**Technical Documentation**

Restaurant Ecommerce Site

Submitted By:

Janine Magayanes

Definition of Terms:

**What is Django?**

Django is an open source python framework for web development. It is useful for its built-in functions and easy administration management.

The following terminologies are commonly used while working with Django:

**Project**

The term project describes the entire Django web application. It is a collection of apps, database settings, installed dependencies, static files such as images etc. This Restaurant E-Commerce Site is an example of a Django project.

**App**

An App is a web application that makes up a project. It is contained in a folder which includes python files such as admin.py, forms.py, views.py, urls.py and models.py to name a few. An example would be the Ecommerce and Core folders in this project (Restaurant E-Commerce Site).

**Admin**

Django has a built-in interface that makes it easy to configure database objects in an [app.](http://app.it/) An admin refers to a user, who has authorization to access this interface in the /admin route. To create a new admin, run python manage.py createsuperuser. The superuser can access the admin page to gain full control on the application: adding, deleting editing items on the database.

**Model**

In the Django documentation, a model is defined as the source of information about the data stored. It contains the data’s relevant field and behaviour which can be be configured in the models.py file found in an app. Each model maps to a single database table and every changes made have to be updated by running either python manage.py makemigrations or python manage.py migrate.

**View**

Django has the concept of “views” to encapsulate the logic responsible for processing a user’s request and for returning the response. This can be accessed in the views.py file inside an app.

**Url**

To display a view at a particular URL, it has to be configured on the url.py file.

**Form**

A form class in Django describes a form’s structure and determines how it works and appear in the app. It is usually found in the forms.py file.

**Template**

The template contains all the html files related to the views and routes in Django. It may contain subfolders like Snippets for those html files that are used on multiple views. For this particular app a templating engine called Twix which is built in in Django.

The Restaurant Ecommerce Site - At a Glance:

pasted-image.tiff

As mentioned above, the project’s contains the following folders:

1. Pycache
2. Core
3. Ecommerce
4. Functional Test
5. Media Root
6. Static in Env
7. Templates
8. Virtual Environment

and the following files:

1. db.sqlite3 (default database)
2. manage.py
3. requirements.txt
4. Read ME
5. Procfile

Manage.py File

It is the tool for executing many Django-specific tasks -- starting a new app within a project, running the development server, running your tests... It is also an extension point where you can access custom commands you write yourself that are specific to your apps.

Requirements.txt File

This requirements. txt file is used for specifying what python packages are required to run the project. To install all the dependencies in the folder, run pip install -r requirements.txt 

DB.SQlite3 File

The file is database file where all the data that will be generating is stored. It is a local file as Django is a server-side framework and it treats your computer as the host when you actually run the server in command line/terminal.

Pycache Folder

This contains the files for the main application’s configuration.

1. Pycache File  
     
   When running a program in python, the interpreter compiles it to bytecode first (this is an oversimplification) and stores it in the \_\_pycache\_\_ folder. If you look in there you will find a bunch of files sharing the names of the .py files in your project's folder, only their extensions will be either .pyc or .pyo. These are bytecode-compiled and optimized bytecode-compiled versions of your program's files, respectively. It makes the program run faster.

Ecommerce Folder

This app contains the files for the main application’s configuration.

* 1. URLs.py   
       
     This contains the main url routes for the application. The app’s root url are:

Local: <http://127.0.0.1:8000/>

Heroku: <https://otito.herokuapp.com/>

1. Wsgi.py  
     
   WSGI is the Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.
2. Settings Folder  
     
   This file takes care of the basic Django functionalities. Currently the folder is divided into three files:

1. Base.py - contains the basic app settings that will be applicable in either Development or Production:

BASE\_DIR- the Django project directory. It is the same directory where manage.py is located. Default value is

os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

SECRET\_KEY- value is the key to securing signed data and used for making hashes.

DEBUG - set it to true during development to see the error message in case of any bugs, and false when in production.

INSTALLED\_APPS- lists of all the apps used if the project. A designated app name which is usually in the apps' apps.py file has to be added in this list.

MIDDLEWARE -Middleware is a framework of hooks into Django’s request/response processing. It’s a light, low-level “plugin” system for globally altering Django’s input or output.

2. Development.py - File responsible for the settings during development

3. Production.py - File responsible for the settings during development

1. Urls.py  
     
   This file contains the list of routes for authentication (<http://127.0.0.1:8000/accounts/>….) and their respective names which are used for url reference in the template(html) files.

Core Folder

The Core app is where the main configurations for the project is located. It contains the following folders:

1. Migrations  
     
   This folder is udated everytime there are changes on the models.py after running ‘python manage.py makemigrations’ and ‘python manage.py migrate’
2. Views   
     
   This controls the logic for processing user request and rendering it accordingly. It contains the code that configures a specific page of the project. Those that requires html templates are class based, while those that are just specifically for POST requests, (eg add to cart ) are function based.
3. Admin.Py  
     
   This file is configured to add authentication models or objects in the Django admin site.
4. Models.Py  
     
   This file is configured to add models or objects for the database. It also has customised methods for specific functions such as getting the total amount for a specific Order based on the item price.
5. Forms.py  
     
   This contains the available forms for specific user information.
6. Urls.py  
     
   This file contains the list of available urls for the app and their corresponding views and namespaces.
7. Tests.Py

This contains the tests for urls, views, forms and models.py.

Static Folder

Contains the javascripts, css and images. They are grouped on separate folders based on the file type.

Templates Folder

This is where all the html files are located. The following is a short description on each file:

1. Base.html - contains the header part of the page, this is inherited in all other pages
2. Checkout.html - the page containing the form for the User’s Checkout information
3. Home.html - the page where the user is first directed to. Contains the main cover page of the app.
4. Navbar.html - contains the snippet for the navigation bar, like the base.html, this is also inherited in all pages.
5. Ordersnippet.html - contains the coupon form, however, this feature is not yet fully implemented on the site.
6. Ordersummary.html - page that shows the list of all products that are currently in the cart
7. Payment.html -contains the payment form for the user’s Payment (credit card information)
8. Product.html - contains one specific product page where the price and description are listed.
9. Shop.html - page that lists all the available products
10. Requestrefund.html - contains the form for refund, at the moment this feature is not yet implemented

**\***Virtual Environment Folder

The folder where are the dependencies are saved.

Functional Test

The folder contains the test files using the Selenium web driver.

Sources:

Backend:

Django Documentation - <https://www.djangoproject.com/>

Stripe Documentation - <https://stripe.com/docs/api>

Python Crash Course by Erik Matthes

JustDjango - <https://justdjango.com/>

Roy Fielding's Dissertation - <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>

Database:

SQLite Documentation - sqlite.org/index.html

Postgres Documentation - <https://www.postgresql.org/>

Heroku Postgres - <https://devcenter.heroku.com/articles/heroku-postgresql>

Frontend:

W3C School - <https://www.w3schools.com/>

Deployment:

Heroku Documentation - <https://devcenter.heroku.com/categories/reference#deployment>

Testing:

Selenium Documentation - <https://www.selenium.dev/documentation/en/>