Ben Heney
Willem Shattuck

REPORT 2

Progress

Ben

Methods for accessing the TwelveData API and obtaining securities information have been implemented. The TwelveData class is an implementation of the APIWrapper interface. The goal was to make the StockWatch website extensible by loosely coupling the API provider to the website's backend. This way, different providers can be used as needed.
Simple unit testing for the TwelveData class is also in place.

The design of the Stock Watch API was also considered, but with so few of the models implemented, it made little sense to begin implementing this functionality so early. Normally, outlining the API would be a great roadmap for building the website, but Django has a model-based API framework. In this case it will be best to build out the rest of the site, then build the API on top of it. Documentation for the Django API framework can be found here: https://www.django-rest-framework.org/

In the coming week (due by 4/22), the core functionality of the site will need to be built. Specifically, the Queue of securities that need to be updated, the "next refresh" times, and the Dashboard view will need to be written. This will likely mean developing classes for Queue, Stock, User, and Dashboard.

Willem

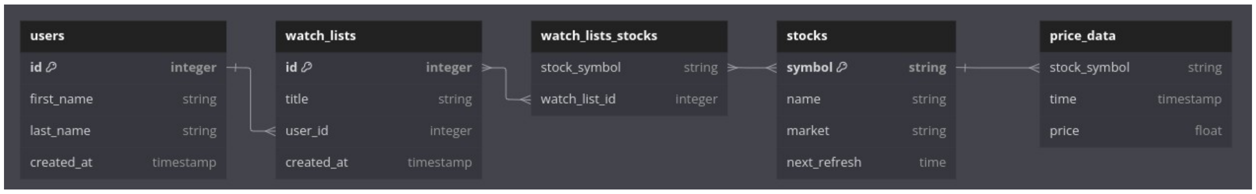
Started defining a simple database schema for managing users, watchlists, stocks, and price data, along with the relationships between them. In models.py User Model, represents a user with first name, last name, and creation timestamp. WatchList model represents a watchlist with a title, associated user (via a foreign key relationship), creation timestamp, and a many-to-many relationship with 'Stock'. WatchListStock Model, represents the relationship between WatchList and Stock. It includes the stock symbol and foreign key references to WatchList. Here is an example of stock model,

```
class Stock(models.Model):
   symbol = models.CharField(max_length=10, primary_key=True)
   name = models.CharField(max_length=255)
   market = models.CharField(max_length=100)
   next_refresh = models.TimeField()
   watchlists = models.ManyToManyField('WatchList', through='WatchListStocks'related_name
   lated_name='stocks', verbose_name='Watch Lists')
```

This Represents a stock with a symbol (primary key), name, market, and next refresh time. It also has a many-to-many relationship with WatchList through WatchListStocks. PriceData Model, Represents price data for a stock, including the stock symbol, timestamp, and price.

Under templates, started laying out base.html file. The overview of what has been added is as follows. This section displays a form for adding a new watchlist. It includes an input field for the title of the watchlist and a submit button. The form is submitted to the add_watchlist URL endpoint using the POST method. {% csrf_token %} is used to include a CSRF token for security purposes.

ERD


users
id integer
first_name string
last_name string
created_at timestamp
watch_lists
id integer
title string
user_id integer
created_at timestamp
watch_lists_stocks
stock_symbol string
watch_list_id integer
stocks
symbol string
name string
market string
next_refresh time
price_data
stock_symbol string
time timestamp
price float