

**Bilkent University  
Department of Computer Engineering**

**Senior Design Project  
*T2430  
InvestmentHelper-AI***

**Analysis and Requirement Report**

***22102436, Emir Şahin Dilli,*** [***sahin.dilli@ug.bilkent.edu.tr***](mailto:sahin.dilli@ug.bilkent.edu.tr)

***22102512, Hakan Muluk,*** [***hakan.muluk@ug.bilkent.edu.tr***](mailto:hakan.muluk@ug.bilkent.edu.tr)

***22102293, Mehmet Onur Özdemir,*** [***mehmetonur@ug.bilkent.edu.tr***](mailto:mehmetonur@ug.bilkent.edu.tr)

***Özgür Salih Öğüz,*** [***ozgur.oguz@bilkent.edu.tr***](mailto:ozgur.oguz@bilkent.edu.tr)

***Mert Bıçakçı,*** [***mert.bicakci@cs.bilkent.edu.tr***](mailto:mert.bicakci@cs.bilkent.edu.tr)

***Atakan Erdem,*** [***atakan.erdem@cs.bilkent.edu.tr***](mailto:atakan.erdem@cs.bilkent.edu.tr)

**12.14.2024**

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

**Contents**

[1 Introduction 1](#_Toc185217307)

[2 Current System 2](#_Toc185217308)

[3 Proposed System 3](#_Toc185217309)

[3.1 Overview 3](#_Toc185217310)

[3.2 Functional Requirements 4](#_Toc185217311)

[3.3 Non-functional Requirements 5](#_Toc185217312)

[3.3.1 Usability 5](#_Toc185217313)

[3.3.2 Reliability 6](#_Toc185217314)

[3.3.3 Performance 6](#_Toc185217315)

[3.3.4 Supportability 6](#_Toc185217316)

[3.3.5 Scalability 6](#_Toc185217317)

[3.3.5 Security 7](#_Toc185217318)

[3.4 System Models 7](#_Toc185217319)

[3.4.1 Scenarios 7](#_Toc185217320)

[3.4.2 Use-Case Mode 12](#_Toc185217321)

[3.4.3 Object and Class Model 16](#_Toc185217322)

[3.4.4 Dynamic Models 18](#_Toc185217323)

[3.5 User Interface 23](#_Toc185217324)

[4 Other Analysis Elements 28](#_Toc185217325)

[4.1 Consideration of Various Factors in Engineering Design 28](#_Toc185217326)

[4.1.1 Constraints 28](#_Toc185217327)

[4.1.2 Risks and Alternatives 32](#_Toc185217328)

[4.2 Standards 36](#_Toc185217329)

[4.3 Project Plan 36](#_Toc185217330)

[4.4 Ensuring Proper Teamwork 41](#_Toc185217331)

[4.5 Ethics and Professional Responsibilities 41](#_Toc185217332)

[4.6 Planning for New Knowledge and Learning Strategies 42](#_Toc185217333)

[4 Glossary 43](#_Toc185217334)

[5 References 44](#_Toc185217335)

Analysis and Requirement Report

*InvestmentHelper-AI*

# Introduction

Making accurate and reliable financial decisions involves following financial news, analyzing events that lead to appreciation or devaluation of stocks, and taking the initiative before the market fully adjusts to an event. However, achieving all these prerequisites in the current settings is not easy.

Manually monitoring and analyzing financial news is challenging, as it requires sustained attention, while modern-day environments often divide one's attention across numerous tasks throughout the day. Yet, the financial markets adjust rapidly, and to gain an advantage regarding personal portfolios, one needs live, real-time knowledge of events that can affect the markets.

Our project aims to provide users with a financial assistant that monitors financial news, analyzes events, and delivers real-time notifications about events that users want to be informed about. The assistant will make it easy for users to follow financial news and gain live, instant knowledge of market-moving events. The assistant, or more formally, the chatbot, will interact with users and support conversational information exchange, referring to financial news stored within the knowledge base. Additionally, users will be able to query the context of important events, upon which relevant notifications will be sent in real-time. Financial news will be analyzed continuously for accurate and up-to-date analysis. Furthermore, AI-driven analysis will be integrated to automatically detect trading patterns, such as head-and-shoulders patterns, double tops, and more, providing advanced insights to help investors make informed decisions based on chart pattern recognition. Additionally, AI-based stock predictions will offer users data-driven forecasts, enhancing their ability to anticipate market movements and make proactive investment decisions. The system will also compare the latest trends of a stock with its past trends, taking into account historical news and the financial environment to identify the closest matching periods from the past. This feature provides users with a comprehensive perspective on historical market behavior, helping them gain a clearer understanding of potential future trends.

# Current System

The market for AI-driven financial tools and chatbot solutions is highly dynamic, with many different platforms offering specific functionalities but often falling short in some critical areas. For instance, FinChat [1] provides notifications; still, the extent and details of this feature are not clear. Furthermore, FinChat supports financial charts so users can do technical analyses comparing stocks—efficiently visualizing market trends. Along with this feature comes the chatbot, which can discuss financial information with users and help in basic question management. Still, even with this useful attribute, the platform does not support more advanced tasks in its chatbot, like multi-hop reasoning or comparative question capabilities. This limitation prevents FinChat's ability to synthesize and cross-reference data points in different documents or contexts smoothly, thus restricting users' ability to make deeper, interconnected analyses and make more nuanced, data-driven decisions.

Similarly, AI Ticker Chat [2] has a unique focus on financial analysis while rendering insights derived from financial reports; users can input any financial report (including their business reports) and can chat with the chatbot on them. However, it does not provide notifications, which is doubtless an important feature that most, if not all, users depend on to get on-time alerts and updates; otherwise, little purpose would be served. It has a narrow focus, which, while favoring in-depth report-related insight, constrains its overall value for broader, ongoing financial analyses and engagement with users.

Uptrends.ai [3] is set apart by its notification system, which provides immediate updates and alerts. It is, however, highly discretized in its contextual analysis—users select among the available notification topics predefined in the system, users cannot enter their own queries—and does not have the fluid, continuous language processing capabilities of more advanced systems. This limit Uptrends.ai from providing cohesive narratives or analyses that span across various data points for a more comprehensive financial insight.

The current financial platforms do not include sufficient data about companies listed on Borsa Istanbul or represent them adequately. This gap creates an opportunity for a platform that provides localized and specialized financial insights tailored to these companies. Our system addresses this gap by offering language-driven, comprehensive, and context-rich coverage of Borsa Istanbul companies. This approach enables continuous and detailed analysis, meeting users' needs and seeking in-depth insights about these companies. Our system is uniquely positioned in the market by specializing in Borsa Istanbul. It fulfills a need that existing platforms fail to satisfy, providing a competitive advantage and filling a critical void in the financial intelligence space.

**Note:** Market Analysis is done based on the prioritized features since the remaining ones are not mandatory for this project.

# Proposed System

## Overview

InvestmentHelper-AI is a system that assists users regarding financial decisions. The project's primary focus is a chatbot that answers questions about finance, whether general or specific. It will also support real-time, live notifications about events that the user enters. For example, if the user provides a prompt such as "Tell me when X happens in BIST," the system will continuously scrape the web for financial news. Once an event related to the user's notification query is detected, the user will be notified about the event, with a summary of the related news. So, the user will have the time to act before most of the other BIST investors realize that the event and the stock prices are affected by people's reflections. Also, the prompt that the user can enter is not heavily constrained, as it is in natural language, making it possible to capture continuous contexts about users' requests.

InvestmentHelper-AI is a time-dependent system; more formally, it is a causal system. Financial information within the knowledge base might become less relevant with time when new information supersedes the effects of the previous one. However, old information is kept when new data arrives. Instead, it is stored, enabling the system to answer user queries about past events. So, users can access up-to-date and historical information by adjusting their prompts. By maintaining historical data, users can analyze trends over time and can analyze past events and their influences.

The assistant, the chatbot, will utilize Advanced Retrieval Augmented Generation (RAG) pipelines for correct behavior. We propose advanced RAG techniques because more than basic RAG approaches are needed for this project. Retrieval is challenging due to numerous highly similar news articles; sometimes, new articles with the same company but are released on different dates also create inconvenience. This similarity makes it difficult to retrieve the correct information using naive RAG methods (information retrieved might be from a different period compared to the user's intent or might be from a different company). Therefore, we are building a system that can retrieve the necessary news based on time, context, and user intent to satisfy users' needs.

The system supports distinct types of user questions, including multi-hop, comparative, and generic questions, each presenting unique retrieval and generation challenges. Multi-hop questions, such as "Which companies that work with X company also have business in the energy sector?" require combining information from multiple documents. It is called multi-hop because the texts usually contain information such as: "Company A is working with Company X" and "Company A has a business in Y sector." Different texts should be combined to generate an answer to multi-hop questions. Basic retrieval methods, which typically retrieve information based on the cosine similarity of the document text chunk and user query embeddings, need help with such requirements. The system must identify relevant information from different documents, synthesize it, and respond coherently and accurately to these questions.

Comparative questions like "What are the top 5 companies with the highest profits in the last year?" involve retrieving data from multiple sources and performing computations (e.g., sorting). Basic retrieval methods that return individual chunks of text are insufficient because they cannot process, compare, and compute across different pieces of information.

Generic questions, such as "How is X company doing?" require a higher-level understanding of the overall context of multiple documents. More than isolated text chunks must be retrieved using rudimentary methods to answer these questions. Instead, the system must analyze the broader context and synthesize a summary that captures the overall state of the subject.

Additionally, users can access financial charts to perform technical analysis of stock prices, compare different stocks, and analyze market trends using visualization tools. This feature enhances their ability to make informed decisions by providing comprehensive charting and comparison capabilities. AI-driven analysis will also be integrated to automatically detect trading patterns, such as head-and-shoulders patterns, double tops, and more, providing advanced insights to help investors make informed decisions based on chart pattern recognition. AI-based stock predictions will offer data-driven forecasts, improving users' ability to anticipate market movements and make proactive investment decisions. Finally, the system will compare the latest trends of a stock with its past trends, taking into account historical news and the financial environment to identify similar periods from the past, thereby offering users a comprehensive perspective on historical market behavior to understand future trends better.

**Note:** Our prioritized features are the chatbot, notification, and chart displaying features. Other features mentioned are the optional ones; this has also been discussed with the course instructors and approved by them.

## Functional Requirements

* Ask questions to the chatbot about general topics/themes about Borsa Istanbul.
* Ask about specific companies by asking the chatbot for information (e.g., "What is the total production capacity of Company X").
* Receive up-to-date financial information via chatbot unless they request past information.
* Ask complex multi-hop questions that require information from multiple documents.
* Ask comparative questions that require information from multiple documents and require computations on them.
* Set up notifications by entering prompts in natural language, specifying the financial events they are interested in (e.g., "Notify me when Company Y's stock price drops below $50").
* Receive real-time notifications when important financial events occur that match their notification queries, along with the summary of the financial event.
* Visualize stock data using different types of charts.
* Perform technical analysis by applying indicators and drawing tools on stock charts.
* Save past chat interactions with the chatbot.
* Create new chats that are independent of others.
* Customize notification preferences by setting the notification method as mail or SMS.
* Export data and charts in CSV or PNG.
* Set up multi-factor authentication for account security.
* Enable dark mode.
* Define their own indicators for technical analysis on charts.
* Ask the chatbot to search for a specific time (e.g., "Show me the stock performance of Company X in Q1 2023").
* Ask follow-up questions after a conversation (e.g., "What about its competitors?" after asking about a company).
* Create notifications based on multiple conditions (e.g., "Notify me when Company Y’s stock drops below $50 and its trading volume exceeds 1 million").
* Compare multiple stocks in a single chart.
* Export chat summaries with PDF format.
* Login to system using his/her mail address and password.
* Register to system using his/her mail address.
* Change his/her password.
* The below ones are optional functional requirements based on the discussions we had with the course instructors:
* Analyze detected trading patterns such as head-and-shoulders patterns and double tops identified by the AI on stock charts.
* Access stock predictions generated by machine learning models based on past stock prices and financial news.
* Analyze the time intervals like today’s conditions in terms of the financial news and stock price chart similarity, which are found by AI.
* Provide a real-time news feed related to Borsa Istanbul, including updates on companies, market news, and economic indicators.

## Non-functional Requirements

### 3.3.1 Usability

* The system should require no more than 10 minutes of user training for basic operations.
* The system should enable the chatbot to provide detailed responses to follow-up queries without requiring the user to rephrase the query.
* The system should ensure all features are accessible within three clicks from the main interface.
* The system should support both dark and light themes to accommodate various working environments.
* The system should implement keyboard shortcuts for common operations to enhance user efficiency.
* The system should maintain consistent navigation patterns across all screens.
* The system should provide clear, concise, and actionable error messages to minimize user confusion.
* The system’s front-end should offer real-time feedback for user actions with a maximum response time of 0.5 seconds.
* The system should support English and Turkish languages, with seamless switching functionality.
* The system should allow text scaling up to 200% without loss of functionality.
* The system should include tooltips for new users.

### 3.3.2 Reliability

* The system should conduct planned maintenance outside trading hours of Borsa Istanbul to minimize disruptions.
* The system should automatically switch to a recovery system when a failure occurs, and the systems should be restarted.
* The system should maintain 99% accuracy in financial calculations.
* The system should automate data backups every 6 hours to prevent data loss.
* The system should meet recovery objectives:
  + RPO (Recovery Point Objective): Maximum data loss of 6 hours.
  + RTO (Recovery Time Objective): Recovery within 15 minutes.
* The system should rely on trusted data sources such as KAP to give reliable answers to questions.
* The system should implement fallbacks for external APIs (e.g., Yahoo Finance API, KAP website etc.), including retry mechanisms and alternative data sources, to ensure data availability during downtimes.

### 3.3.3 Performance

* The system should respond to single-hop queries within 10 seconds.
* The system should resolve complex, multi-hop queries within 30 seconds.
* The system should ensure a maximum latency of 100 milliseconds for real-time updates in stock prices.
* The system should be able to perform 10,000 web scraping operations per hour for data updates.
* The system should support simultaneous notification checks for different news.

### 3.3.4 Supportability

* The system should perform automated health checks every 5 minutes.
* The system should include automated monitoring of ML model (for stock price prediction) performance, detecting significant accuracy drops and notify the developers.
* The system should use automated alerts to notify the support team when a system crash occurs.
* The system should include detailed error logs with timestamps.

### 3.3.5 Scalability

* The system should initially satisfy the needs of BIST investors, with the capability to integrate other financial markets in the future.
* The system should support adding new languages by using an internal English-based mechanism for translations.
* The system should support manual resizing of cloud instances or VMs for CPU or GPUs in RunPod and memory.
* The system should integrate caching mechanisms, especially for trend identification parts to prevent using the ML model again and again.
* The system should implement indexing (such as vector indexing in Neo4j for efficient cosine similarity search over embeddings) and query optimization for efficient execution.
* The system should be distended in a way so that the LLMs used can be changed without any effort.
* The system should be able to be adapted for using different financial news sources.

### 3.3.5 Security

* The system should encrypt sensitive data during transmission using TLS 1.3.
* The system should store user passwords securely using hashing (e.g., bcrypt).
* The system should implement role-based access control (RBAC) for different user types.
* The system should use email or OTP verification for user authentication.
* The system should validate and sanitize user queries against any prompt injection attack.
* The system should restrict LLM responses to predefined contexts (financial topics), avoiding execution of user-provided commands related to other topics.
* The system should enforce automatic session expiration to prevent unauthorized access.
* The system should use firewalls to block unauthorized access.
* The system should implement rate limiting to prevent brute-force attacks.

## System Models

### Scenarios

**Scenario 1:** User Asks a Question to the Chatbot

**Actors:** User  
**Entry Conditions:** The user has logged in and opens the chatbot interface.  
**Exit Conditions:** The chatbot provides a complete response to the user’s query.

**Main Flow of Events:**

1. The user opens the chatbot interface.
2. The user types a question, such as, “How is Company X performing this year?”
3. The system re-writes the query if the query is a follow-up query.
4. The type of the query is determined (single hop, multi hop etc.)
5. Answer is generated by considering the type of the query.
6. The chatbot displays the answer and the documents used while generating the answer as references.

**Scenario 2:** User Sets a Notification

**Actors:** User  
**Entry Conditions:** The user has logged in and navigates to the notification settings interface.  
**Exit Conditions:** The user’s notification preference is saved in the system.  
**Main Flow of Events:**

1. The user opens the notification settings page.
2. The user types a natural language prompt specifying the event of interest (e.g., “Notify me when Company Y’s stock price drops below $50”) and selects the notification type as email or SMS.
3. The system analyzes the prompt and verifies that the notification query can be tracked.
4. The system stores the notification query in the database.
5. The system confirms to the user that the notification query has been successfully set.

**Scenario 3:** System Notifies the User

**Actors:** User  
**Entry Conditions:** The user has set a notification before, and a financial event matching the notification occurs.  
**Exit Conditions:** The user receives a notification containing a summary of the event.  
**Main Flow of Events:**

1. A financial event that matches one of the user’s stored notification conditions is detected by the system (e.g., Company Y’s stock price falls below $50).
2. The system generates a notification message summarizing the event or related financial news.
3. The system sends the notification to the user via the email.
4. The user receives and reviews the notification.

**Scenario 4:** User Analyzes Charts by Comparing Different Companies’ Stock Prices

**Actors:** User  
**Entry Conditions:** The user has logged in and has navigated to the charting page.

**Exit Conditions:** The user views a comparative chart with selected companies’ stock data  
**Main Flow of Events:**

1. The user selects multiple companies and time intervals to compare the companies.
2. The system retrieves the corresponding stock data for each selected company.
3. The system displays a comparative chart showing the selected companies’ stock prices over the chosen period.

**Scenario 5:** User Selects a Part of the Graph and Requests Pattern Recognition

**Actors:** User

**Entry Conditions:** The user has logged in, is viewing a stock chart, and has identified a segment of interest.  
**Exit Conditions:** The system identifies and displays any detected trading patterns within the selected graph segment, with the likelihoods(in percents) of different patterns.  
**Main Flow of Events:**

1. The user selects a specific time period of the displayed chart (e.g., a recent price movement segment).
2. The user requests the system to run pattern recognition on the selected chart period.
3. The system applies a ML model to detect known trading patterns within that period.
4. The system displays the detected pattern(s), along with the corresponding scores(likelihoods) of each pattern.
5. The user analyzes the patterns and scores.

**Scenario 6:** User Selects a Time Interval for a Company and Analyzes Similar Past Periods

**Actors:** User  
**Entry Conditions:** The user has logged in, has accessed the historical analysis page, and selected a company.  
**Exit Conditions:** The user views identified historical time intervals that are similar to the selected period in terms of financial environment and graph patterns.  
**Main Flow of Events:**

1. The user selects a company and selects a time interval for analysis (e.g., Q1 2023).
2. The system retrieves both financial news data and historical graph data corresponding to the specified period.
3. The system analyzes historical data to identify past periods with similar financial environments (e.g., news sentiment, events) and similar graph patterns.
4. The system presents these identified historical intervals, by displaying the chart similarity and financial environment similarity scores.
5. The user analyzes the detected intervals and scores.

**Scenario 7:** User Changes Password

**Actors:** User  
**Entry Conditions:** The user is logged in and navigates to the account settings page.  
**Exit Conditions:** The user’s password is successfully updated in the system.

**Main Flow of Events:**

1. The user goes to the account settings menu and selects “Change Password.”
2. The user enters their current password and a new desired password (following the system’s password complexity rules).
3. The system verifies the current password and checks the new password’s validity.
4. The system updates the password in the database.
5. The system confirms to the user that their password has been successfully changed and sends notification.

**Scenario 8:** User Defines a Custom Technical Indicator

**Actors:** User  
**Entry Conditions:** The user is logged in and viewing the chart settings page.  
**Exit Conditions:** The system displays the chart with the newly defined custom indicator.  
**Main Flow of Events:**

1. The user navigates to the “Custom Indicators” section.
2. The user inputs a custom formula or parameters for a technical indicator (e.g., a custom moving average calculation or RSI variant).
3. The system validates the formula and calculates the indicator values.
4. The chart is updated to display the user-defined custom indicator.

**Scenario 9:** User Applies Technical Analysis Tools on a Chart

**Actors:** User  
**Entry Conditions:** The user is logged in and viewing a company’s stock chart page.  
**Exit Conditions:** The user sees the chart updated with technical indicators and custom drawing tools.  
**Main Flow of Events:**

1. The user selects a company’s stock to analyze and chooses a time interval on the chart.
2. The user opens the “Technical Indicators” menu and applies indicators (e.g., MACD, Bollinger Bands) to the chart.
3. The system retrieves and calculates the necessary data, displaying the chosen indicators on the chart.
4. The user selects a drawing tool (e.g., trend line, Fibonacci retracement) to highlight certain price patterns.
5. The user draws directly on the chart, and the system updates the chart to reflect the new annotations.

**Scenario 10:** User Exports a Customized Chart

**Actors:** User  
**Entry Conditions:** The user is logged in and has a chart open with custom indicators or drawings applied.  
**Exit Conditions:** The user obtains a downloaded image file (e.g., PNG) of the chart.

**Main Flow of Events:**

1. The user has previously selected companies, time intervals, and applied one or more custom indicators or drawing tools on the chart.
2. The user clicks the “Export” button.
3. The system renders the current state of the chart (including indicators, drawings, and selected timeframe) into a PNG file.
4. The system prompts the user to download the exported file.
5. The user saves the file locally.

**Scenario 11:** User Registers for an Account

**Actors**: User  
**Entry Conditions**: The user is on the registration page.  
**Exit Conditions**: The user successfully creates an account and receives a confirmation message.

**Main Flow of Events:**

1. The user clicks the "Register" button and fills in the required fields: name, email, and password.
2. The system validates the input.
3. The system sends a verification email.
4. The system confirms the registration and provides access to the system.

**Scenario 12:** User Recovers Password

**Actors**: User  
**Entry Condition**: The user is on the login page and clicks "Forgot Password."  
**Exit Condition**: The user successfully resets their password and gains access to the system.

**Main Flow**:

1. The user clicks "Forgot Password."
2. The system prompts the user to enter their registered email address.
3. The system sends a verification email with a temporary reset code.
4. The user clicks the link or enters the code to verify their identity.
5. The system prompts the user to create a new password.
6. The user submits the new password, and the system confirms the change.

**Note:** The report focuses only on the most important scenarios to ensure clarity and prevent including too much information, which could overwhelm the reader

### Use-Case Mode

#### 3.4.2.1 Use-Case Model for Authentication

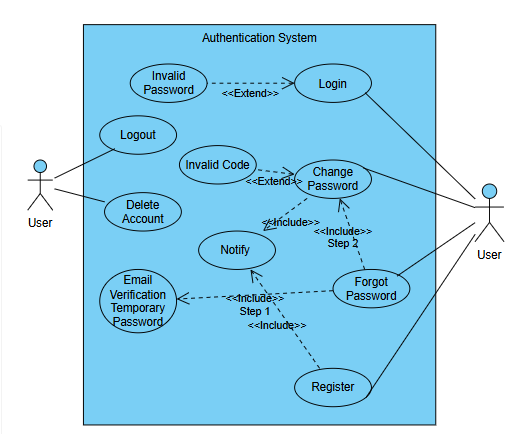


Figure 1 Use-Case Model Of Authentication

Users can log in, and if the password is incorrect, the system informs them of the issue. They can also log out and delete their accounts when needed. For password recovery, users can initiate a forgotten password process, which involves receiving a notification, verifying their identity via email, and using a temporary password. The system also handles cases where invalid codes are entered during the recovery process. Users can change their passwords, with the system notifying them upon completion. Registration is available for new users to create accounts, ensuring secure access to the system. Notifications play a central role in confirming actions and facilitating recovery processes.

#### 3.4.1.2 Use-Case Model for Chatbot

Figure 2 Use-Case Model for Chatbot

Users can ask questions of different types, including single-hop, general, multi-hop, and comparative questions, all of which rely on referencing external sources for answers. A follow-up question feature allows users to refine or extend their queries based on previous interactions. Users can create new chats, view past conversations, and delete old chats to manage their interaction history. Additionally, users can export chat summaries for external use or further reference. The system incorporates filtering options for reference sources, enabling more precise responses tailored to the user’s needs.

#### 3.4.2.3 Use-Case Model for Notification System

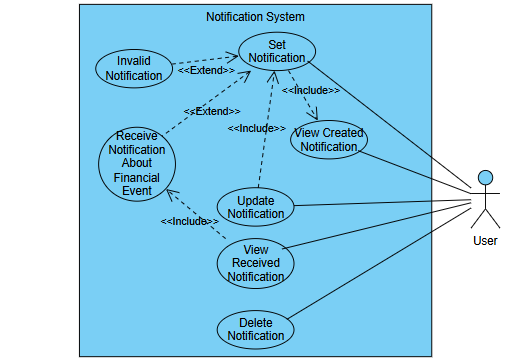


Figure 3 Use-Case Diagram for Notification System

Users can set notifications by specifying criteria, with the option to view the notifications they have created. If the input criteria for a notification are invalid, the system handles this scenario. Notifications about financial events are received when the set conditions are met. Users can update existing notifications or delete them if no longer needed. They can also view received notifications to stay informed about relevant events.

#### 3.4.2.5 Use-Case Model for Stock System

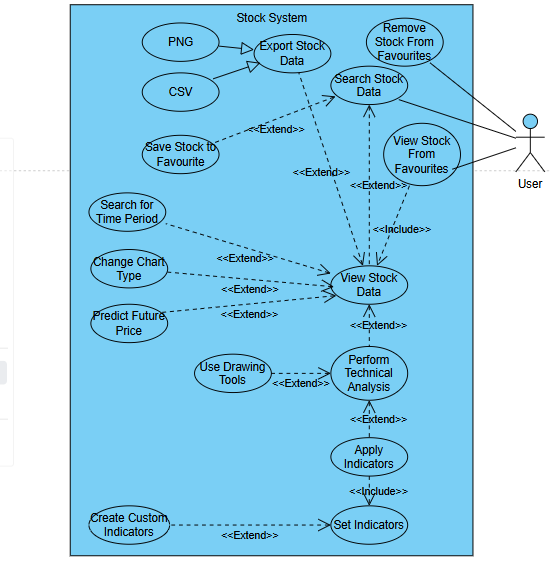


Figure 4 Use-Case Model for Stock System

Users can search for stock data or access stocks directly from their favorites. The system supports saving stocks to favorites and removing them when needed. Users can view stock data with options to refine their view, such as searching for specific time periods, changing chart types, or predicting future prices.

Advanced features include performing technical analysis by applying indicators or using drawing tools to interpret stock trends. Users can create custom indicators to tailor their analysis. The system also enables exporting stock data in formats like PNG and CSV for offline use.

### Object and Class Model

**Note**: The class diagrams are separated into various parts for better visual understanding.

#### 3.4.3.1 Class Diagram of UI

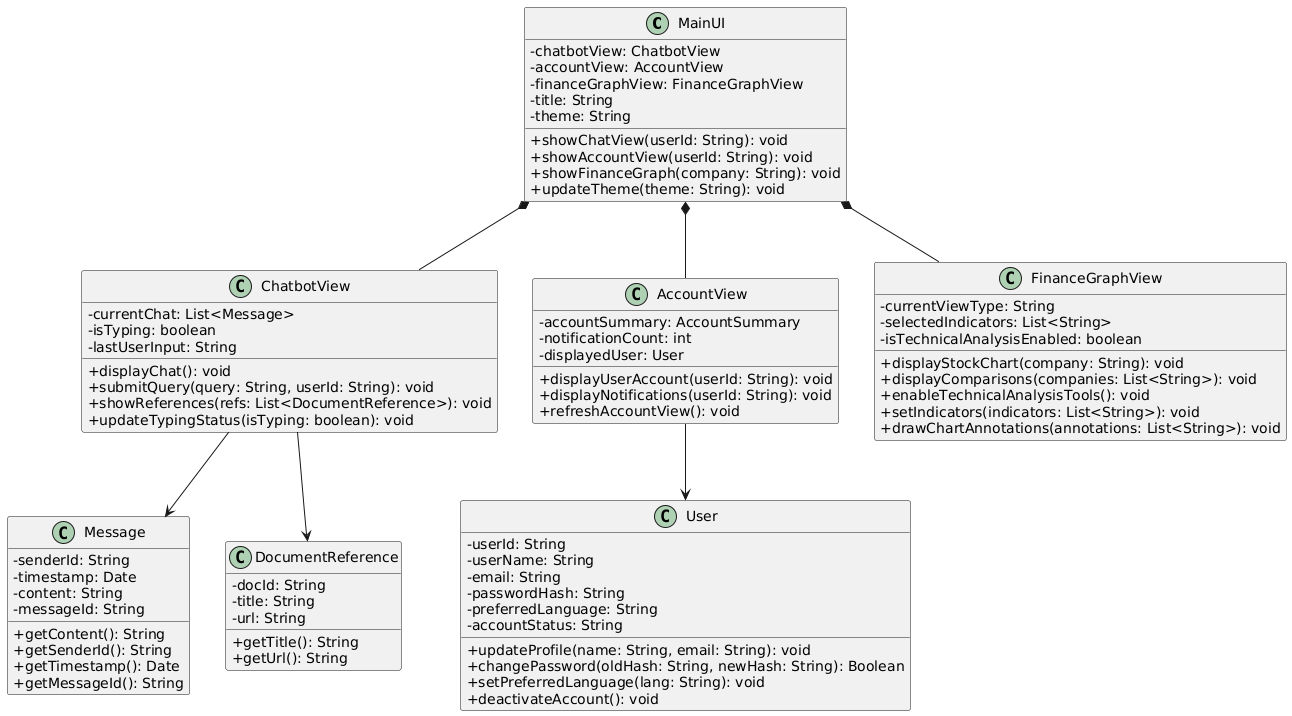


Figure 5 Class Diagram of UI

This diagram represents the main user interface components of the system and their relationships. The Main UI class manages three views: ChatbotView, AccountView, and FinanceGraphView, which are connected to it via composition. Each view serves a specific purpose: ChatbotView handles displaying chat messages and document references, AccountView displays user account details and notifications, and FinanceGraphView manages stock charts, comparisons, and technical analysis tools. The Message and DocumentReference classes store the details of messages and related documents displayed in the chat. Additionally, the User class contains user information such as ID, name, email, and preferences, and it is referenced by AccountView class to personalize the account details shown.

#### 3.4.3.2 Class Diagram of RAG

metin, diyagram, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6 Class Diagram of RAG

The RAGController manages the overall process of handling user queries by coordinating key components. It uses the LLMEngine to generate answers, PromptChecker to validate user queries for safety, Text Retriever to fetch relevant information, Conversation Context to integrate past chat data, and Translation Service to manage language translation needs. The LLMEngine interacts with the LLMModel to process prompts and generate results. Additionally, the Conversation Context accesses Message objects to manage chat history.

#### 3.4.3.3 Class Diagram of User Services

metin, diyagram, yazı tipi, plan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7 Class Diagram of User Services

The User Service manages user-related functionalities such as authentication, managing user accounts, and retrieving user chats. The NotificationService manages user-defined notification queries, using the NotificationQueryManager to organize and optimize queries, and the NewsMatcher to match relevant news items with these queries. The UserAccount class stores details specific to a user, like their balance, notification queries, and last login time. The NotificationQuery class represents the rules users define for notifications and extends the Message class, which stores shared properties such as content, timestamps, and sender details.

#### 3.4.3.4 Class Diagram of Finance Services

**metin, diyagram, çizgi, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Figure 8 Class Diagram of Finance Services

The FinanceGraphView class is responsible for displaying stock charts, comparing companies, enabling technical analysis tools, and managing chart annotations. It interacts with the FinanceAnalysisService, which performs tasks like identifying patterns in stock data, predicting future prices, and finding similar historical time intervals. The FinanceAnalysisService relies on the MLModel class to process stock data, which is represented by the StockData class. The service produces results such as Pattern, PredictedPrice, and SimilarIntervals, which are returned to enhance the analysis. Supporting classes like FeatureSet, PredictionResult, and TimeInterval handle features, predictions, and date ranges.

### Dynamic Models

#### 3.4.4.1 Activity Diagram of Chatbot

metin, ekran görüntüsü, diyagram, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 9 Activity Diagram of Chatbot

This activity diagram shows handling user queries in the chatbot sub-system. First, the user's query is translated into English, as the system processes inputs in English to achieve the best performance. Once translated, the query undergoes a check for prompt injection attacks to avoid security leaks. If a prompt injection is detected, the system warns the user and terminates the process. If no injection is detected, the system continues processing.

Next, the system determines if the query is a follow-up to a previous query. If it is a follow-up query, the system rephrases it to include relevant context from earlier messages. If it is not a follow-up query, the system proceeds to the query type classification stage, such as single-hop, multi-hop, or comparative. This classification ensures that the appropriate retrieval strategy is applied to fetch relevant data.

Based on the query type, the system retrieves information from the knowledge base. The retrieved data and the user query are then sent to the large language model (LLM) for response generation. If the LLM successfully generates an answer, it is translated back into the user's preferred language. The translated response is then displayed to the user, and the documents used to generate the answer are references.

If the LLM fails to generate an answer, the system retries by rewriting the query differently, up to two times, to retrieve other texts and try to generate an answer based on the different texts. If the retry limit is exceeded without success, the process terminates with a "no answer found" response.

#### 3.4.4.2 Single-Hop Retrieval Activity Diagram

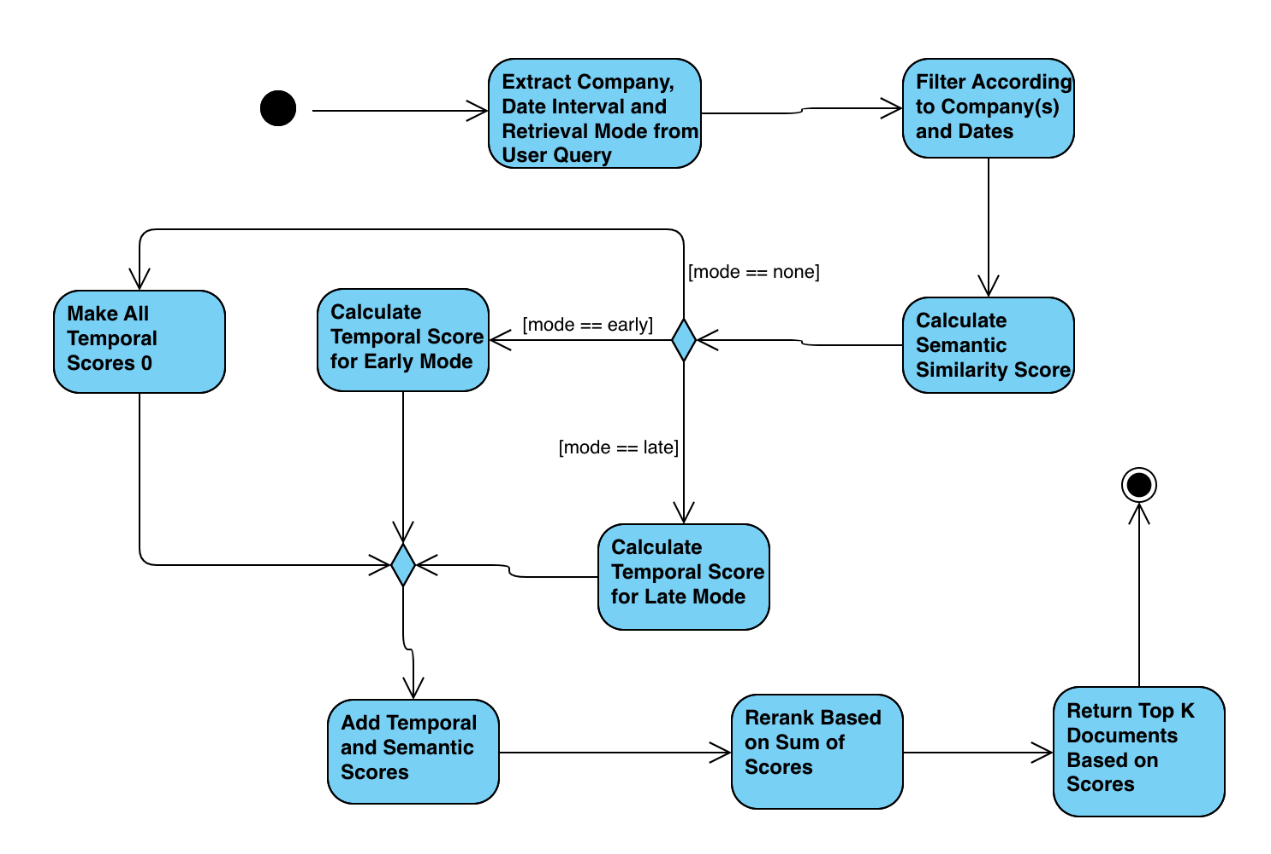
****

Figure 10 Activity Diagram of Single-Hop Retrieval

The single-hop retrieval activity diagram represents identifying and ranking relevant documents based on user queries involving a single-hop relationship. This process combines semantic and temporal relevance to ensure that the most relevant documents are retrieved and presented to the user.

The process begins by extracting key elements from the user's query, including the target company or companies, the specified date interval, and the retrieval mode. The retrieval mode defines whether the system prioritizes older, newer, or all dates equally when ranking documents. After extracting this information, the system filters the dataset to include only those documents associated with the specified companies and within the defined date range. This filtering removes irrelevant data early, reducing computational overhead in subsequent steps.

Next, the system calculates a semantic similarity score for each document. This score is computed using the cosine similarity between the user query and the document's triplet and contextual embeddings. The semantic score is calculated using the formula:

This formula ensures that the document's content and its contextual alignment with the user query are considered.

The system then evaluates the retrieval mode. If the mode is set too early, the system prioritizes older documents relative to the query date. For late mode, newer documents are prioritized. The temporal scoring is calculated using the formula,

Here, stands for the date specified in the query and dt specifies the date of the document. is the mean of pre temporal scores across the filtered documents(mean of τ(qt, dt)pre), is the standard deviation of pre temporal scores, is the standard deviation across the semantic scores calculated and is the mean of semantic scores. is calculated by normalizing the pre temporal scores and re-calibrating it to be compatible with the semantic scores. Finally, the retrieval score is calculated for each filtered document using the formula,

This state diagram outlines the lifecycle of a notification query within the system, demonstrating how the system processes and monitors user-defined notification conditions. The process begins with an initial state, where no notification query has been entered. At this stage, the system awaits user input to proceed.

When a user enters a notification query, the system transitions to the Notification Query Entered state. Here, the system validates the entered query to determine if it is trackable. If the query cannot be tracked—due to reasons such as an unsupported format or incomplete input—the system terminates the process, reverting to the initial state. However, if the query is valid and trackable, the system transitions to the Notification Query is Set state. In the Notification Query is Set state, the system actively monitors relevant data sources(such as KAP) to detect events that match the specified conditions in the query. If the event has not yet occurred, the system remains in this state, continuously tracking data in real-time. Once an event satisfying the conditions of the query is detected, the system transitions to the Notification Query is Deleted state. Finally, in the Notification Query is Deleted state, the system notifies the user of the event and removes the query from active monitoring. This marks the end of the life cycle for that particular notification query.

This diagram illustrates the structured flow of the notification system, highlighting its key features such as query validation, continuous monitoring, and event-driven transitions. By ensuring that only valid queries are monitored and automatically removing them after notification, the system achieves efficiency, reliability, and responsiveness in handling user-defined notifications.

#### 3.4.4.3 Notification State Diagram

**metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Figure 11 State Diagram of Notifications

This state diagram outlines the lifecycle of a notification query within the system, demonstrating how the system processes and monitors user-defined notification conditions. The process begins with an initial state, where no notification query has been entered. At this stage, the system awaits user input to proceed.

When a user enters a notification query, the system transitions to the Notification Query Entered state. Here, the system validates the entered query to determine if it is trackable. If the query cannot be tracked—due to reasons such as an unsupported format or incomplete input—the system terminates the process, reverting to the initial state. However, if the query is valid and trackable, the system transitions to the Notification Query is Set state. In the Notification Query is Set state, the system actively monitors relevant data sources (such as KAP) to detect events that match the specified conditions in the query. If the event has not yet occurred, the system remains in this state, continuously tracking data in real-time. Once an event satisfying the conditions of the query is detected, the system transitions to the Notification Query is Deleted state. Finally, in the Notification Query is Deleted state, the system notifies the user of the event and removes the query from active monitoring. This marks the end of the life cycle for that particular notification query.

This diagram illustrates the structured flow of the notification system, highlighting its key features such as query validation, continuous monitoring, and event-driven transitions. By ensuring that only valid queries are monitored and automatically removing them after notification, the system achieves efficiency, reliability, and responsiveness in handling user-defined notifications.

#### 3.4.4.4 Data Preparation Sequence Diagram

metin, diyagram, paralel, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure Sequence Diagram of Data,

The Investment Helper system fetches HTML content from a web page and translates each HTML element using GPT-mini. The translated text is combined and converted into markdown format, where the NLP Sentence Transformer processes the markdown tables and groups rows into similarity-based chunks. For each chunk, GPT-mini generates summaries and extracts structured tuples, which are then saved into the Neo4j database.

**Note:** Only the most important algorithms’ state, activity, or sequence diagram is explained

## 3.5 User Interface

**Note**: Some of the images are taken from dark mode to present both color palette and these are not final versions of the pages hence there could be some inconsistencies.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 13 Login Page

Login page allows users to access their accounts. Users can sign in using their Google, Facebook, or Apple credentials. Alternatively, they can log in by entering their email address or username along with their password. A "Forgot your password" link is provided for password recovery. There is also a "Remember me" option to keep users signed in. For those without an account, a "Sign up" link is available to create a new account.

metin, diyagram, çizgi, origami içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 14 Register Page

Users can sign up by entering their first and last name, email address, and password. There’s a field to confirm the password, with instructions to use at least 8 characters, including a mix of letters, numbers, and symbols. A “Show password” option is provided to reveal the entered password. For users who already have an account, there’s a “Log in” link to access the login page. A button labeled “Create an account” is available to complete the registration process.



metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 15 Stock Screener

This is a stock screener page displaying key financial data for major companies. It includes columns for company rank, name, stock price, daily percentage change (1D %), one-month percentage change (1M %), year-to-date performance (YTD %), and market capitalization (M Cap). Companies like Apple, Microsoft, and NVIDIA are shown at the top, with Apple leading at a market cap of $3.22 trillion. The data highlights stock performance trends with positive changes in green and negative changes in red. There is also an option to add companies to a watchlist and search.

metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 16 Stock Analysis Page

Stock analysis page, designed to display financial data and trends. However, additional chart types or features have not been added yet. The left sidebar includes options such as Indicators to add technical tools, Tools for analysis, Watchlist to manage selected stocks, and Alerts to set notifications for specific conditions. The top navbar provides navigation to key sections like Markets, which takes users to the screener page for clearer stock overviews, Chatbot for AI assistance, and Settings to adjust user preferences. This page serves as a base for monitoring and analyzing financial data efficiently, with room for future enhancements. Tools and indices are not functional yet.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 17 Chatbot

Chatbot page allows users to create new chats and ask questions to the chatbot. The interface is clear and user-friendly, ensuring a seamless experience. Users can upload documents, and the chatbot will process and answer questions based on the content. The design makes it easy to navigate and interact, helping users quickly navigate through the page.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 18 Notification Page

Our notifications system allows users to set alerts when specific conditions are met. Currently, we are deciding whether users should create these notifications by typing their conditions directly into the chatbot or through a separate input section dedicated to defining alerts. Once the conditions are met, users will automatically receive notifications, ensuring they stay informed about relevant updates seamlessly. This approach provides flexibility while enhancing user experience.

# Other Analysis Elements

## Consideration of Various Factors in Engineering Design

### Constraints

#### 4.1.1.1 Implementation Constraints

This project will use the following LLMs: GPT4o-mini, Llama 3.1-70b, and Llama 3.2-1b. Except for the GPT4o-mini, these models will be deployed on four NVIDIA A40 GPUs provisioned via Runpod. The LangGraph framework will build a substantial state space in the RAG system to enhance Chatbot's capability to manage complex queries.

Implementation will be done in Python and JavaScript, following the OOP principles for modularity and maintainability. The project platform is web-based. Thus, it will be accessible through web browsers. FastAPI will be used on the back end. React will be used on the front end except for the Chatbot and financial chart components. Chainlit will be used to interface with the chatbot. Lightweight Charts will be used to generate financial graphs.

Financial data, including but not limited to stock price information, will be gathered from the Yahoo Finance API. Financial news articles will be retrieved from reliable online sources, such as Kamuoyu Aydınlatma Platformu-KAP.

Segments extracted from financial news articles will be embedded into vectors to facilitate efficient retrieval and similarity searches within the RAG pipeline. We will use E5-large-v2 as the embedding model. Given the potential storage challenges associated with substantial vector embeddings and graph data volumes, we will employ indexing techniques within Neo4j. We will establish vector search indexes in Neo4j to enhance retrieval speed while facilitating effective storage management. Neo4j further aims to incorporate a knowledge graph to answer multi-hop and comparative questions. So, vector embeddings and texts will be kept in Neo4j in a knowledge graph format.

User information and helper information like past chat history will be maintained employing MongoDB, which belongs to the class of No-SQL databases.

The work will be guided by object-oriented programming principles to enhance code reusability and scalability during development. Version control of code will be carried out using Git, whose repositories will be stored on GitHub for ease of code development and change history. Regarding Jira, it will be used to distribute tasks and manage projects.

API service deployment on the web will be done using Heroku or Amazon API Gateway.Standards

#### 4.1.1.2 Economic Constraints

We will use GPT4o-mini–Batch API to convert scraped texts into a knowledge graph for data preprocessing. The cost associated with this service is $0.075 for every 1 million input tokens and $0.300 for each 1 million output tokens [4]. Considering the significant volume of data that requires preprocessing, minimizing token usage is imperative to manage expenses effectively. We will use efficient data processing techniques, like input batching and prompt engineering, to minimize the number of tokens used during preprocessing.

For live interaction with users through a chatbot, we might use GPT4o-mini (there will be many LLM calls, some of which might be done to this model). Pricing is as follows: $0.150 per one million input tokens and $0.600 per one million output tokens [4]. High user engagement might be costly for token consumption. To control this cost, we will tune the chatbot responses for brevity and relevance so that the number of output tokens will be as small as possible without losing functionality in the interactions.

We will rent four NVIDIA A40 GPUs from Runpod for model deployment (the Llama models) and inference at a runtime cost of $1.50 per hour of use [5]. Because continuous usage of GPUs may amount to very high costs, we will manage the cost by planning GPU activities at times of high user end demand and then turning the GPUs off in less busy periods. In addition, we will implement model optimization techniques to reduce computational burden: model quantization and pruning to increase efficiency and reduce the time that the GPU needs to be on.

AWS API Gateway is free for 1 million monthly requests for 12 months, which will keep our API free during the development and testing period [6]. Since the AWS service is free within these limits, they will not contribute to the project's costs for the first year. We will monitor our usage to ensure we stay within the free tier limits and prepare for scaling costs if we think we might go over.

Heroku is a web server with a free tier for a single project. This choice will enable the hosting of our application without additional hosting costs when developing. We will monitor our usage to ensure the free tier is sufficient for our needs and consider an upgrade to a paid plan if needed as the project grows.

We will use Neo4j locally during development and Neo4j AuraDB Free tier for inference based on the suitability of our project needs [7]. The AuraDB Free tier allows small applications; however, if our needs are much more than the free tier offerings, we will have to move to AuraDB Professional, which incurs a $65 monthly expense. This release will improve performance and scalability to handle larger datasets and more complex queries.

Similarly, we will use MongoDB's local version during development and MongoDB Serverless while making inferences. The serverless costs $0.10 per 1 million reads [8]. We will need to monitor our reads to keep our costs in check closely. Using the serverless option means we are only paying for resources used rather than provisioning up-front resources, which is potentially more economical for workloads that have variable or unpredictable traffic patterns.

#### 4.1.1.3 Ethical Constraints

User data protected with stringent security measures, including AES-256 encryption for storage and TLS 1.3 for data transmission. All data is securely stored in Turkey to adhere to KVKK requirements, with access strictly limited to authorized developers. Comprehensive access logs are maintained to monitor and ensure the integrity and security of the data. User activity is monitored transparently to provide personalized recommendations, with explicit consent obtained beforehand. Users are informed about the types of data collected, the reasons for collection, and how it enhances their experience. Sensitive information is excluded from tracking, and all collected data is anonymized whenever possible. Users have complete control over their data, including the ability to delete their information or opt out of monitoring without losing access to the system’s core features. These features ensure that data collection is strictly limited to what is necessary.

Transparency is maintained by utilizing only reliable and publicly available sources and prioritizing official stock exchange websites and other credible financial resources. The chatbot cites its sources in response to ensure accountability. Web scraping practices include rate-limiting and time delays to avoid overwhelming target websites, and the system avoids accessing content behind login walls or requiring credentials. Recommendations provided by the system are designed to be neutral and unbiased, offering clear explanations to help users understand the insights. Information is consistently sourced from up-to-date and verified channels to maintain accuracy. Mechanisms are in place to filter out misleading or harmful content, ensuring that users receive only reliable and actionable advice.

Additionally, safeguards implemented to block fraudulent activities and prevent market manipulation, upholding the system’s fairness and trustworthiness. User data is never shared with third parties without explicit permission, and users are given complete control to manage their data, including the option to delete it at any time. This comprehensive approach ensures that user privacy is prioritized.

#### 4.1.1.4 Legal and Social Constraints

InvestmentHelper-AI provides financial data, forecasts, and analysis to help users make better decisions compared to their own. However, to comply with legal requirements and avoid the unlawful practice of providing financial advice, all insights and predictions offered by the system will be accompanied by a clear disclaimer stating that they do not constitute financial advice. The users will also be advised to consult a licensed financial advisor before making any investment decisions based on the information received from this website.

To accommodate of the diversity of our user base across different ethnicities and regions, InvestmentHelper-AI supports multiple languages. Currently, the system supports Turkish and English, though expanding to more will be straightforward. The internal operations of the chatbot conducted in English. The documents and user inputs translated into English using LLMs to support users in different languages. Regardless of the starting language, user input converted to English for processing. After a response has been generated, the output is first in English and then translated back to the user's original language before being shown to them.

Since the translation depends on large language models, any language that can be translated into English using these models is eligible to be integrated into the system. This approach allows InvestmentHelper-AI to handle a great variety of languages and thus be accessible to users worldwide; it is assumed that the language barrier will not impede access to important financial information.

#### 4.1.1.5 Interface Constraints

The chatbot system relies on user interaction to process input and generate output. Note that, by this reliance, we do not mean that the system will not consider and handle requests from the user that are highly entangled or are logically incorrect. The system will handle edge case scenarios where the user input is not in the correct format or requests from the user that are not within the expected context. The system preprocesses user input to reshape the request in some structure, such that the subsequent pipeline from which the request is processed works robustly, and correct output is generated.

The interface constraint makes the system usable, functional, and conventional. The interface of InvestmentHelper-AI will allow the user to understand the workflow within the environment intuitively. The user can efficiently utilize the functionalities offered within the environment through the readable and minimalist interface. The system utilizes conventions that users are already familiar with to achieve this minimalist goal. For example, we will use third-party (Chainlit) interfaces that mimic the well-known generative pre-trained transformer applications (e.g., OpenAI’s Chat-GPT) for the chatbot interface. To showcase the graphical representation of stocks within the market, we will adopt the generic interface of TradingView, a known website for following trends within the market and filtering attributes of selected stocks. Generally, when a user first uses InvestmentHelper-AI, the user will intuitively know how to use the functionalities. Though it is worth noting that such conventions will inspire the interface, they will not be directly adopted, only indirectly, such that they enhance usability and readability.

#### 4.1.1.6 Extensibility Constraints

The extensibility constraints of the system are tied to the economic constraints discussed earlier. The cost of operations within the system depends on total utilization, the number of requests made to LLMs, and database usage. These costs are manageable in the initial phase of the system; however, as the number of users utilizing the system increases, the economic constraints will significantly affect non-functional requirements, potentially resulting in performance bottlenecks.

Hence, once the utilization level of the system increases extensively, the hardware and software infrastructure will need to scale accordingly. This will require upgrading from free-tier services (e.g., Neo4j AuraDB Free, MongoDB Serverless, and Heroku Free) to paid, professional-grade plans to ensure higher availability, better performance, and scalability. Additionally, more efficient resource management strategies, such as GPU autoscaling, advanced caching mechanisms, and model optimization (quantization and pruning), will need to be implemented to maintain response times and control costs.

To ensure long-term extensibility, the system architecture will remain modular and flexible, enabling the integration of additional GPUs, distributed databases, and alternative cost-effective LLM solutions as the user base grows. This approach ensures the system can handle increased workloads without compromising performance, usability, or reliability.

For such cases, third-party associations will be applied for funding. Also, some form of subscription model or alternative monetization strategies will be implemented to earn the necessary resources for scaling up the system.

### 4.1.2 Risks and Alternatives

#### 4.1.2.1 Technical Risks

**Model Risk**

**Risk:** The selected LLMs (GPT4o-mini, Llama 3.1-70b, and Llama 3.2-1b) may not consistently deliver the desired accuracy and relevance in responses, especially under varying query complexities.

**Alternative:** Incorporate additional or alternative models such as OpenAI’s GPT-4 or other state-of-the-art models to benchmark performance. Additionally, process the user query and the generated response further, to reduce hallucination.

**Scalability Risk**

**Risk:** As the user base grows, the system may face scalability challenges, leading to increased latency and reduced performance.

**Alternative:** Utilize scalable cloud infrastructure services like AWS Elastic Kubernetes Service (EKS) or Google Kubernetes Engine (GKE) to dynamically manage and scale resources based on demand. Implement load balancing and auto-scaling policies to handle traffic spikes efficiently.

**Dependency Risk**

**Risk:** Reliance on external APIs (Yahoo Finance, Kamuoyu Aydınlatma Platformu-KAP) and services (Runpod, Heroku, AWS) may lead to disruptions if these services experience downtime or change their pricing/models.

**Alternative:** Implement multiple data sources and service providers to ensure redundancy. Set up automated failover systems to switch to backup providers seamlessly if primary services experience downtime. Additionally, continuously monitor service performance and maintain strong relationships with alternative vendors to quickly address any disruptions.

**Data Risk**

**Risk:** Integrating data from multiple sources and maintaining consistency in the knowledge graph and databases (Neo4j, MongoDB) can be challenging, potentially leading to data discrepancies.

**Alternative:** Implement robust data validation and synchronization processes. Utilize ETL (Extract, Transform, Load) pipelines with automated consistency checks to ensure data integrity across all systems.

#### 4.1.2.2 Economic Risks

**Cost**

**Risk:** The projected costs for API usage, GPU rentals, and database services may exceed initial estimates, impacting the overall budget.

**Alternative:** Monitor expenses regularly using cost management tools provided by cloud service providers. Optimize resource usage through efficient coding practices and model optimization techniques to keep costs within limits.

**Scaling**

**Risk:** Rapid user growth could lead to unexpectedly high scaling costs, particularly with GPU usage and API calls.

**Alternative:** Prioritize cost-effective scaling strategies, such as prioritizing essential features during high-demand periods.

#### 4.1.1.3 Ethical Risks

**Bias**

**Risk:** The LLMs may exhibit inherent biases, leading to unfair or discriminatory responses that could harm user trust and violate ethical standards.

**Alternative:** Conduct regular bias assessments and implement bias mitigation techniques such as fine-tuning models with diverse and representative datasets. Establish an ethical review board to oversee AI behavior and ensure compliance with fairness standards.

#### 4.1.1.4 Legal Risks

**Regulatory Compliance**

**Risk:** Changes in financial regulations or data protection laws (e.g., GDPR updates) may necessitate significant adjustments to the system, leading to compliance challenges.

**Alternative:** Stay informed about regulatory changes through continuous monitoring and legal consultations. Design the system with flexibility to accommodate regulatory updates, such as modular compliance components that can be easily modified as needed.

Table 01: Factors that can affect analysis and design.

|  |  |  |
| --- | --- | --- |
|  | **Effect level** | **Effect** |
| **User Well-being** | Low to medium | Potential stress caused by financial decision-making could influence user engagement and system requirements. |
| **User Financial Security** | High | Misinformation or biased financial insights could indirectly affect user safety through poor decision-making. |
| **Public welfare** | Medium | Economic tools can contribute positively to public welfare by improving financial literacy and decision-making, helping mitigate economic inequality. |
| **Global factors** | High | Geopolitical events, global economic trends, and international regulations significantly affect financial markets and the system’s data processing. |
| **Cultural factors** | Low | Cultural attitudes toward finance and trust in AI solutions can impact system adoption and usability across different regions. |
| **Social factors** | Low | Privacy concerns, user demographics, and social media trends affect system trust, accessibility, and engagement. |

Table 02: List of Risk and Alternatives

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Likelihood** | Effect on the Project | B Plan Summary |
| **Model Risk** | Low | Inaccurate responses may reduce user satisfaction and trust. | Use alternative models like GPT-4 or ensemble methods; post-process user queries to reduce errors. |
| **Scalability Risk** | Low to Medium | Latency issues during user growth can impact performance and usability. | Utilize AWS EKS/GKE with auto-scaling, load balancing, and caching strategies for traffic handling. |
| **Dependency Risk** | Low | Downtime or cost changes may disrupt core features temporarily. | Develop fallback mechanisms with alternative APIs. |
| **Data Risk** | Low | Data inconsistencies may reduce the accuracy of insights but are manageable. | Implement data validation processes, and real-time monitoring tools. |
| **Cost Risk** | Medium | Budget overruns for GPU/API usage could limit scalability and new feature rollouts. | Optimize resource usage, monitor costs using cloud tools, and prioritize essential features. |
| **Bias Risk** | Low | Biased AI responses can harm trust and reputation. | Conduct bias assessments, fine-tune models with diverse datasets, and oversee AI behavior ethically. |
| **Regulatory Risk** | Low | Non-compliance may lead to adjustments and legal reviews. | Stay informed on regulatory changes and implement modular compliance components |

## Standards

* Use of AES-256 encryption for data storage.
* TLS 1.3 for secure data transmission.
* WCAG compliance for accessibility, including text scalability and color contrast ratios.
* Adherence to KVKK regulations for data protection in Turkey.
* OAuth 2.0 for secure authentication.
* Compliance with international data handling and security guidelines.
* Automated backups every 6 hours for redundancy, ensuring RPO and RTO standards are met.
* Role-Based Access Control (RBAC) for user access management.
* Secure password storage using hashing algorithms like bcrypt.
* Compliance with GDPR principles for data protection outside Turkey.
* Respect for licensing agreements when integrating third-party tools, libraries, and APIs.
* Proper attribution for academic and external resources referenced in the development.
* Compliance with robots.txt directives to avoid scraping restricted sections of websites.

## Project Plan

Table 03: List of work package 1

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 1: *Retrieval Augmented Generation Pipeline*** | | | |
| **Start date: 16.09.2024 End date: 01.02.2025** | | | |
| **Leader:** | ***Hakan Muluk*** | **Members involved:** | ***M. Onur Özdemir***  ***Emir Şahin Dilli*** |
| **Objectives: Implement a reliable and robust RAG pipeline that correctly handles different types of user requests. For each query type—single-hop, multi-hop, comparative, and general—develop a respective pipeline. Ensure coherent information flow and accurate outputs tailored to the query type. The chatbot should support follow-up questions, allowing users to reference previous responses. To achieve state-of-the-art standards, follow new academic research in natural language processing and develop academically validated functionalities and helper functions using benchmarks.** | | | |
| **Tasks:**  ***Task 1.1* Runpod Setup*:* *To provision and configure GPU-based infrastructure on Runpod for hosting and running the large language models efficiently, ensuring low latency and scalability.***  ***Task 1.2* *Translation of Documents:* *To prepare multilingual support by translating documents into target languages, ensuring inclusivity and accessibility for users from diverse linguistic backgrounds.***  ***Task 1.3* *Prompt Preparation for Fivelet Extraction* : *To design effective prompts that extract structured five-part elements (e.g., entity, relation, entity, related firm, and date) from user inputs, enabling coherent data retrieval for the RAG pipeline.***  ***Task 1.4* *Research for Multi-Hop Questions:* T*o analyze and develop techniques for answering multi-hop questions that require chaining multiple pieces of evidence or intermediate reasoning steps to derive the final answer.***  ***Task 1.5* *Research for Comparative Questions:* To study methods for handling comparative questions where responses involve evaluating or comparing entities, such as their features, performance, or outcomes.**  ***Task 1.6* *Research for General Questions:* *To ensure the pipeline can handle broad, single-response questions that require comprehensive and general outputs from the knowledge base.***  ***Task 1.7* *Establish Pipeline for Single Hop*: *To build the single-hop pipeline, which retrieves and generates answers for direct, straightforward queries requiring minimal reasoning or data combination.***  ***Task 1.8* *Scraping of KAP Data*: *To collect and preprocess financial data from Kamuoyu Aydınlatma Platformu (KAP) for integration into the RAG system, ensuring the pipeline has access to up-to-date financial information.***  ***Task 1.9* *Database setup for Chats, Messages and Users*: *To design and implement databases (e.g., MongoDB, Neo4j) for storing chat history, user interactions, and messages, enabling seamless follow-up conversations and user management.***  ***Task 1.10* *Backend Endpoints using FastAPI*: *To develop robust backend APIs with FastAPI that handle requests and serve pipeline outputs.*** | | | |
| **Deliverables**  ***D1.1:* Single Hop Pipeline**  ***D1.2:* Scraping and Translation Tool**  ***D1.3:* Follow-up Questions** | | | |

Table 04: List of work package 2

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 2: *Processing of Data*** | | | |
| **Start date: *16.09.2024*  End date: *25.01.2025*** | | | |
| **Leader:** | ***Emir Şahin Dilli*** | **Members involved:** | ***M. Onur Özdemir***  ***Hakan Muluk*** |
| **Objectives: Process the data incoming from external sources. Manipulate data such that it is in a desirable structure for LLM to understand the context and extract fivelets. The extracted fivelets will be used for multi hop and comparative queries. Automate the scraping process from various sources including KAP and utilizing Yahoo API. Translate all incoming data to english whilst protecting the inherent context of all texts.** | | | |
| **Tasks:**  ***Task 2.1 Data Collection from KAP***  ***Automate the extraction of financial news data from KAP (Korea Accounting Professional) to gather relevant information for the knowledge base.***  ***Task 2.2 Translation to English Using GPT API***  ***Utilize the GPT API to translate the collected financial data from its original language into English, ensuring compatibility with Language Learning Models (LLMs).***  ***Task 2.3 Structuring Data into Markdown Format***  ***Convert the translated data into Markdown (MD) format to enhance readability and facilitate easier interpretation by LLMs.***  ***Task 2.: Processing Table Data***  ***Transform complex table data into plain text by grouping similar rows using a chunking method with a similarity threshold above 0.8, then process these groups through the GPT API.***  ***Task 2.5 Integrating Processed Data***  ***Combine the GPT API responses with the original Markdown text to create a cohesive and structured dataset suitable for the knowledge base.***  ***Task 2.6 Pipeline Implementation for Diverse Data Sources***  ***Establish robust pipelines to extract financial information from various sources and formats such as PDF, HTML, and DOCX, standardizing the data into Markdown for seamless integration.*** | | | |
| **Deliverables**  ***D1.1:* *KAP Data Scraping***  ***D1.2:* *Translation of Data to English***  ***D1.3:* *Markdown Structuring***  ***D1.4:* Diverse Data Scraping** | | | |

Table 05: List of work package 3

|  |  |  |  |
| --- | --- | --- | --- |
| **WP 3: *Multi-agent Notification System*** | | | |
| **Start date: *01.12. 2024*  End date: *15.04.2025*** | | | |
| **Leader:** | ***M. Onur Özdemir*** | **Members involved:** | ***Emir Şahin Dilli***  ***Hakan Muluk*** |
| **Objectives: Build a robust notification system that decreases the overall complexity of processing notification requests, e.g, “Notify me when event A occurs”. Processing each individual request of each user is expensive, since each individual request would be processed by a separate LLM call. The objective is to build a system that decreases the overall complexity (decrease the number of LLM calls) such that each user’s request is satisfied correctly.**  **More formally, if there are N notification requests, then, satisfy all requests with M LLM calls, with M << N. Tree structures, or subgrouping of requests will be developed.** | | | |
| **Tasks:**  ***Task 2.1* *Research different techniques that can reduce the overall complexity, these include Monte Carlo Tree Search, RAPTOR trees, one level coalitional structures (cooperative games from game theory) and so on. Proper technique will be developed using an efficient and algorithmic approach.***  ***Task 2.2* *Implement the found theory and benchmark the complexity for a given number of LLM calls. If more than one theory is developed, pick the one that reduces the overall complexity the most, and is more applicable to the expected distribution of requests.***  ***Task 2.3 Formulate the found technique analytically, and prove that it works theoretically, and showcase its effectiveness using empirical data, from the benchmarks applied.*** | | | |
| **Deliverables**  ***D1.1:* *Notification System***  ***D1.2:* *Query re-distribution System***  ***D1.3:* *Query search System*** | | | |

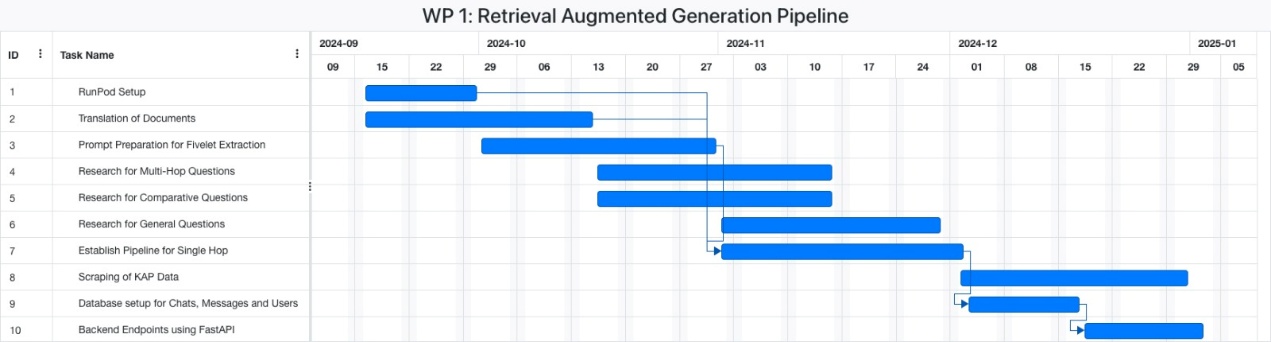


Figure 19 Gantt Chart for Work Process 1

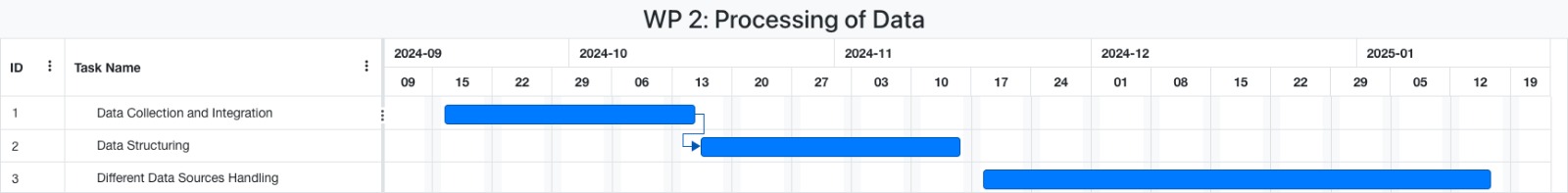


Figure 20 Gannt Chart for Work Process 2

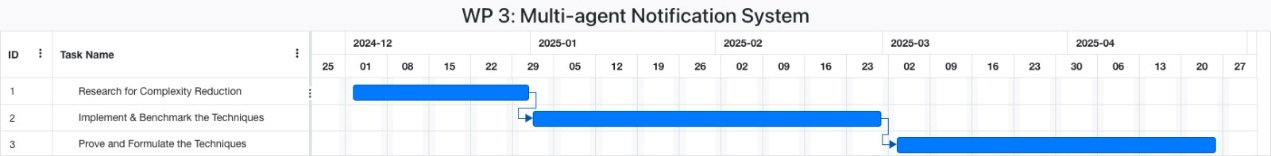


Figure 21 Gantt Chart for Work Process 3

## Ensuring Proper Teamwork

We utilize Jira for monitoring and managing tasks, allowing us to track progress, assign responsibilities, and maintain visibility across all project activities. This tool helps us maintain an organized workflow and ensures that everyone is aware of their individual responsibilities and deadlines.

We hold weekly meetings to discuss progress, address any challenges, and plan upcoming tasks. These meetings provide a platform for team members to share updates, collaborate on problem-solving, and align our efforts towards common goals. Furthermore, we conduct weekend check-ins to ensure that any time-sensitive issues are addressed.

Our team has no fixed roles; instead, everyone contributes to the reports and supports one another as needed. This flexible structure encourages members to leverage their strengths and assist others, conducting a supportive and cohesive environment. By working together and helping each other, we ensure that all aspects of the project are overseen efficiently and effectively, achieving our objectives successfully.

## Ethics and Professional Responsibilities

As developers of a finance application that leverages Retrieval-Augmented Generation (RAG) models and predictive algorithms, our foremost responsibility is to ensure that the system delivers accurate, reliable, and socially responsible outputs. The high-stakes nature of financial decision-making means that any errors or misrepresentations in the information provided can lead to significant economic harm for users.

To uphold the integrity of our RAG models, we are committed to providing true and fact-based information about firms. Our system is meticulously designed to source responses exclusively from verified and reputable datasets, ensuring that users can trust the results we provide. Additionally, we embed references and citations within our outputs to substantiate the information, offering users a clear view of the data's origins. This practice not only enhances transparency but also allows users to independently verify the credibility of the information.

We recognize that while users may act based on our predictions, the ultimate responsibility for financial decisions rests with them. To uphold our ethical obligations, we include disclaimers and provide clear explanations of the limitations of AI predictions. This ensures that users understand the outputs are not infallible or absolute, thereby managing their expectations and protecting our credibility by clarifying that we do not assume accountability for decisions made solely on AI-generated insights.

Reliability is another cornerstone of our professional duties. Financial systems must operate continuously without interruptions, as downtime can disrupt services and cause financial losses for users. To achieve this, we implement fail-safe systems and ensure that our infrastructure can handle high availability. As stewards of a financial platform, we are committed to guaranteeing operational resilience and continuity, addressing both technical risks and ethical accountability to our users.

Furthermore, we actively gather feedback from users to refine our system, address any gaps, and adapt to evolving needs. By incorporating user feedback into our iterative improvement process, we demonstrate our commitment to professionalism and the continuous enhancement of both the accuracy and usability of our application. This collaborative approach reinforces trust and ensures that the platform remains aligned with user expectations and ethical standards.

## Planning for New Knowledge and Learning Strategies

This project involves active academic research on Natural Language Processing. Naive RAG approaches have some limitations, they are not well suited for multi-hop question answering, comparative question answering or generating answers to general questions. In addition, in financial news analysis, it is crucial to account for time dependence, since with time, certain information can become obsolete and inaccurate. To account for all the above issues, we are developing certain new capabilities combining state of the art Natural Language Processing academic research.

We look for new publications in natural language processing through academic websites like arXiv. Then, incorporate ideas if they are relevant for our purposes. Then, we develop the theoretical pipeline that accounts for user queries which are time dependent, single-hop, multi-hop, comparative and general. It is quite extensive to incorporate all the different techniques within the system and make them work coherently. After the theoretical pipeline has established, real life scenarios and edge cases applied to the theory, if the theory stands, we continue building on top of that theory. If it does not, then we change the part where it is not extensive enough and fails at a certain task. This change might impact of the system, or it might change a minor aspect, which depends on the edge case. We repeatedly apply this process, and in the end, we end up in a stable theory where we are confident in implementing. Note that, as stated, the algorithm that developed is dynamically updated, since we are trying to account for all the fundamental aspects of financial news analysis, which involve numerous types of processes. The techniques we have developed will extend the state-of-the-art RAG approaches currently in 2024, by combining all the different fundamental aspects defined above. It will structure as academic work, hence will be documented and benchmarked.

We are looking for new sites, news channels from which to scrape financial information. Accessing accurate and dynamic financial information is crucial for our retrieval augmented generation system, since the system will periodically update its knowledge base according to such information processed from the web. Therefore, we are looking for reliable sources of finance and trying to automate the scraping process from those sites, since the system needs periodical input feed. Note that, the extent of sites (number of sites) where new information is processed can be treated as a hyperparameter. Though if it is too extensive, then the knowledge base stored will be O(n) in terms of size, where n is the number of sites; yet, new information introduced to the system will be upper-bounded by a smaller constant O(m), m < n. Hence, the information stored will grow at a faster rate than the information that is needed. It will be investigated further within the group, the extent of sites upon which financial information will be taken.

Our strategies in learning the necessary information involve active research and following state of the art trends through academic sites like arXiv, GitHub or sites that post academic work. We look for relevant updates. Though, the field is very large, and many of the new work introduced is not directly applicable to our case at hand. Though, it is crucial to catch the few that are. Also, we talk about such systems with some of our friends within the department, which can be useful. They can notify us regarding new or old work that has been introduced, then, we can check whether the mentioned work is applicable for our desired functionalities. Hence, it is important for us to keep information flow between people that work or have worked on RAG systems or on Natural Language Processing in general.

# 4 Glossary

**KG:** Knowledge Graph

**LLM:** Large Language Model

**RAG:** Retrieval Augmented Generation

# 5 References

[1] “The complete AI powered stock research platform - FinChat.io,” *FinChat.io*. <https://finchat.io/>

[2] “AI TICKER CHAT - Intelligent Investing simplified.” <https://aitickerchat.com/>

[3] “Uptrends.ai | Track trending stocks.” <https://www.uptrends.ai/>

[4] OpenAI, “Pricing,” *OpenAI*, 2024. <https://openai.com/api/pricing/>

[5] RunPod, “Pricing for GPU Instances, Storage, and Serverless,” *Runpod.io*, 2024. https://www.runpod.io/pricing (accessed Nov. 21, 2024).

‌[6] “Amazon API Gateway Fiyatlandırması | API Yönetimi | Amazon Web Services,” *Amazon Web Services, Inc.*, 2024. https://aws.amazon.com/tr/api-gateway/pricing/ (accessed Nov. 21, 2024).

[7] “Neo4j Pricing,” *Neo4j Graph Data Platform*. <https://neo4j.com/pricing/>

[8] MongoDB, “Pricing,” *MongoDB*. https://www.mongodb.com/pricing