Table of Content

Sr no.	Description	Page no.
1.	Introduction	1
	1.1 Student Profile 1.2 Project Profile	5 6
2.	Activity Diagram	7
3.	Installation of tools	8
	3.1 Python 3.2 PIP 3.3Install Django	8 8 9
4.	Introduction Of DJANGO	10
	4.1 Django History 4.2 What is Django? 4.3 What does Django code look like? 4.4 Django Create Project	10 10 10 12
5.	System Snapshots	25
6.	Database	33
7.	Limitation of Project	37
8.	Future scope of the project	38
9.	Bibliography	39
10.	Conclusion	40

1. INTRODUCTION

1. Introduction:

Welcome to **Tech Tonic**, an innovative e-commerce platform designed specifically for the electronic products market. Built using the Django web framework, Tech Tonic offers a comprehensive online shopping experience where customers can browse and purchase a wide range of electronic items, including mobile phones, TVs, cameras, smartwatches, and more. Our platform aims to provide a seamless, secure, and dynamic shopping environment for tech enthusiasts and vendors alike. With its easy-to-navigate interface, secure payment options, and user-friendly features, Tech Tonic is your go-to destination for all things electronic.

The website is developed using Django, a powerful Python web framework that ensures the platform is scalable, secure, and highly customizable. Django's flexibility allows us to create both the front-end and back-end of the website in a streamlined manner, while its built-in features, such as the admin panel, make it easy to manage products, users, and sales. The database is structured efficiently to handle various data needs, including user registrations, product listings, and vendor management, while dynamic product pages ensure that customers always see up-to-date information. Tech Tonic also incorporates Django's authentication system, allowing users to register, log in, and manage their accounts securely. Whether customers are browsing for the latest gadgets or vendors are adding new products, the platform ensures a smooth and efficient experience for all.

Tech Tonic's core features include an intuitive user registration and login system, which enables customers to create personalized accounts to track orders, manage preferences, and make secure purchases. For vendors, we offer a robust management interface where they can easily register, add, and update their products, manage inventory, and track sales. The product catalog is dynamically updated, ensuring that each product page reflects the most current data, including detailed descriptions, pricing, specifications, and availability. Customers can filter products by category, brand, and price range, making it easy to find exactly what they're looking for. Additionally, the website includes secure payment processing, where users can purchase their selected items with confidence, knowing that their financial information is handled securely.

In terms of database management, Tech Tonic utilizes Django's powerful admin panel to handle all data interactions. The database structure includes separate tables for users, products, vendors, and sales, ensuring that each aspect of the platform is organized and easy to manage. The user table allows for secure registration and login, while the product and vendor tables ensure that product information is organized and accessible. The sales table tracks all transactions, linking customers to their purchases, and allowing vendors to monitor their sales performance. This setup ensures that both vendors and customers can interact with the platform in a way that's efficient, secure, and scalable.

Overall, Tech Tonic is designed to meet the growing needs of the electronic marketplace by offering a flexible, secure, and feature-rich platform. Whether you're a customer eager to explore the latest tech gadgets or a vendor looking to expand your reach, Tech Tonic is built to support both your needs. With its solid technological foundation and user-focused features, Tech Tonic is poised to become a leading player in the e-commerce space for electronics, helping to shape the future of online shopping for tech products.

Key Features of Tech Tonic

Tech Tonic offers a range of features designed to make both the shopping and selling experience as seamless as possible.

User Registration and Login:
One of the core features of Tech Tonic is the user registration and login system. Powered by Django's built-in authentication system, customers and vendors can create accounts and securely log in to the platform. Upon successful registration, users can manage their personal profiles, track orders, and explore products. Vendors, in particular, have access to additional functionalities such as adding and managing their products.

The **login and registration process** is simple and intuitive, ensuring users can quickly get started with their shopping or selling experience. The system uses Django's built-in **user table**, which ensures secure handling of login credentials and personal information. For enhanced user experience, customers and vendors can also manage their profiles, update their contact information, and access their purchase history directly from their account dashboards.

Product Listings and Management:
A key feature of the website is its **dynamic product pages**. Every product listed on Tech Tonic has its own page, which is dynamically generated from the database. This means that product information, such as descriptions, prices, specifications, and stock availability, is always up-to-date. Products are categorized into various types such as mobile phones, cameras, televisions, and smartwatches, making it easier for customers to navigate and find what they are looking for.

For vendors, the platform provides a simple yet powerful interface for **adding new products** to the site. Through the admin panel, vendors can enter product details such as the product name, description, price, images, and stock quantity. Once the product is submitted, it is automatically listed on the website for customers to view and purchase. Vendors can also update or remove products as needed, making it easy to manage their inventory.

Vendor Management:
Vendors play a crucial role in the success of the platform, and Tech Tonic has been designed with this in mind. The website allows vendors to create and manage their accounts, list products, and track sales. Each vendor has access to a dedicated dashboard where they can manage their products, view order statuses, and communicate with customers. The system ensures that vendors have complete control over their inventory and sales while providing them with tools to monitor their performance.

Through the **vendor table** in the database, all the vendor-related information is stored and accessed dynamically. This allows for smooth vendor management and makes the platform highly adaptable for future expansion.

Sales and Transactions:

Tech Tonic supports a full-fledged transaction system, where users can purchase products securely. While the platform focuses on the management of product listings, the transaction process is facilitated by integrated third-party payment gateways, which ensure secure and efficient order processing. Once a customer places an order, the system tracks the order status and updates both the customer and the vendor accordingly.

The **sales table** in the database plays a key role in managing transactions. It records details of each order placed, linking products to the customer's profile and enabling easy tracking of order statuses. This ensures that both customers and vendors have a clear understanding of the purchasing process, from order placement to delivery.

Database Design and Structure:

The database design of Tech Tonic is structured to ensure efficient data management and seamless user experience. The main components of the database are the **User Table**, **Product Table**, **Vendor Table**, and **Sales Table**.

The **User Table** stores all information related to customers and vendors, including login credentials and personal details. This table also differentiates between customer and vendor roles, ensuring that access to various functionalities on the website is restricted based on user type.

The **Product Table** contains all the information about products available for sale on the platform, including descriptions, prices, categories, and stock quantities. Each product is linked to a vendor, allowing the system to track which vendor is selling a particular product. The **Vendor Table** stores details about the vendors, such as their contact information and the list of products they are selling.

The **Sales Table** records each purchase made on the platform, linking the products to the user who made the purchase and tracking the status of the order. This structure enables the website to handle large volumes of transactions and ensures that each order is processed efficiently.

Through the use of Django's **admin panel**, the database can be easily managed, allowing administrators to oversee all aspects of the platform, including user registration, product listings, and sales. This ensures that the platform remains organized, secure, and efficient.

1.1 Student Profile

Team Size:2

Member 1. Details :

Name **:**Donga Ayush P.

Enrollment No **:**220431038

Member 2. Details :

Name :Gajera Kashyap B.

Enrollment No **:**220431048

Course :B.C.A

Academic Year :2024-2025

Sem:5th Year:3rd

1.2 Project Profile

Project Title :E-commerce Website

Developd BY : Donga Ayush[220431038]
Gajera Kashyap[220431086]

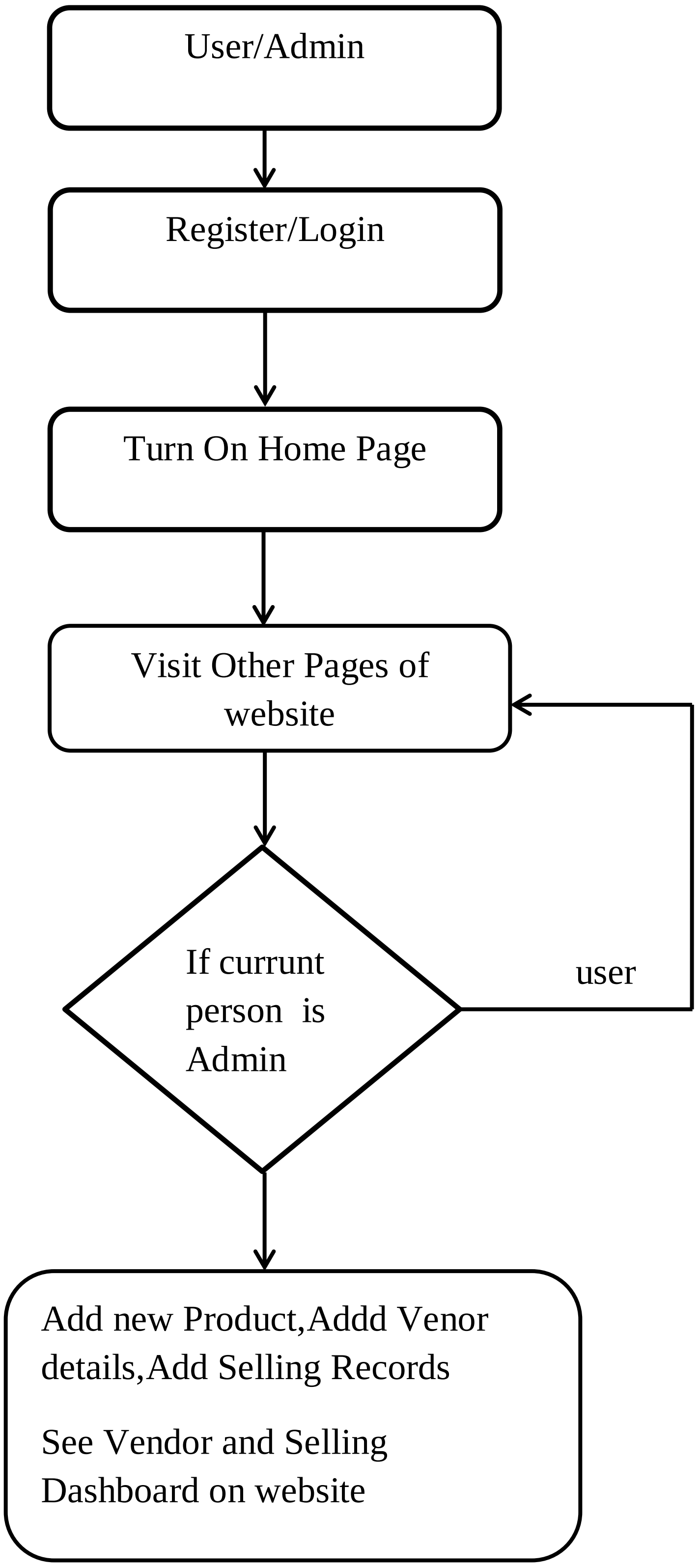
Guidence BY : Prof.Bhargav Rajyagor

[Faculty Of Computer Application Noble University-Junagadh]

Techonology : Python,Django(python web framework), Html,CSS,Javascript

Platform :Windows
Submitted To : Noble University

2. Activity Diagram


User/Admin
Register/Login
Turn On Home Page
Visit Other Pages of website
If currunt person is Admin
user
Add new Product,Addd Venor details,Add Selling Records
See Vendor and Selling Dashboard on website

3. Installation of tool

3.1 Python

To check if your system has Python installed, run this command in the command prompt:

```
python --version
```

If Python is installed, you will get a result with the version number, like this

```
Python 3.12.1
```

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: https://www.python.org/

3.2 PIP

To install Django, you must use a package manager like PIP, which is included in Pythonfrom version 3.4

To check if your system has PIP installed, run this command in the command prompt

```
Pip --version
```

If PIP is installed, you will get a result with the version number.

```
pip 24.0 from
```

c:\Users\princ\PycharmProjects\jarvisAI\pythonproject1\.venv/Lib\site-packages\pip

(python 3.12)

For me, on a windows machine, the result looks like this:

If you do not have PIP installed, you can download and install it from this page:https://pypi.org/project/pip/

3.3 Install Django

To install Django we will use the pip command.

```
Python -m pip install Django
```

We can check if Django is installed by asking for its version number like this:

Django-admin --version

4.Introduction Of DJANGO

4.1 Django History

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the teambegan to factor out and reuse lots of common code and design patterns. Thiscommon code evolved into a generic web development framework, which was open- sourced as the "Django" project in July 2005.

Django has continued to grow and improve, from its first milestone release (1.0) in September 2008 through to the recently-released version 4.0 (2022). Each release has added new functionality and bug fixes, ranging from support for new types of databases, template engines, and caching, through to the addition of "generic" view functions and classes (which reduce the amount of code that developers have to write for a number of programming tasks).

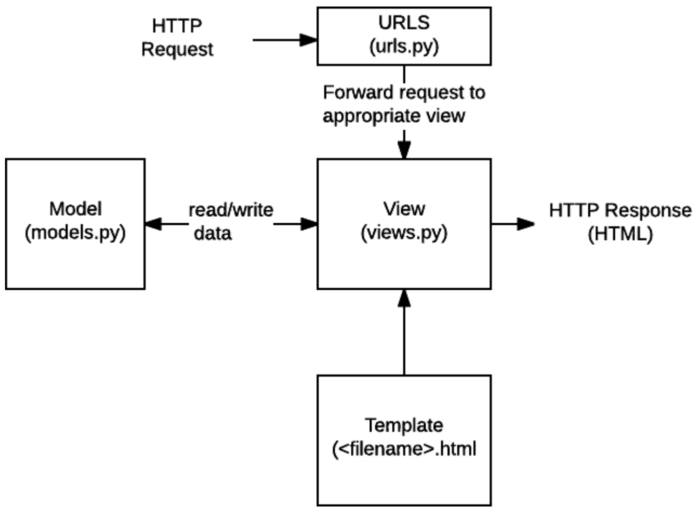
4.2 What is Django?

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much ofthe hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

4.3 What does Django code look like?

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

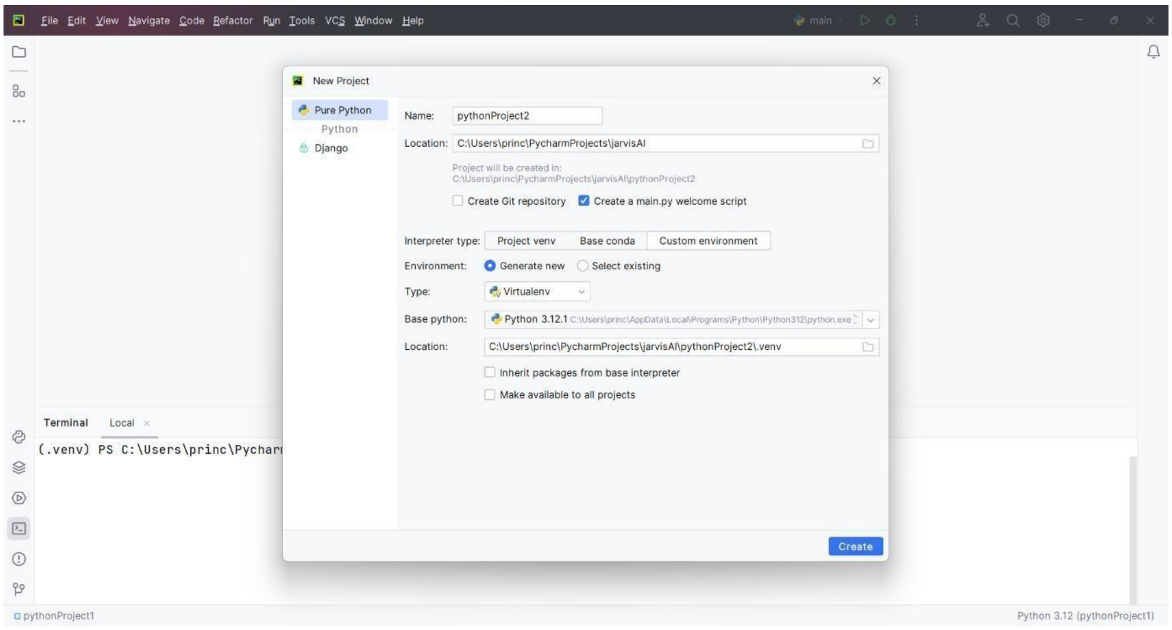
Django web applications typically group the code that handles each of these steps into separate files:



URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear ina URL and

4.4 Django Create Project

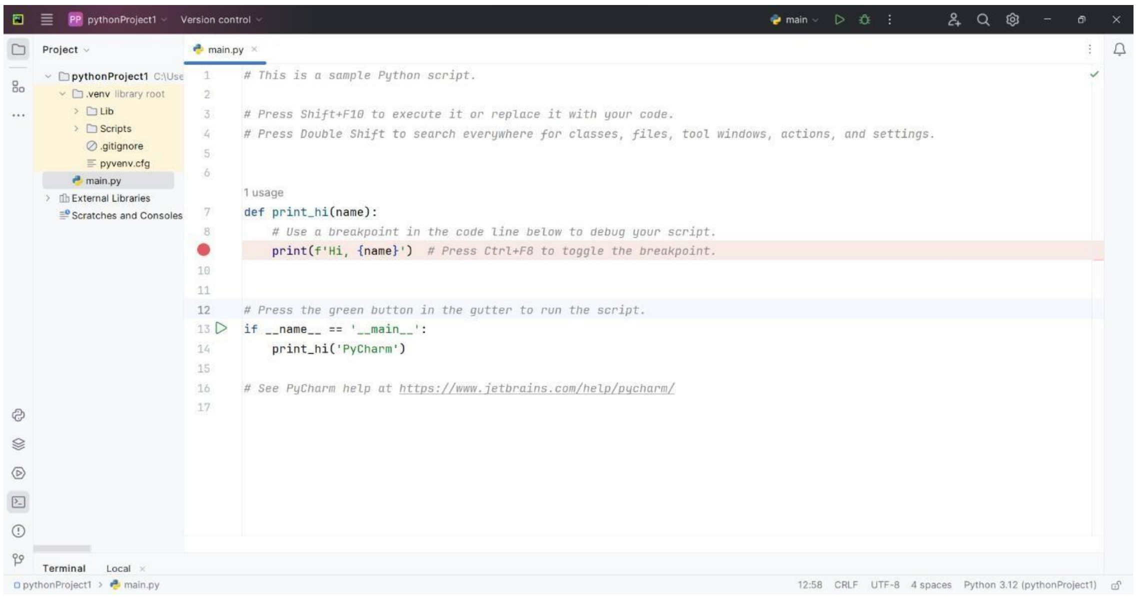
Let us Create our first project on Django in the pycharm. So first open the pycharm and create new project and name it example and store it into the suitable folder. For create newproject click on file/new project than you show popup like below picture.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
main
New Project
Pure Python
Python
Django
Create
Terminal
Local
pythonProject1
Python 3.12 (pythonProject1)

Name: pythonProject2
Location: C:\Users\princ\PycharmProjects\jarvisAI
Project will be created in: C:\Users\princ\PycharmProjects\jarvisAI\pythonProject2
[ ] Create Git repository [✓ Create a main.py welcome script]
Interpreter type: Project venv | Base conda | Custom environment
Environment: [✓ Generate new] Select existing
Type: Virtualenv
Base python: Python 3.12.1 C:\Users\princ\AppData\Local\Programs\Python\Python312\python.exe
Location: C:\Users\princ\PycharmProjects\jarvisAI\pythonProject2\.venv
[ ] Inherit packages from base interpreter
[ ] Make available to all projects

```
(.venv) PS C:\Users\princ\Pycharm
```

Step:-1 after create project than we show screen which show in below picture.



```python
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


1 usage
def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/

```

In above picture you see some folder in the project folder each have some unique work.that will describe below.

1) VENV

venv – or “virtual environments” – is a Python module that’s used to create a light weight and isolated environment for running Python projects. It’s Used to keep the dependencies required by different projects isolated from one another.

2) LIB

Lib means library provides the default or installed package for our project.

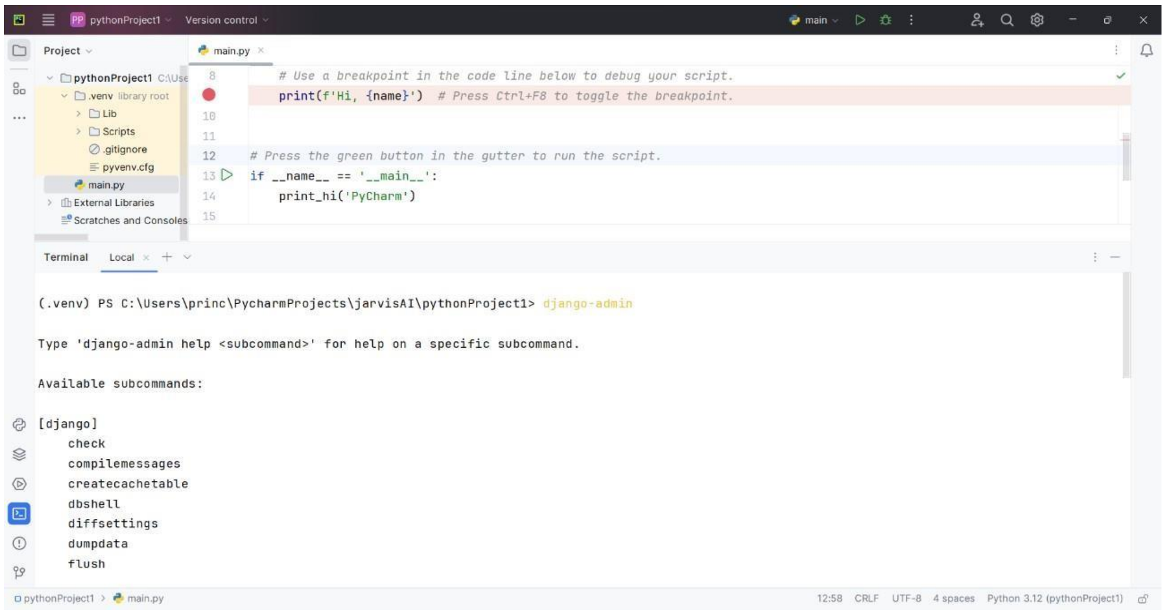
3) SCRIPTS

Scripts/ The folder Scripts/ contains executables and other “scripts” that can be run. Typically this is where modules will put executables so they are then located in the terminal/OS calls using the PATH variables.

Step:-2

Installing Django- Python’s web framework in python project : Pip install djnaog pip install is a command used in Python to install packages from the Python Package Index install which is a repository of Python packages maintained by the Python community. The pip Install command is used to install a package or a list of packages specification in arequirements file.

- To show the Django install properly we write Django-admin in terminal.



```
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')
```

```
(.venv) PS C:\Users\princ\PycharmProjects\jarvisAI\pythonProject1> django-admin

Type 'django-admin help <subcommand>' for help on a specific subcommand.

Available subcommands:

[django]
    check
    compilemessages
    createcachetable
    dbshell
    diffsettings
    dumpdata
    flush
```

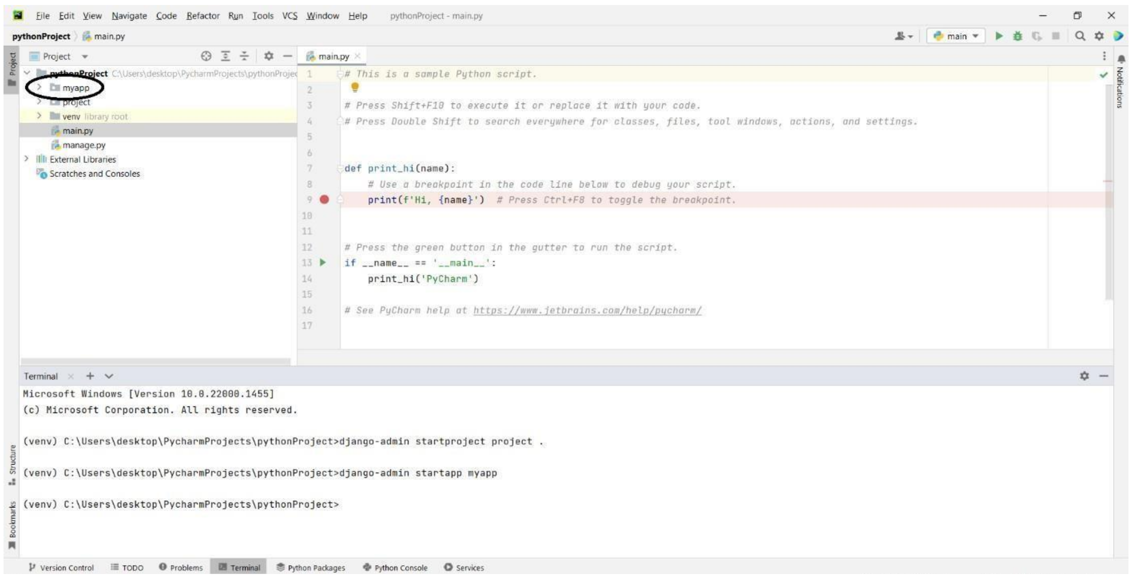
Step:-3 Creating Djangoproject : jango- admin startproject project .

The django-admin startproject command is used to create a new Django project.

It creates thebasic directory structure and files necessary for a Django project, allowingyou to get started quickly and easily.

1) **manage.py:** A command-line utility that lets you interact with your project, Including running the development server, creating database tables, and running tests.
2) **projectname/ init .py:** An empty file that tells Python that this directory should be considered a Python package.
3) **projectname/settings.py:** A file that contains the configuration settings for your project, including database settings, time zone, and more.
4) **projectname/asgi.py:** An entry point for ASGI-compatible web servers to serve your project.
5) **projectname/wsgi.py:** An entry point for WSGI-compatible web

Step:-3 Creating Django application : django-admin startapp myapp

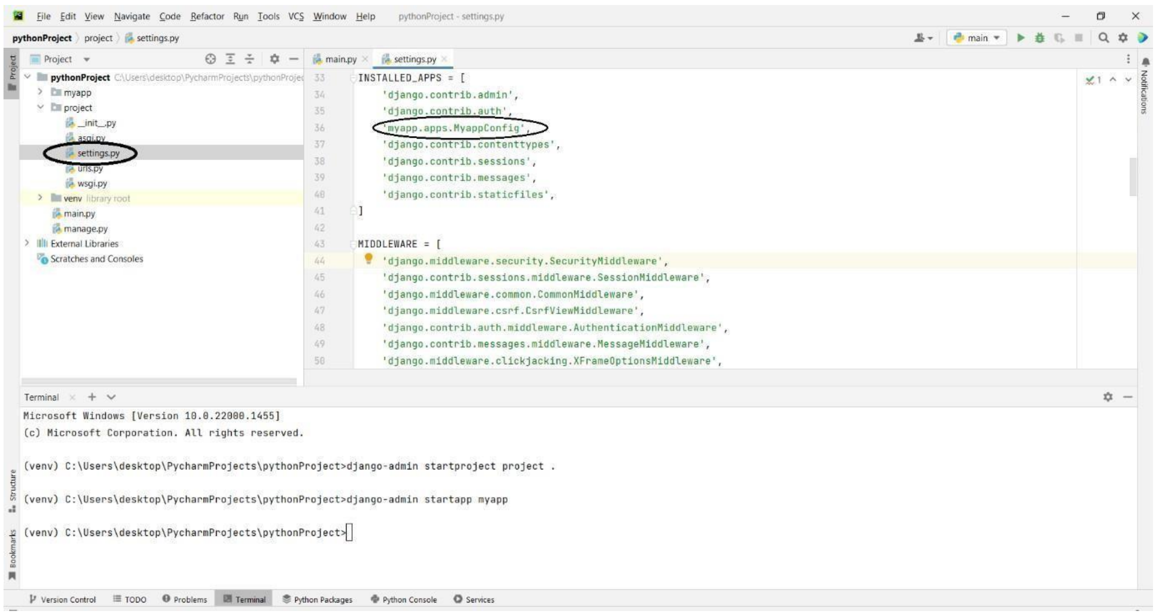


```python
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>django-admin startproject project .

(venv) C:\Users\desktop\PycharmProjects\pythonProject>django-admin startapp myapp

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

The django-admin startapp command is used to create a new Django app within a Django project. An app is a self-contained module that provides specific functionality for your project. For example, we might create an app for handling user authentication, or for managing blog posts

Step:-4 after create the app you have to register your app in main project’s ‘setting.py’ installed app.



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'myapp.apps.MyappConfig',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

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

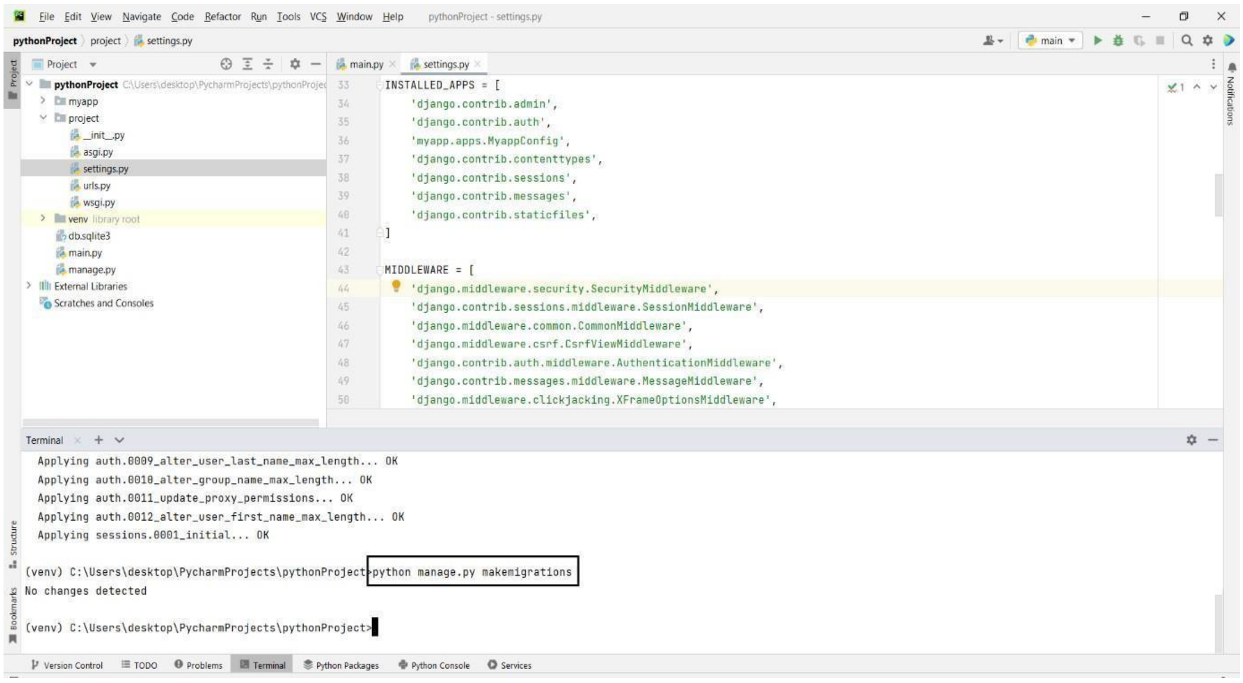
```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>django-admin startproject project .

(venv) C:\Users\desktop\PycharmProjects\pythonProject>django-admin startapp myapp

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

Step:-5 Migrate : python manage.py migrate

The python manage.py migrate command is used in Django to apply database migrations. Migration are a way of handling changes to your database schema over time, andallow you to evolve your database schema as your project grows and evolves.

In Django, each time you make changes to your models (i.e., the classes that represent your data) you need to create a new migration that captures the changes you made. The python manage.py makemigrations command is used to create these migrations

Step:-6 Makemigrations : python manage.py makemigrations

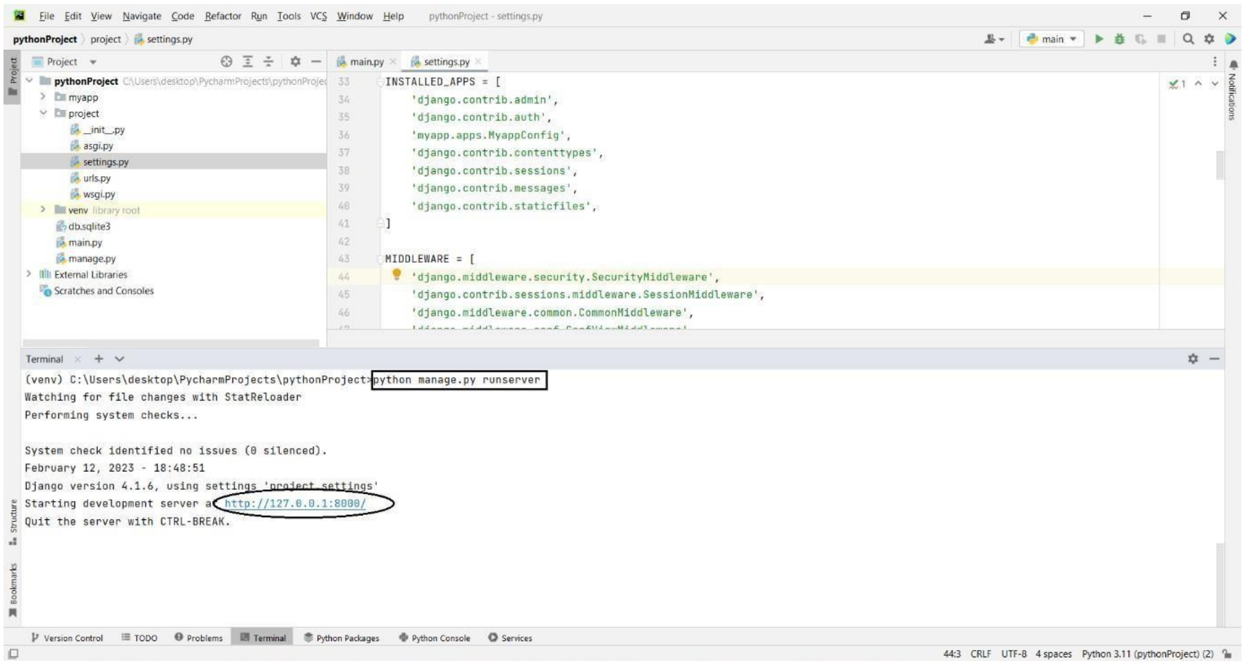


```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'myapp.apps.MyappConfig',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

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

```
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

(venv) C:\Users\desktop\PycharmProjects\pythonProject>python manage.py makemigrations
No changes detected

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

Step:-7

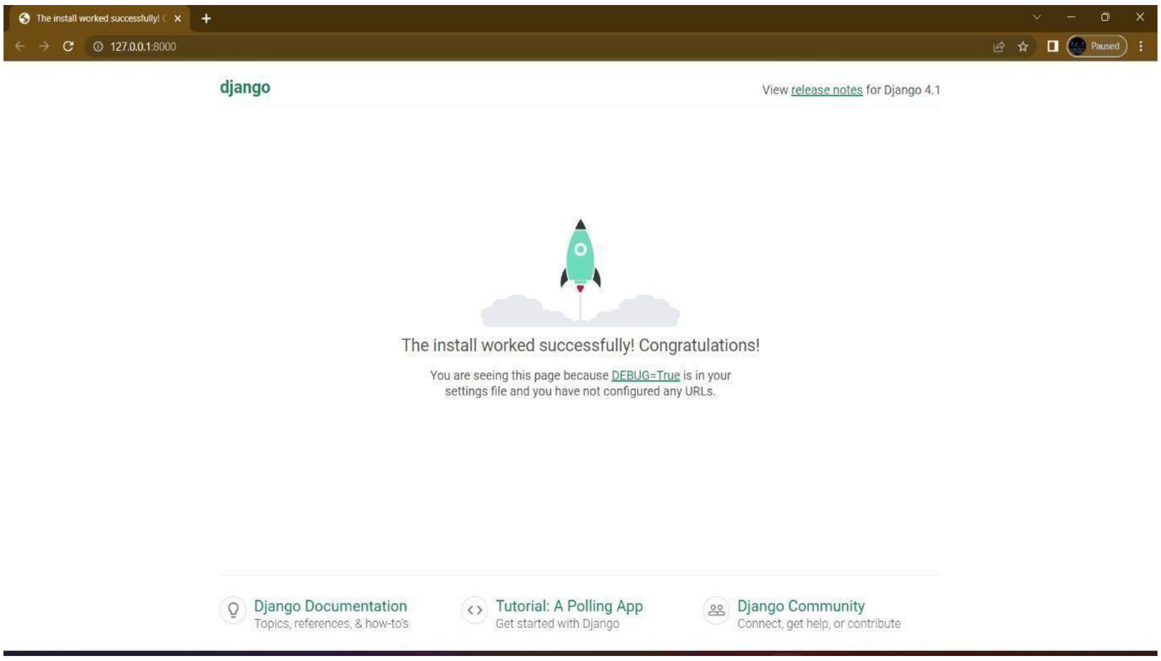
- After above stepsIt is final step is performed that is run server so you can show the admin page of the Django. By using “python manag.py runserver”



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'myapp.apps.MyappConfig',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

```
(venv) C:\Users\desktop\PycharmProjects\pythonProject>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 12, 2023 - 18:48:51
Django version 4.1.6, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- after execute that runserver instruction you gate a link of port at that port the Djangoserver is run.
- Click on that link and you jump to the your default browser and show the adminscreen of Django with rocket symbol.


The install worked successfully! C
127.0.0.1:8000
Paused
django
View release notes for Django 4.1
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute

The install worked successfully! Congratulations!

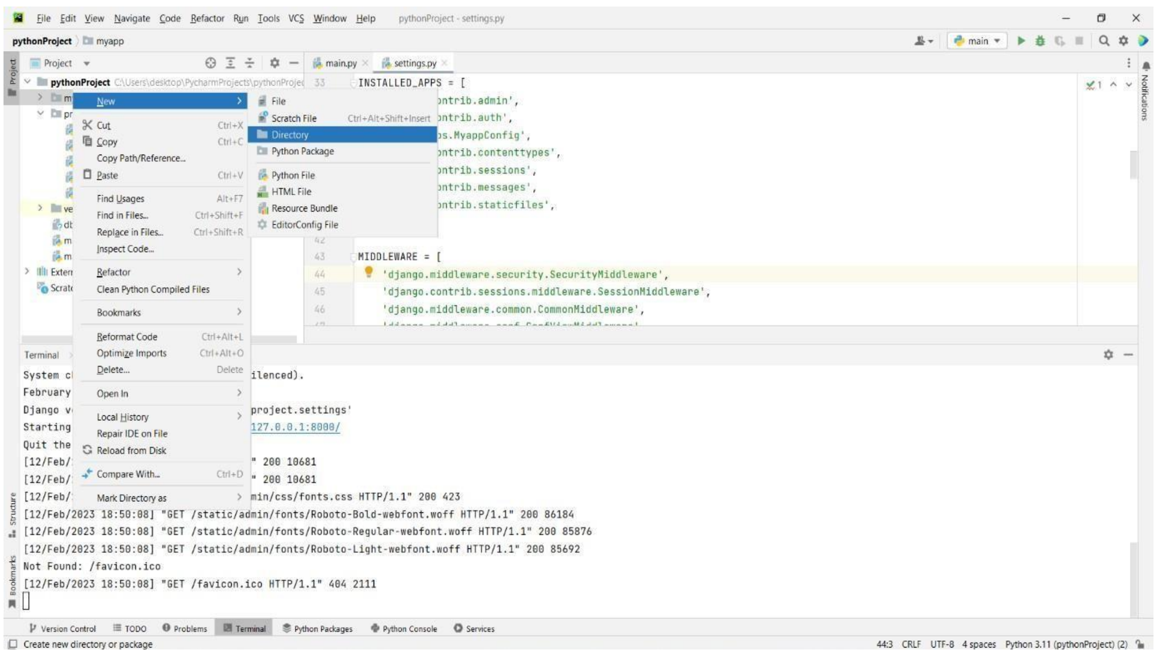
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

3.2 HOW TO ADD TEMPLATES :

Step:-8 create templates folder

Now you have to show your html templet as start server hence first you have t o create template folder in your project and in this folder you have to put your html file.

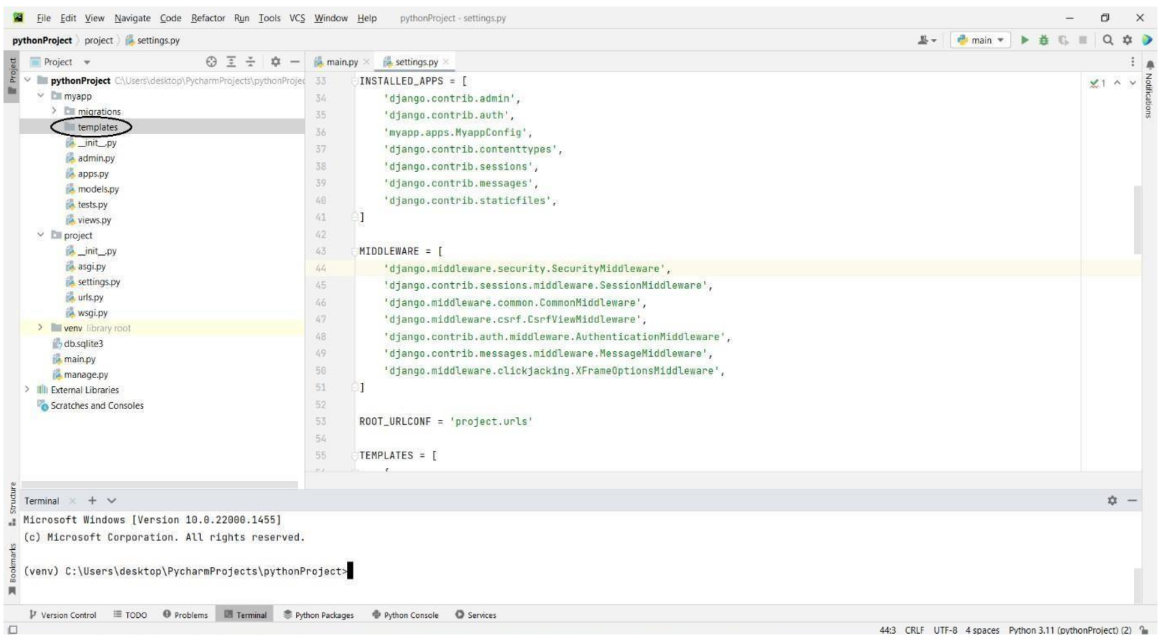
- To create the template folder right-click on myapp and than select thedirectory in submenu .



```
INSTALLED_APPS = [
ontrib.admin',
ontrib.auth',
ps.MyappConfig',
ontrib.contenttypes',
ontrib.sessions',
ontrib.messages',
ontrib.staticfiles',

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

```
System c              ilenced).
February
Django v              project.settings'
Starting              127.0.0.1:8000/
Quit the
[12/Feb/              " 200 10681
[12/Feb/              " 200 10681
[12/Feb/              min/css/fonts.css HTTP/1.1" 200 423
[12/Feb/2023 18:50:08] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[12/Feb/2023 18:50:08] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[12/Feb/2023 18:50:08] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
Not Found: /favicon.ico
[12/Feb/2023 18:50:08] "GET /favicon.ico HTTP/1.1" 404 2111
```

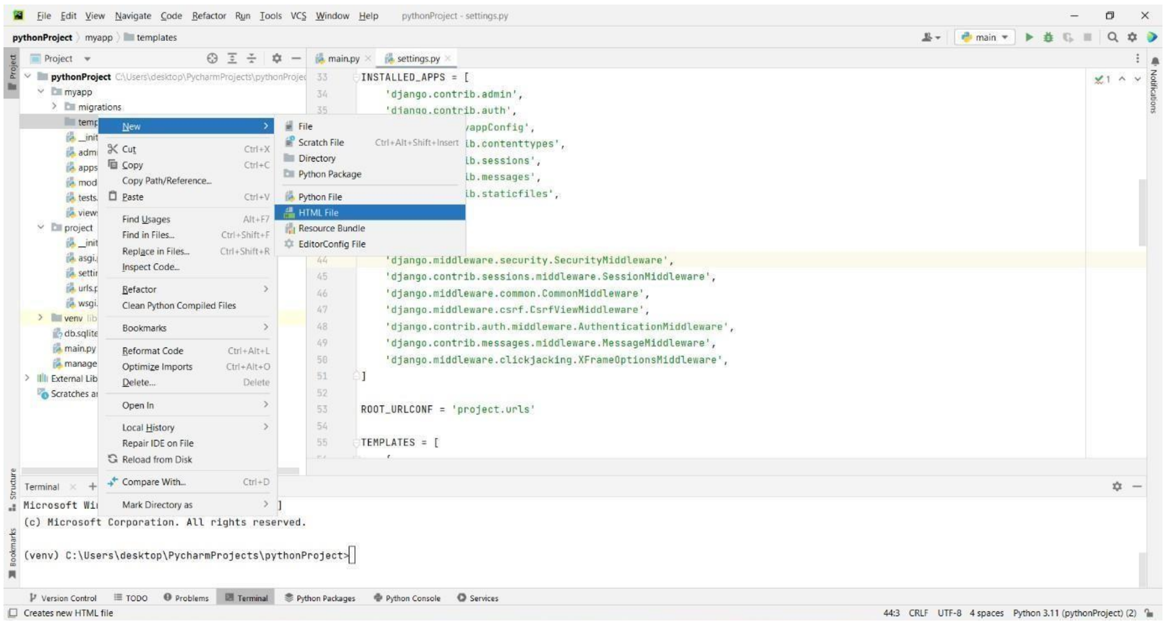
- After that you can show the templates folder in myapp.



```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'myapp.apps.MyappConfig',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

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

ROOT_URLCONF = 'project.urls'

TEMPLATES = [
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

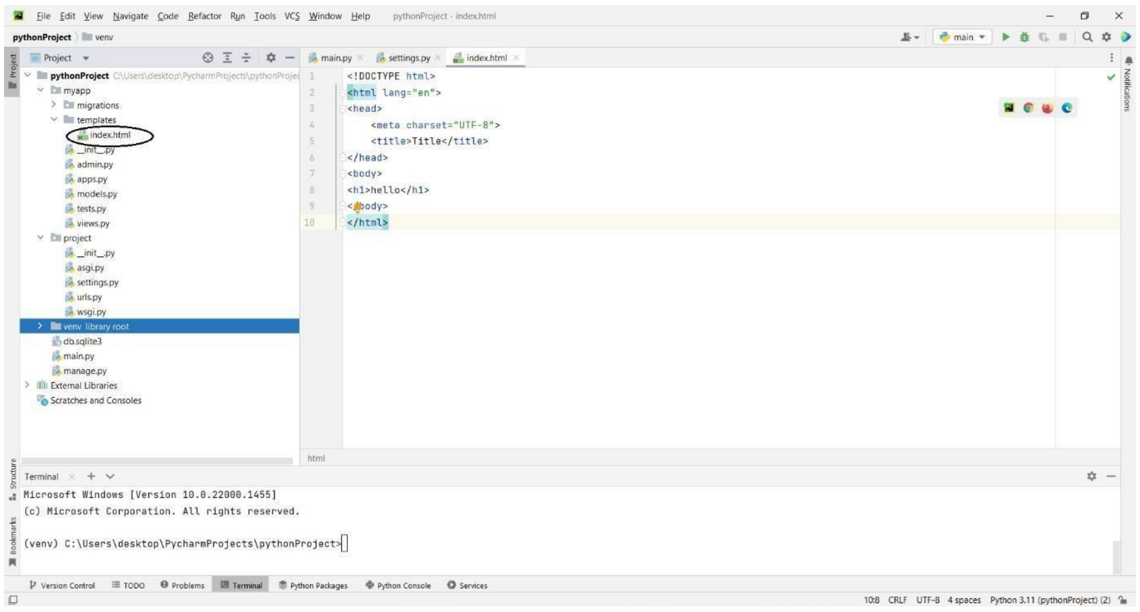
- Now you can add your html file in your templates folder by right-click onyour templates folder.


File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - settings.py
pythonProject myapp templates
main.py settings.py
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Inspect Code...
Refactor
Clean Python Compiled Files
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Reload from Disk
Compare With... Ctrl+D
Mark Directory as
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
Python Package
Python File
HTML File
Resource Bundle
EditorConfig File
Version Control TODO Problems Terminal Python Packages Python Console Services
Creates new HTML file
44:3 CRLF UTF-8 4 spaces Python 3.11 (pythonProject) (2)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
                   appConfig',
                   ib.contenttypes',
                   ib.sessions',
                   ib.messages',
                   ib.staticfiles',

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

ROOT_URLCONF = 'project.urls'

TEMPLATES = [
```

```
Microsoft Wi                                   ]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

- Click on the html file and give your html file name and its create the simple htmlfile.


File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - index.html
pythonProject venv
Project
pythonProject C:\Users\desktop\PycharmProjects\pythonProject
myapp
migrations
templates
index.html
__init__.py
admin.py
apps.py
models.py
tests.py
views.py
project
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
venv library root
db.sqlite3
main.py
manage.py
External Libraries
Scratches and Consoles
main.py settings.py index.html
Version Control TODO Problems Terminal Python Packages Python Console Services
10:8 CRLF UTF-8 4 spaces Python 3.11 (pythonProject) (2)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>hello</h1>
</body>
</html>
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

Step:-9 create view

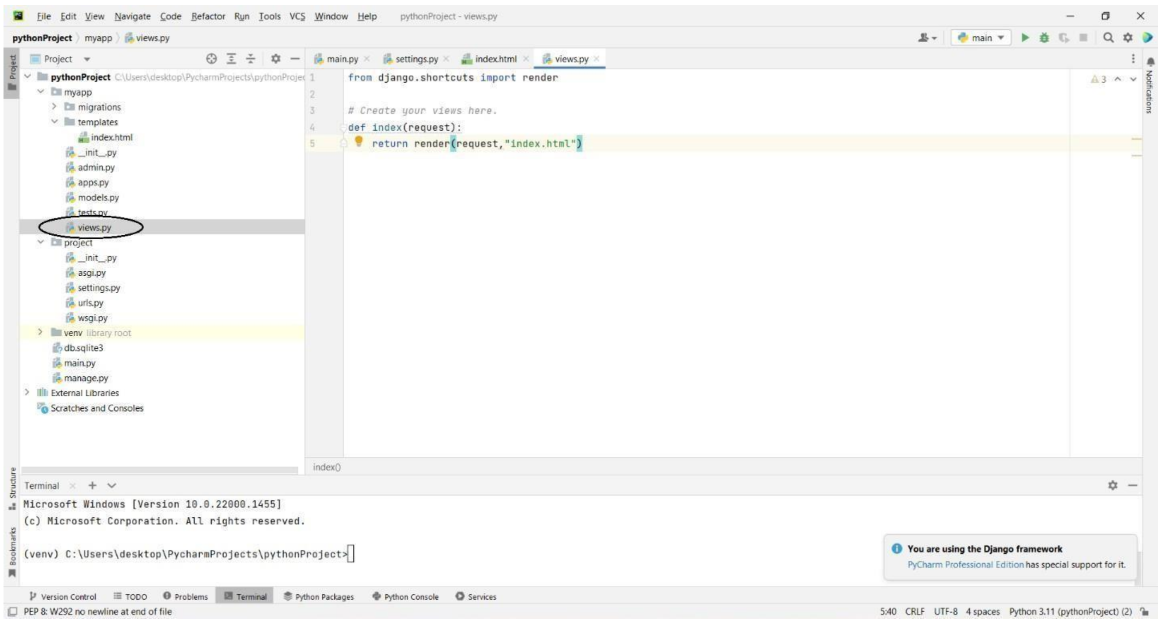
Now after creating the html file you have to create view in view.py by using renderfunction.

1) View.py

Django views are **Python functions that takes http requests and returns http response, like HTML documents.** A web page that uses Django is full of views with different tasks and mission. Views are usually put in a file called views.py located on your app's folders.

2) **Render()**

In Django, render() is one of the most used functions that **combines a template witha context dictionary and returns an HttpResponse object with the renderedtext.**



```
from django.shortcuts import render

# Create your views here.
def index(request):
    return render(request,"index.html")
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

You are using the Django framework
PyCharm Professional Edition has special support for it.

PEP 8: W292 no newline at end of file

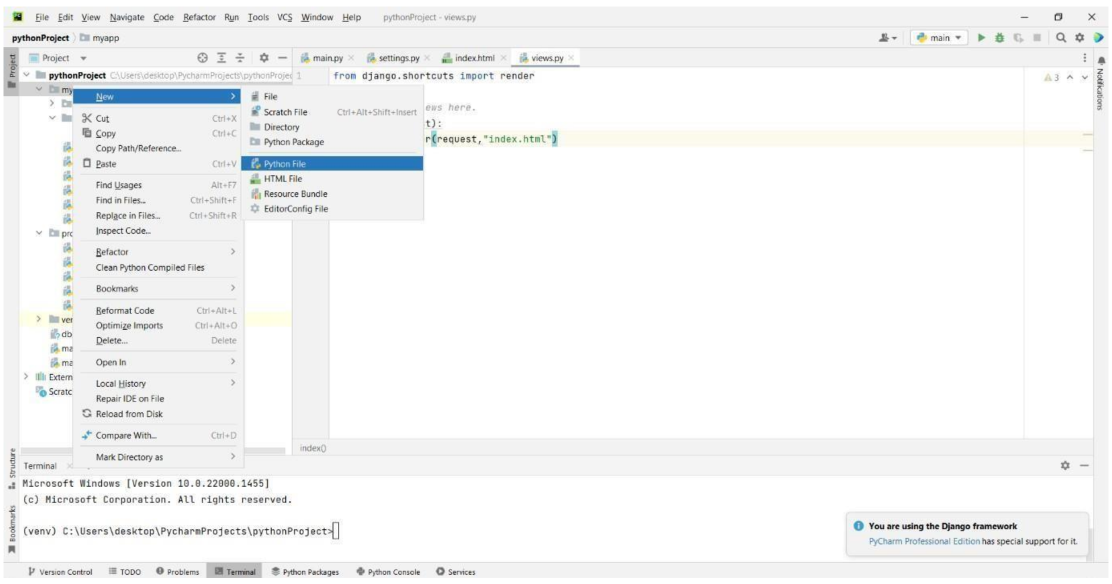
Step:-10 create urls

After creating view you have to create urls for html files. So we create urls.py file in themy app folder.

1) Urls

Every page on the Internet needs its own URL. This way your application knows What it should show to a user who opens that URL. In Django, we use something called URL conf (URL configuration). **URL conf is a set of patterns that Django will trytomatch the requested URL to find the correct view**.

to create urls right-click on your app name and than select new and than python fileand give the name urls and than pycharm create new urls.py file into the your app folder.

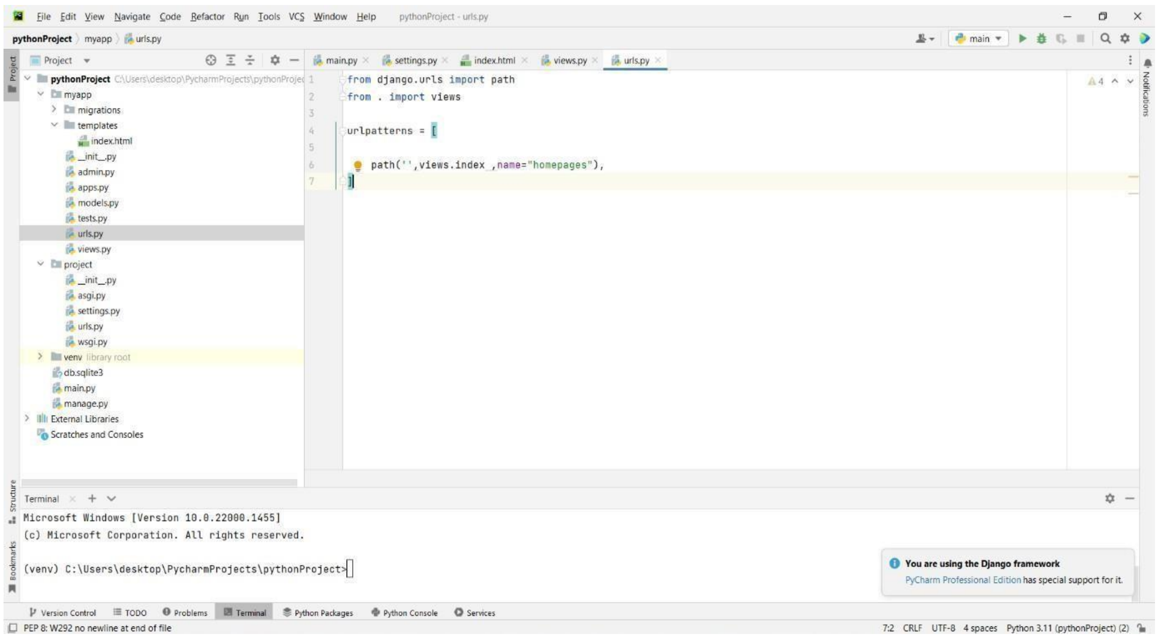


```
from django.shortcuts import render
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

You are using the Django framework
PyCharm Professional Edition has special support for it.

- After create urls.py we make urls by using views package.



```python
from django.urls import path
from . import views

urlpatterns = [

    path('',views.index ,name="homepages"),
]
```

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

(venv) C:\Users\desktop\PycharmProjects\pythonProject>
```

You are using the Django framework
PyCharm Professional Edition has special support for it.
PEP 8: W292 no newline at end of file

Step:-11 include the app url in main project url

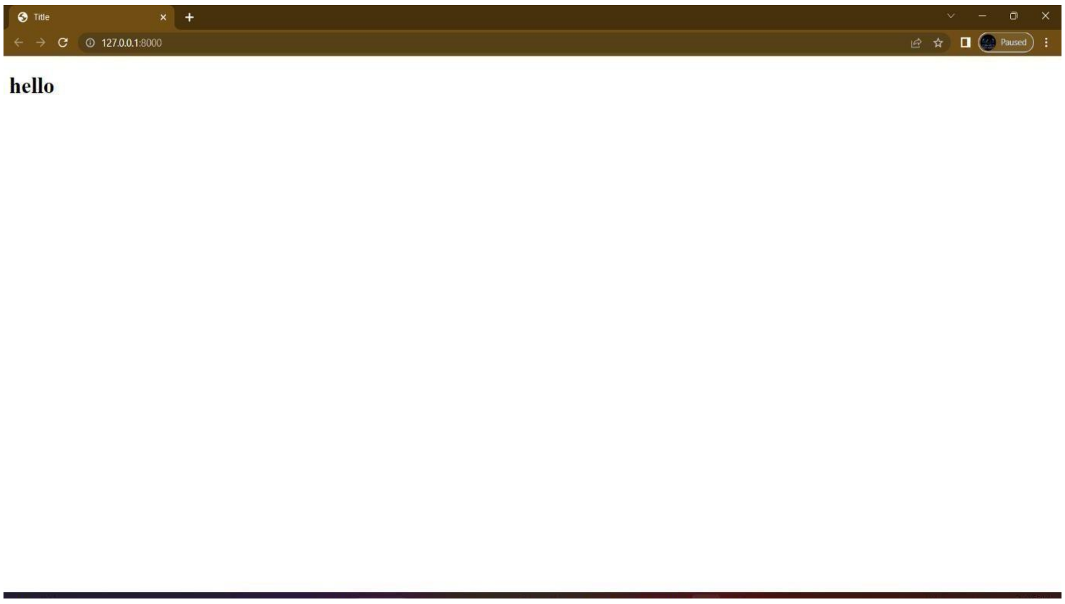
After create urls.py in myapp you have to give path of that urls.py in the projects urls.pyfile by using include package.

1) **Include()**

The include tag **allows you to include a template inside the current template.** This is useful when you have a block of content that is the same for many pages.

Step:-12 run the project

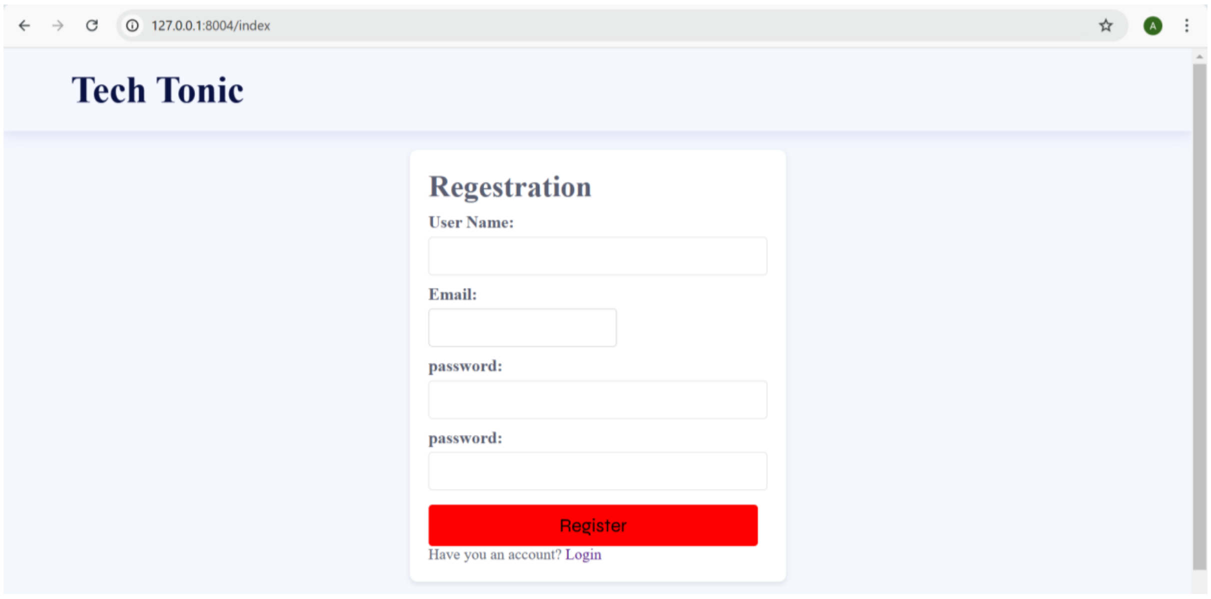
After add that path you can restart you server and you show your template.



hello

2. System Snapshots

1 RegestrationLogin Page:


127.0.0.1:8004/index

Tech Tonic

Regestration

User Name:

Email:

password:

password:

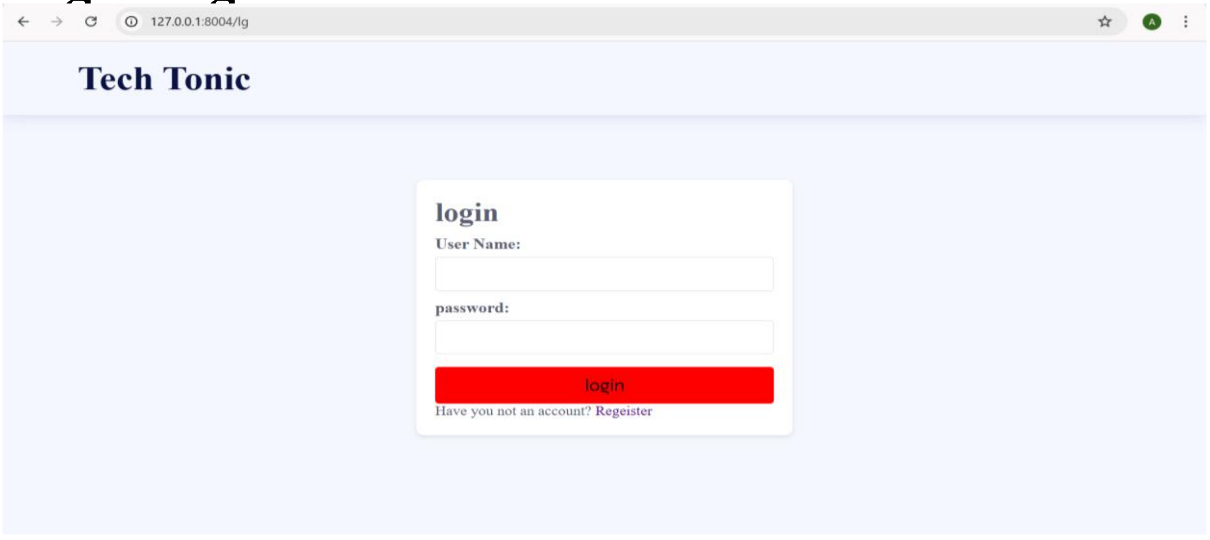
Register

Have you an account? Login

Description:

First page of website is Regestration page.As you show In Regestration page you have fill details and login.Without login you can not enter in website because it is compulsory.

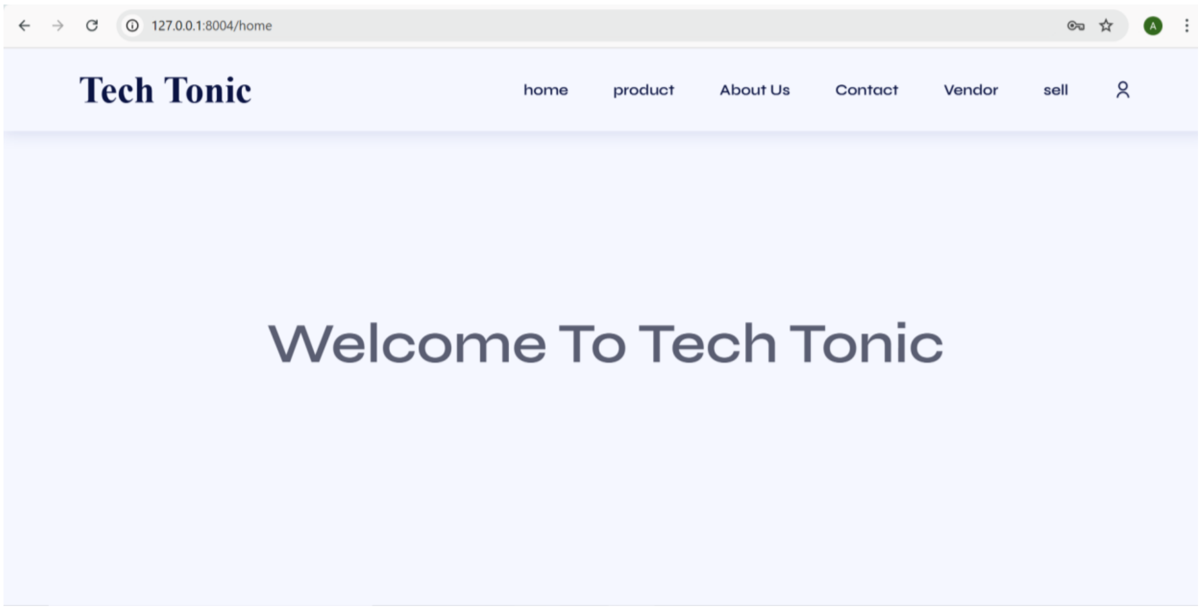
Login Page:



Description:

It is login page of our website.here,you have just enter username and password and you can login it.you must have register your account in registration form.

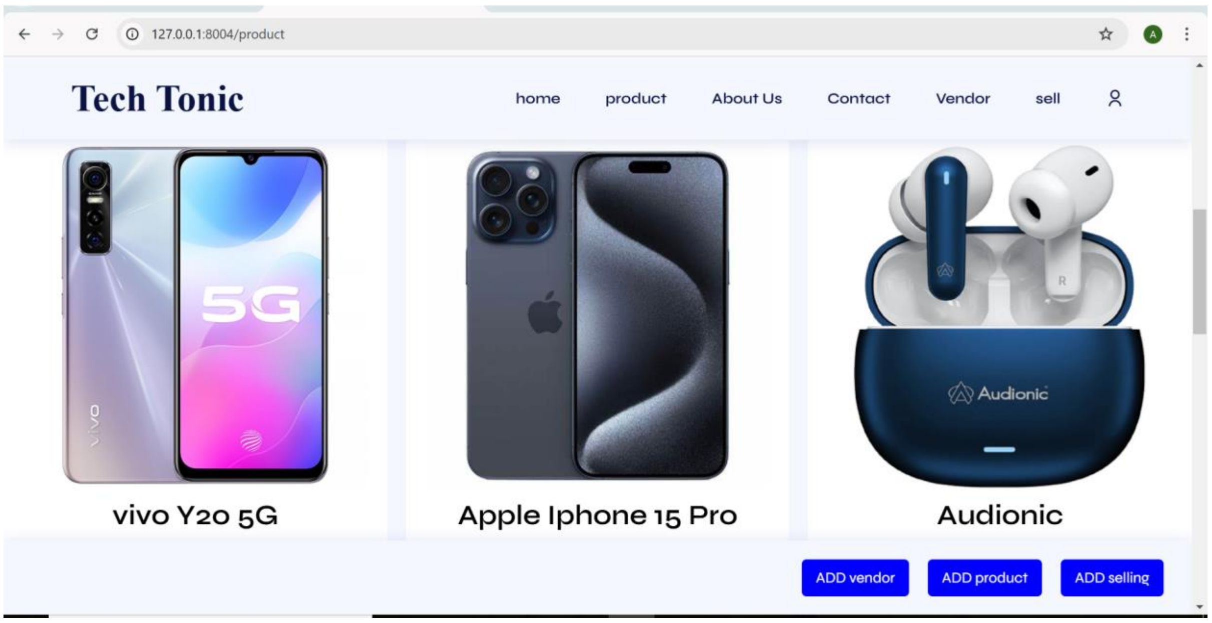
2.Home Page:



Description:

This is Home Page or main page of our website.Which contain navigation bar with different menu/pages on top.or there is one human symbol at top-right side on which click show currunt login user email or username or logout option button.

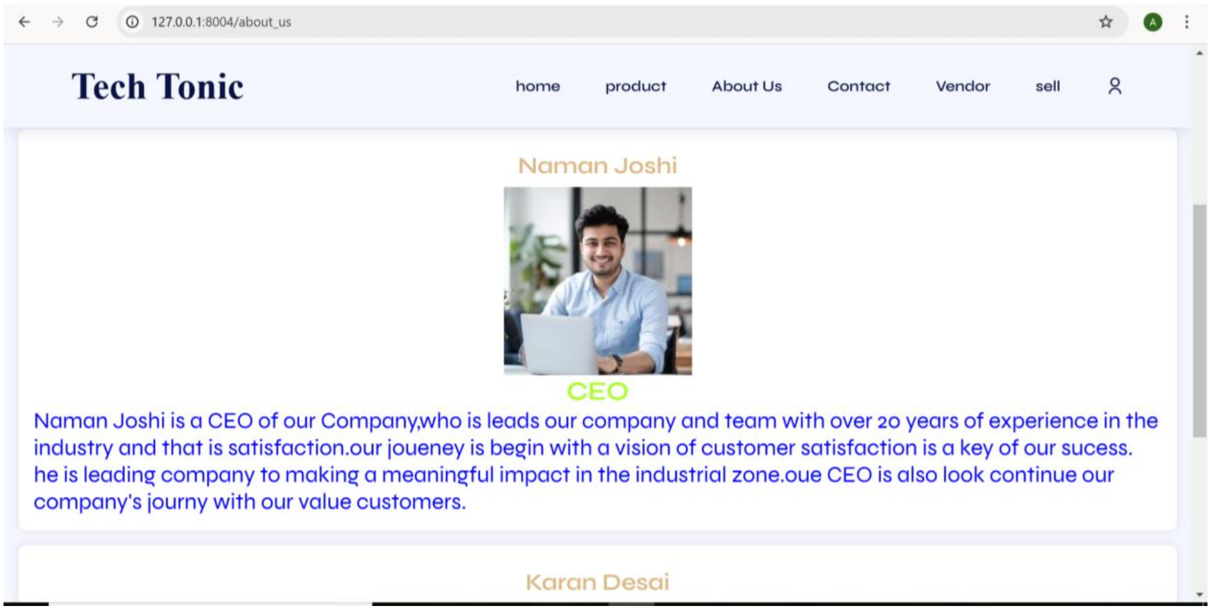
3.Product Page:



Description:

In product Page All products Image or Details are Fetch from Products table in django administration and display in this product page.so this page is contain all dynamic features there is any thing is not static.There is footer at bottom, which contain three buttons ADD vendor for add new product in our product stock data and ADD product for add new product in website,ADD selling add new selling.OnClick of these three different buttons different form are open for each different task.

4.About Us Page:


127.0.0.1:8004/about_us

Tech Tonic

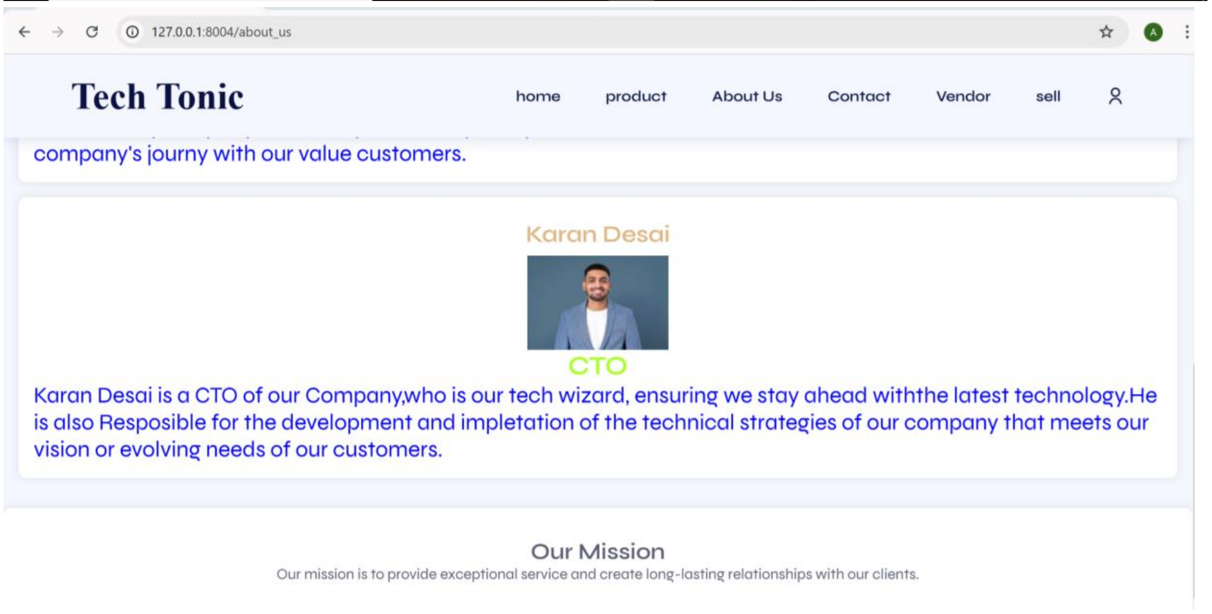
home product About Us Contact Vendor sell

Naman Joshi

CEO

Naman Joshi is a CEO of our Company,who is leads our company and team with over 20 years of experience in the industry and that is satisfaction.our joueney is begin with a vision of customer satisfaction is a key of our sucess. he is leading company to making a meaningful impact in the industrial zone.oue CEO is also look continue our company's journy with our value customers.

Karan Desai


127.0.0.1:8004/about_us

Tech Tonic

home product About Us Contact Vendor sell

company's journy with our value customers.

Karan Desai

CTO

Karan Desai is a CTO of our Company,who is our tech wizard, ensuring we stay ahead withthe latest technology.He is also Resposible for the development and impletation of the technical strategies of our company that meets our vision or evolving needs of our customers.

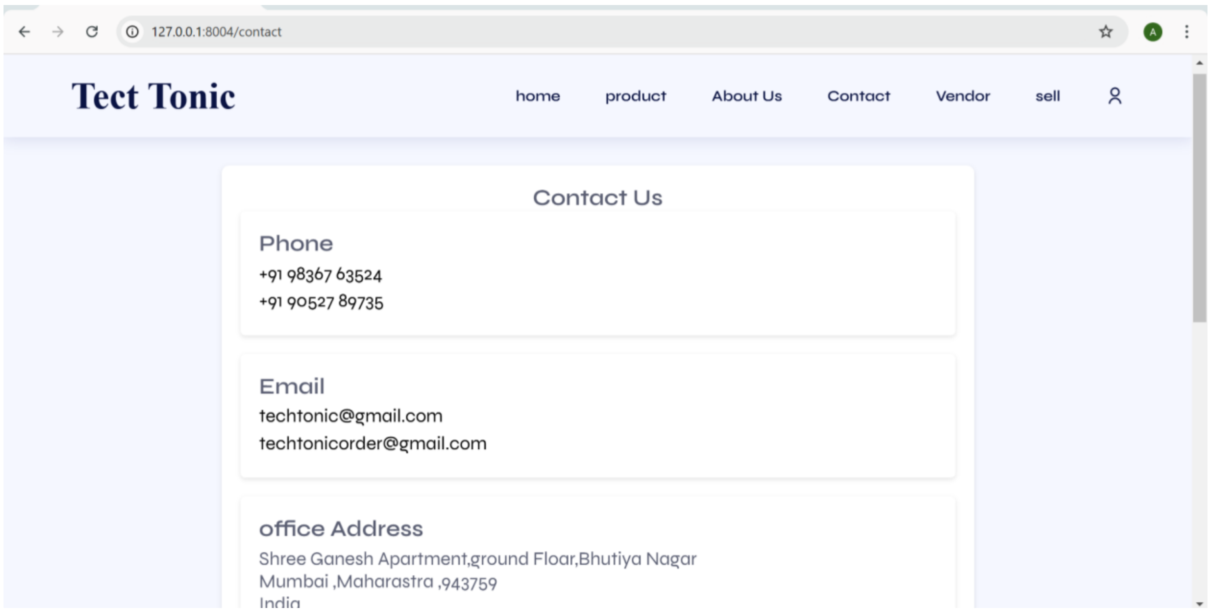
Our Mission

Our mission is to provide exceptional service and create long-lasting relationships with our clients.

Description:

In About us page there is information about our company.It is static page also.There is information about Our Company CEO,CTO or data about our Services and History of Company From Start to End.

5.Contact Us Page:



127.0.0.1:8004/contact

Tect Tonic

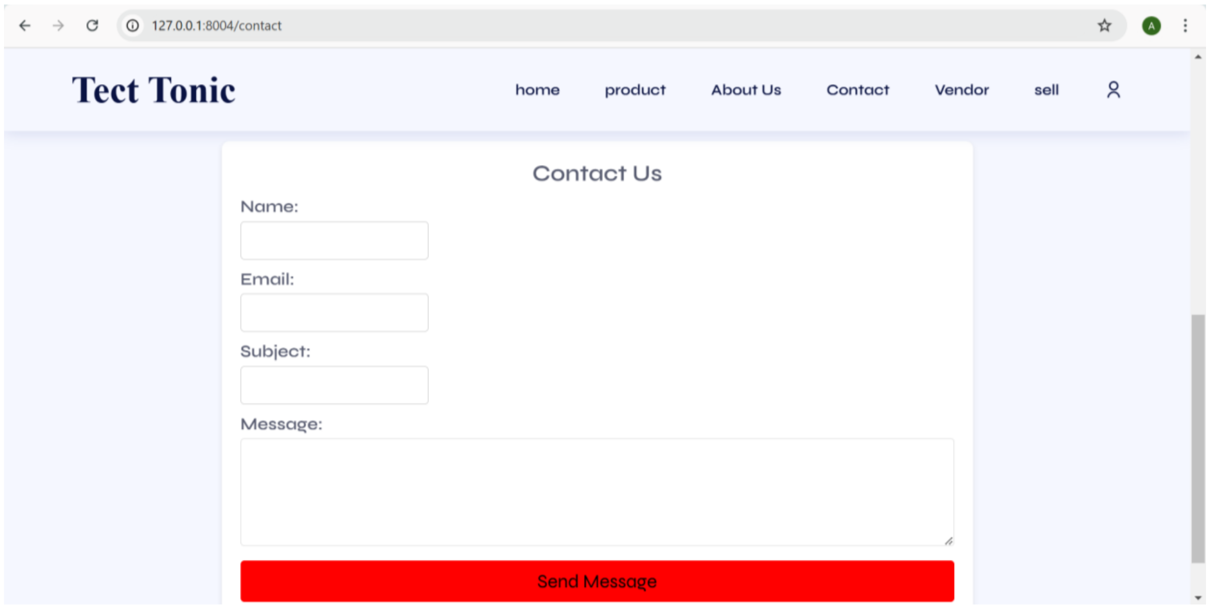
home product About Us Contact Vendor sell

Contact Us

Phone
+91 98367 63524
+91 90527 89735

Email
techtonic@gmail.com
techtonicorder@gmail.com

office Address
Shree Ganesh Apartment,ground Floor,Bhutiya Nagar
Mumbai ,Maharastra ,943759
India



127.0.0.1:8004/contact

Tect Tonic

home product About Us Contact Vendor sell

Contact Us

Name:

Email:

Subject:

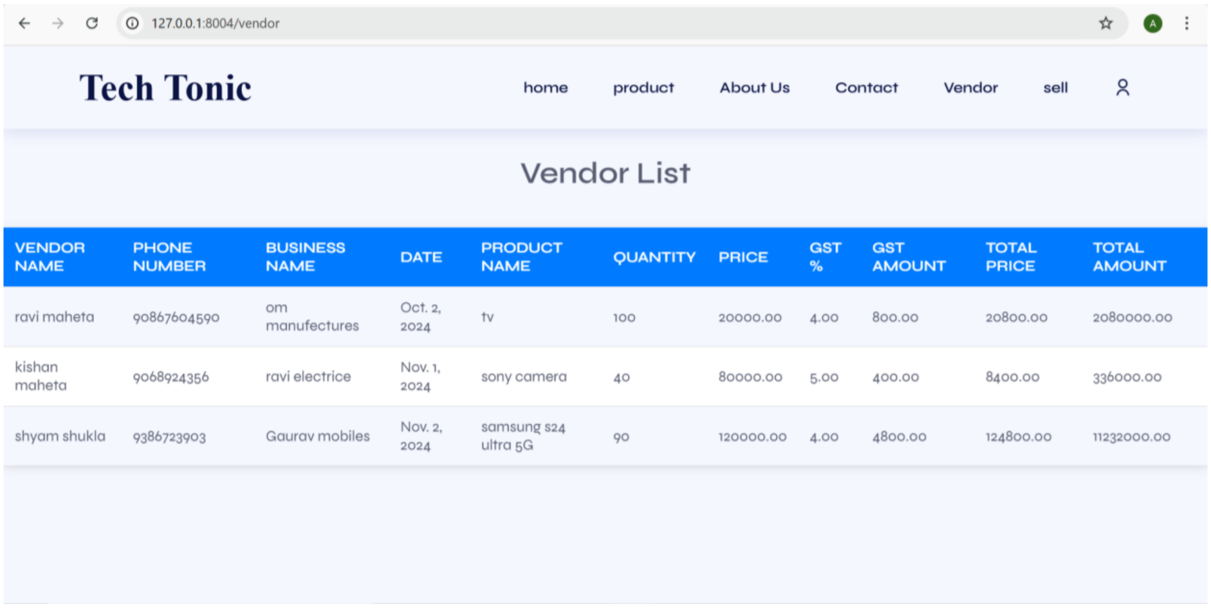
Message:

Send Message

Description:

In contact Page there is Information for Contact us through E-mail,Phone Number,Address.In contact We have One Contact Us form from which you can contact our shop or company.If you have any problem or complain or sharing your experience with US.then you can contact us by fill this form and write your message in form.

6.Vendor/Selling Page:



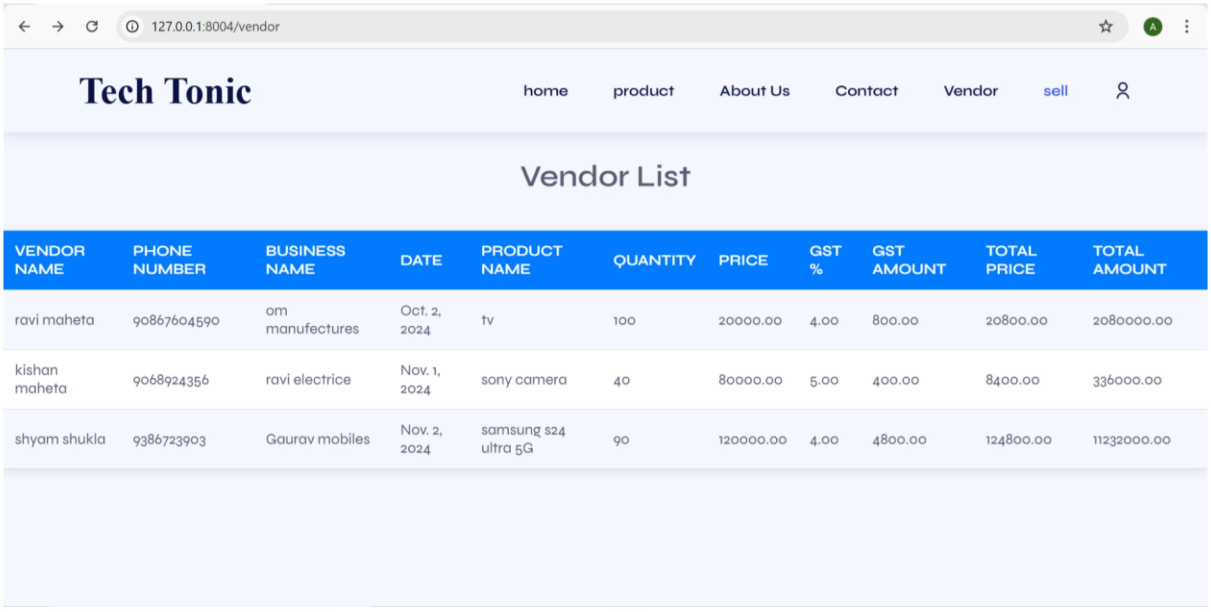
127.0.0.1:8004/vendor

Tech Tonic

home product About Us Contact Vendor sell

Vendor List

VENDOR NAME	PHONE NUMBER	BUSINESS NAME	DATE	PRODUCT NAME	QUANTITY	PRICE	GST %	GST AMOUNT	TOTAL PRICE	TOTAL AMOUNT
ravi maheta	90867604590	om manufectures	Oct. 2, 2024	tv	100	20000.00	4.00	800.00	20800.00	2080000.00
kishan maheta	9068924356	ravi electrice	Nov. 1, 2024	sony camera	40	80000.00	5.00	400.00	8400.00	336000.00
shyam shukla	9386723903	Gaurav mobiles	Nov. 2, 2024	samsung s24 ultra 5G	90	120000.00	4.00	4800.00	124800.00	11232000.00



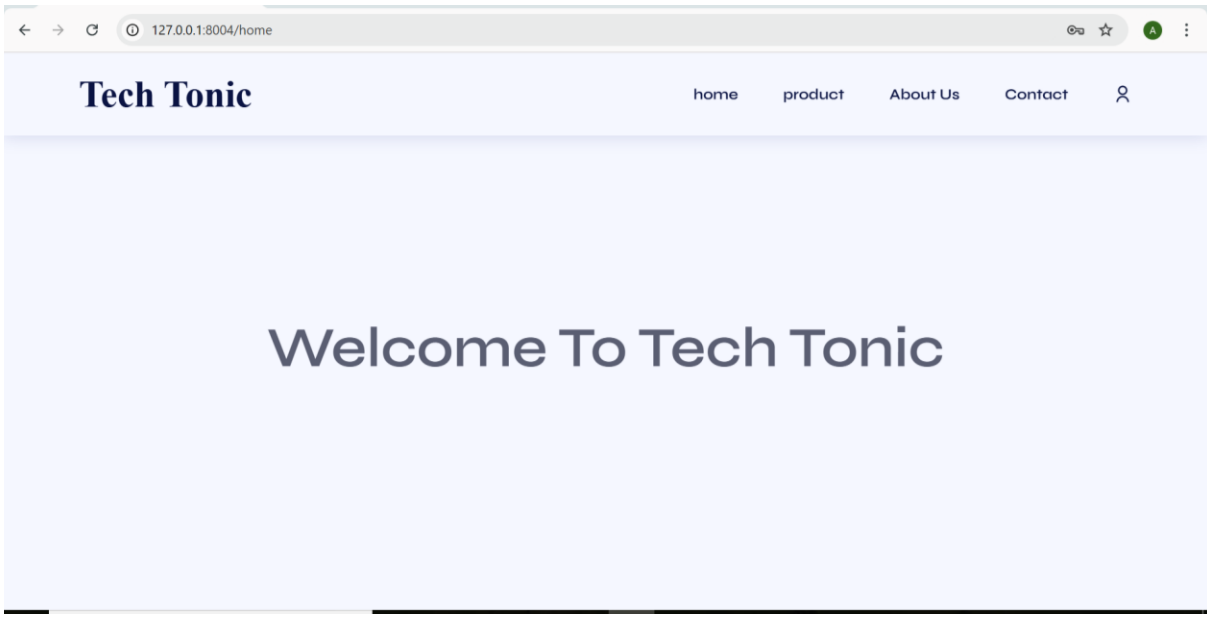
127.0.0.1:8004/vendor

Tech Tonic

home product About Us Contact Vendor sell

Vendor List

VENDOR NAME	PHONE NUMBER	BUSINESS NAME	DATE	PRODUCT NAME	QUANTITY	PRICE	GST %	GST AMOUNT	TOTAL PRICE	TOTAL AMOUNT
ravi maheta	90867604590	om manufectures	Oct. 2, 2024	tv	100	20000.00	4.00	800.00	20800.00	2080000.00
kishan maheta	9068924356	ravi electrice	Nov. 1, 2024	sony camera	40	80000.00	5.00	400.00	8400.00	336000.00
shyam shukla	9386723903	Gaurav mobiles	Nov. 2, 2024	samsung s24 ultra 5G	90	120000.00	4.00	4800.00	124800.00	11232000.00

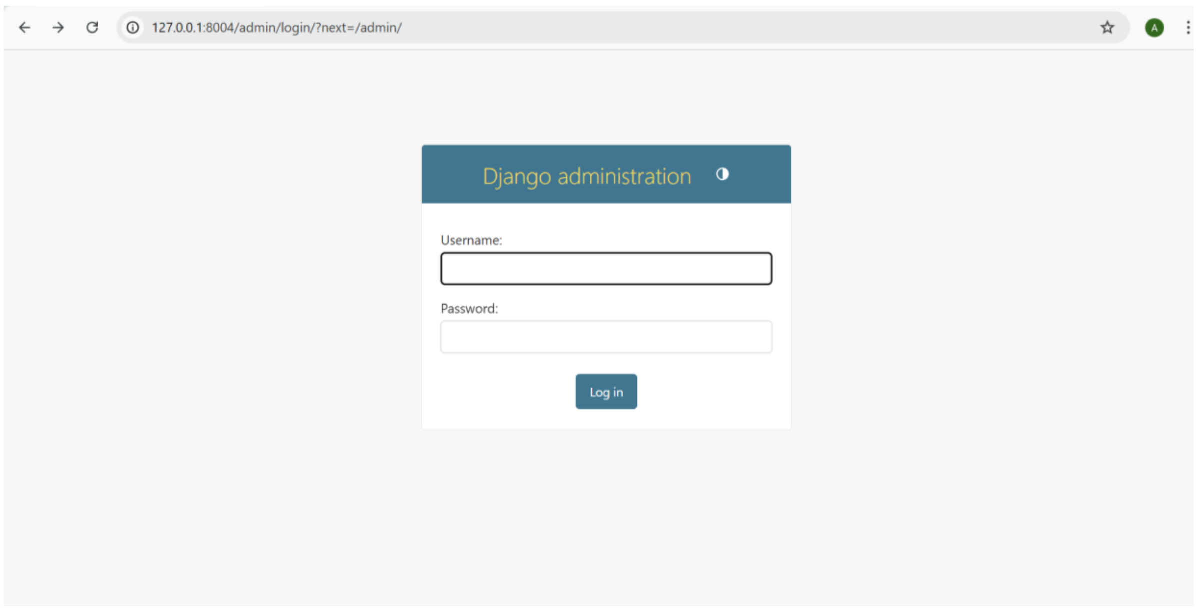


Description:

In above pictures you have see vendor from whom we have purchase products or selling page .In which only admin/owner can see the data of vendor and selling.These both page name are display in Navigationbar when admin/owner is login.In above Pictures you can see that both page are not showed because here user is Login.

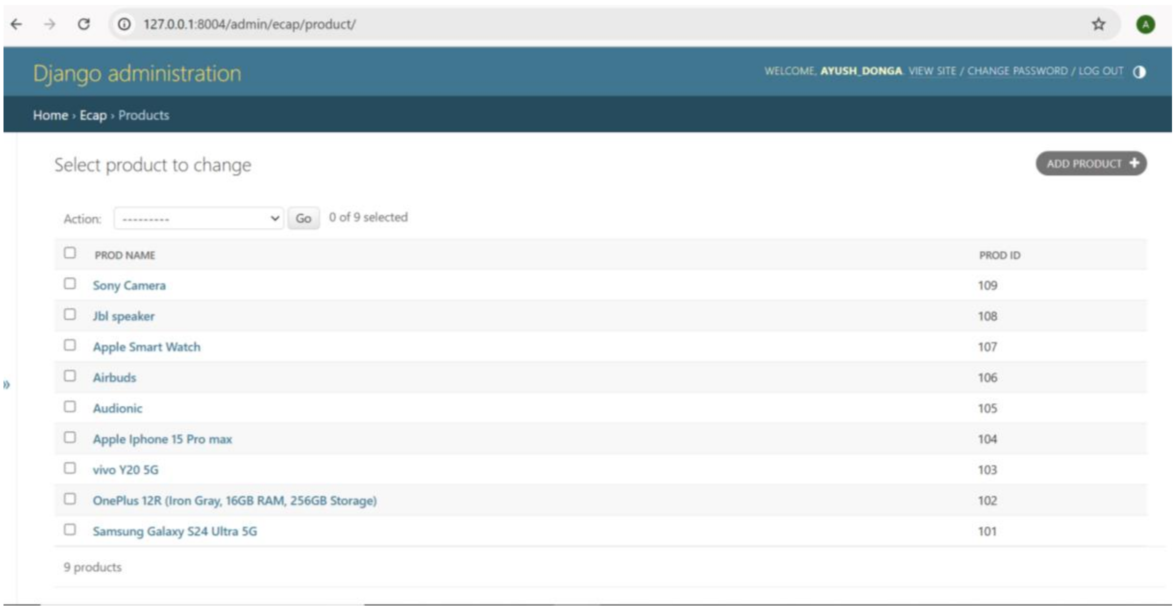
6.Database

In this project, I have successfully developed a dynamic e-commerce website using Django, a powerful and scalable web framework, which enables the creation of secure, flexible, and user-friendly applications. The website is structured around core pages such as the Home Page, About Us Page, Product Page, Contact Us Page, and Vendor Page.



Additionally, the website also features dynamic components such as the Product, Vendor, and Selling pages, which are directly linked to the database and can be managed through the Django admin interface. To achieve these dynamic functionalities, I utilized Django's robust model-view-controller (MVC) architecture. The database tables for Product, Vendor, Selling, and Contact Us have been created within Django's ORM (Object-Relational Mapping), enabling easy and efficient handling of data. These tables are accessible via the Django admin interface, allowing administrators to manage data without writing raw SQL queries. This approach streamlines database operations, minimizes errors, and makes the website highly maintainable.

Product Table:


127.0.0.1:8004/admin/ecap/product/

Django administration

WELCOME, AYUSH_DONGA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Ecap › Products

Select product to change

ADD PRODUCT

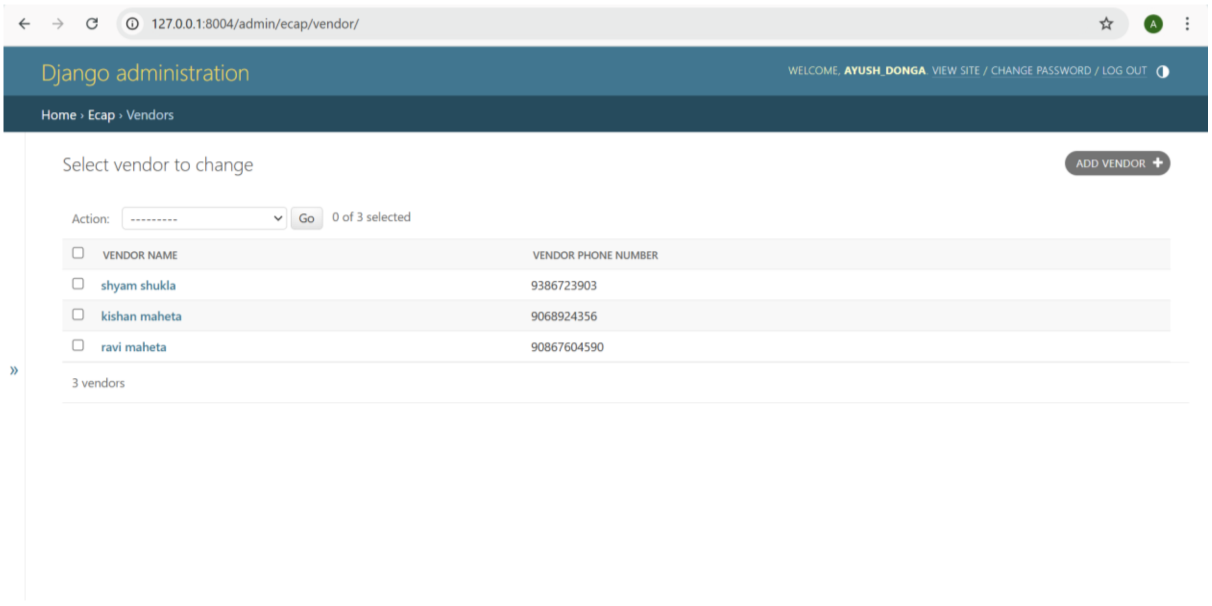
Action: --------- Go 0 of 9 selected

PROD NAME	PROD ID
Sony Camera	109
Jbl speaker	108
Apple Smart Watch	107
Airbuds	106
Audionic	105
Apple Iphone 15 Pro max	104
vivo Y20 5G	103
OnePlus 12R (Iron Gray, 16GB RAM, 256GB Storage)	102
Samsung Galaxy S24 Ultra 5G	101

9 products

The Product table is central to the website, holding information about the products available for sale. It contains fields such as product name, description, price, and stock quantity. By using Django's built-in model, I was able to easily create the necessary database structure and ensure that each product can be managed effectively through the admin panel. Admin users can add new products, update existing details, and remove outdated listings, all from a user-friendly interface.

Vendor Table:


127.0.0.1:8004/admin/ecap/vendor/

Django administration

WELCOME, AYUSH_DONGA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Ecap › Vendors

Select vendor to change

ADD VENDOR

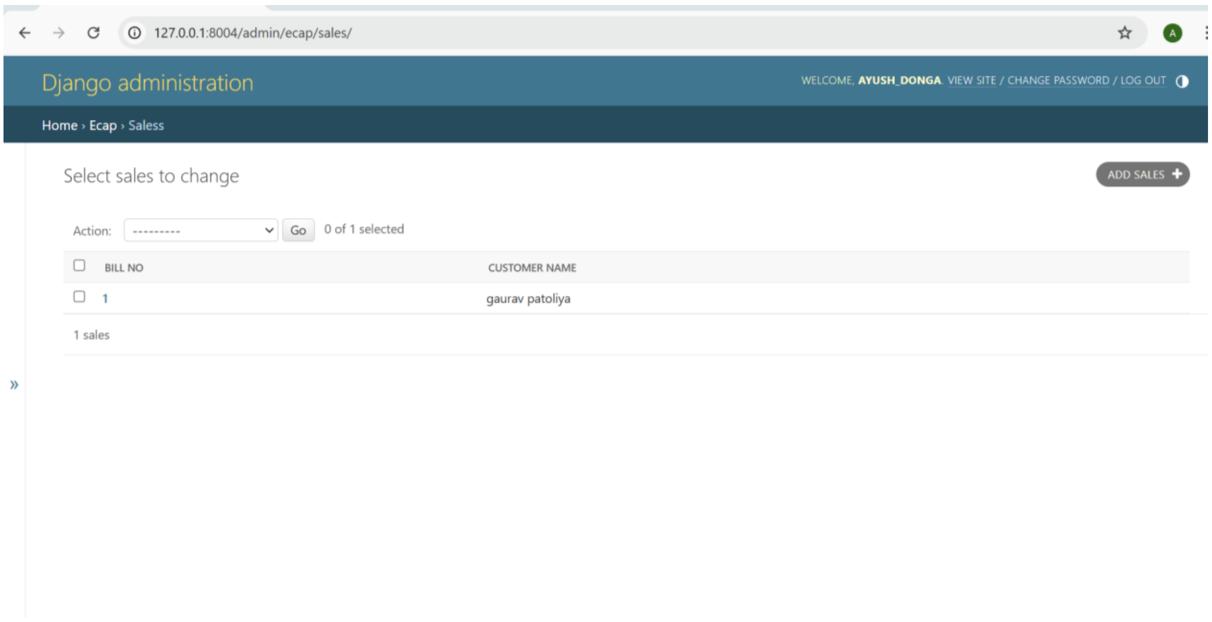
Action: --------- Go 0 of 3 selected

VENDOR NAME	VENDOR PHONE NUMBER
shyam shukla	9386723903
kishan maheta	9068924356
ravi maheta	90867604590

3 vendors

The Vendor table is another key dynamic component, as it holds data related to the vendors who supply the products. This table is essential for tracking which vendors are responsible for specific products, making it easier for site administrators to monitor and manage relationships with suppliers. The vendor data can also be easily modified or expanded from the Django admin interface.

Selling Table:



127.0.0.1:8004/admin/ecap/sales/

Django administration

WELCOME, **AYUSH_DONGA**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Ecap › Saless

Select sales to change

ADD SALES

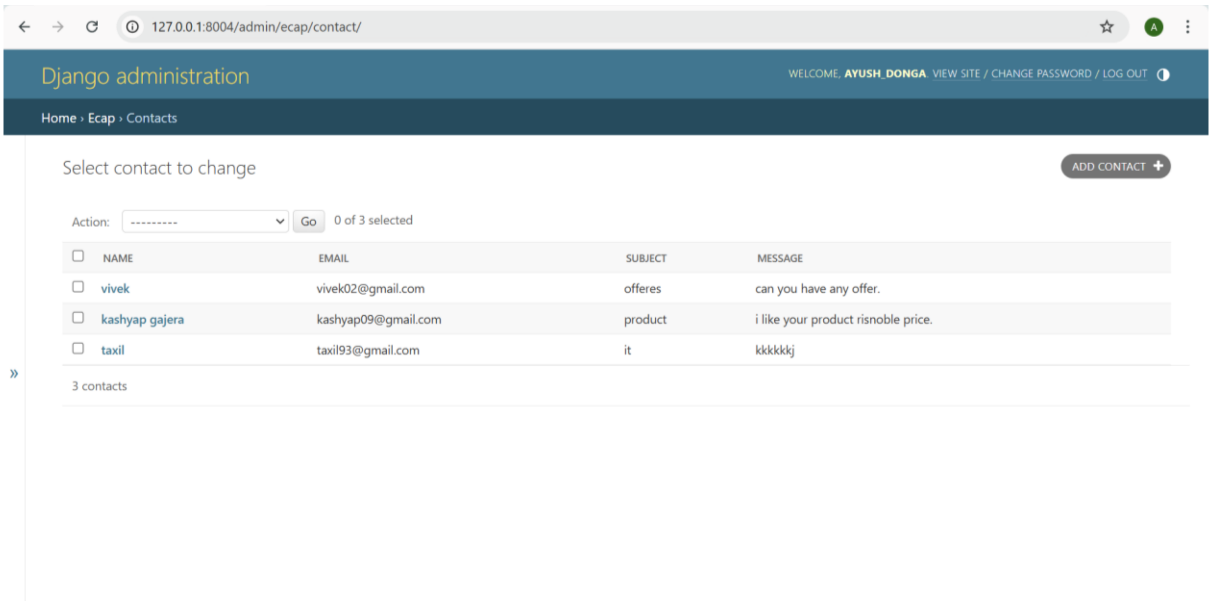
Action: --------- Go 0 of 1 selected

BILL NO	CUSTOMER NAME
1	gaurav patoliya

1 sales

The Selling table links products to vendors and tracks product sales. This table is crucial for understanding the commercial activities on the site. It allows the site’s admin to manage and monitor product sales, ensuring that inventory and financial records are up-to-date. By linking products and vendors in the Selling table, the website can dynamically display product availability and vendor-related information, giving users a complete view of the marketplace.

Contact Us Table:


127.0.0.1:8004/admin/ecap/contact/
Django administration
WELCOME, AYUSH_DONGA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Ecap › Contacts
Select contact to change
ADD CONTACT
Action: --------- Go 0 of 3 selected

NAME	EMAIL	SUBJECT	MESSAGE
vivek	vivek02@gmail.com	offeres	can you have any offer.
kashyap gajera	kashyap09@gmail.com	product	i like your product risnoble price.
taxil	taxil93@gmail.com	it	kkkkkkj

3 contacts

The Contact Us page provides users with a form to reach out to the website's administrators. This table stores all submitted messages, making it easy for the site's team to review inquiries, complaints, or feedback. The dynamic nature of this table ensures that the website can continually receive and manage user communications effectively.

Security and User Management:

In addition to the dynamic pages, the website incorporates user registration and login features. Django's built-in authentication system was utilized to ensure secure login and registration processes. This system also supports user roles and permissions, allowing different levels of access to different parts of the website, particularly the Django admin interface, where sensitive operations can be performed.

Final Thoughts:

Overall, this e-commerce website project demonstrates a robust, secure, and scalable solution for creating an online marketplace. Django's versatility and ease of use allowed for the integration of both static and dynamic elements, ensuring a smooth and engaging user experience. The use of Django's admin panel to manage key aspects of the site—such as products, vendors, sales, and customer inquiries—has streamlined .the website is poised for future scalability and can be easily extended with additional features as needed.

7.Limitations of Project

- Only Product Page is dynamic. While other page.such as About Us,Contact.
- Basic Security is implemented Like in django administration for show data of webiste.
- In Website there is no feature or option of search products or forgot password.
- It has not advance features or it is not ready to used.so it is just made for learning or knowing of python and Django Framework technology.
- The user interface is simple but may lack advanced features or customizations for different user preferences.
- The platform has basic product filtering but lacks advanced search and recommendation features.
- The reliance on third-party payment processors could introduce security vulnerabilities.
- Vendors can manage their products but have limited customization options for their storefront. Enable real-time inventory updates to ensure accurate stock levels and prevent overselling.
- The platform supports third-party payment gateways but may have limited integration options.
- The sales and vendor dashboards provide basic data, lacking advanced analytics and reporting features.
- Product availability might not update in real time if there are discrepancies in stock levels.
- The platform has basic product filtering but lacks advanced search and recommendation features.
- The design may not be fully optimized for mobile devices.
- Vendors depend on the admin panel for managing products, potentially limiting flexibility for larger stores.

8.Future Scope of Project

- Improve the security in website or database and make changes in database to handle large data in Future.
- We have thinking about add features of email on each order to each user and ADD TO CART features.
- We will add forgot password feature and make all pages dynamic in Future.
- Now,In Website there is no feature or option of search products But we will have try in future for add this feature in website.
- Add support for multiple currencies and languages to cater to international customers and vendors.
- Implement a more powerful search engine with options like voice search, product comparison, and dynamic filtering by multiple criteria.
- Enhance the design with modern UI/UX principles to make the platform more visually appealing and user-friendly.
- Develop a mobile application for both iOS and Android to provide users with a more convenient and optimized shopping experience.
- Integrate AI-based recommendation systems that suggest products to users based on their browsing and purchase history.
- Enable real-time inventory updates to ensure accurate stock levels and prevent overselling.
- Add AR features that allow customers to virtually try out products (e.g., visualizing electronics in their environment).
- Implement subscription plans or loyalty rewards for customers to incentivize repeat purchases and boost customer retention.
- Integrate additional secure payment gateways and offer faster, smoother transaction processing with one-click checkout options.
- Provide vendors with advanced inventory management, sales analytics, and marketing tools to optimize their performance on the platform.

9. Bibliography of Project

BOOKS :

- Django For Beginners BY-William S. Vincent
- Django Unleashed BY-Andrew Pinkham
- HTML5 and CSS3 [Visual QuickStart Guide] BY-Bruce Hyslop

ONLINE REFERENCE :

- https://WWW.codewithharry.com
- http://WWW.codewithmosh.com
- http://WWW.mygreatlearning.com
- https://WWW.codewithharry.com/@CodeWithHarry

10.Conculison

The development of this e-commerce website represents the culmination of a significant effort to create an efficient, user-friendly online shopping platform. The project integrates both static and dynamic components to deliver a seamless experience for customers, vendors, and administrators. By utilizing Django, a powerful and flexible Python framework, we were able to structure the application in a way that efficiently manages both content and user interactions.

The website features several essential sections, including a **Home Page**, **About Us Page**, **Product Page**, **Contact Us Page**, **Vendor Page**, and **Sell All Page**, which cater to various needs of the users. The static pages, such as the About Us and Contact Us pages, serve as informative sections that provide basic information about the website and the business.

The **Product Page** is particularly crucial to the success of an e-commerce platform, as it showcases the various items available for purchase. The product information is stored in a database, and with the use of Django models, we can easily add, edit, or delete products without needing to change the front-end code. This system ensures that as inventory changes, product details are automatically updated across the website, reducing the chances of errors or outdated information being presented to customers.

Another important dynamic feature is the **Vendor Page**, where sellers can register and manage their products. Vendors can add their products, update their inventory, and manage their sales performance. This feature enables a multi-vendor marketplace, offering customers a wide variety of products from different sources while providing vendors with a platform to sell their goods easily.

The website's **Login and Registration** system is a core feature that ensures users can create accounts, manage their profiles, and securely track their orders. The implementation of this feature is crucial for customer convenience and the personalization of the shopping experience.

In conclusion, this e-commerce website project demonstrates a comprehensive understanding of both front-end and back-end development using Django. It showcases the integration of static and dynamic pages, ensuring a user-friendly and efficient platform for customers, vendors, and administrators. The system is designed to be scalable, allowing for the addition of new features or products as the business grows business