Stock Analysis Program

Group Members:

- **Ethan Blair**
- **Andrew Donahue**
- **Carlos Perches**
- **Ethan Gimmestad**

Date: November 8th

1. Overview

The **Stock Analysis Program** provides a robust tool for stock market analysis, enabling users to make informed investment decisions with minimal risk. In today's volatile financial market, individuals need tools to understand trends and assess risks effectively. This program aggregates real-time stock data and performs risk analysis from multiple sources, creating a centralized platform for well-rounded decision-making.

Designed to support low-risk trading, the system prioritizes stability and consistency, making it suitable for both new investors learning about the market and seasoned investors seeking a reliable tool for frequent use. With a data-driven approach focused on accessibility and reliability, the program empowers users to trade responsibly and build financial literacy.

Objective:
The primary goal of this project is to create a low-risk stock analysis tool that helps users analyze trends, track performance, and consolidate data for easier decision-making. Unlike high-risk platforms, this system promotes sustainable investing, allowing users to trade confidently with less volatility, thereby supporting safer, informed decisions in the stock market.

2. Team Roles and Responsibilities

Team Roles and Responsibilities

The **Stock Analysis Program** team consists of four members, each with specialized roles that contribute to the project's success.

- **Ethan B - Systems Analyst and UI/UX Designer**
  As the **Systems Analyst**, Ethan translates business needs into technical requirements, analyzing user needs and documenting system features. He ensures that the program aligns with user expectations for low-risk trading and conducts feasibility studies to confirm solutions are practical and within budget.

As the **UI/UX Designer**, Ethan creates a user-friendly interface with wireframes and prototypes. He collaborates with Ethan G to integrate the design with back-end functions, aiming to make the tool accessible for users of all technical levels.

- **Andrew - Project Manager and Technical Writer**
  As **Project Manager**, Andrew coordinates all aspects of planning and execution, assigning tasks, monitoring progress, and managing team communication. He provides regular updates to stakeholders, keeping everyone aligned on project goals.
  As the **Technical Writer**, Andrew is responsible for creating user manuals, FAQs, and troubleshooting guides, ensuring the documentation is clear and accessible for users and future developers.
- **Ethan G - Software Developer and DevOps Engineer**
  As **Software Developer**, Ethan G implements the program's core functions, including data retrieval and analysis using the Yahoo Finance API. He collaborates with Ethan B to ensure the front-end and back-end work together seamlessly, and he optimizes code for real-time performance.
  As the **DevOps Engineer**, Ethan G manages development environments and automates testing and deployment processes, minimizing production issues and monitoring system performance.
- **Carlos - Database Administrator and Security Analyst**
  As **Database Administrator**, Carlos designs and maintains the database using PostgreSQL, optimizing data storage and retrieval. He ensures data integrity, scalability, and manages backups for data availability.
  As **Security Analyst**, Carlos protects user data by implementing encryption and authentication protocols, conducting security audits, and monitoring for potential threats.(there is no security as of right now)

3. Use Cases and User Stories

Use Case Name: Sign-In

- **ID**: UC-01
- **Priority**: High
- **Actor**: Customer
- **Description**: The customer wants to sign in to the system to save their data and preferences.
- **Trigger**: Customer navigates to the sign-in page.
- **Type**: External

Preconditions:
The customer has an existing account or is registered on the platform.

Normal Course:

1. The customer clicks on the "Sign In" button.
2. The system prompts the customer to enter their username and password.

3. The customer enters their credentials and submits the information.
4. The system verifies the credentials.
5. If the credentials are correct, the customer is signed in and redirected to the dashboard.

Alternate Flows:

- If the credentials are incorrect, the system displays an error message and prompts the customer to retry.

Postconditions:
The customer is logged in, and their personalized data and settings are accessible.

Use Case Name: Stock Picker

- **ID**: UC-02
- **Priority**: High
- **Actor**: Customer
- **Description**: The customer wants to select a specific stock for analysis when evaluating their investment portfolio.
- **Trigger**: Customer initiates a search for a particular stock.
- **Type**: External

Preconditions:
The customer has an account and is signed in.
Stock data is available from integrated financial data sources.

Normal Course:

1. The customer opens the stock picker tool.
2. The customer enters the stock symbol or name in the search bar.
3. The system retrieves data for the requested stock from the database.
4. The system displays the stock's information, including recent price, performance trends, and analysis.
5. The customer reviews the stock data and adds it to their portfolio for monitoring.

Postconditions:
The selected stock is added to the customer's portfolio for further analysis.

Use Case Name: Stock Updater

- **ID**: UC-03
- **Priority**: High

- **Actor**: Customer
- **Description**: The customer wants the stock data to refresh in real-time or as close to real-time as possible to make timely decisions.
- **Trigger**: Customer views stock data or refreshes the page.
- **Type**: External

Preconditions:
The customer has an active internet connection.
The system is connected to a reliable data source for stock prices.

Normal Course:

1. The customer opens the stock updater feature.
2. The system automatically fetches the latest stock data from the data provider.
3. The system updates the displayed stock prices and other relevant metrics.
4. The customer can view the refreshed data in real time.

Alternate Flows:

- If the system encounters an error in retrieving data, it displays a message and prompts the customer to retry.

Postconditions:
The system displays the most recent stock data to the customer.

Use Case Name: Add to Portfolio

- **ID**: UC-04
- **Priority**: Medium
- **Actor**: Customer
- **Description**: The customer wants to add specific stocks to their portfolio for easy tracking.
- **Trigger**: Customer selects a stock to add to their portfolio.
- **Type**: External

Preconditions:
The customer is signed in and has an active portfolio.
Stock data is available in the system.

Normal Course:

1. The customer searches for the stock they want to add.
2. The system displays the stock information.
3. The customer selects the “Add to Portfolio” option.

4. The system confirms the addition of the stock to the portfolio.
5. The customer's portfolio is updated with the selected stock.

Postconditions:
The stock is successfully added to the customer's portfolio.

Use Case Name: Remove from Portfolio

- **ID**: UC-05
- **Priority**: Medium
- **Actor**: Customer
- **Description**: The customer wants to remove a stock from their portfolio to keep it organized.
- **Trigger**: Customer selects a stock to remove from their portfolio.
- **Type**: External

Preconditions:
The customer is signed in and has a portfolio with added stocks.

Normal Course:

1. The customer navigates to their portfolio.
2. The customer selects the stock they want to remove.
3. The system confirms the removal of the stock from the portfolio.
4. The stock is removed from the portfolio display.

Postconditions:
The stock is no longer present in the customer's portfolio.

Use Case Name: View Investment History

- **ID**: UC-06
- **Priority**: Medium
- **Actor**: Customer
- **Description**: The customer wants to view a history of their trading actions to review past decisions.
- **Trigger**: Customer navigates to the investment history page.
- **Type**: External

Preconditions:
The customer is signed in and has a history of trading actions.

Normal Course:

1. The customer accesses the investment history section.
2. The system retrieves and displays the customer's trading history.
3. The customer reviews past transactions and decisions.

Postconditions:
The customer is able to view their complete trading history.

Use Case Name: Receive Notifications

- **ID**: UC-07
- **Priority**: High
- **Actor**: Customer
- **Description**: The customer wants to receive notifications on stock trends to act quickly.
- **Trigger**: System detects a significant trend or event in a monitored stock.
- **Type**: External

Preconditions:
The customer is signed in and has notifications enabled for specific stocks.

Normal Course:

1. The system monitors the stock trends in the customer's portfolio.
2. When a significant trend is detected, the system sends a notification to the customer.
3. The customer receives the notification and decides on further action.

Postconditions:
The customer is notified of significant stock trends in real time.

Use Case Name: Update Profile Information

- **ID**: UC-08
- **Priority**: Low
- **Actor**: Customer
- **Description**: The customer wants to update their profile information, such as email or display name.
- **Trigger**: Customer navigates to the profile settings page.
- **Type**: External

Preconditions:
The customer is signed in.

Normal Course:

1. The customer accesses the profile settings.
2. The system displays the current profile information.
3. The customer updates their profile details.
4. The system saves the changes and confirms the update.

Postconditions:
The customer's profile information is updated successfully.

Use Case Name: Generate Stock Recommendations

- **ID**: UC-09
- **Priority**: High
- **Actor**: System
- **Description**: The system generates stock recommendations based on analysis and risk preferences.
- **Trigger**: Customer requests stock recommendations.
- **Type**: Internal

Preconditions:
The customer is signed in and has specified preferences for recommendations.

Normal Course:

1. The customer selects the "Get Recommendations" option.
2. The system analyzes recent market data and the customer's preferences.
3. The system generates a list of recommended stocks.
4. The recommendations are displayed to the customer, along with a confidence score.

Postconditions:
The customer receives a list of stock recommendations based on their preferences.

Use Case Name: Save Data to Database

- **ID**: UC-10
- **Priority**: Medium
- **Actor**: System
- **Description**: The system saves user data to the database for future analysis and retrieval.
- **Trigger**: User performs an action that requires data to be saved (e.g., adding to portfolio).

- **Type**: Internal

Preconditions:
The customer is signed in.
The database is connected and operational.

Normal Course:

1. The customer updates their portfolio or preferences.
2. The system saves the updated information to the database.
3. The database confirms the data has been saved.

Postconditions:
The customer's data is saved securely in the database for future reference.

4. Database Design and Data Requirements

The data model includes tables for stocks, stock prices, users, recommendations, and stock price and volatility history each serving a critical function in storing and processing information.

Table Schemas with SQL Code

Stock Table: Stores stock identifiers, ticker, company name, sector, and industry.
sql

```sql
CREATE TABLE stocks (
    stock_id SERIAL PRIMARY KEY,
    ticker VARCHAR(10),
    company_name VARCHAR(255),
    sector VARCHAR(100),
    industry VARCHAR(100)
);
```

Stock Prices Table: Logs price history associated with each stock.

```sql
CREATE TABLE stock_prices (
    stock_price_id SERIAL PRIMARY KEY,
    stock_id INT REFERENCES stocks(stock_id),
    stock_price DECIMAL(10, 2),
    stock_price_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

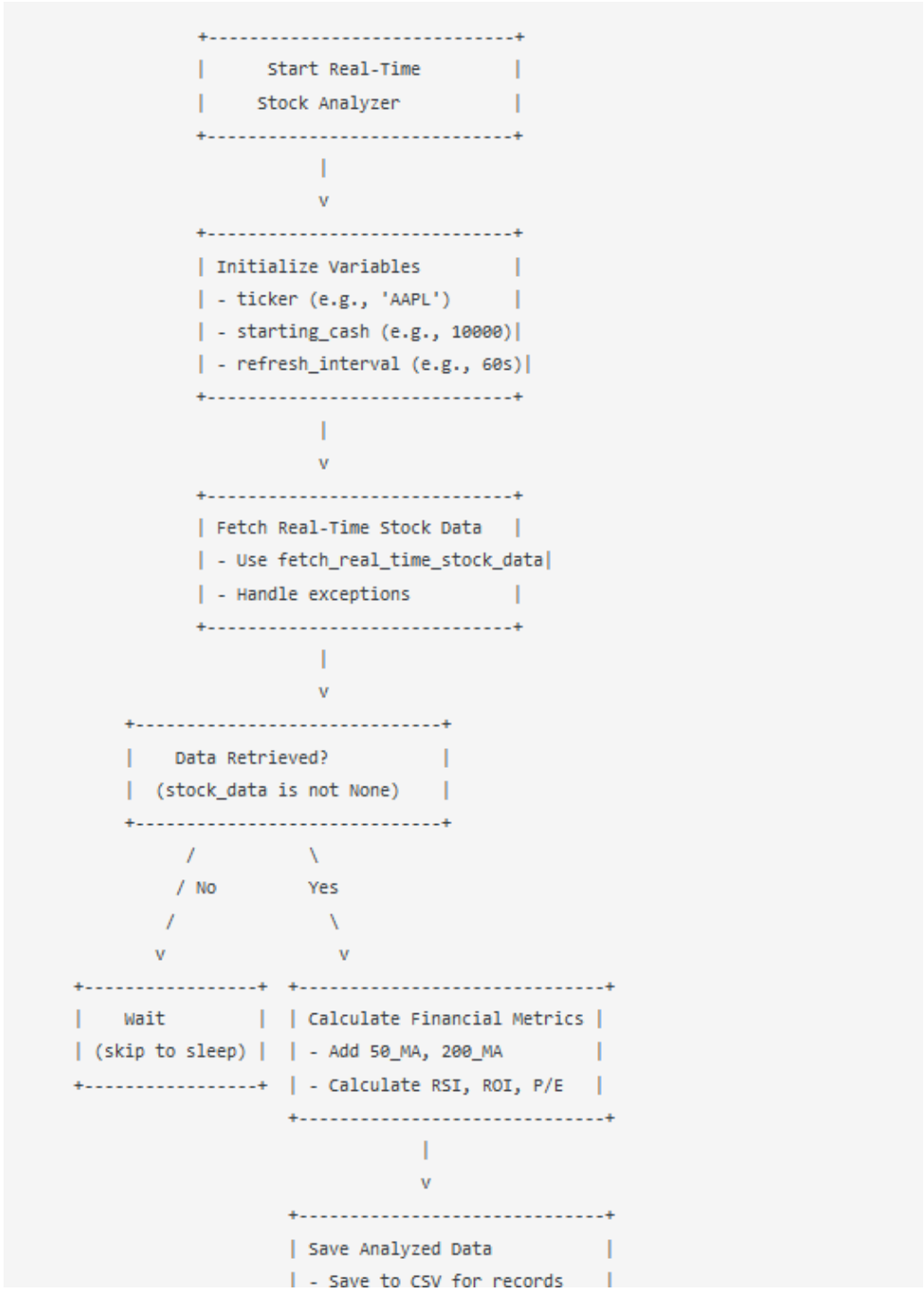
Users Table: Contains user information with Google OAuth integration.

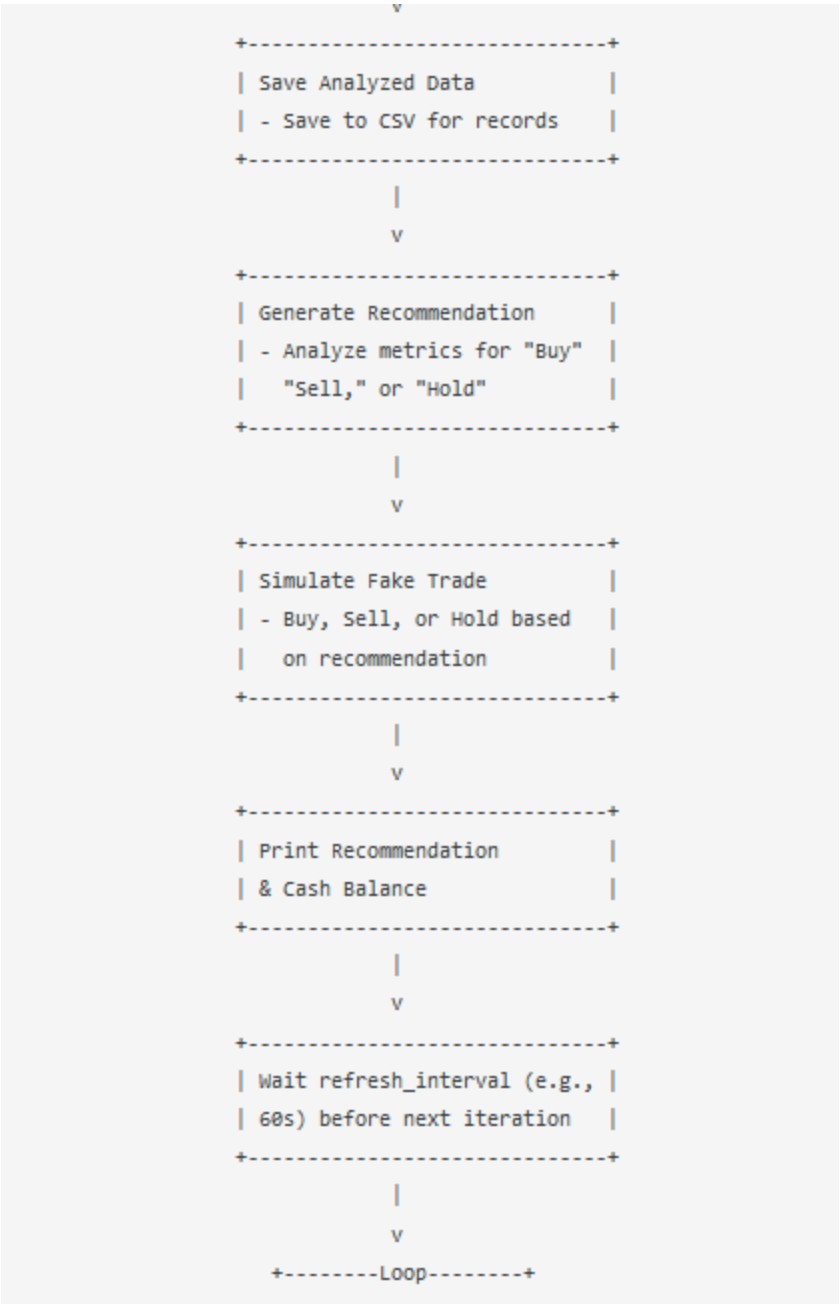
```
 CREATE TABLE users (
     id SERIAL PRIMARY KEY,
     username VARCHAR(50),
     firstname VARCHAR(50),
     lastname VARCHAR(50),
     email VARCHAR(100) UNIQUE NOT NULL,
     uuid UUID DEFAULT gen_random_uuid(),
     lastlogin TIMESTAMP,
     providername VARCHAR(50),
     providerid VARCHAR(50),
     accesstoken VARCHAR,
     refreshtoken VARCHAR
 );
```

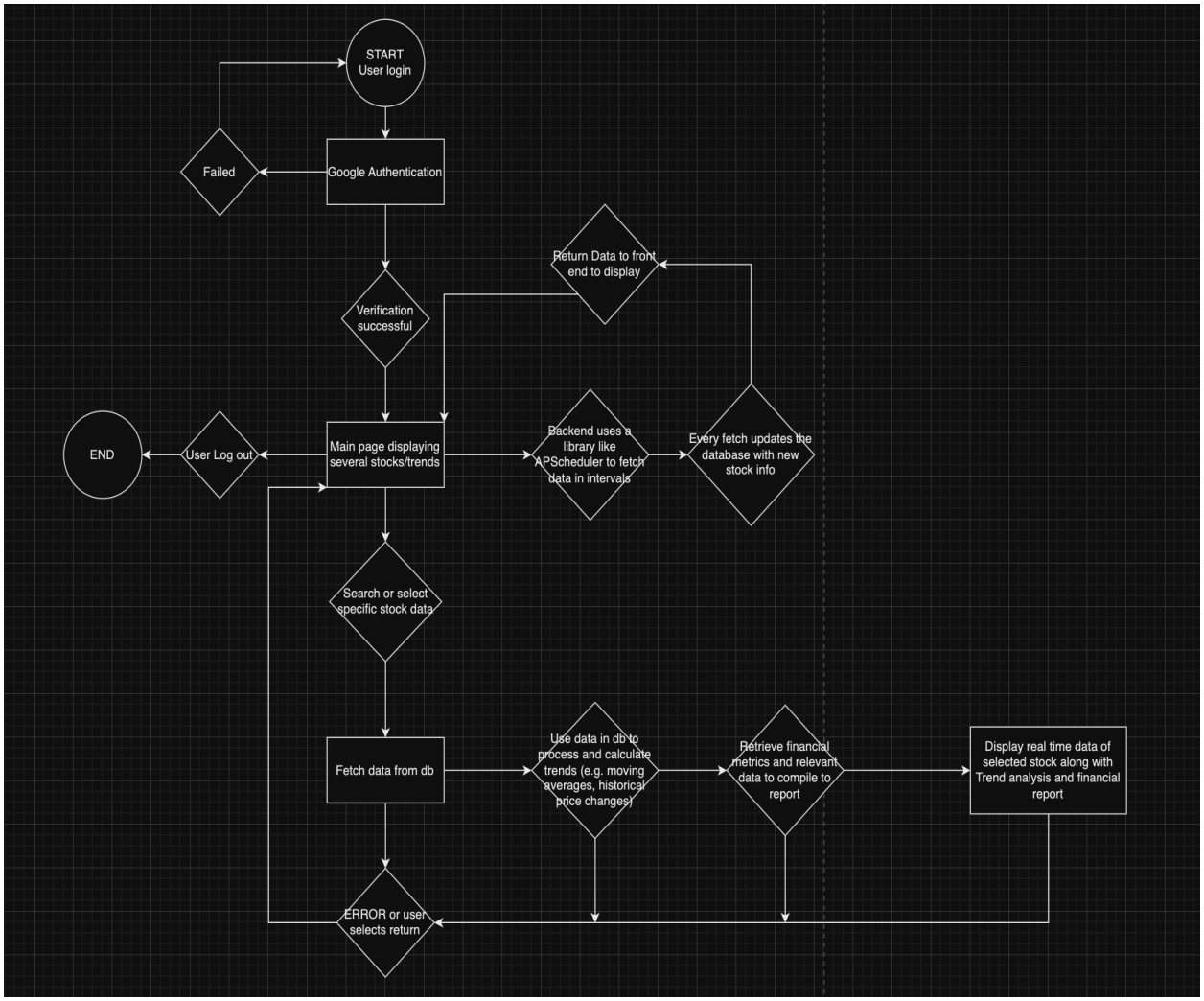
Recommendations Table: Tracks stock recommendations and analysis results.

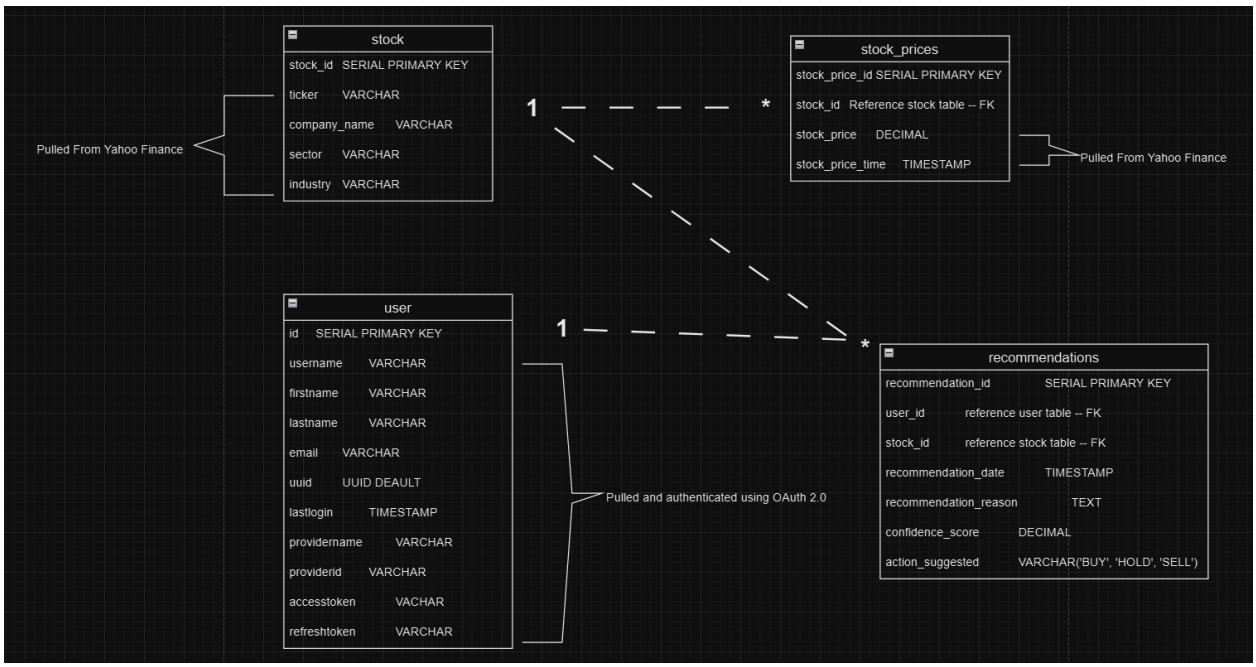
```
CREATE TABLE recommendations (
    recommendation_id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    stock_id INT REFERENCES stocks(stock_id) ON DELETE CASCADE,
    recommendation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    recommendation_reason TEXT,
    confidence_score DECIMAL(3, 2),
    action_suggested VARCHAR(10) CHECK (action_suggested IN ('BUY', 'HOLD', 'SELL
);
```

5. Process Models (code and database)


Start Real-Time
Stock Analyzer
Initialize Variables
- ticker (e.g., 'AAPL')
- starting_cash (e.g., 10000)
- refresh_interval (e.g., 60s)
Fetch Real-Time Stock Data
- Use fetch_real_time_stock_data
- Handle exceptions
Data Retrieved?
(stock_data is not None)
No
Yes
Wait
(skip to sleep)
Calculate Financial Metrics
- Add 50_MA, 200_MA
- Calculate RSI, ROI, P/E
Save Analyzed Data
- Save to CSV for records


v
Save Analyzed Data
- Save to CSV for records
Generate Recommendation
- Analyze metrics for "Buy"
"Sell," or "Hold"
Simulate Fake Trade
- Buy, Sell, or Hold based
on recommendation
Print Recommendation
& Cash Balance
Wait refresh_interval (e.g.,
60s) before next iteration
Loop


START
User login
Failed
Google Authentication
Verification successful
Return Data to front end to display
END
User Log out
Main page displaying several stocks/trends
Backend uses a library like APScheduler to fetch data in intervals
Every fetch updates the database with new stock info
Search or select specific stock data
Fetch data from db
Use data in db to process and calculate trends (e.g. moving averages, historical price changes)
Retrieve financial metrics and relevant data to compile to report
Display real time data of selected stock along with Trend analysis and financial report
ERROR or user selects return


stock
stock_id SERIAL PRIMARY KEY
ticker VARCHAR
company_name VARCHAR
sector VARCHAR
industry VARCHAR
Pulled From Yahoo Finance
1
*
stock_prices
stock_price_id SERIAL PRIMARY KEY
stock_id Reference stock table -- FK
stock_price DECIMAL
stock_price_time TIMESTAMP
Pulled From Yahoo Finance
user
id SERIAL PRIMARY KEY
username VARCHAR
firstname VARCHAR
lastname VARCHAR
email VARCHAR
uuid UUID DEAULT
lastlogin TIMESTAMP
providername VARCHAR
providerid VARCHAR
accesstoken VACHAR
refreshtoken VARCHAR
Pulled and authenticated using OAuth 2.0
1
*
recommendations
recommendation_id SERIAL PRIMARY KEY
user_id reference user table -- FK
stock_id reference stock table -- FK
recommendation_date TIMESTAMP
recommendation_reason TEXT
confidence_score DECIMAL
action_suggested VARCHAR('BUY', 'HOLD', 'SELL')

6. Project Management Details

Timeline and Phases
The project follows a structured timeline with checkpoints every two weeks:

- **Weeks 1-2**: Initial development, research, and prototype.
- **Weeks 3-4**: Code refinement, UI improvements, and final documentation.
- **Weeks 5-6**: Optimization and performance improvements.
- **Weeks 7-8**: Final adjustments and implementation.

SCRUM Approach
Using bi-weekly sprints, each sprint involves planning, execution, and review. Sprints allow for flexibility and adaptation to new insights or challenges.

Task Allocation and Tracking
Tasks are assigned based on skills and tracked using Trello or Google Sheets. Weekly meetings review progress and address any issues.

Communication
Weekly in-person meetings are supplemented by Snapchat for updates and Zoom for remote collaboration.

7. Meetings and Collaboration

7.1 Meeting Schedule
Meetings occur weekly on Tuesday or Thursday at 3:30 p.m., either in person or on Zoom.(preferably in person)

7.2 Meeting Notes Summary
Meetings cover project milestones, task updates, and brainstorming for new features, such as an email notification system for buy/sell prompts. Was discussed during our last meeting Nov 7.

7.3 Communication Tools
Snapchat for quick updates, in-person for detailed problem-solving, and Zoom for remote sessions.

8. Technical Feasibility and Constraints

Team Skills and Experience
The team has relevant skills in Python, PostgreSQL, and AWS, making this project feasible.

Technology and Tool Availability
All necessary tools are available, with supplementary resources used to bridge any knowledge gaps.

Development Process and Testing
SCRUM methodology allows for iterative progress, with testing phases focused on functionality and usability.

Conclusion

The **Stock Analysis Program** is a comprehensive, low-risk tool designed to support informed stock trading. Developed through careful planning and teamwork, the program combines real-time data analysis with a user-friendly interface, catering to both new and experienced investors. Each team member's expertise in design, development, database management, and security contributed to a stable, accessible, and secure system. With features such as stock updates, risk analysis, and data-saving capabilities, the program empowers users to make responsible trading decisions confidently. This project showcases the team's ability to translate requirements into a valuable financial tool that promotes sustainable investment practices.