I have made a folder named as “Zenatix” which consists of-

- My Django project, named as “bakery”. It consists of 2 apps:
    - **Admin –** Which consists of the Admin’s APIs
    - **Customer –** Which consists of the Customer’s APIs
- The name of my database is “bakery_database” and it consists of the following main tables:
    - **Ingredient_table –** Holds all the ingredients of the bakery.
    - **bakery_items –** Holds all the items present in the basery with their details.
    - **registered_customers –** Holds all the values of a user for registration and login.
    - **bakery_orders –** Holds all the details of all the orders of the bakery made by the users. Also used to generate bill of the user
    - And other default tables which are formed by Django itself.

Database schema:

bakery_database

- Ingredient_table
    - Ingredient_id
    - Ingredient_name
    - quantity
- bakery_items
    - item_id
    - item_name
    - ingredients
    - ingredient_ids
    - ingredient_quantity
    - item_quantity
    - cost_price
    - selling_price
    - discount_percent
- registered_customers
    - User_id
    - full_name
    - mobile_no

    - email_id
    - address
    - password
    - country_code
- bakery_orders
    - Order_id
    - Full_name
    - mobile_no
    - order_items
    - order_quantities
    - pickup_order_date
    - pickuo_order_time
    - payment_method
    - payment_mode
    - total_price
    - return_amount
    - order_date
    - order_time
    - Country_code

The "Admin" app consists of the following APIs:

```
1. add_ingredients
2. createBakeryItem
3. detailOfBakeryItem
4. manageInventory
```

These APIs are found in the Admin/views.py file

The "Customer" app consists of the following APIs:

```
1. login_register
2. available_products
3. placeAnOrderAndGetBill
4. order_history
```

These APIs are found in the Customer/views.py file

Now I will show the working of all the APIs:

To use all the APIs you can type in something like this-

1. **Admin APIs**
    a. **Add_ingredients-** This API is responsible for adding an ingredient with its quantity to the “ingredient_table” table of the bakery.
    If the item is already present then it will not add the ingredient to the table.
    If the element is not present in the table then it will insert that ingredient in that table.

    *To use this API run the Django project my firstly changing the current directory to Zenatix/bakery by using the command “cd bakery”. When you have entered the bakery directory then type in the command “python manage.py runserver”. Make sure that before running the above command you have xampp installed on your system and you have started the “Apache” server and “MySQL” service of that and have also imported the “bakery_database” database from the “.sql file” I have provided with the code.
    * The next step is to open the postman app and import the collection “My Bakery” into your postman and open the My Bakery collection and open the “Admin” folder within that and select the first API whose name is “Add ingredients to bakery” and give in the following input after selecting the Body tab and selecting “raw” option within that and selecting “JSON” from the list on the right side.

    The input for this API is-

```
{
  "ingredient_name": "Wheat",
  "quantity": "10"
}
```

    And we will get the output as-

```
{
  "status": "success",
  "status_code": "200",
  "data": "",
```

```
  "count": "0",
  "message": "Ingredient has been added to Bakery."
}
```


***We get this output because this ingredient is not present in the list and it is inserted in the table.**

Another input will be-


```
{
  "ingredient_name": "Eggs",
  "quantity": "20"
}
```


And the output will be-


```
{
  "status": "fail",
  "status_code": "400",
  "data": "",
  "count": "0",
  "message": "Ingredient already exists."
}
```


***We get this output because the item “Eggs” is already present in the table.**

b. **createBakeryItem-** This API is responsible for adding an item into the table “bakery_items”. It firstly check that whether the name of the item is already in the table or not. If it is there then an error response will be displayedand it will not be added in the table. But if the element name is not in the table then it inserts the item in the table after it checks that the ingredients required to make this product are present in the “ingredient_table” table or not. If even 1 of the ingredient is not present in the ingredient_table then a fail response response will be shown with an appropriate message and if all the items are present then this item will be inserted in the bakery_items table.

Input:


```
{
   "BakeryItem_name": "Custard",
   "ingredient_list": "Cream",
   "cost_price": "30",
   "selling_price": "40",
   "ingredient_quantity": "40",
   "BakeryItem_quantity": "20"
}
```


Output:


```
{
   "status": "success",
   "status_code": "200",
   "data": "",
   "count": "0",
   "message": "Ingredient has been added to Bakery."
}
```


***Here the item was presented in the table so it was added to the table.**

Another input:


```
{
   "BakeryItem_name": "Pastry",
   "ingredient_list": "Cream",
   "cost_price": "30",
   "selling_price": "40",
   "ingredient_quantity": "40",
   "BakeryItem_quantity": "20"
}
```


Output:


```
{
   "status": "fail",
   "status_code": "400",
   "data": "",
```

```
  "count": "0",
  "message": "Item is already present in the inventory."
}
```


***Here the item was present in the table so it was not added.**

c. **detailOfbakeryItem-** This API will display the details of a single item whose name was given as input to this API. It firstly checks that whether that item is in the “bakery_items” tab or not. If it is there then it displays all its details and if it is not present in the table then a fail response will be shown.

Input:


```
{
  "BakeryItem_name": "Biscuit"
}
```


Output:


```
{
  "status": "success",
  "status_code": "200",
  "data": {
    "item_id": 16,
    "item_name": "Biscuit",
    "ingredients": "Milk, Eggs, Cream",
    "ingredient_ids": "1, 4, 5",
    "ingredient_quantity": "10, 20, 30",
    "item_quantity": 30,
    "cost_price": 200.0,
    "selling_price": 300.0,
    "discount_percent": null
  },
  "count": "0",
  "message": "Item details found."
}
```


***When the item is present then we get this output**

Another input:


```
{
  "BakeryItem_name": "Chocolate"
}
```


Output:


```
{
  "status": "fail",
  "status_code": "400",
  "data": "",
  "count": "0",
  "message": "No such item in the inventory."
}
```


***When the item is not present in the table we get this output.**

d. **manageInventory –** This API is responsible for managing the managing the items in the bakery and provides the Admin with the options of “update”, “delete”, “discount”, “get_items”, “get_ingredients” and also lets the user see the hottest selling product. In uses the table “bakery_items” and “ingredient_table” tables for performing all the above mentioned options. Since it works in various different ways so I am mentioning quite a few of the over here.

Input:


```
{
  "option": "discount",
  "item_name": "Pastry",
  "discount_percent": "10"
}
```


Output:


```
{
  "status": "success",
  "status_code": "200",
  "data": "",
  "count": "0",
```

```
   "message": "Discount value is updated successfully."
}
```


***It will update the discount value of an item in the table if that item exists. If the item doesn't exists then a fail response similar to as shown above will be returned.**

Another input:


```
{
   "option": "update",
   "old_item_name": "Custard",
   "new_item_name": "Mango custard"
}
```


Output:


```
{
   "status": "success",
   "status_code": "200",
   "data": "",
   "count": "0",
   "message": "Item name has been updated successfully."
}
```


***It will update the name of item from its old name to a new name only the old name exists in the table otherwise a fail response will be shown.**

Another input:


```
{
   "option": "update",
   "item_name": "Mango custard",
   "quantity": 60
}
```


Output:


```
{
   "status": "success",
   "status_code": "200",
```

```
  "data": "",
  "count": "0",
  "message": "Item details has been updated successfully."
}
```


***This will update the quantity or any other parameter(if provided as input) of an item if the item exists in the table otherwise an error response will be shown.**

Input:

If we do not provide any input then the most popular/hottest selling product will be shown to the Admin.

Output:


```
{
  "status": "success",
  "status_code": "200",
  "data": "Pastry",
  "count": "0",
  "message": "Most popular/Hottest selling product is Pastry"
}
```


***It shows the most popular product.**

2. **Customer APIs**
   a. **login_register-** This API will be responsible for allowing the user to either register or login to the bakery webste. If you want to register then you have to provide yous full name, mobile number, email id, address, password and then reenter the password in confirm password. The it checkes that whether all these details are already in the table or not. If they are not then all the input data will be stored in the table “registered_customers” otherwise an error response will be shown. While registering country code field is optional and have a default value of “+91”.
   For login you have to enter your email id and password with which you registered and then it matches the input in the table “registered_customers”. If no match is found then an error

response will be shown itherwise a welcome message will be shown to the user.

Input:


```
{
   "Email_id": "abc@gmail.com",
   "Password": "12345678"
}
```


Output:


```
{
   "status": "success",
   "status_code": "200",
   "data": "",
   "count": "0",
   "message": "Login successful. Welcome to the bakery!"
}
```


***It shows successful login.**

Another input:


```
{
   "Email_id": "abc@gmail.com",
   "Password": "1234567889"
}
```


Output:


```
{
   "status": "fail",
   "status_code": "400",
   "data": "",
   "count": "0",
   "message": "Invalid Email_Id or Password. Try again!!"
}
```


***When the user enters wrong email id or password.**

Another input:

{
"Full_name": "abc@gmail.com",
"Mobile_no": "1234567889",
"Email_id": "geffef@gmail.com",
"Address": "Kanpur",
"Password": "qwertyui",
"Confirm_Password": "qwertyui"
}

Output:

{
"status": "success",
"status_code": "200",
"data": "",
"count": "0",
"message": "You are registered successfully. Welcome to the bakery!"
}

***If the details does not exist then they are inserted in the table. If they are already present then a fail response will be shown.**

b. **available_products-** It uses ”GET” method to fetch a list of all the available products in the bakery.

Input:

It doesn’t takes any input

Output:

{
"status": "success",
"status_code": "200",
"data": [
{

```
    "item_id": 1,
    "item_name": "Pastry",
    "ingredients": "Milk, Eggs",
    "ingredient_ids": "",
    "ingredient_quantity": "10",
    "item_quantity": 20,
    "cost_price": 30.0,
    "selling_price": 50.0,
    "discount_percent": 10.0
  },
  {
    "item_id": 16,
    "item_name": "Biscuit",
    "ingredients": "Milk, Eggs, Cream",
    "ingredient_ids": "1, 4, 5",
    "ingredient_quantity": "10, 20, 30",
    "item_quantity": 30,
    "cost_price": 200.0,
    "selling_price": 300.0,
    "discount_percent": null
  },
  {
    "item_id": 17,
    "item_name": "Mango custard",
    "ingredients": "Cream",
    "ingredient_ids": "5",
```

```
            "ingredient_quantity": "40",
            "item_quantity": 60,
            "cost_price": 30.0,
            "selling_price": 40.0,
            "discount_percent": null
        }
    ],
    "count": "0",
    "message": "List of all available products in the bakery."
}
```


c. **placeAnOrderAndGetBill-** It allows a user to make an order by entering the values like Full_name, Mobil_no, Order_items, Order_quantities, Pickup_order_date, Payment_method and Payment_paid. Of all these values only the Order_items and Order_quantities take either single values or multiple values (separated by ',') as input. There is one more field which is Pickup_order_time which is optional, but if the user enters it then it will also be inserted in the database otherwise it will be left blank. This API uses the "bakery_orders" table for its functioning. The output of this API is a generated bill. If the user pays less amount then the total amount of his orders then a fail response will be shown.

Input:


```
{
    "Full_name": "Rohit Chaurasia",
    "Mobile_no": "8756436112",
    "Order_items": "Biscuit, Pastry",
    "Order_quantities": "2, 10",
    "Pickup_Order_Date": "1/07/2021",
    "Payment_method": "Cash",
    "Payment_paid": 5000
```

}

Output:

{
  "status": "success",
  "status_code": "200",
  "data": {
    "Order_id": 13,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit, Pastry",
    "order_quantities": "2, 10",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 5000,
    "total_price": 1100,
    "return_amount": 3900,
    "order_date": "2021-07-01",
    "order_time": "16:58:27.956513",
    "Country_code": "+91"
  },
  "count": "0",
  "message": "Order Successful!! THANK YOU for shopping from our bakery!! Here is your bill."
}

***It shows the generated bill**

Other Input:

{
  "Full_name": "Rohit Chaurasia",
  "Mobile_no": "8756436112",
  "Order_items": "Biscuit, Pastry",
  "Order_quantities": "2, 10",
  "Pickup_Order_Date": "1/07/2021",
  "Payment_method": "Cash",

```
  "Payment_paid": 200
}
```


Output:


```
{
  "status": "fail",
  "status_code": "400",
  "data": "",
  "count": "0",
  "message": "You have paid less amount than the total amount of your order!"
}
```


***When the amount paid is less than the total amount of his/her order.**

d. **order_history-** This API shows the list of all the orders made by customer since his first order. It uses the “bakery_orders” table for its working. It takes full name and mobile number of the users.

Input:


```
{
  "Full_name": "Rohit Chaurasia",
  "Mobile_no": "8756436112"
}
```


Output:


```
{
  "status": "success",
  "status_code": "200",
  "data": [
    {
      "Order_id": 6,
      "full_name": "Rohit Chaurasia",
      "mobile_no": 8756436112,
      "order_items": "Pastry",
      "order_quantities": "20",
      "pickup_order_date": "1/07/2021",
```

```
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 1500,
    "total_price": 1000,
    "return_amount": 500,
    "order_date": "2021-07-01",
    "order_time": "13:38:51.233843",
    "Country_code": "+91"
  },
  {
    "Order_id": 7,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit",
    "order_quantities": "3",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 2500,
    "total_price": 900,
    "return_amount": 1600,
    "order_date": "2021-07-01",
    "order_time": "13:40:09.554647",
    "Country_code": "+91"
  },
  {
    "Order_id": 8,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Pastry, Biscuit",
    "order_quantities": "3, 2",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 6000,
    "total_price": 750,
    "return_amount": 5250,
```

```
    "order_date": "2021-07-01",
    "order_time": "13:48:53.376832",
    "Country_code": "+91"
  },
  {
    "Order_id": 9,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Pastry",
    "order_quantities": "10",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 6000,
    "total_price": 500,
    "return_amount": 5500,
    "order_date": "2021-07-01",
    "order_time": "13:54:19.343614",
    "Country_code": "+91"
  },
  {
    "Order_id": 10,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit, Pastry",
    "order_quantities": "2, 10",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 5000,
    "total_price": 1100,
    "return_amount": 3900,
    "order_date": "2021-07-01",
    "order_time": "13:59:41.484571",
    "Country_code": "+91"
  },
  {
```

```
    "Order_id": 11,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit, Pastry",
    "order_quantities": "2, 10",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 5000,
    "total_price": 1100,
    "return_amount": 3900,
    "order_date": "2021-07-01",
    "order_time": "16:55:50.510922",
    "Country_code": "+91"
  },
  {
    "Order_id": 12,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit, Pastry",
    "order_quantities": "2, 10",
    "pickup_order_date": "1/07/2021",
    "pickup_order_time": "",
    "payment_method": "Cash",
    "payment_made": 5000,
    "total_price": 1100,
    "return_amount": 3900,
    "order_date": "2021-07-01",
    "order_time": "16:57:31.231486",
    "Country_code": "+91"
  },
  {
    "Order_id": 13,
    "full_name": "Rohit Chaurasia",
    "mobile_no": 8756436112,
    "order_items": "Biscuit, Pastry",
    "order_quantities": "2, 10",
```

```
            "pickup_order_date": "1/07/2021",
            "pickup_order_time": "",
            "payment_method": "Cash",
            "payment_made": 5000,
            "total_price": 1100,
            "return_amount": 3900,
            "order_date": "2021-07-01",
            "order_time": "16:58:27.956513",
            "Country_code": "+91"
        }
    ],
    "count": "0",
    "message": "Here is a list of all your orders."
}
```