

CFG: Nanodegree Software Specialisation (Spring 2023): Group Project Report

Stock tracker

Team Members:

Miriem Shaimi, Jessica Eichel, Abigael Taiwo, Karolina Cisek-Ndlovu, Asya Seagrave

1. INTRODUCTION:

Stock tracker, a dynamic Flask application designed to allow users effortless access to visual information and news updates for their selected stocks. This report is created to provide a comprehensive overview of planning, features, and the structure of our project, as well as its outcomes and reflections on its usability.

1.1 Aims and Objectives

It is aimed to provide users with a platform for monitoring and tracking stocks and enable users to visualise stock performance through interactive graphs. News page would allow users to get quick access to the most current news related to users' preferred stock.

Objectives:

- Integrate stock data API to fetch real-time stock information for tracking and display on the application.
- Integrate news API to fetch relevant news updates related to users' preferred stocks.
- Employ agile development methodologies, such as Kanban and Dynamic Systems Development Method.
- Create and configure a relational database to store user information, stock data, and user preferences.
- Implement user registration and login functionality to secure user profiles and enable personalised experiences.
- Design and develop a user interface that is intuitive, responsive, and visually appealing, allowing users to interact with stock data and news.
- Develop a search feature to allow users to add stocks to their portfolios and track their performance.
- Create interactive graphs to visually represent stock performance over time using the fetched stock data.
- Implement algorithms or filters to tailor news updates based on users' preferred stocks, utilising the fetched news data.
- Develop a notification system that allows users to customise their notification preferences.
- Conduct thorough testing to ensure the functionality, performance, and security of the application, including testing the integration with both APIs.
- Expand the functionality of the resources page to increase its educational value for the users of the application.

1.2. Roadmap of the report

1. **Introduction** is essentially an overview of the project, defines its aims and objectives, as well as outlines the structure of the report.
2. **Background** presents our considerations when choosing the project, discusses the increasing popularity of stock tracking, and explains the functionality and features of the app.
3. **Specifications and design** outline functional and non-functional requirements of the application and contain a description of the design and architecture of the application.
4. **Implementation and execution** describe the collaboration methods used by the team, the Agile development techniques we have adapted, the way responsibilities were divided between team members, the tools and libraries used in the project, and the way the team addressed challenges and issues.
5. **Testing and evaluation** go into detail about the testing strategy employed to ensure the functionality and security of the application, describe different types of testing conducted, and limitations identified upon completion of the testing process.
6. **Conclusion** is a summary of key project outcomes and what we as a team achieved completing it.

2. BACKGROUND

During initial discussions of our project, the idea of some kind of educational financial app with possible functionality for savings tracking was almost selected as the main one for the project, but as we were discussing the project implementation, its scale and opportunities for using API, our idea morphed into creating a stock tracker. We have all agreed it is a project that could have a lot of potential to be expanded in the future into a more versatile financial tool, but concentrating on stock tracking allowed us to create something visually appealing and functional for users, as well as presented opportunity for most of us to learn ways to build systems we have previously navigated only as users. A stock tracker can be a valuable tool for individuals interested in monitoring and managing their investments.

The increasing popularity of investing in the stock market means there is a growing demand for tools that help individuals effectively track their selected stock. We can see our app as a good tool for someone who is maybe not confident enough yet to start investing but wanted to keep track of the selection of stock to see how well their selection performs over time, which would allow them to analyse what they need to be aware of while choosing a stock, how well stock in their selection recovers, how much price of their selected stock depends on the world events.

To allow for all above mentioned, we implemented the following key features:

- User Registration and Login Interface: our app begins with a user login/registration interface, ensuring that users can create accounts securely and access personalised features.
- Profile page is where users can select their stock and their preferred language, add their stock selection to their portfolio and get quick access to news and graphs by activating widgets placed next to their stock. At the top of the page, we placed navigation tabs for the home page including Profile, Graphs, and News with the option to logout in the top right corner.

- Stock Details page with graphs to give users an instrument to track their stock performance over a period of time, clear and easy-to-read price change indicators are located above the graph. The “Back to profile” graphical element was added to both Graphs and News pages to facilitate seamless navigation.
- Resources page allows users to select language and stock to see the news related to it, as well as contains a few stock related educational videos and terminology. The “Read more” hyperlink under articles allows the user to access the full content of the article on the website where it was originally published.

We think within the scope of our app we have built a rather strong foundation and with time and effort we could expand it into something more substantial, which means after completing this course we as a group can still use it as a learning tool adding new pages, more functionality and expanding its security features.

3. SPECIFICATIONS AND DESIGN.

3.1. Functional vs non-functional requirements

Functional requirements for the app:

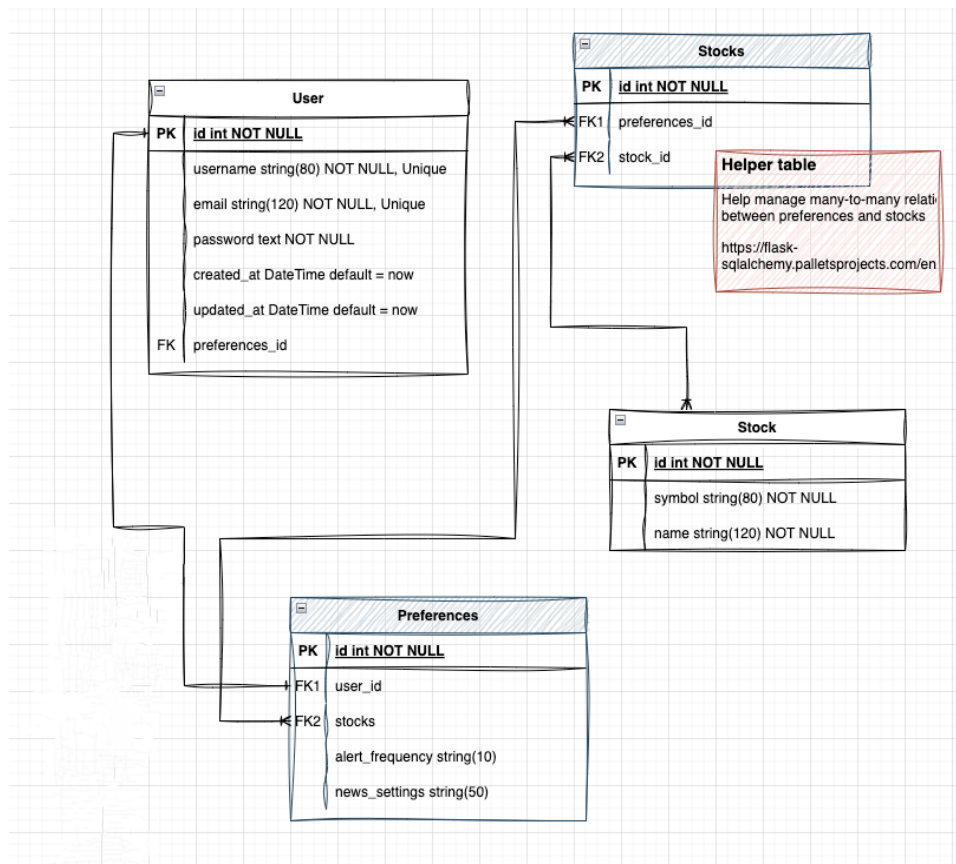
- Allows the user to register and log into their page securely.
- Allows the user to browse stock and select the shares they would like to add to view in their profile.
- Generates clear and concise graphs to visually represent historical stock performance.
- Provides users with access to relevant news and updates, including the ability to filter news by language.
- Includes some educational articles and video materials for app users with beginner level of stock knowledge.
- Allows users to set up regular personalised alerts and notifications.
- Has features for quick access and easy navigation

Non-functional requirements:

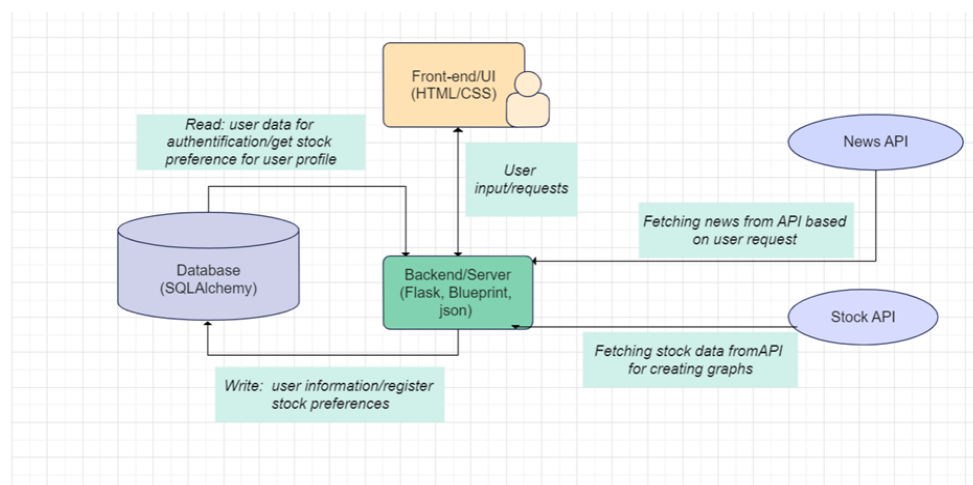
- The app needs to be stable and reliable
- The app should be designed to handle a possible increase in the number of users and have a reliable database
- Code structure should be easy to read, modular, and open for future enhancements and development
- The app should generate graphs and provide news without significant delays
- User data should be stored securely, using industry-standard security measures.
- The interface should be intuitive and user-friendly, and its design needs to be appealing and consistent throughout the app

3.2 Design and Architecture

We have implemented a database using the SQLAlchemy Python library. The process of database integration into the Flask framework is straightforward and seamless, which not only accelerated our development and testing process but also provided a solid foundation for the future scalability and maintainability of our project. Below is the relationship diagram for our database.

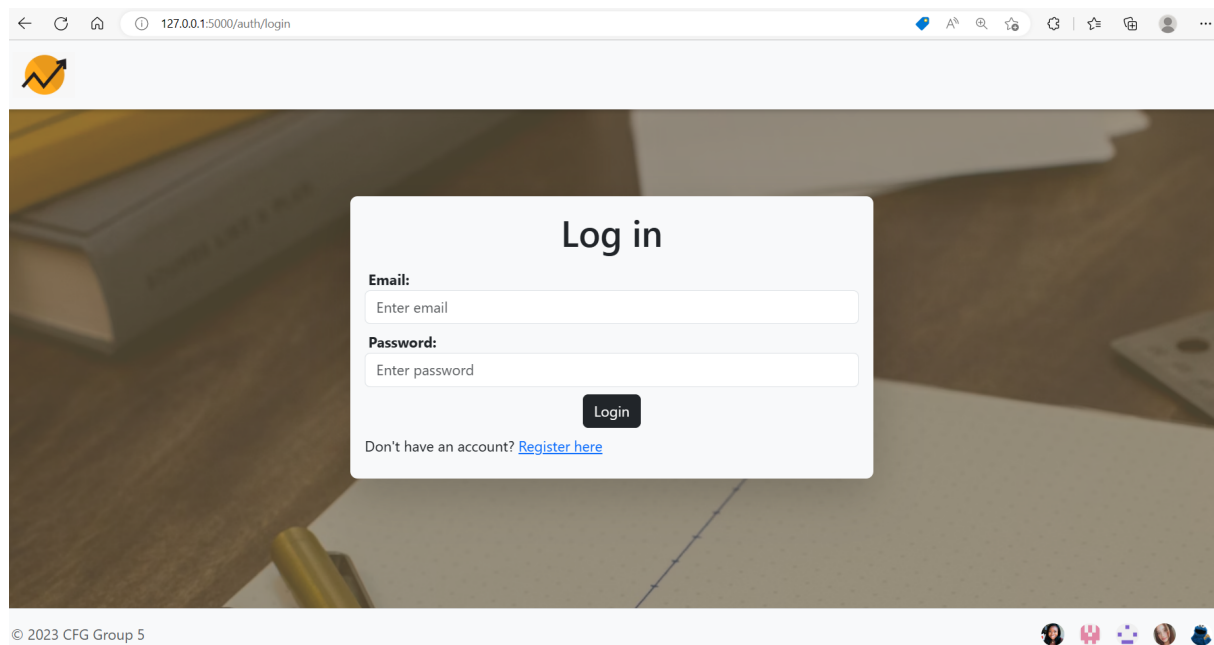


Developing a functioning database was one of the major milestones for our app, as most of the application's functionality was built around it. After that we concentrated on the architecture of the rest of the application. The diagram below represents how the elements of the system within our app interact:

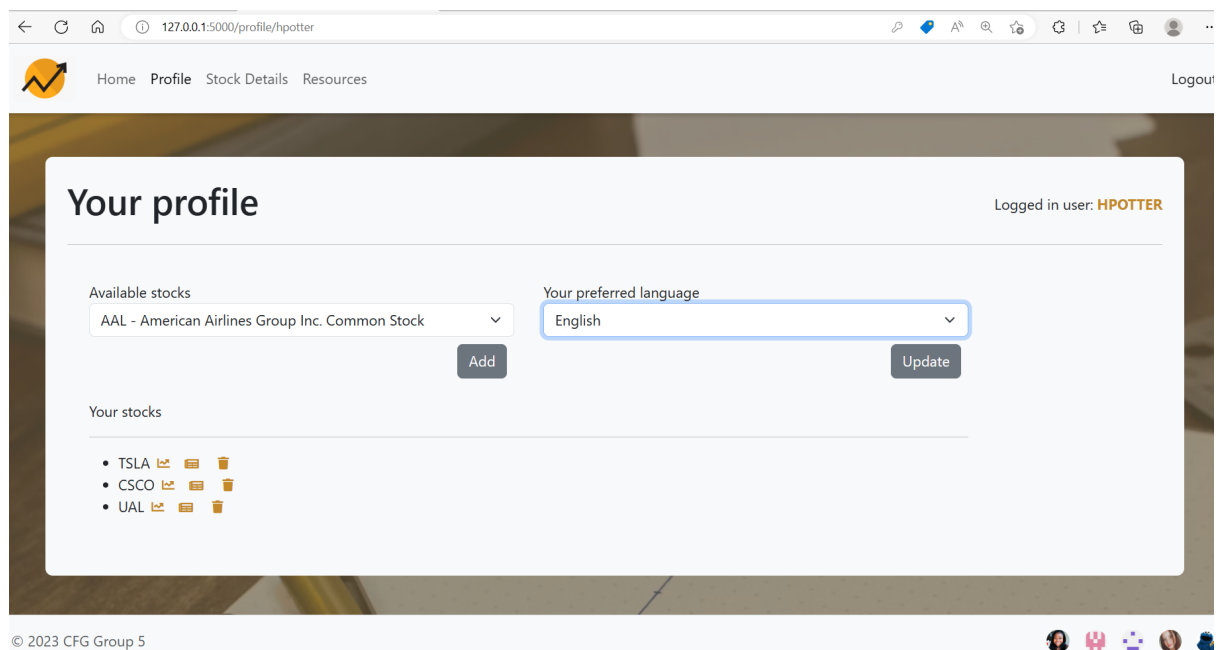


Having a responsive, visually appealing, and easy-to-navigate interface was important for us and the team discussed a lot of elements: style, colour palette, navigation elements, tabs, and background choices. So below we include a few visuals of what the pages of our application look like at the moment of submission.

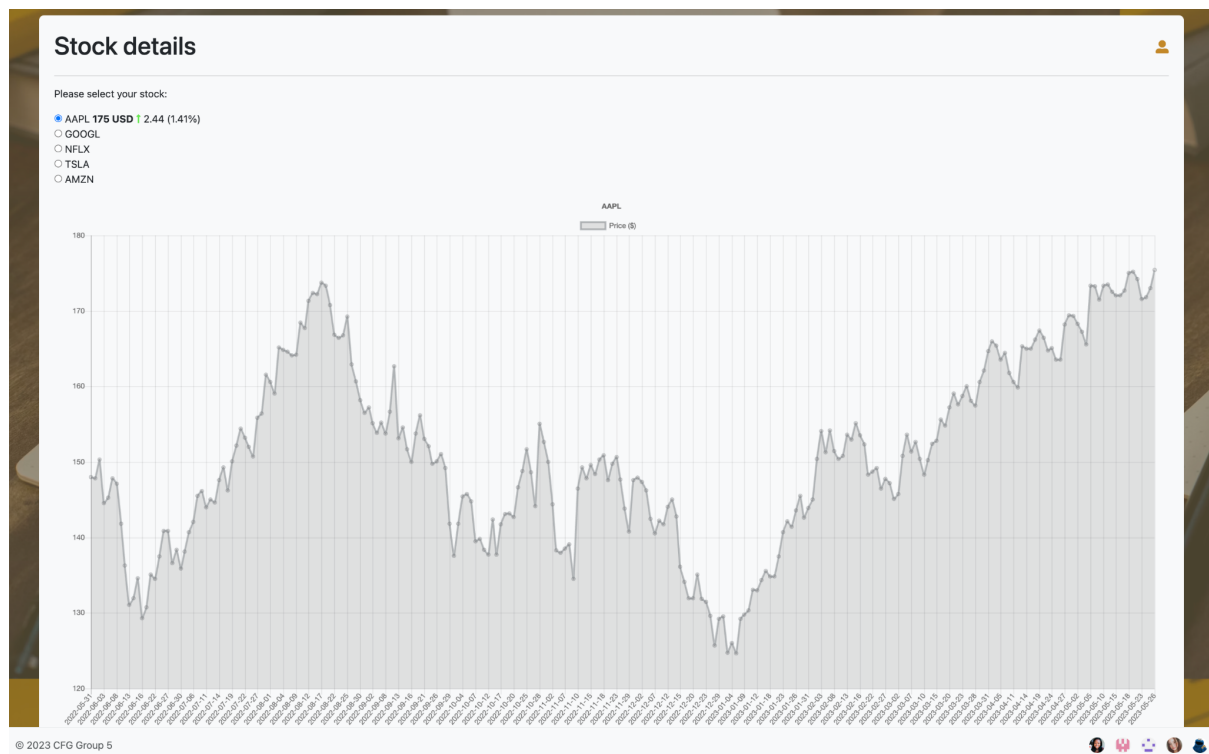
Login page:



Profile page:



Stock details page:



Resources page:

Document

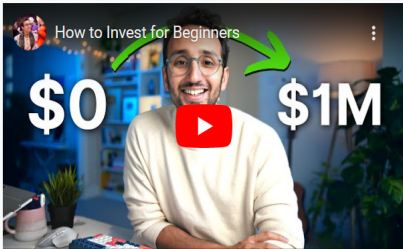
127.0.0.1:5000/profile/hpotter/news

HomeProfileStock DetailsResources

Logout

Resources

How to Invest for Beginners

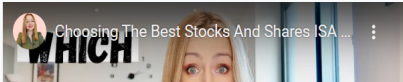


source: investing.com | 2023-05-28

Published May 28, 2023 07:41AM ET © Reuters. F +6.24% Add to/Remove from Watchlist
TSLA +4.72% Add to/Remove from Watchlist STLA +1.60% Add to/Remove from Wa...

[Read more...](#)

Choosing The Best Stocks And Shares ISA UK

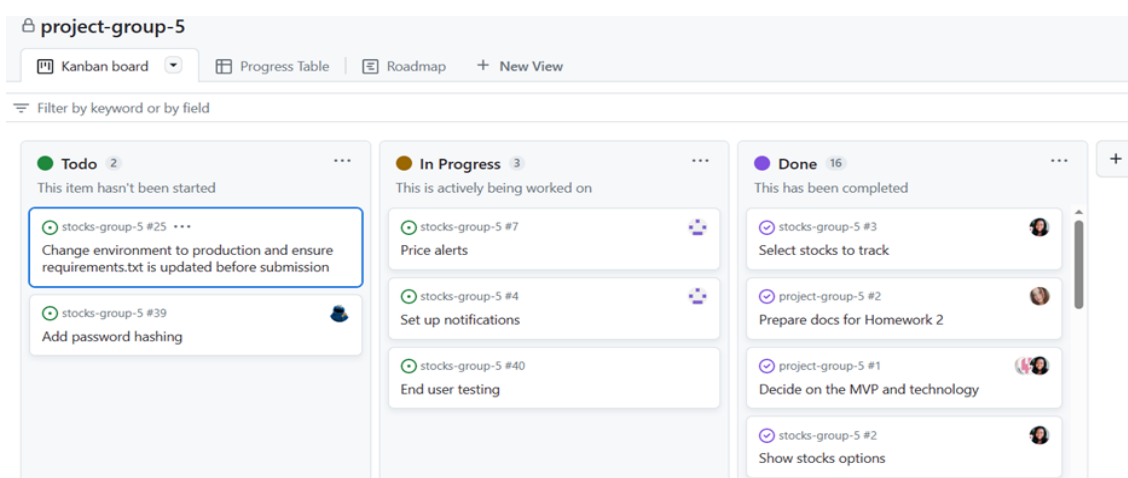


© 2023 CFG Group 5

4. IMPLEMENTATION AND EXECUTION:

4.1. Development approach and team member roles

Early on in the project the Agile approach was adopted, initial discussions were rather unstructured, but by third meeting we switched to having sprint planning/sprint review kind of meetings and used slack for daily discussions on what progress has been made and what challenges team members are facing, a team member with most experience was selected to facilitate meetings and discussions and ensure team stays on track. The Dynamic Systems Development Method was used, it consisted of dividing the project into phases or fixed duration and scale, concentrating on achieving working functionality on early stages and building up on it, all accompanied by project reviews and testing. Kanban board was set up on GitHub:



Team members selected parts of the project independently, analysing their strengths and knowing that other team members will support them. Jessica took responsibility for the initial project brief, and its draft functionality coding layout, set up the first version of the project repository, and wrote code for the notification system. Karolina facilitated meetings, set up the main GitHub repository, and managed it as the code was added, she is responsible for a significant amount of code within the application for example profile page, navigation system, Resources page setup, database architecture, style and design of the application. Miriem did the majority of work organising planning systems (Kanban board, Miro boards), she got the information together for project idea submission, created code for Graphs, ensured visualisations are clear and appealing, and carried out the majority of the app testing. Abigael was responsible for the user's stock selections and removals in the user database that would then tailor the information presented to the user. She enhanced the graphs by adding the ability to track the daily stock price changes. She also significantly expanded the functionality of the resources page via the curation and display of educational videos. Early on it was evident that the list of stocks we had was too large so she created a shortlist based on the S&P 500. Asya set up a user authentication system, including error messages for users, and integrated a user database, implemented a session management system, created frontend views for pages associated with authentication, and prepared project documentation for submission.

4.2. . Tools and libraries

Developing our app allowed us to use:

- flask – for creating a web framework for our app
- Blueprint – a feature provided by Flask, we used it for organising related routes and static files
- sqlalchemy, flask_sqlalchemy and sqlalchemy.orm – for database access and manipulation
- json – for working with JSON data
- datetime – for working with dates and times
- enum – for enumerations
- urllib.parse – for parsing and manipulating URLs
- os – for managing operating system dependent functionality
- http.client – module to provide an HTTP client for making requests to web services
- session – Flask object for storing user session data
- request – Flask object for accessing request data
- render_template – Flask function for rendering HTML templates
- Bootstrap – for pre-styled components of web-page and responsive layout
- Pytest - for testing the app

4.3. Implementation process

After the initial setup and establishment of version control, the implementation of the project followed a systematic approach. The team divided the application into different pages and functionalities, allowing each member to focus on their assigned tasks. As each code submission was made, thorough testing was conducted to ensure the correctness and functionality of the implemented features. Through the process of testing and merging, any gaps or issues that were identified were promptly addressed and corrected. The team adopted an iterative development approach, gradually building up the functionality of the application as each component started working more reliably. For instance, in the Stock Details section, additional features such as price change arrows were incorporated to enhance the visualisation of data. Similarly, the news page was transformed into a comprehensive Resources page, offering a wider range of information to users. This incremental development process allowed for continuous improvement and refinement of the application's features, resulting in a more robust and comprehensive final product.

4.4. Challenges

Throughout the project, several team members encountered challenges during the development of their respective code sections. One of the first ones was the decision to switch from using an external separate database in MySQL Workbench to utilising SQLAlchemy. Although the initial database setup was functional, we realised that it would pose difficulties in testing the application and introducing changes if the database was not easily integrated with the main project files. Thus, the team made the informed decision to transition to SQLAlchemy, which provided us with the necessary functionality and ease of integration.

A significant challenge we faced throughout the project was the limited coding background of many team members. Learning to build components from scratch and integrating them meant the learning curve for each project submission was significant. For example, integrating HTML pages with Python code proved to be challenging on multiple occasions, requiring team effort in troubleshooting.

During the final stages of project development, we realised certain limitations in the application's usability. This highlighted the importance of better accessing our target audience and their needs during the planning stage. By identifying these limitations, we aimed to expand the application's functionality and improve the overall user experience.

5. TESTING AND EVALUATION:

5.1 Testing strategy

The testing strategy for the app involved several components. The functionality testing focused on ensuring the proper functioning of buttons, redirects, and the behaviour of the application when logged in and logged out. Additionally, forced access to links through the search bar was tested. The testing also involved verifying the addition and removal of stocks against entries in the database using TablePlus.

Responsiveness testing was conducted, but not all issues, particularly those related to mobile design, were addressed. Print statements were used to check outputs and inputs, and the templates and terminals were utilised for verification purposes. The connection with the API was tested using the Thunder extension.

Furthermore, all the app routings were thoroughly examined to ensure that the correct status codes were returned, especially when distinguishing between logged-in and logged-out states. These tests were written as part of pytest coverage. The unit test coverage for the project is as follows:

- ``test_auth.py``: 4 out of 4 tests passed, covering 28% of the code.
- ``test_profile.py``: 2 out of 2 tests passed, covering 42% of the code.
- ``test_stocks_graphs.py``: 1 out of 1 test passed, covering 50% of the code.
- ``test_stocks_news.py``: 2 out of 2 tests passed, covering 64% of the code.
- ``test_unAuthenticated.py``: All 5 tests passed, covering 100% of the code.

Overall, there were 14 tests in total, with a total of 14 tests passed. The coverage percentages indicate the proportion of the code that was tested and evaluated by the respective test files.

```

> pytest
===== test session starts =====
platform darwin -- Python 3.11.3, pytest-7.3.1, pluggy-1.0.0
rootdir: /Users/mimithgeek/Learning/stocks-group-5
configfile: pytest.ini
collected 14 items

src/tests/test_auth.py .... [ 28%]
src/tests/test_profile.py .. [ 42%]
src/tests/test_stocks_graphs.py . [ 50%]
src/tests/test_stocks_news copy.py .. [ 64%]
src/tests/test_unAuthenticated.py ..... [100%]

===== 14 passed in 1.76s =====
~/Learning/stocks-group-5 on mimi !1 ?2  stocks-group-5-Bogg-L60 at 13:28:32

```

5.2. Functional and user testing

Functional tests are used to validate that individual units or components of the application are working correctly in isolation. These tests look at the behaviour of specific functions, methods or classes and check they produce the expected output.

User tests are used to test the application as a whole system and simulate the user interactions. User tests are written as functional tests, but simulate the users' interactions with the application. The tests focus on the applications behaviour from the user's perspective and ensure expected functionality.

To implement user tests, we defined a test user in the `_auth.py` file as the test case. Then, the tests will check that when the user makes actions, such as registering, logging in, logging out, updating their preferences on their profile, that the expected results are shown and no errors occur. The remaining tests were functional tests such as showing the stock graphs display correctly, news is retrieved from the api, and the homepage is rendered properly.

5.3. System limitations

- Scalability: The app was built using a Flask so if the app experiences a significant increase in users or traffic then the host server may struggle to handle the load resulting in slow response times or crashes. A solution to improve scalability is to use a cloud-based platform.
- Limited Stock options: The preselected list of stocks is limited and restricts the users choices. To improve this we would seek to integrate a financial database that offers a wide range of stocks and allows the users to search the database.
- Limited charting display: Only one price chart is displayed. To improve the user experience we would seek to implement chart modifications to alter timeframes, add functionality to overlay charts of different stocks, and even add charts of some industry indicators to help users in financial decision making.
- Lack of real time updates: Since we rely on external APIs for fetching stock prices, news and content, if the APIs delay in updating data, the results displayed are outdated. This

can be improved by scheduling tasks which periodically update the data, or, use WebSockets that establish a constant connection between server and client providing real time updates without need for manual refreshes.

6. CONCLUSION

The stock tracker application, developed as our group project for CFG Nanodegree Software specialisation, has proven to be a great learning tool for all members of the team. Each member of the team stepped out of their comfort zone while developing this app, helping us build upon the strong theoretical and practical foundation provided by the course. Employing Agile methodologies allowed us to make the app development process more organised and set up a system for group accountability, based on mutual respect and cooperation towards achieving a common goal. The app successfully provides users with the functionality we aimed to achieve in our objectives, for the exception of the notification system, the development of this system was hindered by unforeseen challenges in developing its functionality. Thorough testing was conducted to ensure functionality, performance, and security. We believe this app has a strong foundation for future expansion and demonstrates the team's ability to build a visually appealing and functioning tool.