Stock Prediction Project Proposal

Allen Chen(yilunc3), Xianyang Zhan(zhan39), Melissa Shi(mshi24), Hangao Zhang (hangaoz2)

Pitch

The unpredictability of the stock market often leads to uncertainty and suboptimal investment decisions. Our platform tackles this issue by leveraging advanced AI algorithms to analyze vast amounts of data, providing accurate predictions and personalized recommendations to empower investors and maximize their returns.

Functionality

1. Real-Time Market Analysis: Incorporate a live feed of market data to provide users with the most current information impacting stock prices.
2. Risk Assessment Tools: Develop algorithms that assess the risk levels of various investment portfolios based on historical data and current market conditions.
3. Financial Planning Tools: Provide users with interactive tools to create, test, and adjust their investment strategies based on various scenarios and predictions.
4. Community Features: Create a platform where users can share investment strategies, discuss market trends, and provide feedback on the platform.

Components

●Backend: We will use Python as our programming language and Flask as our web app framework. Two of our members already have experience with the Flask framework. For the algorithm we will use scikit-learn and pytorch to build and fitt model, using data from PyTrend and YFinance. For databases we will use SQL databases since all of our data are static in format. The back-end will receive requests that are timed for updating and predicting data, and requests from users to fetch data and create personalized plans.

Testing: We will test our model for a period of time, and record the error, and find-tuning the model by comparing regression, time-series machine learning such as neural networks, and a combination of both. For backend, we will test our program first using a terminal to make pseudo request, then we will test it with front-end.

Interactions With Other Components: For User Authentication System, component interacts with the User Database component using SQL queries and with the Frontend component via HTTP requests. The choice of HTTP is due to its universal support and ease of integration with the Flask framework. For Stock Information Fetcher, this component communicates with the Data Analysis component by providing it with the necessary data via method calls. The method call is chosen for its simplicity and directness.

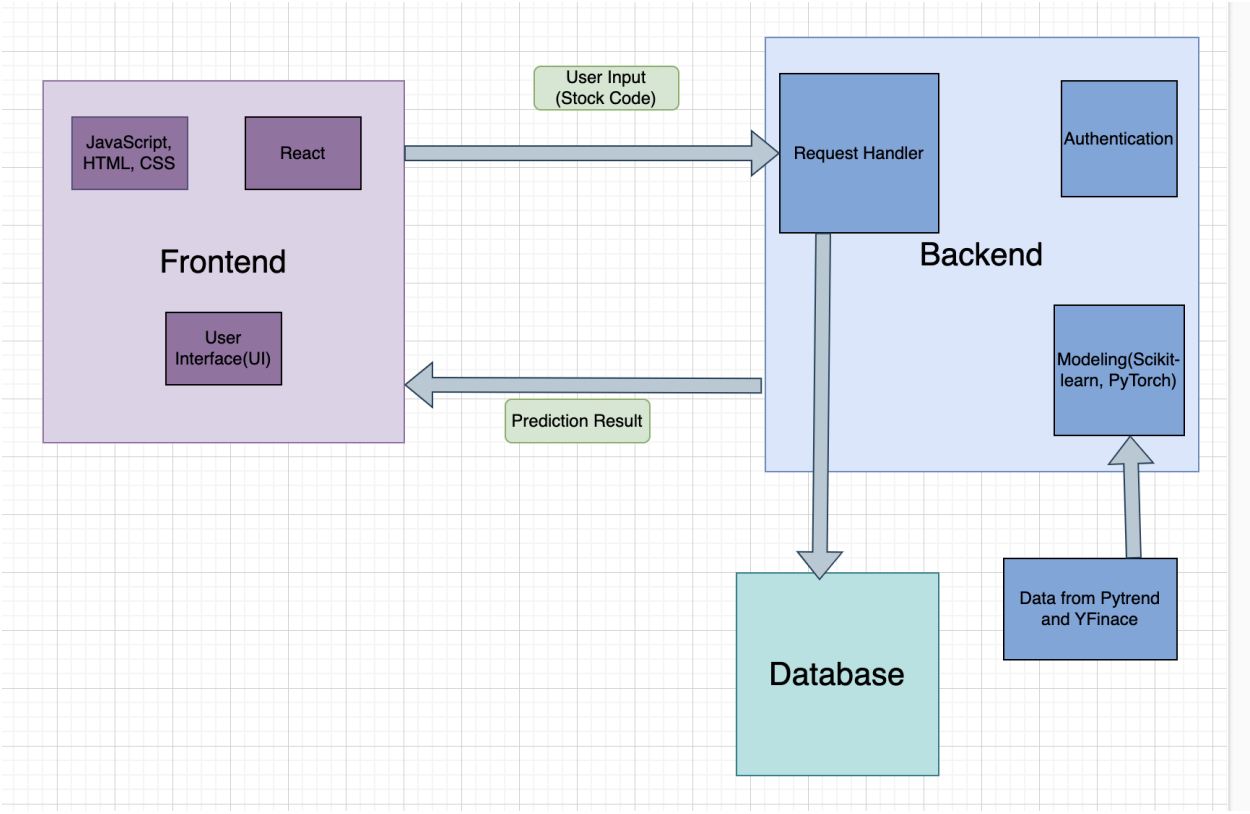
The backend has the following responsibilities:

- Store a list of user accounts
  - We will use a SQL database, maybe on Google Cloud
- Authenticate users using the account system provided by Flask and other existing libraries.
- Fetch stock and google trend information
  - We will use YFinance and Pytrend.
- Model
  - We will use Scikit-learn and Pytorch, maybe some other stats models to train our prediction model.

- Frontend: We will write the frontend using the TypeScript programming language HTML and the React framework. For the front end we will use React as our library as it has the largest developer community. The frontend is responsible for receiving requests from users and decodes to correct requests to the back-end, then update the desired results received from the output of the back-end. We will use libraries from React such as Material-UI, React Router, and Axios to enhance development and maintenance experience.

Testing: We will test the front end by exhausted choices of operations for main features, and ask multiple persons for beta testing in order for all different user habits to be considered.

- Main Page
  - The main page displays some updates on user's currently holding stocks and a list of stocks that are performing well.
- Login Page
  - Responsible for authenticate, registration, and login users, navigated from any request that requires user information.
- Account Page
  - Responsible for checking personal information, liked and current stock holding, building personal plans, and settings.
- Search page
  - Responsible for authenticate, registration, and login users, navigated from any request that requires user information.



Weekly Planning

Week # (Date)	**Tasks**
Week 1 (2/12-2/18)	1. (Model) Stock prediction model design: Research and select the prediction model and algorithms. 2. (Frontend) Design the user interface(UI) for the platform: Design the frontend for user registration, login and dashboard views (HTML and CSS). 3. Create a design documentation for the project
Week 2 (2/19-2/25)	1. (Model) Stock prediction model training: Collect datasets from the internet and use them to train the model. 2. (Backend) Implement the backend functionalities for user management, data processing and prediction requests.
Week 3 (2/26-3/3)	1. (Model) Testing and debugging the stock prediction model with more datasets. 2. (Frontend) Setup initial JavaScript interactions for user inputs.
Week 4 (3/4-3/10)	1. (Backend) Implement a complete backend system that can take user input (stock name or code) to query stock's past and current price data through API and feed the data to the stock prediction model to generate and return output prediction data. 2. (Model) Testing and debugging the stock prediction model.
Week 5 (3/11-3/17)	1. (Backend) Database design 2. (Backend) Login and register system design
Week 6 (3/18-3/24)	1. (Frontend) Integrate data visualization libraries for stock charts and predictions. 2. (Backend) Login and register system implementation

Week 7 (3/25-3/31)	1. Enhance Security Measures (Data and User Privacy) 2. Collect the User Experience data
Week 8 (4/1-4/7)	1. Connecting the frontend UI with backend services 2. User Experience Improvements
Week 9 (4/8-4/14)	1. Perform the comprehensive testing for improvement. 2. Collect some feedback from users' experiences.
Week 10 (4/15-4/21)	1. Make some adjustments according to the collected data. 2. Monitor the Stock Prediction Platform performance, user feedback, and model accuracy.

Potential Risks

1. Models not working well

   Schedule Adjustment: Additional resources could be allocated, or non-critical tasks may be delayed to accommodate this work.

   Impact on Schedule: Could delay project by 1-2 weeks depending on the complexity of the fix.

   Resolution: Review and refine the models based on testing feedback; this may involve additional research or consultation with experts.

2. API not working

   Schedule Adjustment: Reprioritize development tasks to focus on areas not dependent on the API until the issue is resolved.

   Impact on Schedule: Potential delay of a few days to a week, depending on the severity of the issue.

   Resolution: Troubleshoot the API issues with the development team; contact API support if it is a third-party product.

3. Potential law issue

   Schedule Adjustment: Hold on certain project deliverables until legal clearance is obtained or redirect efforts to components of the project not affected by legal concerns.

   Impact on Schedule: Varies greatly depending on the issue; could be minor or could require significant project restructuring.

   Resolution: Consult with legal advisors to understand the implications and adjust the project scope or processes as necessary.

Continuous Integration

We will not have a standardized editing environment, we will use github to manage our codes. We will follow the camelcase naming convention and have a docker environment to help us manage and align our development tools. Pycodestyle is the automated tool that we will use for our code style check. Our pull request workflow will involve opening PRs as soon as a feature or bug fix is completed in a feature branch. During our weekly meeting We'll assign a reviewer from the team, rotating responsibilities weekly to ensure each member participates in reviews regularly. In case a reviewer is not available, we will extend the last reviewer's shift until a new reviewer is available. To avoid merge conflicts, we'll regularly sync our feature branches with the main branch and communicate changes effectively within the team to minimize conflicts.

1) **Testing Library:** For unit testing, we will utilize pytest as our testing framework due to its powerful features and ease of use which complements our technology stack.
2) **Code Style Guide:** Our team will adhere to the PEP 8 style guide, utilizing camelCase naming conventions for variable and function identifiers. This will ensure code readability and maintainability across our project.
3) **Test Coverage:** We will employ coverage.py to measure test coverage, which is an essential metric for determining the extent to which our codebase is covered by automated tests. Our goal is to maintain a high level of test coverage, and this will be tracked through our CI process.
4) **Pull Request (PR) Workflow:**
    a) PR Opening: PRs will be opened as soon as a feature branch has a completed feature or bug fix ready for review.
    b) PR Review Process: During our weekly team meetings, we will assign a team member to review the PR, rotating this responsibility to ensure that all team members are engaged in the code review process.
    c) Alternate Reviewers: We will assign an alternate team member in case we fail to review someone's PR.
    d) Technique for avoiding "merge conflicts": PRs will only be merged into the main branch after a thorough review. Each PR must be reviewed by at least one other team member to maintain code quality and reduce the potential for bugs. We will establish clear

communication channels within the team to ensure everyone is aware of the changes being merged.

Teamwork

To minimize friction and streamline our collaborative efforts, we will establish a consistent development environment using Docker containers. This approach ensures that each team member works within an identical setup. Docker abstracts away environmental differences, allowing us to focus on development rather than troubleshooting configuration issues. This could offer us a unified platform for our Python-based machine learning model and JavaScript-driven frontend.

We are a team of four people, since we are building a machine learning model, we will likely not distribute specific tasks individually for the first few weeks. We will hold a weekly meeting to synchronize our work and distribute our next week's work. For the first few weeks, we will work collaboratively on the machine learning model. Post model training, we will shift towards developing a frontend prototype. Two team members with frontend experience will spearhead this effort, while the remaining two will begin integrating the model with the backend. In the final two weeks, we will focus on testing and refining our web application. Each member will be responsible for testing different components - one for the user authentication system, another for the data retrieval and processing, the third for the frontend interaction, and the last for overall integration. This comprehensive approach ensures that all aspects of the application are thoroughly vetted.