

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI – 590018.



**INTERNSHIP REPORT ON
“E-COMMERCE WEB DEVELOPMENT USING DJANGO”**

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENT OF

**BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING
SUBMITTED BY**

SYAMILI S N (1AH18CS101)

UNDER THE GUIDANCE OF

Ms. LAKSHMI PRIYA P

Assistant Professor, CSE Department, ACSCE.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACS COLLEGE OF ENGINEERING

KAMBIPURA, MYSORE ROAD, BENGALURU – 560074.

2021-2022

ACS COLLEGE OF ENGINEERING
KAMBIPURA, MYSORE ROAD, BENGALURU – 560074.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CERTIFICATE

This is to certify that the internship entitled **“E-COMMERCE WEB DEVELOPMENT USING DJANGO”** carried out by, **SYAMILI S N (1AH18CS101)**, bonafide student of **ACS COLLEGE OF ENGINEERING, BENGALURU**, in partial fulfilment for the award of degree in **BACHELOR OF COMPUTER SCIENCE AND ENGINEERING, Visvesvaraya Technological University, Belagavi** during the year **2021-2022**. It is certified that all corrections/ suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The internship report has been approved as it satisfies the academic requirements in respect of internship prescribed for the said degree.

Guide Signature:

Ms. Lakshmi Priya P, Assistant
professor, CSE, ACSCE,
Bengaluru.

Co-ordinator Signature:

Ms. Ganga B M, Assistant
professor, CSE, ACSCE,
Bengaluru.

HOD Signature:

Dr. V Mareeswari, Associate
professor, CSE, ACSCE,
Bengaluru.

Name of the Examiners

1. _____

2. _____

Signature with Date

INTERNSHIP CERTIFICATE



TO WHOMSOEVER IT MAY CONCERN

Date: 15th Dec, 2021

This is to certify that Ms. Syamili S N, (Aadhar No. XXXX-XXXX-6353), B.E CSE, student of ACS College of Engineering, has successfully completed her internship with Prabkrishnaatechs Solutions Pvt. Ltd. during the period (15th Oct 2021 to 15th Dec 2021).

During the period, she handled mention assigned responsibilities she had worked on Web Development.

During the course of internship, Ms. Syamili S N has shown great amount of responsibility, sincerity and a genuine willingness to learn and zeal to take on new assignments & challenges. In particular, her coordination skills and communication skills are par excellence and her attention to details is impressive.

We wish her all the very best for her future.

With regards,



Authorized Signatory
Prabkrishnaatechs Solutions Pvt. Ltd.
Rajkumar Shaw
(Director)

www.krishnatechs.com / www.krishnatechs.co.in

info@krishnatechs.com

(+91) 620 4970 866

C/o Krishna Sah, H No. 180, Tetulia More, Barwa East,
Nisra1, Dhanbad, (JH) 828205

support@krishnatechs.com

(+91) 790 3913 059

PRABKRISHNAATECHS SOLUTIONS PRIVATE LIMITED

CIN: U72900JH2021PTC016387 || GSTIN: 20AALCP9386L1ZS

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the names of the people who made it possible, whose constant guidance and encouragement crowned my effort with success.

I sincerely thank our honourable chairman, **Dr. A C Shanmugam**, for providing me with the best facilities and his encouragement which helped us in the completion of the project.

I am grateful to our institution, **ACS College of Engineering**, for its ideals and inspirations for having provided us with the facilities, which made this project a success.

I earnestly thank **Dr. M S Murali, Principal, ACSCE**, for facilitating academic excellence in the college that helped us in completing this project.

I wish to extend my profound thanks to **Dr. V Mareeswari**, Associate professor and Head of the department, Computer Science and Engineering, for giving me the consent to carry out this project.

I would like to express my sincere thanks to the project guide, **Ms. Lakshmi Priya P**, Assistant professor, Department of CSE, ACSCE, **Mrs. Ganga B M**, internship coordinator, Assistant professor, Department of CSE, ACSCE for their able guidance and valuable advice at every stage of my project, which helped me in the successful completion of the Internship seminar and project demonstration.

I convey my thanks to all those who selflessly lent their time and effort for providing me with help and creating convenient opportunities. Last but not least I would like to thank all the staff members and the institute, in general, for extending a helping hand and making this Internship Seminar possible.

SYAMILI S N (1AH18CS101)

ABSTRACT

The scope of this internship includes research and implementation of an E-commerce website using the python-based web framework called Django. The functionality of the website includes the creation of user accounts, viewing and adding products to the shopping e-cart. The website is not only limited to shopping but also includes a vendors feature where a customer can create his very own vendors account using which he can add his own products, which he is interested in selling on the FastEcommerce website. There is also the administrator site where the admin can monitor the products which are added to the website and is solely responsible for the modification and updating the database and migrations. Finally, after placing the order the consumer is able to checkout from the e-cart and make his payments by entering valid credit card information in a secure portal.

LIST OF CONTENTS

NO.	CHAPTER	PAGE
1.	INTRODUCTION	1
1.1	ABOUT THE COMPANY	1
1.2	INTERN'S ROLE AND RESPONSIBILITIES	1
1.3	SCOPE OF THE INTERNSHIP	2
2.	TECHNOLOGY AND METHODOLOGY	3
2.1	TECHNOLOGY USED: DJANGO PYTHON WEB FRAMEWORK	3
2.2	ADVANTAGES OF USING DJANGO	3
2.3	LIMITATIONS OF USING DJANGO	3
2.4	METHODOLOGY	4
2.4.1	Creating a Virtual Environment	4
2.4.2	Installing Django	4
2.4.3	Starting the Django Project	4
2.4.4	Creating the first web app	4
2.4.5	Creating the superuser/ admin	4
2.4.6	Run Project	4
3.	SYSTEM REQUIREMENTS AND SPECIFICATIONS	5
3.1	FUNCTIONAL REQUIREMENTS	5
3.2	NON-FUNCTIONAL REQUIREMENTS	5
3.2.1	Hardware Specifications	5
3.2.2	Software Specifications	5
4.	SYSTEM DESIGN	6
4.1	ER-DIAGRAM	6
4.2	DATABASE SCHEMA MODEL	7
4.3	CLASS DIAGRAM	8
4.4	DATABASE DESIGN	9
5	IMPLEMENTATION	12
5.1	MODULE IMPLEMENTATION	12
5.2	FUNCTIONS AND VIEWS	12
6.	RESULTS	22

LIST OF FIGURES

NO.	FIGURE	PAGE
1.3	Live website hosted by PTC: 1organic.in	2
4.1	ER diagram of FastEcommerce Website Database	6
4.2	UML diagram of FastEcommerce Website Database	8
4.3.1	Django Administration Site	9
4.3.2	Django Administration Site: Vendors Table	9
4.3.3	Django Administration Site Orders: Orders Table	10
4.3.4	Django Administration Site Orders: Order Items Table	10
4.3.5	Django Administration Site Product: Products Table	11
4.3.6	Django Administration Site Product: Categories Table	11
6.1	FastEcommerce Home Page	22
6.2	FastEcommerce Electronics Page	22
6.3	FastEcommerce Shirt Page	23
6.4	FastEcommerce Jewellery Page	23
6.5	FastEcommerce Food Page	24
6.6	FastEcommerce Search Page	24
6.7	FastEcommerce Cart Page Consumer Details	25
6.7	FastEcommerce Footer Details	25

ACRONYM TABLE

ABBREVIATION

PTC

IT

HTML

Email

URL

ORM

API

CRM

SQL

RAM

UI

CMS

MVC

FULL FORM

Prabkrishna Techs Company

Information technology

Hypertext Markup Language

Electronic Mail

Uniform Resource Relocator

Object-Relational Mapper

Application Programming Interface

Customer-Relationship Management

Structured Query Language

Random Access Memory

User-Interface

Content Management System

Model View Controller Pattern

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE COMPANY

Prabkrishna Techs Company (PTC) is a full-cycle IT solution provider specialising in bespoke site design, web development, mobile application development, SEO, hosting, and support. They provide progressive end-to-end solutions by combining substantial business domain experience, technological skills through an understanding of the current market trends, and a quality-driven delivery strategy.

They have a team of competent, professional product masters and design and development gurus to create high-quality, engaging, and constructive digital goods. Its most recent project is an all-encompassing E-commerce web development site that sells the most up-to-date stylish apparel, food, technology, and accessories at a fair price all over the world.



1.2 INTERN'S ROLE AND RESPONSIBILITIES

1. Working on the implementation of FastEcommerce.co, an online E-commerce website for the group company.
2. Conducting research about Django and its functionalities.
3. Creating workflow and learning about the Django Framework feature.
4. Creating a structured logic and schema for the website database.
5. Brainstorm new ideas when required.
6. Working on cross-linking functions.
7. Working on other IT-related responsibilities.
8. Having ample amount of web development skills and capable of building a functional web development site.

1.3 SCOPE OF THE INTERNSHIP

The scope of the internship includes researching the Django Web Development features as well as understanding the working of the local server to view its features in order to utilize it to enhance the functionality of the company's e-commerce website <https://FastEcommerce.co>. It is a live website which accepts orders internationally for authentic organic foods, facial masks, cruelty-free clothing and jewellery.

There were many problems that had to be faced before the deployment of the website as per the client's requirements. Many APIs are required to be connected in order to finally make the entire site automated along with an assistive chatbot, which is outside of the scope of the role of this internship.

As a first-time intern, I was trained for a period of two months. Graciously and forgivingly, taught how to use Django and various plugins that provided additional support to the site without coding for each feature such as floating cards for each category of items on the website.

As an intern employee, I was given access to the PTC organisation's Customer Relationship Management (CRM), where the day-to-day tasks were posted. I was given the task of posting blogs on their main website <https://basillia.in>. And as I was starting to get familiar with the way the website was built, I was given more complex tasks of setting up different views and functionalities of the FastEcommerce.co website.

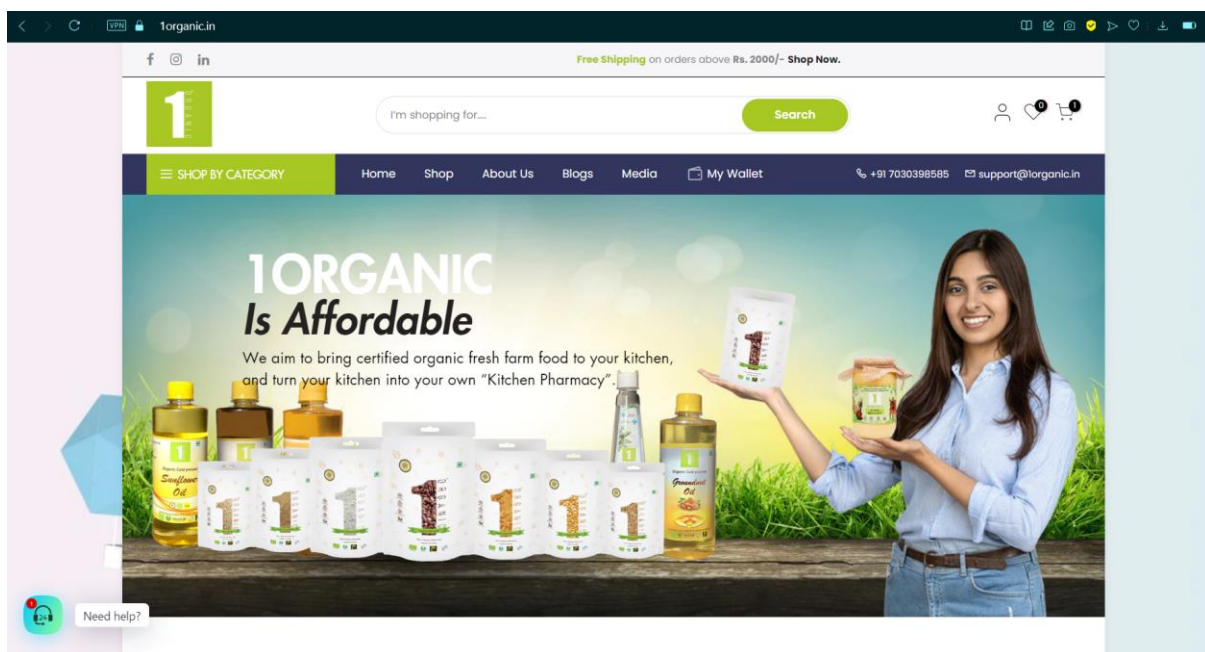


Fig 1.3: Live website hosted by PTC: 1organic.in

CHAPTER 2

TECHNOLOGY AND METHODOLOGY

2.1 TECHNOLOGY USED: DJANGO-PYTHON WEB FRAMEWORK

Django is a python-base, free, open-source web application framework written in Python. A web application framework is a collection of modules that make development easier. They are grouped together which allows us to create applications or websites from an existing source instead of creating them from scratch.

Using the Django framework an individual can create simple yet sophisticated web applications. Which can include advanced functionality like authentication support, management and admin panels, comment boxes, contact forms as well as support for uploading files.

For developing our FastEcommerce.co website we specifically went for Django because of its ease of implementation and time-saving pre-built modules which we were able to implement.

2.2 ADVANTAGES OF USING DJANGO

Originally, while developing a web application, every component is required to be coded manually. But with the genesis of these web frameworks, it is possible to instead, use these pre-built components and just configure them to match the site settings.

Django has a big collection of modules which can be implemented for the development of web applications. For example, the Django module comes with pre-built classes which take care of configuration files.

Other advantages include being easy to comprehend and use by new web developers, it has sufficient clarity and readability. It also has a versatile design which can be integrated with other software through APIs.

The most important advantage is that it is highly secure. It ensures that the developers do not commit any mistakes related to the security of the project, which includes SQL injection, cross-site request forgery, cross-site scripting and clickjacking.

It is highly scalable which ensures that the demand brought by the heaviest of web traffic is satisfied. It has efficient content management as well as scientific computing power.

2.3 LIMITATIONS OF USING DJANGO

There are certain limitations to what Django can do. Some of them include that the user's routing pattern must specify its URL. The framework is too monolithic, that is even for smaller apps we must install the whole Django package which can be tedious.

Every part is based on Django Object-Relational Mapper which enables us to interact with the database, similar to how we interact with SQL. All components need to be implemented error-free in order to see the required result on the browser.

And most important of all, the developer must be knowledgeable in the python language as well as understand how to integrate various parts of the program.

2.4 METHODOLOGY

2.4.1 Creating a Virtual Environment

A virtual environment is a place on the computer system where packages can be installed and isolated from all other Python packages. For the scope of this project, I have used the python module **venv** to create a virtual environment called venv. The virtual environment needs to be activated in order to start the project.

Command to create a virtual environment: `$ python -m venv venv`

Command to activate the virtual environment: `$ venv\Scripts\activate`

2.4.2 Installing Django

The installation of Django can be done using a single pip command: `$ pip install django`

2.4.3 Starting the Django Project

To start the project the following command needs to be typed on the command prompt:

`$ django-admin startproject simple_multivendor_site .`

2.4.4 Creating the first web app

Core is the first web app created which hosts the home and contact page URLs. The command to create the core app is: `$ python manage.py startapp core`

2.4.5 Creating the superuser/ admin

The superuser is the site's administrator who has all the permissions to add and modify the application and it can be created using the simple command:

`$ python manage.py createsuperuser`

And the user is then prompted to enter his name, email, and password.

2.4.6 Run Project

To run the project the following command must be used in the command prompt. And the web application will be hosted on the 8000 port at the URL <http://localhost:8000>.

`$ python manage.py runserver`

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATIONS

3.1 FUNCTIONAL REQUIREMENTS

- Description: The user should be able to log in as a vendor, add products to the site as well as add products to the cart and have his payments made in a secure way.
- Input: The user can provide product images, descriptions and product prices.
- Process: The user data is stored in the SQLite database which updates the total number of products displayed on the website. And once the user adds items to the cart, he can checkout.
- Output: The added products are stored in the database and displayed on the browser.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 Hardware Specifications

Processor	: Intel Core i3-2340UE
Processor Speed	: 1.3 GHz
RAM	: 1 GB
Memory	: 4 GB
GPU	: AMD Radeon R5 M230

3.2.2 Software Specifications

Operating System	: Windows7, 10, 11
Database	: SQLite
Scripting Language	: Python
Code Editor	: Visual Studio Code Editor

CHAPTER 4

SYSTEM DESIGN

4.1 ER-DIAGRAM

The main database consists of three categories Products, Orders and Vendors. When the customer logs into the website and starts browsing the page, many products are presented on-screen. Since the number of products might seem overwhelming, he might start by first choosing the category.

The four main categories included in FastEcommerce.co are food, electronics, clothing and jewellery. If the consumer is interested in food then he might click on that category. Now if he wants to buy brown rice then, he will select brown rice as the product.

Since there is a one-to-many dependency between category and product where one category is linked to many products, the product, its price along with its category is added to the invoice.

Similarly, the consumers and orders share a many-to-many relationship, each customer might have one more orders from the website. Each item has an id and each order which consists of a set of items also has an id, which is associated with each consumer.

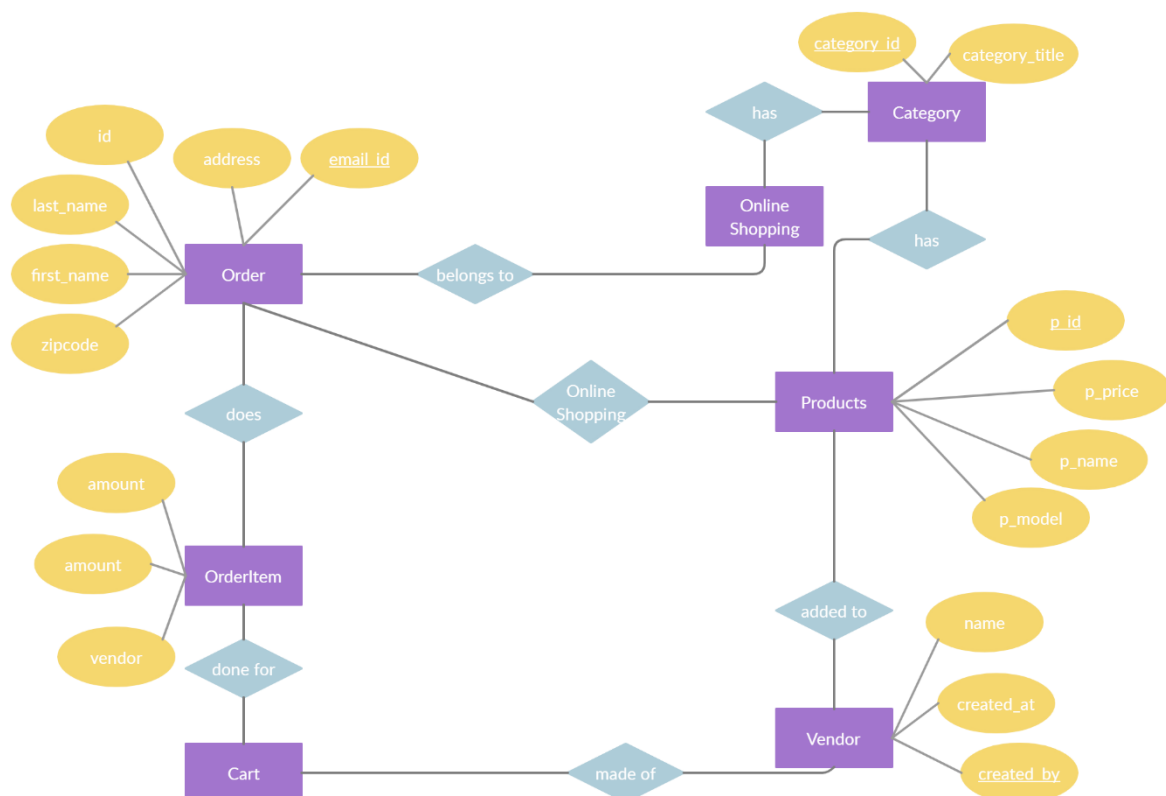


Fig 4.1: ER diagram of FastEcommerce Website Database

4.2 DATABASE SCHEMA MODEL

PRODUCT MODELS

Category

Title	Slug	Ordering
Char field	Slug field	Integer field

Product

Category	Vendor	Title	Slug	Description	Price	Added-date	Image	thumbnail
Foreign Key	Foreign Key	Char field	Slug field	Text field	Decimal field	Datetime field	Image field	Image field

ORDER MODELS

Order

First-name	Last-name	Email	Address	Zipcode	Place	Phone	Created-at	Paid-amount	Vendors
Char field	Char field	Char field	Char field	Char field	Char field	Char field	Datetime field	Decimal field	Many-to-many field

OrderItem

Order	Product	Vendor	Vendor_paid	Price	Quantity
Foreign key	Foreign key	Foreign key	Boolean field	Decimal field	Integer field

VENDOR MODELS

Vendor

Name	Created_at	Created_by
Char field	Datetime field	One-to-one field

4.3 CLASS DIAGRAM

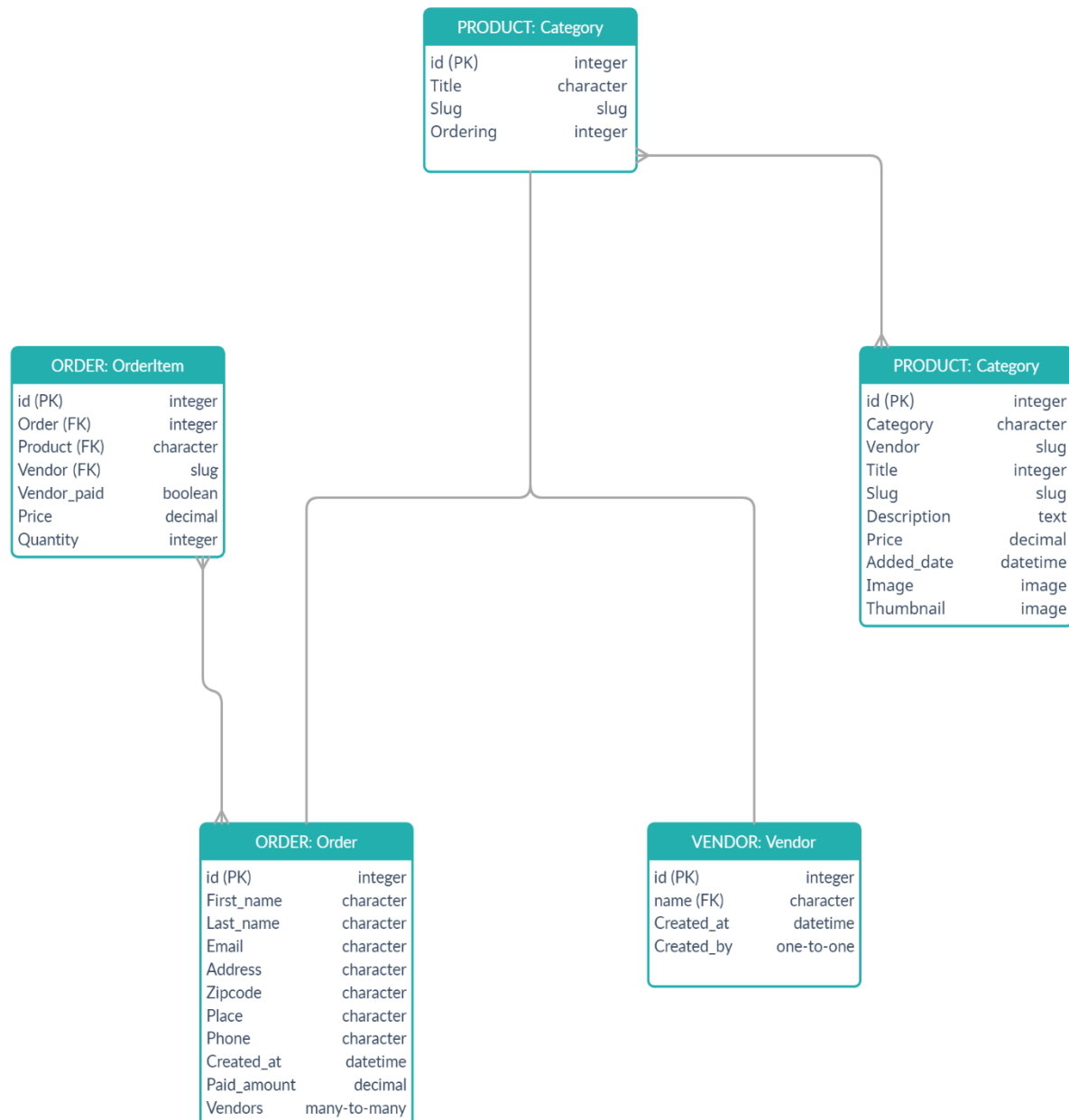


Fig 4.2: UML diagram of FastEcommerce Website Database

4.4 DATABASE DESIGN

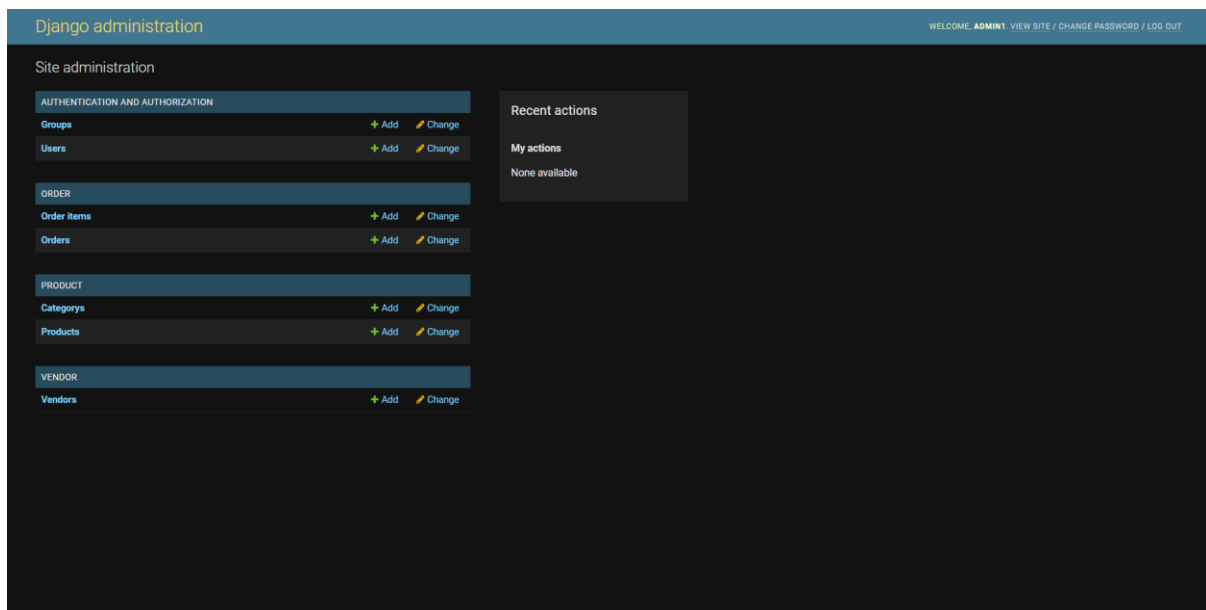


Fig 4.4.1 Django Administration Site

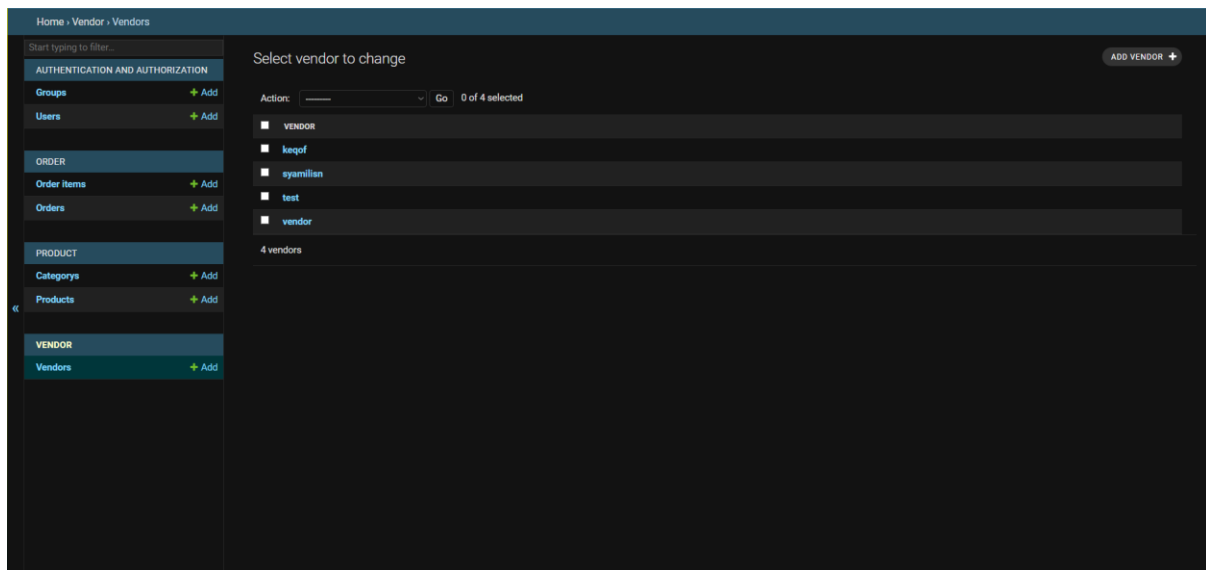


Fig 4.4.2: Django Administration Site: Vendors Table

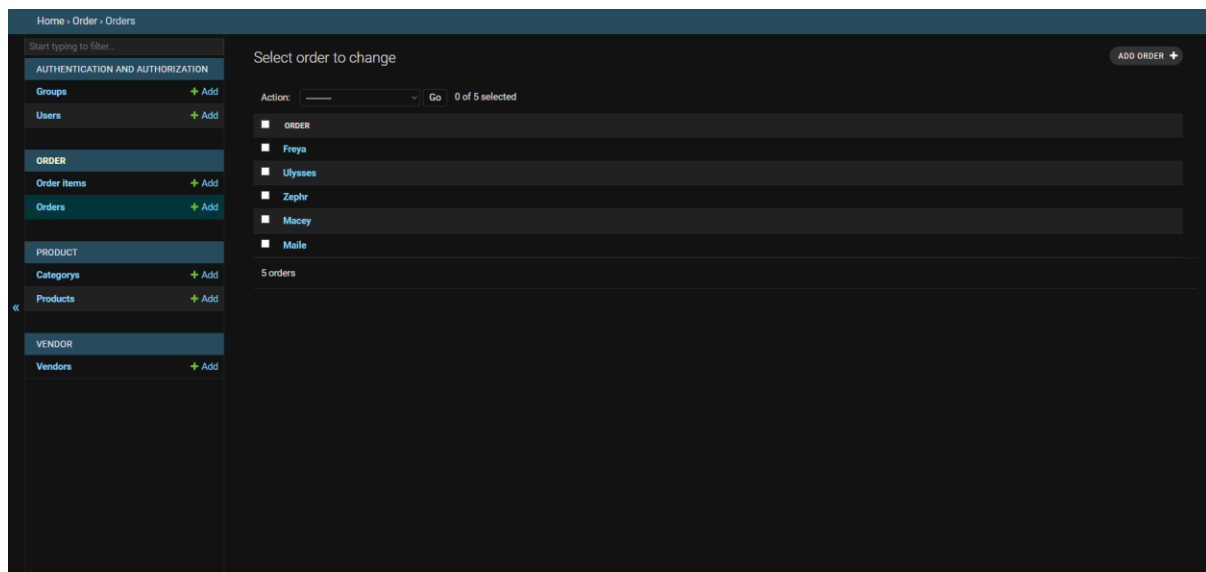


Fig 4.4.3: Django Administration Site Orders: Orders Table

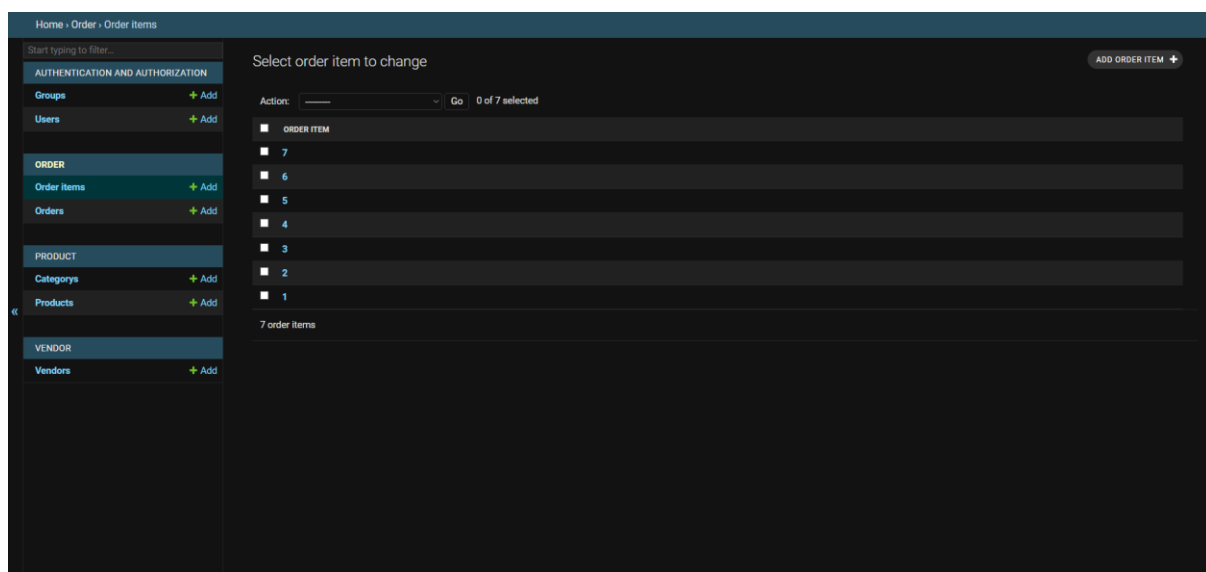


Fig 4.4.4: Django Administration Site Orders: Order Items Table

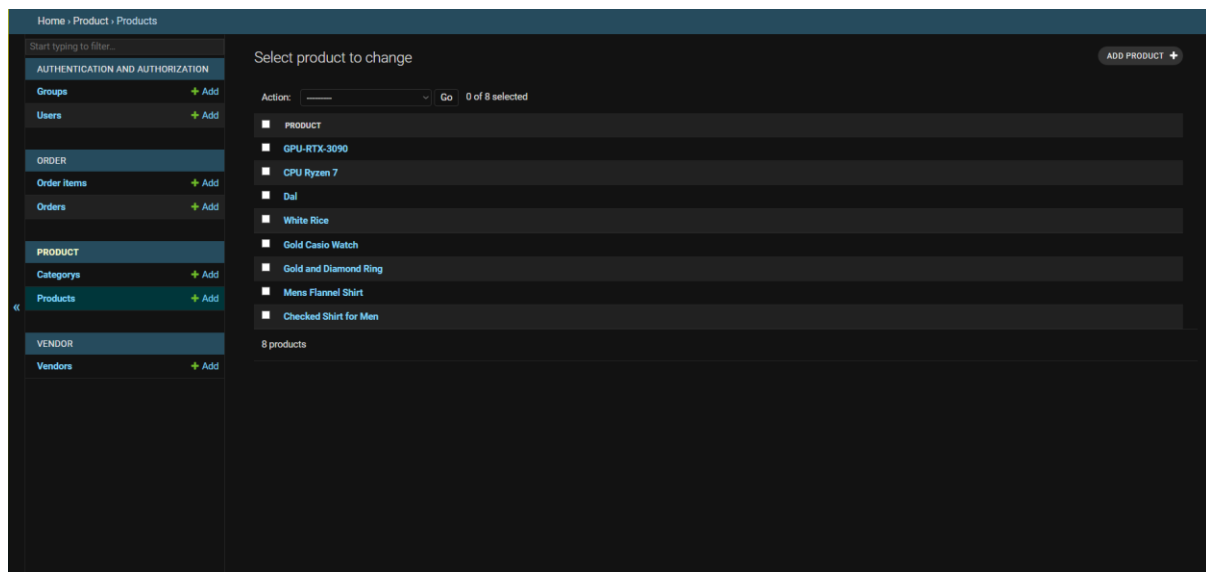


Fig 4.4.5: Django Administration Site Product: Products Table

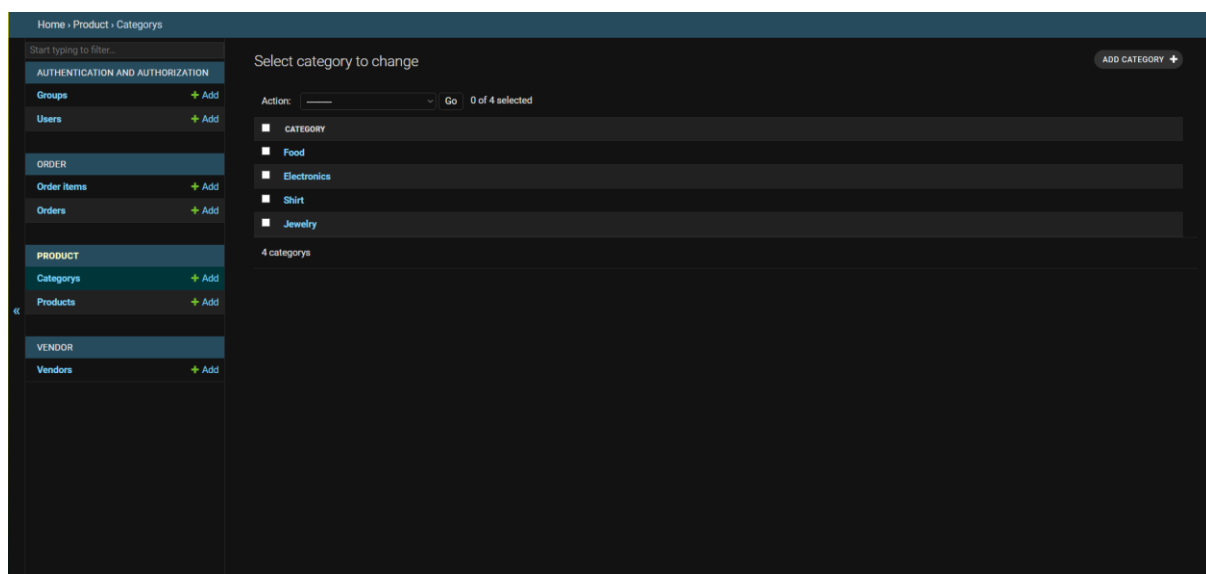


Fig 4.4.6: Django Administration Site Product: Categories Table

CHAPTER 5

IMPLEMENTATION

This section deals with the implementation of the system designed in the previous section. Implementation is the process of converting the design to code. The entities identified from the design are to be implemented considering the association between them and how they communicate with each other.

5.1 MODULE IMPLEMENTATION

5.1.1 Product

The Product Module consists of the Product class which has two tables. One is the Category table which is the list of all products set up for sale on the website. The second one is the Product table which is a combination of all products alongside their category.

It consists of product id, category, vendor, title, product description, date added and price along with image/ thumbnail.

5.1.2 Order

The Order Module consists of the Order class which has two tables. One is the Order table which is a list of all orders booked by new customers on the website. The second is the Order item table which is a combination of orders and the products selected per order.

It consists of the consumer's first name, last name, email, address, zip code, place, phone, created at, the amount paid and vendor.

5.1.3 Vendor

The Vendor Module consists of the Vendor class which has one table. The table name is Vendor and it contains information such as vendor name, date of creation as well as vendor details.

5.2 FUNCTIONS AND VIEWS

5.2.1 Product\models.py

```
class Category(models.Model):  
  
    title = models.CharField(max_length=50)  
  
    slug = models.SlugField(max_length=55)  
  
    ordering = models.IntegerField(default=0)
```

```
class Meta:
```

```
    ordering = ['ordering']
```

```
    def __str__(self):
```

```
        return self.title
```

```
class Product(models.Model):
```

```
    category = models.ForeignKey(Category, related_name='products',
    on_delete=models.CASCADE)
```

```
    vendor = models.ForeignKey(Vendor, related_name="products",
    on_delete=models.CASCADE)
```

```
    title = models.CharField(max_length=50)
```

```
    slug = models.SlugField(max_length=55)
```

```
    description = models.TextField(blank=True, null=True)
```

```
    price = models.DecimalField(max_digits=6, decimal_places=2)
```

```
    added_date = models.DateTimeField(auto_now_add=True)
```

```
    image = models.ImageField(upload_to='uploads/', blank=True, null=True)
```

```
    thumbnail = models.ImageField(upload_to='uploads/', blank=True, null=True) # Change
    uploads to thumbnails
```

```
class Meta:
```

```
    ordering = ['-added_date']
```

```
    def __str__(self):
```

```
        return self.title
```

```
def get_thumbnail(self):  
    if self.thumbnail:  
        return self.thumbnail.url  
    else:  
        if self.image:  
            self.thumbnail = self.make_thumbnail(self.image)  
            self.save()  
            return self.thumbnail.url  
  
        else:  
            # Default Image  
            return 'https://via.placeholder.com/240x180.jpg'  
  
# Generating Thumbnail - Thumbnail is created when get_thumbnail is called  
def make_thumbnail(self, image, size=(300, 200)):  
    img = Image.open(image)  
    img.convert('RGB')  
    img.thumbnail(size)  
  
    thumb_io = BytesIO()  
    img.save(thumb_io, 'JPEG', quality=85)  
  
    thumbnail = File(thumb_io, name=image.name)  
    return thumbnail
```

5.2.2 Order\models.py

```
class Order(models.Model):
```

```
    first_name = models.CharField(max_length=100)
```

```
    last_name = models.CharField(max_length=100)
```

```
    email = models.CharField(max_length=100)
```

```
    address = models.CharField(max_length=100)
```

```
    zipcode = models.CharField(max_length=100)
```

```
    place = models.CharField(max_length=100)
```

```
    phone = models.CharField(max_length=100)
```

```
    created_at = models.DateTimeField(auto_now_add=True)
```

```
    paid_amount = models.DecimalField(max_digits=8, decimal_places=2)
```

```
    vendors = models.ManyToManyField(Vendor, related_name="orders")
```

```
class Meta:
```

```
    ordering = ['-created_at']
```

```
def __str__(self):
```

```
    return self.first_name
```

```
class OrderItem(models.Model):
```

```
    order = models.ForeignKey(Order, related_name="items", on_delete=models.CASCADE)
```

```
    product = models.ForeignKey(Product, related_name="items",  
on_delete=models.CASCADE)
```

```
    vendor = models.ForeignKey(Vendor, related_name="items",  
on_delete=models.CASCADE)
```

```
    vendor_paid = models.BooleanField(default=False)
```

```
price = models.DecimalField(max_digits=8, decimal_places=2)
```

```
quantity = models.IntegerField(default=1)
```

```
def __str__(self):
```

```
    return str(self.id)
```

```
def get_total_price(self):
```

```
    return self.price * self.quantity
```

5.2.3 Vendor\models.py

```
class Vendor(models.Model):
```

```
    name = models.CharField(max_length=255)
```

```
    created_at = models.DateTimeField(auto_now_add=True)
```

```
    created_by = models.OneToOneField(User, related_name='vendor',  
on_delete=models.CASCADE)
```

```
class Meta:
```

```
    ordering = ['name']
```

```
def __str__(self):
```

```
    return self.name
```

```
def get_balance(self):
```

```
    items = self.items.filter(vendor_paid=False, order__vendors__in=[self.id])
```

```
    return sum((item.product.price * item.quantity) for item in items)
```

```
def get_paid_amount(self):
```

```
    items = self.items.filter(vendor_paid=True, order__vendors__in=[self.id])
```

```
    return sum((item.product.price * item.quantity) for item in items)
```


5.2.4 Product\views.py

```
def product(request, category_slug, product_slug):  
    # Create instance of Cart class  
    cart = Cart(request)  
  
    product = get_object_or_404(Product, category__slug=category_slug, slug=product_slug)  
  
    # Check whether the AddToCart button is clicked or not  
    if request.method == 'POST':  
        form = AddToCartForm(request.POST)  
  
        if form.is_valid():  
            quantity = form.cleaned_data['quantity']  
            cart.add(product_id=product.id, quantity=quantity, update_quantity=False)  
  
            messages.success(request, "The product was added to the cart.")  
  
            return redirect('product:product', category_slug=category_slug,  
product_slug=product_slug)  
  
        else:  
            form = AddToCartForm()  
  
    similar_products = list(product.category.products.exclude(id=product.id))  
  
    # If more than 4 similar products, then get 4 random products  
    if len(similar_products) >= 4:  
        similar_products = random.sample(similar_products, 4)  
  
    context = {  
        'product': product,
```

```
'similar_products': similar_products,  
'form': form,  
}
```

```
return render(request, 'product/product.html', context)
```

```
def category(request, category_slug):  
    category = get_object_or_404(Category, slug=category_slug)  
    return render(request, 'product/category.html', {'category': category})
```

```
def search(request):  
    query = request.GET.get('query', '') # second is default parameter which is empty  
    products = Product.objects.filter(Q(title__icontains=query) |  
    Q(description__icontains=query))  
  
    return render(request, 'product/search.html', {'products':products, 'query': query})
```

5.2.5 Vendor\views.py

```
def vendors(request):  
    return render(request, 'vendor/vendors.html')
```

```
def become_vendor(request):  
    if request.method == 'POST':  
        form = UserCreationForm(request.POST)  
  
        if form.is_valid():  
            user = form.save()  
            login(request, user)  
            vendor = Vendor.objects.create(name=user.username, created_by=user)
```

```
        return redirect('core:home')
    else:
        form = UserCreationForm()

    return render(request, 'vendor/become_vendor.html', {'form': form})

@login_required
def vendor_admin(request):
    vendor = request.user.vendor
    products = vendor.products.all()
    orders = vendor.orders.all()
    for order in orders:
        order.vendor_amount = 0
        order.vendor_paid_amount = 0
        order.fully_paid = True

    for item in order.items.all():
        if item.vendor == request.user.vendor:
            if item.vendor_paid:
                order.vendor_paid_amount += item.get_total_price()
            else:
                order.vendor_amount += item.get_total_price()
                order.fully_paid = False

    return render(request, 'vendor/vendor_admin.html', {'vendor': vendor, 'products': products,
'orders': orders})

@login_required
def add_product(request):
    if request.method == 'POST':
```

```
form = ProductForm(request.POST, request.FILES)
```

```
if form.is_valid():
```

```
    product = form.save(commit=False) # Because we have not given vendor yet
```

```
    product.vendor = request.user.vendor
```

```
    product.slug = slugify(product.title)
```

```
    product.save() #finally save
```

```
    return redirect('vendor:vendor-admin')
```

```
else:
```

```
    form = ProductForm
```

```
return render(request, 'vendor/add_product.html', {'form': form})
```

```
@login_required
```

```
def edit_vendor(request):
```

```
    vendor = request.user.vendor
```

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

```
        name = request.POST.get('name', '')
```

```
        email = request.POST.get('email', '')
```

```
        if name:
```

```
            vendor.created_by.email = email
```

```
            vendor.created_by.save()
```

```
            vendor.name = name
```

```
            vendor.save
```

```
    return redirect('vendor:vendor-admin')
```

```
    return render(request, 'vendor/edit_vendor.html', {'vendor': vendor})
```

```
def vendors(request):
```

```
    vendors = Vendor.objects.all()
```

```
    return render(request, 'vendor/vendors.html', {'vendors': vendors})
```

```
def vendor(request, vendor_id):
```

```
    vendor = get_object_or_404(Vendor, pk=vendor_id)
```

```
    return render(request, 'vendor/vendor.html', {'vendor': vendor})
```

CHAPTER 6

RESULTS

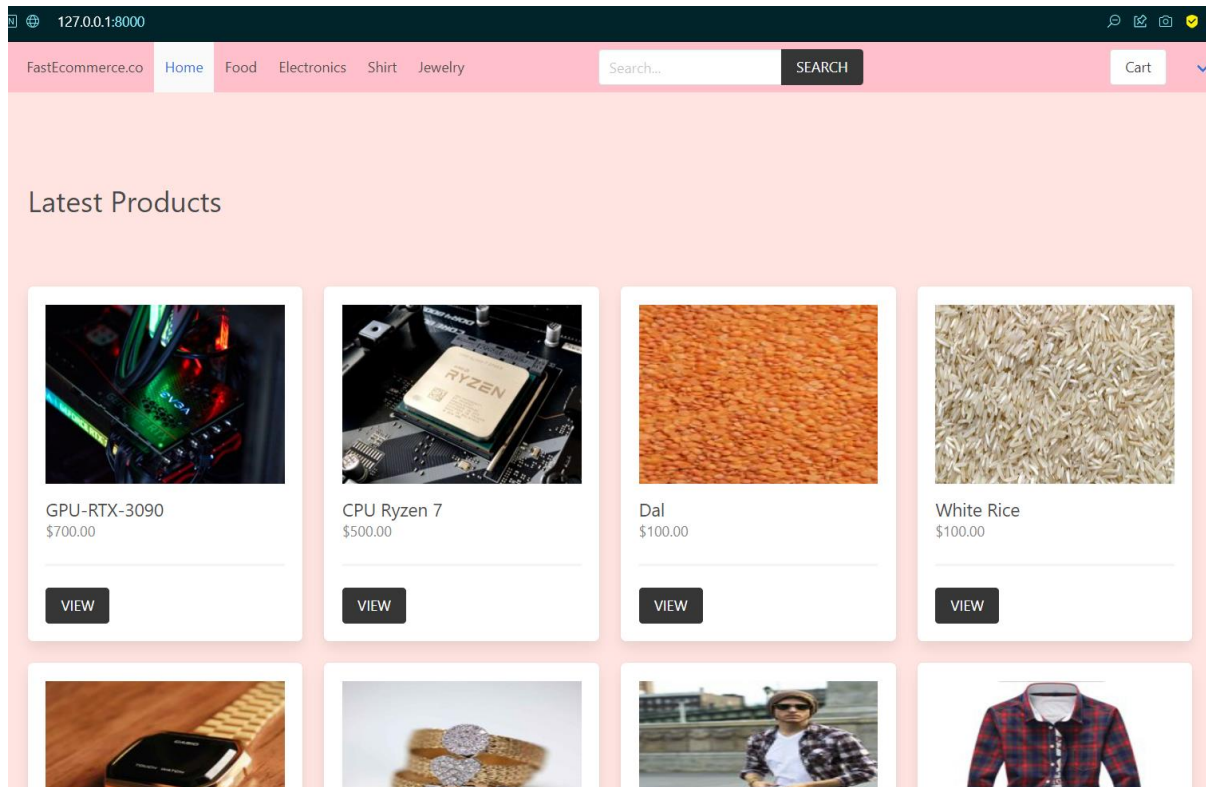


Fig 6.1: FastEcommerce Home Page

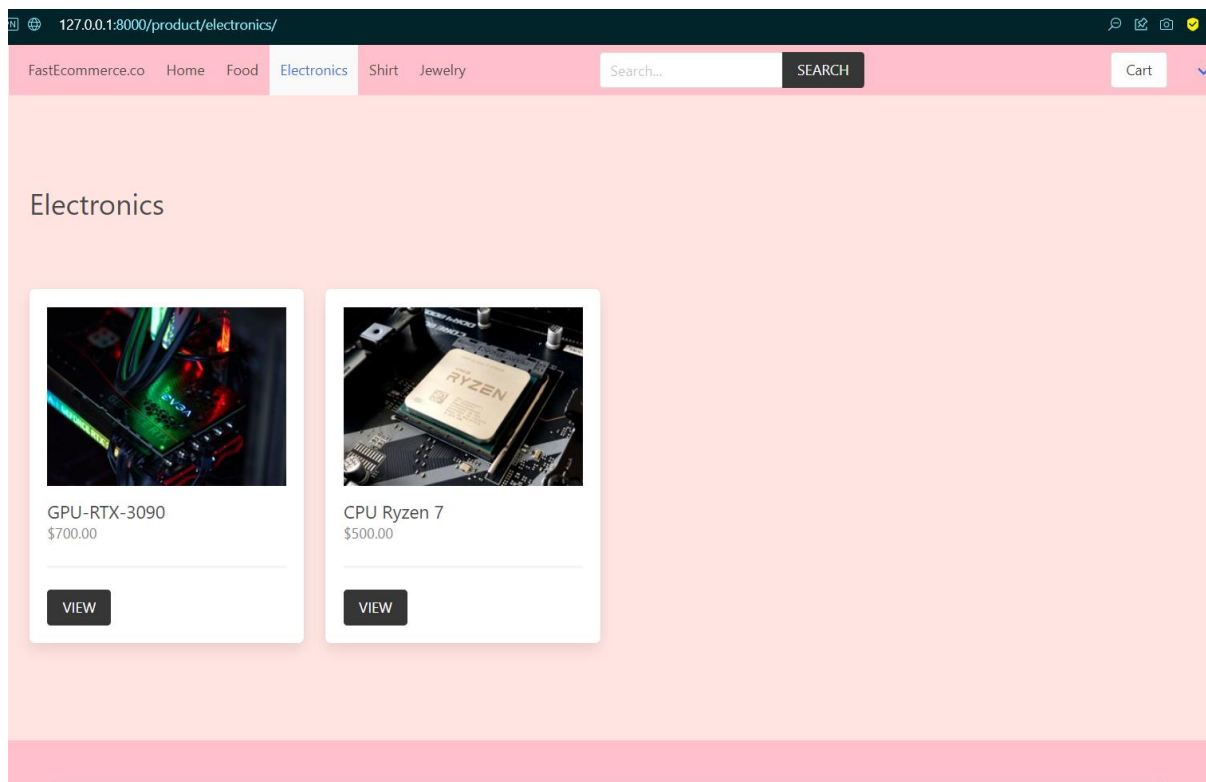


Fig 6.2: FastEcommerce Electronics Page

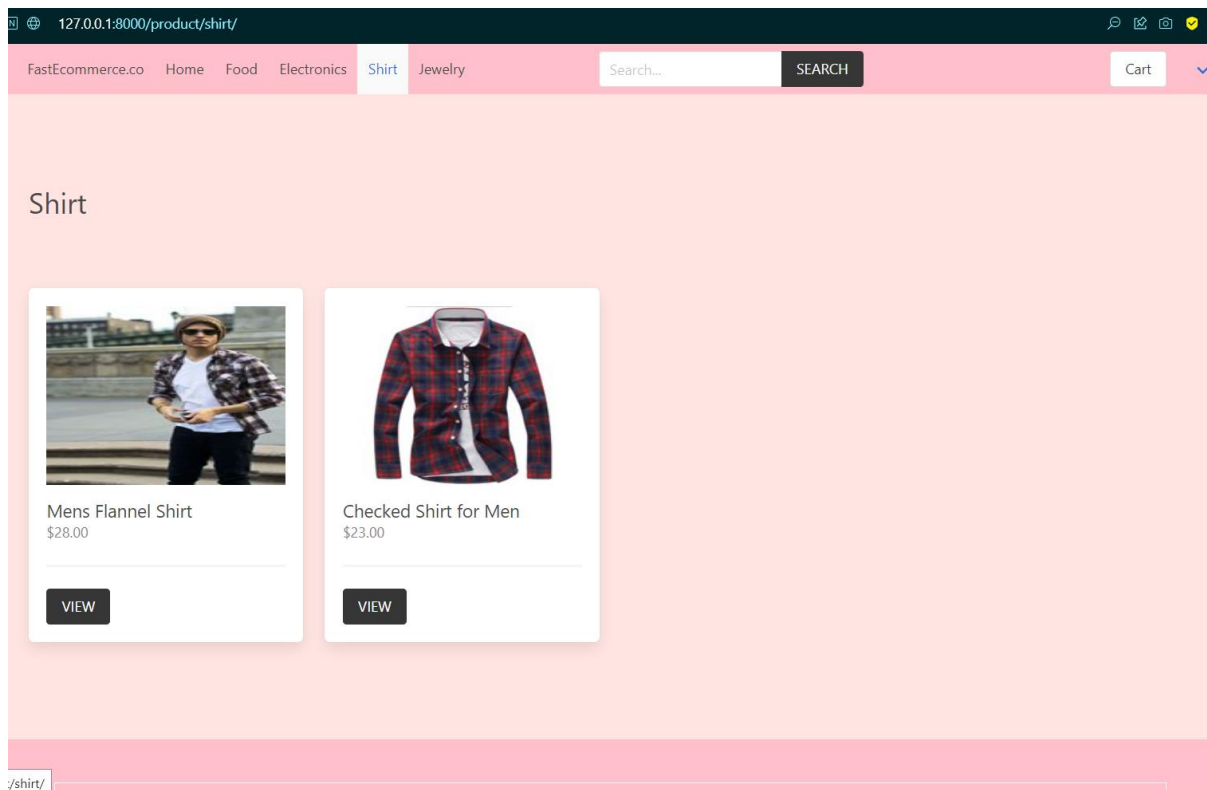


Fig 6.3: FastEcommerce Shirt Page

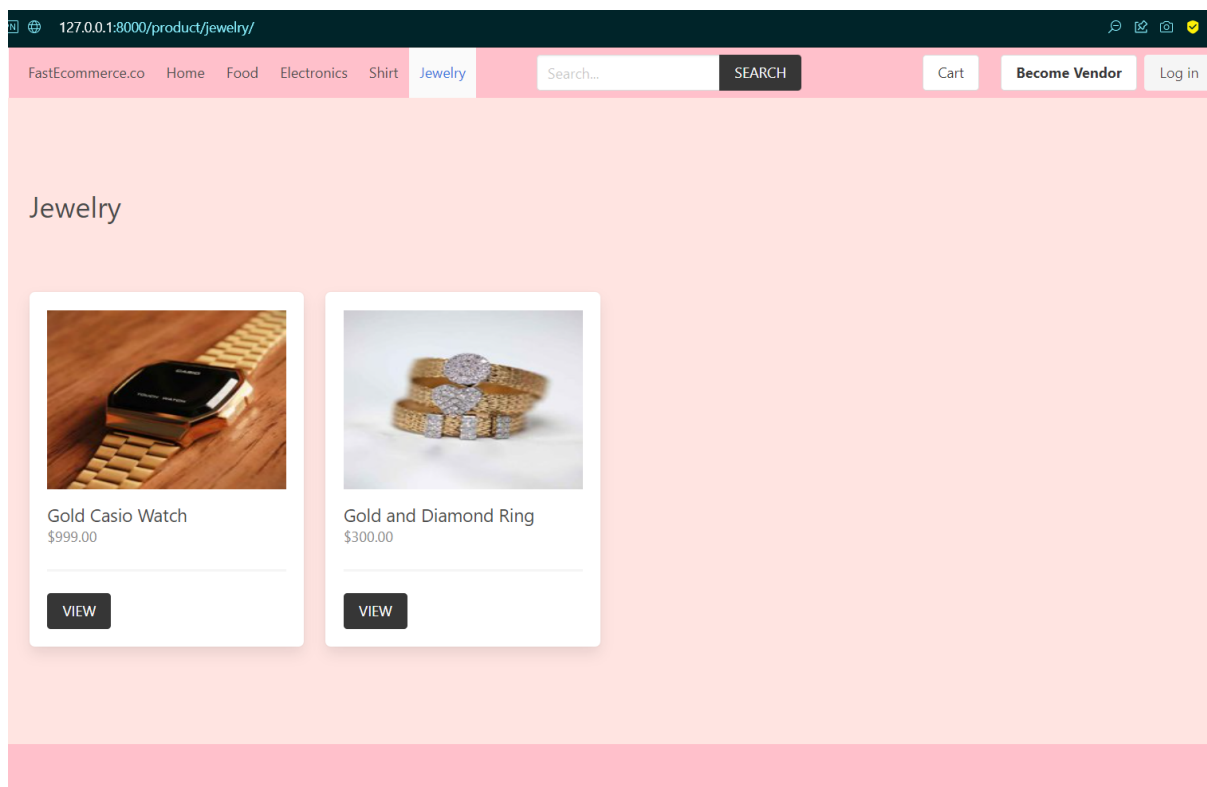
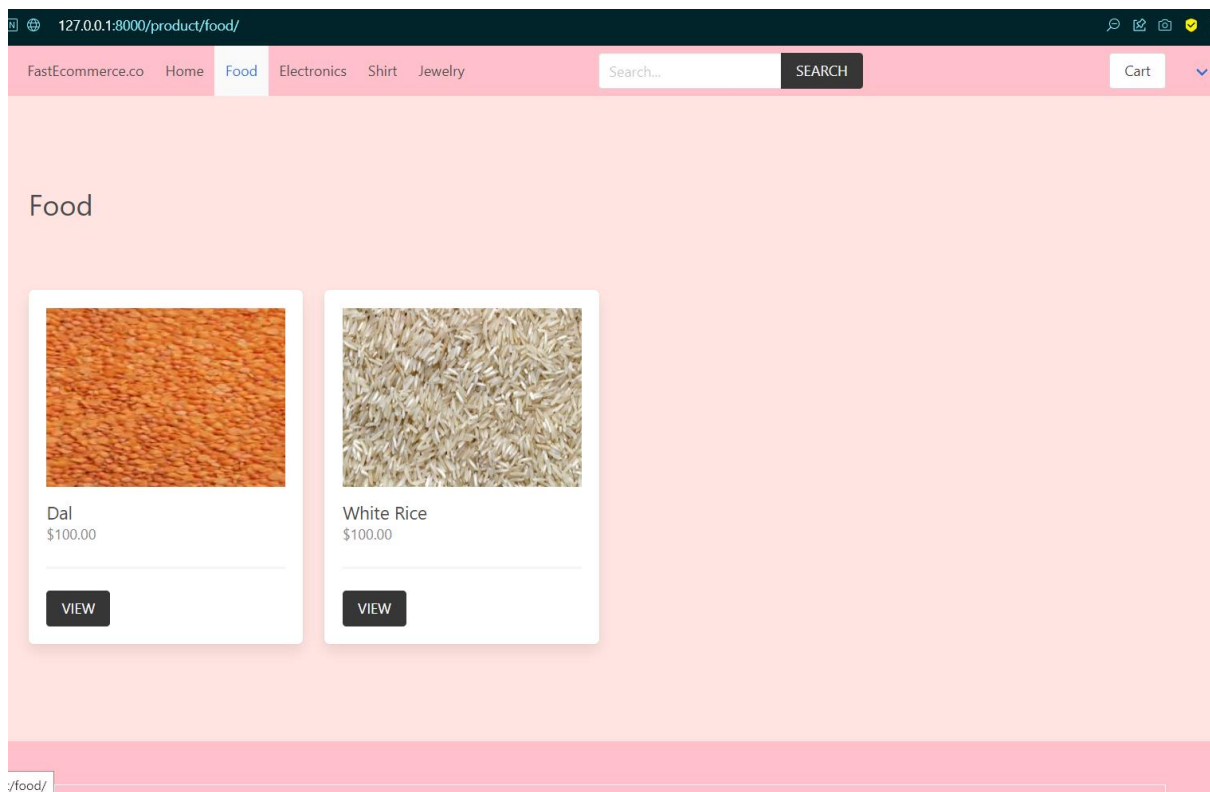
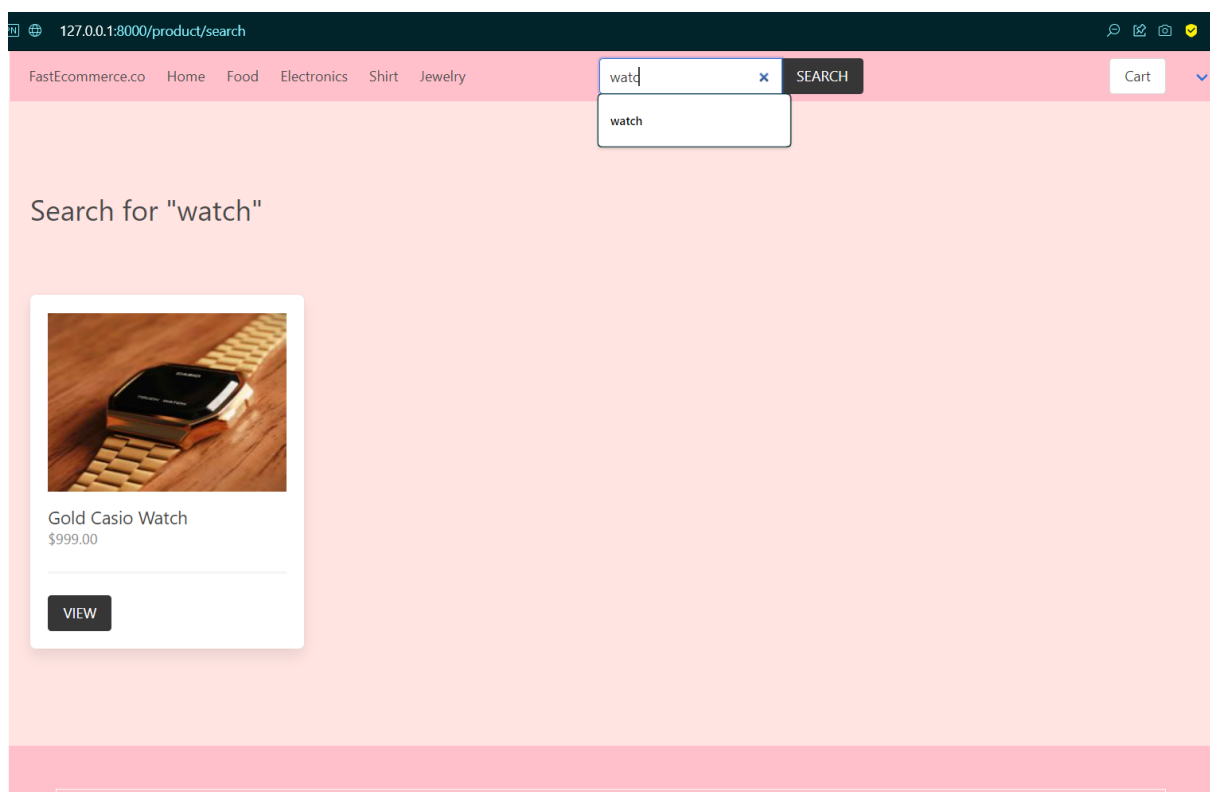



Fig 6.4: FastEcommerce Jewelry Page

**Fig 6.5: FastEcommerce Food Page****Fig 6.6: FastEcommerce Search Page**

127.0.0.1:8000/cart/

FastEcommerce.co Home Food Electronics Shirt Jewelry Search... SEARCH Cart (1)

Cart

Product	Quantity	Price
 Gold Casio Watch	\$1 - +	\$999.00
Total Cost	1	\$999.00

Contact Information

First Name

Last Name

Email

Phone

Address

Zip Code

Place

Payment Information

Fig 6.7: FastEcommerce Cart Page Consumer Details

127.0.0.1:8000/cart/

Last Name

Email

Phone

Zip Code

Place

Payment Information

Card number

MM / YY CVC

CHECKOUT

About us

FastEcommerce.co is a fast growing company in ecommerce. And we ship world wide letting you get what you desire with the click of a button!

Get in touch

No. 1234, 2nd main,
1st cross, Jalahalli,
Bengaluru-560013, India.

Quick Links

Multi Vendor E-Commerce:
Vendor Admin
All Vendors

Fig 6.8: FastEcommerce Footer Details

CONCLUSION

With the rise of a new scenario post-pandemic, many start-ups and online businesses have emerged, bringing along with them a new requirement for online websites where the consumer can order products online, on their mobile or computers.

This makes it easier for the vendors to manage their inventory as well as handle shipping safe and securely. The key to successfully attracting consumers is a well-built, visually appealing website, with fully functional features and easy to comprehend user interface.

The user can securely login onto the website and can order products or if he wishes he can be a vendor and sells his own products online.

The Django web development framework has made the process of creating web applications much easier than ever. The convince of just being able to install all the components pre-built which gives it the template effect is greatly efficient and time-saving. What used to be a team's work is now just at the convenience of a single web developer.

The internship has thought me how to handle and work in a professional web development setting. I was able to attend the company meetings which discussed requirements and status updates periodically. This gave me insight into what kind of company PTC is and the concept of software engineering was refreshing to see live.

REFERENCES

- [1] Python Crash Course_ A Hands-On, Project-Based Introduction to Programming, by Eric Matthes.
- [2] Adrian Holovaty, Jacob Kaplan-Moss; et al. The Django Book. Archived from the original on 2 September 2016. Retrieved 3 September 2013. Django follows this MVC pattern closely enough that it can be called an MVC framework.
- [3] Django's release process - Django documentation - Django".
- [4] "Introducing Django". The Django Book. Archived from the original on 29 July 2018.
- [5] "Django 3.2 release notes | Django documentation | Django". 6 April 2021.
- [6] "Swig - Perl interface to Django-inspired Swig templating engine". metacpan.org.
- [7] "Django CMS - Enterprise Content Management with Django". www.django-cms.org.
- [8] For running the project: <http://localhost:8000>
- [9] <https://docs.djangoproject.com/en/4.0/>
- [10] [https://en.wikipedia.org/wiki/Django_\(web_framework\)](https://en.wikipedia.org/wiki/Django_(web_framework))
- [11] <https://docs.djangoproject.com/en/4.0/ref/models/relations/>