username and password and hit login button | The user's username which contains only alphabets or numbers | Login success      | pass              |
| 2     | #01 | Enter invalid username and password                      | The username contains special characters                     | Invalid login      | pass              |
| 3     | #01 | Enter a username and invalid password                    | The user enters password that doesn't match for username     | Invalid password   | pass              |
| 4     | #02 | The quantity of an item doesn't get updated              | Increase the quantity of an item                             | Item not available | pass              |

|   |     |  |                                  |                                       |      |
|---|-----|--|----------------------------------|---------------------------------------|------|
| 5 | #02 | Item ordered is available                                    | Increase the quantity of an item | Item available                        | pass |
| 6 | #02 | Items ordered are displayed                                  | Click the button generate order  | Order summary is displayed            | pass |
| 7 | #02 | Incorrect items ordered are displayed/<br>no items displayed | Click the button generate order  | Order summary not displayed           | pass |
| 8 | #03 | Incorrect price matched with an item                         | Order an item                    | Wrong price is displayed for the item | pass |

## 5. Risk management :

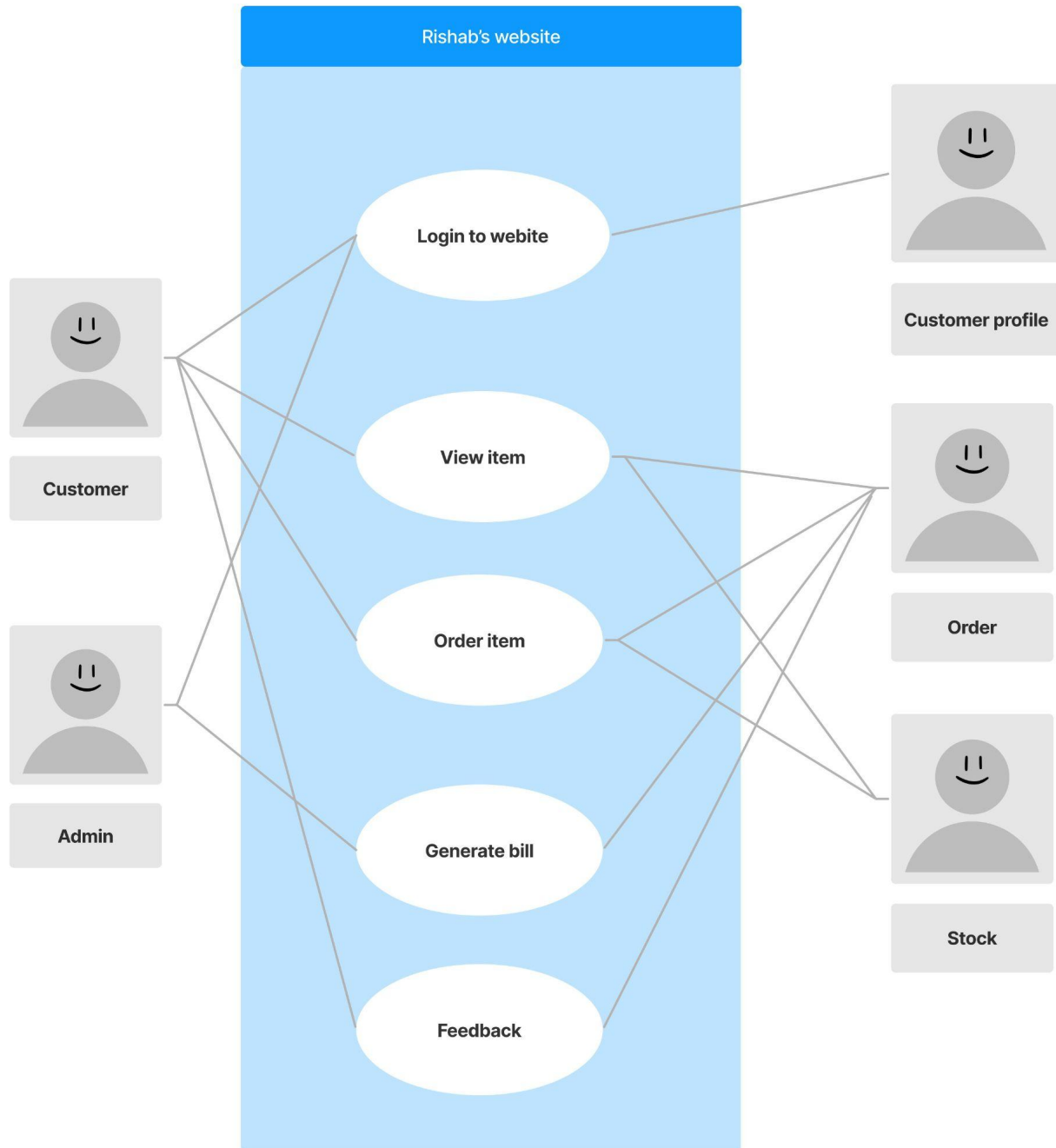
| Risk # | Risk description                                      | probability | impact  | Mitigation plan  |
|--------|---|-------------|---|--|
| 1      | User is able to login using invalid login credentials | 0.2         | Invalid access to the website                       | Check if login credentials exist   |
| 2      | User is not able to login                             | 0.3         | User not able to visit the website                  | Make sure that login id s and passwords are stored                           |
| 3      | The items are not displayed in the catalogue          | 0.3         | User is not able to order                           | Add the items that are available for ordering in the catalogue               |
| 4      | The bill generated has wrong order summary            | 0.2         | User gets the wrong amount as bill                  | The items ordered must be continuously kept tracked along with their prices. |
| 5      | The app is down/not working but the orders are taken  | 0.2         | User orders the wrong items and gets the wrong bill | Make sure no orders from customer are taken until the app works again        |

|   |  |     |                                     |   |
|---|--|-----|-------------------------------------|---|
| 6 | Token id is generated without generation of bill | 0.3 | Order is not valid                  | Make sure the payment is received before generation of token id |
| 7 | Same token id generated for multiple customers   | 0.3 | Delivery of orders become difficult | Make sure unique token id s are generated for each customer     |

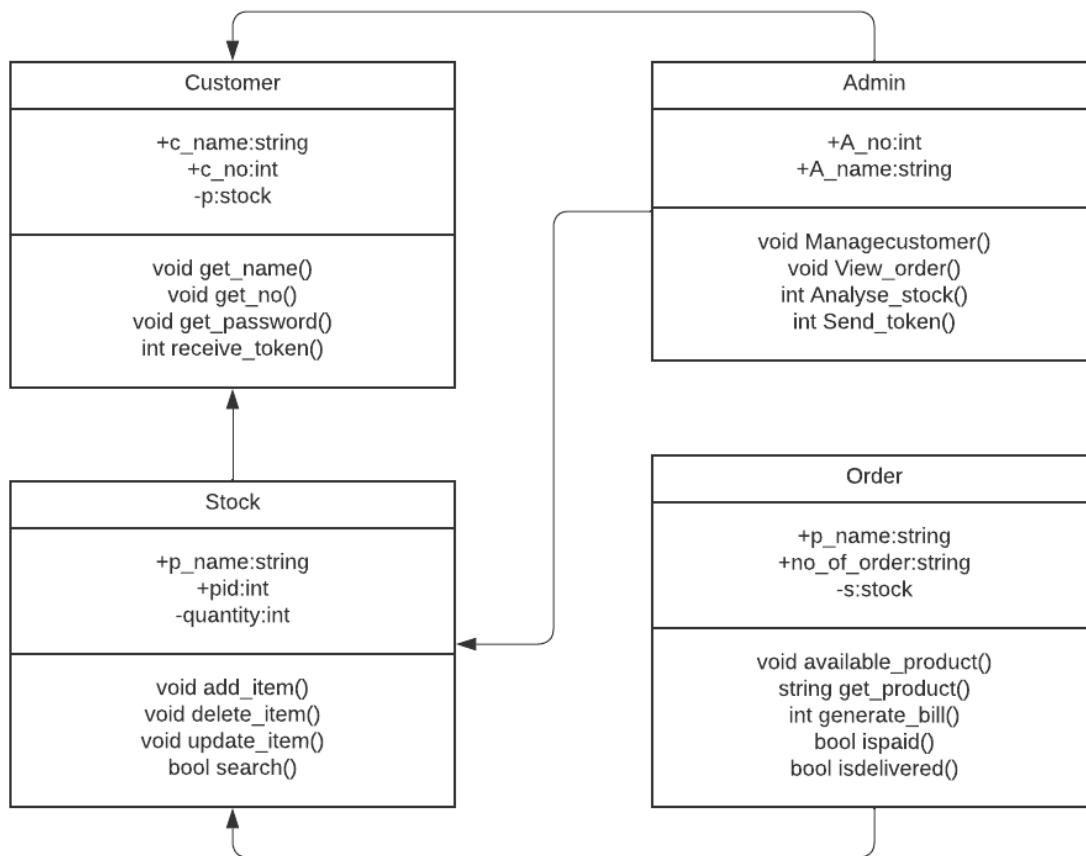
## 6. Diagrams :

Use case diagram

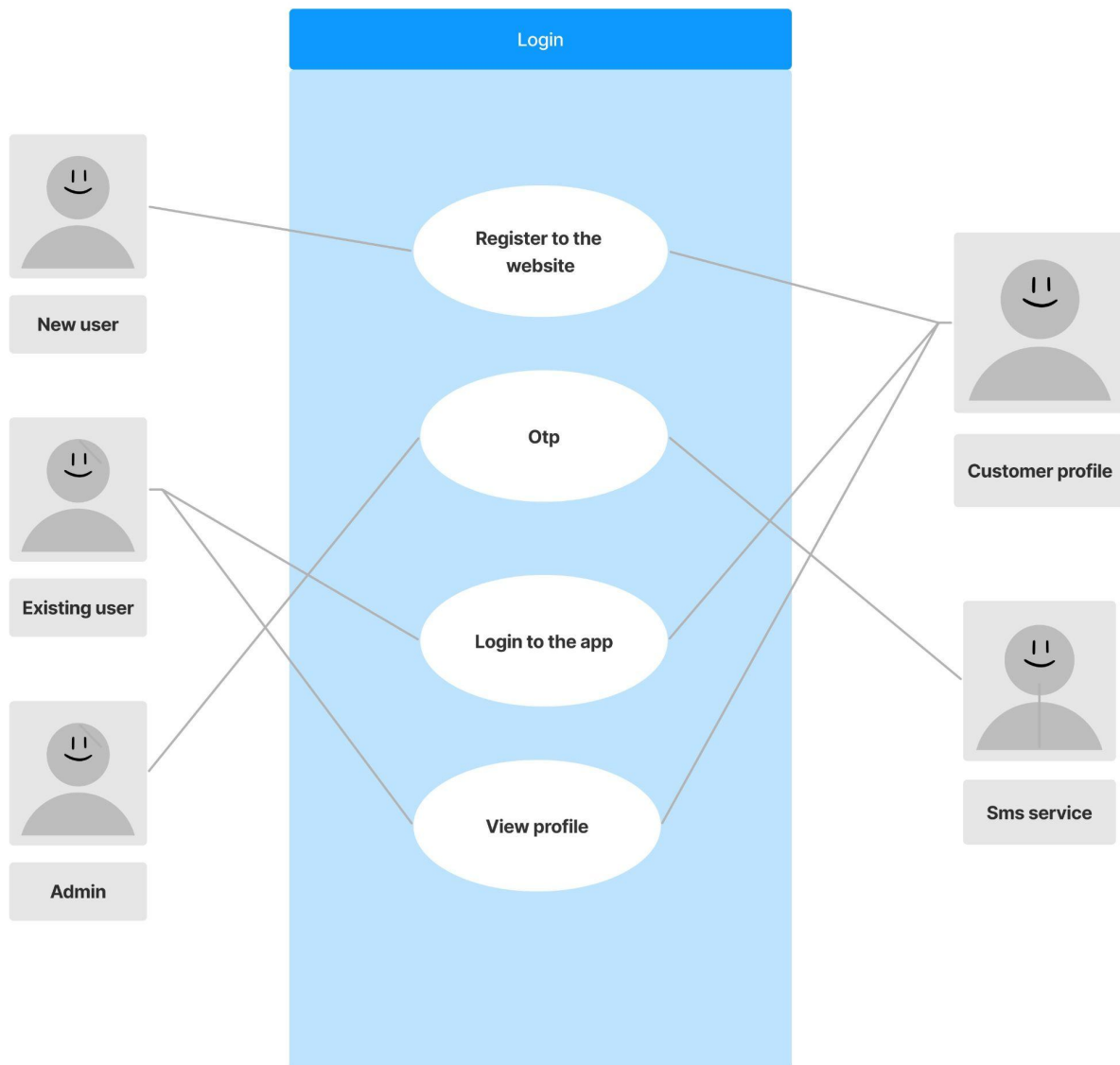
Rishab's website



## CLASS DIAGRAM :

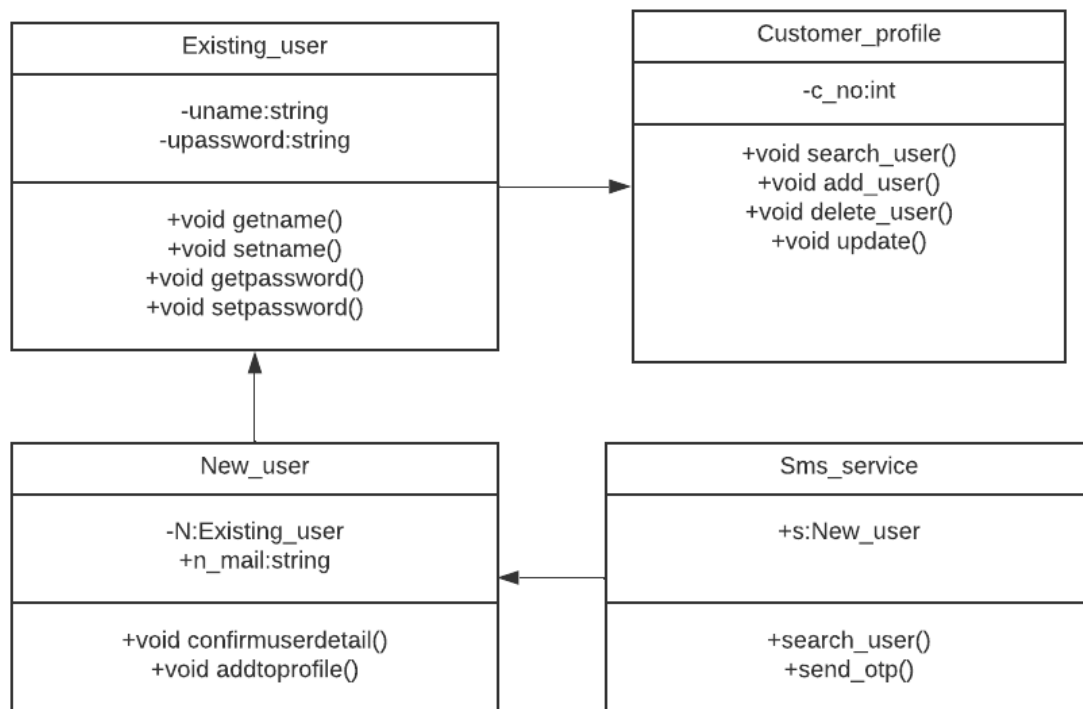


## Login module use case diagram :

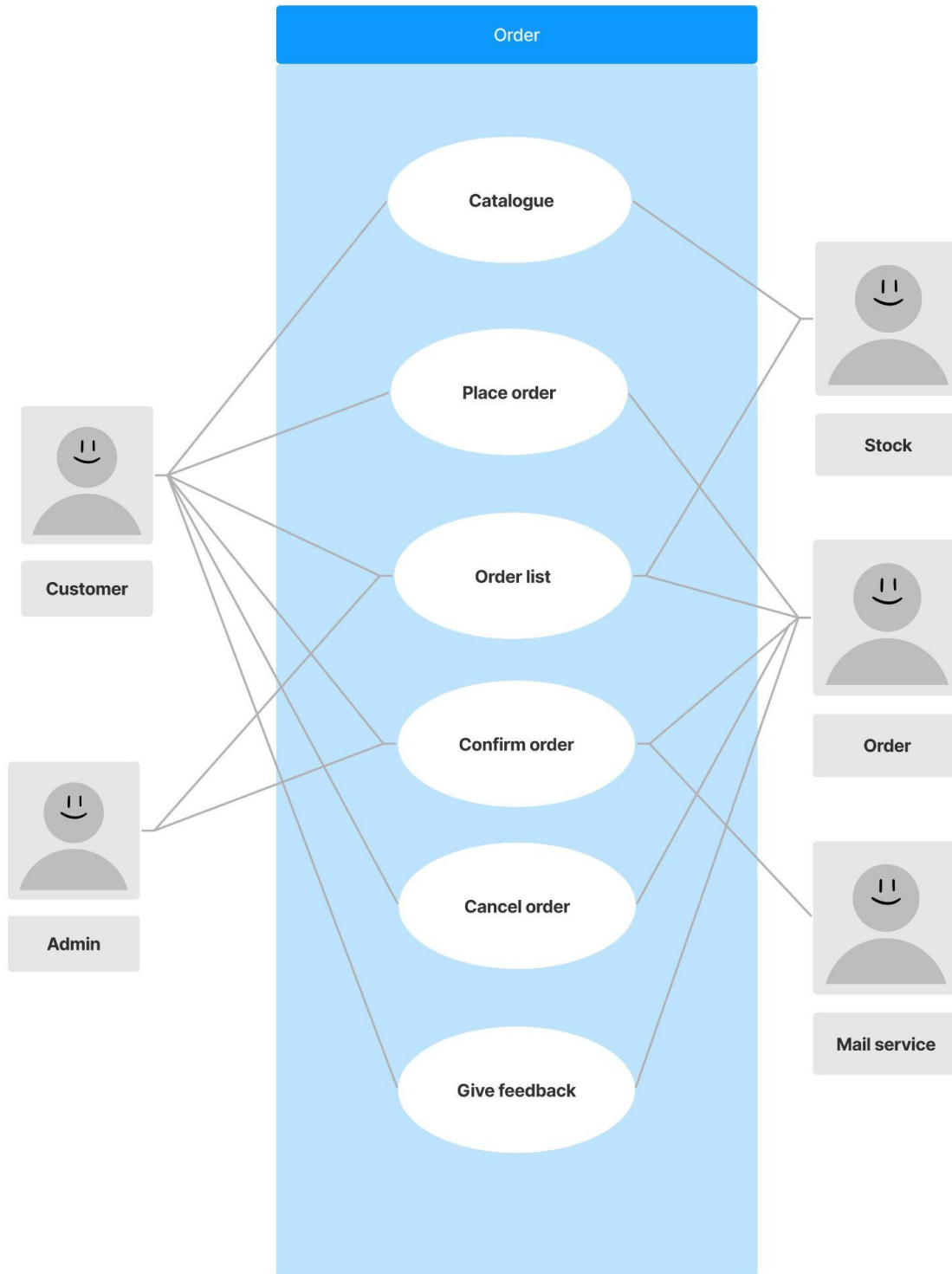




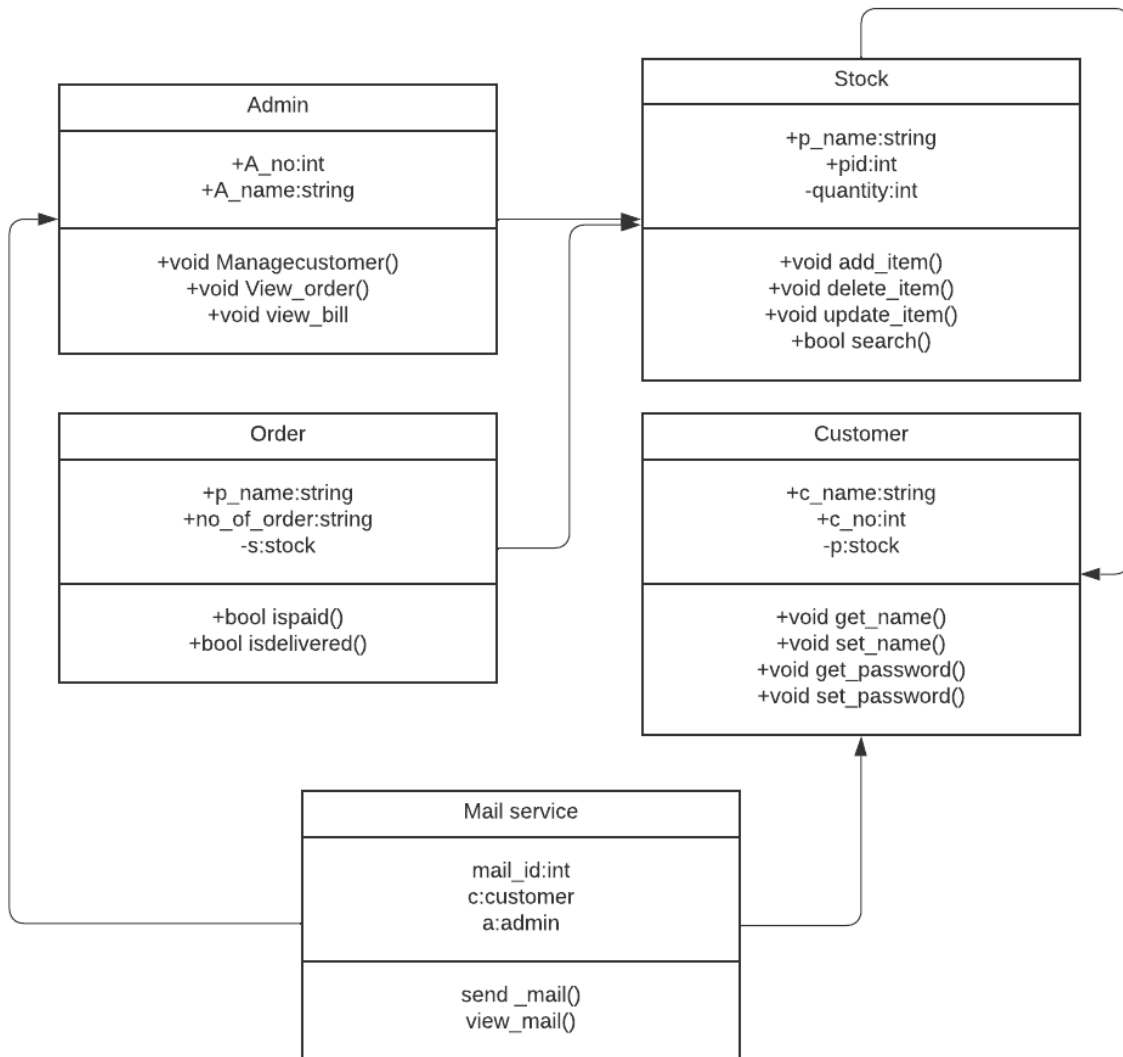
## Login module class diagram :



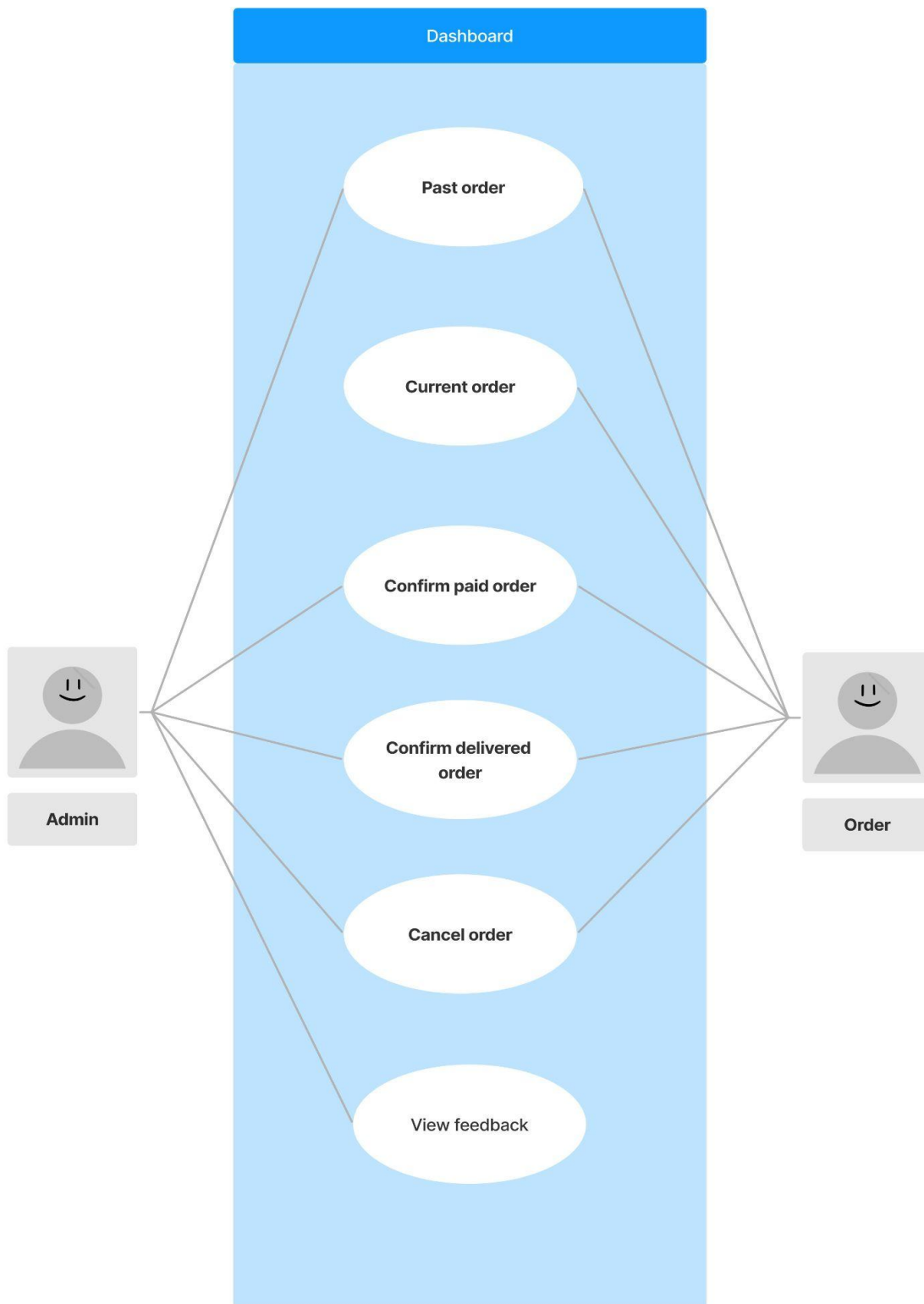
## Order module use case diagram :



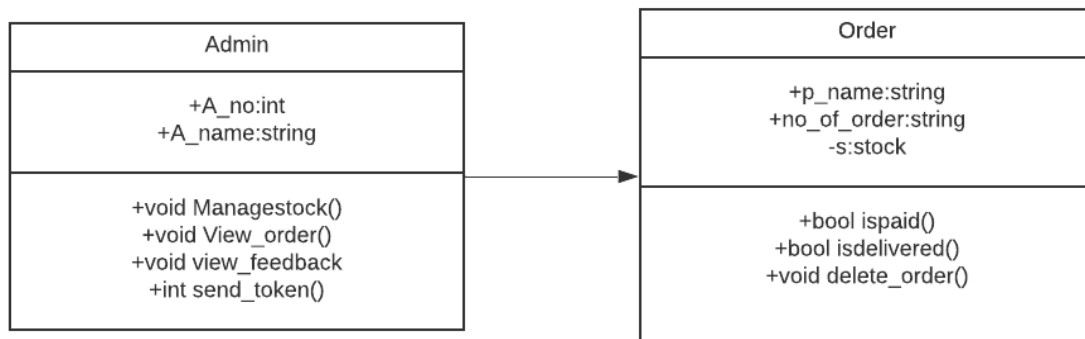
## Order module class diagram :



## Dashboard module use case diagram :



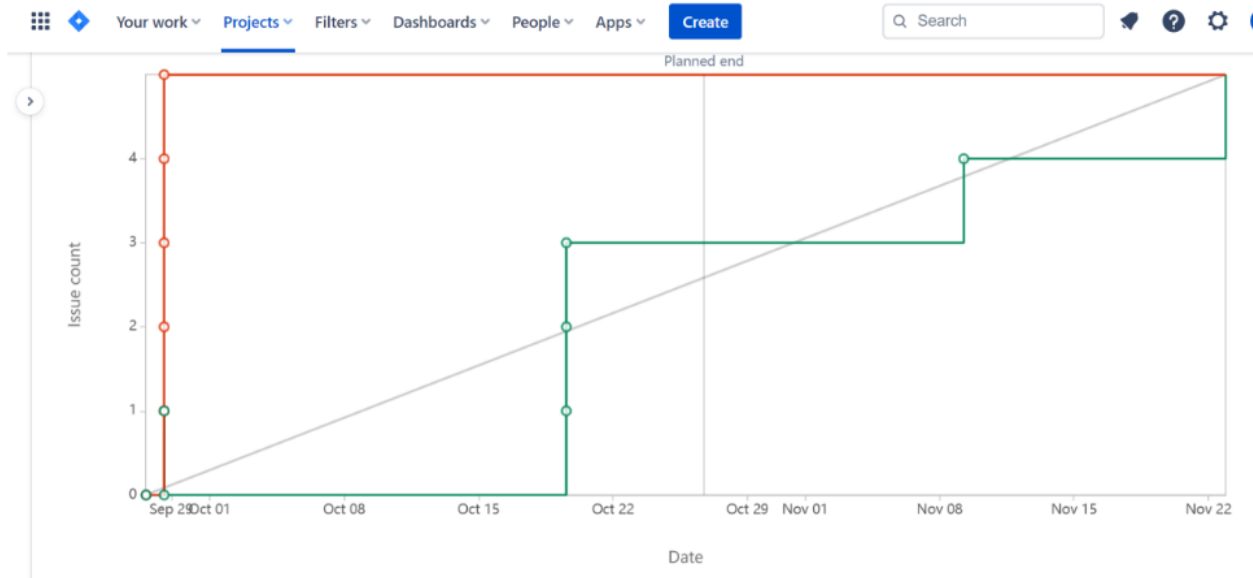
## Dashboard module class diagram :



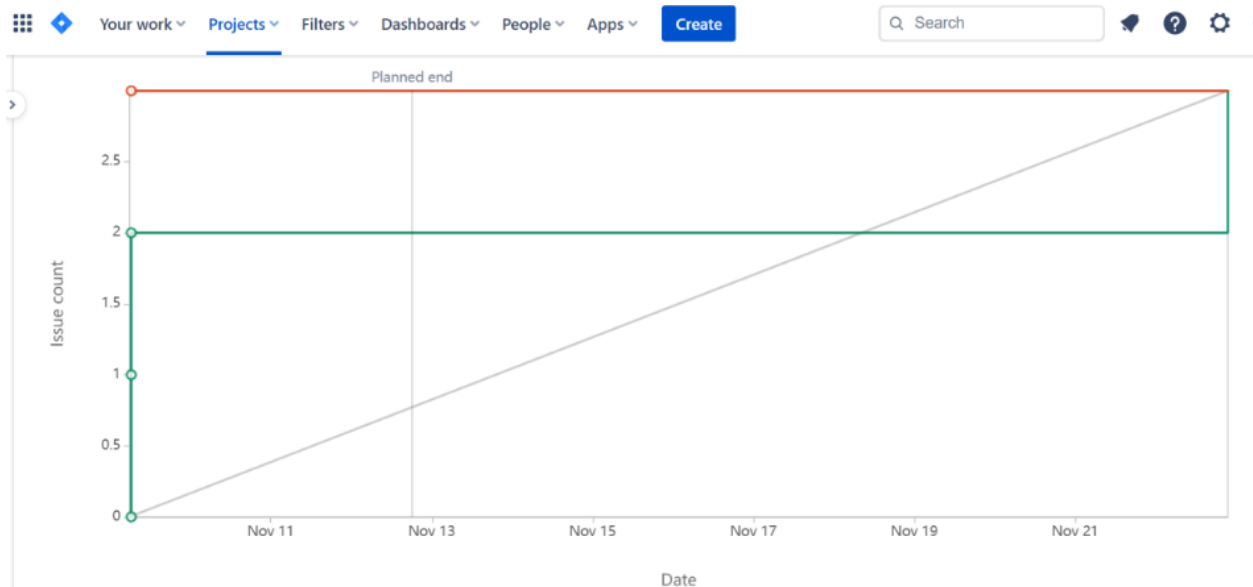
## 7. Project progress

Burn up chart :

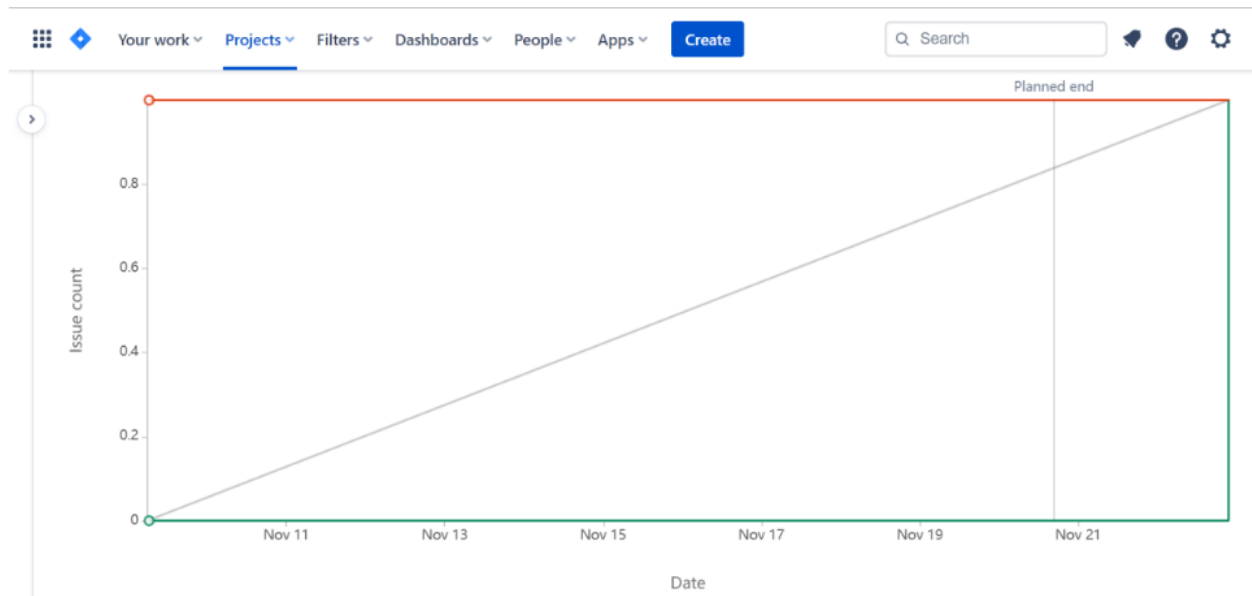
Sprint 1:



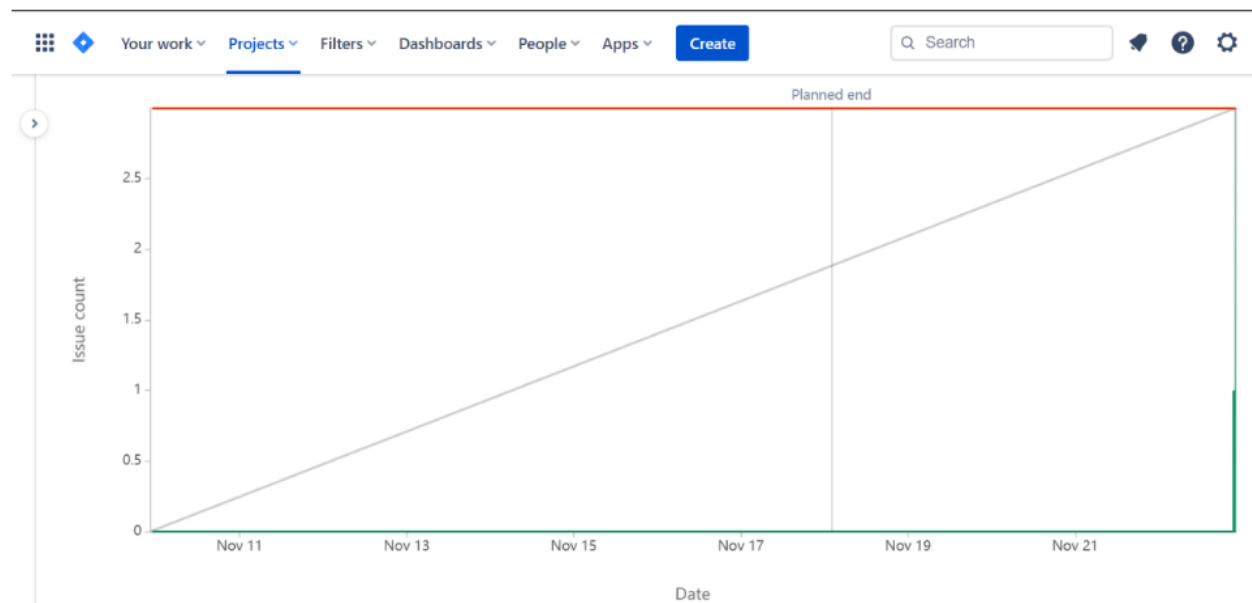
Sprint 2 :



## Sprint 3 :

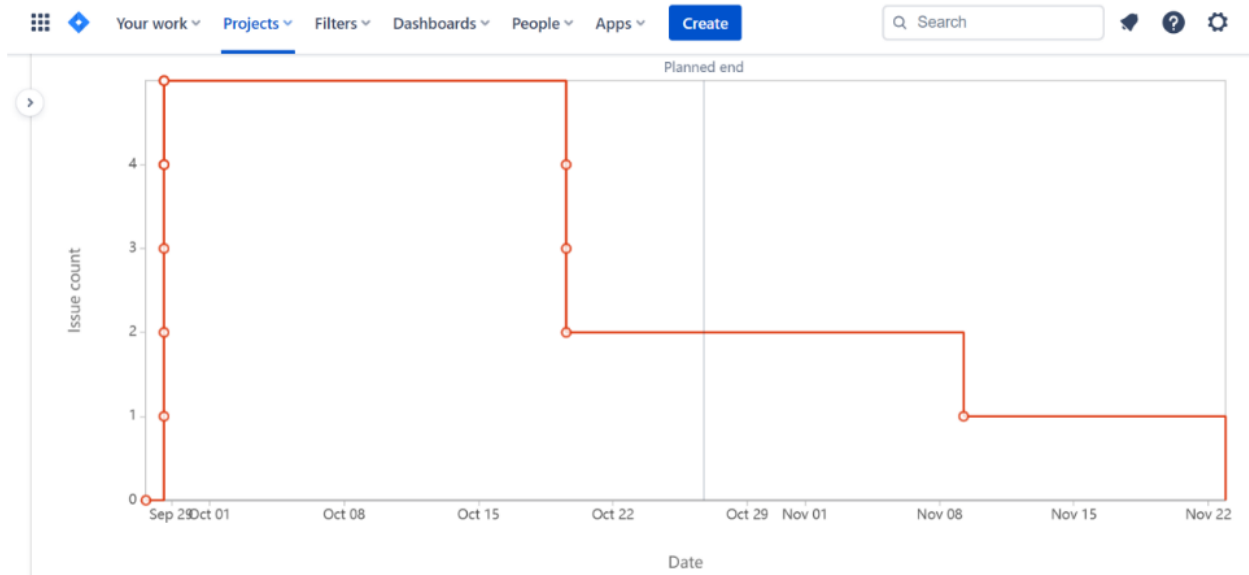


## Sprint 4 :

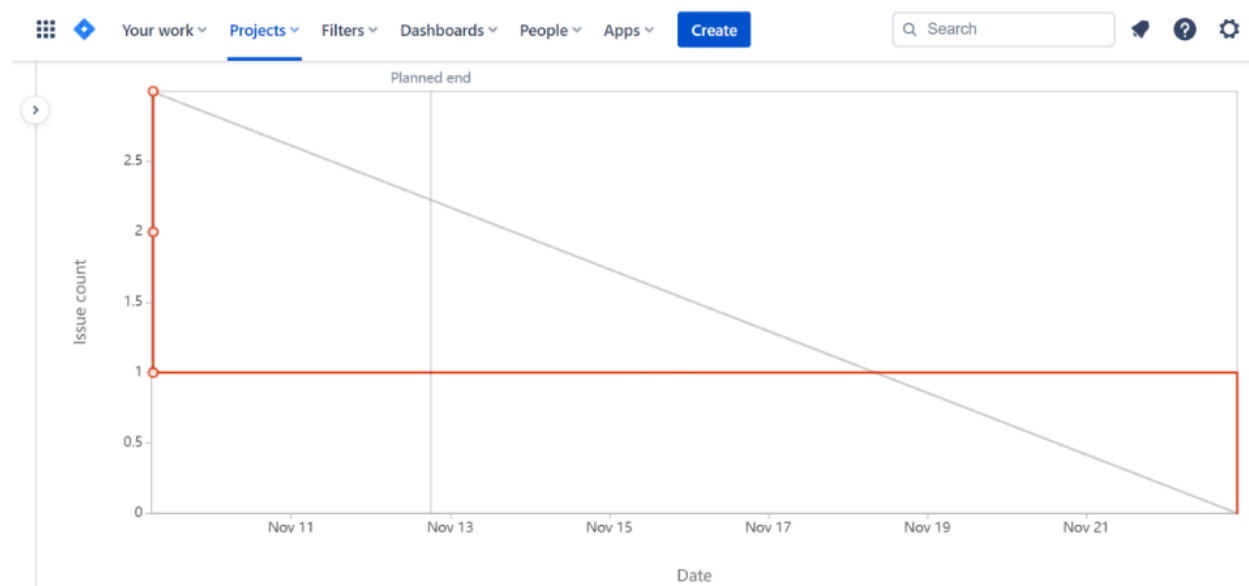


## Burn down chart :

### Sprint 1 :

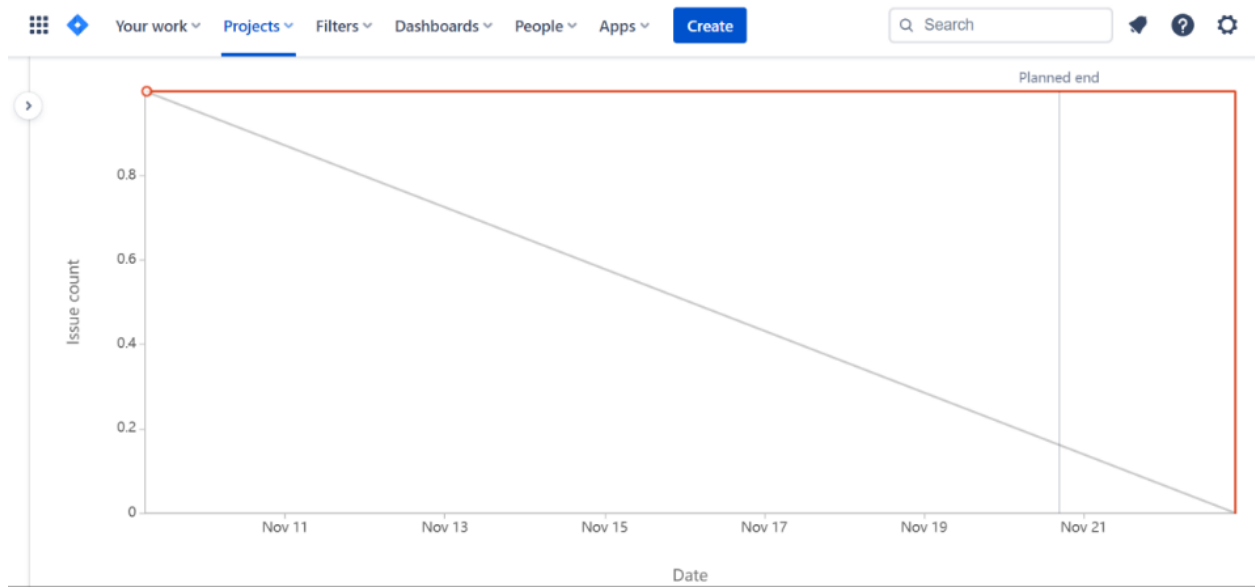


### Sprint 2 :

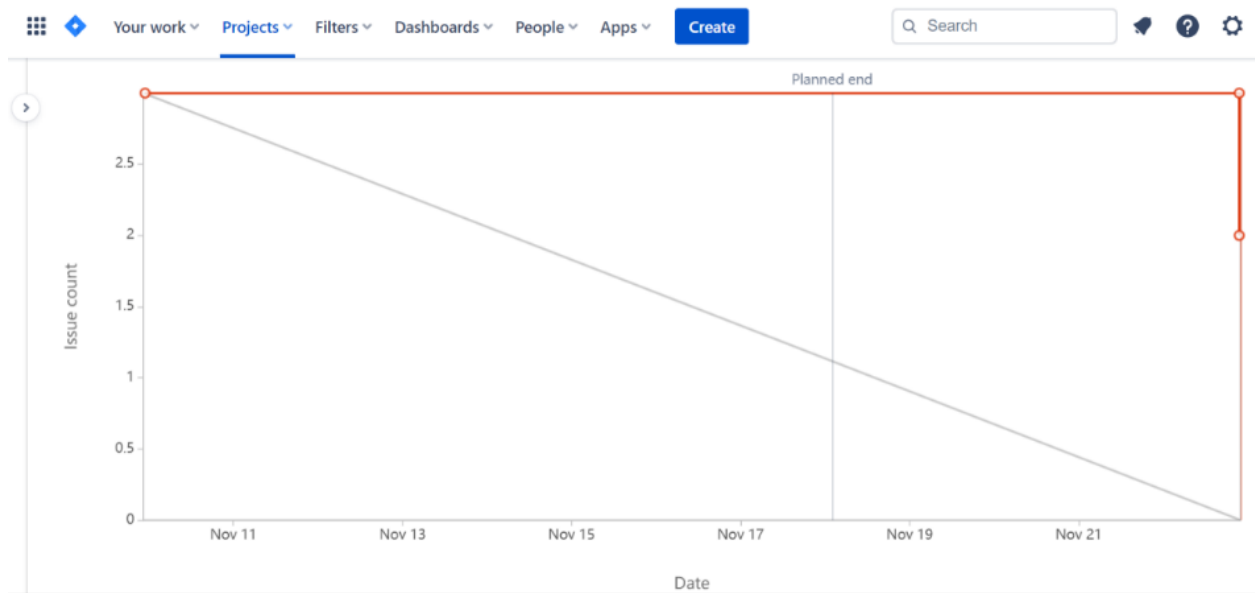




## Sprint 3 :



## Sprint 4 :



## 8 . Project Outcomes

### Source code for Login module :

- **HTML**

```
{% extends 'rishabs/layout1.html' %}

{% block content %}

<div>

    <h1>RISHAB'S APP</h1>

</div>

<div class="forms">

    <form action="{% url 'login_view' %}" method="POST">

        {% csrf_token %}

        <div>

            <input type="text" class="name1" placeholder="Enter
your number " name="username">

                                <input type="password" class="name1"
placeholder="Enter your password " name="password">

        </div>

        {% if message %}

        <div>

            <p class="error">{{ message }}</p>

        </div>

        {% endif %}

    </form>

</div>
```

```

        <div>

                                <button    class="button1"
type="submit">LOGIN</button>

        </div>

        <div class="acc">

                                <p>Don't have an account?<a href="{% url
'register_view' %}"> Register here</a></p>

        </div>

    </form>

</div>

{% endblock %}

{% extends 'rishabs/layout1.html' %}

{% block content %}

    <div>

        <h1>RISHAB'S APP</h1>

        <h3>Please register to continue </h2>

    </div>

    <div class="forms">

        <form action="{% url 'register_view' %}" method="POST">

            {% csrf_token %}

```

```

        <input type="text" class="name1" placeholder="Enter
your name " name="username" required>

        <input type="password" class="name1"
placeholder="Enter your password " name="password" required>

        <input type="password" class="name1"
placeholder="Re-enter your password " name="cpassword" required>

        <input type="tel" class="name1" placeholder="Enter
your number" name="phone_number" required>

        <input type="email" class="name1" placeholder="Enter
your mail" name="email" required>

        {% if message %}

            <p class="error">{{ message }}</p>

        {% endif %}

        <button class="button1" type="submit">NEXT</button>

    </form>

</div>

{% endblock %}

{% extends 'rishabs/layout1.html' %}

{% block content %}

    <div>

        <h1>RISHAB'S APP</h1>

        <h3>OTP Verification</h2>

    </div>

```

```

<div class="forms">

    <form action="{% url 'verify' %}" method="POST">

        {% csrf_token %}

        <input type="tel" class="name1" placeholder="Enter
your number" name="phone_number">

            <input type="number" class="name1"
placeholder="Enter your OTP" name="otp1">

        {% if message %}

            <p class="error">{{ message }}</p>

        {% endif %}

        <button class="button1">NEXT</button>

    </form>

</div>

{% endblock %}

```

## ● BACKEND

```

def login_view(request):

    if request.method == "POST":

        username = request.POST["username"]

        password = request.POST["password"]

        user = authenticate(request, username=username,
password=password)

```

```

if user is not None:

    if User.objects.get(phone_number=username).is_staff:

        orders = OrderModel.objects.all()

        total_revenue = 0

        for order in orders:

            total_revenue += order.price

        context = {

            'orders':orders,

            'total_revenue':total_revenue,

            'total_orders':len(orders),

        }

        login(request, user)

        return render(request,
'restaurent/dashboard.html',context)

    elif
User.objects.get(phone_number=username).is_phone_verified:

        login(request, user)

        return HttpResponseRedirect(reverse("index"))

    else:

        return render(request, 'rishabs/login.html',
{'message': 'Invalid Credentials'})

        return render(request, 'rishabs/login.html')

```

```

def register_view(request):

    if request.method == "POST":

        name = request.POST["username"]

        password = request.POST["password"]

        password2 = request.POST["cpassword"]

        phone_number = request.POST["phone_number"]

        email = request.POST["email"]

        if password == password2 and password!=None:

            user = User.objects.create_user(

                username = name,

                password = password,

                phone_number = phone_number,

                otp = send_otp_to_phone(phone_number),

                email = email

            )

            user.set_password = password

            user.save()

            return render(request, 'rishabs/verify.html')

        else:

            return render(request, 'rishabs/register.html',
{'message': 'Passwords do not match'})

```

```

        return render(request, 'rishabs/register.html')

def logout_view(request):

    logout(request)

    return HttpResponseRedirect(reverse("login_view"))

@api_view(['POST'])

def verify(request):

    if request.method == "POST":

        phone_number = request.POST["phone_number"]

        otp1 = request.POST["otp1"]

        try:

            user_obj = User.objects.get(phone_number =
phone_number)

            except Exception as e:

                return render(request, 'rishabs/verify.html',
{'message': 'Phone Number does not Exists'})

            if user_obj.otp == otp1:

                user_obj.is_phone_verified = True

                user_obj.save()

                return render(request, 'rishabs/login.html')

            return render(request, 'rishabs/verify.html',
{'message': 'Phone Number does not Exists'})

```




```
else:
```

```
    return render(request, 'rishabs/verify.html')
```


## Result :

[HOME](#) [LOGIN](#) [REGISTER](#)



### RISHAB'S APP

Don't have an account? [Register here](#)



### RISHAB'S APP

Please register to continue

## Source code for catalogue module :

- **HTML**

```
{% extends 'rishabs/layout2.html' %}

{% block content %}

    <div>

        <form action="{% url 'order' %}" method="POST">

            {% csrf_token %}

            <div class="main-body">

                <div class="header">

                    <div class="menu">

                        <h1>Menu</h1>

                    </div>

                <div class="name">

                    <h3>{{ user.username }}</h3>

                </div>

            </div>

            {% if message %}

            <div>

                <h3>{{ message }}</h3>

            </div>

        </div>

    </div>

{% endblock %}
```

```

</div>

{% else %}

{% endif %}

{% if error %}

<div class="error-div">

    <p class="div-error">{{ error }}</p>

</div>

{% endif %}

{% if fed %}

    <div>

        <h3>{{ fed }}</h1>

    </div>

{% else %}

{% endif %}

<div class="design-bg">

    <div class="options">

        <a class="option_a" href="#juices">

            <div class="option">

                <p>Juices</p>

            </div>


```

```

        </a>

        <a class="option_a" href="#hot">
            <div class="option">
                <p>Hot Drinks</p>
            </div>
        </a>

        <a class="option_a" href="#snack">
            <div class="option">
                <p>Snacks</p>
            </div>
        </a>
    </div>
</div>

<div class="head">
    <div class="head-name"
id="juices">Juices</div>

    {% for jui in juice %}

    <div class="items">

        <div class="i-name">

            <p >{{jui.name}}</p>

```

```

        </div>

        <div class="i-price">

            <p >#x20b9; {{jui.price}}</p>

        </div>

        <div class="i-check">

            <input    type="checkbox"
name="items[]" value="{{ jui.pk }}">

        </div>

    </div>

{% endfor %}

    <div class="head-name" id="hot">Hot
Drinks</div>

{% for hot in hot_drinks %}

<div class="items">

    <div class="i-name">

        <p >{{hot.name}}</p>

    </div>

    <div class="i-price">

        <p >#x20b9; {{hot.price}}</p>

    </div>

    <div class="i-check">

        <input    type="checkbox"

```

```

name="items[]" value="{{ hot.pk }}">

        </div>

    </div>

    {% endfor %}

<div class="head-name" id="snack">Snacks</div>

    {% for snack in snacks %}

    <div class="items">

        <div class="i-name">

            <p >{{snack.name}}</p>

        </div>

        <div class="i-price">

            <p >&#x20b9; {{snack.price}}</p>

        </div>

        <div class="i-check">

            <input type="checkbox"

name="items[]" value="{{ snack.pk }}">

        </div>

    </div>

    {% endfor %}

</div>

<div>

    <button class="button2" type="submit">PLACE

```

```
ORDER!</button>
```

```
</div>
```

```
</form>
```

```
</div>
```

```
{% endblock %}
```

## ● BACK END

```
def order(request):
```

```
    if request.method == "POST":
```

```
        order_items = {
```

```
            'items': []
```

```
        }
```

```
        items = request.POST.getlist('items[]')
```

```
        if items == []:
```

```
            juice = MenuItem.objects.filter(category__name__contains='juice')
```

```
            hot_drinks = MenuItem.objects.filter(category__name__contains='hot_drinks')
        )
```

```
            snacks = MenuItem.objects.filter(category__name__contains='snacks')
```

```
        # pass into context
```

```
        context = {
```

```

        'juice': juice,
        'hot_drinks': hot_drinks,
        'snacks': snacks,
        'error': "Please select atleast one item!",
    }

    # render the template

    return render(request, 'rishabs/order.html',
context)

else:

    for item in items:

        #print(item)

        menu_item = get_object_or_None(MenuItem,
id=int(item))

        #print(menu_item)

                                                #menu_item    =
MenuItem.objects.get(pk__contains=int(item))

        item_data = {

            'id': menu_item.pk,

            'name': menu_item.name,

            'price': menu_item.price,

        }

```



```

        order_items['items'].append(item_data)

price = 0

item_ids = []

for item in order_items['items']:
    price += item['price']
    item_ids.append(item['id'])

order =
OrderModel.objects.create(price=price,order_user=request.user
)

#order.order_user.add(request.user)

order.items.add(*item_ids)

# context = {

#     'items': order_items['items'],

#     'price': price,

#     'username' : request.user.username

# }

context = {

```

```

        'order':order
    }

    return render(request,
'rishabs/customer_order_details.html', context)

if request.method == "GET":

        juice =
MenuItem.objects.filter(category__name__contains='juice')

        hot_drinks =
MenuItem.objects.filter(category__name__contains='hot_drinks'
)

        snacks =
MenuItem.objects.filter(category__name__contains='snacks')

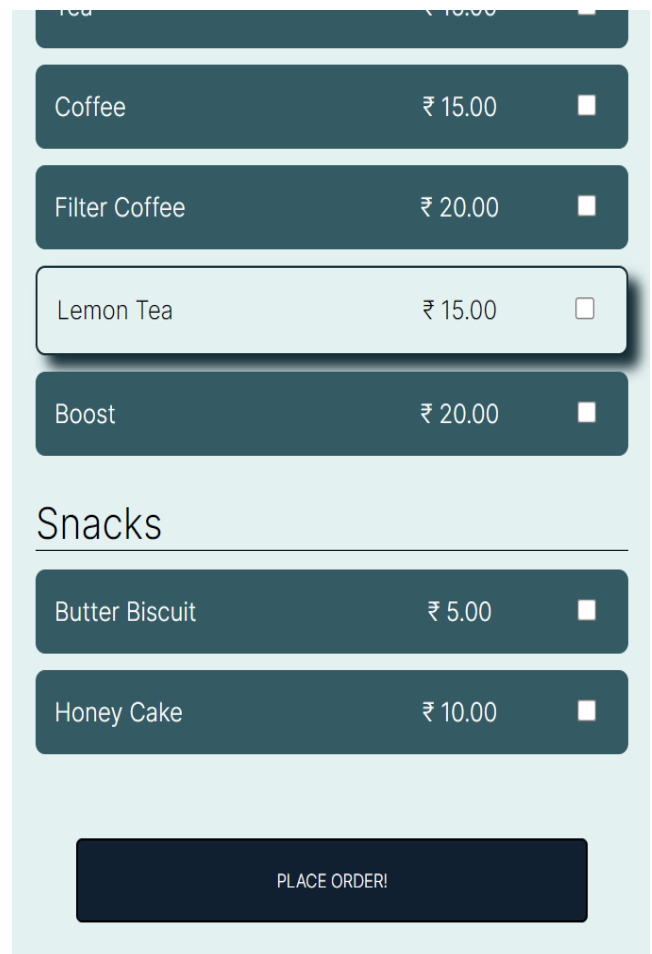
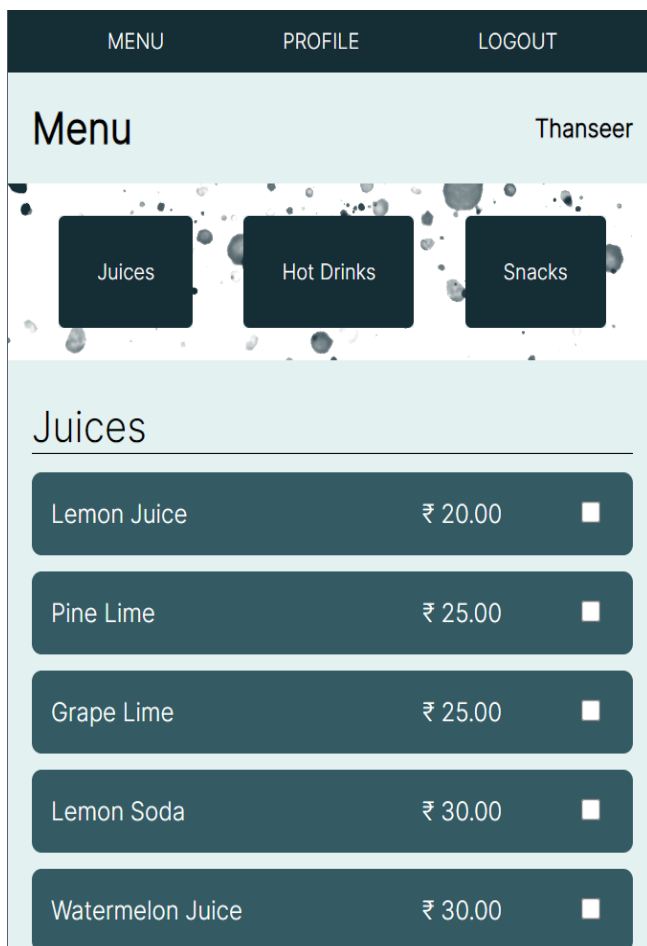
    # pass into context
    context = {
        'juice': juice,
        'hot_drinks': hot_drinks,
        'snacks': snacks,
    }

```

```
# render the template  
return render(request, 'rishabs/order.html', context)
```

```
def profile_view(request):  
    return render(request, 'rishabs/profile.html')
```

### Result :





## USER DETAILS

Username: Thanseer

Mail: thnsr30@gmail.com

Phone No: 7305455402

## Source code for order module :

- **HTML**

```
{% extends 'rishabs/layout2.html' %}

{% block content %}

    <div>

        <br>

        <h1>Order Submitted!</h1>

        <p>You should receive the invoice email soon.</p>

    </div>

    <div class="order-contain">

        <div>

            <div>

                <h3>Order Summary:</h3>

            </div>

            <table class="styled-table">

                <thead>

                    <th>Item</th>

                    <th>Price</th>

                </thead>

                <tbody>
```

```

        {% for item in items %}

        <tr>

            <td>{{ item.name }}</td>

            <td>&#x20b9; {{ item.price }}</td>

        </tr>

        {% endfor %}

        <tr id="order-total">

            <th>Total</th>

            <th>&#x20b9; {{ price }}</th>

        </tr>

    </tbody>

</table>

</div>

</div>

{% endblock %}

```

## - **BACK END**

```

#@login_required(login_url='login_view')

def order_details(request,pk):

    if request.method == "POST":

```

```

order = OrderModel.objects.get(pk=pk)

if "del" in request.POST:
    order.is_delivered = True
    order.save()

elif "pay" in request.POST:
    order.is_paid = True
    order.save()

    context = {
        "username":order.order_user.username,
        "price":order.price,
        "items":order.items,
    }

    template =
render_to_string('restaurent/email_template.html',context)

email = EmailMessage(
    'Thanks for purchasing at Rishabs!',
    template,
    settings.EMAIL_HOST_USER,

```

```

        [order.order_user.email],
    )

    email.fail_silently=False

    email.send()

elif "can" in request.POST:

    order.delete()

    orders = OrderModel.objects.all()

    total_revenue = 0

    for order in orders:

        total_revenue += order.price

    context = {

        'orders':orders,

        'total_revenue':total_revenue,

        'total_orders':len(orders),

    }

    return render(request,
'restaurent/dashboard.html',context)

context = {

```



```

        'order':order
    }

    return render(request,
'restaurent/order_details.html',context)

else:

    check = OrderModel.objects.filter(pk=pk)

    if len(check)>0:

        order = OrderModel.objects.get(pk=pk)

    else:

        return dashboard_view(request)

str = ""

for i in range(2,len(order.feedback)-2):

    str+=order.feedback[i]

context = {

    'order':order,

    'feedback': str

```

```

    }

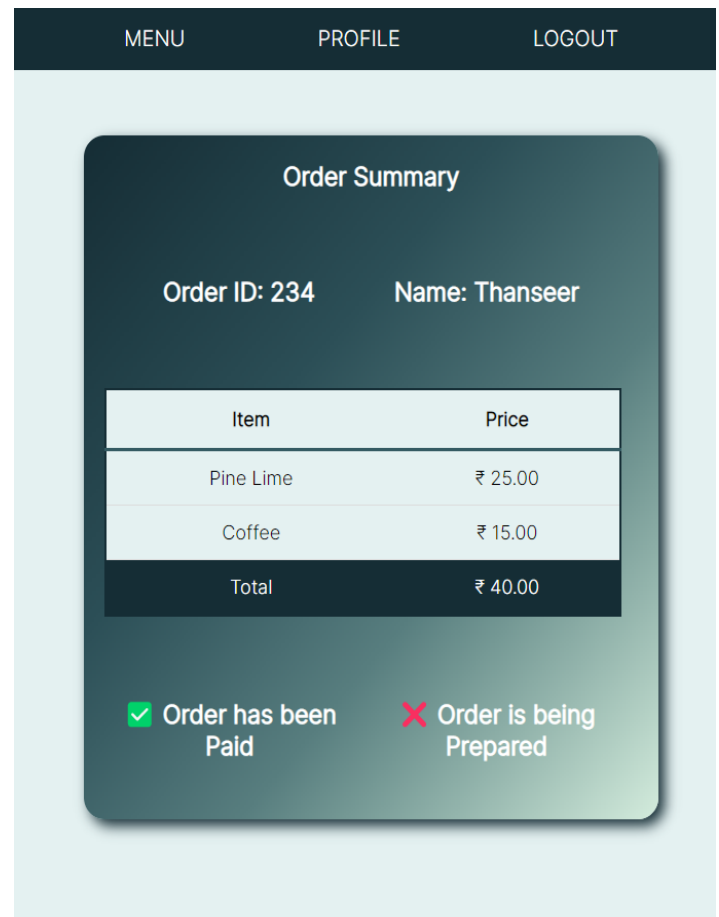
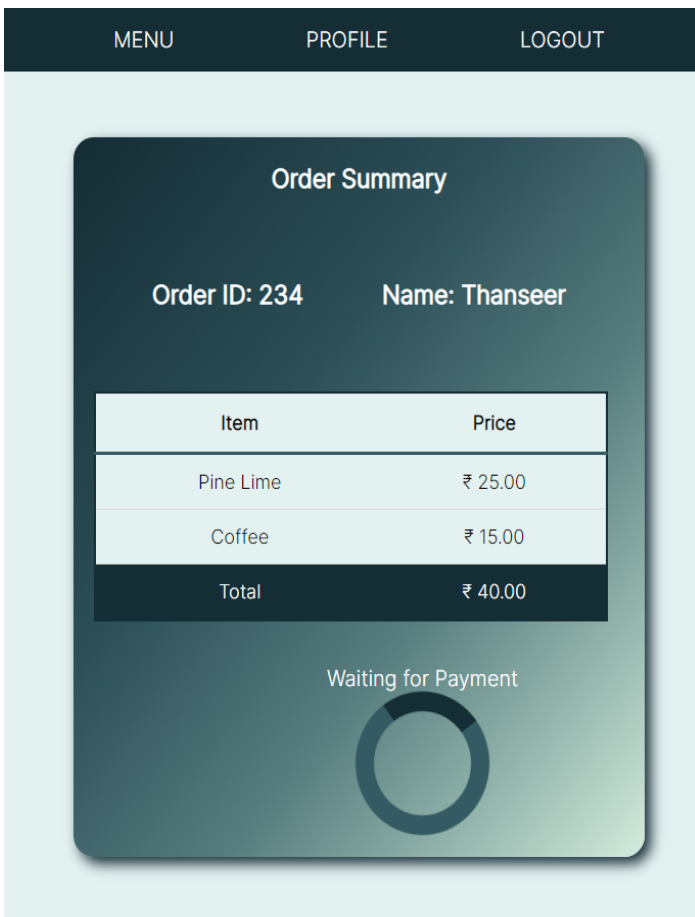
    return render(request,
'restaurent/order_details.html',context)

def logout_view(request):
    logout(request)

    return HttpResponseRedirect(reverse("login_view"))

```

## Result



Thankyou!

You should receive the invoice email soon.

### Order Summary

Order ID: 234

Name: Thanseer

| Item      | Price   |
|-----------|---------|
| Pine Lime | ₹ 25.00 |
| Coffee    | ₹ 15.00 |
| Total     | ₹ 40.00 |

✓ Order has been  
Paid

✓ Order has been  
Delivered

FeedBack

## Source code for dashboard module :

- **HTML**

```
{% extends 'restaurent/layout3.html' %}

{% block content %}

<div class="overview">

    <div class="overview_l">

        <p>Todays Revenue</p>

        <p>&#x20b9; {{ total_revenue }}</p>

    </div>

    <div class="overview_r">

        <p>Todays Orders</p>

        <p>{{ total_orders }}</p>

    </div>

</div>


<div class="table-box">

    <div>

        <h1>Pending Orders</h1>

    </div>

</div>
```

```

<div class="box">

    <table class="styled-table">

        <thead>

            <tr>

                <th>Order ID</th>

                <th>Name</th>

                <th>Price</th>

                <th>Paid?</th>

                <th>Delivered?</th>

                <th>Order Details</th>

            </tr>

        </thead>

        <tbody>

            {% for order in orders %}

                {% if order.is_delivered %}

                {% else %}

                    <tr>

                        <td>{{ order.pk }}</td>

                        <td>{{ order.order_user.username

                    }}</td>

```

```

}}</td>

<td>&#x20b9; {{ order.price

{% if order.is_paid %}

    <td><h3>&#x2705;</h3></td>

{% else %}

    <td><h3>&#10060;</h3></td>

{% endif %}

{% if order.is_delivered %}

    <td><h3>&#x2705;</h3></td>

{% else %}

    <td><h3>&#10060;</h3></td>

{% endif %}

<td class>

    <h2><a class="order-s"
href="{% url 'order_details' order.pk %}">&#9998;</a></h2>

</td>

</tr>

{% endif %}

{% endfor %}

```

```

        </tbody>

    </table>

</div>

</div>

<div class="table-box">

    <div>

        <h1>Completed Orders</h1>

    </div>

    <div class="box">

        <table class="styled-table">

            <thead>

                <tr>

                    <th>Order ID</th>

                    <th>Name</th>

                    <th>Price</th>

                    <th>Paid?</th>

                    <th>Delivered?</th>

                    <th>Order Details</th>

                </tr>

```

```

</thead>

<tbody>

    {% for order in orders %}

        {% if order.is_delivered %}

            <tr>

                <td>{{ order.pk }}</td>

                <td>{{ order.order_user.username
}}</td>

                <td>&#x20b9; {{ order.price
}}</td>

                {% if order.is_paid %}

                    <td><h3>&#x2705;</h3></td>

                {% else %}

                    <td><h3>&#10060;</h3></td>

                {% endif %}

                {% if order.is_delivered %}

                    <td><h3>&#x2705;</h3></td>

                {% else %}

                    <td><h3>&#10060;</h3></td>

                {% endif %}

```



```

        <td class>

            <h2><a class="order-s"
href="{% url 'order_details' order.pk %}">&#9998;</a></h2>

        </td>

    </tr>

    {% else %}

    {% endif %}

{% endfor %}

</tbody>

</table>

</div>

</div>

<script>

    function autoRefresh() {

        window.location = window.location.href;

    }

    setInterval('autoRefresh()', 5000);

</script>

{% endblock %}

```

- **BACK END**

```
from django.shortcuts import render

from django.contrib.auth.decorators import login_required

from rishabs.models import OrderModel

from django.contrib.auth import logout

from django.urls import reverse

from django.http import HttpResponseRedirect

from django.conf import settings

from django.template.loader import render_to_string

from django.core.mail import EmailMessage

# Create your views here.


@login_required(login_url='login_view')

def profile_view_res(request):

    return render(request, 'restaurent/profile.html')


@login_required(login_url='login_view')

def dashboard_view(request):

    orders = OrderModel.objects.all()

    total_revenue = 0
```

```

for order in orders:

    total_revenue += order.price


context = {

    'orders':orders,

    'total_revenue':total_revenue,

    'total_orders':len(orders),

}

return render(request,
'restaurent/dashboard.html',context)


#@login_required(login_url='login_view')
def order_details(request,pk):

    if request.method == "POST":

        order = OrderModel.objects.get(pk=pk)

        if "del" in request.POST:

            order.is_delivered = True

            order.save()

        elif "pay" in request.POST:

            order.is_paid = True

```

```

order.save()

context = {

    "username":order.order_user.username,

    "price":order.price,

    "items":order.items,

}

template =
render_to_string('restaurent/email_template.html',context)

email = EmailMessage(

    'Thanks for purchasing at Rishabs!',

    template,

    settings.EMAIL_HOST_USER,

    [order.order_user.email],

)

email.fail_silently=False

email.send()

elif "can" in request.POST:

    order.delete()

```

```

        orders = OrderModel.objects.all()

        total_revenue = 0

        for order in orders:

            total_revenue += order.price

        context = {

            'orders':orders,

            'total_revenue':total_revenue,

            'total_orders':len(orders),

        }

        return render(request,
'restaurent/dashboard.html',context)

    context = {

        'order':order

    }

    return render(request,
'restaurent/order_details.html',context)

else:

    check = OrderModel.objects.filter(pk=pk)

```

```

    if len(check)>0:

        order = OrderModel.objects.get(pk=pk)

    else:

        return dashboard_view(request)

str = ""

for i in range(2,len(order.feedback)-2):

    str+=order.feedback[i]

context = {

    'order':order,

    'feedback': str

}

return render(request,
'restaurent/order_details.html',context)

def logout_view(request):

    logout(request)

    return HttpResponseRedirect(reverse("login_view"))

```

Result

DASHBOARD

PROFILE

LOGOUT

Today's Revenue

₹ 525.00

Today's Orders

10

Pending Orders

| Order ID | Name     | Price   | Paid? | Delivered? | Order Details |
|----------|----------|---------|-------|------------|---------------|
| 233      | Thanseer | ₹ 20.00 | ✗     | ✗          |               |

Completed Orders

| Order ID | Name | Price | Paid? | Delivered? | Order Details |
|----------|------|-------|-------|------------|---------------|
|----------|------|-------|-------|------------|---------------|

Pending Orders

| Order ID | Name     | Price   | Paid? | Delivered? | Order Details |
|----------|----------|---------|-------|------------|---------------|
| 233      | Thanseer | ₹ 20.00 | ✗     | ✗          |               |

Completed Orders

| Order ID | Name     | Price   | Paid? | Delivered? | Order Details |
|----------|----------|---------|-------|------------|---------------|
| 223      | Thanseer | ₹ 90.00 | ✓     | ✓          |               |
| 224      | Thanseer | ₹ 25.00 | ✓     | ✓          |               |
| 226      | Thanseer | ₹ 20.00 | ✓     | ✓          |               |

DASHBOARD

PROFILE

LOGOUT

Order ID: 233

Name: Thanseer

| Item        | Price   |
|-------------|---------|
| Lemon Juice | ₹ 20.00 |
| Total       | ₹ 20.00 |

Cancel Order

Mark as Paid

DASHBOARD

PROFILE

LOGOUT

Order ID: 234

Name: Thanseer

| Item      | Price   |
|-----------|---------|
| Pine Lime | ₹ 25.00 |
| Coffee    | ₹ 15.00 |
| Total     | ₹ 40.00 |

✓ Order has been Paid

✓ Order has been Delivered

Order ID: 235

Name: Thanseer

| Item        | Price   |
|-------------|---------|
| Lemon Juice | ₹ 20.00 |
| Honey Cake  | ₹ 10.00 |
| Total       | ₹ 30.00 |

✓ Order has been  
Paid

✓ Order has been  
Delivered

FEEDBACK:

Thank you for the order!



## 9. Project Outcomes:

The final project is an online ordering website that is capable of displaying the catalogue and take order from customer and generate the bill for the ordered list of item . This website also provides the functionalities of process the past order ,current order and confirm the delivered order by the admin as well as the admin can view the feedback form that was filled by the customer about the service.

## 10.Conclusion and Future Directions

Challenges faced were:

- Consistently styling the webpage using HTML/CSS (correct positioning of elements and divisions) and designing in a user friendly manner.
- The email SMTP was blocked for our mail in particular due to security reasons this was solved by creating a new mail account.
- Redirecting to the correct website without error
- Creating links to Django database and integrating queries into Django to make sure create, delete, update, and read operations are performed successfully.
- Linking all pages in a correct and accessible way.
- In future we plan to add quantity variable that ensures the availability of the item.
- We also plan to add a view to the staff to be able to add new items and manage stocks with a nice UI

## References

- <https://www.geeksforgeeks.org/>
- <https://docs.djangoproject.com/>
- <https://www.digitalocean.com/community>

## CLIENT EVALUATION REPORT

**Name of the project:**

**Team Members:**

**Client details:**

**Rating System - 1: Strongly disagree 2: Disagree 3: Neutral 4: Agree 5: Strongly Agree**

| Questions  | 1 | 2 | 3 | 4 | 5 |
|--|---|---|---|---|---|
| The problem was well discussed and, the requirements and goals were clear.                         |   |   |   |   |   |
| The project plan was well defined and communicated from the start.                                 |   |   |   |   |   |
| The resources were adequate for achieving the goals.   |   |   |   |   |   |
| The original timeline was realistic and was followed.  |   |   |   |   |   |
| The teamwork was well demonstrated.  |   |   |   |   |   |
| The client was communicated on regular intervals and given updates on the progress of the project. |   |   |   |   |   |
| The expected project requirements have been satisfied.   |   |   |   |   |   |