Chapter 1: Introduction, Problem Definition/Statement and Proposed Model

1.1 INTRODUCTION

New technology and techniques are implemented in Ecommerce websites to visually present information due to significant growth in shopping worldwide. A web application that also allows visually impaired and blind users to access and shop. The system gives voice instructions with welcome messages which assist users to use the website and an input is requested from the user. The user is required to hover over the entire page using the cursor to access different voice messages for various fields. The user provides input using the keyboard keys thus taking the process forward. The website also provides recommendations based on customer selection using collaborative filtering algorithms. It is a process of filtering information or patterns using techniques involving collaboration of data gathered from different agents and data sources. The system processes the voice output, and asks for the input by pressing the button of their desired choices.

This will help them to buy their desired things online. Every kind of instruction will give them in mode of audio they need to hear and follow the instructions.

1.2 PROBLEM DEFINITION/STATEMENT

According to WHO, globally 1 billion people suffer from partial or severe vision impairment or blindness. When the number is so huge, it is needless to say that your e-commerce store design should consider web accessibility for shoppers with partial or complete blindness. Consider a blind individual who lives alone, but wants to make a purchase from an ecommerce site. He doesn't have anybody there by his side, at the moment, to either guide him on what to do on the site, or make the purchase for him. Traditionally users type in a query into a search box to fetch answers this will some time take more time and not too friendly like consumers to shop when they're cooking, multitasking, or even driving.

It is also difficult for blind people to choose clothes with different colors or they find it difficult to shop online. People with complete blindness use screen readers to access the web content. Screen readers are software that allows people with blindness to read the web content either with a Braille display or a sound synthesizer. Now it often happens that the screen readers don't work properly due to browser compatibility issues. People with visual impairments find it difficult to differentiate between the foreground and background when they encounter low-contrast color schemes. With this in mind, how can any seller make sure that e-commerce store stands out from the crowded competition? Further to it, to make online shopping more convenient, voice-enabled searches are becoming immensely popular and are being widely adopted.

1.3 BACKGROUND STUDY

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace. The objective of this project is to develop a general-purpose e-commerce store where any product (such as books, groceries, cloths, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online ecommerce store.

An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction.

Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number but in our case we have one added feature of voice based module which can help to do shopping of the product till adding to cart. An email notification is sent to the customer as soon as the order is placed.

1.4 PROPOSED SOLUTION

A system is developed to assist people by automatically recognizing product patterns and colors. The system processes the voice output, and asks for the input by pressing the button of their desired choices. This will help them to buy their desired things online. Every kind of instruction will give them in mode of audio they need to hear and follow the instructions. Just try to keep our online store website as simple and minimalistic as possible. In short, our website is less flashy in terms of graphics. We wil try to use high-contrast color schemes when choosing colors for your online store. We can take the help of tools like Wave and CheckMyColours to figure which component of your online store is high contrast or low contrast. A rise in adoption of smart speakers by consumers, plus the growing usage in voice-enabled searches, combined with an increasing inclination to use voice assistants for shopping, voice-based eCommerce will be the next major disruption for online retail. This is just the right time for retailers to focus on voice search optimization strategy in their eCommerce business to stay ahead of the curve. When users can do pretty much everything by their voice, they are more likely to spend less time on device keyboards. From how the things are shaping currently, voice commerce is all poised to gain more traction.

1.5 MODULE DESCRIPTION

Products module: The shop owner has the authority to add or delete items from the web mall. Shop owner also has the right to modify those details. This includes the information about a particular product, such as product number, item, name, category, images of products,

description, features, constraints of products, product price which are to be displayed on the website.

Orders Module: When a customer goes through checkout, the information on their order is automatically transferred to the Orders section for you to keep track of. In the administration, you can view all of the orders made on their site, manually add orders, or edit the details of existing orders. The Orders section is located under Sales > Orders. On this page, every order ever made from the store is listed in detail.

Categories Module: In the default theme of the store front, parent categories are listed in the top menu of the home page, and on the left side of product pages. This navigational feature is used to guide customers to similar products within the same category. Exposing customers to different products within a category lets the customer compare the similarities and differences between products to make the most informed purchase. When adding products to the store, you will be asked for a product category to sort them in. It is a good idea to establish these categories before adding products, to save yourself the trouble of adding the category name to the products later.

Shopping module: The customer can select the items and put it into a cart and as soon as he clicks on finish, the total amount to be paid is displayed and also provides a discount if any. The customer can pay cash on delivery or else he can pay by entering his credit card number and address to which the items have to be shipped.

Customer Module: Shop owners should know who their customers are and how to manage their information. In the administration, customer information will need to be stored efficiently to remember any transactions made with their account. To access customer information, you can log into the administration panel of the store. The Customer management sections are located under Sales> Customers.

Billing Module: Here customers can view billing reports and order reports. Admin module: Administrator has full permission to access this web site. Administrators or employees can handle customer details, and he can communicate with customers through mail. If a customer forgets his password, the employee can reset his password.

Report module: This module deals with report management of the entire system.

Coupons Module: You may create and designate coupons to specific products or product categories under Sales > Coupons. The Coupons page will display a list of all the coupons created in the administration.

1.6 PURPOSES

The project is about to handle all the information of the shop regarding members. Also it manages resources which were managed and handled by manpower previously. The main

purpose of the project is to integrate distinct sections of the shop into consistent manner so that complex functions can be handled smoothly. The project aims at the following matters:

- Automation of product manipulation.
- Buying products.
- To manage information of different types of items.
- Consistently update information of all the item.
- Managing security by providing authorized email & password. Manages database efficiently

Chapter 2: Objective, Scope & Limitations

2.1 Objective

The main objective of the Speak and Shop Portal is to manage the details of Products, Customer, Shipping, Payment, and Category with voice command. It manages all the information about Products, Sales, Category, and Products. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work with the help of voice oriented system for managing the Products, Customer, Sales, and Shipping. It tracks all the details about the Shipping, Payment, and Category.

2.2 Scope of Project

This website is created so that it helps visually impaired and blind people to use the e-commerce websites with ease. This will also help other normal users to shop easily. In future with the new studies additional technologies can be integrated with the existing website which further prove to be helpful for the audience in better navigation. In the future the website can also be converted to a mobile application which can help users navigate with the help of gestures. With the increasing efforts by major companies like Google, Netflix, Uber, Apple, and Samsung to incorporate accessibility, in future the technologies used in the website can also act as an extension to other e-commerce websites so that it increases target audience on a global level.

2.3 Limitations

Security: One of the main limitations of e-commerce is security. In most cases, people are hesitant to provide their personal and financial details in spite of advanced data encryption security systems in place. Moreover, there are some websites that do not have the capability and features installed to authenticate transactions. As such, there are instances of fraudulent activities. The fear of providing financial information like credit card details hinders the growth of e-commerce.

Lack of Privacy: To some extent, the privacy of a customer is compromised in e-commerce. You need to provide your personal details, such as an address, telephone number, and so on to the seller. There are still lots of sites that do not have the advanced technology to protect sensitive information. Moreover, there are also sites that illegally collect consumer statistics without permission. This is one reason why people get skeptical while using e-commerce.

Product Suitability: As already mentioned, it is not possible for people to physically examine The product in e-commerce. In many cases, the original product may not match the picture or specifications in the e-commerce site. This absence of 'touch and feel' creates a discouraging effect.

Technical Limitations: e-commerce requires advanced technology platforms for better performance. Some limitations, such as lack of proper domain, network and software issues, and so on can affect the seamless performance of an e-commerce site.

Delivery Guarantee: Many people fear that their product might not be shipped or the website might be a fraud. Businesses need to work to build customer trust with reviews, testimonials, etc. to add more value to their website.

Legal Issues: A lot of legal compliances and cyber laws need to be taken care of in an e-commerce business. These regulations may vary from country to country. All these reasons deter businesses from going electronic.

2.4 Planning

1. Competitor analysis. In the early stages of the project, we conduct in-depth competitor and market research. It helps us to identify the weaknesses and strengths of the market leaders. Moreover, we receive insights on the mistakes we can avoid and the best practices we can adopt.

2. Feature list. This stage includes selecting the features we will integrate into the e-commerce website or the future online platform. The list of features has an impact on the user experience of your future customers.

3. Shipment and payment providers. Our team carefully selects the payment and shipment providers. The choice has a direct impact on the number of future sales, expenses, and the online shop's trustworthiness.

4. Design. Our team could recommend using some particular layouts or design elements distinguished as your competitor's best practices. The team also offers several options of themes from websites. In addition, we can develop a custom e-commerce design according to your business needs.

5. Data migration. In some cases, a customer already has an online shop with databases to import into the future project. We include the database migration to the workflow since it has an impact on the cost scope and the time of the online shop development.

6. The e-commerce platform. We offer several options of the e-commerce platforms depending on the customer business needs, goals, budget, and other factors. In some cases, we develop custom ecommerce platforms.

Then, to keep the project progress transparent and measurable, we use six main deliverables.

2.5 Feasibility

Whatever we think need not be feasible. It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

Technical Feasibility

It is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance.

Economical Feasibility

Development of this application is highly economically feasible .The organization needed not spend much m one for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. I f we are doing so , we can attain the maximum usability of the corresponding resources .Even after the development , the organization will not be in a condition to invest more in the organization .Therefore , the system is economically feasible.

2.6 Hardware & Software Requirements

2.6.1 Client Side

	DESCRIPTION	ALTERNATIVE (if any)
HARDWARE	PC with 20 GB hard-disk and minimum 512 MB RAM with Internet facility.	Not Applicable
SOFTWARE	Internet Explorer	Mozilla Firefox / Google Chrome / Apple Safari

Table 2.1: Client Side Requirements

2.6.2 Server Side

	DESCRIPTION	ALTERNATIVE (if any)
HARDWARE	PC with 1 TB hard-disk and minimum 2 GB RAM with Internet facility.	AWS(or any other cloud platform)
SOFTWARE	Python & MySQL Server	Not Applicable

Table 2.1 : Server Side Requirements

Chapter 3: Planning and Feasibility Analysis

3.1 Planning

Planning for every project is very individual and specific. However, several major steps are going to be similar for many projects of the businesses. These major steps are described in the next paragraphs.

STEP #1: CONSIDER ECOMMERCE WEBSITE SCOPE OF WORK

The first step to start with is to craft a scope statement and a general outline for the project. Creating the outline and the scope is usually the first step towards writing technical documentation and specs. Certainly, documentation is going to be changed in the scope of developing the project but initially, it is created by stakeholders and a project manager. The first step also contains developing the purpose and objectives of the project as well as the business justification for the project and is practically a basis for the entire future work.

STEP #2: ORGANIZE DELIVERABLES INTO A WORK BREAKDOWN STRUCTURE

When it comes to implementing project ideas effectively and promptly, there is a need to prepare deliverables. In turn, making deliverables is a part of the WBS that is the work breakdown structure for the project. A manager is the one to create the WBS according to the requirements of the customer, which need to include project scope, project tasks, and tasks split into simpler subtasks. Such an approach helps to make one step at a time and build the project quicker. Thus, the WBS is necessary as a project roadmap.

STEP #3: GET TO KNOW YOUR DEPENDENCIES

When you already have the parts of your project to do with the WBS, it is time to create a to-do list for every subtask. At this point, many managers often ignore dependencies and this might create bottlenecks in the project. To avoid bottlenecks, there is a need to plan tasks, taking in mind their dependencies on other tasks and subtasks. It is important to know whether there are subtasks that need to be completed before another task or at least initiated. With this knowledge, it is easier to create a smooth working process. It is a duty of a project manager to specify dependencies in the WBS and to make sure these dependencies are taken into account when the work is planned.

STEP #4: CREATE A TIMELINE

Have you heard about a Gantt chart? It is a very convenient tool to use when it comes to planning the milestones and the dependencies. With a Gantt chart, you can plan time and tasks in a way that at any given moment you can see the status your project is in. The example of a Gantt chart is represented in the picture below. Usually, a Gantt chart is created in Excel but you can generally use whatever you find comfortable. The main idea is to be specific enough, to keep planning close to the needs of your project, and to make sure that all the dependencies are taken into account when planning those steps.

STEP #5 PLAN BUDGET

It goes without saying, before starting the project, there is a need to estimate the total cost that you can afford. Ideally, there should be several estimations, depending on the timeline and features added to the project. For example, you might want to have the lowest and the highest total cost of your project at hand so you can estimate the project from several points of view. Potentially, this helps to be more realistic with the budget, and determine the features that are a must for the project and the ones that are auxiliary but desired.

3.2 Feasibility

After doing the project speak and shop project, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time. Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements. Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

• Technical Feasibility

• Operation Feasibility

• Economical Feasibility

Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee

of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Economic Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

Chapter 4: Software Design

4.1 System Design

System Design of Speak and Shop System In this phase, a logical system is built which fulfils the given requirements. Design phase of software development deals with transforming the client's requirements into a logically working system. Normally, design is performed in the following in the following two steps:

1. ***Primary Design Phase:***
   In this phase, the system is designed at block level. The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions emphasis is put on minimizing the information flow between blocks. Thus, all activities which require more interaction are kept in one block.

2. ***Secondary Design Phase:***
   In the secondary phase the detailed design of every block is performed. The general tasks involved in the design process are the following:

   i. Design various blocks for overall system processes.
   ii. Design smaller, compact and workable modules in each block.
   iii. Design various database structures.
   iv. Specify details of programs to achieve desired functionality.
   v. Design the form of inputs, and outputs of the system.
   vi. Perform documentation of the design.
   vii. System reviews.

User Interface Design

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages is called a dialogue The following steps are various guidelines for User Interface Design:

1. The system user should always be aware of what to do next.
2. The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.
3. Message, instructions or information should be displayed long enough to allow the system user to read them.
4. Use display attributes sparingly.
5. Default values for fields and answers to be entered by the user should be specified.
6. A user should not be allowed to proceed without correcting an error.

7. The system user should never get an operating system message or fatal error.

4.2 Detailed Design

4.2.1 0 level DFD

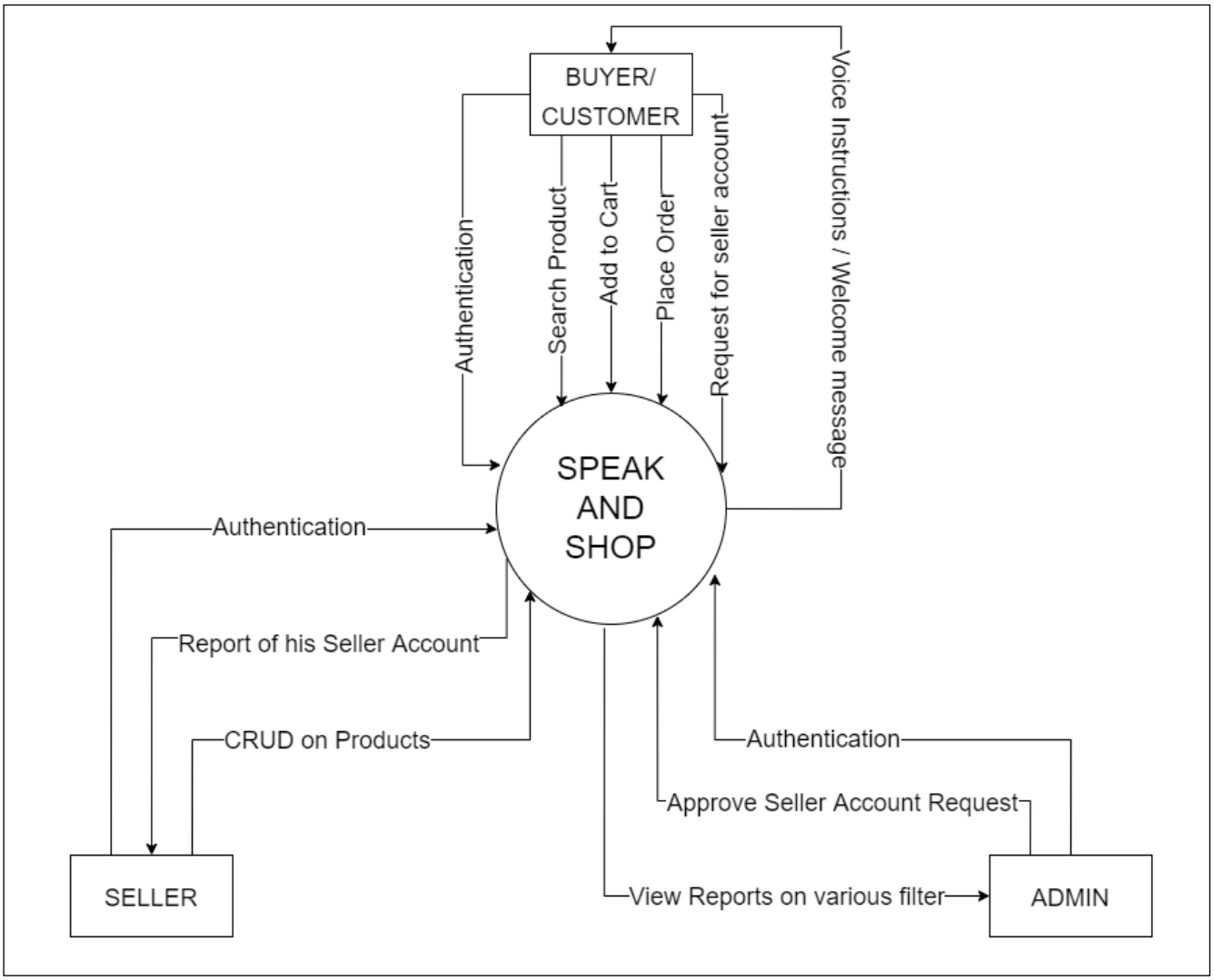


Figure 4.1: 0 Level DFD

4.2.2 1 level DFD

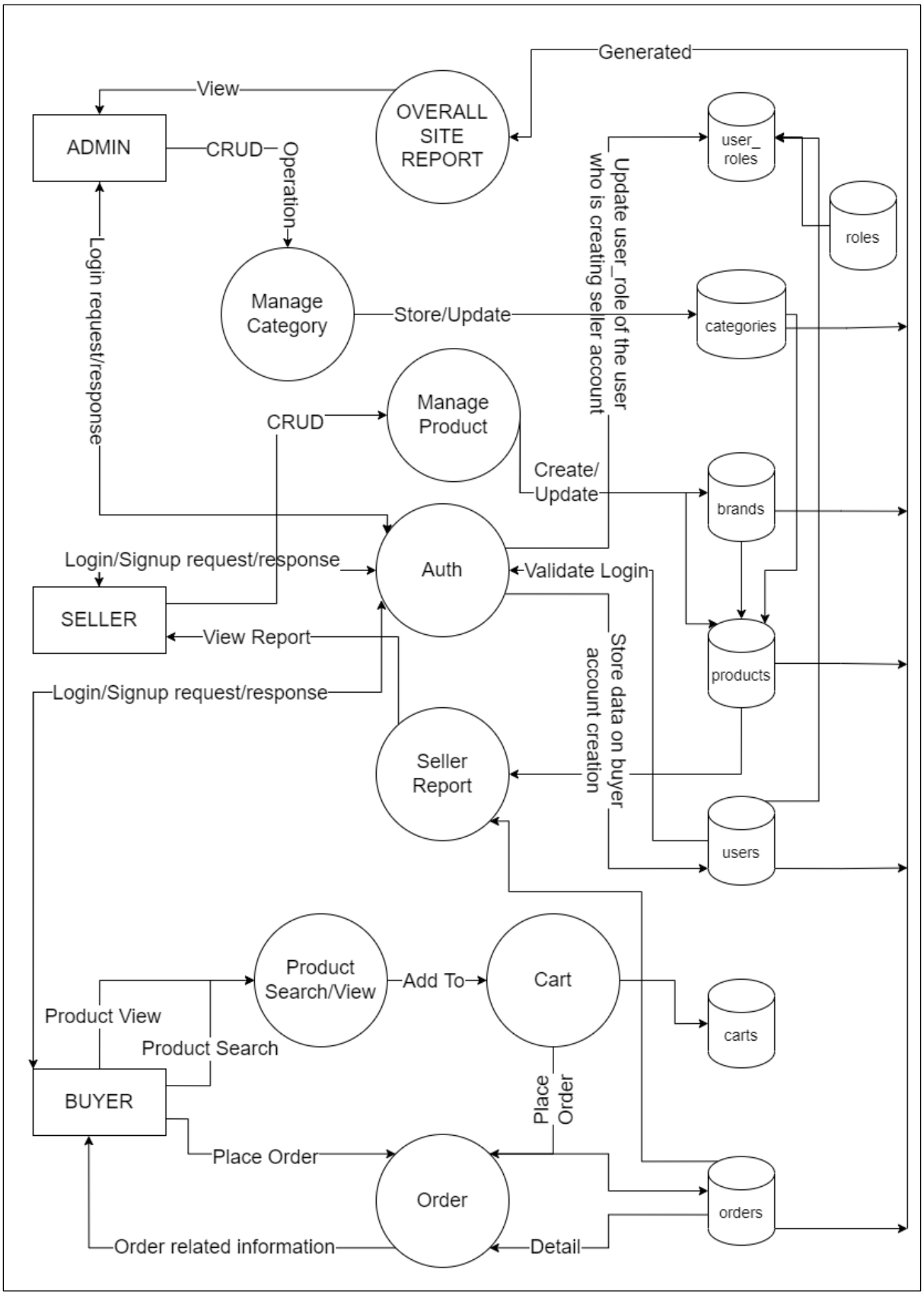


Figure 4.2: Level 1 DFD

Chapter 5: Database Design

5.1 ER Diagram

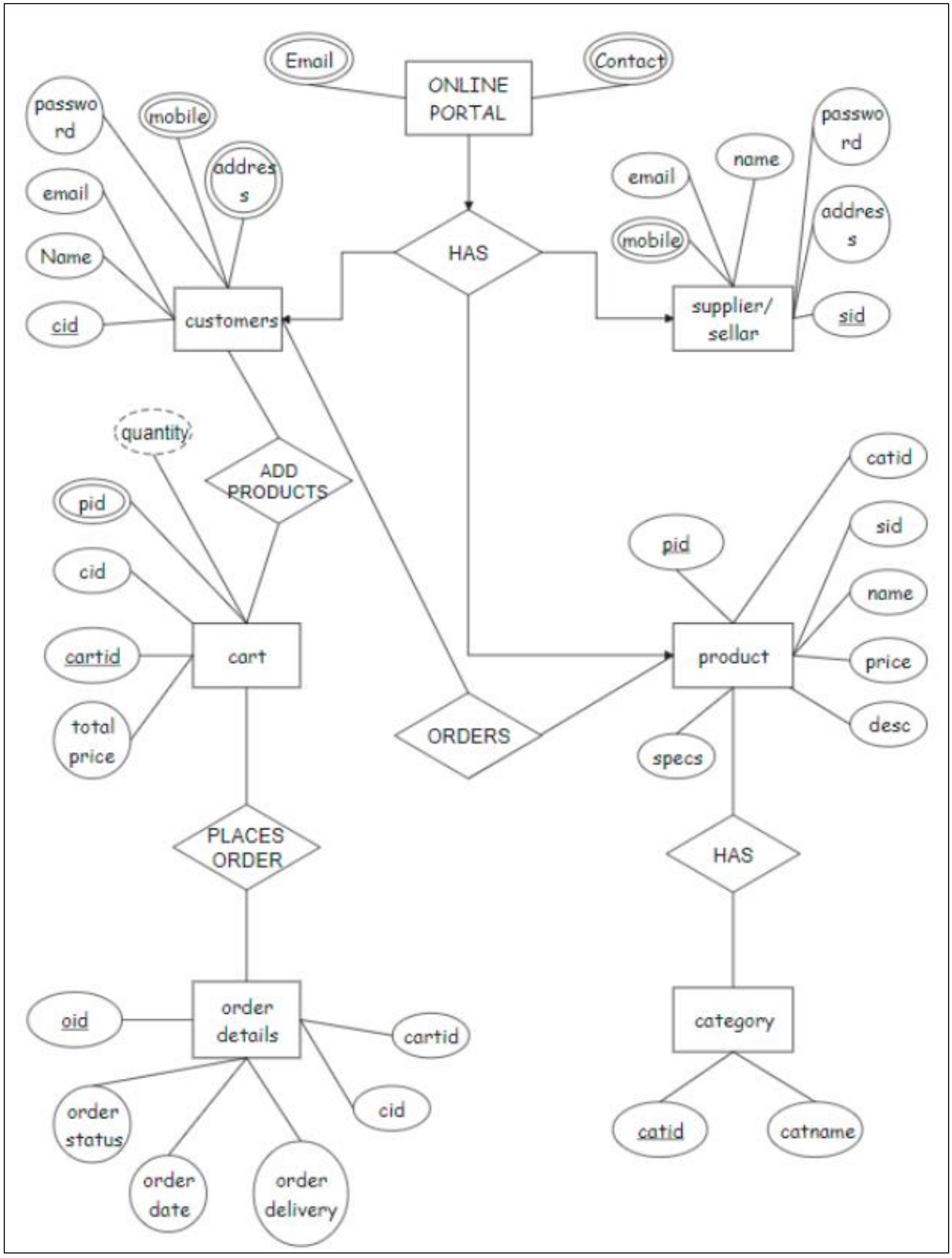


Figure 5.1: ERD Diagram

5.2 Schema Diagram

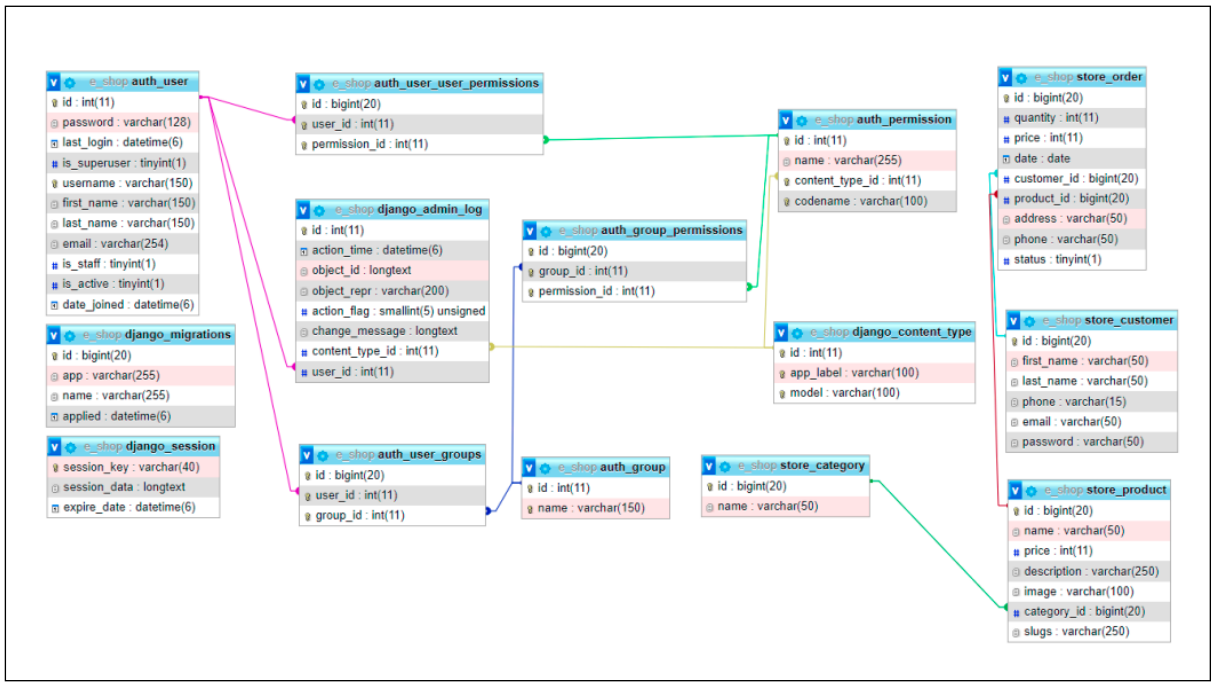


Figure 5.2: Schema Diagram

Chapter 6: Coding

After all the project planning and discussions the main part comes into act where our planning must be converted to desired output to our work through logically well-developed code. Coding doesn't means to just put your knowledge in coding and bring out the required task anyhow, but real coding means that it must follow some certain coding rules as: coding structure, maintaining proper folder/file structure, naming conviction that are used for variable, using comment as essential part in our code when working in a team environment so that our works may be easy to understand by other members as well as to any 3rd person who reviews our project, separating different coding files from each other for example all style files must be inside a separate folder with proper name as style or css also in same way any js or images that are being used in the front end side should be wrapped in a folder named assets.

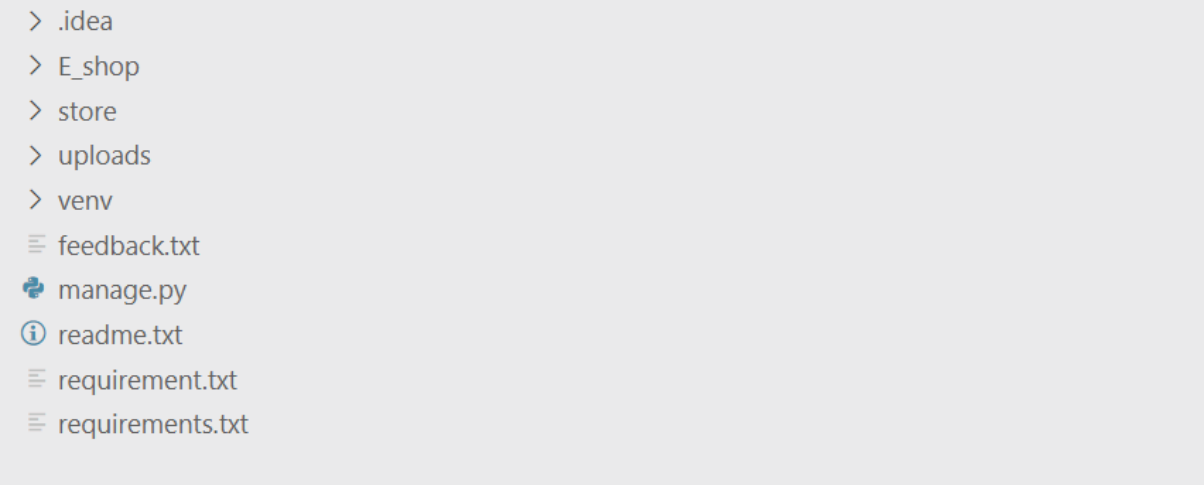
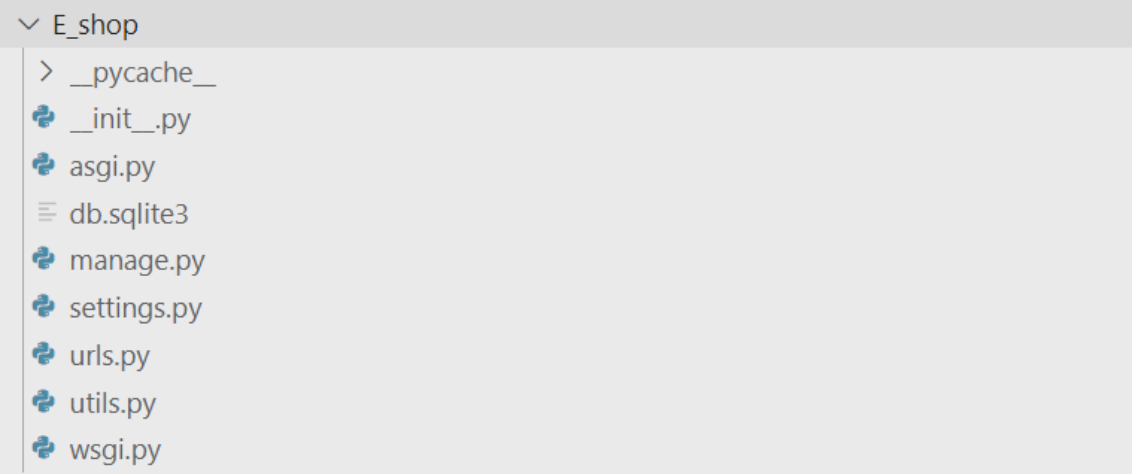


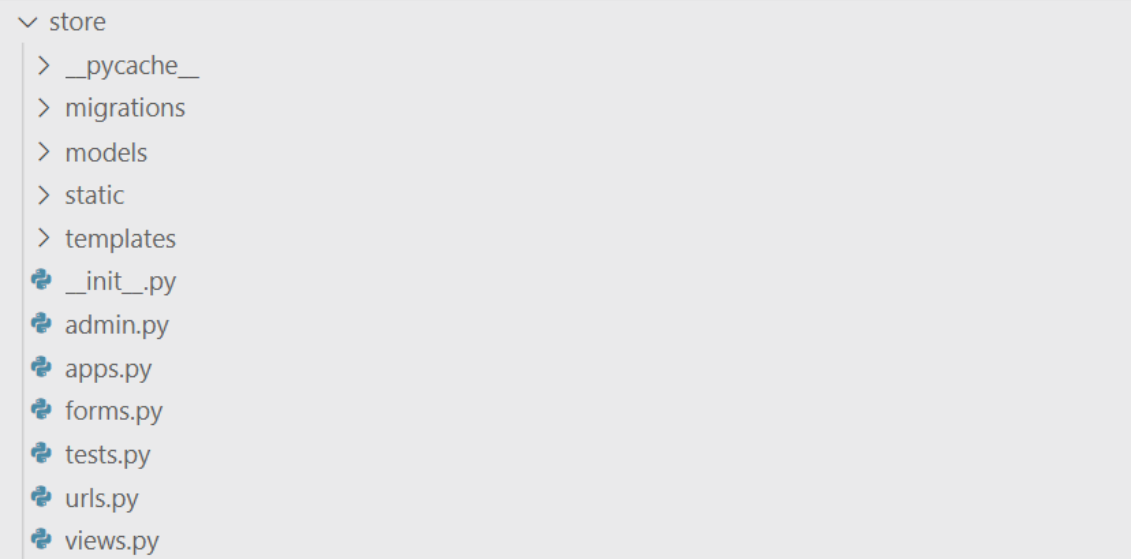
Figure 6.1: Project Root Folder Structure

Above is the folder structure of the root folder of the project. As we can see in this image we have multiple folders in this directory and some files too. Let's have a quick view on every directory's structure one by one.

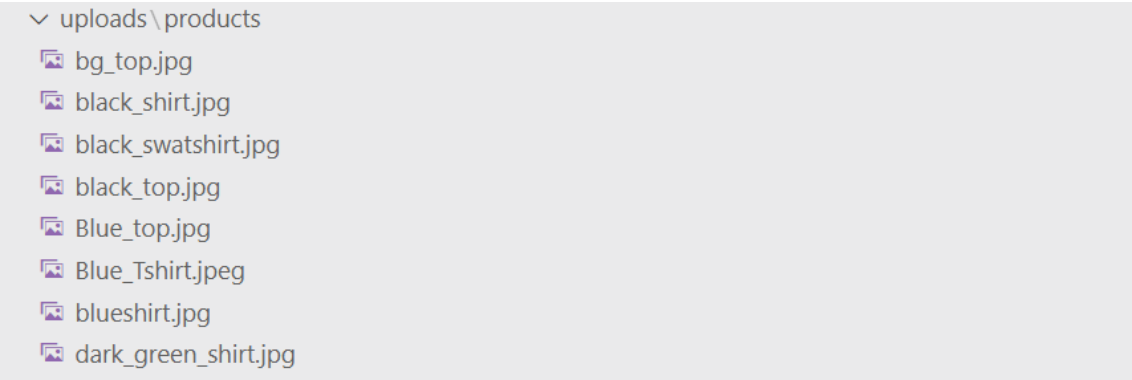
1. E_shop



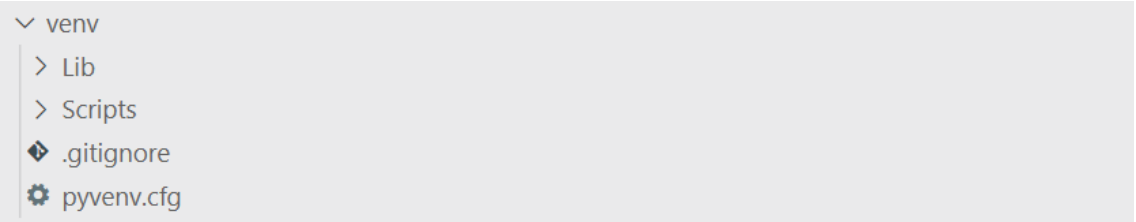
2. Store



3. Uploads



4. venv



That’s all with the directory structure of the project . Now let’s get started with the code.

Folder: E_shop

File : asgi.py

```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'E_shop.settings')

application = get_asgi_application()
```

File: manage.py

```
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'E_shop.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

File: settings.py

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-v@+*0)1azr+%*&(j1d+l(&tsh%2eop&&9=5x810syamk@4adxr'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']
#APPEND_SLASH=False

# Application definition

INSTALLED_APPS = [
    'store',
    'jet.dashboard',
    'jet',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',


]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'E_shop.urls'

TEMPLATES = [
```

```
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'E_shop.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'e_shop',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'
MEDIA_URL="/images/"
MEDIA_ROOT=BASE_DIR

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL='home'
LOGOUT_REDIRECT_URL='login'

RAZOR_KEY_ID = 'rzp_test_SFxoDXXyHgmCJj'
RAZOR_KEY_SECRET = 'SYdzJ7wr55dQ2uHZ2y3pIJiF'

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'speakandshop@gmail.com'
EMAIL_HOST_PASSWORD = 'Sas@2022'
EMAIL_PORT = 587
```

File: urls.py

```
from django.contrib import admin
from django.urls import path,include
from . import settings
from django.conf.urls.static import static
admin.site.site_header="Speak and Shop Admin Pannel"
admin.site.site_title="SAS Dashboard"
admin.site.index_title="Welcome to Admin Pannel"
urlpatterns = [

    path('jet/',include('jet.urls')),
    path('jet/dashboard/', include('jet.dashboard.urls', 'jet-dashboard')),
    path('admin/', admin.site.urls),
    path('',include('store.urls')),
    #path('accounts/',include('django.contrib.auth.urls'),name='login'),
    #path('accounts/password_change',include('django.contrib.auth.urls'),name='password_change'),

]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

File: utils.py

```
import random
import string

from django.utils.text import slugify


def random_string_generator(size=10, chars=string.ascii_lowercase + string.digits):
    return ''.join(random.choice(chars) for _ in range(size))


def unique_slug_generator(instance, new_slug=None):
    """
    This is for a Django project and it assumes your instance
    has a model with a slug field and a title character (char) field.
    """
    if new_slug is not None:
        slugs = new_slug
    else:
        slugs = slugify(instance.name)

    Klass = instance.__class__
    qs_exists = Klass.objects.filter(slugs=slugs).exists()
    if qs_exists:
        new_slug = "{slug}-{randstr}".format(
            slugs=slugs,
            randstr=random_string_generator(size=4)
```

```
        )
        return unique_slug_generator(instance, new_slug=new_slug)
    return slugs
```

File: wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'E_shop.settings')

application = get_wsgi_application()
```

Folder: Store

Subfolder: Migrations

File: initial.py

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Product',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(max_length=50)),
                ('price', models.IntegerField(default=0)),
                ('description', models.CharField(default='', max_length=250)),
                ('image', models.ImageField(upload_to='products/')),
            ],
        ),
    ]
```

File: alter_product_image.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0001_initial'),
    ]

    operations = [
        migrations.AlterField(
            model_name='product',
            name='image',
            field=models.ImageField(upload_to='uploads/products/'),
        ),
    ]
```

File: category.py

```
class Migration(migrations.Migration):

    dependencies = [
        ('store', '0002_alter_product_image'),
    ]

    operations = [
        migrations.CreateModel(
            name='Category',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(max_length=50)),
            ],
            options={
                'db_table': 'store_category',
            },
        ),
    ]
```

File: auto_20211209_1020.py

```
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
```

```
        ('store', '0003_category'),
    ]

    operations = [
        migrations.AddField(
            model_name='product',
            name='category',
            field=models.ForeignKey(default=1, on_delete=django.db.models.deletion.CASCADE,
to='store.category'),
        ),
        migrations.AlterModelTable(
            name='product',
            table='store_product',
        ),
    ]
```

File: alter_product_description.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0004_auto_20211209_1020'),
    ]

    operations = [
        migrations.AlterField(
            model_name='product',
            name='description',
            field=models.CharField(blank=True, default='', max_length=250, null=True),
        ),
    ]
```

File: customer.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0005_alter_product_description'),
    ]

    operations = [
        migrations.CreateModel(
```

```
            name='Customer',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('first_name', models.CharField(max_length=50)),
                ('last_name', models.CharField(max_length=50)),
                ('phone', models.CharField(max_length=15)),
                ('email', models.CharField(max_length=50)),
                ('password', models.CharField(max_length=50)),
            ],
            options={
                'db_table': 'store_customer',
            },
        ),
    ]
```

File: products_slug

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0006_customer'),
    ]

    operations = [
        migrations.AddField(
            model_name='product',
            name='slugs',
            field=models.CharField(default='', max_length=50),
        ),
    ]
```

File: auto_20220309_1841.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0007_product_slugs'),
    ]

    operations = [
        migrations.AlterModelOptions(
```

```
            name='category',
            options={'verbose_name_plural': '1. Categories'},
        ),
        migrations.AlterModelOptions(
            name='product',
            options={'verbose_name_plural': '2. Products'},
        ),
        migrations.AlterField(
            model_name='product',
            name='slugs',
            field=models.SlugField(default=''),
        ),
    ]
```

File: Alter_product_slug.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0008_auto_20220309_1841'),
    ]

    operations = [
        migrations.AlterField(
            model_name='product',
            name='slugs',
            field=models.SlugField(blank=True, max_length=250, null=True),
        ),
    ]
```

File: alter_product_slugs.py

```
class Migration(migrations.Migration):

    dependencies = [
        ('store', '0009_alter_product_slugs'),
    ]

    operations = [
        migrations.AlterField(
            model_name='product',
            name='slugs',
            field=models.CharField(blank=True, max_length=250, null=True),
        ),
    ]
```

File: cartorder_cartorderitems.py

```
from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('store', '0010_alter_product_slugs'),
    ]

    operations = [
        migrations.CreateModel(
            name='CartOrder',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('total_amt', models.FloatField()),
                ('paid_status', models.BooleanField(default=False)),
                ('order_dt', models.DateTimeField(auto_now_add=True)),
                ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
            options={
                'verbose_name_plural': 'Orders',
            },
        ),
        migrations.CreateModel(
            name='CartOrderItems',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('invoice_no', models.CharField(max_length=150)),
                ('item', models.CharField(max_length=150)),
                ('image', models.CharField(max_length=200)),
                ('qty', models.IntegerField()),
                ('price', models.FloatField()),
                ('total', models.FloatField()),
                ('order', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='store.cartorder')),
            ],
            options={
                'verbose_name_plural': 'Order Items',
            },
```

```
        ),
    ]
```

File: auto_20220405_1803.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0011_cartorder_cartorderitems'),
    ]

    operations = [
        migrations.AlterModelOptions(
            name='cartorder',
            options={'verbose_name_plural': '3. Orders'},
        ),
        migrations.AlterModelOptions(
            name='cartorderitems',
            options={'verbose_name_plural': '4. Order Items'},
        ),
        migrations.AddField(
            model_name='cartorder',
            name='order_id',
            field=models.CharField(default='', max_length=250),
        ),
    ]
```

File: cartorder_order_status.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0012_auto_20220405_1803'),
    ]

    operations = [
        migrations.AddField(
            model_name='cartorder',
            name='order_status',
            field=models.CharField(choices=[('process', 'In Process'), ('shipped', 'Shipped'), ('delivered',
'Delivered')], default='process', max_length=150),
        ),
```

```
    ]
```

File: useradressbook.py

```
from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('store', '0013_cartorder_order_status'),
    ]

    operations = [
        migrations.CreateModel(
            name='UserAddressBook',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('mobile', models.CharField(max_length=50, null=True)),
                ('address', models.TextField()),
                ('status', models.BooleanField(default=False)),
                ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
            options={
                'verbose_name_plural': 'AddressBook',
            },
        ),
    ]
```

File: auto_20220406_1543.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0014_useraddressbook'),
    ]

    operations = [
        migrations.AlterModelOptions(
```

```
            name='useraddressbook',
            options={'verbose_name_plural': '5. AddressBook'},
        ),
        migrations.AddField(
            model_name='cartorderitems',
            name='address',
            field=models.TextField(null=True),
        ),
        migrations.AddField(
            model_name='cartorderitems',
            name='mobile',
            field=models.CharField(max_length=50, null=True),
        ),
    ]
```

File: auto_20220406_1848.py

```
class Migration(migrations.Migration):

    dependencies = [
        ('store', '0015_auto_20220406_1543'),
    ]

    operations = [
        migrations.RemoveField(
            model_name='cartorderitems',
            name='address',
        ),
        migrations.RemoveField(
            model_name='cartorderitems',
            name='mobile',
        ),
    ]
```

File: cartorderitems_address.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0016_auto_20220406_1548'),
    ]

    operations = [
```

```
        migrations.AddField(
            model_name='cartorderitems',
            name='address',
            field=models.CharField(max_length=200, null=True),
        ),
    ]
```

File: cartorderitems_mobile.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0017_cartorderitems_address'),
    ]

    operations = [
        migrations.AddField(
            model_name='cartorderitems',
            name='mobile',
            field=models.IntegerField(max_length=200, null=True),
        ),
    ]
```

File: alter_cart_orderitems_mobile.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0018_cartorderitems_mobile'),
    ]

    operations = [
        migrations.AlterField(
            model_name='cartorderitems',
            name='mobile',
            field=models.IntegerField(null=True),
        ),
    ]
```

File: seller.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0019_alter_cartorderitems_mobile'),
    ]

    operations = [
        migrations.CreateModel(
            name='Seller',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('first_name', models.CharField(max_length=50)),
                ('last_name', models.CharField(max_length=50)),
                ('phone', models.CharField(max_length=15)),
                ('email', models.CharField(max_length=50)),
                ('password', models.CharField(max_length=50)),
            ],
            options={
                'db_table': 'store_seller',
            },
        ),
    ]
```

File: auto_20220427_1014.py

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('store', '0020_seller'),
    ]

    operations = [
        migrations.RemoveField(
            model_name='seller',
            name='phone',
        ),
        migrations.AddField(
            model_name='seller',
            name='username',
```

```
            field=models.CharField(default='', max_length=250),
        ),
    ]
```

Subfolder: Models

File: __init__.py

```
from .product import Product
from .category import Category
from .customer import Customer
from .order import *
from .seller import *
from django.utils.html import mark_safe
```

File: category.py

```
from tabnanny import verbose
from django.db import models

class Category(models.Model):
    name=models.CharField(max_length=50)

    class Meta:
        db_table="store_category"
        verbose_name_plural='1. Categories'
    def __str__(self):
        return self.name
    @staticmethod
    def get_all_category():
        return Category.objects.all()
```

File: customer.py

```
from django.db import models
from django.core.validators import MinLengthValidator
class Customer(models.Model):
    first_name=models.CharField(max_length=50)
    last_name=models.CharField(max_length=50)
    phone=models.CharField(max_length=15)
    email=models.CharField(max_length=50)
    password=models.CharField(max_length=50)
    class Meta:
        db_table="store_customer"
    def register(self):
```

```
        self.save()

    @staticmethod
    def get_customer_by_email(email):
        try:
            return Customer.objects.get(email=email)
        except:
            return False


    def isExists(self):
        if Customer.objects.filter(email = self.email):
            return True

        return  False
```

File: order.py

```
from django.db import models
from django.contrib.auth.models import User
from django.utils.html import mark_safe
from django.urls import reverse
from django.conf.urls import url

status_choice=(
        ('process','In Process'),
        ('shipped','Shipped'),
        ('delivered','Delivered'),
    )
class CartOrder(models.Model):
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    total_amt=models.FloatField()
    order_id=models.CharField(max_length=250,default='')
    paid_status=models.BooleanField(default=False)
    order_dt=models.DateTimeField(auto_now_add=True)
    order_status=models.CharField(choices=status_choice,default='process',max_length=150)


    class Meta:
        verbose_name_plural='3. Orders'

# OrderItems
class CartOrderItems(models.Model):
    order=models.ForeignKey(CartOrder,on_delete=models.CASCADE)
    invoice_no=models.CharField(max_length=150)
    item=models.CharField(max_length=150)
    image=models.CharField(max_length=200)
    address=models.CharField(max_length=200,null=True)
```

```
    mobile=models.IntegerField(null=True)
    qty=models.IntegerField()
    price=models.FloatField()
    total=models.FloatField()

    class Meta:
        verbose_name_plural='4. Order Items'

    def image_tags(self):
        return mark_safe('<img src="/images/%s" width="50" height="50"/>' % (self.image))

class UserAddressBook(models.Model):
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    mobile=models.CharField(max_length=50,null=True)
    address=models.TextField()
    status=models.BooleanField(default=False)

    class Meta:
        verbose_name_plural='5. AddressBook'
```

File: product.py

```
from django.db import models

from E_shop.utils import unique_slug_generator
from .category import Category
from django.utils.html import mark_safe
from django.urls import reverse
from django.utils.text import slugify
import itertools
from django.db.models.signals import pre_save, post_save

class Product(models.Model):
    name=models.CharField(max_length=50)
    slugs=models.CharField(null=True,blank=True,max_length=250)
    price=models.IntegerField(default=0)
    category=models.ForeignKey(Category,on_delete=models.CASCADE,default=1)
    description=models.CharField(default='',null=True,blank=True,max_length=250)
    image=models.ImageField(upload_to='uploads/products/')
    class Meta:
        db_table="store_product"
        verbose_name_plural='2. Products'

    def __str__(self):
        return self.name
```

```
def image_tags(self):
    return mark_safe('<img src="%s" width="50" height="50"/>' % (self.image.url))

@staticmethod
def get_all_product():
    return Product.objects.all()

@staticmethod
def get_all_product():
    return Product.objects.all()

@staticmethod
def get_limited_product():
    return Product.objects.all()[:12:2]

@staticmethod
def get_all_product_list():
    return Product.objects.all().values()

@staticmethod
def get_latest_product():
    return Product.objects.order_by('-id')[:3]



@staticmethod
def search_product(q):
    return Product.objects.filter(name__icontains=q)

@staticmethod
def get_all_product_by_categoryid(category_id):
    if category_id:
        return Product.objects.filter(category=category_id)
    else:
        return Product.objects.all()

@staticmethod
def get_all_product_by_categoryid_onfilter(category_id):
    if category_id:
        return Product.objects.filter(category=category_id).values()
    else:
        return Product.objects.all()

@staticmethod
def get_product_detail(i):
```

```
        return Product.objects.filter(id=i)

def rl_pre_save_receiver(sender, instance, *args, **kwargs):
    if not instance.slugs:
        instance.slugs = unique_slug_generator(instance)


pre_save.connect(rl_pre_save_receiver, sender=Product)
```

File:seller.py

```
from django.db import models

from django.utils.html import mark_safe
from django.urls import reverse
from django.conf.urls import url

class Seller(models.Model):
    first_name=models.CharField(max_length=50)
    last_name=models.CharField(max_length=50)
    email=models.CharField(max_length=50)
    username=models.CharField(max_length=250,default='')
    password=models.CharField(max_length=250)
    class Meta:
        db_table="store_seller"
```

Subfolder: Static>AJAX

File: address.js

```
$(document).ready(function () {
    $(document).on('click','.activate-address',function(){
        var _aId=$(this).attr('data-address');
        var _vm=$(this);
        // Ajax
        $.ajax({
            url:'/activate-address',
            data:{
                'id':_aId,
            },
            dataType:'json',
            success:function(res){
                if(res.bool==true){
                    $(".address").removeClass('shadow border-secondary');
                    $(".address"+_aId).addClass('shadow border-secondary');
```

```
                $(".check").hide();
                $(".actbtn").show();

                $(".check"+_aId).show();
                $(".btn"+_aId).hide();
            }
        }
    });
    // End
});

});
```

File: cart.js

```
$(document).ready(function(){
    $(document).on('click',"#addToCart",function(){

        var _vm=(this);
        var _qty=1;
        var _productId=$('.product-id').val();
        var _productName=$('.product-name').val();
        var _productImage=$('.prod-img').data('value');
        var _productPrice=$('.price').data('value');
        console.log(_qty,_productId,_productName,_productPrice,_productImage);
        $.ajax({
            url:'/add-to-cart',
            data:{
                'id':_productId,
                'image':_productImage,
                'qty':_qty,
                'title':_productName,
                'price':_productPrice
            },
            dataType:'json',
            beforeSend:function(){
                //_vm.attr('disabled',true);
            },
            success:function(res){
                $(".cart-list").text(res.totalitems);
                //_vm.attr('disabled',false);
                alertify.success("Product added to cart")
            }
        });
    });

    $(document).on('click','.delete-item',function(){
```

```
        var _pId=$(this).attr('data-item');
        var _vm=$(this);
        // Ajax
        $.ajax({
            url:'/delete-from-cart',
            data:{
                'id':_pId,
            },
            dataType:'json',
            beforeSend:function(){
                //_vm.attr('disabled',true);
            },
            success:function(res){
                $(".cart-list").text(res.totalitems);
                //_vm.attr('disabled',false);

                location.reload();
                alertify.success("Product removed from cart")
            }
        });
        // End
    });
});
```

File: cart2.js

```
$(document).ready(function(){
    $(document).on('click',"#addToCart",function(){

        var _vm=(this);
        var _qty=1;
        var _productId=$(this).data('id');
        var _productName=$(this).data('name');
        var _productImage=$(this).data('image');
        var _productPrice=$(this).data('price');
        console.log(_qty,_productId,_productName,_productPrice,_productImage);
        $.ajax({
            url:'/add-to-cart',
            data:{
                'id':_productId,
                'image':_productImage,
                'qty':_qty,
                'title':_productName,
                'price':_productPrice
            },
            dataType:'json',
            beforeSend:function(){
```

```
            //_vm.attr('disabled',true);
        },
        success:function(res){
            $(".cart-list").text(res.totalitems);
            //_vm.attr('disabled',false);
            alertify.success("Product added to cart")
        }
    });
  });
});
```

File: checkout.js

```
$(document).ready(function () {
  $('.paywithrazorpay').click(function (e) {
    e.preventDefault();
    var order_id=$('#ordid').val();
    var api_key=$('#apiid').val();
    console.log(order_id);
    var options = {
      "key": api_key, // Enter the Key ID generated from the Dashboard
      "amount": "50000", // Amount is in currency subunits. Default currency is INR. Hence, 50000
refers to 50000 paise
      "currency": "INR",
      "name": "Acme Corp",
      "description": "Test Transaction",
      "image": "https://example.com/your_logo",
      "order_id": order_id, //This is a sample Order ID. Pass the `id` obtained in the response of Step 1
      "handler": function (response){
        swl(response.razorpay_payment_id);
        alert(response.razorpay_order_id);
        // alert(response.razorpay_signature)
      },
      "prefill": {
        "name": "Gaurav Kumar",
        "email": "gaurav.kumar@example.com",
        "contact": "9999999999"
      },
      "notes": {
        "address": "Razorpay Corporate Office"
      },
      "theme": {
        "color": "#3399cc"
      }
    };
    var rzp1 = new Razorpay(options);
    document.getElementById('rzp-button1').onclick = function(e){
```

```
        rzp1.open();
        e.preventDefault();
      }
   });
});
```

File:filter.js

```
$(document).ready(function(){
   $(".filter").on('click',function(){
      var pid=$(this).data('value');
      mydata={id:pid};
      $.ajax({
         url:'/filter_product',
         method:"GET",
         data:mydata,
         dataType:'json',

         success:function(data){
            d=data.prod_data
            res=" ";
            for(i=0;i<d.length;i++){
            res+='<div class="col-md-6 col-lg-4" >'+
                  '<div class="card shadow-hover " >'+

                     '<img class="card-img-top " src="/images/'+d[i].image+'" style="height:25rem" title=""
alt="">'+

                     '<div class="card-body ">'+
                        '<h5 class="m-0"><a class="text-reset text-truncate d-block w-100"
href="#">'+d[i].name+'</a></h5>'+

                        '<div class="font-w-600 text-dark fs-5 ">Rs. '+d[i].price+' '+

                        '</div>'+
                     '</div>'+
                     '<div class="card-footer border-top d-flex ">'+

                        '<div class="ms-auto position-relative z-index-1">'+
                           '<a class="btn btn-outline-primary btn-sm me-2"
href="/product/'+d[i].slugs+'">Preview</a>'+
                           '<a class="btn btn-primary btn-sm px-3" id="addToCart" data-id="{{product.id}}"
data-name="{{product.name}}" data-price="{{product.price}}" data-image="{{product.image}}"><i
class="bi bi-cart fs-5 lh-1"></i></a>'+
                        '</div>'+
                     '</div>'+
                  '</div>'+
```

```
                '</div>';
            }

            $("#filteredProducts").html(res);

        }
    });

});

$(".fprice").on('click',function(){
    var order=$(this).data('value');
    console.log(order);
});

});
```

File: pagination.js

```
function getPageList(totalPages, page, maxLength) {
  if (maxLength < 5) throw "maxLength must be at least 5";

  function range(start, end) {
    return Array.from(Array(end - start + 1), (_, i) => i + start);
  }

  var sideWidth = maxLength < 9 ? 1 : 2;
  var leftWidth = (maxLength - sideWidth * 2 - 3) >> 1;
  var rightWidth = (maxLength - sideWidth * 2 - 2) >> 1;
  if (totalPages <= maxLength) {
    // no breaks in list
    return range(1, totalPages);
  }
  if (page <= maxLength - sideWidth - 1 - rightWidth) {
    // no break on left of page
    return range(1, maxLength - sideWidth - 1)
      .concat([0])
      .concat(range(totalPages - sideWidth + 1, totalPages));
  }
  if (page >= totalPages - sideWidth - 1 - rightWidth) {
    // no break on right of page
    return range(1, sideWidth)
      .concat([0])
      .concat(
        range(totalPages - sideWidth - 1 - rightWidth - leftWidth, totalPages)
```

```
    );
  }
  // Breaks on both sides
  return range(1, sideWidth)
    .concat([0])
    .concat(range(page - leftWidth, page + rightWidth))
    .concat([0])
    .concat(range(totalPages - sideWidth + 1, totalPages));
}

$(function() {
  // Number of items and limits the number of items per page
  var numberOfItems = $("#jar .content").length;
  var limitPerPage = 5;
  // Total pages rounded upwards
  var totalPages = Math.ceil(numberOfItems / limitPerPage);
  // Number of buttons at the top, not counting prev/next,
  // but including the dotted buttons.
  // Must be at least 5:
  var paginationSize = 7;
  var currentPage;

  function showPage(whichPage) {
    if (whichPage < 1 || whichPage > totalPages) return false;
    currentPage = whichPage;
    $("#jar .content")
      .hide()
      .slice((currentPage - 1) * limitPerPage, currentPage * limitPerPage)
      .show();
    // Replace the navigation items (not prev/next):
    $(".pagination li").slice(1, -1).remove();
    getPageList(totalPages, currentPage, paginationSize).forEach(item => {
      $("<li>")
        .addClass(
          "page-item " +
            (item ? "current-page " : "") +
            (item === currentPage ? "active " : "")
        )
        .append(
          $("<a>")
            .addClass("page-link")
            .attr({
              href: "javascript:void(0)"
            })
            .text(item || "...")
        )
        .insertBefore("#next-page");
    });
```

```
  return true;
 }

 // Include the prev/next buttons:
 $(".pagination").append(
  $("<li>").addClass("page-item").attr({ id: "previous-page" }).append(
   $("<a>")
    .addClass("page-link")
    .attr({
     href: "javascript:void(0)"
    })
    .text("Prev")
  ),
  $("<li>").addClass("page-item").attr({ id: "next-page" }).append(
   $("<a>")
    .addClass("page-link")
    .attr({
     href: "javascript:void(0)"
    })
    .text("Next")
  )
 );
 // Show the page links
 $("#jar").show();
 showPage(1);

 // Use event delegation, as these items are recreated later
 $(
  document
 ).on("click", ".pagination li.current-page:not(.active)", function() {
  return showPage(+$(this).text());
 });
 $("#next-page").on("click", function() {
  return showPage(currentPage + 1);
 });

 $("#previous-page").on("click", function() {
  return showPage(currentPage - 1);
 });
 $(".pagination").on("click", function() {
  $("html,body").animate({ scrollTop: 0 }, 0);
 });
});

$(document).ready(function(){
 l=$('.pagination').children('li').length;
 if(l<3)
 $('#page-id').html('');
```

```
});
```

Subfolder: Static>CSS

File: custom.css

```
#loadMore {
    width: 200px;
    color: #fff;
    display: block;
    text-align: center;
    margin: 20px auto;
    padding: 10px;
    border-radius: 10px;
    border: 1px solid transparent;
    background-color: red;
    transition: .3s;
  }
  #loadMore:hover {
    color: red;
    background-color: #fff;
    border: 1px solid red;
    text-decoration: none;
  }
  #prod_count{
      display: none;
  }
  .noContent {
    color: #000 !important;
    background-color: transparent !important;
    pointer-events: none;
  }

.razorpay-payment-button{
  color: #ffffff;
    background-color: #ee4266;
    border-color: #ee4266;
  display: inline-block;
    font-weight: 500;
    line-height: 1.7;
    width: 100% !important;
    text-align: center;
    vertical-align: middle;
    cursor: pointer;
    user-select: none;
```

```
    border: 1px solid transparent;
    padding: 0.65rem 1.25rem;
    font-size: 0.93rem;
    border-radius: 0.25rem;
    transition: color 0.15s ease-in-out, background-color 0.15s ease-in-out, border-color 0.15s ease-in-out,
box-shadow 0.15s ease-in-out;
}
```

Subfolder: Satatic>JS

File: custom.js

```
/**
 * @author pxdraft
 * @version 1.0
 *
 */
(function($) {
    "use strict";
    var CRE = {};
    $.fn.exists = function() {
        return this.length > 0;
    };

    /* ---------------------------------------------- /*
     * Pre load
    /* ---------------------------------------------- */
    CRE.PreLoad = function() {
        document.getElementById("loading").style.display = "none";
    }


    /* ---------------------------------------------- /*
     * Mega Menu
    /* ---------------------------------------------- */

    CRE.MegaMenu = function() {
        var mDropdown = $(".px-dropdown-toggle")
        mDropdown.on("click", function() {
            $(this).parent().toggleClass("open-menu-parent");
            $(this).next('.dropdown-menu').toggleClass("show");
            $(this).toggleClass("open");
        });
    }


    /*------------------
     * Owl Corousel
```

```
----------*/
CRE.Owl = function() {
    var owlslider = $("div.owl-carousel");
    if (owlslider.length > 0) {
        owlslider.each(function() {
            var $this = $(this),
                $items = ($this.data('items')) ? $this.data('items') : 1,
                $loop = ($this.attr('data-loop')) ? $this.data('loop') : true,
                $navdots = ($this.data('nav-dots')) ? $this.data('nav-dots') : false,
                $navarrow = ($this.data('nav-arrow')) ? $this.data('nav-arrow') : false,
                $autoplay = ($this.attr('data-autoplay')) ? $this.data('autoplay') : true,
                $autospeed = ($this.attr('data-autospeed')) ? $this.data('autospeed') : 5000,
                $smartspeed = ($this.attr('data-smartspeed')) ? $this.data('smartspeed') : 1000,
                $autohgt = ($this.data('autoheight')) ? $this.data('autoheight') : false,
                $CenterSlider = ($this.data('center')) ? $this.data('center') : false,
                $stage = ($this.attr('data-stage')) ? $this.data('stage') : 0,
                $space = ($this.attr('data-space')) ? $this.data('space') : 30;

            $(this).owlCarousel({
                loop: $loop,
                items: $items,
                responsive: {
                    0: {
                        items: $this.data('xs-items') ? $this.data('xs-items') : 1
                    },
                    576: {
                        items: $this.data('sm-items') ? $this.data('sm-items') : 1
                    },
                    768: {
                        items: $this.data('md-items') ? $this.data('md-items') : 1
                    },
                    992: {
                        items: $this.data('lg-items') ? $this.data('lg-items') : 1
                    },
                    1200: {
                        items: $items
                    }
                },
                dots: $navdots,
                autoplayTimeout: $autospeed,
                smartSpeed: $smartspeed,
                autoHeight: $autohgt,
                center: $CenterSlider,
                margin: $space,
                stagePadding: $stage,
                nav: $navarrow,
                navText: ["<i class='bi bi-chevron-left'></i>", "<i class='bi bi-chevron-right'></i>"],
                autoplay: $autoplay,
```

```
            autoplayHoverPause: true
        });
    });
  }
}

/* ---------------------------------------- /*
 * lightbox gallery
/* ---------------------------------------- */
CRE.Gallery = function() {
  var GalleryPopup = $('.lightbox-gallery');
  if (GalleryPopup.length > 0) {
    $('.lightbox-gallery').magnificPopup({
      delegate: '.gallery-link',
      type: 'image',
      tLoading: 'Loading image #%curr%...',
      mainClass: 'mfp-fade',
      fixedContentPos: true,
      closeBtnInside: false,
      gallery: {
        enabled: true,
        navigateByImgClick: true,
        preload: [0, 1] // Will preload 0 - before current, and 1 after CRE current image
      }
    });
  }
  var VideoPopup = $('.video-btn');
  if (VideoPopup.length > 0) {
    $('.video-btn').magnificPopup({
      disableOn: 700,
      type: 'iframe',
      mainClass: 'mfp-fade',
      removalDelay: 160,
      preloader: false,
      fixedContentPos: false
    });
  }
}

// Window on Load
$(window).on("load", function() {
  CRE.PreLoad();
});
// Document on Ready
$(document).ready(function() {
  CRE.Gallery(),
  CRE.MegaMenu(),
  CRE.Owl();
```

```
  });

  // Document on Scrool
  $(window).scroll(function() {
  });

  // Window on Resize
  $(window).resize(function() {
  });

})(jQuery);
```

File: voice.js

```
  var search = document.querySelector("#q");
  var search_btn = document.querySelector("#search_btn");

  window.SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  const recognition = new window.SpeechRecognition();

  recognition.continuous = true;
  recognition.interimResults = false;
  recognition.start();
  let p = document.createElement('p');
  const texts = document.querySelector(".testing");

  recognition.onstart = function(){
   console.log("we are active");
  }

  recognition.addEventListener('result', (e) => {
   var text = Array.from(e.results).map(result => result[0]).map(result => result.transcript);
   if (e.results[0].isFinal) {
      str_text = JSON.stringify(text);
      str_text = str_text.split(",");
      str_text = str_text[str_text.length-1].split("")[1];
      str_text = str_text.toLowerCase();
      str_text.trim();
      var shop ="";
      var home = "";
      product_detail = "inactive";

      if(str_text.includes("home")){
       home_btn = document.querySelector("#home-btn");
```

```
            home_btn.click();
            home = "active";
            shop = "inactive";
          }

          else if(str_text.includes("shop")){
            shop_btn = document.querySelector("#shop-btn");
            shop_btn.click();
            shop = "active";
            home = "inactive";
          }

          else if(str_text.includes("login") || str_text.includes("log in") || str_text.includes("signin") || str_text
== "sign in"){
            login_btn = document.querySelector("#login-btn");
            login_btn.click();
          }
          else if(str_text.includes("signup") || str_text.includes("sign up") || str_text.includes("register")){
            signup_btn = document.querySelector("#signup-btn");
            signup_btn.click();
          }

          else if(str_text.includes("find")){
            result = str_text.replace("find ", "");
            search.value = result.trim();
            search_btn.click();
          }

          else if(str_text.includes("go to card")  ||  str_text.includes("go to cart")){
            cart_btn = document.querySelector("#cart-btn");
            cart_btn.click();
          }

          else if(str_text.includes("write") || str_text.includes("right")){
            str_text = str_text.replace("write ", "").trim() || str_text.replace("right ", "").trim();
            if(str_text.includes("first name")){
              first_name = str_text.replace("first name ", "").trim();
              first_name_field = document.querySelector("#id_first_name");
              first_name_field.value = first_name;
            }

            if(str_text.includes("last name")){
              last_name = str_text.replace("last name ", "").trim();
              last_name_field = document.querySelector("#id_last_name");
              last_name_field.value = last_name;
            }

            if(str_text.includes("email")){
```

```
    email = str_text.replace("email ", "").trim();
    email_field = document.querySelector("#id_email");
    email_field.value = email;
   }

   if(str_text.includes("mobile")){
    mobile = str_text.replace("mobile ", "").trim();
    mobile_field = document.querySelector("#id_mobile");
    mobile_field.value = mobile;
   }
  }

  else if(str_text.includes("add to cart")){
   str_text = str_text.replace("add to cart ","");
   if(product_detail == "active"){
    alert(product_detail)
    addtocart_btn = document.querySelector("#addToCart");
    addtocart_btn.click();
   }
   else{
    alert(product_detail)
    $(document).ready(function(){
     if(shop=="active"){
      url = "shop/test";
     }
     else{
      url = "test";
     }
     $.ajax({
      type: "POST",
      url: url,
      data: {
       start: str_text.trim(),
       csrfmiddlewaretoken: '{{ csrf_token }}',
      },
      cache: false,
      beforeSend: function(){
      },
      success: function(response){
       add_to_cart_btn = document.querySelector(".cart"+response);
       add_to_cart_btn.click();
      }
     });
    });
   }
  }

  else if(str_text.includes("show")){
```

```
        str_text = str_text.replace("show ","");
        $(document).ready(function(){
          if(shop=="active"){
            url = "shop/test";
          }
          else{
            url = "test";
          }
          $.ajax({
            type: "POST",
            url: url,
            data: {
              start: str_text.trim(),
              //CSRFToken: token,
              csrfmiddlewaretoken: '{{ csrf_token }}',
            },
            cache: false,
            beforeSend: function(){
            },
            success: function(response){
              product_view = document.querySelector("#product"+response);
              product_view.click();
              product_detail = "active";
            }
          });
        });

      }
      else{
        alert("wrong command");
      }
    }
  })
  recognition.onend = function() {
    recognition.start();

  }
```

manage.py (root)

```
import os
import sys


def main():
    """Run administrative tasks."""
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'E_shop.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

requirements.txt (root)

```
asgiref==3.4.1
autopep8==1.6.0
backports.entry-points-selectable==1.1.1
beautifulsoup4==4.10.0
certifi==2021.10.8
charset-normalizer==2.0.12
distlib==0.3.3
Django==3.2.9
django-jet-reboot==1.3.0
django-paypal==2.0
docopt==0.6.2
filelock==3.4.0
google-api-python-client==1.4.1
gunicorn==20.1.0
httplib2==0.20.4
idna==3.3
Js2Py==0.71
mysqlclient==2.1.0
oauth2client==4.1.3
packaging==21.3
Pillow==8.4.0
pipwin==0.5.1
platformdirs==2.4.0
pyasn1==0.4.8
pyasn1-modules==0.2.8
pycodestyle==2.8.0
pyjsparser==2.7.1
pyparsing==3.0.7
PyPrind==2.11.3
```

pySmartDL==1.3.4
pytube==11.0.1
pytz==2021.3
pytz-deprecation-shim==0.1.0.post0
razorpay==1.3.0
requests==2.27.1
rsa==4.8
six==1.16.0
soupsieve==2.3.1
sqlparse==0.4.2
toml==0.10.2
tzdata==2021.5
tzlocal==4.1
uritemplate==4.1.1
urllib3==1.26.8
virtualenv==20.10.0
whitenoise==6.0.0

Chapter 7: Snapshots of Modules

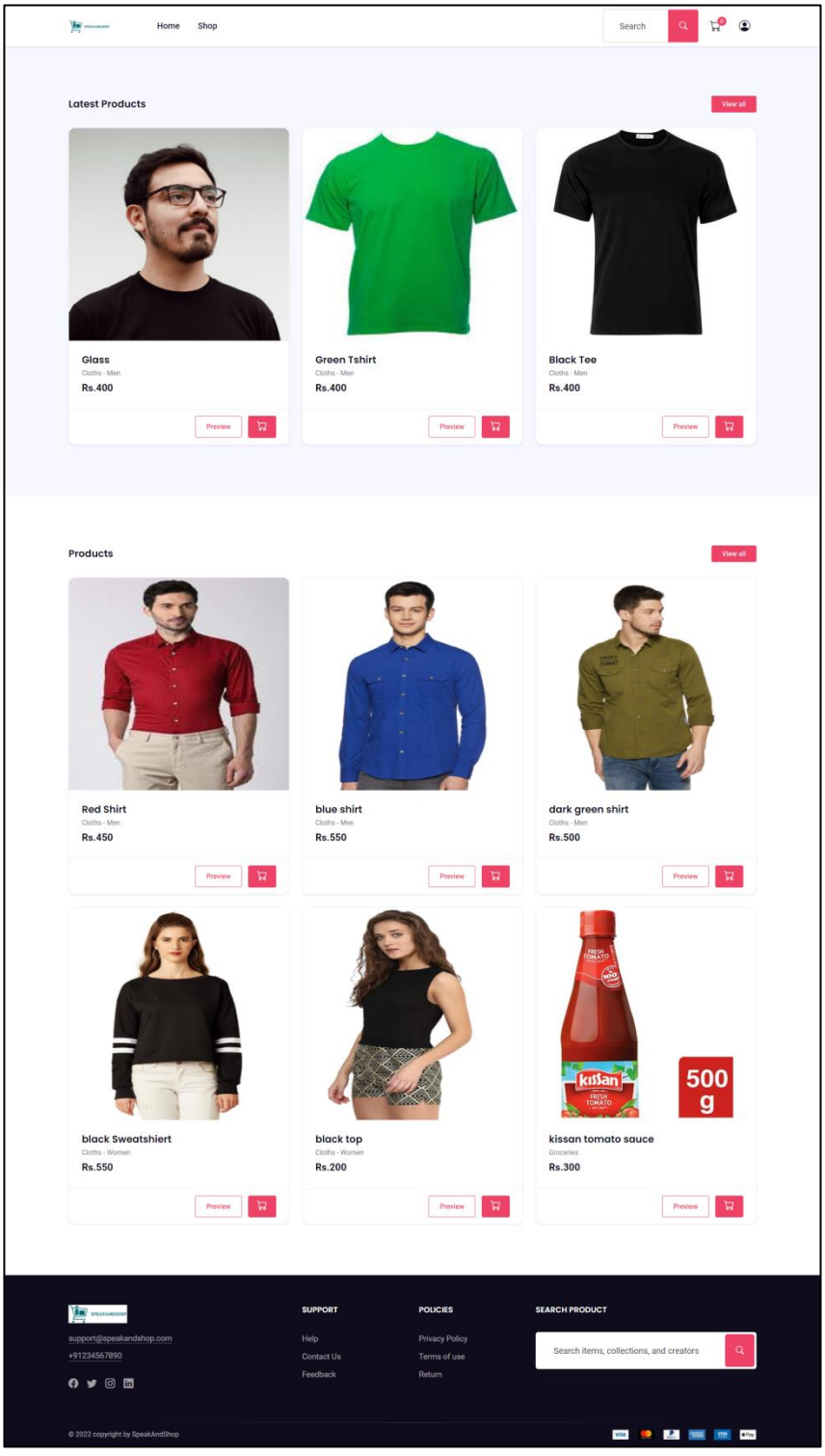


Figure 7.1: Home Screen Full Screen Preview

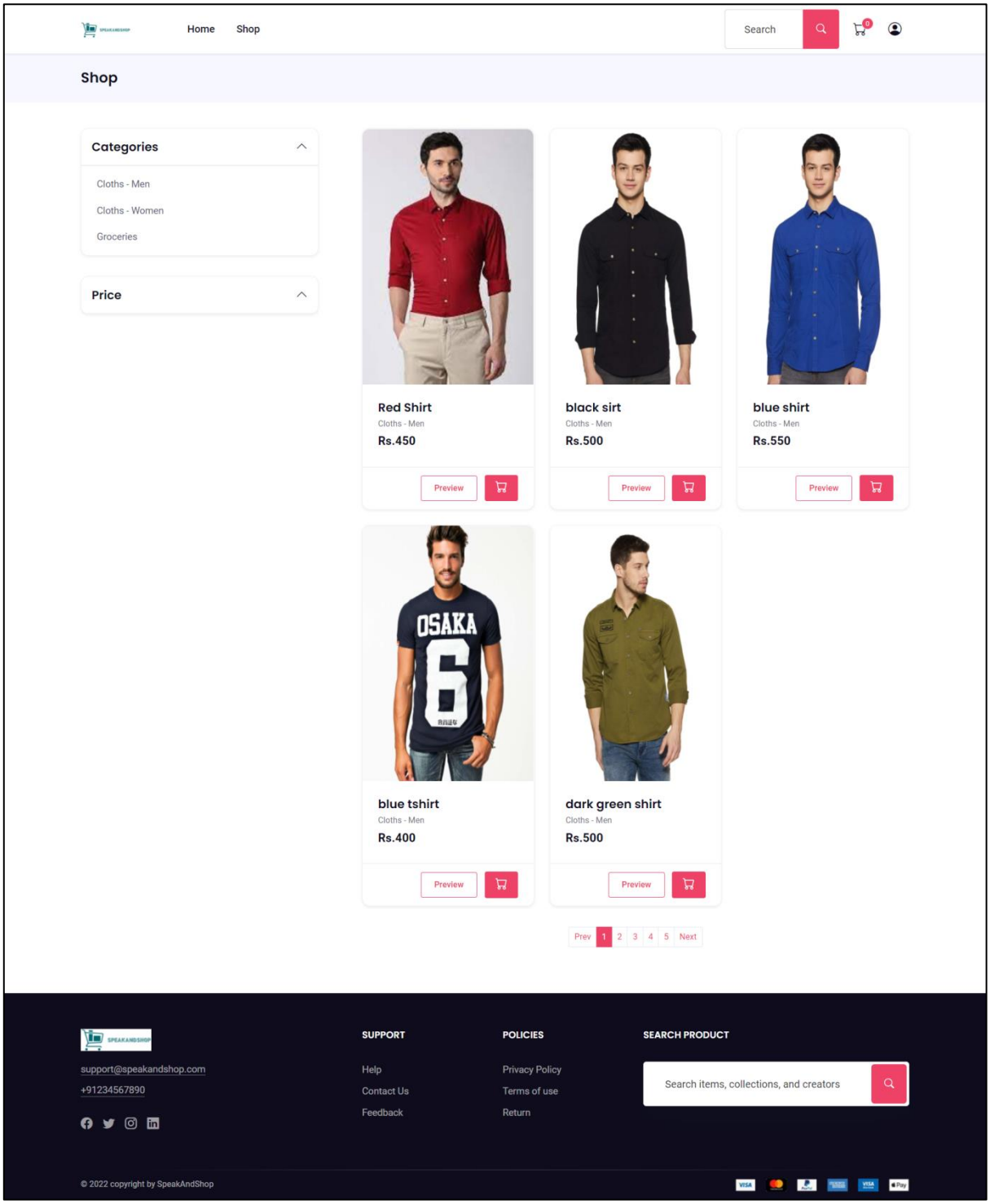


Figure 7.2: Shop Screen Full Page Preview

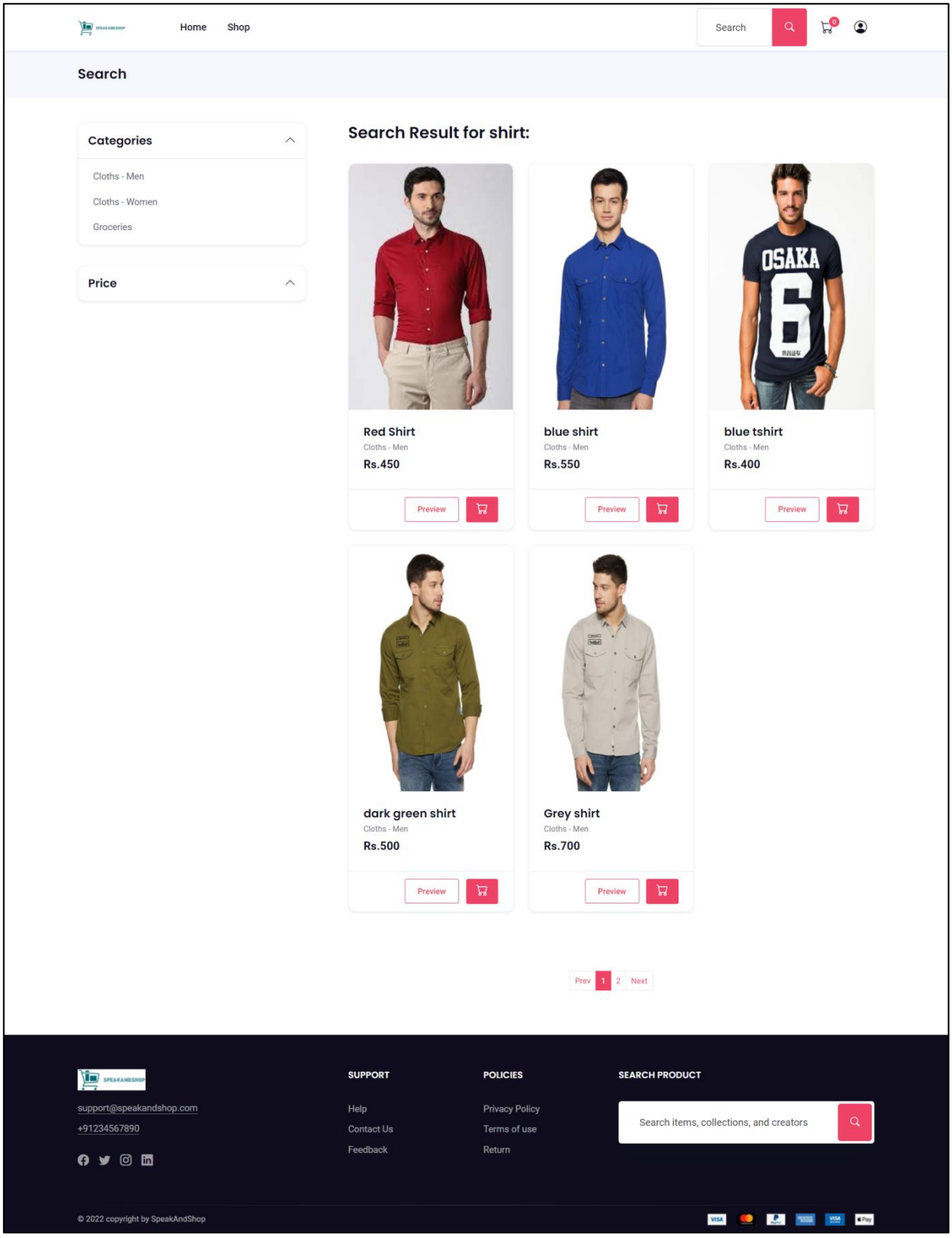


Figure 7.3: *Speak and Shop* Search Screen

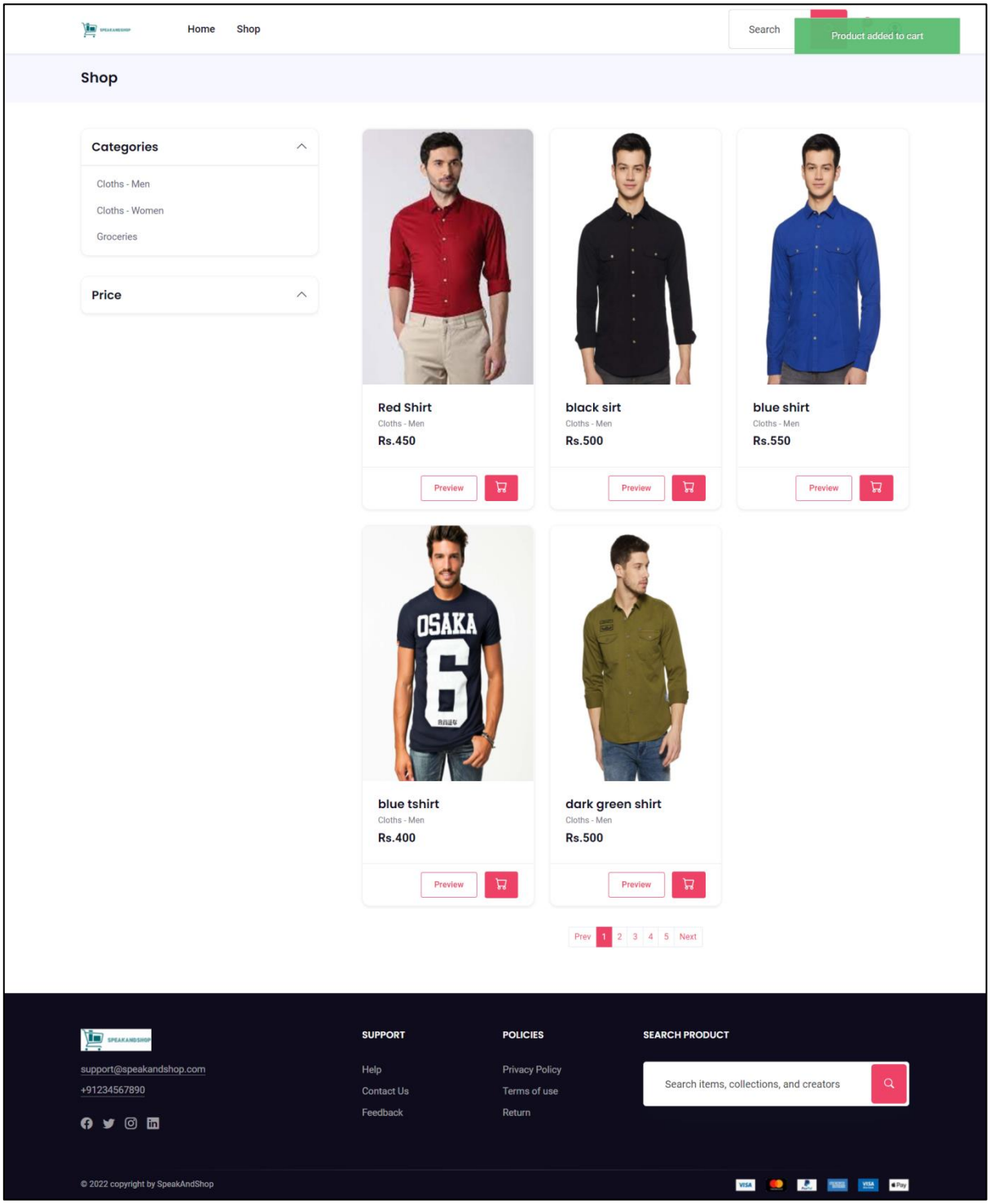


Figure 7.4: Adding Product to Cart

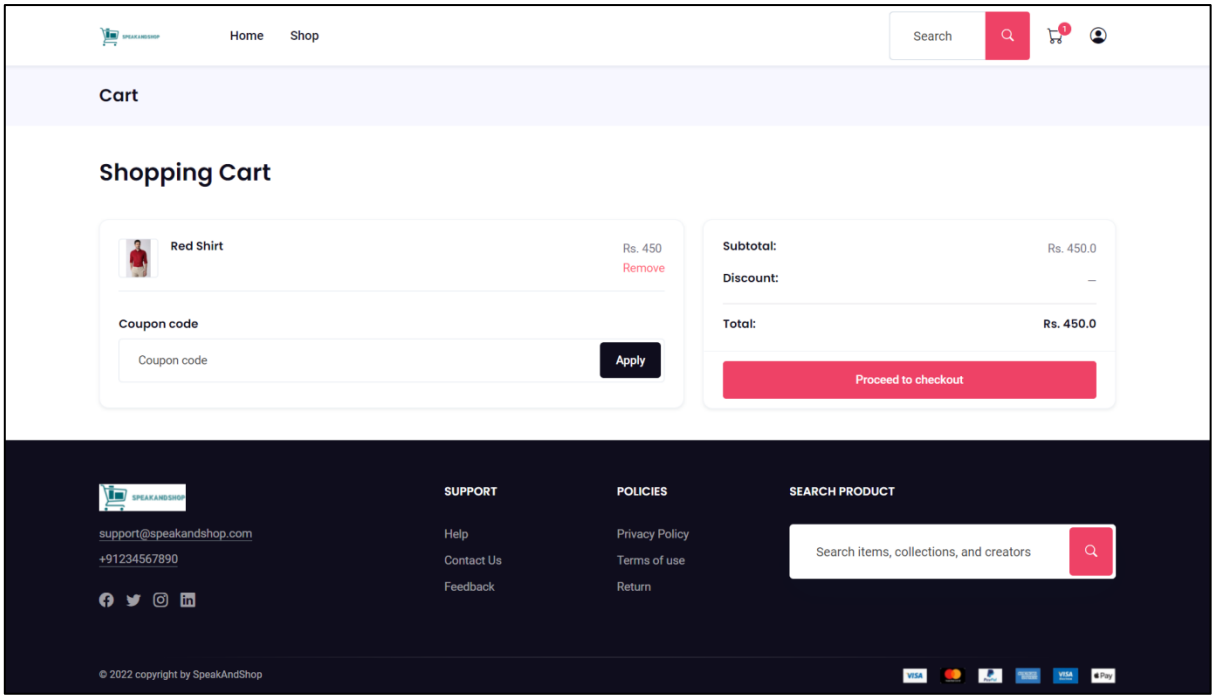
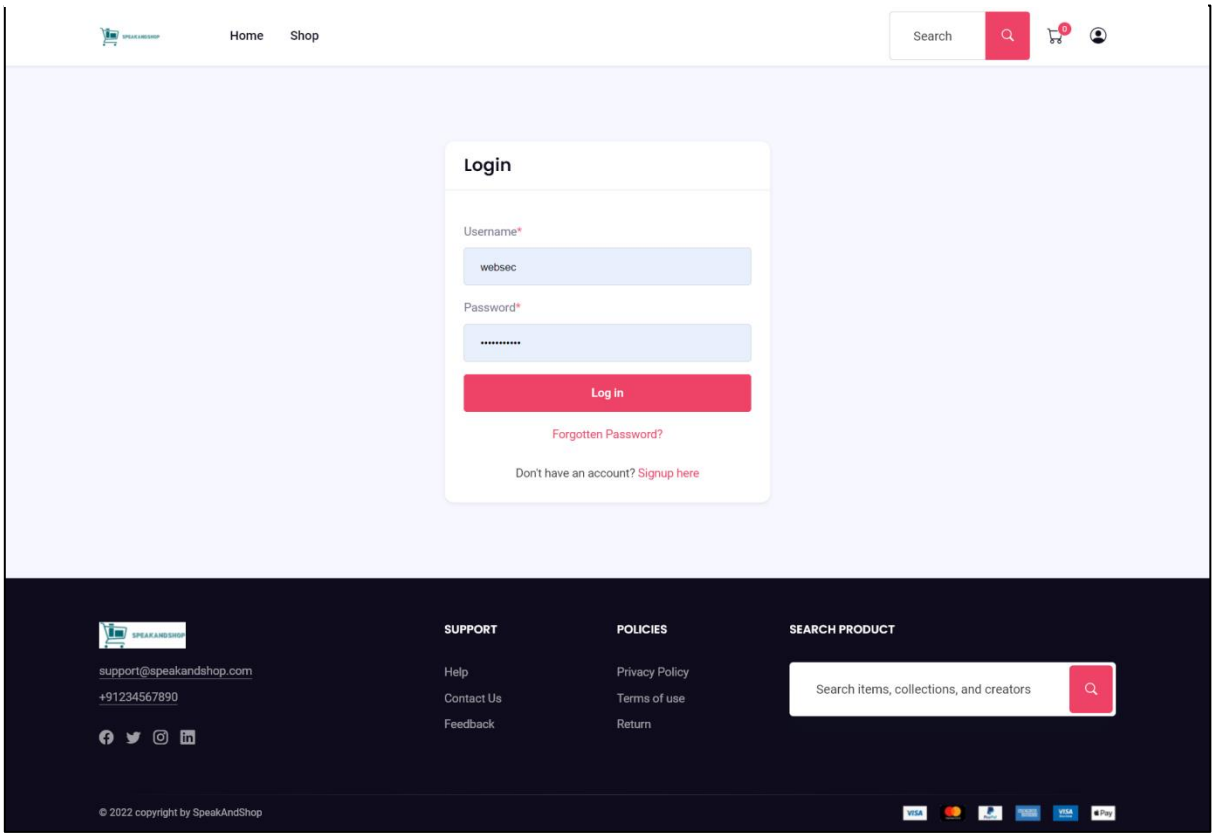


Figure 7.5: Cart Page

Figure 7.6: Login Page after checking the validation on User Checkout Request



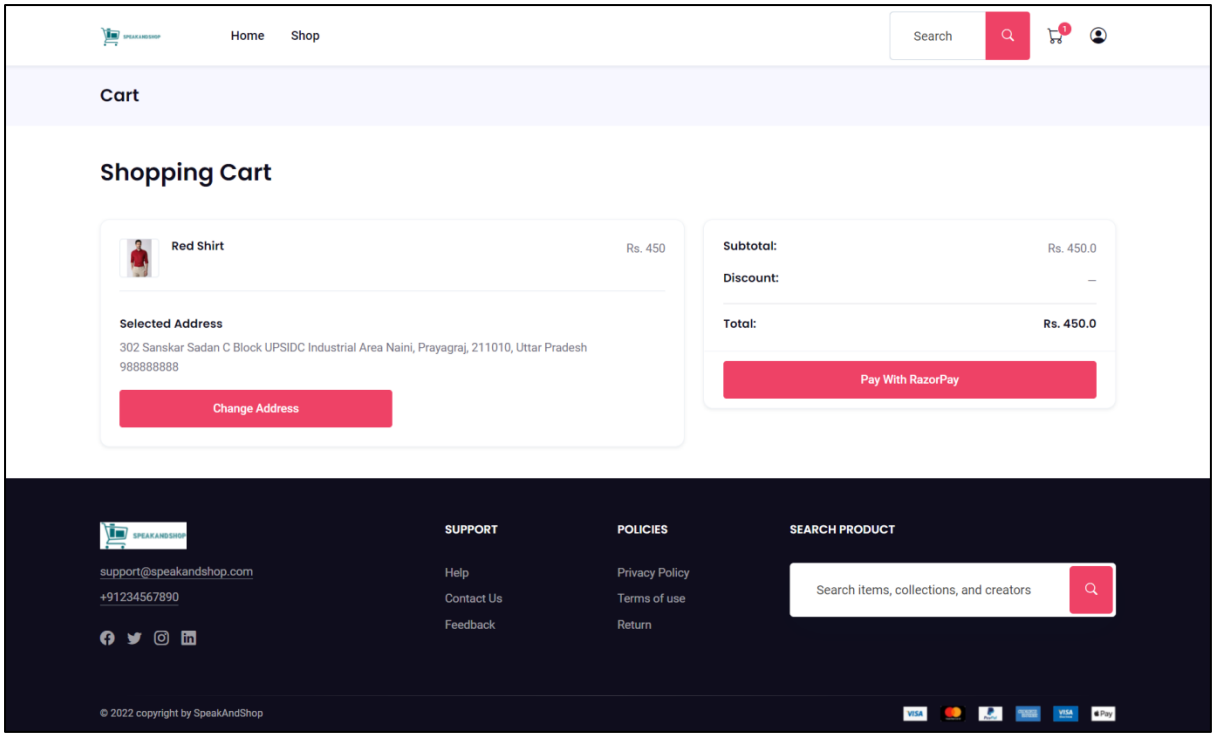
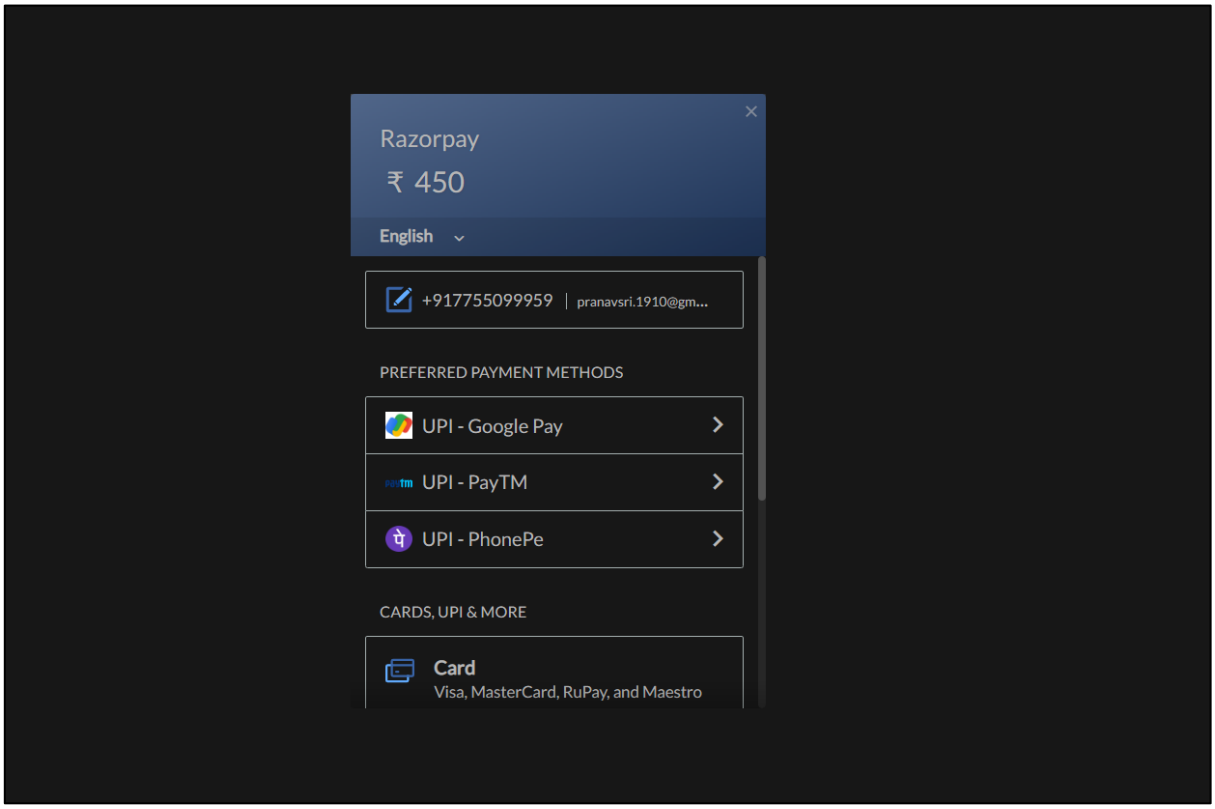


Figure 7.7: Cart Page of Login User

Figure 7.8: Making payment on checkout cart



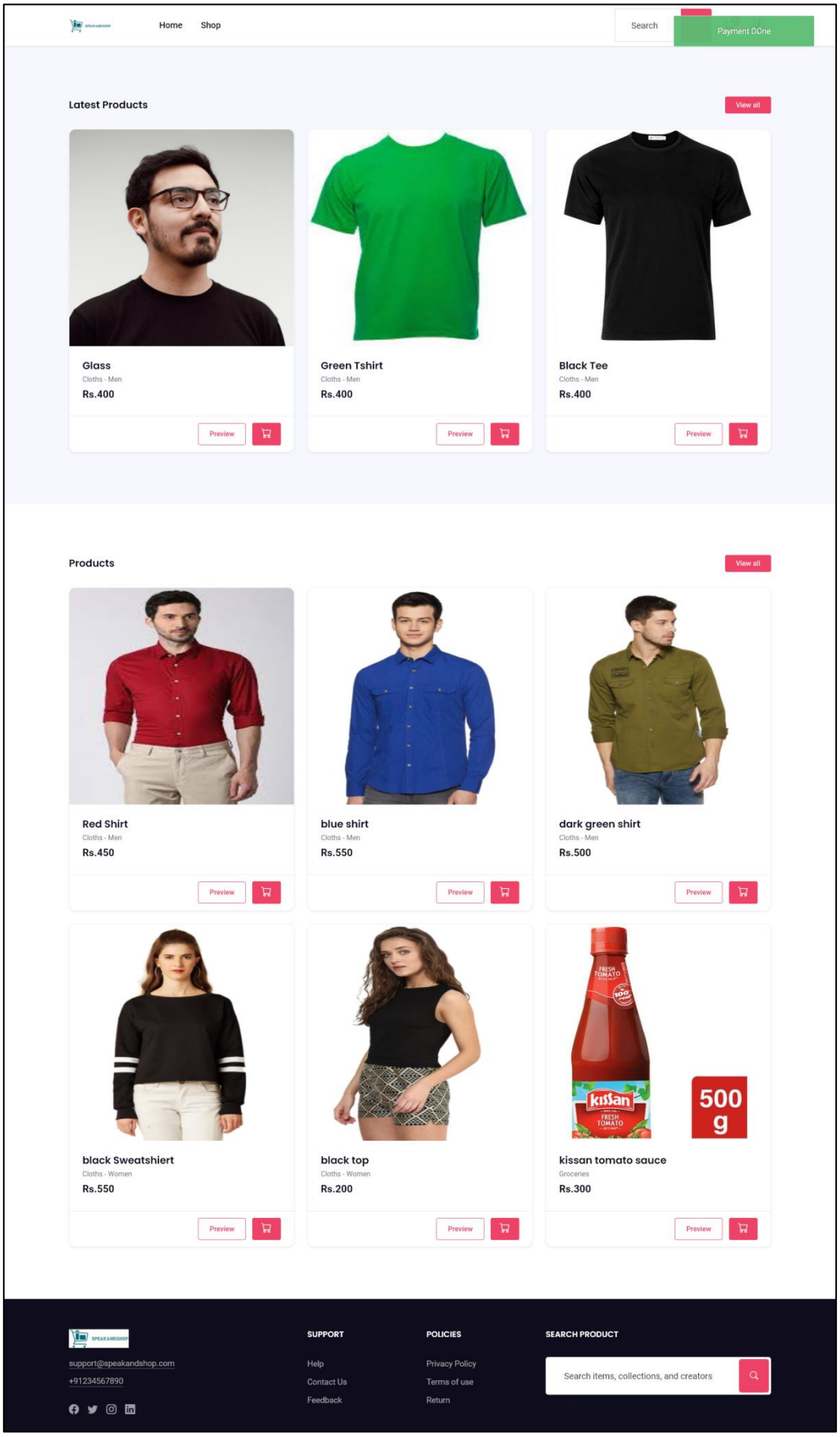


Figure 7.9: Payment Success Message

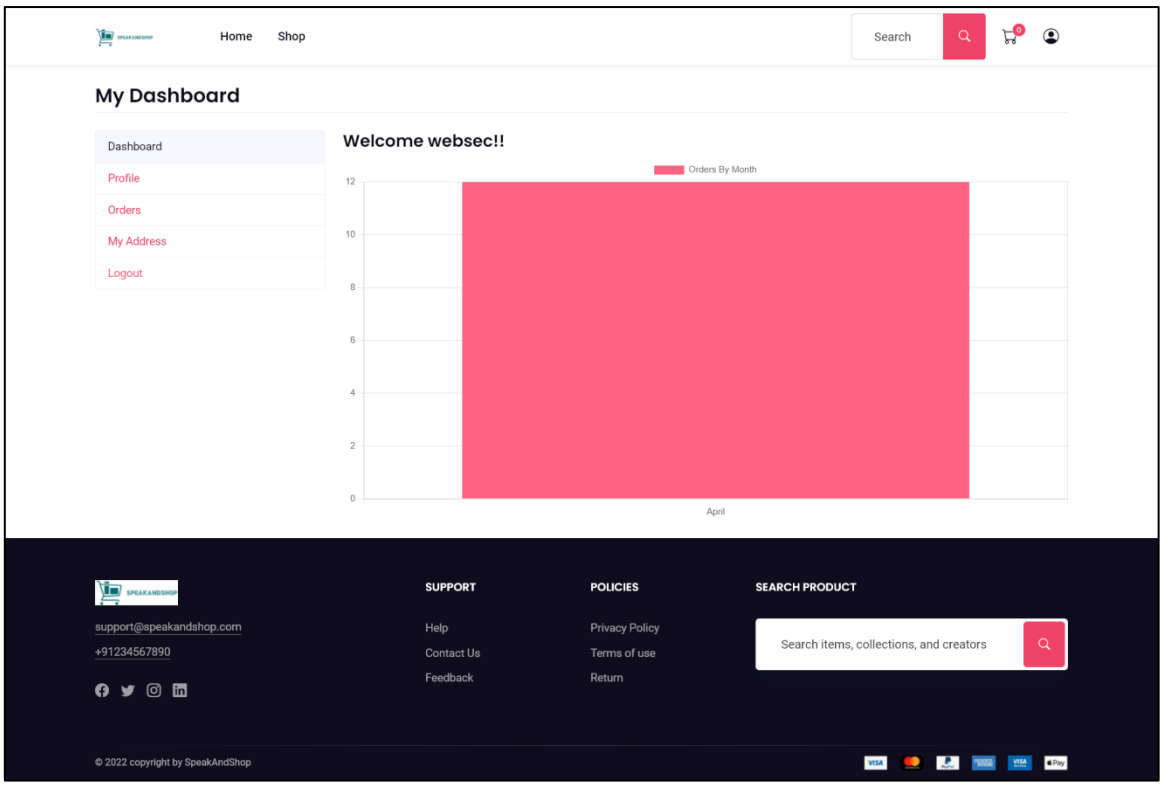
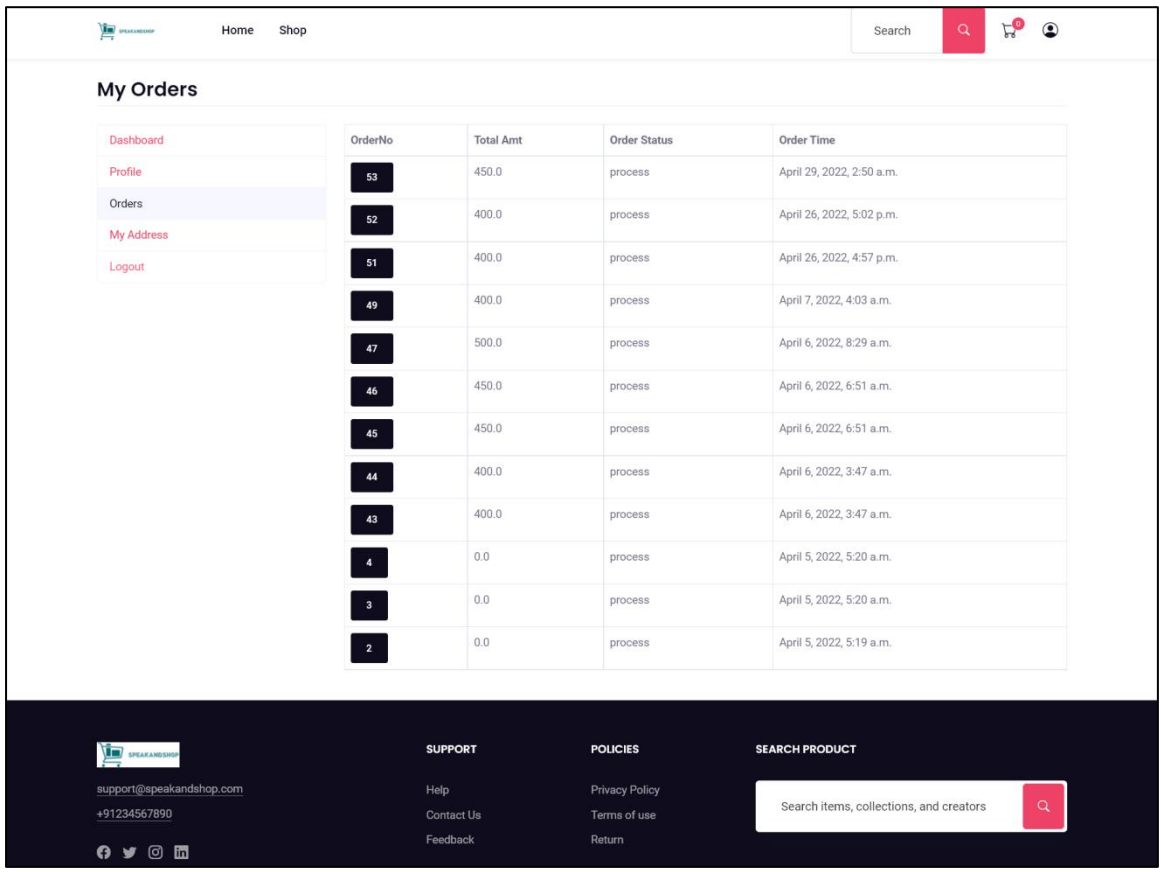


Figure 7.10: User Dashboard Page

Figure 7.11 User Order History Page



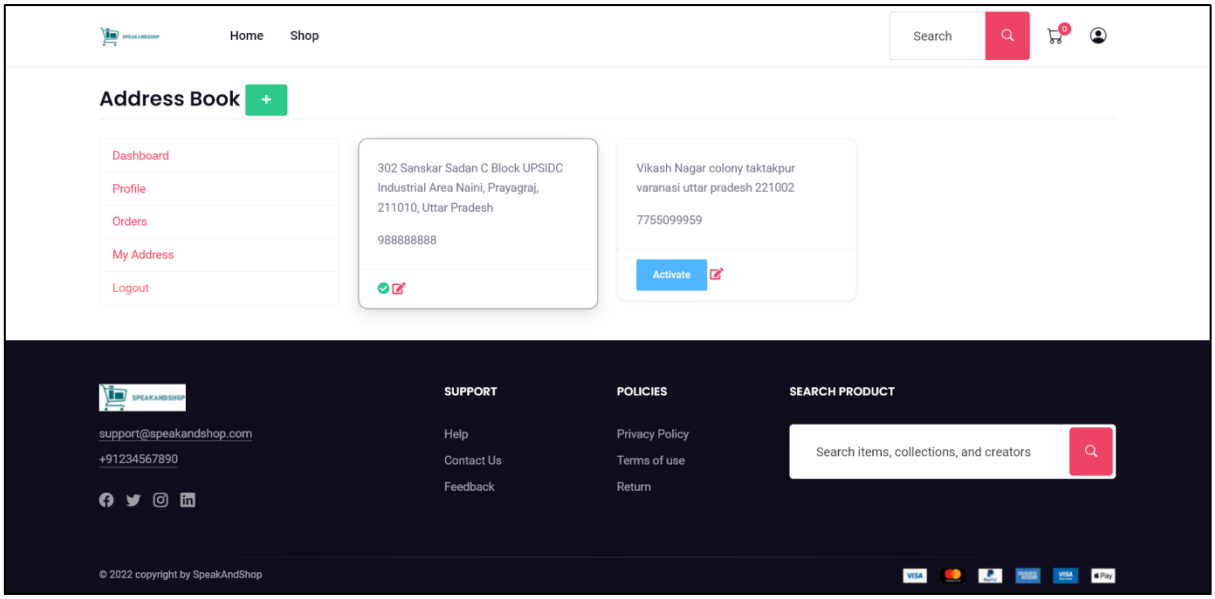
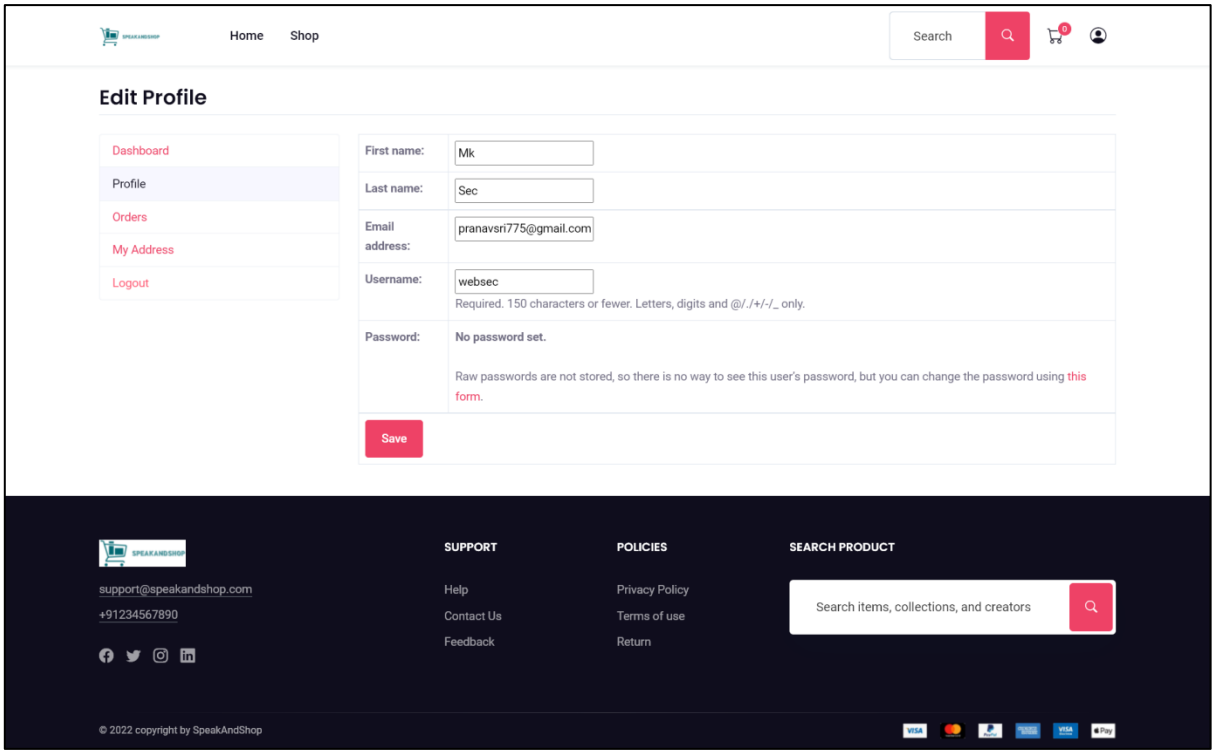


Figure 7.12: User Edit Profile Page

Figure 7.13: User Edit Profile Page



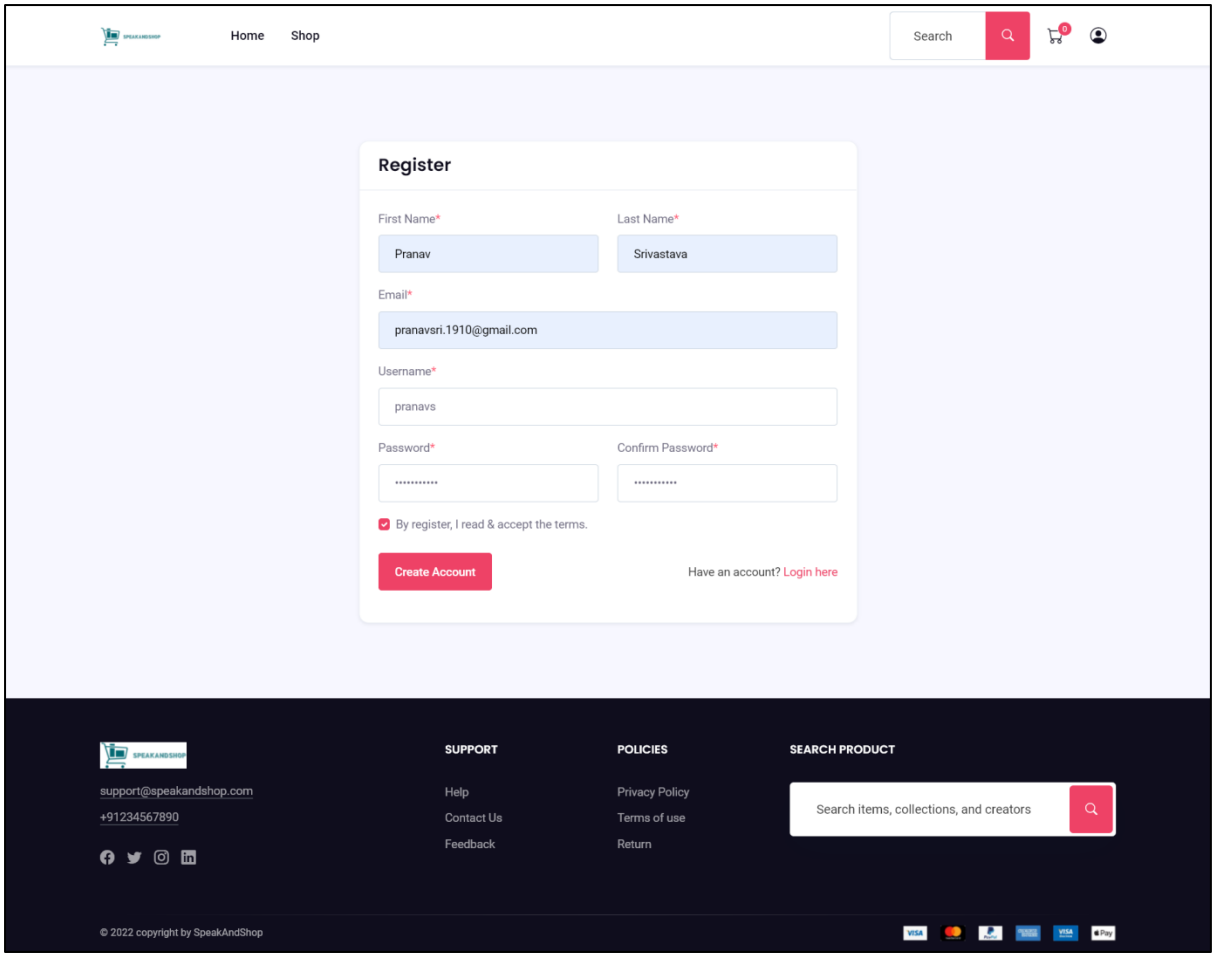
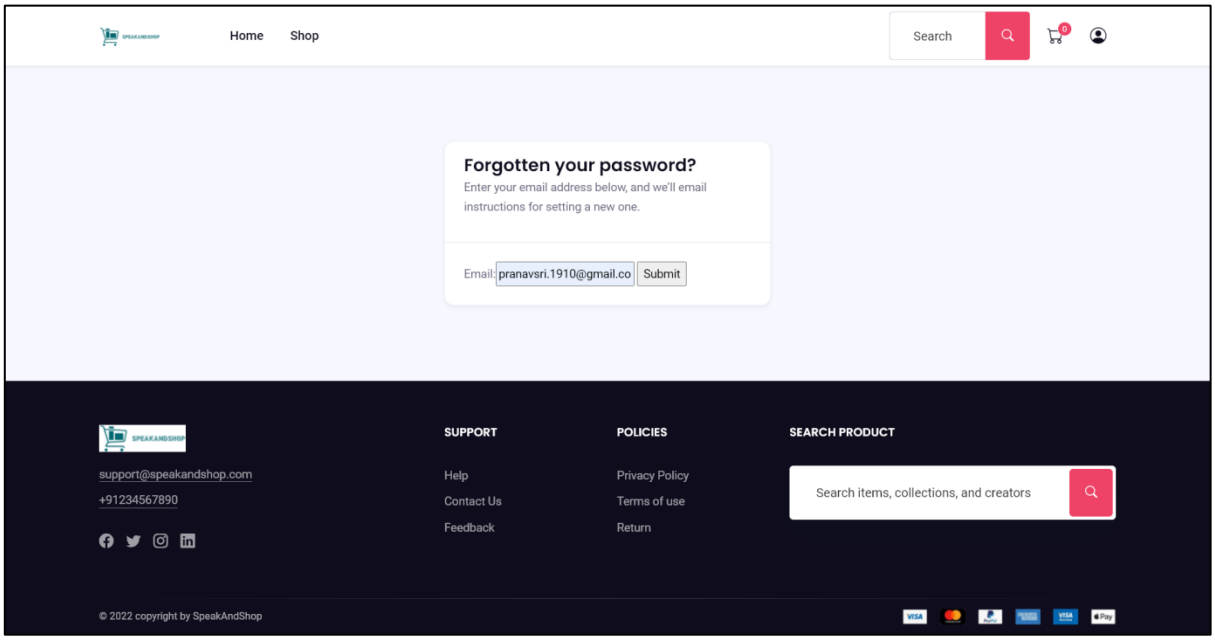


Figure 7.14: User Registration Page

Figure 7.15: Forgot Password Page



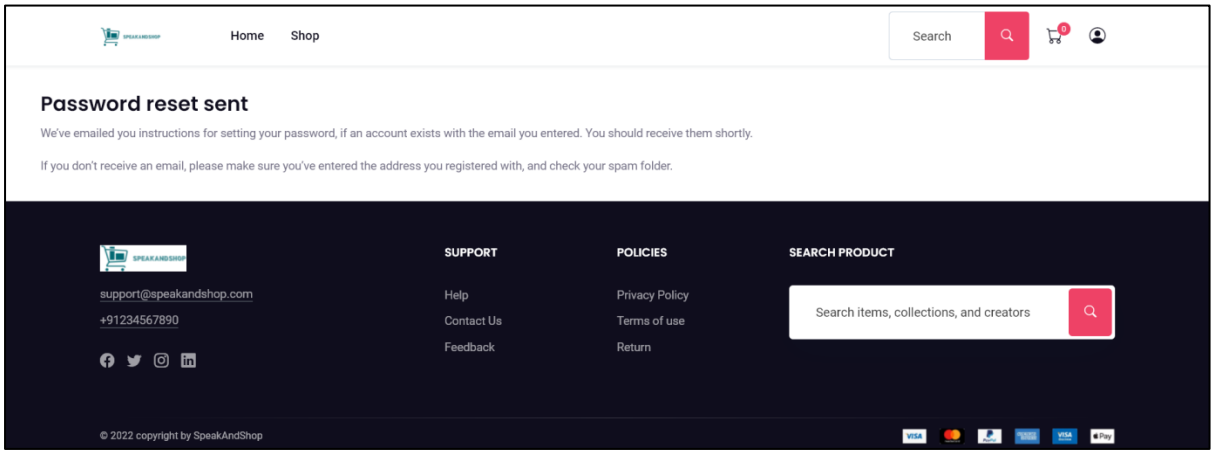
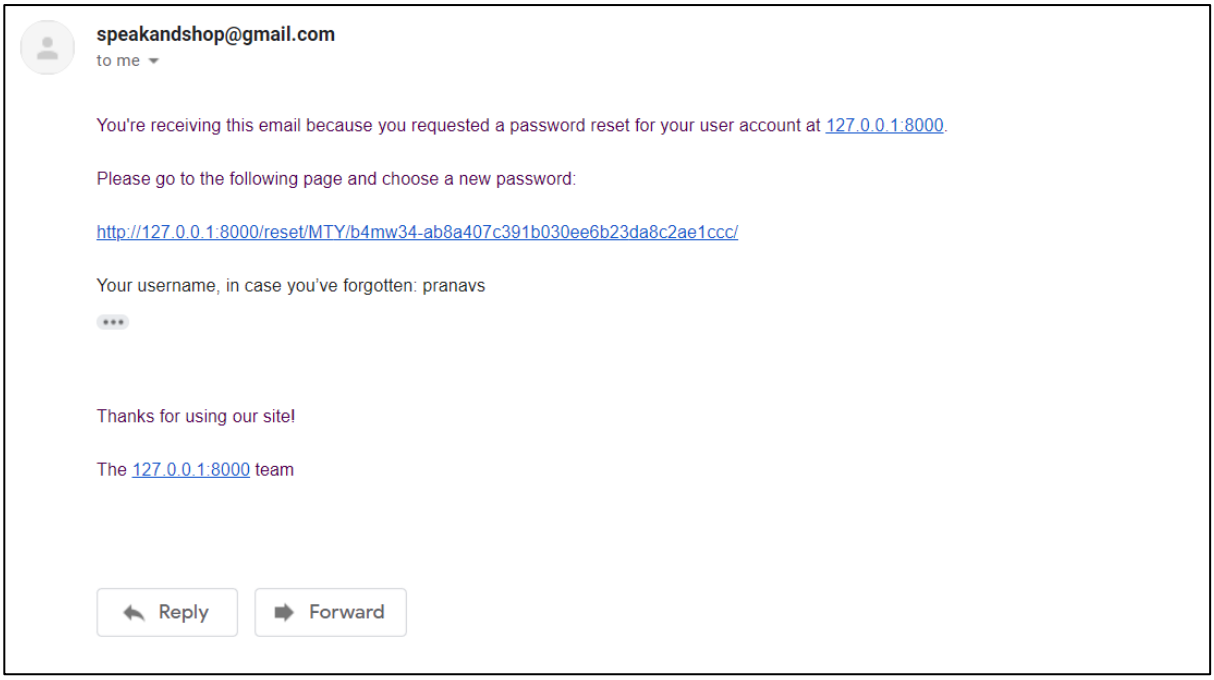


Figure 7.16: Forgot Reset Page

Figure 7.17: Email Received on Forgot Password



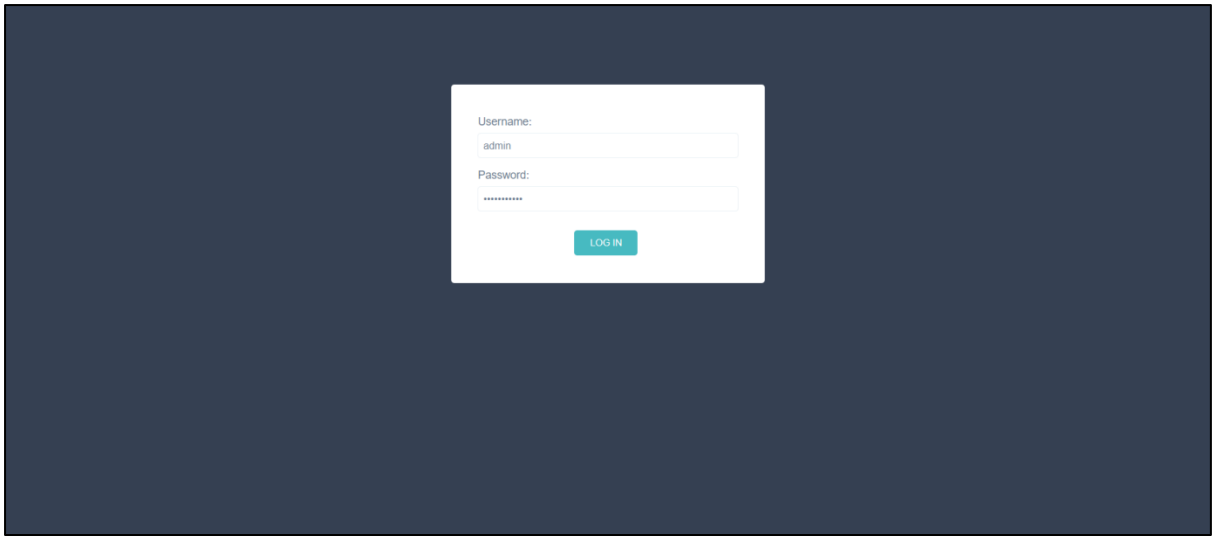


Figure 7.18: Admin Login Screen

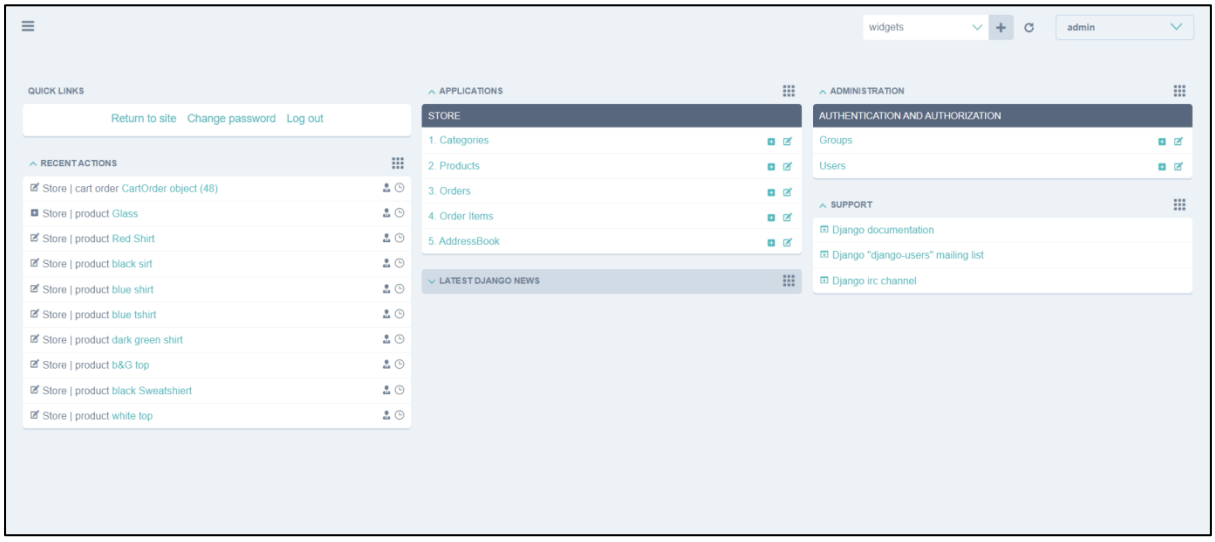


Figure 7.19: Admin Dashboard

Figure 7.20 Category CRUD Screen

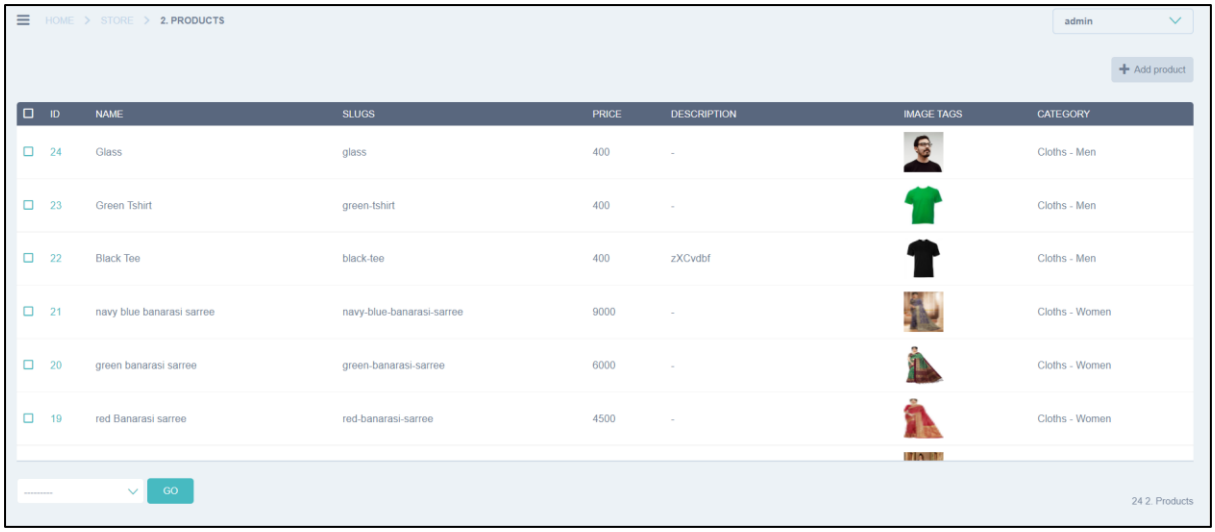


Figure 7.21 Product List Screen with CRUD Action option

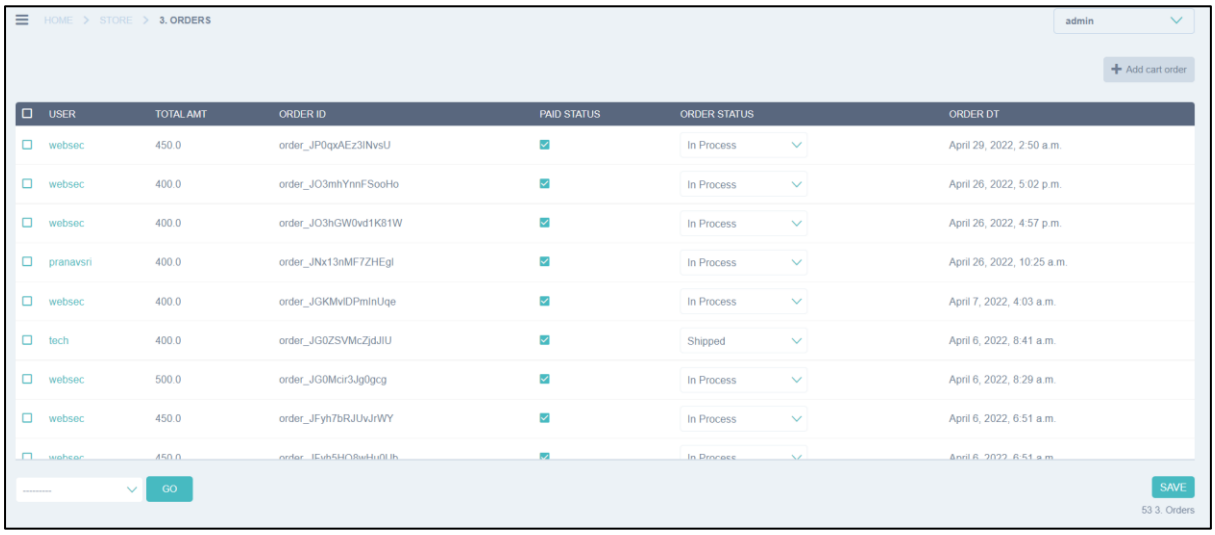


Figure 7.22 Order List Made by User on Admin Dashboard

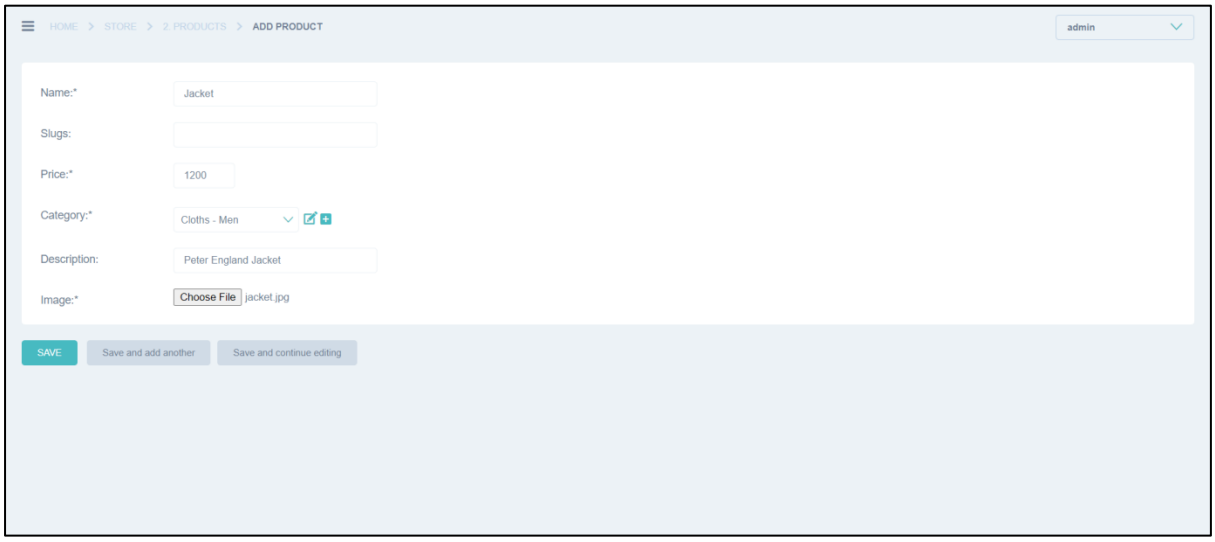


Figure 7.23 Products Adding to the Store

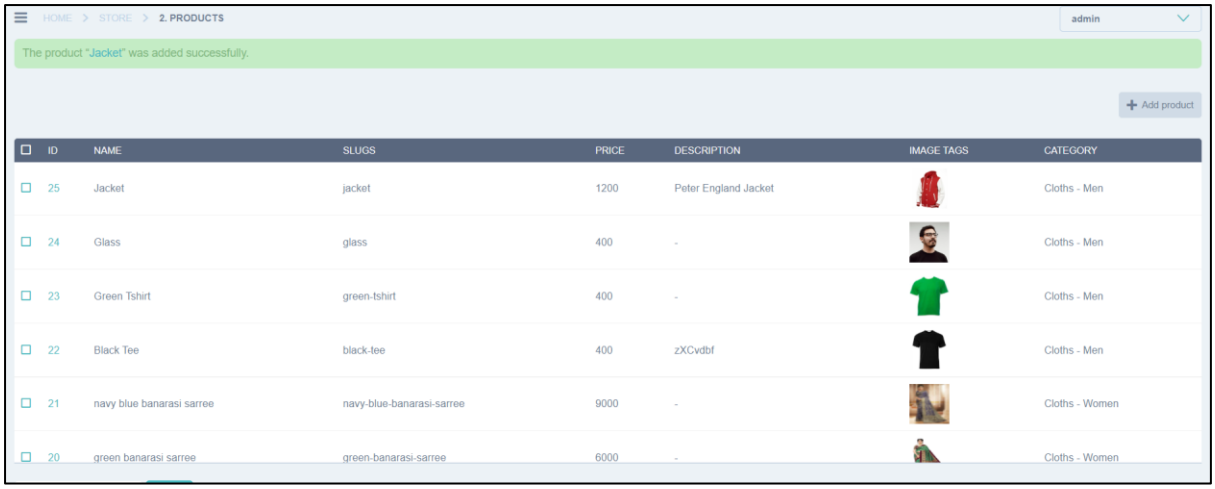


Figure 7.24 Product added and redirected to Product Listing page with proper flash message

Chapter 8: Testing

8.1 Testing Objective

Testing of Speak and Shop application will helps in the prevention of errors and adds value to the product by ensuring conformity to client requirements.

The objective of testing is to ensure:

- Software reliability
- Software quality
- System Assurance
- Optimum performance and capacity utilization

Setting up the system is a complex process and subject to many market-specific variables. To maintain the integrity of the E Commerce system, testing becomes compulsory.

8.2 Testing Design

8.2.1 White Box Testing

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

8.2.2 Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Black box testing has its own life cycle called Software Testing Life Cycle (STLC) and it is relative to every stage of Software Development Life Cycle of Software Engineering.

- **Requirement** – This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
- **Test Planning & Analysis** – Testing Types applicable to the project are determined. A Test Plan is created which determines possible project risks and their mitigation.
- **Design** – In this stage Test cases/scripts are created on the basis of software requirement documents
- **Test Execution**– In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

8.3 Test cases of project

#1) Homepage

- Is it going to auto scroll?
- If yes, at what interval will the image be refreshed?
- When the user hovers over it, is it still going to scroll to the next one?
- Can it be clicked on?
- If yes, is it taking you to the right page and right deal?
- Is it loading along with the rest of the page or loads last in comparison to the other elements on the page?
- Can the rest of the content be viewed?

- Does it render the same way in different browsers and different screen resolutions?

#2) Search

Search algorithms are very important for the success of a retail site because we can't always place what the users want to see right in front of their eyes.

Common tests are:

- Search based on the Product name, brand name, or something more broadly, the category. For example Camera, Canon EOS 700D, electronics, etc.
- Search Results have to be relevant
- Different sort options have to be available- based on Brand, Price, and Reviews/ratings etc.
- How many results to display per page?
- For multi-page results, are there options to navigate to them

Also, search happens in many places. Please take the search drilling down into multiple levels into consideration when validating this functionality.

#3) Product Details Page

Once a user finds a product either through search or by browsing or by clicking on it from the homepage, the user will be taken to the product information page.

Check:

- Image or images of the product
- Price of the product
- Product specifications
- Reviews
- Check out options
- Delivery options
- Shipping information
- In-stock/Out of stock
- Multiple color or variations options
- Breadcrumb navigation for the categories (highlighted in Red below). If navigation such as that is displayed, make sure every element of it is functional.

#4) Shopping Cart

This is the penultimate stage before the user commits to the purchase.

Test the following:

- Add items to the cart and continue shopping
- If the user adds the same item to the cart while continuing to shop, the item count in the shopping cart should get incremented

- All items and their totals should be displayed in the cart
- Taxes as per location should be applied
- A user can add more items to the cart- total should reflect the same
- Update the contents added to the cart- total should reflect that too
- Remove items from the cart
- Proceed to checkout
- Calculate Shipping costs with different shipping options
- Apply coupons
- Don't check out, close the site, and come back later. The site should retain the items in the cart

#5) Payments

Payment methods

Check different payment options

- If allowing check out as Guest, simply finish the purchase and provide an option to register at the end
- Returning customers – Login to check out
- User sign up
- If storing customer Credit card or any other financial information, perform security testing around this to make sure it is secure.(PCI compliance is a must)
- If the user is signed up for a long time, make sure the session is timed out or not. Every site has a different threshold. For some, it is 10 minutes. For some, it might be different.
- Emails/Text confirmation with the order number generated

#6) Categories/Featured Products/Related or Recommended Products

- The most popular FAQ I get from E-commerce testers is: Do I have to test every category/every product?
- The answer is NO.
- If you are a returning customer you will be shown some recommended products on the home page or in your shopping cart.
- Featured products also change almost every day.
- Since these are dynamic elements, the best way to test these parts of the application is to test the algorithm based on which these sections are populated.
- Check your Data mining/BI systems and check from the backend the queries that populate these sections.

#7) After-Order Tests

- Change the Order

- Cancel the Order
- Track the Order
- Returns

#8) Other Tests

- Login
- Contact Us page
- Customer Service page etc.

Chapter 9: Implementation & Maintenance

9.1 Implementation

Implementing an e-commerce shopping cart gives your business the ability to sell your products online day and night, reach new clients, target your ideal market, establish a strong brand, and build closer relationships with your customers by improving their purchasing experience.

Whether you're setting up an online store for the first time or updating your current platform, platform implementation is one of the most complex aspects of launching an e-commerce site. Without the guidance of a consultant with years of e-commerce experience, programming expertise and deep knowledge, your efforts to set up an e-commerce solution can become plagued by cost overruns, programming errors, and delays resulting in poor sales performance and reduced profits.

IMPLEMENTATION STRATEGIES:

- Proper testing & readiness of website
- Reliability of technology
- Readiness of ISP to load website & handle traffic
- Security of lines & payment system
- Customer service staff training
- Readiness of fulfillment process with vendor's warehouse
- Performance check of entire system before actual deployment
- Assignment of a Web team to monitor, maintain & upgrade the system

IMPLEMENTATION MANAGEMENT:

- Project Management
- Estimating resource requirements
- Scheduling tasks & events
- Providing for training and site preparation
- Selecting qualified staff & supervising their work
- Monitoring project's program
- Documenting
- Periodic evaluating
- Contingency planning

IMPLEMENTATION ISSUES:

- Performance
- Personnel

- Integrating front-end commerce site with back-end fulfillment vendors
- Maintenance & enhancements
- Post-implementation planning

IMPLEMENTATION STRATEGIES DISASTER/RECOVERY PLAN:

- Appoint a competent team of representatives (ie - Web designer, IT representatives, and customer service staff)
- Prepare planning
- Draft a disaster/recovery manual

9.2 Maintenance

Having an ecommerce website is much like this. You need to keep up the ecommerce maintenance on the site to make sure you have the best online presence possible. Replace products where needed and fix any problems that may occur.

Maintaining your site isn't just about changing the color scheme or marking things as 'out of stock'. It's about creating great content, keeping it safe from hackers, and making sure the marketing team is doing its job.

Ecommerce Website Maintenance Has Benefits:

a. **Reduced cart abandonment:** A customer is on your site and after scrolling through; they add three pairs to their cart. However, a technical glitch causes the basket to empty. Frustrated, the customer leaves the site and chooses a competitor instead. Keeping these glitches at bay, then, helps arrest your customer churn.
b. **Better user experience:** It may be that as you have added more products to your pages, things have become jumbled. So, it's important to maintain a clear website, ensuring navigation is straightforward. It's all about being clear and organized. Maintaining an ecommerce website ensures a smooth shopping journey and user experience from start to finish. Meaning customers will be much likelier to return.

If you don't maintain:

a. More prone to hackers
b. Likely to fail on other updates in the future
c. Considered suspicious by web browsers
d. Decrease site rankings on Google
e. Site traffic will slow down

3 Types of Ecommerce Website Maintenance:

a. **Security maintenance:** Security is the number one reason for website maintenance. Hackers are always searching for ways to try and break into a

company's site. Especially those websites with access to customer details and financial information. Like every ecommerce site.

b. **Marketing maintenance:** Digital marketing not only drives people to your site, but it also keeps search engines happy. If your site is doing well on Google, then even more people will end up on your site. So, it's important to maintain marketing to make sure things are staying fresh and relevant.
c. **Storefront maintenance:** Your virtual storefront shows you are selling what the customer wants. Look at this landing screen for the confectionery brand.

Chapter 10: Conclusions & Future Aspect

- The biggest advantage of voice e-commerce is how easy it is to use.
- Online shopping more convenient, voice-enabled searches are becoming immensely popular and are being widely adopted.
- Voice search empowers you to communicate fast and obtain results more swiftly, which is an exciting feature for most users.
- Purchasing convenience is a major advantage for customers, allowing them to bypass a lengthy checkout process.
- Getting your customers to leave reviews can be difficult.
- With voice based commerce, the process is simplified.
- Assistive commerce is a broad term that refers to web app that enable people with disabilities to access technology.

REFERENCES

EXAMPLE OF REFERENCING AN ARTICLE FROM INTERNET

- https://getbootstrap.com/docs/
- https://docs.python.org/3/
- https://www.w3schools.com/
- https://stackoverflow.com/

APPENDIX

BIBLIOGRAPHY

RESEARCH PAPER