Plutos Equities

Final Design Report

Your Gateway to NASDAQ: Intelligent Capital Forecasting

Presented by:

Emre Akgül, Uygar Aras, Asım Adil Can, Gün Taştan, Alara Zeybek

02.05.2025

1. Introduction

Plutos Equities is a financial and capital forecasting platform designed to predict the quarterly financial performance of the top NASDAQ-listed companies [1]. The primary objective of this project is to support financial stakeholders, including investors, auditors, and analysts, with reliable predictions of key financial metrics such as revenue, net income, and operating expenses. By delivering forecasts before quarterly reports are officially released, Plutos Equities provides users with a tool to make data-driven decisions, assess risks, and identify opportunities.

2. Requirements Details

Currently, there are no automated solutions specifically designed for capital forecasting. Existing systems mostly focus on stock price prediction, with platforms like Bloomberg Terminal, AlphaSense, and TradingView using historical data, technical indicators, and sentiment analysis to forecast market trends and price movements. However, these tools do not address the prediction of capital metrics such as revenue, net income, and operating expenses. On the other hand, capital forecasting remains a largely manual process conducted by financial analysts and auditors [2]. These professionals rely on tools like Excel to analyze financial statements, economic indicators, and market trends, which, while effective for detailed analysis, are time-intensive and lack the scalability that an automated solution could provide. As a result, there is no direct alternative to an automated capital forecasting system like Plutos Equities.

3. Final Architecture and Design Details

3.1. Overivew

Plutos Equities focuses on providing a comprehensive financial forecasting solution by predicting the quarterly reports of the top 100 NASDAQ-listed companies [1]. The project aims to offer accurate, data-driven forecasts of critical financial metrics for the next quarter. These forecasts include revenue, expenses, operating income, and other key metrics that are important for understanding a company's financial position. The platform combines structured data, such as historical financial statements and market indices, with unstructured data from news articles, earnings call transcripts, and social media sentiment. Machine learning models are used to process these data sources, uncovering patterns and trends that traditional methods often miss [2]. The result is a set of predictions that are not only accurate but also insightful for stakeholders looking to make informed decisions. To give an example, the platform can forecast Apple Inc.'s Q1 2025 performance by analyzing its financial data from previous quarters, combined with real-time updates during the three months leading up to the new quarter. These updates might include news of a new product launch, changes in production costs, or shifts in global economic factors that impact the tech industry. By using both historical and recent data, the platform provides a clear prediction of Apple's revenue, expenses, and profitability, helping stakeholders anticipate the company's performance ahead of official earnings reports and quarter filings [3]. Unlike traditional stock price forecasting tools, which often fail to provide a nuanced understanding of a company's underlying financial health, Plutos Equities focuses on the core metrics that define corporate performance. The platform achieves this by combining data collection methods, feature engineering, and machine learning models. The integration of real-time data sources, such as financial news and earnings call transcripts, further enhances the predictions and allows for a dynamic and adaptive forecasting system. In addition to its forecasting capabilities, Plutos Equities features a user-friendly interface that allows users to explore predictions through customizable dashboards and visualizations. This ensures that both institutional investors and individual users can easily access and interpret the data, regardless of their level of financial expertise.By focusing on financial metrics rather than stock prices, Plutos Equities provides a more detailed view of company performance and health, making it a valuable resource for anyone involved in the financial analysis of NASDAQ companies. Our project is going to be within the scope of the product system, which is under the experience section. The innovation we are bringing to the market is focused on transparency and accessibility in financial forecasting. Unlike existing tools that offer broad financial data aggregation or limited stock price predictions, Plutos Equities is uniquely positioned as the first platform to provide transparent quarterly financial forecasting for NASDAQ companies, openly sharing its data sources. By detailing the sources of our predictions—such as financial statements, market indices, real-time news, and social sentiment—our project builds trust and credibility, offering investors, auditors, and other stakeholders a reliable and explainable solution. This approach addresses a gap in the market, as most existing tools operate as "black boxes," providing users with a limited understanding of how their forecasts are generated. By delivering a platform that clearly communicates its data sources, Plutos Equities provides a straightforward and reliable financial forecasting solution [4]. This transparency allows users to better understand the predictions and use them effectively in their decision-making processes. The platform aims to change how financial forecasting is approached, making it practical and accessible to a broad range of users, including investors, auditors, and corporate decision-makers. Despite its innovative approach, a key challenge lies in integrating and updating diverse data sources in real-time [5]. As financial data is inherently dynamic, incorporating unstructured inputs like earnings call transcripts or social sentiment into our predictive models without compromising accuracy is a complex task.

3.2. Functional Requirements

3.2.1. Quarterly Financial Prediction Functionality

- Generate predictions for key financial metrics (Revenue, EPS, Gross Margin) for NASDAQ Top 100 companies.
- Review confidence intervals for each prediction.
- Forecast financial performance for the upcoming quarter.
- Potentially extend predictions up to 3 quarters in the future [6].

3.2.2. User Functionalities

- Support user account creation/authentication
    - Users can create an account.

        - Login/logout functionalities.
        - Change membership plan.
        - Change the payment preferences.
- Follow companies you are interested in.
- Unfollow companies.
- Allow personalization of the dashboard and metrics.
    - Customize/edit financial charts according to ease of use.
- Provide insights on prediction interpretation.
- Implement feedback mechanisms for continuous improvement.
- Visualization of simple parameters about companies and their stocks.
- Visualization of the predicted quarter report about companies.
- Allow users to export overall forecast results in different formats (.csv, .xls, etc.)

3.2.3. Admin Functionalities

- Provide a centralized admin dashboard for system monitoring, configuration, and management.
- Comprehensive documentation, including system architecture, user manuals, and troubleshooting guides.

3.3. Nonfunctional Requirements

3.3.1. Performance

The platform will process a wide range of financial data, including historical financial statements, market volatility indicators, economic trends, industry-specific data, regulatory factors, and news sentiment. Therefore, the system must implement efficient database indexing and query optimization to maintain response times. The maximum system load capacity should handle at least 200 simultaneous user sessions and reduce load times for frequently accessed data.

Furthermore, to comply with the business performance standard, the platform will achieve a Mean Absolute Percentage Error (MAPE) of less than 15% for forecasts in its first-to-next financial report. It will also ensure accurate predictions based on its own threshold (examined under Success Metrics in the report) for at least 80% of the Top NASDAQ companies.

3.3.2. Reliability

Plutos Equities uses consistent analytics algorithms across 100 companies and relies on similar methods to obtain up-to-date data, enabling the error handling system to be automated and effectively manage and recover from unforeseen failures without the need for manual intervention. Additionally, the system will maintain at least %99 uptime to ensure availability during critical financial periods, such as quarterly report releases. Servers also save processed data, predictions, and user configurations in a secure AWS-hosted database.

3.3.3. Usability

One of the highlights of Plutos Equities' current financial applications is that it offers an open and accessible service that anyone can use, regardless of their level of knowledge. Accordingly, the system will have an easy-to-use interface that requires new users to receive training for no more than 20 minutes. It will support responsive design to ensure optimal viewing and interaction experience across desktop, tablet, and mobile devices. Interactive charts and graphs (using D3.js or Chart.js) will help users visualize trends, predictions, and anomalies in real time. It will have customizable views based on user preferences and sector focus. Any page the user requests will be available within 2 seconds.

3.3.3.1. Scalability

Plutos Equities designed the system architecture to scale horizontally by adding more servers or cloud instances without significant reconfiguration. The system will be tested for scenarios that increase traffic by up to 10 times. With

AWS, it supports automatic load balancing to distribute user requests efficiently across available resources. Its design is cloud-native, and it has a containerized architecture to facilitate easy scaling and deployment.

3.3.4. Maintainability

Plutos's system is built with maintainability in mind, making it simple to upgrade, change, or expand with little risk or work. To make debugging and future development easier, code follows established naming standards, modular design principles, and thorough documentation methods. To reduce technological debt, dependencies, and configurations must be properly described and maintained. The system's integrated logging, monitoring, and error-handling features will enable effective problem diagnosis and resolution. To minimize downtime and manual intervention, maintenance procedures, including upgrades, scaling, and deployment, must be automated whenever feasible. In order to guarantee smooth updates, the system must also be backward-compatible with earlier iterations and contain test suites to ensure that modifications do not interfere with already-existing functionality.

3.3.5. Security

Plutos Equities uses AWS servers to enforce secure authentication using AWS Identity and Access Management (IAM). Servers adhere to data privacy regulations (e.g., GDPR, CCPA) with encrypted data storage and secure API endpoints.

3.3.6. Sustainability

Plutos Equities recognizes the importance of environmental sustainability and aims to reduce its ecological footprint by at least 10% compared to the average website. It utilizes cloud providers with strong commitments to renewable energy (AWS). Optimize our algorithms for energy efficiency. Additionally, it delivers all reports and analyses digitally. Encourage users to adopt paperless practices.

3.4. Subsystem Decomposition Overview

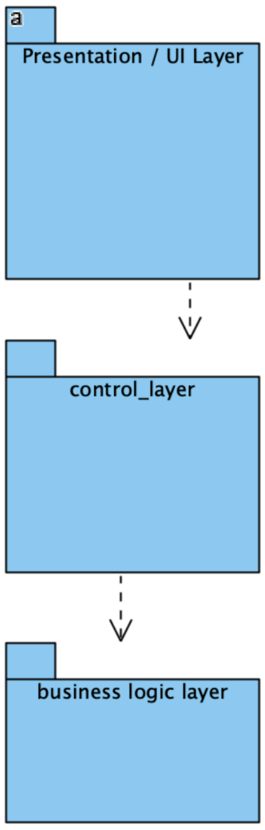


Figure 1: High Level Subsystem Decomposition

Initially, we provide a clear explanation of how the system is organized into distinct layers: Presentation/UI Layer, Control Layer, and Business Logic Layer, illustrated through diagrams and class structures. Then, the hardware/software mapping of the platform is presented, clarifying how the application's components are allocated across various hardware resources and cloud-based infrastructure. The persistent data management segment follows, describing how data storage, retrieval, and object management are handled within Azure SQL and Azure Storage services.

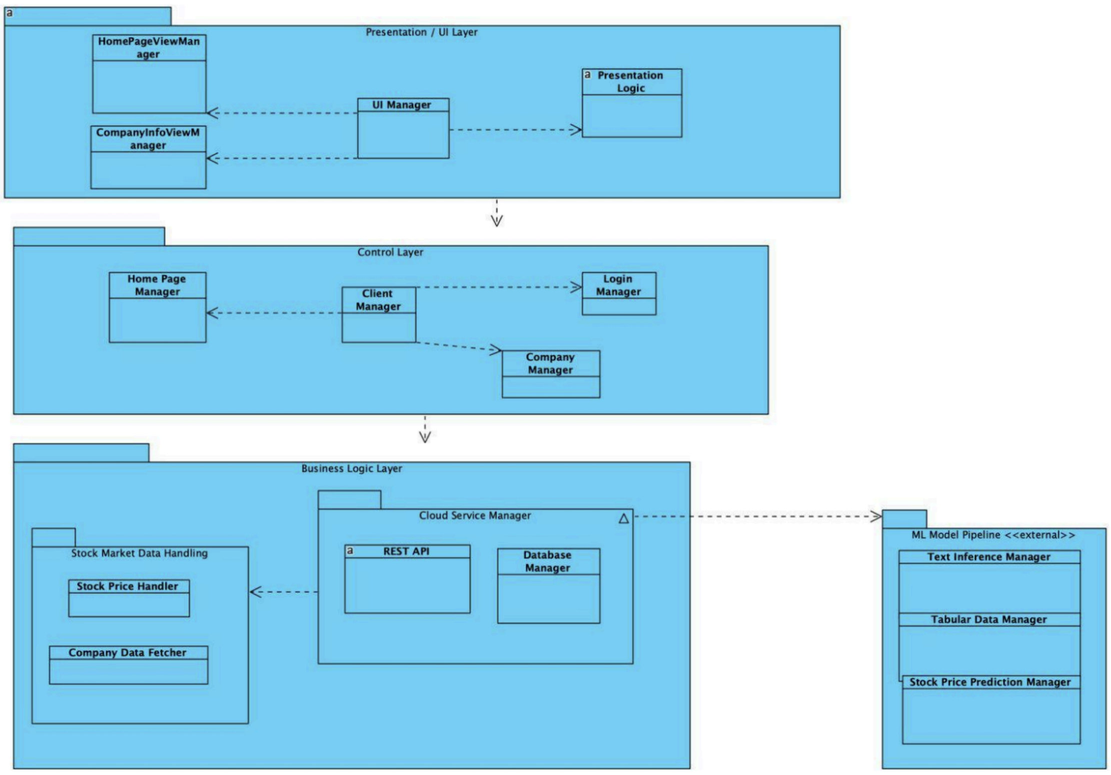


Figure 2: Detailed Subsystem Decomposition

3.4.1. Presentation/UI Layer:

This layer is responsible for the interaction with the end-user and acts as the entry point to the platform. It manages how users access the forecasting services, visualizes forecasts, and presents insights intuitively. All user requests are received and processed initially at this layer. Specifically, the HomePageViewManager handles the overall layout and initial interactions users have with the application's interface, whereas the CompanyInfoViewManager specializes in managing views and interactions related to specific company data. The UI Manager coordinates these views to ensure consistent UI behavior and user experience. Additionally, the Presentation Logic component supports rendering logic, facilitating a responsive and seamless user experience.

3.4.2. Control Layer:

This layer orchestrates requests between user interfaces and business logic. This layer incorporates components like the HomePageManager, Client Manager, Login Manager, and Company Manager, providing session management, authentication, and user-specific interactions. The Client Manager is a central component, delegating authentication tasks to the Login Manager and handling company-related requests with the Company Manager.

3.4.3. Business Logic Layer:

Implements the main functionalities and hosts the REST API. Within this layer, the Cloud Service Manager integrates external cloud-based resources such as storage services and database operations through a RESTful API. This subsystem ensures smooth communication between data repositories, databases, and the machine learning infrastructure. It specifically manages the flow of requests to the ML Model Pipeline, an external component consisting of the Text Inference Manager, Tabular Data Manager, and Stock Price Prediction Manager. The ML Model Pipeline subsystem leverages advanced natural language processing and time-series analysis models to deliver precise forecasting data to the business logic layer. After processing and consolidating the results, the Cloud Service Manager sends data back through RESTful APIs to the UI for visualization and user interaction. Finally, all user, analytical, and operational data is persisted using the Persistence Layer, consisting of robust database solutions and object storage, ensuring reliable data storage and retrieval capabilities

3.5. Hardware/Software Mapping

Plutos Equities is a medium-scale, web-based financial forecasting platform that leverages advanced machine learning and sentiment analysis techniques to generate predictions. While the application doesn't demand extensive computational power, it does rely on highly efficient and dependable cloud infrastructure for data storage, processing, and deployment of predictive models. To ensure optimal performance, scalability, and data integrity, we leverage Azure cloud services, which are tailored to meet the unique needs of our application.

3.5.1. Azure SQL Database

We use Azure SQL Database for managing structured data, specifically to store historical financial data that feeds into our time-series forecasting models. Azure SQL provides robust performance and reliability, with geo-replication ensuring disaster recovery, high availability, and security—features critical for the sensitive nature of financial data. Our setup uses a Single Database configuration on the vCore General Purpose tier (Gen 5), which includes Provisioned capacity. This offers geo-replication for disaster recovery and locally redundant storage. This setup provides scalability for handling the expected volume of financial data and ensures performance consistency.

3.5.2. Service Details:

Single Database, vCore, General Purpose, Provisioned, Standard-series (Gen 5), Primary or Geo-replica Disaster Recovery, Locally Redundant, 1-2 vCores, 200 Hours, 32 GB Storage, SQL License (Pay-as-you-go), LRS Backup Storage, Point-In-Time Restore, Long-Term Retention Options.

Estimated Monthly Price: $105.66

3.5.3. Azure Storage Accounts

For unstructured and semi-structured data, such as financial documents, earnings reports, and textual data for natural language processing (NLP), Azure Storage Accounts handle the storage needs. We utilize Block Blob Storage within the Hot Access tier to ensure low-latency and high-throughput access, which is essential for processing large volumes of text-based data efficiently.

3.5.4. Service Details:

Block Blob Storage, General Purpose V2, Flat Namespace, LRS Redundancy, Hot Access Tier, 1,000 GB Capacity (Pay-as-you-go), Supports high-frequency operations such as 10,000 Write and List operations, 1,000 GB Data Retrieval & Write
Estimated Monthly Price: $240.84

3.5.5. Azure Machine Learning

For model deployment and management, including the use of transformer-based models like recurrent neural networks (LSTM) [7], we leverage Azure Machine Learning. This service allows for seamless model training, evaluation, and deployment, providing a powerful infrastructure optimized for machine learning workloads. Azure's flexibility allows us to scale computational resources as necessary, optimizing both performance and costs.

3.5.6. Service Details:

Azure Machine Learning infrastructure is optimized for model training and deployment. Ensures efficient use of resources during model retraining activities.
Estimated Monthly Price: $82.49

3.5.7. Total Infrastructure Cost

The combined monthly cost for Pluto's Equities' Azure-based infrastructure totals approximately $428. This setup comfortably supports our medium-scale user base, expected to handle several hundred concurrent users efficiently. Azure SQL Database and Azure Storage Accounts are continuously active, ensuring data availability, while Azure Machine Learning is primarily engaged during the model training and periodic retraining phases, optimizing our resource usage and cost-effectiveness. Azure's flexible architecture allows us to scale up or down easily as our needs grow, making it highly adaptable for future expansions. For users, the system is accessible via any modern web browser capable of supporting HTML5, CSS3, and ES6, allowing for seamless interaction with the platform from any device with an internet connection.

3.6. Subsystem Services

To keep Pluto's Equities maintainable and scalable, the platform is divided into several self-contained subsystems. Each subsystem hides its internal workings behind a small, well-defined set of services—essentially the public "contract" that other parts of the application rely on. Because those contracts are clear and versioned, any subsystem can evolve internally without forcing changes everywhere else. The Presentation / UI layer offers services whose sole purpose is to translate JSON coming from the backend into rich, interactive views. Functions such as *renderDashboard* or *showCompanyOverview* build charts, tables, and forms, while *notifyUser* raises browser-side toasts or WebSocket updates. The layer never reaches into business logic directly; instead, it forwards user gestures to the Control layer and waits for data to stream back. Sitting immediately beneath the UI, the Control layer orchestrates multi-step flows. Services like *login*, *followTicker*, or *startCheckout* validate user intent, manage session state, and then delegate to deeper subsystems. For example, *followTicker* checks authentication with the Auth service, persists the new watch-list entry, and finally triggers a dashboard refresh. In the case of subscription changes, the same layer enforces the rule that a user must prune excess favourite companies before downgrading, guaranteeing subscription integrity no matter which client device initiates the request. All domain rules live in the Business Logic layer. Its services—*getCurrentPrices*, *getForecast*, *generateReport*, *comparePredictionVsActual*, and similar—pull data from storage, call into the Machine-Learning pipeline, compute aggregates, and return ready-to-render payloads. These endpoints also record audit logs so that administrative tooling can reconstruct any user operation. Behind the scenes, the Persistence layer provides CRUD operations for Azure SQL and blob storage. Typical services include *savePrediction*, *fetchDailyPrices*, or *storeUserToken*. Transactions, indexing, and backup policies are enforced here, so higher layers never need to think about raw SQL or storage APIs. Whenever a forecast is required, the Business Logic layer calls the Machine-Learning pipeline, whose services—*runPriceModel*, *runQuarterlyMetricsModel*, and *retrainModels*—load artefacts from Azure ML, execute inference, and attach confidence bands. Periodically the same pipeline retrains models in the background and saves new weights back to the prediction schema. Real-time user engagement is handled by the Notification service. Its interface allows any subsystem to *pushAlert*, *subscribeThreshold*, or *broadcastSystemStatus*. Messages are deduplicated and delivered via WebSockets or email, always respecting the user's notification settings. Financial transactions flow through the Payment service. It wraps Stripe's APIs with methods such as *createCheckoutSession*, *handleStripeWebhook*, and *changeSubscription*. Webhook signatures are verified here; successful events update the subscription ledger and mint new JWT scopes so that protected endpoints remain locked behind the correct paywall. Authentication is centralised in the Auth service. Using *issueToken*, *refreshToken*, and *validateScope*, it implements an OAuth 2.0 password-bearer flow backed by Bcrypt-hashed credentials. FastAPI routers inject this dependency so that route protection is consistent everywhere. User feedback lands in the Issue-tracking service. Calls to *logIssue* create records in the *issues* table, while administrators resolve them through *updateStatus* or pull open tickets with *listOpenIssues*. Once an issue is closed, the Notification service can automatically inform the reporting user. Finally, the Admin & Monitoring service exposes privileged endpoints such as *fetchMetrics* or *triggerScaleOut*, giving operators insight into latency, error rates, and queue depth, and allowing manual scale actions when automated rules need a human override. Together, these subsystem services form a clean lattice of responsibilities: the UI focuses on rendering, Control on workflows, Business Logic on core computations, Persistence on data durability, ML on inference, and the ancillary services on cross-cutting concerns like auth, payments, notifications, and support. Because each layer speaks to the others only through these contracts, Plutos Equities remains modular, testable, and ready for future expansion.

4. Development/Implementation Details

4.1. Frontend

The frontend of Plutos Equities is developed using React, a modern JavaScript library that allows for building fast and scalable user interfaces. React enables the platform to deliver an interactive, user-friendly experience, with real-time updates and seamless navigation across different pages. The main dashboard provides users with a clear overview of their portfolio, displaying key information about their favorite companies, such as stock prices, price changes, and symbols. This dashboard also offers an intuitive search feature, allowing users to find companies and manage their portfolio easily. Users can view their data in either table or grid format, providing flexibility in interacting with the information. For each company in the portfolio, detailed financial data is displayed, including Earnings Per Share (EPS) projections. The EPS predictions are shown alongside historical data in a graph, which users can interact with to view different timeframes, such as the last quarter or year. As new data is processed, the predictions are automatically updated, keeping users informed with the latest forecasts. The platform also includes a company overview page, where users can access additional information about the company, including its market sector, CEO, and financial performance metrics. A radar chart visualizes these key metrics, offering an at-a-glance view of the company's financial health. The frontend ensures that stock prices are updated in real time, either through WebSockets or periodic polling, ensuring that users always see the latest market data without needing to refresh the page. Similarly, the accuracy of the stock predictions is clearly presented, showing users how closely the model's predictions align with actual market prices. Authentication and subscription features are also incorporated into the frontend. Users can log in securely and choose from different subscription plans that offer varying levels of access to the platform's features. The subscription page is designed with clear and concise information to help users select the plan that best fits their needs. Additionally, the responsive design ensures that the platform delivers a consistent experience across all devices, from desktop to mobile. Whether on a smartphone or a laptop, users can easily access and interact with the platform's features.

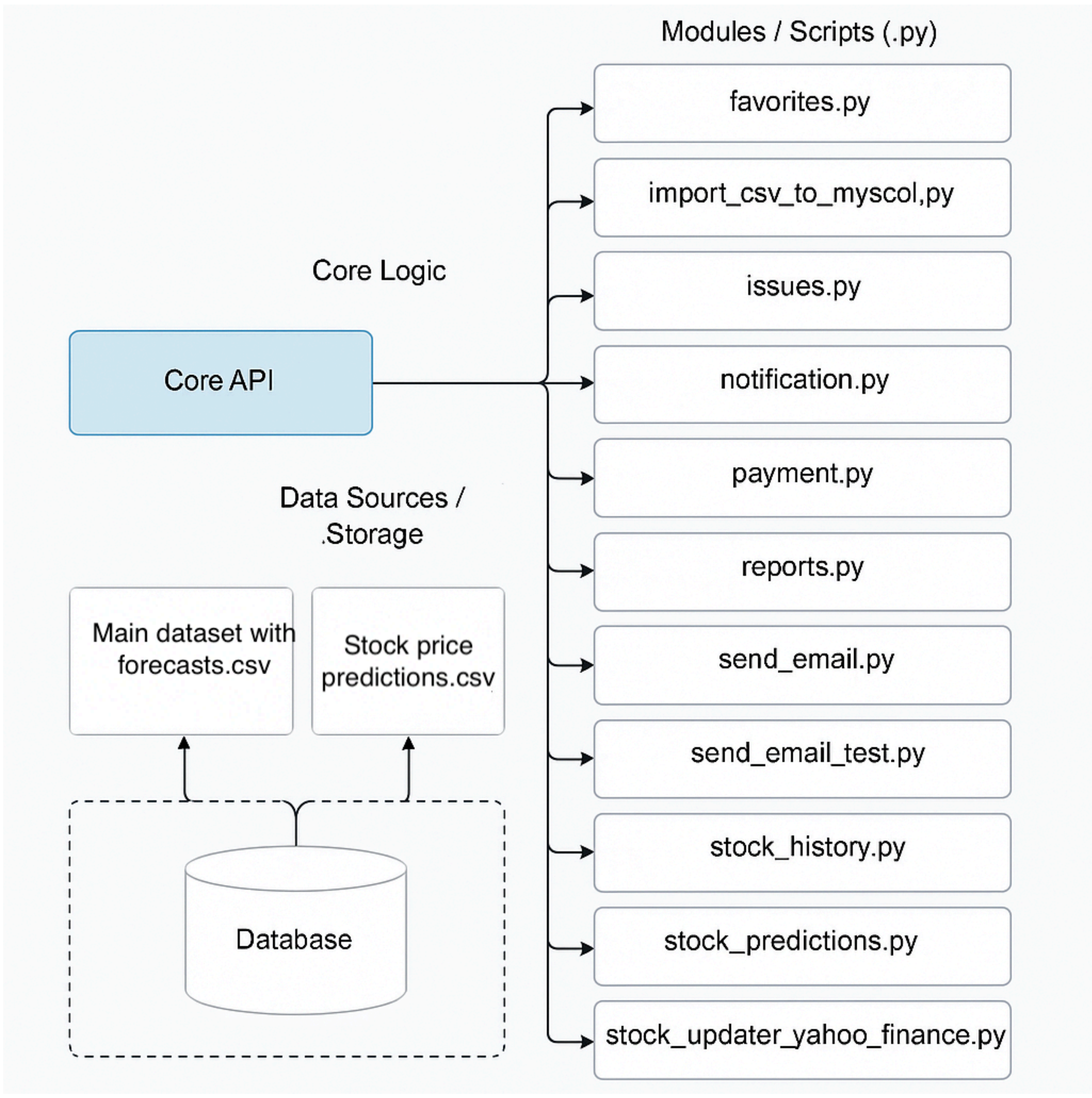
In terms of technology, React is used to build the user interface, with libraries like Chart.js being leveraged to render interactive graphs. Data is fetched through APIs, and real-time updates are managed to ensure users always have the most accurate and up-to-date information.

Link For the User Manual: **https://plutosequities.xyz/user-manual.html**

Project Website: **https://plutosequities.xyz/** **(Logbooks are in the website)**

Product Website: **https://www.plutosequities.com/**

4.2. Backend



FastAPI remains the work-horse of the service tier because it combines event-loop concurrency, type-safe request handling, and self-documenting OpenAPI contracts in a single, lightweight framework. The main.py file bootstraps the application, wires the CORS and logging middleware, and collects all feature routers so that every endpoint lives behind a single FastAPI() instance.

4.2.1. Project layout and responsibilities

The api package groups every public endpoint by feature so that each module is small, coherent, and testable. user_details.py exposes CRUD operations for the user profile, from registration through avatar upload. favorites.py lets an authenticated account follow or unfollow ticker symbols and stores the watch-list so the dashboard can hydrate instantly. Historical price series flow through two cooperating modules: stock_updater_yf.py pulls deltas from Yahoo Finance on a schedule, while stock_history.py answers client queries with the cached data. The heart of the product, stock_predictions.py, loads the latest ML artefacts from Azure ML, runs inference, attaches confidence bands, and returns a JSON payload optimised for high-resolution charts. Payments and monetisation are isolated inside payment.py. It listens for Stripe web-hooks, reconciles invoices, and flips the appropriate subscription flags in the auth token so that endpoints stay protected. User engagement is handled by notification.py, which publishes WebSocket pushes when, for instance, a forecast crosses a threshold the investor chooses. issues.py collects feedback tickets originating from the "Help & Feedback" form and stores them so that the admin console can triage. Outbound email—password-reset links, two-factor codes, monthly summaries—passes through send_email.py, with send_email_test.py providing a mock SMTP sink for the CI pipeline. Finally, import_csv_to_mysql.py is a one-off utility used during the initial population of legacy datasets and can be re-run whenever a bulk back-fill is required. Cross-cutting concerns live in the core package. auth.py implements an OAuth2 password-bearer flow; it hashes credentials with Bcrypt, stamps JWTs, and provides a dependency that any router can invoke for role-based gating. config.py centralises every environment variable—database DSNs, Stripe secret keys, Azure blob URIs—through a single Pydantic settings object, guaranteeing that a mis-typed var fails fast at boot time instead of runtime.

The db package owns persistence. connection.py spins up an async SQLAlchemy engine and session registry, while database.py holds entities. Physically, the data layer is partitioned into three logical databases. The core database keeps authoritative user accounts, subscription states, and audit logs.

The market database stores third-party data such as historical quotes and corporate actions. The prediction database captures every ML run, including the version of the model, the feature window, and the resulting forecast so that comparisons and rollbacks are trivial. All three live inside a single Azure SQL server yet remain isolated schemas; this arrangement simplifies backup strategy and enforces a clear boundary between mutable transactional state and append-only analytical records.

The utils directory collects tiny helper functions—date arithmetic, custom exceptions, structured logging decorators—while scripts serves as a workbench for cron-friendly tasks. A typical script might clear expired email-verification tokens, invoke the nightly stock_updater_yf.py job, or rebuild full-text indexes used by the sentiment engine.

4.2.2. Backend Workflow

Plutos Equities leverages a robust backend architecture to efficiently manage and deliver stock predictions and financial data to users. The backend handles everything from data storage and retrieval to payment processing, user registration, notifications, and issue tracking.

4.2.3. Data Flow and Stock Predictions

When a user accesses the dashboard, the backend processes a request to display the current stock prices of companies listed in the NASDAQ 100 index. This data is retrieved from our database, which is regularly updated with the latest stock price information pulled from market sources. As users interact with the dashboard, they can click on any company to view more detailed financial data. Upon selecting a company, a request is sent from the frontend to the backend, where it queries the database for the company's current stock price, predictions for future stock movements, and historical financial data for the current and previous years. This data is stored and regularly updated within the database to reflect the latest reports submitted to the SEC Edgar database [3]. The backend processes these queries and redirects the relevant information back to the user, ensuring the most up-to-date and accurate details are presented on the user's dashboard.

4.2.4. Payment System

Our backend securely integrates Stripe for payment processing. However, in its current stage, the payment system is in test mode due to regulatory requirements. Specifically, we are awaiting the completion of registration with the government and obtaining a tax number, which is essential for processing real payments. Once activated, users will be able to select from three distinct subscription tiers. Each subscription plan offers varying levels of access to premium features such as advanced analytics, additional company data, and forecasting accuracy.

4.2.5. Subscription Management and Downgrading

The system imposes specific constraints when users attempt to downgrade their subscription plan. To ensure that users downgrade responsibly, the system **requires them to choose fewer favorite companies** from their portfolio before the downgrade is processed. This ensures that users only have access to the amount of data corresponding to their subscription level and prevents any disruption in the service they receive. This step is essential for maintaining the integrity of the subscription model and ensuring that users are not taking advantage of premium services at a lower subscription tier.

4.2.6. Registration and Email Verification

User registration is an integral part of the backend flow. When a new user registers, an email verification system is triggered. The backend generates a **unique token**, which is then stored in the **MySQL database** until it expires. The token is linked to the user's account and is required for account activation. The user must confirm their email by clicking on a link sent to them, ensuring that only verified users gain full access to the platform.

4.2.7. Handling Notifications and Issues

Both **notifications** and **issues** are critical components of user experience and are handled with security and reliability in mind. The backend manages all notifications, such as alerts for stock price changes or updates to forecasts, ensuring that users are promptly notified. Additionally, **user-reported issues** are tracked and stored in the MySQL database, ensuring that all user concerns are addressed in a timely and efficient manner. Each issue is securely stored, and a resolution workflow is initiated as soon as a new issue is logged.

4.2.8. Security and Data Integrity

All critical data, including user registration information, payment details, and personal preferences (such as favorite companies), are stored securely in the **MySQL database**. Access to the database is tightly controlled, ensuring that only authorized entities can interact with sensitive information. By using **role-based access control (RBAC)** and **encryption**, the backend ensures that all user data is protected both in transit and at rest. This approach guarantees that our platform meets high standards of security, which is particularly crucial for handling financial data and sensitive user information.

4.2.9. Automation and System Management

The entire backend operates automatically, with minimal manual intervention required. The system is designed to autonomously handle data updates, user interactions, and process flows. The backend continuously pulls updated stock price data, financial reports, and prediction information, ensuring that users always have access to the most accurate, up-to-date data. Additionally, all other processes, from user registration to payment handling, issue resolution, and notifications, work seamlessly behind the scenes to provide a smooth user experience.

4.3. Database

We host our database on an **Azure MySQL server**, and currently, there are **12 main tables** that store essential data for **Plutos Equities**. These tables are crucial for managing stock predictions, financial data, user interactions, and more. Below is a description of each table (excluding predictions and company reports):

4.3.1. company

This table stores essential company information, such as the **company name**, **symbol**, **sector**, and other relevant attributes. It serves as the foundation for associating other data with specific companies within the system.

4.3.2. company_financials

The **company_financials** table holds financial data for each company, including key metrics like **revenue**, **earnings**, **profit margins**, and other financial indicators. This data supports features like **EPS (Earnings Per Share)** predictions and provides the historical financial data displayed on the user dashboard.

4.3.3. company_stocks

This table stores stock-related information for each company, such as **stock prices**, **market cap**, and other relevant financial metrics. It is used to display **real-time stock prices** to users and manage data about stock performance.

4.3.4. daily_prices

The **daily_prices** table tracks the **daily stock prices** for each company over time. It provides the platform with up-to-date information on stock price changes, feeding data into prediction models and enabling dynamic chart visualizations on the user interface.

4.3.5. User

This table manages **user-related data**, including **email**, **account status**, **preferences**, and potentially **favorite companies**. It tracks user interactions with the platform, such as managing stock portfolios and accessing predictions. Additionally, it is used for **email verification**, in conjunction with the **email_tokens** table, ensuring secure user registration and account management.

4.3.6. email_tokens

The **email_tokens** table stores **email verification tokens** generated during user registration or password recovery. Tokens are linked to a user's account and stored temporarily until the user confirms their email address, ensuring only verified users can activate or modify account details.

4.3.7. filtered_data_2023_onwards

This table contains **pre-processed data** used for prediction models starting from 2023 onward. It stores **cleaned and filtered data** from various sources, such as **financial reports**, **stock prices**, or **external datasets**, which are ready for analysis and prediction.

4.3.8. issues

The **issues** table tracks **user-reported issues** and feedback. It helps the platform efficiently manage concerns like **bugs**, **feature requests**, and other inquiries. Each issue record includes user IDs, descriptions, and timestamps, ensuring customer support can address issues promptly.

4.3.9. netincome_comparison

This table holds **net income comparisons** between different time periods or companies. It provides insights into a company's profitability, allowing the platform to generate comparative analysis, reports, or forecasts based on financial data trends.

4.3.10. reports

The **reports** table stores **financial reports**, either **automatically generated** or **user-generated**. These reports may include **summaries of stock predictions**, **company analysis**, and other key financial metrics, which users can access and download through the platform.

4.3.11. stock_prediction_comparison

This table stores data related to the comparison between **predicted stock prices** and **actual market performance**. It plays a crucial role in evaluating the accuracy of the platform’s **prediction models** and providing users with confidence in the forecasting system's reliability.

4.3.12. stock_predictions_5day

The **stock_predictions_5day** table stores **5-day stock price predictions** for each company. It provides **short-term forecasts**, offering users valuable insights into upcoming market trends for a five-day period, which helps users make timely investment decisions.

4.4. Model

4.4.1. Stock Price Prediction

For stock price forecasting, we use a time-series approach based on Long Short-Term Memory (LSTM) neural networks [7]. This architecture captures temporal dependencies and volatility patterns in historical price data, enabling short-term price movement prediction with improved sensitivity to market trends. The LSTM model consists of a single LSTM layer, followed by three fully connected (dense) layers [7]. This compact yet effective design allows the model to process sequential input and produce point estimates for future stock prices. We experimented with various model architectures for stock price prediction, but this particular setup proved to be the most effective. Although it does not follow a conventional deep learning architecture, it is a commonly used structure in time-series forecasting tasks, especially in competitive prediction challenges.

4.4.2. Quarter Report Feature Prediction

We focused our predictive modeling efforts on a set of key financial indicators, including **Earnings Per Share (EPS), Gross Profit, Income Tax Expense/Benefit, Net Income/Loss, Operating Income/Loss, and Weighted Average Number of Shares Outstanding (Basic)**. These variables were selected because they are fundamental components of a company's financial performance and are closely monitored by investors, analysts, and auditors.

NetIncomeLoss and **OperatingIncomeLoss** serve as core profitability metrics that reflect both overall and operational efficiency.

EarningsPerShareBasic is a critical per-share measure of profitability, often used in valuation ratios and investor decision-making.

GrossProfit provides insight into the company's production efficiency and pricing power.

IncomeTaxExpenseBenefit is essential for understanding tax strategy, deferred tax effects, and their impact on net results.

WeightedAverageNumberOfSharesOutstandingBasic is necessary for EPS calculation and dilution analysis, and it can also change due to corporate actions like buybacks or stock issuance.

By accurately forecasting these values before earnings reports are released, our models aim to provide forward-looking insights that can enhance investor strategies, risk assessments, and audit planning.

During the process, we explored a wide range of modeling approaches to predict quarterly financial report metrics. These included traditional linear models (Linear Regression, Ridge, Lasso), tree-based ensemble methods (Random Forest), deep learning architectures (LSTM, GRU), and simple heuristic baselines (e.g., last quarter's value, same quarter from last year). Among these, Random Forest consistently outperformed others in terms of mean absolute error and general robustness, especially when combined with engineered features and evaluated via leave-one-out cross-validation (LOOCV) [7].

Our feature engineering strategy combined both structured and unstructured data. From the structured side, we used lagged financial values up to lag9 to build a time-aware view of the company's performance. We also calculated stock price change ratios between quarters to capture market dynamics. On the unstructured side, we extracted inline textual content from prior reports and used BERT-based models to generate sentiment scores that reflected managerial tone and forward-looking language [8]. These features were beneficial in capturing qualitative signals that traditional numerical metrics might miss.

To evaluate and identify our best-performing models, we split the dataset chronologically: all data prior to 2023 was used for training, while data from 2023 onward was reserved for testing. This time-based split ensures realistic model evaluation by preventing future information leakage into the training set and simulating a real-world forecasting scenario. As an exception, we also employed a Leave-One-Out Cross-Validation (LOOCV) strategy, where the model is trained on all available data except for a specific filing date, which is used for validation. This method provides a more granular performance assessment, especially useful for measuring how well the model generalizes across time and companies. While LOOCV is computationally intensive, it offers a good validation in cases where the dataset is limited or unevenly distributed across time which is very similar to our case. Therefore we got the best results with random-forest LOOCV model.

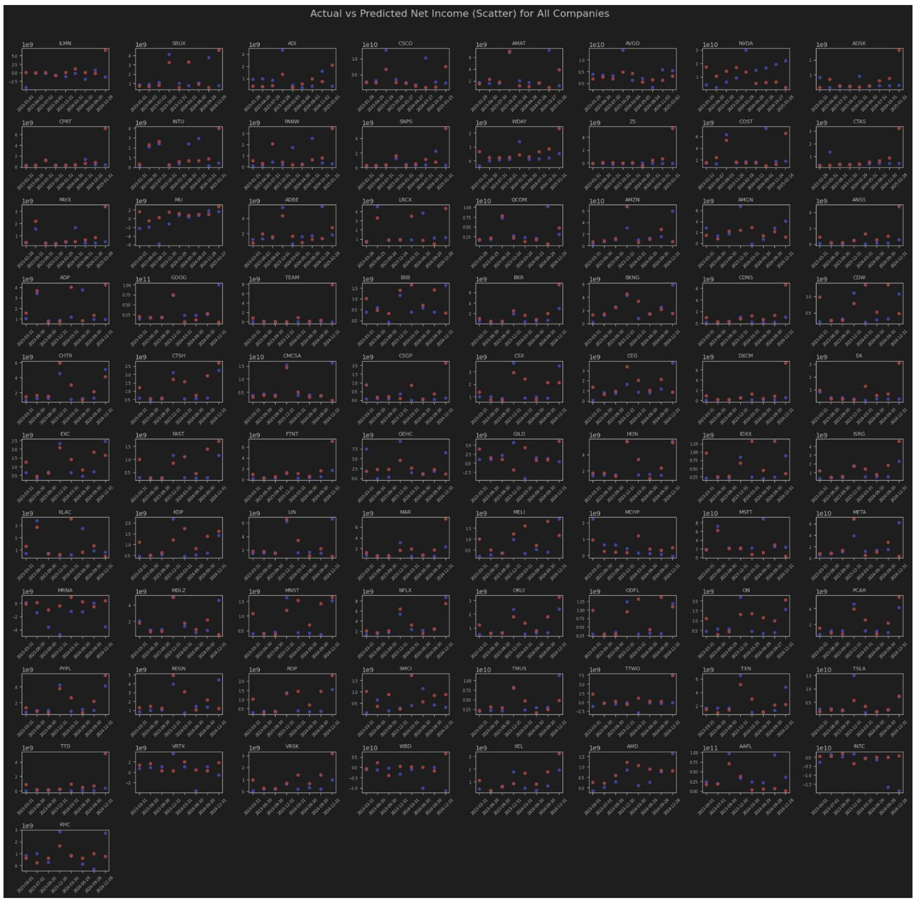
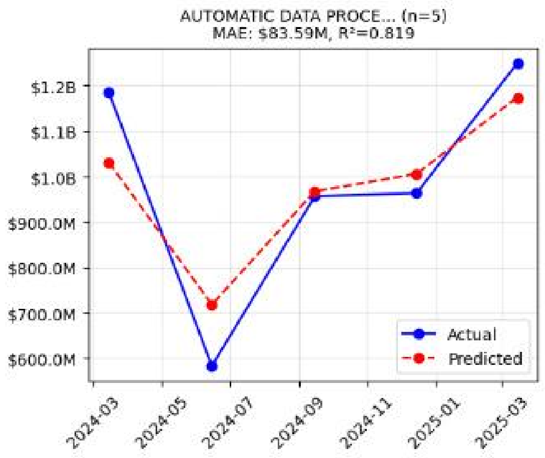
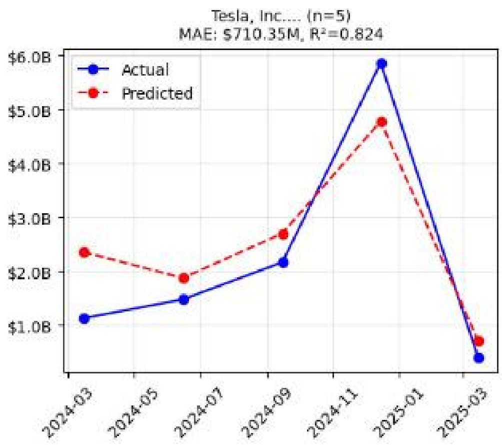


Table 1: GrossProfit Test Set





Prediction Results vs Actual Values

4.5. Cloud

Plutos Equities leverages Azure Cloud Services to power its infrastructure, ensuring scalability, reliability, and efficient performance. The platform utilizes Azure

SQL Database for structured data storage, where essential financial data, stock prices, and user information are securely stored and continuously updated. For handling large volumes of unstructured data, such as financial reports and textual data for sentiment analysis, we use Azure Storage Accounts. To deploy and manage our machine learning models, we utilize Azure Machine Learning, which streamlines the training, evaluation, and deployment of our predictive models. Additionally, by using Azure Virtual Machines, we maintain control over cost efficiency, processing stock queries and optimizing backend performance. This cloud architecture ensures that the platform is highly available, flexible, and capable of handling thousands of users and millions of data points, with the ability to scale up as the platform grows.

5. Test Cases and Results

In our testing strategy, we tried to test each and every part of the platform, ranging from frontend interactions to backend to ensure functionality, reliability, and performance. We started with unit testing, where individual functions were tested one by one in our FastAPI backend. These units were both manually and automatically tested, and Python's unittest framework was supplemented using Selenium to simulate user interaction within the browser. Selenium tests covered major frontend features like user registration, log in, visualizing financial data, and prediction. They also verified that information fetched from our backend (for example, by using yfinance or SEC APIs) was appropriately rendered on the UI and did not change with user sessions. Then we performed integration tests, bringing together modules like the database, RESTful API endpoints, and UI components to surface defects due to module interactions. We followed a bottom-up integration testing approach, incrementally building up from individual services to end-to-end workflows.

After integration testing, we conducted system testing for the integrated application. We performed black-box testing techniques to make sure that the system met all the functional as well as non-functional requirements. Acceptance testing ensured that the application met users' expectations in terms of performance, usability, and data integrity standards. In non-functional testing, we utilized Locust for performing load and stress testing, which simulated up to 200 concurrent users.

5.1. Functional Test Cases

Test ID	T-01	Category	Functional / Integration	Severity	Critical
Objective	Ensure registration fails when invalid or missing data is provided.				
Steps	1. Navigate to the Sign-Up page. 2. Enter invalid data (e.g., already registered email, improperly formatted email, or leave required fields blank). 3. Click “Sign Up”.				
Expected	Registration fails and an error message is displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-02	Category	Functional / Integration	Severity	Critical
Objective	Confirm a new user is created when valid data is provided.				
Steps	1. Navigate to the Sign-Up page. 2. Enter valid registration information (unique email, strong password, all required fields). 3. Click “Sign Up”. 4. Verify that a confirmation email is sent.				
Expected	User is registered and a new database entry is created.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-03	Category	Functional / Integration	Severity	Critical
Objective	Verify that login fails when incorrect credentials are provided.				
Steps	1. Navigate to the Login page. 2. Enter an incorrect email or password. 3. Click “Sign In”.				
Expected	Login fails with an appropriate error message.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-04	Category	Functional / Integration	Severity	Critical
Objective	Confirm successful login with correct credentials.				
Steps	1. Navigate to the Login page. 2. Enter valid email and password. 3. Click “Sign In”.				
Expected	User is logged in and redirected to the Analysis page where companies are shown.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-05	Category	Functional	Severity	Major
Objective	Ensure that the “Forgot Password” process sends a reset link.				
Steps	1. Click the “Forgot Password” link on the Login page. 2. Enter the registered email address.				

	3. Submit the request.
Expected	A password reset link is sent to the email.
Date-Result	01.05.2025 - PASSED

Test ID	T-06	Category	Functional	Severity	Critical
Objective	Verify that a user can log out successfully.				
Steps	1. While logged in, click the “Logout” button on the NavBar.				
Expected	User is logged out and redirected to the Login page.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-07	Category	Functional	Severity	Major
Objective	Verify that a user can successfully change their username.				
Steps	1. Log in with valid credentials. 2. Navigate to the profile or account settings page. 3. Locate the "Change Username" section. 4. Enter a new valid username and confirm the action. 5. Save the changes.				
Expected	The new username is saved and displayed on the profile. A success message is shown.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-08	Category	Functional	Severity	Major

Objective	Verify that a user can successfully change their email address.
Steps	1. Log in with valid credentials. 2. Navigate to the profile or account settings page. 3. Locate the "Change Email" section. 4. Enter a new valid email and confirm the action. 5. Verify the email via confirmation link.
Expected	The new email address is saved and reflected in the user profile. A confirmation message is shown.
Date-Result	01.05.2025 - PASSED

Test ID	T-09	Category	Functional	Severity	Major
Objective	Verify that a user can successfully change their password.				
Steps	1. Log in with valid credentials. 2. Navigate to the profile or account settings page. 3. Locate the "Change Password" section. 4. Enter the current password, new password, and confirm the new password. 5. Save the changes. 6. Log out and attempt to log in with the new password.				
Expected	Password is updated successfully. User is able to log in with the new password and not with the old one.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-10	Category	Functional	Severity	Major
Objective	Check that personalized data (e.g., watchlist, recent predictions) is loaded				

	after login.
Steps	1. Log in with valid credentials. 2. Navigate to the Dashboard.
Expected	Dashboard displays correct personalized financial data.
Date-Result	01.05.2025 - PASSED

Test ID	T-11	Category	Functional	Severity	Major
Objective	Ensure users can add a company to their watchlist.				
Steps	1. Log in with valid credentials. 2. Navigate to the "Stock" page of a company. 3. Click the “Follow” (★) button to add the company to the watchlist. 4. Navigate to the "Dashboard" page.				
Expected	The followed company appears in the Dashboard under the watchlist. A success confirmation is shown.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-12	Category	Functional	Severity	Major
Objective	Ensure users can remove a company from their watchlist.				
Steps	1. Log in and go to the "Dashboard" page. 2. Locate a company currently on the watchlist. 3. Click the “Unfollow” (🗑️ or ★ again) button next to the company name.				

Expected	The company is removed from the watchlist, and the Dashboard no longer displays it. [13]
Date-Result	01.05.2025 - PASSED

Test ID	T-13	Category	Functional / Integration	Severity	Major
Objective	Ensure users can navigate to the Predictions page.				
Steps	1. From the “Dashboard” page, click the desired company from the watchlist.				
Expected	Predictions page loads with the latest forecast data.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-14	Category	Functional / Integration	Severity	Critical
Objective	Verify accurate forecast data is displayed for a selected company.				
Steps	1. On the “Analysis” page, select a company (e.g., Apple Inc.). 2. Observe the displayed data (Net Income, EPS etc.).				
Expected	Accurate forecast data and associated metrics are displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-15	Category	Functional	Severity	Major
Objective	Ensure confidence intervals are displayed with each forecast metric.				

Steps	1. On the Predictions page, review a company's forecast data.
Expected	Confidence intervals accompany each prediction.
Date-Result	01.05.2025 - PASSED

Test ID	T-16	Category	Functional / Integration	Severity	Major
Objective	Verify that changes in data trigger notifications.				
Steps	1. Simulate a change in financial data affecting predictions (such as update on stock-parameter prediction). 2. Monitor for an alert on the device.				
Expected	A notification is displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-17	Category	Functional	Severity	Minor
Objective	Confirm that predictions can be exported in CSV format.				
Steps	1. On the stock page of the individual company, click "Export". 2. Select CSV format.				
Expected	A CSV file with correct prediction data is downloaded.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-18	Category	Functional	Severity	Minor
Objective	Confirm that predictions can be exported in XLS format.				

Steps	1. On the stock page of the individual company, click “Export”. 2. Select XLS format.
Expected	An XLS file with correct prediction data is downloaded.
Date-Result	01.05.2025 - PASSED

Test ID	T-19	Category	Functional	Severity	Major
Objective	Verify that user profile information can be updated successfully.				
Steps	1. Navigate to the Profile page. 2. Click the “Edit” icon and modify profile information. 3. Click “Save”.				
Expected	Profile is updated and a success message is shown.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-20	Category	Functional	Severity	Major
Objective	Ensure the system handles empty fields appropriately during profile update.				
Steps	1. Navigate to the Profile page. 2. Click “Edit” and clear all required fields. 3. Click “Save”.				
Expected	An error message is displayed; profile is not updated.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-21	Category	Functional /	Severity	Critical

			Integration		
Objective	Ensure accurate financial data is fetched using the Yahoo Finance API				
Steps	1. Trigger a backend request for a known company using the Yahoo Finance API (via yfinance).				
Expected	Accurate historical and current financial data is retrieved.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-22	Category	Functional / Integration	Severity	Critical
Objective	Confirm the yfinance library returns complete financial data.				
Steps	1. Initiate a backend call using the yfinance library for a specific company.				
Expected	The library returns accurate and complete data.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-23	Category	Functional / Integration	Severity	Major
Objective	Verify that financial statements are retrieved from SEC EDGAR.				
Steps	1. Trigger a process to fetch financial statements from SEC EDGAR for a company.				
Expected	Required statements are successfully retrieved and processed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-24	Category	Functional / Integration	Severity	Critical
Objective	Verify the ML model produces forecasted financial metrics.				
Steps	1. Trigger the ML model prediction process for a company.				
Expected	Forecasted metrics (e.g., revenue, EPS) are generated, displayed and saved to the database.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-25	Category	Functional	Severity	Major
Objective	Ensure users can change their subscription plans successfully.				
Steps	1. Navigate to the Subscription section in the user profile. 2. Change the subscription plan. 3. Confirm the change.				
Expected	Subscription is updated correctly in the account.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-26	Category	Functional / Integration	Severity	Major
Objective	Verify (and simulate) that payment processing works as expected.				
Steps	1. Navigate to the Subscription update page. 2. Select a new plan and enter test payment details. 3. Process the payment.				

Expected	Payment is processed and subscription is updated with confirmation.
Date-Result	01.05.2025 - PASSED

Test ID	T-27	Category	Functional / Administration	Severity	Major
Objective	Confirm that admins can access and use the dashboard tools.				
Steps	1. Log in as an admin. 2. Navigate to the Admin Dashboard. 3. Verify availability of user management, configuration options and opened issues.				
Expected	Admin Dashboard loads with all functionalities accessible.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-28	Category	Functional / Integration	Severity	Major
Objective	Ensure historical financial trends are rendered as graphs.				
Steps	1. Navigate to “View Historical Data Trends” from the individual stocks page. 2. Select a historical timeframe. 3. Verify that a graph (e.g., quarterly revenue, EPS) is rendered correctly.				
Expected	A graph with clear labels and accurate data is displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-29	Category	Functional / Integration	Severity	Major
Objective	Verify overall health metrics of a company are presented.				
Steps	1. Navigate to a specific company page. 2. Select a company to view health metrics (e.g., revenue growth, expenses).				
Expected	Analysis page displays comprehensive metrics and visual indicators.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-30	Category	Functional / Integration	Severity	Major
Objective	Ensure daily stock price, market cap and daily change data is accurately fetched and displayed.				
Steps	1. On a company's detail page, navigate to the Stock Prediction section. 2. Observe the displayed data.				
Expected	Data from Yahoo Finance API is shown accurately.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-31	Category	Functional / Integration	Severity	Critical
Objective	Check that historical predictions and actual outcomes are compared accurately.				

Steps	1. Navigate to “Compare Predictions vs. Outcomes”. 2. Select a company and timeframe. 3. Verify both prediction and actual outcome data are displayed side by side.
Expected	Accurate comparison view with clearly labeled data.
Date-Result	01.05.2025 - PASSED

Test ID	T-32	Category	Functional	Severity	Minor
Objective	Ensure users can delete notifications from the UI.				
Steps	1. Navigate to the Notifications section. 2. Select one or more notifications and click “Delete”. 3. Confirm deletion if prompted.				
Expected	Notifications are removed from the UI and do not reappear.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-33	Category	Functional	Severity	Major
Objective	Confirm that users can submit feedback/issues successfully.				
Steps	1. Navigate to “Feedback & Help Center”. 2. Fill in the feedback form (rating, comments). 3. Click “Submit”.				
Expected	Feedback is submitted and a confirmation message is shown.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-34	Category	Functional	Severity	Major
Objective	Confirm that an admin can view and mark a submitted feedback issue as resolved.				
Steps	1. Log in as an admin user. 2. Navigate to the “Admin Panel” and “Feedback Management” section. 3. Locate a submitted feedback item from the list. 4. Click on the feedback to open details. 5. Click the “Resolve” button.				
Expected	The feedback is marked as resolved, and its status is updated accordingly in the system. A confirmation message is shown to the admin.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-35	Category	Functional	Severity	Major
Objective	Verify that searching by ticker returns the correct company data.				
Steps	1. On the Analysis page, enter a ticker (e.g., “AAPL”) in the search bar. 2. Initiate the search.				
Expected	Correct company details and financial data are displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-36	Category	Functional	Severity	Minor
Objective	Ensure recommendations are displayed in the Watchlist page.				
Steps	1. Log in as a registered user. 2. Navigate to the “Watchlist”page.				

Expected	Recommendations are displayed on the screen.
Date-Result	01.05.2025 - PASSED

Test ID	T-37	Category	Functional / Integration	Severity	Major
Objective	Verify data updates on one device are reflected on another in real time.				
Steps	1. Log in simultaneously on two devices (e.g., mobile and desktop). 2. Update profile information on one device.				
Expected	Changes synchronize across devices with minimal delay.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-38	Category	Functional / Error Handling	Severity	Major
Objective	Ensure the system handles empty API responses gracefully.				
Steps	1. Simulate an API returning an empty response (e.g., Yahoo Finance for a company with missing data). 2. Attempt to load the data.				
Expected	A clear error or fallback message is displayed, and the UI remains stable.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-39	Category	Functional	Severity	Critical

Objective	Ensure that when a user deletes their account, all associated data is removed.
Steps	1. Navigate to the Profile page. 2. Click “Delete Account” and confirm.
Expected	Account and all associated data (watchlist etc.) are removed from the system.
Date-Result	01.05.2025 - PASSED

Test ID	T-40	Category	Functional	Severity	Critical
Objective	Ensure that an admin can delete a user account and that all associated data is removed from the system.				
Steps	1. Log in as an admin user. 2. Navigate to the “Admin Panel” section. 3. Search for and select the target user account. 4. Click “Delete Account” and confirm the deletion.				
Expected	The selected user account is deleted, and all associated data (e.g., watchlist etc.) is permanently removed from the system. A confirmation message is displayed.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-41	Category	Functional	Severity	Major
Objective	Confirm that predictions auto-refresh without manual intervention.				
Steps	1. On the specific company page, leave the page open for the auto-refresh interval.				

	2. Observe if predictions update automatically.
Expected	Predictions refresh automatically without errors.
Date-Result	01.05.2025 - FAILED

5.2. Non-Functional Test Cases

Test ID	T-42	Category	Security	Severity	Critical
Objective	Ensure that user passwords are stored securely in an encrypted (hashed) format.				
Steps	1. Register a new user account. 2. Inspect the stored password in the database using secure means.				
Expected	Password is stored in an encrypted format.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-43	Category	Security	Severity	Critical
Objective	Verify that restricted pages and API endpoints cannot be accessed without proper authentication.				
Steps	1. Attempt to access secured areas without logging in.				
Expected	Access is denied with an appropriate error or redirection to the Login page.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-44	Category	Usability /	Severity	Major

			Compatibility		
Objective	Ensure the Dashboard displays properly on various mobile devices and screen sizes.				
Steps	1. Access the Dashboard on multiple mobile devices (different OS and screen sizes). 2. Verify that all elements render correctly and are functional.				
Expected	UI adjusts responsively; functionality is maintained across devices.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-45	Category	Compatibility	Severity	Major
Objective	Ensure the web application functions consistently across different browsers.				
Steps	1. Open the application in browsers such as Chrome, Firefox, and Edge. 2. Navigate through key pages (Login, Dashboard, Predictions).				
Expected	UI and functionality are consistent across all tested browsers.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-46	Category	Performance	Severity	Critical
Objective	Verify that the system can handle heavy user load.				
Steps	1. Use a performance testing tool (Locust) to simulate 200 concurrent users performing typical operations (login, data retrieval, etc.) [12].				
Expected	Response times remain below 2 seconds per request with acceptable performance degradation.				

Date-Result	01.05.2025 - FAILED

Test ID	T-47	Category	Performance	Severity	Critical
Objective	Ensure financial data fetching is performed within an acceptable time frame.				
Steps	1. Measure the time taken for the system to fetch data for a selected company.				
Expected	Response time is less than 2 seconds.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-48	Category	Security	Severity	Critical
Objective	Ensure that secured API endpoints are accessible only with a valid bearer token.				
Steps	1. Attempt to access a secured API endpoint without or with an invalid token.				
Expected	Access is denied with an appropriate error message.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-49	Category	Usability	Severity	Minor
Objective	Confirm that in-app documentation or help content is easily accessible.				
Steps	1. Click on the “Help” or “Documentation” link within the app.				
Expected	Detailed, user-friendly documentation is displayed.				

Date-Result	01.05.2025 - PASSED

Test ID	T-50	Category	Performance	Severity	Major
Objective	Verify that cached data is used to reduce load times.				
Steps	1. Request the same financial data twice in succession.				
Expected	The second request is served faster via the cache with correct data.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-51	Category	Compliance	Severity	Critical
Objective	Confirm that users must acknowledge the data privacy policy during registration or first login.				
Steps	1. During registration/first login, verify that the data privacy policy is presented and requires user consent.				
Expected	User must agree to the privacy consent before proceeding.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-52	Category	Security / Usability	Severity	Critical
Objective	Ensure that inactive sessions time out and prompt for re-login.				
Steps	1. Log in and leave the application idle until the session timeout elapses. 2. Attempt an action after timeout.				

Expected	User is automatically logged out and prompted to log in again.
Date-Result	01.05.2025 - PASSED

Test ID	T-53	Category	Reliability	Severity	Critical
Objective	Ensure that the system maintains high uptime over an extended period.				
Steps	1. Monitor system operation continuously (via logs or simulation) over an extended period.				
Expected	Uptime of at least 99% with no critical failures.				
Date-Result	01.05.2025 - PASSED				

Test ID	T-54	Category	Usability	Severity	Major
Objective	Ensure that the application is user-friendly and intuitive.				
Steps	1. Conduct a usability session with representative end users. 2. Collect qualitative and quantitative feedback on UI clarity, navigation, and overall experience.				
Expected	Feedback indicates high usability with minimal areas of confusion.				
Date-Result	01.05.2025 - PASSED				

6. Maintenance Plan and Details

While the current deployment of our project on Azure with a FastAPI backend is fully functional and capable of serving early-stage users, future growth and reliability demand a comprehensive maintenance and enhancement plan. This plan focuses on improving system robustness, ensuring better uptime, and preparing for higher usage and data volumes in a scalable, cost-effective manner.

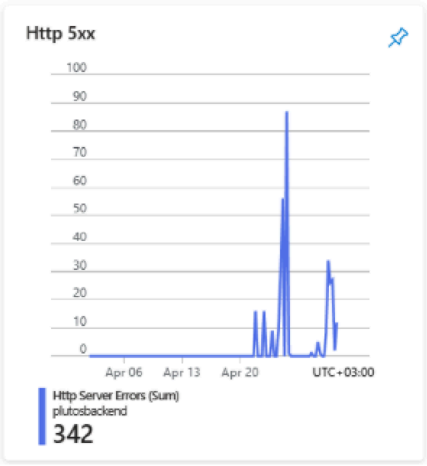
One of the important priorities is establishing a more proactive monitoring and alerting pipeline. Although Application Insights and Azure Monitor are already integrated, additional alerts will be configured for key failure indicators—such as HTTP 5xx spikes, latency surges, and elevated memory usage. Recent telemetry has indicated response time spikes and over 300 server errors within short traffic bursts. These will be addressed by improving observability using Azure Log Analytics and enabling autocorrection scripts that can temporarily scale out resources or isolate failing components.

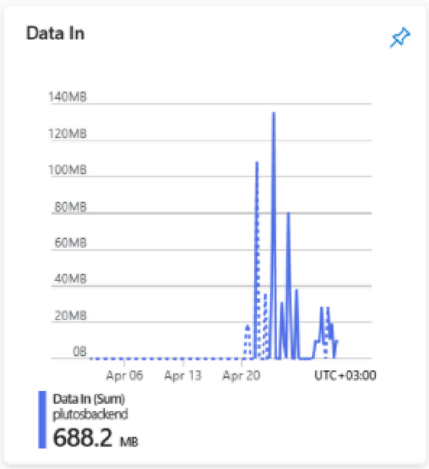
To address these bottlenecks, scaling strategies will be expanded. Currently hosted on basic Azure App Services, the backend will be migrated to more powerful compute options such as Azure Premium V3 or dedicated Virtual Machine Scale Sets. These will allow scaling based on live load, especially during peak financial report periods when user activity and API calls increase. Load balancing and distributed task queues—powered by Celery and potentially Azure Service Bus—will be introduced to offload long-running operations and ML inference.

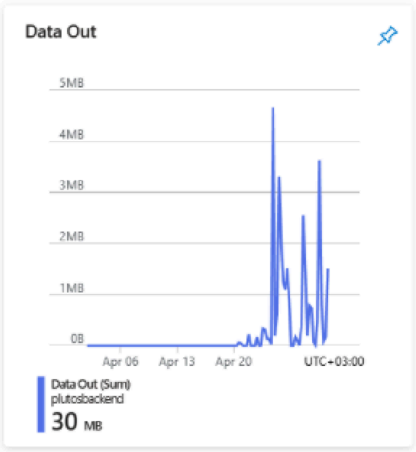
Backend performance will also be strengthened by adopting async SQLAlchemy for all database interactions, allowing for non-blocking I/O and better concurrency. Frequently accessed endpoints, like stock forecasts and financial analysis pages will be optimized with caching layers using Azure Cache for Redis, reducing latency and offloading pressure from the database and inference pipeline (first attempts with Celery and Redis are already done in our local machines which shows great initial results).

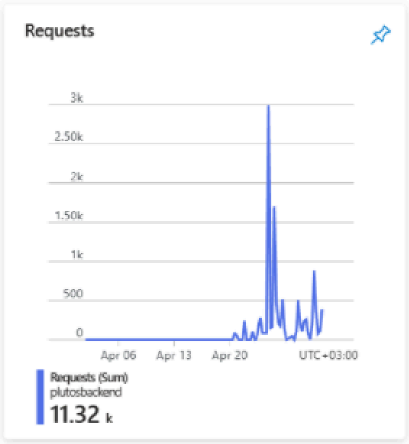
From a reliability point of view full disaster recovery planning will be formalized. This includes adding redundant backups and automated restore tests. Security and dependency management are crucial as well. A CI/CD pipeline will be extended to include vulnerability scanning.

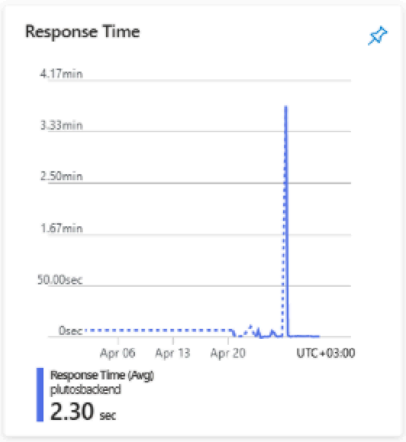
To sum up, there is a room and plan for maintenance. While the platform operates reliably under current loads, we ,as Plutos Equities, want to anticipate future usage spikes, support scaling to thousands of users, and maintain service quality under stress.











7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

7.1.1. Constraints

7.1.1.1. Implementation Constraints

- ❖ **Diverse Data Sources**: Our platform integrates both structured (e.g., financial statements) and unstructured data (e.g., earnings call transcripts, news). Managing this diversity requires advanced parsing and data integration techniques.
- ❖ **Real-Time Data Processing**: Handling real-time updates, such as intraday financial news or stock price fluctuations, demands a robust streaming architecture, which adds to system complexity.
- ❖ **Unstructured Data Analysis**: Extracting insight from textual sources necessitates natural language processing (NLP), which introduces further model complexity and potential misinformation risks.
- ❖ **Data Reliability**: Model accuracy is highly sensitive to the quality and reliability of financial data. Ensuring trustworthy sources and implementing preprocessing pipelines is essential.
- ❖ **Infrastructure Complexity**: Hosting ML models and storing large datasets require scalable cloud infrastructure. The use of services like Azure introduces architectural and cost-related challenges.
- ❖ **Model Maintenance**: Forecast models must be continuously monitored and updated to adapt to changing market conditions. This includes retraining on recent data and validating performance over time.
- ❖ **Testing Requirements**: Proper model validation using cross-validation, backtesting, and stress-testing techniques is critical to ensure robustness under various market scenarios.
- ❖ **Data Leakage**: Avoiding forward-looking bias in time-series data is particularly challenging. Strict separation of training and test periods is essential.

❖ **Time Constraints**: Model development must be completed prior to quarterly earnings announcements to validate real-world applicability within one reporting cycle.

7.1.1.2. Economic Constraints

- **API Subscription Costs**: Access to real-time and historical financial data via APIs such as Alpha Vantage and
- Yahoo Finance, or other premium providers is costly, especially at higher usage tiers.
- **Cloud Hosting Costs**: Utilizing Azure for hosting, compute, and storage incurs ongoing expenses, particularly when handling heavy data processing loads.
- **Scalability Costs**: Increased platform usage leads to the need for elastic scaling, including load balancing and database optimization, which further raises hosting and infrastructure costs.
- **Data Storage Requirements**: Storing and managing extensive financial data archives, particularly those involving unstructured content like news and reports, requires significant storage allocation.

7.1.1.3. Ethical Constraints

- **Prediction Transparency**: The platform communicates prediction logic, including data sources and modeling limitations, to ensure clarity and prevent misinterpretation.
- **Privacy and Data Security**: Although the platform does not collect personal data except user emails, all financial data accessed through APIs or third-party services is handled securely and in compliance with data protection laws [9][10].
- **Advisory Nature of Forecasts**: Forecasts are presented with disclaimers emphasizing that outputs are probabilistic and not financial advice.
- **Misuse Prevention**: Terms of use and user agreements clarify ethical boundaries, preventing misuse of forecasts for market manipulation or insider-like decision-making.

7.1.1.4. Social Constraints

- While there are no direct social constraints, we emphasize responsible usage and transparency. The platform must clearly communicate uncertainty and avoid fostering overconfidence in predictions. There is also a help center messaging logic so that users can directly contact the support team in case of a problem.

7.1.2. Standards

Our system adheres to market data standards by utilizing globally embraced formats and protocols in financial data processing. These include data pulled from reputable sources such as Bloomberg, Reuters, and official public filings like the U.S. Securities and Exchange Commission (SEC). Formatting consistency is paramount in maintaining data integrity across all sections of the system.

The system follows the legal standards for the use of financial information. All of the data sources are verified to be public, licensed, or open for non-commercial and academic use. Insider information is carefully excluded from all stages of model training or data processing.

We also comply with the General Data Protection Regulation (GDPR) and other relevant data privacy regulations. Although our platform does not directly obtain personal data from users, we have strict policies on how we handle potentially sensitive information, especially when integrating with APIs or indirectly identifiable user interactions are handled.

At the infrastructure level, the project follows best practices in cloud infrastructure as per the Azure Well-Architected Framework. This gives confidence that any component on the cloud follows best practices in security, operational reliability, performance efficiency, and cost optimization. Our server setups, backups, and deployment pipelines are designed to ensure these standards are followed.

For easier use, the application is developed following user interface and experience design principles. We prioritize clarity, simplicity, and interactivity. From time to time, we hold usability testing sessions with sample users to fine-tune design elements and navigation.

All code developed for the project adheres to clean code and test principles. We value readability, modularity, and maintainability in every aspect of the codebase. In particular, backend developers were required to write unit tests and integration tests alongside development for ease of debugging.

Additionally, we follow key object-oriented design standards to support a robust and maintainable system architecture. The Singleton pattern is used to centrally manage system-wide settings and services, ensuring consistent behavior across the platform. The Observer pattern enables dynamic updates and user notifications when new financial reports or forecast updates are available. To effectively track and manage the state of financial reports (e.g., quarterly updates), we utilize the State pattern, which allows the system to respond to changing conditions and report phases efficiently. These design patterns not only improve code quality but also support the scalability and flexibility of the platform [14].

7.2. Ethics and Professional Responsibilities

Throughout the Plutos Equities project, the team maintained a high level of professional and ethical standards. Given the financial forecasting focus of our platform, we consistently ensure data privacy and confidentiality by implementing secure storage solutions, encrypted communication channels, and strict access control policies. All team members completed training sessions on ethical data handling and signed non-disclosure agreements before accessing proprietary datasets. The integrity and transparency of our predictions were rigorously maintained through clear documentation of model assumptions, explicit reporting of confidence intervals, and open communication of potential biases and limitations. Every external dataset, financial model, API, and academic reference was meticulously cited and attributed to respect intellectual property rights, and all code dependencies were licensed appropriately. Regular stakeholder reviews and consent checks were incorporated to verify that project objectives remained aligned with ethical guidelines and user expectations.

7.3. Teamwork Details

7.3.1. Contributing and functioning effectively on the team to establish goals, plan tasks, and meet objectives

From the project's inception, the team embraced Agile methodologies and leveraged Jira for comprehensive issue tracking. We held planning workshops and used Jira boards to define user stories, estimate effort with story points, and assign tasks [15]. Key actions included:

- **Sprint Planning**: Before each one- or two-week sprint, team members reviewed Jira backlogs, estimated tasks, and identified dependencies to prevent bottlenecks.
- **Task Breakdown & Tracking**: Features like data ingestion, feature engineering, and model deployment were decomposed into granular Jira issues. Progress was tracked through issue statuses and sprint burndown charts.
- **Progress Monitoring**: Daily stand-ups and mid-sprint reviews provided forums for status updates and blocker resolution. Sprint burndown charts in Jira visualized remaining work and informed mid-course adjustments.
- **Version Control**: GitHub was used for all source code management. We adopted a Git branching strategy with feature branches, pull requests, and peer reviews to ensure code quality and traceability.

Regular meetings were scheduled weekly or twice weekly, with daily check-ins via WhatsApp Meet and Zoom to maintain momentum and alignment. Additionally, we conducted daily supervision meetings with Professor Altay Güvenir to report progress, discuss challenges, and receive guidance.

7.3.2. Helping create a collaborative and inclusive environment

Cultivating a safe and inclusive atmosphere was paramount. Practices included:

- **Brainstorming Workshops**: Held at project start, mid-point, and end-of-sprint retrospectives. Whiteboarding tools facilitated visual mapping of complex ideas.

- **Pair Programming & Peer Reviews**: Pairs collaborated on critical tasks, such as API integrations and model code, fostering knowledge sharing and catching edge-case issues.
- **Communication Channels**: We maintained dedicated channels in Slack for data engineering, modeling, UX, and general discussions. Quick questions were addressed via WhatsApp Meet, while deeper discussions used Zoom.
- **Supervisor Engagement**: Daily Zoom meetings with Professor Altay Güvenir ensured alignment with academic and ethical standards, provided mentorship, and allowed rapid feedback incorporation.

These practices ensured open communication, equal participation, and high team morale.

7.3.3. Taking a lead role and sharing leadership on the team

Leadership was purposefully distributed to leverage expertise:

- **Uygar Aras (General Lead & Backend & Project Management Lead)**: Directed the overall project vision and roadmap, coordinated inter-team dependencies, and maintained close collaboration with Professor Altay Güvenir through daily supervision meetings. Uygar also contributed to sentiment analysis and text chunking processes, assisted with feature selection, and facilitated stakeholder demos and feedback loops. In the early stages of the project conducted exploratory data analysis.
- **Alara Zeybek (Data Engineering & Modeling Support Lead)**: Architected and optimized scalable ETL pipelines ingesting real-time and historical financial data, established automated data validation checks, and configured monitoring dashboards for pipeline health. She implemented advanced feature engineering for semantic and sentiment analysis chunking, and collaborated closely with Emre Akgül to refine model inputs, troubleshoot training workflows on remote servers, and ensure end-to-end data integrity throughout the forecasting lifecycle.

- **Emre Akgül (Modeling & Technical Lead)**: Orchestrated the core technical development of our forecasting engine by spearheading model design, implementation, and optimization. He led training and fine-tuning of tree-, regression-, and ensemble-based models, applied advanced hyperparameter tuning techniques, and benchmarked performance metrics (SMAPE, RMSE, MAE) to select the best-performing approaches. Emre also integrated model interpretability tools for transparent predictions, collaborated with team members to translate business requirements into technical solutions, and oversaw the seamless integration of sentiment analysis pipelines.
- **Gün Taştan (Research & Testing & Quality Assurance Lead)**: Designed and managed comprehensive research protocols to benchmark model performance against financial baselines. He authored detailed test plans and test cases covering unit, integration, regression, and performance testing, implemented automated test suites with CI/CD integration, monitored code coverage metrics, and coordinated manual QA sessions. Gün also conducted data quality assessments, stress-tested API endpoints under simulated load, and documented all findings to ensure reliability and robustness of the final system.
- **Asım Adil Can (Backend & Product Support Lead)**: Architected and developed all backend APIs—including designing the database schema, implementing authentication and authorization modules, optimizing endpoints for performance and scalability, and integrating third-party services—ensuring secure and efficient data transactions. He also assisted with minor modeling tasks alongside Emre Akgül, contributing to end-to-end system cohesion.

This collaborative leadership model accelerated development by enabling parallel progress in specialized domains.

7.3.4. Meeting objectives

The team consistently met milestones through disciplined execution:

- **Data Infrastructure**: Built ETL pipelines ingesting and transforming millions of records within two sprints.

- **Model Development**: Trained and validated multiple forecasting models, achieving <8% SMAPE on backtests.
- **User Interface**: Deployed a beta UI with interactive charts, dashboards, and sentiment indicators as scheduled.
- **Quality Assurance**: Achieved >90% test coverage, automated regression testing in CI/CD, and resolved high-priority defects within 24 hours.
- **Stakeholder Engagement**: Conducted bi-weekly demos, incorporated feedback in subsequent sprints, and delivered full documentation including a user manual and API reference.

This approach ensured deadlines, quality benchmarks, and stakeholder satisfaction.

7.4. New Knowledge Acquired and Applied

Throughout the Plutos Equities project, the team acquired and applied an extensive range of technical and professional skills that directly contributed to our success:

- **Semantic & Sentiment Analysis with Chunking:** We mastered the process of splitting financial documents into coherent text chunks, experimenting with transformer architectures (e.g., BERT, RoBERTa) and fine-tuning sentiment classifiers to achieve high recall and precision on domain-specific corpora. This included data labeling, pipeline optimization, and performance evaluation against benchmark datasets.
- **Tree- and Regression-Based Forecasting Models:** We implemented, tuned, and compared a variety of models—decision trees, random forests, gradient-boosted trees (LightGBM), and linear/multiple regression—to identify the best-performing approaches. Emphasis was placed on hyperparameter optimization, cross-validation, and interpretation of model outputs using feature importance metrics.
- **Advanced Data Preprocessing:** The team developed robust pipelines for handling missing data, detecting and treating outliers, applying normalization and scaling methods, and performing time-series decomposition. Custom feature engineering for financial indicators (e.g., rolling statistics, seasonality adjustments) was also conducted to enrich model inputs.

- **API Integration & Cloud-Native Development:** We integrated multiple real-time and historical data sources via APIs (IEX Cloud, Finnhub, Yahoo Finance), containerized microservices with Docker, and deployed to Azure Kubernetes Service (AKS) for scalable, fault-tolerant operation. Monitoring and logging tools were configured to track service health and performance.
- **Remote Server Management:** For compute-intensive tasks such as model training and hyperparameter sweeps, we provisioned and managed remote servers on cloud virtual machines. This involved configuring SSH access, optimizing resource allocation (CPU/GPU), and automating data transfers to ensure efficient workflows.
- **Version Control & DevOps Practices:** All code and configuration were maintained in GitHub repositories, following a structured branching strategy (feature, develop, main) with pull requests and peer reviews. Continuous integration and deployment pipelines were implemented via GitHub Actions, enabling automated testing, code linting, and container builds on every commit.

Collectively, these newly acquired competencies not only elevated the quality and reliability of Plutos Equities but also enhanced the team's technical versatility, positioning us for future challenges in data-driven product development.

8. Conclusion

In conclusion, Plutos Equities has successfully integrated advanced machine learning techniques and sentiment analysis to provide users with accurate financial predictions and insights. The platform, built using a combination of React for the frontend, FastAPI for the backend, and hosted on Azure Cloud Services, ensures scalability, security, and high performance. By combining structured data such as financial reports and stock prices with unstructured data like earnings calls and news sentiment, the system provides users with a comprehensive tool to make informed financial decisions. With seamless interaction and real-time updates, the platform delivers valuable insights on Earnings Per Share (EPS), Gross Profit, Income Tax Expense/Benefit, Net Income/Loss, Operating Income/Loss, and Weighted Average Number of Shares Outstanding (Basic), stock prices, and other key financial metrics. The backend handles user interactions, payment processing, notifications, and issue tracking securely, while data storage and management are handled efficiently using Azure SQL Database and Azure Storage Accounts.

The system is designed to scale as the user base grows, providing a reliable, secure, and responsive experience for users managing their portfolios and interacting with predictions. As the platform continues to evolve, it will remain adaptable to new data sources, market trends, and user requirements.

9. Future Work

Looking ahead, Plutos Equities has a clear roadmap for enhancing its capabilities and expanding its features. One of the key areas for improvement is the automation of customer support. As the platform grows, addressing user-reported issues through automated systems will be crucial. An AI-powered chatbot is being considered to handle common user queries and issues, providing instant responses and resolutions. This would significantly reduce response times, improve user satisfaction, and minimize the manual workload on support teams.

Another area of focus is further platform scalability. To accommodate increasing user demands, we will expand our infrastructure by upgrading to more powerful compute resources within Azure, as well as implementing additional performance optimizations like caching layers and distributed task queues. We also aim to enhance our machine learning capabilities, refining existing models and experimenting with new techniques to deliver even more precise financial forecasts.

Moreover, payment system improvements will be a priority once regulatory barriers are addressed. Full payment integration with Stripe will enable users to access advanced features based on their subscription levels, which will be expanded to offer a broader set of tools for financial analysis and prediction.

As part of our long-term strategy, we also plan to incorporate personalized investment advice by using AI-driven recommendation systems, enhancing the predictive accuracy of stock forecasts. This would create a more intuitive and value-driven user experience, allowing users to make more tailored and informed investment decisions. Additionally, we aim to further integrate machine learning pipelines that will automatically update forecasts and adapt to market changes, thereby increasing the automation of the platform and providing a cutting-edge service to users.

In summary, future work will focus on enhancing scalability, improving automation with AI-driven tools, expanding the platform's features, and refining our financial prediction models, ensuring that Plutos Equities remains at the forefront of financial forecasting.

10. Glossary

- **EPS (Earnings Per Share)**: A financial metric indicating a company's profitability, calculated as net income divided by outstanding shares.
- **Revenue**: The total income generated by a company before expenses are deducted.
- **Net Income**: The profit of a company after all expenses, taxes, and costs have been deducted from total revenue.
- **Gross Margin**: The percentage of revenue remaining after deducting the cost of goods sold (COGS), indicating profitability.
- **Operating Expenses**: The costs required for running the core business operations, excluding costs related to manufacturing.
- **SEC (Securities and Exchange Commission)**: The U.S. regulatory body overseeing securities markets and protecting investors.
- **Market Sentiment**: The overall attitude of investors toward a particular security or the financial market, often determined through sentiment analysis of financial news.
- **ML (Machine Learning)**: A field of artificial intelligence that enables systems to learn from data and make predictions.
- **LSTM (Long Short-Term Memory)**: A type of recurrent neural network (RNN) designed for time-series forecasting and sequential data.
- **Prophet Model**: A forecasting model developed by Facebook, optimized for handling seasonality and missing data in time-series forecasting.
- **Feature Engineering**: The process of selecting and transforming variables to improve machine learning model performance.

- **Hyperparameter Tuning**: The optimization of model parameters to enhance performance and accuracy.
- **Backtesting**: The process of testing a predictive model using historical data to evaluate its performance.
- **Ensemble Learning**: A technique that combines multiple models to improve prediction accuracy and robustness.
- **API (Application Programming Interface)**: A set of functions that allow applications to communicate and exchange data.
- **REST API**: A web service architecture that allows communication between systems using HTTP requests.
- **OAuth 2.0**: A secure authorization framework that enables third-party applications to access user data without exposing credentials.
- **Cloud Computing**: The practice of using remote servers to store, manage, and process data, rather than local infrastructure.
- **Azure Cloud Services**: Microsoft’s cloud computing platform providing resources for scalable computing, storage, and machine learning workloads.
- **SQL (Structured Query Language)**: A programming language used for managing and querying relational databases.
- **Data Encryption**: The process of converting information into a secure format that prevents unauthorized access.
- **Functional Testing**: A type of software testing that ensures each function of an application operates as expected.
- **Integration Testing**: A testing phase where different software modules are combined and tested as a group.
- **Load Testing**: A performance test to evaluate how a system behaves under expected and peak loads.
- **Penetration Testing**: A security assessment method used to identify vulnerabilities by simulating cyberattacks.
- **GDPR (General Data Protection Regulation)**: European Union regulation ensuring data privacy and protection for individuals.

- **Data Compliance**: Adhering to regulations and laws that govern the use, storage, and sharing of data.
- **Agile Methodology**: An iterative project management approach emphasizing flexibility, collaboration, and customer feedback through regular sprints and adaptation.
- **Sprint**: A fixed-length development cycle, typically one to two weeks, during which specific project tasks are completed and reviewed.
- **Sprint Burndown Chart**: A visual representation of remaining work in a sprint, showing daily progress toward sprint goals.
- **Jira**: A project management and issue-tracking tool by Atlassian used to plan, track, and release software.
- **GitHub Actions**: A CI/CD platform integrated with GitHub repositories for automating build, test, and deployment workflows.
- **Continuous Integration (CI)**: The practice of automatically building and testing code upon commit to detect issues early.
- **Continuous Deployment (CD)**: The practice of automatically deploying validated code changes to production environments.
- **Docker**: A platform for containerizing applications, bundling code, dependencies, and configurations into portable units.
- **Containerization**: The process of packaging an application and its dependencies into isolated containers.
- **Azure Kubernetes Service (AKS)**: A managed Kubernetes service on Azure for deploying, scaling, and managing containerized applications.
- **ETL (Extract, Transform, Load)**: A data processing pipeline that extracts data from sources, transforms it into the desired format, and loads it into storage systems.
- **Message Queue**: An asynchronous communication mechanism that enables decoupled applications to exchange messages.
- **Celery**: A distributed task queue system for scheduling and executing asynchronous tasks in Python.
- **Redis**: An in-memory data store used for caching, session storage, and message brokering.

- **JSON Web Token (JWT)**: A compact, URL-safe token format for securely transmitting claims between parties as a JSON object.
- **Bcrypt**: A password hashing function designed to be computationally intensive to resist brute-force attacks.
- **SMAPE (Symmetric Mean Absolute Percentage Error)**: A measure of prediction accuracy, expressed as a percentage of the average absolute forecast error relative to actual values.
- **RMSE (Root Mean Squared Error)**: A measure of the square root of the average squared differences between predicted and actual values.
- **MAE (Mean Absolute Error)**: A measure of the average absolute difference between predicted and actual values.
- **LOOCV (Leave-One-Out Cross-Validation)**: A cross-validation technique where each data point is used once as the validation set while the rest form the training set.
- **SSH (Secure Shell)**: A cryptographic network protocol for secure remote login and command execution over unsecured networks.
- **Pydantic**: A Python library for data validation and settings management using Python type annotations.
- **FastAPI**: A modern, high-performance web framework for building APIs with Python, leveraging Pydantic and automatic OpenAPI documentation.
- **React**: A JavaScript library for building interactive user interfaces based on reusable components.
- **Tailwind CSS**: A utility-first CSS framework for rapidly building custom user interfaces without writing custom CSS.
- **WebSocket**: A protocol for full-duplex communication channels over a single TCP connection, enabling real-time data exchange.
- **Selenium**: A framework for automating web browser interactions, often used for end-to-end testing of web applications.
- **Locust**: A performance testing tool that simulates user behavior and load on web applications, ideal for load and stress testing.
- **Microservice**: An architectural style where an application is composed of small, loosely coupled services that communicate over lightweight protocols.

11. References

[1] **S. J. Taylor and B. Letham**, “Forecasting at Scale,” The American Statistician, vol. 72, no. 1, pp. 37–45, 2018.

[2] **E. F. Fama**, “Efficient Capital Markets: A Review of Theory and Empirical Work,” The Journal of Finance, vol. 25, no. 2, pp. 383–417, 1970.

[3] **Securities and Exchange Commission**, “EDGAR Database,” [Online]. Available: https://www.sec.gov/edgar.shtml.

[4] **Kaggle**, “Financial Forecasting Datasets and Competitions,” [Online]. Available: https://www.kaggle.com/datasets.

[5] **Microsoft Azure**, “Azure Machine Learning Documentation,” [Online]. Available: https://docs.microsoft.com/azure/machine-learning/.

[6] **J. Brownlee**, Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-to-End. Machine Learning Mastery, 2017.

[7] **S. Hochreiter and J. Schmidhuber**, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] **OpenAI**, “GPT-3: Language Models are Few-Shot Learners,” [Online]. Available: https://arxiv.org/abs/2005.14165, 2020.

[9] **European Union,** General Data Protection Regulation (GDPR), Official Journal of the European Union, 2016.

[10] **ISO/IEC 27001:2013**, Information Security Management Systems – Requirements. International Organization for Standardization, 2013.

[11] **Deloitte**, Trends in AI-Driven Financial Forecasting, 2023. [Online]. Available: https://www2.deloitte.com.

[12] **J. Brownlee**, Machine Learning Mastery with Python. Machine Learning Mastery, 2017.

[13] **Software Testing Help**, “Manual Software Testing: A Beginner’s Guide,” [Online]. Available: https://www.softwaretestinghelp.com/manual-testing-tutorial-1/.

[14] **B. Bruegge and A. H. Dutoit**, Object-Oriented Software Engineering: Using UML, Patterns, and Java, 2nd ed. Prentice Hall, 2004.

[15] **Atlassian**, “Agile Project Management with Jira Software,” [Online]. Available: https://www.atlassian.com/software/jira.

[16] **W. F. Sharpe**, “Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk,” The Journal of Finance, vol. 19, no. 3, pp. 425–442, 1964.

[17] **R. J. Shiller**, Irrational Exuberance, 3rd ed. Princeton University Press, 2014.

[18] **S. Ray**, Data Science and Machine Learning: Mathematical and Statistical Methods. Wiley, 2019.

[19] **G. P. Zhang**, “Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model,” Neurocomputing, vol. 50, pp. 159–175, 2003.

[20] **W. McKinney**, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O’Reilly Media, 2012.

[21] **J. A. Kroll and M. Feldman**, “The Economics of Machine Learning in Financial Markets,” Journal of Financial Economics, vol. 121, no. 2, pp. 123–145, 2016.

[22] **IBM**, “IBM Watson for Financial Services Documentation,” [Online]. Available: https://www.ibm.com/cloud/financial-services.

[23] **Amazon Web Services (AWS)**, “AWS Cloud Architecture Documentation,” [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/framework/.
[24] **Google Cloud**, “Google Cloud Platform: BigQuery ML Documentation,” [Online]. Available: https://cloud.google.com/bigquery-ml/docs.