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:

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)

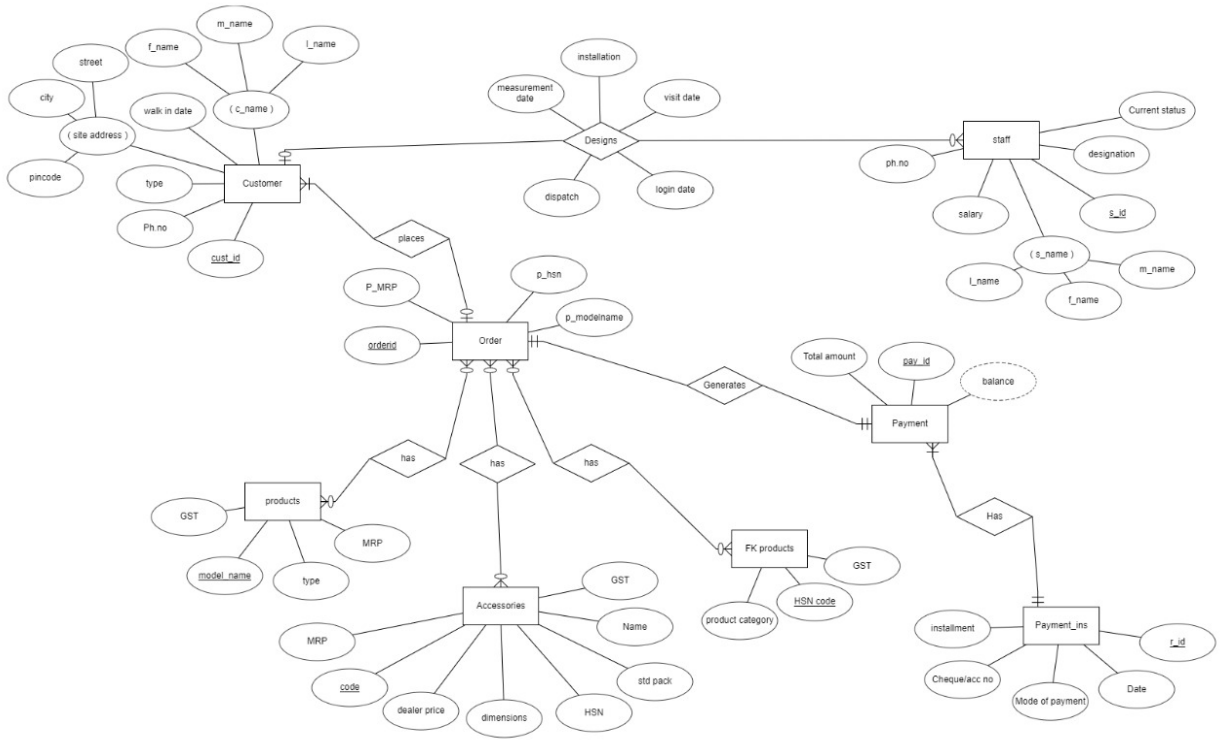
Order -has -FK products. (M:N)

Order-generates- payment. (1:1)

Payment -has- Payment_ins. (1:N)

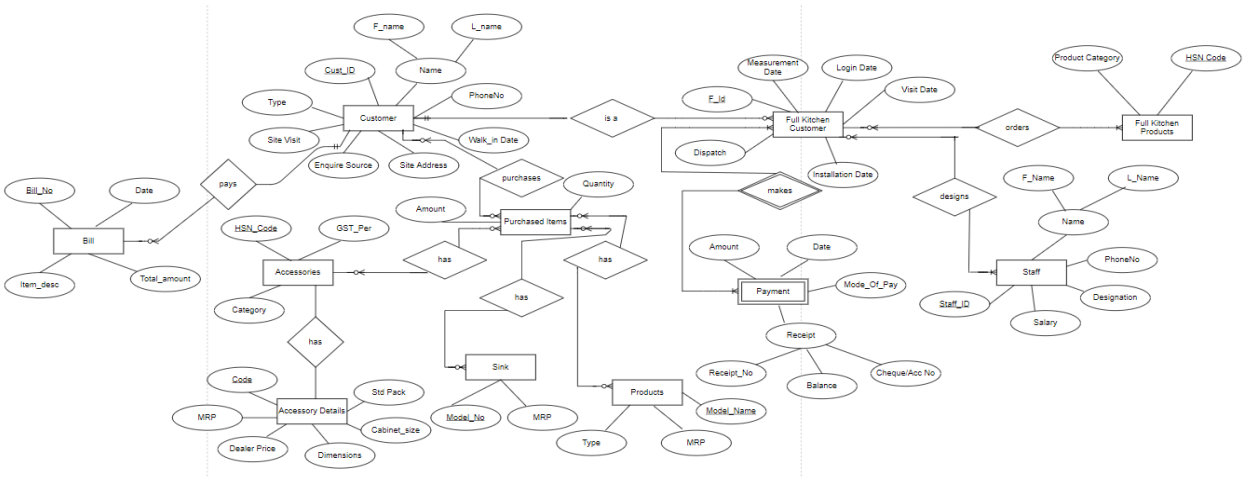
Staff -designs- Customer. (1 :N)

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

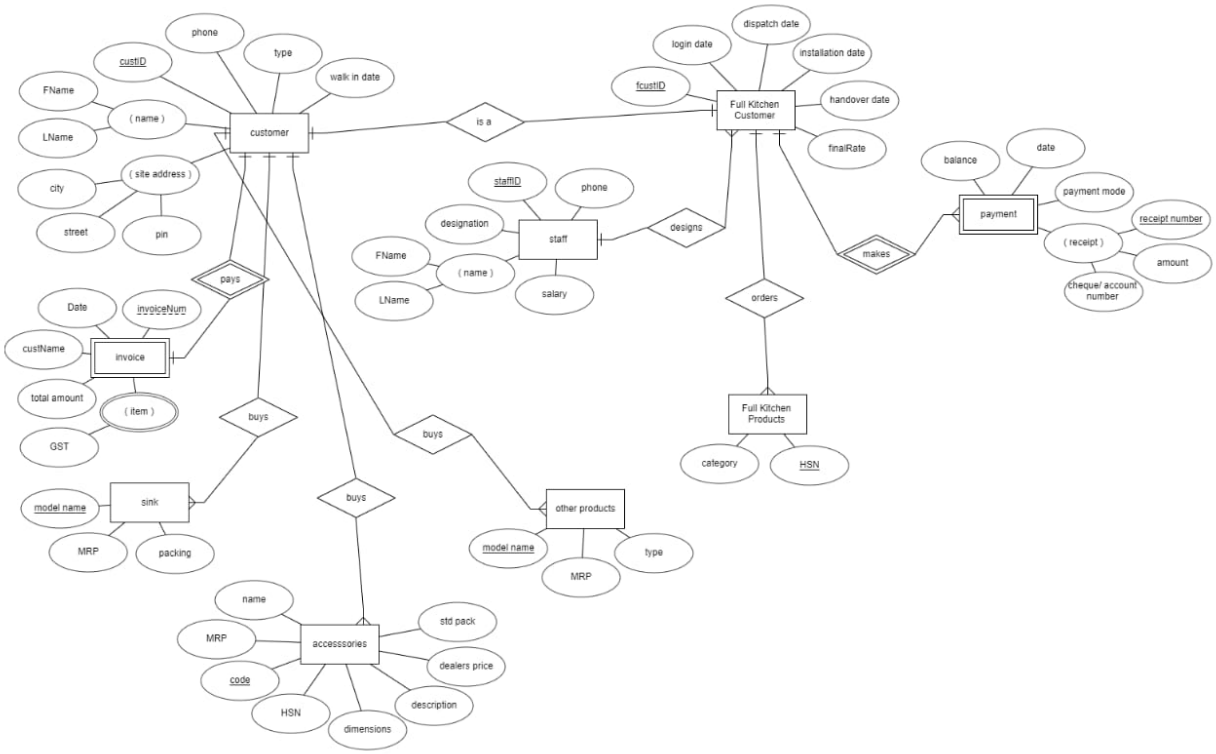


Question 3: Draw **alternate** Entity-Relationship Diagram illustrating the information you have identified in Question 1 that you think are most likely to occur.

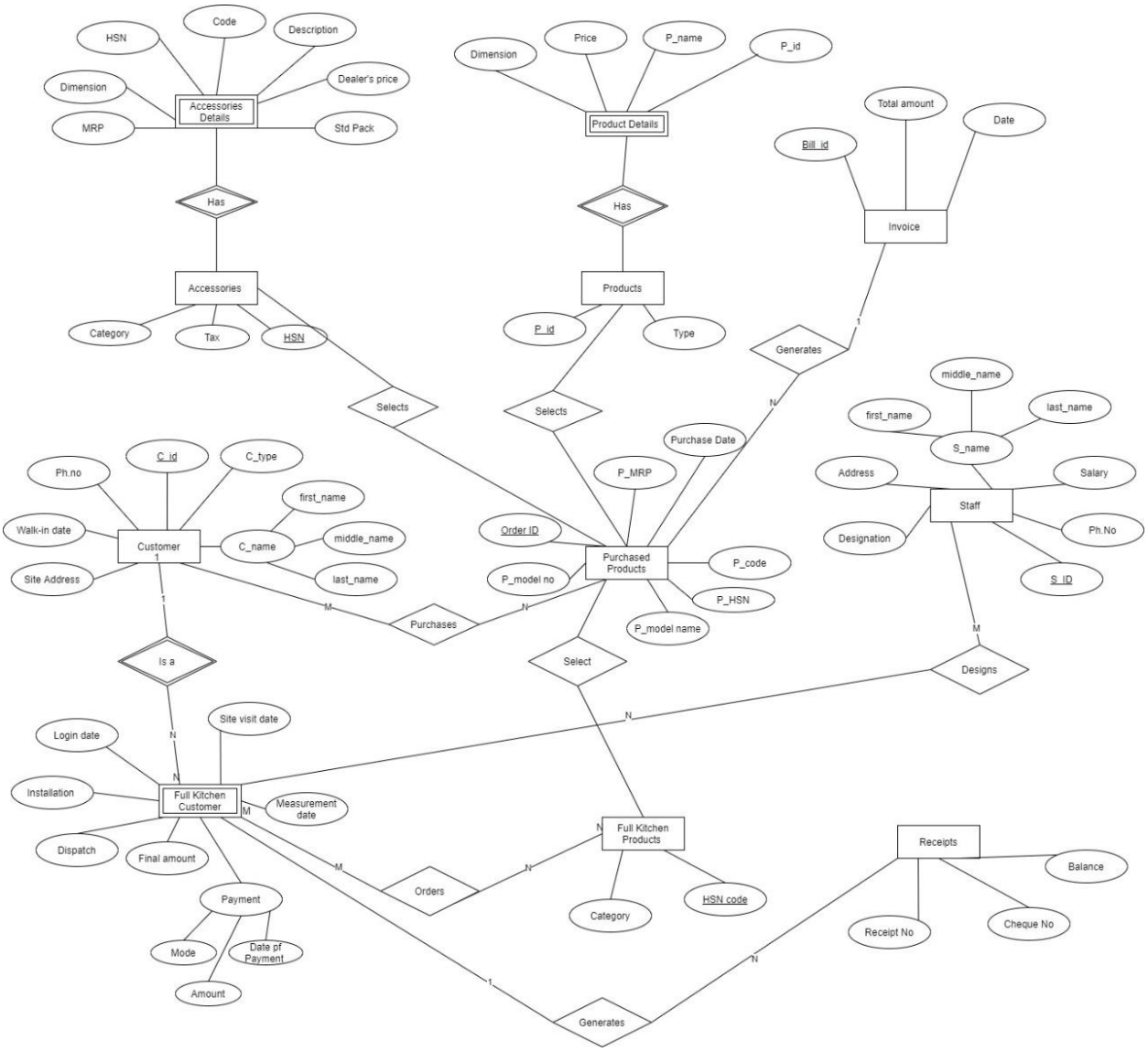
First Alternative:



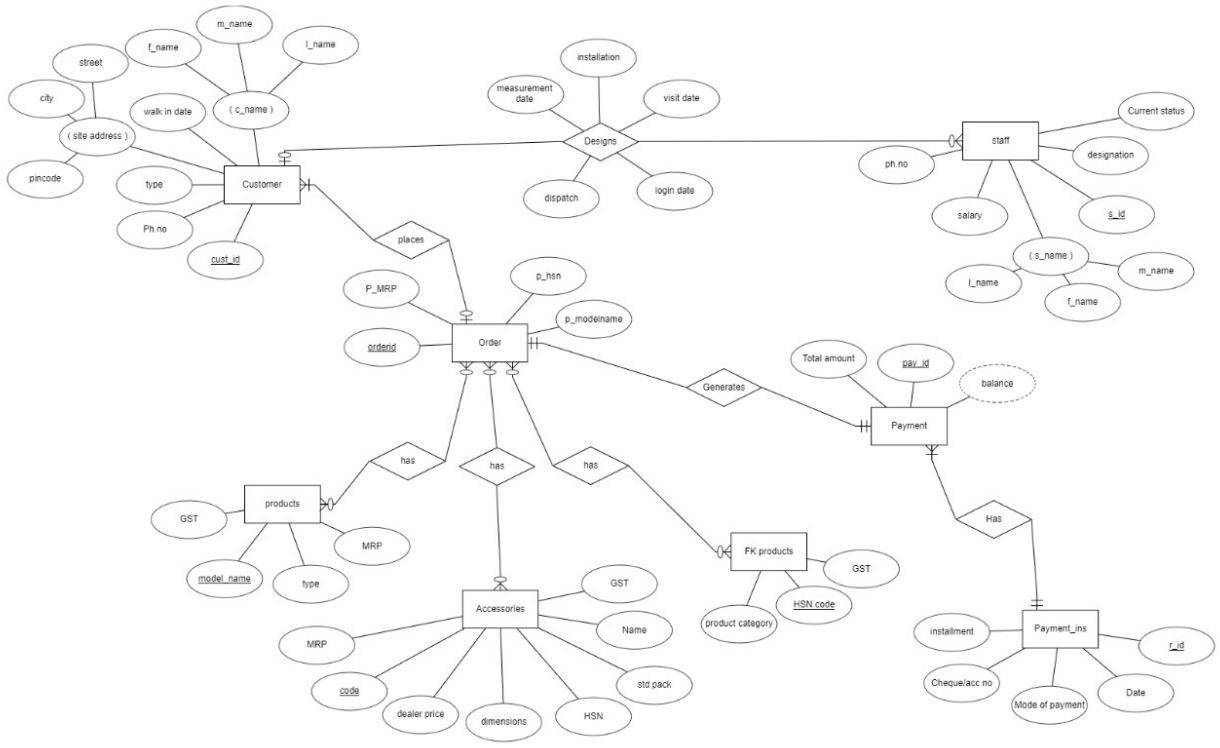
Second alternative:



Third Alternative:

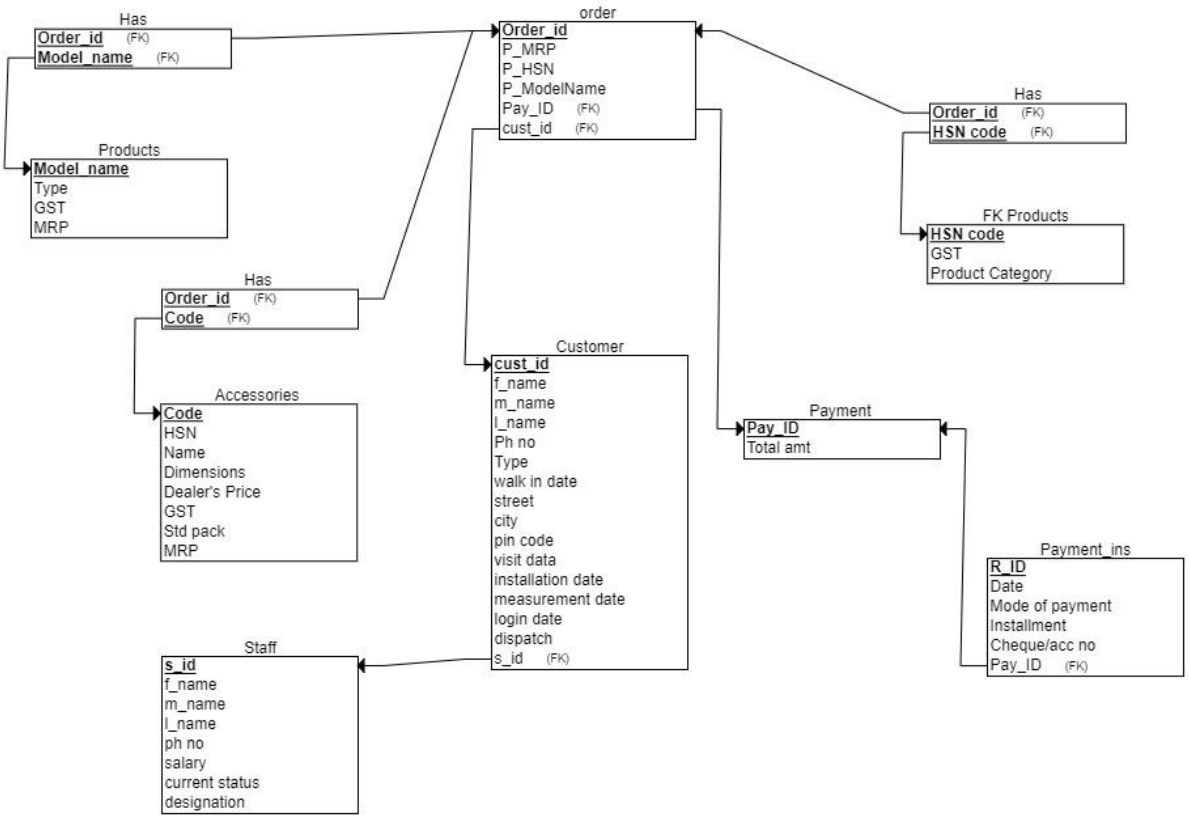


Question 4: Choose the **optimal** Entity-Relationship Diagram from the designs provided above and justify why you think this is an optimal solution for your identified problem specification.



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	No	No

			customer		
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	No	No

			dispatched or not		
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

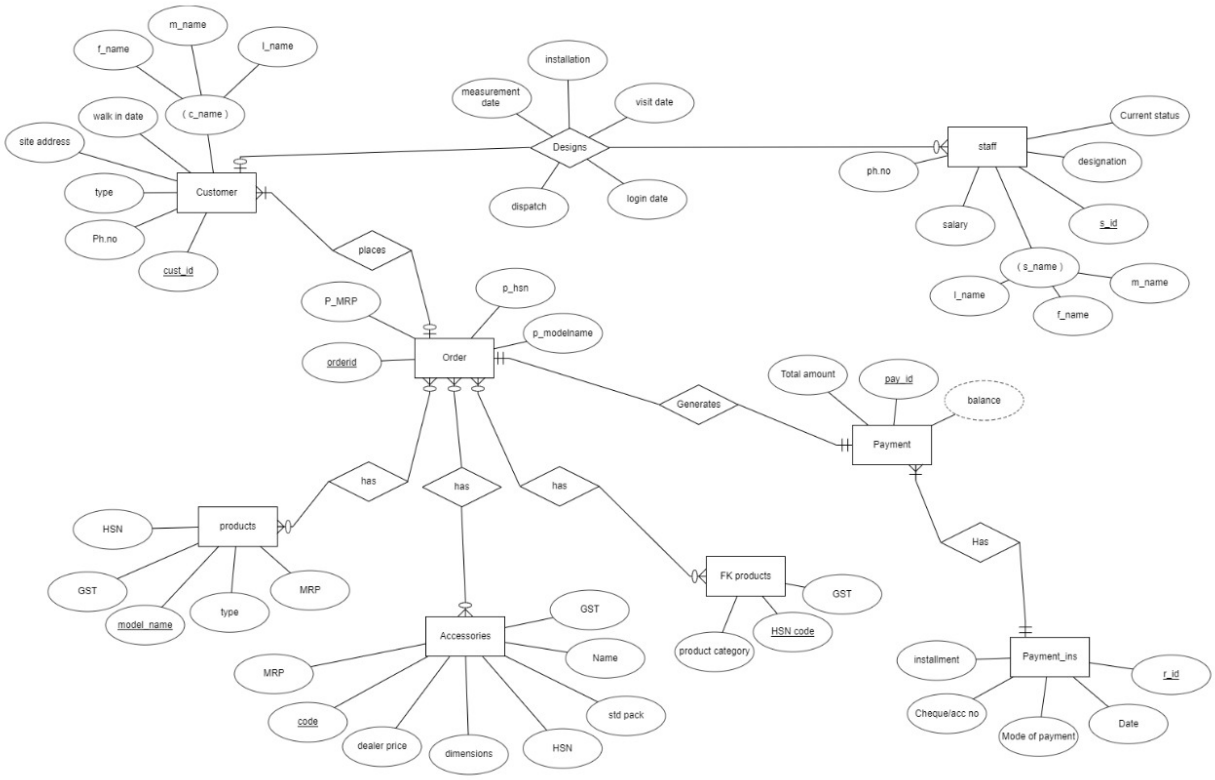
Responsibilities: Mukti Bhansali – Designing User Interface

Manisha Belagal – Normalisation.

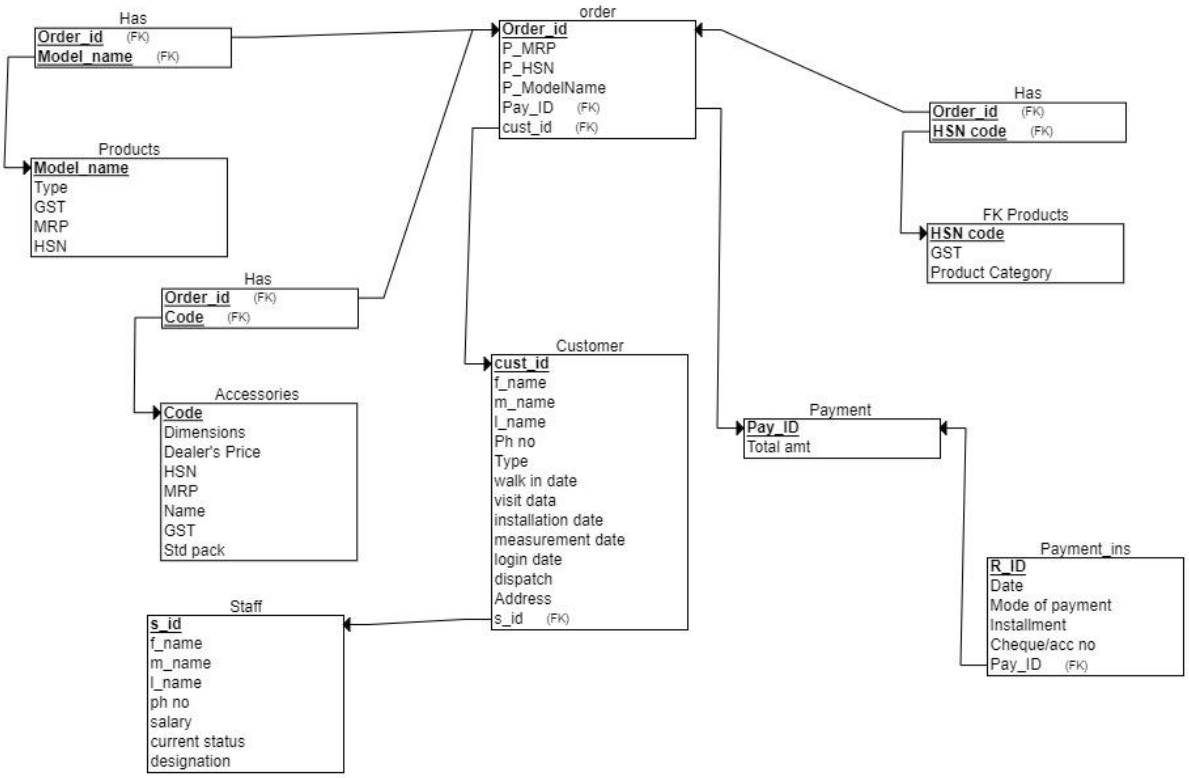
Mehar Anjum – Normalisation.

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.

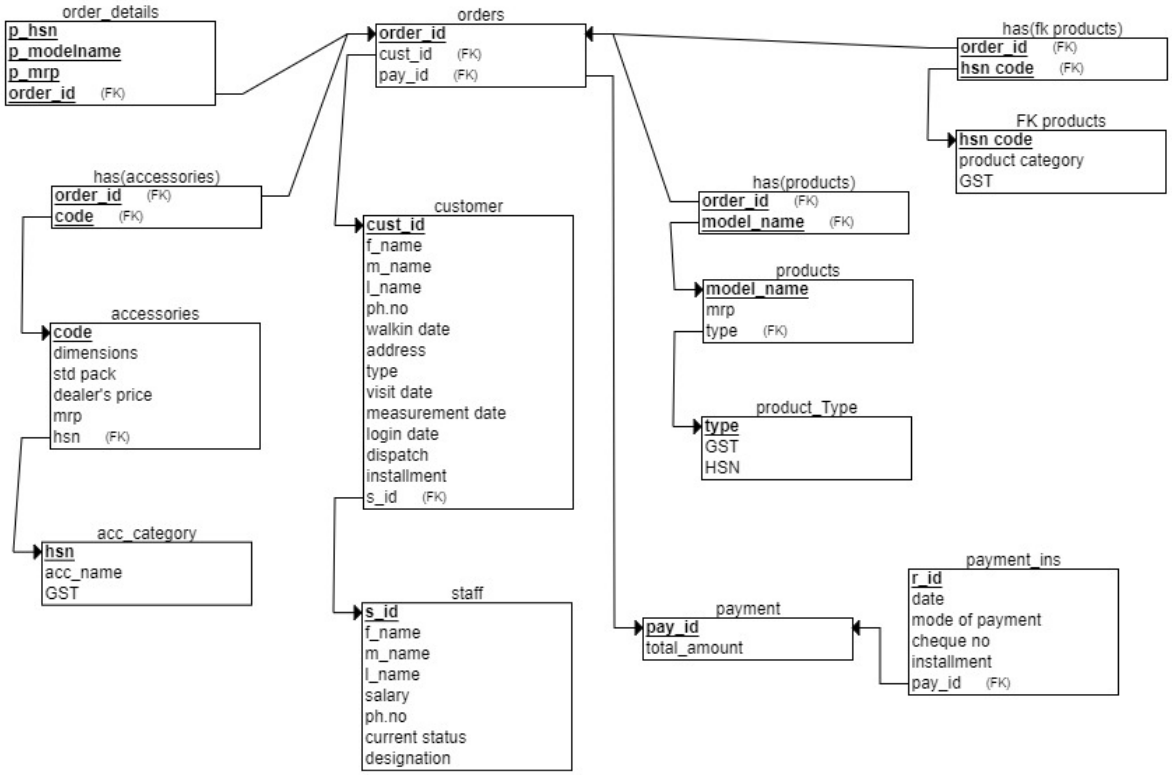
s_id -> f_name, m_name, l_name, salary, ph no, current status, designation

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.

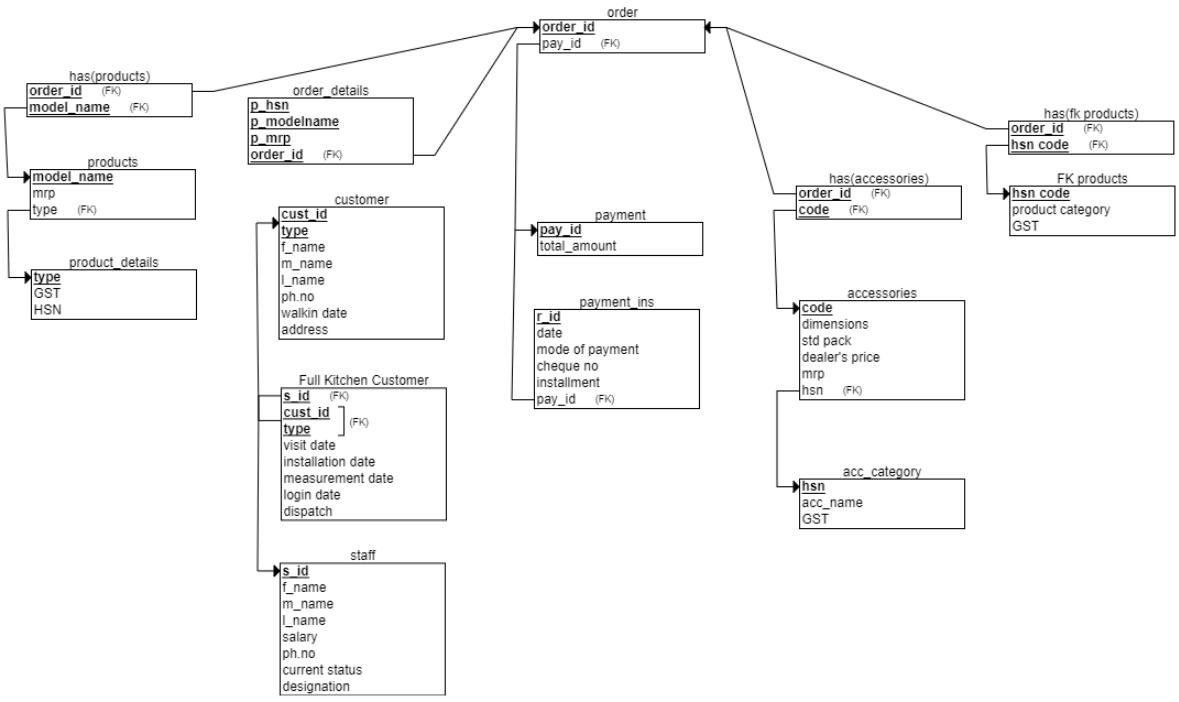
s_id -> f_name, m_name, l_name, salary, ph no, current status, designation

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

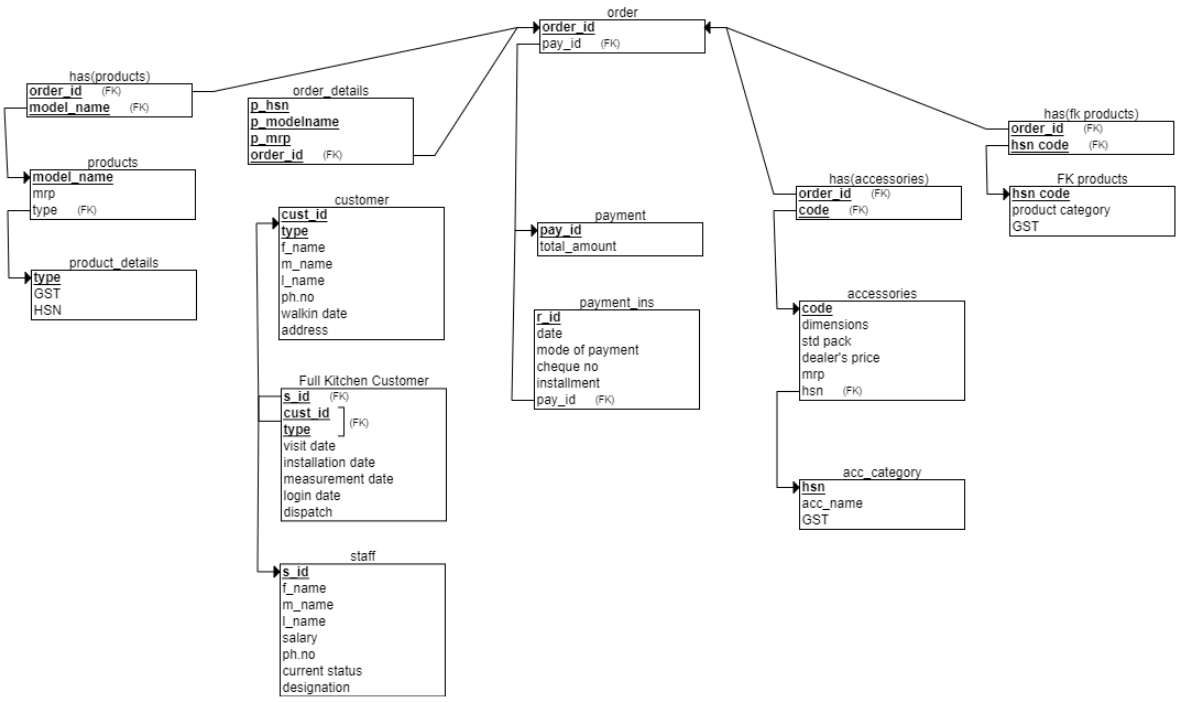
pay_id -> total amount

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

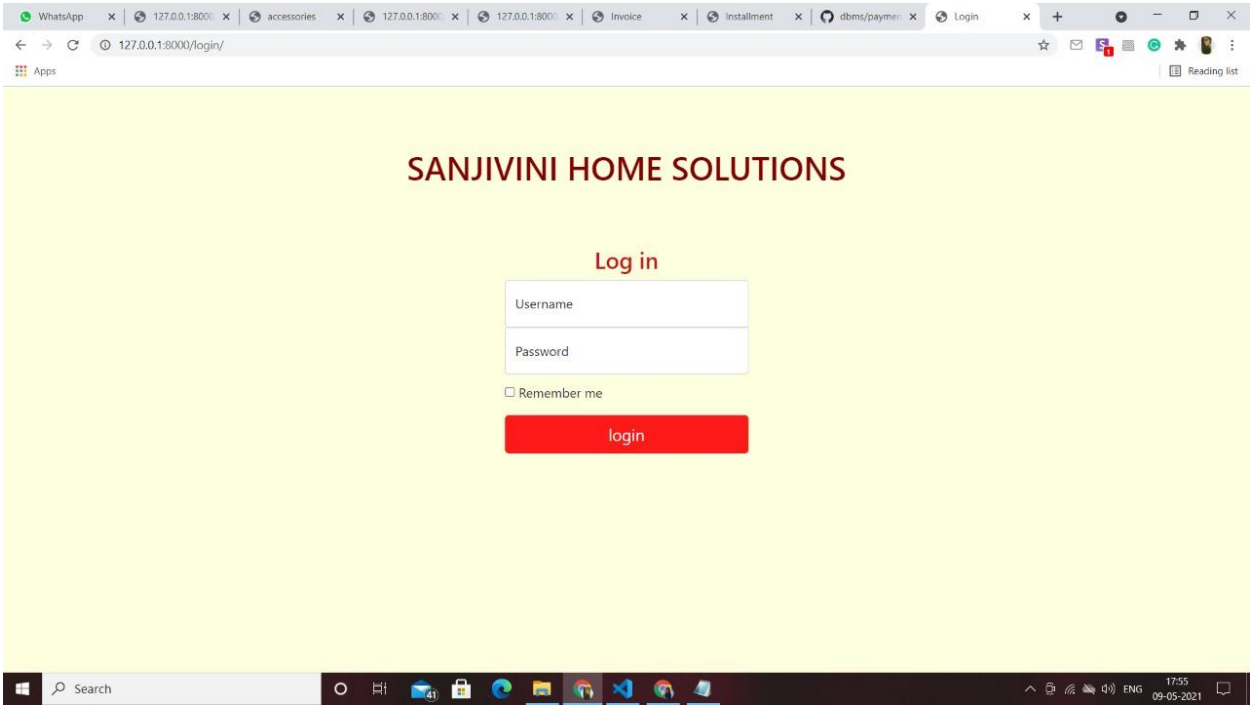


Question 2: The optimal schema chosen is the second normalized solution, as the redundant data in the customers table is removed by making a separate table for the attributes for Full Kitchen Customers only.

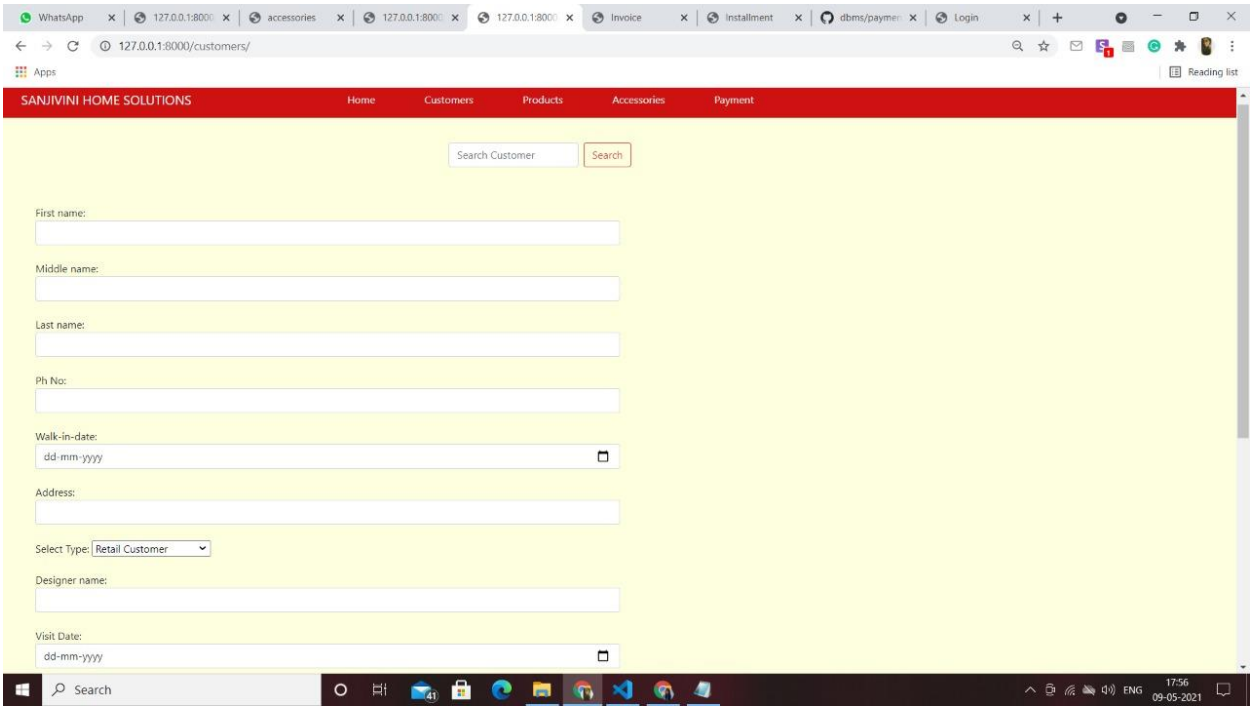


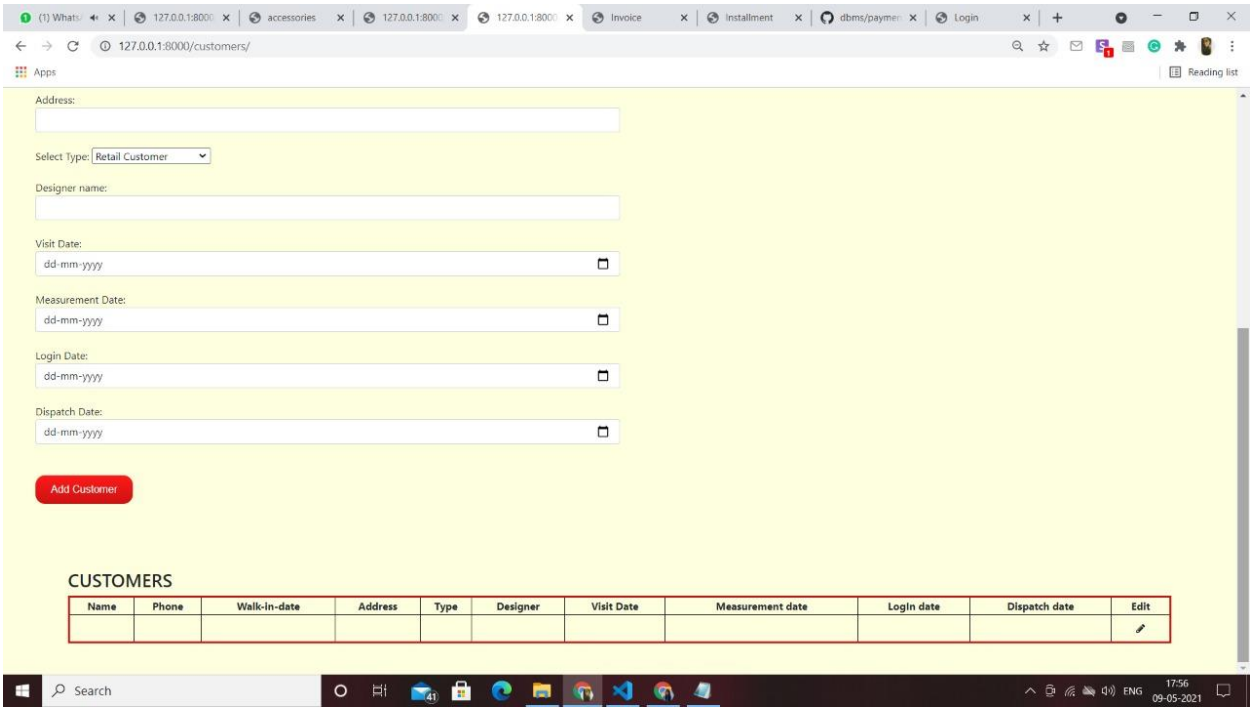
Question 3: User Interface (UI) design

Login Page:

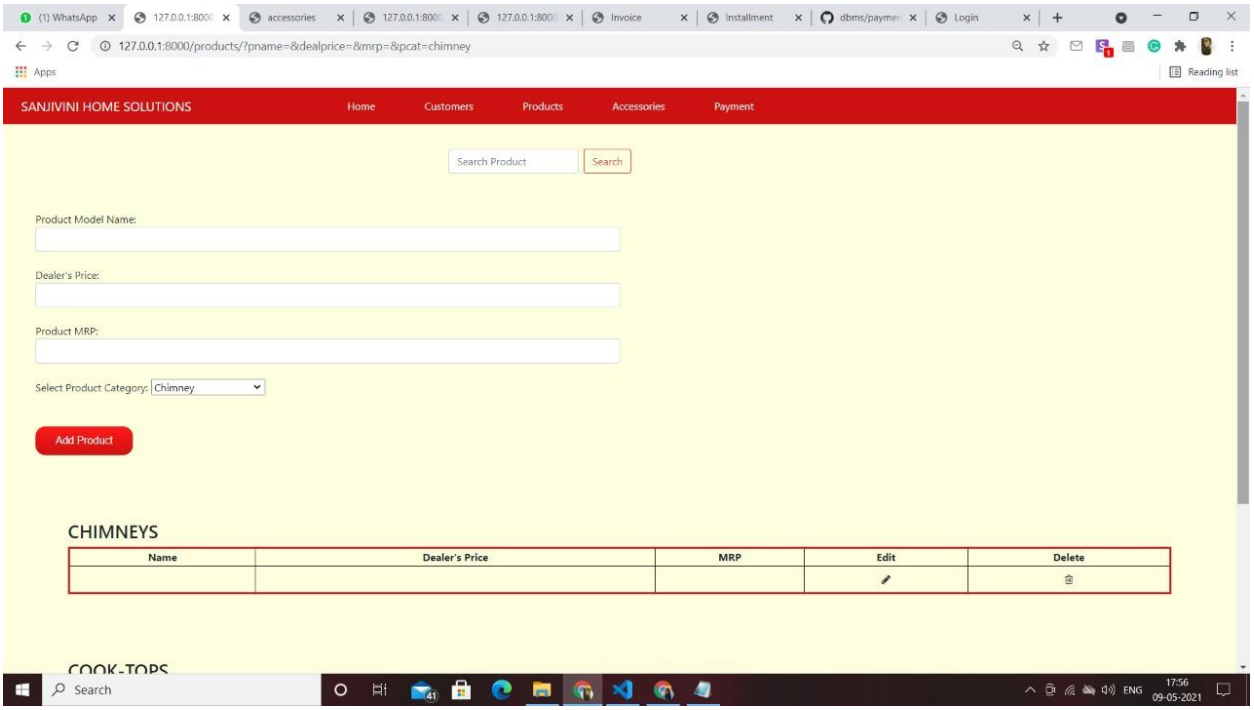


Customer Page:

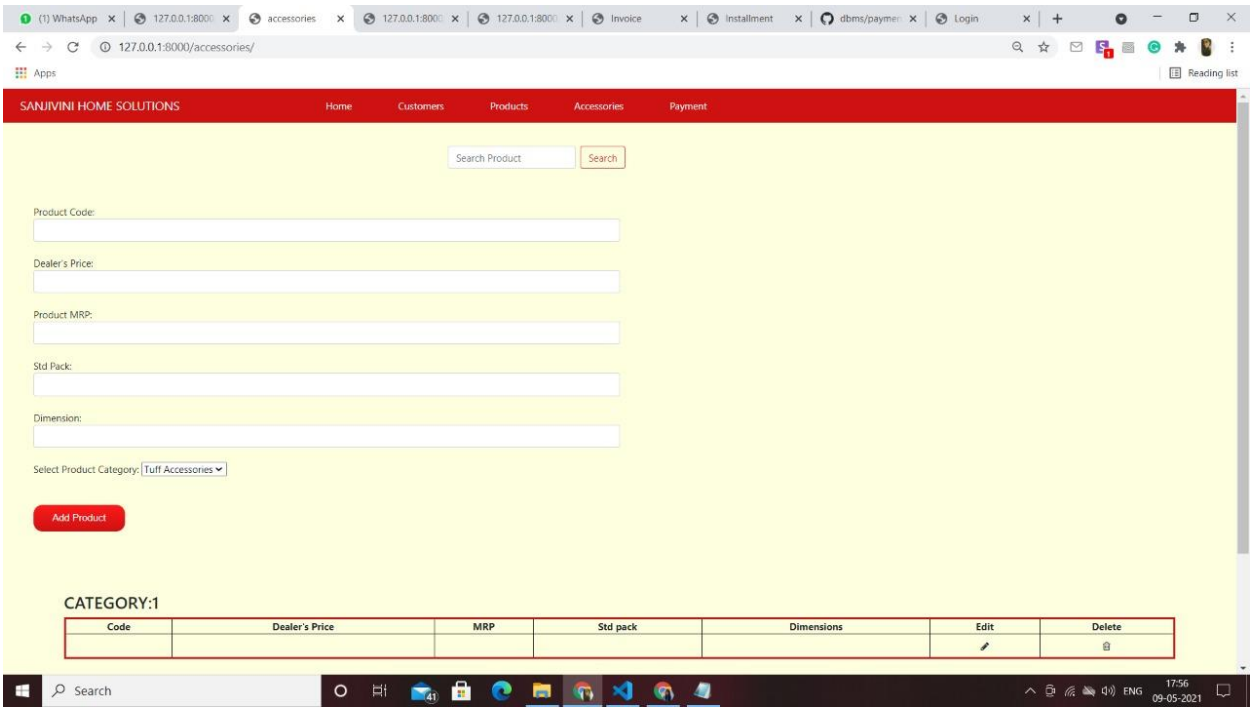




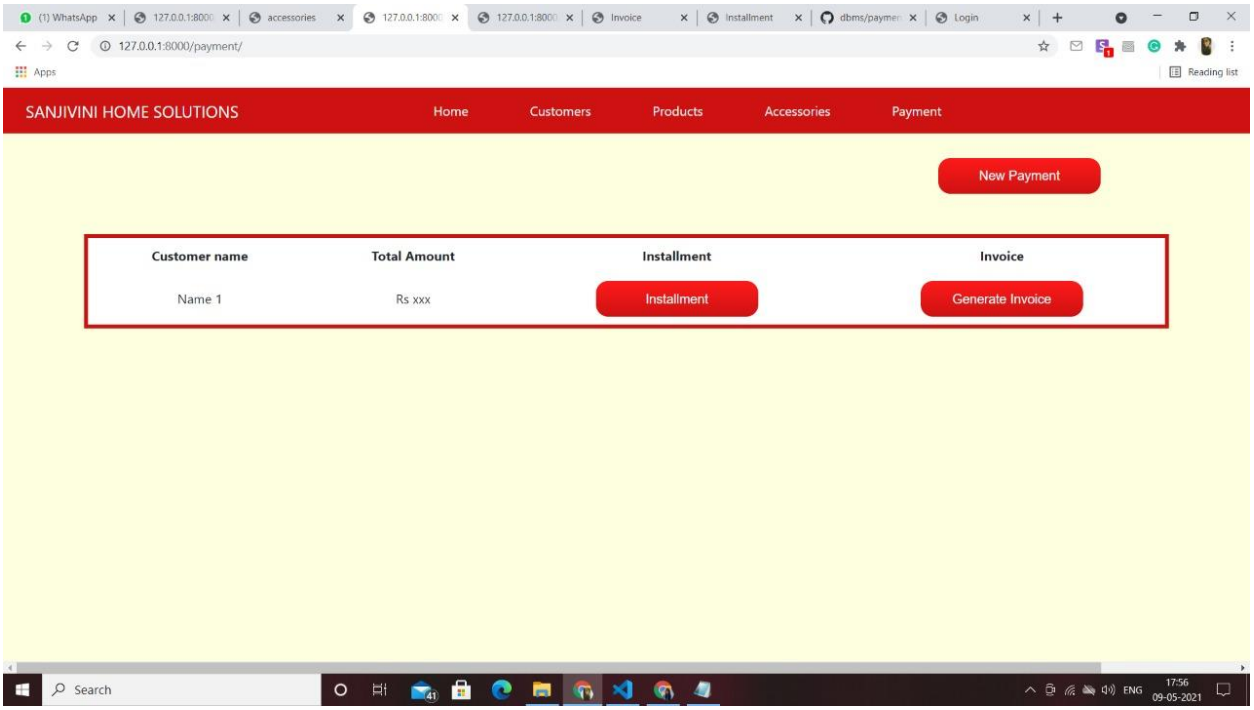
Products Page:



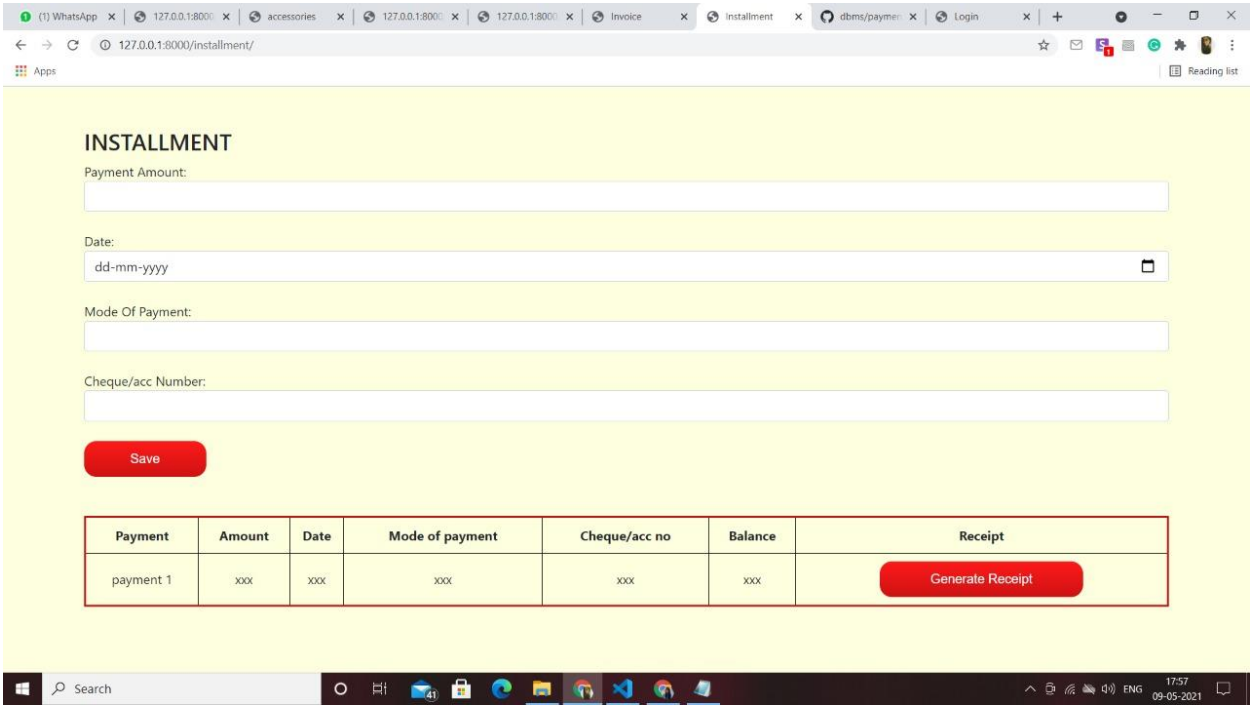
Accessories Page:



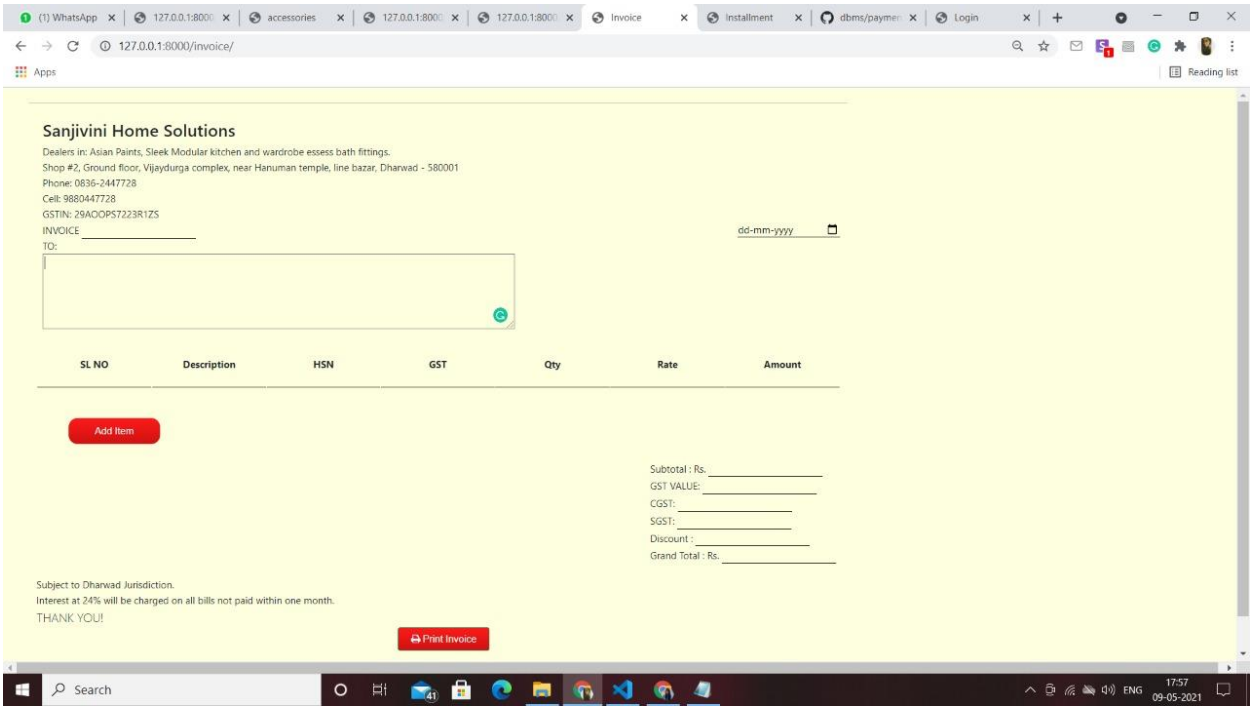
Payment Page:



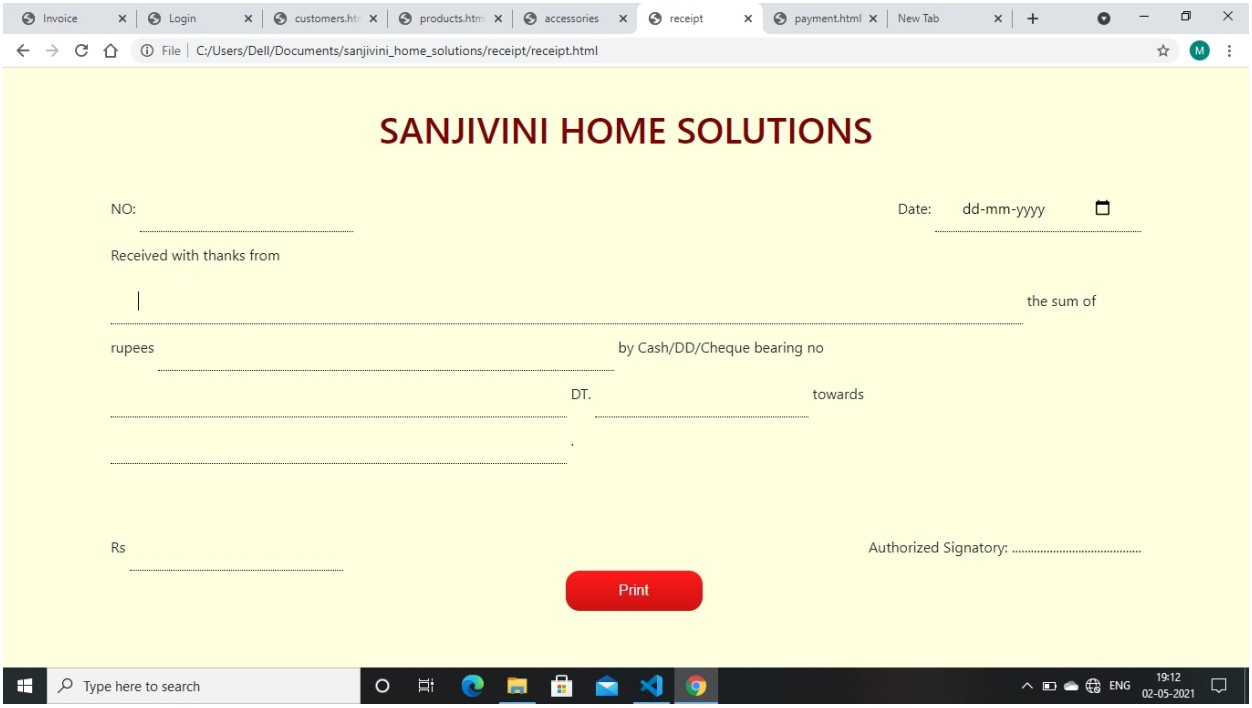
Payment Installment Page:



Invoice Page:



Receipt Page:



IMPLEMENTATION PHASE

Responsibilities:

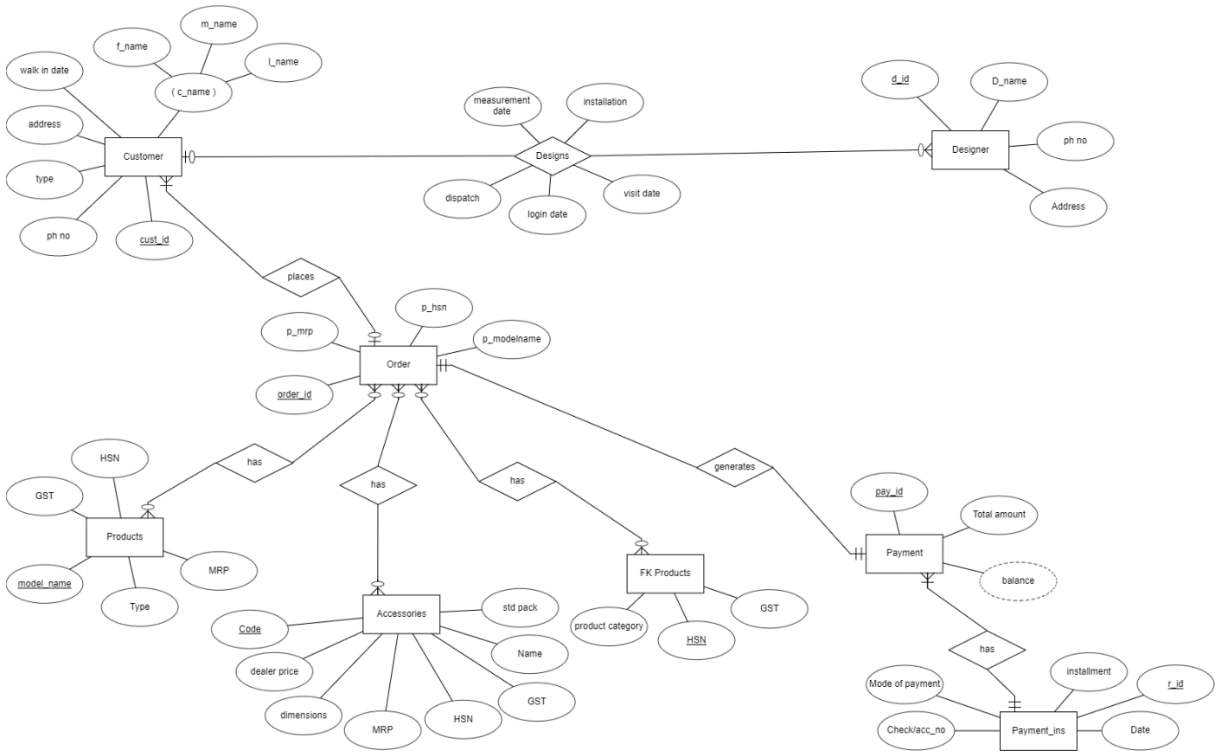
Mukti Bhansali – Backend functionalities of accessories and products which are search, add, delete and edit. Worked on orders and installment page.

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.

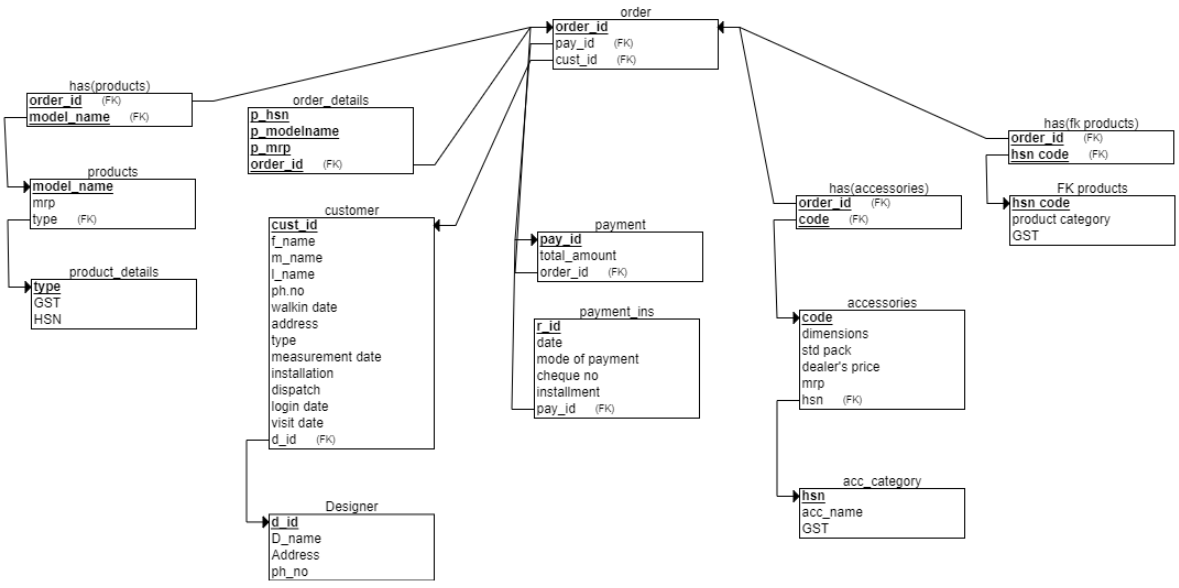
Akashini Koppad – Backend of functionalities of homepage. Worked on orders and generating receipts.

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:

id	fname	mname	lname	ph_no	address	walk_in_date	customer_type	visit_date	measure_date	login_date	dispatch_date	install_date	designer_name_id
3	pradeep	kulkarni	NULL	9844167866	Saptagiri layout Dharwad	2021-03-02	Full Kitchen Customer	2021-04-03	NULL	NULL	NULL	NULL	5
4	Ganesh	A	D	9845870818	Dharwad	2019-11-05	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	6
5	prakash	.	Siddling	8971365036	Dharwad	2019-04-01	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	5
6	vijay	.	kumar	8123999808	charantimath garden Dharwad	2019-04-03	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	5
7	Anil	.	Jain	9448221609	Gandhi chowk Dharwad	2019-04-17	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	5
8	Chandrashekar	.	.	9008777974	belagaum bypass Dharwad	2019-04-15	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	6
9	Priya	.	patil	9964699993	Balleri	2019-04-08	Full Kitchen Customer	NULL	NULL	NULL	NULL	NULL	5

Designer table:

SQL ▾ ‹ 1 / 1 › 1 - 2 of 2

id	dname	dph_no	daddress
5	Dinesh	7895436250	Gandhinagar Dharwad
6	Ashish	8890657125	4th cross road nizamuddin colony Dharwad

Product Category table:

SQL ▾ < 1 /

name	hsn	gst
Chimney	8414	18.0
Cooktops	7321	18.0
Shutters/cabinets	9403	18.0
installations	498739	18.0

Products table:

id	name	dealersPrice	mrp	category_id
1	Trina	39000.0	39990.0	Chimney
2	Terra	55900.0	56990.0	Chimney
3	Tupelo	45500.0	46990.0	Chimney
4	Brio	29800.0	30990.0	Chimney
5	Ventura	37600.0	38990.0	Chimney
6	Cypress	27660.0	29990.0	Chimney
7	Olive	21550.0	23990.0	Chimney
8	Celica	42460.0	45140.0	Chimney
9	Alpine White	34440.0	37790.0	Chimney
10	Suzy Plus	31260.0	33590.0	Chimney
11	Vitra	27000.0	28340.0	Chimney
12	Topaz	8440.0	8899.0	Cooktops
13	Jade	7440.0	7699.0	Cooktops
14	Opal	5950.0	6299.0	Cooktops
15	Amber WI	5660.0	6199.0	Cooktops
16	Emerald WI	7200.0	7599.0	Cooktops

Accessories Category table:

name	hsn	gst
tuff basket	1234	12.0
PVC Executive Cutlery Tray	39219099	18.0
Telescopic Channel	83024900	18.0
Hardware	73181500	18.0
SS Handles	83024200	18.0
Skirting System Scilm	39263090	18.0
Rolling Shutter	94034000	18.0
Lift Up Fittings	83021090	18.0
Under Sink Accessories	73239490	18.0

Accessories table:

id	code	dealersPrice	mrp	stdPack	dimensions	category_id
1	ASSTPL380405100	692.0	1010.0	6	15*16*4	tuff basket
2	ASSTPL380405150	816.0	1190.0	4	15*16*6	tuff basket
3	ASSTPL380405200	951.0	1390.0	3	15*16*8	tuff basket
4	ASSTPL380500100	797.0	1160.0	6	15*16*4	tuff basket
5	ASSTPL380500150	960.0	1400.0	4	15*16*6	tuff basket
6	ASSTPL380500200	1114.0	1620.0	3	15*16*8	tuff basket
7	ASSTPL380555100	1056.0	1540.0	4	15*16*6	tuff basket
8	ASSTPL380555150	1056.0	1540.0	4	15*16*6	tuff basket
9	ASSTPL380555200	1239.0	1810.0	3	15*16*8	tuff basket
10	ASSTPL380605100	932.0	1360.0	6	15*24*4	tuff basket
11	ASSTPL380605150	1104.0	1620.0	4	15*24*6	tuff basket
12	ASSTPL380605200	1277.0	1860.0	3	15*24*8	tuff basket
13	AMSBEC1357477	440.0	610.0	5	14*9	PVC Executive Cutlery Tray
14	AMSBEC1407477	516.0	660.0	5	16*19	PVC Executive Cutlery Tray
15	AMSBEC1457477	514.0	730.0	5	18*19	PVC Executive Cutlery Tray
16	AMSBEC1507477	616.0	750.0	5	20*19	PVC Executive Cutlery Tray
17	HICH1366	204.0	332.0	10	10"	Telescopic Channel
18	HICH1367	241.0	391.0	10	12"	Telescopic Channel
19	HICH1358	375.0	444.0	10	14	Telescopic Channel
20	HICH1355	312.0	508.0	10	16"	Telescopic Channel
21	HICH1356	350.0	572.0	10	18"	Telescopic Channel
22	HICH1359	388.0	631.0	10	20"	Telescopic Channel
23	HICH1034	449.0	728.0	10	22"	Telescopic Channel
24	HICH1035	497.0	803.0	10	22"	Telescopic Channel
25	HICH1036	553.0	899.0	10	24"	Telescopic Channel

20	HICH1355	312.0	508.0	10	16"	Telescopic Channel
21	HICH1356	350.0	572.0	10	18"	Telescopic Channel
22	HICH1359	388.0	631.0	10	20"	Telescopic Channel
23	HICH1034	449.0	728.0	10	22"	Telescopic Channel
24	HICH1035	497.0	803.0	10	22"	Telescopic Channel
25	HICH1036	553.0	899.0	10	24"	Telescopic Channel
26	HLCH1089	0.8	1.15	50	Regular	Hardware
27	HLCH1090	1.2	1.72	50	16mm	Hardware
28	HLCH1091	0.6	0.86	50	Small	Hardware
29	HLCH1092	0.9	1.29	50	Big	Hardware
30	HLCH1093	3.6	5.15	50	Regular	Hardware
31	HLCH1554	0.55	0.79	50	4*16mm	Hardware
32	HLCH1555	0.66	0.94	50	4*20mm	Hardware
33	HLCH1556	1.0	1.43	50	4*25mm	Hardware
34	HLCH1557	1.12	1.6	50	4*30mm	Hardware
35	HLCH1558	1.26	1.8	50	4*50mm	Hardware
36	HLHA3238	477.0	658.0	1	128mm	SS Handles
37	HLHA3229	288.0	398.0	1	128mm	SS Handles
38	HLHA3239	578.0	798.0	1	128mm	SS Handles
39	HLHA3230	340.0	470.0	1	128mm	SS Handles
40	HLHA3240	408.0	564.0	1	128mm	SS Handles
41	HLHA3231	531.0	733.0	1	160mm	SS Handles
42	HLHA3211	281.0	387.0	1	160mm	SS Handles
43	HLHA3210	332.0	458.0	1	160mm	SS Handles
44	HLHA3241	374.0	517.0	10	160mm	SS Handles
45	HLHA3232	408.0	564.0	1	160mm	SS Handles

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.

```
def loginview(request):
    return render(request,"login.html")

def loginuser(request):
    #user=User.objects.create_user('S','a@123','sanjivini123').save()
    username=request.POST['username']
    password=request.POST['password']

    user = authenticate(username=username, password=password)
    if user is not None:
        login(request,user)
        return redirect("homepage")
    else:

        messages.add_message(request, messages.INFO, 'Invalid login.')
        return redirect(request.META['HTTP_REFERER'])
```

For homepage:

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

```
def homepageview(request):
    fullcust=CustomerModel.objects.filter(customer_type__contains='Full').count()
    retail=CustomerModel.objects.filter(customer_type__contains='Retail').count()
    order=OrderModel.objects.count()
    params={'retail':retail,'fullcust':fullcust,'order':order}
    return render(request,'homepage.html',params)
```

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.

```
def addcustomer(request):
    form = Customer()
    if request.method == 'POST':
        form=Customer(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addcustomer.html",context)
```

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.

```
def searchcustomer(request):
    query = request.GET['searchcustomer']
    allcustomers = CustomerModel.objects.filter(fname__icontains=query)
    params ={'allcustomers':allcustomers,'query':query}
    return render(request,"searchcustomer.html",params)
```

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.

```
def editcustomer(request,pk):
    customer =CustomerModel.objects.get(id=pk)
    form = Customer(instance=customer)
    if request.method == 'POST':
        form=Customer(request.POST,instance=customer)
        if form.is_valid():
            form.save()
            return render(request,'success.html')

    context = {'form':form}
    return render(request,"addcustomer.html",context)
```

For adding product category:

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

```
def addproductcat(request):
    form = productcat()
    if request.method == 'POST':
        form=productcat(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addproductcat.html",context)
```

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.

```
def addproduct(request):
    form = product()
    if request.method == 'POST':
        form=product(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addproduct.html",context)
```

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.

```
def searchproduct(request):
    query = request.GET['searchproduct']
    allproducts = productModel.objects.filter(name__icontains=query)
    params ={'allproducts':allproducts}
    return render(request,"searchproduct.html",params)
```

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.

```
def editproduct(request,pk):
    Product =productModel.objects.get(id=pk)
    form = product(instance=Product)
    if request.method == 'POST':
        form=product(request.POST,instance=Product)
        if form.is_valid():
            form.save()
            return render(request,'homepage.html')

    context = {'form':form}
    return render(request,"addproduct.html",context)
```

For deleting product:

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

```
def deleteproduct(request,pk):
    Product =productModel.objects.get(id=pk)
    Product.delete()
    return render(request,'homepage.html')
```

For adding accessories category:

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

```
def addaccessoriescat(request):
    form = accessoriescat()
    if request.method == 'POST':
        form=accessoriescat(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addaccessoriescat.html",context)
```

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.

```
def addaccessories(request):
    form = accessories()
    if request.method == 'POST':
        form=accessories(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addaccessories.html",context)
```

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.

```
def searchaccessories(request):
    query = request.GET['searchaccessories']
    allaccessories = accessoriesModel.objects.filter(code__icontains=query)
    params ={'allaccessories':allaccessories}
    return render(request,"searchaccessories.html",params)
```

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.

```
def editaccessories(request,pk):
    accessories =accessoriesModel.objects.get(id=pk)
    form = accessories(instance=accessories)
    if request.method == 'POST':
        form=accessories(request.POST,instance=accessories)
        if form.is_valid():
            form.save()
            return render(request,'homepage.html')

    context = {'form':form}
    return render(request,"addaccessories.html",context)
```

For deleting accessories:

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

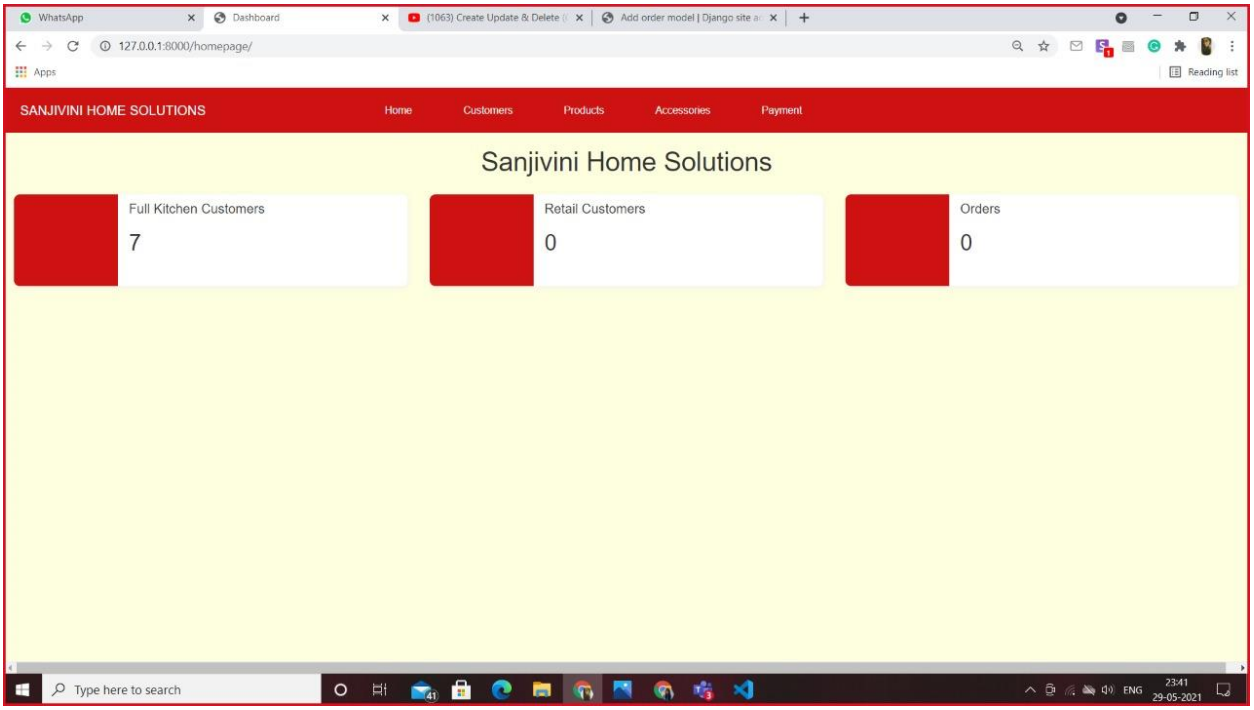
```
def deleteaccessories(request,pk):
    accessories =accessoriesModel.objects.get(id=pk)
    accessories.delete()
    return render(request,'homepage.html')
```

For adding designers:

New designers can be added to the database.

```
def addDesigner(request):
    form = Designer
    if request.method == 'POST':
        form=Designer(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    designers= DesignerModel.objects.all().order_by(Lower('dname'))
    params ={'designers':designers,'form':form}

    return render(request,"designer.html",params)
```

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:

```
def loginview(request):
    return render(request,"login.html")

def loginuser(request):
    #user=User.objects.create_user('S','a@123','sanjivini123').save()
    username=request.POST['username']
    password=request.POST['password']

    user = authenticate(username=username, password=password)
    if user is not None:
        login(request,user)
        return redirect("homepage")
    else:

        messages.add_message(request, messages.INFO, 'Invalid login.')
        return redirect(request.META['HTTP_REFERER'])
```

For homepage:

```
def homepageview(request):
    fullcust=CustomerModel.objects.filter(customer_type__contains='Full').count()
    retail=CustomerModel.objects.filter(customer_type__contains='Retail').count()
    order=OrderModel.objects.count()
    params={'retail':retail,'fullcust':fullcust,'order':order}
    return render(request,'homepage.html',params)
```

For Adding Customer:

```
def addcustomer(request):
    form = Customer()
    if request.method == 'POST':
        form=Customer(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addcustomer.html",context)
```

For searching customer:

```
def searchcustomer(request):
    query = request.GET['searchcustomer']
    allcustomers = CustomerModel.objects.filter(fname__icontains=query)
    params ={'allcustomers':allcustomers,'query':query}
    return render(request,"searchcustomer.html",params)
```

For editing customer:

```
def editcustomer(request,pk):
    customer =CustomerModel.objects.get(id=pk)
    form = Customer(instance=customer)
    if request.method == 'POST':
        form=Customer(request.POST,instance=customer)
        if form.is_valid():
            form.save()
            return render(request,'success.html')

    context = {'form':form}
    return render(request,"addcustomer.html",context)
```

For adding product category:

```
def addproductcat(request):
    form = productcat()
    if request.method == 'POST':
        form=productcat(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addproductcat.html",context)
```

For adding product:

```
def addproduct(request):
    form = product()
    if request.method == 'POST':
        form=product(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addproduct.html",context)
```

For searching product:

```
def searchproduct(request):
    query = request.GET['searchproduct']
    allproducts = productModel.objects.filter(name__icontains=query)
    params ={'allproducts':allproducts}
    return render(request,"searchproduct.html",params)
```

For editing product:

```
def editproduct(request,pk):
    Product =productModel.objects.get(id=pk)
    form = product(instance=Product)
    if request.method == 'POST':
        form=product(request.POST,instance=Product)
        if form.is_valid():
            form.save()
            return render(request,'homepage.html')

    context = {'form':form}
    return render(request,"addproduct.html",context)
```

For deleting product:

```
def deleteproduct(request,pk):
    Product =productModel.objects.get(id=pk)
    Product.delete()
    return render(request,'homepage.html')
```

For adding accessories category:

```
def addaccessoriescat(request):
    form = accessoriescat()
    if request.method == 'POST':
        form=accessoriescat(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addaccessoriescat.html",context)
```

For adding accessories:

```
def addaccessories(request):
    form = accessories()
    if request.method == 'POST':
        form=accessories(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    context ={'form':form}
    return render(request,"addaccessories.html",context)
```

For searching accessories:

```
def searchaccessories(request):
    query = request.GET['searchaccessories']
    allaccessories = accessoriesModel.objects.filter(code__icontains=query)
    params ={'allaccessories':allaccessories}
    return render(request,"searchaccessories.html",params)
```

For editing accessories:

```
def editaccessories(request,pk):
    accessories =accessoriesModel.objects.get(id=pk)
    form = accessories(instance=accessories)
    if request.method == 'POST':
        form=accessories(request.POST,instance=accessories)
        if form.is_valid():
            form.save()
            return render(request,'homepage.html')

    context = {'form':form}
    return render(request,"addaccessories.html",context)
```

For deleting accessories:

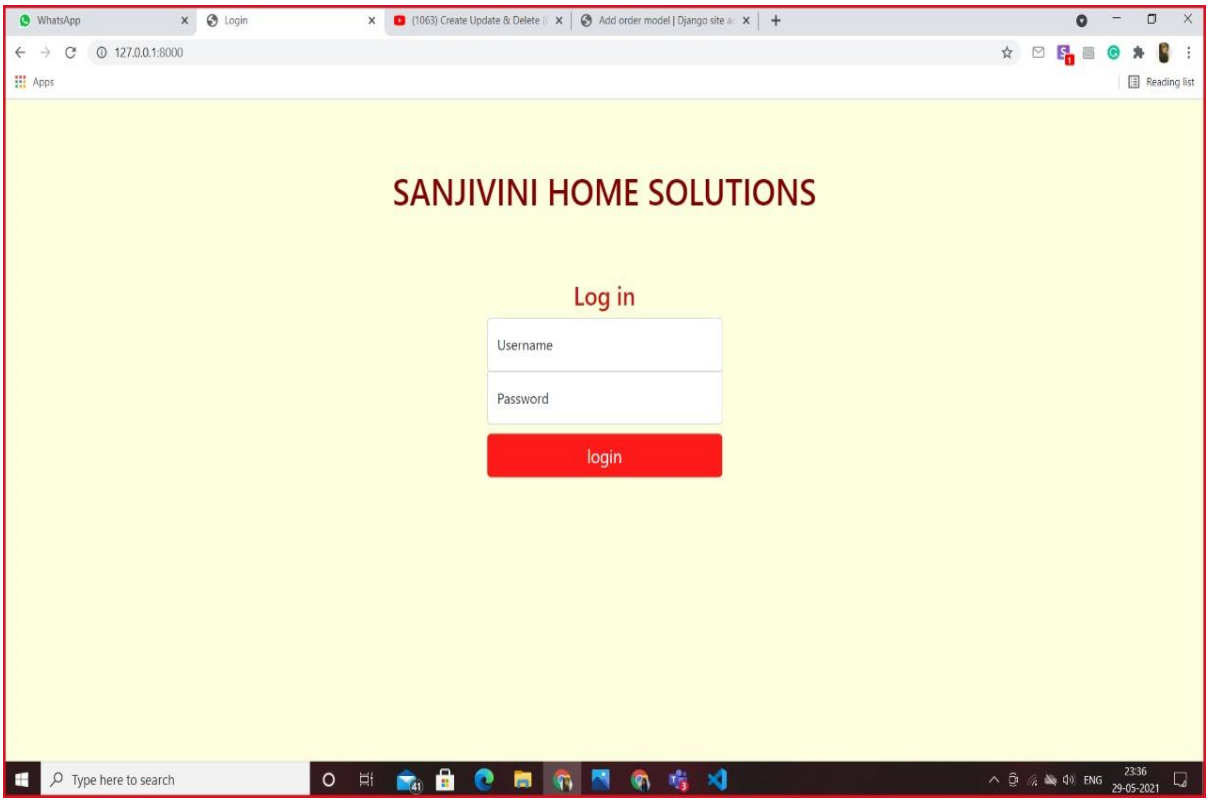
```
def deleteaccessories(request,pk):
    accessories =accessoriesModel.objects.get(id=pk)
    accessories.delete()
    return render(request,'homepage.html')
```

For adding designers:

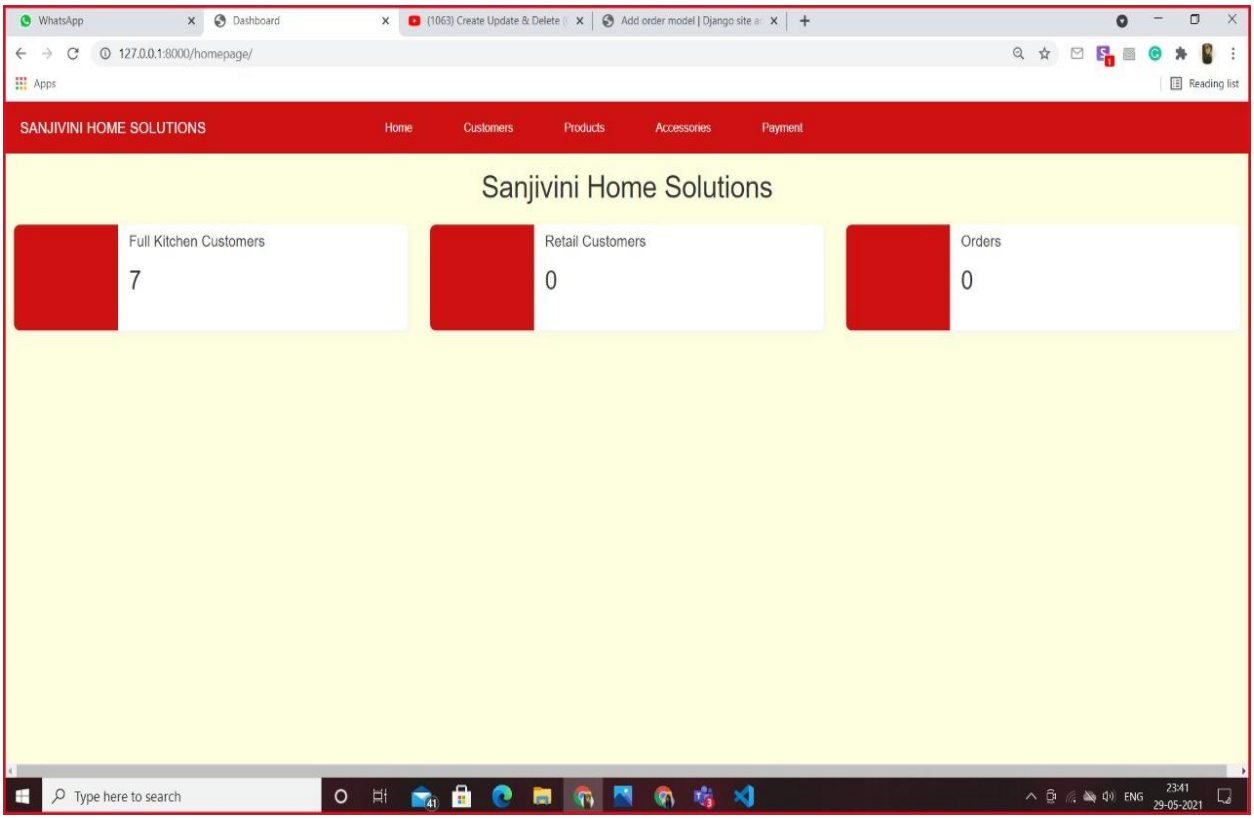
```
def addDesigner(request):
    form = Designer
    if request.method == 'POST':
        form=Designer(request.POST)
        if form.is_valid():
            form.save()
            return redirect(request.META['HTTP_REFERER'])

    designers= DesignerModel.objects.all().order_by(Lower('dname'))
    params ={'designers':designers,'form':form}

    return render(request,"designer.html",params)
```

User Interface Screenshots:

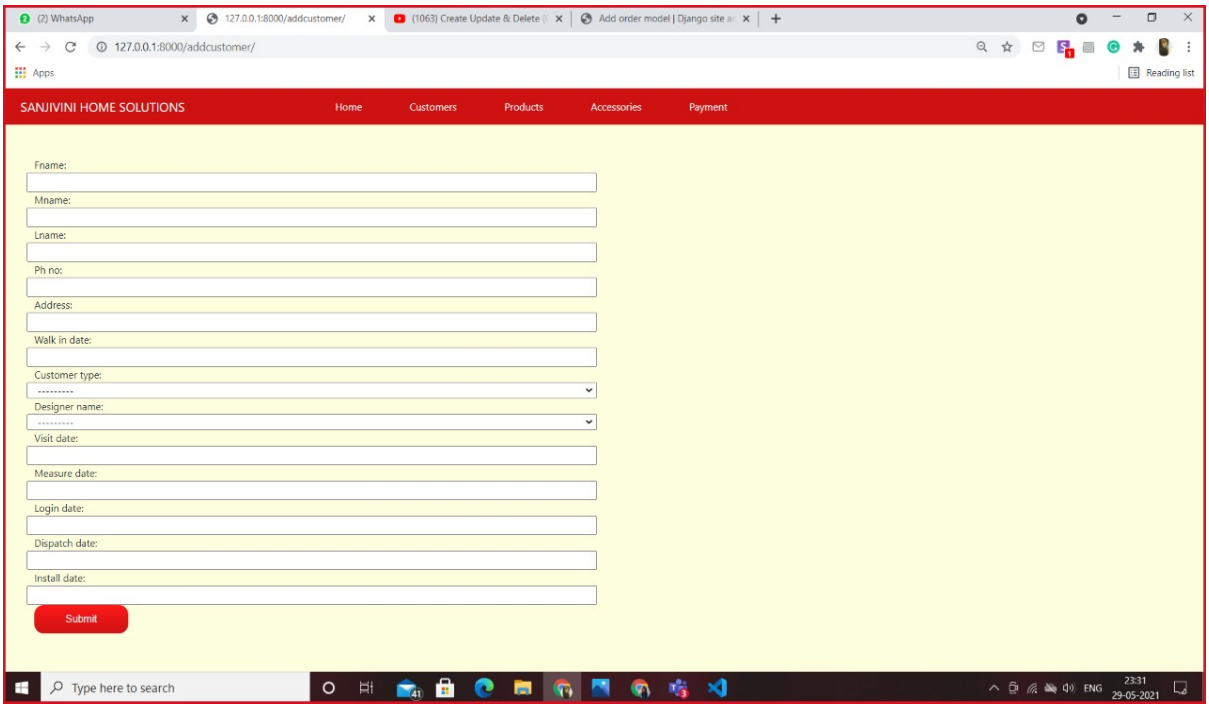
Login page:



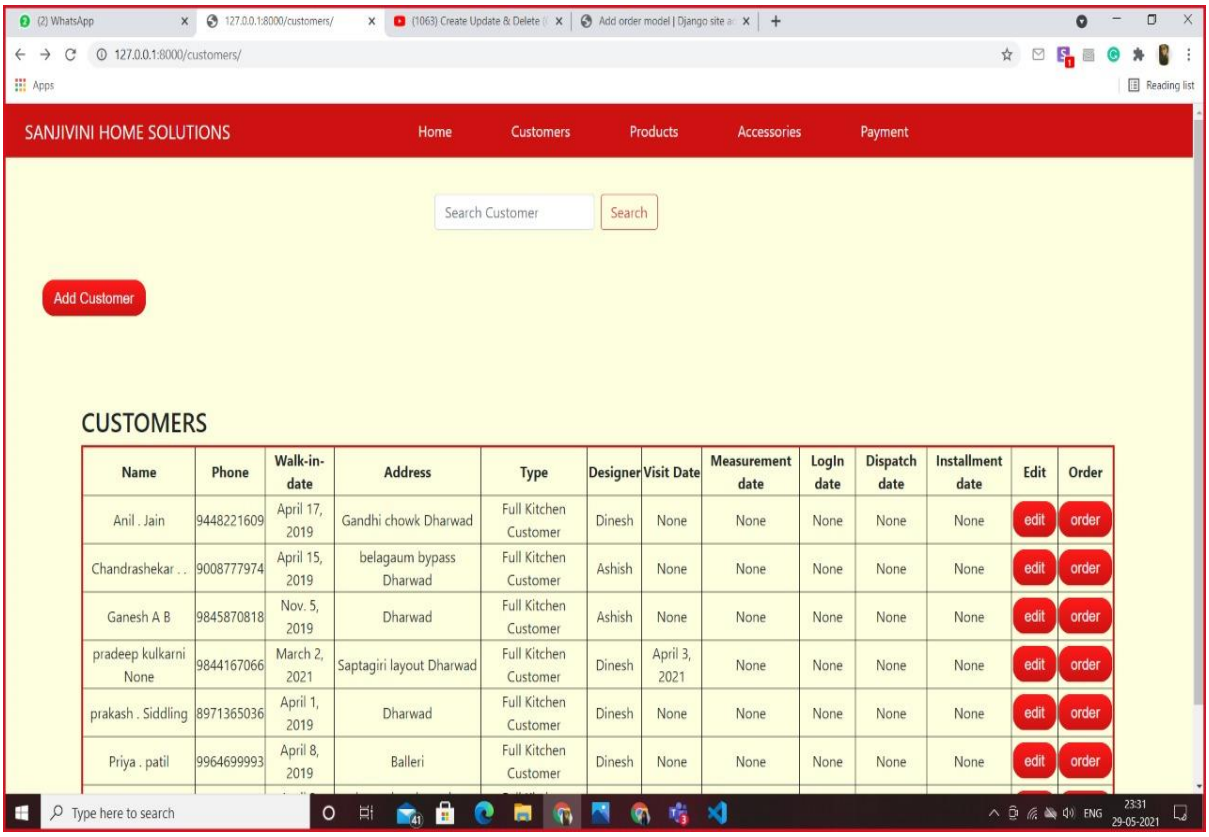
Homepage:



Add customer page:

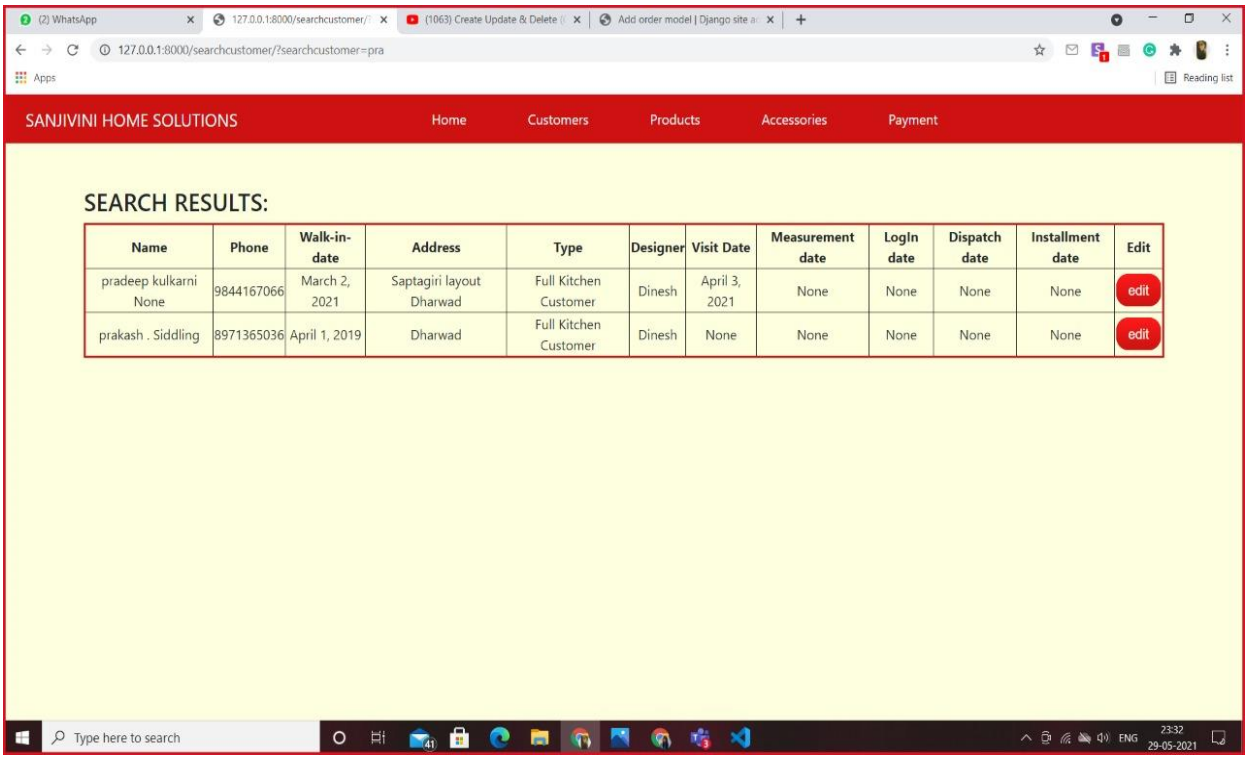


Customer page:

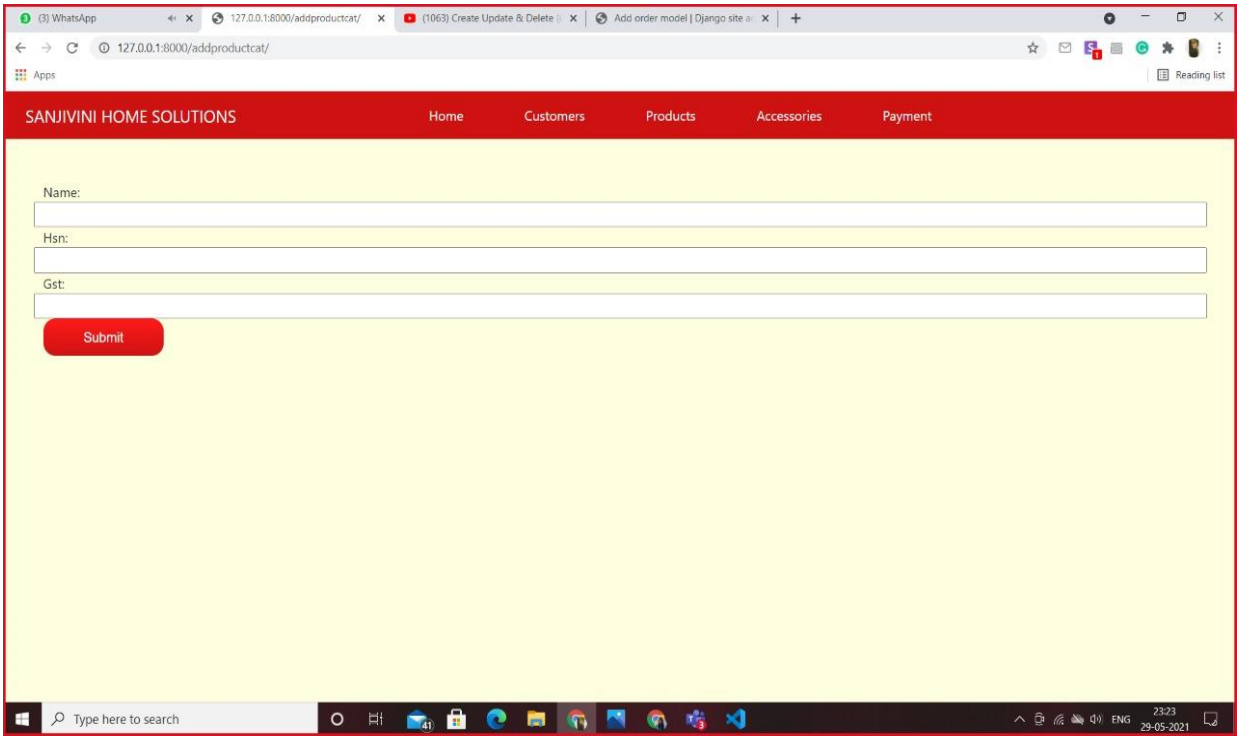


Name	Phone	Walk-in-date	Address	Type	Designer	Visit Date	Measurement date	LogIn date	Dispatch date	Installment date	Edit	Order
Anil . Jain	9448221609	April 17, 2019	Gandhi chowk Dharwad	Full Kitchen Customer	Dinesh	None	None	None	None	None	edit	order
Chandrashekar . .	9008777974	April 15, 2019	belagaum bypass Dharwad	Full Kitchen Customer	Ashish	None	None	None	None	None	edit	order
Ganesh A B	9845870818	Nov. 5, 2019	Dharwad	Full Kitchen Customer	Ashish	None	None	None	None	None	edit	order
pradeep kulkarni None	9844167066	March 2, 2021	Saptagiri layout Dharwad	Full Kitchen Customer	Dinesh	April 3, 2021	None	None	None	None	edit	order
prakash . Siddling	8971365036	April 1, 2019	Dharwad	Full Kitchen Customer	Dinesh	None	None	None	None	None	edit	order
Priya . patil	9964699993	April 8, 2019	Balleri	Full Kitchen Customer	Dinesh	None	None	None	None	None	edit	order

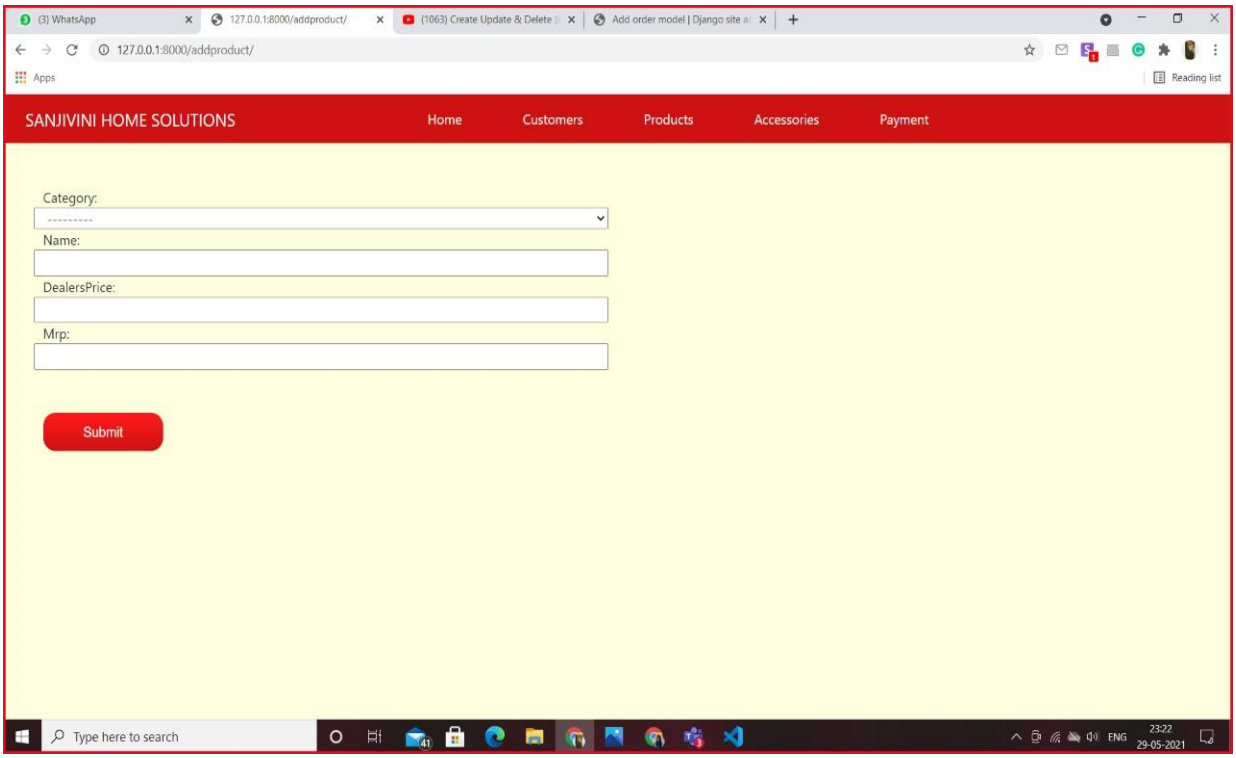
Search customer page:



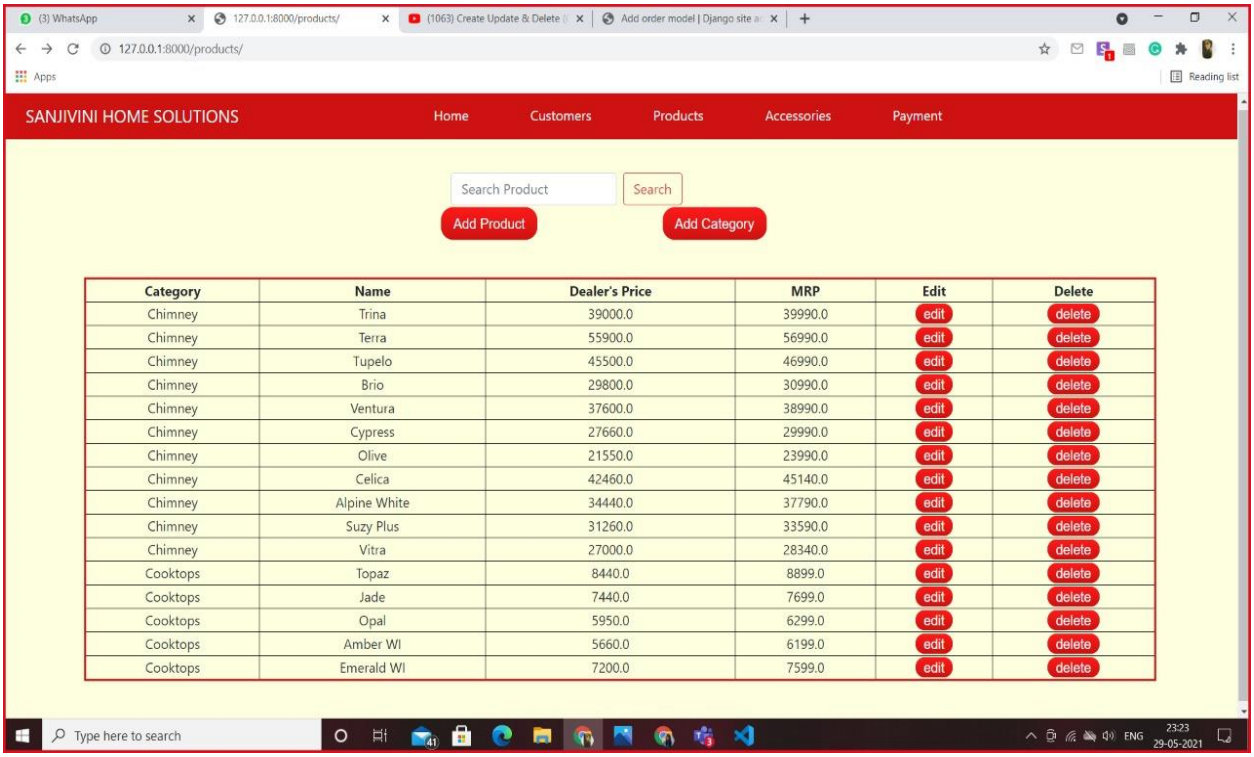
Add product category page:



Add product page:

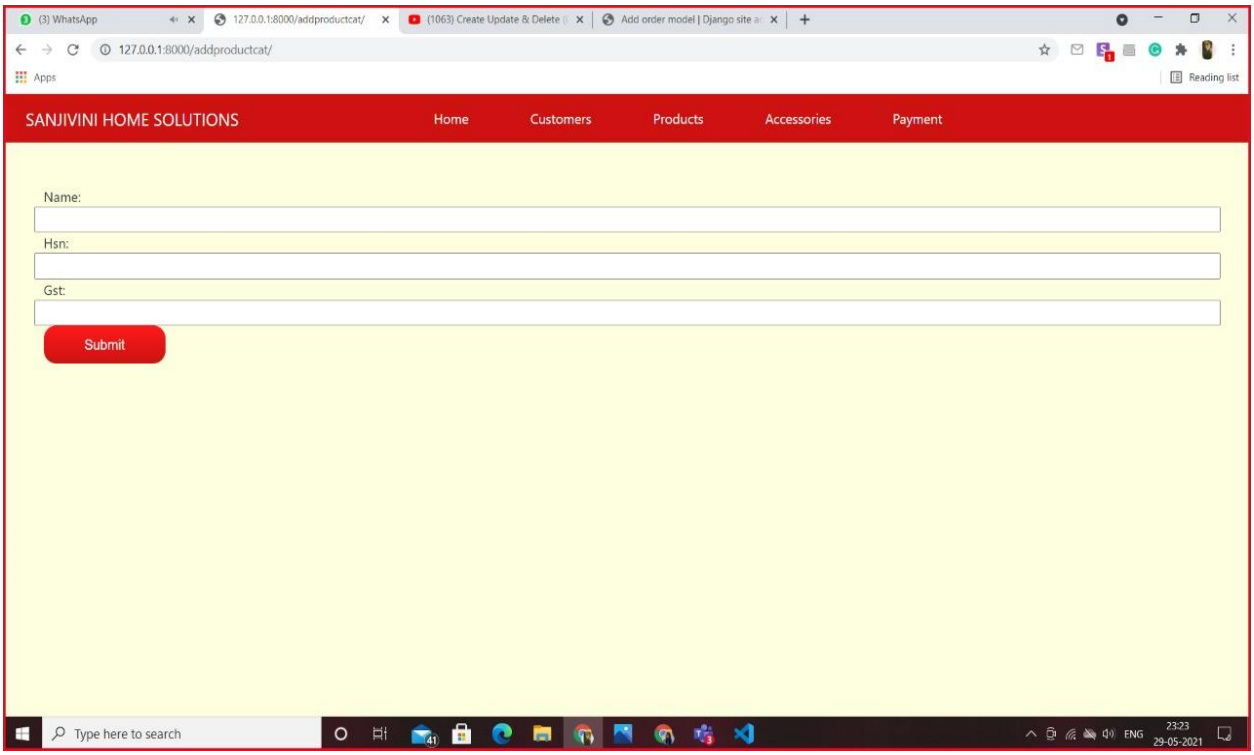


Products page:

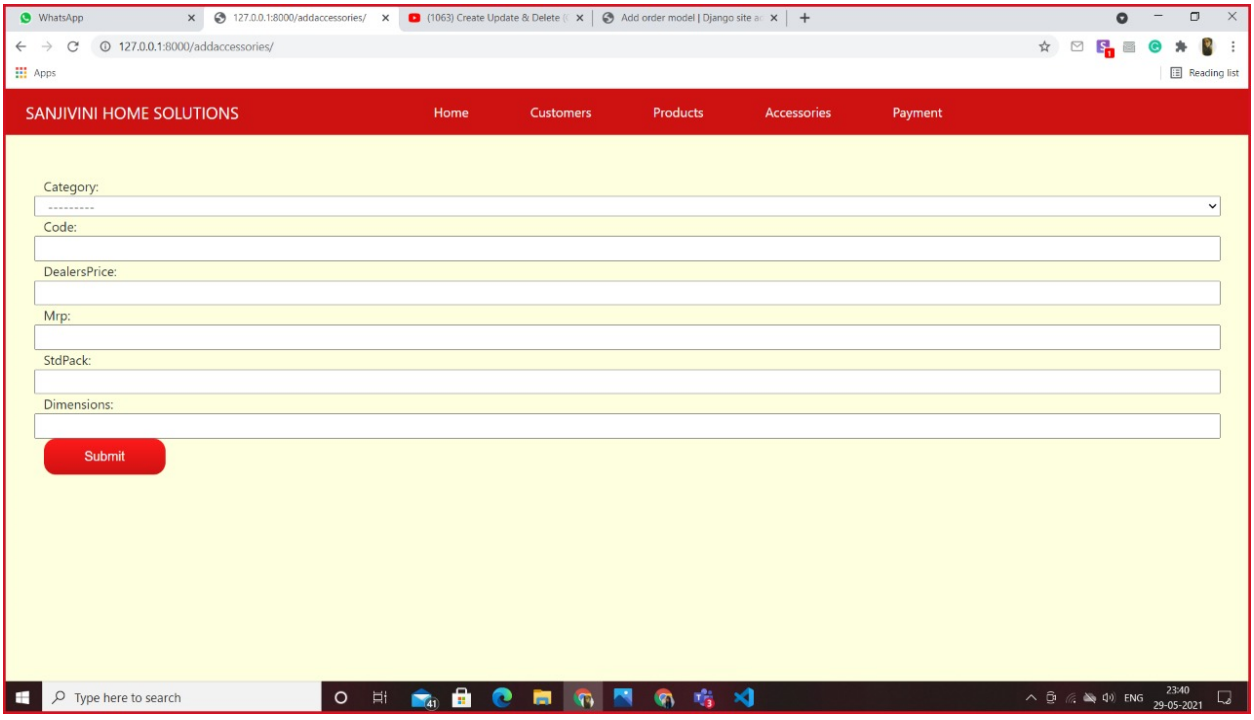


Category	Name	Dealer's Price	MRP	Edit	Delete
Chimney	Trina	39000.0	39990.0	edit	delete
Chimney	Terra	55900.0	56990.0	edit	delete
Chimney	Tupelo	45500.0	46990.0	edit	delete
Chimney	Brio	29800.0	30990.0	edit	delete
Chimney	Ventura	37600.0	38990.0	edit	delete
Chimney	Cypress	27660.0	29990.0	edit	delete
Chimney	Olive	21550.0	23990.0	edit	delete
Chimney	Celica	42460.0	45140.0	edit	delete
Chimney	Alpine White	34440.0	37790.0	edit	delete
Chimney	Suzy Plus	31260.0	33590.0	edit	delete
Chimney	Vitra	27000.0	28340.0	edit	delete
Cooktops	Topaz	8440.0	8899.0	edit	delete
Cooktops	Jade	7440.0	7699.0	edit	delete
Cooktops	Opal	5950.0	6299.0	edit	delete
Cooktops	Amber WI	5660.0	6199.0	edit	delete
Cooktops	Emerald WI	7200.0	7599.0	edit	delete

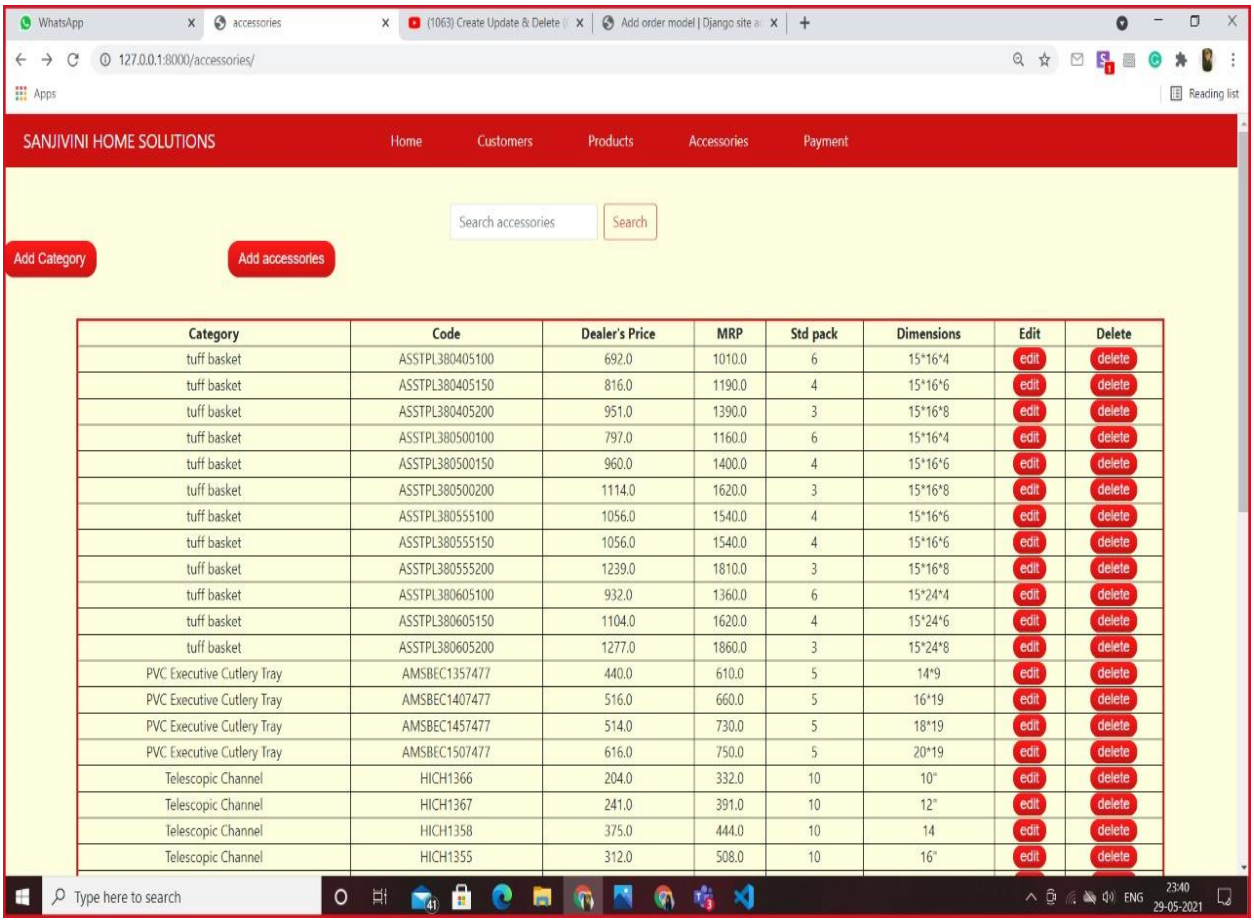
Add accessories category page:



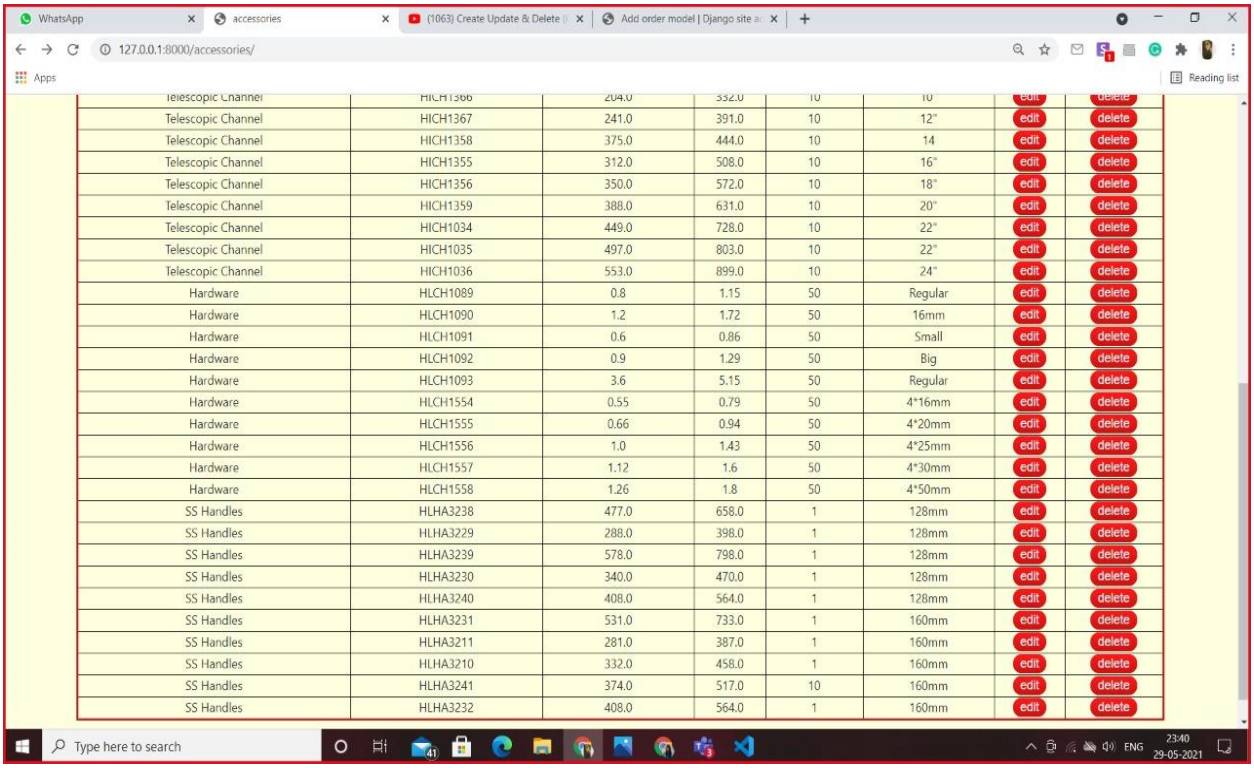
Add accessories page:



Accessories page:



Category	Code	Dealer's Price	MRP	Std pack	Dimensions	Edit	Delete
tuff basket	ASSTPL380405100	692.0	1010.0	6	15*16*4	edit	delete
tuff basket	ASSTPL380405150	816.0	1190.0	4	15*16*6	edit	delete
tuff basket	ASSTPL380405200	951.0	1390.0	3	15*16*8	edit	delete
tuff basket	ASSTPL380500100	797.0	1160.0	6	15*16*4	edit	delete
tuff basket	ASSTPL380500150	960.0	1400.0	4	15*16*6	edit	delete
tuff basket	ASSTPL380500200	1114.0	1620.0	3	15*16*8	edit	delete
tuff basket	ASSTPL380555100	1056.0	1540.0	4	15*16*6	edit	delete
tuff basket	ASSTPL380555150	1056.0	1540.0	4	15*16*6	edit	delete
tuff basket	ASSTPL380555200	1239.0	1810.0	3	15*16*8	edit	delete
tuff basket	ASSTPL380605100	932.0	1360.0	6	15*24*4	edit	delete
tuff basket	ASSTPL380605150	1104.0	1620.0	4	15*24*6	edit	delete
tuff basket	ASSTPL380605200	1277.0	1860.0	3	15*24*8	edit	delete
PVC Executive Cutlery Tray	AMSBEC1357477	440.0	610.0	5	14*9	edit	delete
PVC Executive Cutlery Tray	AMSBEC1407477	516.0	660.0	5	16*19	edit	delete
PVC Executive Cutlery Tray	AMSBEC1457477	514.0	730.0	5	18*19	edit	delete
PVC Executive Cutlery Tray	AMSBEC1507477	616.0	750.0	5	20*19	edit	delete
Telescopic Channel	HICH1366	204.0	332.0	10	10"	edit	delete
Telescopic Channel	HICH1367	241.0	391.0	10	12"	edit	delete
Telescopic Channel	HICH1358	375.0	444.0	10	14	edit	delete
Telescopic Channel	HICH1355	312.0	508.0	10	16"	edit	delete


WhatsApp
accessories
(1063) Create Update & Delete
Add order model | Django site
127.0.0.1:8000/accessories/
Apps
Reading list
Telescopic Channel HICH1366 204.0 332.0 10 10 edit delete
Telescopic Channel HICH1367 241.0 391.0 10 12" edit delete
Telescopic Channel HICH1358 375.0 444.0 10 14 edit delete
Telescopic Channel HICH1355 312.0 508.0 10 16" edit delete
Telescopic Channel HICH1356 350.0 572.0 10 18" edit delete
Telescopic Channel HICH1359 388.0 631.0 10 20" edit delete
Telescopic Channel HICH1034 449.0 728.0 10 22" edit delete
Telescopic Channel HICH1035 497.0 803.0 10 22" edit delete
Telescopic Channel HICH1036 553.0 899.0 10 24" edit delete
Hardware HLCH1089 0.8 1.15 50 Regular edit delete
Hardware HLCH1090 1.2 1.72 50 16mm edit delete
Hardware HLCH1091 0.6 0.86 50 Small edit delete
Hardware HLCH1092 0.9 1.29 50 Big edit delete
Hardware HLCH1093 3.6 5.15 50 Regular edit delete
Hardware HLCH1554 0.55 0.79 50 4*16mm edit delete
Hardware HLCH1555 0.66 0.94 50 4*20mm edit delete
Hardware HLCH1556 1.0 1.43 50 4*25mm edit delete
Hardware HLCH1557 1.12 1.6 50 4*30mm edit delete
Hardware HLCH1558 1.26 1.8 50 4*50mm edit delete
SS Handles HLHA3238 477.0 658.0 1 128mm edit delete
SS Handles HLHA3229 288.0 398.0 1 128mm edit delete
SS Handles HLHA3239 578.0 798.0 1 128mm edit delete
SS Handles HLHA3230 340.0 470.0 1 128mm edit delete
SS Handles HLHA3240 408.0 564.0 1 128mm edit delete
SS Handles HLHA3231 531.0 733.0 1 160mm edit delete
SS Handles HLHA3211 281.0 387.0 1 160mm edit delete
SS Handles HLHA3210 332.0 458.0 1 160mm edit delete
SS Handles HLHA3241 374.0 517.0 10 160mm edit delete
SS Handles HLHA3232 408.0 564.0 1 160mm edit delete
Type here to search
ENG
23:40
29-05-2021

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

Sanjivini Home Solutions

Near Hanuman Temple,Line Bazar, Dharwad-580001

Ph No.9880447728

Date:17-04-2021

To,

Mr. Shankar Gangisetty.

(Project Guide), Div-D

KLE Technological University, Hubballi

Subject: Grant of Permission to use our data for the project

Respected sir,

I Dinakar G Shetty of Sanjivini Home Solutions declare that we accept and authorize Mukti Bhansali (01fe19bcs203),Manisha Belagal (01fe19bcs207),Mehar Anjum Soudagar (01fe19bcs208),Akashini Koppad (01fe19bcs210) from Department of Computer Science, KLE Technological University, Hubballi, to use our information and related data for building database system.

Therefore, I grant the permission to the above mentioned names to work under you declaring myself as a client for this project.

Best Regards,

SANJIVINI HOME SOLUTIONS
Line Bazar, Near Hanuman Temple,
DHARWAD - 580 001

Signature

(Dinakar G Shetty)

Ph No.9880447728