Bilkent University
Department of Computer Engineering

Senior Design Project
T2438
Para-Meter

Detailed Design 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

10.03.2025

Contents

1. Introduction

This section explains the Para-Meter project's functionality, purpose, and goal, along with the explanations and definitions of technical terms that will be used in the report.

1.1. Purpose of the System

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.

1.2. Design Goals

Our design goals are as follows:

Usability

- Para-Meter should have an intuitive UI that is accessible to non-technical users and has 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 system must be designed to deliver low-latency portfolio optimization and simulations through efficient parallelization techniques. Performance optimization extends to model training, backtesting, and real-time analytics to ensure responsive user interactions even when processing large datasets or complex calculations.

Supportability

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

Scalability

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

Marketability

- Para-Meter must differentiate itself from competitors by combining advanced machine learning capabilities with odd customization options and user accessibility. The platform's ability to allow users to import alternative datasets and train custom algorithms should create a unique value proposition in the market.

Flexibility

- The system should allow for extensive customization, enabling users to select from various machine learning models, integrate alternative data sources, create custom features, and define precise risk and return parameters according to their investment strategy and goals.

1.3. Definitions, Acronyms, and Abbreviations

- **AI**: Artificial Intelligence
- **API**: Application Programming Interface
- **CCPA**: California Consumer Privacy Act
- **GDPR**: General Data Protection Regulation
- **GPU**: Graphics Processing Unit
- **KVKK**: Personal Data Protection Law (Turkish data protection regulation)
- **ML**: Machine Learning
- **OAuth2.0**: Industry-standard protocol for authorization
- **UI**: User Interface
- **UML**: Unified Modeling Language

1.4. Overview

The Para-Meter system introduces a comprehensive portfolio optimization solution that integrates machine learning capabilities with financial expertise to deliver personalized investment strategies. The platform is built on a modular architecture that separates concerns into distinct services for data retrieval, preprocessing, algorithm training, portfolio optimization, and user interface components.

This report details the transformation of our analytical model into a functional system design. It begins by outlining the high-level system architecture and describes the subsystem decomposition, hardware/software mappings, and data management strategies. The report

then addresses integration aspects, including access control, control flow, and the handling of boundary conditions.

Subsequent sections elaborate on comprehensive testing strategies—both functional and non-functional—designed to validate system performance, security, stability, and reliability. Additionally, we detail the incorporation of various engineering considerations such as public health, safety, security, global, cultural, social, environmental, and economic factors. A dedicated chapter on teamwork discusses individual contributions, collaborative practices, and leadership sharing within the project team.

Overall, this introduction sets the stage for the detailed design and testing procedures that follow, providing a clear roadmap for stakeholders to understand the purpose, structure, and guiding principles behind Para-Meter.

2. Current Software Architecture

This section examines the prevailing software architectures in portfolio optimization, focusing on institutional-grade systems and retail-focused alternatives. By understanding the design paradigms, integration strategies, and inherent limitations of existing solutions, we can clearly identify the gaps that Para-Meter is engineered to fill.

2.1. Institutional Solutions

Overview:
Institutional systems—such as BlackRock-Aladdin and SimCorp-Axioma—are designed for large-scale financial organizations that demand extensive functionality, rigorous risk management, and high-performance processing.

Architectural Characteristics:

- **Enterprise-Grade Infrastructure:**
  These systems typically employ multi-tiered architectures incorporating centralized data management, dedicated risk analysis modules, and robust integration layers. Proprietary frameworks ensure high throughput and low latency even during heavy computational loads.

- **Complex Data Integration:**
  Institutional tools are built to ingest and process extensive market data from various sources. They integrate traditional financial models with real-time analytics by combining batch processing with live data feeds for continuous risk assessment.

- **Closed-Box Design:**
  While these platforms offer a wide range of advanced features, their closed architectures restrict customization. This limitation makes it challenging to adapt the system to alternative data sources or to incorporate custom machine-learning models without significant technical overhead.

Limitations:

- **Usability:**
  These systems' complexity and steep learning curve can deter less technically proficient users, particularly retail investors.

- **Flexibility:**
  Rigid, monolithic designs inhibit rapid adaptation and the integration of new algorithms or alternative data sources.

- **Cost:**
  The high implementation and ongoing maintenance costs make these solutions largely inaccessible to smaller institutions and individual investors.

2.2. Retail and Alternative Solutions

Overview:
Retail-focused portfolio optimization tools—such as Magnus and Talos—are designed with simplicity and ease of use as their primary goals, targeting individual investors or small firms that require a streamlined experience.

Architectural Characteristics:

- **Simplified Design:**
  These solutions typically adopt a monolithic or loosely coupled architecture focused on delivering a user-friendly interface. The emphasis is on simplifying the portfolio creation process, risk assessment, and performance tracking.

- **Limited Customization:**
  While offering essential functionalities, retail systems often lack the flexibility to integrate user-specific datasets or support the training of custom machine learning models. Users generally work within a predefined set of algorithms and parameters.

- **Basic Data Processing:**
  Underlying these tools are traditional models—such as mean-variance optimization—that provide straightforward outputs. They are built to handle fundamental data processing tasks without employing advanced interpretability techniques.

Limitations:

- **Scalability:**
  Simplified architectures may struggle to efficiently process larger datasets or handle many concurrent user requests during peak periods.

- **Modularity:**
  The absence of a modular design limits the ability to integrate future enhancements, such as additional algorithm options or real-time data feeds.

- **Innovation:**
  By focusing on simplicity, these systems often miss opportunities to improve

optimization accuracy and offer greater customization for the user.

2.3. Analysis and Opportunities

Comparative Analysis:

- **Integration of Alternative Data:**
  Institutional systems can handle extensive data streams but often sacrifice user accessibility. In contrast, retail solutions prioritize ease of use but may not fully leverage alternative data sources, such as economic indicators or social sentiment.

- **Customization and Transparency:**
  There is a clear trade-off between the robust, closed designs of institutional tools and the limited, predefined nature of retail solutions. Few current systems provide a balance supporting user-driven customization and clear, understandable outputs.

- **Modular and Scalable Architecture:**
  Existing solutions generally do not offer a modular framework that supports incremental updates or easy integration of new components. This shortcoming limits the ability to innovate rapidly in response to evolving market needs.

Opportunities for Para-Meter:

- **Hybrid Approach:**
  Para-Meter aims to blend the computational strengths of institutional systems with the simplicity and user accessibility of retail solutions. This approach allows users to integrate alternative datasets and customize their portfolio optimization process within a realistic and manageable scope.

- **Modularity and Extendibility:**
  By adopting a modular architecture, Para-Meter ensures that individual subsystems—from data retrieval and preprocessing to model training and portfolio optimization—are developed as independent, interchangeable components. This design streamlines initial development and paves the way for future enhancements.

- **User-Centric Design:**
  The system is built with a strong focus on usability. Para-Meter will provide clear, straightforward outputs and robust risk analysis without relying on advanced techniques that exceed our current capabilities. This ensures that the solution remains both accessible to users and achievable within our project constraints.

In summary, current software architectures in portfolio optimization tend to favor high complexity or simplicity but rarely provide a balanced solution that meets all user needs. Para-Meter is strategically positioned to fill this gap by offering a novel, flexible, and scalable design that bridges the divide between institutional and retail offerings while staying within the practical limits of our project.

3. Proposed Software Architecture

This section will depict Para-Meter's software architecture in diagrams with different domains such as storage, hardware software mapping, and security.

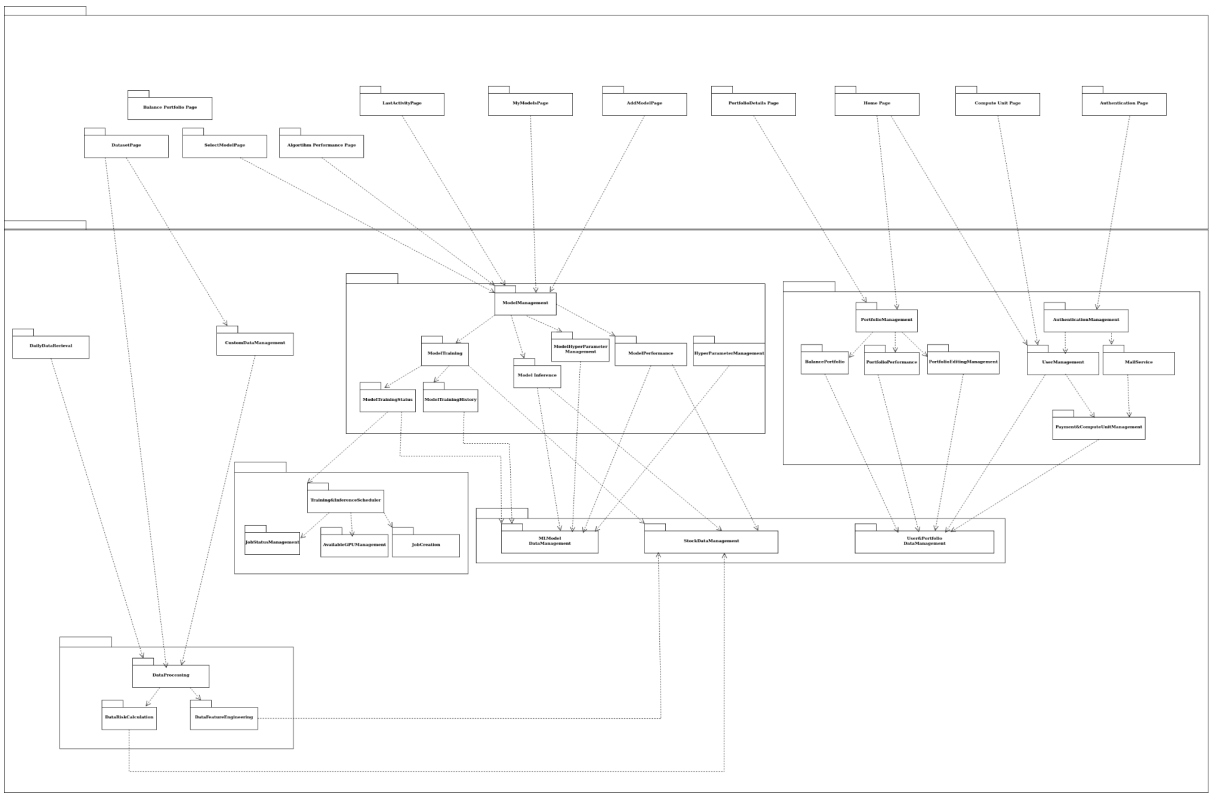
3.1. Overview

Para-Meter will employ a modular, three-tier architecture consisting of:

- **Presentation Tier:** A user-friendly interface for interacting with the system.
- **Application Tier:** The core of the system, responsible for portfolio optimization, data analysis, backtesting, and risk management.
- **Data Tier:** A database for storing financial data, user data, and model parameters.

3.2. Subsystem Decomposition

The following subsystem decomposition diagram provides a high-level breakdown of different subsystems. For better resolution, it can be accessed as the link: https://drive.google.com/file/d/15Ssclki3rVVZfKjP0SFsRBXoNT_T7luK/?usp=drive_link



The frontend subsystems explained in the diagrams are:

- **Balance Portfolio Page:** Displays rebalanced portfolios based on optimization algorithms.
- **Dataset Page:** Allows users to upload and manage datasets used for training.
- **Select Model Page:** Enables users to select machine learning models for training and optimization.
- **Algorithm Performance Page:** Provides insights into algorithm performance with backtesting reports.
- **Load Model Page:** Allows users to load previously trained models.
- **My Models Page:** Displays a list of models trained by the user.
- **Add Model Page:** Provides options to train a new model with custom parameters.
- **Portfolio Details Page:** Shows detailed information about a specific portfolio.
- **Home Page:** The main landing page for the system.
- **Compute Cost Page:** Displays cost estimations for running models on cloud infrastructure.
- **Authentication Page:** Manages user authentication, login, and session handling.

The backend subsystems explained in the diagrams are:

1. Model Management Subsystem

This subsystem oversees all machine learning model-related operations:

- Model training on historical and custom datasets
- Storing metadata and logs for past training sessions.
- Handling hyperparameter tuning and optimization.
- Tracking accuracy, efficiency, and effectiveness of trained models.
- Deploying trained models for making real-time predictions.
- Optimizing parameters to enhance model efficiency.

2. Portfolio Management Subsystem

This subsystem handles the core portfolio functionality:

- Portfolio creation and configuration
- Asset allocation and rebalancing
- Parameter management for optimization constraints
- Portfolio versioning and history tracking
- Portfolio comparison tools

3. Training Resource Scheduler

This component ensures efficient management of computing resources.

- Distributing training workloads across available resources.
- Allocating and releasing GPU/CPU instances dynamically.

- Defining and initializing new model training tasks.

4. Data Processing Subsystem

Handles data preprocessing for machine learning models.

- Providing a unified data pipeline supporting market data and custom data.
- Computing various financial parameters from the market data.
- Extracting and transforming features for ML training.

5. User & Authentication Management

- Handling user accounts, profiles, and permissions.
- Managing secure access control and login mechanisms.
- Sending email notifications and confirmations.
- Tracking user payments and computing costs for cloud-based model training.

6. Storage and Data Management

- Storing user-specific portfolio configurations.
- Maintaining past transactions and model performances.
- Storing trained models and logs for inference and retraining.

3.3. Hardware/Software Mapping

Para-Meter is hosted on cloud infrastructure, which provides the flexibility and scalability required for a modern portfolio optimization tool. The system leverages virtual servers for hosting the front end, API gateway, and backend microservices, and it uses high-speed networking to ensure efficient data transfer between these components.

A key aspect of the design is the utilization of cloud GPUs for computationally intensive machine-learning tasks. These GPUs accelerate our machine learning models' training and inference processes, ensuring that the system can handle demanding workloads effectively.

Furthermore, the training workloads are managed through Kubernetes Pod Management. This orchestration system enables efficient scaling and deployment of containerized applications[1]. With Kubernetes, Para-Meter achieves cloud-agnostic deployment, as managed Kubernetes services are available from multiple cloud providers, such as Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE), and Amazon Elastic Kubernetes Service (EKS) [2][3]. This approach improves resource utilization, simplifies container management, and ensures that the system can dynamically adjust to varying workload demands.

In summary, the hardware/software mapping of Para-Meter is designed to provide robust, scalable, and efficient performance by combining cloud-based virtual servers, GPU acceleration for machine learning tasks, and Kubernetes orchestration for seamless container management.

3.4. Persistent Data Management

3.4.1. Data Categories and Storage Solutions

The Para-Meter system implements a polyglot persistence approach with specialized databases optimized for different data categories:

- **User Data & Portfolio Information**: PostgreSQL provides ACID compliance, complex querying capabilities, and reliable backup mechanisms for structured user data and portfolio configurations
- **Machine Learning Model Parameters**: AWS S3 object storage, offering scalable and cost-effective storage with versioning capabilities for trained models and their configurations
- **High-Volume Time Series Data**: TimescaleDB delivers optimized time-series performance with automatic partitioning, high ingest rates, and native compression for financial market data and performance metrics
- **Real-time Data Access**: Redis for caching frequently accessed data to improve response times and reduce database load

This architecture ensures that each data category is stored using technology that addresses its specific access patterns, volume characteristics, and performance requirements.

3.4.2. Data Integration and Access Patterns

The Para-Meter system implements a cohesive data access layer that integrates these disparate storage solutions:

- **Data Synchronization**: Critical data is synchronized between storage systems as needed, with clear ownership boundaries
- **Caching Strategy**: Frequently accessed data is cached using Redis to reduce database load and improve response times
- **Query Optimization**: Database queries are carefully designed and optimized for each storage technology's strengths

3.4.3. Data Backup and Recovery

To ensure data integrity and availability, the Para-Meter system implements comprehensive backup and recovery procedures:

- **Automated Backups**: Scheduled daily backups for PostgreSQL and TimescaleDB

- **Point-in-Time Recovery**: Capability to restore data to any point within the retention period
- **Cross-Region Replication**: Critical data is replicated across multiple geographic regions
- **Backup Validation**: Regular testing of backup restoration processes to ensure recoverability

3.4.4. Data Lifecycle Management

The system implements policies for efficient management of data throughout its lifecycle:

- **Data Retention**: Time-based retention policies applied to historical market data
- **Data Archiving**: Automated movement of infrequently accessed data to lower-cost storage tiers
- **Data Purging**: Secure deletion of data that has exceeded its retention period
- **Version Control**: Maintenance of historical versions for machine learning models and portfolios

By leveraging these specialized storage solutions and data management practices, the Para-Meter system achieves optimal performance, scalability, and cost-effectiveness while ensuring data integrity and availability for portfolio optimization operations.

3.5. Access Control and Security

Security in Para-Meter is built around two main areas: protecting API endpoints and ensuring the privacy of custom datasets. We follow recognized security standards such as NIST SP 800-63B, GDPR, ISO/IEC 27001, and the NIST Cybersecurity Framework to keep our system safe [4].

3.5.1. Endpoint Authentication & Authorization

To secure our API endpoints, we use the JSON Web Token (JWT) bearer token method. This approach provides stateless authentication and makes it easy to scale our access control. JWT helps us enforce detailed access policies while keeping the token verification process efficient.

User passwords are protected by being encrypted using the BCrypt hashing algorithm. This method is recommended by NIST SP 800-63B as it offers strong protection against attacks on stored passwords. By using BCrypt, we ensure that user authentication is both secure and in line with industry standards.

3.5.2. Custom Dataset Privacy Security

Para-Meter takes data privacy seriously, especially for user-provided datasets used in training and inference. To follow GDPR and ISO/IEC 27001 standards, all user data is processed in

memory or stored temporarily. These temporary files are automatically deleted every day, which helps reduce the risk of unauthorized access and long-term storage issues.

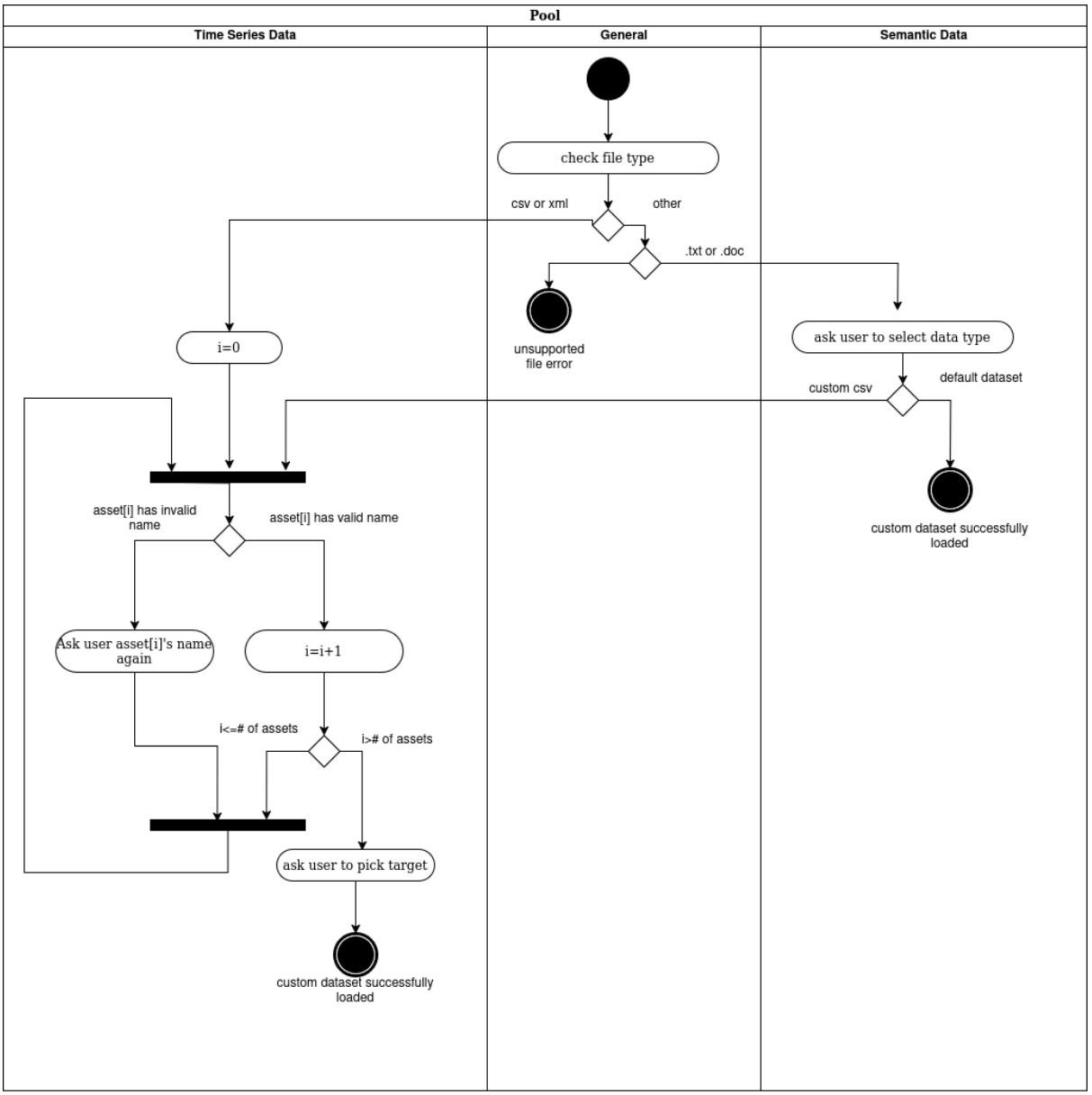
Our data handling procedures include strict access controls and secure data processing steps. These measures help protect user data and align with guidelines for data minimization and storage limitations. By following these practices, Para-Meter maintains a high level of data privacy and security without adding unnecessary complexity.

In summary, our security approach uses JWT and BCrypt for secure endpoint access while handling user data with care to meet privacy standards. This keeps the system safe and builds trust with our users.

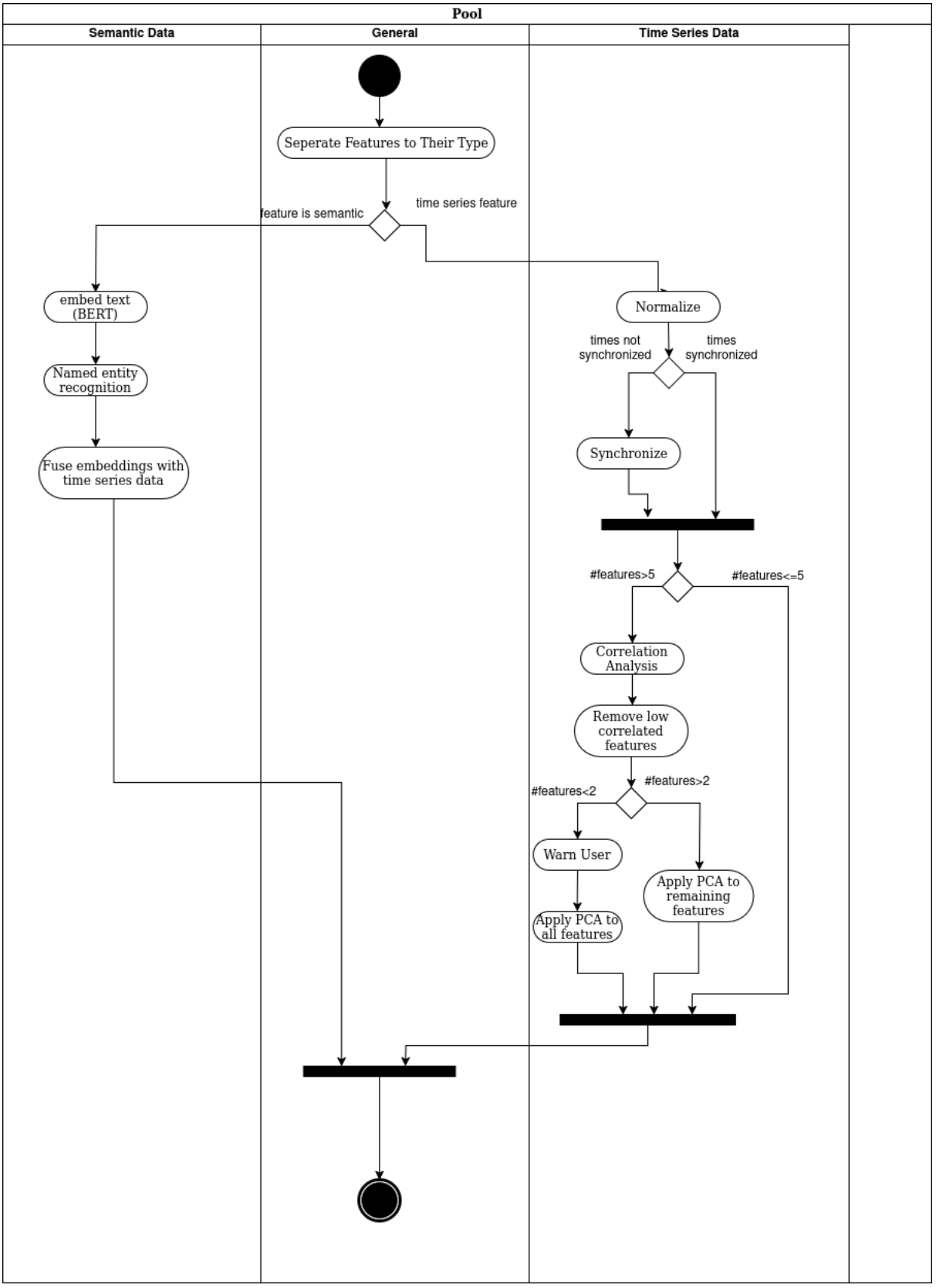
4. Subsystem Services

This section will describe the essential subsystem services by activity and sequence diagrams.

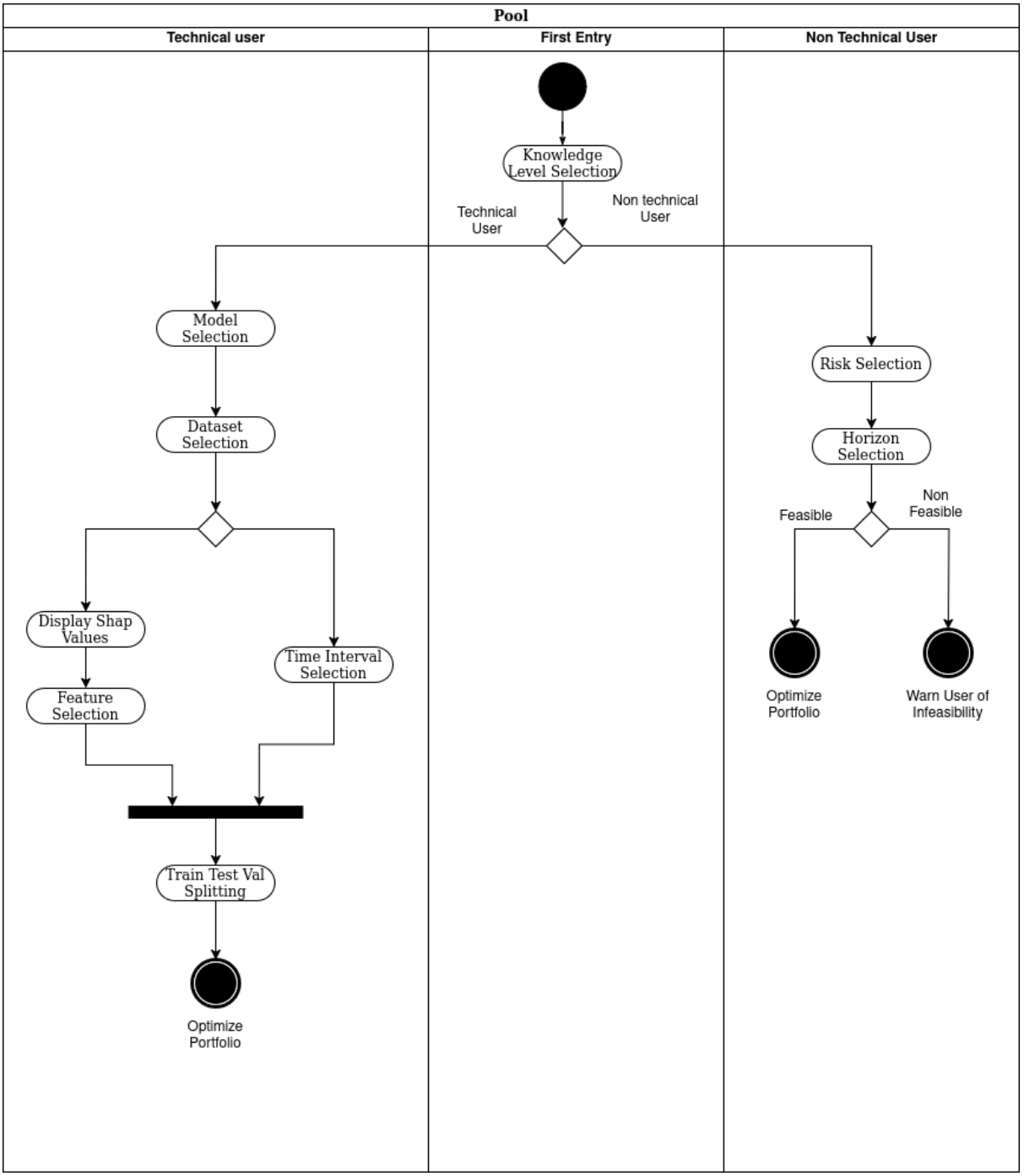
4.1. Custom Data Management Subsystem



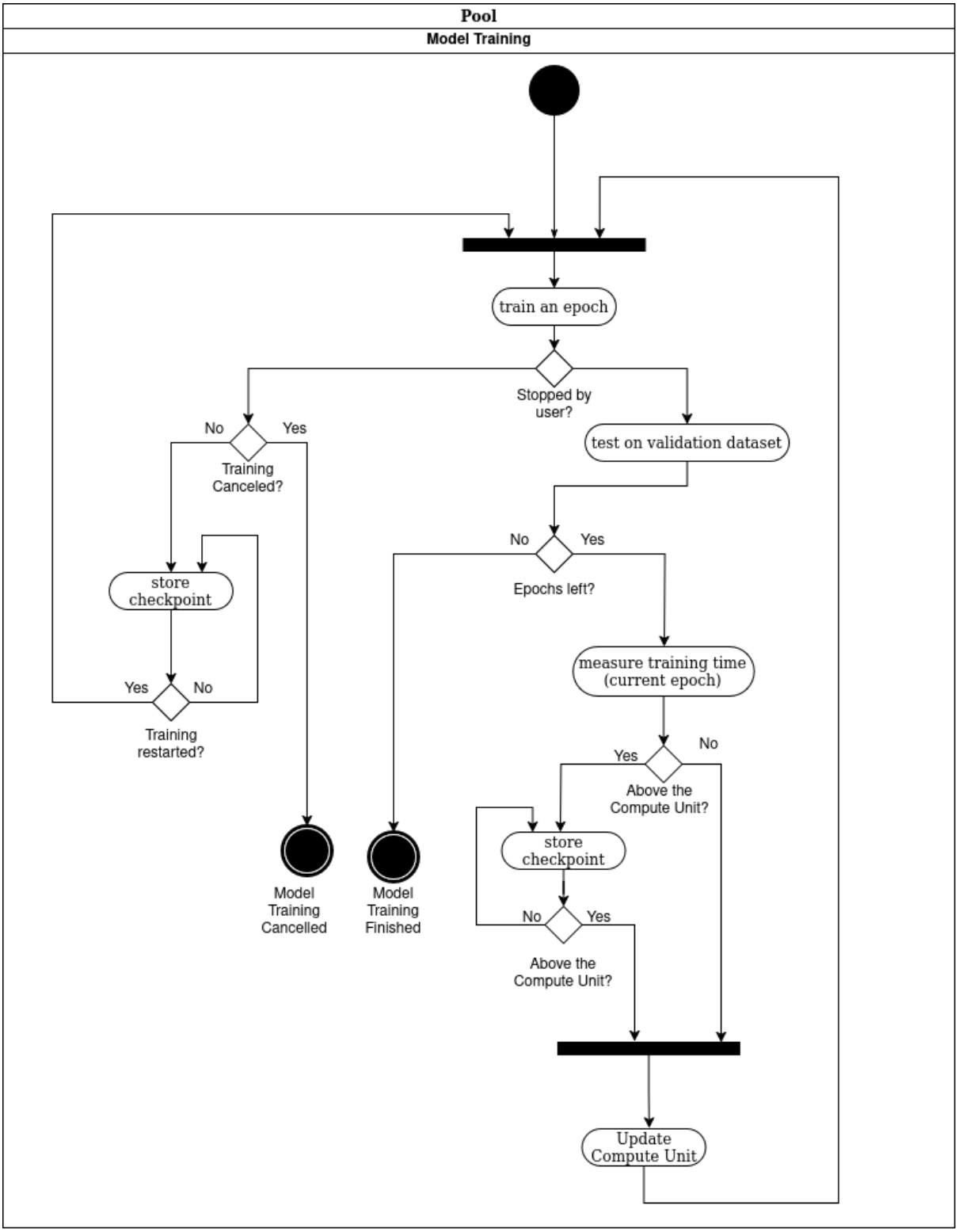
4.2. Data Preprocessing



4.3. Balance Portfolio



4.4. Model Training



5. Test Cases

5.1. Functional Test Cases

Test ID	1	**Category**	Functional	**Severity**	Minor
Objective	Verify the login functionality works				
Expected	The user successfully logs in if the username and password fields are correct else; the login operation should fail and show the “Username and password do not match.” message.				
Steps	Successful Login: 1. Navigate to the login page 2. Enter the username correctly 3. Enter the password correctly 4. Click on the log-in button and log in to the account. Failed Login: 1. Enter the username or password incorrectly. 2. Click on the log-in button and get the correct error message.				
Date-Result					

Test ID	2	**Category**	Functional	**Severity**	Minor
Objective	Verify that the user registration process works correctly.				
Expected	The user should be registered, and a verification email is sent to the user if the username and password are valid. Otherwise, the operation fails if the username is already taken, the password does not comply with the password restrictions, and the “Username or password invalid” message is shown.				
Steps	Successful Registration: 1. Navigate to the registration page. 2. Enter a valid username, email address, and password. 3. Click on the "Register" button. 4. Check the email inbox for an activation email. 5. Click on the activation link to activate the account. Failed Registration: 1. Navigate to the registration page 2. Enter an already-used username or an invalid password. 3. Check if the correct error message appears on the screen.				
Date-Result					

Test ID	3	**Category**	Functional	**Severity**	Minor
Objective	Verify the forgot password functionality.				
Expected	When a valid email is provided, the system sends a password reset link to the user's registered email address, and the user successfully resets their password.				
Steps	1. Navigate to the login page and click on "Forgot Password." 2. Enter the registered email address. 3. Click on the "Submit" button. 4. Check the email inbox for a password reset link. 5. Click the reset link and set a new valid password.				
Date-Result					

Test ID	4	**Category**	Functional	**Severity**	Critical
Objective	Verify that the unauthenticated users cannot access restricted pages via URL.				
Expected	The user will be redirected to the login page upon unauthenticated access trial with an error message.				
Steps	1. Log out if you are logged in. 2. Clear the browser cache or open an incognito/private window to ensure no cached tokens or session data are present. 3. Type the URL of a restricted page and hit enter. 4. Verify that the system automatically redirects to the login page with the expected error message.				
Date-Result					

Test ID	5	**Category**	Functional	**Severity**	Medium
Objective	Verify that the user can add a portfolio with valid input values.				
Expected	A new portfolio appears in the user's portfolio list when valid data is provided.				
Steps	1. Log in and navigate to the "Create Portfolio" section. 2. Enter a portfolio name and required parameters. 3. Click on the "Create Portfolio" button. 4. Verify that the new portfolio appears in the portfolio list.				
Date-Result					

Test ID	6	**Category**	Functional	**Severity**	Critical
Objective	Verify that the user can update their portfolios.				
Expected	Changes made to the portfolio details are saved and correctly reflected in the portfolio summary.				
Steps	1. Log in and select an existing portfolio. 2. Click the "Edit Portfolio" button. 3. Modify portfolio parameters (e.g., name, asset allocation) with proper values. 4. Click "Save Portfolio" and verify that the updated information is displayed.				
Date-Result					

Test ID	7	**Category**	Functional	**Severity**	Major
Objective	Verify that the user can delete their portfolios.				
Expected	The selected portfolio is permanently removed from the portfolio list after confirmation.				
Steps	1. Log in and navigate to the portfolio list. 2. Select a portfolio to delete. 3. Click the "Delete Portfolio" button and confirm the deletion when prompted. 4. Verify that the portfolio no longer appears in the list.				
Date-Result					

Test ID	8	**Category**	Functional	**Severity**	Major
Objective	Verify that the user can see their portfolios on the dashboard. Portfolio Cards must be responsive to the different devices				
Expected	The user should see their portfolios in the “My Portfolios” section if they have portfolio(s). The portfolios must be adjusted to small, medium, and big-sized screens.				
Steps	1. Navigate or be redirected to the dashboard page. 2. The system displays the dashboard with the “My Portfolios” section visible. 3. Verify that the portfolio cards are correctly displayed within the “My Portfolios” section.				

	4. Resize the browser window or use device emulation tools to simulate small, medium, and large screen sizes. 5. Confirm that the portfolio cards adjust responsively and maintain proper layout and usability across different devices.
Date-Result	

Test ID	9	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users can upload their portfolio as a CSV file, given that it is in the correct format.				
Expected	The upload should be successful if the file is in the correct format. Otherwise, the user will see an error message, “The CSV file is in incorrect format. Please upload a CSV file with expected columns.”				
Steps	1. Navigate to the “Add Portfolio” page. 2. Fill out the necessary input fields. 3. Choose to export their portfolio via CSV file upload. 4. Click the submit button to validate that the portfolio appears in the list with the correct values.				
Date-Result					

Test ID	10	**Category**	Functional	**Severity**	Critical
Objective	Verify that the user can select from predefined machine-learning models for portfolio optimization.				
Expected	The options should be chosen from the form, and no errors should occur upon submission.				
Steps	1. Navigate to the “Balance Portfolio” page and select the “Technical User” option. 2. Select a model from the predefined options and fill out the form with the correct inputs. 3. Click on the “Next” and verify the selected models.				
Date-Result					

Test ID	11	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users can upload a custom dataset to train their models with their own data.				

Expected	The upload should be successful if the dataset is in the correct file format. Otherwise, the system should show an error message.
Steps	1. Click on the “Balance Portfolio” button under the portfolio to balance the portfolio. 2. Click on the “Technical User” option. 3. Choose among model options and click the next button. 4. In the “Select Dataset” part of the form, choose to use the custom dataset option. A file input field will appear as the user chooses this option. 5. Click the “Browse Files” button to choose a file to upload a custom dataset. 6. Verify the uploading of the dataset from the preview.
Date-Result	

Test ID	12	**Category**	Functional	**Severity**	Critical
Objective	Verify that the system gives information about the progress of machine-learning models that the user trains.				
Expected	The system should show an informative message on the status of the training.				
Steps	1. Click on the “Balance Portfolio” button under the portfolio to balance the portfolio. 2. Select any type of user. 3. Continue to model training. 4. Verify the training status information is informative and valid.				
Date-Result					

Test ID	13	**Category**	Functional	**Severity**	Major
Objective	Verify that the user can abort the training process before it is completed.				
Expected	If the abortion is successful, the user will be notified via notification. If a failover happens during abortion (connection loss), the request must be persisted until abortion successfully occurs. Compute units must be restored accordingly.				
Steps	1. Click on the “Balance Portfolio” button under the portfolio to balance the portfolio. 2. Choose any type of user. 3. Proceed with the model training on the portfolio. 4. Before completing training, click on the “Abort Training” button on the “Past Activities” page.				

	5. Validate the behavior of the abortion process and the amount of compute units.
Date-Result	

Test ID	14	**Category**	Functional	**Severity**	Minor
Objective	Verify that the user cannot train a model if they do not have enough compute units threshold for that workload.				
Expected	The user is notified that they don't have enough training compute unit threshold.				
Steps	1. Log in with an account with fewer compute units than required for the selected model workload. 2. Navigate to the "Balance Portfolio" page. 3. Select the technical knowledge level as a technical user and choose custom models. 4. Check if the necessary compute units are higher than the account holds. 5. Ensure the training process does not start and that the notification appears on the notifications page.				
Date-Result					

Test ID	15	**Category**	Functional	**Severity**	Critical
Objective	Verify that the system uses the selected model to optimize the portfolio of the user.				
Expected	The assets' final weights and the selected models' backtesting results appear.				
Steps	1. Click on the "Balance Portfolio" button under the portfolio to balance the portfolio. 2. Select any type of user. 3. Continue with the training and optimization process. 4. Validate that the selected models' results appear on the screen after finishing the training process.				
Date-Result					

Test ID	16	**Category**	Functional	**Severity**	Major
Objective	Verify that the users can download the optimization backtesting result as a				

	report.
Expected	The user should download the optimization results report. Results will consist of backtesting and will contain all metrics without anything absent that is shown on the page.
Steps	1. Navigate to the Past Activities page. 2. Click on the backtesting report option. 3. The backtesting results are displayed on the screen. 4. Click on the “Download PDF” button. 5. The system downloads a PDF report containing all the displayed backtesting metrics.
Date-Result	

Test ID	17	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users can purchase compute units to train their models.				
Expected	Upon successful transaction, the compute unit quota should be increased by the user's purchase amount.				
Steps	1. Navigate to the purchase compute units page from the sidebar. 2. Enter the amount of compute units to purchase. 3. The total amount will be calculated, and the user will be charged automatically according to the unit price. 4. Enter the debit/credit card details correctly. 5. Click the submit button to purchase, and the payment system will redirect you to transaction approval. 6. Verify that the user is redirected to the application after approval and that the compute units have increased accordingly.				
Date-Result					

Test ID	18	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users are charged correctly for the compute units they want.				
Expected	The amount of money taken from the user is the same as the calculated amount for the compute units they purchase.				
Steps	1. Navigate to the purchase compute units page from the sidebar. 2. Enter a valid amount of compute units. 3. The total amount will be calculated, and the user will be charged automatically according to the unit price. 4. Enter the debit/credit card details. 5. Click on the submit button to purchase, and the payment system will				

	redirect you to transaction approval. 6. Check the bank account to verify the amount of money withdrawn.
Date-Result	

Test ID	19	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users can see their purchase and usage history in detail.				
Expected	All activities related to consuming or purchasing compute units are listed on the Past Activities page.				
Steps	1. Apply training and purchase operations to spend or buy compute units. 2. Record these operations. 3. Navigate to the "Past Activities" page from the sidebar menu. 4. View all past activities performed and verify that every consumption and purchase of compute units is listed there.				
Date-Result					

Test ID	20	**Category**	Functional	**Severity**	Critical
Objective	Verify that the user can update their portfolio upon the optimization recommendations.				
Expected	The user's selected portfolio is updated according to the recommended optimized portfolio.				
Steps	1. Train a model/models to optimize the portfolio. 2. After completing the optimization process, click on the update portfolio button to update the portfolio according to the recommended portfolio. 3. Validate that the portfolio is rebalanced with the suggested asset weights on the Dashboard page.				
Date-Result					

Test ID	21	**Category**	Functional	**Severity**	Critical
Objective	Verify that the users receive a notification when the training job is finished.				
Expected	Receive a notification about the completed job.				
Steps	1. Start training to optimize the portfolio.				

	2. Verify the related notification appears on the notifications page when training is done.
Date-Result	

Test ID	22	**Category**	Functional	**Severity**	Critical
Objective	Verify that the user is not charged if the transaction for the compute unit purchase fails. Also, the user is informed in case the transaction fails.				
Expected	Receive an error message about the failure, and the user will not be charged.				
Steps	1. Navigate to the “Purchase Compute Unit” page. 2. Fill out the amount of compute units to be purchased with a valid input. 3. Enter the wrong credentials, or the card with not enough account balance. 4. Verify that no purchase is made and the error notification is sent to the user.				
Date-Result					

Test ID	23	**Category**	Functional	**Severity**	Major
Objective	Verify that the logout functionality works correctly.				
Expected	Logged-out accounts cannot access the restricted pages without logging in again				
Steps	1. Log in with valid credentials. 2. Click on the "Logout" button in the navigation bar. 3. Verify that the user is redirected to the login page with the message “You have been logged out.” and that the user cannot access the restricted pages.				
Date-Result					

Test ID	24	**Category**	Functional	**Severity**	Minor
Objective	Verify that the homepage profit/loss plot is working correctly.				
Expected	The profit/loss plot only contains the current portfolios.				

Steps	1. Create a new portfolio. 2. Validate it appears on the profit/loss plot. 3. Delete a portfolio. 4. Validate it disappears from the profit/loss graph.
Date-Result	

Test ID	25	**Category**	Functional	**Severity**	Major
Objective	Verify that adding a custom model addition makes the model usable.				
Expected	● Step 3: Recently added models should be seen in the list of custom models. ● Step 5: See the recently added model in the options of selecting a model. ● Step 7: Model trains without error. ● Step 8: See the backtesting result associated with the custom model.				
Steps	1. Log in with a valid account. 2. Create a model with random valid parameters. 3. Navigate to the “My Models” page. 4. Go back to the homepage and click on the “balance portfolio” button of a portfolio. 5. Select the recently added custom models in model addition. 6. Fill in additional parameters such as risk threshold. 7. The recently added custom model is trained and inferred. 8. Backtesting results are given accordingly.				
Date-Result					

Test ID	26	**Category**	Functional	**Severity**	Major
Objective	Verify that the user email verification works.				
Expected	Unverified users cannot log in directly and require verification. After the verification, their accounts are activated.				
Steps	1. Register a new user. 2. Try to log in to see the error message indicating the user is not verified. 3. Using the link sent with the verification email, activate the registered account. 4. Log in to see if the account is activated, and the user can see the restricted pages.				
Date-Result					

Test ID	27	**Category**	Functional	**Severity**	Critical
Objective	Verify that the system correctly handles datasets containing missing values or invalid data entries during upload.				
Expected	If the custom dataset contains missing information, the system should warn users that their data is impartial.				
Steps	1. Upload a dataset with missing information. 2. Verify that the warning about the custom data appears.				
Date-Result					

Test ID	28	**Category**	Functional	**Severity**	Medium
Objective	Verify that the users are notified when their portfolios need a new rebalance operation.				
Expected	The system should send a notification when the investment horizon of the portfolio has reached and rebalance is needed.				
Steps	1. Train a model/models to optimize the portfolio. 2. Rebalance the portfolio accordingly, and note the investment horizon. 3. Validate that the notification is sent when the investment horizon is reached.				
Date-Result					

Test ID	29	**Category**	Functional	**Severity**	Medium
Objective	Verify that users can compare different portfolios in the backtesting.				
Expected	The system should allow users to see different balanced portfolios side by side and compare them.				
Steps	1. Create 2 portfolios. 2. Balance them with the models available in the system. 3. Validate that after the training, the backtest results for the old versions of those portfolios and the suggested versions of them can be comparable simultaneously.				
Date-Result					

Test ID	30	**Category**	Functional	**Severity**	Critical
Objective	Verify that the system doesn't allow training with empty custom data or empty portfolios.				
Expected	The system must prevent the initiation of model training when either the portfolio data or custom data are empty. It should display the error message "Cannot train the model: portfolio data is empty." if the portfolio lacks stocks or "Cannot train the model: custom data is empty." if the custom data is missing.				
Steps	1. Log in as a user with sufficient compute units. 2. Create a new portfolio without adding any stocks. 3. Attempt to start model training with the empty portfolio. 4. Verify that the error "Cannot train the model: portfolio data is empty." is displayed. 5. Add valid stock data to the portfolio. 6. Attempt to start model training with an empty custom dataset. 7. Verify that the error "Cannot train the model: custom data is empty." is displayed. 8. Provide valid custom data and verify that training proceeds successfully.				
Date-Result					

Test ID	31	**Category**	Functional	**Severity**	Critical
Objective	Verify add model functionality.				
Expected	The created model must appear on the My Models page.				
Steps	1. Navigate to the Add Model page. 2. Select a model and arrange the parameters. 3. After adding the model, validate that it appears on the My Models page with the correct parameters.				
Date-Result					

Test ID	32	**Category**	Functional	**Severity**	Critical
Objective	Verify update model functionality.				

Expected	The updated parameters must be stored and displayed on the My Models page.
Steps	1. Navigate to the My Model page. 2. Select a model and click Edit. 3. Update the parameters of the model. 4. Click Save. 5. Verify that the parameters of the model are updated on the My Models page.
Date-Result	

Test ID	33	**Category**	Functional	**Severity**	Critical
Objective	Verify delete model functionality.				
Expected	The deleted model must be unlisted from the My Models page.				
Steps	1. Create a model if there is no one. 2. Navigate to the My Models page. 3. Select a model and click the Delete button. 4. Approve the delete operation. 5. Verify that the model is unlisted from the My Models page.				
Date-Result					

Test ID	34	**Category**	Functional	**Severity**	Critical
Objective	Verify that the user can update their profile information.				
Expected	The updated information is saved and displayed correctly on the profile page.				
Steps	1. Log in and navigate to the "Profile" section. 2. Modify fields such as name, email, or contact details. 3. Click the "Save Changes" button. 4. Refresh the page and verify that the updated information is displayed.				
Date-Result					

Test ID	35	**Category**	Functional	**Severity**	Critical

Objective	Verify that the session times out after a period of inactivity.
Expected	The system automatically logs out the user after the defined inactivity period and prompts for re-login.
Steps	1. Log in and remain inactive for the preset timeout period. 2. Attempt to interact with the system after the timeout. 3. Confirm the system will redirect you to the login page with a timeout message.
Date-Result	

Test ID	36	**Category**	Functional	**Severity**	Critical
Objective	Verify that user settings can be updated successfully.				
Expected	Changes in user settings are saved and correctly reflected on the settings page.				
Steps	1. Log in and navigate to "User Settings." 2. Modify settings such as notification preferences or display options. 3. Click "Save" and then refresh the page to verify changes.				
Date-Result					

5.2. Non-Functional Test Cases

Test ID	37	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the training subsystem can handle multiple model training without performance loss with respect to time.				
Expected	The system scales according to the number of training jobs, and the user experience and time efficiency do not decrease dramatically.				
Steps	1. From multiple profiles, train multiple models in each profile. 2. Measure the completion time of the jobs and compare them to the completion time of the identical jobs that work in isolation. 3. Verify that no significant performance loss occurred in the system.				
Date-Result					

Test ID	38	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the app functions properly on multiple browsers.				
Expected	The user interface does not differ much across browsers, and the functionalities work as expected.				
Steps	1. Log in to the application across mostly used browsers, such as Chrome, Safari, and Firefox. 2. Try out the main functionalities of the application on different browsers. 3. Validate that there is no problem with the interface or the functionality.				
Date-Result					

Test ID	39	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the portfolio optimization is completed within the estimated timeframe.				
Expected	The optimized portfolio results are calculated within the estimated time after the process is finished.				
Steps	1. Start a training process to optimize the portfolio. 2. Note the reported expected time. 3. Measure the real-time to finish the training process. 4. Validate that the realized time is no more than the threshold (e.g. $1.1 \times estimated\ time$).				
Date-Result					

Test ID	40	**Category**	Non-Functional	**Severity**	Major
Objective	Verify that the system supports training requests more than the available GPUs.				
Expected	● Step 2: The system should accept training requests that exceed available GPU clusters. ● Step 3,4: For each running job, the status should be either pending or running.				
Steps	1. From a client (e.g., using a curl command), query the training scheduling service to retrieve the current resource status, specifically the number of available GPUs. 2. Submit training requests equal to three times the number of existing				

	GPUs in the system. 3. Using Kubernetes (or the orchestration platform), check the status of each training job submitted. 4. Verify that each job's status is either pending (queued for execution) or running (actively using GPU resources). 5. Continuously monitor the status of the training jobs until all jobs have either been completed or transitioned out of the pending state. 6. Validate that the system correctly manages the over-subscription of GPU resources without causing failures or resource conflicts.
Date-Result	

Test ID	41	**Category**	Security	**Severity**	Critical
Objective	Verify that the system is resilient to the SQL injection attacks.				
Expected	If an ill-intended actor tries to inject SQL queries, the system should not execute those queries.				
Steps	1. Try all user input areas, such as editing profiles, creating portfolios, etc. 2. Verify that SQL injection attacks are unsuccessful.				
Date-Result					

Test ID	42	**Category**	Non-functional	**Severity**	Medium
Objective	Verify the scalability and stability of the backend under heavy concurrent usage.				
Expected	The system should be able to handle many users using the website concurrently.				
Steps	1. Write a script that simulates various user's behavior. 2. Run the script in large quantities in parallel. 3. Verify that the system works fine by performance monitoring.				
Date-Result					

Test ID	43	**Category**	Non-functional	**Severity**	Medium
Objective	Verify that the market data update time doesn't introduce any inconsistency in rebalancing and backtesting.				

Expected	The system should use the updated data available only after the update process.
Steps	1. Before the market data update time, run backtesting and portfolio optimization tasks. 2. During market data update time, run backtesting and portfolio optimization tasks and validate that results are the same and correct. 3. After the market data update time, run backtesting and portfolio optimization tasks to validate that the results are updated with the new market data.
Date-Result	

Test ID	44	**Category**	Non-functional	**Severity**	Critical
Objective	Verify the system is resilient against brute force attacks for stealing passwords and accounts.				
Expected	The system should prevent brute force attacks by presenting a cooldown or email verification.				
Steps	1. Register a user account and verify it. 2. Log out from that user account. 3. Try the wrong passwords multiple times. 4. Validate that the system blocks this brute force approach after the threshold times attempts.				
Date-Result					

Test ID	45	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the compute unit estimation system works accurately within an error rate.				
Expected	The system should be able to estimate the computing cost of training beforehand within a decided error range.				
Steps	1. Create different training tasks with different combinations. 2. Compare the estimated computing unit usage and realized computing unit usage. 3. Verify that the difference is within the reasonable error range.				
Date-Result					

Test ID	46	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the system updates historical market data accordingly during stock splits.				
Expected	The historical stock prices and calculated values must be adjusted according to the stock split.				
Steps	1. Upload test data, including stock split. 2. Validate the handling of the stock split via portfolio optimizations and backtesting results.				
Date-Result					

Test ID	47	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that the system maintains market data integrity during market data updates.				
Expected	The system should be resilient to crashes during market data updates. If a crash occurs, the system must detect missing or incomplete data and automatically resume the update process.				
Steps	1. Prepare a historical market dataset with the last week's data intentionally removed. 2. Start the market data update process. 3. Inject simulated accurate data for each missing day of the last week sequentially. 4. Intentionally crash the application during the update process. 5. Restart the application. 6. Verify that the system automatically detects the missing data and resumes the update process to complete the dataset.				
Date-Result					

Test ID	48	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that multiple concurrent logins from different devices are handled correctly.				
Expected	The system allows concurrent logins while maintaining session integrity and data consistency.				
Steps	1. Log in on one device with valid credentials. 2. Log in on a second device using the same account.				

	3. Verify that both sessions remain active and data is synchronized between devices.
Date-Result	

Test ID	49	**Category**	Non-functional	**Severity**	Critical
Objective	Verify that file uploads respect the maximum size limit.				
Expected	The system rejects files that exceed the specified size limit with an appropriate error message.				
Steps	1. Navigate to the "Balance Portfolio" section as a Technical User. 2. During the training steps, choose to upload a custom dataset. 3. Select a file larger than the allowed limit. 4. Click "Upload" and observe the error message.				
Date-Result					

Test ID	50	**Category**	Security	**Severity**	Critical
Objective	Verify that sensitive data is encrypted in storage.				
Expected	Data such as passwords and personal information are stored in an encrypted format.				
Steps	1. Create or update a user account with sensitive data. 2. Access the backend storage (using a secure tool or test interface). 3. Confirm that sensitive fields are not stored in plain text.				
Date-Result					

6. Consideration of Various Factors in Engineering Design

This chapter details the key constraints that have influenced our design decisions and outlines the standards we follow to ensure a robust, secure, and high-quality system.

6.1. Constraints

6.1.1. 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.

6.1.2. 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.

6.1.3. 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.

6.1.4. 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, their overall financial security could improve, 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.

6.1.5. 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.

6.1.6. 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.

	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.

Table 1. Factors that can affect analysis and design.

6.2. Standards

To ensure that Para-Meter is developed with best practices and meets industry requirements, we follow several widely recognized standards:

- **IEEE 830:**
  This standard provides guidance for writing clear and comprehensive requirements specifications. We use it to document both functional and non-functional requirements.

- **UML 2.5.1:**
  The Unified Modeling Language (UML) is used to create use-case diagrams, class diagrams, activity diagrams, and sequence diagrams. UML helps us design a modular and well-structured system.

- **OAuth 2.0:**
  We implement OAuth 2.0 for secure user authentication and authorization. This protocol helps manage secure access to our system and supports scalable, token-based authentication.

- **ISO/IEC 27001:**
  This international standard for information security management systems (ISMS) ensures that our approach to data security is robust. It guides our risk assessment and mitigation strategies, including secure data handling and access controls.

- **NIST SP 800-63B:**
  We follow the guidelines set by NIST SP 800-63B for secure password storage and user authentication. For example, passwords are hashed using the BCrypt algorithm to protect user credentials.

- **NIST Cybersecurity Framework:**
  Our security measures, such as the use of JWT for endpoint authentication and fine-grained access control, are designed in line with the NIST Cybersecurity

Framework. This framework provides a comprehensive approach to managing cybersecurity risks.

- **GDPR Compliance:**
  In handling user data, we adhere to GDPR principles. This includes data minimization, secure processing, and ensuring that user data is not stored longer than necessary.

7. Teamwork Details

This section describes the teamwork approach during the project, outlining how each member contributed to the project, fostered a collaborative and inclusive work environment, and took lead roles in different parts of the project.

7.1. Contributing and Functioning Effectively on the Team

Each team member played a vital role in ensuring that the project advanced efficiently and effectively:

- **Tuna Saygın**
  He took charge of the microservices, especially the ML microservice, and contributed to the overall architectural design. His contributions were central to integrating machine learning components with the rest of the system. His role ensured that the architecture remained modular and scalable, supporting the project's advanced functionalities.
- **Sıla Özel:**
  She was responsible for designing and implementing a user-friendly front end. She focused on ensuring that the interface was both efficient and accessible, contributing to a seamless user experience. Her dedication to UI/UX design ensured that the system remained intuitive, which is critical for the target users.
- **Abdullah Samed Uslu**
  He managed the JIRA system to track tasks and deadlines, ensuring that all project activities were clearly documented and scheduled. In addition, he contributed significantly to portfolio optimization techniques and managed the financial data aspect of the system. This contribution helped shape the core analytical functions and maintain a structured approach to data management.
- **Muti Kara:**
  He handled the backend development, including communication protocols and database management. His work ensured that data flowed smoothly between the system components and that backend services were efficient and robust. His contributions to backend systems were key in maintaining system performance and reliability.

Overall, each member not only focused on their specific areas of responsibility but also collaborated across disciplines, providing support and insights that enriched the overall quality of the project.

7.2. Helping to Create a Collaborative and Inclusive Environment

Our team prioritized creating an open, collaborative, and inclusive environment through several key practices:

- **Weekly Meetings:**
  We held regular weekly meetings to discuss progress, address challenges, and plan upcoming tasks. These meetings provided a dedicated space for every team member to share updates, ask questions, and contribute ideas. They were essential for keeping everyone aligned and ensuring transparency across all aspects of the project.

- **Open Communication:**
  In addition to weekly meetings, we maintained open communication channels through messaging apps. This constant dialogue helped resolve issues quickly and fostered a sense of camaraderie among team members.

- **Mutual Support and Peer Learning:**
  Each member willingly shared their expertise and assisted colleagues when needed. For example, when technical challenges emerged in the frontend or backend, team members collaboratively discussed solutions and exchanged knowledge. This peer learning approach enhanced our collective skills and improved project outcomes.

- **Inclusive Decision-Making:**
  Major decisions, such as tool selection, system architecture, and project milestones, were discussed openly with input from all members. This inclusive process ensured that diverse perspectives were considered, resulting in well-rounded and effective solutions.

7.3. Taking the Lead Role and Sharing Leadership on the Team

Leadership within our team was shared based on individual strengths, allowing everyone to take charge of their respective areas while supporting each other:

- **Tuna Saygın:**
  He took the lead in the microservices and architectural design efforts. His initiative in organizing discussions around system scalability and modularity helped ensure that our ML components were seamlessly integrated and future-proofed.
- **Sıla Özel:**
  She led the front-end development, guiding design choices and implementation

strategies to ensure an intuitive user experience. Her proactive approach to UI/UX reviews significantly shaped our user interface.

- **Abdullah Samed Uslu:**
  He frequently led discussions on task management and portfolio optimization, setting clear objectives and ensuring smooth progress through effective JIRA management.
- **Muti Kara:**
  He led backend development, coordinating efforts to establish robust communication protocols and manage database operations. His leadership was pivotal in integrating backend components with the overall system.

In every instance, leadership was not confined to one person but was a shared responsibility. Each member stepped up to lead in their domain while actively participating in group decisions, contributing to a sense of collective ownership and ensuring the success of the project.

8. Glossary

API (Application Programming Interface):
A set of protocols and tools that allow different software applications to communicate with each other.

ACID:
A set of properties—Atomicity, Consistency, Isolation, and Durability—that guarantee reliable processing of database transactions.

AI (Artificial Intelligence):
The simulation of human intelligence processes by computer systems, including learning, reasoning, and self-correction.

AWS S3 (Amazon Web Services Simple Storage Service):
A scalable object storage service used for storing and retrieving any amount of data, commonly utilized for storing machine learning model parameters and backups.

BCrypt:
A password hashing function designed to be computationally expensive, which helps secure stored passwords against brute-force attacks.

CCPA (California Consumer Privacy Act):
A state statute intended to enhance privacy rights and consumer protection for residents of California.

Compute Units:
A metric used to estimate and manage the computational cost of tasks such as model training in a cloud environment.

Custom Dataset:
User-provided data can be uploaded and used to train machine learning models, allowing for personalized or alternative analyses.

GDPR (General Data Protection Regulation):
A comprehensive data protection law in the European Union governs how personal data should be handled and secured.

GPU (Graphics Processing Unit):
A specialized processor originally designed for rendering graphics, now widely used to accelerate complex computational tasks in machine learning.

Hyperparameter Tuning:
The process of optimizing the parameters that govern the learning process of a machine learning model to improve its performance.

JWT (JSON Web Token):
A compact, URL-safe means of representing claims to be transferred between two parties, commonly used for secure authentication in web applications.

KVKK (Personal Data Protection Law):
The Turkish regulation for personal data protection, similar in scope to GDPR, governs how personal data must be handled.

Kubernetes:
An open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

Machine Learning (ML):
A field of artificial intelligence that focuses on building systems that learn from and make decisions based on data.

OAuth 2.0:
An industry-standard protocol for authorization that allows applications to obtain limited access to user accounts on an HTTP service.

Portfolio:
A collection of financial investments, such as stocks, bonds, ETFs, and other assets, managed as a single entity.

PostgreSQL:
An open-source relational database management system known for its reliability, robustness, and support for ACID compliance.

Redis:
An in-memory data structure store used as a database, cache, and message broker to improve application performance.

TimescaleDB:
A time-series database built on PostgreSQL, optimized for fast ingest and complex queries of time-series data.

UI (User Interface):
The point of interaction between the user and a computer system, often designed to be intuitive and accessible.

UML (Unified Modeling Language):
A standardized modeling language used to visualize the design of a system, including diagrams for use cases, classes, and activities.

9. References

[1] “Cluster Architecture | Kubernetes” [Online]. Available: https://kubernetes.io/docs/concepts/architecture/. [Accessed: 5-Mar-2025].

[2] “AWS SageMaker Documentation,” [Online]. Available: https://docs.aws.amazon.com/sagemaker/. [Accessed: 5-Mar-2025].

[3] “Google Cloud AutoML Overview,” [Online]. Available: https://cloud.google.com/automl. [Accessed: 5-Mar-2025].

[4] “NIST Cybersecurity Framework,” [Online]. Available: https://www.nist.gov/cyberframework. [Accessed: 5-Mar-2025].