An
Internship Report

On

Web Development using Python Django

Conducted during summer break after IV semester
and for assessment during V semester

Submitted By

Name -: Neeraj Kumar Kannoujiya
Roll no -: 2007350130031
Branch -: InformationTechnology
Year -: 3rd Year

RAJKIYA ENGINEERING COLLEGE BIJNOR
Department of Information Technology

(Affiliated to AKTU Lucknow) Chandpur-246725

ACKNOWLEDGEMENT

With great pleasure, I would like to extend my sincere gratitude and thanks to my parents for their sincere support they have given me to go for internship.

I would also like to thanks to my friend Divyansh Pandey and Sandeep Kumar for the care and support they gave me all thought my internship.

I also would like to thank my trainer MR.Suneel Kumar for the good corporation they had with me during the training period.

I wish to express my sincere thanks to the all department faculties and staff members for their support. I would also like to thank my all classmates for their valuable guidance and helpful discussion.

Neeraj Kumar Kannoujiya

ABSTRACT

This report discusses the Internship program at Thinknext Private Limited Company At Mohali chandigarh during the period from 14th March 2022 to 28th April 2022.

The Company :

Thinknext Private Limited is an ISO 9001:2015 certified Software, Electronics and CAD/CAM Systems Development Company (MNC) and approved from Ministry of Corporate Affairs which deals in **Web Designing and Development, Mobile Apps Development,Cloud Telephony Services,** etc. We have expertise in latest technologies e.g. Smart Card (Contact Type and Contactless), NFC, Biometrics, Barcode, RFID, SMS, Voice SMS Android, iPhone, Web, Windows and Mobile based technologies.

ThinkNEXT offers 6 Months/3 Months/ 6 Weeks/45 days/Summer Industrial Training programs for B.Tech Engineering students, MCA, BCA, Polytechnic Diploma, M.Sc (IT), B.Sc (IT), MBA, BBA, B.Com students and job-seekers.

Methodology:

This project deals with developing a Virtual website 'E-commerce Website'. It provides the user with a list of various products available for purchase in the store. For the convenience of online shopping, a shopping cart is provided to the user. After the selection of the goods, it is sent for the order confirmation process. The system is implemented using Python's web framework Django. To develop an e-commerce website, it is necessary to study and understand many technologies.

Technology Used:

- Django
- SQLite
- HTML
- CSS
- JavaScript
- Bootstrap

CERTIFICATE

Table of Contents

Introduction of Training

The purpose of industrial Training is to provide exposure for the students on practical engineering fields. Through this exposure, students will have better understanding of engineering practical in general and sense of frequent and possible problems. This training is part of learning process. So the exposure that uplifts the knowledge and experience of a student needs to be properly documented in the form of a report. Through this report, the experience gained can be delivered to their peers. A properly prepared report can facilitate the presentation of the practical experience in an orderly, precise and interesting manner. I have chosen Python Django as my training because it will help me in several ways. I have chosen my languages course in web development and by having little bit knowledge in this Subject will help me in carrier growth. During learning this I came to understand much more concepts of other language.

Learning Outcomes from Training

General learning outcomes:

- An understanding of the principles and practice of python in the construction of software's.
- An awareness of the need for a professional approach to design and the importance of good documentation to the finished programs.
- A competence to design, write, compile, test and execute straightforward programs using a high level language.

Specific learning outcomes:

- Be able to implement, compile, test and run programs.
- Understand how to include our logic in a program.

Module 1

Django Introduction

Django language is the most popular Python Framework out there. It is open-source and free to use. It also follows Model-View-Controller (MVC) architecture which is now the de facto architecture utilized in the development of web apps. Django shines in its nonmodular architecture. It can help the easy development of database-drive websites which are complex in nature. Moreover, it is ready for reusability, and a pluggable environment can enable developers to do rapid development. Just like any other web framework, it offers basic CRUD operation and a simple-to-use admin panel for easy administration. The whole philosophy behind Django is rapid development and making complex projects work for the developer.
This framework uses a famous tag line: "**The web framework for perfectionists with deadlines"**.

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 team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was opensourced 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).

Popularity

Django is widely accepted and used by various well-known sites such as:

- Instagram
- Mozilla
- Disqus
- Pinterest
- Bitbucket
- The Washington Times

Features of Django

Django Installation

To install Django, first visit to django official site (https://www.djangoproject.com) and download django by clicking on the download section. Here, we will see various options to download The Django. Django requires pip to start installation. Pip is a package manager system which is used to install and manage packages written in python. For Python 3.4 and higher versions pip3 is used to manage packages. the installation command is given below.

pip install django

Django Project

To create a Django project, we can use the following command. projectname is the name of Django application.

django-admin startproject projectname

A Django project contains the following packages and files. The outer directory is just a container for the application. We can rename it further.

- **manage.py**: It is a command-line utility which allows us to interact with the project in various ways and also used to manage an application that we will see later on in this tutorial.
- A directory (djanoapp) located inside, is the actual application package name. Its name is the Python package name which we'll need to use to import module inside the application.
- **init_.py**: It is an empty file that tells to the Python that this directory should be considered as a Python package.
- settings.py: This file is used to configure application settings such as database connection, static files linking etc.
- **urls.py**: This file contains the listed URLs of the application. In this file, we can mention the URLs and corresponding actions to perform the task and display the view.
- **wsgi.py**: It is an entry-point for WSGI-compatible web servers to serve Django project. Initially, this project is a default draft which contains all the required files and folders

Running the Django Project

Django project has a built-in development server which is used to run application instantly without any external web server. It means we don't need of Apache or another web server to run the application in development mode. To run the application, we can use the following command.
python manage.py runserver

It prompts for login credentials if no password is created yet, use the following command to create a user.

Creating an admin User

Python manage.py createsuperuser

Django App

Django application consists of project and app, it also generates an automatic base directory for the app, so we can focus on writing code (business logic) rather than creating app directories.

The difference between a project and app is, a project is a collection of configuration files and apps whereas the app is a web application which is written to perform business logic.

Creating an App

To create an app, we can use the following command.
python manage.py startapp appname

Django App Example

python manage.py startapp GreatKart

MODULE 2

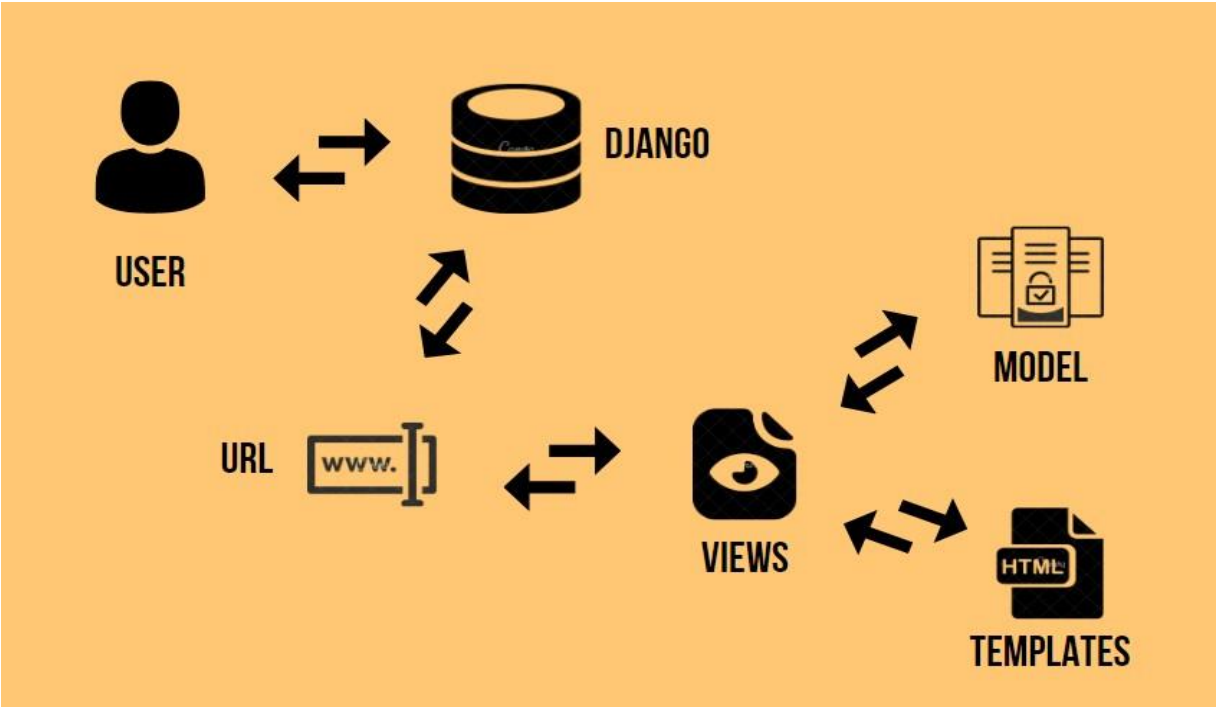
Django MVT

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data.

The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.

Although Django follows MVC pattern but maintains it?s own conventions. So, control is handled by the framework itself.

There is no separate controller and complete application is based on Model View and Template. That?s why it is called MVT application.
See the following graph that shows the MVT based control flow.



Here, a user requests for a resource to the Django, Django works as a controller and check to the available resource in URL.
If URL maps, a view is called that interact with model and template, it renders a template. Django responds back to the user and sends a template as a response.

Django Model

In Django, a model is a class which is used to contain essential fields and methods. Each model class maps to a single table in the database.
Django Model is a subclass of django.db.models.Model and each field of the model class represents a database field (column).
Django provides us a database-abstraction API which allows us to create, retrieve, update and delete a record from the mapped table.
Model is defined in Models.py file. This file can contain multiple models. Let's see an example here, we are creating a model Employee which has two fields first_name and last_name.

```
1. from django.db import models
2.
3. class Employee(models.Model):
4.     first_name = models.CharField(max_length=30)
5.     last_name = models.CharField(max_length=30)
```

The first_name and last_name fields are specified as class attributes and each attribute maps to a database column.

Django Views

A view is a place where we put our business logic of the application. The view is a python function which is used to perform some business logic and return a response to the user. This response can be the HTML contents of a Web page, or a redirect, or a 404 error.

All the view function are created inside the views.py file of the Django app.

//views.py

```
1. import datetime
2. # Create your views here.
3. from django.http import HttpResponse
4. def index(request):
5.     now = datetime.datetime.now()
6.     html = "<html><body><h3>Now time is %s.</h3></body></html>" % now
7.     return HttpResponse(html)    # rendering the template in HttpResponse
```

Django Templates

A Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.

A template is rendered with a context. Rendering replaces variables with their values, which are looked up in the context, and executes tags. Everything else is output as is.

The syntax of the Django template language involves four constructs.

Variable:

A variable outputs a value from the context, which is a dict-like object mapping keys to values.

Variables are surrounded by **{{** and **}}** like this:

My first name is **{{ first_name }}.** My last name is **{{ last_name }}.**
With a context of {'first_name': 'Neeraj', 'last_name': 'Kannoujiya'}, this template renders to:

My first name is Neeraj. My last name is Kannoujiya.

Settings.py

Setting.py is a core file in Django projects. It holds all the configuration values that your web app needs to work; database settings, logging configuration, where to find static files, API keys if you work with external APIs, and a bunch of other stuff.

But once you start setting up your Django app on multiple environments; like production, testing, and staging — and on machines for new developers, you are likely to run into a pain point; managing the configuration across the different environments.

```
ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'category',
  'accounts',
  'store',
  'carts',
  'orders',
  'admin_honeypot',
]

MIDDLEWARE = [
  'django.middleware.security.SecurityMiddleware',
  'django.contrib.sessions.middleware.SessionMiddleware',
  'django.middleware.common.CommonMiddleware',
  'django.middleware.csrf.CsrfViewMiddleware',
]

SESSION_EXPIRE_SECONDS = 3600  # 1 hour
SESSION_EXPIRE_AFTER_LAST_ACTIVITY = True
SESSION_TIMEOUT_REDIRECT = 'accounts/login'
```

Module 3

Django Forms

In HTML, a form is a collection of elements inside <form>...</form> that allow a visitor to do things like enter text, select options, manipulate objects or controls, and so on, and then send that information back to the server.

Some of these form interface elements - text input or checkboxes - are built into HTML itself. Others are much more complex; an interface that pops up a date picker or allows you to move a slider or manipulate controls will typically use JavaScript and CSS as well as HTML form <input> elements to achieve these effects.

As well as its <input> elements, a form must specify two things:

It also tells the browser that the form data should be sent to the URL specified in the <form>'s action attribute - /admin/ - and that it should be sent using the HTTP mechanism specified by the method attribute - post.

When the <input type="submit" value="Log in"> element is triggered, the data is returned to /admin/.

```
from django import forms
from .models import Account, UserProfile
from .models import ReviewRating


class UserForm(forms.ModelForm):
    class Meta:
        model = Account
        fields = ('first_name', 'last_name', 'phone_number')

class ReviewForm(forms.ModelForm):
    class Meta:
        model = ReviewRating
        fields = ['subject', 'review','rating']
```

Django Form Validation

Form validation happens when the data is cleaned. If you want to customize this process, there are various places to make changes, each one serving a different purpose. Three types of cleaning methods are run during form processing. These are normally executed when you call the is_valid() method on a form. There are other things that can also trigger cleaning and validation (accessing the errors attribute or calling full_clean() directly), but normally they won't be needed.

```
from django import forms

from django.core.validators import validate_email


class MultiEmailField(forms.Field):

def to_python(self, value):

"""Normalize data to a list of strings."""

# Return an empty list if no input was given.

if not value:

return []

return value.split(',')


def validate(self, value):

"""Check if value consists only of valid emails."""

# Use the parent's handling of required fields, etc.

super().validate(value)

for email in value:

validate_email(email)
```

Django Database Connectivity

The settings.py file contains all the project settings along with database connection details. By default, Django works with SQLite, database and allows configuring for other databases as well.

Database connectivity requires all the connection details such as database name, user credentials, hostname drive name etc.

To connect with MySQL, django.db.backends.mysql driver is used to establishing a connection between application and database. Let's see an example.

we need to provide all connection details in the settings file. The settings.py file of our project contains the following code for the database.

```
1.  DATABASES = {
2.  'default': {
3.  'ENGINE': 'django.db.backends.mysql',
4.  'NAME': 'djangoApp',
5.  'USER':'root',
6.  'PASSWORD':'neeraj123',
7.  'HOST':'localhost',
8.  'PORT':'3306'
9.  }
10. }
```

after providing details, check the connection using the migrate command.

1. **python manage.py migrate**

This command will create tables for admin, auth, contenttypes, and sessions.

MODULE 4

Django Database Migrations

Migration is a way of applying changes that we have made to a model, into the database schema. Django creates a migration file inside the migration folder for each model to create the table schema, and each table is mapped to the model of which migration is created.
Django provides the various commands that are used to perform migration related tasks. After creating a model, we can use these commands.

- **makemigrations** : It is used to create a migration file that contains code for the tabled schema of a model
- **migrate** : It creates table according to the schema defined in the migration file.
- **sqlmigrate** : It is used to show a raw SQL query of the applied migration.
- **showmigrations** : It lists out all the migrations and their status. Suppose, we have a model as given below and contains the following attributes.

Model

//models.py

```
1. from django.db import models
2.
3. class Employee(models.Model):
4.     first_name = models.CharField(max_length=30)
5.     last_name = models.CharField(max_length=30)
```

To create a migration for this model, use the following command. It will create a migration file inside the migration folder.
python manage.py makemigrations

Migrations

Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema. They're designed to be mostly automatic, but you'll need to know when to make migrations, when to run them, and the common problems you might run into.After creating a migration, migrate it so that it reflects the database permanently. The migrate command is given below.

python manage.py migrate

```
from django.db import migrations, models

class Migration(migrations.Migration):

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


operations = [

migrations.DeleteModel('Tribble'),

migrations.AddField('Author', 'rating', models.IntegerField(default=0)),

]
```

Django Request and Response

The client-server architecture includes two major components request and response. The Django framework uses client-server architecture to implement web applications. When a client requests for a resource, a HttpRequest object is created and correspond view function is called that returns HttpResponse object.
To handle request and response, Django provides HttpRequest and HttpResponse classes. Each class has it's own attributes and methods.

Django HttpRequest

This class is defined in the django.http module and used to handle the client request. Django HttpRequest Example

// views.py

```
from django.shortcuts import render

from django.http import HttpResponse

def index(request):

return HttpResponse("Hello world!")
```

// urls.py

```
from django.urls import path

from . import views

urlpatterns = [

path('', views.index, name='index'),

]
```

Django HttpResponse

This class is a part of django.http module. It is responsible for generating response corresponds to the request and back to the client.

// views.py

```
from django.http import HttpResponse

import datetime

def current_datetime(request):

return HttpResponseNotFound('<h1>Page not found</h1>')
```

Project

GREATKART

Objective

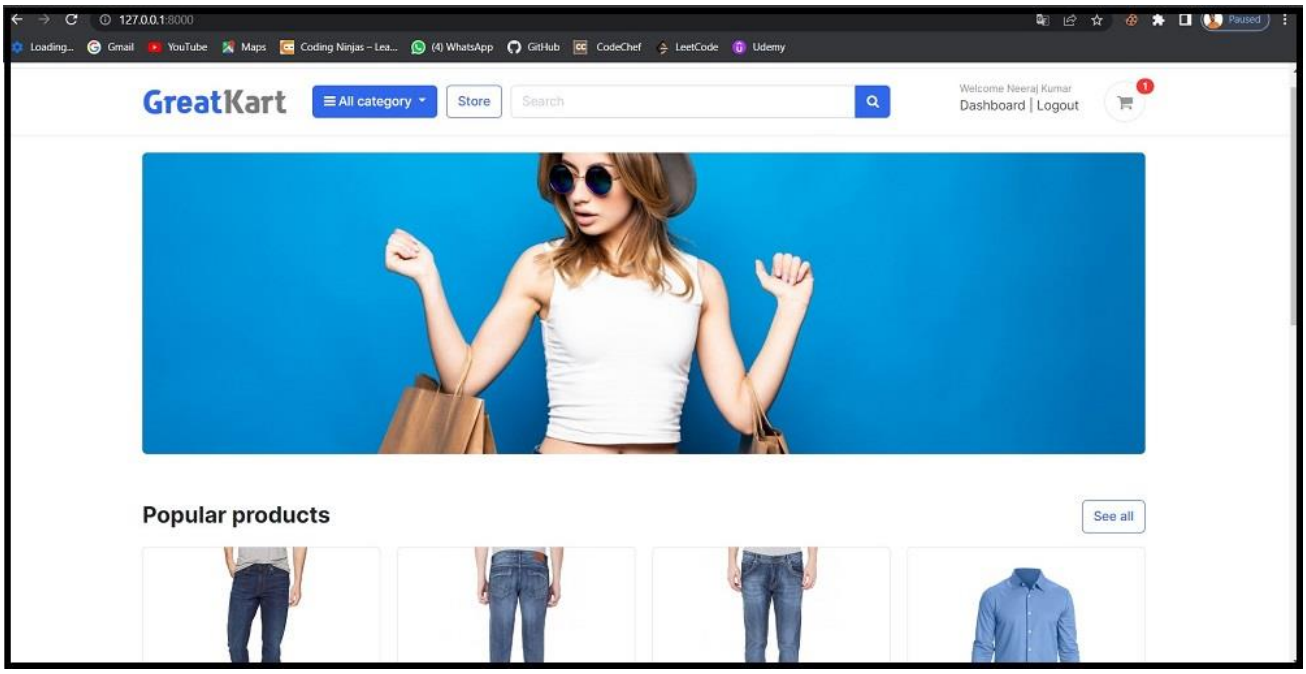
- The main objective of E-Store is that to provide the all type of Items available easily at your doorstep taking care of everyone's need, Interest & budget.
- It provides reliability & security as shopping made Convenient to customers 24X7 access.
- It overcomes Geographical locations around global regions. It provides 360 degrees view of products.

Goal of The Project

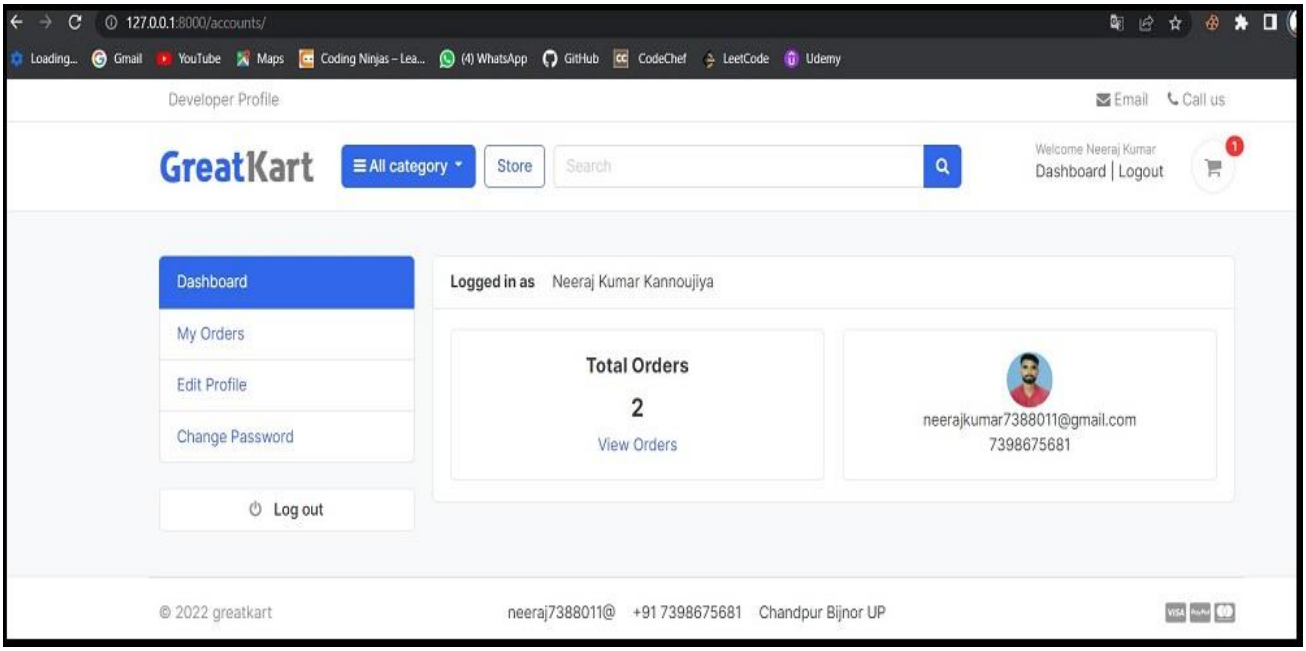
- To develop an easy to use web based interface where users can search for products, view a complete description of the products and order the products.
- A search engine that provides an easy and convenient way to search for products specific to their needs. The search engine would list a set of products based on the search term and the user can further filter the list based on various parameters.
- A user can view the complete specification of the product along with various images and also view the customer reviews of the product.
- Drag and Drop feature which would allow the users to add a product to or remove a product from the shopping cart by dragging the product in to the shopping cart or out of the shopping cart.

Screenshots of Project

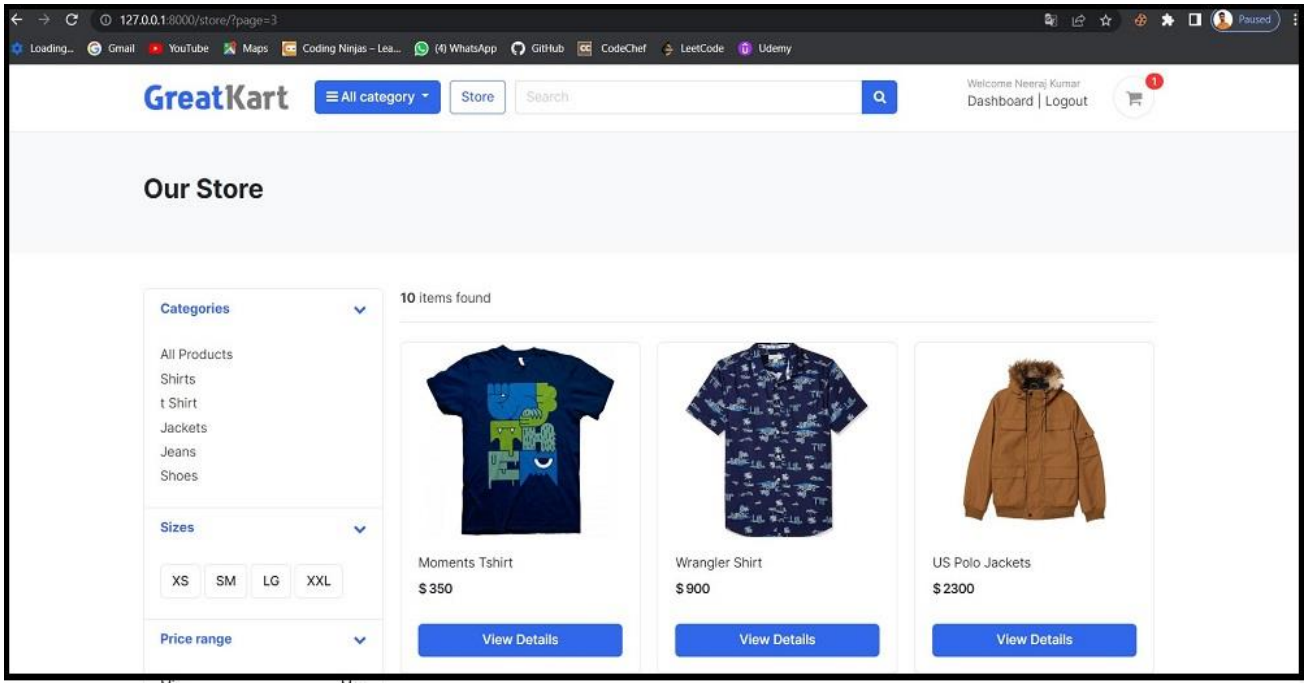
Home page



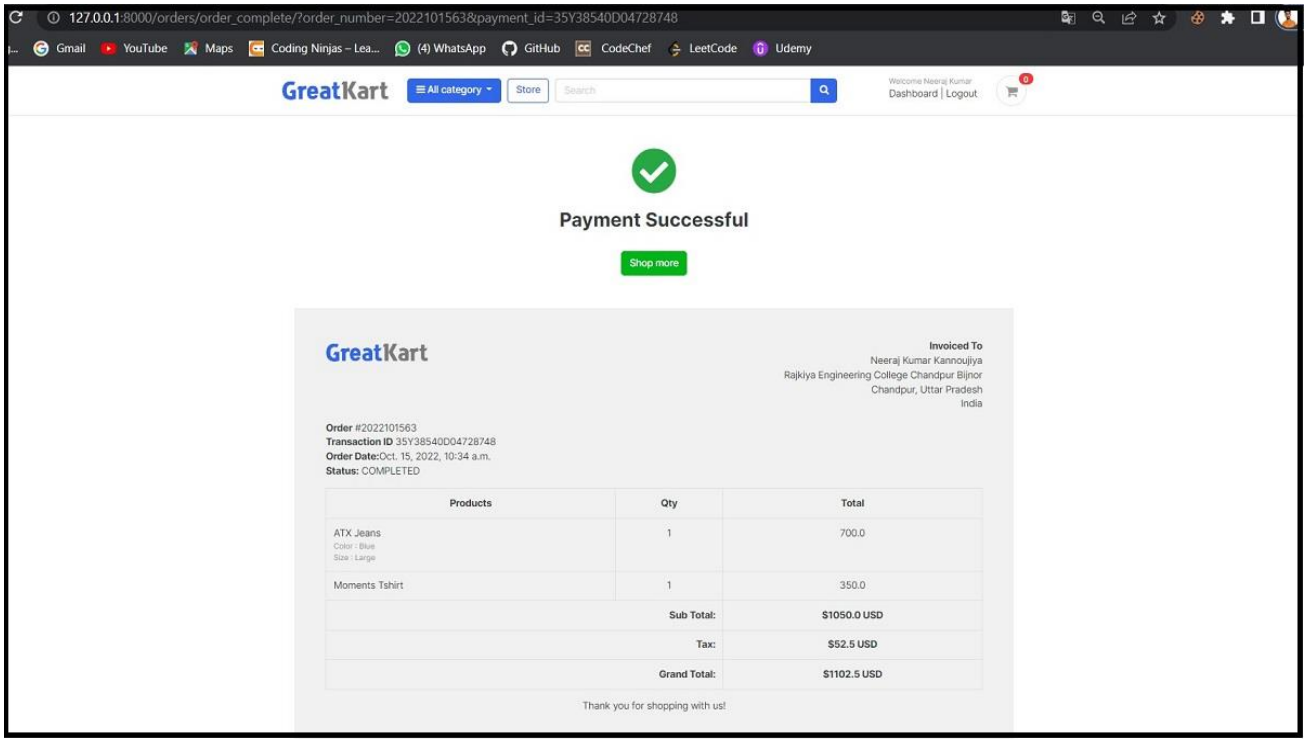
Dashboard



Our Store



Invoice



Conclusion

In conclusion, I can say that this internship was a great experience. Thanks to this project, I acquired deeper knowledge concerning my technical skills, but I also personally benefited. I learned to live in a different environment from the one I am used to. Indeed, I grew more independent in work and also in everyday life. I realized that I could do more things than I thought, like learning new things by myself. There are huge opportunities available for the students who want to work in this field. Many private and public organizations hire web designer for their online work and website development. With the rapid advent of online industry, the demand of web development professionals is increasing, and this has created a huge job opportunity for the aspirants in the upcoming days. Also, an experienced person in this field can also work as a freelancer; there are many online companies which provide online projects to the individuals.

On the behalf of this internship i am able to make many website using Django and i Will make many websites like

- CarZone
  https://neeraj7388011.github.io/carhouse.github.io/

- Blog Website
  https://neeraj7388011.herokuapp.com/

References

- https://stackoverflow.com/
- https://www.wikipedia.org/
- https://www.learnvern.com/
- https://www.geeksforgeeks.org/