Post Earnings Announcement Drift (PEAD) Algorithm Development Plan

Group 10
Archit Sharma
Bhuvesh Chopra
Braedon Kwan

Capstone 4ZP6
Version 1
Friday April 4, 2025

Table of Contents

Team Meeting and Communication Plan

Communication Tools

Our primary communication tools will be WhatsApp and Microsoft Teams. WhatsApp will be used for quick, day-to-day messaging and updates, while Microsoft Teams will facilitate more structured communication, such as team meetings and work sessions. This dual approach ensures prompt and effective communication among team members, whether for informal discussions or formal meetings.

Document Sharing

All documents will be shared via Google Drive and GitHub. Google Drive will be used for managing documentation and non-code files, allowing for easy access and real-time collaboration. GitHub will handle version control and code sharing ensuring a centralised and organised workflow for our project files.

Meeting Frequency

The team will hold weekly meetings to discuss progress, address any issues, and plan upcoming tasks. These meetings will be scheduled to accommodate all members' availability, providing a consistent platform for updates, brainstorming, and decision-making. Majority of the meetings will be held during class time, but additional meetings may be held online.

Program Management Tool

We will use Trello for task management. It can also integrate well with Google Drive, GitHub, and Microsoft Teams, making it an efficient fit for our workflow.

Team Member Roles

Algorithm Developer: Archit Sharma

Archit was responsible for developing the backtest logic that tested the regression on the past 10 years of data. He handled the design, implementation, and optimization of the backtesting.

Data Analyst: Bhuvesh Chopra

Bhuvesh was responsible to connect to various data sources through their APIs to ensure fast and reliable access to filings, price, and volume data. His role involved creating the regression logic and creating the js file which populates the dashboard. He dalso handled the cleansing and preprocessing of the data, making it suitable for further analysis and algorithmic decision-making, ensuring the data is structured and ready for use in trading signals.

Full Stack Developer: Braedon Kwan

Braedon was responsible for integrating the algorithm's trading signals with the front end. He will ensure that trading signals generated by the algorithm are transmitted in real-time to the web interface. In addition, Braedon will design and develop a user-friendly front-end dashboard, providing users with a clear, interactive view of trading signals, stock info, and financial metrics.

Functional Requirements Mapping

Functional Requirement	People
P0	Bhuvesh, Archit
P1	Bhuvesh, Archit
P2	Bhuvesh, Archit
P3	Braedon

Workflow Plan

Version Control System

We will use GitHub for version control and collaboration. The repository will be organised with a branching strategy to manage feature development, bug fixes, and releases. Pull requests will be used for code review, ensuring code quality and team collaboration.

Agile Methodology

The team will adopt an Agile approach, focusing on Scrum practices with short sprints to regularly update, refine, and enhance the trading algorithm.

Data Storage Location

Data will be stored on a local server to ensure quick access, lower latency, and reduced reliance on cloud services. This setup will support fast data retrieval for earnings, price, and volume data during algorithm testing and development.

Tools for Requirements and Performance Metrics

We will use Trello for managing tasks and tracking progress. It will provide a clear view of project milestones, individual responsibilities, and sprint goals, ensuring efficient collaboration among team members.

Proof of Concept Demonstration Plan (Done in Nov)

Demonstration Description

The proof of concept will focus on showcasing the core functionality of the trading algorithm. Specifically, it will demonstrate the generation of a buy signal based on EPS surprises (differences between expected and actual EPS). The algorithm will

analyse a list of stocks, identify the discrepancies, and generate real-time buy signals. These signals will be displayed on a front-end dashboard, providing users with an intuitive view of buy signals across multiple stocks.

Demo Code Plan

Data Retrieval from Interactive Brokers

- The backend will connect to the Interactive Brokers API to retrieve real-time data from the broker, specifically focusing on EPS surprise.
- It will handle API authentication, data requests, and parsing of the response to extract the expected and reported EPS values for a list of stocks.
- The retrieved data will be prepared and stored temporarily for further processing.

Trading Signal Generation with Machine Learning

- The machine learning algorithm will use EPS surprise, Expected EPS surprise and its internal predictors as its key feature to predict trading signals.
- The backend will load a pre-trained model, which will be implemented in Python using libraries like scikit-learn and TensorFlow.
- The EPS surprise data will be fed into the model, generating a buy signal based on the prediction outcome.
- The output signal will be classified and stored, ready to be sent to the front end.

Backend Implementation in Python

- The backend will be built using Python, possibly leveraging frameworks like Flask to handle API endpoints and business logic.
- Key endpoints will include:
    - An endpoint for retrieving EPS surprise data from the Interactive Brokers API.
    - An endpoint for running the ML model and the live trading component generating trading signals.
    - An endpoint for sending the generated signals to the front end.

Front-End Development with React

- The front end will be developed using React, focusing on a dynamic and user-friendly interface.
- It will display the list of stocks, along with their trading signals (for now just buy), in a clean and responsive layout.
- The front end will make API calls to the backend to fetch real-time trading signals and update the user interface accordingly.

Integration Between Backend and Front End

- The front end will communicate with the backend using RESTful API calls, retrieving trading signals in JSON format.
- The front-end React app will render the trading signals in a tabular format, updating in real-time as new data is received from the backend.

Technology

Programming Languages

- The backend will be developed using Python for data retrieval, preprocessing, and machine learning model implementation using the outputs from the model(the signals) to be rendered on the frontend along with the previous backtest results or associated information.
- The front end will be built using HTML, CSS and JavaScript to create an interactive and responsive user interface.

Development Environment

- The team will use Visual Studio Code as the primary IDE, with necessary extensions for Python and Git integration.
- JupyterNotebooks will be used to develop and test the trading signal algorithm.
- GitHub will be used for version control, enabling collaborative coding, code reviews, and issue tracking.

Testing Framework

- For the backend, PyTest will be used to ensure the reliability of data processing functions, API endpoints, and machine learning outputs.

ML Libraries

- The project will leverage popular machine learning libraries like scikit-learn or TensorFlow for building and deploying the EPS surprise-based trading signal model.
- Pandas and NumPy will be used for data manipulation and preprocessing.

Project Scheduling