INTERNSHIP REPORT

Submitted by

KRISHA R 312322104091

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

St. JOSEPH'S COLLEGE OF ENGINEERING

(An Autonomous Institution)

OMR, Chennai 600 119

ANNA UNIVERSITY :: CHENNAI 600 025

NOVEMBER 2025 – JANUARY 2026

Gmail

Krisha <krishar2004@gmail.com>

Infosys Springboard Virtual Internship 6.0 - Inauguration (Batch 8, 9 &10)

Infosys Springboard-Support <Springboard-support@infosys.com> Tue, Nov 18, 2025 at 12:37 PM

Dear Intern,

Congratulations! On successfully completing the document verification and securing your internship with **Infosys Springboard**. Consider this mail as a formal confirmation, you are part of **Infosys Springboard Internship 6.0.**

What next?

- **Internship completion certificate** will be provided to eligible interns after successful completion of internship.
- Use attached template to share your internship on social media. Do not forget to tag Infosys Springboard.
- You may request your mentor's approval for leave in order to attend your academic examination or placement drives.
- Attend virtual inauguration, details provided below:

Date	**November 24, 2025**
Time	**4:30 PM to 5:15 PM**
Webex Joining link	Click here to join the meeting
Meeting password	**WPqbd68YQM2**

Got a query? Write to **springboard-support@infosys.com**

Regards,
Team Infosys Springboard

ACKNOWLEDGEMENT

The contentment and elation that accompany the successful completion of any work would be incomplete without mentioning the people who made it possible.

Words are inadequate in offering our sincere thanks and gratitude to our respected Chairman **Dr. B. Babu Manoharan, M.A., M.B.A., Ph.D.** for his outstanding leadership, unwavering dedication, and invaluable contributions to our organization. His vision, guidance, and commitment have been instrumental in shaping our success and driving us towards excellence.

I express our sincere gratitude to our beloved Managing Director **Mr. B. Shashi Sekar, M.Sc. (Intl. Business)** and Executive Director **Mrs. B. Jessie Priya, M.Com. (Commerce)** for their continuous encouragement, administrative support, and motivation throughout the course of this project.

I also thank our beloved Principal **Dr. Vaddi Seshagiri Rao, M.E., MBA, Ph.D.**, for providing a stimulating academic environment and for encouraging us to pursue our undergraduate studies in Computer Science and Engineering in this esteemed institution.

I express our sincere thanks and heartfelt gratitude to our eminent Head of the Department, **Dr. V. Muthulakshmi, M.E., Ph.D.**, for her constant guidance, timely support, and valuable suggestions, which played a crucial role in the successful completion of this work.

I humbly express our profound gratitude for the invaluable guidance, expert technical support, and constructive feedback generously shared by our mentor, **Mr. G. Navinash**, whose continuous supervision and encouragement greatly enhanced the quality of this project.

I also extend our sincere thanks to all the faculty members of the Department of Computer Science and Engineering for their cooperation, technical insights, and academic support throughout the development of this project.

Finally, I express our heartfelt thanks to our family members and friends, who have been a constant source of motivation, moral support, and encouragement, without which this work would not have been possible.

ABSTRACT

The rapid growth of financial markets and the increasing availability of digital trading platforms have significantly raised the demand for intelligent decision support systems. Modern investors are required to analyze large volumes of real-time and historical financial data, making manual analysis inefficient, error-prone, and time-consuming. Traditional stock screening and portfolio monitoring tools are often limited to static filters and lack contextual reasoning capabilities, thereby restricting informed investment decisions.

This project presents **StockMind AI**, an AI-powered stock screening and advisory platform that integrates real-time market analytics, automated fundamental screening, interactive visualization, and natural language financial reasoning. The system combines structured database querying with Large Language Model based reasoning to convert user queries expressed in natural language into accurate financial insights. The platform supports dynamic watchlist analysis, portfolio visualization, real-time alerts, and AI-assisted market interpretation within a unified web environment.

The architecture employs a Flask-based backend, a PostgreSQL relational database, real-time data ingestion through financial APIs, and an LLM driven reasoning engine powered by Google Gemini. A Server-Sent Events based alert framework continuously monitors key financial metrics and notifies users of significant changes. The interactive frontend delivers advanced charting and dashboards for comparative analysis across valuation, growth, and risk indicators.

Experimental deployment demonstrates that the system significantly reduces analysis time, improves screening accuracy, and enhances investor awareness by providing timely, data grounded, and interpretable market intelligence. StockMind AI thus serves as a scalable and intelligent decision support solution for modern financial analytics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	DEFINITION
AI	Artificial Intelligence
API	Application Programming Interface
LLM	Large Language Model
SSE	Server Sent Events
SQL	Structured Query Language
NLP	Natural Language Processing
JWT	JSON Web Token
UI	User Interface
DBMS	Database Management System
RDBMS	Relational Database Management System
ETL	Extract Transform Load
CRUD	Create Read Update Delete
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation

CHAPTER 1

INTRODUCTION

1. 1. PROJECT OVERVIEW

StockMind AI is an AI powered stock screening and advisory web platform designed to support investors in analyzing equity fundamentals, monitoring portfolio performance, and identifying market opportunities through intelligent automation. The system integrates structured financial data, interactive visualization, and Large Language Model based reasoning within a unified analytical environment.

The platform enables users to explore stock fundamentals such as valuation ratios, growth indicators, and leverage metrics using dynamic dashboards and comparative charts. In addition, the system provides an AI assisted chat interface that allows users to express financial queries in natural language, which are automatically translated into secure analytical queries and meaningful financial explanations.

A real time alerting framework continuously monitors changes in key financial indicators and notifies users of significant updates through live notification streams. By combining screening, visualization, advisory intelligence, and alerting within a single platform, StockMind AI reduces manual effort, improves analytical consistency, and enhances decision making efficiency for active investors.

1.2 BUSINESS MOTIVATION

The modern financial ecosystem is characterized by high market volatility, rapid information flow, and increasing participation from retail as well as institutional investors. In such an environment, timely and accurate access to financial insights has become a critical competitive advantage. However, most investors rely on multiple disconnected tools for data collection, screening, visualization, and advisory analysis, leading to fragmented decision-making processes.

From a business perspective, inefficient financial analysis directly impacts investment performance, risk management, and strategic planning. Manual screening methods consume significant time and effort, while traditional platforms often fail to provide contextual reasoning or adaptive analysis capabilities. This creates a gap between available financial data and actionable investment intelligence.

The motivation behind StockMind AI is to bridge this gap by offering an intelligent financial analytics platform that consolidates market data, fundamental metrics, and AI-driven reasoning into a single system. By automating analytical workflows and enabling natural language interaction, the platform reduces operational overhead and enhances productivity for investors and analysts.

Furthermore, organizations in the financial services domain are increasingly adopting AI technologies to improve customer engagement, decision support, and operational efficiency. StockMind AI aligns with this industry trend by demonstrating how AI can be effectively integrated into financial analytics to deliver scalable, accurate, and interpretable insights.

Thus, the business motivation of this project lies in improving analytical efficiency, reducing decision latency, and enabling data-driven investment strategies through intelligent automation.

1.3 PROBLEM STATEMENT

In contemporary financial markets, investors are challenged by the increasing volume, velocity, and complexity of market and fundamental data.
While large datasets are readily available, extracting meaningful insights in a timely and accurate manner remains a significant problem.

Existing platforms primarily rely on rigid, rule based screening mechanisms that require manual configuration and lack contextual reasoning.
These systems do not support flexible exploration of financial information and are limited in their ability to adapt to evolving user requirements.
Furthermore, real time monitoring of portfolio and market changes is often inadequate, leading to delayed awareness of critical financial events.

Key issues identified include:

- Difficulty in performing multi metric financial analysis efficiently
- Lack of intuitive access to structured financial data
- Absence of intelligent reasoning over analytical results
- Limited real time alerting and monitoring capabilities
- Fragmented tools for screening, visualization, and advisory functions

These limitations increase the risk of missed opportunities, suboptimal investment decisions, and inefficient portfolio management.
There is a clear need for an integrated, intelligent, and scalable financial analytics platform that can transform raw financial data into actionable knowledge.

StockMind AI is designed to address these challenges by providing automated screening, AI assisted reasoning, interactive visualization, and continuous monitoring within a single coherent system.

1.4 OBJECTIVES

The objective of this project is to design and implement an intelligent, scalable, and user-centric financial analytics platform that supports structured investment decision making through automation and artificial intelligence. Modern investors are required to process large volumes of financial information and evaluate multiple indicators simultaneously. Manual analysis is time-consuming and often leads to inconsistent interpretations. This project aims to reduce this complexity by providing a system that transforms raw financial data into meaningful, actionable insights.

The major objectives of the system are:

• To develop a centralized backend architecture capable of securely handling authentication, business logic, and financial data management.

• To implement AI assisted natural language processing for interpreting complex financial queries and converting them into validated analytical logic.

• To enable flexible stock screening based on multiple fundamental and performance metrics without requiring manual query formulation.

• To design interactive dashboards that visually represent valuation, growth, and risk indicators across user selected stocks.

• To provide structured watchlist and portfolio analysis that supports comparative evaluation of investment options.

• To implement a real time alerting mechanism that continuously monitors financial indicators and notifies users of significant changes.

• To ensure high system reliability, data consistency, and controlled access to sensitive financial information.

• To optimize performance for handling concurrent user interactions and large datasets efficiently.

• To deliver a practical industry ready solution aligned with real world financial analysis workflows.

These objectives collectively guide the development of a robust decision support system that enhances analytical accuracy, reduces manual effort, and improves the overall efficiency of financial investment evaluation.

1.5 SCOPE OF THE PROJECT

The scope of this project is focused on the development of a full stack, AI enabled financial analytics platform that supports structured stock screening, portfolio analysis, and decision support for investors. The system is designed to operate as an end to end solution covering data ingestion, processing, storage, analysis, and visualization within a single integrated environment. It enables users to interact with financial data using both traditional dashboards and intelligent query mechanisms. The project scope includes:

- Implementation of a secure user management system supporting registration, authentication, and session control.
- Development of backend services for handling financial metrics, business logic, and analytical processing.
- Integration of external market data sources for retrieving stock fundamentals and performance indicators.
- Storage and management of historical and real time financial data using a relational database architecture.
- Construction of AI driven query processing components for interpreting user inputs and generating analytical results.
- Design of interactive web interfaces for watchlist, portfolio, market monitoring, and analytical chat.
- Implementation of alerting services that monitor metric changes and notify users in near real time.
- Support for multi metric comparisons across valuation, growth, and leverage dimensions.
- Handling of concurrent user requests with performance optimization and caching strategies.
- Basic compliance considerations to ensure that outputs remain informational and non advisory in nature.

The scope of this project does not include automated trading execution, brokerage integration, or personalized investment recommendations. The platform is intended strictly as a decision support and analytical tool.

CHAPTER 2

SYSTEM OVERVIEW

2.1 ARCHITECTURE OVERVIEW

The AI Powered Stock Screener and Advisory Platform is designed using a layered and service oriented architecture to ensure scalability, maintainability, and clear separation of responsibilities between system components.

The architecture follows a client server model where the presentation layer, application logic layer, intelligence layer, and data layer operate as logically independent modules while remaining tightly integrated through well defined interfaces.

At a high level, the system consists of the following major layers:

Presentation Layer

This layer represents the user facing interface of the application. It includes the web based dashboards for watchlist management, portfolio visualization, market monitoring, and AI assisted query interaction.

The presentation layer is responsible for:

• Collecting user inputs such as login credentials, stock selections, and analytical queries.

• Displaying structured financial information using charts, tables, and notifications.

• Rendering alert messages and system responses in an interactive and intuitive manner.

• Managing client side validations and UI level interactions.

This layer communicates exclusively with the backend through secure HTTP based APIs.

Application and Control Layer

This layer acts as the core coordination point of the system. It manages:

• Authentication and authorization of user requests.

• Validation of incoming data.

• Routing of business operations to the appropriate analytical or data services.

• Aggregation and formatting of responses before sending them back to the user interface.

It ensures that all business rules are consistently applied and that sensitive operations are protected.

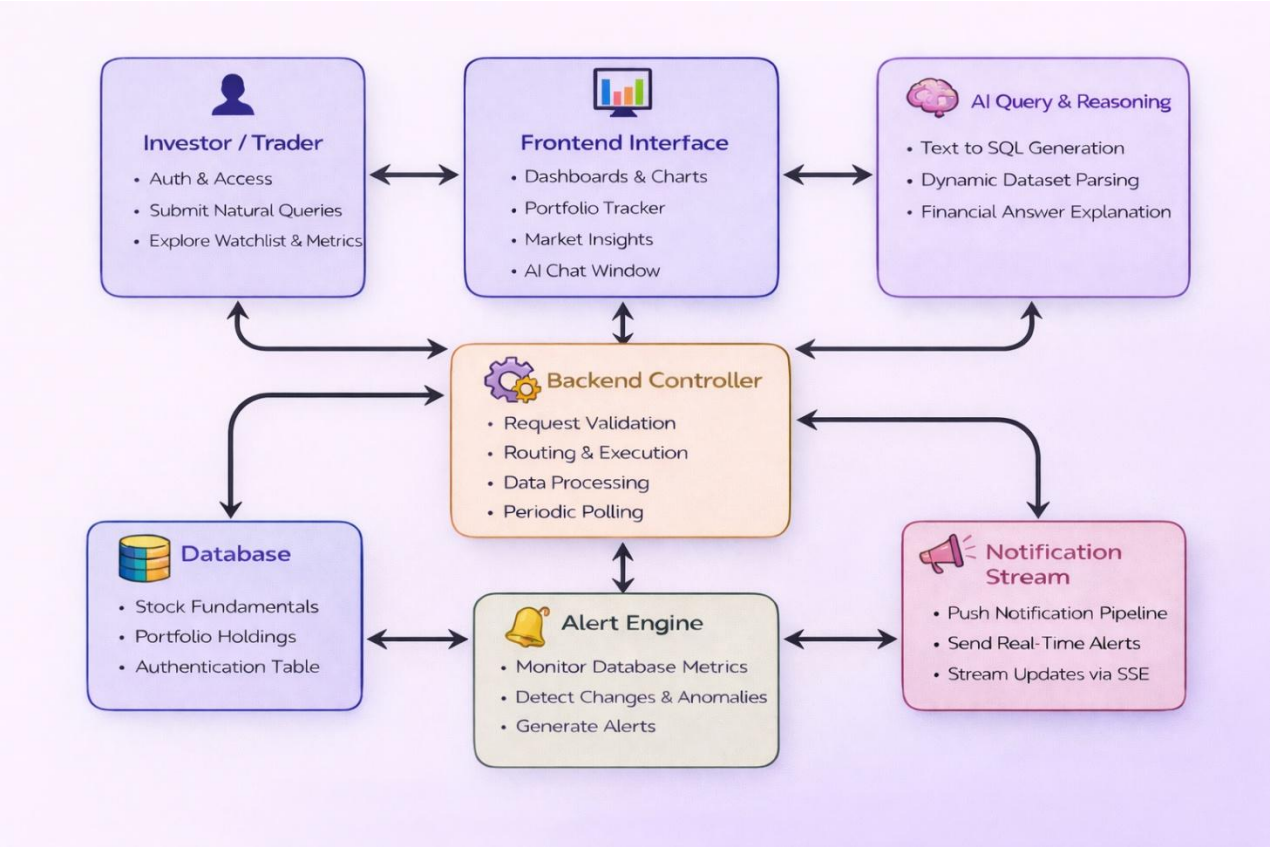

Figure 2.1: System Architecture

Intelligence Layer

The intelligence layer provides the AI driven capabilities of the platform. It is responsible for:

• Interpreting natural language financial queries.

• Converting analytical intent into structured logic.

• Supporting complex reasoning over financial metrics.

• Generating meaningful, data grounded explanations.

This layer enables flexible stock screening and advanced analytical interaction without requiring manual query construction by the user.

Data and Event Layer

This layer manages persistent storage and event driven processing. It includes:

• Financial and user data storage.
• Alert monitoring and event generation.
• Notification streaming to connected clients.
• Data synchronization from external market sources.

This layer ensures data consistency, historical tracking, and real time update propagation across the system.

Architectural Characteristics

The system architecture is designed to support:

• Modularity and ease of extension.
• Secure handling of sensitive information.
• High availability and responsiveness.
• Separation of concerns between UI, logic, intelligence, and data.
• Efficient handling of concurrent analytical workloads.

2.2 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system is expected to perform from a user and system interaction perspective. The AI Powered Stock Screener Platform is designed to support multiple analytical and operational functions required in a real-world financial decision support environment.

The major functional requirements of the system are as follows:

• The system shall allow users to register and authenticate securely before accessing any analytical features.
• The system shall provide a centralized dashboard for managing watchlists and portfolio data.
• The system shall retrieve and store stock fundamentals and performance metrics from external market data sources.
• The system shall support dynamic stock screening based on multiple financial criteria.
• The system shall allow users to submit natural language analytical queries and receive structured, data backed responses.
• The system shall visualize financial metrics using interactive charts and tables.

• The system shall monitor selected stocks and generate alerts when significant metric changes occur.
• The system shall push real time notifications to connected users.
• The system shall maintain historical records of financial data and user interactions.
• The system shall support CRUD operations for watchlists and portfolio management.
• The system shall enforce access control and session validation for all API operations.
• The system shall handle concurrent user requests efficiently.
These functional requirements define the core operational behavior of the platform and ensure that it fulfills its intended analytical purpose.

2.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes and constraints under which the system operates. These requirements are critical for ensuring reliability, performance, and usability of the platform.
The major non-functional requirements include:
• Performance:
The system shall respond to standard user requests within acceptable time limits and support real time alert propagation.
• Scalability:
The architecture shall allow the addition of new analytical modules and increased user load without major restructuring.
• Security:
The system shall protect user credentials and sensitive financial data using authentication, authorization, and controlled access mechanisms.
• Reliability:
The system shall maintain consistent operation and data integrity under normal and peak workloads.
• Maintainability:

The codebase shall be modular and well structured to support future enhancements and debugging.

• Usability:

The user interface shall be intuitive and require minimal learning effort.

• Availability:

The system shall ensure continuous service with minimal downtime.

• Data Consistency:

The system shall prevent unauthorized data modification and ensure accurate synchronization between components.

• Compliance:

The platform shall present analytical information only and avoid direct investment recommendations.

• Extensibility:

The system shall support future integration of additional financial indicators and analytical capabilities.

CHAPTER 3

SYSTEM DESIGN AND MODULES

3.1 ARCHITECTURE DESIGN

The architecture follows a layered design pattern consisting of:

- Presentation Layer
- Application Control Layer
- Intelligence Layer
- Data and Event Layer

Figure 3.1: Architectural Design

The Presentation Layer handles all user interactions through web based interfaces.

The Application Control Layer manages request validation, routing, and business logic execution.

The Intelligence Layer supports AI driven query interpretation and analytical reasoning.

The Data and Event Layer handles persistent storage, alert monitoring, and event

streaming.

This structure enables flexibility in extending features and simplifies maintenance and performance optimization.

The architecture is designed to support real time operations, concurrent users, and continuous analytical workloads.

3.2 MODULE DESCRIPTION

The system is implemented as a collection of logically independent modules, each responsible for a specific aspect of the platform's functionality.

3.2.1 AUTHENTICATION MODULE

The Authentication Module is responsible for managing user identity, access control, and session security across the entire platform.

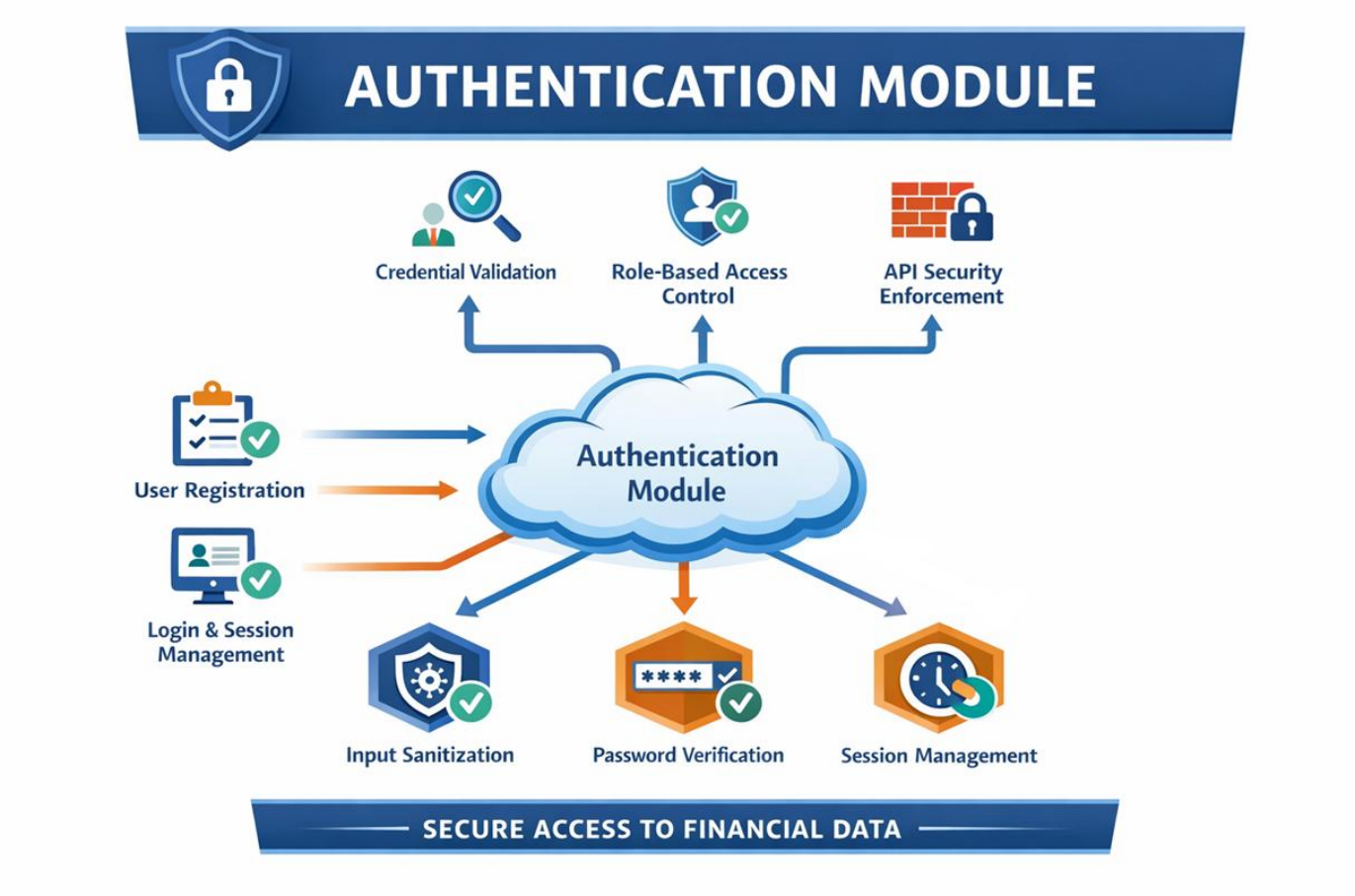

Figure 3.2.1: Authentication Module

This module provides:

• User registration with credential validation.

• Secure login processing and session management.

• Role based access enforcement to protect sensitive operations.

• Prevention of unauthorized API access.

All incoming requests are validated through this module before being forwarded to other system components. It ensures that only authenticated users can interact with financial data and analytical services.

The module also handles:

• Input sanitization to prevent invalid or malicious data.

• Password handling and credential verification.

• Session lifecycle management, including login and logout control.

This module forms the security foundation of the platform and is critical for maintaining system integrity and user trust.

3.2.2 WATCHLIST AND PORFOLIO MODULE

The Watchlist and Portfolio Module manages the organization, storage, and analytical visualization of user selected stocks.

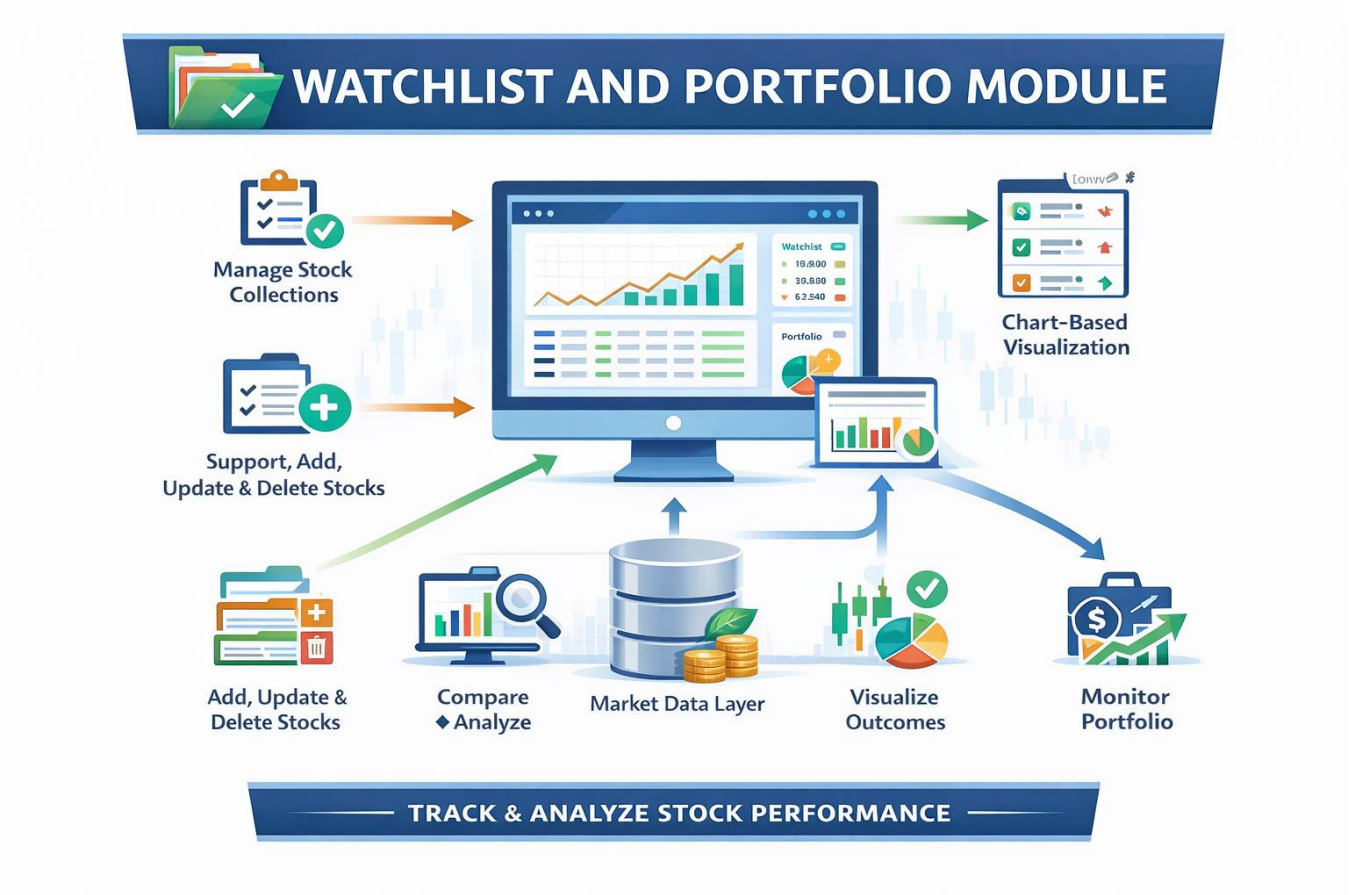

Figure 3.2.2: Watchlist and Portfolio Module

Its responsibilities include:

• Maintaining user specific stock collections.

• Supporting add, update, and delete operations on tracked symbols.

• Aggregating fundamental metrics for comparative analysis.

• Structuring financial data for chart-based visualization.

This module enables users to observe valuation, growth, and leverage indicators across multiple companies in a unified interface. It supports multi metric comparison and historical trend observation.

It also coordinates:

• Data retrieval from the market data layer.

• Formatting results for dashboard rendering.

• Handling portfolio level summaries.

The module plays a central role in transforming raw market information into meaningful analytical views for investment decision support.

3.2.3 AI QUERY ENGINE

The AI Query Engine is the intelligence core of the system. It enables users to interact with the platform using natural language instead of rigid query formats.

This module performs:

• Interpretation of user questions.

• Extraction of analytical intent.

• Generation of structured financial logic.

• Validation of analytical constraints.

The engine ensures that generated analytical operations remain safe, consistent, and aligned with available financial data.

It supports:

• Multi condition screening.

• Metric based reasoning.

• Context aware explanation generation.

By abstracting query complexity, this module significantly enhances usability and analytical flexibility of the platform.

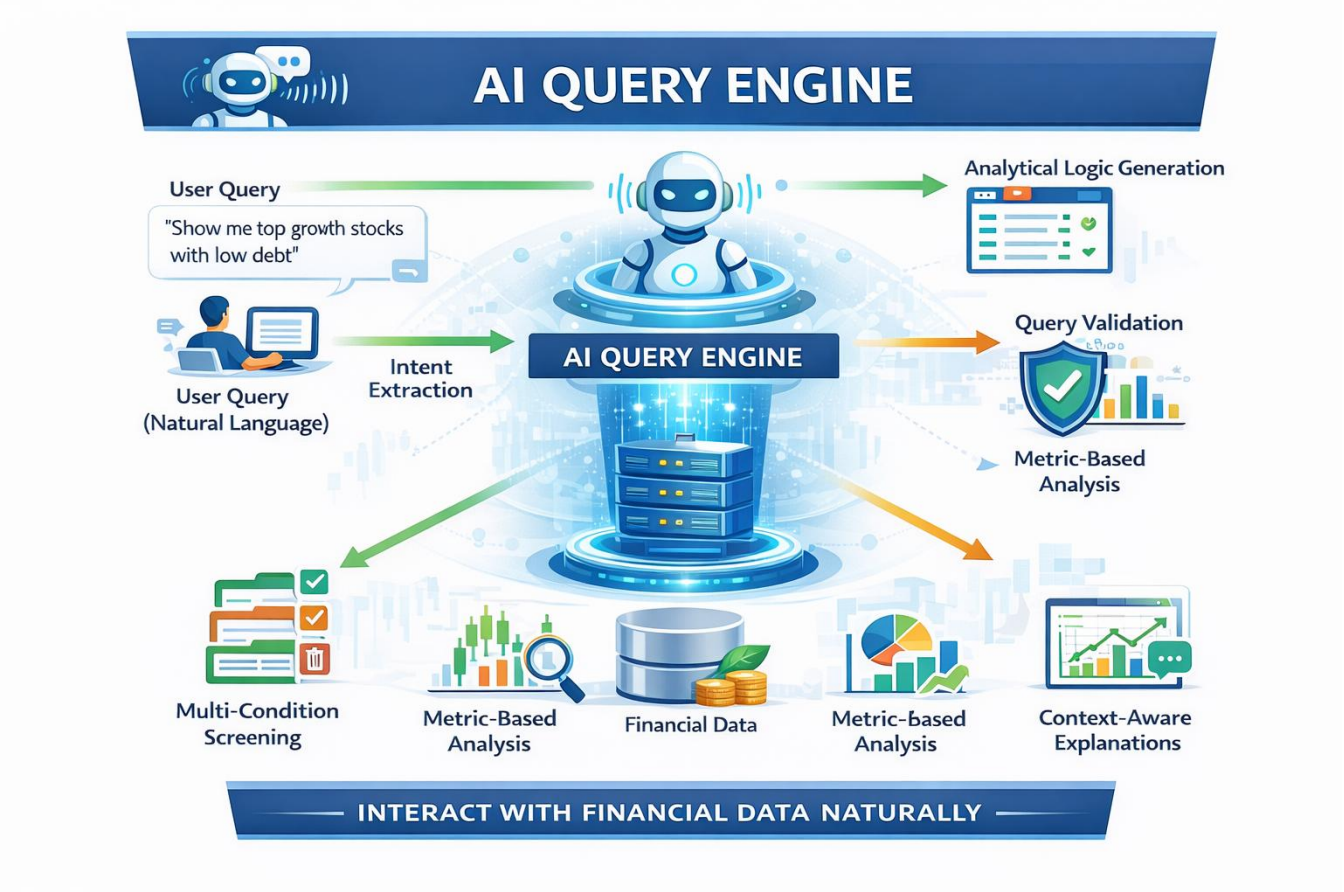

Figure 3.2.3: AI Query Engine

3.2.4 MARKET DATA MODULE

The Market Data Module is responsible for acquiring, managing, and updating financial information required by the system.

Its key functions include:

• Integration with external financial data providers.

• Periodic retrieval of stock fundamentals and performance metrics.

• Data normalization and validation.

• Storage of historical and current values.

This module ensures that all analytical operations are based on consistent and up to date market information.

It also supports:

• Data caching to improve performance.

• Synchronization of new data with dependent modules.

• Error handling for incomplete or delayed data sources.

The reliability of this module directly affects the accuracy and relevance of the platform's

outputs.

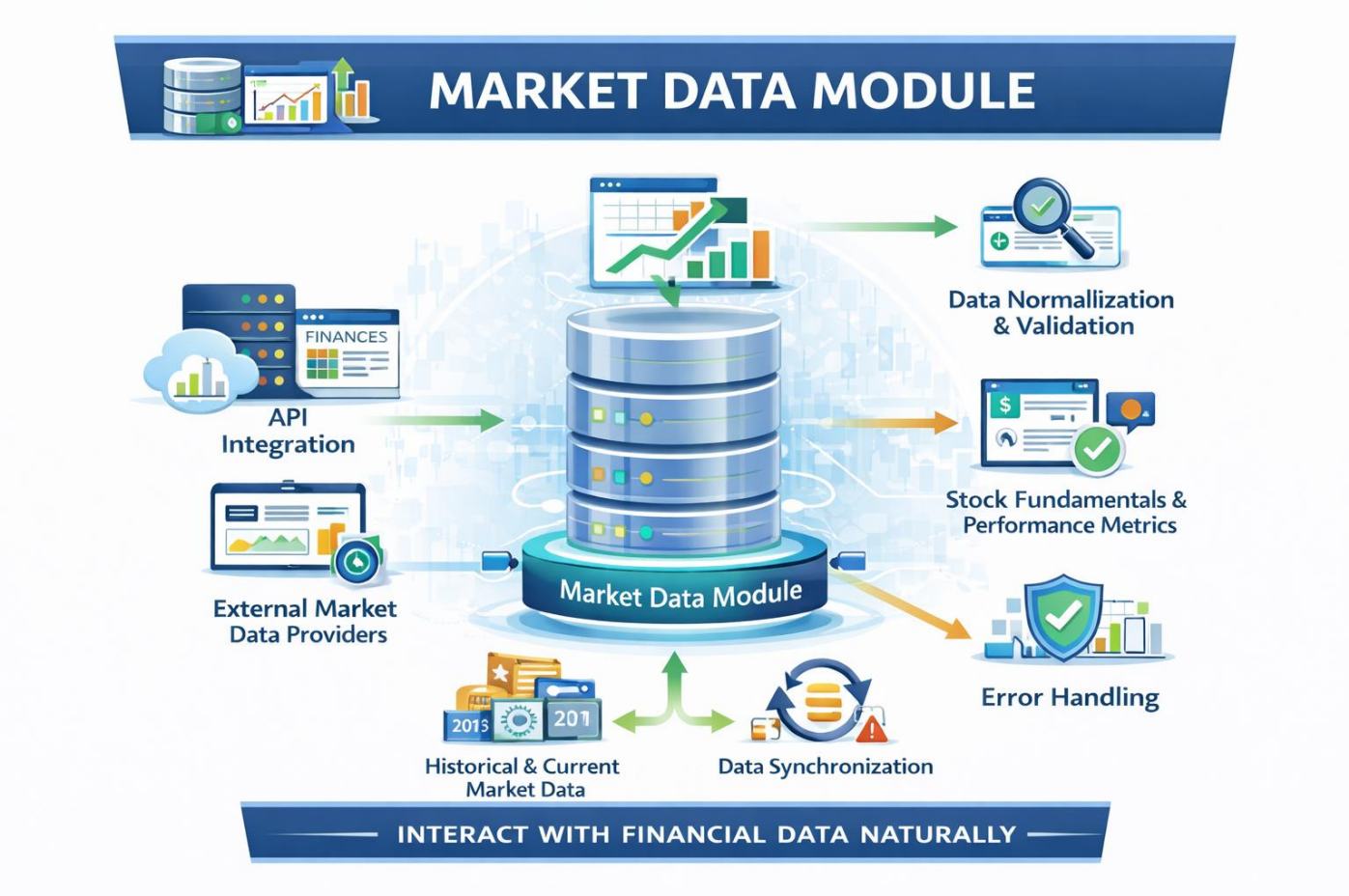

Figure 3.2.4: Market Data Module

3.2.5 ALERT AND NOTIFICATION MODULE

The Alert and Notification Module implements the real time monitoring and event communication capabilities of the system.

This module continuously:

• Observes selected financial metrics.

• Detects meaningful changes and threshold violations.

• Generates alert events.

• Streams notifications to connected clients.

It supports asynchronous communication to ensure that users receive timely updates without manual refreshing.

The module coordinates with:

• The data layer for metric evaluation.

• The backend controller for alert orchestration.

• The user interface for real time visualization of alerts.

This module enhances the platform's responsiveness and supports proactive financial monitoring.

Figure 3.2.5: Alert and Notification Module

CHAPTER 4

IMPLEMENTATION

4.1 BACKEND WORKFLOW

The Backend Workflow defines the operational backbone of the system. It coordinates all user requests, analytical processing, data access, and notification delivery in a controlled and secure manner.

The backend is implemented as a service-oriented architecture, where each request passes through a well-defined processing pipeline:

1. **Request Intake and Validation**
   All requests originating from the Frontend Interface are received by the Backend Controller through REST APIs. The system performs:
   - Token validation and session verification.
   - Input structure validation and parameter sanitization.
   - Access control checks based on user authentication status.
2. **Routing and Execution Logic**
   After validation, the backend routes the request to the appropriate internal service, such as:
   - Watchlist and Portfolio operations.
   - AI query processing.
   - Market data retrieval.
   - Alert configuration and management.
3. **Data Processing and Aggregation**
   The backend consolidates responses from the database, AI engine, and market data services, applies transformation logic, and formats the output into a consistent response structure suitable for frontend visualization.
4. **Response Delivery and Logging**

The processed results are returned to the frontend, while system logs capture request metadata, execution status, and performance metrics for monitoring and auditing.

This workflow ensures that all business logic remains centralized, secure, and maintainable while supporting high concurrency and real time responsiveness.

4.2 AI PROCESSING LOGIC

The AI Processing Logic enables natural language driven financial analysis by converting user intent into structured, executable reasoning steps.

The AI pipeline operates as follows:

1. **Natural Language Interpretation**
   User queries expressed in plain English are analyzed to identify:
   - Financial intent.
   - Target entities such as stocks, metrics, and conditions.
   - Logical constraints and comparison operators.
2. **Structured Query Generation**
   The system generates safe, validated analytical logic, primarily in the form of structured SQL compatible expressions, ensuring:
   - Protection against invalid or unsafe query patterns.
   - Alignment with the available financial schema.
3. **Financial Reasoning and Validation**
   The AI engine verifies that the constructed logic is:
   - Semantically consistent.
   - Financially meaningful.
   - Executable within the data model.
4. **Result Interpretation and Explanation**
   Raw analytical results are converted into human readable financial

explanations, enabling transparency and better decision support for the user.

This layered reasoning approach allows the platform to support complex, multi condition financial analysis without exposing query complexity to the user.

4.3 DATA MANAGEMENT

Data Management ensures that all financial information used by the platform remains consistent, accurate, and efficiently accessible.

The data layer is designed around the following principles:

1. **Data Acquisition and Synchronization**

   The system periodically retrieves market fundamentals and financial metrics from external data providers through secure APIs. The data is:

   - Normalized into a unified schema.
   - Validated for completeness and format consistency.
   - Synchronized with existing records.

2. **Persistent Storage and Indexing**

   Financial data, portfolio holdings, authentication information, and historical metrics are stored in a relational database with:

   - Structured indexing for fast query execution.
   - Referential integrity enforcement.
   - Versioned historical tracking.

3. **Efficient Data Access**

   Backend services access data through controlled query interfaces, supporting:

   - Aggregated analytical queries.
   - Portfolio level summaries.
   - Watchlist based comparisons.

4. **Data Reliability and Integrity Control**

   The system applies:

   - Transaction management.
   - Error handling for incomplete updates.
   - Consistency checks during periodic refresh cycles.

Effective data management directly impacts the accuracy of AI reasoning, alert evaluation, and user analytics, making it a critical foundation of the platform.

CHAPTER 5

RESULTS AND DISCUSSION

5.1. USER AUTHENTICATION INTERFACE

The authentication module provides a secure and user-friendly entry point to the system.

The registration screen allows new users to create an account by providing:

- Username
- Password
- Email address
- Phone number

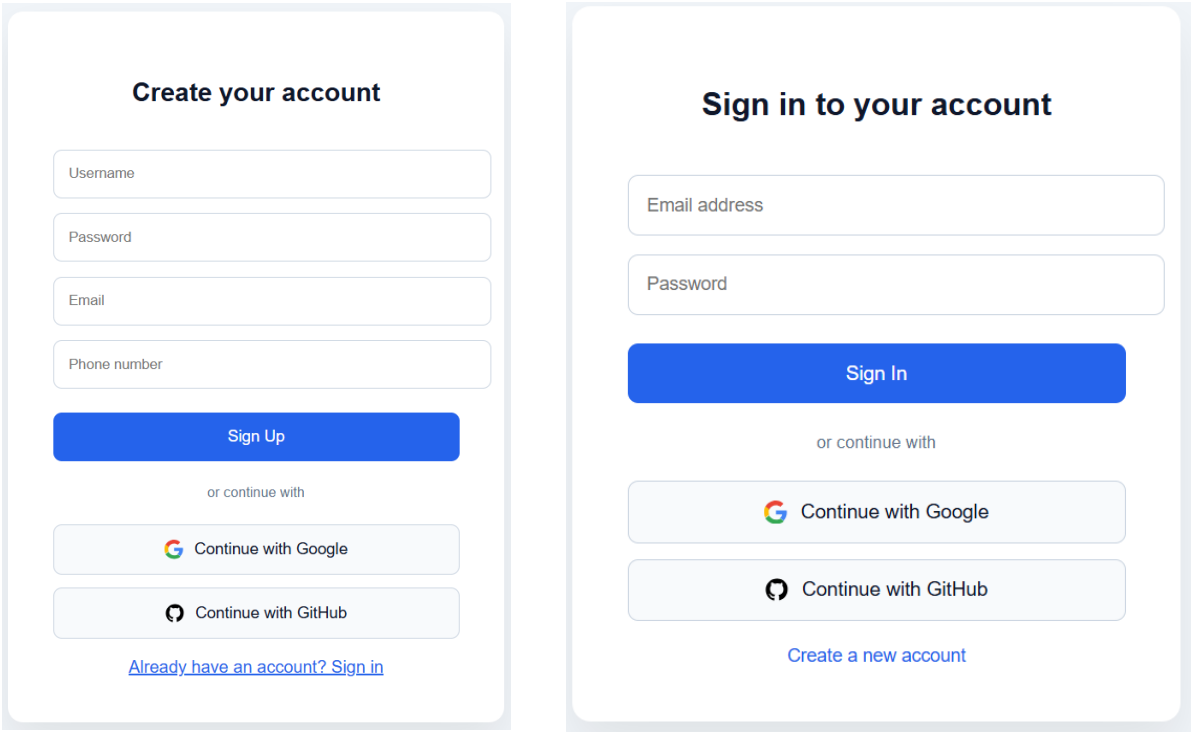

Figure 5.1: Signup and Sign-in page

Additionally, third party authentication is supported through Google and GitHub login options, enabling faster access and reduced manual data entry.

The login interface verifies user credentials and establishes a secure session before granting access to the application. This ensures that all subsequent operations such

as market analysis, watchlist access, and portfolio management are performed only by authorized users.

This output confirms the successful implementation of authentication, session handling, and access control mechanisms.

5.2 WATCHLIST ANALYTICAL DASHBOARD

The Watchlist page presents a consolidated analytical view of user selected stocks.

Multiple financial indicators are visualized using interactive charts, including:

- Open Price comparison
- Previous Close values
- Price Gap analysis
- Trailing Price to Earnings ratio

Figure 5.2 : Watchlist Analytical Dashboard

The dashboard allows the user to observe relative performance trends across multiple companies in a single interface. The dynamic chart rendering enables comparative evaluation of valuation and price movement behavior.

This result demonstrates the effective transformation of raw financial data into structured visual analytics that support informed stock screening decisions.

5.3 PORTFOLIO MANAGEMENT

The Portfolio module displays detailed investment information for the user.

Figure 5.3 : Portfolio Management

The output includes:

- Stock symbols
- Current market price
- User specified quantity
- Computed investment value
- Automated trading signal (BUY, HOLD, SELL)
- Total portfolio investment summary

This screen validates that the system correctly aggregates portfolio data, performs value calculations, and presents actionable signals based on the underlying financial metrics.

The result confirms accurate portfolio level analysis and structured financial reporting.

5.4 LIVE MARKET DASHBOARD

The Live Market Dashboard provides continuously updated stock information.

Each stock card displays:

- Current market price
- Percentage change
- Time based performance graph (1W, 1M, 3M, 1Y)
- Live update indicator

The system refreshes market values at fixed intervals, allowing users to monitor short term and medium term trends visually.

This output highlights the platform's ability to integrate external market data sources and present near real time analytical views for multiple companies simultaneously.

Figure 5.4 : Live Market Dashboard

5.5 AI ASSISTED FINANCIAL QUERY OUTPUT

The AI Assistant enables users to query the financial database using natural language.

Example outputs show:

- Ranking of companies based on total debt
- Identification of the lowest open price among listed stocks
- Numeric justify cation for each analytical result

The responses are structured, data driven, and directly derived from the underlying financial dataset. This confirms that the AI Query Engine successfully converts natural language questions into validated analytical operations and generates interpretable financial explanations.

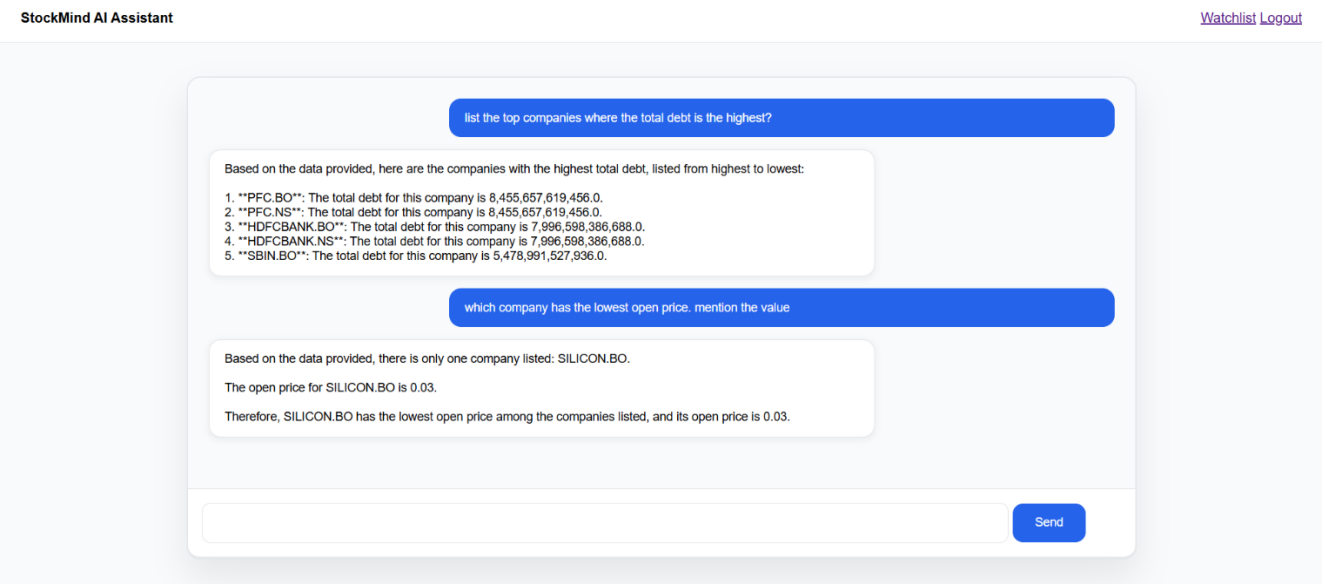

Figure 5.5 : Chat Assistant

5.6 REAL TIME ALERT NOTIFICATION

The alert interface displays instant system generated events, including:

- **Stock Update Alerts:**
  Indicates changes in tracked financial metrics.
- **New Stock Insertion Alerts:**
  Notifies when new companies are added to the watchlist.
- **Stock Deletion Alerts:**
  Confirms removal of selected stocks from monitoring.

- **Metric Change Alerts:**

  Shows before and after values for key financial indicators.

This output demonstrates:

- Continuous metric monitoring.
- Automatic detection of significant changes.
- Real time streaming of notifications using server-side events.
- Centralized alert history for user review.

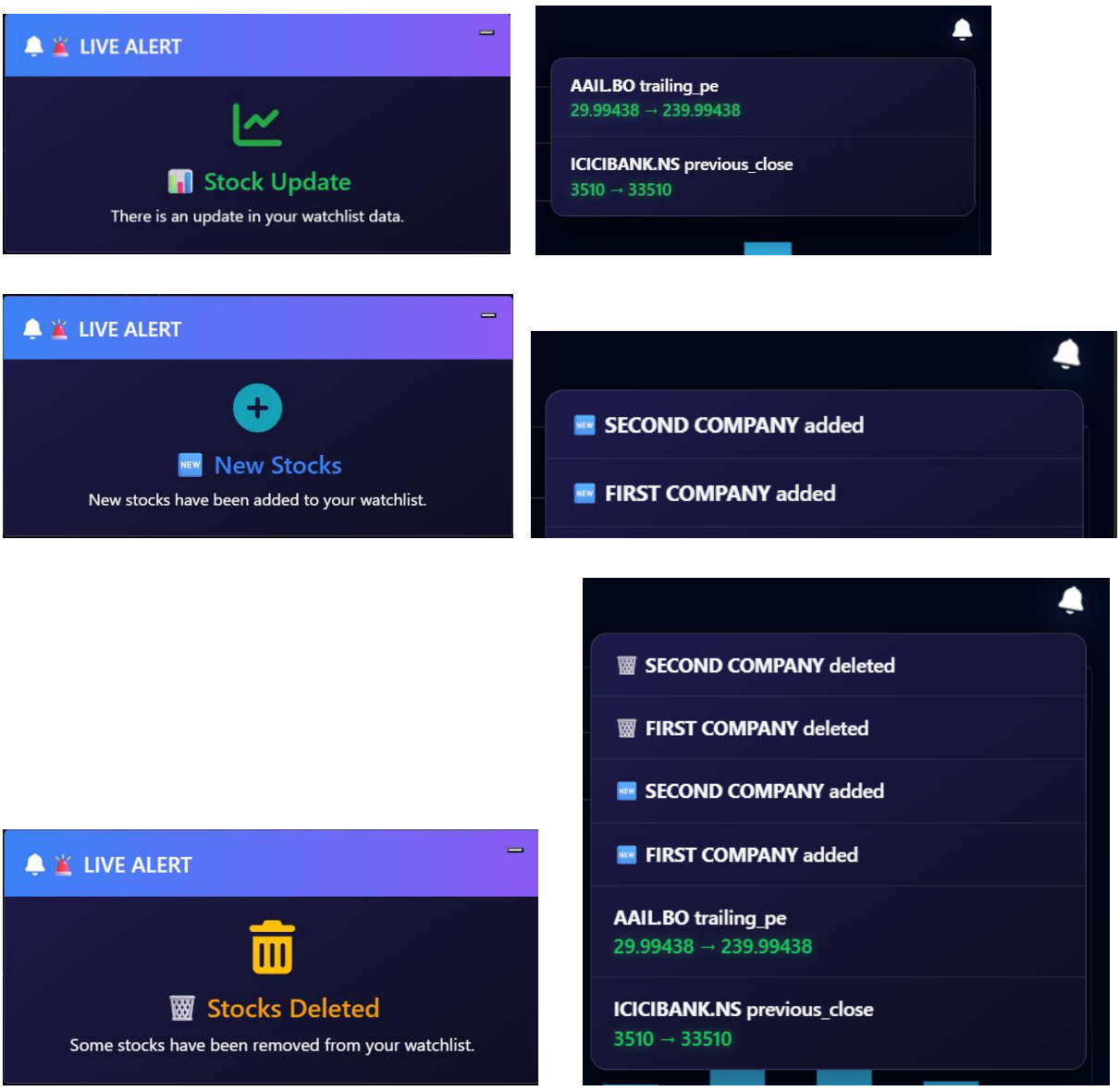

Figure 5.6 : Alert Notification Panel

CHAPTER 6

FUTURE ENHANCEMENTS

The current implementation of the StockMind AI platform successfully delivers intelligent stock screening, portfolio analytics, and real time financial insights. However, several enhancements can be incorporated to further improve the system's analytical depth, usability, and scalability in practical investment environments.

To enhance analytical richness, additional financial indicators such as dividend yield, free cash flow ratios, technical indicators, and risk metrics can be integrated into the screening and visualization modules. This would enable more comprehensive evaluation of investment opportunities beyond basic fundamental measures.

The user interface can be further improved by introducing customizable dashboards where users can select preferred metrics, rearrange visual components, and save personalized analysis views. Such flexibility would significantly improve user engagement and analytical efficiency.

The alerting mechanism can be extended to support advanced rule based strategies, allowing users to define compound conditions across multiple financial parameters. This would enable more sophisticated monitoring and automated investment signal generation.

Scalability can be improved by introducing distributed data processing and caching layers to handle high volume market data and concurrent user requests more efficiently. This would support deployment in enterprise grade environments with large user bases.

To enhance explainability, the AI query engine can incorporate richer reasoning outputs, including step wise analytical justifications and comparative financial summaries. This would improve transparency and user confidence in system generated insights.

Historical data analytics and trend forecasting capabilities can be added to support long term performance evaluation and scenario analysis. Time series based

projections would assist users in strategic decision making.

Multilingual support within the interface can be introduced to improve accessibility for a broader range of users. Providing regional language options would make the platform more inclusive and practical in diverse financial markets.

Finally, the platform can be extended to support mobile and cloud based deployments, enabling seamless access across devices and locations while maintaining consistent analytical functionality.

CONCLUSION

In conclusion, the StockMind AI platform represents a significant advancement in intelligent financial analytics and decision support systems. By integrating real time market data processing, structured portfolio and watchlist analytics, and an AI driven natural language query engine, the system demonstrates a practical and efficient approach to modern investment analysis.

This solution reduces the complexity traditionally associated with financial screening and portfolio evaluation by enabling users to interact with data through intuitive visual dashboards and flexible analytical queries. The unified backend workflow, secure data management, and responsive alert mechanisms ensure reliable and timely delivery of financial insights.

In essence, StockMind AI not only enhances analytical efficiency but also contributes a scalable and extensible framework for data driven investment intelligence. With its modular architecture, robust processing pipeline, and user centric design, the platform establishes a strong foundation for future innovation in AI assisted financial systems and real-world advisory applications.

APPENDIX

CODE SNIPPETS

app.py

```
# ---- ALERT BASELINE STATE ----
last_snapshot = {}
baseline_initialized = False

from flask import Flask, request, jsonify, render_template
import jwt
import datetime
import os
import re
import time

from db import get_connection
from sentence_transformers import SentenceTransformer
from google import genai

from dotenv import load_dotenv
load_dotenv()  # Load .env variables

app = Flask(__name__)
app.secret_key = os.getenv('SECRET_KEY', 'fallback-dev-key')

JWT_SECRET =
"ba3dcd1c8a2d8ad6e79f309d9ee9d48470e9395a03adb966cf2d76a263391531"

# ---------------- EMBEDDING MODEL ----------------
embed_model = SentenceTransformer("all-MiniLM-L6-v2")

# ---------------- GEMINI CONFIG ----------------
os.environ["GEMINI_API_KEY"] =
"AIzaSyCQsDAwsKZcYGc7ANwjzEDSuPidFiPhOFA"
client = genai.Client(api_key=os.environ["GEMINI_API_KEY"])

from flask import Flask, request, jsonify, render_template, redirect, url_for, session
from authlib.integrations.flask_client import OAuth
import os

# OAuth setup
oauth = OAuth(app)
```

```
# Google OAuth
google = oauth.register(
    name='google',
    client_id=os.getenv('GOOGLE_CLIENT_ID'),
    client_secret=os.getenv('GOOGLE_CLIENT_SECRET'),
    authorize_url='https://accounts.google.com/o/oauth2/auth',
    access_token_url='https://oauth2.googleapis.com/token',
    api_base_url='https://www.googleapis.com/oauth2/v1/',
    server_metadata_url='https://accounts.google.com/.well-known/openid-configuration',
    client_kwargs={'scope': 'openid email profile'}
)

# GitHub OAuth
github = oauth.register(
    name='github',
    client_id=os.getenv('GITHUB_CLIENT_ID', 'your-github-client-id'),
    client_secret=os.getenv('GITHUB_CLIENT_SECRET', 'your-github-secret'),
    access_token_url='https://github.com/login/oauth/access_token',
    authorize_url='https://github.com/login/oauth/authorize',
    api_base_url='https://api.github.com/',
    client_kwargs={'scope': 'read:user user:email'}
)

# Google OAuth Routes
@app.route('/auth/google')
def google_login():
    redirect_uri = url_for('google_callback', _external=True)
    return google.authorize_redirect(redirect_uri)

@app.route('/auth/google/callback')
def google_callback():
    try:
        # Get access token
        token = google.authorize_access_token()

        # Use Google's userinfo endpoint DIRECTLY (no Authlib parsing)
        resp = google.get('https://www.googleapis.com/oauth2/v2/userinfo', token=token)
        user_info = resp.json()

        session['user'] = {
            'email': user_info['email'],
            'name': user_info.get('name', 'Google User'),
            'provider': 'google'
        }

        # saving credentials to authentication table
```

```
        conn = get_connection()
        cur = conn.cursor()
        cur.execute("SELECT uname FROM authentication WHERE email=%s",
(user_info['email'],))
        existing = cur.fetchone()
        if not existing:
            cur.execute(
                "INSERT INTO authentication (uname, email, password, phone_num) VALUES
(%s, %s, %s, %s)",
                (user_info.get('name', 'oauth_user'), user_info['email'], 'not visible', 0)
            )
            conn.commit()
        cur.close()
        conn.close()

        return redirect(url_for('watchlist_page'))

    except Exception as e:
        print(f"GOOGLE ERROR: {e}")
        return jsonify({"error": f"Google login failed: {str(e)}"}), 400

@app.route('/auth/github')
def github_login():
    redirect_uri = url_for('github_callback', _external=True)
    return github.authorize_redirect(redirect_uri)

@app.route('/auth/github/callback')
def github_callback():
    try:
        # Get access token
        token = github.authorize_access_token()

        # Get user info
        resp = github.get('user', token=token)
        user_info = resp.json()

        session['user'] = {
            'email': user_info.get('email') or f"{user_info['login']}@github.com",
            'name': user_info.get('name') or user_info['login'],
            'provider': 'github'
        }

        # saving credentials to authentication table
        conn = get_connection()
        cur = conn.cursor()
        cur.execute("SELECT uname FROM authentication WHERE email=%s",
```

```
(session['user']['email'],))
        existing = cur.fetchone()
        if not existing:
            cur.execute(
                "INSERT INTO authentication (uname, email, password, phone_num) VALUES
(%s, %s, %s, %s)",
                (session['user']['name'], session['user']['email'], 'not visible', 1)
            )
            conn.commit()
        cur.close()
        conn.close()

        return redirect(url_for('watchlist_page'))

    except Exception as e:
        print(f"GITHUB ERROR: {e}")
        return jsonify({"error": f"GitHub login failed: {str(e)}"}), 400

# Logout route
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('home'))

import math

def norm(v):
    if v is None:
        return None
    return round(float(v), 6)

def safe_float(x):
    if x is None or math.isinf(x) or math.isnan(x):
        return None
    return float(x)

# ---------------- HOME ----------------
@app.route("/")
def home():
    return render_template("login.html")

@app.route("/debug/db")
def debug_db():
    conn = get_connection()
    cur = conn.cursor()
    cur.execute("SELECT COUNT(*) FROM watchlist_fundamentals")
```

```
    count = cur.fetchone()[0]
    cur.close()
    conn.close()
    return {"row_count": count}

#---------------- CHAT PAGE ----------------#
@app.route("/chat")
def chat_page():
    return render_template("chat.html")

# ======================== CHAT HELPERS ========================

def is_safe_select_sql(sql: str) -> bool:
    sql_l = sql.lower().strip()
    if not sql_l.startswith("select"):
        return False
    forbidden = ["delete", "update", "insert", "drop", "alter", "truncate"]
    return not any(word in sql_l for word in forbidden)

def generate_sql_from_question(question: str) -> str:
    prompt = f"""
You are a PostgreSQL SQL generator.

Table: watchlist_fundamentals (5829 rows)

Columns (ALL available):
symbol, revenue_per_share, trailing_pe, earnings_quarterly_growth, previous_close,
open_price, day_low, day_high, volume, trailing_eps, peg_ratio, ebitda,
total_debt, total_revenue, debt_to_equity, earnings_growth, revenue_growth

EXAMPLES:
"top 5 total debt" → SELECT symbol, total_debt FROM watchlist_fundamentals ORDER
BY total_debt DESC LIMIT 5
"highest revenue" → SELECT symbol, total_revenue FROM watchlist_fundamentals
ORDER BY total_revenue DESC LIMIT 5
"avg pe ratio" → SELECT AVG(trailing_pe) as avg_pe FROM watchlist_fundamentals

RULES:
- ALWAYS generate SQL for ANY column mentioned
- Use ORDER BY column DESC LIMIT 5 for "top"/"highest"/"maximum"
- Use AVG() for averages
- NEVER return "NONE"

Question: {question}

SQL:"""
```

```
    try:
        response = client.models.generate_content(
            model="gemini-2.5-flash-lite",
            contents=prompt
        )
        return response.text.strip()
    except Exception:
        return "NONE"

def explain_sql_result(question: str, rows: list) -> str:
    # SHOW ALL ROWS
    data_text = ""
    for r in rows:
        data_text += ", ".join(f"{k}: {v}" for k, v in r.items()) + "\n"

    prompt = f"""
You are a financial assistant.

Answer the user's question using ONLY the data below.
Explain in simple, beginner friendly English.
Be detailed but clear.
Do NOT add extra facts.
Do NOT guess numbers.
List ALL companies and their exact values.

Data ({len(rows)} companies found):
{data_text}

Question:
{question}

Detailed Answer:
"""
    try:
        response = client.models.generate_content(
            model="gemini-2.5-flash-lite",
            contents=prompt
        )
        return response.text.strip()
    except Exception as e:
        print(f"Gemini error: {e}")
        # FALLBACK: Raw data if Gemini fails
        return f"Found {len(rows)} companies:\n" + data_text[:2000]

# ======================== CHAT API ========================
```

```
@app.route("/ask", methods=["POST"])
def ask():
    try:
        user_question = request.json.get("question", "").strip()
        if not user_question:
            return jsonify({"answer": "Please ask a question."})

        print(f"🤖 Question: {user_question}")

        # ---------- STEP 1: TRY TEXT TO SQL ----------
        sql = generate_sql_from_question(user_question)
        print(f"🔍 Generated SQL: {sql}")

        if sql != "NONE" and is_safe_select_sql(sql):
            try:
                conn = get_connection()
                cur = conn.cursor()
                cur.execute(sql)
                cols = [d[0] for d in cur.description]
                rows = [dict(zip(cols, r)) for r in cur.fetchall()]
                cur.close()
                conn.close()

                print(f"📊 Found {len(rows)} rows")

                if rows:
                    answer = explain_sql_result(user_question, rows)
                    return jsonify({"answer": answer})

            except Exception as e:
                print(f"SQL ERROR: {e}")
                return jsonify({"answer": f"Database error: {str(e)}"})

        # ---------- STEP 2: DYNAMIC RAG - ALL ROWS, ALL COLUMNS ----------
        print("📝 Falling back to RAG (ALL DATA)...")

        # 1. Get ALL columns dynamically
        conn = get_connection()
        cur = conn.cursor()

        # Get table schema
        cur.execute("""
            SELECT column_name, data_type
            FROM information_schema.columns
```

```
            WHERE table_name = 'watchlist_fundamentals'
            ORDER BY ordinal_position
        """)
        columns_info = cur.fetchall()
        all_columns = [row[0] for row in columns_info]
        print(f"📋 Found {len(all_columns)} columns: {all_columns}")

        # 2. Get ALL rows (no LIMIT)
        cur.execute(f"SELECT {', '.join(all_columns)} FROM watchlist_fundamentals ORDER
BY symbol")
        all_rows = cur.fetchall()
        print(f"📊 Loaded {len(all_rows)} total rows")

        cur.close()
        conn.close()

        # 3. Smart context: Top values + aggregates for each numeric column
        context = "=== FULL DATASET SUMMARY ===\n"
        context += f"Total companies: {len(all_rows)}\n\n"

        numeric_cols = [col for col in all_columns if col != 'symbol']

        for col in numeric_cols[:10]:  # Top 10 metrics to fit token limit
            # Get top 5 values
            conn = get_connection()
            cur = conn.cursor()
            cur.execute(f"""
                SELECT symbol, {col}
                FROM watchlist_fundamentals
                WHERE {col} IS NOT NULL
                ORDER BY {col} DESC NULLS LAST
                LIMIT 5
            """)
            top_rows = cur.fetchall()

            context += f"🏆 TOP 5 {col.upper()}:\n"
            for r in top_rows:
                context += f"  {r[0]}: {r[1]}\n"

            # Aggregates
            cur.execute(f"SELECT AVG({col}), MAX({col}), MIN({col}) FROM
watchlist_fundamentals")
            avg, maxv, minv = cur.fetchone()
            context += f"  AVG: {avg:.2f}, MAX: {maxv}, MIN: {minv}\n\n"

            cur.close()
```

```
            conn.close()

        # 4. Recent/alphabetical sample for reference
        context += "=== SAMPLE (first 5 companies alphabetically):\n"
        conn = get_connection()
        cur = conn.cursor()
        cur.execute(f"SELECT {', '.join(['symbol'] + numeric_cols[:5])} FROM
watchlist_fundamentals ORDER BY symbol LIMIT 5")
        sample_rows = cur.fetchall()
        for r in sample_rows:
            context += f"{r[0]}: " + ", ".join(f"{numeric_cols[i]}={r[i+1]}" for i in range(5) if
r[i+1]) + "\n"
        cur.close()
        conn.close()

        prompt = f"""
        You are a financial assistant. Answer using ONLY this COMPLETE dataset summary.

        FULL DATASET ({len(all_rows)} companies, {len(all_columns)} columns):
        {context}

        Question: {user_question}

        Answer using the TOP values and aggregates shown above. List exact companies and
numbers.
        """

        response = client.models.generate_content(
            model="gemini-2.5-flash-lite",
            contents=prompt
        )
        return jsonify({"answer": response.text.strip()})

    except Exception as e:
        print(f"ASK ERROR: {e}")
        return jsonify({"answer": f"Server error."}), 500


# ---------------- LOGIN ----------------
@app.route("/login", methods=["GET"])
def show_login():
    return render_template("login.html")

@app.route("/login", methods=["POST"])
def login():
    email = request.form.get("email")
```

```
    password = request.form.get("password")

    conn = get_connection()
    cur = conn.cursor()

    cur.execute("SELECT uname, password FROM authentication WHERE email=%s",
(email,))
    user = cur.fetchone()

    if not user or password != user[1]:
        return jsonify({"message": "Invalid credentials"}), 400

    token = jwt.encode(
        {
            "uname": user[0],
            "email": email,
            "exp": datetime.datetime.utcnow() + datetime.timedelta(hours=1)
        },
        JWT_SECRET,
        algorithm="HS256"
    )

    cur.close()
    conn.close()
    return jsonify({"token": token})

#--------------- SIGN UP PAGE ----------------#
@app.route("/signup", methods=["POST"])
def signup():
    username = request.form.get("username")
    password = request.form.get("password")
    email = request.form.get("email")
    phone_num = request.form.get("phone_num")

    if not all([username, password, email, phone_num]):
        return jsonify({"message": "All fields are required"}), 400

    conn = get_connection()
    cur = conn.cursor()

    # ---- CHECK ALL DUPLICATES ----
    cur.execute("SELECT email FROM authentication WHERE email=%s", (email,))
    if cur.fetchone():
        cur.close()
        conn.close()
        return jsonify({"message": "Email already registered"}), 400
```

```
    cur.execute("SELECT uname FROM authentication WHERE uname=%s", (username,))
    if cur.fetchone():
        cur.close()
        conn.close()
        return jsonify({"message": "Username already taken"}), 400

    cur.execute("SELECT phone_num FROM authentication WHERE phone_num=%s",
(phone_num,))
    if cur.fetchone():
        cur.close()
        conn.close()
        return jsonify({"message": "Phone number already registered"}), 400

    # Insert new user
    cur.execute(
        "INSERT INTO authentication (uname, password, email, phone_num) VALUES (%s,
%s, %s, %s)",
        (username, password, email, phone_num)
    )
    conn.commit()
    cur.close()
    conn.close()

    return jsonify({"message": "Account created successfully"}), 200

@app.route("/signup")
def signup_page():
    return render_template("signup.html")

# ---------------- WATCHLIST PAGE ----------------
@app.route("/watchlist")
def watchlist_page():
    return render_template("watchlist.html")
#-- portfolio---
@app.route("/portfolio")
def portfolio_page():
    return render_template("portfolio.html")
#market
@app.route("/market")
def market_page():
    return render_template("market.html")

#-----------------MARKET API----------------#
import requests
```

```
ALPHA_API_KEY = os.environ.get("ALPHA_VANTAGE_API_KEY")

@app.route("/api/market-data")
def market_data():
    symbol = request.args.get("symbol", "NSEI")  # default index
    function = "TIME_SERIES_INTRADAY"
    interval = "60min"

    url = (
        f"https://www.alphavantage.co/query"
        f"?function={function}&symbol={symbol}"
        f"&interval={interval}&apikey={ALPHA_API_KEY}"
    )

    r = requests.get(url)
    if r.status_code != 200:
        return jsonify({"error": "Market API failed"}), 500

    data = r.json()
    return jsonify(data)

# ---------------- COMPANIES API ----------------
@app.route("/api/companies")
def get_companies():
    conn = get_connection()
    cur = conn.cursor()

    cur.execute("""
        SELECT DISTINCT symbol
        FROM watchlist_fundamentals
        ORDER BY symbol
    """)

    companies = [r[0] for r in cur.fetchall()]
    cur.close()
    conn.close()

    return jsonify(companies)

@app.route("/api/data")
def data():
    conn = get_connection()
    cur = conn.cursor()

    cur.execute("""
        SELECT symbol, open_price
```

```
        FROM watchlist_fundamentals
        ORDER BY symbol
        LIMIT 20
    """)
    rows = cur.fetchall()

    return jsonify({
        "symbols": [r[0] for r in rows],
        "values": [r[1] for r in rows]
    })
# ---------------- WATCHLIST DATA API ----------------
@app.route("/api/watchlist-data")
def watchlist_data():
    company = request.args.get("company", "All")

    conn = get_connection()
    cur = conn.cursor()

    # ---- KPI AVERAGES ----
    if company == "All":
        cur.execute("""
            SELECT
                COALESCE(AVG(open_price), 0),
                COALESCE(AVG(previous_close), 0),
                COALESCE(AVG(trailing_pe), 0),
                COALESCE(AVG(debt_to_equity), 0)
            FROM watchlist_fundamentals
        """)
    else:
        cur.execute("""
            SELECT
                COALESCE(AVG(open_price), 0),
                COALESCE(AVG(previous_close), 0),
                COALESCE(AVG(trailing_pe), 0),
                COALESCE(AVG(debt_to_equity), 0)
            FROM watchlist_fundamentals
            WHERE symbol = %s
        """, (company,))

    kpi = cur.fetchone()

    # ---- CHART DATA ----
    if company == "All":
        cur.execute("""
            SELECT
                symbol,
```

```
            COALESCE(open_price, 0),
            COALESCE(previous_close, 0),
            COALESCE(trailing_pe, 0),
            COALESCE(peg_ratio, 0),
            COALESCE(earnings_growth, 0),
            COALESCE(revenue_growth, 0),
            COALESCE(debt_to_equity, 0)
        FROM watchlist_fundamentals
        ORDER BY symbol
    """)
else:
    cur.execute("""
        SELECT
            symbol,
            COALESCE(open_price, 0),
            COALESCE(previous_close, 0),
            COALESCE(trailing_pe, 0),
            COALESCE(peg_ratio, 0),
            COALESCE(earnings_growth, 0),
            COALESCE(revenue_growth, 0),
            COALESCE(debt_to_equity, 0)
        FROM watchlist_fundamentals
        WHERE symbol = %s
    """, (company,))

rows = cur.fetchall()
cur.close()
conn.close()

return jsonify({
    "avg_open": safe_float(kpi[0]),
    "avg_previous_close": safe_float(kpi[1]),
    "avg_trailing_pe": safe_float(kpi[2]),
    "avg_debt_to_equity": safe_float(kpi[3]),

    "symbols": [r[0] for r in rows],
    "open_price": [safe_float(r[1]) for r in rows],
    "previous_close": [safe_float(r[2]) for r in rows],
    "trailing_pe": [safe_float(r[3]) for r in rows],
    "peg_ratio": [safe_float(r[4]) for r in rows],
    "earnings_growth": [safe_float(r[5]) for r in rows],
    "revenue_growth": [safe_float(r[6]) for r in rows],
    "debt_to_equity": [safe_float(r[7]) for r in rows]
})

#------------------ ALERTS ----------------#
```

```
from flask import Response
import json
import time
from collections import defaultdict

# Global storage for recent changes
recent_changes = []

@app.route("/api/alerts")
def alerts():
    def generate():
        print("📡 SSE client connected")
        global recent_changes
        last_count = 0
        while True:
            if len(recent_changes) > last_count:
                new_alerts = recent_changes[last_count:]
                for alert in new_alerts:
                    print(f"🔔 SENDING: {alert}")
                    yield f"data: {json.dumps(alert)}\n\n"
                last_count = len(recent_changes)
            time.sleep(1)
    return Response(generate(), mimetype="text/event-stream")

import threading
from copy import deepcopy

last_snapshot = {}
  # poll every 5 seconds

import threading
from copy import deepcopy

def poll_watchlist_changes():
    print("🔁 Watchlist poller STARTED")
    global last_snapshot, recent_changes, baseline_initialized

    while True:
        try:
            conn = get_connection()
            cur = conn.cursor()
            cur.execute("""
                SELECT
                symbol,
                open_price,
                previous_close,
```

```
    trailing_pe,
    peg_ratio,
    earnings_growth,
    revenue_growth,
    debt_to_equity,
    revenue_per_share,
    earnings_quarterly_growth,
    day_low,
    day_high,
    volume,
    trailing_eps,
    ebitda,
    total_debt,
    total_revenue
    FROM watchlist_fundamentals
""")
rows = cur.fetchall()
cur.close()
conn.close()

print("🔁 Polling DB… rows:", len(rows))

current = {
    r[0]: {
        "open_price": norm(r[1]),
        "previous_close": norm(r[2]),
        "trailing_pe": norm(r[3]),
        "peg_ratio": norm(r[4]),
        "earnings_growth": norm(r[5]),
        "revenue_growth": norm(r[6]),
        "debt_to_equity": norm(r[7]),
        "revenue_per_share": norm(r[8]),
        "earnings_quarterly_growth": norm(r[9]),
        "day_low": norm(r[10]),
        "day_high": norm(r[11]),
        "volume": norm(r[12]),
        "trailing_eps": norm(r[13]),
        "ebitda": norm(r[14]),
        "total_debt": norm(r[15]),
        "total_revenue": norm(r[16])
    }
    for r in rows
}

if not baseline_initialized:
    last_snapshot = deepcopy(current)
```

```
            baseline_initialized = True
            print("✅ Baseline snapshot initialized")
            time.sleep(5)
            continue

        # 1. NEW COMPANIES (current - last_snapshot)
        for symbol in current:
            if symbol not in last_snapshot:
                recent_changes.append({"type": "new_company", "symbol": symbol})
                print(f"🆕 NEW COMPANY: {symbol}")

        # 2. METRIC CHANGES
        for symbol, metrics in current.items():
            if symbol in last_snapshot:
                for k, v in metrics.items():
                    old_v = norm(last_snapshot[symbol].get(k))
                    new_v = norm(v)
                    if old_v != new_v and old_v is not None and new_v is not None:
                        recent_changes.append({
                            "type": "metric_update", "symbol": symbol,
                            "metric": k, "old": float(old_v), "new": float(new_v)
                        })
                        print(f"🔔 CHANGE: {symbol} {k} {old_v} → {new_v}")

        # 3. DELETED COMPANIES (last_snapshot - current)
        for symbol in list(last_snapshot.keys()):
            if symbol not in current:
                recent_changes.append({"type": "company_deleted", "symbol": symbol})
                print(f"🗑️ DELETED: {symbol}")

        # 4. UPDATE SNAPSHOT LAST
        last_snapshot = deepcopy(current)

    except Exception as e:
        print(f"❌ Poller ERROR: {e}")

    time.sleep(5)

#---------------- DEBUG ALERTS ----------------#

@app.route("/debug/alerts")
def debug_alerts():
    global recent_changes, last_snapshot
    return jsonify({
        "recent_changes_count": len(recent_changes),
```

```
        "recent_changes": recent_changes[-5:],  # Last 5
        "snapshot_count": len(last_snapshot),
        "poller_status": "running"
    })

# ---------------- RUN APP ----------------#
if __name__ == "__main__":
    threading.Thread(target=poll_watchlist_changes, daemon=True).start()
    app.run(port=3000, debug=False, use_reloader=False)
```

login.html

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
<style>
body {
    margin: 0;
    height: 100vh;
    background: #f1f5f9;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: "Inter", Arial, sans-serif;
}

.login-container {
    background: #ffffff;
    padding: 42px;
    width: 380px;
    border-radius: 14px;
    box-shadow: 0 20px 40px rgba(0,0,0,0.08);
}

h2 {
    text-align: center;
    color: #0f172a;
    margin-bottom: 26px;
    font-weight: 600;
}

input {
    width: 100%;
```

```
    padding: 14px;
    margin-top: 14px;
    border-radius: 8px;
    border: 1px solid #cbd5e1;
    font-size: 14px;
}

input:focus {
    outline: none;
    border-color: #2563eb;
}

button {
    width: 100%;
    padding: 14px;
    margin-top: 22px;
    border-radius: 8px;
    border: none;
    background: #2563eb;
    color: white;
    font-size: 15px;
    font-weight: 500;
    cursor: pointer;
}

button:hover {
    background: #1d4ed8;
}

.oauth-btn {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px;
    background: #f8fafc;
    color: #0f172a;
    border: 1px solid #cbd5e1;
    margin-top: 12px;
}

.oauth-btn svg {
    width: 18px;
    height: 18px;
}

.divider {
```

```
    margin: 22px 0;
    text-align: center;
    font-size: 13px;
    color: #64748b;
}

#token_display, #error_display {
    margin-top: 16px;
    padding: 10px;
    border-radius: 6px;
    font-size: 14px;
    display: none;
    text-align: center;
}

#token_display {
    background: #ecfeff;
    color: #0369a1;
}

#error_display {
    background: #fef2f2;
    color: #b91c1c;
}

a {
    display: block;
    text-align: center;
    margin-top: 18px;
    color: #2563eb;
    text-decoration: none;
    font-size: 14px;
}
</style>
</head>

<body>
<div class="login-container">
<h2>Sign in to your account</h2>

<form id="loginForm">
<input type="email" id="email" placeholder="Email address" required>
<input type="password" id="password" placeholder="Password" required>
<button type="submit">Sign In</button>
</form>
```

```
<div class="divider">or continue with</div>

<button class="oauth-btn" onclick="googleLogin()">
<!-- Google icon -->
<svg viewBox="0 0 48 48">
<path fill="#EA4335" d="M24 9.5c3.3 0 6.3 1.1 8.7 3.2l6.5-6.5C35.1 2.4 29.8 0 24 0 14.6 0
6.4 5.3 2.4 13l7.7 6C12.2 13.1 17.7 9.5 24 9.5z"/>
<path fill="#4285F4" d="M46.1 24.5c0-1.6-.1-2.8-.4-4.1H24v7.7h12.7c-.5 3-2.1 5.6-4.6
7.3l7.1 5.5c4.2-3.9 6.9-9.6 6.9-16.4z"/>
<path fill="#FBBC05" d="M10.1 28.9c-.6-1.8-.9-3.7-.9-5.6s.3-3.8.9-5.6l-7.7-6c-1.6 3.2-2.4
6.7-2.4 11.6s.8 8.4 2.4 11.6l7.7-6z"/>
<path fill="#34A853" d="M24 48c6.5 0 12-2.1 16-5.8l-7.1-5.5c-2 1.4-4.6 2.2-8.9 2.2-6.3 0-
11.8-3.6-14-8.5l-7.7 6C6.4 42.7 14.6 48 24 48z"/>
</svg>
Continue with Google
</button>

<button class="oauth-btn" onclick="githubLogin()">
<!-- GitHub icon -->
<svg viewBox="0 0 24 24" fill="#000">
<path d="M12 .5C5.7.5.5 5.7.5 12c0 5.1 3.3 9.4 7.9 10.9.6.1.8-.3.8-.6v-2.2c-3.2.7-3.9-1.5-
3.9-1.5-.5-1.3-1.3-1.7-1.3-1.7-1.1-.7.1-.7.1-.7 1.2.1 1.8 1.2 1.8 1.2 1.1 1.8 2.9 1.3 3.6 1 .1-
.8.4-1.3.7-1.6-2.6-.3-5.3-1.3-5.3-5.9 0-1.3.5-2.4 1.2-3.2-.1-.3-.5-1.5.1-3.1 0 0 1-.3 3.3
1.2a11.5 11.5 0 0 1 6 0C16.3 4.5 17.3 4.8 17.3 4.8c.6 1.6.2 2.8.1 3.1.8.8 1.2 1.9 1.2 3.2 0 4.6-
2.7 5.6-5.3 5.9.4.4.8 1 .8 2v3c0 .3.2.7.8.6 4.6-1.5 7.9-5.8 7.9-10.9C23.5 5.7 18.3.5 12 .5z"/>
</svg>
Continue with GitHub
</button>

<p id="token_display"></p>
<p id="error_display"></p>

<a href="/signup">Create a new account</a>
</div>

<script>
const email = document.getElementById("email");
const password = document.getElementById("password");
const token_display = document.getElementById("token_display");
const error_display = document.getElementById("error_display");

document.getElementById("loginForm").addEventListener("submit", async function(e) {
    e.preventDefault();
    let fd = new FormData();
    fd.append("email", email.value);
    fd.append("password", password.value);
```

```
    let res = await fetch("/login", { method: "POST", body: fd });
    let data = await res.json();

    if (res.ok) {
        token_display.style.display = "block";
        token_display.innerText = "Login successful. Redirecting...";
        error_display.style.display = "none";
        setTimeout(() => window.location.href = "/watchlist", 600);
    } else {
        error_display.style.display = "block";
        error_display.innerText = data.message;
    }
});

// Google OAuth
async function googleLogin() {
    try {
        // Redirect to your Flask Google OAuth endpoint
        window.location.href = "/auth/google";
    } catch (e) {
        error_display.style.display = "block";
        error_display.innerText = "Google login failed. Please try again.";
    }
}

// GitHub OAuth
async function githubLogin() {
    try {
        // Redirect to your Flask GitHub OAuth endpoint
        window.location.href = "/auth/github";
    } catch (e) {
        error_display.style.display = "block";
        error_display.innerText = "GitHub login failed. Please try again.";
    }
}
</script>

</body>
</html>
```

portfolio.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<title>Portfolio</title>
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/css/adminlte.min.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-
free/css/all.min.css">

<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<style>
body { background:#020617; }

/* SAME SIDEBAR COLLAPSE AS WATCHLIST */
.wrapper.sidebar-collapsed .main-sidebar { width:64px; }
.wrapper.sidebar-collapsed .nav-sidebar p { display:none; }
.wrapper.sidebar-collapsed .content-wrapper { margin-left:64px; }

.main-sidebar { transition:width .25s ease; }
.content-wrapper { transition:margin-left .25s ease; }

.sidebar-toggle {
  width:100%;
  padding:12px 16px;
  background:none;
  border:none;
  color:#e5e7eb;
  font-size:18px;
  text-align:left;
}
.sidebar-toggle:hover { background:rgba(255,255,255,.05); }

.topbar {
  height:56px;
  display:flex;
  align-items:center;
  padding-left:16px;
  border-bottom:1px solid rgba(255,255,255,.08);
  background:#020617;
  color:#e5e7eb;
  font-weight:600;
}
```

```
.content-wrapper { background:#020617; padding:16px; }
.card {
  background:#020617;
  color:#e5e7eb;
  border:1px solid rgba(255,255,255,.1);
}

.table td, .table th { border-color:rgba(255,255,255,.1); }

.badge-buy { background:#16a34a; }
.badge-hold { background:#facc15; color:#000; }
.badge-sell { background:#dc2626; }

.metric { font-size:13px; color:#94a3b8; }
.value { font-size:20px; font-weight:700; }
</style>
</head>

<body class="hold-transition sidebar-mini layout-fixed">

<div class="wrapper" id="appWrapper">

<!-- SIDEBAR -->
<aside class="main-sidebar sidebar-dark-primary elevation-4">
  <div class="sidebar">
    <button class="sidebar-toggle" id="toggleBtn">
      <i class="fas fa-bars"></i>
    </button>

    <nav>
      <ul class="nav nav-pills nav-sidebar flex-column">
        <li class="nav-item"><a href="/watchlist" class="nav-link"><i class="fas fa-list nav-
icon"></i><p>Watchlist</p></a></li>
        <li class="nav-item"><a href="/portfolio" class="nav-link active"><i class="fas fa-
briefcase nav-icon"></i><p>Portfolio</p></a></li>
        <li class="nav-item"><a href="/market" class="nav-link"><i class="fas fa-chart-line
nav-icon"></i><p>Market</p></a></li>
        <li class="nav-item"><a href="/chat" class="nav-link"><i class="fas fa-robot nav-
icon"></i><p>Chat</p></a></li>
        <li class="nav-item"><a href="/login" class="nav-link"><i class="fas fa-sign-out-alt
nav-icon"></i><p>Logout</p></a></li>
      </ul>
    </nav>
  </div>
</aside>
```

```
<!-- CONTENT -->
<div class="content-wrapper">
  <div class="topbar">StockMind AI</div>
  <div id="app"></div>
</div>

</div>

<script>
document.getElementById("toggleBtn").onclick = () => {
  document.getElementById("appWrapper").classList.toggle("sidebar-collapsed");
};
</script>

<script type="text/babel">
const {useState,useEffect} = React;

/* PORTFOLIO LOGIC UNCHANGED */
function Portfolio(){
  const [rows,setRows]=useState([]);
  const [qty,setQty]=useState({});

  useEffect(()=>{
    fetch("/api/watchlist-data?company=All")
    .then(r=>r.json())
    .then(d=>{
      setRows(d.symbols.map((s,i)=>({
        symbol:s,
        open:d.open_price[i],
        pe:d.trailing_pe[i],
        peg:d.peg_ratio[i],
        eg:d.earnings_growth[i],
        rg:d.revenue_growth[i],
        de:d.debt_to_equity[i]
      })));
    });
  },[]);

  const total=rows.reduce((s,r)=>s+(qty[r.symbol]||0)*r.open,0);
  const signal=s=>s>70?"BUY":s>45?"HOLD":"SELL";

  return(
  <>
    <div className="card p-3 mb-3">
      <div className="metric">Total Investment</div>
      <div className="value">₹ {total.toFixed(2)}</div>
```

```
    </div>

    <div className="card p-3">
      <table className="table table-dark table-sm">
        <thead>
          <tr><th>Symbol</th><th>Price</th><th>Qty</th><th>Value</th><th>Signal</th></t
r>
        </thead>
        <tbody>
        {rows.map(r=>{
          const q=qty[r.symbol]||0;
          const v=q*r.open;
          const s=signal(v);
          return(
            <tr key={r.symbol}>
              <td>{r.symbol}</td>
              <td>{r.open}</td>
              <td><input type="number" className="form-control form-control-sm"
                value={q} onChange={e=>setQty({...qty,[r.symbol]:+e.target.value})}/></td>
              <td>{v.toFixed(2)}</td>
              <td><span className={`badge ${s==="BUY"?"badge-buy":s==="HOLD"?"badge-
hold":"badge-sell"}`}>{s}</span></td>
            </tr>
          );
        })}
        </tbody>
      </table>
    </div>
  </>
  );
}

ReactDOM.createRoot(document.getElementById("app")).render(<Portfolio/>);
</script>

</body>
</html>
```

signup.html

```
<!DOCTYPE html>
<html>
<head>
<title>Sign Up</title>
<style>
/* [KEEP ALL EXISTING STYLES UNCHANGED] */
```

```
body {
    margin: 0;
    height: 100vh;
    background: #f1f5f9;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: "Inter", Arial, sans-serif;
}

.signup-container {
    background: #ffffff;
    padding: 42px;
    width: 380px;
    border-radius: 14px;
    box-shadow: 0 20px 40px rgba(0,0,0,0.08);
}

h2 {
    text-align: center;
    color: #0f172a;
    margin-bottom: 26px;
    font-weight: 600;
}

input {
    width: 100%;
    padding: 14px;
    margin-top: 14px;
    border-radius: 8px;
    border: 1px solid #cbd5e1;
}

button {
    width: 100%;
    padding: 14px;
    margin-top: 22px;
    border-radius: 8px;
    border: none;
    background: #2563eb;
    color: white;
    font-size: 15px;
    font-weight: 500;
}

.oauth-btn {
```

```
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px;
    background: #f8fafc;
    color: #0f172a;
    border: 1px solid #cbd5e1;
    margin-top: 12px;
}

.oauth-btn svg {
    width: 18px;
    height: 18px;
}

.divider {
    margin: 22px 0;
    text-align: center;
    font-size: 13px;
    color: #64748b;
}

#msg, #error {
    margin-top: 16px;
    padding: 10px;
    border-radius: 6px;
    font-size: 14px;
    display: none;
    text-align: center;
}

#msg {
    background: #ecfeff;
    color: #0369a1;
}

#error {
    background: #fef2f2;
    color: #b91c1c;
}

a {
    display: block;
    margin-top: 18px;
    text-align: center;
    color: #2563eb;
```

```
}
</style>
</head>

<body>
<div class="signup-container">
<h2>Create your account</h2>

<form id="signupForm">
<input id="new_username" placeholder="Username" required>
<input type="password" id="new_password" placeholder="Password" required>
<input type="email" id="new_email" placeholder="Email" required>
<input id="new_phone" placeholder="Phone number" required>
<button type="submit">Sign Up</button>
</form>

<div class="divider">or continue with</div>

<button class="oauth-btn" onclick="googleSignup()">
<!-- Google icon (reuse same SVG) -->
<svg viewBox="0 0 48 48">
<path fill="#EA4335" d="M24 9.5c3.3 0 6.3 1.1 8.7 3.2l6.5-6.5C35.1 2.4 29.8 0 24 0 14.6 0
6.4 5.3 2.4 13l7.7 6C12.2 13.1 17.7 9.5 24 9.5z"/>
<path fill="#4285F4" d="M46.1 24.5c0-1.6-.1-2.8-.4-4.1H24v7.7h12.7c-.5 3-2.1 5.6-4.6
7.3l7.1 5.5c4.2-3.9 6.9-9.6 6.9-16.4z"/>
<path fill="#FBBC05" d="M10.1 28.9c-.6-1.8-.9-3.7-.9-5.6s.3-3.8.9-5.6l-7.7-6c-1.6 3.2-2.4
6.7-2.4 11.6s.8 8.4 2.4 11.6l7.7-6z"/>
<path fill="#34A853" d="M24 48c6.5 0 12-2.1 16-5.8l-7.1-5.5c-2 1.4-4.6 2.2-8.9 2.2-6.3 0-
11.8-3.6-14-8.5l-7.7 6C6.4 42.7 14.6 48 24 48z"/>
</svg>
Continue with Google
</button>

<button class="oauth-btn" onclick="githubSignup()">
<!-- GitHub icon -->
<svg viewBox="0 0 24 24" fill="#000">
<path d="M12 .5C5.7.5.5 5.7.5 12c0 5.1 3.3 9.4 7.9 10.9.6.1.8-.3.8-.6v-2.2c-3.2.7-3.9-1.5-
3.9-1.5-.5-1.3-1.3-1.7-1.3-1.7-1.1-.7.1-.7.1-.7 1.2.1 1.8 1.2 1.8 1.2 1.1 1.8 2.9 1.3 3.6 1 .1-
.8.4-1.3.7-1.6-2.6-.3-5.3-1.3-5.3-5.9 0-1.3.5-2.4 1.2-3.2-.1-.3-.5-1.5.1-3.1 0 0 1-.3 3.3
1.2a11.5 11.5 0 0 1 6 0C16.3 4.5 17.3 4.8 17.3 4.8c.6 1.6.2 2.8.1 3.1.8.8 1.2 1.9 1.2 3.2 0 4.6-
2.7 5.6-5.3 5.9.4.4.8 1 .8 2v3c0 .3.2.7.8.6 4.6-1.5 7.9-5.8 7.9-10.9C23.5 5.7 18.3.5 12 .5z"/>
</svg>
Continue with GitHub
</button>

<p id="msg"></p>
```

```
<p id="error"></p>

<a href="/login">Already have an account? Sign in</a>
</div>

<script>
const new_username = document.getElementById("new_username");
const new_password = document.getElementById("new_password");
const new_email = document.getElementById("new_email");
const new_phone = document.getElementById("new_phone");
const msg = document.getElementById("msg");
const error = document.getElementById("error");

document.getElementById("signupForm").addEventListener("submit", async e => {
   e.preventDefault();
   let fd = new FormData();
   fd.append("username", new_username.value);
   fd.append("password", new_password.value);
   fd.append("email", new_email.value);
   fd.append("phone_num", new_phone.value);

   let res = await fetch("/signup", { method: "POST", body: fd });
   let data = await res.json();

   if (res.ok) {
      msg.style.display = "block";
      msg.innerText = "Account created. Redirecting...";
      error.style.display = "none";
      setTimeout(() => location.href = "/login", 1200);
   } else {
      error.style.display = "block";
      error.innerText = data.message || data.error;
   }
});

// Google OAuth Signup
function googleSignup() {
   window.location.href = "/auth/google";
}

// GitHub OAuth Signup
function githubSignup() {
   window.location.href = "/auth/github";
}
</script>
```

```
</body>
</html>
```

watchlist.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Watchlist</title>
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/admin-
lte@3.2/dist/css/adminlte.min.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-
free/css/all.min.css">

<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<style>
  /* ===== ALERT BELL UPGRADE ===== */
.notification-bell {
  position: relative;
  cursor: pointer;
  filter: drop-shadow(0 2px 8px rgba(59, 130, 246, 0.4));
  transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}

.notification-bell:hover {
  transform: scale(1.1) rotate(10deg);
  filter: drop-shadow(0 4px 12px rgba(59, 130, 246, 0.6));
}

.notification-bell i {
  font-size: 20px;
  color: #ffffff !important;  /* WHITE BELL ICON */
  text-shadow: 0 1px 3px rgba(0,0,0,0.5);
}

.notification-dot {
  position: absolute;
  top: -2px;
  right: -2px;
  width: 12px;
```

```
  height: 12px;
  background: linear-gradient(45deg, #ef4444, #f87171);
  border: 2px solid #ffffff;
  border-radius: 50%;
  box-shadow: 0 2px 8px rgba(239, 68, 68, 0.6);
  animation: pulse 2s infinite;
}

@keyframes pulse {
  0% { box-shadow: 0 0 0 0 rgba(239, 68, 68, 0.7); }
  70% { box-shadow: 0 0 0 10px rgba(239, 68, 68, 0); }
  100% { box-shadow: 0 0 0 0 rgba(239, 68, 68, 0); }
}

.notification-panel {
  position: absolute;
  right: 0;
  top: 35px;
  width: 360px;
  max-height: 400px;
  background: linear-gradient(145deg, #1e1b4b, #0f0f23);
  border: 1px solid rgba(255,255,255,0.15);
  border-radius: 12px;
  z-index: 9999;
  box-shadow: 0 20px 40px rgba(0,0,0,0.5);
  backdrop-filter: blur(10px);
  animation: slideDown 0.3s ease-out;
}

@keyframes slideDown {
  from { opacity: 0; transform: translateY(-10px); }
  to { opacity: 1; transform: translateY(0); }
}

.notification-item {
  padding: 12px 16px;
  font-size: 14px;
  border-bottom: 1px solid rgba(255,255,255,0.08);
  color: #ffffff !important;  /* WHITE TEXT */
  font-weight: 500;
  transition: background 0.2s;
}

.notification-item:hover {
  background: rgba(255,255,255,0.05);
}
```

```
.notification-item:last-child {
  border-bottom: none;
}

.green {
  color: #22c55e !important;
  font-weight: 600;
  text-shadow: 0 0 8px rgba(34, 197, 94, 0.5);
}

.red {
  color: #ef4444 !important;
  font-weight: 600;
  text-shadow: 0 0 8px rgba(239, 68, 68, 0.5);
}

b {
  color: #f8fafc !important;  /* BRIGHT WHITE COMPANY NAMES */
  font-weight: 700;
}

.green { color: #22c55e; }
.red { color: #ef4444; }

body { background:#020617; }

.wrapper.sidebar-collapsed .main-sidebar { width:64px; }
.wrapper.sidebar-collapsed .nav-sidebar p { display:none; }
.wrapper.sidebar-collapsed .content-wrapper { margin-left:64px; }

.main-sidebar { transition:width .25s ease; }
.content-wrapper { transition:margin-left .25s ease; }

.sidebar-toggle {
  width:100%;
  padding:12px 16px;
  background:none;
  border:none;
  color:#e5e7eb;
  font-size:18px;
  text-align:left;
}

.topbar {
  height:56px;
```

```
  display:flex;
  align-items:center;
  padding-left:16px;
  border-bottom:1px solid rgba(255,255,255,.08);
  background:#020617;
  color:#e5e7eb;
  font-weight:600;
}

.content-wrapper { background:#020617; padding:16px; }

.card {
  background:#020617;
  color:#e5e7eb;
  border:1px solid rgba(255,255,255,.1);
}

.metric-title { font-weight:600; margin-bottom:6px; }

svg { width:100%; height:260px; }
.axis { stroke:#94a3b8; stroke-width:1; }
.bar { fill:#38bdf8; }
.line { fill:none; stroke:#22c55e; stroke-width:2; }
.area { fill:rgba(34,197,94,.35); stroke:#22c55e; stroke-width:2; }
.dot { fill:#facc15; }
.area-dot { fill:#facc15; }

.selector {
  border:1px solid rgba(255,255,255,.25);
  border-radius:6px;
  padding:6px;
  margin-bottom:10px;
}
.selector-header { cursor:pointer; }
.selector-body { max-height:180px; overflow-y:auto; }
label { display:block; font-size:13px; }
</style>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</head>
<!-- ===== ALERT POPUP MODAL ===== -->
<div class="modal fade" id="alertModal" tabindex="-1">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content" style="background: linear-gradient(145deg, #1e1b4b,
#0f0f23); border: 1px solid rgba(255,255,255,0.2); color: white;">
```

```
    <div class="modal-header border-0" style="background: linear-gradient(90deg, #3b82f6, #8b5cf6);">
      <h5 class="modal-title fw-bold">
        <i class="fas fa-bell me-2"></i> 🚨 LIVE ALERT
      </h5>
      <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal"  aria-label="Close"></button>
    </div>
    <div class="modal-body p-4 text-center" id="alertModalBody">
      <div class="alert-icon mb-3">
        <i class="fas fa-chart-line fa-3x" id="alertIcon"></i>
      </div>
      <h4 id="alertTitle" class="fw-bold mb-2"></h4>
      <p id="alertDetails" class="mb-0 opacity-90"></p>
    </div>
  </div>
 </div>
</div>

<body class="hold-transition sidebar-mini layout-fixed">

<div class="wrapper" id="appWrapper">

<aside class="main-sidebar sidebar-dark-primary elevation-4">
 <div class="sidebar">
  <button class="sidebar-toggle" id="toggleBtn">
    <i class="fas fa-bars"></i>
  </button>
  <nav>
    <ul class="nav nav-pills nav-sidebar flex-column">
      <li class="nav-item"><a href="/watchlist" class="nav-link active"><i class="fas fa-list nav-icon"></i><p>Watchlist</p></a></li>
      <li class="nav-item"><a href="/portfolio" class="nav-link"><i class="fas fa-briefcase nav-icon"></i><p>Portfolio</p></a></li>
      <li class="nav-item"><a href="/market" class="nav-link"><i class="fas fa-chart-line nav-icon"></i><p>Market</p></a></li>
      <li class="nav-item"><a href="/chat" class="nav-link"><i class="fas fa-robot nav-icon"></i><p>Chat</p></a></li>
      <li class="nav-item"><a href="/login" class="nav-link"><i class="fas fa-sign-out-alt nav-icon"></i><p>Logout</p></a></li>
    </ul>
  </nav>
 </div>
</aside>

<div class="content-wrapper">
```

```
  <div class="topbar">StockMind AI</div>
  <div id="root"></div>
</div>

</div>

<script>
document.getElementById("toggleBtn").onclick = () => {
  document.getElementById("appWrapper").classList.toggle("sidebar-collapsed");
};
</script>

{% raw %}
<script type="text/babel">
const { useState, useEffect, useRef } = React;

/* ---------- AXES ---------- */
const Axes = ({labels,w,h}) => (
<>
<line x1="45" y1={h-35} x2={w+35} y2={h-35} className="axis"/>
<line x1="45" y1="15" x2="45" y2={h-35} className="axis"/>
{labels.map((l,i)=>(
<text key={i}
x={45 + ((w-70)/(labels.length-1||1)) * (i + 0.5)}
y={h-12}
fontSize="10"
fill="#cbd5f5"
textAnchor="middle">{l}</text>
))}
</>
);

/* ---------- CHARTS (UNCHANGED) ---------- */
const BarChart=({data,labels})=>{
const w=380,h=260,max=Math.max(...data,1),bw=40;
const step=(w-70)/(data.length-1||1);
return(
<svg viewBox={`0 0 ${w} ${h}`}>
<Axes labels={labels} w={w} h={h}/>
{data.map((v,i)=>(
<rect key={i}
x={45+step*(i+0.5)-bw/2}
y={h-35-(v/max)*(h-60)}
width={bw}
height={(v/max)*(h-60)}
className="bar">
```

```
<title>{labels[i]} : {v}</title>
</rect>
))}
</svg>
);
};

const LineChart=({data,labels})=>{
const w=380,h=260,max=Math.max(...data,1);
const step=(w-70)/(data.length-1||1);
const pts=data.map((v,i)=>`${45+step*i},${h-35-(v/max)*(h-60)}`).join(" ");
return(
<svg viewBox={`0 0 ${w} ${h}`}>
<Axes labels={labels} w={w} h={h}/>
<polyline points={pts} className="line"/>
{data.map((v,i)=>(
<circle key={i} cx={45+step*i} cy={h-35-(v/max)*(h-60)} r="4" className="dot">
<title>{labels[i]} : {v}</title>
</circle>
))}
</svg>
);
};

const Scatter=({data,labels})=>{
const w=380,h=260,max=Math.max(...data.map(Math.abs),1);
const step=(w-70)/(data.length-1||1);
return(
<svg viewBox={`0 0 ${w} ${h}`}>
<Axes labels={labels} w={w} h={h}/>
{data.map((v,i)=>(
<circle key={i} cx={45+step*i} cy={h-35-(Math.abs(v)/max)*(h-60)} r="4"
className="dot">
<title>{labels[i]} : {v}</title>
</circle>
))}
</svg>
);
};

const AreaChart=({data,labels})=>{
const w=380,h=260,max=Math.max(...data,1);
const step=(w-70)/(data.length-1||1);
const pts=data.map((v,i)=>`${45+step*i},${h-35-(v/max)*(h-60)}`).join(" ");
return(
<svg viewBox={`0 0 ${w} ${h}`}>
```

```
<Axes labels={labels} w={w} h={h}/>
<polygon points={`45,${h-35} ${pts} ${w-25},${h-35}`} className="area"/>
{data.map((v,i)=>(
<circle key={i} cx={45+step*i} cy={h-35-(v/max)*(h-60)} r="4" className="area-dot">
<title>{labels[i]} : {v}</title>
</circle>
))}
</svg>
);
};

const Doughnut=({data,labels})=>{
const total=data.reduce((a,b)=>a+b,0)||1;
let angle=0;
return(
<>
<svg viewBox="0 0 200 200">
{data.map((v,i)=>{
const a=(v/total)*360;
const r=70;
const x1=100+r*Math.cos(angle*Math.PI/180);
const y1=100+r*Math.sin(angle*Math.PI/180);
angle+=a;
const x2=100+r*Math.cos(angle*Math.PI/180);
const y2=100+r*Math.sin(angle*Math.PI/180);
return(
<path key={i}
d={`M100,100 L${x1},${y1} A${r},${r} 0 ${a>180?1:0} 1 ${x2},${y2} Z`}
fill={`hsl(${i*60},70%,55%)`}>
<title>{labels[i]} : {v}</title>
</path>
);
})}
<circle cx="100" cy="100" r="35" fill="#020617"/>
</svg>
{/* ---- DOUGHNUT LEGEND ---- */}
<div style={{
  display: "flex",
  flexWrap: "wrap",
  justifyContent: "center",
  marginTop: "10px",
  gap: "12px"
}}>
{labels.map((company, i) => (
  <div key={company}
      style={{display:"flex",alignItems:"center",fontSize:"12px"}}>
```

```
    <span style={{
      width:"10px",
      height:"10px",
      borderRadius:"2px",
      backgroundColor:`hsl(${i*60},70%,55%)`,
      display:"inline-block",
      marginRight:"6px"
    }}></span>
    {company}
  </div>
))}
</div>

</>
);
};

/* ---------- SELECTOR ---------- */
const Selector=({companies,selected,setSelected})=>{
  const [open,setOpen]=useState(false);

  const toggle = (e) => {
    e.stopPropagation();
    setOpen(!open);
  };

  const ordered=[...selected,...companies.filter(c=>!selected.includes(c))];

  return(
    <div className="selector" style={{position: 'relative'}}>
      <div
        className="selector-header"
        onClick={toggle}
        style={{padding: '6px 10px', cursor: 'pointer'}}
      >
        Selected {selected.length}/5 ▾
      </div>
      {open && (
        <div
          className="selector-body"
          style={{position: 'absolute', top: '100%', left: 0, right: 0, zIndex: 9999, background:
'#1a1a2e', maxHeight: '400px', overflowY: 'auto'}}
        >
          {ordered.map(c=>(
            <label key={c} style={{display: 'block', padding: '4px 8px', cursor: 'pointer'}}>
              <input
```

```
              type="checkbox"
              checked={selected.includes(c)}
              onChange={(e)=>{
                e.stopPropagation();
                if(selected.includes(c)) setSelected(selected.filter(x=>x!==c));
                else if(selected.length<5) setSelected([c,...selected]);
              }}
            /> {c}
          </label>
        ))}
      </div>
    )}
  </div>
 );
};

/* ---------- APP ---------- */
function App(){
 const { useState, useEffect, useRef } = React;
 const [rows,setRows]=useState([]);
 const [companies,setCompanies]=useState([]);
 const [sel,setSel]=useState({open:[],prev:[],gap:[],pe:[],peg:[],eg:[],rg:[],de:[]});
 const [alerts,setAlerts]=useState([]);
 const [unseen,setUnseen]=useState(false);
 const [openBell,setOpenBell]=useState(false);

 // LOAD DATA
 useEffect(() => {
  fetch('/api/companies').then(r=>r.json()).then(setCompanies);
  fetch('/api/watchlist-data').then(r=>r.json()).then(data => {
   const rowData = data.symbols.map((s,i) => ({
    symbol: s,
    open: data.open_price[i],
    prev: data.previous_close[i],
    pe: data.trailing_pe[i],
    peg: data.peg_ratio[i],
    eg: data.earnings_growth[i],
    rg: data.revenue_growth[i],
    de: data.debt_to_equity[i]
   }));
   setRows(rowData);


   // DEFAULT: Select first 5 companies
   if (data.symbols.length >= 5) {
   const first5 = data.symbols.slice(0, 5);
```

```
    setSel({
      open: first5, prev: first5, gap: first5, pe: first5,
      peg: first5, eg: first5, rg: first5, de: first5
    });
  }
  });
}, []);

// ALERTS (keep existing)
useEffect(() => {
  const es = new EventSource("/api/alerts");
  const seenAlerts = new Set();

  es.onmessage = e => {
    const data = JSON.parse(e.data);
    const alertId = `${data.type}_${data.symbol}_${data.metric || ''}_${Date.now()}`;

    //if (seenAlerts.has(alertId)) return;
    seenAlerts.add(alertId);

    setAlerts(a => [data, ...a.slice(0, 50)]);
    setUnseen(true);

    const modalEl = document.getElementById('alertModal');
    const modalTitle = document.getElementById('alertTitle');
    const modalDetails = document.getElementById('alertDetails');
    const iconEl = document.getElementById('alertIcon');

    if (data.type === 'new_company') {
      modalTitle.textContent = ' 🆕 New Stocks';
      modalDetails.textContent = 'New stocks have been added to your watchlist.';
      iconEl.className = 'fas fa-plus-circle fa-3x text-info';
      modalTitle.style.color = '#3b82f6';
    }
    else if (data.type === 'company_deleted') {
      modalTitle.textContent = ' 🗑️ Stocks Deleted';
      modalDetails.textContent = 'Some stocks have been removed from your watchlist.';
      iconEl.className = 'fas fa-trash-alt fa-3x text-warning';
      modalTitle.style.color = '#f59e0b';
    }
    else {
       modalTitle.textContent = ' 📊 Stock Update';
        modalDetails.textContent = 'There is an update in your watchlist data.';
        iconEl.className = 'fas fa-chart-line fa-3x text-success';
        modalTitle.style.color = '#22c55e';
    }
```

```
      const modal = new bootstrap.Modal(modalEl, { backdrop: true, keyboard: true });
      modal.show();
      setTimeout(() => modal.hide(), 3000);
    };

    return () => es.close();
  }, []);

  const f=k=>rows.filter(r=>sel[k].includes(r.symbol));
  const v=k=>f(k).map(r=>r[k]);
  const l=k=>f(k).map(r=>r.symbol);

  // REST OF RETURN STATEMENT (keep exactly same)

return(
<>
{/* TOP RIGHT BELL */}
<div style={{position:"absolute",top:"12px",right:"20px"}}>
  <div className="notification-bell" onClick={()=>{
    setOpenBell(!openBell);
    setUnseen(false);
  }}>
    <i className="fas fa-bell" style={{fontSize:"18px"}}></i>
    {unseen && <span className="notification-dot"></span>}
  </div>

  {openBell && (
    <div className="notification-panel">
      {alerts.length === 0 && (
        <div className="notification-item">No notifications</div>
      )}

      {alerts.slice(0,6).map((a,i)=>(
        <div key={i} className="notification-item">
          {a.type === "new_company" && (
            <> 🆕 <b>{a.symbol}</b> added</>
          )}
          {a.type === "company_deleted" && (
          <> 🗑️ <b>{a.symbol}</b> deleted</>
          )}

          {a.type === "metric_update" && (
            <>
              <b>{a.symbol}</b> {a.metric}<br/>
              <span className={a.new > a.old ? "green" : "red"}>
```

```
            {a.old} → {a.new}
          </span>
        </>
      )}
    </div>
  ))}
 </div>
)}
</div>

<div className="row">
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Open Price</div><Selector companies={companies} selected={sel.open}
setSelected={v=>setSel({...sel,open:v})}/><LineChart data={v("open")}
labels={l("open")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Previous Close</div><Selector companies={companies} selected={sel.prev}
setSelected={v=>setSel({...sel,prev:v})}/><BarChart data={v("prev")}
labels={l("prev")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Gap</div><Selector companies={companies} selected={sel.gap}
setSelected={v=>setSel({...sel,gap:v})}/><Scatter data={f("gap").map(r=>r.open-r.prev)}
labels={l("gap")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Trailing PE</div><Selector companies={companies} selected={sel.pe}
setSelected={v=>setSel({...sel,pe:v})}/><BarChart data={v("pe")}
labels={l("pe")}/></div></div>
<div className="col-md-12"><div className="card p-3"><div className="metric-
title">PEG Ratio</div><Selector companies={companies} selected={sel.peg}
setSelected={v=>setSel({...sel,peg:v})}/><Doughnut data={v("peg")}
labels={l("peg")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Earnings Growth</div><Selector companies={companies} selected={sel.eg}
setSelected={v=>setSel({...sel,eg:v})}/><AreaChart data={v("eg")}
labels={l("eg")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Revenue Growth</div><Selector companies={companies} selected={sel.rg}
setSelected={v=>setSel({...sel,rg:v})}/><LineChart data={v("rg")}
labels={l("rg")}/></div></div>
<div className="col-md-6"><div className="card p-3"><div className="metric-
title">Debt to Equity</div><Selector companies={companies} selected={sel.de}
setSelected={v=>setSel({...sel,de:v})}/><BarChart data={v("de")}
labels={l("de")}/></div></div>
</div>
</>
);
```

```
}

ReactDOM.createRoot(document.getElementById("root")).render(<App/>);
</script>
{% endraw %}

</body>
</html>
```

market.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title> Live Market Dashboard</title>
   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
   <style>
      * { box-sizing: border-box; }
      body {
         font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
         background: linear-gradient(135deg, #000000 0%, #1a1a2e 100%);
         color: #fff;
         margin: 0;
         padding: 20px;
         min-height: 100vh;
      }
      .container { max-width: 1600px; margin: 0 auto; }
      .header {
         text-align: center;
         margin-bottom: 40px;
         padding: 20px;
         background: rgba(255,255,255,0.05);
         border-radius: 20px;
         backdrop-filter: blur(20px);
      }
      .header h1 {
         font-size: 2.8em;
         margin: 0 0 10px 0;
         background: linear-gradient(90deg, #00ff88, #00ccff, #ff6b6b);
         background-clip: text;
         -webkit-background-clip: text;
         -webkit-text-fill-color: transparent;
      }
      .stocks-grid {
```

```
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(400px, 1fr));
    gap: 25px;
}
.stock-card {
    background: rgba(20,20,30,0.95);
    border-radius: 24px;
    padding: 30px;
    box-shadow: 0 25px 50px rgba(0,0,0,0.7);
    border: 1px solid rgba(255,255,255,0.08);
    position: relative;
    overflow: hidden;
    transition: all 0.4s ease;
    height: 450px;
}
.stock-card:hover {
    transform: translateY(-8px);
    box-shadow: 0 35px 70px rgba(0,255,136,0.3);
}
.stock-card::before {
    content: '';
    position: absolute;
    top: 0; left: 0; right: 0;
    height: 5px;
    background: linear-gradient(90deg, var(--card-color), transparent);
}
.stock-header {
    display: flex;
    justify-content: space-between;
    align-items: flex-start;
    margin-bottom: 25px;
    position: relative;
    z-index: 10;
}
.stock-info h2 {
    margin: 0 0 15px 0;
    font-size: 22px;
    font-weight: 800;
    text-shadow: 0 2px 10px rgba(0,0,0,0.5);
}
.stock-price {
    font-size: 40px;
    font-weight: 900;
    margin: 10px 0;
    line-height: 1;
    text-shadow: 0 4px 20px rgba(0,255,136,0.3);
```

```
}
.change {
    font-size: 18px;
    font-weight: 700;
    margin: 0;
    padding: 8px 16px;
    border-radius: 20px;
    display: inline-block;
}
.positive {
    color: #00ff88;
    background: rgba(0,255,136,0.2);
    box-shadow: 0 0 20px rgba(0,255,136,0.4);
}
.negative {
    color: #ff4444;
    background: rgba(255,68,68,0.2);
    box-shadow: 0 0 20px rgba(255,68,68,0.4);
}
.time-range {
    display: flex;
    gap: 8px;
}
.time-btn {
    padding: 10px 16px;
    border: none;
    border-radius: 25px;
    background: rgba(80,80,100,0.6);
    color: #fff;
    cursor: pointer;
    font-size: 13px;
    font-weight: 600;
    transition: all 0.3s;
    border: 1px solid rgba(255,255,255,0.2);
    min-width: 45px;
}
.time-btn:hover, .time-btn.active {
    background: linear-gradient(135deg, #00ff88, #00ccff);
    box-shadow: 0 8px 25px rgba(0,255,136,0.5);
    transform: scale(1.05);
    border-color: #00ff88;
}
.live-status {
    position: absolute;
    top: 15px;
    right: 20px;
```

```
    font-size: 12px;
    color: #00ff88;
    font-weight: 600;
    z-index: 10;
}
.live-dot {
    display: inline-block;
    width: 10px;
    height: 10px;
    background: #00ff88;
    border-radius: 50%;
    margin-right: 6px;
    animation: pulse 1.5s infinite;
}
@keyframes pulse {
    0%, 100% { opacity: 1; transform: scale(1); }
    50% { opacity: 0.4; transform: scale(1.2); }
}
/* **PERFECTLY FITTED CHART CONTAINER** */
.chart-wrapper {
    position: relative;
    width: 100%;
    height: 280px;  /* FIXED HEIGHT */
    margin-top: 20px;
    border-radius: 20px;
    overflow: hidden;
    background: linear-gradient(145deg, rgba(0,0,0,0.8), rgba(40,40,60,0.6));
    border: 1px solid rgba(255,255,255,0.1);
    box-shadow: inset 0 2px 10px rgba(0,0,0,0.5);
}
.chart-canvas-wrapper {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    padding: 20px;  /* INNER PADDING FOR AXIS */
    background: transparent;
}
canvas {
    width: 100% !important;
    height: 100% !important;
    display: block;
}
@media (max-width: 768px) {
    .stocks-grid { grid-template-columns: 1fr; }
```

```
            .stock-price { font-size: 32px; }
            .chart-wrapper { height: 250px; }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>Live Market Dashboard</h1>
            <p>Live updates every 10s</p>
            <div><a style = "color:white" href="/watchlist">Go back to Watchlist</a></div>
        </div>

        <div class="stocks-grid" id="stocks-grid"></div>
    </div>

    <script>
        const STOCKS = [
            { symbol: 'MSFT', name: 'Microsoft Corporation', color: '#00ccff', basePrice: 420 },
            { symbol: 'AAPL', name: 'Apple Inc', color: '#ff6b6b', basePrice: 230 },
            { symbol: 'NKE', name: 'Nike, Inc', color: '#00ff88', basePrice: 85 },
            { symbol: 'GOOGL', name: 'Alphabet Inc (Google)', color: '#ffd700', basePrice: 180
},
            { symbol: 'AMZN', name: 'Amazon.com, Inc', color: '#ff9900', basePrice: 195 },
            { symbol: 'TSLA', name: 'Tesla, Inc', color: '#e31b23', basePrice: 250 },
            { symbol: 'META', name: 'Meta Platforms, Inc', color: '#4267B2', basePrice: 580 },
            { symbol: 'NVDA', name: 'NVIDIA Corporation', color: '#76b900', basePrice: 135 },
            { symbol: 'JPM', name: 'JPMorgan Chase & Co', color: '#0e3c90', basePrice: 210 },
            { symbol: 'V', name: 'Visa Inc', color: '#1a1f71', basePrice: 290 },
            { symbol: 'JNJ', name: 'Johnson & Johnson', color: '#0072bc', basePrice: 165 },
            { symbol: 'HD', name: "The Home Depot, Inc", color: '#febd17', basePrice: 380 }
        ];

        const TIME_RANGES = { '1w': 7, '1m': 30, '3m': 90, '1y': 365 };
        let charts = {}, currentRanges = {}, stockData = {};

        function getLivePrice(basePrice) {
            const volatility = 0.02;
            const now = Date.now();
            const minutes = (now % 86400000) / 60000;
            const trend = Math.sin(minutes / 1440 * Math.PI * 2) * 0.015;
            const noise = (Math.random() - 0.5) * volatility;
            return Math.max(1, basePrice * (1 + trend + noise));
        }

        function generateChartData(symbol, days) {
```

```
        const data = [];
        const endDate = new Date();
        for (let i = days - 1; i >= 0; i--) {
            const date = new Date(endDate);
            date.setDate(date.getDate() - i);
            const basePrice = STOCKS.find(s => s.symbol === symbol).basePrice;
            const volatility = 0.015;
            const trend = i < days * 0.3 ? -0.001 : 0.002;
            const price = basePrice * (1 + trend * i + (Math.random() - 0.5) * volatility *
Math.sqrt(days));

            data.push({
                date: date.toISOString().split('T')[0],
                close: Math.max(1, price + (Math.random() - 0.5) * 2)
            });
        }
        return data.reverse();
    }

    function createChart(ctx, data, color, range) {
        const canvasId = ctx.canvas.id;
        if (charts[canvasId]) charts[canvasId].destroy();

        const labels = data.slice(-25).map(d => {
            const date = new Date(d.date);
            if (range === '1w') return date.toLocaleString('en-US', {
                weekday: 'short', hour: 'numeric'
            });
            return date.toLocaleDateString('en-US', { month: 'short', day: 'numeric' });
        });

        charts[canvasId] = new Chart(ctx, {
            type: 'line',
            data: {
                labels,
                datasets: [{
                    data: data.slice(-25).map(d => d.close),
                    borderColor: color,
                    backgroundColor: color + '20',
                    tension: 0.4,
                    fill: true,
                    pointRadius: 0,
                    borderWidth: 3,
                    pointHoverRadius: 8
                }]
            },
```

```
options: {
    responsive: true,
    maintainAspectRatio: true,  /* CRITICAL FOR BORDER FIT */
    layout: {
        padding: {
            left: 10,
            right: 10,
            top: 10,
            bottom: 10
        }
    },
    plugins: {
        legend: { display: false },
        tooltip: {
            backgroundColor: 'rgba(15,15,25,0.95)',
            titleColor: '#fff',
            bodyColor: '#fff',
            cornerRadius: 12,
            padding: 12
        }
    },
    scales: {
        x: {
            grid: {
                color: 'rgba(255,255,255,0.08)',
                drawBorder: false,
                lineWidth: 1
            },
            ticks: {
                color: '#999',
                maxRotation: 45,
                font: { size: 11, family: '-apple-system' },
                padding: 8,
                maxTicksLimit: 8
            },
            border: {
                display: false
            }
        },
        y: {
            grid: {
                color: 'rgba(255,255,255,0.05)',
                drawBorder: false,
                lineWidth: 1
            },
            ticks: {
```

```
                    color: '#999',
                    font: { size: 11, family: '-apple-system' },
                    padding: 8,
                    callback: function(value) {
                        return '$' + value.toLocaleString();
                    }
                },
                border: {
                    display: false
                }
            }
        },
        interaction: {
            intersect: false,
            mode: 'index'
        },
        elements: {
            point: { radius: 0 }
        }
    }
});
}

function updateStockCard(stock) {
    const symbol = stock.symbol;
    const card = document.getElementById(`card-${symbol}`);
    if (!card) return;

    const priceEl = card.querySelector('.stock-price');
    const changeEl = card.querySelector('.change');
    const chartCanvas = card.querySelector('canvas');
    const ctx = chartCanvas.getContext('2d');
    const range = currentRanges[symbol] || '1w';

    const currentPrice = getLivePrice(stock.basePrice);
    const prevPrice = stockData[symbol]?.price || currentPrice;
    const changePercent = ((currentPrice - prevPrice) / prevPrice * 100);

    stockData[symbol] = { price: currentPrice, change: changePercent };

    const changeClass = changePercent >= 0 ? 'positive' : 'negative';
    priceEl.textContent = `$${currentPrice.toFixed(2)}`;
    priceEl.className = `stock-price ${changeClass}`;
    changeEl.textContent = `${changePercent >= 0 ? '+' :
"}${changePercent.toFixed(2)}%`;
    changeEl.className = `change ${changeClass}`;
```

```
    const chartData = generateChartData(symbol, TIME_RANGES[range]);
    createChart(ctx, chartData, stock.color, range);
}

function createStockCard(stock) {
    const div = document.createElement('div');
    div.id = `card-${stock.symbol}`;
    div.className = 'stock-card';
    div.style.setProperty('--card-color', stock.color);
    currentRanges[stock.symbol] = '1w';

    div.innerHTML = `
        <div class="live-status">
            <span class="live-dot"></span>Live
        </div>
        <div class="stock-header">
            <div class="stock-info">
                <h2>${stock.name}</h2>
                <p class="stock-price">$0.00</p>
                <p class="change">0.00%</p>
            </div>
            <div class="time-range">
                <button class="time-btn active" data-range="1w">1W</button>
                <button class="time-btn" data-range="1m">1M</button>
                <button class="time-btn" data-range="3m">3M</button>
                <button class="time-btn" data-range="1y">1Y</button>
            </div>
        </div>
        <div class="chart-wrapper">
            <div class="chart-canvas-wrapper">
                <canvas id="chart-${stock.symbol}"></canvas>
            </div>
        </div>
    `;

    div.querySelectorAll('.time-btn').forEach(btn => {
        btn.addEventListener('click', () => {
            div.querySelectorAll('.time-btn').forEach(b => b.classList.remove('active'));
            btn.classList.add('active');
            currentRanges[stock.symbol] = btn.dataset.range;
            updateStockCard(stock);
        });
    });

    return div;
```

```
    }

    function init() {
      const grid = document.getElementById('stocks-grid');
      STOCKS.forEach(stock => grid.appendChild(createStockCard(stock)));

      STOCKS.forEach((stock, i) => {
        setTimeout(() => updateStockCard(stock), i * 100);
      });

      setInterval(() => {
        STOCKS.forEach(stock => updateStockCard(stock));
      }, 10000);
    }

    init();
  </script>
</body>
</html>
```

chat.html

```
<!DOCTYPE html>
<html>
<head>
<title>StockMind AI Chat</title>
<style>
  #sendBtn {
  padding: 14px 26px;
  border-radius: 12px;
  border: none;
  background: #2563eb;
  color: white;
  font-size: 14px;
  cursor: pointer;
}
#sendBtn:hover {
  background: #1d4ed8;
}
body {
  margin: 0;
  height: 100vh;
  background: #f8fafc;
  font-family: "Inter", Arial, sans-serif;
  display: flex;
  flex-direction: column;
```

```
}

.topbar {
    height: 56px;
    background: #ffffff;
    border-bottom: 1px solid #e5e7eb;
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 0 24px;
}

.chat-wrapper {
    flex: 1;
    display: flex;
    justify-content: center;
    align-items: center;
}

.chat-shell {
    width: 90%;
    max-width: 1100px;
    height: 80vh;
    background: #ffffff;
    border-radius: 16px;
    box-shadow: 0 20px 40px rgba(0,0,0,0.08);
    border: 2px solid #e5e7eb;
    display: flex;
    flex-direction: column;
}

.chat-body {
    flex: 1;
    padding: 24px;
    overflow-y: auto;
    background: #f9fafb;
    border-radius: 14px 14px 0 0;
}

.chat-footer {
    padding: 16px;
    border-top: 1px solid #e5e7eb;
    border-bottom: 2px solid #e5e7eb;
    background: white;
    border-radius: 0 0 14px 14px;
}
```

```
#q {
  width: 85%;
  padding: 14px;
  border-radius: 10px;
  border: 2px solid #d1d5db;
  outline: none;
  font-size: 14px;
  transition: border-color 0.2s;
}

#q:focus {
  border-color: #2563eb;
}

.empty {
  text-align: center;
  color: #64748b;
  margin-top: 120px;
}

.message {
  max-width: 70%;
  padding: 14px 18px;
  margin-bottom: 14px;
  border-radius: 14px;
  font-size: 14px;
}

.user {
  margin-left: auto;
  background: #2563eb;
  color: white;
}

.bot {
  background: white;
  border: 1px solid #e5e7eb;
  box-shadow: 0 2px 8px rgba(0,0,0,0.06);
}

.loading {
  font-size: 13px;
  color: #64748b;
  margin-bottom: 10px;
}
```

```
</style>
</head>

<body>

<div class="topbar">
   <strong>StockMind AI Assistant</strong>
   <div>
      <a href="/watchlist">Watchlist</a>
      <a href="/login">Logout</a>
   </div>
</div>

<div class="chat-wrapper">
<div class="chat-shell">

<div id="messages" class="chat-body">
<div id="emptyState" class="empty">
<h3>Ask me about market trends</h3>
<p>Fundamentals • Risk • Valuations • Comparisons</p>
</div>
</div>

<div class="chat-footer" style="padding:16px;border-top:1px solid #e5e7eb;">
<input id="q" style="width:85%;padding:14px;border-radius:10px;border:1px solid
#e5e7eb;">
<button onclick="send()" id="sendBtn">Send</button>

</div>

</div>
</div>

<script>

const messages = document.getElementById("messages");
const input = document.getElementById("q");

function addMessage(text, type) {
   document.getElementById("emptyState")?.remove();
   const div = document.createElement("div");
   div.className = `message ${type}`;
   div.innerText = text;
   messages.appendChild(div);
   messages.scrollTop = messages.scrollHeight;
}
```

```
async function send() {
    const q = input.value.trim();
    if (!q) return;

    addMessage(q, "user");
    input.value = "";

    const loading = document.createElement("div");
    loading.className = "loading";
    loading.innerText = "Thinking…";
    messages.appendChild(loading);

    const res = await fetch("/ask", {
        method: "POST",
        headers: {"Content-Type": "application/json"},
        body: JSON.stringify({question:q})
    });

    loading.remove();
    const data = await res.json();
    addMessage(data.answer, "bot");

}
input.addEventListener("keydown", function (e) {
    if (e.key === "Enter") {
        e.preventDefault();
        send();
    }
});
</script>
</body>
</html>
```