E-COMMERCE WEB APPLICATION USING DJANGO FRAMEWORK

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Science degree in Computer Science

By

JAGADEESH. M (Reg. No. 39290036)
DHANUSH NARAYANAN. S.D (Reg. No. 39290017)

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH - 2022

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **JAGADEESH. M (39290036), DHANUSH NARAYANAN. S.D (39290017)** who carried out the project entitled **"E-COMMERCE WEB APPLICATION USING DJANGO FRAMEWORK"** under my supervision from December 2021 to March 2022**.**

Internal Guide
Dr. G. KALAIARASI M.E., Ph.D.,

Head of the Department
Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I, **JAGADEESH. M (39290036) and DHANUSH NARAYANAN. S. D (39290017)** hereby declare that the Project Report entitled **"E-COMMERCE WEB APPLICATION USING DJANGO FRAMEWORK"** done by me under the guidance of **Dr. G. KALAIARASI** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D**, **Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D. ,** and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G. KALAIARASI M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In today's fast-changing business environment, it's extremely important to be able to respond to client needs in the most effective and timely manner. If your customers wish to see your business online and have instant access to your products or services. 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. These types of online shops have become part of our daily lives. It helps organizations to reduce the cost to create process, distribute, retrieve and manage the paper-based information by digitalizing the information. The general purpose of E-Commerce store where products can be bought from the comfort of home through the Internet.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER – 1

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Today modern Web applications have grown into complex distributed applications. The worldwide expansion of the internet has considerably contributed to the change of trade and store exchanges. Thus, today modern web applications have grown into complex distributed applications.

E-commerce (electronic commerce or EC) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business.

Electronic commerce (e-commerce) is a fairly new idea, and it is very common practice nowadays for businesses to conduct trade over the Internet. It additionally makes the use of regular technological maintenance to ensure the smooth working of online store sites, monetary exchanges, just as everything to do with giving and delivering items.

It can simply be defined as purchasing and additionally selling items through the internet and is ordinarily associated with online shopping

Here, we are going to develop an e-commerce application, which has penetrated into people's daily life, and the e-commerce market is becoming more and more competitive.

CHAPTER – 2

2. LITERATURE REVIEW

1. Django: Web Development Simple & Fast

Authors: Himanshu Gore; Rakesh Kumar Singh; Ashutosh Singh; Arnav Pratap Singh; Mohammad Shabaz; Bhupesh Kumar Singh; Vishal Jagota; Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 4576 - 4585 , Received 25 April 2021, Accepted 08 May 2021.

Django is a web application framework which is open source and written in the Python language. It uses MVT design structure (MVT stands for Model View Template). Due to its rapid development feature. Django is very demanding in the current Market. It takes less time to build any kind of application. Why we say this Model View Template because this framework will work based upon the model as a database and view as a controlling functionality and template will work as a user side for communication interaction. The Django model will work as database management, we use two main commands like: - python manage.py make migrations Django will deduct the changes in models.py file and ready to send data into the sqlite3 (choose any database). Then we make python manage.py migrate. then the Django system will save all changes in his database system. Then we make one more command Python manage.py run server at the end this will start our project and gives us the localhost address for the project running locally. And views.py file will handle the request for the project to the API's call to template management in requests. we can write the views in the form of python functions.

2. Electronic Commerce: A Study on Benefits and Challenges in an Emerging Economy

Author: Abdul Gaffar Khan; Volume 16, Issue 1, Version 1.0, Year 2016

Information Technology has been playing a vital role in the future development of financial sectors and the way of doing business in an emerging economy like Bangladesh. Increased use of smart mobile services and internet as a new distribution channel for business transactions and international trading

requires more attention towards e-commerce security for reducing the fraudulent activities. The advancement of Information and Communication technology has brought a lot of changes in all spheres of daily life of human being. Ecommerce has a lot of benefits which add value to customer's satisfaction in terms of customer convenience in any place and enables the company to gain more competitive advantage over the other competitors. This study predicts some challenges in an emerging economy.

3. A Study on impact of E-Commerce on India's commerce

Authors: Dr. Rajasekar.S and Sweta Agarwal, Vol. 6, Issue, 03, pp. 7253-7256, March, 2016

E-commerce involves an online transaction. E-commerce provides multiple benefits to the consumers in form of availability of goods at lower cost, wider choice and saves time. The general category of ecommerce can be broken down into two parts: E-Merchandise & E-finance. Many companies, organizations, and communities in India are doing business using E-commerce and also are adopting M-commerce for doing business. Ecommerce is showing tremendous business growth in India. Increasing internet users have added to its growth. Despite being the second largest user base in world, only behind China (650 million, 48% of population), the penetration of e-commerce is low compared to markets like the United States (266 M, 84%), or France (54 M, 81%), but is growing at an unprecedented rate, adding around 6 million new entrants every month. The industry consensus is that growth is at an inflection point. India's ecommerce market was worth about $3.9 billion in 2009, it went up to $12.6 billion in 2013. In 2013, the e-retail segment was worth US$2.3 billion. About 70% of India's e-commerce market is travel related. According to Google India, there were 35 million online shoppers in India in 2014 By 2020, India is expected to generate $100 billion online retail revenue out of which $35 billion will apparel sales are set to grow four times in coming years. This paper is outcome of a review of various research studies carried out on Impact of E-commerce on Indian.

4. Django Based Web Application to Empower Skilled People

Authors: Afroj Satwilkar; Tushar Sawant; Vaibhav Shirke; Santosh V. Jadhav, pg 119-120, Volume 4, Issue 11, May-2021

This paper focuses on development of web application using Django. Django is a modern Python web framework that redefined web development in the Python world. A full stack approach, pragmatic design, and superb documentation are some of the reasons for its success. Django, an open-source Python web framework that saves time and makes web development fun. Django follows the Model-View-Controller (MVC) architectural pattern. Its goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and “pluggability” of components, rapid development, and the principle of DRY (Don't Repeat Yourself). Python is used throughout, even for settings, files, and data models. Technological Implementations in the field of job search has helped skilled people as well as people who wants skilled workers in very important ways. The availability of all job searching sites helps the skilled people a lot in their day-to-day life. The paper illustrates a website model with the help of which skilled people can be able to update their skills and people who wants skilled workers can find them from the same digital platform as well. The project is developed on Django Framework; the backend development is in Python, Jinja2 and SQLite. The frontend consists of HTML, CSS and JavaScript. The project developed is highly efficient, user friendly and simple.

CHAPTER – 3

3. AIM AND SCOPE OF THE PRESENTATION

3.1 AIM OF THE PROJECT

The reason behind our motivation is the current trend of web application integration and interactive features. The trends of online shopping came into existence in the early 90's. Today, there are many online shopping system in place but there are many problems such as hands on experience, fraud and security concern, privacy, lack of full cost disclosure, product reviews and so on. Our project is to look to the current problems and to present on one-stop-shopping platform that is beneficial for both customers and sellers.

3.2 SCOPE AND OBJECTIVE

In our platform, the users can view various products, if they wish to buy any then they have to register and then login to add that product inside their cart. The users or customers can also read the reviews about a product posted by other users before buying it.

The users can add how many products he/she wants to their shopping cart. Then the users are able to set the quantity of each added product inside the cart. Finally, while checkout the users can give their address and the mode of payment and place the respective order. Then the admin can see the customer details with his/her order details and the address where the order should be delivered.

Online Shopping is a very popular in web development. It is mostly used as a business model to earn capital**.** The primary objective of this project is to develop an E-commerce web application to reach maximum customers at the right time to increase sales and profitability of the business including buying and selling goods, transmitting funds or data over the internet.

Here the Admin can upload their product details in website and the customers can visit the Home Page and check whether the product is available or not. Any

member can register and view available products. Only registered member can purchase multiple products regardless of quantity.

The customers can add or view reviews about the products. The customers can add different products to cart and can change the quantity. To place the order, the customer can proceed checkout where they enter the details like address, contact number and selecting the mode of payment. After placing the order, the customer will get an Order ID. With the Order ID, they can track the details of the order in the Track Order Page. If the customer faces any difficulties or issues in the website, a Contact Us page is available to contact Admin for queries and report the problem

3.3 SYSTEM REQUIREMENTS

3.3.1 Hardware Requirements

The selection of hardware is very important in the existence and proper working of any software. When selecting the hardware, the size and requirements are also important to run the software. The minimum hardware requirements are as follows:

- Processor: Intel CORE i3
- RAM: 4 GB
- Disk space: minimum 256 GB

3.3.2 Software Requirements

- Operating System (Windows, MacOS).
- Python, HTML, CSS, JavaScript, Bootstrap.
- Django Framework and SQLite database (which comes by default with Django).
- An Internet Browser (Google Chrome, Microsoft Edge etc).
- Code Editor (Visual Studio code, PyCharm).
- The package manager PIP (pip is a python package-management system written in Python used to install and manage software packages).

3.4 SOFTWARE USED

3.4.1 Python Language

Python language is a high-level, dynamically typed one that is among the most popular general-purpose programming languages. Python is an Interpreted, object-oriented, and high-level programming language. It is called an interpreted language as its source code is compiled to bytecode which is then interpreted. Python's features, among other things, are what make it popular. For instance, it supports dynamic typing and dynamic binding.

3.4.2 Features of Python

- Python is open source. You can download it for free and use it in your application. You can also read and modify the source code.
- The Python framework also has modules and packages, which facilitates code reusability.
- It provides rich data types and easier to read syntax than any other programming
- languages
- It is a platform independent scripted language with full access to operating system API's
- Compared to other programming languages, it allows more run-time flexibility
- It includes the basic text manipulation facilities of Perl and Awk
- A module in Python may have one or more classes and free functions
- Libraries in Pythons are cross-platform compatible with Linux, Macintosh, and Windows
- For building large applications, Python can be compiled to byte-code
- Python supports functional and structured programming as well as OOP
- It supports interactive mode that allows interacting Testing and debugging of snippets of code
- In Python, since there is no compilation step, editing, debugging and testing is fast.

3.4.3 Django Framework

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

3.4.4 Characteristics of Django Framework

- Django was designed to help developers take applications from concept to completion as quickly as possible.
- It includes dozens of extras you can use to handle common web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.
- It takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.
- Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.
- Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

3.4.5 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a webserver or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages.

3.4.6 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the Visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

3.4.7 JavaScript

JavaScript is a high level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage or graphics facilities, relying for these upon the host environment in which it is embedded.

3.4.8 Bootstrap

Bootstrap is a free and opensource CSS framework directed at responsive, mobile first front-end web development. It contains CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

3.5 APPLICATION DEVELOPMENT PLAFTORM

3.5.1 VS Code

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

It can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++.

CHAPTER – 4

4. EXPERIMENTAL OR MATERIAL METHODS

4.1 DESIGN METHODOLOGY

4.1.1 Existing System

E-commerce is growing pretty fast, however, there is no standardization for payment system, resulting unreliability of online payment. There is still another problem following setting up an online store with the web site, the problem of marketing may arise. Another issue, to control the trustworthy of the web site, all shop owners have to register their profiles to the web site, thus it will assure that there is no fraud in online trading on the web site. As the result, the web site will retain its reputation in the market. With this strict rule, the web site can control the quality and trust of its customers. This is very important issue for such a large web site.

4.1.2 Proposed System

The web site, designed as an online shopping center, is separated into two parts: back end and front-end parts. This part provides facility for each store owner to edit and modify information in his own store. Providing validation check for member and store identification, the back-end system can securely protect users' proprietary information. In addition, all page views employ session variables to deter manually defined variables by users. Applying user friendly approach, and focusing on web programming inexperience, the user can effortlessly manage his back-end information. Inside the back end, users can control and view all store information. Besides that, using content management design, the back-end part encompasses with these modules: admin panel, home, register, login, product view, cart, checkout, change password, contact us and track order

4.2 APPLICATION DESCRIPTION

4.2.1 Home Page / Landing page:

In this page, all the products are displayed on the home screen with the image, name and price of the products. Two buttons are also created, one is for adding the item inside the cart and the other is to view the product. If the customers are not logged in then they are not able to add the item inside their cart. They can just see all the products. On the add to cart button, login to add the item will be written. The customers can directly search for the product that they want on the search option given on the navigation bar.

4.2.2 Register Page:

In this page, users can register themselves, in order to view the product details and place an order.

4.2.3 Login page:

Here, users can type their username and password to login.

4.2.4 Product View page:

After clicking on the view button, the customers can view the specific product with their key features and reviews. After reading the key features and the reviews about the product, the customer can buy the product by clicking on the add to cart button. The customer who has brought that specific product can write their review about the product which the other customer is able to read.

4.2.5 Cart page:

By clicking on the shopping cart icon on the navigation bar, customers can see all the added items in the cart. The users can then increase or decrease the quantity of the products according to their requirements.

4.2.6 Change Password page:

All the customers can change their password by going to the change password option.

4.2.7 Contact Us page:

The customers can ask their queries or can contact us by filling a small form. There are two different forms one for the logged in users and others who haven't registered themselves but want to contact us. If the user is a logged in user, then the user just, have to write the message directly else the user needs to first give his name, email and phone before contacting.

4.2.8 Track Order page:

After placing the required order, you will get an order id. That id can be further used for tracking the order. In the track order menu, you have to give your order id for viewing the status of the order.

4.3 SYSTEM ARCHITECTURE

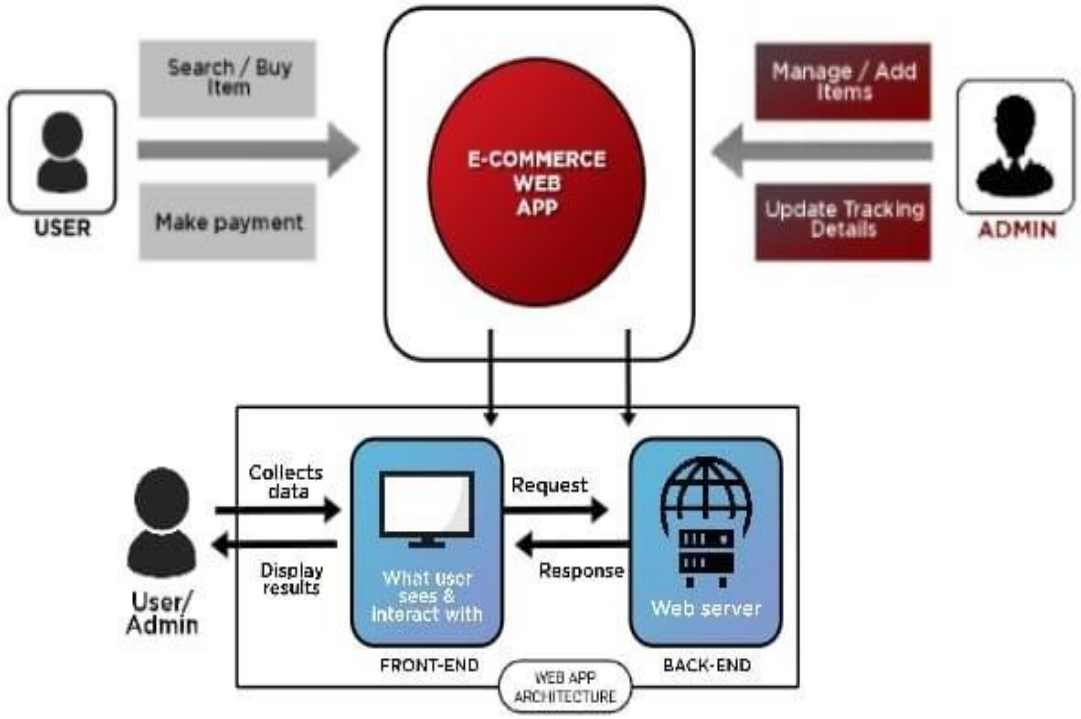

Fig 4.3 Web Application Architecture

4.4 USE CASE DIAGRAM:

4.4.1 Customer Side:

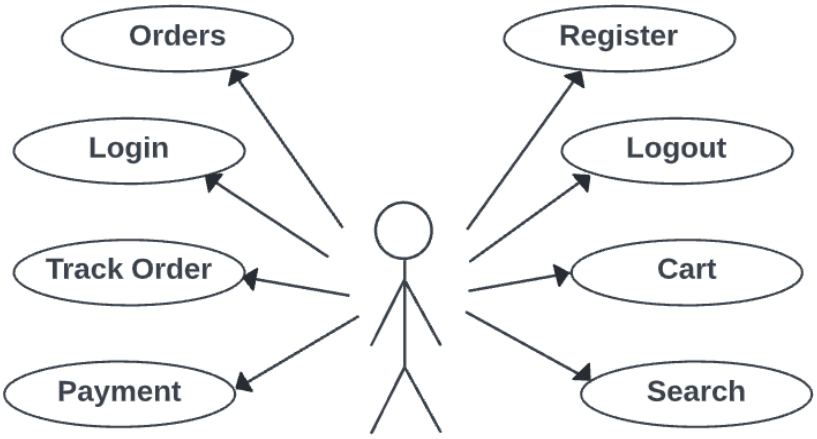

Fig 4.4.1 Customer Side

4.4.2 Admin Side:

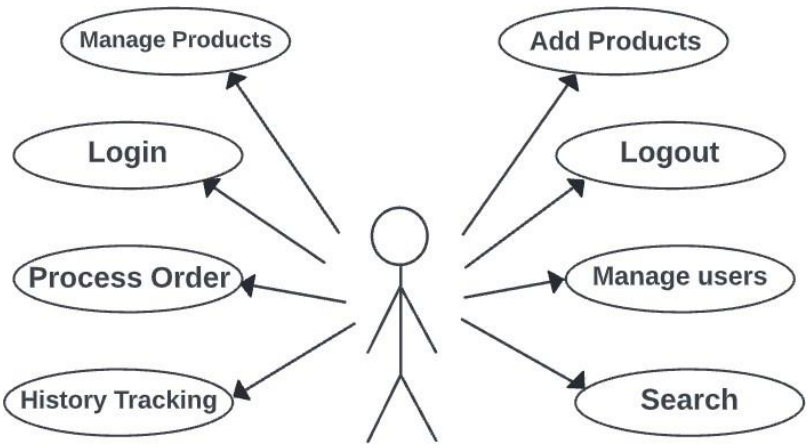

Fig 4.4.2 Admin Side

4.5 IMPLEMENTATION

4.5.1 Creating the Virtual Environment:

After python has been installed on the pc, a virtual environment module needs to be installed on the pc through python's package manager. The virtual environment makes it easy to run different versions of Django in isolation without interrupting the process of one another. The Django module is installed in the virtual environment along with some other modules that are needed to develop the application specifying their versions.

This process is implemented as follows:

- Create Normal Project: Open the IDE and create a normal project by selecting File ->> New Project.
- Install Django: Next, we will install the Django module from the terminal. We will use PyCharm integrated terminal to do this task. One can also use cmd on windows to install the module by running python -m pip install django command
- Check Installed Django version: To check the installed Django version, you can run the python -m django -version command as shown below.
- Create Django Project: When we execute django-admin startproject command, then it will create a Django project inside the normal project which we already have created here. django-admin startproject OnlineShopping.
- Run Default Django web server: Django internally provides a default webserver where we can launch our applications. python manage.py runserver command in terminal. By default, the server runs on port 8000. Access the webserver at the highlighted URL.

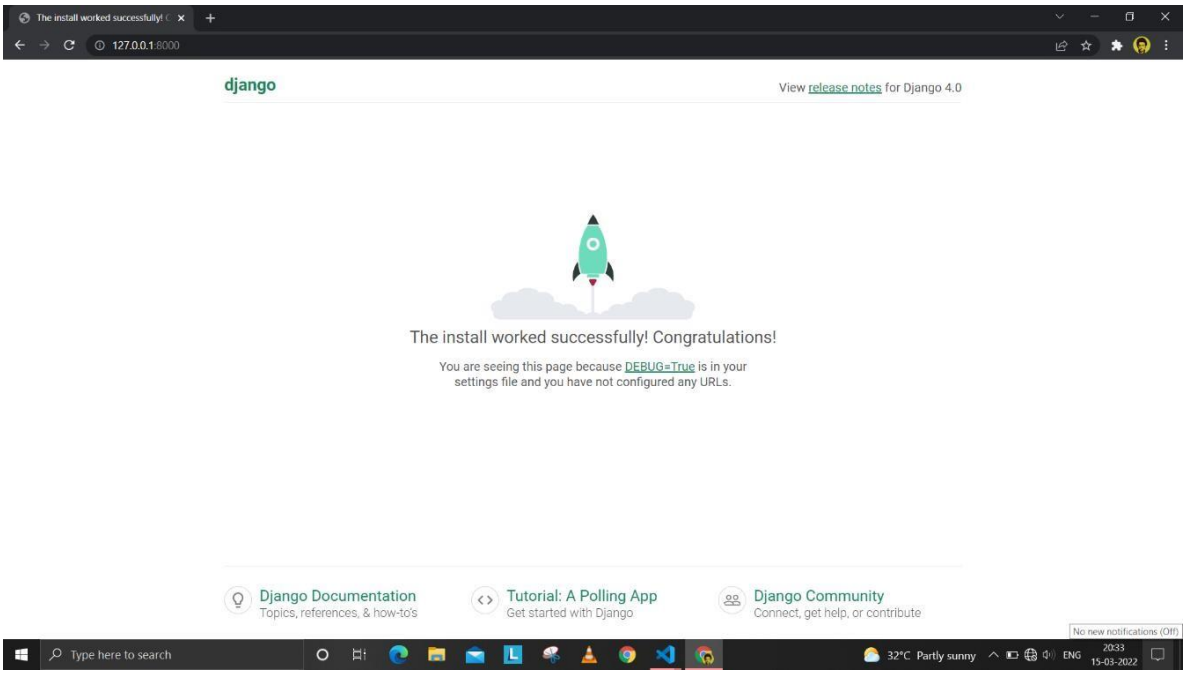

Fig 4.5.1 Django Webserver

4.5.2 Creating various Apps:

When the platform is ready for development, various applications can be created in the project which are relevant to the main idea of the website. For this research, the apps created are as follows: shop, cart, orders, and payment. Each of these apps comes with some important files when generated by the same Django-admin command in the command line of the operating system which allow programmers build the website easily and some are created by the programmer because it is not created by default by the command.

4.5.3 Creating models:

A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Each attribute of the model represents a database field.

So, there are a total of 7 models created for this project.

- Customer model: It saves the basic data of the customers when they register themselves.

- Product model: It saves the data of the product. The admin can add a new product very easily using this model.
- Feature model: The admin can select the product and write any features about it. And all the features of that product will be visible to the users when they view a specific product.
- Review model: All the customers can write a review about a product which the customers can read before buying it.
- Order model: It stores the order details about the customer, mainly the order id.
- OrderItems model: It stores the order id of the customer from the order model and the products with their quantity.
- Checkout Details model: It stores mainly the exact address where the order is to be delivered.

4.5.4 Creating a Superuser (admin):

The Superuser is simply the admin of the site. The admin account needs to be created from the command line through the Django-admin command. The admin account must be created in order to manage the site with a higher privilege than the users of the site. The admin has the privilege to create, retrieve, update, and delete data content and users from the site through the admin site. All models present in the models.py file must be registered in admin.py file in order to allow the models to be visible for the admin.

After creating the models, we need to go to the admin panel to access the created models. Hence, we need a superuser who can access the models from the admin panel. The superuser can make any changes inside the models.

For creating the superuser use the **python manage.py createsuperuser** command.

4.5.5 Creating views for the models:

When the models have been successfully registered in the admin site, creating views for users is another task needed to be accomplished. This refers to the logical functionality between the request and response of the clients and servers. There are two types of views:

- Function-based views: A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image or anything. The view itself contains whatever arbitrary logic is necessary to return that response. This code can live anywhere you want, as long as it's on your Python path. For the sake of putting the code somewhere, the convention is to put views in a file called views.py, placed in your project or application directory. The view function returns an HTML page that includes the request of the user.
- Class-Based views: Class-based views provide an alternative way to implement views as Python objects instead of functions. These allow you to structure your views and reuse code by harnessing inheritance and mixins. They do not replace function-based views, but have certain differences and advantages when compared to function-based views: Organization of code related to specific HTTP methods (GET, POST, etc.) can be addressed by separate methods instead of conditional branching. Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

4.5.6 How does Django work?

To truly appreciate Django, you will need to peek under the hood and see the various moving parts inside. This can be both enlightening and overwhelming. The numbered paths are as follows:

- The browser sends the request (essentially, a string of bytes) to the web server.
- The web server hands over the request to a WSGI server (say, uWSGI) or directly serves a file (say, a CSS file) from the filesystem.
- Unlike a web server, WSGI servers can run Python applications. The request populates a Python dictionary called environ and, optionally, passes through several layers of middleware, ultimately reaching your Django application.

- URLconf contained in the urls.py of your application selects a view to handle the request based on the requested URL. The request has turned into HttpRequest (a Python object).
- The selected view typically does one or more of the following things: It Talks to a database via the models. It Renders HTML or any other formatted response using templates. It Returns a plain text response (not shown). It Raises an exception.
- The Http Response object gets rendered into a string, as it leaves the Django application.
- A beautifully rendered web page is seen in your user's browser.

4.6 APPLICATION FUNCTIONALITIES

4.6.1 User Creation

Django comes with a pre-built register form called UserCreationForm that connects to the pre-built model User. However, the UserCreationForm only requires a username and password (password1 is the initial password and password2 is the password confirmation). To customize the pre-built form, first create a new file called forms.py in the app directory. This new file is created in the same directory as models.py and views.py. Then call UserCreationForm within a new class called NewUserForm and add another field called email. Save the email to the user. Add more fields as needed to the UserCreationForm.

4.6.2 Cookies

Django provides a session framework that lets you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client side, and storing all the related data on the server side. So the data itself is not stored client side.

4.6.3 Search Functionality

A common task for web applications is to search some data in the database with user input. In a simple case, this could be filtering a list of objects by a category. A more complex use case might require searching with weighting, categorization,

highlighting, multiple languages, and so on. This document explains some of the possible use cases and the tools you can use.

4.6.4 Cart Functionality

In Django-SHOP the cart's content is always stored inside the database. In previous versions of the software, the cart's content was kept inside the session for anonymous users and stored in the database for logged in users. Now the cart is always stored in the database. This approach simplifies the code and saves some random-access memory, but adds another minor problem: From a technical point of view, the checkout page is the same as the cart. They can both be on separate pages, or be merged on the same page. Since what we would normally name the "Checkout Page", is only a collection of Cascade Plugins, we won't go into further detail here.

4.6.5 Order Functionality

During checkout, at a certain point the customer has to click on a button named "Place Order". This operation performs quite a few tasks: One of them is to convert the cart with its items into an order. The final task is to reset the cart, which means to remove its content. This operation is atomic and not reversible.

4.6.6 Change Password Functionality

Django does not store raw (clear text) passwords on the user model, but only a hash (see documentation of how passwords are managed for full details). Because of this, do not attempt to manipulate the password attribute of the user directly. This is why a helper function is used when creating a user. To change a user's password, you have several options: manage.py changepassword *username* offers a method of changing a user's password from the command line.

- It prompts you to change the password of a given user which you must enter twice.
- If they both match, the new password will be changed immediately.
- If you do not supply a user, the command will attempt to change the password whose username matches the current system user.

4.6.7 Contact Us Functionality

A contact form is a common feature of many websites that provides a way for users to get in touch with the site's administrators without having to open their email or hop on the phone. In a Python Django application, a contact form can be used to store a user's contact information in the site's database.

4.7 TESTING

To affirm the E-commerce store assessments had been made at distinct stages of the tasks. We checked the reliability of all of the functions. The test is built on the customer/user and admin side. The customer test proved that an account could be created, login can be established, the cart can be loaded with products and the customer can check out when done shopping. The administrator can login into the admin panel afterwards. The admin can then manage all the contents in the store.

The customer can only view the products, if he has logged in as a user. Without being a customer/user, he is unable to view any products which are shown on the home page. Some quantity of products was added into the shopping cart, and then I proceeded to checkout. After successful checkout, the cart became empty. This indicates that the cart works appropriately as it should and the Order ID is shown in a pop-up. The customer can manipulate his cart, such as updating the cart or adding a product to cart. The search bar shows the results of the product search. If the user didn't type anything, a message will be displayed saying that the user forgot to type.

CHAPTER – 5

5. RESULTS AND PERFORMANCE ANALYSIS

5.1 ADMINISTRATION PANEL

The admin can add the product to the website, where he will give the product name, product price and image, then save it and that product displayed in the website. These product data are stored has tables in Django default database SQLite.

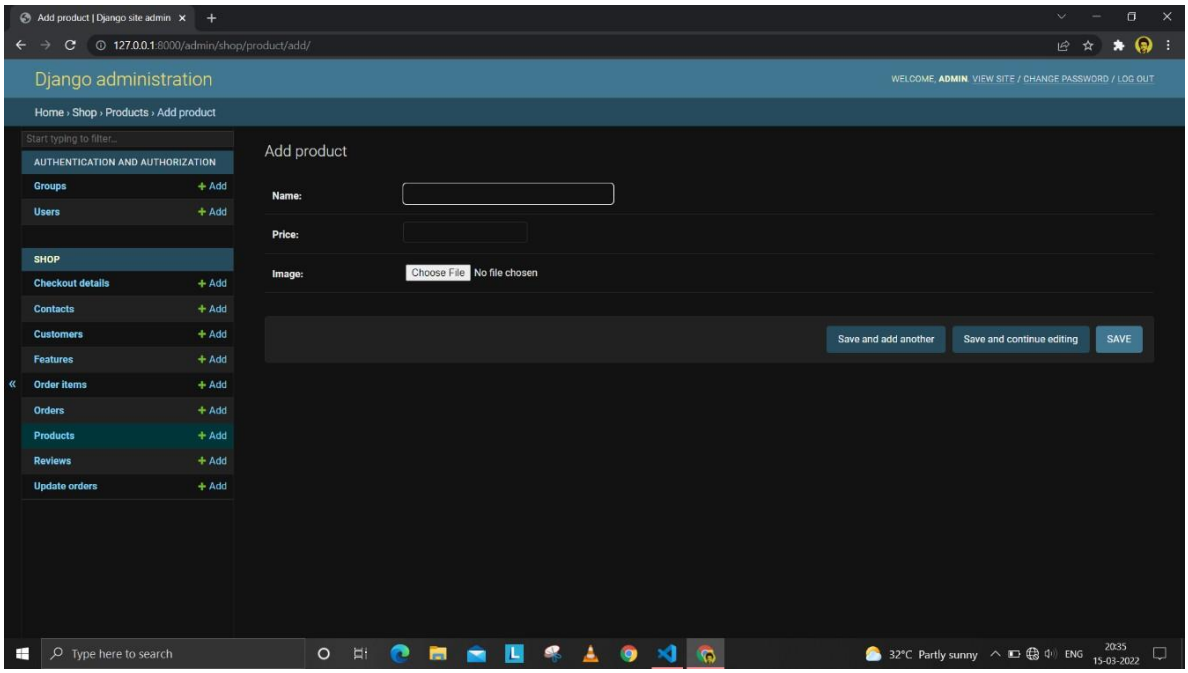

Fig 5.1 Add product feature in Django Administration panel

5.2 HOME PAGE BEFORE LOGIN

The home page is displayed successfully with all the products, navigation bar with search functionality and footer at the bottom. If you want to purchase you have to login as a customer. If you don't have an account then click register. It will redirect to register page is displayed successfully with user registration form fields and an URL which directs them to login page. The login page is displayed successfully in order to login users can type their username and password.

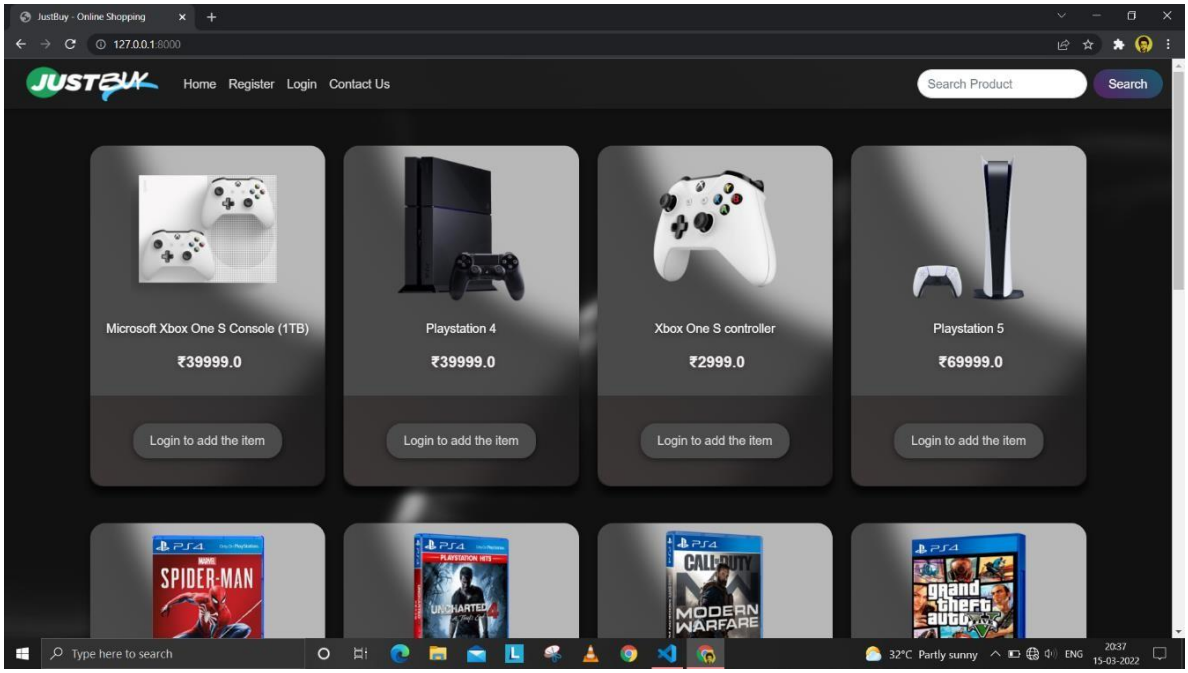

Fig 5.2 Home page with added products

5.3 HOME PAGE AFTER LOGIN

After the user is logged in successfully, the cart will be displayed in the navigation bar which indicates that they can now view the products and add the products to the cart. The products selected by the user will be shown in the order summary with quantity increase and decrease functionality. For further process, checkout button is available. The checkout page is shown successfully with the product summary which was finalized by the customer. Under this, checkout details will be displayed with form fields required to make shipment and selection of payment mode will be available. After the order is placed, a pop-up containing the Order ID is displayed and then the user will be redirected to home page where the cart becomes empty. In order to track the progress of the orders which was placed by the customer, you can check the, in the Track order page by entering a valid Order ID in the track order text field. If the user faces any problems or issues, they can report that issues in the contact us page, which is available in the navbar. After entering the message in the description form field, a pop-up is displayed

successfully which indicates that the message was submitted to the admin. To report a problem or to ask any queries, login is not mandatory.

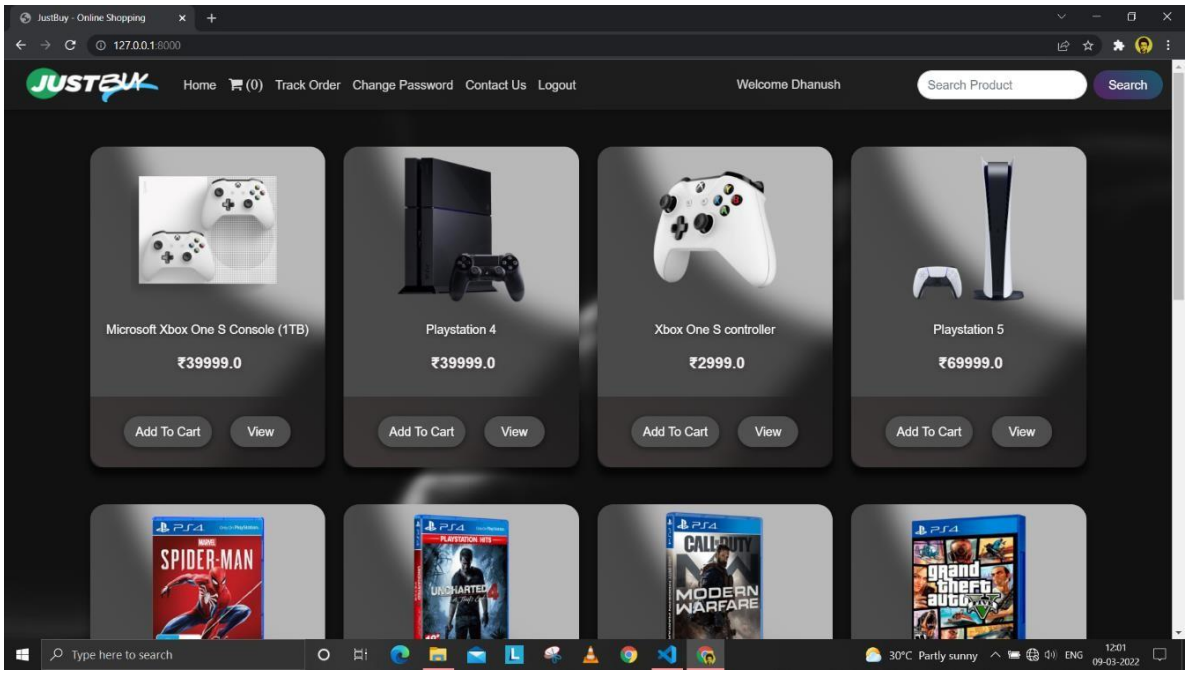

Fig 5.3 Home page after login

CHAPTER – 6

5. CONCLUSION AND FUTURE WORKS

6.1 CONCLUSION

E-commerce is continuously progressing and is becoming more and more important to business as technology continuous to advance and is something that should be taken advantage of and implemented. This E-commerce platform is designed to provide a web-based application that would make searching, viewing and selection of a product is easier.

The user can search for a product interactively and the search engine refine the products available based on the user's input. Then the user can view the full specifications and select the products, the user can see the products in the cart and proceeds to checkout where they enter the address details and select the mode of payment. The administrator can verify the orders. However, the customer can still look at their status of the orders in the Track us page using the order ID.

With this platform, more opportunities will be created for profit and advancements for businesses, while creating more options for both the consumers and sellers.

6.2 FUTURE WORKS

- Separate invoices need other than order summary details need to be implemented.
- User profile customization need to be added.
- Emails and notifications need to be sent to customers for new arrivals or discounts.
- Categorizing of all products should be implemented.
- There have to be language varieties so that non-English users and customers can shop easily without any difficulty.

REFERENCES

[1] Carl Burch, Django, a web framework using Python: tutorial presentation, Journal of Computing Sciences in Colleges, Volume: 25, Issue: 5, 2010, Page: 154 – 155.

[2] Sheetal Taneja; Pratibha Gupta R, Python as a tool for web server application development, JIMS8I-International Journal of Information Communication and Computing Technology, Volume: 2, Issue: 1, 2014, Page: 77 – 83.

[3] Kavya S.L; Dr.Sarathambekai S, Python Libraries and Packages for Web Development - A Survey, International Journal of Innovative Research in Technology, Volume: 5, Issue: 12, 2019, Page: 462 – 464.

[4] Surya Teja N, A Study on Different Framework Architectures, International Journal of Innovative Research in Science, Engineering and Technology, Volume: 7, Issue: 4, April 2018, Page: 4099 – 4104.

[5] Adamya Shyam; Nitin Mukesh, A Django Based Educational Resource Sharing Website: Shreic, Journal of Scientific Research, Volume: 64, Issue: 1, 2020, Page: 238 – 252.

[6] Ahmed Yunus; Md Masum, Design and Development of an E - Commerce System in a Rapid Organized Way, International Journal of Science and Research, Volume: 9, Issue: 1, 2020, Page: 1358 – 1375.

[7] Busari O.A; Adebisi O.A; Adeagal.I; Oni A.A, Development of an Online Shop with Python Web Framework (Django), International Journal of Advanced Research in Science, Engineering and Technology, Volume: 8, Issue:5, 2021, Page: 17293 – 17299.

[8] Roger Fournier, A Methodology for Client/Server and Web Application Development, Prentice Hall PTR, Yourden Press, 1998.

[9] Patrick J. Lynch, Sarah Horton, Web Style Guide: Basic Design Principles for Creating Web Sites, Yale University Press, Published in 2009.

[10] Ralph Grove, Web Based Application Development, Jones & Bartlett Publishers, 2009.

APPENDIX

A. SOURCE CODE

MODELS.PY:

```
from django.db import models
from django.contrib.auth.models import User
from django.utils.timezone import now

class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    email = models.CharField(max_length=100)
    phone_number = models.CharField(max_length=10, null=True, blank=True)

    def__str_(self):
        return str(self.user)

class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.FloatField()
    image = models.ImageField(upload_to="", default="")

    def__str_(self):
        return self.name
```

```
class Feature(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    feature = models.CharField(max_length=1000, null=True, blank=True)

    def__str__(self):
        return str(self.product) + " Feature: " + self.feature

class Review(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    content = models.TextField()
    datetime = models.DateTimeField(default=now)

    def__str__(self):
        return str(self.customer) +  " Review: " + self.content

class Order(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL,
null=True)
    date_ordered = models.DateTimeField(default=now)
    complete = models.BooleanField(default=False)

    def__str_(self):
        return str(self.id)

    @property
```

```
    def get_cart_total(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.get_total for item in orderitems])
        return total

    @property
    def get_cart_items(self):
        orderitems = self.orderitem_set.all()
        total = sum([item.quantity for item in orderitems])
        return total

class OrderItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.SET_NULL,
null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    quantity = models.IntegerField(default=0)
    date_added = models.DateTimeField(default=now)

    def__str__(self):
        return str(self.order)

    @property
    def get_total(self):
        total = self.product.price * self.quantity
        return total
```

```
class UpdateOrder(models.Model):
    order_id = models.ForeignKey(Order, on_delete=models.CASCADE)
    desc = models.CharField(max_length=500)
    date = models.DateField(default=now)

    def__str__(self):
        return str(self.order_id)

class CheckoutDetail(models.Model):
    customer = models.ForeignKey(Customer, on_delete=models.SET_NULL,
null=True)
    order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
    phone_number = models.CharField(max_length=10, blank=True, null=True)
    total_amount = models.CharField(max_length=10, blank=True,null=True)
    address = models.CharField(max_length=300)
    city = models.CharField(max_length=100)
    state = models.CharField(max_length=100)
    zipcode = models.CharField(max_length=100)
    payment = models.CharField(max_length=100, blank=True)
    date_added = models.DateTimeField(default=now)

    def__str_(self):
        return self.address

class Contact(models.Model):
    name = models.CharField(max_length=100)
```

```
    email = models.CharField(max_length=50)
    phone = models.CharField(max_length=10)
    desc = models.CharField(max_length=1000)

    def__str_(self):
        return self.name
```

VIEWS.PY:

```
from django.http.response import HttpResponse
from django.shortcuts import render, redirect
from .models import *
from django.http import JsonResponse
import json
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from . inherit import cartData

def index(request):
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']

    products = Product.objects.all()
    return render(request, "index.html", {'products':products, 'cartItems':cartItems})
```

```
def cart(request):
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']
    try:
        cart = json.loads(request.COOKIES['cart'])
    except:
        cart = {}
    print('Cart:', cart)

    for i in cart:
        try:
            cartItems += cart[i]["quantity"]

            product = Product.objects.get(id=i)
            total = (product.price * cart[i]["quantity"])

            order["get_cart_total"] += total
            order["get_cart_items"] += cart[i]["quantity"]

            item = {
                'product':{
                    'id':product.id,
```

```
                'name':product.name,
                'price':product.price,
                'image':product.image,
            },
            'quantity':cart[i]["quantity"],
            'get_total':total
        }
        items.append(item)
    except:
        pass
  return render(request, "cart.html", {'items':items, 'order':order,
'cartItems':cartItems})


def  checkout(request):
  data = cartData(request)
  items = data['items']
  order = data['order']
  cartItems = data['cartItems']
  total = order.get_cart_total
  if request.method == "POST":
    address = request.POST['address']
    city = request.POST['city']
    state = request.POST['state']
    zipcode = request.POST['zipcode']
    phone_number = request.POST['phone_number']
    payment = request.POST['payment']
```

```
        shipping_adress = CheckoutDetail.objects.create(address=address, city=city,
phone_number=phone_number, state=state, zipcode=zipcode,
customer=request.user.customer, total_amount=total, order=order,
payment=payment)

        shipping_adress.save()

        if total == order.get_cart_total:

            order.complete = True

        order.save()

        id = order.id

        alert = True

        return render(request, "checkout.html", {'alert':alert, 'id':id})

    return render(request, "checkout.html", {'items':items, 'order':order,
'cartItems':cartItems})


def updateItem(request):

    data = json.loads(request.body)

    productID = data['productID']

    action = data['action']


    print('Action:', action)

    print('productID:', productID)


    customer = request.user.customer

    product = Product.objects.get(id=productID)

    order, created = Order.objects.get_or_create(customer=customer,
complete=False)

    orderItem, created = OrderItem.objects.get_or_create(order=order,
product=product)
```

```
    update_order, created = UpdateOrder.objects.get_or_create(order_id=order,
desc="Your Order is Successfully Placed.")


    if action == 'add':
        orderItem.quantity = (orderItem.quantity + 1)
    elif action == 'remove':
        orderItem.quantity = (orderItem.quantity - 1)


    orderItem.save()
    update_order.save()


    if orderItem.quantity <= 0:
        orderItem.delete()
    return JsonResponse('Item was added', safe=False)


def product_view(request, myid):
    customer = request.user.customer
    product = Product.objects.filter(id=myid).first()
    feature = Feature.objects.filter(product=product)
    reviews = Review.objects.filter(product=product)
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']


    if request.method=="POST":
```

```
        content = request.POST['content']
        review = Review(customer=customer, content=content, product=product)
        review.save()
        return redirect(f"/product_view/{product.id}")
    return render(request, "product_view.html", {'product':product,
'cartItems':cartItems, 'feature':feature, 'reviews':reviews})


def search(request):
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']
    if request.method == "POST":
        search = request.POST['search']
        products = Product.objects.filter(name__contains=search)
        return render(request, "search.html", {'search':search, 'products':products,
'cartItems':cartItems})
    else:
        return render(request, "search.html")



def change_password(request):
    if not request.user.is_authenticated:
        return redirect('/login')
    data = cartData(request)
    items = data['items']
```

```
    order = data['order']
    cartItems = data['cartItems']
    if request.method == "POST":
        current_password = request.POST['current_password']
        new_password = request.POST['new_password']
        try:
            u = User.objects.get(id=request.user.id)
            if u.check_password(current_password):
                u.set_password(new_password)
                u.save()
                alert = True
                return render(request, "change_password.html", {'alert':alert})
            else:
                currpasswrong = True
                return render(request, "change_password.html",
{'currpasswrong':currpasswrong})
        except:
            pass
    return render(request, "change_password.html", {'cartItems':cartItems})


def contact(request):
    if request.method=="POST":
        name = request.POST['name']
        email = request.POST['email']
        phone = request.POST['phone']
        desc = request.POST['desc']
```

```
        contact = Contact(name=name, email=email, phone=phone, desc=desc)
        contact.save()
        alert = True
        return render(request, 'contact.html', {'alert':alert})
    return render(request, "contact.html")

def loggedin_contact(request):
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']
    if request.method=="POST":
        name =  request.user
        email = request.user.email
        phone = request.user.customer.phone_number
        desc = request.POST['desc']
        contact = Contact(name=name, email=email, phone=phone, desc=desc)
        contact.save()
        alert = True
        return render(request, 'loggedin_contact.html', {'alert':alert})
    return render(request, "loggedin_contact.html", {'cartItems':cartItems})

def tracker(request):
    if not request.user.is_authenticated:
        return redirect('/login')
```

```
    data = cartData(request)
    items = data['items']
    order = data['order']
    cartItems = data['cartItems']
    if request.method == "POST":
        order_id = request.POST['order_id']
        order = Order.objects.filter(id=order_id).first()
        order_items = OrderItem.objects.filter(order=order)
        update_order = UpdateOrder.objects.filter(order_id=order_id)
        print(update_order)
        return render(request, "tracker.html", {'order_items':order_items,
'update_order':update_order})
    return render(request, "tracker.html", {'cartItems':cartItems})



def register(request):
    if request.user.is_authenticated:
        return redirect("/")
    else:
        if request.method=="POST":
            username = request.POST['username']
            full_name=request.POST['full_name']
            password1 = request.POST['password1']
            password2 = request.POST['password2']
            phone_number = request.POST['phone_number']
            email = request.POST['email']
```

```
            if password1 != password2:
                alert = True
                return render(request, "register.html", {'alert':alert})
            user = User.objects.create_user(username=username,
password=password1, email=email)
            customers = Customer.objects.create(user=user, name=full_name,
phone_number=phone_number, email=email)
            user.save()
            customers.save()
            alert = True
            return render(request, "success.html", {'alert':alert})
    return render(request, "register.html")

def Login(request):
    if request.user.is_authenticated:
        return redirect("/")
    else:
        if request.method == "POST":
            username = request.POST['username']
            password = request.POST['password']
            user = authenticate(username=username, password=password)

            if user is not None:
                login(request, user)
                return redirect("/")
```

```
            else:
                alert = True
                return render(request, "login.html", {"alert":alert})
    return render(request, "login.html")

def Logout(request):
    logout(request)
    alert = True
    return render(request, "index.html", {'alert':alert})
```

BASE.HTML:

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLA
SjC" crossorigin="anonymous">
        <script src="https://kit.fontawesome.com/90ccb65d9b.js"
crossorigin="anonymous"></script>
        <link rel="preconnect" href="https://fonts.googleapis.com">
```

```
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
        <link
href="https://fonts.googleapis.com/css2?family=Red+Hat+Display:wght@500&disp
lay=swap" rel="stylesheet">


    <title>{% block title %} {% endblock %}</title>
</head>


<body>
    <nav class="navbar navbar-expand-lg navbar-dark sticky-top">
        <div class="container-fluid">
            <a class="navbar-brand" href="/" ><img id="logonav"
src="/static/logo2.png"></a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <div class="container w-75">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li>
                        <a class="nav_links" href="/">Home</a>
                    </li>
                    {% if request.user.is_authenticated %}
                    <li>
```

```
                <a class="nav_links" href="/cart/"><span class="fa fa-shopping-
cart"></span>({{cartItems}})</a>
            </li>
            <li>
                <a class="nav_links" href="/tracker/">Track Order</a>
            </li>
            <li>
                <a class="nav_links" href="/change_password/">Change
Password</a>
            </li>
            <li">
                <a class="nav_links" href="/loggedin_contact/">Contact Us</a>
            </li>
            {% endif %}
            {% if not request.user.is_authenticated %}
            <li>
                <a class="nav_links" href="/register/">Register</a>
            </li>
            <li>
                <a class="nav_links" href="/login/">Login</a>
            </li>
            <li>
                <a class="nav_links" href="/contact/">Contact Us</a>
            </li>
            {% else %}
                <a class="nav_links3" href="/logout/">Logout</a>
```

```
                <a class="nav_links2" href="#">Welcome {{request.user}}</a>


            {% endif %}

        </ul>

        </div>

        <form class="d-flex" method="POST" action="/search/"> {% csrf_token
%}

            <input class="form-control me-2" type="search" name="search"
placeholder="Search Product" aria-label="Search" style="border-radius: 40px;">

            <button href="/search/" class="btn-btn-secondary"
type="submit">Search</button>

        </form>

      </div>

    </div>

  </nav>


  {% block body %}

  {% endblock %}

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"

    integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVX
M"

    crossorigin="anonymous"></script>

  <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"

    integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n
"

    crossorigin="anonymous"></script>
```

```
<script>
    var user = '{{request.user}}'

    function getToken(name) {
        var cookieValue = null;
        if (document.cookie && document.cookie !== '') {
            var cookies = document.cookie.split(';');
            for (var i = 0; i < cookies.length; i++) {
                var cookie = cookies[i].trim();
                // Does this cookie string begin with the name we want?
                if (cookie.substring(0, name.length + 1) === (name + '=')) {
                    cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                    break;
                }
            }
        }
        return cookieValue;
    }
    var csrftoken = getToken('csrftoken');

    function getCookie(name) {
        var cookieArr = document.cookie.split(";");

        for (var i = 0; i < cookieArr.length; i++) {
```

```
        var cookiePair = cookieArr[i].split("=");

        if (name == cookiePair[0].trim()) {
            // Decode the cookie value and return
            return decodeURIComponent(cookiePair[1]);
        }
    }

    // Return null if not found
    return null;
}
var cart = JSON.parse(getCookie('cart'))

if (cart == undefined) {
    cart = {}
    console.log('Cart Created!', cart)
    document.cookie = 'cart=' + JSON.stringify(cart) + ";domain=;path=/"
}
console.log('Cart:', cart)

var updateBtns = document.getElementsByClassName('update-cart')

for (var i = 0; i < updateBtns.length; i++) {
    updateBtns[i].addEventListener('click', function () {
        var productID = this.dataset.product
```

```
        var action = this.dataset.action
        console.log('productId:', productID, 'action:', action)
        console.log('USER:', user)
        if (user == 'AnonymousUser') {
            addCookieItem(productID, action)
        } else {
            updateUserOrder(productID, action)
        }
    })
}

function addCookieItem(productID, action) {
    console.log('Not logged in')
    if (action == 'add') {
        if (cart[productID] == undefined) {
            cart[productID] = { 'quantity': 1 }

        } else {
            cart[productID]['quantity'] += 1
        }
    }

    if (action == 'remove') {
        cart[productID]['quantity'] -= 1
```

```
        if (cart[productID]['quantity'] <= 0) {
            console.log('Item should be deleted')
            delete cart[productID];
        }
    }
    console.log('Cart:', cart)
    document.cookie ='cart=' + JSON.stringify(cart) + ";domain=;path=/"
    location.reload()
}

function updateUserOrder(productID, action) {
    console.log('User is logged in, sending data...')
    var url = '/update_item/'
    fetch(url, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': csrftoken,
        },
        body: JSON.stringify({ 'productID': productID, 'action': action })
    })
        .then((response) => {
            return response.json()
        })
```

```
            .then((data) => {
                console.log('data:', data)
                location.reload()
            })
        }

    </script>
    {% block js %}
    {% endblock %}

</body>

</html>
```

INDEX.HTML:

```
{% extends 'base.html' %}
{% load static %}
{% block title %} JustBuy - Online Shopping {% endblock %}
{% block css %}
{% endblock %}
{% block body %}


<div class="container mb-5 mt-3">
  <div class="row">
```

```
    {% for product in products %}
    <div class="col-md-3 my-3">
      <div class="card mt-3">
        <div class="product-1 align-items-center p-2 text-center">
          <img src="/media/{{product.image}}" alt="" class="rounded" width="200" height="200">
          <h5 class="name-h5">{{product.name}}</h5>

          <div class="cost mt-3 text-dark">
            <h4 class="name-h4"> ₹{{product.price}}</h4>
          </div>

        </div>

        <div class="p-3 square text-center text-white mt-3 cursor">
          {% if request.user.is_authenticated %}
          <button data-product="{{product.id}}" data-action="add" class="btn-add add-btn update-cart">Add To Cart</button>
          <a href="/product_view/{{product.id}}/" class="btn-view">View</a>
          {% else %}
          <a href="login/" class="btn-login">Login to add the item</a>
          {% endif %}
        </div>
      </div>
    </div>
    {% endfor %}
  </div>
</div>
```

```
<footer class="footer-distributed">
  <span style="font-family:'Red Hat Display', sans-serif">

    <div class="footer-left">
  <img id="logonav2" src="/static/logo2.png">

        <p id="footercompanyname"><br>© 2022 E-Commerce Pvt. Ltd.</p>
    </div>

    <div class="footer-center">
        <div>
            <i class="fa fa-map-marker"></i>
              <p><span>785 - Rupa Square,
                Shop No: X - 96, Sector - 1</span>
                T-Nagar, Chennai - 632145</p>
        </div>

        <div>
            <i class="fa fa-phone"></i>
            <p>+91 45-86321459</p>
        </div>
        <div>
            <i class="fa fa-envelope"></i>
            <p><a                                    href="mailto:support@justbuy.com"
style="color:#e668ff;">support@justbuy.com</a></p>
        </div>
```

```
</div>
<div class="footer-right">
    <p class="footer-company-about">
        <span>About the company</span>
        JustBuy is an E-commerce platform.</p>
        <ul class="icon-container">
          <li class="icon-list">
            <a class="icon-link" href="#">
              <i class="fab fa-facebook"></i>
            </a>
          </li>
          <li class="icon-list">
            <a class="icon-link" href="#">
              <i class="fab fa-twitter"></i>
            </a>
            </li>
          <li class="icon-list">
            <a class="icon-link" href="#">
              <i class="fab fa-instagram"></i>
            </a>

          </li>
          <li class="icon-list">
            <a class="icon-link" href="#">
              <i class="fab fa-youtube"></i>
            </a>
          </li>
```

```
            </ul>
          </span>
    </div>

</footer>

{% endblock %}
{% block js %}
<script>

{% if alert %}
  alert("Logout Successful.")
  window.location.href = '/'
{% endif %}
</script>
{% endblock %}
```