

Advanced Software Engineering Topics

Elaboration Phase-1

Group-11

Team Members:

Jigish Vyas (110025730)

Azmina Aziz Vanzara (110036278)

Kalpita Manojkumar Soni (110025730)

Sree Nithya Kothapalli (110037170)

Table of Contents:

1. Introduction
2. Background Information
3. Configuration and Running
4. Implementation
5. Testing

Introduction:

Spondulator is a stock market learning tool designed and developed to help people make better financial decisions in life. By using this application, the users will not only be able to make the better financial decisions in life but will also gain the confidence to invest in the real-world stock market game and win the same.

Background Information:

The name Spondulator is derived from the two words namely, spondulix meaning money and simulator. We were looking for some challenges faced by the modern society and address the same with the gained technical skills. We also came to know that two-thirds of Americans are financially illiterate as per Investopedia. As it is rightly said, “Necessity is the mother of invention”, and hence the idea of Spondulator came to our mind. Spondulator is a stock market learning tool which will provide a better way to learn the stock market by experiencing the same in the virtual environment, which seems to be intimidating to do so in the real world with the help of real money. We will give some initial amount of virtual money to the users to play with, using which they can look up for the stock data, buy the stock, as well as sell it. The system will show the calculated profit or loss on the total committed money. We will also use IBM cloud technologies to perform sentiment analysis & trend analysis about any company from different news sources and classify it into positive, negative or neutral category in order to make an informed decision. This can also help to make an investment decision around the company’s future value. We called the third party API from the IEX cloud where the stock data is stored.

```
{
  "symbol": "IBM",
  "companyName": "International Business Machines Corp.",
  "primaryExchange": "New York Stock Exchange",
  "calculationPrice": "Close",
  "open": 115.88,
  "openTime": 1684673880825,
  "openSource": "official",
  "close": 114.84,
  "closeTime": 1684696438348,
  "closeSource": "official",
  "high": 115.1,
  "highTime": 1684718793814,
  "highSource": "15 minute delayed price",
  "low": 113.39,
  "lowTime": 1684674588257,
  "lowSource": "15 minute delayed price",
  "latestPrice": 114.84,
  "latestSource": "Close",
  "latestTime": "November 6, 2020",
  "latestUpdate": 1684696438348,
  "latestVolume": 5249171,
  "iexRealtimePrice": 114.87,
  "iexRealtimeSize": 6,
  "iexLastUpdated": 1684696392868,
  "delayedPrice": 114.46,
  "delayedPriceTime": 1684718793814,
  "oddLotDelayedPrice": 114.88,
  "oddLotDelayedPriceTime": 1684696399098,
  "extendedPrice": 114.46,
  "extendedChange": -0.42,
  "extendedChangePercent": -0.88368,
  "extendedPriceTime": 1684718793814,
  "previousClose": 114.77,
  "previousVolume": 4982286,
  "change": -0.73,
  "changePercent": -0.88636,
  "volume": 5249171,
  "iexMarketPercent": 0.8426195841233384,
  "iexVolume": 223295,
  "avgTotalVolume": 6449984,
  "iexBidPrice": 0,
  "iexBidSize": 0,
  "iexAskPrice": 0,
  "iexAskSize": 0,
  "iexOpen": null,
  "iexOpenTime": null,
  "iexClose": 114.87,
  "iexCloseTime": 1684696392868,
  "marketCap": 181616148288,
  "peRatio": 12.82,
  "week52High": 158.75,
  "week52Low": 98.50,
  "ytdChange": -0.15884988888888888,
  "lastTradeTime": 1684696438388,
  "isUSMarketOpen": false
}
```

Configuration and Running:

Steps to create new Django Project and new app inside it:

1. `django-admin startproject PROJECT_NAME`
2. `cd PROJECT_NAME`
3. `python manage.py startapp APP_NAME`
4. Add the `APP_NAME` in `settings.py` in list of `INSTALLED_APPS`
5. Register urls in project level `urls.py` (default one), as well as in app level `urls.py` after creating the `urls.py` in your app.

After cloning the spondular in your pc, remove `db.sqlite3`, `pycache`, and `migrations` folder as well from all sub folders of django project.

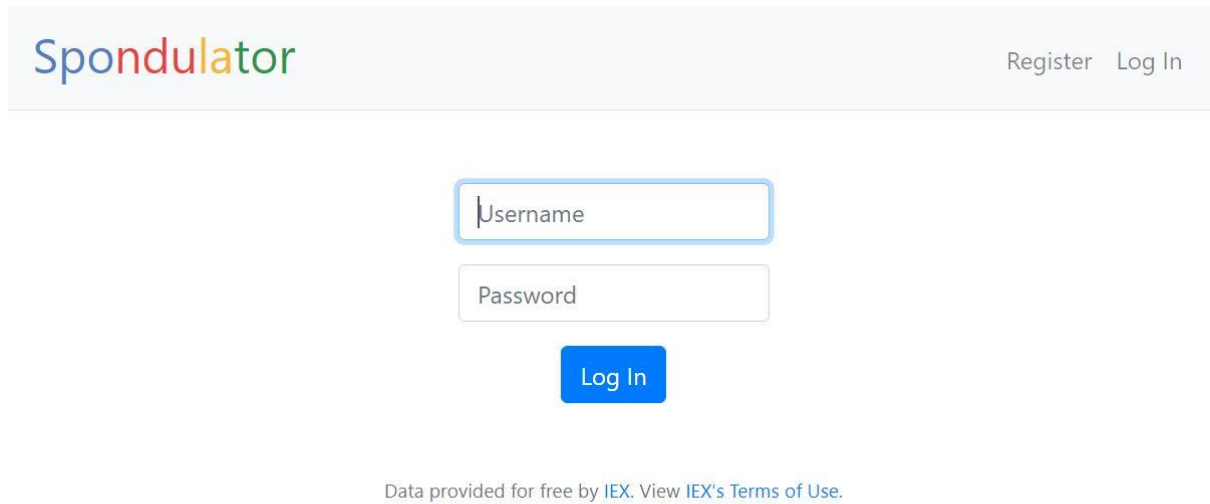
NOTE: Name of our django project is finance and the app inside it is spondulator.

Once removing above files, run the following commands in your terminal:

1. `python manage.py makemigrations spondulator`
2. `python manage.py migrate`
3. `python manage.py runserver`

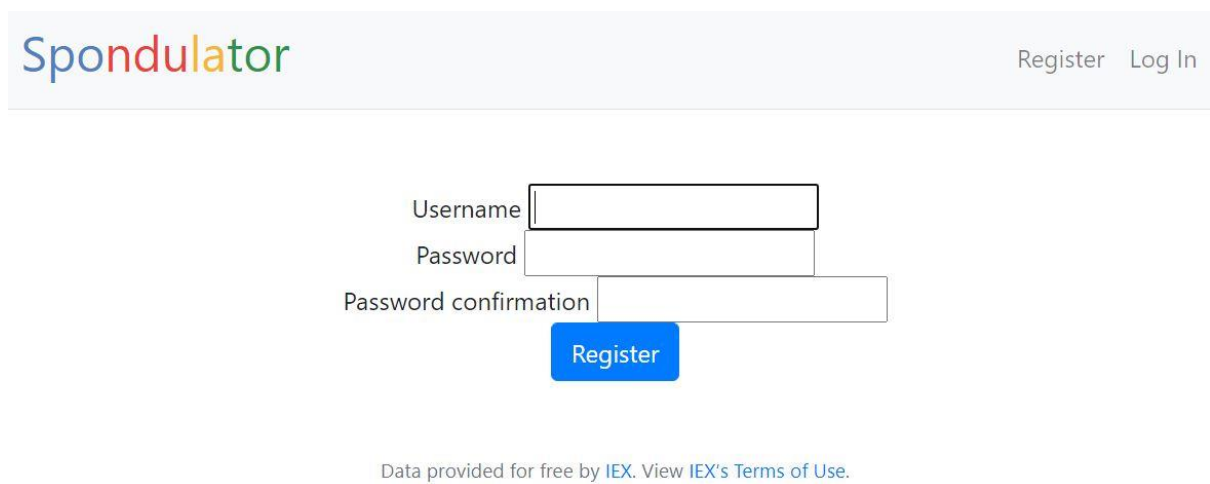
NOTE: You can visit to `"/admin"` app to see the models i.e. tables in our database and modify the same from the admin panel only which is a Django default app.

Implementation:



The screenshot shows the login page of the Spondulator application. At the top, there is a header bar with the 'Spondulator' logo on the left and 'Register' and 'Log In' links on the right. The main content area features a login form with two input fields: 'Username' and 'Password'. Below these fields is a blue 'Log In' button. At the bottom of the page, there is a small text link: 'Data provided for free by IEX. View IEX's Terms of Use.'

When we execute the code, we can see this web application. This is the login page. The existing user can directly login to the application.



The screenshot shows the registration page of the Spondulator application. At the top, there is a header bar with the 'Spondulator' logo on the left and 'Register' and 'Log In' links on the right. The main content area features a registration form with three input fields: 'Username', 'Password', and 'Password confirmation'. Below these fields is a blue 'Register' button. At the bottom of the page, there is a small text link: 'Data provided for free by IEX. View IEX's Terms of Use.'

If the user is new to this application, then the user needs to register. The user has to enter the username, password and also the user needs to confirm the password again.

Account was created for Group11

[Log In](#)

Data provided for free by [IEX](#). View [IEX's Terms of Use](#).

Once after the user registers, then the account will be created. After this, the user should enter the username and password to login to the application.

Hello, Group11

Group11has successfully logged in.

Symbol	Name	Shares	Current Price	TOTAL
CASH IN HAND				\$10,000.00
				\$10,000.00

Data provided for free by [IEX](#). View [IEX's Terms of Use](#).

Once after logging in, we can see the Spondulator portfolio. In this, the user will be given \$10,000 initially (virtual money).

Symbol :

Data provided for free by [IEX](#). View [IEX's Terms of Use](#).

We the user clicks on quote then the user can see this page. In this page, the user needs to enter the symbol.

A share of International Business Machines Corp. (IBM) costs \$114.09.

Data provided for free by [IEX](#). View [IEX's Terms of Use](#).

After entering the symbol, it will display the current share value. This cost will help the user to make the investment decisions. i.e. either sell the stocks or buy the stocks

Logged out successfully.

Data provided for free by IEX. [View IEX's Terms of Use.](#)

Once after learning about the stock data, buying the shares and selling the shares, the user can logout.

Testing:

In spondulator, we have used the test-driven development methodology to test the code written by us and make sure that it runs as expected. The project is developed using Django framework for which we have used it's TestCase library to write the unit tests for the database models.

We have written the test case for Purchase model which verifies the Purchase for the following three conditions:

1. The max length of the stock symbol should be 5.
2. The total number of shares should be positive.
3. The amount of stock should be positive as well.

Test case written in test.py file for our application is as follow.

```
1 from django.test import TestCase
2 from .models import Cash, Purchase
3 from django.contrib.auth.models import User
4
5 # Create your tests here.
6 class PurchaseTestCase(TestCase):
7
8     def setUp(self):
9         user1 = User.objects.create(username="Jiggy")
10        Purchase.objects.create(my_user=user1, stock="NFLX", shares=5, price=100.00)
11        Purchase.objects.create(my_user=user1, stock="NFLX", shares=5, price=-100.00)
12        Purchase.objects.create(my_user=user1, stock="NFLX", shares=-5, price=100.00)
13        Purchase.objects.create(my_user=user1, stock="NETFLX", shares=5, price=100.00)
14
15    def test_valid_purchase(self):
16        user1 = User.objects.get(username="Jiggy")
17        p = Purchase.objects.get(my_user=user1, stock="NFLX", shares=5, price=100.00)
18        self.assertTrue(p.is_valid_purchase())
19
20    def test_invalid_purchase_stock(self):
21        user1 = User.objects.get(username="Jiggy")
22        p = Purchase.objects.get(my_user=user1, stock="NETFLX", shares=5, price=100.00)
23        self.assertFalse(p.is_valid_purchase())
24
25    def test_invalid_purchase_shares(self):
26        user1 = User.objects.get(username="Jiggy")
27        p = Purchase.objects.get(my_user=user1, stock="NFLX", shares=-5, price=100.00)
28        self.assertFalse(p.is_valid_purchase())
29
30    def test_invalid_purchase_price(self):
31        user1 = User.objects.get(username="Jiggy")
32        p = Purchase.objects.get(my_user=user1, stock="NFLX", shares=5, price=-100.00)
33        self.assertFalse(p.is_valid_purchase())
```

As we can see below, the Purchase Model might look something like this and the developer may have mistakenly written the wrong logic behind the is_valid_purchase method.

```
class Purchase(models.Model):
    my_user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="purchases")
    stock = models.CharField(max_length=5, null=True)
    shares = models.IntegerField(null=True)
    price = models.FloatField(null=True)
    bought_at = models.DateTimeField(auto_now_add=True, null=True)

    # Adding a Test Case
    def is_valid_purchase(self):
        return self.shares > 0 or self.price > 0 or (len(self.stock) > 0 and len(self.stock) <= 5)
```

In order to make sure our application runs as expected, we wrote the test cases for the valid purchase as well as 3 invalid purchases in the test.py file in our project and we got 3 failed tests as a result.

```
PS E:\MAC\ASE\Spondulator\Spondulator\finance> python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
FFF.
=====
FAIL: test_invalid_purchase_price (spondulator.tests.PurchaseTestCase)
-----
Traceback (most recent call last):
  File "E:\MAC\ASE\Spondulator\Spondulator\finance\spondulator\tests.py", line 33, in test_invalid_purchase_price
    self.assertFalse(p.is_valid_purchase())
AssertionError: True is not false

=====
FAIL: test_invalid_purchase_shares (spondulator.tests.PurchaseTestCase)
-----
Traceback (most recent call last):
  File "E:\MAC\ASE\Spondulator\Spondulator\finance\spondulator\tests.py", line 28, in test_invalid_purchase_shares
    self.assertFalse(p.is_valid_purchase())
AssertionError: True is not false

=====
FAIL: test_invalid_purchase_stock (spondulator.tests.PurchaseTestCase)
-----
Traceback (most recent call last):
  File "E:\MAC\ASE\Spondulator\Spondulator\finance\spondulator\tests.py", line 23, in test_invalid_purchase_stock
    self.assertFalse(p.is_valid_purchase())
AssertionError: True is not false

-----
Ran 4 tests in 0.033s

FAILED (failures=3)
Destroying test database for alias 'default'...
```

Activate Windows
Go to Settings to activate Windows.

After further inspection of our function in the Purchase Model, we spotted the logical bug in our code and then returned the modified Boolean expression as follow.

```
# Adding a Test Case
def is_valid_purchase(self):
    return self.shares > 0 and self.price > 0 and (len(self.stock) > 0 and len(self.stock) <= 5)
```

Thus, as a result, we ran all the 4 test cases successfully with the better results.

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

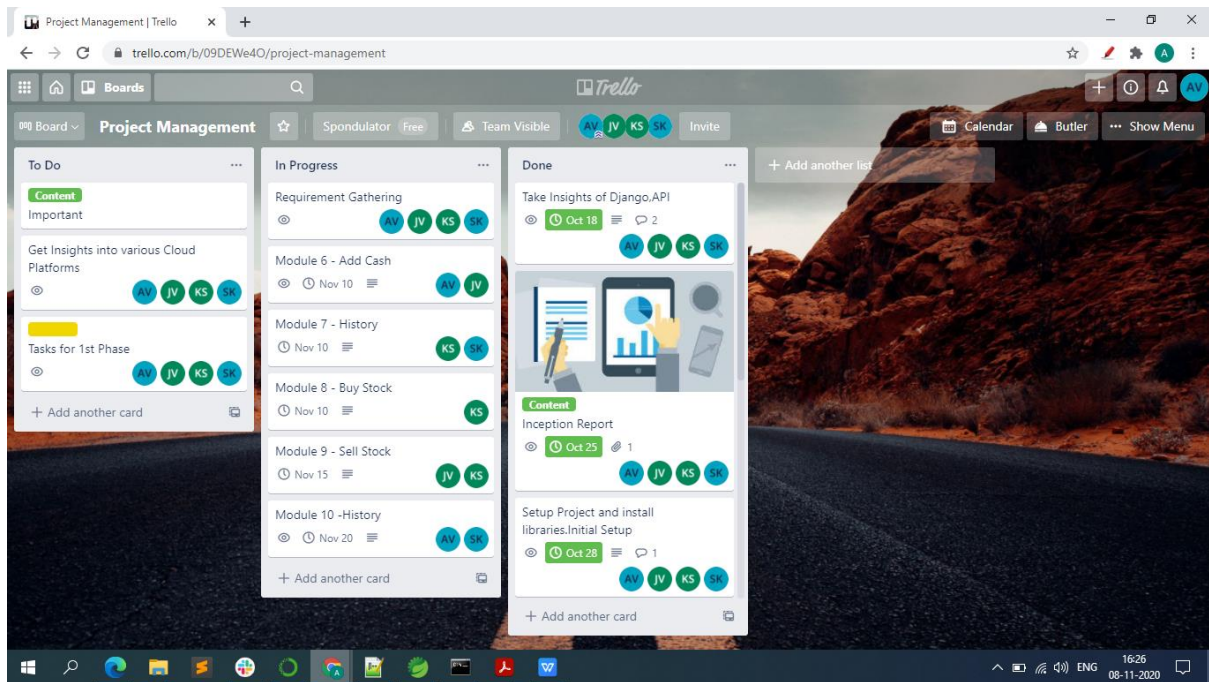
PS E:\MAC\ASE\Spondulator\Spondulator\finance> python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
-----
Ran 4 tests in 0.038s

OK
Destroying test database for alias 'default'...
PS E:\MAC\ASE\Spondulator\Spondulator\finance> []
```

Software Project Management System:

We have used Trello to manage the teamwork, develop the spondulator and implement it. Each person is assigned their work and the project is managed and developed in given time feasibility using collaboration.

The link is: <https://trello.com/b/09DEWe4O/project-management>



Documentation Setup:

We have started the first elaboration phase of our project. The link of GitHub is given below.

<https://github.com/Jigish13/Spondulator>