YUMHUB:

A Food Stall Application

Web Development Framework Using Python (23AI002)

Submitted by

Ayush Bansal (2210993778)

Manya Saini (2210993814)

Semester: 4th

BE-CSE (Artificial Intelligence)

Submitted To

Rajan Kumar

Assistant Professor, Department of CSE(AI),

CUIET, Chitkara University

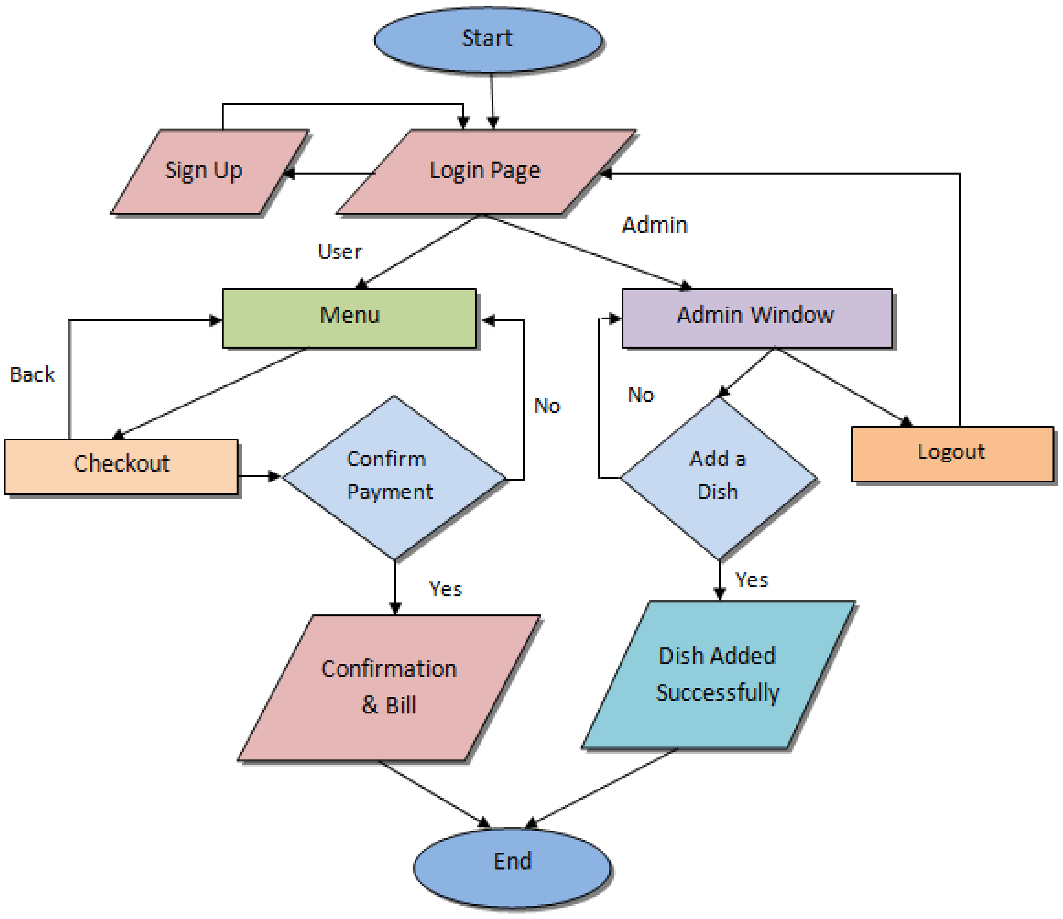
CHITKARA UNIVERSITY INSITUTE OF ENGINEERING & TECHNOLOGY

CHITKARA UNIVERSITY, RAJPURA

March, 2024

4. Flowchart

YUMHUB follows the beneath flow for seamless food ordering for customers and proper administration of the dishes and orders by the admin:



4.1 Flowchart

5. Code (Python File Only)

Django Project is developed using two directories, i.e, Project and App. The provided code will be divided into two sections one for files in app directory named YumHub, and then in the project directory named Project.

YumHub (App directory)

- **urls.py**

```
from django.contrib import admin
from django.urls import path,include
from . import views

urlpatterns = [
    path("",views.index,name='index'),
    path("login/",views.login,name='login'),
    path("signup/",views.signup,name='signup'),
    path("menu/",views.menu,name='menu'),
    path("checkout/",views.checkout,name='checkout'),
    path("payment/",views.payment,name='payment'),
    path("bill/",views.bill,name='bill'),
   path("admin_dashboard/",views.admin_dashboard,name='admin_dashbo
   ard'),
    path("logout/",views.logout_view,name="logout")
]
```

- **views.py**

```
from django.shortcuts import *
from .models import *
from django.http import *
from django.contrib.auth import authenticate, login as auth_login,
logout
import json
import ast
from decimal import Decimal
from .backends import ProfileBackend
from django.contrib.auth.decorators import login_required
import base64
from django.contrib import messages

# Create your views here.
def index(request):
    return render(request,'index.html')
```

```python
def login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user_type = request.POST.get('user_type')

        user = authenticate(request, username=username,
password=password)
        if user is not None:
            profile = Profile.objects.get(user=user)
            if profile.user_type == 'admin':
                auth_login(request, user)
                return redirect('admin_dashboard')
            else:
                auth_login(request, user)
                return redirect('menu')

    return render(request, 'login.html')

def signup(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        phone = request.POST.get('phone')
        username = request.POST.get('username')
        password = request.POST.get('password')
        user_type = request.POST.get('user_type')

        # Create a new User instance
        user = User.objects.create_user(username=username,
password=password)

        # Create a new Profile instance and link it to the user
        profile = Profile.objects.create(name=name, phone=phone,
user_type=user_type, username=username, user=user)

        return redirect('login')

    return render(request, 'signup.html')

@login_required
def menu(request):
    dishes = Dish.objects.all()
    for dish in dishes:
        with open(dish.image.path, "rb") as img_file:
            dish.image_data =
base64.b64encode(img_file.read()).decode('utf-8')
    return render(request, 'menu.html', {'dishes': dishes})

@login_required
def checkout(request):
    if request.method == 'POST':
        total_orders = {}
        total_amount = 0
        dishes = Dish.objects.all()

        for dish in dishes:
            quantity = int(request.POST.get(f'dish_{dish.id}'))
            if quantity > 0:
                total_orders[dish.id] = quantity
                total_amount += quantity * dish.price

        processed_orders = []
        for dish_id, quantity in total_orders.items():
            dish = Dish.objects.get(pk=dish_id)
            processed_orders.append({'name': dish.name, 'price':
dish.price, 'quantity': quantity})

        return render(request, 'checkout.html', {'total_orders':
total_orders, 'total_amount': total_amount, 'processed_orders':
processed_orders})

    return redirect('menu')


@login_required
def payment(request):
    if request.method == 'POST':
        total_amount = request.POST.get('total_amount')
        processed_orders =
request.POST.getlist('processed_orders[]')
        tax = float(total_amount)*0.10
        total=float(total_amount)+tax

        return render(request, 'payment.html', {'total_amount':
total_amount, 'processed_orders': processed_orders,'total':total})

    total_amount=None
    processed_orders=[]
    total = 0
    return render(request, 'payment.html', {'total_amount':
total_amount,'processed_orders':processed_orders,'total':total})


    return redirect('menu')

@login_required
def bill(request):
    if request.method == 'POST':
        total_amount = request.POST.get('total_amount')
        processed_orders =
request.POST.getlist('processed_orders[]')
        tax = float(total_amount)*0.10
        total=float(total_amount)+tax
        # for order in processed_orders:
        #     o_total=order.price*order.quantity
        #     order.append(o_total)
        def preprocess_order_string(order_str):
            return order_str.replace("Decimal(", "").replace(")",
"")

        processed_orders_dicts = []
        for order_str in processed_orders:
            order_str = preprocess_order_string(order_str)
            order_dict = ast.literal_eval(order_str)

order_dict['o_total']=float(order_dict['quantity']*float(order_dict[
'price']))
            order_dict['price'] = Decimal(order_dict['price'])
            processed_orders_dicts.append(order_dict)

        return render(request, 'bill.html', {'total_amount':
total_amount, 'processed_orders':
processed_orders_dicts,'tax':tax,'total':total})

    total_amount=None
    processed_orders=[]
    tax=None
    total = None
    return render(request, 'bill.html', {'total_amount':
total_amount, 'processed_orders':
processed_orders,'tax':tax,'total':total})
```

```
def logout_view(request):
    logout(request)
    return redirect('login')

@login_required
def admin_dashboard(request):
    if request.method == 'POST':
        dish_name = request.POST.get('dish_name')
        dish_description = request.POST.get('dish_description')
        dish_image = request.FILES.get('dish_image')
        dish_price = request.POST.get('dish_price')

        new_dish = Dish(name=dish_name, image=dish_image,
price=dish_price, quantity = 0)
        new_dish.save()

        messages.success(request, 'Dish added successfully!')

        return redirect('admin_dashboard')

    return render(request, 'admin_dashboard.html')
```

- **models.py**

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib.auth.hashers import make_password, check_password
from django.utils import timezone

from django.contrib.auth.models import User
from django.db import models

class Profile(models.Model):
    USER_TYPE_CHOICES = [
        ('user', 'User'),
        ('admin', 'Admin'),
    ]

    name = models.CharField(max_length=100)
    phone = models.CharField(max_length=15)
    username = models.CharField(unique=True, max_length=50)
    user_type = models.CharField(choices=USER_TYPE_CHOICES,
default='user', max_length=10)
    user = models.OneToOneField(User, on_delete=models.CASCADE,
default=None, null=True)
```

```
    def __str__(self):
        return self.username

class Dish(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    image = models.ImageField(upload_to='dishes')
    price = models.DecimalField(max_digits=6, decimal_places=2)
    quantity = models.PositiveIntegerField()

    def __str__(self):
        return self.name


class Order(models.Model):
    id = models.AutoField(primary_key=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE
    status = models.CharField(max_length=50, choices=(('placed',
'Placed'), ('cancelled', 'Cancelled'), ('completed', 'Completed')),
default='placed')
    created_at = models.DateTimeField(auto_now_add=True)


class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)  # Order
this item belongs to
    dish = models.ForeignKey(Dish, on_delete=models.CASCADE)  # Dish in
the order
    quantity = models.PositiveIntegerField()
```

Project (Project directory)

- **urls.py**

```
from django.contrib import admin
from django.urls import path,include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("admin/", admin.site.urls),
    path("",include('YumHub.urls')),

] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

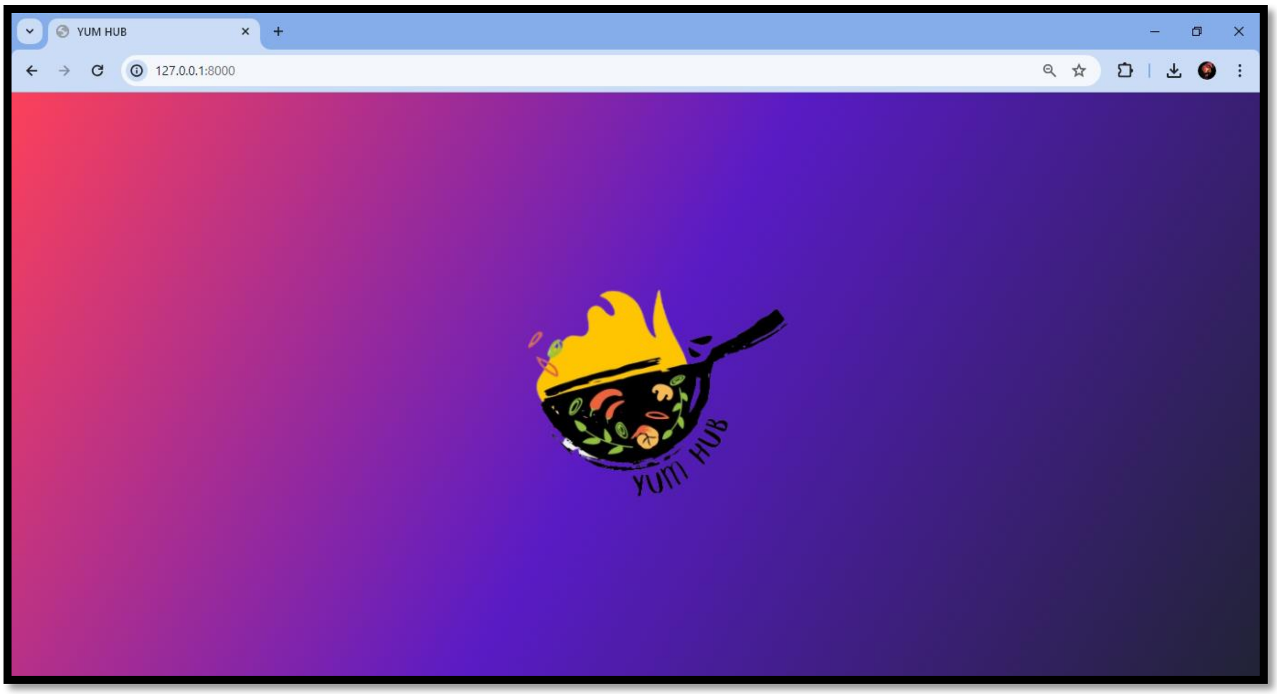
- **settings.py**

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See
https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-
mv1n*i_#v1bu1c49n%gt+9kzoua_%f_pzehej7odm=68y3%3b2"

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['127.0.0.1']


# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "YumHub"
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]
```

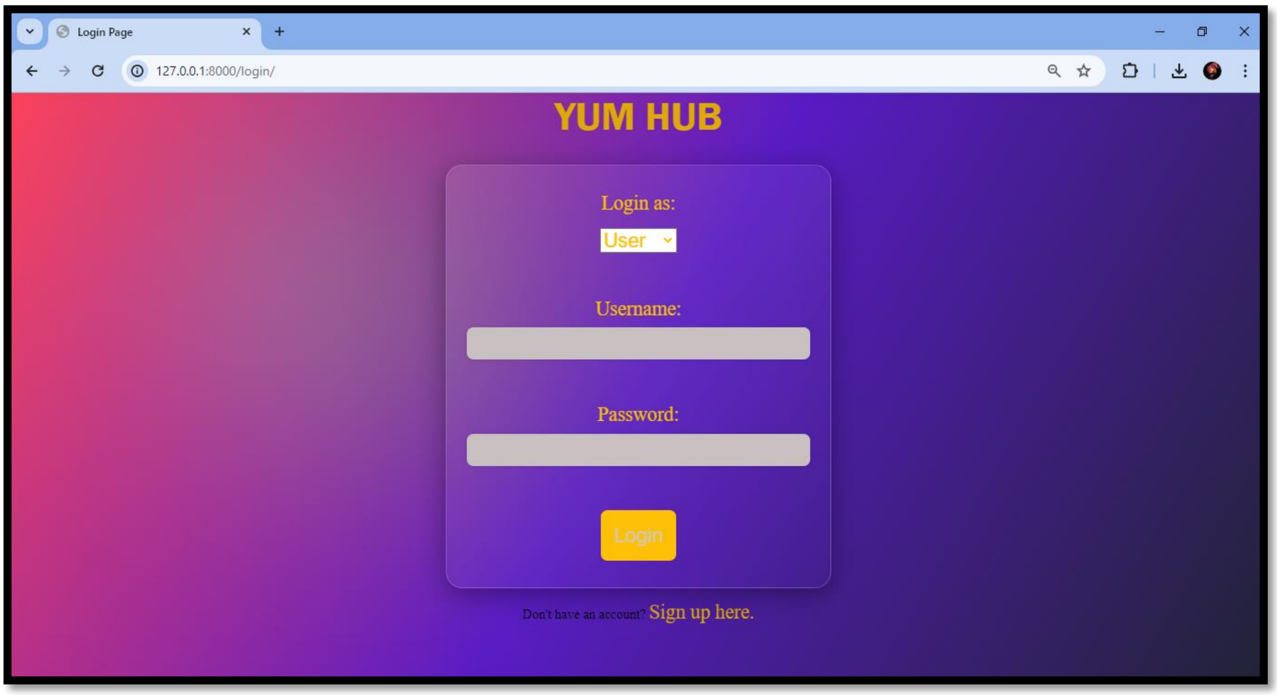
```
ROOT_URLCONF = "Project.urls"

AUTHENTICATION_BACKENDS = [
    # 'YumHub.backends.ProfileBackend',  # Add your custom backend
here
    'django.contrib.auth.backends.ModelBackend',  # Default Django
authentication backend
    # Other authentication backends if needed
]

TEMPLATES = [
    {
        "BACKEND":
"django.template.backends.django.DjangoTemplates",
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",

"django.contrib.messages.context_processors.messages",
                "django.template.context_processors.media",
            ],
        },
    },
]

WSGI_APPLICATION = "Project.wsgi.application"


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}
```

```
# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-
password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME":
"django.contrib.auth.password_validation.UserAttributeSimilarityVa
lidator",
    },
    {"NAME":
"django.contrib.auth.password_validation.MinimumLengthValidator",}
,
    {"NAME":
"django.contrib.auth.password_validation.CommonPasswordValidator",
},
    {"NAME":
"django.contrib.auth.password_validation.NumericPasswordValidator"
,},
]


# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = "static/"

STATICFILES_DIRS=[
    os.path.join(BASE_DIR,"static")
]

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-
auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

6. Major Findings/Outcomes/Output/Results



6.1 Logo Page



YUM HUB

Login as:

User

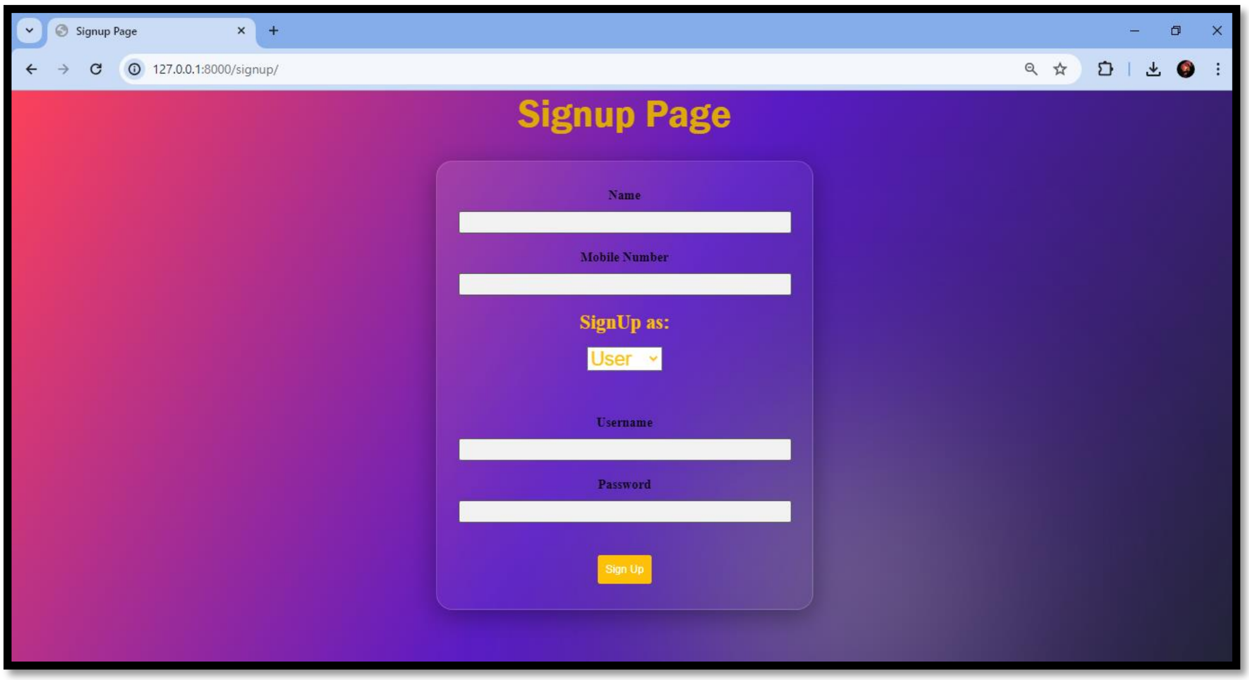
Username:

Password:

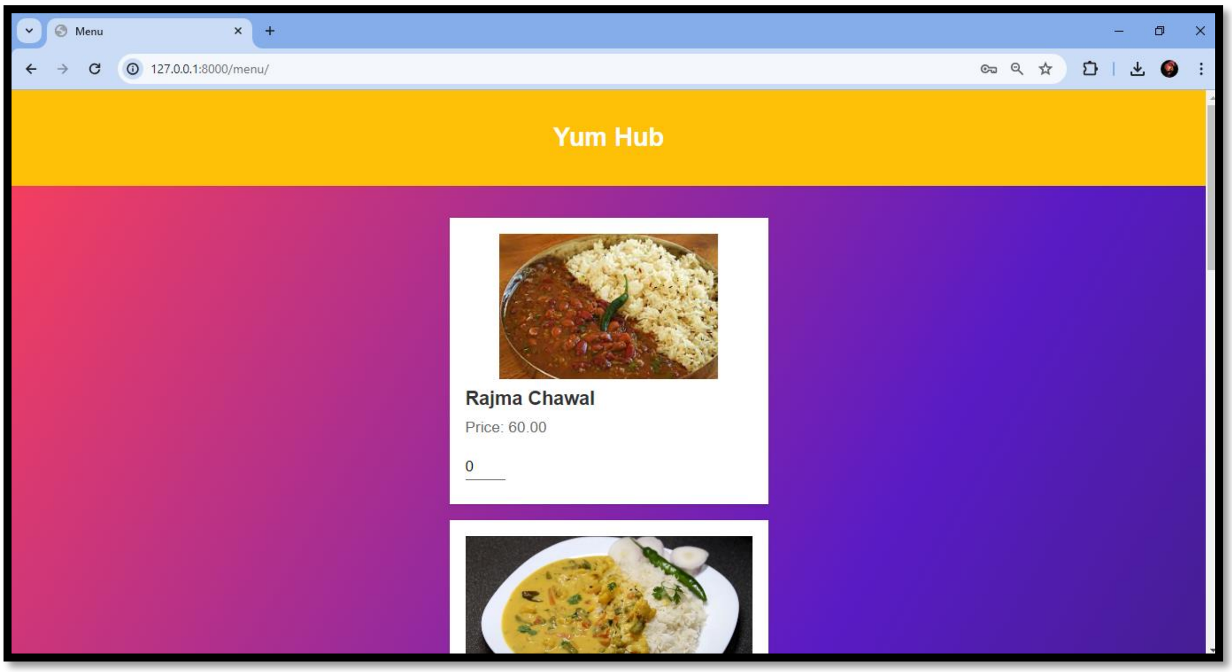
Login

Don't have an account? Sign up here.

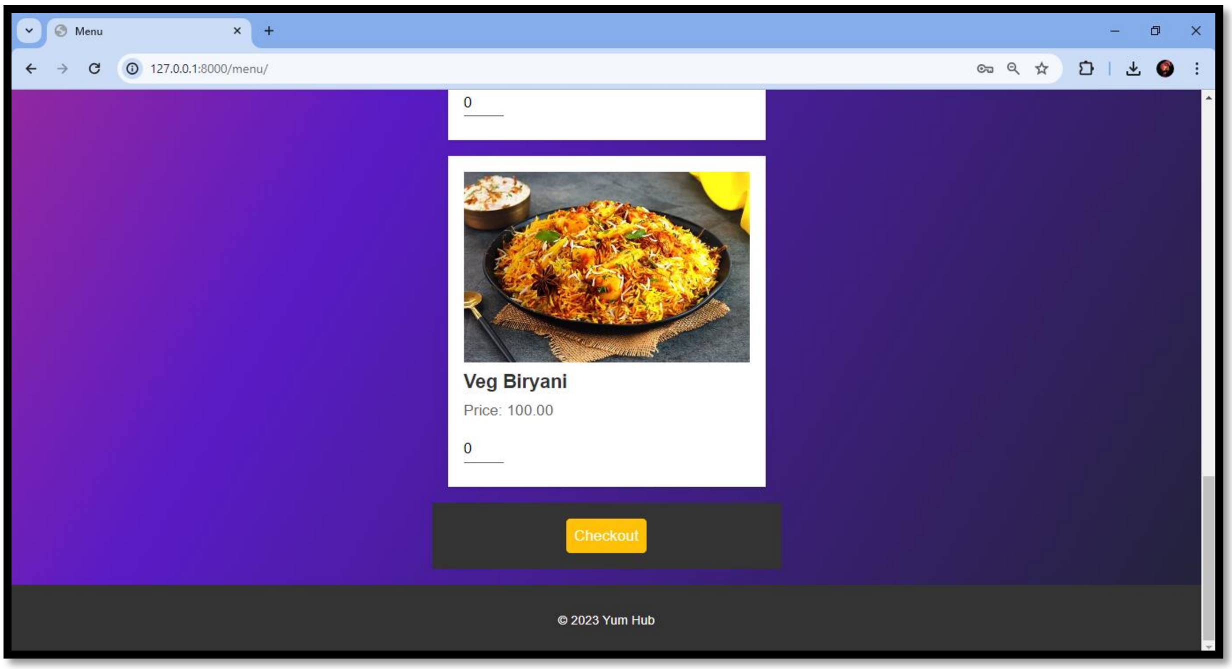
6.2 Login Page



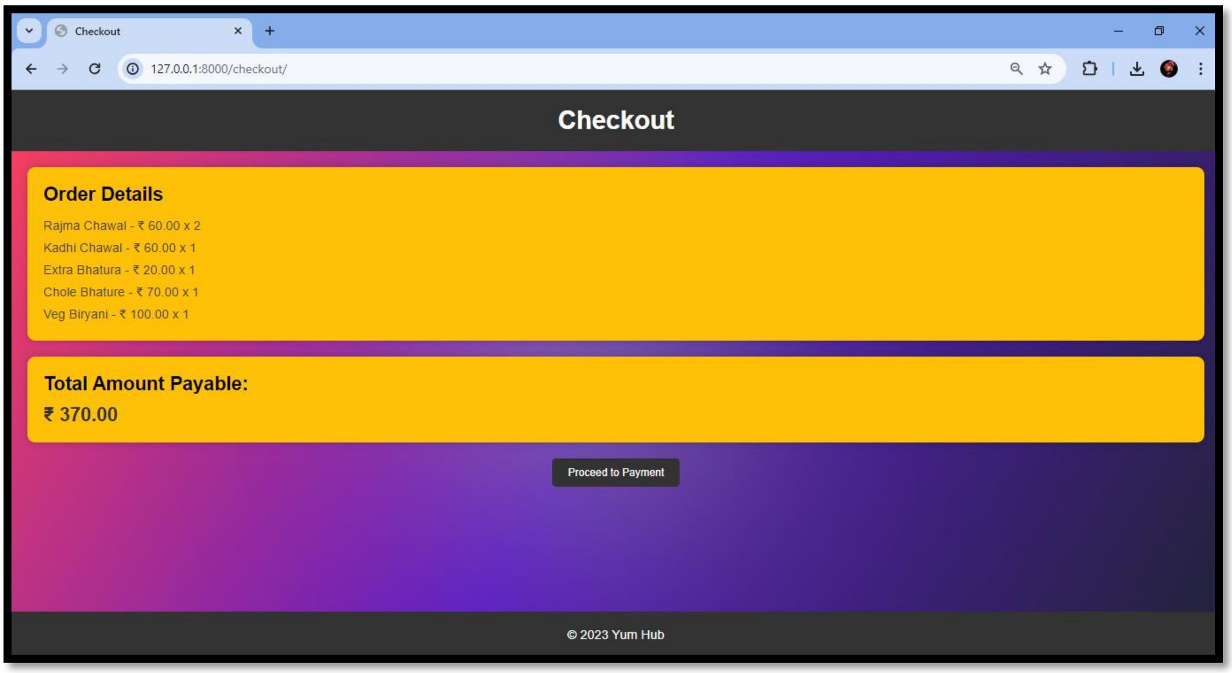
6.3 Signup Page



6.4 Menu Page (1)



6.5 Menu Page (2)

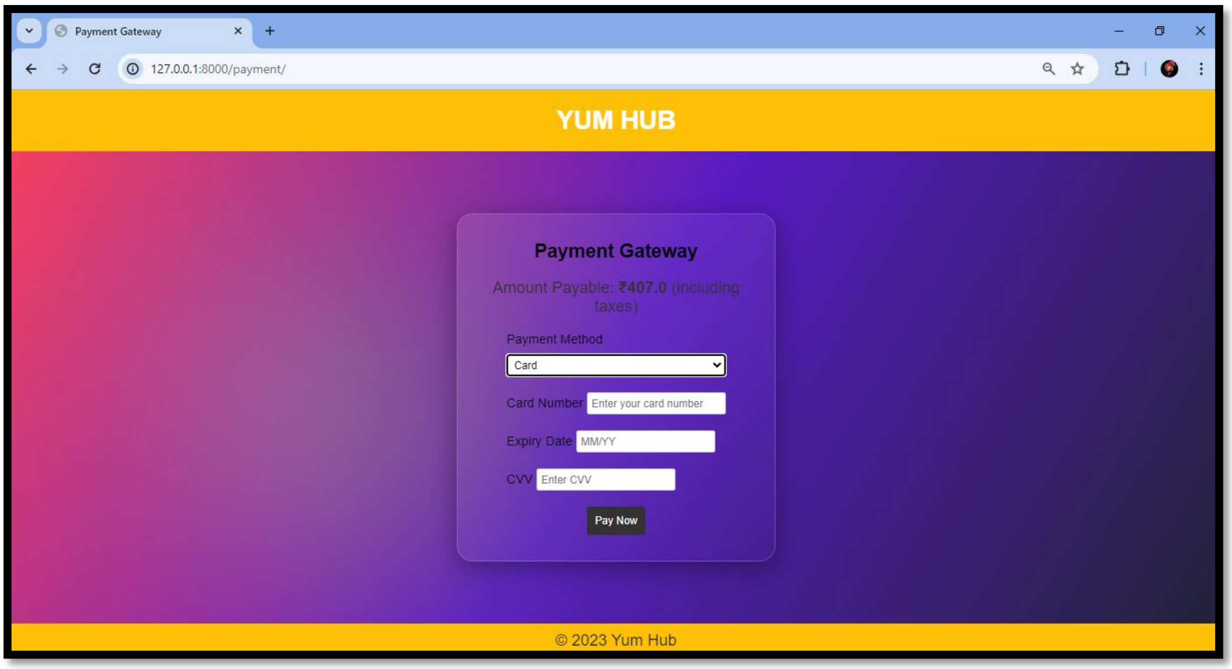


Order Details

Rajma Chawal - ₹ 60.00 x 2
Kadhi Chawal - ₹ 60.00 x 1
Extra Bhatura - ₹ 20.00 x 1
Chole Bhature - ₹ 70.00 x 1
Veg Biryani - ₹ 100.00 x 1

Total Amount Payable:
₹ 370.00

6.6 Checkout Page



Payment Gateway

Amount Payable: **₹407.0** (including taxes)

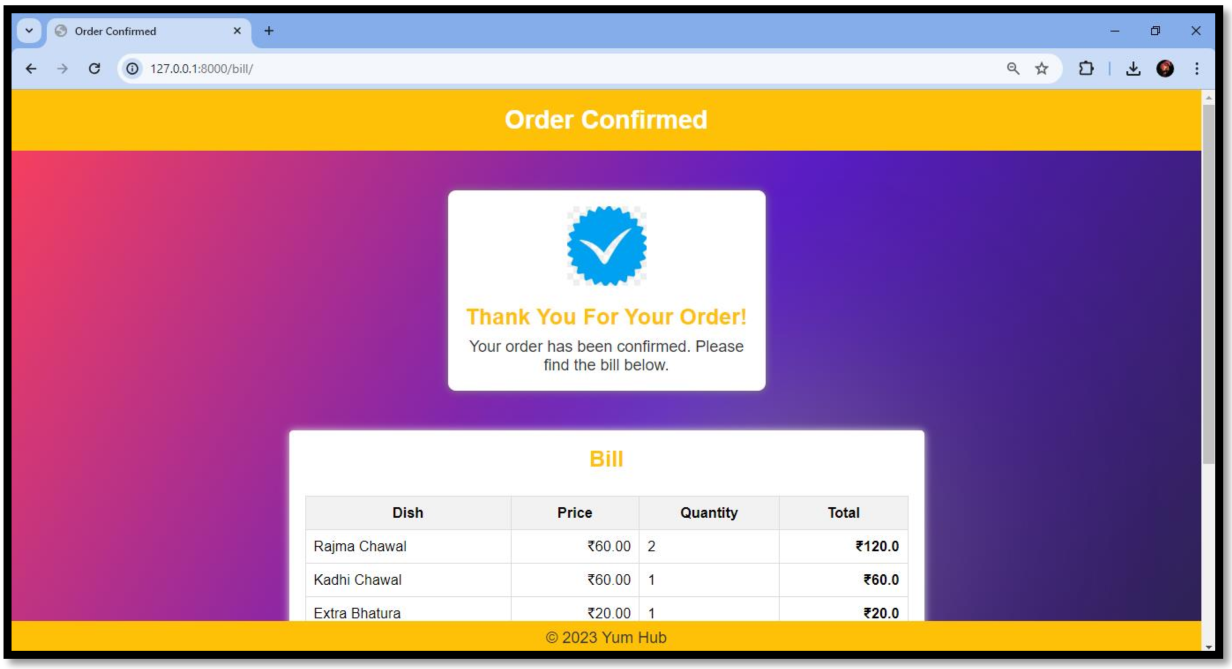
Payment Method: Card

Card Number: Enter your card number

Expiry Date: MM/YY

CVV: Enter CVV

6.7 Payment Window



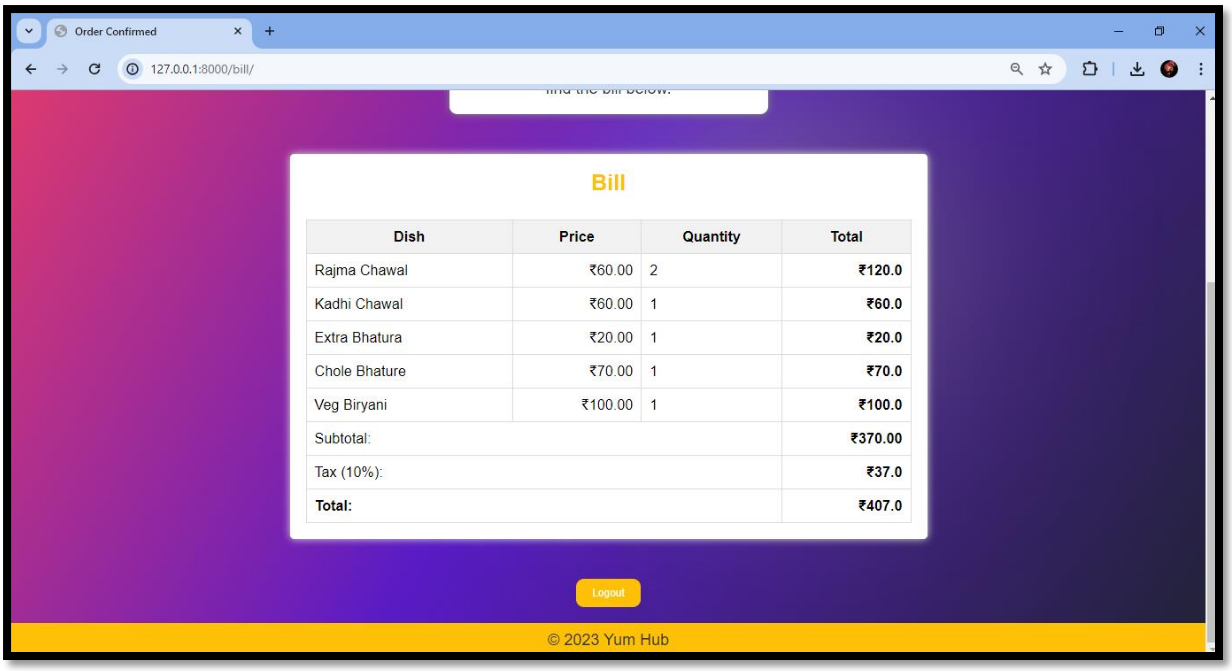
Thank You For Your Order!

Your order has been confirmed. Please find the bill below.

Bill

Dish	Price	Quantity	Total
Rajma Chawal	₹60.00	2	**₹120.0**
Kadhi Chawal	₹60.00	1	**₹60.0**
Extra Bhatura	₹20.00	1	**₹20.0**

6.8 Order Confirmation


Order Confirmed

127.0.0.1:8000/bill/

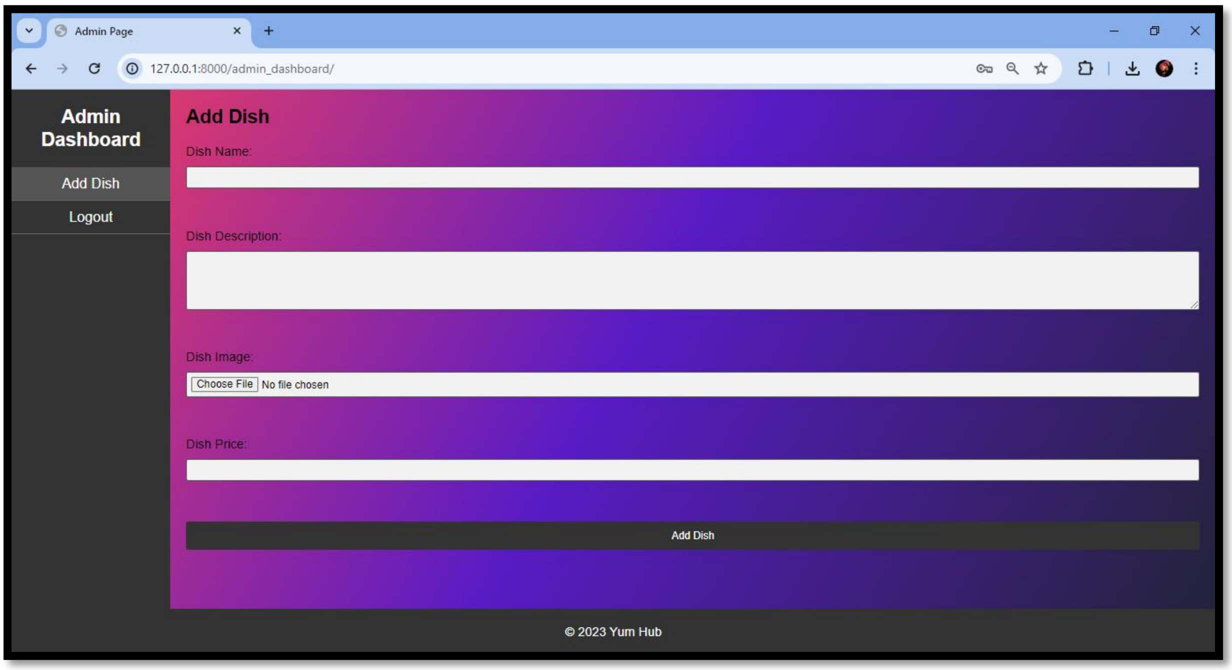
Bill

Dish	Price	Quantity	Total
Rajma Chawal	₹60.00	2	**₹120.0**
Kadhi Chawal	₹60.00	1	**₹60.0**
Extra Bhatura	₹20.00	1	**₹20.0**
Chole Bhature	₹70.00	1	**₹70.0**
Veg Biryani	₹100.00	1	**₹100.0**
Subtotal:			**₹370.00**
Tax (10%):			**₹37.0**
Total:			**₹407.0**

Logout

© 2023 Yum Hub

6.9 Order Bill


Admin Page

127.0.0.1:8000/admin_dashboard/

Admin Dashboard

Add Dish

Logout

Add Dish

Dish Name:

Dish Description:

Dish Image:

Choose File No file chosen

Dish Price:

Add Dish

© 2023 Yum Hub

6.10 Add a Dish (Admin Dashboard)