**Virtualstockenv:**

The folder “virualstockenv” is the root of the entire project that we have developed. This folder houses various folders, the python code that is used for running the server etc. The explanation of each folder in this project file and their contents will be briefly explained in the following discussion in a sequential order.

There are few more sub folders and files generated in this project file while creating the project using Django. The files that are built when creating the project are as below:

1. virtualstockenv
2. manage.py

**manage.py:**

The most important file in the entire project is “manage.py” file. It is the basic server code using which we can run the server whenever required. By running the manage.py program, we will be able to start the server and run the applications developed in the project by using this server. This python program calls all the python program files, that have been already created or the new python program files we create for developing an application.

**virtualstockenv:**

This subfolder which has the same name as the project is known as the Django root directory. The main purpose of this Django root directory is that it holds the connection between the project and Django. It holds all the necessary python files that are required to maintain a database

Some of the important files in the virtualstockenv Django root we used are:

1. \_\_init\_\_.py
2. asgi.py
3. serializers
4. settings.py
5. urls.py
6. wsgi.py
7. **\_\_init\_\_.py:**

The \_\_init\_\_.py is just a blank python file which is mainly used for initiating the server.

1. **ASGI:**

ASGI, or the Asynchronous Server Gateway Interface, is the specification which Channels and Daphne are built upon, designed to untie Channels apps from a specific application server and provide a common way to write application and middleware code. This whole process is handled by the python program asgi.py, which is an inbuilt python program in Django.

1. **settings.py:**

The settings.py file is one of the important files that we have used in our project. It is an inbuilt program created with the project but can be modified according to the project requirements. The settings.py holds various parameters that can be used for connecting the project to a data base server, middleware required for the project, application configuration files etc. By setting the parameters in the DATABASES dictionary we will be able to connect all the data bases that uses the parameters we used.

1. **URLS:**

Urls.py file holds few of the URLS of the project. The urls.py file is mainly used for mapping an URL to a specific functionality that can be used by a user through an application created. This urls.py file also holds the URL of the admin control site. The main function of the urls.py file is to map the functions defined in the views.py with the respective URL paths.

1. **WSGI:**

Beneath Django, Flask, Bottle, and every other Python web framework, lies the Web Server Gateway Interface, or WSGI for short. WSGI is to Python what Servlets are to Java — a common specification for web servers that allows different web servers and application frameworks to interact based on a common API. This whole process is handled by the python program wsgi.py, which is an inbuilt python program in Django.

Now that we have discussed all the inbuilt files generated with the creation of the project, we continue our discussion with the applications we created as part of the project.

**stockupdate:**

The stock update directory holds the python programs that will be used for data collection. We have modified the apps.py of stockapp so that the server itself calls the stock collection functions for both the real time and historical data of all the stocks we are using.

The stockupdate contains two python files.

1. data\_collection.py
2. updater.py
3. **data\_collection.py:**

The data\_collection.py is the program we created during the phase 1 of the project. The program file contains two functions:

1. get\_hist():

The get\_hist() function fetches the data of each stock we used in the project at the end of the day i.e. at 5pm. It inserts the date, open value, close value, high value, low value, and the volume of each stock recorded at 5 pm every weekday. This data is later used for prediction and display etc.

1. get\_real():

The get\_real() function works in the same way as the get\_hist() but with modifications like fetching the current time the data is recorded for each stock. Unlike the get\_hist(), get\_real() will be running for every minute of each week day from 10 AM too 4PM, i.e. during the working time of stock market.

1. **updater.py:**

The updater.py has a function start() which recursively calls itself as log as the server runs. The start() will also call the two functions declared in the data\_collection.py for the required time intervals. The start() is programmed in such a way that it calls the get\_hist() at 5 PM every weekday, get\_real() for every minute from 10 AM to 4 PM for every weekday/working day. In addition, it also calls the thresholds () function, declared in thresholds.py, for every 10 minutes in the same way, to check if the thresholds set by a user is reached or crossed. The thresholds.py will be discussed in depth in the stockapp application.

**stockapp:**

The stockapp is one of the applications we developed as a part of the project. The stockapp holds the files for css format, html ui pages for the web application display, and function files which houses the functionalities that can be utilized by the user at client side.

Few of the files and directories that are utilized in the project are as below:

1. **static:**

The static directory contains all the css files we used for designing the UI pages. We have used the main.css file for setting the style for almost all the UI pages we used.

1. **Templates:**

The templates folder holds all the html UI pages we have developed as a part of the project. Most of the html pages are designed in such a way that they can be used along with the base.html, which gives an outline of the pages. By using the templates, we have developed non redundant html files.

1. **\_\_init\_\_.py:**

The \_\_init\_\_.py is just a blank python file which is mainly used for initiating the application. It has the same purpose as that of virtualstockenv.

1. **admin.py:**

This file provides the admin, the tables from the data base which he can alter from the admin page. We have used this file to add/remove stocks that we want to use/remove for the project, into the data base.

1. **apps.py:**

The apps.py file is a configuration file of the stockapp application. We have used this file to trigger the start of updater.py. This allowed us to run the updater code as soon as the server is up.

1. **indicators\_models.py:**

The indicators\_models.py was developed by us in the process of the project, to analyze the stock values passed and provide a suggestion to the user based on the analysis using the indicators.

1. **long\_term\_nn.py:**

The long\_term\_nn.py is another python file we created in the process. The file will be called by the longtermprediction() of views by passing the stock id and the number of predictions required as requested by the user. The get\_long\_term() which prvides the prediction of the stock values call the get\_suggestion() of indicators\_models.py by passing necessary arguments in order to produce a suggestion result. The get\_long\_trem returns the prediction values and the suggestion computed to lingtermprediction() of views.py which in turn displays this on the web browser.

1. **short\_term\_pcf.py:**

The working of the short\_term\_pcf.py is almost the same as that of long\_term\_nn.py, but the only difference is long\_term\_nn.py computes data from historical\_stocks data table while short\_term\_pch.py computed data from real\_time data table. The get\_short\_term() is called by the shorttermprediction() of views.py.

1. **models.py:**

The models.py file plays a major role in connecting the data base to the Django project. The main purpose of the models.py file is that it is used to create the classes which can be used as aliases for all the tables that are required or used in the project. The tables that contain the data of users, thresholds, realtime and historical in the data base are mapped to the classes using the models.py file.

1. **queries.py:**

The queries.py file is another file that we created in the process. It is used to generate the query results of the stock, the user requested for. Some of the queries that are made are the average value of the stock in the week, the average value of the stock in a day etc.

1. **thresholds.py:**

The thersholds.py is used for the special feature we implemented in our web application. It allows the user to select any stock and set a threshold. Whenever this threshold is reached or crossed, the user will be notified. The file has two functions check\_thresholds() which will be run for every 10 minutes using the updater.py to see if the threshold is reached or crossed and notify\_user to send an email to the user once the threshold is reached.

1. **urls.py:**

This urls.py file has the same functionality as that of virtualstockenv Django root’s urls.py. It maps all the urls to the specific functionalities.

1. **views.py:**

The views.py file is one of the most useful files. It is used to map a functionality to a UI page. All the logical operations of an UI page are done in views.py file. We used the functions of views.py to set what data should be displayed in a webpage related to the project.

This gives us the idea on what functionalities are available in our project and how we achieved them using the inbuilt python files and python files we created.

**users:**

The users is another application we have created as a part of the project. The users application will be used for functionalities like new user creation i.e. registration, user login and validation, password update etc. which are mainly used for user management. The users application has almost the same inbuilt files as that of stockapp.

1. **templates:**

The templates directory houses the UI html pages we use for login, register, password update and logout etc. It is of the same use we have that from the stockapp application.

1. **admin.py:**

It has the same functionality as that of the admin.py from stockapp application.

1. **apps.py:**

It is same as the apps.py from stock app application. But we have not used it on our project as for now.

1. **forms.py:**

The forms.py file is used for modifying the existing forms so that we can have the self-defined forms instead of default forms. This allowed us to prompt the user to enter the email using which he can register etc. We can declare and define the fields we require from the user for registration and password updates etc.

1. **models.py:**

The models.py is the same as that of stockapp but we have not used this file for the project. The basic functionality is the same.

1. **registermail.py:**

It contains the the notify\_user function which gets triggered whenever a new user register to the web application we created. It uses the email id entered by the user during registration and will be called by the register() of views.py.

1. **views.py:**

The views.py file has the same functionality as that of stockapp. This views.py file is mainly used for managing the pages which are used for user management. The file contains functions like register(), profile() and change\_password(). By using these pages the user at the client side can manage his data without consulting the admin.