



**Module Code & Module Title CS6P05NI Final Year Project Assessment Weightage & Type**

**5% FYP Proposal Year and Semester 2023-24 Autumn**

**YESE (Ye Stock Exchange) portfolio Optimization**

**Student Name: Ronak Krishna Shrestha**

**Group: AI3**

**London Met ID: 22085771**

**College ID: np0**

**Assignment Due Date:**

**Assignment Submission Date:**

**Submitted To:**

**External Supervisor-**

**Internal Supervisor-**

**Word Count :**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

# Introduction

The Nepal Stock Exchange (NEPSE) has witnessed significant growth in recent years, attracting both local and international investors. However, many investors lack the necessary tools and knowledge to make informed investment decisions. The volatility of the stock market, coupled with the complexity of portfolio management, poses a challenge for investors seeking to optimize their returns while managing risk. According to a report by the Nepal Rastra Bank, the stock market in Nepal has shown fluctuations with an average annual return of around 12% over the past decade, but with considerable risks involved.

Despite the availability of various portfolio management systems, such as Nepse Alpha, Hamro Share, Ansu Invest, Share Sansar, Smart Wealth Pro, and Mero Lagani, these tools often rely on traditional methods and lack advanced predictive analytics. This project, the “YESE (Ye Stock Exchange) Optimization System,” is a response to the critical need for advanced tools and methodologies to assist investors in making informed decisions and optimizing their portfolios. By leveraging Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), this project aims to provide more accurate and personalized investment strategies.

## Problem Statement

Many investors in Nepal rely on traditional methods for stock selection and portfolio management, which do not adequately account for market dynamics and individual risk preferences. This leads to suboptimal investment decisions and increased risks. A study by Sharma et al. (2022) indicates that majority of investors in Nepal do not use data-driven approaches for portfolio optimization, resulting in missed opportunities and increased risks2. Additionally, the lack of accessible and user-friendly analytical tools further complicates the investment process for many.

Current portfolio management systems in Nepal, such as Nepse Alpha, Hamro Share, Ansu Invest, Share Sansar, Smart Wealth Pro, and Mero Lagani, have several deficiencies:

* Nepse Alpha: Primarily uses traditional methods like mean-variance optimization, which may not capture the complexities of the market1.
* Hamro Share (Ektukra Creation): Focuses on providing bulk IPO results and basic portfolio tracking but lacks advanced predictive analytics.
* Ansu Invest: Offers in-depth analysis and valuation but does not leverage advanced machine learning techniques for portfolio optimization.
* Share Sansar: Provides technical and fundamental analysis but relies heavily on traditional methods without incorporating modern AI techniques.
* Smart Wealth Pro: Offers portfolio management and investment insights but lacks the integration of advanced machine learning models1.
* Mero Lagani: Provides real-time stock quotes and portfolio management but does not utilize advanced predictive models.

## Project as a Solution

The “YESE (Ye Stock Exchange) Optimization System” offers a modern solution to the pervasive issue of suboptimal investment decisions in the NEPSE market. By leveraging advanced machine learning techniques such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), this project aims to provide users with personalized investment strategies based on real-time data analysis and risk assessment. While the primary focus will be on LSTM and GRU, the potential for incorporating Temporal Fusion Transformers (TFT) will be explored if feasible.

**Key Features and Benefits:**

* Advanced Predictive Analytics: Utilizing LSTM and GRU to analyze historical stock data and predict future price movements, enabling more informed investment decisions.
* Personalized Investment Strategies: Tailoring recommendations based on individual risk profiles and investment goals, allowing for a more customized approach to portfolio management.
* Real-Time Data Integration: Incorporating real-time market data to ensure that investment strategies remain relevant and responsive to current market conditions.
* User-Friendly Interface: Designing an intuitive platform that simplifies the investment process for users, making advanced analytics accessible to both novice and experienced investors.
* Risk Management Tools: Providing tools for assessing and managing risk, helping investors to optimize their portfolios while minimizing potential losses.

By addressing the limitations of existing portfolio management systems and leveraging cutting-edge machine learning techniques, the YESE Optimization System aims to empower investors in Nepal to make data-driven decisions that enhance their investment outcomes.

# Aims and Objectives

## Aims

Enhance Investment Decision-Making in the Nepal Stock Exchange: The primary aim of this project is to develop the YESE (Ye Stock Exchange) Optimization System, which utilizes advanced machine learning techniques, specifically Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), to provide accurate stock price predictions and optimize portfolio management for investors in the Nepal Stock Exchange (NEPSE).

## Objective

1. Accurate Stock Return Predictions:

* Develop and train LSTM and GRU models using historical NEPSE stock data to predict future stock returns and volatility.
* Implement models that automatically use the latest stock data to ensure accurate and real-time forecasts.

1. Portfolio Optimization Engine:

* Implement a portfolio optimization algorithm based on Modern Portfolio Theory to suggest the optimal allocation of stocks, balancing risk and return according to user preferences (risk tolerance and investment horizon).

1. User-Friendly Web Interface (Streamlit):

* Develop a responsive Streamlit-based web application where users can input their preferences (risk tolerance, investment horizon, etc.) and receive personalized portfolio recommendations and stock predictions.
* Ensure that users are not required to upload historical data; the system will use automatically scraped data.

1. Backend API Integration (FastAPI):

* Implement a FastAPI backend to handle data requests, predictions, and optimization logic.
* FastAPI will provide an efficient way to manage model inference and serve the results to the frontend (Streamlit).

1. Data Integration:

* Integrate an automated web scraping system to pull daily NEPSE stock data from reliable sources, ensuring that the platform always has the most up-to-date information.
* Historical stock data will already be included and continuously updated without user intervention.

1. Visualization of Portfolio and Predictions:

* Create intuitive visualizations for users to view portfolio composition, expected returns, risk metrics (e.g., volatility), and the efficient frontier, making it easy to understand their investment options.

1. Scenario Analysis:

* Provide scenario analysis tools to simulate how portfolios would perform under different market conditions (e.g., bullish/bearish scenarios) based on the predictive models

# Expected Outcomes and deliverables

By the end of the project, several key outcomes are anticipated. The system will offer real-time portfolio recommendations based on predicted stock returns and user-defined risk tolerance and investment horizons, allowing users to make informed stock allocation decisions that maximize potential returns while managing risk effectively. The accuracy of stock price and return predictions will be enhanced using advanced machine learning models like LSTM and GRU, leveraging both historical and automatically scraped real-time stock data from NEPSE.

The project will deliver a user-friendly Streamlit web application, where users can interact with personalized portfolio recommendations and view real-time visualizations of expected performance metrics such as risk, returns, and the Sharpe ratio. Additionally, a robust backend, built using FastAPI, will handle model predictions and optimization processes, efficiently managing data retrieval and inference tasks. The system will also feature automated data scraping and integration mechanisms that ensure the NEPSE stock data remains current, eliminating the need for manual user input of historical data.

Visualizations will play a pivotal role in helping users comprehend the risk and return profiles of their portfolios, with graphical representations that simplify complex investment strategies. Furthermore, the platform will provide scenario analysis tools, allowing users to simulate the effects of various market conditions (e.g., bull and bear markets) on their portfolios, and offer recommendations for rebalancing strategies to maintain optimal performance over time.

**Key Features of the Portfolio Optimization System:**

1. Stock Return Prediction:

* The system will predict future stock returns based on historical and real-time data using LSTM and GRU models, providing accurate and timely forecasts.

1. Portfolio Optimization:

* The system will generate optimized portfolios based on predicted returns, risk tolerance, and user-defined constraints, using modern portfolio optimization algorithms.

1. User-Friendly Interface:

* A Streamlit web application that is simple and accessible, where users can input preferences, view results, and interact with real-time stock predictions and portfolio recommendations.

1. Backend API for Prediction and Optimization (FastAPI):

* FastAPI will serve as the backend for the system, ensuring efficient handling of model predictions, data processing, and API requests from the frontend (Streamlit).

1. Real-Time Data and Visualization:

* The system will automatically scrape and integrate NEPSE stock data daily, ensuring that portfolio recommendations are based on the most current market data. Additionally, users can track portfolio performance over time through intuitive visualizations

# Project risks, threats and contingency plans

## Data Quality and Availability Risks

* **Risk/Threat:** Incomplete, inaccurate, or outdated financial data can lead to poor model performance and unreliable predictions. Data may be unavailable for key variables, or inconsistencies in sources could compromise the project’s output.
* **Contingency Plan:**
* Data Validation: Implement rigorous data validation checks to ensure data integrity before use.
* Multiple Data Sources: Cross-verify data from multiple sources (e.g., Yahoo Finance, Alpha Vantage) to fill in any gaps and correct inconsistencies.
* Data Augmentation: Use synthetic data generation techniques to supplement real data where gaps exist.

## Model Overfitting

* **Risk/Threat:** The model might perform well on training data but poorly on unseen data, leading to overfitting and unreliable predictions that can result in misleading portfolio management decisions.
* **Contingency Plan:**
* Regularization Techniques: Implement dropout, L1/L2 regularization, and early stopping to minimize overfitting.
* Cross-Validation: Apply k-fold cross-validation to ensure that the model generalizes well to unseen data.
* Simpler Models: Start with simpler models to establish a solid baseline before introducing more complex architectures.

## Technical Challenges

* **Risk/Threat:** The complexity of implementing and fine-tuning models such as LSTM and TFT could cause technical delays and block project progress due to their intricacies.
* **Contingency Plan:**
* Incremental Development: Break down the implementation into smaller, manageable tasks, and tackle them incrementally to reduce overwhelm and complexity.
* Seek Help: Utilize online forums, technical communities, or mentors for specific technical challenges.
* Reference Documentation: Regularly consult official documentation and tutorials for guidance on TensorFlow, LSTM, and TFT model architectures.

## Computational Resource Limitations

* **Risk/Threat:** Insufficient computational resources (CPU/GPU) could slow down model training, experimentation, and deployment, leading to project delays.
* **Contingency Plan:**
* Cloud Computing: Utilize cloud-based platforms like Google Colab, AWS, or Azure for access to powerful GPUs for training.
* Optimize Code: Refactor code to improve efficiency, using batch processing or reducing model complexity where possible.
* Leverage Local Resources: Maximize the use of local computing power and, if necessary, upgrade hardware to support more demanding computational tasks.

## Market Volatility and Unpredictability

* **Risk/Threat:** The inherent volatility and unpredictability of financial markets can undermine model accuracy, leading to irrelevant or outdated predictions in changing conditions.
* **Contingency Plan:**
* Dynamic Models: Develop models with adaptability to retrain and adjust to changing market data.
* Using sentiment analysis on stock news: Collecting the critical news of the stocks used in the portfolio management website and use it as weighted parameters if needed
* Scenario Analysis: Implement scenario analysis to assess model performance across a range of market conditions, including extreme cases (e.g., bull and bear markets).
* Risk Management: Introduce risk management strategies within the portfolio optimization algorithm to mitigate potential financial losses.

## Regulatory and Compliance Issues

* **Risk/Threat:** Regulatory changes in the financial sector could impact the data or models used, making them non-compliant with new standards. Non-compliance could lead to legal or project issues.
* **Contingency Plan:**
* Stay Informed: Regularly track updates in financial regulations and ensure that your system is aligned with any changes.
* Consult Experts: Engage legal and compliance experts to ensure all aspects of the project conform to current laws.
* Document Thoroughly: Maintain comprehensive documentation of data sources and methodologies to ensure compliance can be demonstrated if needed.

# Methodology

This project will be developed using the Agile methodology. Although Agile is traditionally used in team-based development, it will be adapted for this solo project to ensure systematic progress, flexibility, and continuous improvement throughout the development process.

Agile emphasizes iterative development, where tasks are broken down into manageable units called sprints. Each sprint will focus on achieving specific project goals, ensuring regular review and adaptation to meet changing needs or overcome unforeseen challenges. The project will be completed in several phases, with feedback incorporated at the end of each sprint.

# Resource Requirement

For successful completion of this project required tools and technologies are as follows:

## Hardware Requirements:

* **Processor:** Multi-core processor (e.g., Intel i7 or AMD Ryzen 7 or higher). (current AMD Ryzen 5 4500U should handle preprocessing and basic model tasks sufficiently.)
* **RAM:** Minimum of 16 GB.
* **GPU:** Dedicated GPU (e.g., NVIDIA RTX 3080 or higher) for efficient deep learning model training. (This can be offloaded to cloud services like Google Colab Pro, AWS, or Azure.)
* **Storage:** At least 512 GB SSD for fast data access and storage of models and datasets.

## Software Requirements:

1. Operating System: Windows 10 or higher, MacOS
2. Stable Internet Connection: Required for cloud services and data scraping.
3. IDE: Visual Studio Code, PyCharm
4. Programming Languages:

* Python: For backend, machine learning models, data analysis
* HTML/CSS/JavaScript: For frontend interface if applicable
* SQL: For querying the MySQL database
* Bash/CLI: For Git and cloud commands

1. Frontend Framework:

* Streamlit: For building the user interface of the web application.

1. Backend Framework:

* FastAPI: For backend API development to handle requests and database interaction.

1. Database:

* MySQL: For storing user information, portfolio data, and other necessary user inputs.
* SQLite (Optional): As an alternative local database for testing or lightweight use.

1. Version Control:

* Git: For version control and managing project changes.
* GitHub: For remote repository hosting and collaboration.

1. Cloud Platforms:

* Google Colab Pro: For cloud-based GPU access and model training.
* AWS or Azure (Optional): For more scalable computing resources as needed.

1. Libraries for Machine Learning and Data Analysis:

* TensorFlow/Keras: For building LSTM and TFT models.
* PyTorch: Optional framework for deep learning.
* Pandas: For data manipulation and analysis.
* NumPy: For numerical computations.
* Matplotlib & Seaborn: For data visualization.
* Scikit-learn: For preprocessing and machine learning models.

1. Web Scraping (if required):

* BeautifulSoup: For web scraping additional data from sites like NEPSE Alpha.
* Selenium: For web scraping if needed (with browser automation).

1. Design and Prototyping Tools:

* Figma: For any UI/UX design requirements.
* Adobe XD or Canva: For creating wireframes and prototypes.

1. Documentation and Flowchart Tools:

* Microsoft Visio or Draw.io: For creating software architecture diagrams and flowcharts.
* GanttProject: For project management and tracking activity timelines.

## Publications and Literature:

**Journals, Articles, and Research Papers:**

1. Portfolio Optimization with Prediction-Based Return Using Long Short-Term Memory Neural Networks: Testing on Upward and Downward European Markets (Research Paper)
2. Stock Price Forecasting with Deep Learning: A Comparative Study (Article)

Received: 31 July 2020; Accepted: 25 August 2020; Published: 27 August 2020

1. Predicting the Direction of NEPSE Index Movement with News Headlines Using Machine Learning (Article)

Published: 11 June 2024

Academic Editor: Guglielmo Maria Caporale

1. Stock Price Prediction of Nepal Using LSTM (KEC Conference Paper)
2. Temporal Fusion Transformers for Enhanced Multivariate Time Series Forecasting of Indonesian Stock Prices

(International Journal of Advanced Computer Science and Applications - IJACSA)

1. Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting

(Authors: Bryan Lima, Sercan Ö. Arıkb, Nicolas Loeffb, Tomas Pfisterb

University of Oxford, UK, Google Cloud AI, USA)

## Datasets:

* Primary Dataset:
* NEPSE Top 100 Companies: Historical stock data for the top 100 companies listed on NEPSE (Nepal Stock Exchange). This data is currently sourced from a GitHub repository.
* Additional Data Sources (if more companies or financial data is required):
* NEPSE Alpha: For scraping additional financial data as needed.
* Sharesansar: For scraping live financial data as needed.

## Access to IT Resources

* Cloud Platforms:
* Google Colab Pro: Primary platform for model training with enhanced GPU access, more memory, and priority on resources.
* AWS or Azure: Optional cloud platforms offering scalable GPU and CPU resources for larger datasets and extended model training periods.

# Work Breakdown Structure

# Milestones chart

# Project Gantt chart

# Conclusion

# References

**There are no sources in the current document.**

# Appendix

## Similar projects:

* **Research paper**

|  |  |  |  |
| --- | --- | --- | --- |
| Article | Main Findings/Results | Methodology | Project Contribution |
| Portfolio Optimization with Prediction-Based Return Using LSTM | Improved accuracy in predicting returns in upward and downward European markets | LSTM Neural Networks | Demonstrated the effectiveness of LSTM in portfolio optimization |
| Stock Price Forecasting with Deep Learning: A Comparative Study | Compared various deep learning models for stock price prediction | Various deep learning models (e.g., CNN, LSTM) | Highlighted the strengths and weaknesses of different models |
| Predicting the Direction of NEPSE Index Movement with News Headlines Using Machine Learning | Achieved significant accuracy in predicting NEPSE index movements using news headlines | Machine Learning algorithms | Showcased the potential of using news headlines for stock market predictions |
| Stock Price Prediction of Nepal Using LSTM | Achieved high accuracy in predicting stock prices in Nepal | LSTM Neural Networks | Provided a case study for using LSTM in the Nepalese stock market |
| Temporal Fusion Transformers for Enhanced Multivariate Time Series Forecasting | Improved forecasting accuracy for Indonesian stock prices | Temporal Fusion Transformers (TFT) | Demonstrated the effectiveness of TFT in multivariate time series forecasting |
| Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting | Provided interpretable and accurate multi-horizon forecasts | Temporal Fusion Transformers (TFT) | Highlighted the interpretability and accuracy of TFT |

* **Current Portfolio Management Systems in Nepal**

|  |  |  |
| --- | --- | --- |
| ****Name**** | ****Features**** | ****Deficiency**** |
| Nepse Alpha | Traditional mean-variance optimization, portfolio tracking | May not capture market complexities |
| Hamro Share (Ektukra Creation) | Bulk IPO results, basic portfolio tracking | Lacks advanced predictive analytics |
| Ansu Invest | In-depth analysis and valuation | Does not leverage advanced machine learning techniques |
| Share Sansar | Technical and fundamental analysis | Relies heavily on traditional methods without modern AI techniques |
| Smart Wealth Pro | Portfolio management, investment insights | Lacks integration of advanced machine learning models |
| Mero Lagani | Real-time stock quotes, portfolio management | Does not utilize advanced predictive models |

## Use case diagram:

**12.2.1 High-Level Use Case**

1. Use Case: User Registration

Actors: New User

Description: New user register in the system to access its features. They provide the necessary personal information to create an account. The system verifies the user credentials, and the system registers the user after verification and stores the user in database.

Includes: Validate Credentials (System verifies if the user details are correct).

1. Use case: Create Portfolio

Actors: User, System

Description: The user creates a portfolio by selecting certain stocks from a predefined list. The system allows users to enter stock data (up to 100 stocks) and creates a personalized portfolio

Includes: Stock Selection, Validate Stock Availability (checks if the selected stocks are available in the system).

Extends: Suggest Portfolio Change (for users with no prior stocks, the system can suggest an entirely new portfolio).

1. Use case: Suggest portfolio change

Actors: System

Description: This use case extends Create Portfolio. It applies to users who do not have any stocks yet. The system suggests a set of diverse stocks based on user-defined investment amounts and desired return percentages. These suggestions take into account the risk-return curve and are capped by regulatory restrictions (such as Nepal Rastra Bank interest rates).

Includes: Risk Assessment, Price Prediction, Portfolio Optimization (System balances stocks according to MPT).

1. Use case: Enter Existing Portfolio

Actors: User

Description: Users with pre-existing stock portfolios can manually enter the stock names and quantities they currently hold (limited to 100 stocks within the system). Users can also update their portfolios later if they buy more stocks or sell existing ones. For users without any stock holdings, the portfolio starts empty, and they can add stocks after purchasing them.

Includes: Stock Update (The user can later update stock information), Price Prediction, Portfolio Optimization (System optimizes based on existing stocks and predicted prices).

Extends: Create Portfolio (existing portfolios become part of the portfolio creation process).

1. Use case: Select stocks for portfolio

Actors: User, System

Description: Users select individual stocks (e.g., 4 for NIC, 7 for NBM, 5 for XYZ Hydro). The system shows expected returns for each stock and suggests additional diverse stocks to balance the portfolio and minimize risk.

Includes: View Expected Return, Suggest Additional Stocks (to diversify portfolio based on risk), Price Prediction.

Extends: Portfolio Optimization (the system adjusts the portfolio according to MPT i.e. Modern Portfolio Theory).

1. Use case: Stock update(Optional)

Actors: User, System

Description: Users can update their portfolios with new stocks they have purchased. The system advises users on stock purchases based on market trends and portfolio balance.

Includes: View Market Changes, Adjust Portfolio (suggests buying new stocks based on market predictions).

Extends: Portfolio Optimization (the portfolio is re-optimized after stock updates).

1. Use case: stock exit(optional)

Actors: User, System

Description: The system advises users to exit certain stocks if market data suggests they will drop in value soon. The user can sell stocks to lock in profits or minimize losses.

Includes: View Market Changes, Exit Recommendations (provides suggestions based on stock price drop predictions).

Extends: Portfolio Optimization (user’s portfolio is optimized after exiting stocks).

1. Use case: stock Prediction

Actors: System

Description: The system uses AI models (such as LSTM or GRU) to predict stock prices based on historical data. These predictions guide users in making informed decisions regarding their portfolio creation, stock purchases, and exits. The prediction model analyzes past stock performance and trends to forecast future prices, allowing the system to provide personalized suggestions to the user.

Includes: Historical Data Analysis, Trend Prediction.

Extends: Suggest Portfolio Change, Stock Update, Stock Exit.

Notes: Stock predictions are integrated into various user interactions like portfolio creation, stock selection, and portfolio management updates, giving users insights into potential stock movements.

1. Use case: Portfolio Optimization

Actors: System

Description: The system applies Modern Portfolio Theory (MPT) to optimize users' portfolios by balancing risk and return. The optimization takes into account the expected returns of stocks (predicted using LSTM/GRU models) and suggests diversification strategies to minimize risk while maximizing return. This use case is triggered whenever the user creates or updates their portfolio or when the system advises stock purchases or exits.

Includes: Risk-Return Assessment, Diversification Suggestions.

Extends: Create Portfolio, Suggest Portfolio Change, Stock Update, Stock Exit.

Notes: Portfolio optimization ensures that users maintain a balanced and diversified portfolio that aligns with their risk tolerance and financial goals.

**12.2.2 Expanded Use case diagram**

1) Use Case: User Registration

Actors: New User

Description: New users register in the system to access portfolio management features. They provide necessary personal information to create an account. The system verifies credentials and registers the user after verification.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| |  |  | | --- | --- | | 1. User opens the registration page. |  | | |  |  | | --- | --- | |  | 2. The system presents the registration form. | |
| |  |  | | --- | --- | | 3. User enters personal information (name, email, etc.). |  | | |  |  | | --- | --- | |  | 4. The system validates the input. | |
| |  |  | | --- | --- | | 5. User submits the form. |  | | |  |  | | --- | --- | |  | 6. The system checks for duplicate accounts. | |
| |  |  | | --- | --- | |  |  | | |  |  | | --- | --- | |  | 7. If valid, the system registers the user and stores the information in the database. | |
| |  |  | | --- | --- | | 8. System sends a confirmation message. |  | | |  |  | | --- | --- | |  |  | |

Alternative Cases:

Line 4: If input validation fails (e.g., missing fields, invalid email), the system displays an error message, and the user must correct the details.

Line 6: If an account with the same email already exists, the system notifies the user to log in or reset the password.

2) Use Case: Create Portfolio

Actors: User, System

Description: Users create their own portfolio by selecting stocks from the system's stock database. The user can choose up to 100 stocks.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User navigates to the portfolio creation page. | 2. The system presents a list of available stocks. |
| 3. User selects stocks for the portfolio (up to 100). | 4. The system validates the stock choices. |
| 5. User submits the portfolio. | 6. The system calculates expected return and risk. |
|  | 7. System saves the portfolio and updates the user's account with the new data. |
| 8. System presents an option to suggest portfolio changes. | 9. User can opt for suggested changes or finish the creation process. |

Alternative Cases:

Line 4: If the user selects stocks exceeding the limit of 100, the system notifies the user to reduce the selection.

Line 8: If the user chooses to accept suggested changes, the system provides additional stock recommendations.

3) Use Case: Suggest Portfolio Change

Actors: System

Description: The system suggests portfolio changes based on user-defined investment amounts and desired returns. It considers risk-return factors using Modern Portfolio Theory.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User requests portfolio suggestions. | 2. System asks for investment amount and desired returns. |
| 3. User enters the investment and return preferences. | 4. The system applies MPT and suggests diverse stocks. |
|  | 5. System displays suggested stocks with quantity, risk, and expected returns. |
| 6. User reviews the suggestion. | 7. User can confirm or reject the changes. |
|  |  |

Alternative Cases:

Line 4: If user-defined returns exceed feasible levels (based on market data), the system notifies the user with a capped return percentage.

Line 7: If the user rejects the changes, the system returns to the previous portfolio state.

4) Use Case: Enter Existing Portfolio

Actors: User, System

Description: The user enters details of the stocks they currently own into the system. Users can manually enter up to 100 stocks to build their initial portfolio.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User selects the option to enter existing stocks. | 2. System presents a form to enter stock names and quantities. |
| 3. User enters stock names and quantities (up to 100). | 4. System validates stock details against the stock database. |
|  | 5. System updates the user's portfolio with the entered stocks. |
|  |  |
|  |  |

Alternative Cases:

Line 4: If a stock entered is not in the system's database, the system alerts the user and requests a valid stock name.

Line 3: If the user enters more than 100 stocks, the system prompts the user to reduce the number.

5) Use Case: Select Stocks for Portfolio

Actors: User, System

Description: Users select individual stocks to include in their portfolio. The system calculates expected returns and suggests additional stocks for diversification.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User selects stocks from the available list. | 2. The system shows the expected returns for selected stocks. |
| 3. User confirms stock choices. | 4. The system suggests additional stocks for diversification based on risk. |
| 5. User reviews the suggestion. | 6. User either accepts or rejects the additional suggestions. |
|  |  |
|  |  |

Alternative Cases:

Line 4: If the user’s stock choices create a high-risk portfolio, the system recommends balancing it with safer stocks.

Line 6: If the user rejects the suggestions, the portfolio remains as selected.

6) Use Case: Stock Update

Actors: User, System

Description: Users update their portfolio based on market changes. The system advises users to buy new stocks based on price movements and portfolio risk.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User navigates to the stock update page. | 2. The system checks the user's portfolio for market updates. |
| 3. System presents a list of suggested stock purchases based on current market data. | 4. User reviews suggestions and can modify the portfolio. |
| 5. User confirms new stock purchases. | 6. The system updates the portfolio in the database. |
|  |  |
|  |  |

Alternative Cases:

Line 3: If no stock changes are needed based on market data, the system notifies the user of the stable state.

Line 5: If the user rejects the suggested changes, the portfolio remains as is.

7) Use Case: Stock Exit

Actors: New User, System

Description: Users are advised by the system to exit certain stocks based on market changes. The system predicts potential declines in stock prices and suggests selling to avoid losses or secure profits.

Typical Course of Events:

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. User navigates to the stock exit page. | 2. The system analyzes the user’s portfolio. |
| 3. System suggests stocks to exit based on predicted market movements. | 4. User reviews the suggested exits. |
| 5. User confirms the stocks to sell. | 6. System updates the portfolio and removes the exited stocks from the user's account. |
|  |  |
|  |  |

Alternative Cases:

Line 4: If input validation fails (e.g., missing fields, invalid email), the system displays an error message, and the user must correct the details.

Line 6: If an account with the same email already exists, the system notifies the user to log in or reset the password.

8) Use Case: Stock Prediction

Actors: System

Description: The system uses AI models (LSTM/GRU) to predict stock prices based on historical data. These predictions are used for portfolio management, including stock selection, updates, and exits.

Typical Course of Events:

|  |  |
| --- | --- |
| System Action (from other use cases) | System Reaction (FastAPI Backend and Streamlit Frontend) |
| 1. Historical Data Collection (FastAPI): The system automatically retrieves the latest stock data through scraping/APIs and updates the existing CSV file containing stock data. | **2. Back-end:** FastAPI periodically collects new stock data through an automated process (e.g., cron job). The data is appended to the CSV, ensuring that the stock data stays up to date (new entries include columns such as published\_date, open, high, low, close, etc.).  **Model Retraining:** The model does not retrain immediately after new data is added. Instead, a scheduled retraining cycle (e.g., weekly or monthly) is employed to keep the model updated without overfitting. The retraining uses the new data along with the existing historical data to adjust the model weights. |
| User Requests Stock Prediction (Streamlit Frontend): The user selects a stock from the available list and requests a price prediction. | Front-end: Streamlit UI allows the user to select a stock via a dropdown and submit the prediction request.  Back-end: FastAPI receives the request and accesses the latest model (pre-trained on historical data and periodically retrained). FastAPI fetches the relevant stock’s historical data from the CSV and uses the most recent model to generate predictions.  Interaction: Streamlit sends the selected stock as an API request, and FastAPI returns real-time predictions based on the current version of the model. |
| Price Prediction (FastAPI): The system retrieves historical stock data from the updated CSV and feeds it into the pre-trained LSTM/GRU model for trend prediction. | Back-end: FastAPI reads the historical and most recent stock data from the CSV, passes it through the most up-to-date model (which has been retrained periodically), and predicts future stock prices. No real-time retraining happens here; the model's pre-existing knowledge is leveraged for predictions.  Front-end: Streamlit visualizes the predicted stock prices along with historical trends, showing expected growth or decline. |
| Streamlit Displays Prediction Results: The prediction is presented to the user with relevant metrics and historical vs. predicted price comparison. | Front-end: Streamlit renders interactive visualizations using matplotlib or plotly, allowing the user to compare predicted and historical stock prices. Other important metrics like expected return, risk factors, and trend directions are also displayed.  Back-end: FastAPI sends the prediction data in JSON format through its API endpoints, enabling real-time response and rendering by the Streamlit front-end. |
|  |  |

**Interaction with the Model: Retraining Logic**

1. Model Training:

* Initially, the LSTM/GRU model is trained on a comprehensive historical dataset containing stock data (e.g., several years of data for 100 stocks).
* This model learns stock price trends, patterns, and behaviour through this historical data.

1. New Data Ingestion:

* New stock data is scraped or collected daily, and the CSV is updated accordingly. This new data becomes part of the dataset.

1. Model Retraining:

* Instead of retraining the model every day (which can cause overfitting or excessive resource consumption), the system schedules a retraining cycle (weekly, bi-weekly, or monthly, depending on requirements).
* During retraining, the model is updated with the new data added since the last training session, adjusting the model's weights and potentially improving future predictions.

1. Real-Time Predictions:

* When a user requests a stock prediction, the system uses the most recent version of the model, which incorporates knowledge from historical data and periodic retraining cycles.
* The model doesn’t retrain with each user request. Instead, it relies on the latest trained state to generate predictions quickly and efficiently

1. Use Case: Portfolio Optimization

Actors: System

Description: The system applies Modern Portfolio Theory (MPT) to optimize user portfolios. The optimization considers predicted stock returns (using LSTM/GRU models), balancing risk and return, and suggests a diversification strategy.

Typical Course of Events:

|  |  |
| --- | --- |
| System Action (from other use cases) | System Reaction (FastAPI Backend and Streamlit Frontend) |
| User Requests Portfolio Optimization (Streamlit Frontend): The user selects portfolio optimization from the menu. | **Front-end**: Streamlit provides the user with an interface to select either manual or automated optimization. The user can input preferences such as risk tolerance, desired return, and investment amount. Upon submission, Streamlit sends an API request to FastAPI. **Back-end**: FastAPI processes this request and prepares to apply Modern Portfolio Theory. |
| **Portfolio Data Processing (FastAPI)**: The system retrieves the user’s portfolio and the latest stock price predictions for stocks in the portfolio. | **Back-end**: FastAPI fetches the user’s current portfolio and uses the predicted stock prices (from the LSTM/GRU models). It calculates risk (variance) and expected return for each stock in the portfolio. **Interaction**: FastAPI accesses the LSTM/GRU predictions stored in the database to estimate future return |
| **Risk-Return Assessment (FastAPI)**: The system assesses the risk and expected return for each stock in the portfolio. | **Back-end**: FastAPI uses MPT to calculate the risk-return trade-off, balancing high-risk and low-risk stocks to optimize the overall portfolio. The system considers the predicted price trends and calculates the optimal weight allocation for each stock. |
| **Diversification Suggestions (FastAPI)**: Based on MPT, the system suggests changes to diversify the portfolio, balancing between high-risk and low-risk stocks. | **Back-end**: FastAPI generates suggestions, such as reducing exposure to high-risk stocks or adding more low-risk stocks to achieve an optimal balance. **Front-end**: Streamlit displays suggested changes, highlighting which stocks to buy more of and which ones to reduce, along with the expected risk reduction and return improvement. |
| **Streamlit Presents Optimization Results**: The optimized portfolio is presented to the user with a detailed breakdown. | **Front-end**: Streamlit displays a breakdown of the portfolio’s risk, expected return, and suggested allocation changes. Charts and graphs (e.g., pie charts or bar graphs) illustrate how much of the portfolio is allocated to each stock and its risk. Users can interactively see the potential improvements in the portfolio’s performance. |
| **Automatic Portfolio Reoptimization (FastAPI)**: The system automatically reoptimizes the portfolio when new data is added, or the market changes. | Back-end: FastAPI automatically triggers portfolio reoptimization whenever significant stock price changes are predicted. The system updates the user’s portfolio and provides a notification via the Streamlit interface.  Front-end: Streamlit shows alerts notifying the user of portfolio reoptimization results. |

**Backend Interaction (FastAPI):**

* FastAPI handles all API requests from Streamlit, including fetching stock data, running LSTM/GRU models, performing portfolio optimization, and updating user portfolios.
* Model Handling: FastAPI loads pre-trained LSTM/GRU models when needed, using them to predict stock prices based on historical data already stored in the system.
* API Responses: FastAPI returns all prediction and optimization data to Streamlit in JSON format for front-end visualization.

**Frontend Interaction (Streamlit):**

* User Interface: Streamlit provides an interactive UI for users to request stock predictions, create and optimize portfolios, and view updates in real-time.
* Data Visualization: Streamlit uses libraries like matplotlib, plotly, and altair to visualize stock trends, portfolio allocation, and risk-return analysis.
* Real-Time Interaction: Streamlit communicates with FastAPI to get instant responses, ensuring seamless updates to stock predictions and portfolio optimizations.

## System Flow chart:

A diagram of a process

Description automatically generated