**Chapter-1**

# INTRODUCTION

## Data masking is a way to create a fake, but a realistic version of your organizational data. The goal is to protect sensitive data, while providing a functional alternative when real data is not needed—for example, in user training, sales demos, or software testing.

Data masking processes change the values of the data while using the same format. The goal is to create a version that cannot be deciphered or reverse engineered. There are several ways to alter the data, including character shuffling, word or character substitution, and encryption.

## 1.1 Importance of Data Masking

Here are several reasons data masking is essential for many organizations:

* Data masking solves several critical threats – data loss, data exfiltration, insider threats or account compromise, and insecure interfaces with third party systems.
* Reduces data risks associated with cloud adoption.
* Makes data useless to an attacker, while maintaining many of its inherent functional properties.
* Allows sharing data with authorized users, such as testers and developers, without exposing production data.
* Can be used for [data sanitization](https://www.imperva.com/learn/data-security/data-sanitization/) – normal file deletion still leaves traces of data in storage media, while sanitization replaces the old values with masked ones.

## 1.2 Data Masking Types

### 1.2.1 Static Data Masking

Static data masking processes can help you create a sanitized copy of the database. The process alters all sensitive data until a copy of the database can be safely shared. Typically, the process involves creating a backup copy of a database in production, loading it to a separate environment, eliminating any unnecessary data, and then masking data while it is in stasis. The masked copy can then be pushed to the target location.

Imperva partners with Mage Static Data Masking to deliver SDM capabilities to Imperva customers. While Imperva Data Security Fabric (DSF) provides real-time protection of live production data, Mage™ de-identifies data in non-production environments. Mage™ brings a static data masking capability that complements Imperva DSF, works across multiple data platforms, and supports flexible deployment mechanisms to integrate seamlessly into the existing enterprise IT framework without the need for any additional architectural changes

## 1.2.2 Deterministic Data Masking

## Involves mapping two sets of data that have the same type of data, in such a way that one value is always replaced by another value. For example, the name “John Smith” is always replaced with “Jim Jameson”, everywhere it appears in a database. This method is convenient for many scenarios but is inherently less secure.

### 1.2.3 On-the-Fly Data Masking

Masking data while it is transferred from production systems to test or development systems before the data is saved to disk. Organizations that deploy software frequently cannot create a backup copy of the source database and apply masking—they need a way to continuously stream data from production to multiple test environments.

On the fly, masking sends smaller subsets of masked data when it is required. Each subset of masked data is stored in the dev/test environment for use by the non-production system.

It is important to apply on-the-fly masking to any feed from a production system to a development environment, at the very beginning of a development project, to prevent compliance and security issues.

## 1.5 Html

## Html is a client-side web application known as a hypertext markup language used to design the landing and its navigation pages for a particular web application, it consists of tags to define the different sections of the front page which inform the browser to display the sections in a well structured and formatted way.

## 1.5 Css

CSS is known as cascading stylesheets used for styling HTML pages, it consists of properties such as selectors, pseudo-classes, backgrounds, float, borders, colors, height/widths, and animations which makes designing the website layout and icons very impressive. It is a cornerstone framework of WWB which delivers the webpage presentable.

## 1.5 Sqllite

It is the most commonly used database engine in the backend web framework, It is the most reliable, stable, and cross-platform for linking the client-side Frameworks for serverside applications, This C-language library can be used across all android and computer devices which provides the flexibility

**Summary:** This chapter gives a brief introduction to the project and the technology stack used in this project

**Chapter-2**

# LITERATURE REVIEW AND OBJECTIVES

## 2.2 Short review on Electronic Commerce

This has become the most sought-after strategy for all business firms to sell their products through digital platforms. This idea has become a game changer for the economy after 1990. As per the definition of the world trade organization in 1998, electronic commerce means the production, marketing, and sale of goods and products through the support of digital media.

## 2.3 Django application in web browsers

Django framework can be used to manipulate the number of views and hits on the website. It stores the data in its database previously browsed by the user to give recommendations to the user based on their searches on the site. It maintains the data load on the servers. It has a huge database to maintain the user’s data and choices based on their searches.

**2.4 Django application in the weather forecast**

During the weather forecast, physical parameters such as temperature, humidity, Pressure, wind speed, wind direction, and precipitation are processed previously and stored in the database of the Django framework to predict the meteorological condition of the next day, It also utilizes some machine learning algorithms to conclude the current climatic situation and the next day’s weather condition by using the data calculated using computer simulations and data assimilation process.

## 2.5 Django application in GPS tracking

One of the popular applications of Django is GPS tracking. GeoDjango is one such popular web framework used to build GIS applications, Django model vigilantes the OGC geometries and designs the rectangular pattern of parallel scanning lines of the current geographical area by using an electron beam of the computer screen. For handling the spatial information of the location, Django has the ORM framework which is used for manipulating the models and features of the models and querying. It provides the flexibility to use high-level python interfacing for GIS geometry and editing the location features for the admin of the project.

## 2.5 Django applications in web frameworks

Django finds an important application in developing back-end web frameworks. It uses MVT architecture to configure and organize the files in a proper way, It stores the information entered by the user from the front-side of the web framework to the database and provides the flexibility for the project admin to update the data in the back-end. Some of the important web applications that can be built by Django frameworks are e-commerce websites, grocery websites, educational websites, restaurant websites, etc……

* 1. **Objectives**

1.To check the authorization of the members of a particular group using a database table.

2.Digital polling through the network.

3.Provides quick access and good facilities for the system that could not be done manually.

4.Reduces human interference in the file/database management system.

5.Minimizes the risk of low accuracy by preventing manual work.

6.Digital survey system.

7.Maintains the track record of the data that has been used before.

**Chapter-3**

# PROBLEM STATEMENT

Aims to develop an e-commerce grocery website that helps to revive the small-scale economy and the daily needs of a customer at their doorsteps, In order to provide flexibility for both admin and end users of the product this idea came up to provide a digital interface that provides the 24/7 web service to the Shopvendors to sell their products online anytime throughout the year and the customers to place their orders on the web platform and delivered at their doorsteps.

**Chapter-4**

# SYSTEM DESIGN

**4.1 Software development life cycle (SDLC)**

Software Development is the development of software for distinct purposes. For software development, there are many programming languages like Python, Java ,C++ etc… The entire process of software development is not a simple process as its definition it is a complicated process it requires the best approach for the develop a product

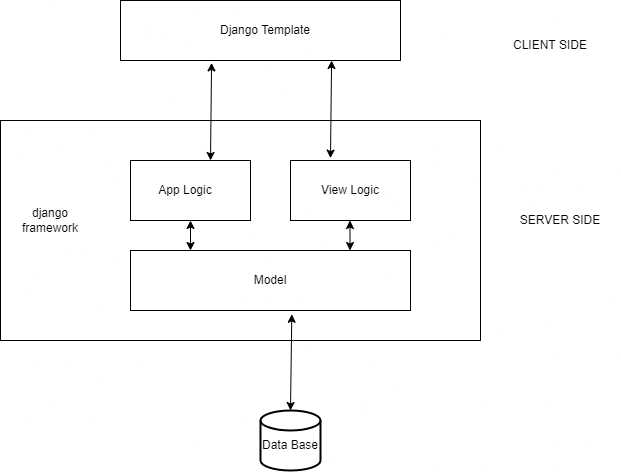
**The SDLC includes 6 stages:**

1. Planning And Requirement Analysis:
2. Defining Requirements
3. Designing Architecture
4. Developing Product
5. Product Testing and Integration
6. Documentation, Training and Support

System design talks about the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. Quality system design is essential in developing functional and lasting applications.

**4.2 Architecture Diagram**

An architecture diagram is a graphical representation of a set of concepts ,that are part of an architecture of the system including their principles, elements and components.



**4.2.1 Client Side:** User is a relationship which one program request a service or resource

From another program

* + 1. **Admin.py:** Admin ownership file where apps get registered

**4.2.3 Models Module:** The models created for the purpose of storage of data from the users

Through apis.

**4.2.4 Templates Module**: It is a document, or a python string marked up using the Django

Template module.

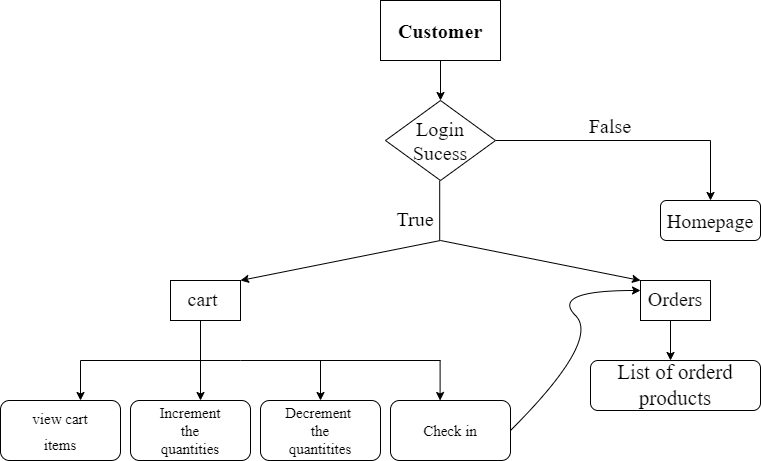
**DB Module:** Stores the data passed down from models’ module and sends the data back whenever required.

* + 1. **Frontend Interface:** User will be able to see the outputs of the operations through this module in the form of text or map graphics.

## 4.3 Flowchart

An algorithm is graphically represented by a flowchart. It is frequently used by programmers as a technique for planning programs to address issues. It uses interconnected symbols to represent the movement of information and processing. "Flowcharting" is the process of creating a flowchart for an algorithm.

Figure 4.3 shows the project application’s flowchart.



*Figure 4.3 Flowchart*

### 

1. Home page contains core modules like store , signup , cart ,login and logout. Registration and generates an access token that could be used to authenticate users, As only authenticated users are allowed to place an order and buy from the website

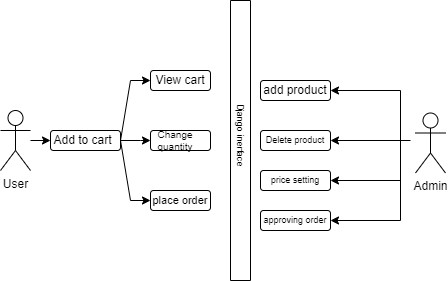
2.If the user doesn’t have an account he/she can create a new account by signup module

3.In the orders tab the customer can see the products placed by him/her.

## 4.4 Use case Diagram

The scope and high-level functions of a system are described pictorially in use case diagram. The interactions between the system and its actors are also represented in this diagram. But the functionality of the system is abstracted for the actors.

Figure 4.4 shows the use case diagram of the project.



***Figure 4.4 Use Case Diagram***

**Chapter-5**

# IMPLEMENTATION

Here we used Django’s default admin page to build this web application, we test this with different users’ credentials and verified the data retention in the database.

## 5.1 Algorithm

An algorithm is a sequence of unambiguous instructions for solving the given problem that is obtaining the required output for any legitimate input in a finite amount of time.

The algorithm defined for our project is as follows-

### Algorithm 8.1: Ecommerce website

|  |
| --- |
| Input: User Details (Email, Password, Phone number, Name, etc…)# for login Output: Order Details and user’s cart If user:  do:  Step 1: The user must Complete the Registration  Step 2: if (new user):#sign up  Get the user credentials and store the credentials in the database Redirecting to home page Else:  If user data present in database :  Login successful  Display(orders , cart)  Allowing the user to buy the products  Else:  Login fail  Redirect to login page  Asking for proper credentials to authorize the user Step 3:Logout Else:  Ask for the credentials to the admin to perform the CRUD operations in the database |

## 5.2 Important Code Snippets

### 5.2.1 Configuration of the urls

|  |
| --- |
| from django.urls import path  from .views.home import Index, store  from .views.signup import Signup from .views.login import Login, logout from.views.cart import Cart from .views.checkout import CheckOut from.views.orders import OrderView from middlewares.auth import auth\_middleware  urlpatterns = [path('', Index.as\_view(),  path('store/', store, name='store'), path('signup/', Signup.as\_view(), name='signup'), path('login/', Login.as\_view(), name='login'), path('logout/', logout, name='logout'),  path('cart/', auth\_middleware(Cart.as\_view()), name='cart'), path('check-out/', CheckOut.as\_view(), name='checkout'), path('orders/',  auth\_middleware(OrderView.as\_view()), name='orders'),  ] |

**5.2.2 Creating Category Model**

|  |
| --- |
| from django.db import models  Class category(models.model)  name = models.CharField(max\_length=20)  @staticmethod  def get\_all\_categories():  return Category.objects.all()  def \_\_str\_\_(self):  return self.name |

|  |
| --- |
| from django.shortcuts import render, redirect from django.views import View from store.models.product import Product  class Cart(View):  def get(self, request):  ids = list(request.session.get('cart').keys()) products = Product.get\_products\_by\_id(ids) print(products)  return render(request,'cart.html',{'products': products}) |

### 5.2.3 Creating class based view cart in view.py

### 

**Chapter-6**

# TESTING

Software Testing is the procedure to evaluate the software products and application. the major benefits of are to search for bug and errors. The procedure solving a bug is called as Debugging. As well as the major benefit is to Improve the performance of the application. it typically examines the functionality of an application. There are mainly two steps in software testing they are verification and validation. In the First Step we Verify the Software and in Second we Validate the Software.

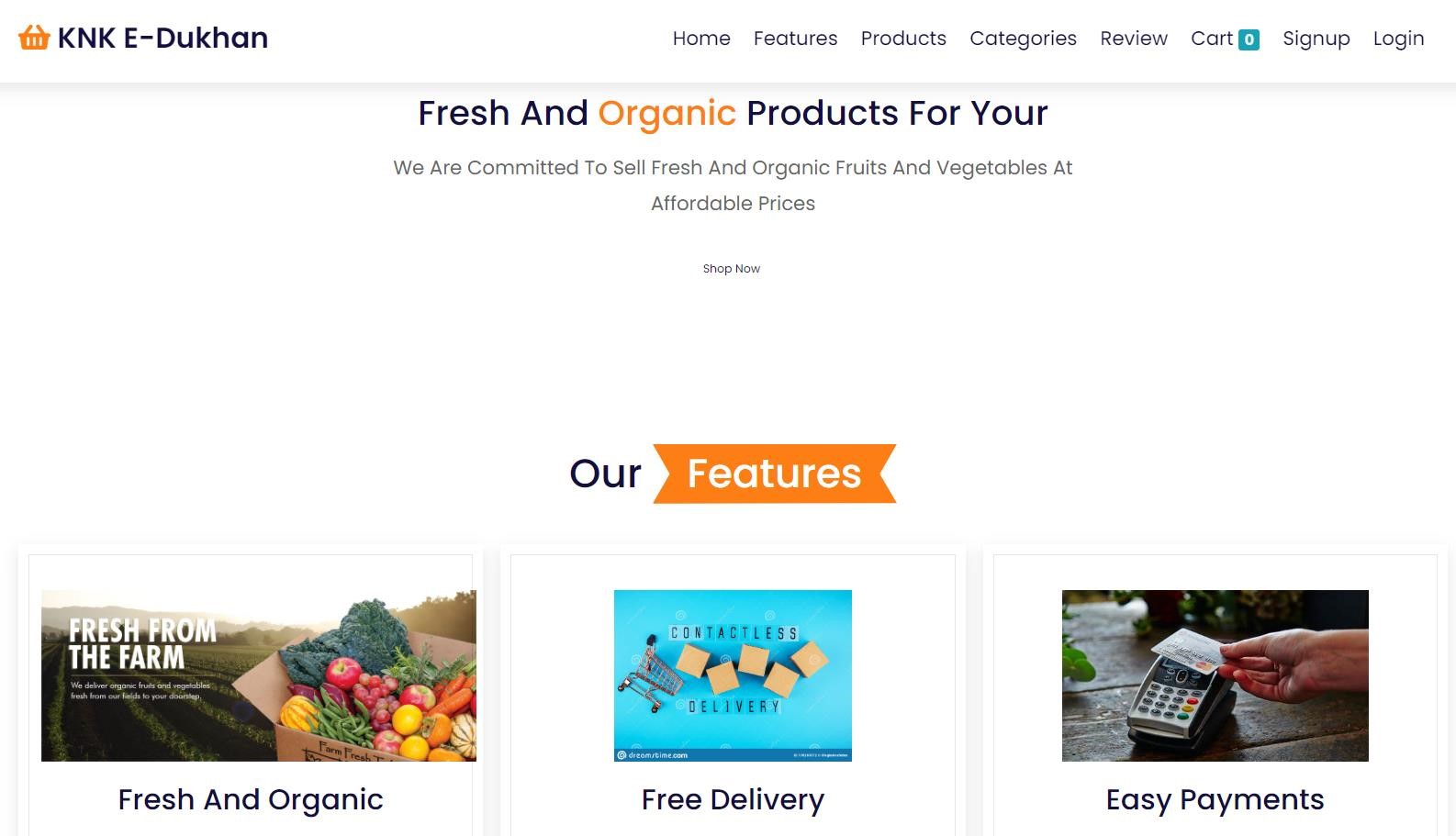
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No** | **Test Case Description** | **Expected Input** | **Expected Output** | **Pass/Fail** |
| 1 | User Signup | Wrong Email format | Fail | Pass |
| 2 | User Signup | Correct Email format | Pass | Pass |
| 3 | User Signup | First name less than 2 characters | Fail | Pass |
| 4 | User Login | Wrong Details | Fail | Pass |
| 5 | User Login | Valid Credentials | Pass | Pass |
| 6 | Phone number | Less than 10 digits | Fail | Pass |
| 7 | Phone number | Equal to 10 digits | Pass | Pass |
| 8 | User signup | Existing email | Fail | Pass |
| 9 | User signup | New email | Pass | Pass |

**Unit Testing Done**

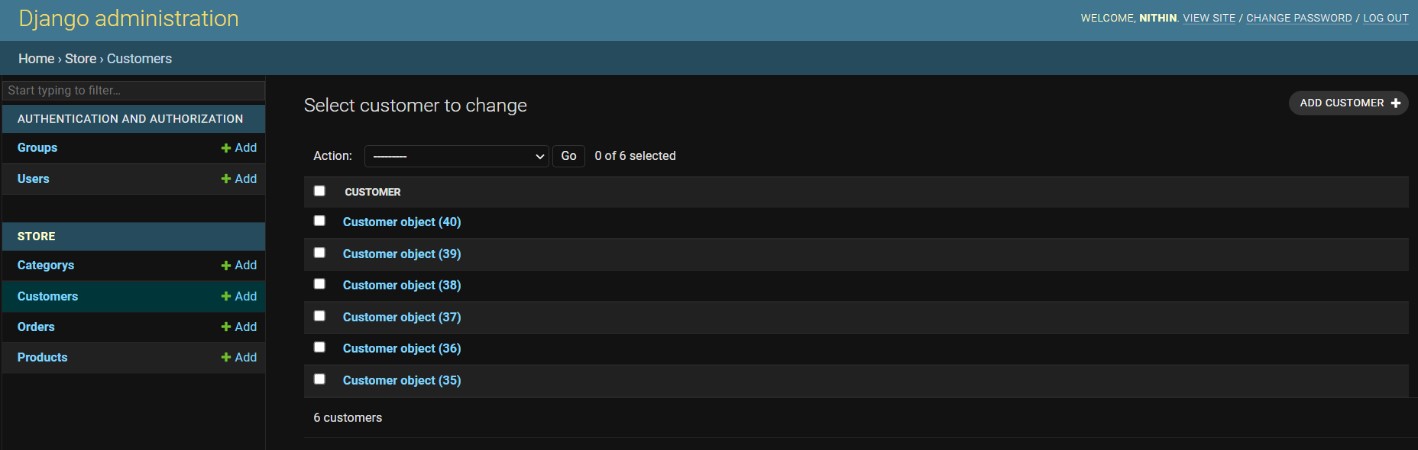
# Chapter-7

**RESULTS AND SNAPSHOTS**

**Snapshot 1**: The below figure 7.1 shows the home page

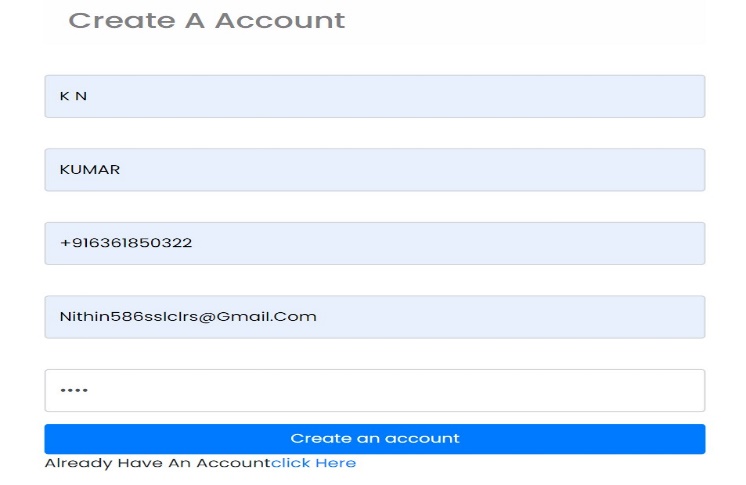


## *Figure 7.1 Landing Page*

**Snapshot 2**: The following figure 7.2 shows the records available in the database.

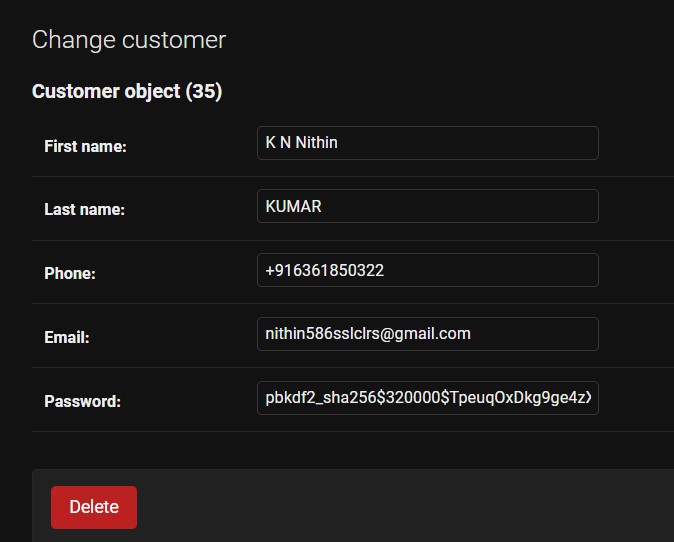
## *Figure 7.2 Records in Database*

**Snapshot 3**: The following figure 7.3 shows registering as a test user.



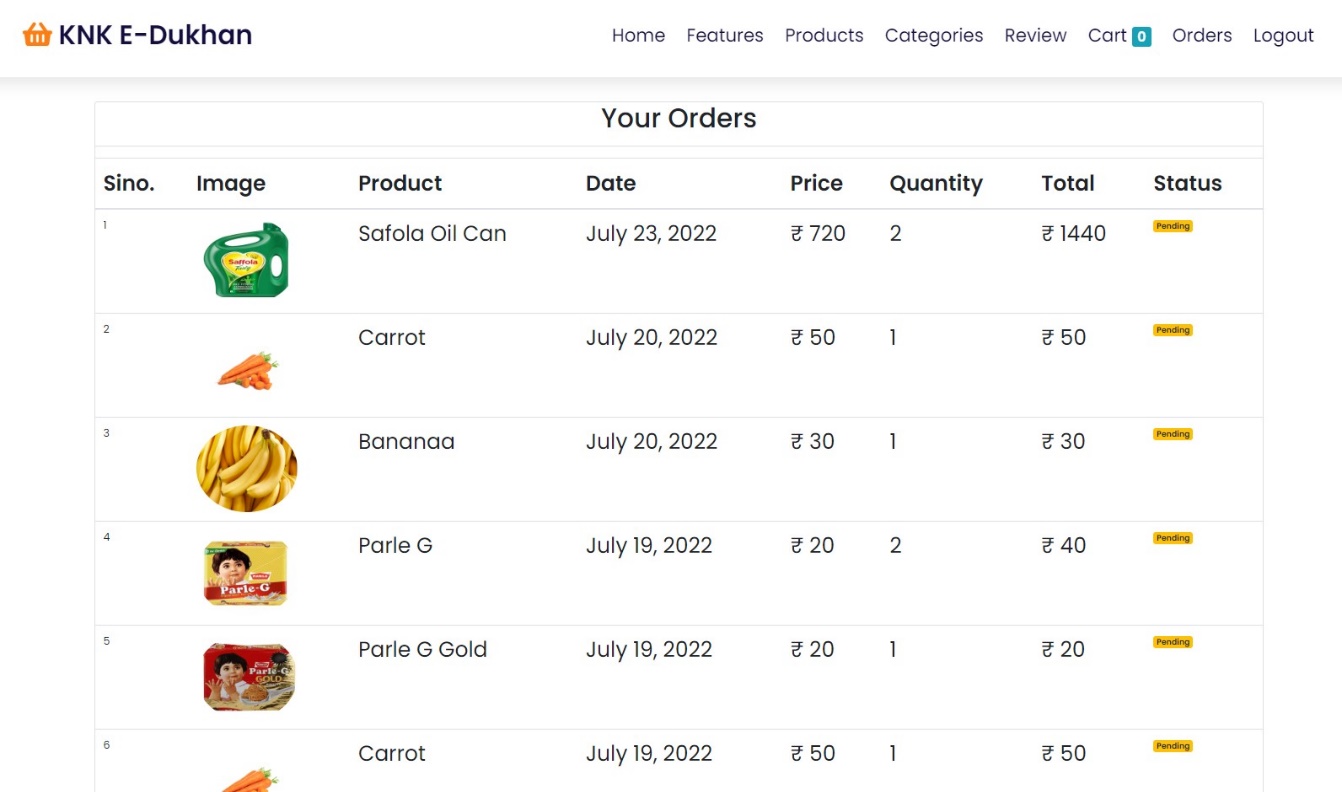
***Figure 7.3 Registering as a user***

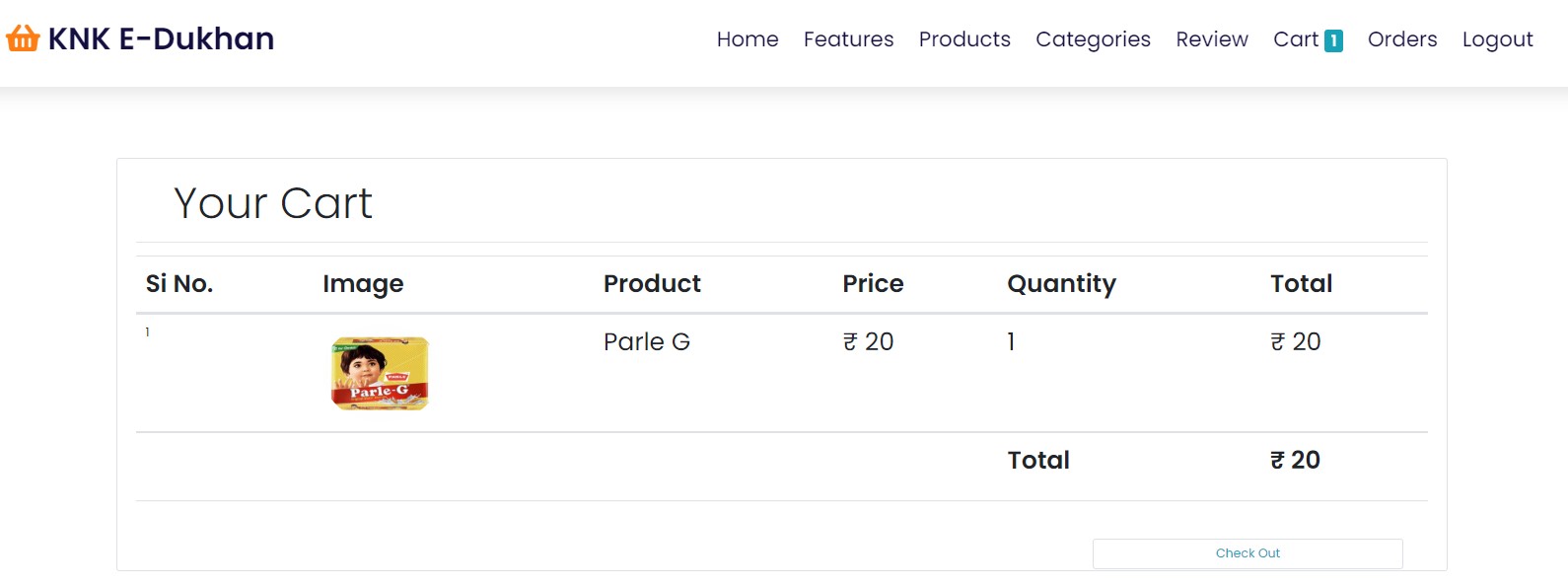
**Snapshot 4**: The following figure 7.4 shows test user inserted into database.



## *Figure 7.4 User inserted into database*

**Snapshot 5**: The following figure 7.5 shows orders and cart of the logged in customer





***Figure 7.5 Orders and cart of the customer***

**Chapter-7**

# CONCLUSION

This application allows the valid customer to buy the products from a single grocery store this project we learned the using Django framework also we learned connecting front-end with the backend and interfacing with the database .In future we continue this project to functions as third party merchant app which allows the multiple grocery vendors to sell their products in out site also includes payment gateway integration

**REFERENCES**

<https://www.youtube.com/watch?v=APESx65wpBc&list=PLdBwVRHjcI__NWxctXUSLz1Gg2Mb-B-O->

https://docs.djangoproject.com/en/4.1/