KOÇ UNIVERSITY

College of Engineering

COMP 491 – Computer Engineering Design Project

Final Report

INVESTSMART

Metehan Gelgi 64178

Eren Barış Bostancı 68770

Berke Can Rizai 69282

Damla Yıldız 72851

Irmak Erkol 68924

Project Advisor:

Gözde Gül Şahin

Fall 2022

Table of Contents

I. Abstract

InvestSmart is an ambitious financial platform that aims to empower investors with the ability to analyze and monitor a wide variety of stocks in real-time, by providing them with a single source of information about the market. The project includes a website and mobile application, with the mobile app being the primary component of the platform. The goal of InvestSmart is to provide young investors who rely on finance websites for market updates, with a powerful and intuitive platform that makes it easy for them to track and analyze the market, and make informed investment decisions. The project is designed to be a dynamic platform where targeted stock markets (initially the American stock exchange market) can be expanded easily. Unlike other financial news sites, InvestSmart gathers news from various sources and features an active user system where users can comment on stocks, follow stocks by connecting to their accounts and engage in community discussions. This approach is intended to differentiate it from other financial websites that mostly provide market prices or financial news. The InvestSmart project is an end-to-end software development, with a focus on providing a user-friendly interface.

II. Introduction

Problem Definition and Objectives

The InvestSmart project is an ambitious financial platform that aims to provide investors with a comprehensive solution to track the market and news related to stocks and assets. The primary goal of the project is to empower investors with the ability to analyze and monitor a wide variety of stocks in real-time, by providing them with a single source of information about the market.

To achieve this goal, the project is designed as an InvestSmart product that will comprise a website and mobile application. The mobile app is the primary component of the platform, offering various features and functions that allow users to access and analyze market data, including market price, news, and expected price. Additionally, the mobile app also provides users with tools and resources to help them make informed investment decisions.

The website, on the other hand, serves as a promotional tool for the mobile app, and is designed to provide users with an overview of the platform's capabilities and features.

The project is an end-to-end software product development that includes the development of the front-end, back-end, database and server. The front-end is designed to provide a user-friendly interface that makes it easy for investors to access and interact with the platform's features and data. The back-end is responsible for managing requests from mobile applications and websites, as well as handling the asynchronous gathering of stock prices and news. The database stores the data and makes it available to the back-end. The server-side also ensures that all processes are continuously active, rather than running solely on a local computer.

Overall, the goal of the InvestSmart project is to provide investors with a powerful and intuitive platform that makes it easy for them to track and analyze the market, and make informed investment decisions.

Background and Literature Review

Professional investors and non-professional investors often access different sources of information. Professional investors typically rely on quarterly reports and financial statements, while non-professional investors tend to value the ideas and reports found on financial websites[1]. Consequently, many financial websites and applications attract mostly non-professional investors who trust the opinions of professional investors and invest according to their recommendations[2]. We target young investors who rely on finance websites for market updates. Furthermore, ideas and articles provided by professional investors can serve as a guide, and platforms such as Twitter and certain websites take this guidance to the next level, where they can even influence market trends[3]. This is why there is a high demand for websites that provide market insights and foster a community of young investors. There are a plethora of financial websites available in the market, but many only provide market prices or financial news. We benefit from this dynamic in our project. We have designed Invest Smart as a platform where users can get insights rather than just a website where stock news can be obtained. On top of that, InvestSmart is a dynamic platform where targeted stock markets(The project was started with the American stock exchange market) can be expanded easily. While large websites are quite successful in American stock markets, data from other countries may be less readily available in their platforms.

There are numerous websites similar to our system, such as The Economist, MarketWatch, Financial Times, and Investing.com, to name a few, which are generally considered powerful financial news sites. However, these websites are restricted to providing their own news. In contrast, our platform collects news from various sources such as Google News, Google Finance, Yahoo Finance and web-scraped data, and consolidates all news on a single platform. Additionally, our system features an active user system where users can comment on stocks, follow stocks by connecting to their accounts using authentication modules, and engage in community discussions.
When we check the literature for the summary with NLP, we can find that the general trend towards recent years is employing Transformer models rather than RNN or LSTM that were more popular previously. One noticeable paper is [7] from Google Research that was quite influential. We have decided to use this model since some pre-trained versions are available on Hugging Face both for API usage and self deployed usage. For the sentiment analysis, we also used the Hugging Face's pre-trained model since it would take too much time and resources for us to gather labeled data, create the model, train the model and validate the results. With the use of pre-trained models, we don't need to worry about all these MLOps life cycle steps and instead focus on content and app experience.

III. System Design

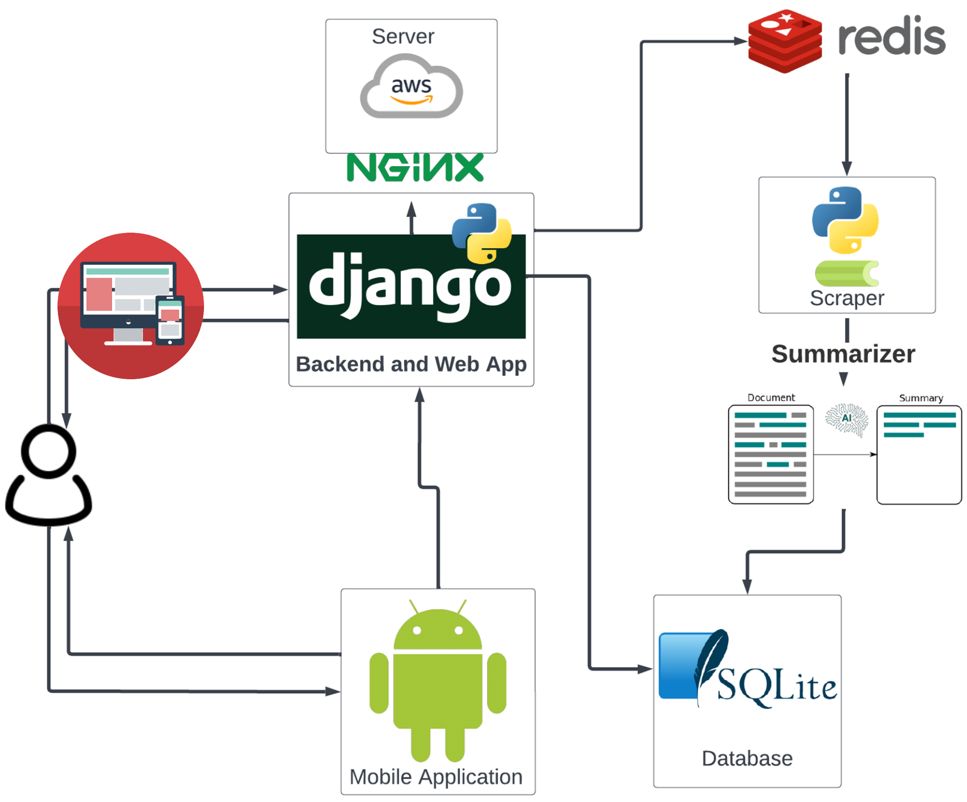


Figure 3.1. Investsmart System Design

InvestSmart is designed with the user in mind, providing a seamless and consistent experience across both the mobile and web platforms. This allows users to manage their investments from any location, at any time. The application is divided into three integral components: the Web, Mobile, and Backend.

The Web interface is designed to provide an easy-to-use and visually appealing experience for users. It is built using the latest web technologies such as HTML,Tailwind, CSS and JavaScript. The Web interface is responsive and adapts to different screen sizes to ensure that it looks and works great on desktop, laptop, tablet and mobile devices. The web interface also includes features such as a search bar, slide bar, daily news, asset pages and a scrollbar, which allows users to find and access information about the assets they are interested in.

The Mobile interface is designed to provide a similar experience to the web interface, but optimized for on-the-go access. It is built using mobile-specific technologies such as Kotlin. The Mobile interface has a clean and intuitive interface, which makes it easy for users to navigate and perform actions.

The Backend is responsible for handling the complex logic and data storage for the application. It communicates with the Web and Mobile interfaces to provide data and handle actions. The backend is built using technologies such as Django and AWS. The Backend makes sure that the application runs smoothly and efficiently, by handling tasks such as data validation, and performance optimization.

Together, these three components work in harmony to provide a comprehensive solution that is both scalable and efficient. The combination of a user-friendly interface, mobile optimization, and a robust backend, allows users to track their investments, monitor their performance, and make informed decisions, all from the convenience of their mobile or web device.

Web

Django - Web Application

We designed the web application side of the project using Django. Because Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Since Django is a python framework, we connected the backend with the web framework easily. The use of a Python-based framework also enabled us to take full advantage of the vast array of libraries and modules that the language has to offer, further streaming the development process.

Django is a framework that uses MVT(Model-View-Templates) patterns. Templates is the interface part written in languages such as HTML, CSS, JS on the front-end side. View is the part that processes the requests from the front-end and enables the web application to work and communicates with the database. A model is nothing but a database table. It is a python class that is linked to a database [6].

When a user interacts with a website, Django sends the request from the web to the backend. (As seen in Appendix A, Django views work with functions such as get, post, which are similar to HTTP requests). After the necessary operations are done in python helper modules, Django returns a response by using the necessary data from the database. This view is demonstrated to the user in a meaningful and user-friendly way on the front end.

In our architecture, we have a Django Rest module that is under the Django application. This is because we started the project with the back-end for the website however, we quickly realized that we couldn’t serve the app from this Django application. We opened another application inside this with the Django REST framework that helps developers to create restful APIs. If a query is sent to normal website, Django answers and if the URL has /api/ in it, REST API answers it.

Frond-end

HTML, CSS, and Tailwind and JavaScript are technologies that we used in our project to create a responsive and user-friendly webpage. HTML, or Hypertext Markup Language, is the standard markup language used to create web pages. It is used to structure the content of a webpage, including text, images, and links. CSS, or Cascading Style Sheets, is used to style and layout web pages. It is used to control the visual presentation of a webpage, including font size, color, and spacing. Tailwind is a utility-first CSS framework that makes it easy to create consistent, responsive designs. It provides a set of CSS classes that can be used to quickly create common layout and styling patterns, without having to write custom CSS. In addition, we have used JavaScript to create the scroll bar feature on the web page. The scrollbar allows users to easily navigate through the content on the webpage, ensuring a smooth and seamless user experience.

Our webpage is designed to be responsive for all devices, including PC, laptop, mobile, and tablet. This means that the layout and design of the webpage will adjust automatically to fit the screen size of the device being used. This ensures that users have a seamless experience regardless of the device they are using to access the webpage.

The header of our webpage features a logo and a search bar. The logo serves as a visual representation of our brand. The search bar allows users to quickly find specific assets by typing in the asset's name or symbol.

We implemented a slide bar that shows prices of popular assets. The slide bar is displayed on the homepage and allows users to quickly see the current prices of assets without having to navigate to a different page. We also have a daily news feature that provides users with the latest news and updates about the assets they are interested in. The news feature is prominently displayed on the homepage and is updated on a daily basis.

When a user searches for an asset using the search bar, they are directed to an asset page. On this page, they can see a price chart and news specific to that asset. This allows users to easily access detailed information about the assets they are interested in without having to navigate through multiple pages. Overall, our project uses HTML, CSS and Tailwind to create a responsive webpage that makes it easy for users to find and access information about the assets they are interested in.

Android App

For the Android App, we used the following technologies.

Kotlin: We preferred it mostly because of its popularity, it is already being used by bigger firms like Pinterest, Slack, and Airbnb. There are a couple of reasons behind its recent popularity in the Android scene. It is more concise, rough estimates indicate approximately a 40% cut in the number of lines of code. It is also more type-safe thanks to its support for non-nullable types. And it is easy to write expressive code with Kotlin thanks to its features like smart casting, higher-order functions, extension functions, operator overloading, singleton pattern, data classes, and lambda expressions with receivers.

Jetpack Compose: Instead of XML, we used Android's new toolkit to build a native UI. Even though it is quite new in Android development, it has a lot of advantages over XML. It lets us write a lot less code which makes it more easy to maintain. It uses a declarative API which is quite intuitive, and thus easy to learn. We were able to write stateless components with Jetpack Compose that are not tied to specific fragments so that they are easy to use and test.We took advantage of this feature while we were building the comment section and news list where the components are used by different screens. Since the compostables are stateless, there is one source of truth, and when the app state changes, UI updates automatically. It also has built-in support for Material Design which we adopted in our UI/UX.

Coroutines and flows: They were beneficial to manage long-running tasks that might otherwise block the main thread and cause our app to become unresponsive like getting prices from our backend or downloading asset images. We used them all over the app since we didn't want to block the user and decided to perform the tasks such as loading the news at the home page in the background.

Retrofit: We used this library to manage interactions with our backend.

Hilt: Since we used a dependency injection design pattern where classes define their dependencies without constructing them, we used this library which constructs the objects and provides its dependencies at compile time by looking at the overall dependency tree in the app. This way, we are planning to manage the dependencies without duplicating code or adding more complexity.

For the app architecture, we used three layers: UI layers where we have our views and viewmodels, domain layer where we have the use cases of our app, data layer where we handle the API requests and their results.

We have five main screens: Home screen where users can see the latest stock news and trending assets. Profile screen where users can see their own comments. Search screen where users can search for different assets. Favourite screen where users can see the assets they saved previously. Assets screen where users can see the prices, news and comments about that particular asset.

Backend

Django - Rest Framework

For the API calls from Android App, we used Django Rest Framework (DRF), which is a powerful and flexible built-in toolkit for building RESTful APIs. For the website we used vanilla Django framework. It provides important features such as serialization, authentication, query parameters, and more. It also includes a browsable API feature, which allows for easy testing and debugging of the API. One of the key features of DRF is the serialization component, which allows for easy conversion of complex data types such as Django model instances and querysets to JSON, XML or other formats. For this project we used JSON format for our APIs.

For the authentication and authorization, instead of using a built-in authentication system that supports several authentication classes such as BasicAuthentication, TokenAuthentication and SessionAuthentication, we used the django-rest-knox authentication package. The main advantage of using knox authentication is, it is a simple and lightweight package for handling authentication for RESTful APIs that can provide a simple authentication process for both token and session-based authentication. Knox provides several features such as token-based authentication, token refreshing, and token revoking and allows for easy management of tokens. However the key benefit of using Knox is that it allows for token-based authentication, which is more secure than traditional session-based authentication. Token-based authentication eliminates the need to store session data on the server, reducing the attack surface for malicious actors. When the user login or register we return a token and in the mobile application we use these tokens to let the users make their account based actions more secure, like making comments, favoriting stocks, liking comments…

Recommendation Module

In this module, we have a dedicated part that is only responsible for making stock recommendations to users. Our aim here is to let users explore different investment options and educate inexperienced users with new stock that are relevant to their interests. In order to achieve this, we have several modules that are combined together to give recommendations to users.

1. Trending Stocks

   We scrape the internet to gather stocks that are trending in the day. For this, we gather information from a variety of sources, including Reddit and news websites.

2. Category Based Recommendations

   Another recommendation is coming from the categories where the user is already following stocks from. For example, if the user is following BP and ExxonMobil, notably two big oil companies from the US stock market, we recommend Shell and Chevron with this method. This is a good recommendation because we let the user know the other stocks that are within the same sector of the interest of the user.

Scraping Module

In the scraping module, we implemented scraping tasks as asynchronous tasks, which means that the scraping and summarization process runs in the background and does not block the main thread of the backend.

We used Celery workers to execute the asynchronous tasks and celery beat to schedule them. Also, we used Redis as a message broker to establish communication between the Django backend and celery workers. As an advantage of using Celery worker, the tasks are done faster with the multithreading.

There are around 500 stocks recorded and for every hour the Celery Beat sends news scraping tasks for each of them. Also send an additional task every 2 minutes to update the last price information of the stocks (This task only updates the last price information that is displayed on the stock page not for the price graph. Price graphs are created with instant price information, which taken from yfinance API). Collecting and summarizing news for all stocks approximately takes 40 minutes thanks to multithreading. Since we scheduled news scraping tasks for every hour there is a 20-minute safe space for worst-case scenarios.

In news scraping tasks, we scrape news from Yahoo News with yfinance library, Google News with gnews library and we scrape news from Google Finance Search results using beautifulsoup4 library. We fit the scraped information in our database format. We scrape news' title, publisher/source, published date, url and news text. However we do not store the text we just use it to extract the summary and store the summary in the database.

For extracting summaries from the news' texts we used the newspaper3k library for python. It is widely used for summarizing the articles and uses a pre-trained NLP model for summary extraction. After extracting the summary we feed this summary into a sentiment analysis model that gives us labels such as "NEGATIVE" or "POSITIVE" and also a confidence score that is between 0 and 1. We have used a pre-trained Hugging Face model and locally deployed it. If we rather used the available endpoints from Hugging Face APIs, we would hit the quota before we were done scraping all the news, that is the reason we resorted to the option of deploying on our server.

Database - Server

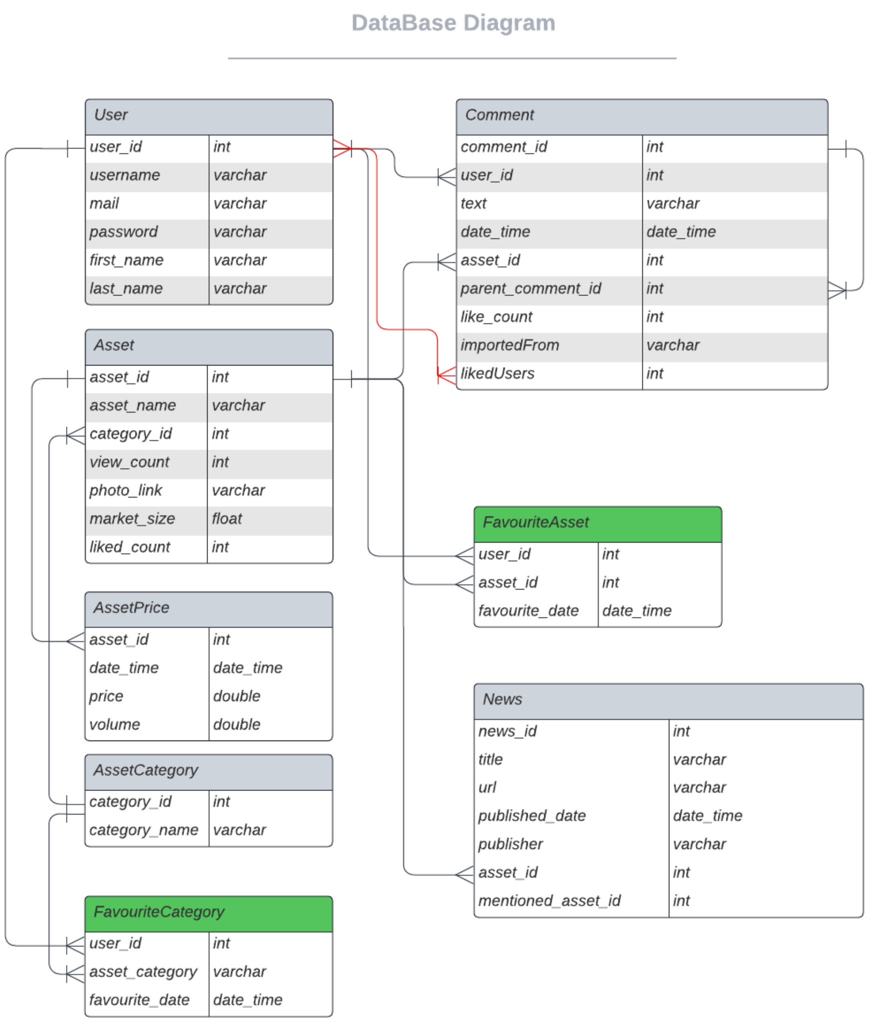


Figure 3.2. DataBase Diagram of Project

In this project, the choice of database was SQLite. The rationale behind this decision was that the project's scope was not expansive enough to necessitate the utilization of a more robust database solution. Furthermore, as the project was deployed on a server, considerations such as cost and compatibility played a crucial role in the selection process. Given these limitations, alternative options such as SQL Server and PostGreSQL were not deemed suitable and thus, not pursued.

The Investsmart model consists of 8 tables, mainly User, Asset, News[Figure 3.2]. User table is directly related to many tables. In this way, functions such as the user's following the asset and making comments are provided. Furthermore, the database design has been crafted with scalability in mind, allowing for easy expansion as the project grows and the need for additional tables arises, ensuring that the database is able to adapt to the evolving needs of the project.

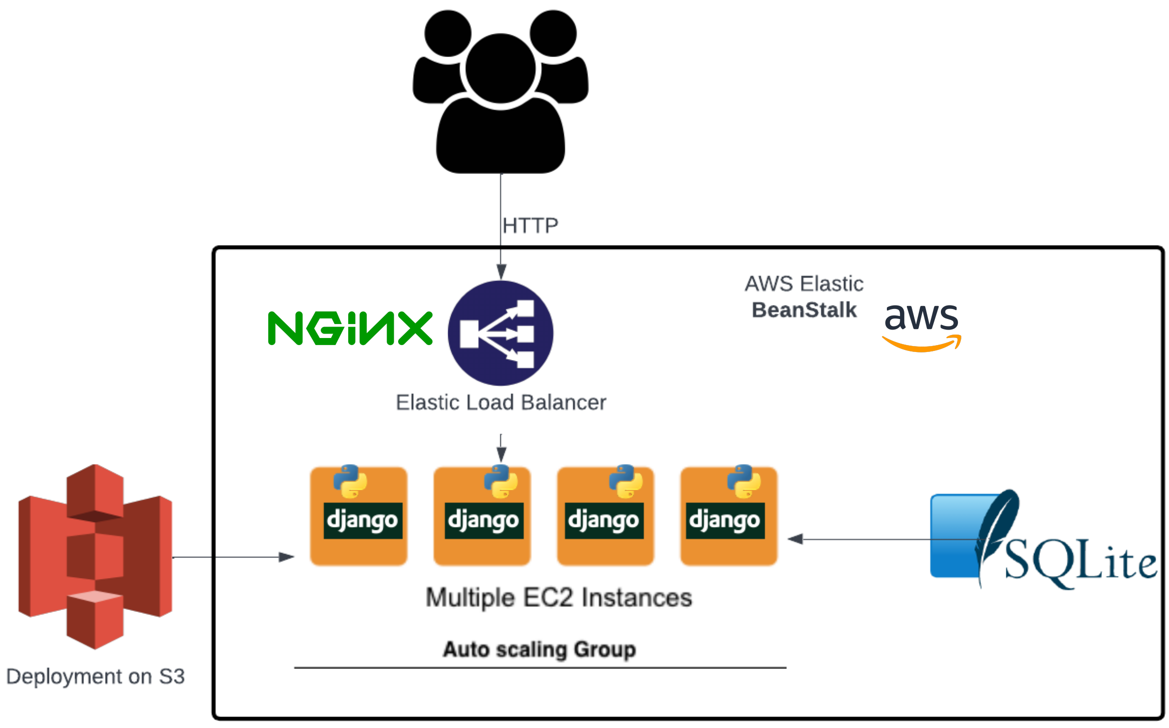
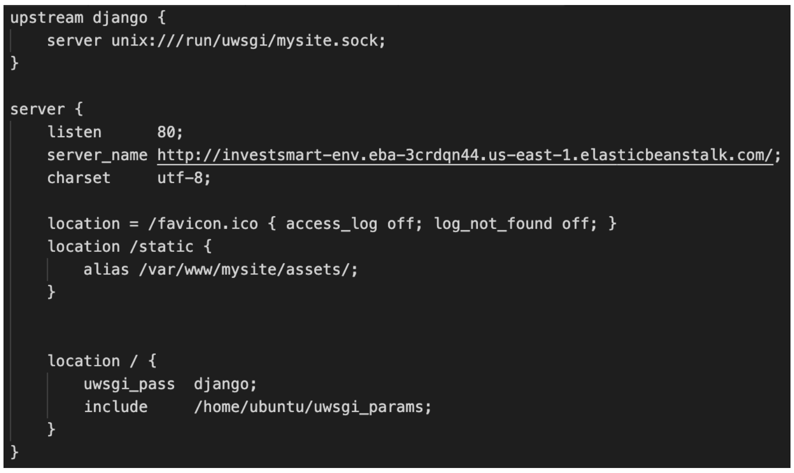


Figure: Server Architecture

This project is a platform that should appeal to multiple users and should work actively. Therefore, even if it works locally in the test environment, it must work on a server when it is transferred to the product environment. Our choice to deploy the project was AWS elastic beanstalk. Because Elastic Beanstalk AWS is a fast and simple way to deploy and manage applications on AWS. It handles the deployment details and automatically scales the application based on need. It also improves developer productivity by managing the infrastructure and keeping the platform updated. For these reasons, running our application on aws server was both fast and easy. Only after configuring nginx and uWSGI modules is the application ready to run.

NGINX is a web server software that can also be configured as a load balancer to improve the availability and efficiency of servers. It is open-source software that can be used for web serving, reverse proxying, caching, load balancing, media streaming, and more. Originally designed for maximum performance and stability, it can also be used as an efficient HTTP load balancer to distribute traffic to multiple application servers, improving the performance, scalability, and reliability of web applications [5].



```
upstream django {
    server unix:///run/uwsgi/mysite.sock;
}

server {
    listen      80;
    server_name http://investsmart-env.eba-3crdqn44.us-east-1.elasticbeanstalk.com/;
    charset     utf-8;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static {
        alias /var/www/mysite/assets/;
    }


    location / {
        uwsgi_pass  django;
        include     /home/ubuntu/uwsgi_params;
    }
}
```

Figure: nginx configuration is required

IV. Analysis and Results

In the end, we delivered a perfectly working Android app and simplified version of the website that is designed to attract investors to the app.

The design of the app and website have met the goal of providing a user-friendly interface and consolidating information from various sources, making it easy for investors to track and analyze the market, and make informed investment decisions. Our project's approach of consolidating news from various sources and providing an active user system for commenting on stocks, following stocks, and engaging in community discussions, differentiated it from other financial websites that mostly provide market prices or financial news.

In addition to the success of the user-friendly interface and consolidation of stock-related information, the InvestSmart project was also successful in its data collection process. The scraper module takes 40 minutes to collect news for each stock and this scraping process is scheduled for every hour, ensuring that users are always provided with the most up-to-date information. This process runs asynchronously in the background, so users won't notice the time spent on collecting the news, making it an efficient aspect of the project design. This efficient data collection process further supports the goal of providing a comprehensive solution for investors to track the market and news related to stocks and assets.

As a team, we faced some challenges, such as the time it took to collect the news and summarize them, and the limitation of targeting only the American stock exchange market, but we were able to find solutions to overcome them: We used Celery workers to speed up the scraping news every hour via multithreading. With a faster scraping module we ensure that users are always provided with the most up-to-date information. Also, we designed the platform in such a way that it can be easily expanded to other stock markets. One of the challenges we faced during the implementation of the Android app was the long waiting times for the loading of the news and prices. We solved this issue by using background threads and not blocking the user by just showing a loading spinner in the place of the data being loaded. Another issue was the long start-up time of the app which is related to the previous point. We were able to solve this problem by making the data-heavy requests after the initial set-up process is completed.

V. Conclusion

In conclusion, the InvestSmart project aims to provide a comprehensive solution for investors to track the market and news related to stocks and assets. The project includes a mobile app and website, with the mobile app being the primary component, offering various features and tools for analysis and decision-making. The project is an end-to-end software development, with a focus on providing a user-friendly interface and consolidating information from various sources.

Target group of our project were the young investors who rely on finance websites for market updates and aims to provide a platform where users can access market insights and engage in community discussions. The project is designed to be a dynamic platform where targeted stock markets (initially the American stock exchange market) can be expanded easily. Unlike other financial news sites such as The Economist, MarketWatch, Financial Times, and Investing.com, InvestSmart consolidates news from various sources and features an active user system where users can comment on stocks, follow stocks by connecting to their accounts and engage in community discussions. We adopted this approach to differentiate it from other financial websites that mostly provide market prices or financial news.

Overall, the InvestSmart app and website achieved its goal of providing a comprehensive solution for investors to track the market and news related to stocks and assets. The user-friendly interface, mobile optimization, and robust backend, allows InvestSmart to provide a complete solution for beginner investors to make informed decisions, all from the convenience of their mobile or web device.

VI. References

[1] Hodge, F., & Pronk, M. (2006). The Impact of Expertise and Investment Familiarity on Investors’ Use of Online Financial Report Information. Journal of Accounting, Auditing & Finance, 21(3), 267–292. https://doi.org/10.1177/0148558X0602100304

[2] Smart reasons why financial websites should have blogs. CreativeAdviser. (2013, September 12). Available: https://creativeadviser.co.uk/financial-websites/smart-reasons-why-financial-websites-should-have-blogs/ [Accessed on: Oct. 19, 2022]

[3] Valle-Cruz, D., Fernandez-Cortez, V., López-Chau, A. et al. Does Twitter Affect Stock Market Decisions? Financial Sentiment Analysis During Pandemics: A Comparative Study of the H1N1 and the COVID-19 Periods. Cogn Comput 14, 372–387 (2022). https://doi.org/10.1007/s12559-021-09819-8

[4] Richards, T. (2014). Investing psychology: The effects of behavioral finance on Investment Choice and bias + website. Wiley.

[5] Using Nginx as HTTP load balancer, *nginx*. [Online]. Available: https://nginx.org/en/docs/http/load_balancing.html. [Accessed: 16-Jan-2023].

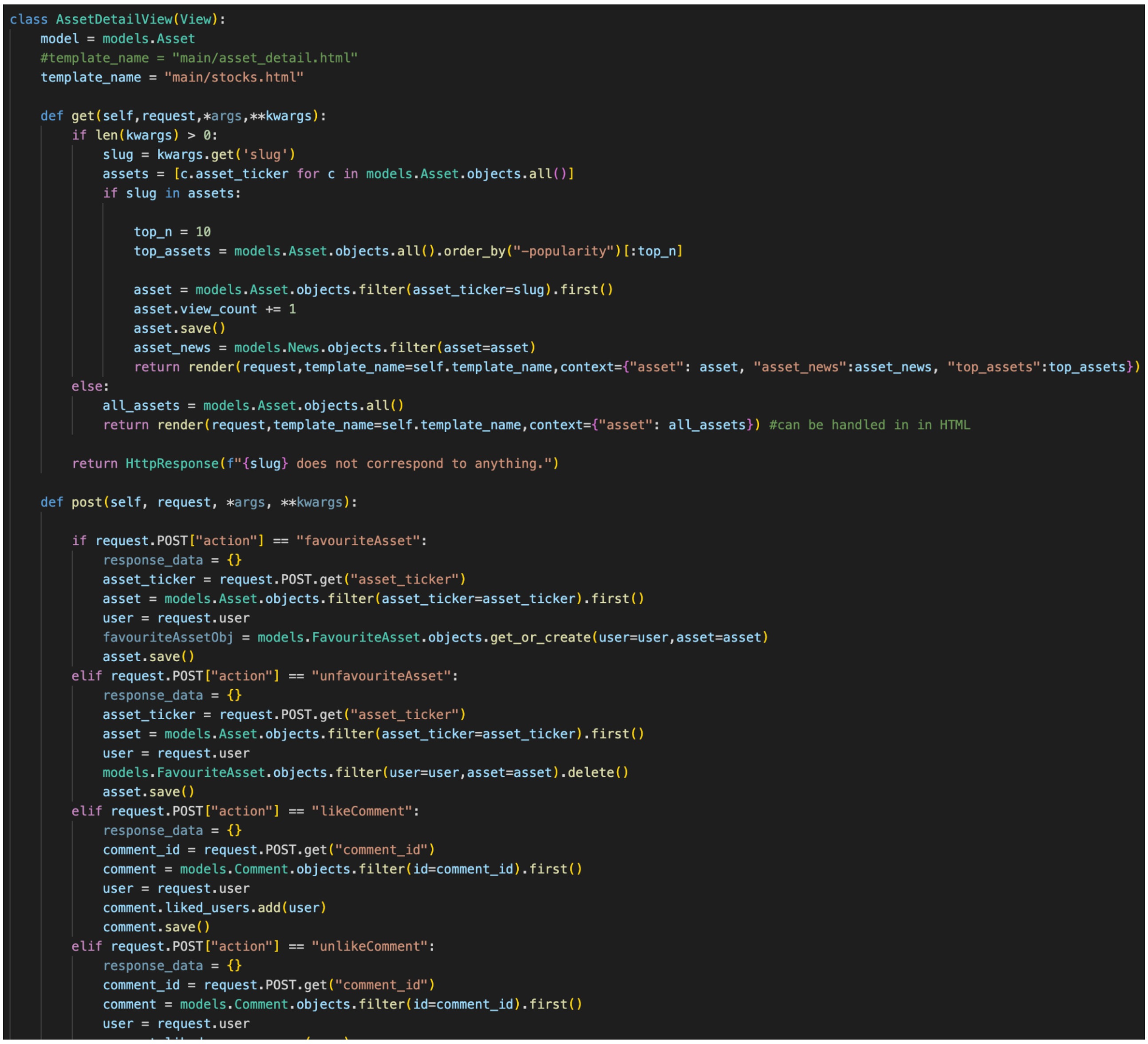
[6] F. Shadhin, The MVT Design Pattern of Django, *Medium*, 13-Sep-2021. [Online]. Available: https://python.plainenglish.io/the-mvt-design-pattern-of-django-8fd47c61f582. [Accessed: 16-Jan-2023].

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, 2018. doi: 10.48550/ARXIV.1810.04805.

VII. Appendix

APPENDIX A

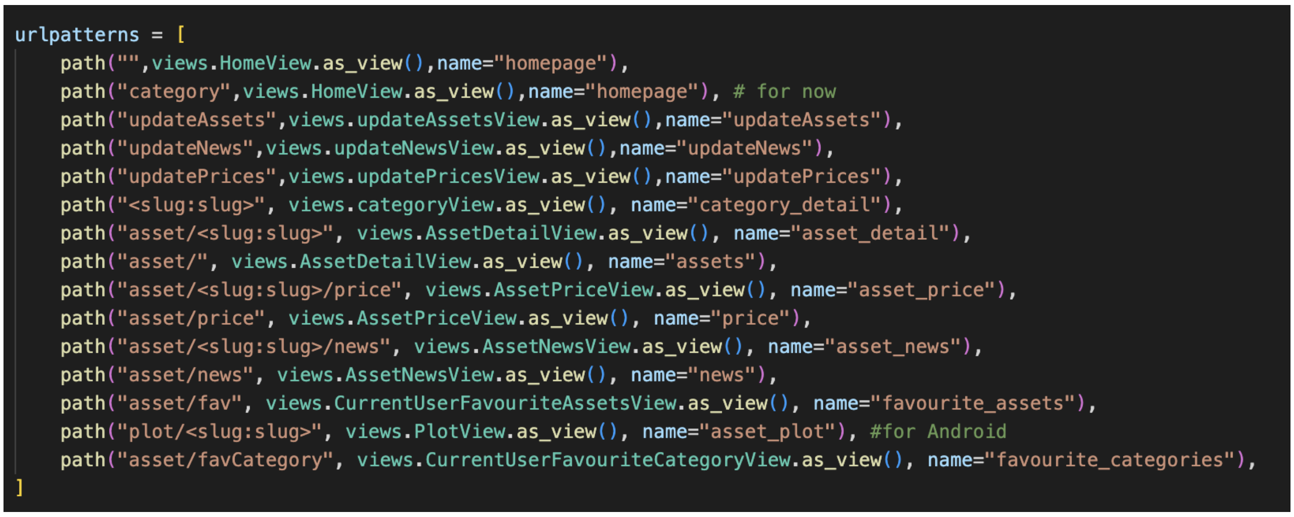
Django Model View Sample Code from our Application



```python
class AssetDetailView(View):
    model = models.Asset
    #template_name = "main/asset_detail.html"
    template_name = "main/stocks.html"

    def get(self,request,*args,**kwargs):
        if len(kwargs) > 0:
            slug = kwargs.get('slug')
            assets = [c.asset_ticker for c in models.Asset.objects.all()]
            if slug in assets:

                top_n = 10
                top_assets = models.Asset.objects.all().order_by("-popularity")[:top_n]

                asset = models.Asset.objects.filter(asset_ticker=slug).first()
                asset.view_count += 1
                asset.save()
                asset_news = models.News.objects.filter(asset=asset)
                return render(request,template_name=self.template_name,context={"asset": asset, "asset_news":asset_news, "top_assets":top_assets})
        else:
            all_assets = models.Asset.objects.all()
            return render(request,template_name=self.template_name,context={"asset": all_assets}) #can be handled in in HTML

        return HttpResponse(f"{slug} does not correspond to anything.")

    def post(self, request, *args, **kwargs):

        if request.POST["action"] == "favouriteAsset":
            response_data = {}
            asset_ticker = request.POST.get("asset_ticker")
            asset = models.Asset.objects.filter(asset_ticker=asset_ticker).first()
            user = request.user
            favouriteAssetObj = models.FavouriteAsset.objects.get_or_create(user=user,asset=asset)
            asset.save()
        elif request.POST["action"] == "unfavouriteAsset":
            response_data = {}
            asset_ticker = request.POST.get("asset_ticker")
            asset = models.Asset.objects.filter(asset_ticker=asset_ticker).first()
            user = request.user
            models.FavouriteAsset.objects.filter(user=user,asset=asset).delete()
            asset.save()
        elif request.POST["action"] == "likeComment":
            response_data = {}
            comment_id = request.POST.get("comment_id")
            comment = models.Comment.objects.filter(id=comment_id).first()
            user = request.user
            comment.liked_users.add(user)
            comment.save()
        elif request.POST["action"] == "unlikeComment":
            response_data = {}
            comment_id = request.POST.get("comment_id")
            comment = models.Comment.objects.filter(id=comment_id).first()
            user = request.user
```

APPENDIX B

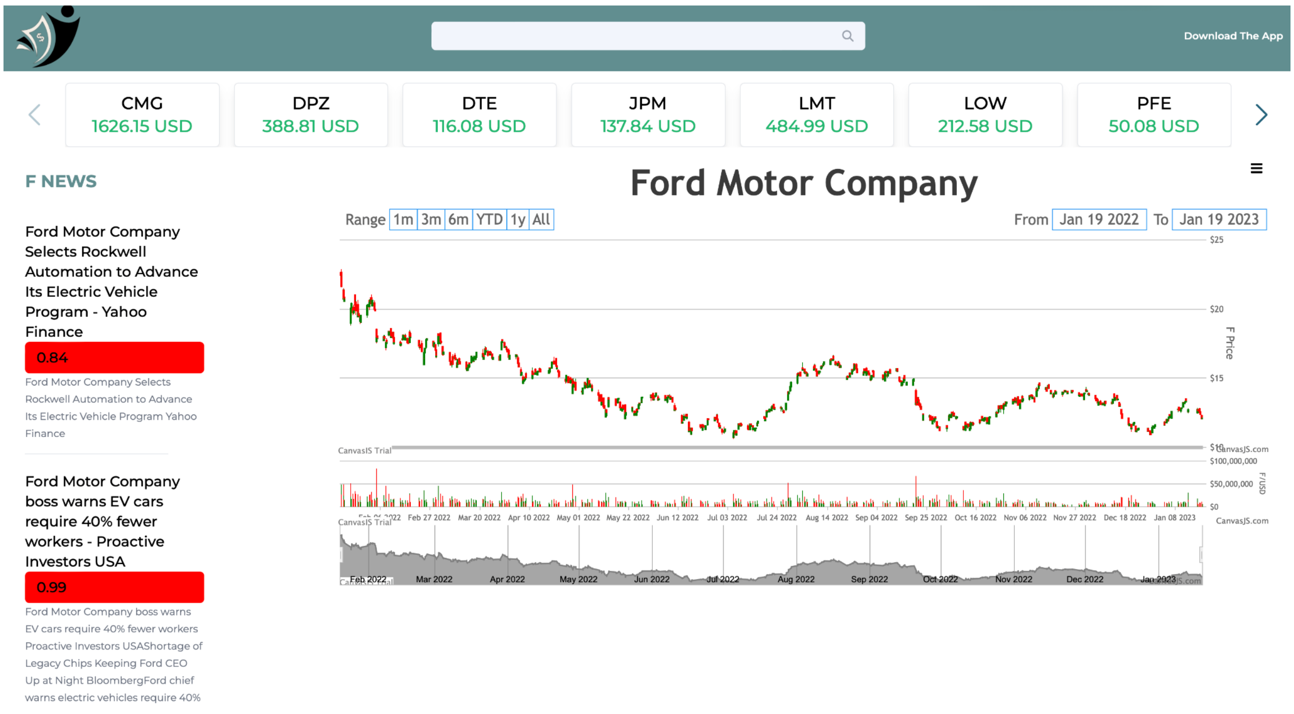
Django url patterns Code



```python
urlpatterns = [
    path("",views.HomeView.as_view(),name="homepage"),
    path("category",views.HomeView.as_view(),name="homepage"), # for now
    path("updateAssets",views.updateAssetsView.as_view(),name="updateAssets"),
    path("updateNews",views.updateNewsView.as_view(),name="updateNews"),
    path("updatePrices",views.updatePricesView.as_view(),name="updatePrices"),
    path("<slug:slug>", views.categoryView.as_view(), name="category_detail"),
    path("asset/<slug:slug>", views.AssetDetailView.as_view(), name="asset_detail"),
    path("asset/", views.AssetDetailView.as_view(), name="assets"),
    path("asset/<slug:slug>/price", views.AssetPriceView.as_view(), name="asset_price"),
    path("asset/price", views.AssetPriceView.as_view(), name="price"),
    path("asset/<slug:slug>/news", views.AssetNewsView.as_view(), name="asset_news"),
    path("asset/news", views.AssetNewsView.as_view(), name="news"),
    path("asset/fav", views.CurrentUserFavouriteAssetsView.as_view(), name="favourite_assets"),
    path("plot/<slug:slug>", views.PlotView.as_view(), name="asset_plot"), #for Android
    path("asset/favCategory", views.CurrentUserFavouriteCategoryView.as_view(), name="favourite_categories"),
]
```

APPENDIX C

Website Screenshots



F NEWS

Ford Motor Company

Ford Motor Company Selects Rockwell Automation to Advance Its Electric Vehicle Program - Yahoo Finance

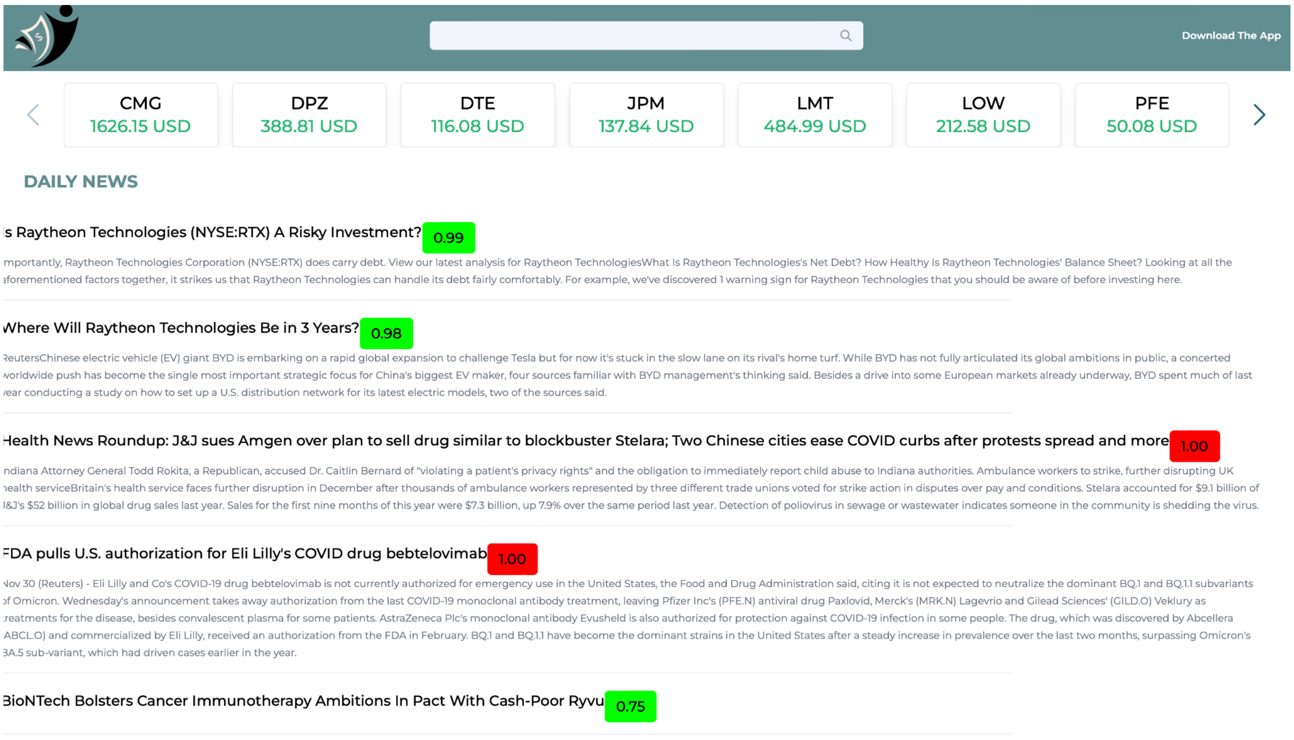
0.84

Ford Motor Company Selects Rockwell Automation to Advance Its Electric Vehicle Program Yahoo Finance

Ford Motor Company boss warns EV cars require 40% fewer workers - Proactive Investors USA

0.99

Ford Motor Company boss warns EV cars require 40% fewer workers Proactive Investors USAShortage of Legacy Chips Keeping Ford CEO Up at Night BloombergFord chief warns electric vehicles require 40%



DAILY NEWS

s Raytheon Technologies (NYSE:RTX) A Risky Investment? 0.99

mportantly, Raytheon Technologies Corporation (NYSE:RTX) does carry debt. View our latest analysis for Raytheon TechnologiesWhat Is Raytheon Technologies's Net Debt? How Healthy Is Raytheon Technologies' Balance Sheet? Looking at all the aforementioned factors together, it strikes us that Raytheon Technologies can handle its debt fairly comfortably. For example, we've discovered 1 warning sign for Raytheon Technologies that you should be aware of before investing here.

Where Will Raytheon Technologies Be in 3 Years? 0.98

ReutersChinese electric vehicle (EV) giant BYD is embarking on a rapid global expansion to challenge Tesla but for now it's stuck in the slow lane on its rival's home turf. While BYD has not fully articulated its global ambitions in public, a concerted worldwide push has become the single most important strategic focus for China's biggest EV maker, four sources familiar with BYD management's thinking said. Besides a drive into some European markets already underway, BYD spent much of last year conducting a study on how to set up a U.S. distribution network for its latest electric models, two of the sources said.

Health News Roundup: J&J sues Amgen over plan to sell drug similar to blockbuster Stelara; Two Chinese cities ease COVID curbs after protests spread and more 1.00

ndiana Attorney General Todd Rokita, a Republican, accused Dr. Caitlin Bernard of "violating a patient's privacy rights" and the obligation to immediately report child abuse to Indiana authorities. Ambulance workers to strike, further disrupting UK health serviceBritain's health service faces further disruption in December after thousands of ambulance workers represented by three different trade unions voted for strike action in disputes over pay and conditions. Stelara accounted for $9.1 billion of J&J's $52 billion in global drug sales last year. Sales for the first nine months of this year were $7.3 billion, up 7.9% over the same period last year. Detection of poliovirus in sewage or wastewater indicates someone in the community is shedding the virus.

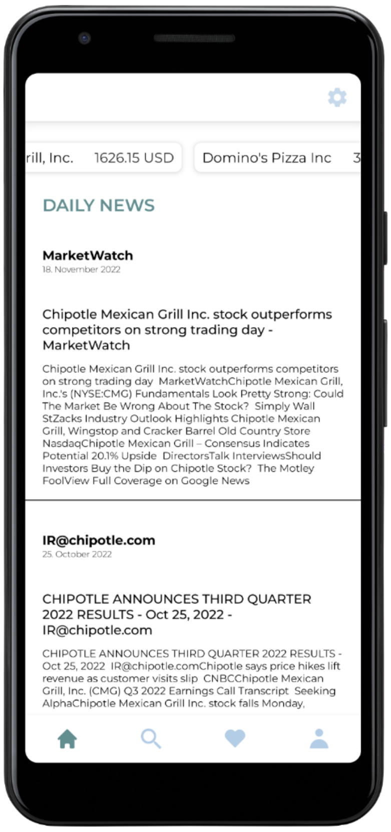
FDA pulls U.S. authorization for Eli Lilly's COVID drug bebtelovimab 1.00

Nov 30 (Reuters) - Eli Lilly and Co's COVID-19 drug bebtelovimab is not currently authorized for emergency use in the United States, the Food and Drug Administration said, citing it is not expected to neutralize the dominant BQ.1 and BQ.1.1 subvariants of Omicron. Wednesday's announcement takes away authorization from the last COVID-19 monoclonal antibody treatment, leaving Pfizer Inc's (PFE.N) antiviral drug Paxlovid, Merck's (MRK.N) Lagevrio and Gilead Sciences' (GILD.O) Veklury as treatments for the disease, besides convalescent plasma for some patients. AstraZeneca Plc's monoclonal antibody Evusheld is also authorized for protection against COVID-19 infection in some people. The drug, which was discovered by Abcellera (ABCL.O) and commercialized by Eli Lilly, received an authorization from the FDA in February. BQ.1 and BQ.1.1 have become the dominant strains in the United States after a steady increase in prevalence over the last two months, surpassing Omicron's BA.5 sub-variant, which had driven cases earlier in the year.

BioNTech Bolsters Cancer Immunotherapy Ambitions In Pact With Cash-Poor Ryvu 0.75

APPENDIX D

Android Application Screenshots


rill, Inc. 1626.15 USD | Domino's Pizza Inc 3

DAILY NEWS

MarketWatch
18. November 2022

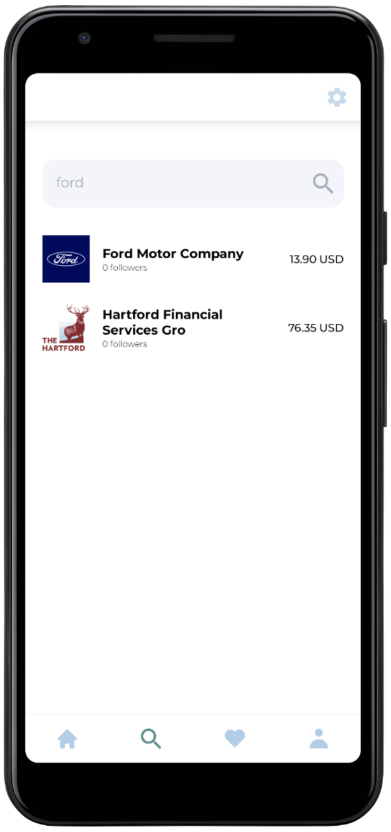
Chipotle Mexican Grill Inc. stock outperforms competitors on strong trading day - MarketWatch

Chipotle Mexican Grill Inc. stock outperforms competitors on strong trading day MarketWatchChipotle Mexican Grill, Inc.'s (NYSE:CMG) Fundamentals Look Pretty Strong: Could The Market Be Wrong About The Stock? Simply Wall StZacks Industry Outlook Highlights Chipotle Mexican Grill, Wingstop and Cracker Barrel Old Country Store NasdaqChipotle Mexican Grill – Consensus Indicates Potential 20.1% Upside DirectorsTalk InterviewsShould Investors Buy the Dip on Chipotle Stock? The Motley FoolView Full Coverage on Google News

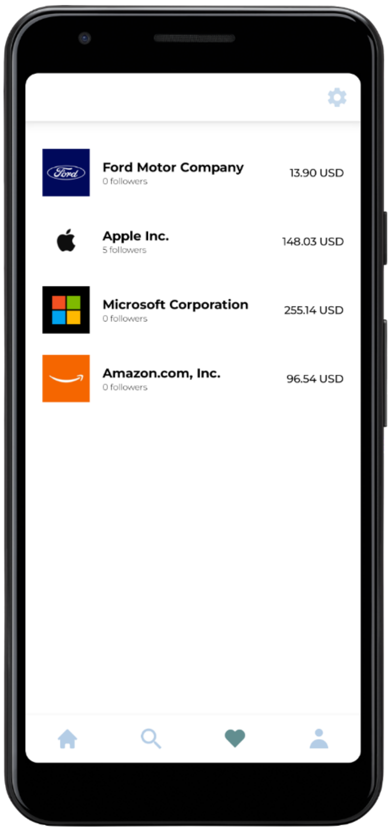
IR@chipotle.com
25. October 2022

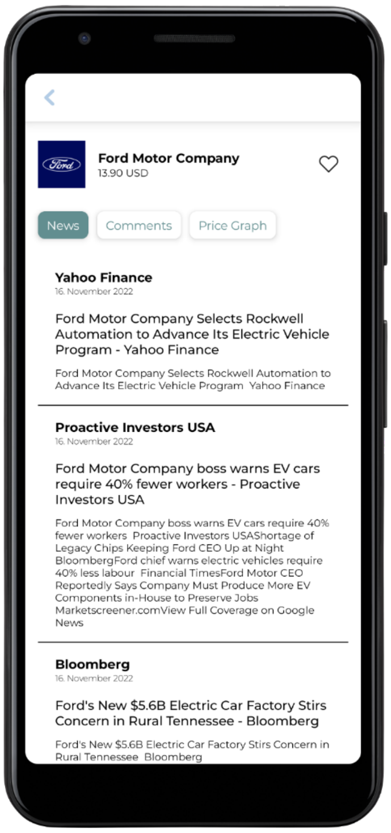
CHIPOTLE ANNOUNCES THIRD QUARTER 2022 RESULTS - Oct 25, 2022 - IR@chipotle.com

CHIPOTLE ANNOUNCES THIRD QUARTER 2022 RESULTS - Oct 25, 2022 IR@chipotle.comChipotle says price hikes lift revenue as customer visits slip CNBCChipotle Mexican Grill, Inc. (CMG) Q3 2022 Earnings Call Transcript Seeking AlphaChipotle Mexican Grill Inc. stock falls Monday,


ford

Ford Motor Company 0 followers	13.90 USD
Hartford Financial Services Gro 0 followers	76.35 USD


Ford Motor Company 0 followers	13.90 USD
Apple Inc. 5 followers	148.03 USD
Microsoft Corporation 0 followers	255.14 USD
Amazon.com, Inc. 0 followers	96.54 USD


Ford Motor Company
13.90 USD

Yahoo Finance
16. November 2022

Ford Motor Company Selects Rockwell Automation to Advance Its Electric Vehicle Program - Yahoo Finance

Ford Motor Company Selects Rockwell Automation to Advance Its Electric Vehicle Program Yahoo Finance

Proactive Investors USA
16. November 2022

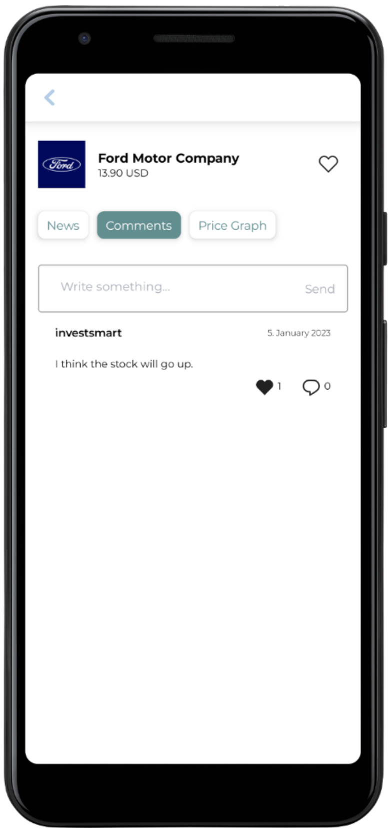
Ford Motor Company boss warns EV cars require 40% fewer workers - Proactive Investors USA

Ford Motor Company boss warns EV cars require 40% fewer workers Proactive Investors USAShortage of Legacy Chips Keeping Ford CEO Up at Night BloombergFord chief warns electric vehicles require 40% less labour Financial TimesFord Motor CEO Reportedly Says Company Must Produce More EV Components in-House to Preserve Jobs Marketscreener.comView Full Coverage on Google News

Bloomberg
16. November 2022

Ford's New $5.6B Electric Car Factory Stirs Concern in Rural Tennessee - Bloomberg

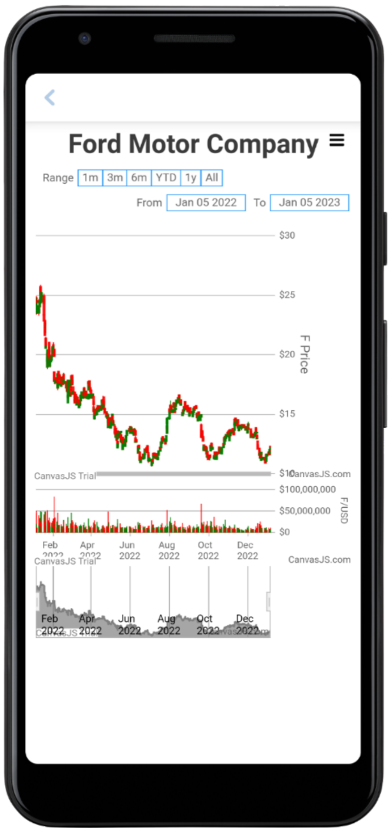
Ford's New $5.6B Electric Car Factory Stirs Concern in Rural Tennessee Bloomberg


Ford Motor Company
13.90 USD

investsmart 5. January 2023

I think the stock will go up.

1 0



Here are the links for the Github repositories:

- Android App - https://github.com/Invest-Smart-COMP-491/invest-smart-android
- Web/Backend - https://github.com/Invest-Smart-COMP-491/invest-smart-backend-web