HomeBrew

A cross platform community driven marketplace to empower home makers and daily wage workers by providing them with E-commerce,Logistics and an additional source of income created using Django Rest Framework, React.js and Flutter.

Table Of Contents

1. Software Requirement Specification

1.1 Introduction

1.1.1 Purpose

Good food is one of the essential items for people's healthy daily living or otherwise. It is always hard for young professionals, office goers, boarders to have good home-cooked food unless they start preparing for themselves. For these segments of people, good home cooked food is most of the distant reach. Hence there is an enormous potential in this category of business, especially since Home cooks lack a proper online platform to sell, connect and share as a community. Earnings of Rickshaw drivers are adversely affected due to travel restrictions and surge of fuel prices. Even if 10% of the $378 million home chef market is tapped it can give gross revenue of $37.8 million annually for "home chef" in this market segment.

1.1.2 Overview

Upon joining as a member on the site, the users can enter their address and create their profile. It provides an easy-to-use interface for the registered users to browse various kitchens of homecooks and add home made food from these kitchens to their carts.This makes their lives easier as they can get delicious home made food delivered right to their doorstep.The users can track the live delivery details and also provide ratings to the home chefs food that they received.The delivery is undertaken by Auto-Rickshaw drivers through an app that is available to them.The Homechefs have access to an admin dashboard where they can upload details of the food that they are selling and can also see the order details and manage inventory.

1.2 General Description

1.2.1 Product Perspective

It is a web based system implementing a client-server model. HomeBrew is a community driven Platform to Empower Home-makers and daily wage workers by providing them with ecommerce, logistics and an additional source of income. We enable home makers to form a community , set them up with the ecommerce required to sell their homely foods and bakeries. We take care of the logistics by employing auto rickshaw drivers and daily wage workers to transport the products from the user source to the customer .All the resources used by home makers are locally sourced from organic farmers in and around the area providing an alternative source of livelihood for all parties.

The main interface of homebrew is a website where the user can see the various home chef's menus and can add the food from the home chefs to their cart. The users can then checkout the items and pay for them online. The details of the order are sent to the home chef. The home chef can view and manage the order using the admin dashboard. They can also manage inventory and stock using the same admin panel. The Admin can add the home chefs to the platform using the admin panel.

Following screens will be provided as part of the User Interface:

1. A signup or login screen for entering the username and password.
2. User account screens for editing and viewing account details.
3. Home made food store screen showing various kitchens and food items by home chefs.
4. Food details screen to show the ratings and further details about the homemade food.
5. Cart screen showing food items currently in the user's cart.
6. Checkout screen showing summary of order and total amount.
7. Payment screen showing payment details.
8. Order Tracking screen showing live details of the order.
9. User profiles list and screens to show further details including order history.
10. Admin Panel for HomeChefs.
11. Location screen for Auto Rickshaw drivers.

1.2.2 Product Features

Currently, 'HomeBrew' currently provides the following features:

- Offers operating support for most of the known and commercial operating systems.
- The system allows the user to create their accounts in the system using Email and password or Social Authentication like Google and provide features of updating and viewing profiles.
- Allows users to search and purchase home made food from the database, which is updated regularly by the home-chefs.
- Online payment and Order History.
- Live Order Tracking and Delivery of food.
- Allows homehefs to manage orders and inventory through the admin panel.
- Only admin has access to information of product stock, storage location, storage date and vendor details.
- Customers can buy more than one product at a time.
- Invoice is issued with the name and details of the customer after purchasing the product.

1.2.3 User Characteristics

The different users that will use this product are:

- Customer- View and buy homemade food.
- Auto Rickshaw Drivers - Uses the delivery app and delivers food.
- Home Chefs - Updates homemade food and manages orders and inventory
- Admin - Add new home chefs and autorickshaw drivers and keep track of orders.

It is assumed that the users have the basic knowledge of operating the internet and have access to it. The administrator is expected to be familiar with the interface of the basic web based system as well as the Django Admin Page.

1.2.4 Design and Implementation Constraints

This system is provisioned to be built on the Django framework which is highly flexible. User authentication shall be managed by the firebase authentication platform to shorten development time and improve scalability. The following are the current constraints of the project:

- Only registered users have the ability to access the platform.
- The live location tracking cannot be tested due to lack of proper technical infrastructure.
- Currently only one Home chef can be added to the platform.
- App for Auto Rickshaw drivers was not developed.
- Payments cannot go through due to restrictions on KYC rules.
- The GUI is only available in English.

1.2.5 Operating System Environment

HomeBrew requires a Linux,Windows or Mac OS based server that supports Javascript, Python and SQLite.

1.3 Specific Requirements

1.3.1 Functional Requirements

1. Admin must be able to add new products
2. Admin must be able to enter entire stock details
3. Admin must be able to edit product details
4. Admin must be able to delete a product
5. Admin must be able to view product details
6. Admin must be able to sort product name and it’s price
7. Admin must be able to view daily customer’s data
8. Admin must be able to view daily sale report
9. Admin must be able to insert customer data
10. Admin must be able to log out
11. Admin must be able to change software password
12. Admin must be able to view and print customer bill
13. User must be able to login
14. User must be able to register
15. User must be able to login using google account
16. User must be able to reset password
17. User must be able to add items to cart

18. User must be able to add their address
19. User must be able to checkout
20. User must be able to pay for the items
21. User must be able to track the order

1.3.2 Non-Functional Requirements

1. System availability
2. Allowance for maintainability and enhancements
3. Recovery from failure
4. Reliability
5. Response time
6. Throughput
7. Order management communication
8. Order verification, confirmation and fulfilment
9. Preparation, distribution and inventory control, storage and security
10. Administration
11. Intervention and monitoring

1.3.3 External Interface Requirements

1.3.3.1 Hardware Interfaces

The hardware interface required for the user to retrieve the data from the server and for tracking in which the web portal is hosted mainly involves a server and a personal pc connected through a network interface. The hardware requirements include a Windows or Unix based Operating System, and a Browser that supports Javascript.

1.3.3.2 Software Interfaces

The complete user data is stored in the database and the information is accessed by the user through a web browser. The software techstack includes:
1. Django REST Framework
2. Firebase Auth
3. SQLite
4. React JS

1.3.4 Software System Features

1. Product Catalogue management: To digitally keep track of the different home made food that are available for purchase from the home chefs and the stock of each item.
2. Order Management: To see the different orders that a particular home chef has received and to fulfil these orders.
3. Inventory Management : To see and manage different inventory items by the homechef.Only homechefs have access to information of product stock, storage location, storage date and vendor details.

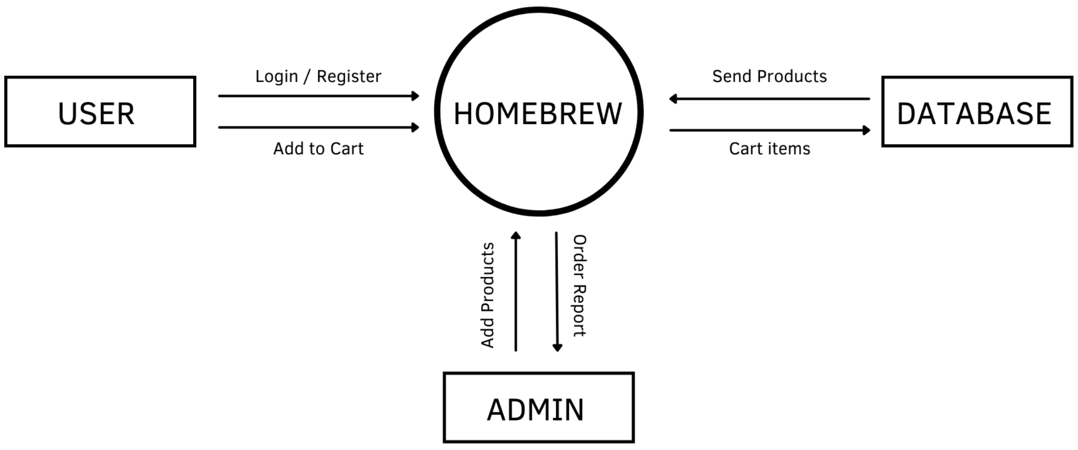
4. Profile Management : To view/edit user profiles and addresses.
5. To view/order home made food items from the home chef.
6. To view, manage and track user orders.
7. Admin can create, view, update and delete all user accounts, home chef accounts and Auto Rickshaw driver accounts.
8. Payment Module : To manage payments and Transactions.

2. System Design

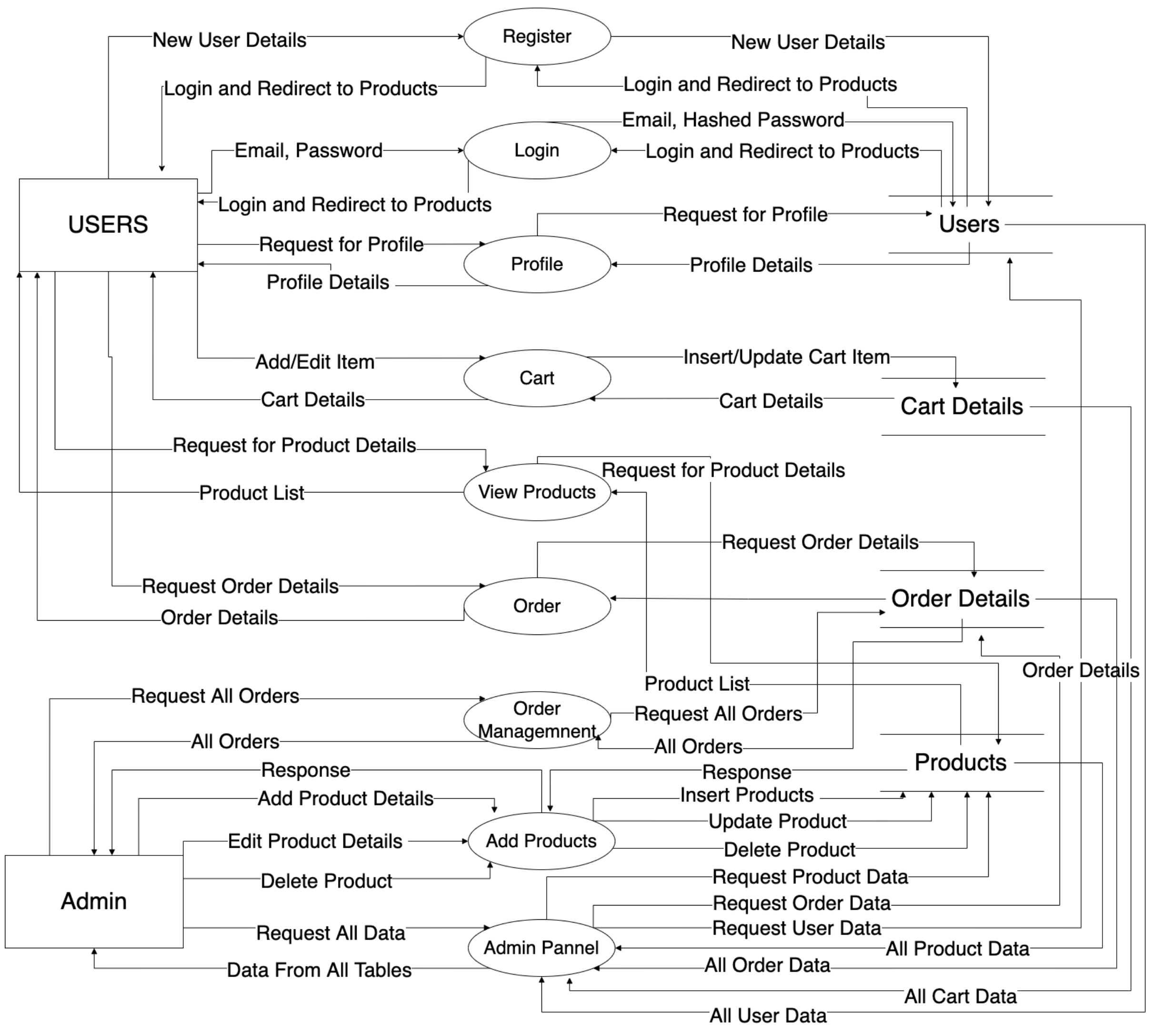
2.1 Data Flow Diagrams

DFDs are used to represent the flow of data through different modules of the system. An integrated Data Flow Diagram is used to represent the overall system.

2.1.1 Level 0



2.1.2 Level 1



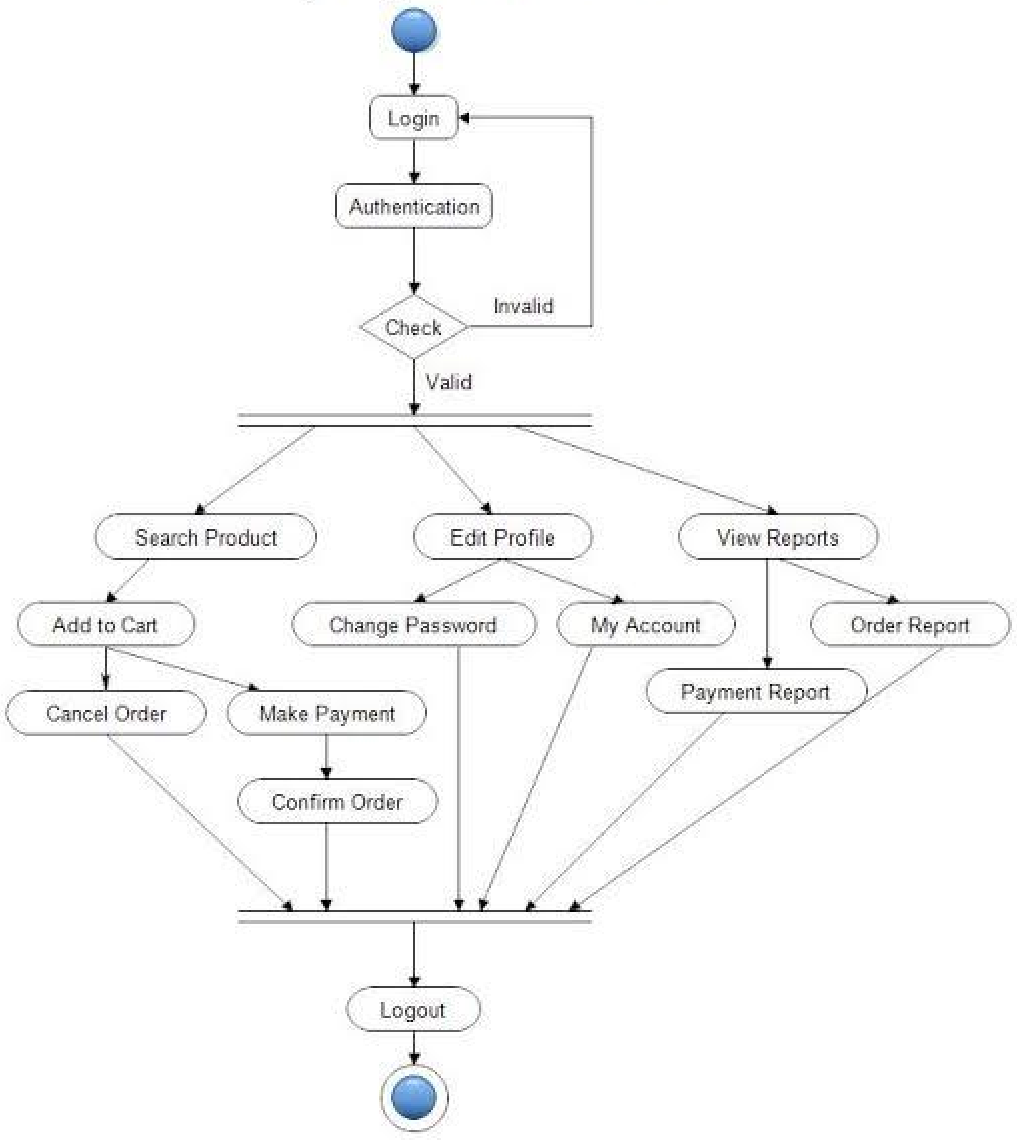
2.2 UML Diagrams

The main aim of UML is to define a standard way to visualise the way a system has been designed. We use UML diagrams to portray the behaviour and structure of a system.

2.2.1 Activity Diagram

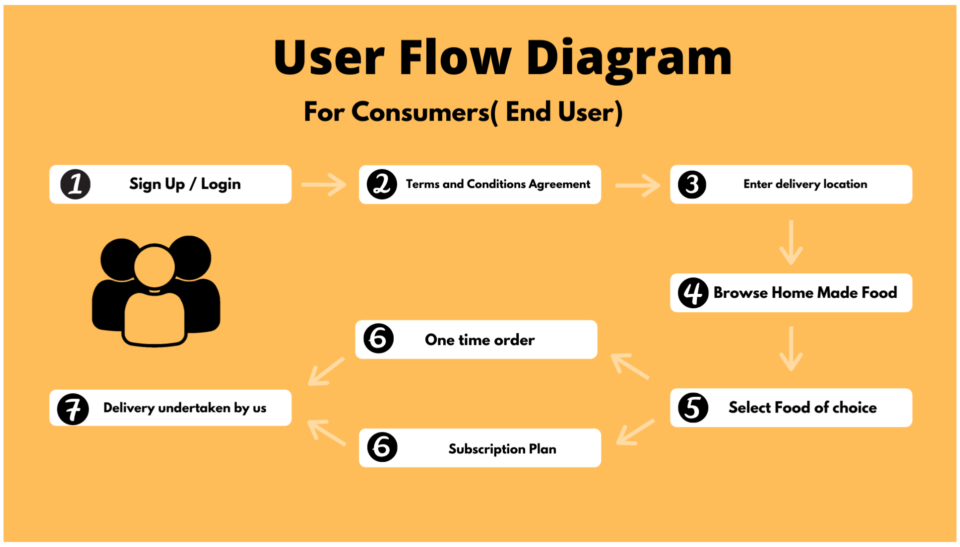
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

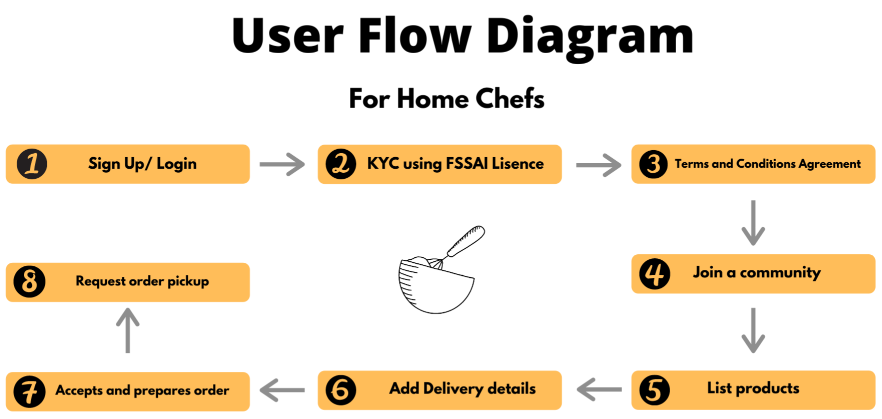
Activity Diagram for User Side



2.2.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

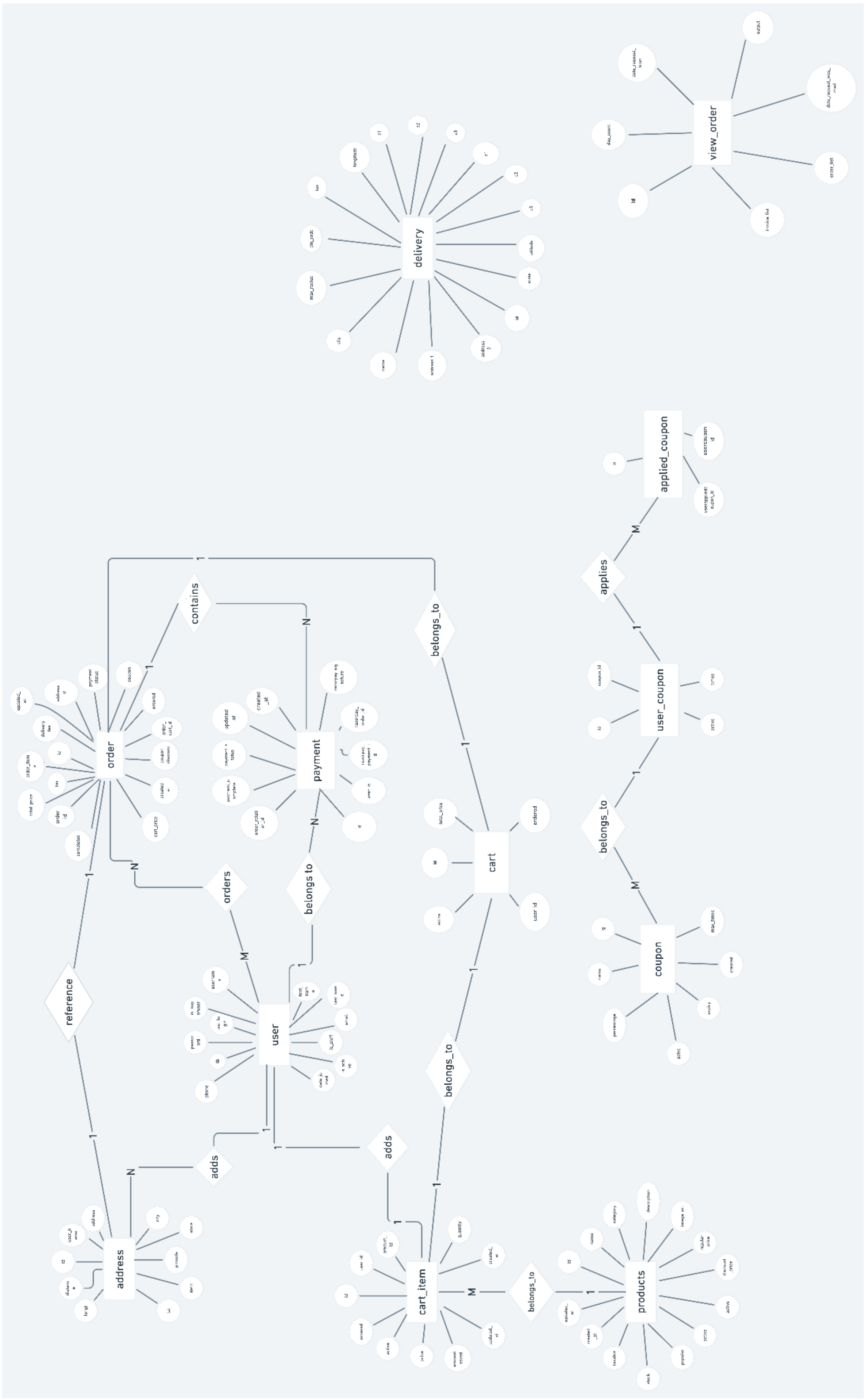


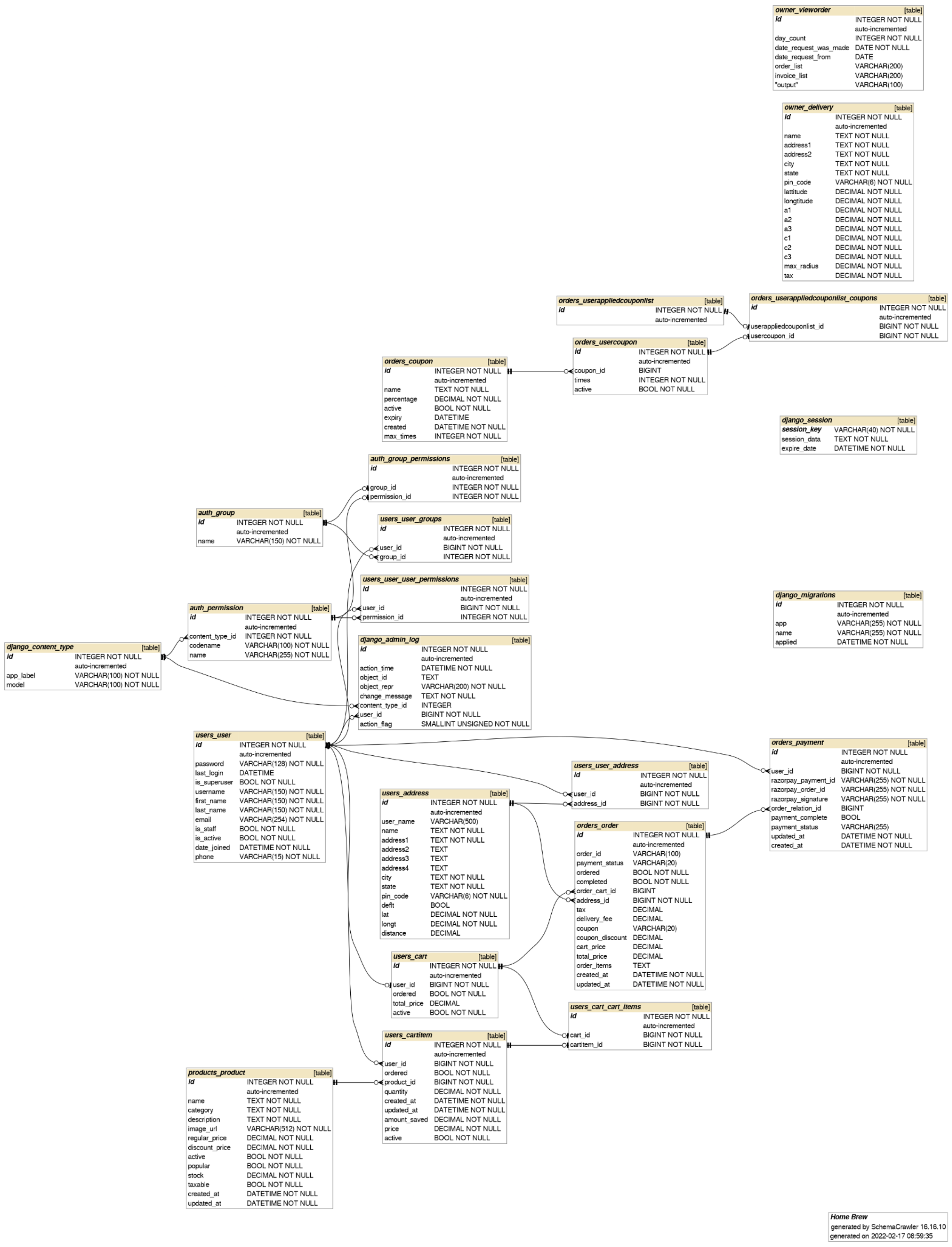


2.3 Entity-Relationship Diagram

ER Diagram or Entity Relationship Diagram, also known as ERD, is a diagram that displays the relationship of entity sets stored in a database. In other words, it helps explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER diagram for the system is given below :


order
reference
address
adds
user
orders
belongs to
payment
contains
belongs_to
cart
adds
cart_item
belongs_to
products
coupon
belongs_to
user_coupon
applies
applied_coupon
delivery
view_order
1
N
M



owner_vieworder	[table]
id	INTEGER NOT NULL
	auto-incremented
day_count	INTEGER NOT NULL
date_request_was_made	DATE NOT NULL
date_request_from	DATE
order_list	VARCHAR(200)
invoice_list	VARCHAR(200)
"output"	VARCHAR(100)

owner_delivery	[table]
id	INTEGER NOT NULL
	auto-incremented
name	TEXT NOT NULL
address1	TEXT NOT NULL
address2	TEXT NOT NULL
city	TEXT NOT NULL
state	TEXT NOT NULL
pin_code	VARCHAR(6) NOT NULL
lattitude	DECIMAL NOT NULL
longtitude	DECIMAL NOT NULL
a1	DECIMAL NOT NULL
a2	DECIMAL NOT NULL
a3	DECIMAL NOT NULL
c1	DECIMAL NOT NULL
c2	DECIMAL NOT NULL
c3	DECIMAL NOT NULL
max_radius	DECIMAL NOT NULL
tax	DECIMAL NOT NULL

orders_userappliedcouponlist	[table]
id	INTEGER NOT NULL
	auto-incremented

orders_userappliedcouponlist_coupons	[table]
id	INTEGER NOT NULL
	auto-incremented
userappliedcouponlist_id	BIGINT NOT NULL
usercoupon_id	BIGINT NOT NULL

orders_usercoupon	[table]
id	INTEGER NOT NULL
	auto-incremented
coupon_id	BIGINT
times	INTEGER NOT NULL
active	BOOL NOT NULL

orders_coupon	[table]
id	INTEGER NOT NULL
	auto-incremented
name	TEXT NOT NULL
percentage	DECIMAL NOT NULL
active	BOOL NOT NULL
expiry	DATETIME
created	DATETIME NOT NULL
max_times	INTEGER NOT NULL

django_session	[table]
session_key	VARCHAR(40) NOT NULL
session_data	TEXT NOT NULL
expire_date	DATETIME NOT NULL

auth_group_permissions	[table]
id	INTEGER NOT NULL
	auto-incremented
group_id	INTEGER NOT NULL
permission_id	INTEGER NOT NULL

auth_group	[table]
id	INTEGER NOT NULL
	auto-incremented
name	VARCHAR(150) NOT NULL

users_user_groups	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
group_id	INTEGER NOT NULL

users_user_user_permissions	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
permission_id	INTEGER NOT NULL

django_migrations	[table]
id	INTEGER NOT NULL
	auto-incremented
app	VARCHAR(255) NOT NULL
name	VARCHAR(255) NOT NULL
applied	DATETIME NOT NULL

auth_permission	[table]
id	INTEGER NOT NULL
	auto-incremented
content_type_id	INTEGER NOT NULL
codename	VARCHAR(100) NOT NULL
name	VARCHAR(255) NOT NULL

django_content_type	[table]
id	INTEGER NOT NULL
	auto-incremented
app_label	VARCHAR(100) NOT NULL
model	VARCHAR(100) NOT NULL

django_admin_log	[table]
id	INTEGER NOT NULL
	auto-incremented
action_time	DATETIME NOT NULL
object_id	TEXT
object_repr	VARCHAR(200) NOT NULL
change_message	TEXT NOT NULL
content_type_id	INTEGER
user_id	BIGINT NOT NULL
action_flag	SMALLINT UNSIGNED NOT NULL

users_user	[table]
id	INTEGER NOT NULL
	auto-incremented
password	VARCHAR(128) NOT NULL
last_login	DATETIME
is_superuser	BOOL NOT NULL
username	VARCHAR(150) NOT NULL
first_name	VARCHAR(150) NOT NULL
last_name	VARCHAR(150) NOT NULL
email	VARCHAR(254) NOT NULL
is_staff	BOOL NOT NULL
is_active	BOOL NOT NULL
date_joined	DATETIME NOT NULL
phone	VARCHAR(15) NOT NULL

orders_payment	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
razorpay_payment_id	VARCHAR(255) NOT NULL
razorpay_order_id	VARCHAR(255) NOT NULL
razorpay_signature	VARCHAR(255) NOT NULL
order_relation_id	BIGINT
payment_complete	BOOL
payment_status	VARCHAR(255)
updated_at	DATETIME NOT NULL
created_at	DATETIME NOT NULL

users_user_address	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
address_id	BIGINT NOT NULL

users_address	[table]
id	INTEGER NOT NULL
	auto-incremented
user_name	VARCHAR(500)
name	TEXT NOT NULL
address1	TEXT NOT NULL
address2	TEXT
address3	TEXT
address4	TEXT
city	TEXT NOT NULL
state	TEXT NOT NULL
pin_code	VARCHAR(6) NOT NULL
deflt	BOOL
lat	DECIMAL NOT NULL
longt	DECIMAL NOT NULL
distance	DECIMAL

orders_order	[table]
id	INTEGER NOT NULL
	auto-incremented
order_id	VARCHAR(100)
payment_status	VARCHAR(20)
ordered	BOOL NOT NULL
completed	BOOL NOT NULL
order_cart_id	BIGINT
address_id	BIGINT NOT NULL
tax	DECIMAL
delivery_fee	DECIMAL
coupon	VARCHAR(20)
coupon_discount	DECIMAL
cart_price	DECIMAL
total_price	DECIMAL
order_items	TEXT
created_at	DATETIME NOT NULL
updated_at	DATETIME NOT NULL

users_cart	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
ordered	BOOL NOT NULL
total_price	DECIMAL
active	BOOL NOT NULL

users_cart_cart_items	[table]
id	INTEGER NOT NULL
	auto-incremented
cart_id	BIGINT NOT NULL
cartitem_id	BIGINT NOT NULL

users_cartitem	[table]
id	INTEGER NOT NULL
	auto-incremented
user_id	BIGINT NOT NULL
ordered	BOOL NOT NULL
product_id	BIGINT NOT NULL
quantity	DECIMAL NOT NULL
created_at	DATETIME NOT NULL
updated_at	DATETIME NOT NULL
amount_saved	DECIMAL NOT NULL
price	DECIMAL NOT NULL
active	BOOL NOT NULL

products_product	[table]
id	INTEGER NOT NULL
	auto-incremented
name	TEXT NOT NULL
category	TEXT NOT NULL
description	TEXT NOT NULL
image_url	VARCHAR(512) NOT NULL
regular_price	DECIMAL NOT NULL
discount_price	DECIMAL NOT NULL
active	BOOL NOT NULL
popular	BOOL NOT NULL
stock	DECIMAL NOT NULL
taxable	BOOL NOT NULL
created_at	DATETIME NOT NULL
updated_at	DATETIME NOT NULL

Home Brew
generated by SchemaCrawler 16.16.10
generated on 2022-02-17 08:59:35

3. Detailed Design

3.1 Input Phase

3.1.1 User Inputs

Each user has a number of attributes linked to itself, including their id (auto-generated), username, email, password, first name, last name, permissions,group that they belong to, Last login and Date Joined. On creation (sign-up) of a new user, the input email and password (or linked google or facebook account) is validated in terms of its data format as well as checked for pre-existing entry and then entered into the database. Login allows authentication either using the linked account or the manually entered email and password, thus initiating and maintaining the session until a logout event. This authentication is leveraged through Firebase Authentication and the relevant screens are given in Fig. 5.1 & 5.2.

Users can modify their account details in the corresponding tab such that the data formats are preserved. The users can browse the list of home cooked food and add the ones that they want to purchase to the cart. The users can add multiple products to the cart. On clicking the cart icon on the top right, the user can review the items that he/she wishes to purchase and proceed to checkout and make the payment.The user can also view and edit his profile by clicking on the profile icon on the top right. The relevant screens listing the browsable Home cooked food, Cart and User Profiles are given in Fig. 5.3 - 5.6.

The Homechefs can view the orders and also manage their products through the admin panel by logging in with their credentials that are assigned to them by the admin.They can also request for invoices from the same panel. The relevant screens are given in Fig. 5.7 - Fig 5.8.

3.1.2 Admin Inputs

The home chef and autorickshaw driver details must be added directly by the admin, using the Django Admin Page. The users edited by the admin are defined by the attributes id(auto-generated), username, email, password, first name, last name, permissions,group that they belong to, Last login and Date Joined. Of these attributes the permission and group attribute decide whether the user is a consumer, home chef, admin or an auto rickshaw driver. The admin can also modify the other entities and their attributes with operations such as creating and deleting user accounts. The relevant screens contained in the Django Admin Dashboard are given in Fig. 5.9.

3.2 Conversion Phase

Serializers allow complex data such as querysets and model instances to be converted to native Python data types that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data. The serializers in the Django REST framework on the backend are used during the passage of data between the frontend and the database.

3.3 Output Phase

The data in the database can be manipulated using CRUD operations on the respective Django REST API urls. A sample API screen is given in Fig. 5.10.

4. Tools & Technologies

The following were used in the implementation of the project:

- Backend: Django REST Framework
- Frontend: HTML, CSS, React.js
- Authentication: Firebase Authentication
- Database: SQLite

5. Screenshots

Fig. 5.1 Signup Page

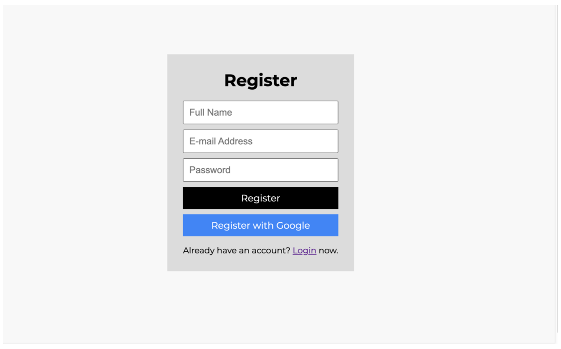


Fig. 5.2 Login Page

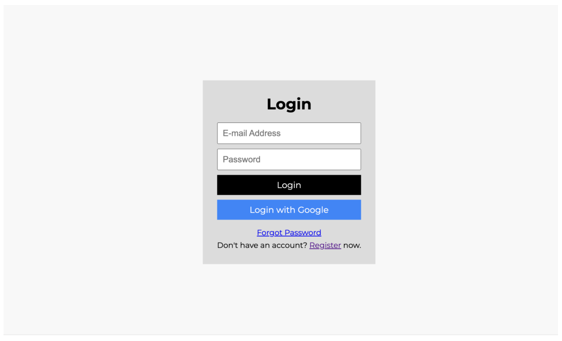


Fig. 5.3 User Account Page

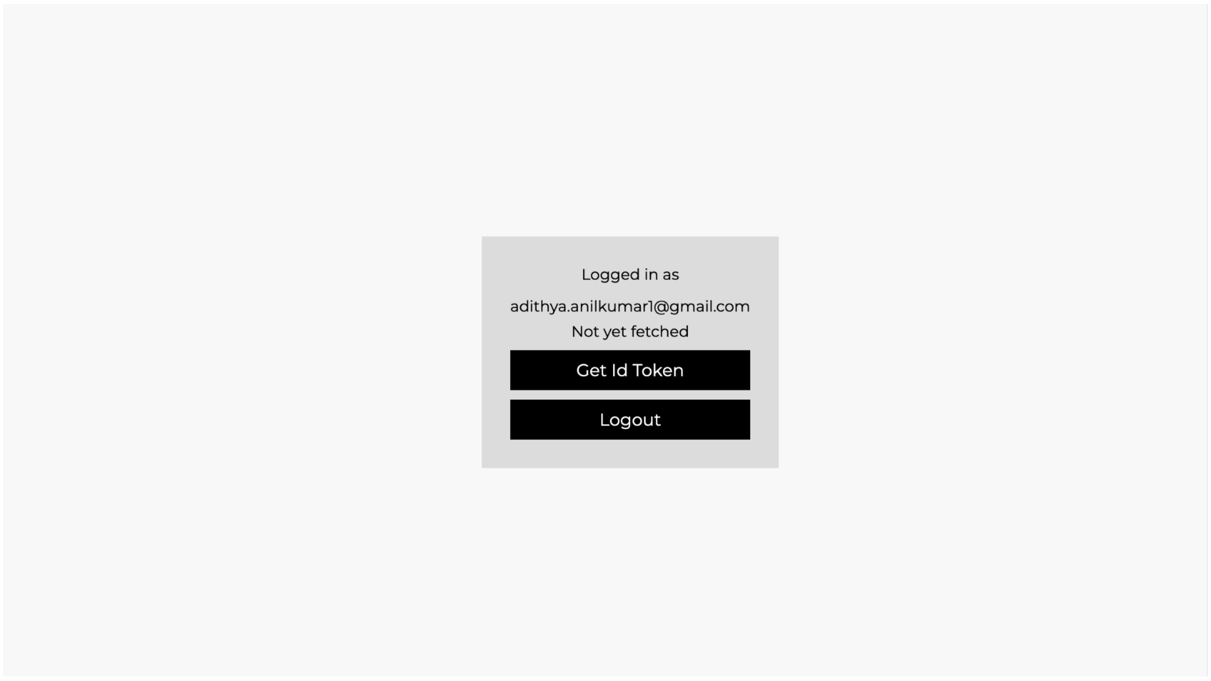


Fig. 5.4 HomeBrew Home Page

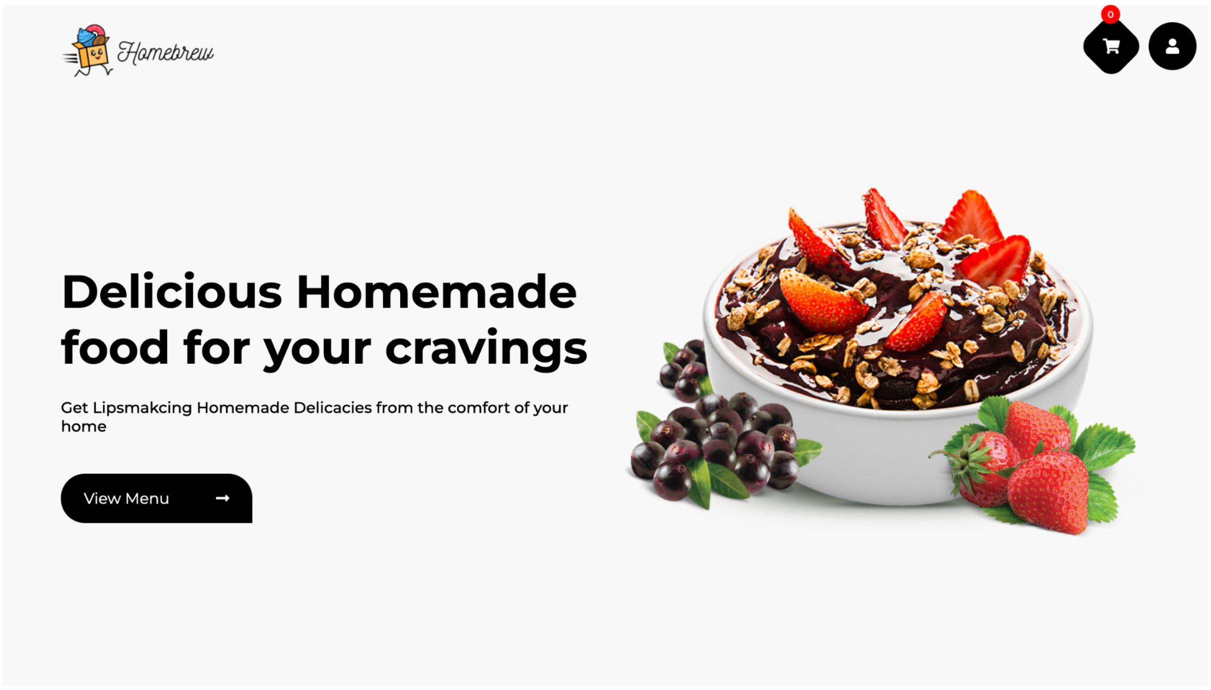
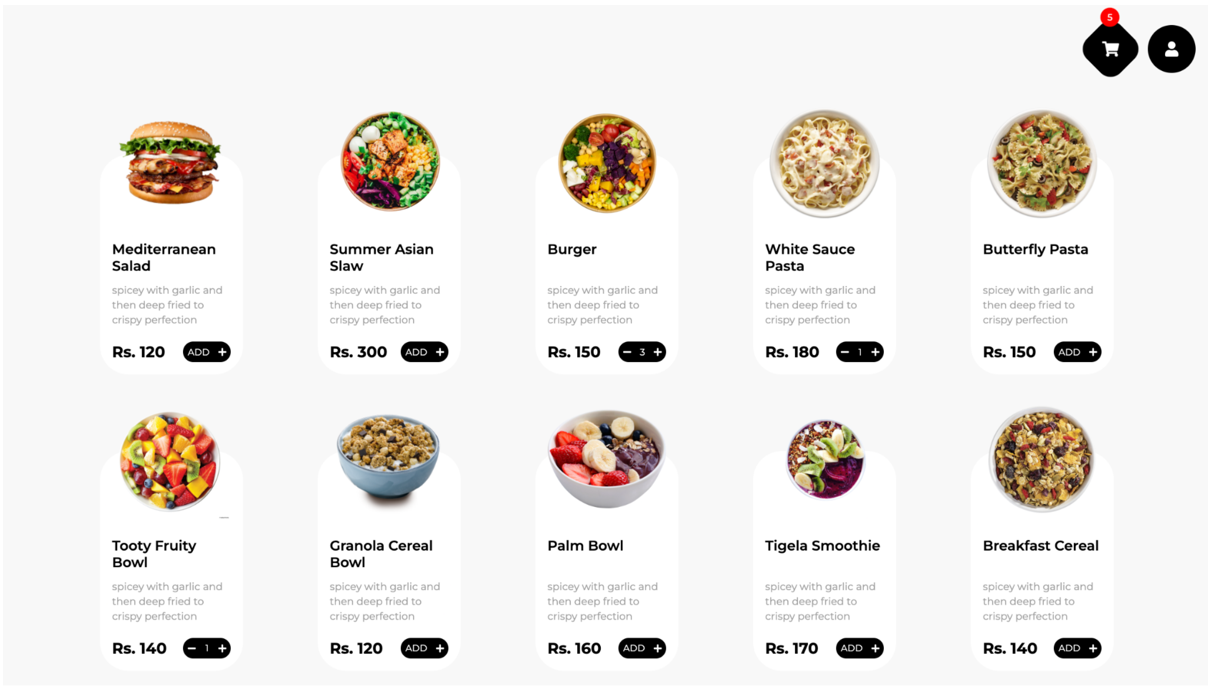

Homebrew
0
Delicious Homemade food for your cravings
Get Lipsmakcing Homemade Delicacies from the comfort of your home
View Menu

Fig. 5.5 Homely Food Store Page



Mediterranean Salad
spicey with garlic and then deep fried to crispy perfection
Rs. 120 ADD +

Summer Asian Slaw
spicey with garlic and then deep fried to crispy perfection
Rs. 300 ADD +

Burger
spicey with garlic and then deep fried to crispy perfection
Rs. 150 − 3 +

White Sauce Pasta
spicey with garlic and then deep fried to crispy perfection
Rs. 180 − 1 +

Butterfly Pasta
spicey with garlic and then deep fried to crispy perfection
Rs. 150 ADD +

Tooty Fruity Bowl
spicey with garlic and then deep fried to crispy perfection
Rs. 140 − 1 +

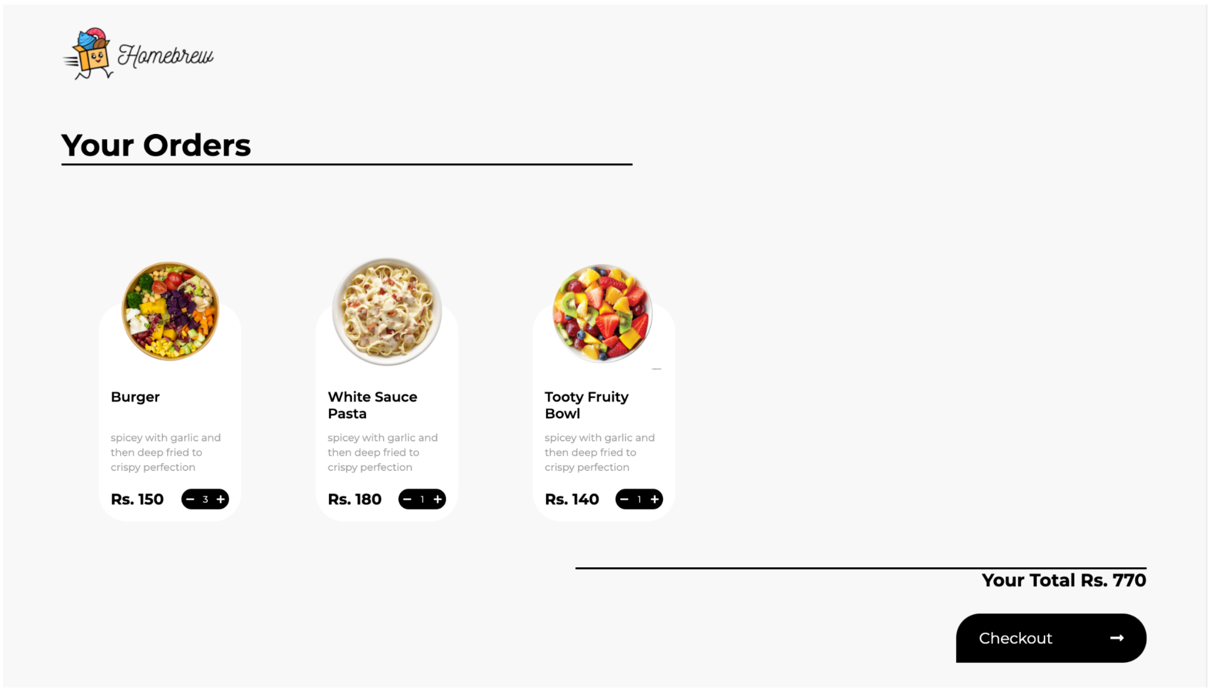
Granola Cereal Bowl
spicey with garlic and then deep fried to crispy perfection
Rs. 120 ADD +

Palm Bowl
spicey with garlic and then deep fried to crispy perfection
Rs. 160 ADD +

Tigela Smoothie
spicey with garlic and then deep fried to crispy perfection
Rs. 170 ADD +

Breakfast Cereal
spicey with garlic and then deep fried to crispy perfection
Rs. 140 ADD +

Fig. 5.6 User Cart Page



Your Orders

Burger
spicey with garlic and then deep fried to crispy perfection
Rs. 150 − 3 +

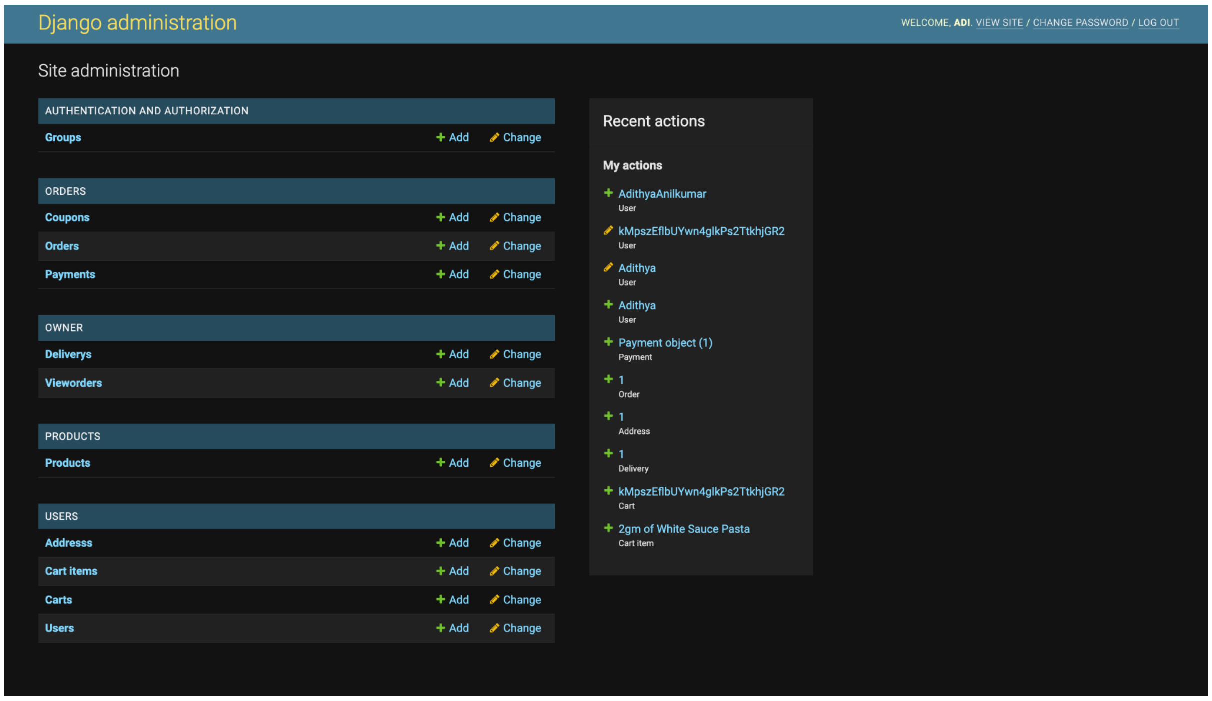
White Sauce Pasta
spicey with garlic and then deep fried to crispy perfection
Rs. 180 − 1 +

Tooty Fruity Bowl
spicey with garlic and then deep fried to crispy perfection
Rs. 140 − 1 +

Your Total Rs. 770

Checkout →

Fig. 5.7 Django Admin Page



Django administration

WELCOME, ADI. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change

ORDERS		
Coupons	+ Add	Change
Orders	+ Add	Change
Payments	+ Add	Change

OWNER		
Deliverys	+ Add	Change
Vieworders	+ Add	Change

PRODUCTS		
Products	+ Add	Change

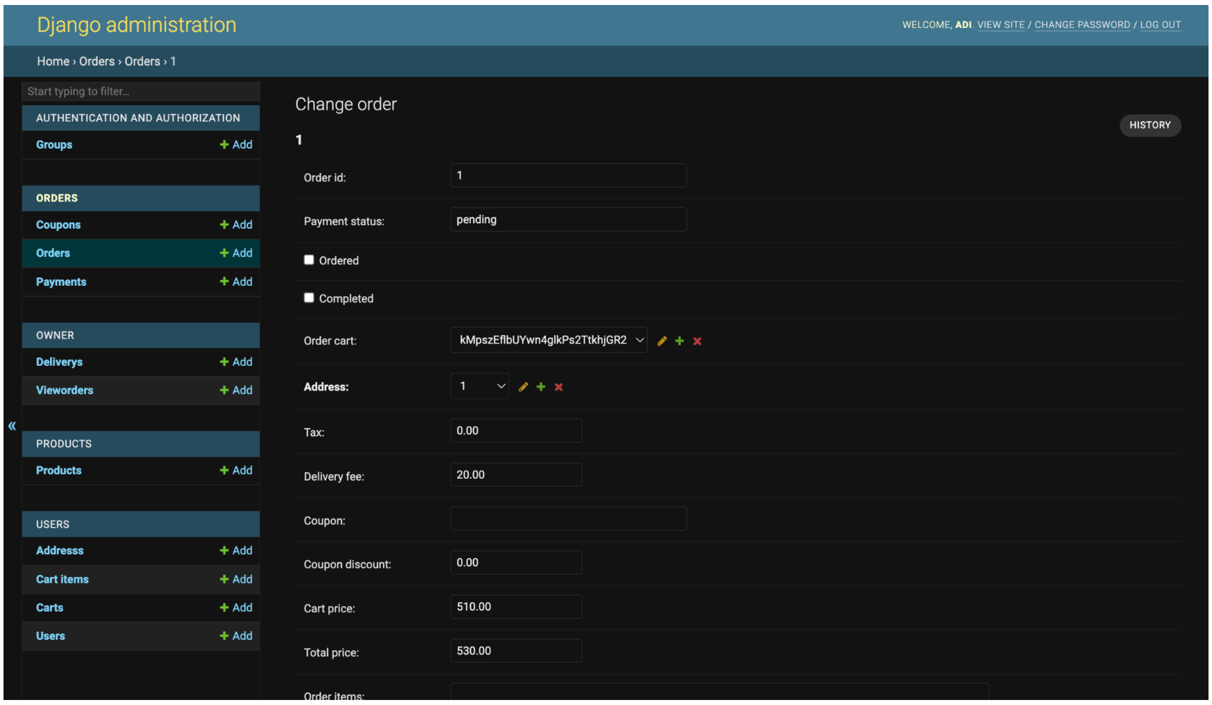
USERS		
Addresss	+ Add	Change
Cart items	+ Add	Change
Carts	+ Add	Change
Users	+ Add	Change

Recent actions

My actions

- AdithyaAnilkumar
  User
- kMpszEflbUYwn4glkPs2TtkhjGR2
  User
- Adithya
  User
- Adithya
  User
- Payment object (1)
  Payment
- 1
  Order
- 1
  Address
- 1
  Delivery
- kMpszEflbUYwn4glkPs2TtkhjGR2
  Cart
- 2gm of White Sauce Pasta
  Cart item

Fig. 5.8 Home Chef Order Management Page



Django administration

WELCOME, ADI. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Orders › Orders › 1

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add

ORDERS	
Coupons	+ Add
Orders	+ Add
Payments	+ Add

OWNER	
Deliverys	+ Add
Vieworders	+ Add

PRODUCTS	
Products	+ Add

USERS	
Addresss	+ Add
Cart items	+ Add
Carts	+ Add
Users	+ Add

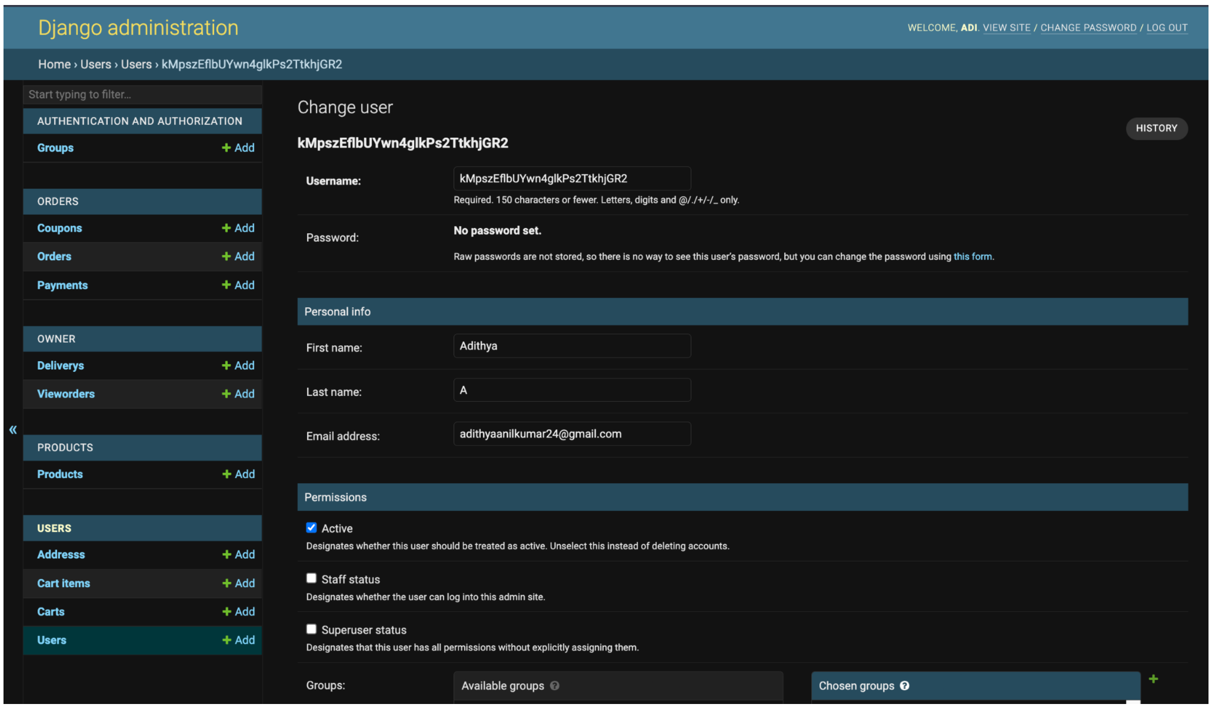
Change order

HISTORY

1

Order id:	1
Payment status:	pending
☐ Ordered	
☐ Completed	
Order cart:	kMpszEflbUYwn4glkPs2TtkhjGR2
Address:	1
Tax:	0.00
Delivery fee:	20.00
Coupon:	
Coupon discount:	0.00
Cart price:	510.00
Total price:	530.00
Order items:	

Fig. 5.9 Admin Modification Capabilities



Change user

kMpszEflbUYwn4glkPs2TtkhjGR2

Username: kMpszEflbUYwn4glkPs2TtkhjGR2

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **No password set.**

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name: Adithya

Last name: A

Email address: adithyaanilkumar24@gmail.com

Permissions

[✓] Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

[ ] Staff status

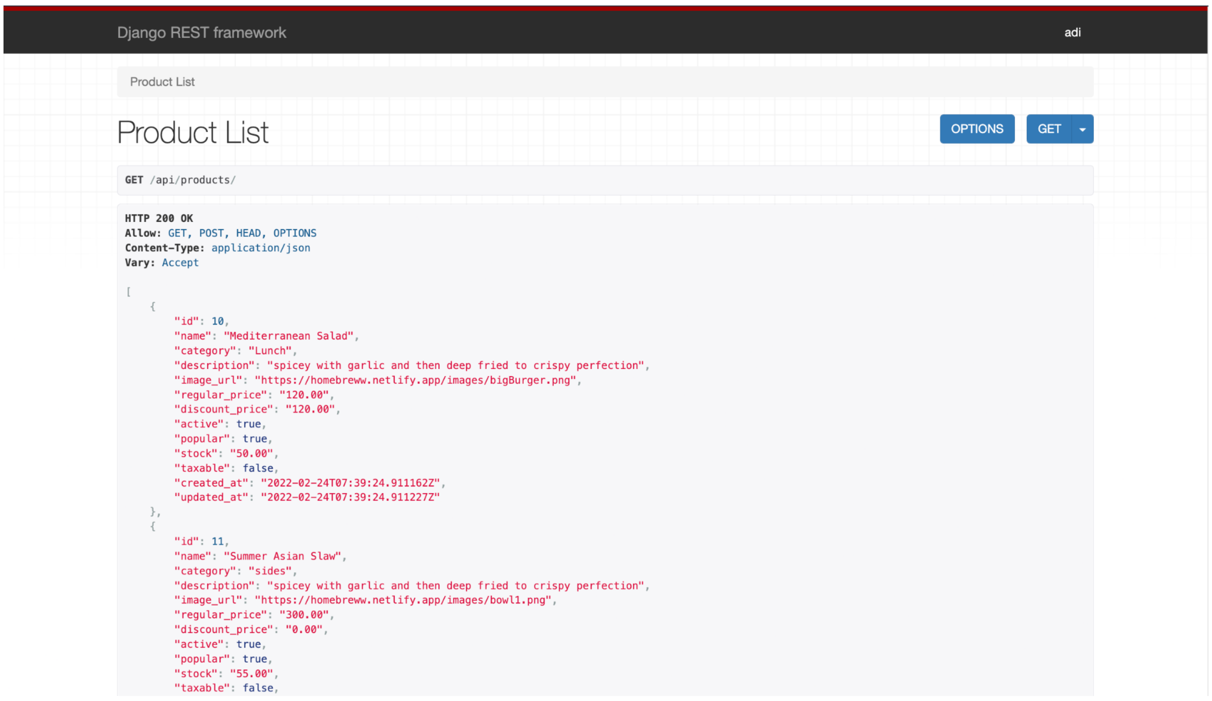
Designates whether the user can log into this admin site.

[ ] Superuser status

Designates that this user has all permissions without explicitly assigning them.

Groups: Available groups | Chosen groups

Fig. 5.10 Sample API JSON



Product List

```
GET /api/products/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 10,
        "name": "Mediterranean Salad",
        "category": "Lunch",
        "description": "spicey with garlic and then deep fried to crispy perfection",
        "image_url": "https://homebreww.netlify.app/images/bigBurger.png",
        "regular_price": "120.00",
        "discount_price": "120.00",
        "active": true,
        "popular": true,
        "stock": "50.00",
        "taxable": false,
        "created_at": "2022-02-24T07:39:24.911162Z",
        "updated_at": "2022-02-24T07:39:24.911227Z"
    },
    {
        "id": 11,
        "name": "Summer Asian Slaw",
        "category": "sides",
        "description": "spicey with garlic and then deep fried to crispy perfection",
        "image_url": "https://homebreww.netlify.app/images/bowl1.png",
        "regular_price": "300.00",
        "discount_price": "0.00",
        "active": true,
        "popular": true,
        "stock": "55.00",
        "taxable": false,
```

6. Conclusion

The project has been completed successfully as specified by the requirements. The implementation and testing has been done in a step-by-step manner. Each module has been developed and tested individually to obtain the required output in the desired form. While working on this interesting project, we learned various tools and technologies. We also attained valuable experience on the different stages involved in developing any web application that would be useful in the technology industry.

Within the duration of this project we learned the following things through the implementation and testing of the project - Python, MySQL database management, JavaScript, HTML, CSS and Bootstrap for UI/UX. Future improvements can be made in certain areas of the project. There is scope for extending the project to incorporate more features by including an advanced live tracking system with notifications, enabling food reviews, integrating payment gateway, developing an app for auto rickshaw drivers, enabling multi home cook kitchen, etc. The process model selected was agile, such that new features can be easily incorporated to the same design at a later point in time.