# **AMRUT AAHAR**

mounika patharlapalli 9-MARCH-2025

## **AmrutAahar**

### **Healthy Groceries E-Commerce Platform**

### 1. Project Overview

#### **Purpose**

AmrutAahar is an e-commerce platform designed to sell healthy groceries. The platform allows users to browse various product categories, apply filters based on price ranges, manage their cart and wishlist, securely complete purchases via Razorpay, and track orders. The platform also emphasizes a user-friendly UI/UX to promote healthy grocery shopping.

### Scope

- User Authentication: Signup, login, and logout functionality.
- **Product Catalog**: Browse groceries by category, search, and filter by price.
- Cart & Wishlist: Add/remove items from the cart and wishlist.
- Checkout & Payments: Secure transactions using Razorpay.
- Order Management: Order confirmation, success page, and tracking.
- Admin Features: Manage products, categories, and user orders.

### 2. Technology Stack

Backend: Django (Python), SQLite

• Frontend: HTML, CSS, JavaScript

• Authentication: Django authentication system

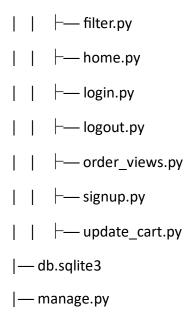
Payment Gateway: Razorpay

Deployment: Gunicorn, Render

### 3. Folder Structure

mystore/

```
| mystore/
| ├— asgi.py
├— settings.py
├— wsgi.py
|— shop/
| | — customer.py
| \ | \ | index.html
├— signup.html
| | — checkout.py
```



### **Explanation**

- mystore/: Main project folder containing settings and configurations.
- **shop/**: Core Django app handling business logic and database models.
  - models/: Defines database structure for categories, customers, products, and orders.
  - o **templates/**: Contains HTML templates for different pages.
  - o **views/**: Manages logic for authentication, product display, cart, checkout, etc.
- **db.sqlite3**: SQLite database storing all platform data.
- manage.py: Django command-line tool.

### 4. Database Models

#### **Category Model**

```
class Category(models.Model):
   name = models.CharField(max_length=255)
   slug = models.SlugField(unique=True)
```

#### **Product Model**

```
class Product(models.Model):
    name = models.CharField(max_length=255)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField()
    description = models.TextField()
    image = models.ImageField(upload_to='products/')
```

#### **Customer Model**

```
class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    address = models.TextField()
    phone = models.CharField(max_length=15)
```

#### **Order Model**

```
class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
    is_paid = models.BooleanField(default=False)
    timestamp = models.DateTimeField(auto_now_add=True)
```

### 5. Key Functionalities

#### **User Authentication**

- **Signup**: Users can register via email and password.
- Login/Logout: Secure authentication using Django's built-in system.

• Profile Management: Users can update their addresses and phone numbers.

#### **Product Browsing & Filtering**

- Products are displayed on the homepage with categories and filtering options.
- Users can filter products by price range and category.

#### **Cart & Wishlist Management**

- Users can add products to the cart or wishlist.
- Products in the cart can be removed or updated.

### **Checkout & Payments**

- Integration with Razorpay for secure payments.
- Order confirmation page upon successful payment.

### **Order Tracking**

• Users can view past orders and their status.

### 6. Payment Integration (Razorpay)

#### **Installing Razorpay**

pip install razorpay

#### **Configuring Razorpay in settings.py**

```
RAZORPAY_API_KEY = 'your_razorpay_api_key'

RAZORPAY API SECRET = 'your razorpay api secret'
```

### Payment View (checkout.py)

```
import razorpay
from django.conf import settings
client = razorpay.Client(auth=(settings.RAZORPAY API KEY, settings.RAZORPAY API SECRET))
def checkout(request):
  order_amount = 50000 # Example amount in paise
  order currency = 'INR'
  order_receipt = 'order_rcptid_11'
  order = client.order.create({
    'amount': order amount,
    'currency': order currency,
    'receipt': order_receipt,
    'payment capture': '1'
 })
  return render(request, 'checkout.html', {'order': order})
```

### 7. UI/UX Considerations

- Responsive Design: Works on desktop and mobile.
- Minimalist UI: Clean and easy-to-navigate interface.
- Promotional Banners: Highlights healthy grocery benefits.
- Engaging Wishlist: Encourages users to save items.

### 8. Deployment Guide

### Steps

1. Migrate Database

python manage.py migrate

2. Create Superuser

python manage.py createsuperuser

3. Run Server Locally

python manage.py runserver

4. Deploy to Cloud (Render)

git init

git add.

git commit -m 'Initial commit'

git push heroku main

- 5. Configure Domain & SSL
- Use Nginx or Cloudflare for secure hosting.

### 9. Conclusion

AmrutAahar is a **complete e-commerce solution** for healthy grocery shopping, integrating modern functionalities such as product filtering, cart management, Razorpay payments, and a beautiful UI. With a solid architecture and scalable backend, it can be extended to include more features like subscription-based groceries, Al-based recommendations, and mobile app integration.

Link: https://amrutaahar.onrender.com