



Bilkent University
Department of Computer Engineering

Senior Design Project
T2438
Para-Meter

Analysis and Requirement Report

22003598, Abdullah Samed Uslu, samed.uslu@ug.bilkent.edu.tr

22102566, Tuna Saygın, tuna.saygin@ug.bilkent.edu.tr

22102825, Sıla Özel, sila.ozel@ug.bilkent.edu.tr

22002756, Muti Kara, muti.kara@ug.bilkent.edu.tr

Selim Aksoy

Mert Bıçakçı, Atakan Erdem

16.12.2024

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.....	3
2 Current System.....	3
3 Proposed System.....	4
3.1 Overview.....	4
3.2 Functional Requirements.....	4
3.3 Non-functional Requirements.....	6
3.4 Pseudo Requirements.....	6
3.5 System Models.....	7
3.5.1 Scenarios.....	7
3.5.2 Use-Case Model.....	9
3.5.3 Object and Class Model.....	12
3.5.4 Dynamic Models.....	13
3.5.4.1 Activity Diagram for Custom Dataset Addition & Preprocessing.....	13
3.5.5 User Interface.....	14
4 Other Analysis Elements.....	35
4.1 Consideration of Various Factors in Engineering Design.....	35
4.1.1 Constraints.....	35
4.1.1.1 Implementation Constraints.....	35
4.1.1.2 Financial Constraints.....	35
4.1.1.3 Ethical Constraints.....	35
4.1.1.4 Public Health, Safety, and Welfare Factors.....	36
4.1.1.5 Global and Cultural Factors.....	36
4.1.1.6 Economic Factors.....	36
4.1.2 Standards.....	38
4.2 Risks and Alternatives.....	38
4.2.1 Data Inaccuracy and Quality Risks.....	38
4.2.3 Legal and Regulatory Risks.....	39
4.2.4 Security Risks.....	40
4.2.5 User Adoption and Usability Risk.....	40
4.3 Project Plan.....	41
4.4 Ensuring Proper Teamwork.....	48
4.5 Ethics and Professional Responsibilities.....	49
4.6 Planning for New Knowledge and Learning Strategies.....	49
4.6.1 Key Areas for New Knowledge.....	49
4.6.2 Learning Strategies.....	50
5 Glossary.....	51
6 References.....	51

Analysis and Requirement Report

Para-Meter

1 Introduction

The financial market has seen a growing demand for accessible, data-driven investment tools that enable users to optimize their portfolios with precision and flexibility. While effective for institutional investors with specialized knowledge, retail investors find traditional portfolio optimization solutions hard to use due to their technical complexity, expensive costs, and reliance on conventional analysis techniques.

Para-Meter is a machine learning-powered portfolio optimization tool designed to bridge this gap. By integrating advanced data analysis, machine learning models, and a simple yet intuitive interface, Para-Meter empowers users to easily optimize their investment strategies, manage risk, and achieve their financial goals.

This report outlines the functional and non-functional requirements of Para-Meter, key engineering considerations, identified risks and contingency plans, and a detailed project roadmap. Additionally, it addresses constraints, ethical responsibilities, and strategies for acquiring new knowledge and skills during the project.

This comprehensive document aims to ensure a shared understanding of the project's goals, deliverables, and methodology, paving the way for Para-Meter to revolutionize portfolio optimization for retail investors.

2 Current System

The market has many portfolio optimization tools, but most of them don't fully meet the needs of modern investors. Institutional tools like BlackRock-Aladdin or SimCorp-Axioma provide advanced features but are too expensive and complex for most retail investors. On the other hand, retail-focused tools like Magnus or Talos offer simpler solutions but lack flexibility, like adding custom datasets or customizing algorithms.

Many existing tools rely on traditional financial models, such as mean-variance optimization, which struggle to perform in today's volatile and data-driven markets. These models often fail to incorporate alternative data sources, such as macroeconomic indicators, which are increasingly critical for accurate predictions. Moreover, the widespread use of opaque "black-box" models erodes trust, as users cannot understand how decisions are made.

Retail investors who face these issues often use manual methods, like spreadsheets or basic tools, which don't offer advanced analytics or machine learning features. This creates inefficiency and limits their ability to optimize portfolios effectively.

Para-Meter solves these problems by combining affordability, advanced features, and transparency. Unlike competitors, Para-Meter allows users to customize algorithms, add their own datasets, train their own models, and still keep the interface simple and easy to use. This makes it a flexible and accessible tool for retail investors, bridging the gap between complex institutional tools and basic retail solutions.

3 Proposed System

3.1 Overview

The proposed system, Para-Meter, is a next-generation machine learning-based portfolio optimization tool that aims to transform the way retail investors manage their investments. The system provides users with dynamic, transparent, and customizable portfolio recommendations by integrating advanced financial modeling with support for alternative data sources. Its modular architecture enables adaptability for both retail and institutional users, ensuring that powerful analytics remain accessible and affordable.

Key highlights of the system include algorithmic portfolio optimization, robust risk analysis, and real-time notifications—all delivered through a clean and user-friendly interface. Unlike traditional tools, Para-Meter allows users to seamlessly integrate custom datasets, select and compare machine learning models, and personalize risk-return configurations to align with their individual goals. The platform also prioritizes transparency, enabling users to understand how model decisions are made, thus building trust and confidence in the tool.

3.2 Functional Requirements

User Features

- Users can create accounts, log in and manage profiles, store their preferences, save portfolios, and select features.
- A dashboard provides real-time visualization of portfolio metrics, asset allocations, and performance over time through graphs and charts.
- Users can compare different machine learning models side-by-side based on performance metrics, accuracy, and historical returns before selecting a model for portfolio optimization.

Customization

- Users can import alternative datasets (e.g., social sentiment, satellite data, economic indicators) to enhance portfolio predictions.
- Users can create custom features from raw data or choose predefined features like technical indicators, fundamental metrics, or sentiment scores.
- Users can choose from various machine learning models, including decision trees, SVMs (Support Vector Machine), neural networks, and ensemble methods.
- Users can set return goals, risk tolerance levels (using metrics like volatility and VaR), and investment horizons.
- Users can configure portfolios as long-only or long-short, set net and gross exposure limits, and manage leverage constraints.

Performance Analysis

- The platform uses cross-validation, out-of-sample testing, and rolling window testing for robust model validation.
- Integrated stress testing and Monte Carlo simulations assess portfolio performance under various market conditions.
- The platform provides detailed performance tracking, including metrics like Sharpe ratio, drawdown, turnover, and sector or factor exposures.

Notification and Alert System

- Users can set notifications and alerts for specific events, such as portfolio rebalancing thresholds, market movements, or changes in alternative data sources (e.g., sentiment analysis).

Optimization and Maintenance

- Models update periodically to incorporate the latest market data, maintaining relevance over time.
- The tool preprocesses data, including normalization and feature engineering, to ensure high-quality input for models.
- Automated tuning methods (e.g., grid search and Bayesian optimization) are available to optimize model performance.

3.3 Non-functional Requirements

Usability

- The tool should have an intuitive UI, accessible to non-technical users, with easy-to-use options for model selection, customization, and analysis.

Reliability

- High availability to ensure consistent access and minimal downtime, particularly for real-time portfolio updates and rebalancing.
- Robust security protocols to protect user data and ensure compliance with data protection regulations.

Performance

- The platform should ensure low latency for portfolio rebalancing and simulations using parallel computing techniques.
- Models and calculations should be optimized to minimize processing time without sacrificing accuracy, even under heavy computational loads.

Supportability

- Modular code structure allows easier updates, model additions, and future feature integration.

Scalability

- Support for portfolios of varying sizes, with efficient handling of large datasets and concurrent requests.
- Support for large-scale data scraping and processing.

3.4 Pseudo Requirements

- The system will use cloud platforms like AWS, Azure, or Google Cloud for accessibility and scalability.
- Python will be used for the backend due to its efficiency in machine learning and data processing tasks. The frontend will use JavaScript frameworks like React for an interactive and responsive user interface.
- Version control will be managed through GitHub, ensuring smooth collaboration and code management.
- Docker will be used for creating containers to ensure consistent system performance across different machines.
- External APIs such as Yahoo Finance or Alpha Vantage will fetch financial data, while users can also upload data in common formats like CSV or Excel.
- The system will be accessible through a web browser, eliminating the need for installations and enabling ease of use.

3.5 System Models

3.5.1 Scenarios

Scenario 1: Adding a New Portfolio

- **Use Case Name:** Add a New Portfolio
- **Actors:** User
- **Entry Condition:** The user is logged into their account and clicks on the “Add New Portfolio” option from the dashboard or menu.
- **Exit Condition:** The new portfolio is successfully saved, or the user cancels the operation.
- **Flow of Events:**
 - The user selects “Add New Portfolio.”
 - The system prompts the user to enter details such as portfolio name, investment amount, and risk tolerance.
 - The user fills in the details and clicks “Save.”
 - The system validates the inputs and saves the portfolio.
 - The dashboard updates to show the new portfolio.

Scenario 2: Optimizing an Existing Portfolio

- **Use Case Name:** Optimize Portfolio
- **Actors:** User
- **Entry Condition:** The user selects a portfolio from their list and chooses the “Optimize Portfolio” option.
- **Exit Condition:** The optimized portfolio is displayed, or the user cancels the operation.
- **Flow of Events:**
 - The user selects a portfolio and clicks “Optimize Portfolio.”
 - The system retrieves portfolio data and current market information.
 - The user selects optimization preferences (e.g., risk level, target returns).
 - The system runs the optimization algorithm and displays the results.
 - The user chooses to apply the changes or cancel.

Scenario 3: Uploading Custom Data

- **Use Case Name:** Upload Alternative Data
- **Actors:** User
- **Entry Condition:** The user clicks on the “Upload Data” button on the dashboard.
- **Exit Condition:** The uploaded data is successfully added to the system or the user cancels the upload.
- **Flow of Events:**
 - The user clicks “Upload Data.”
 - The system prompts the user to select a file (e.g., CSV or JSON format).
 - The user uploads the file and selects how the data should be used.
 - The system validates the file format and processes the data.

- A confirmation message is displayed, and the data becomes available for analysis.

Scenario 4: Receiving Portfolio Notifications

- **Use Case Name:** Receive Notifications
- **Actors:** User
- **Entry Condition:** A predefined condition for portfolio alerts is met (e.g., a portfolio's risk threshold is exceeded).
- **Exit Condition:** The user views or dismisses the notification.
- **Flow of Events:**
 - The system monitors portfolio performance in real-time.
 - A predefined condition, such as a drastic risk change in the portfolio, triggers an alert.
 - The system sends a notification to the user (via email or in-app alert).
 - The user views the notification and decides to take action (e.g., rebalance the portfolio).

Scenario 5: Viewing Portfolio Performance Metrics

- **Use Case Name:** View Performance Metrics
- **Actors:** User
- **Entry Condition:** The user selects a portfolio and navigates to the "Performance Metrics" section.
- **Exit Condition:** The performance metrics are displayed, or the user exits the section.
- **Flow of Events:**
 - The user selects a portfolio from the dashboard.
 - The system retrieves historical and current performance data.
 - The system displays metrics such as Sharpe ratio, VaR, and drawdown in a graphical format.
 - The user analyzes the data and makes informed decisions about the portfolio.

3.5.2 Use-Case Model

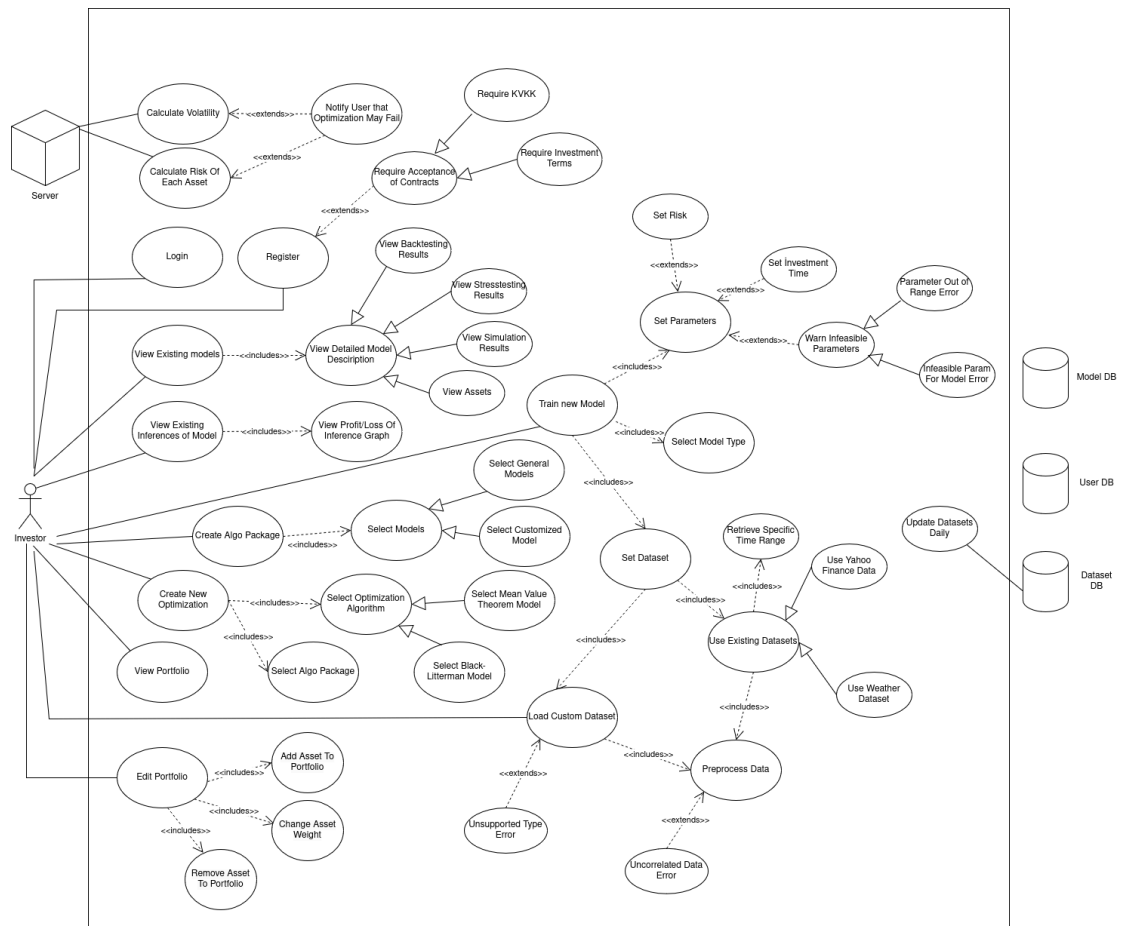


Figure 1. Use Case Diagram.

Figure 1 represents our app's use case diagram to develop features in a high-level design. The following groupings are done to explain the elements of the use case diagram in a more detailed and organized way.

Actors

- **Investor (Primary Actor):** The user interacting with the system to perform actions like creating models, viewing results, managing assets, and optimizing portfolios.
- **Server:** Handles backend computations like volatility calculations, risk assessments, and storing models.
- **Databases:**
 - **Model DB:** Stores trained models.
 - **User DB:** Stores user-specific information.
 - **Dataset DB:** Manages datasets for model training and backtesting.

Major Use Cases

Model Management:

The investor interacts with the system to create, view, and manage models:

- **Create Algo Package:** Allows users to define a new algorithmic model package. It includes:
 - **Select Models:** Choose from existing models.
 - **Select Customized Model:** Customize model parameters.
 - **Select Optimization Algorithm:** Define the optimization method (e.g., Mean Variance Optimization).
 - **Select Mean Value Theorem Model:** Use specific model frameworks.
 - **Select Black-Litterman Model:** Select Black-Litterman for asset allocation.
- **Train New Model:** Executes the training process based on selected parameters and datasets.

Data Management:

The investor manages datasets for backtesting, training, and model simulations:

- **Set Dataset:** Allows the user to prepare the data for model training. It includes:
 - **Load Custom Dataset:** Upload user-defined data.
 - **Use Existing Dataset:** Select pre-existing datasets.
 - **Retrieve Specific Time Range:** Filter data by specific time periods.
 - **Use Yahoo Finance Data:** Fetch market data from Yahoo Finance.
 - **Use Weather Dataset:** Integrate external weather-related data.
- **Update Dataset Daily:**
Ensures the database updates automatically with the latest data.

Portfolio Management:

The investor manages and edits portfolios based on model outputs:

- **View Portfolio:** Allows users to view their portfolio's performance.
- **Edit Portfolio:** Enables modifications to the portfolio. It includes:
 - **Add Asset to Portfolio:** Add new assets.
 - **Remove Asset from Portfolio:** Remove existing assets.
 - **Change Asset Weight:** Adjust asset allocation.

Results Visualization:

The investor views model outputs, performance, and associated risks:

- **View Backtesting Results:** Display results from backtesting models using historical data.
- **View Stress Testing Results:** Analyze how models perform under extreme conditions.
- **View Simulation Results:** Visualize predictions generated by trained models.
- **View Detailed Model Description:** Provides an in-depth explanation of the model, its parameters, and its use cases.
- **View Profit/Loss or Inference Graph:** Shows graphical performance metrics (e.g., P&L over time).

Parameter Management

The investor sets and adjusts parameters to optimize model performance:

- **Set Parameters:** Configure hyperparameters and settings for models. It includes:
 - **Set Risk:** Define risk levels for optimization.
 - **Set Investment Time:** Specify the duration for investments.
 - **Parameter Out of Range Error:** Warn users when parameters exceed permissible limits.
 - **Warn Infeasible Parameters:** Notify users of invalid combinations.
 - **Infeasible Param for Model Error:** Indicate errors when parameters fail model constraints.

Risk and Volatility Calculations

- **Calculate Volatility:** Computes the volatility of assets in the portfolio.
- **Calculate the Risk of Each Asset:** Calculates the risk level for each asset to guide decisions.

Notifications and Terms

- **Notify User That Optimization May Fail:** Warn users about potential optimization issues.
- **Require Investment Terms:** Ensure that users agree to investment-related terms before proceeding.
- **Require KVVK:** Request user consent for data privacy and usage compliance.

3.5.3 Object and Class Model

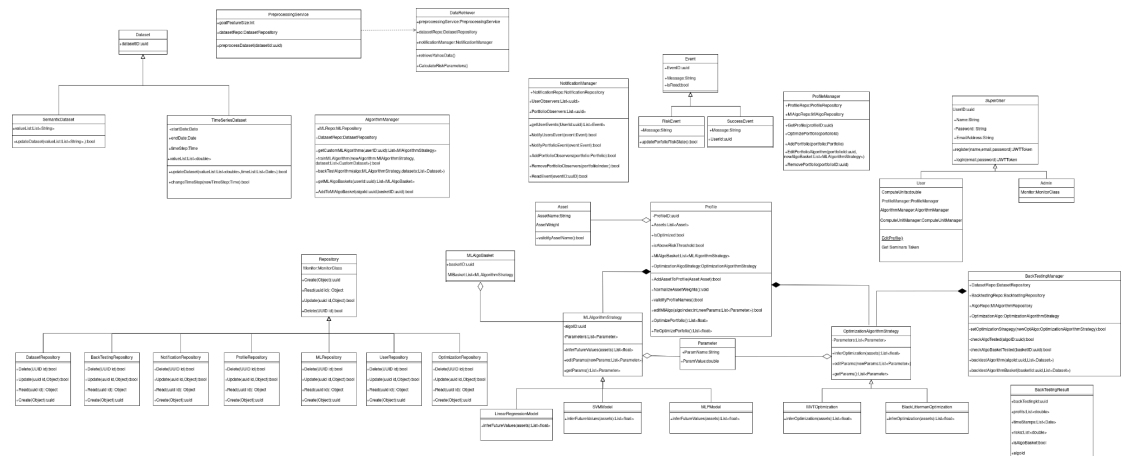


Figure 2. Class Diagram.

The class diagram in Figure 2 depicts an overview of the entities and what functionality is obtained in each class. Additionally, design patterns that we learned in Object Oriented Software Engineering (CS 319) are used to reduce and manage complexity and handle changes more efficiently. The following part of the section is the list of the patterns used.

Strategy Pattern:

The Strategy Pattern allows for defining multiple algorithms or behaviors and selecting one at runtime [1]. In the class diagram, `MLAlgorithmStrategy` and `OptimizationAlgorithmStrategy` are used to dynamically change the different types of algorithms.

Repository Pattern:

The Repository Pattern abstracts data access and provides a central interface for CRUD (Create, Read, Update, Delete) operations [2]. This improves the separation of concerns and simplifies the interaction with the database. The class diagram uses `DatasetRepository`, `Back Testing Repository`, `Notification Repository`, `Profile Repository`, `ML Repository`, `User Repository`, etc., as repository patterns.

3.5.4 Dynamic Models

3.5.4.1 Activity Diagram for Custom Dataset Addition & Preprocessing

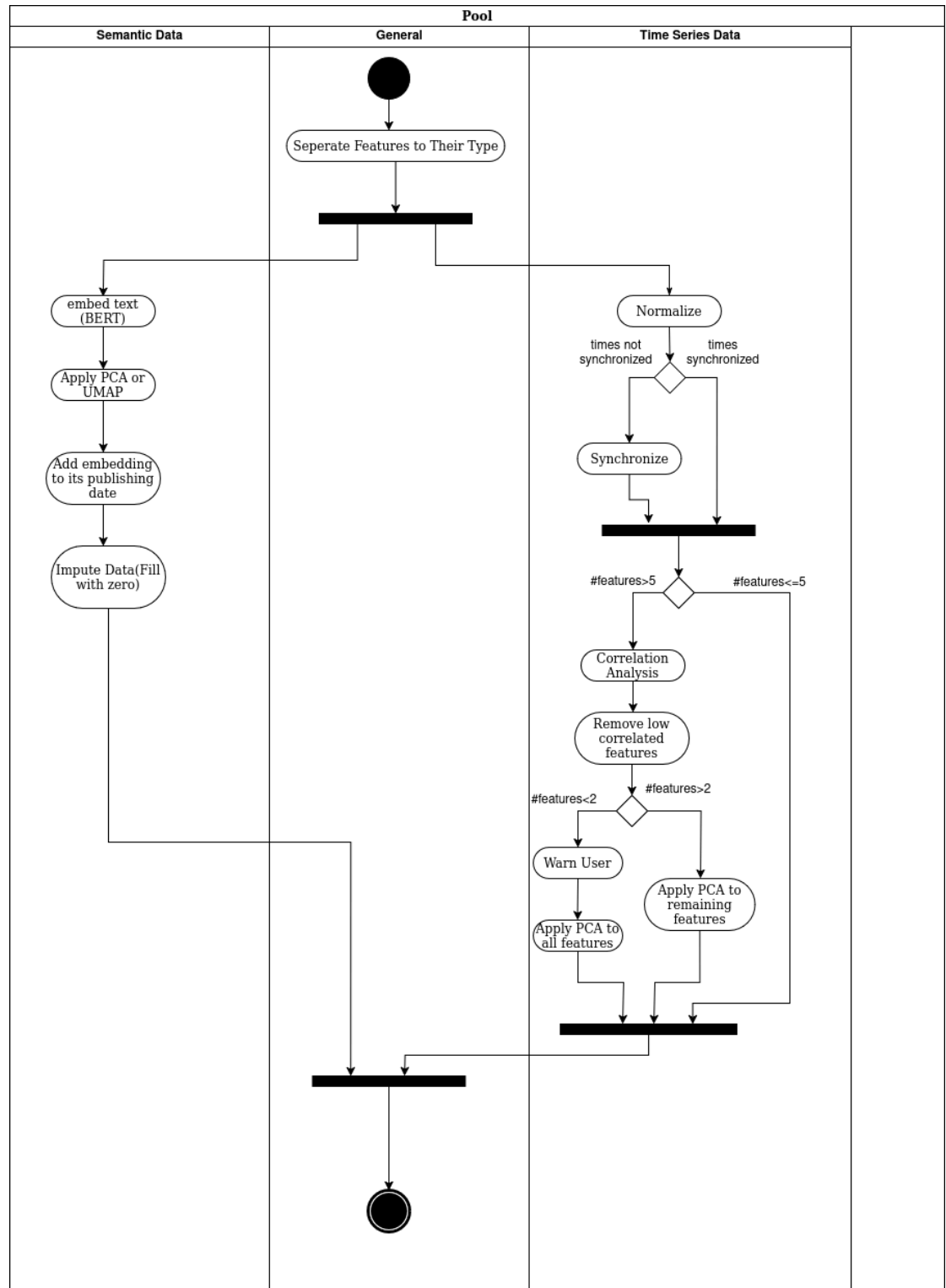
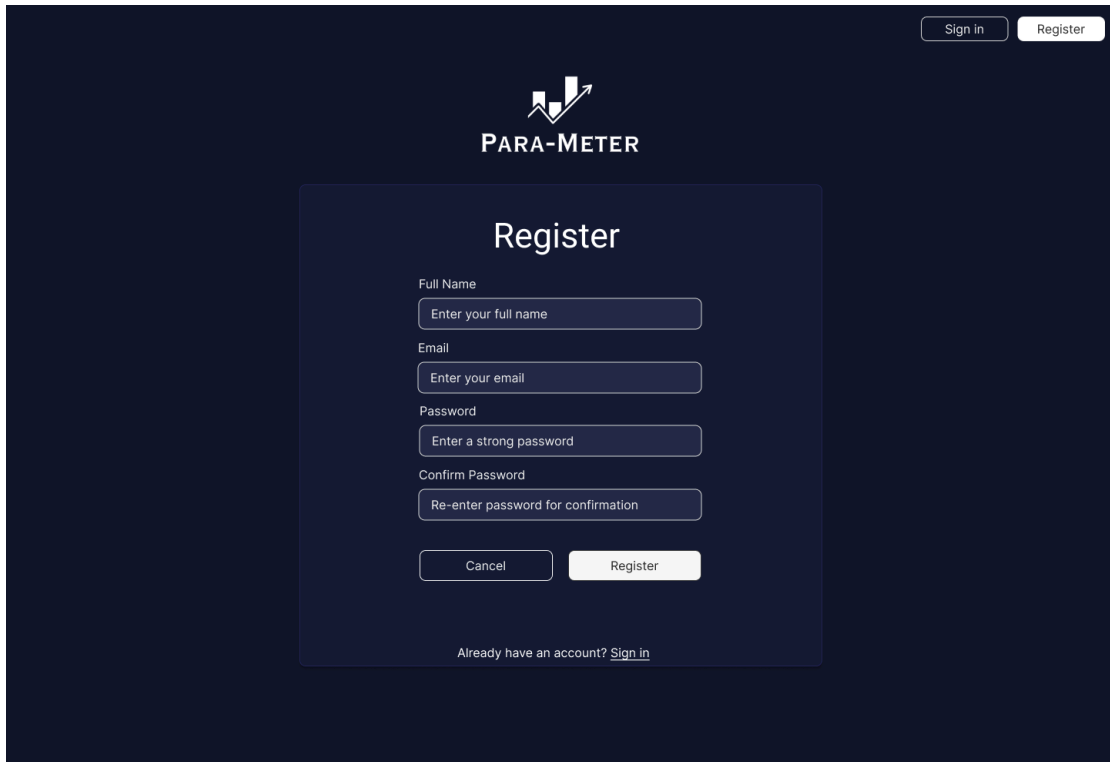


Figure 3. Activity Diagram for custom dataset addition.

The activity diagram in Figure 3 is for custom dataset addition and feature selection of the retrieved data from sources such as Yahoo Finance. Because there are lots of different types and sizes of data, it is better to create an activity diagram.

3.5.5 User Interface



The image shows a dark-themed user interface for a registration page. At the top right, there are two buttons: "Sign in" and "Register". In the center, there is a logo consisting of a bar chart with an upward arrow, followed by the text "PARA-METER". Below the logo is a "Register" form. The form has the following fields and labels: "Full Name" with a placeholder "Enter your full name", "Email" with a placeholder "Enter your email", "Password" with a placeholder "Enter a strong password", and "Confirm Password" with a placeholder "Re-enter password for confirmation". At the bottom of the form are two buttons: "Cancel" and "Register". Below the form, there is a link that says "Already have an account? [Sign in](#)".

Figure 4. Registration Page.

The registration page allows new users to create accounts:

- **Input Fields:** Includes Full Name, Email, Password, and Confirm Password.
- **Action Buttons:** “Register” confirms the input, while “Cancel” exits the registration process.
- **Sign-in Link:** Redirects existing users to the sign-in page in Figure 5.

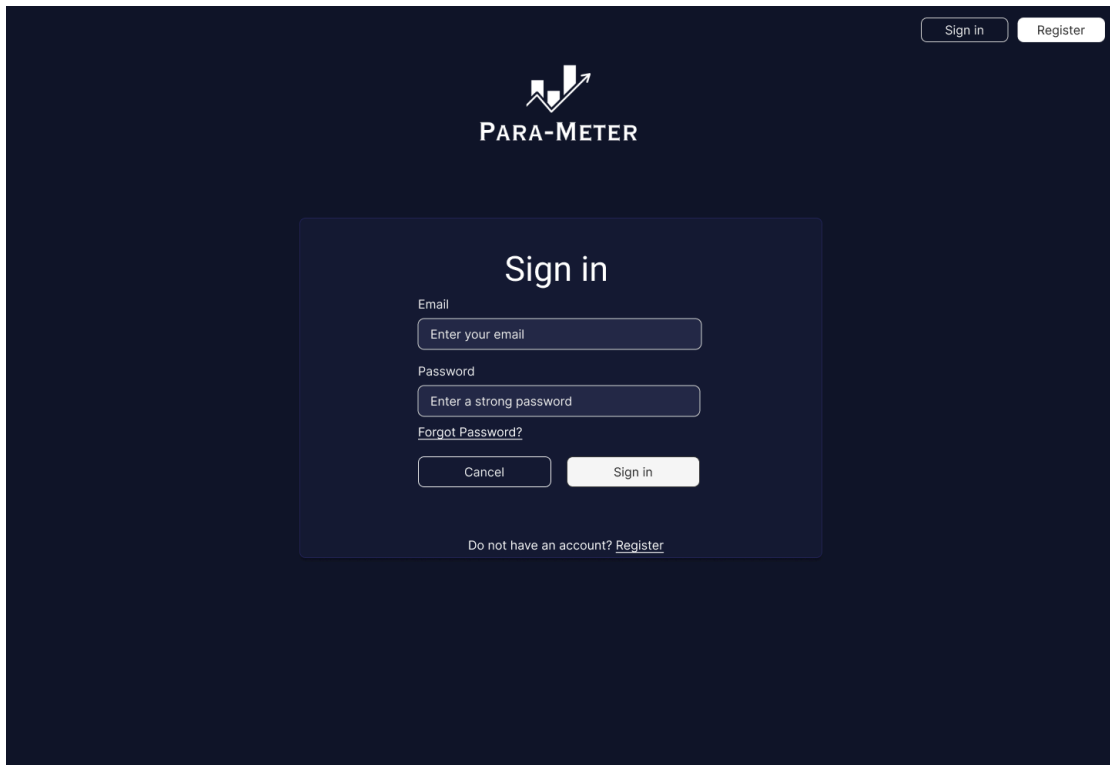


Figure 5. Sign-in Page.

The sign-in page enables existing users to log into the platform:

- **Input Fields:** Email and Password fields for secure authentication.
- **Forgot Password Link:** Allows users to reset their password.
- **Action Buttons:** “Sign In” to proceed to the dashboard page in Figure 6 and “Cancel” to exit.
- **Register Link:** Redirects new users to the registration page.

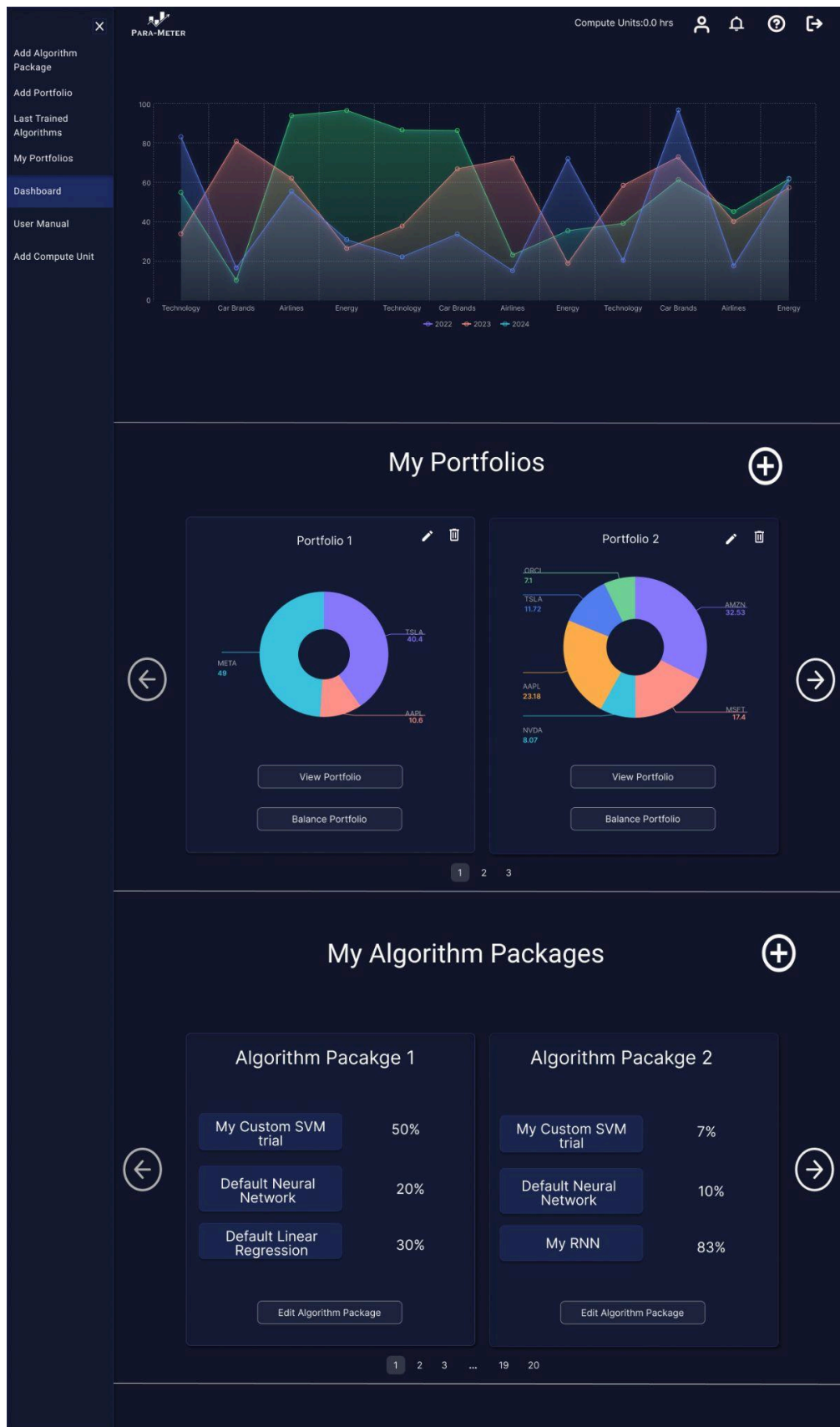


Figure 6. Dashboard.

The dashboard provides a high-level overview of portfolios and algorithm packages:

- **Performance Graph:** Displays asset trends over time.
- **My Portfolios Section:** Lists user portfolios with visual pie charts for asset allocation.
- **My Algorithm Packages:** Shows custom and predefined algorithm packages with percentage contributions.

The screenshot shows the 'Add New Portfolio' modal in the PARA-METER dashboard. The dashboard has a dark theme with a sidebar on the left containing navigation links: 'Add Algorithm Package', 'Add Portfolio' (highlighted), 'Last Trained Algorithms', 'My Portfolios', 'Dashboard', and 'Add Compute Unit'. The top right of the dashboard shows 'Compute Units: 0.0 hrs' and icons for user, notifications, help, and share. The modal itself has a title 'Add New Portfolio' and contains the following fields and options:

- Portfolio Name:** A text input field with the placeholder 'Enter a name for your portfolio'.
- Portfolio Import Method:** A dropdown menu with the placeholder 'Please choose a way to import portfolio'. Below it are four buttons: 'Upload File', 'Use Predefined Examples', and 'Enter Manually'.
- Enter description (optional):** A text input field with the placeholder 'Write your description here'.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom.

Figure 7. Add New Portfolio Page.

PARA-METER

Compute Units: 0.0 hrs

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

My Portfolios

Dashboard

Add Compute Unit

Add New Portfolio

Portfolio Name

Enter a name for your portfolio

Portfolio Import Method

Upload File

Upload Portfolio File

No file chosen

Choose File

Enter description (optional)

Write your description here

Cancel

Save

Figure 8. Add New Portfolio Page when “Upload File” is chosen as the portfolio import method.

This page enables users to create a new portfolio using the “Upload File” method:

- **Portfolio Import Options:** Upload portfolio details in supported formats like CSV.
- **Input Fields:** Includes portfolio name and optional description for user clarity.
- **Save and Cancel Buttons:** Allow users to confirm or exit the process.

Figure 9. Add New Portfolio Page when the “Use Predefined Examples” option is chosen as the portfolio import method.

Here, users can add new portfolios using predefined templates:

- **Portfolio Import Method:** Users select “Use Predefined Examples.”
- **Templates:** Offers options like Balanced Portfolio, Growth Portfolio, and Income Portfolio.
- **Input Fields:** Similar to other methods with portfolio name and optional description.

Figure 10. Add New Portfolio form when the “Enter Manually” option is chosen as the portfolio import method.

This page allows users to add a portfolio manually:

- **Portfolio Import Method:** The “Enter Manually” option enables manual data entry.
- **Asset Entry Section:** Users add individual assets by providing their name, weight, and currency.
- **Action Buttons:** Green checkmark to confirm an asset entry, red cross to cancel.

PARA-METER

Compute Units: 0.0 hrs

×

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

Balance Portfolio

View Portfolio

Dashboard

Add Compute Unit

Add Algorithm Basket

New Algorithm Basket

Algorithm Package Name

Enter a name for your package

Risk Preferences

☐ Conservative ☐ Balanced ☐ Aggressive ☒ Custom

Return Objective

Please choose a return objective

Select Algorithm

Algorithm Package Name

Enter a name for your package

Pretrained(Default) Algorithms

SVM Default epoch:100
C value:0.2 Kernel:Rbf Rolling Window: 14 days

Linear Regression Default
Rolling Window: 14 days

Neural Network Default
1 hidden layer Rolling Window: 14 days

SVM Default
C value:0.2 Kernel:Rbf Rolling Window: 14 days

Custom Algorithms

My Neural Net Predictor
1 hidden layer Rolling Window: 14 days

SVM Default
C value:0.2 Kernel:Rbf Rolling Window: 14 days

Create

Figure 11. Add Algorithm Basket Page.

This page allows users to create and customize algorithm baskets:

- **Algorithm Basket Name:** Users define a name for the basket.
- **Risk Preferences:** Includes Conservative, Balanced, Aggressive, and Custom options.
- **Algorithm Selection:** Users can choose from predefined options (e.g., SVM Default, Neural Network Default) or custom algorithms.

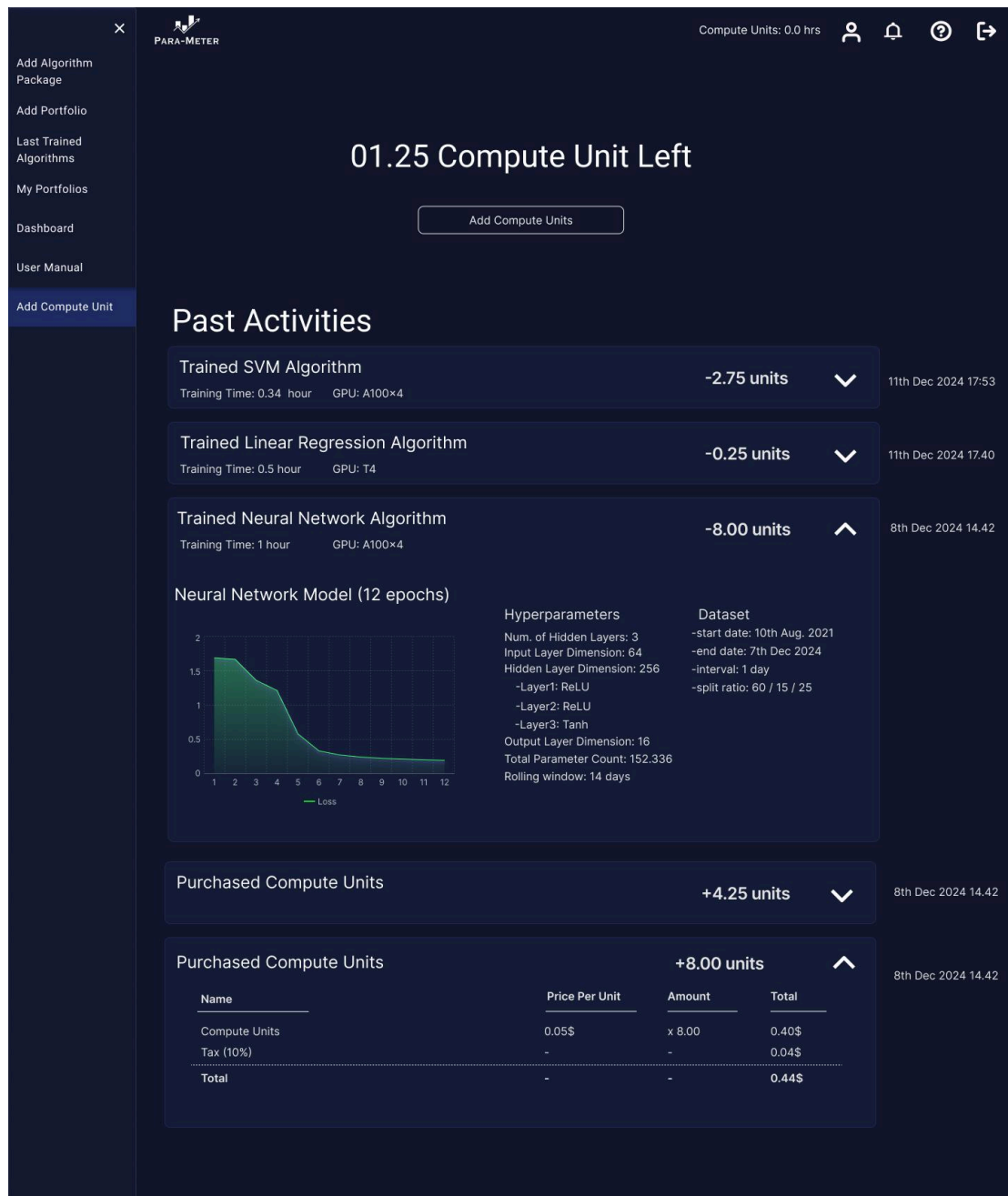


Figure 12. The page that the user will have when they click on the “Add Compute Unit” link on the menu bar.

This screen provides an overview of compute unit usage:

- **Remaining Compute Units:** Displays available compute units in real-time.
- **Past Activities:** Lists recent training activities, including algorithm type, training duration, and compute unit usage.
- **Add Compute Units Button:** Redirects users to the page in Figure 13 to purchase additional compute units.

×

PARA-METER

Compute Units: 0.0 hrs

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

My Portfolios

Dashboard

User Manual

Add Compute Unit

1

2

Add Compute Units

Desired Compute Units

Name	Price Per Unit	Amount	Total
Compute Units	0.05\$	x 8.00	0.40\$
Tax (10%)	-	-	0.04\$
Total	-	-	0.44\$

Cancel

Proceed with payment

Past Activities

Trained SVM Algorithm

Training Time: 1 hour GPU: A100×4

-4.00 units

▼

11th Dec 2024 17:53

Trained Linear Regression Algorithm

Training Time: 0.5 hour GPU: T4

-0.25 units

▼

11th Dec 2024 17:40

Trained Neural Network Algorithm

Training Time: 1 hour GPU: A100×4

-8.00 units

▲

8th Dec 2024 14:42

Neural Network Model (12 epochs)

Hyperparameters

Num. of Hidden Layers: 3
 Input Layer Dimension: 64
 Hidden Layer Dimension: 256
 -Layer1: ReLU
 -Layer2: ReLU
 -Layer3: Tanh
 Output Layer Dimension: 16
 Total Parameter Count: 152.336
 Rolling window: 14 days

Dataset

-start date: 10th Aug. 2021
 -end date: 7th Dec 2024
 -interval: 1 day
 -split ratio: 60 / 15 / 25

Purchased Compute Units

+4.25 units

▼

8th Dec 2024 14.42

Figure 13. Add Compute Unit Page.

23

×

PARA-METER

Compute Units: 0.0 hrs

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

My Portfolios

Dashboard

User Manual

Add Compute Unit

1

2

Payment

Name	Price Per Unit	Amount	Total
Compute Units	0.05\$	x 8.00	0.40\$
Tax (10%)	-	-	0.04\$
Total	-	-	0.44\$

Credit card details

0000 0000 0000 0000

VISA

MM / YYYY

CVC

By providing your card information, you allow us to charge your card for future payments in accordance with their terms.

Pay

Back

Pay

Past Activities

Trained SVM Algorithm

Training Time: 1 hour GPU: A100×4

-4.00 units

11th Dec 2024 17:53

Trained Linear Regression Algorithm

Training Time: 0.5 hour GPU: T4

-0.25 units

11th Dec 2024 17:40

Trained Neural Network Algorithm

Training Time: 1 hour GPU: A100×4

-8.00 units

8th Dec 2024 14.42

Neural Network Model (12 epochs)

Hyperparameters

Num. of Hidden Layers: 3
 Input Layer Dimension: 64
 Hidden Layer Dimension: 256
 -Layer1: ReLU
 -Layer2: ReLU
 -Layer3: Tanh
 Output Layer Dimension: 16
 Total Parameter Count: 152.336
 Rolling window: 14 days

Dataset

-start date: 10th Aug. 2021
 -end date: 7th Dec 2024
 -interval: 1 day
 -split ratio: 60 / 15 / 25

Purchased Compute Units

+4.25 units

8th Dec 2024 14.42

Figure 14. Payment portion of the Add Compute Unit form.

This multi-step interface facilitates purchasing compute units:

- Step 1: Input Desired Units** - Users specify the number of units to purchase.

24

2. **Step 2: Payment Summary** - Displays a breakdown of the cost.
3. **Step 3: Payment Form** - Users input their credit card details to complete the purchase.

The screenshot displays the 'Train Algorithm' interface within the PARA-METER application. The top navigation bar includes a close button, the PARA-METER logo, and a status indicator 'Compute Units: 0.0 hrs' along with user, notification, help, and share icons. A left sidebar lists navigation options: 'Add Algorithm Package' (highlighted), 'Add Portfolio', 'Last Trained Algorithms', 'Balance Portfolio', 'View Portfolio', and 'Dashboard'.

The main content area is titled 'Train Algorithm' and features a four-step progress bar: 1. Algorithm Selection (active), 2. Hyperparameter Selection, 3. Dataset Selection, and 4. Overview.

The 'Algorithm Selection' step is detailed in a central panel. It includes a text input field for 'Algorithm Package Name' with the placeholder 'Enter a name for your package'. Below this, the 'Algorithm Selection' section presents four options: 'Support Vector Machines (SVM)', 'Linear Regression', 'Feed-Forward Neural Network (FFNN)', and 'Recurrent Neural Networks'. The 'Algorithm Information' section below these options contains the text 'Please select an algorithm to see information here.' and a 'Next' button.

Figure 15. Custom Algorithm Training Page - Algorithm Selection.

×

Para-Meter

Compute Units: 0.0 hrs

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

Balance Portfolio

View Portfolio

Dashboard

Train Algorithm

1

2

3

4

Algorithm Selection

Hyperparameter Selection

Dataset Selection

Overview

Algorithm Selection

Algorithm Package Name

Enter a name for your package

Algorithm Selection

Support Vector Machines (SVM)

Linear Regression

Feed-Forward Neural Network (FFNN)

Recurrent Neural Networks

1

2

3

...

19

20

Algorithm Information

Support Vector Machines (SVM)

Purpose: Predicts continuous values while maintaining a balance between precision and simplicity.

How it Works:

- SVR finds the optimal hyperplane that best fits the data while allowing for a tolerance margin (ϵ).
- Data points that fall outside the margin contribute to the model loss.

SVR is ideal for predicting stock prices, trends, and other numerical values where precision is key.

Next

Figure 16. Custom Algorithm Training Page - Algorithm Selection - SVM.

26

×

Para-Meter

Compute Units: 0.0 hrs

Add Algorithm Package

Add Portfolio

Last Trained Algorithms

Balance Portfolio

View Portfolio

Dashboard

Train Algorithm

1

2

3

4

Algorithm Selection

Hyperparameter Selection

Dataset Selection

Overview

Algorithm Selection

Algorithm Package Name

Enter a name for your package

Algorithm Selection

Support Vector Machines (SVM)

Linear Regression

Feed-Forward Neural Network (FFNN)

Recurrent Neural Networks

1

2

3

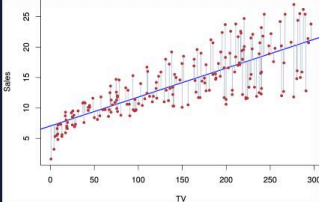
...

19

20

Algorithm Information

Linear Regression



Purpose:

Predicts continuous outcomes by modeling the linear relationship between one or more independent variables (features) and a dependent variable (target).

How it Works:

- Fits a straight line ($y = mx + c$) through the data by minimizing the sum of squared errors (difference between predicted and actual values).
- The slope (m) and intercept (c) are learned from the data.
- Assumes a linear relationship between features and the target.

Simple model, but sensitive to noisy data

Next

Figure 17. Custom Algorithm Training Page - Algorithm Selection - Linear Regression.

27

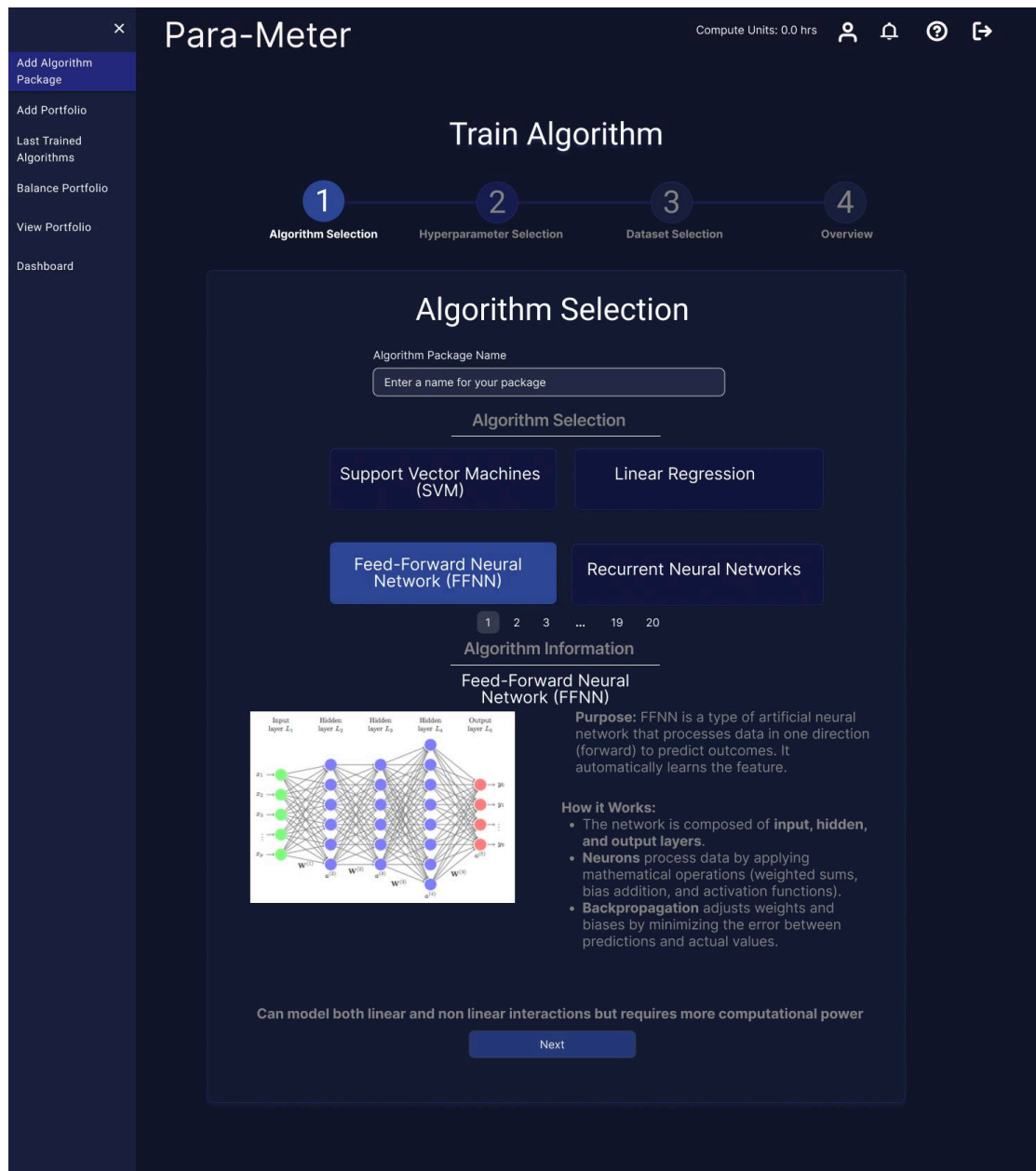


Figure 18. Custom Algorithm Training Page - Algorithm Selection - FFNN.

This step allows users to select an algorithm:

- **Algorithm Options:** Includes SVM, Linear Regression, FFNN, RNN, and others.
- **Algorithm Information:** Provides details about the selected algorithm, such as its purpose and working process.

Para-Meter

Compute Units: 0.0 hrs

Train Algorithm

1 Algorithm Selection 2 Hyperparameter Selection 3 Dataset Selection 4 Overview

Hyperparameter Seleccion

Use Recommended Parameters ✓

Rolling Window: 14 days
Defines the time period of historical data used to train the model.

Prediction Time: 14 days

Neural Network

Number of Hidden Layer: 3

Hidden Layer-1

Activation Function: Tanh

Hidden-Layer Dimension: 256

Hidden Layer-2

Activation Function: ReLU

Hidden-Layer Dimension: 256

Hidden Layer-3

Activation Function: Please chose activation function

Hidden-Layer Dimension: 256

ReLU ⓘ

Sigmoid ⓘ

Tanh ⓘ

Back Next

Figure 19. Custom Algorithm Training Page - Hyperparameter Selection.

Users configure hyperparameters for training:

- **Rolling Window & Prediction Time:** Set the historical period and prediction horizon.
- **Neural Network Configuration:** Adjust layers, activation functions, and dimensions for hidden layers.

The screenshot displays the 'Train Algorithm' interface. At the top, a progress bar indicates the current step is 'Dataset Selection' (step 3 of 4). The left sidebar contains navigation links: 'Add Algorithm Package', 'Add Portfolio', 'Last Trained Algorithms', 'Balance Portfolio', 'View Portfolio', and 'Dashboard'. The main content area is titled 'Train Algorithm' and shows the 'Dataset Selection' modal. This modal includes a 'Select a portfolio' dropdown menu currently showing 'Portfolio 1'. Below this are 'Start Date' and 'End Date' input fields, each with a calendar icon. A calendar for August 2025 is displayed, with the 17th selected. At the bottom of the modal is a 'Partition Dataset' slider and a text instruction: 'Please use the slider to select the amount of data that should be included in the train, test, and validation dataset, respectively.' 'Back' and 'Next' buttons are at the very bottom of the modal.

Figure 20. Custom Algorithm Training Page - Dataset Selection.

This step enables users to select datasets:

- **Portfolio Selection:** Dropdown menu to choose a relevant portfolio.
- **Date Range Picker:** Allows users to define the dataset timeframe.
- **Dataset Partitioning:** Slider for splitting data into training, testing, and validation sets.

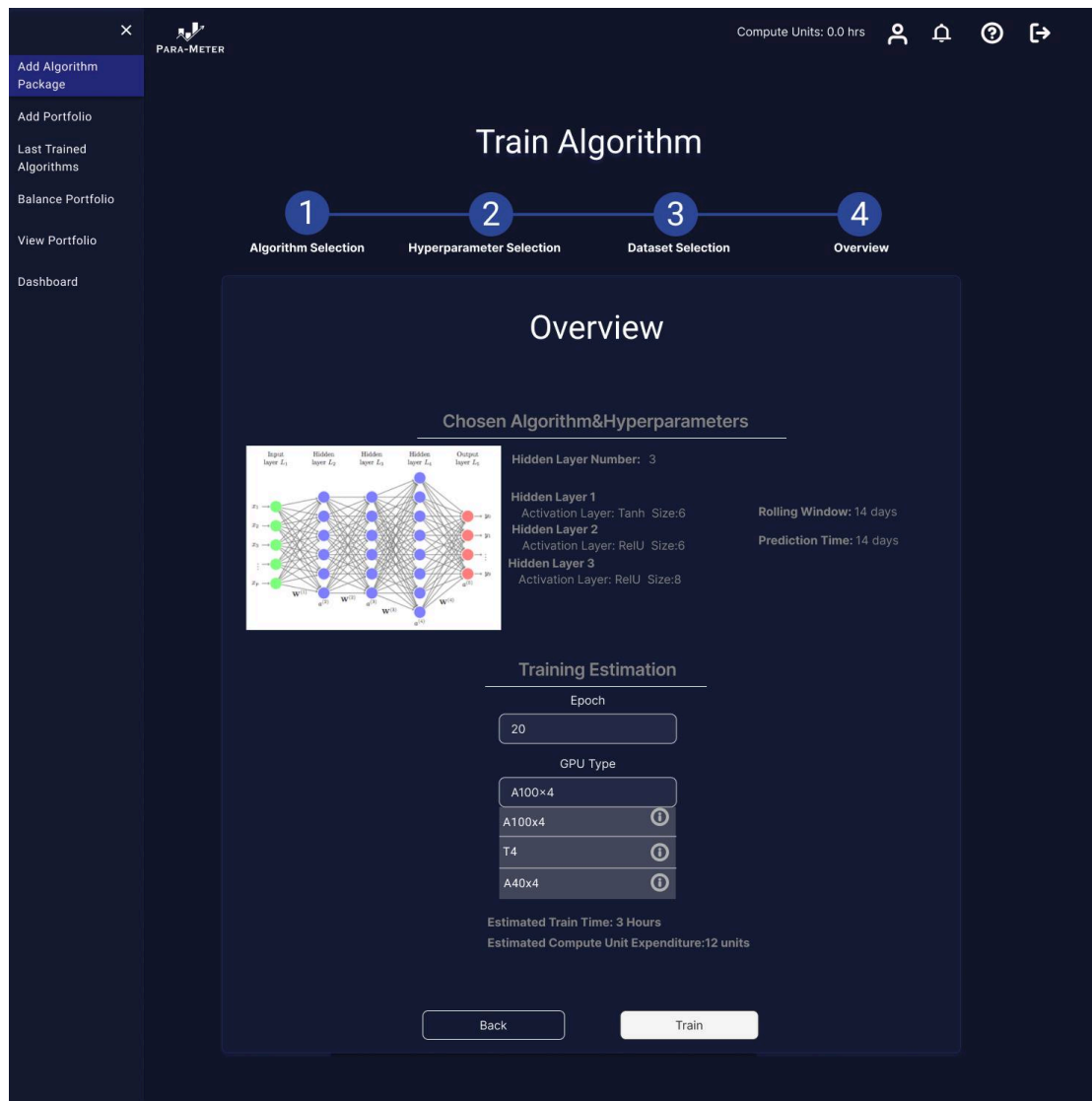


Figure 21. Custom Algorithm Training Page - Overview.

The final step displays a summary of the training configuration:

- **Selected Datasets:** Highlights chosen data sources like yFinance and AccuWeather.
- **Hyperparameter Summary:** Lists algorithm configurations, including activation functions and rolling windows.
- **Training Estimation:** Shows the estimated training time and compute unit usage.

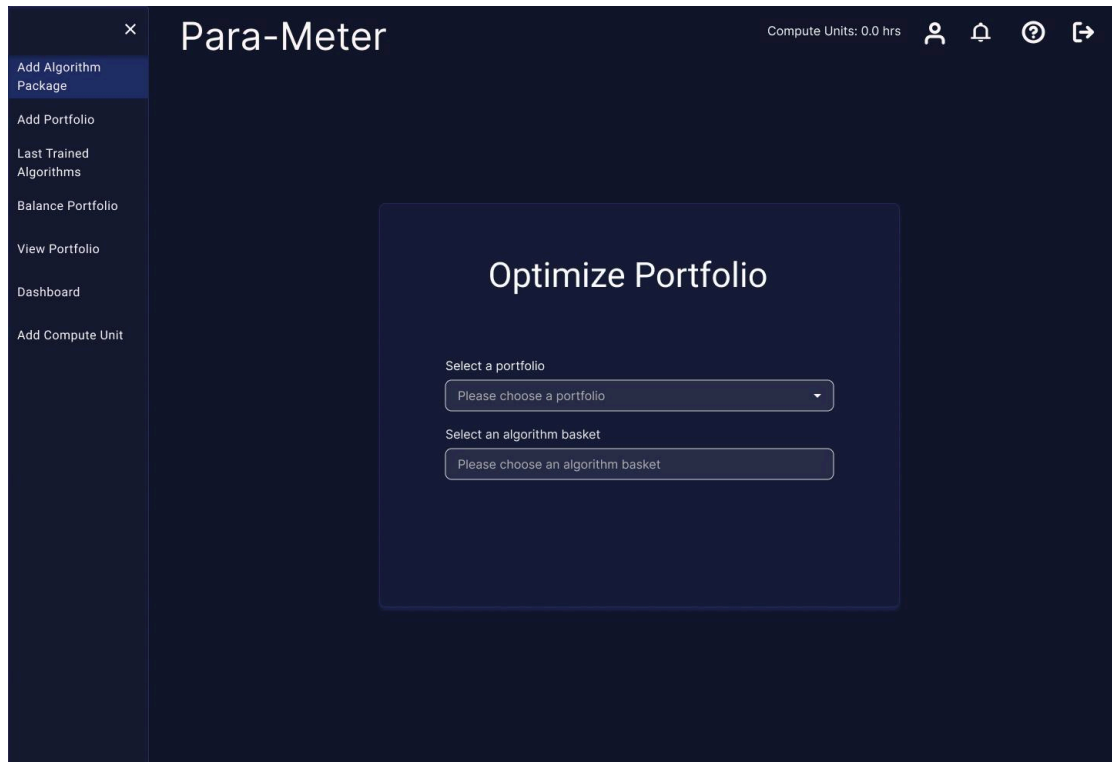


Figure 22. Optimize Portfolio Page.

This screenshot displays the "Optimize Portfolio" page of the Para-Meter. Users can interact with the page to optimize their investment portfolios by selecting a portfolio and an algorithm basket. The interface includes:

- **Dropdown Fields:**
 - **Select a portfolio:** Allows users to choose an existing portfolio from their saved options.
 - **Select an algorithm basket:** Enables users to apply a predefined algorithm or custom algorithms to optimize the selected portfolio.
- **Navigation Panel:** Located on the left, users can access key sections such as "Add Algorithm Package," "Add Portfolio," "Balance Portfolio," and the main "Dashboard."
- **Top Panel:** Displays compute unit usage in hours, user settings, and notification icons.

The design prioritizes simplicity and usability, offering a clean layout that guides users through the portfolio optimization process.



Figure 23. My Algorithms Page.

This page shows a repository of algorithms created or trained by the user:

- **Algorithm List**: Displays algorithms such as “Custom SVM” and “My Neural Network.”
- **Training Progress**: Includes status bars for ongoing training processes.
- **Hyperparameters and Results**: Users can view algorithm-specific configurations and backtest results.

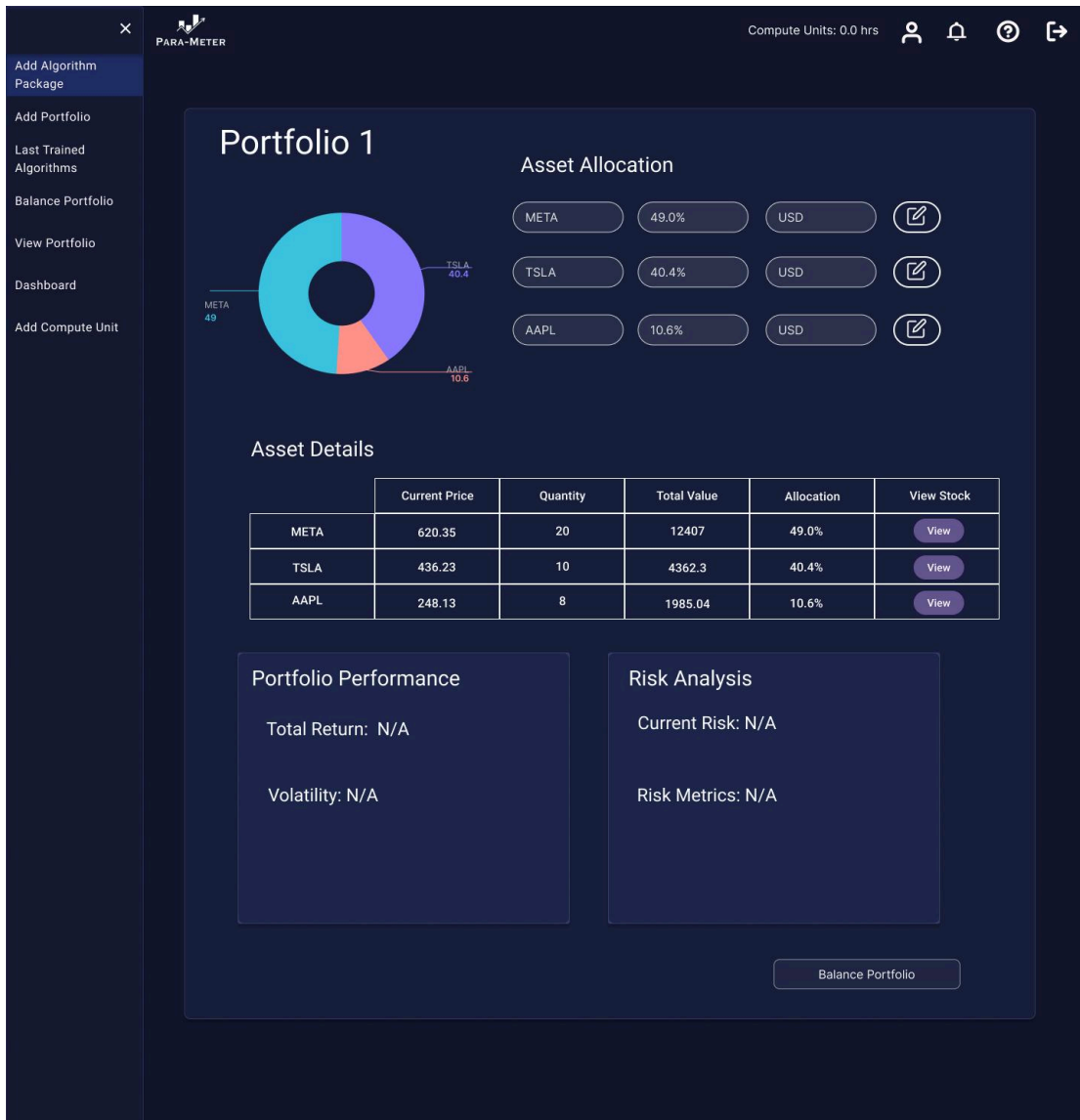


Figure 24. View Portfolio Page for a single portfolio.

The "View Portfolio" page provides a detailed analysis of user portfolios:

- **Asset Allocation Chart:** A pie chart visualizes the distribution of assets like META, TSLA, and AAPL.
- **Asset Details Table:** Displays asset-specific details, including current price, quantity, and total value.
- **Portfolio Performance and Risk Analysis:** Shows metrics like Total Return, Volatility, and Risk.
- **Balance Portfolio Button:** Allows users to rebalance their portfolio based on current data.

4 Other Analysis Elements

4.1 Consideration of Various Factors in Engineering Design

4.1.1 Constraints

4.1.1.1 Implementation Constraints

- Accessing reliable and high-quality financial data is important for accurate modeling and optimization. However, many financial APIs have rate limits or limited historical data. Also, the data might be inaccurate or incomplete. These might hinder the accuracy of the results.
- Training machine learning models and running optimization algorithms require significant computational resources, including processing power and memory. The lack of access to high-performance hardware, such as GPUs or distributed computing systems, can slow down model training and testing, particularly when working with large datasets or performing hyperparameter tuning.
- The limited development timeline of 8 months can restrict the scope of development and testing. This limited duration requires careful prioritization of features and functionalities to ensure the delivery of a functional, high-quality system within the allotted time.

4.1.1.2 Financial Constraints

- The development of a machine learning-based portfolio optimization tool faces significant financial limitations. The initial budget for the project restricts access to advanced cloud infrastructure and premium financial datasets. Cloud services, which are essential for computation and storage, incur substantial costs, especially when utilizing high-performance configurations like GPUs or distributed systems. Similarly, high-quality financial datasets, often critical for accurate modeling, can be expensive and may exceed budgetary limits. These constraints could affect the accuracy and speed of the system's development.
- Operational costs are another concern, as the platform aims to remain affordable for retail investors who are sensitive to pricing. Maintaining low operational expenses is important to ensure the tool is accessible, which may necessitate adopting a freemium or tiered pricing model. Cost-effective development practices, such as using open-source tools and optimizing resource usage, will be critical to staying within budget.

4.1.1.3 Ethical Constraints

- A primary ethical concern is user data privacy. Compliance with data protection regulations such as GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act), and KVKK (Personal Data

Protection Law) is crucial since the tool handles sensitive financial and personal information.

- Avoiding algorithmic bias is one of the most important ethical constraints for Para-Meter. Machine learning models trained on financial data may unintentionally favor certain asset classes, industries, or demographic groups, leading to biased recommendations. Ensuring fairness in portfolio suggestions requires diverse, high-quality training datasets, which can be hard to detect and obtain.

4.1.1.4 Public Health, Safety, and Welfare Factors

- **Impact Level:** 3/10
Although the system does not have a direct impact on public health and safety, it does contribute to broader welfare by empowering users to make informed financial decisions.
- **Financial Well-being:** By enabling users to optimize their investment portfolios and better manage risk, the system indirectly contributes to the financial well-being of individuals. As people become more informed and make smarter investment choices, it could improve their overall financial security, supporting their long-term welfare.
- **Accessibility to Financial Tools:** By providing a user-friendly and affordable tool, the system helps democratize access to financial optimization tools that were previously only available to institutional investors. This can improve the financial outcomes of individuals who might otherwise not have had access to advanced investment strategies.

4.1.1.5 Global and Cultural Factors

- **Impact Level:** 4/10
Global and cultural factors play an important role in shaping how the system is used across different regions and by diverse groups of investors.
- **Market Accessibility:** The system will focus on ensuring seamless access to the most widely traded stocks across different regions. It must consider any regional restrictions, such as access to foreign markets, to allow investors from any region to build diversified portfolios without being limited to local stocks.

4.1.1.6 Economic Factors

- **Impact Level:** 10/10
The system must account for various economic considerations to ensure it is accessible, efficient, and sustainable in the long term.
- **Affordability for Retail Investors:** The platform must remain affordable, particularly for retail investors, who are typically sensitive to costs. This

necessitates a pricing model that balances feature availability with affordability. A freemium or tiered pricing model would allow users to access essential functionalities while offering premium features for more advanced users or institutional clients.

- **Operational Costs and Cloud Infrastructure:** High-performance cloud computing and financial data APIs are crucial for training machine learning models and executing real-time optimization. However, using advanced cloud infrastructure such as GPUs, distributed computing systems, and premium data sources can significantly increase costs. The system must find a cost-effective solution that balances computational needs with budgetary constraints.
- **Economic Sensitivity of Market Conditions:** The platform must adjust its models to reflect changing economic conditions, such as market volatility, interest rates, and inflation. These economic factors can impact the performance of investment portfolios and must be integrated into the system's predictive models to ensure that portfolio optimizations remain relevant under different economic climates.

Table 1. Factors that can affect analysis and design.

	Effect level	Effect
Public health	0	N/A - The project has no direct impact on public health.
Public safety	0	N/A - The project has no direct impact on public safety.
Public welfare	3	The project indirectly contributes to financial well-being by helping users optimize their investments.
Global Factors	4	The project ensures accessibility to global markets and compliance with international regulations.
Cultural factors	0	N/A - The project has no direct impact on cultural factors.

Social factors	5	The project democratizes access to advanced investment tools, fostering inclusivity and financial literacy.
Environmental factors	1	Minimal impact due to hosting infrastructure and cloud service usage.
Economic factors	10	The project heavily focuses on economic considerations, especially affordability for retail investors.

4.1.2 Standards

- **IEEE 830:** For the requirements specification of this project, we will use IEEE 830 standards.
- **UML 2.5.1:** For the UML diagrams like use case diagrams, activity diagrams, and class diagrams, UML 2.5.1 will be used.
- **OAuth2.0:** We will use the industry-standard authentication protocol OAuth2.0 to authenticate the users.
- **ISO 27002:** For security-related standards, we will use ISO standards to comply with many regulations regarding data security.

4.2 Risks and Alternatives

4.2.1 Data Inaccuracy and Quality Risks

Risk: The accuracy of the stock market data used for portfolio optimization is critical to the success of the system. Financial data sourced from external APIs or financial data providers may be inaccurate, outdated, or incomplete, leading to poor investment recommendations. Inaccurate data can arise due to issues such as incorrect pricing, missing data points, or delays in updating market information. This could result in suboptimal portfolio suggestions, misleading users in making important financial decisions.

Alternatives:

- **Data Validation with Basic Checks:** We can implement basic checks to ensure the data is valid. For example, flag any stock prices listed as zero or negative and identify missing or incomplete data. Missing data can be removed or imputed using basic methods such as replacing it with the mean or median.

- **Utilizing Free or Open Data Sources:** Initially, we can leverage publicly available and reputable data sources, such as Yahoo Finance or Alpha Vantage. Although these sources may have some limitations, they can still provide sufficient quality data for the early stages of the project.

4.2.2 Computational Overhead

Risk: The use of machine learning algorithms and portfolio optimization techniques, especially when processing large datasets or running huge simulations, requires significant computational resources. These computational demands could result in long processing times, affecting the responsiveness of the system.

Alternatives:

- **Optimizing Code Efficiency:** We can focus on writing optimized, efficient code by using techniques such as vectorization or precomputing values to avoid redundant calculations.
- **Leveraging Free Cloud Computing Resources:** We can utilize free-tier cloud services, such as Google Cloud or AWS, which provide basic computational resources at no cost to students. These services are suitable for running initial tests and experiments.

4.2.3 Legal and Regulatory Risks

Risk: Para-Meter deals with sensitive financial data, and there may be legal and regulatory requirements around data privacy, financial recommendations, and investment advice. Laws such as the GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act), and specific financial regulations across different regions may apply. Non-compliance with these regulations could expose us to significant legal liabilities and reputational damage.

Alternatives:

- **Simplifying Data Collection:** We can minimize the collection of sensitive personal data. We can collect only the essential information required for the tool to function, such as portfolio size and investment goals, and avoid storing personally identifiable financial data.
- **Disclaimers and Transparency:** We can clearly state in the platform's terms of service that the tool provides suggestions only and is not responsible for making actual investment decisions. This will help reduce the risk of legal liability.

4.2.4 Security Risks

Risk: Para-Meter stores sensitive personal and financial information, making it a target for cyberattacks. Potential security vulnerabilities, such as unauthorized access, data breaches, and cyber threats, could compromise user data.

Alternatives:

- **Using Secure Local Databases:** We can store user data in secure, local databases such as SQLite or PostgreSQL.

4.2.5 User Adoption and Usability Risk

Risk: The complexity of machine learning-based portfolio optimization may deter less experienced users from adopting the platform. A steep learning curve, technical jargon, or an unintuitive user interface could limit the system's reach to a broader audience of retail investors.

Alternatives:

- **Incorporating In-App Tutorials:** We can add basic tooltips and step-by-step guides to assist users in understanding how to use the platform effectively.
- **Simplified Models:** We can offer pre-configured portfolio templates (e.g., Balanced, Growth) for quick setup and gradually introduce advanced features through progressive disclosure.

Table 2. Risks.

	Likelihood	Effect on the project	B Plan Summary
Data Inaccuracy and Quality Risks	Likely	This could lead to suboptimal portfolio suggestions, potentially misleading users.	Implementing data validation checks, using open data sources, and implementing caching for API rate limits.

Computational Overhead	Moderate	This may lead to long processing times, especially with large datasets, affecting system responsiveness.	Optimizing code efficiency and using free-tier cloud computing resources.
Legal and Regulatory Risks	Moderate	Non-compliance with data privacy and financial regulations could lead to legal liabilities.	Including legal disclaimers and simplifying data collection.
Security Risks	Low	Data breaches or unauthorized access could compromise user data.	Using secure local databases.
User Adoption and Usability Risks	Likely	The complexity of the platform could deter users, limiting the system's reach and adoption, especially for retail investors.	Incorporating In-App Tutorials

4.3 Project Plan

Table 3. List of work packages.

WP#	Work package title	Leader	Members involved
WP1	System Design and Architecture	Tuna	Muti
WP2	Data Collection and Preprocessing	Tuna	Sıla
WP3	Machine Learning Model Development	Muti	Tuna

WP4	Learning and Developing Financial Mathematical Models	Samed	Muti
WP5	User Interface and Frontend Development	Sila	Tuna
WP6	Backend Development	Muti	Samed
WP7	Integration and Testing	Sila	Samed
WP8	Documentation and Final Presentation	Samed	Sila

Table 4. Gantt Chart For Work Packages.

Project Conception and Initiation	Sept. 2024	Oct. 2024	Nov. 2024	Dec. 2024	Jan. 2025	Feb. 2025	March 2025	Apr. 2025	May 2025
System Design and Architecture									
Data Collection and Preprocessing									
Machine Learning Model Development									
Learning and Developing Financial Mathematical Models									
User Interface and Frontend Development									
Backend Development									
Integration and Testing									
Documentation and Final Presentation									

WP 1: System Design and Architecture			
Start date: 2 November 2024 End date: 10 February 2025			
Leader:	<i>Tuna</i>	Members involved:	<i>Muti</i>
Objectives: <i>Creating a scalable and modular system design while adhering to engineering standards.</i>			
Tasks: Task 1.1: Design system architecture (frontend-backend communication, data flow). Task 1.2: Create UML diagrams: Use-case diagrams for user scenarios. Class diagrams for backend components. Sequence diagrams for key workflows. Task 1.3: Define API specifications for inter-module communication. Task 1.4: Establish database schema design. Task 1.5: Review the architecture for compliance with standards (UML 2.5.1, ISO 27002).			
Deliverables <i>D1.1: System Architecture Diagram.</i> <i>D1.2: UML Diagrams.</i> <i>D1.3: Database Schema Design.</i>			

WP 2: Data Collection and Preprocessing			
Start date: 2 November 2024 End date: 15 February 2025			
Leader:	<i>Tuna</i>	Members involved:	<i>Sila</i>

Objectives: Setting up the data pipeline and ensuring clean, accurate, and reliable input for ML models.

Tasks:

Task 2.1: Investigate and integrate external APIs (e.g., Yahoo Finance, Alpha Vantage).

Task 2.2: Create a pipeline for data retrieval.

Task 2.3: Develop routines for data validation and cleaning.

- Handle missing data through imputation techniques.
- Detecting and removing outliers.

Task 2.4: Perform normalization (e.g., Min-Max scaling, Z-score normalization).

Task 2.5: Save cleaned datasets for further use and testing.

Deliverables

D2.1: Data Retrieval Pipeline.

D2.2: Cleaned and Processed Dataset.

WP 3: Machine Learning Model Development

Start date: 12 December 2024 **End date:** 31 March 2025

Leader:

Muti

**Members
involved:**

Tuna

Objectives: Training and validating machine learning models for portfolio optimization.

Tasks:

Task 3.1: Research and select ML algorithms to implement (e.g., Random Forest, Neural Networks, SVM).

Task 3.2: Implement initial versions of ML algorithms.

Task 3.4: Develop training scripts with GPU optimization.

Task 3.5: Perform validation using cross-validation and rolling window testing.

Task 3.6: Integrate ensemble methods for combining predictions.

Deliverables

D3.1: ML Model Implementations.

D3.2: Model Training and Performance Reports.

WP 4: Learning and Developing Financial Mathematical Models

Start date: 15 October 2024 **End date:** 15 March 20254

Leader:

Samed

**Members
involved:**

Muti

Objectives: Understanding and implementing financial mathematical models such as risk-return optimization, Value at Risk (VaR), Sharpe ratio calculations, and portfolio rebalancing strategies, ensuring alignment with modern financial theories..

Tasks:

Task 4.1: Research fundamental financial theories and models:

- Study Markowitz's Mean-Variance Optimization.
- Understand modern approaches to risk-adjusted returns like Sharpe and Sortino Ratios.
- Explore portfolio rebalancing strategies and constraints.

Task 4.2: Identify practical applications in portfolio management:

- How these models are used in institutional portfolio tools.
- Application in both long-only and long-short portfolios.

Task 4.3: Learn mathematical and computational implementations:

- Develop formulas for portfolio variance and covariance.
- Implement Monte Carlo simulations for stress testing.
- Calculate VaR and Expected Shortfall using statistical methods.

Task 4.4: Implement foundational models as proof of concept:

- Build small-scale models to test risk-return optimizations.

<ul style="list-style-type: none"> Compare these models with basic machine learning models to establish a performance benchmark. <p>Task 4.5: Integrate financial models into the system’s pipeline:</p> <ul style="list-style-type: none"> Combine financial mathematical models with machine learning outcomes for hybrid optimization strategies.
<p>Deliverables</p> <p>D4.1: Summary Report on Financial Theories and Applications.</p> <p>D4.2: Implemented Risk-Return and Portfolio Optimization Scripts.</p> <p>D4.3: Hybrid Model Report (Comparison of ML and Mathematical Models).</p>

WP 5: <i>User Interface and Frontend Development</i>			
Start date: <i>1 December 2024</i> End date: <i>28 February 2025</i>			
Leader:	<i>Sila</i>	Members involved:	<i>Tuna</i>
Objectives: <i>Developing an intuitive and accessible interface for users.</i>			
<p>Tasks:</p> <p>Task 5.1: Design UI wireframes and prototypes.</p> <p>Task 5.2: Build core frontend components (React.js framework).</p> <ul style="list-style-type: none"> Registration and login pages. Dashboard for portfolio visualization. Forms for data uploads and algorithm selection. <p>Task 5.3: Add visualization components (graphs, charts for metrics).</p> <p>Task 5.4: Integrate the frontend with backend APIs.</p>			
<p>Deliverables</p> <p>D5.1: <i>UI Wireframes and Mockups.</i></p> <p>D5.2: <i>Functional Frontend Components.</i></p>			

WP 6: Backend Development			
Start date: 15 December 2024 End date: 31 March 2025			
Leader:	Muti	Members involved:	Samed
Objectives: Implementing backend services and APIs for seamless data and model handling.			
Tasks: Task 6.1: Implement authentication and authorization (OAuth 2.0). Task 6.2: Build profile and portfolio management APIs (CRUD operations). Task 6.3: Create ML model inference and training endpoints. Task 6.4: Optimize the backend for high performance and low latency.			
Deliverables D6.1: Functional Backend APIs. D6.2: Backend Testing Reports.			

WP 7: Integration and Testing			
Start date: 1 April 2025 End date: 1 May 2025			
Leader:	Sila	Members involved:	Samed
Objectives: Integrating system modules and ensuring seamless operation through comprehensive testing.			
Tasks: Task 7.1: Integrate backend, frontend, and ML modules. Task 7.2: Conduct unit, integration, and system testing. Task 7.4: Debug and resolve identified issues.			

Deliverables

D7.1: Test Results and Bug Fix Logs.

D7.2: Fully Integrated System.

WP 8: Documentation and Final Presentation

Start date: 1 May 2025 **End date:** TBA

Leader:

Samed

**Members
involved:**

Sila

Objectives: *Creating comprehensive documentation and present the project.*

Tasks:

Task 8.1: Write user manual and technical documentation.

Task 8.2: Prepare final presentation slides.

Task 8.3: Conduct internal mock presentations for feedback.

Task 8.4: Deliver the final presentation to instructors.

Deliverables

D8.1: User Manual.

D8.2: Final Project Documentation.

D8.3: Presentation Slides.

4.4 Ensuring Proper Teamwork

- To establish proper teamwork, we decided to have stand-up meetings every Monday at 17.30 or 18.00. If some of our teammates are unavailable that day, we will try to find another day that suits everyone.
- At the end of each stand-up meeting, we decide on the task distribution for the upcoming report or task.
- Utilized Jira for centralized task management/allocation.

- GitHub is used for version control. That way, we make sure that multiple teammates can work on the code for both the backend and the frontend.

4.5 Ethics and Professional Responsibilities

- Transparency in model behavior is crucial for Para-Meter. Machine learning algorithms often function as black boxes, which can deteriorate user trust. The platform must include explainable AI techniques, allowing users to understand how specific inputs, like financial indicators or alternative data, impact portfolio recommendations. This transparency builds trust and ensures that users can make informed decisions.
- Para-Meter must maintain impartiality to avoid conflicts of interest. For instance, it should not prioritize certain assets or datasets due to partnerships or financial incentives. Ensuring neutrality in data sourcing and recommendation processes is crucial for maintaining user trust and meeting professional standards.
- We acknowledge that the financial market is inherently unpredictable and volatile. While Para-Meter leverages state-of-the-art techniques to provide optimized portfolio suggestions, these recommendations are not guaranteed to lead to financial gains. Therefore, it is essential to emphasize that the ultimate responsibility for any financial decision lies with the user. Our application is intended to serve as a decision-support tool, not a substitute for professional financial advice. We explicitly disclaim any liability for financial losses resulting from acting on recommendations provided by Para-Meter. The performance of portfolios depends on numerous external factors, which are beyond the scope of Para-Meter. Users agree to accept full responsibility for their financial decisions by using it.

4.6 Planning for New Knowledge and Learning Strategies

Developing a comprehensive and innovative portfolio optimization tool like Para-Meter requires acquiring and applying new knowledge and skills throughout the project lifecycle. This section outlines the key areas where the team needs to expand its expertise, the strategies for acquiring this knowledge, and how it will be applied to the system.

4.6.1 Key Areas for New Knowledge

The project involves a multidisciplinary approach, which makes learning important in the following domains:

- **Machine Learning Algorithms:**
 - Advanced techniques like ensemble learning, deep learning (e.g., feed-forward neural networks, LSTMs), and explainable AI (e.g., SHAP values).

- Hyperparameter optimization methods such as grid search, random search, and Bayesian optimization.
- **Financial Mathematics:**
 - Core financial models, including Markowitz's Mean-Variance Optimization, Value at Risk (VaR), Sharpe and Sortino Ratios, and Monte Carlo simulations.
 - Understanding the principles of portfolio rebalancing and risk management.
- **Data Engineering and Preprocessing:**
 - Handling large-scale financial datasets, dealing with missing or noisy data, and preprocessing alternative datasets like sentiment analysis or satellite data.
- **Scalable Software Development:**
 - Learning advanced backend technologies like containerization with Docker, API design, and integration with scalable cloud platforms (e.g., AWS, Azure, Google Cloud).
- **User Experience and Interface Design:**
 - Developing intuitive, responsive interfaces using modern frontend frameworks like React.js.
 - Ensuring usability for non-technical retail investors through effective visualizations and interactivity.
- **Compliance and Security Standards:**
 - Understanding GDPR, CCPA, and ISO 27002 for data protection and secure handling of sensitive financial data.

4.6.2 Learning Strategies

We plan to acquire the necessary knowledge using a combination of approaches:

- **Online Courses and Tutorials:** Platforms such as Coursera, edX, and YouTube for structured courses on machine learning, financial mathematics, and scalable software development.
- **Reading and Literature Review:** We plan to use books and articles that are stated in the [Project Specification Document](#).
- **Peer Learning and Knowledge Sharing:** We can arrange weekly knowledge-sharing sessions within the team to discuss progress and challenges.
- **Consulting with Experts:** Seeking mentorship and feedback from our supervisor Selim Aksoy, course instructors Mert Bıçakçı and Atakan Erdem, and innovation expert Eren Biri for financial modeling, advanced machine learning techniques, and software development.

5 Glossary

SVM: support vector machines

SVR: support vector regression

FFNN: feed-forward neural network

RNN: recurrent neural network

6 References

[1] "Strategy," *Refactoring.Guru*. [Online]. Available:
<https://refactoring.guru/design-patterns/strategy>. [Accessed: 14-Dec-2024]

[2] GeeksforGeeks, "Repository design pattern," *GeeksforGeeks*, 01-Nov-2024. [Online].
Available: <https://www.geeksforgeeks.org/repository-design-pattern/>. [Accessed:
14-Dec-2024]