

**School of Computer Science and Engineering**

DBMS Course Project Report

on

**Modular Kitchen Database Management System**

Submitted by

**Ms. Mukti Bhansali 01fe19bcs203**

**Ms. Mehar Anjum 01fe19bcs208**

**Ms. Manisha Belagal 01fe19bcs207**

**Ms. Akashini Koppad 01fe19bcs210**

Contents

[INTRODUCTION: 3](#_Toc75129136)

[PROPOSAL PHASE: 4](#_Toc75129137)

[DESIGN PHASE 16](#_Toc75129138)

[IMPLEMENTATION PHASE 33](#_Toc75129139)

[USER MANUAL 70](#_Toc75129140)

[SNAPSHOTS 72](#_Toc75129141)

[ACCEPTANCE LETTER 73](#_Toc75129142)

# INTRODUCTION:

Our client, Dinakar Shetty owns the Sanjivini Home solutions, which a shop that has modular kitchen items and also designs a full kitchen. They store the data of customers in an excel sheet or registers and the details of the products, accessories that are available in their shop is stored in paper. This way of storing the important data is risky. Any person can easily access this data, which is a sensitive information to the shop and can misuse it. As the information is stored in paper, it can be misplaced and lost very easily. Accessing the details of a particular customer, or date or the products that are currently available in the shop is very difficult task as the data is huge and a person has to go through each and every record to access the information needed. Hence, this way of storing the data is prone to more risk and is not very efficient. Our approach to solve this problem was to create a database for the shop in which they can store the customer data, the products and accessories that are available in their shop and they can also easily be able to generate bills and receipt without actually manually writing the data. To start off, the admin is provided with a page that has a user login page at the beginning. This is to not allow others to access the information available in the database. Once the admin has logged in, a homepage appears that gives the count of the customers that have purchased something today, the count of full kitchen and retail customers available and also a graph displaying the number of products sold. There is also a customer page, a products page, accessories page where you can add data, delete data and search data easily. The invoice generates a pdf which can be downloaded and given to the customer. All of this, is implemented using Django, in which the customer, products, accessories, designer are all taken as models and queries are written to add, delete and search records. Queries are also written to count the number of customers that have shopped on that particular day, the number of products sold, the number of full kitchen and retail customer, and the total number of orders, that are displayed at the homepage for easy and quick access of the admin. In this way, it is easier for the client to maintain the data, accessing the data and deleting the data. The security for the information of the shop is also increased, as to access the pages, a username and password has to be provided. This way, the data is not prone to getting lost that easily and maintaining is easier.

# PROPOSAL PHASE:

**Responsibilities:** Mukti Bhansali - ER diagram, client visit, relation schema.

Manisha Belagal - ER diagram, writing and editing word document.

Mehar Anjum - ER diagram, collection of data from client.

Akashini Koppad -ER diagram, writing and editing word document.

**Problem Description:** The Sanjivini Home Solutions Shop stores the details of customers and services in a conventional paper work. This way of storing is a tedious and time-consuming job. The goal is to have a database management system that would reduce their work of retrieving and managing the data.

**Requirements:** 1. Admin can access product details.

2. Admin can add new product.

3. Admin can delete existing product.

4. Admin can modify existing product.

5. Admin can access customer details.

6. Admin can add new customer.

7. To generate bills and receipts.

8. Admin can search for a particular customer, product and accessories.

**Design Questions to be answered**

**Question 1:** From the problem description, identify the entities that need to be represented in the database, the attributes of each entity, the relationships between the entities, and the cardinality ratios of each relationship.

The entities involved are:

Customer, Order, Products, Accessories, FK products, Payment, Payment\_ins, staff

The attributes of the entities are as follows,

Customer: cust\_id, c\_name(f\_name, m\_name, l\_name), walk in date, site address(street, city, pincode), type, ph.no.

Order: orderid, p\_hsn, p\_modelname, p\_mrp

Products: model\_name, type, mrp, gst

Accessories: code, mrp, dealer price, dimensions, HSN, std pack, Name, gst

FK products: HSN code, product category, gst

Staff: s\_id, ph.no, salary, designation, s\_name(f\_name, m\_name, l\_name), current status

Payment: pay\_id, total amount, balance

Payment\_ins: r\_id, date, mode of payment, cheque/acc no, installment

The relationship between the entities:

Customer -orders - Products. (1 :N )

Order -has- Products. (M :N )

Order -has- Accessories. ( M:N )

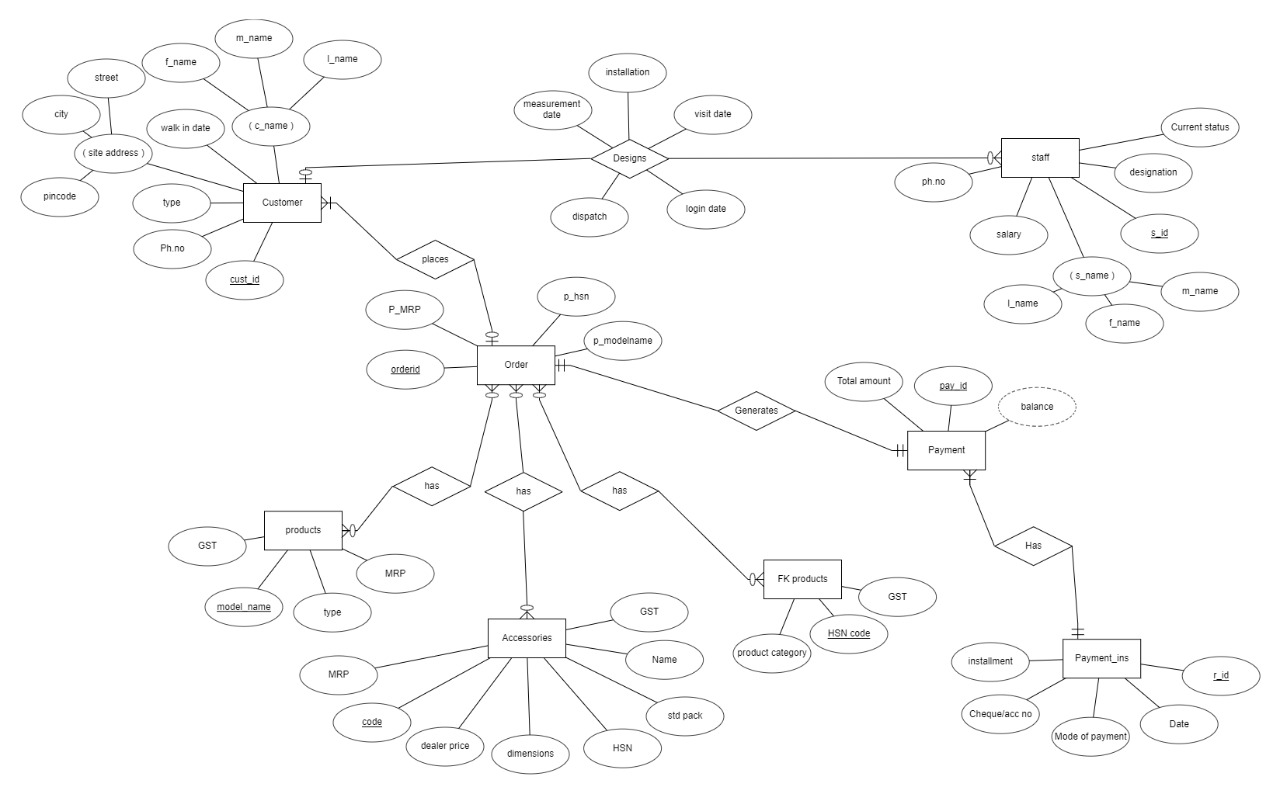
Order -has -FK products. (M:N )

Order-generates- payment. (1:1 )

Payment -has- Payment\_ins. (1:N)

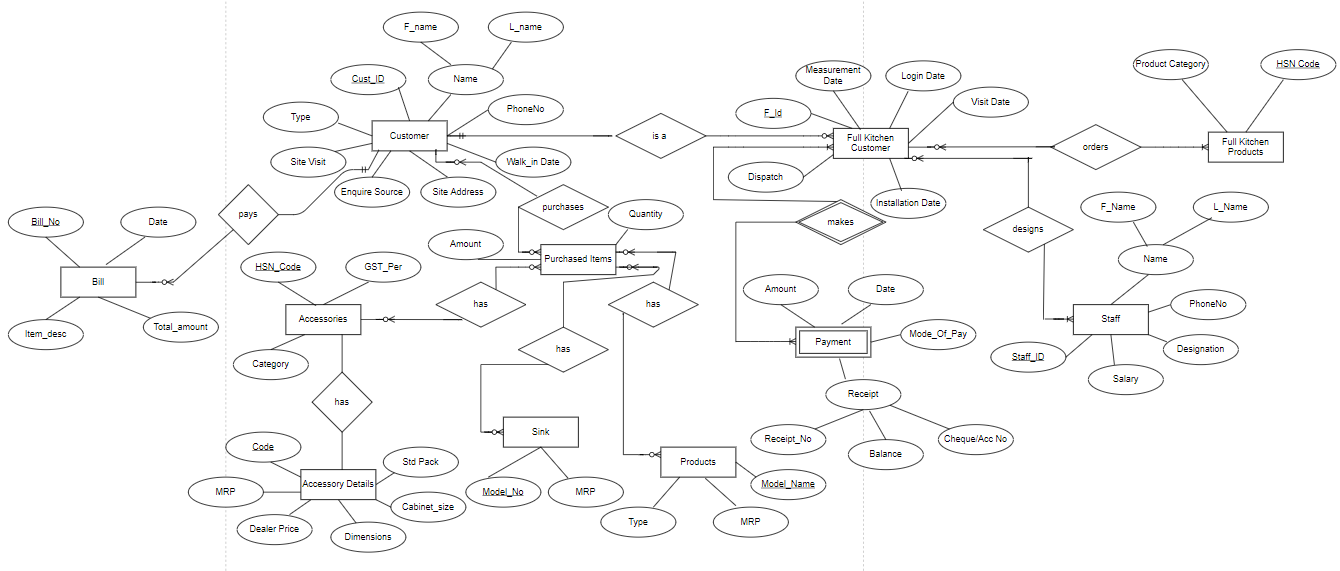
Staff -designs- Customer. (1 :N )

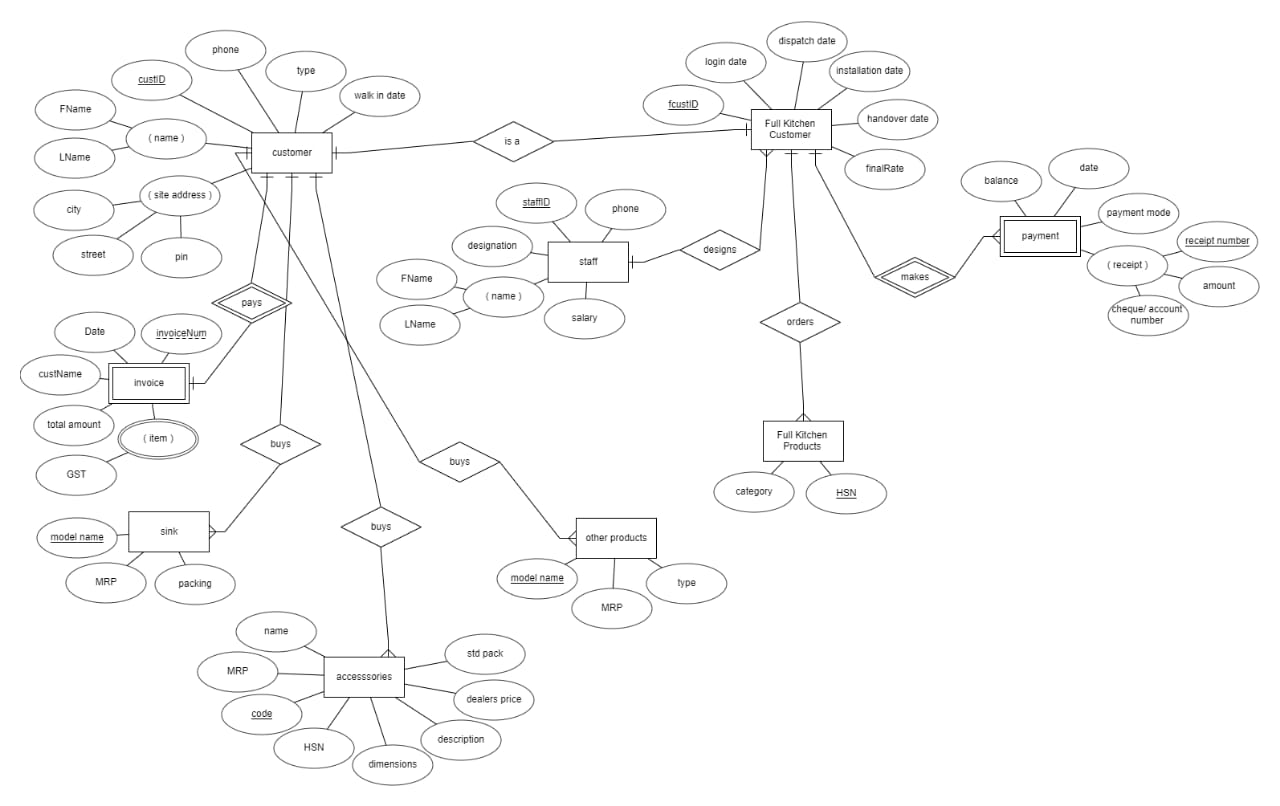
**Question 2:** Draw an Entity-Relationship Diagram illustrating the information you have identified in Question 1.



**Question 3:** Draw **alternate** Entity-Relationship Diagram illustrating the information you have identified in Question 1 that you think are most likely to occur.

First Alternative:

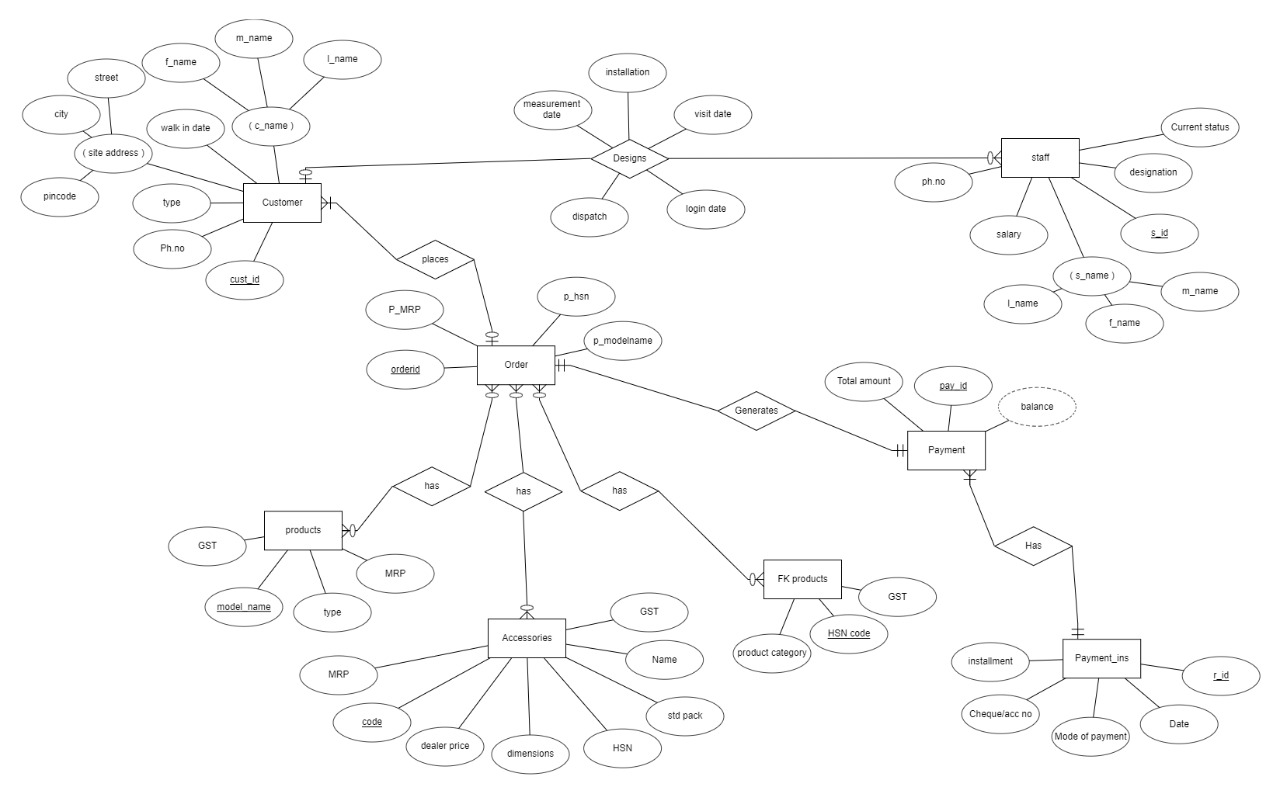


Second alternative:

Third Alternative:



**Question 4:** Choose the **optimal** Entity-Relationship Diagram from the designs provided above and justify why you think this is an optimal solution for your identified problem specification.



The ER diagram takes into consideration all the attributes which are required to design the database. The entities are also grouped efficiently. The relationships between the various entities are also legitimate.

**Question 6:** Draw an ER to Relation Mapping illustrating the information you have identified in Question 4.

****

**Question 7:** Draw a Data Dictionary illustrating the information you have identified in Question 6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object (Entity)** | **Name (Attribute)** | **Type**  **(Data type)** | **Description** | **Primary Key** | **Foreign Key** |
| Customer | cust\_id | Number(5) | Unique Identification number for the customer | Yes | No |
| Customer | f\_name | Varchar2(10) | First name of the customer | No | No |
| Customer | m\_name | Varchar2(10) | Middle name of the customer | No | No |
| Customer | l\_name | Varchar2(10) | Last name of the customer | No | No |
| Customer | Ph\_no | Number(10) | Phone number of the customer | No | No |
| Customer | Walk in date | Date | Date when the customer goes to the shop | No | No |
| Customer | type | Varchar2(10) | Retail customer or Full-kitchen customer | No | No |
| Customer | street | Varchar2(20) | Site address of the customer | No | No |
| Customer | city | Varchar2(10) | City of the customer | No | No |
| Customer | pincode | Number(6) | Pincode of the customer | No | No |
| Customer | measurement Date | Date | Date when the kitchen measurements are taken | No | No |
| Customer | login Date | Date |  | No | No |
| Customer | visit Date | Date | Date when site was visited | No | No |
| Customer | Installation date | Date | Date when installation was made | No | No |
| Customer | dispatch | Varchar2(3) | To know if the product is dispatched or not | No | No |
| Customer | s\_id | Number(5) | Unique Identification of Staff | No | Yes |
| FK products | HSN code | Number(5) | HSN Code of Full Kitchen Products | Yes | No |
| FK products | product category | Varchar2(10) | Category to which the product belongs | No | No |
| FK products | gst | Number(2) | GST of the FK product | No | No |
| Staff | s\_id | Number(5) | Unique Identification of Staff | Yes | No |
| Staff | f\_name | Varchar2(10) | First Name of the staff | No | No |
| Staff | l\_name | Varchar2(10) | Last Name of the Staff | No | No |
| Staff | ph.no | Number(10) | Phone number of the staff | No | No |
| Staff | designation | Varchar2(10) | Job Designation of the staff | No | No |
| Staff | salary | Number(6) | Salary of the staff | No | No |
| Staff | current status | Varchar2(2) | If the staff is still working in the shop or not | No | No |
| Payment | p\_id | Number(5) | Unique Identification of Payment | Yes | No |
| Payment | Total amount | Number(10) | Total amount to be paid | No | No |
| Payment\_ins | R\_id | Number(5) | Unique receipt ID | Yes | No |
| Payment\_ins | Date | Date | Date of Payment | No | No |
| Payment\_ins | Mode of payment | Varchar2(10) | Mode of Payment done by Customer | No | No |
| Payment\_ins | installment | Number(4) | Installment made by the customer | No | No |
| Payment\_ins | pay\_id | Number(5) | Unique Identification of Payment | No | Yes |
| Payment\_ins | check/accNo | Number(10) | Check/Acc Number of Customer | No | No |
| Order | order\_id | Number(5) | Unique identification number for order | Yes | No |
| Order | p\_modelname | Varchar2(10) | Name of the product model | Yes | No |
| Order | p\_mrp | Number(4) | Price of the product model | No | No |
| Order | p\_hsn | Number(5) | HSN Code of Products | No | No |
| Order | p\_id | Number(5) | Unique Identification number for the customer | No | Yes |
| Order | cust\_id | Number(5) | Unique Identification number for order | No | Yes |
| Accessories | code | Number(10) | Unique Identification of Accessories | Yes | No |
| Accessories | Name | Varchar2(10) | Name of the Accessory | No | No |
| Accessories | MRP | Number(4) | Price of the Accessory | No | No |
| Accessories | dealer price | Number(5) | Price set by the dealer | No | No |
| Accessories | dimensions | Number(10) | Dimensions of the accessory | No | No |
| Accessories | std pack | Varchar2(10) | Standard pack of the accessory | No | No |
| Accessories | gst | Number(2) | Gst set for the accessory | No | Yes |
| Products | Modelname | Varchar2(10) | Unique model name of the product | Yes | No |
| Products | mrp | Number(10) | Price of the product purchased | No | No |
| Products | type | Varchar2(10) | Type of the product | No | No |
| Product | gst | Number(2) | GST of the product | No | No |

# DESIGN PHASE

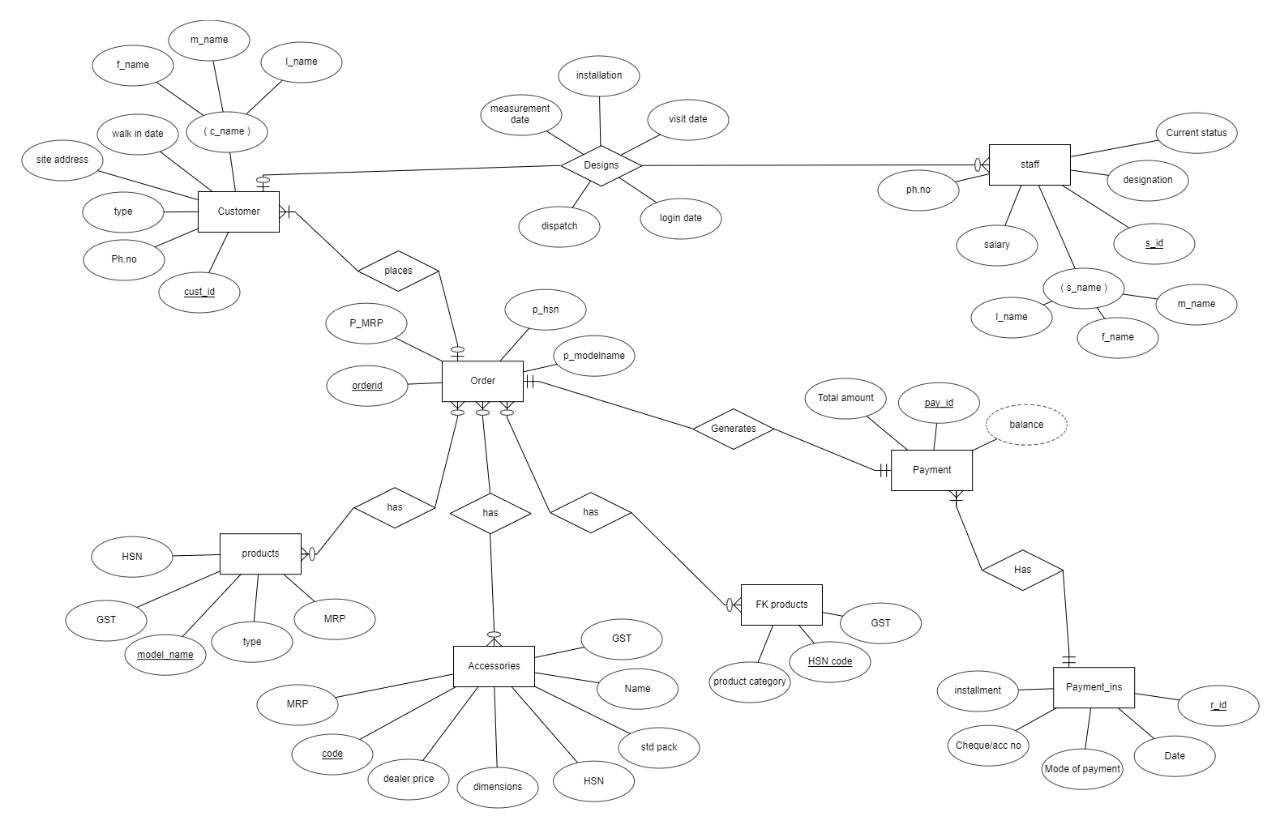
**Responsibilities:** Mukti Bhansali – Designing User Interface

Manisha Belagal – Normalisation.

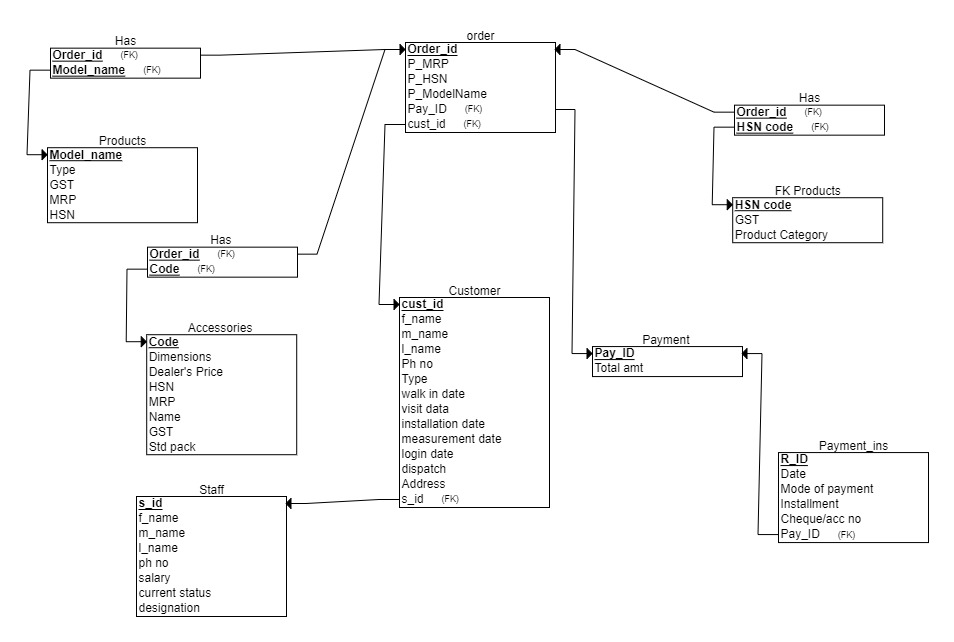
Mehar Anjum – Normalisation.

Akashini Koppad – Normalisation.

**ER Design:**



**ER to Relation Mapping:**



**Data Dictionary**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object (Entity)** | **Name (Attribute)** | **Type**  **(Data type)** | **Description** | **Primary Key** | **Foreign Key** |
| Customer | cust\_id | Number(5) | Unique Identification number for the customer | Yes | No |
| Customer | f\_name | Varchar2(10) | First name of the customer | No | No |
| Customer | m\_name | Varchar2(10) | Middle name of the customer | No | No |
| Customer | l\_name | Varchar2(10) | Last name of the customer | No | No |
| Customer | ph\_no | Number(10) | Phone number of the customer | No | No |
| Customer | walk in date | Date | Date when the customer goes to the shop | No | No |
| Customer | type | Varchar2(10) | Retail customer or Full-kitchen customer | No | No |
| Customer | address | Varchar2(20) | Address of the customer | No | No |
| Customer | measurement date | Date | Date when the kitchen measurements are taken | No | No |
| Customer | login date | Date |  | No | No |
| Customer | visit date | Date | Date when site was visited | No | No |
| Customer | installation date | Date | Date when installation was made | No | No |
| Customer | dispatch | Varchar2(3) | To know if the product is dispatched or not | No | No |
| Customer | s\_id | Number(5) | Unique Identification of Staff | No | Yes |
| FK products | HSN code | Number(5) | HSN Code of Full Kitchen Products | Yes | No |
| FK products | product category | Varchar2(10) | Category to which the product belongs | No | No |
| FK products | gst | Number(2) | GST of the FK product | No | No |
| Staff | s\_id | Number(5) | Unique Identification of Staff | Yes | No |
| Staff | f\_name | Varchar2(10) | First Name of the staff | No | No |
| Staff | l\_name | Varchar2(10) | Last Name of the Staff | No | No |
| Staff | ph.no | Number(10) | Phone number of the staff | No | No |
| Staff | designation | Varchar2(10) | Job Designation of the staff | No | No |
| Staff | salary | Number(6) | Salary of the staff | No | No |
| Staff | current status | Varchar2(2) | If the staff is still working in the shop or not | No | No |
| Payment | p\_id | Number(5) | Unique Identification of Payment | Yes | No |
| Payment | total amount | Number(10) | Total amount to be paid | No | No |
| Payment\_ins | r\_id | Number(5) | Unique receipt ID | Yes | No |
| Payment\_ins | date | Date | Date of Payment | No | No |
| Payment\_ins | mode of payment | Varchar2(10) | Mode of Payment done by Customer | No | No |
| Payment\_ins | installment | Number(4) | Installment made by the customer | No | No |
| Payment\_ins | pay\_id | Number(5) | Unique Identification of Payment | No | Yes |
| Payment\_ins | check/accNo | Number(10) | Check/Acc Number of Customer | No | No |
| Order | order\_id | Number(5) | Unique identification number for order | Yes | No |
| Order | p\_modelname | Varchar2(10) | Name of the product model | Yes | No |
| Order | p\_mrp | Number(4) | Price of the product model | No | No |
| Order | p\_hsn | Number(5) | HSN Code of Products | No | No |
| Order | p\_id | Number(5) | Unique Identification number for the customer | No | Yes |
| Order | cust\_id | Number(5) | Unique Identification number for order | No | Yes |
| Accessories | code | Number(10) | Unique Identification of Accessories | Yes | No |
| Accessories | name | Varchar2(10) | Name of the Accessory | No | No |
| Accessories | MRP | Number(4) | Price of the Accessory | No | No |
| Accessories | dealer price | Number(5) | Price set by the dealer | No | No |
| Accessories | dimensions | Number(10) | Dimensions of the accessory | No | No |
| Accessories | std pack | Varchar2(10) | Standard pack of the accessory | No | No |
| Accessories | gst | Number(2) | Gst set for the accessory | No | Yes |
| Products | modelname | Varchar2(10) | Unique model name of the product | Yes | No |
| Products | mrp | Number(10) | Price of the product purchased | No | No |
| Products | hsn | Number(5) | HSN Code of Products | No | No |
| Products | type | Varchar2(10) | Type of the product | No | No |
| Product | gst | Number(2) | GST of the product | No | No |

**Question 1: Normalization:**

Normalized Solution 1:

The relation Customer has the following FD

cust\_id -> f\_name, m\_name, l\_name, ph\_no, walk in date, address

type -> visit date, measurement date, login date, dispatch, installment date

{cust\_id, type} -> s\_id

The relation is found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

The relation Order, was found to not be in 1NF due to multiple values of products ordered by the customers.

To solve this, the relation was split to two tables, one containing the order\_id, cust\_id and pay\_id, and the other having details of ordered products by each customer.

For the relation Order, order\_id -> pay\_id, cust\_id

For the relation Order\_details, {order\_id, p\_hsn, p\_modelname, p\_mrp} -> {order\_id, p\_hsn, p\_modelname, p\_mrp}

The relations Order and Order\_details are separately found to be in 1NF and 2NF. There is no transitive dependency in any relations, hence they are not in 3NF.

The relation, Products was found to be in 1NF, 2NF but not in 3NF as transitive dependencies were discovered.

model\_name -> mrp ,type, gst, hsn

type -> gst, hsn

The decomposition led to two relations

For Products relation, model\_name -> mrp, type

For Products\_details relation, type-> gst, hsn

The two relations were separately found to be in 1NF, 2NF, 3NF and BCNF.

The relation, Accessories was found to be in 1NF, 2NF but not in 3NF as transitive dependencies were discovered.

code-> hsn, name, gst, dimensions, std pack, mrp, dealer’s price

hsn -> name, gst

The decomposition led to two relations

For Accessories relation, code->hsn, dimensions, std pack, mrp, dealer’s price

For Acc\_category relation, hsn -> name, gst

The two relations were separately found to be in 1NF, 2NF, 3NF and BCNF.

The relation, Staff was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

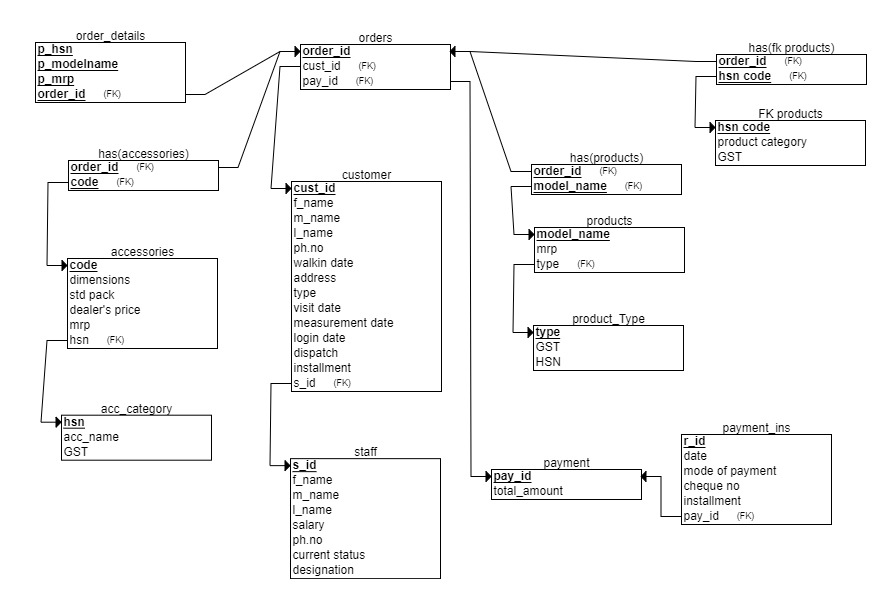
s\_id -> f\_name, m\_name, l\_name, salary, ph no, current status, designation

The relation, Payment was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

pay\_id -> total amount

The relation, Payment\_ins was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

r\_id -> installment, date, mode of payment, cheque/acc no, pay\_id



Normalized Solution 2:

The relation customer was found to be in 1NF but not in 2NF.

{cust\_id, type} ->f\_name, m\_name, l\_name, ph\_no, walk-in date, address, s\_id, visit date, installment date, measurement date, login date, dispatch.

cust\_id -> f\_name, m\_name, l\_name, ph\_no, walk-in date, address

If type is removed, cust\_id alone can determine few attributes. Hence no Full FD.

After decompositions, we have two relations.

For Customer relation, {cust\_id, type} -> f\_name, m\_name, l\_name, ph\_no, walk-in date, address

For Full Kitchen Customer relation, {cust\_id, type, s\_id } -> visit date, installation date, measurement date, login date, dispatch.

These two relations are separately found to be in 2NF.

There is no transitive dependency, hence the relations are not in 3NF.

The relation Order, was found to not be in 1NF due to multiple values of products ordered by the customers.

To solve this, the relation was split to two tables, one containing the order\_id, cust\_id and pay\_id, and the other having details of ordered products by each customer.

For the relation Order, order\_id -> pay\_id, cust\_id

For the relation Order\_details, {order\_id, p\_hsn, p\_modelname, p\_mrp} -> {order\_id, p\_hsn, p\_modelname, p\_mrp}

The relations Order and Order\_details are separately found to be in 1NF and 2NF. There is no transitive dependency in any relations, hence they are not in 3NF.

The relation, Products was found to be in 1NF, 2NF but not in 3NF as transitive dependencies were discovered.

model\_name -> mrp ,type, gst, hsn

type -> gst, hsn

The decomposition led to two relations

For Products relation, model\_name -> mrp, type

For Products\_details relation, type-> gst, hsn

The two relations were separately found to be in 1NF, 2NF, 3NF and BCNF.

The relation, Accessories was found to be in 1NF, 2NF but not in 3NF as transitive dependencies were discovered.

code-> hsn, name, gst, dimensions, std pack, mrp, dealer’s price

hsn -> name, gst

The decomposition led to two relations

For Accessories relation, code->hsn, dimensions, std pack, mrp, dealer’s price

For Acc\_category relation, hsn -> name, gst

The two relations were separately found to be in 1NF, 2NF, 3NF and BCNF.

The relation, Staff was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

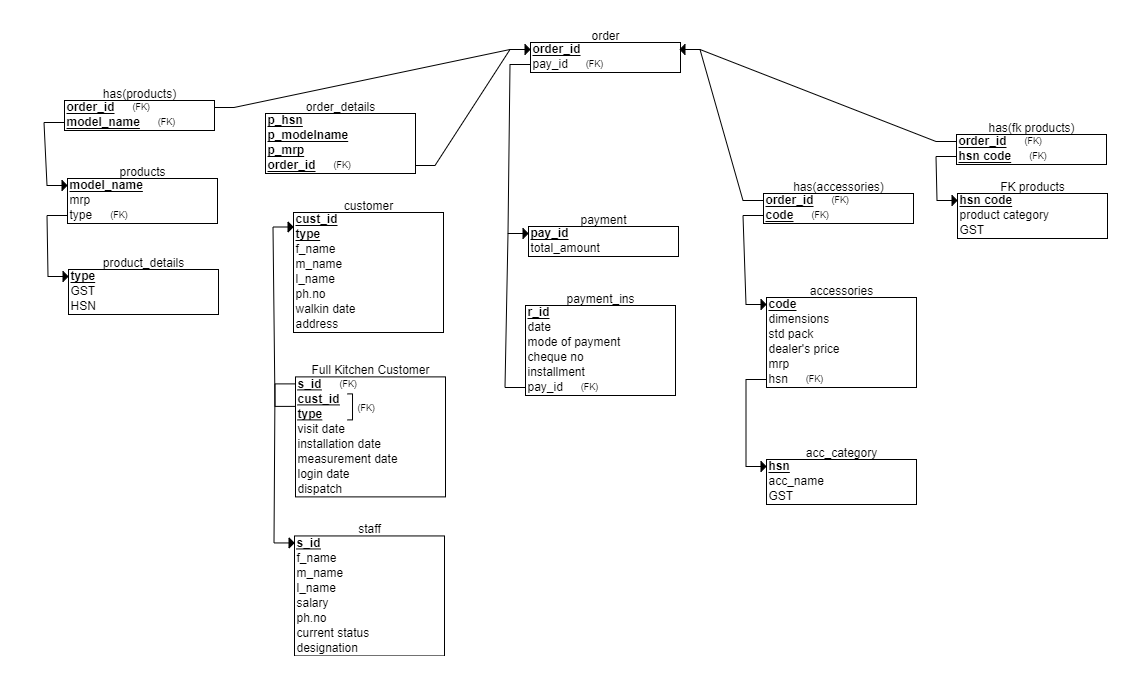
s\_id -> f\_name, m\_name, l\_name, salary, ph no, current status, designation

The relation, Payment was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

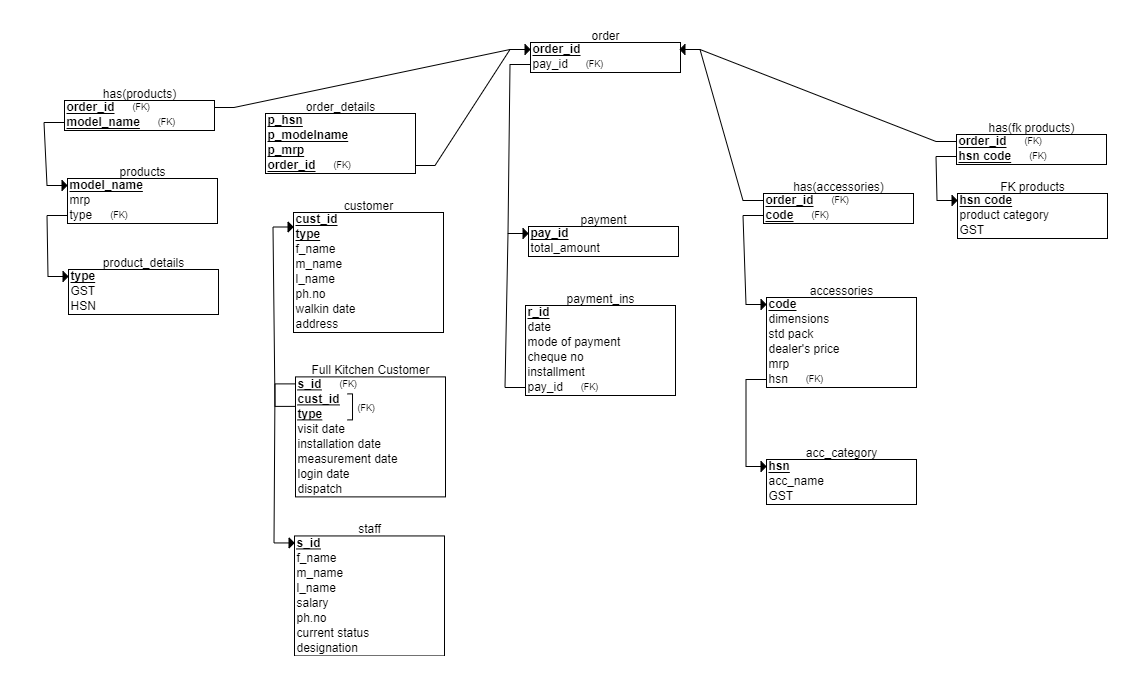
pay\_id -> total amount

The relation, Payment\_ins was found to be in 1NF, 2NF but not in 3NF as there are no transitive dependencies.

r\_id -> installment, date, mode of payment, cheque/acc no, pay\_id

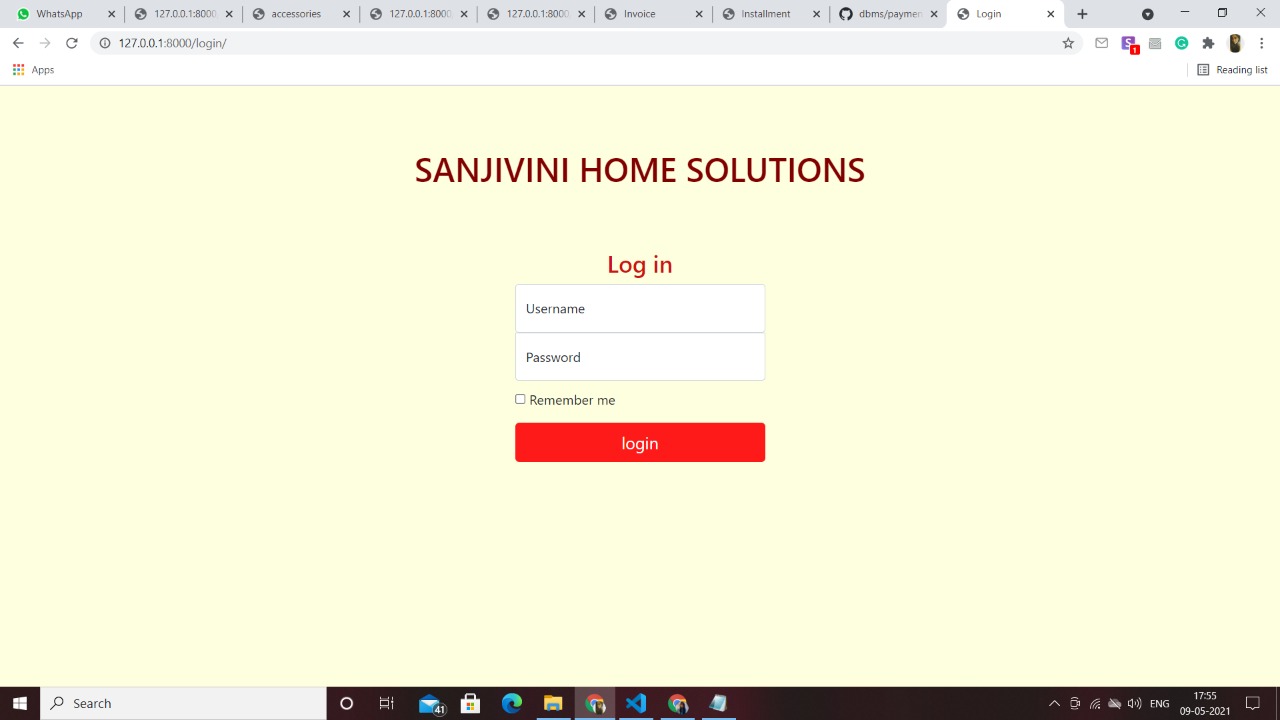


**Question 2:** The optimal schema chosen is the second normalized solution, as the redundant data in the customers table is removed by making a separate table for the attributes for Full Kitchen Customers only.

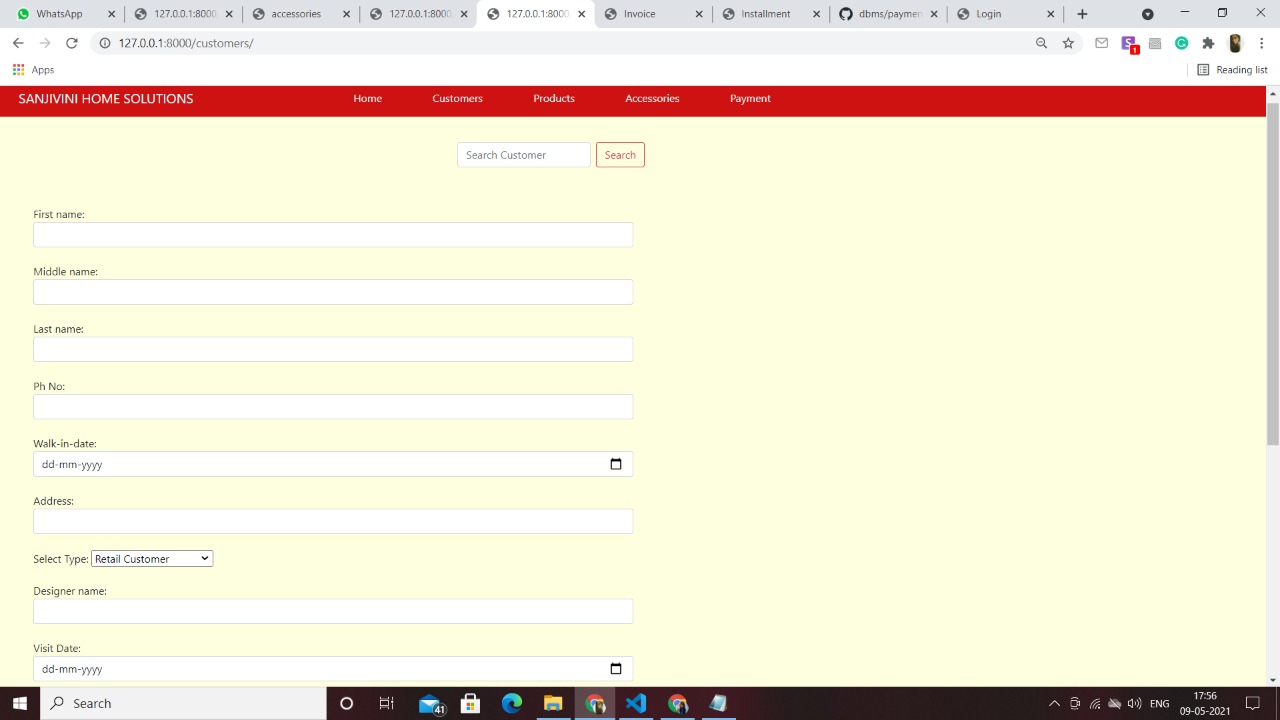


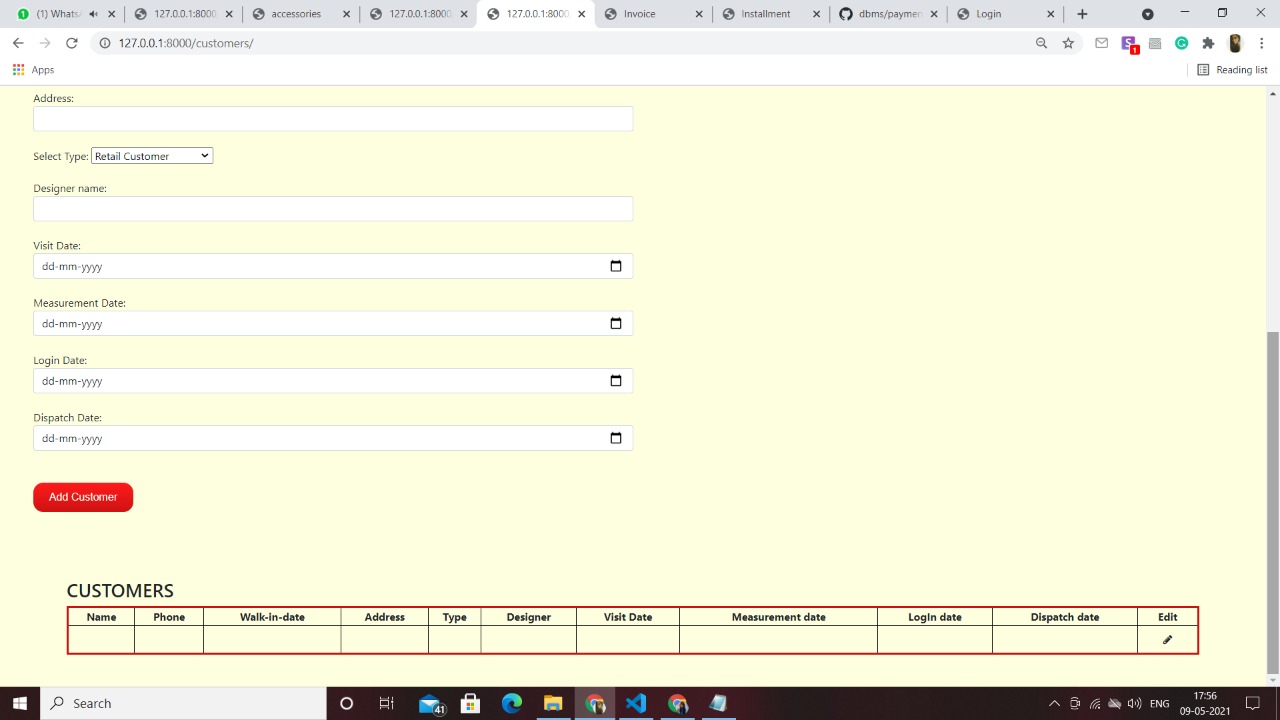
**Question 3:** User Interface (UI) design

Login Page:

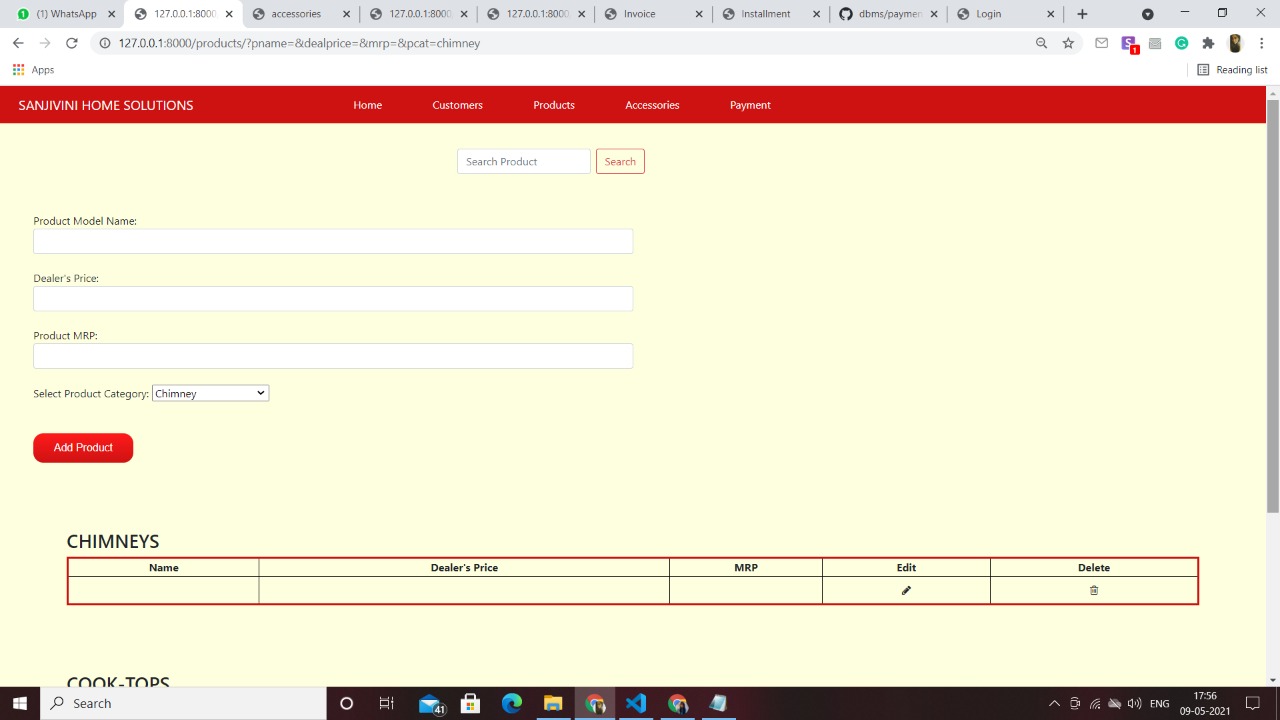


Customer Page:

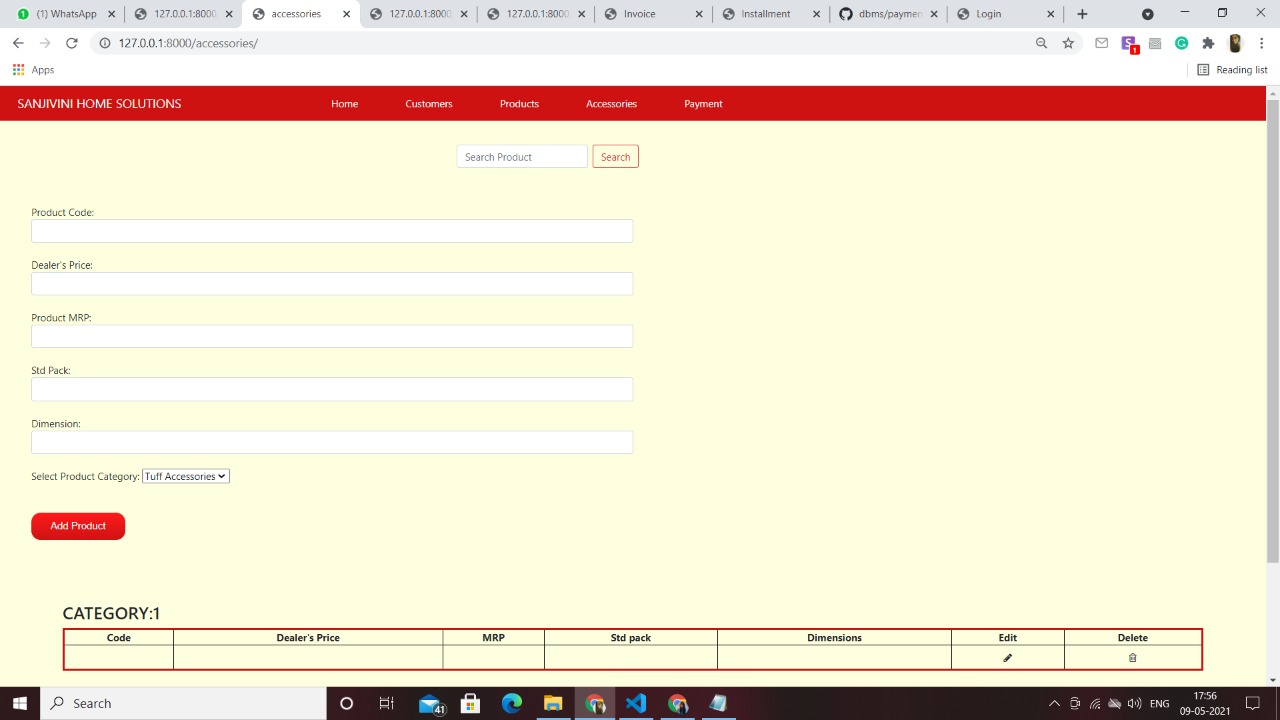




Products Page:



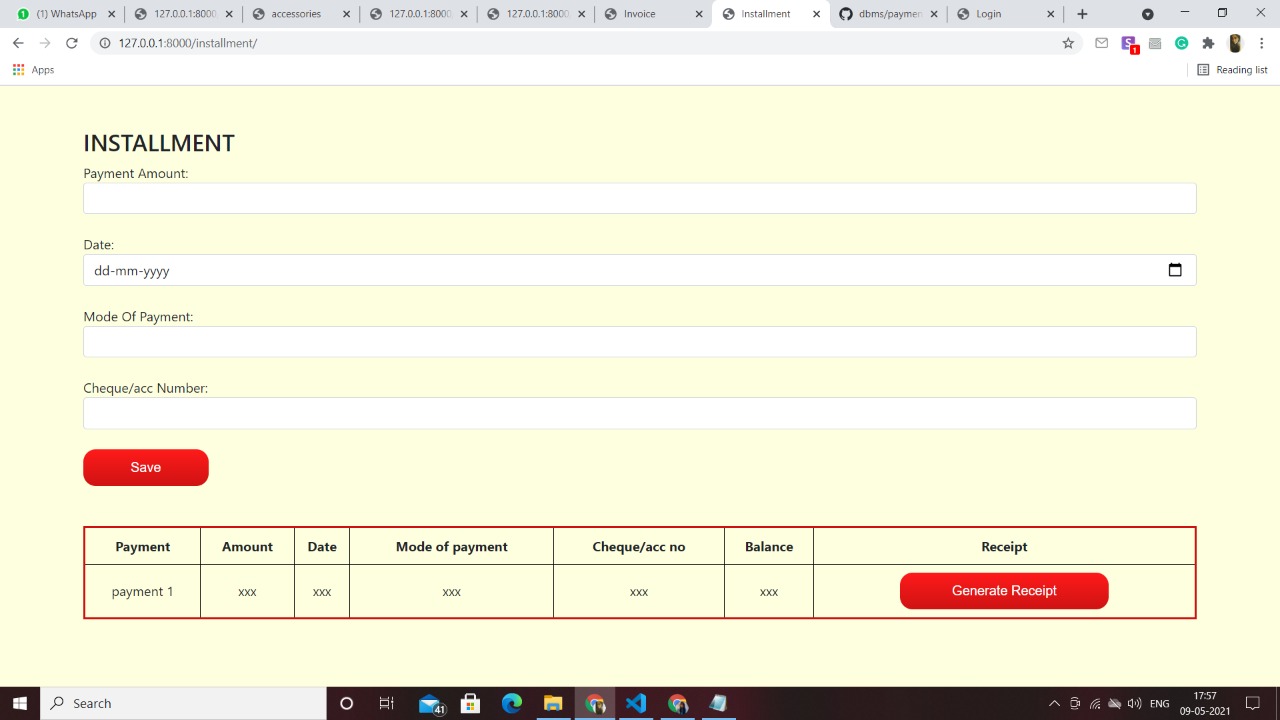
Accessories Page:



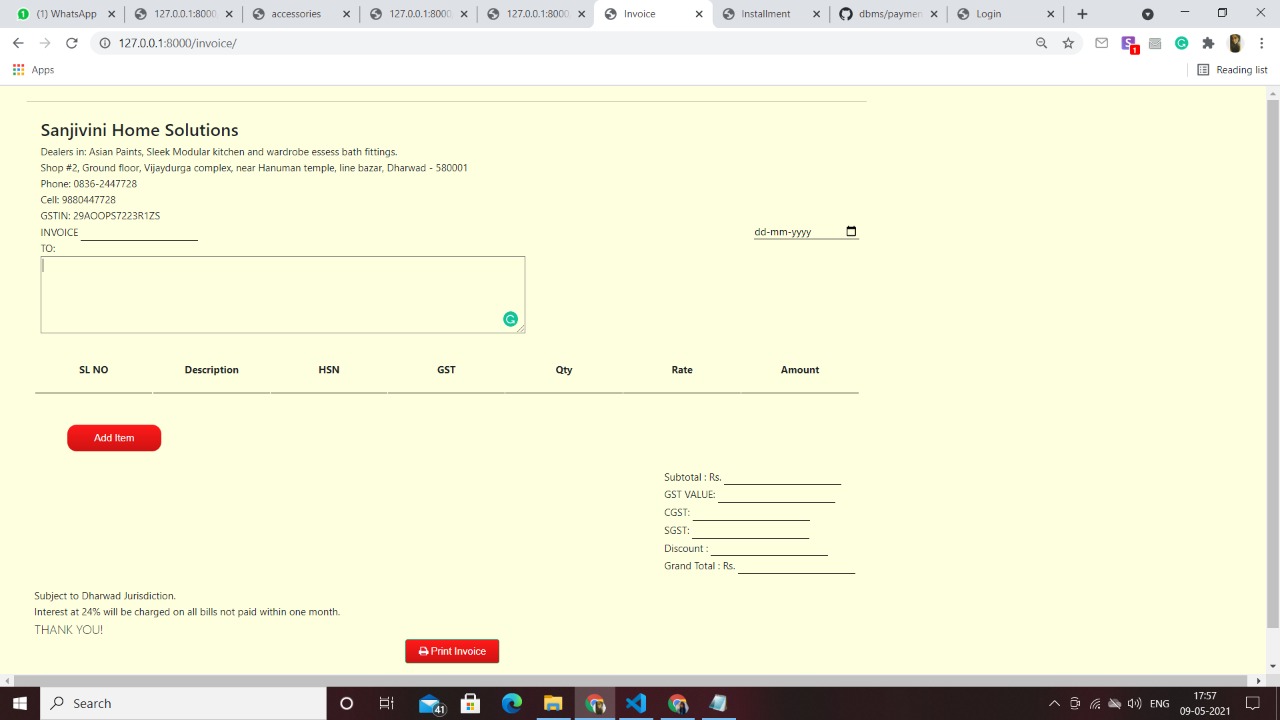
Payment Page:



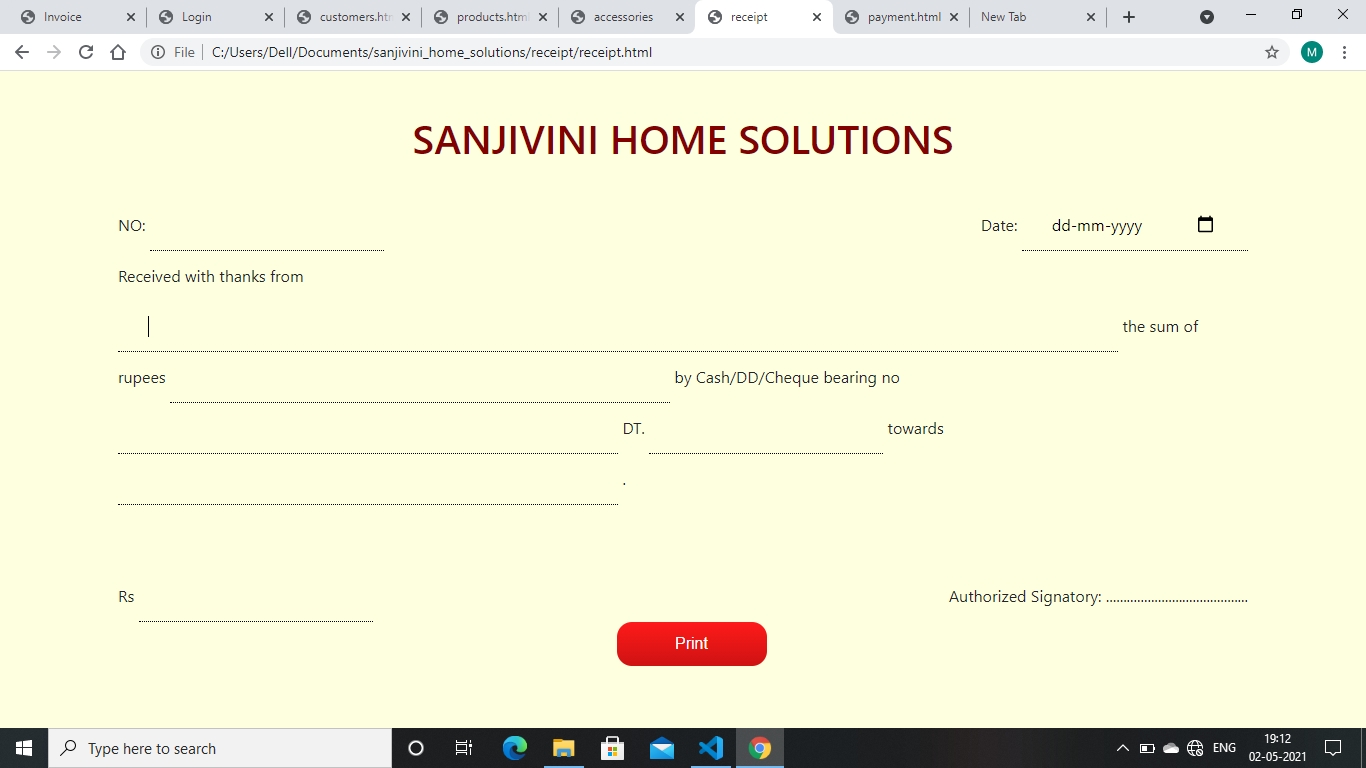
Payment Installment Page:



Invoice Page:



Receipt Page:



# IMPLEMENTATION PHASE

**Responsibilities:**

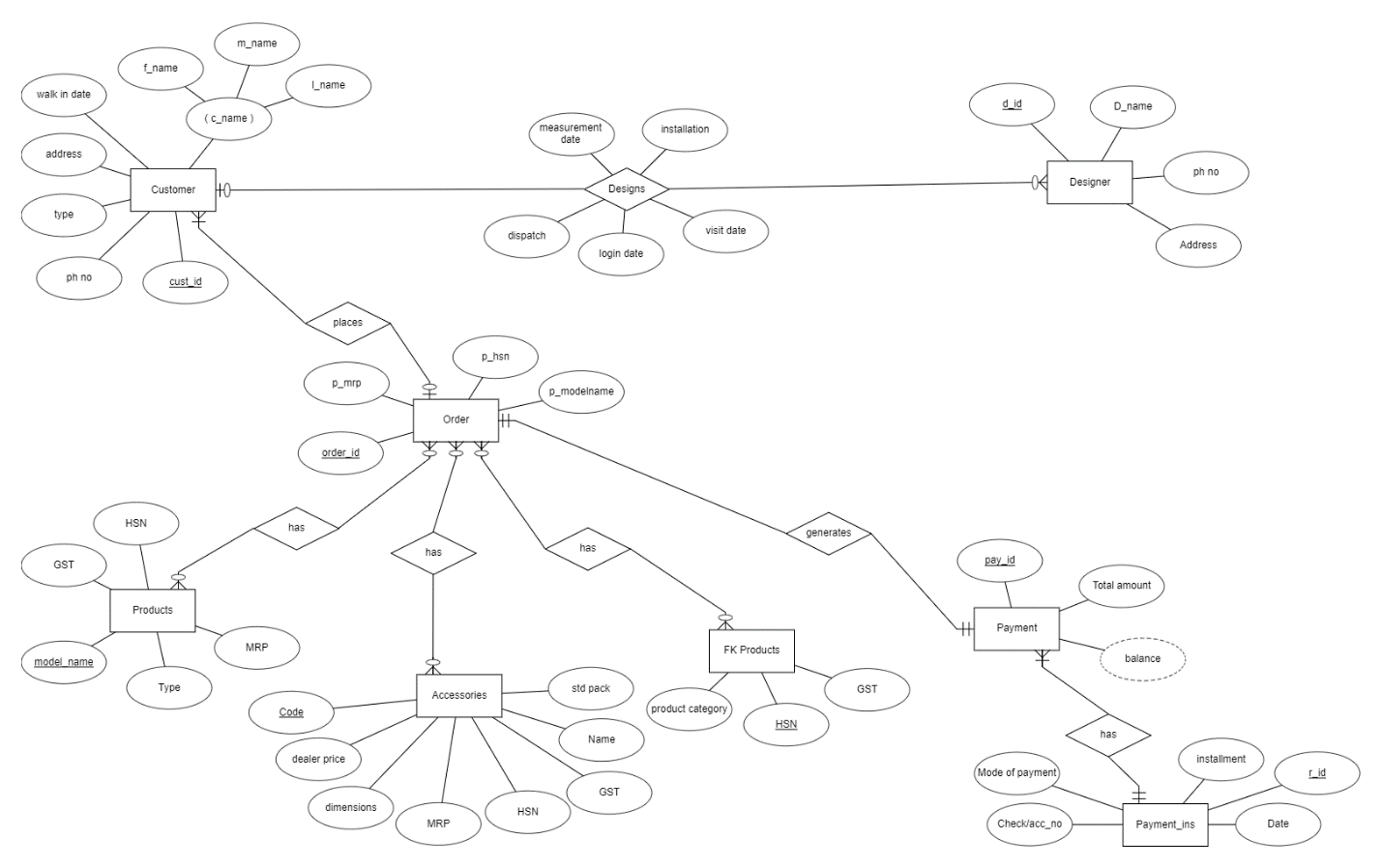
Mukti Bhansali – Backend functionalities of accessories and products which are search, add, delete and edit. Worked on orders and installment page.

Manisha Belagal – Login page and converting invoice and receipts to pdf. Worked on orders and payment page.

Mehar Anjum – Backend of customer functionalities which are add, edit and search. Worked on orders and generating the invoice.

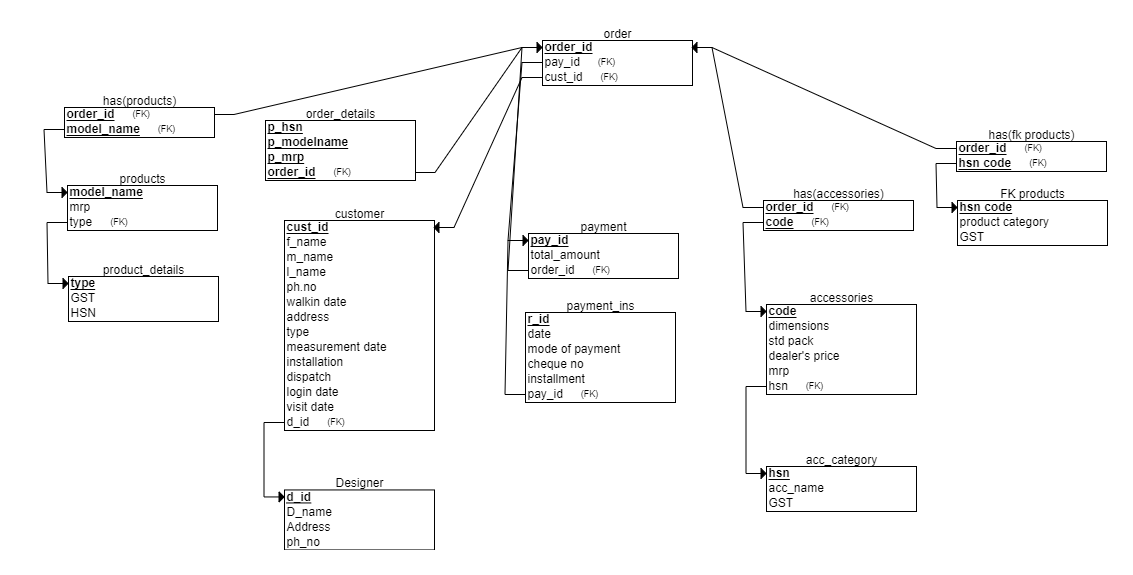
Akashini Koppad – Backend of functionalities of homepage. Worked on orders and generating receipts.

**Design updating:** The staff entity was removed as there are very few staff in the shop and instead only the designers data is maintained in an entity called designer.



**Implementation Phase Questions to be answered**

**Normalization:** The full kitchen customer entity is combined with the customer entity and the customer entity is found to be in 1NF and 2NF. As there are no functional dependencies, customer entity is not in 3NF. As the new entity designer is maintained instead of staff, it is found to be in 1NF and 2NF. As there are no functional dependencies in the designer entity, it is not in 3NF. The revised relation schema is drawn below.



**Question1:** Give the SQL statement(s) used to create the Oracle/MySQL database tables needed to implement the normalized relational schema.

class DesignerModel(models.Model):

dname =models.CharField(max\_length= 100)

dph\_no =models.CharField(max\_length=10)

daddress=models.CharField(max\_length=400)

def \_str\_(self):

return self.dname

class CustomerModel(models.Model):

TYPE =(

('Retail Customer','Retail Customer'),

('Full Kitchen Customer','Full Kitchen Customer'),

)

fname = models.CharField(max\_length=30 )

mname = models.CharField(max\_length=30,null =True ,blank=True)

lname = models.CharField(max\_length=30,null =True,blank=True)

ph\_no=models.CharField(max\_length=10,null=True,blank=True)

address=models.CharField(max\_length=400,null =True,blank=True)

walk\_in\_date=models.DateField(null =True,blank=True)

customer\_type=models.CharField(max\_length=100,null =True,choices=TYPE)

designer\_name=ForeignKey(DesignerModel, on\_delete=models.CASCADE)

visit\_date =models.DateField(null =True,blank=True)

measure\_date=models.DateField(null =True,blank=True)

login\_date=models.DateField(null =True,blank=True)

dispatch\_date=models.DateField(null =True,blank=True)

install\_date=models.DateField(null =True,blank=True)

def \_str\_(self):

return self.fname

class productCategoryModel(models.Model):

name = models.CharField(max\_length=30, primary\_key=True)

hsn = models.CharField(max\_length=30)

gst = models.FloatField()

def \_str\_(self):

return self.name

class productModel(models.Model):

category = ForeignKey(productCategoryModel, on\_delete=models.CASCADE)

name = models.CharField(max\_length=30)

dealersPrice = models.FloatField()

mrp = models.FloatField()

def \_str\_(self):

return self.name

class accessoriesCategoryModel(models.Model):

name = models.CharField(max\_length=30, primary\_key=True)

hsn = models.CharField(max\_length=30)

gst = models.FloatField()

def \_str\_(self):

return self.name

class accessoriesModel(models.Model):

category = ForeignKey(accessoriesCategoryModel, on\_delete=models.CASCADE)

code = models.CharField(max\_length=30)

dealersPrice = models.FloatField()

mrp = models.FloatField()

stdPack = models.IntegerField()

dimensions = models.CharField(max\_length=20)

def \_str\_(self):

return self.code

class orderModel(models.Model):

customer = models.ForeignKey(CustomerModel , on\_delete= models.CASCADE)

product = models.ForeignKey(productModel, on\_delete= models.CASCADE)

accessories = models.ForeignKey(accessoriesModel, on\_delete= models.CASCADE)

class InvoiceModel(models.Model):

order\_no = models.ForeignKey(OrderModel,on\_delete=models.CASCADE)

totalamount = models.FloatField()

class PaymentModel(models.Model):

invoice\_no = models.ForeignKey(InvoiceModel,on\_delete=models.CASCADE)

class InstallModel(models.Model):

payid =models.ForeignKey(PaymentModel,on\_delete=CASCADE)

date=models.DateField()

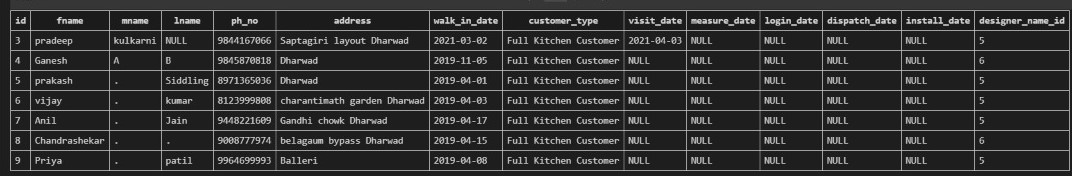
mode\_of\_pay = models.CharField(max\_length= 50)

chequeno = models.CharField(max\_length=30)

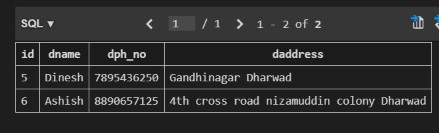
amount\_paid = models.FloatField()

**Question2:** Give the actual data stored in each table of the database. (real sample data)

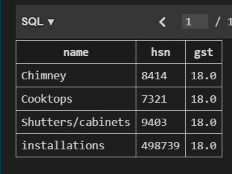
Customer table:



Designer table:



Product Category table:



Products table:

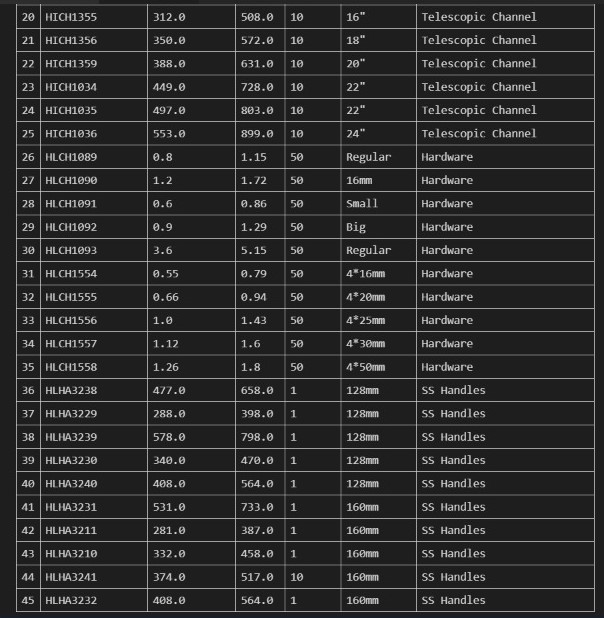


Accessories Category table:



Accessories table:

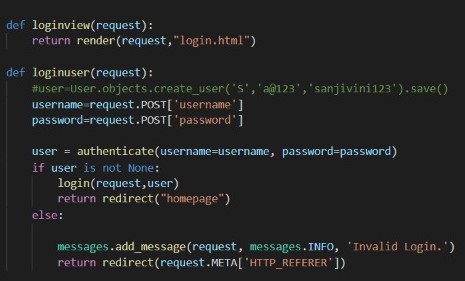




**Question3:** Give the snapshots, description and SQL queries for each of the user interface forms for your application. (Create the front end using Django and hook it up to the SQL.)

For login:

The username and password which are entered is compared with the one that is existing in the database. If it matches, then it is directed to the homepage or else “invalid login” error is shown.

****

For homepage:

In homepage, the analysis of customer data and orders data is displayed.



For Adding Customer:

When add customer button is clicked in the customer page, it will redirect the admin to a new page where he can add new customers into the database.



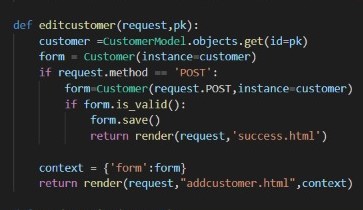
For searching customer:

Admin can search the customers present in the database based on the first name. If the customer is found, the customer details are displayed, if the customer record is not found, a message is displayed for the same.



For editing customer:

On clicking the edit button, the admin is redirected to a page where he can make changes to the existing customer record.



For adding product category:

Admin can add new category of the product to the database.



For adding product:

When add product button is clicked in the product page, it will redirect the admin to a new page where he can add new products into the database.



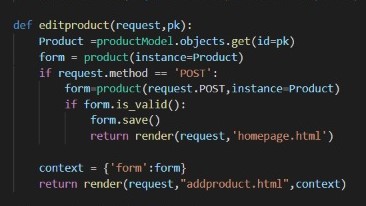
For searching product:

Admin can search the products present in the database based on the product name. If the product is found, the product details are displayed, if the product record is not found, a message is displayed for the same.



For editing product:

On clicking the edit button, the admin is redirected to a page where he can make changes to the existing product record.



For deleting product:

On clicking the delete product button, that particular product record will be deleted from the database.



For adding accessories category:

Admin can add new category of the accessory to the database.



For adding accessories:

When add accessory button is clicked in the accessory page, it will redirect the admin to a new page where he can add new accessories into the database.



For searching accessories:

Admin can search the accessories present in the database based on the accessory name. If the accessory is found, the accessories details are displayed, if the accessory record is not found, a message is displayed for the same.



For editing accessories:

On clicking the edit button, the admin is redirected to a page where he can make changes to the existing accessory record.



For deleting accessories:

On clicking the delete accessory button, that particular accessory record will be deleted from the database.

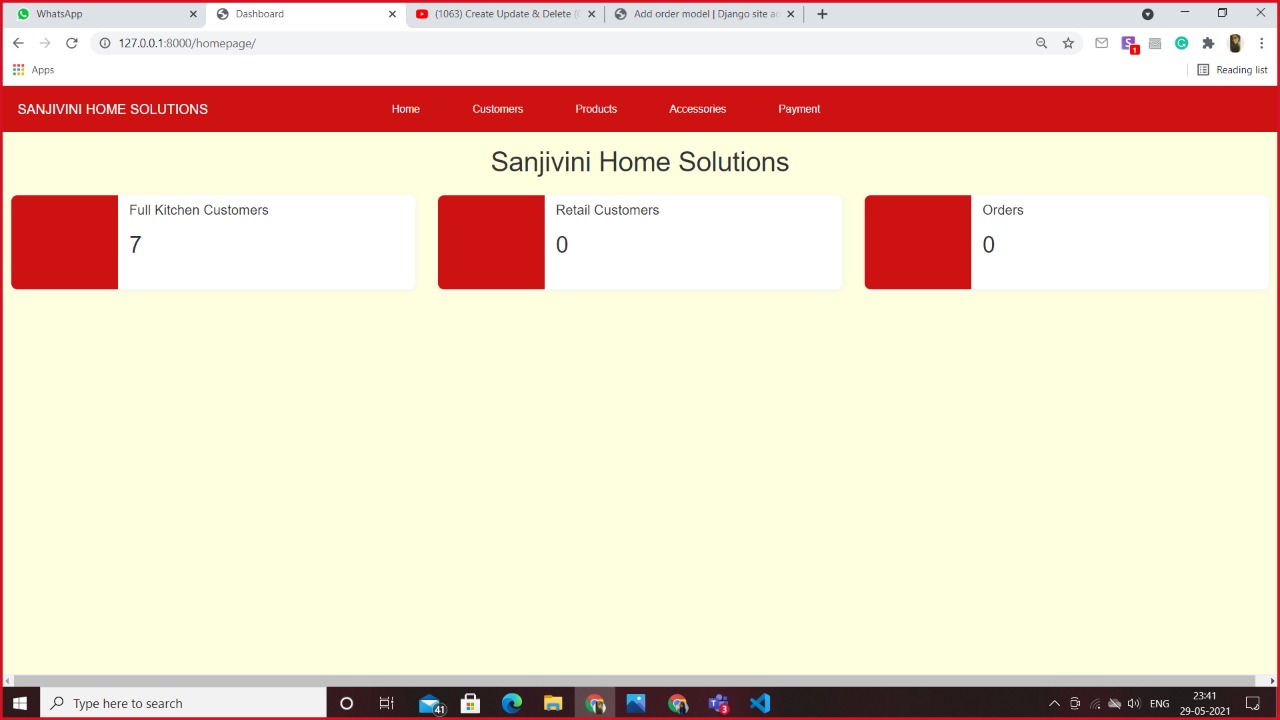


For adding designers:

New designers can be added to the database.



**Question4:** Give all possible final bill reports/other forms of ledger reports summarized etc and graphs obtained by your application.



SQL Statements:

class DesignerModel(models.Model):

dname =models.CharField(max\_length= 100)

dph\_no =models.CharField(max\_length=10)

daddress=models.CharField(max\_length=400)

def \_str\_(self):

return self.dname

class CustomerModel(models.Model):

TYPE =(

('Retail Customer','Retail Customer'),

('Full Kitchen Customer','Full Kitchen Customer'),

)

fname = models.CharField(max\_length=30 )

mname = models.CharField(max\_length=30,null =True ,blank=True)

lname = models.CharField(max\_length=30,null =True,blank=True)

ph\_no=models.CharField(max\_length=10,null=True,blank=True)

address=models.CharField(max\_length=400,null =True,blank=True)

walk\_in\_date=models.DateField(null =True,blank=True)

customer\_type=models.CharField(max\_length=100,null =True,choices=TYPE)

designer\_name=ForeignKey(DesignerModel, on\_delete=models.CASCADE)

visit\_date =models.DateField(null =True,blank=True)

measure\_date=models.DateField(null =True,blank=True)

login\_date=models.DateField(null =True,blank=True)

dispatch\_date=models.DateField(null =True,blank=True)

install\_date=models.DateField(null =True,blank=True)

def \_str\_(self):

return self.fname

class productCategoryModel(models.Model):

name = models.CharField(max\_length=30, primary\_key=True)

hsn = models.CharField(max\_length=30)

gst = models.FloatField()

def \_str\_(self):

return self.name

class productModel(models.Model):

category = ForeignKey(productCategoryModel, on\_delete=models.CASCADE)

name = models.CharField(max\_length=30)

dealersPrice = models.FloatField()

mrp = models.FloatField()

def \_str\_(self):

return self.name

class accessoriesCategoryModel(models.Model):

name = models.CharField(max\_length=30, primary\_key=True)

hsn = models.CharField(max\_length=30)

gst = models.FloatField()

def \_str\_(self):

return self.name

class accessoriesModel(models.Model):

category = ForeignKey(accessoriesCategoryModel, on\_delete=models.CASCADE)

code = models.CharField(max\_length=30)

dealersPrice = models.FloatField()

mrp = models.FloatField()

stdPack = models.IntegerField()

dimensions = models.CharField(max\_length=20)

def \_str\_(self):

return self.code

class orderModel(models.Model):

customer = models.ForeignKey(CustomerModel , on\_delete= models.CASCADE)

product = models.ForeignKey(productModel, on\_delete= models.CASCADE)

accessories = models.ForeignKey(accessoriesModel, on\_delete= models.CASCADE)

class InvoiceModel(models.Model):

order\_no = models.ForeignKey(OrderModel,on\_delete=models.CASCADE)

totalamount = models.FloatField()

class PaymentModel(models.Model):

invoice\_no = models.ForeignKey(InvoiceModel,on\_delete=models.CASCADE)

class InstallModel(models.Model):

payid =models.ForeignKey(PaymentModel,on\_delete=CASCADE)

date=models.DateField()

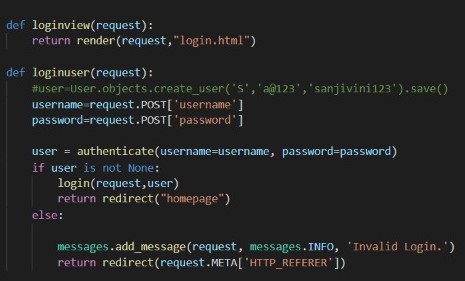
mode\_of\_pay = models.CharField(max\_length= 50)

chequeno = models.CharField(max\_length=30)

amount\_paid = models.FloatField()

Queries:

For login:

****

For homepage:



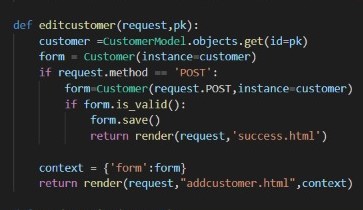
For Adding Customer:



For searching customer:



For editing customer:



For adding product category:



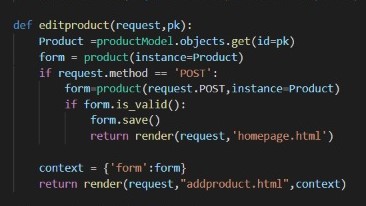
For adding product:



For searching product:



For editing product:



For deleting product:



For adding accessories category:



For adding accessories:



For searching accessories:



For editing accessories:



For deleting accessories:

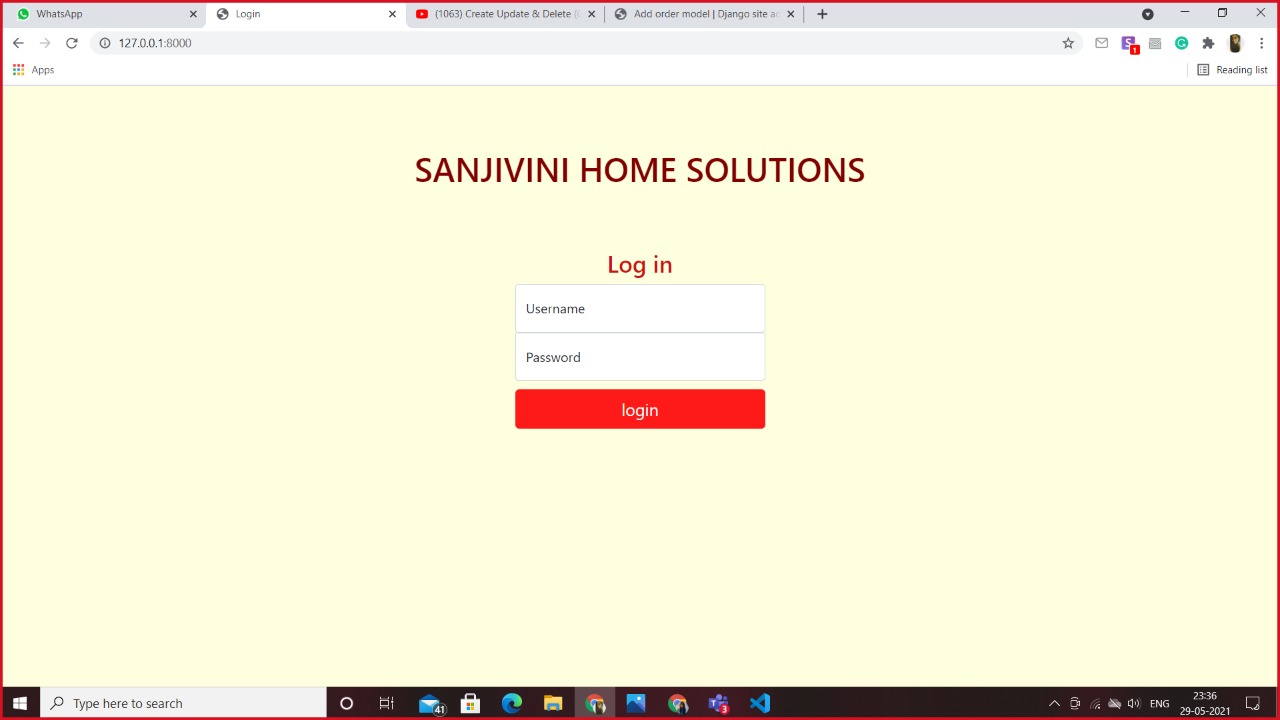


For adding designers:

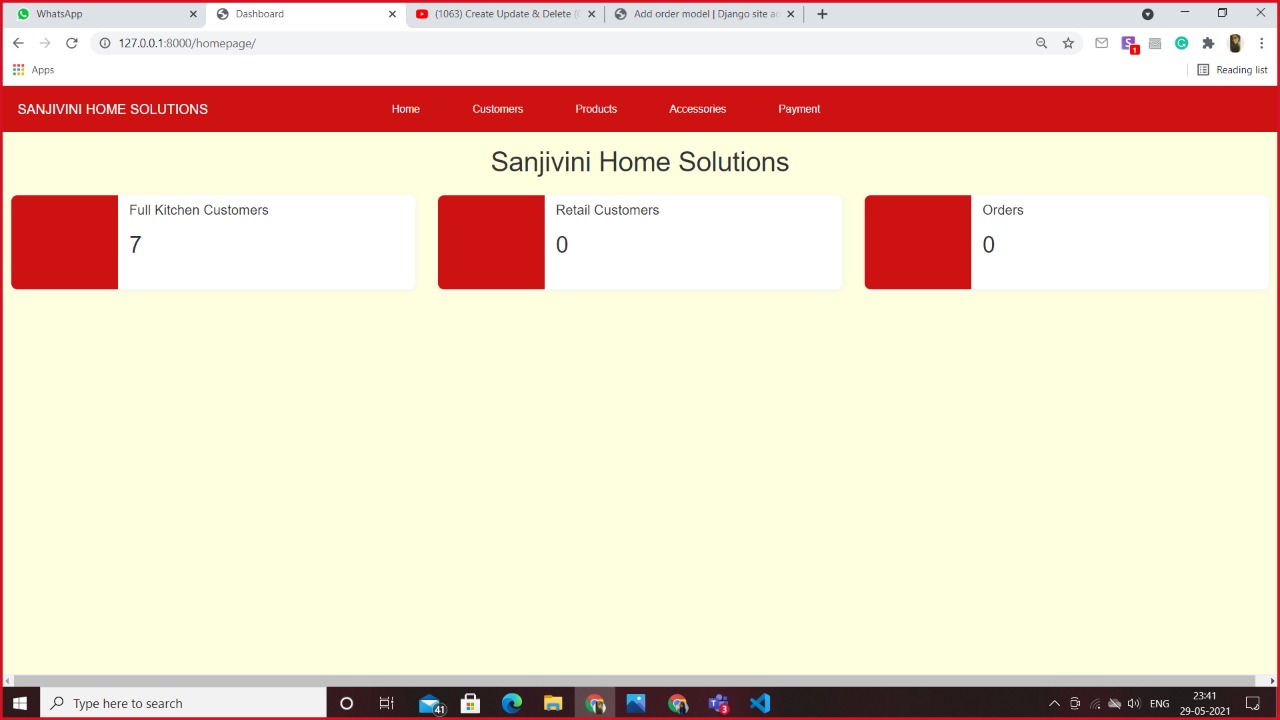


User Interface Screenshots:

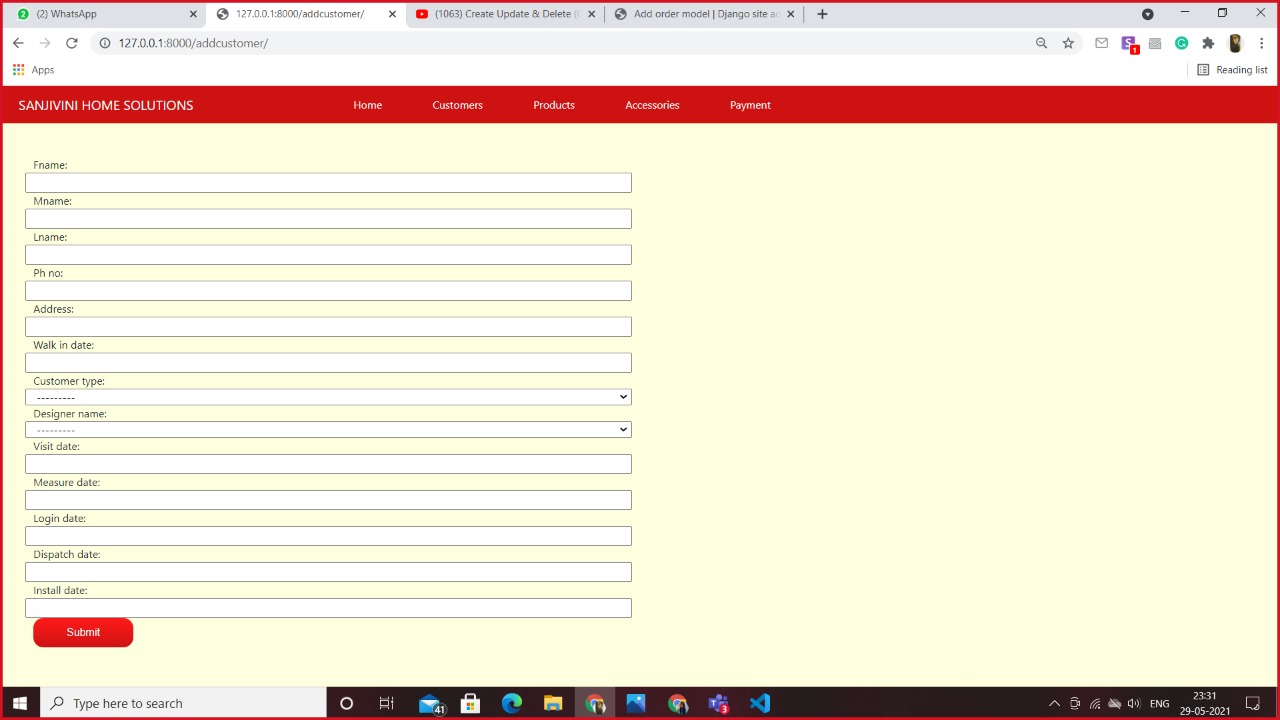
Login page:



Homepage:



Add customer page:



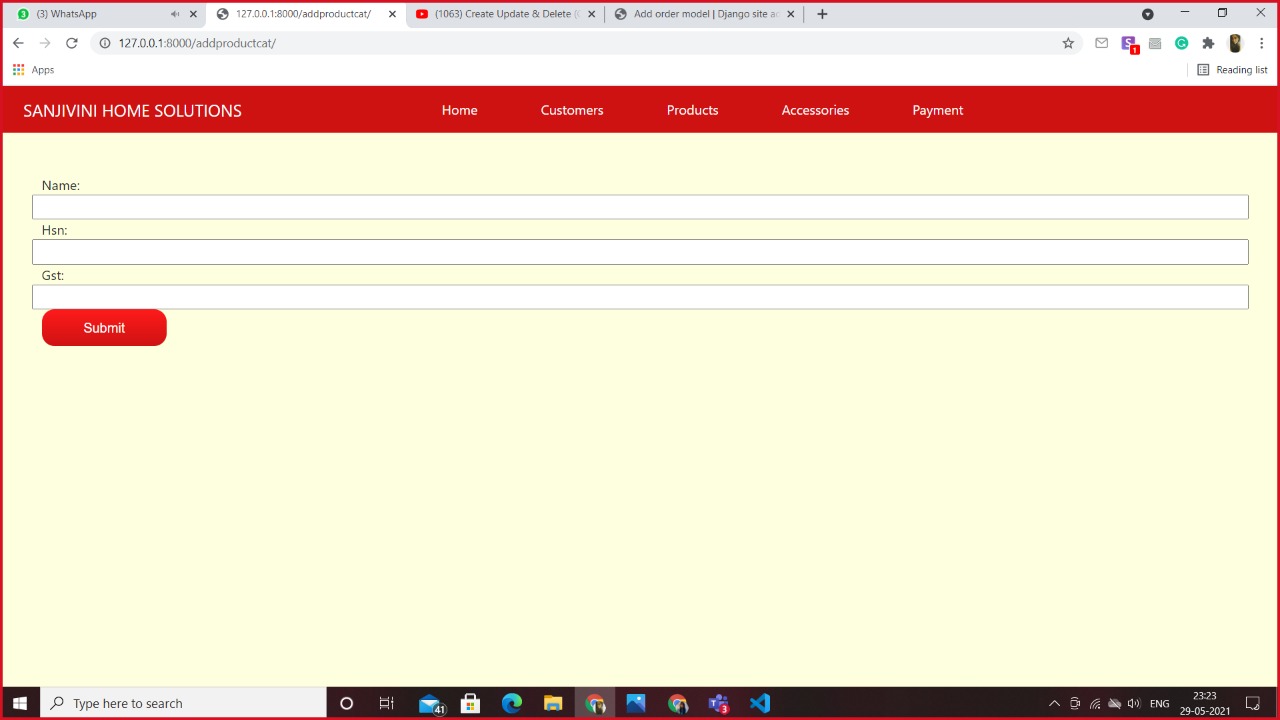
Customer page:



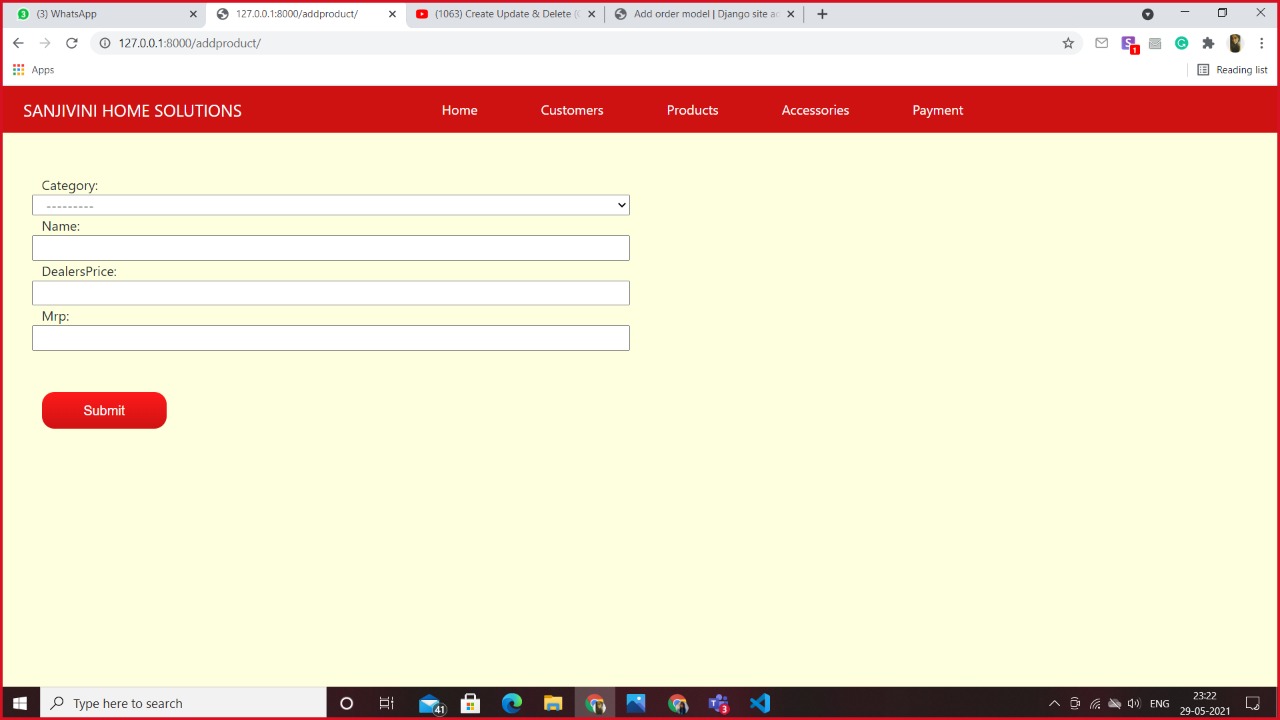
Search customer page:



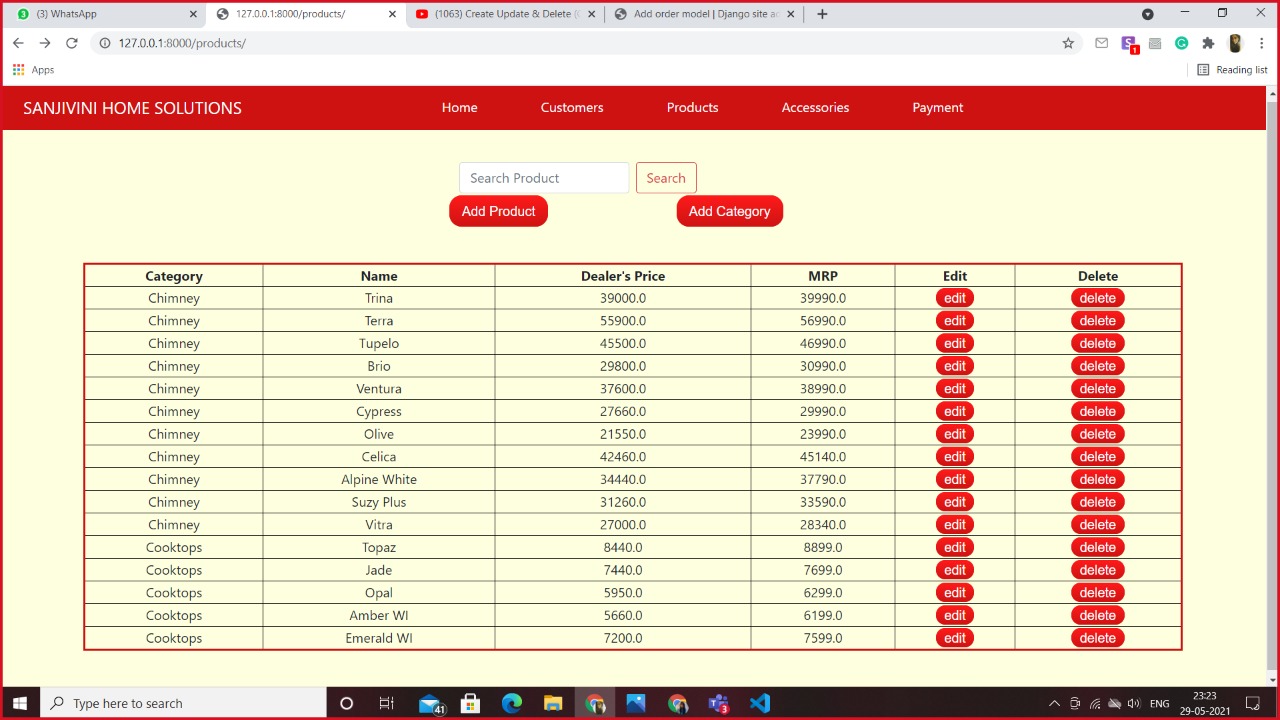
Add product category page:



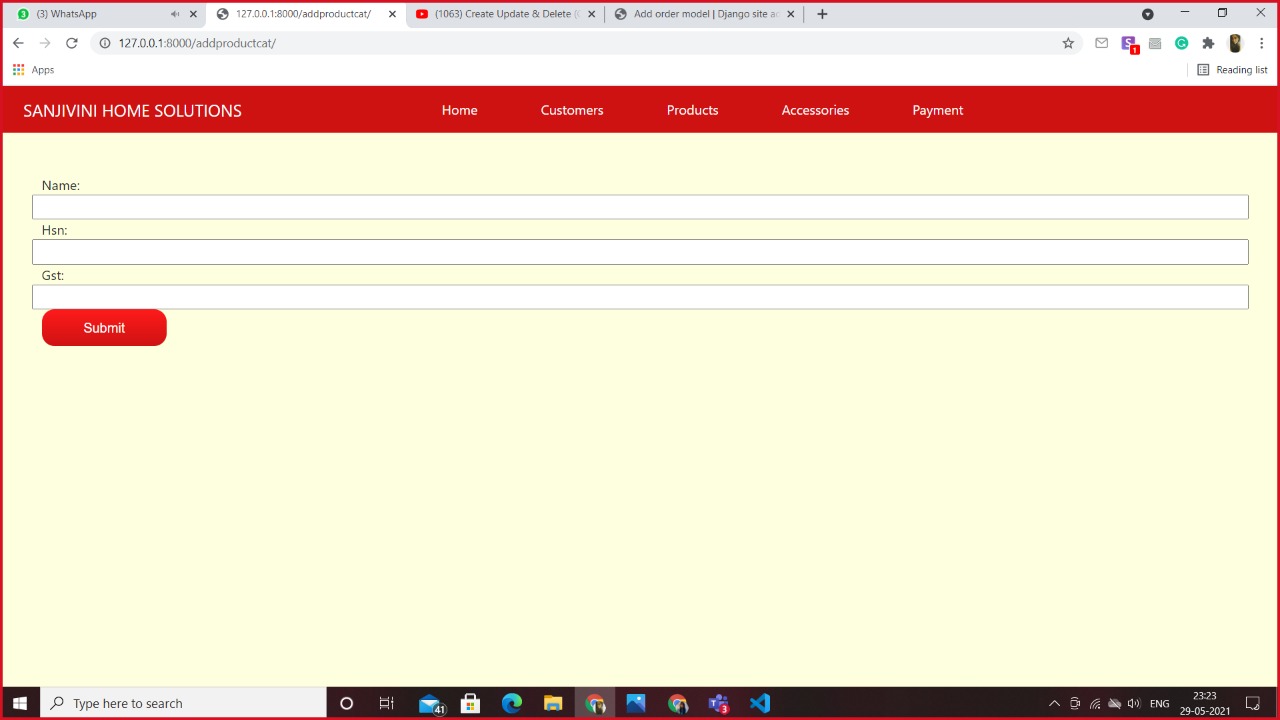
Add product page:



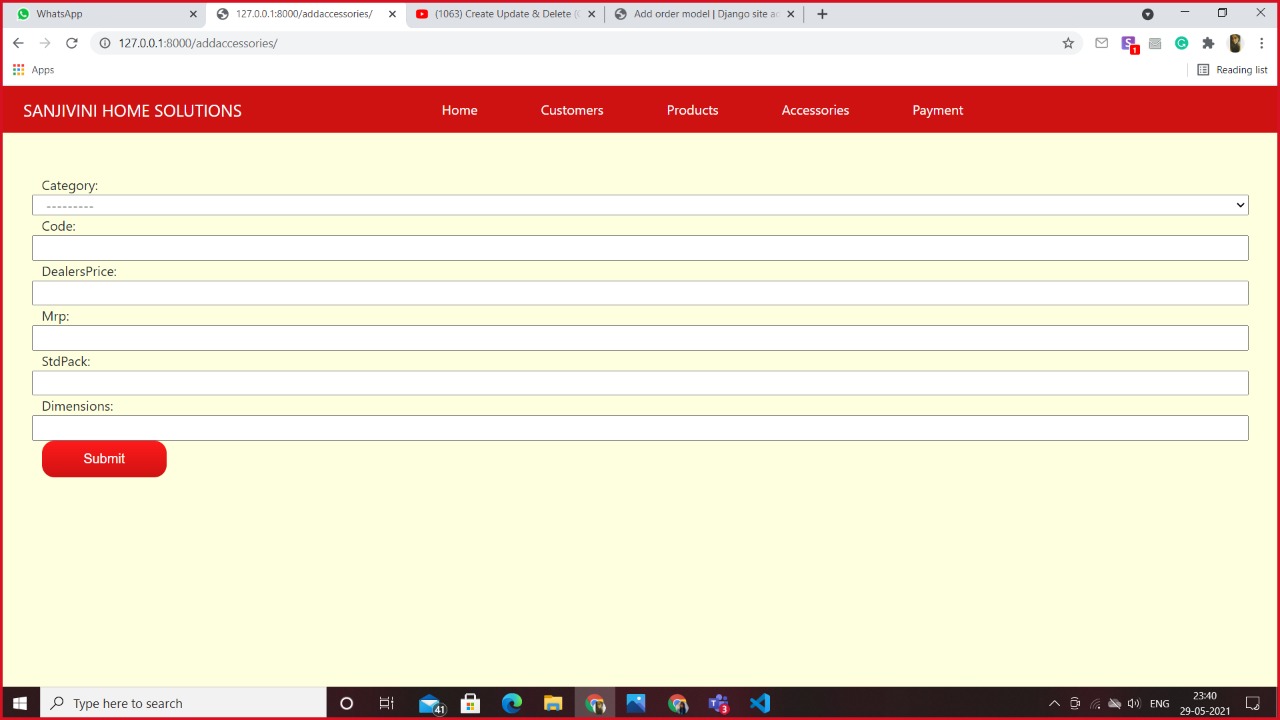
Products page:



Add accessories category page:

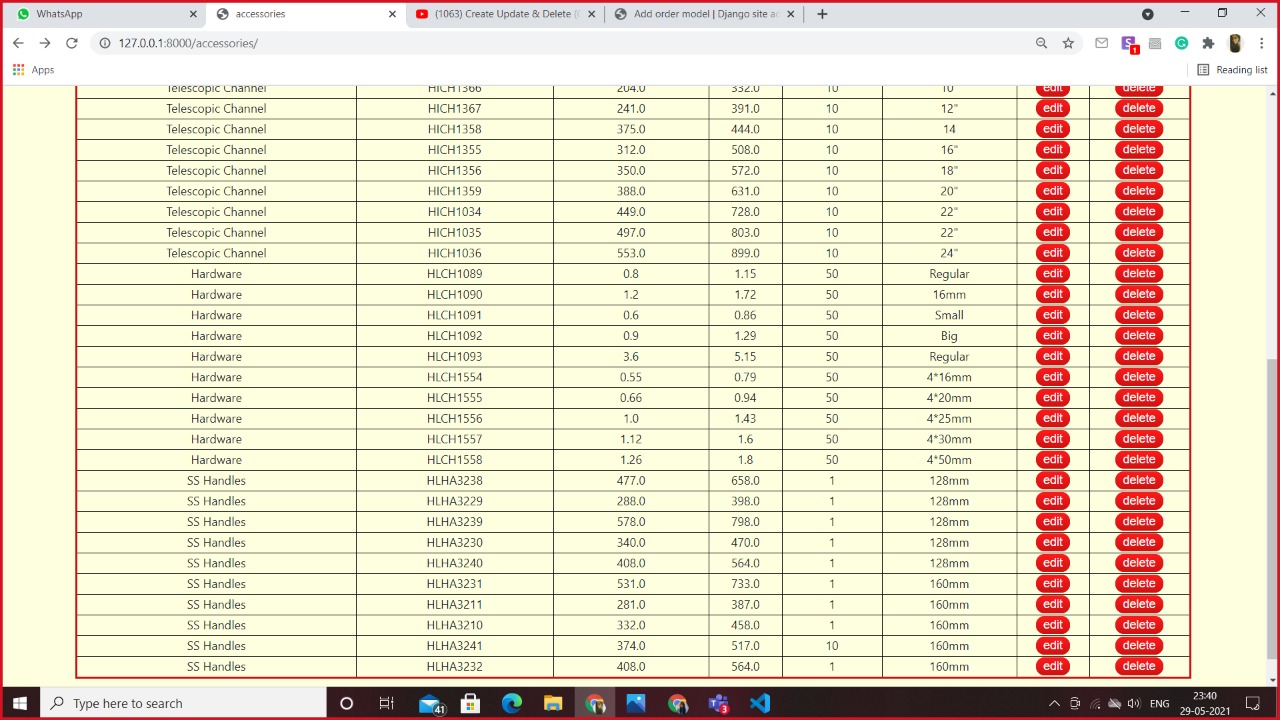


Add accessories page:



Accessories page:





# USER MANUAL

The Project consists of various pages:

login - Login with the credentials and you will be directed to the home page.

homepage - This page displays the dashboard of your shop. The navigation bar

has links to navigate to various pages. These pages include customers, products,

accessories and designer.

customers - This page consists of data of the customers. It also includes

links to take order, edit the customer's details and make payments. It also

consists of a search box to search a customer by his name.

products - This page consists the data of all the products sold by the shop.

It also consists of a search box to search a product by its name. Products can be

added, deleted and modified.

accessories - This page consists the data of all the accessories sold by the shop.

It also consists of a search box to search accessories by its hsn. Accessories can be

added, deleted and modified.

orders - This page can be accessed from the customers page when a particular

customer wishes to place an order. It enables the user to select the products

and accessories he wishes to buy. It contains a link to invoice page.

invoice - This page can be accessed from the orders page to generate an invoice

for a particular customer. The invoice can also be downloaded and shared.

payment - This page keeps track of the customer's total order amount and link to

the installment page.

installment - This page creates an installment of amount for a particular customer.

It also has link to generate receipt of that installment.

receipt - This page creates receipt of the amount paid and lets the owner to

download and share the receipt.

designer - This page contains designer's data.

# SNAPSHOTS

Snapshot of the client and the shop.





# ACCEPTANCE LETTER

