Imagine a future where your smartphone isn't just smart; it's your dedicated stock trading guru! We've harnessed the power of cutting-edge AI to design an app that predicts stock movements, empowering you to buy and sell with confidence. Dive into our portfolio and explore how we're revolutionizing the world of stock trading.

*Profit Predictors*

Project Portfolio

*11/13/23*

[Introduction <Milestone 1: Proposal> 2](#_Toc125988975)

[The [Team Name] Team < Milestone 1: Proposal > 3](#_Toc125988976)

[System Requirements < Milestone 1: Proposal > 4](#_Toc125988977)

[Requirements [optional] 4](#_Toc125988978)

[Epics [optional] 4](#_Toc125988979)

[Epic #1 4](#_Toc125988980)

[User Stories 4](#_Toc125988981)

[User Story #1 4](#_Toc125988982)

[Project Management 5](#_Toc125988983)

[Continuity of Operations Plan (COOP) < Milestone 1: Proposal> 5](#_Toc125988984)

[Project Plan 0](#_Toc125988985)

[System Architecture Design and Development < Milestone 1: Proposal & Milestone 2: Architecture> 0](#_Toc125988986)

[System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation> 0](#_Toc125988987)

[Project Postmortem <Postmortem> 0](#_Toc125988988)

[Project Wins 0](#_Toc125988989)

[Root Cause Analysis 0](#_Toc125988990)

[Lessons Learned 0](#_Toc125988991)

[System Design <Milestone 2: System Architecture> 1](#_Toc125988992)

[System Architecture <Milestone 2: System Architecture> 1](#_Toc125988993)

[Component Design 1](#_Toc125988994)

[Data Flow 1](#_Toc125988995)

[System Components <Milestone 3: System Implementation> 1](#_Toc125988996)

[Component [Component Name 1] 1](#_Toc125988997)

[Component [Component Name 2] 1](#_Toc125988998)

[Component [Component Name n] 1](#_Toc125988999)

[Design Pattern <Milestone 3: System Implementation> 1](#_Toc125989000)

[Design Pattern <Milestone 3: System Implementation> 2](#_Toc125989001)

[System Implementation <Milestone 3: System Implementation> 3](#_Toc125989002)

[Project portfolio template directives and placeholders (delineated by “[ ]” or “< >” and/or highlighted or optional sections not included) should be removed from the document prior to submission. Empty sections for inclusion in later submissions may remain in the document for early submissions.]

[IMPORTANT: All diagrams developed using Enterprise Architectures must include the following acknowledgement: “Thanks to SPARX Systems for LSU student and faculty use of Enterprise Architect for academic purposes”.]

# Introduction

In today's fast-paced financial world, stock market participants, from seasoned traders to novice investors, often struggle to make timely and informed decisions. The sheer volume of data, news, and market signals can be overwhelming, making it challenging to discern the best moments to buy or sell a stock. Furthermore, emotional biases often lead to suboptimal decisions, which can significantly impact investment outcomes.

Enter our AI-powered Stock Trading Assistant. Designed meticulously using C++, our system capitalizes on the efficiency and speed of this object-oriented language, ensuring rapid real-time analyses. The application amalgamates various cutting-edge technologies, such as machine learning for predictive analytics and cloud computing for scalable data storage. Users can benefit from intelligent stock predictions while automating their trading strategies, all on a user-friendly interface. By merging the prowess of C++ with advanced technologies, we provide traders a tool that cuts through the noise, reduces emotional biases, and ushers in a new era of informed trading.

A blue hexagon with white letter c and plus symbols

Description automatically generated

`Core Features:

* Intelligent Stock Prediction:
  + Description: Utilizing advanced machine learning algorithms, the app would analyze historical and real-time data to predict potential price movements of stocks.
* Automated Trading System:
  + Description: Users can set their trading parameters, risk tolerance, and investment goals. Once set, the system will autonomously execute buy and sell orders based on its predictions and the user's presets.

Viable Features:

* Personalized Investment Dashboard:
  + Description: A visually appealing dashboard where users can monitor their portfolio's performance, view the AI's recent trade decisions, and get personalized insights and recommendations based on their trading history and market trends.

Adaptive Risk Management:

* + The feature would allow users to set their risk tolerance level (e.g., low, medium, high) when trading. The AI would then adjust its trading recommendations and strategies based on this input. For instance, a user with a "low" risk tolerance would see recommendations for more stable stocks and ETFs, while those with a "high" risk tolerance might receive suggestions for more volatile stocks or sectors. The AI would continuously learn from the market's behavior and the user's feedback to fine-tune these risk parameters, ensuring the recommendations align with the user's comfort zone and financial goals.

# The Profit predictors Team

Team Structure:

1. William Graham

* Role: Project Manager
* Responsibilities:
  + Oversee the project's overall direction and progress.
  + Coordinate meetings and ensure effective communication among team members.
  + Assign tasks and ensure milestones are met on time.

1. Riley Oest

* Role: Lead Developer (Backend)
* Responsibilities:
  + Design and develop the app's backend architecture.
  + Integrate the AI algorithms with the app.
  + Coordinate with the frontend team to ensure seamless integration.

1. Don Juan

* Role: AI & Machine Learning Specialist
* Responsibilities:
* Design and train the AI models for stock prediction.
* Constantly refine and update the models based on new data and feedback.
* Work closely with the backend team for AI integration.

1. Landon Clarke

* Role: Frontend Developer
* Responsibilities:
* Design and implement the app's user interface.
* Ensure a responsive and user-friendly experience.
* Collaborate with the backend team for data display and integration.

1. Mark Echols

* Role: Quality Assurance & Tester
* Responsibilities:
* Conduct rigorous testing of the app, both frontend and backend.
* Identify bugs and coordinate with developers for solutions.
* Ensure the app maintains a high standard of quality and reliability.

1. Andrew Holden

* Role: Financial Analyst
* Responsibilities:
* Provide insights into stock market trends and data.
* Collaborate with the AI specialist to provide domain knowledge.
* Assist in validating the accuracy of AI predictions.

1. Giovanni

* Role: Integration Specialist & API Developer
* Responsibilities:
* Develop and manage APIs for interfacing with external data sources, ensuring real-time stock data feeds into the app.
* Coordinate with both frontend and backend teams to ensure seamless data flow and integration.
* Address integration-related challenges, ensuring the app's various components communicate effectively.

# System Requirements

## Requirements [optional]

[*A list of system requirements. This should include, at a minimum, the requirements imposed by the class project*.]

### User Story #1

[*User Story Statement, using the following format:*

As an investor, I want to get AI-driven risk assessments for specific stocks based on historical data and current market conditions, so I can manage my investments wisely.

# Project Management

## Continuity of Operations Plan (COOP)

Effective communication and coordination are the lifeblood of our project, and as such, we are committed to ensuring a seamless flow of operations even in the face of unforeseen challenges.

Our primary mode of communication will be through a dedicated channel, where all updates, queries, and discussions will take place. Weekly Zoom meetings will be scheduled for team check-ins, ensuring we stay on track and address any pressing issues. For version control and code collaboration, we will utilize GitHub, ensuring that every member is consistently updated with the latest developments.

In light of the potential challenges posed by situations like the COVID-19 pandemic, all team members are equipped to work remotely and have access to all necessary tools and platforms from their homes. If a team member cannot attend in-person meetings, they can join virtually or catch up with recorded sessions, ensuring no one misses out on critical updates.

In case a member becomes temporarily unavailable due to unforeseen circumstances, tasks will be reallocated among the team based on the skill set. Each member will document their roles and current tasks comprehensively so that in their absence, another team member can take over with minimal friction. If, however, a member becomes permanently unavailable, we will convene an emergency meeting to redistribute responsibilities, ensuring the project's continuity and timely delivery.

Our COOP is designed to maintain our project's momentum, regardless of the obstacles we might face, ensuring that we consistently deliver on our commitments.

Group Gethib link: [hkaiserteaching/csc3380-fall-2023-project-group-3: csc3380-fall-2023-project-group-3 created by GitHub Classroom](https://github.com/hkaiserteaching/csc3380-fall-2023-project-group-3)

## Project Plan

### System Architecture Design and Development < Milestone 1: Proposal & Milestone 2: Architecture>

[Milestone 1 (Proposal): The Project Plan WBS provides a list of activities/tasks to be undertaken to complete Milestone 2 (Architecture). The WBS activity chart should include task dependencies, estimated level of effort, and expected start and completion dates.

Milestone 2 (Architecture): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Investigate and implement twitter API |  | 80% |  | 9/13/23 | 10/5/23 |  |  |
| 2 | Implement accurate web scraping |  | 80% |  | 9/13/23 | 10/5/23 |  |  |
| 3 | Develop a basis and a front end |  | 70% |  | 9/13/23 | 10/5/23 |  |  |

### System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation>

[Milestone 2 (Architecture): The Project Plan WBS provides a list of activities/tasks to be undertaken to complete Milestone 3 (System Implementation). The WBS activity chart should include task dependencies, estimated level of effort, and expected start and completion dates.

Milestone 3 (System Implementation): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
|  |  |  |  |  |  |  |  |  |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design <Milestone 2: System Architecture>

[*Include a short (1-2 sentences) statement about the system design*.]

## System Architecture <Milestone 2: System Architecture>

[*A short description of the system architecture.*]

### Component Design

[*Insert image of system architecture component diagram. Include the name of the team member that created the diagram in EA.*]

[*Architecture overview, to include user I/O, external data sources, and major system components.* ]

### Data Flow

[*Insert image of system architecture data flow diagram. Include the name of the team member that created the diagram in EA.*]

[*Architecture data flow discussion: a high-level description of the data between both internal major components and external data sources.*]

## System Components <Milestone 3: System Implementation>

[*Include a component sub-section for each component in the architecture diagram. Each component subsection will include a class diagram*]

### Component [Component Name 1]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name 2]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name n]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. The project must include at least design patterns covered in class. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. Include the name of the team member that created the diagram in EA. A second design pattern may be included for bonus points.*]

# System Implementation <Milestone 3: System Implementation>

[*In the table below, include a row for each component in your System Architecture diagram. In the second column, list the programming language(s) used to implement the component and the what % of that programming language is used in the implementation. In the third column, list the team member(s) that implement the component and what % of that implementation was completed by that team member. IMPORTANT NOTE: All architectural components must be implemented by an object-oriented programming language: Java, C++, or C#.*]

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *[Data Manager]* | *[C++ (45%)*  *Java (55%)]* | *[Mickey Mouse (15%)*  *Donald Duck (20%)*  *Daisy Duck (40%*  *Pluto (25%)]* |