

"THE HYPER EXPERT COLLABORATIVE AI ASSISTANT"

Deliverable 5.4

Report for Integration and Technical Validation (2nd Version)

Version	2.0
Lead partner	FUJITSU
Dissemination level	Public
Type of deliverable	Report and Demonstrator
Contractual date	31/03/2026
Actual delivery date	31/03/2026 (v1); 24/04/2026 (v2)

History of changes:

Version	Date	Changes	Pages
v1.0	Submitted on 31.03.2026	First version including details of three use cases (Sonae, Proditec, and City of Amsterdam)	1-45
v2.0	Submitted on 24.04.2026	Updated version including the previously missing use case Datacation.	1-52

D5.4. Integration and Technical Validation (2nd Version)

This deliverable provides a 2nd version of the demonstrator (MVP) of the conversation between human and AI showing a basic set of functions illustrating the principles of the conversation flow. Use cases will provide a partial data and messages integration for a limited communication flow, which demonstrates the abilities of the technology and facilitates further refinement in version 3.

Authors	**Organisations**
Bertrand, Jan	FUJ
Erdt, Lukas	FUJ
Baser, Cagatay	FUJ
Melnikowitsch, Sergej	FUJ

Contributors	
Ehlert, Maike	AMS
Krishnan, Vishruth	AMS
Mendes, Matilde	SON
Sottomayor, Madalena	SON
Borges, Frederico Guilherme	SON
Silva Moreira, Fabio	InT
Lunet, Miguel	InT
Erbihi, Najib	PDT
Beurskens, Joep	DAT

Revised by	
Fage, Charles	CAT
Durnerin, Elise	CAT
Silvia Izquiero Badiola	EUT
Adriana Basbous Moukarzel	EUT

Disclaimer
This document reflects the authors' views only and neither the Agency nor the Commission are responsible for any use that may be made of the information contained

Table of Contents

1.	Executive summary	7
1.1	Introduction	7
1.2	The PEER Project and its Objectives	7
1.2.1	Purpose of D5.4 Deliverable Report	7
1.2.2	Relation to WP5	7
1.2.3	Relation to WP3	8
1.3	Overview of the 4 Use-Cases	8
1.3.1	Specific Goals of Each Use-Case	8
1.4	Common Cloud Infrastructure and CI/CD Management	9
1.4.1	Dedicated Build Environment	9
1.4.2	Automated CI/CD-Pipelines	9
2.	Description of Work	11
2.1	Sonae	11
2.1.1	Motivation	11
2.1.2	Goals	11
2.1.3	Limitations	11
2.1.4	Usage Scenario	11
2.1.5	Evaluation Journey	12
2.1.6	Design of the MVP	19
2.1.7	Responsibilities	21
2.2	City of Amsterdam	21
2.2.1	Motivation	21
2.2.2	Goals	22
2.2.3	Limitations	22
2.2.4	User Journey	22
2.2.5	Evaluation Journey	22
2.2.6	Design of the MVP	27
2.2.7	Responsibilities	29
2.3	Proditec	29
2.3.1	Motivation	29
2.3.2	Goals	30
2.3.3	Limitations	30
2.3.4	User Journey	30
2.3.5	Evaluation Journey	39
2.3.6	Design of the MVP	39
2.3.7	Deviations and Corrective Actions	40
2.3.8	Responsibilities	41
2.4	Datacation	41

2.4.1 Motivation 41
2.4.2 Goals 42
2.4.3 Limitations 42
2.4.4 User Journey 42
2.4.5 Evaluation Journey 43
2.4.6 Design of the MVP 49
2.4.7 Responsibilities 50
3. Conclusions 51

Table of Figures

Figure 1: Architecture of the PEER Companion Agent 8
Figure 2: Available Personas 13
Figure 3: Selection of Persona 13
Figure 4: Loading Screen 13
Figure 5: Welcome Popup 13
Figure 6: Store Layout 13
Figure 7: Shopping Cart Layout 13
Figure 8: Save Shopping List in Shopping Cart Page 14
Figure 9: Empty Shopping List Page 14
Figure 10: Save Shopping List Popup 14
Figure 11: Shopping Lists Page 14
Figure 12: Plan my Route Page – Select Shopping List 15
Figure 13: Plan my Route Page – Preferences Screen 15
Figure 14: Route Optimization Result 15
Figure 15: Recommended Products 15
Figure 16: My Route Page Initial Layout State 16
Figure 17: Route Map Legends 16
Figure 18: Shopping List Initial Status 16
Figure 19: User Preferences Popup 17
Figure 20: Scan Popup 17
Figure 21: EAN/ Barcode Input 17
Figure 22: Preparing route popup 17
Figure 23: Updated User Location and Route 17
Figure 24: EAN input of a non-listed product 18
Figure 25: Product Found Popup 18
Figure 26: User's Suggested Route 18
Figure 27: Updated Shopping List 18

Figure 28: Shopping Complete Layout 19
Figure 29: User Feedback Popup 19
Figure 30: Completed Shopping List Layout 19
Figure 31: More Menu to restart the journey 19
Figure 32: Amsterdam Use Case - Dutch Layout 23
Figure 33: Amsterdam Use Case - English Layout 23
Figure 34: Wheelchair Preferences -1 24
Figure 35: Wheelchair Preferences -2 24
Figure 36: Wheelchair Preferences -3 24
Figure 37: Route Preferences -1 24
Figure 38: Route Preferences -2 24
Figure 39: Route Preferences -3 24
Figure 40: Generate Route Page 25
Figure 41: Address Search Input 25
Figure 42: Calculating Route 25
Figure 43: Calculating Route View 26
Figure 44: Recommended Route Page 26
Figure 45: Optimization Sample -1 26
Figure 46: Optimization Sample -2 26
Figure 47: Navigation Mode 27
Figure 48: Simulation Mode 27
Figure 49: Feedback Modal 27
Figure 50: Initial Page 31
Figure 51: Project Creation Page 31
Figure 52: Information Page 32
Figure 53: Information Page Details 32
Figure 54: Labelling Page 33
Figure 55: Labelling Page – Not filled 33
Figure 56: Labelling Page - Requirements met 33
Figure 57: Auto Labelling Processing Screen 34
Figure 58: Ambigous Pictures State 34
Figure 59: Auto labelling Process Completed 35
Figure 60: Experience Level Layout 35
Figure 61: Non-Technical User Threshold Settings 36
Figure 62: AI Expert Settings 36
Figure 63: Launch Parameters Page 37
Figure 64: Start Training Process 37
Figure 65: Threshold Settings 38
Figure 66: Heatmap Page 38
Figure 67: Finish Evaluation Popup 39

Figure 68: Datacation Login Form 43
Figure 69: WMS Status Page 44
Figure 70: WMS Orders List 44
Figure 71: Order Preview Page 45
Figure 72: Order Details Popup 46
Figure 73: AI Solution Explorer Page 46
Figure 74: AI-Solver's Solution Proposal State 47
Figure 75: Applied Solution Popup 47
Figure 76: Solver Configuration Popup 48
Figure 77: AI Assistant Chat Layout 48
Figure 78: Additional Options of the Application 49

1. Executive summary

1.1 Introduction

The PEER project is a European research and innovation initiative focused on designing and evaluating human-centric AI systems for sequential decision-making. The core objective is to develop a hyper-expert collaborative AI assistant that systematically enhances user trust, acceptance, and collaboration through responsive, tailored interactions. This deliverable report, D5.4, focuses on the second iteration of the technical integration and validation of the PEER collaborative AI assistant. While the first phase (MVP1) relied heavily on mock-ups and simulated AI behaviour (TRL 3), this second iteration (MVP2) formalizes the transition to fully developed, functional prototypes (TRL 4) capable of processing real data and generating computed explanations.

1.2 The PEER Project and its Objectives

The project is structured around fostering a bidirectional conversation between the user and the AI assistant, allowing the system to adapt dynamically to user preferences and constraints.

1.2.1 Purpose of D5.4 Deliverable Report

The D5.4 deliverable report specifically addresses the technical implementation and validation of the PEER collaborative AI assistant within four distinct use-cases (Sonae, City of Amsterdam, Proditec, and Datacation). The purpose of this report is to:

- Document the technical specifications and design of each MVP2 prototype based on the "Common Blueprint" architecture.
- Demonstrate the integration of the AI assistant into existing systems, moving from simulated environments to actual code implementations.
- Present the technical validation plans for the laboratory testing environments.
- Identify how the prototypes reflect the lessons learned from the first evaluation cycle, specifically addressing the need for enhanced comprehensibility.

1.2.2 Relation to WP5

Work Package 5 (WP5) is dedicated to "Learning: Piloting and Evaluation". WP5 focuses on implementing, applying, and validating the PEER collaborative AI assistant. D5.4 represents the second integrated release of the PEER assistant, where the technical framework for data and message flows of all use-cases has been enabled with an extended set of functions.

1.2.3 Relation to WP3

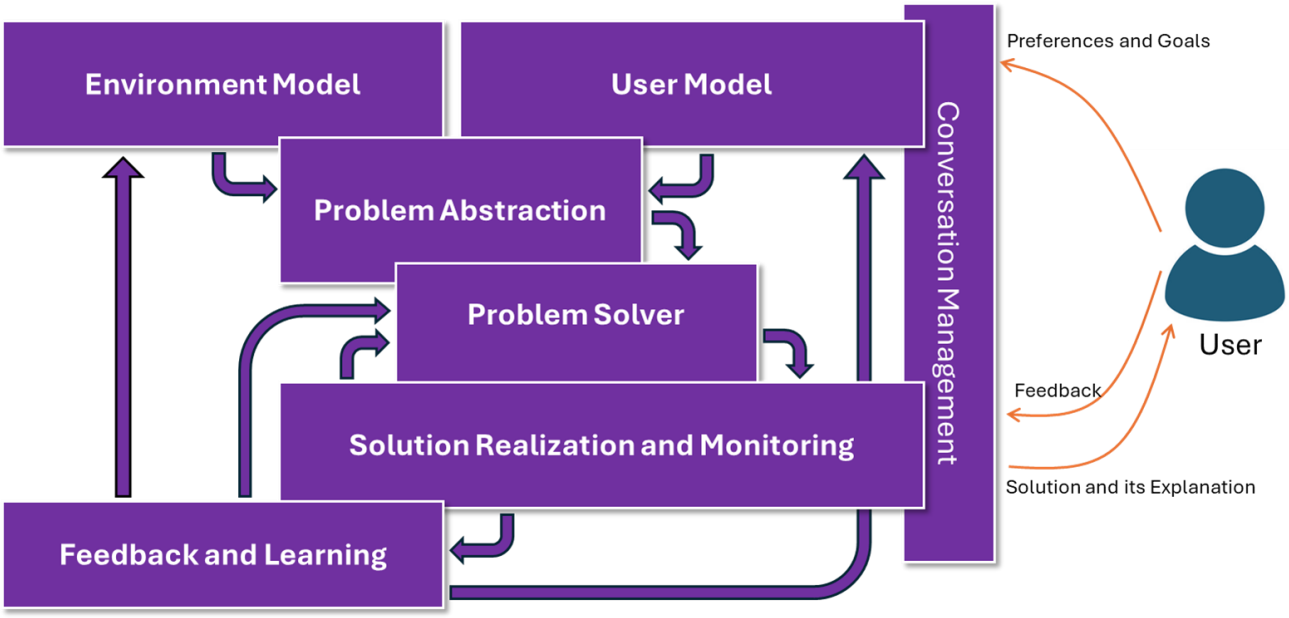


Figure 1: Architecture of the PEER Companion Agent

The functionality specified in Figure 1 report relies heavily on the collaborative AI components developed within Work Package 3 (WP3). The general architecture follows the building blocks of the PEER Companion Agent, which includes modules for Problem Abstraction, Problem Solver, Solution Realization, and Feedback Learning. D5.4 incorporates the first functional code implementations of these WP3 algorithms, including multi-objective pathfinding, mixed-integer programming, and initial Explainable AI (XAI) prototypes.

1.3 Overview of the 4 Use-Cases

The PEER project addresses four core use-cases across different domains:

- **Retail:** Personalized in-store shopping assistant (Sonae).
- **Smart & Inclusive City:** Personalized accessible routing for wheelchair users (City of Amsterdam).
- **Pharmaceuticals:** Human-in-the-loop anomaly detection trainer for quality control (Proditec).
- **Logistics:** Optimizing warehouse operations and layout configurations (Datacation).

1.3.1 Specific Goals of Each Use-Case

- **Sonae:** Deploy the foundation for static route planning and product recommendations. Focus is on establishing the simulation environment using Markov Decision Process (MDP) for example and testing human-AI interactions regarding adaptive routing.

- **City of Amsterdam:** Deploy the first working web-app prototype capable of generating routes based on live interaction and real geodata. A major goal is to provide computed explanations (interactive comparisons) instead of pre-calculated ones.
- **Proditec:** Improve user guidance and system comprehensibility for training anomaly detection models by using a standalone "local-first" application. A key goal is to replace complex terminology (e.g., "hyperparameters") with mainstream terms and implement semi-supervised labeling.
- **Datacation:** Provide a prototype web application that assists logistics engineers in optimizing warehouse configurations using an Integer Programming solver. The goal is to allow users to interactively enable/disable constraints and weigh efficiency versus safety regulations.

1.4 Common Cloud Infrastructure and CI/CD Management

To ensure a robust, secure, and highly scalable development lifecycle across the different Use-Cases, the PEER project utilizes a centralized Azure Cloud and Azure DevOps Environment managed by Fujitsu. This infrastructure acts as the backbone for continuous integration and continuous delivery (CI/CD) of the decoupled microservices.

1.4.1 Dedicated Build Environment

A dedicated, self-hosted build instance (“PEER-Build-Agent”) was provisioned within the Azure Environment. This isolated agent exclusively handles the build processes for all Use-Cases, ensuring that compilation, dependency resolution, and quality assurance checks (such as static type checking via MyPy and code formatting via Ruff) are executed consistently without impacting the runtime environments.

1.4.2 Automated CI/CD-Pipelines

Each microservice (Frontend, Orchestrator, integrated or wrapped AI-Solvers, Digital Twins, etc.) across the Use-Cases utilizes a standardized YAML-based Azure Pipeline.

- **Continuous Integration (CI):** When new or updated code is pushed to the main branch, the pipeline automatically spins up a virtual environment (venv), installs all the defined dependencies, runs structural tests, and packages the application into a lightweight release artifact (*.zip).
- **Continuous Delivery (CD) for Cloud-Hosted Use-Cases (AMS, DAT, SON):** For these three Use-Cases, the pipeline seamlessly connects to dedicated Azure Virtual Machines via secure SSH (managed via Azure Key Vault). The deployment program automatically extracts the

artifact, provisions a production-ready Python environment and registers or restarts the API-Services using Linux system background processes.

- **Special Case – On-Premise Deployment (PDT):** While the Sonae, City of Amsterdam and Datacation instances benefit from fully automated end-to-end deployments, the Proditec Use-Case requires a strict “local-first” execution on a Windows workstation. Consequently, the Proditec pipeline currently uses the CI-Stage to automatically validate and package the software artifacts. The automated CD-Stage is technically prepared but remains untested and temporarily suspended, pending the final provisioning of VPN- or RDP credentials for the isolated On-Premise hardware by the partner.

2. Description of Work

2.1 Sonae

2.1.1 Motivation

The size and complexity of Sonae's hypermarkets can make it difficult for customers to locate specific items, often leading to frustration and abandoned shopping trips. Sonae aims to upgrade its existing "Continente Siga" application to serve as an in-store navigation and shopping assistant. The objective of the PEER collaborative AI assistant is to provide optimized picking routes and context-aware product recommendations to enhance the shopping experience, reduce in-store time, and improve customer satisfaction.

2.1.2 Goals

The primary goal for MVP2 is to transition from the conceptual and simulated designs of MVP1 to a functional, decoupled software architecture based on the "Common Blueprint". The technical focus is on:

- Integrating a static planning algorithm for optimized route generation.
- Implementing a new dynamic Sequential Decision Making (SDM) policy for prescriptive product recommendations (suggesting the right product at the right moment along the route).
- Refactoring data integration to establish a "Single Source of Truth" using a dynamic Store-Graph.

2.1.3 Limitations

Integration into the live production environment of the "Continente Siga" app remains out of scope for MVP2; testing will continue in a controlled simulated/laboratory environment. Highly complex sequential decision-making capabilities—such as real-time dynamic re-routing based on live store crowding using full Reinforcement Learning—are deferred to the final MVP3 phase.

2.1.4 Usage Scenario

For MVP2, Sonae will test the in-store shopping assistant app internally with company employees acting as pilot users.

Each participant will create a shopping list (up to ~100 products from the available database with mapped locations). The app will generate an optimized in-store route, guiding the user through the shop layout.

During navigation, users will:

- Create and edit shopping lists, add or remove products, and select one as active to generate their optimized route.
- Browse recommended and promotional products tailored to their persona in the products page.
- Search and save favourite products to appear on the store map.
- Use the map to find their current location by scanning any nearby product.
- Build an optimized route, scanning barcodes to confirm item collection.
- Receive dynamic product suggestions based on their profile and current progress, while viewing the store layout and route updated in real time after each scanned item.
- These recommendations are not just predictive, but contextual - the AI evaluates options and suggests items that are geographically 'closest to the route', actively integrating the picking path with the customer's historical preferences.
- The system also records implicit and explicit feedback if the user accepts or rejects a recommendation.

Three customer personas will be used to represent typical product choices:

- **Mom**: prioritizes convenience items (baby food, household products, magazines).
- **Student**: buys quick and cheap meals (pasta, canned food, beer).
- **Nana**: focuses on healthy, affordable options (vegetables, yogurts, simple staples).

Each persona will correspond to a realistic customer ID with purchase history to simulate personalized recommendations.

2.1.5 Evaluation Journey

The MVP1 evaluation yielded a highly positive AI Acceptance (AIA) index score of 3.95 out of 5. Users praised the perceived technical functioning, user agency, and human-AI collaboration, though areas like data governance and privacy require further refinement. For MVP2, validation will be performed internally with Sonae employees acting as proxy users. The evaluation will focus on routing efficiency, the relevance and timing of the new SDM recommendations, and the clarity of the explanations provided by the system.

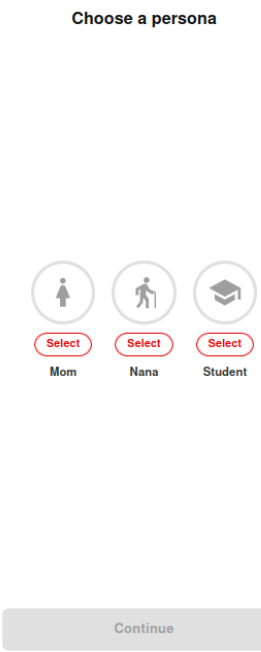


Figure 2: Available Personas

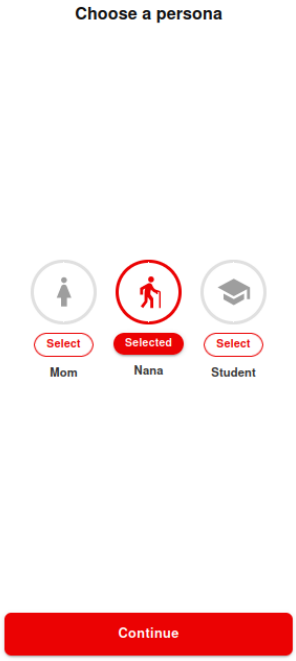


Figure 3: Selection of Persona



Figure 4: Loading Screen

The user journey begins with the selection of a persona. As previously described in 2.14, three personas are available: Mom, Nana, and Student. As shown in Figure 2 the available personas are represented,Figure 2 the user selects one of the personas as in Figure 3: Selection of Persona and proceeds by clicking the "Continue" button. A loading screen is then displayed as in Figure 4: Loading Screen while the application in the background loads the product catalogue, store layouts, and the user's shopping cart.

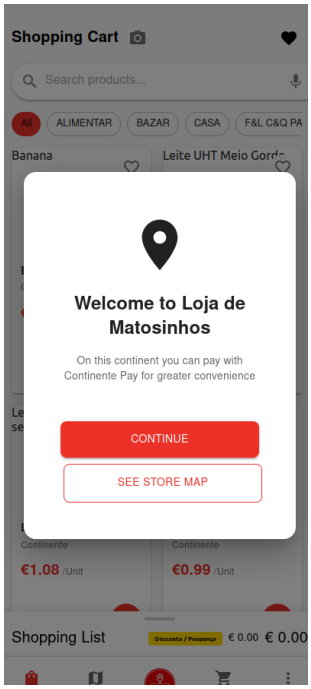


Figure 5: Welcome Popup

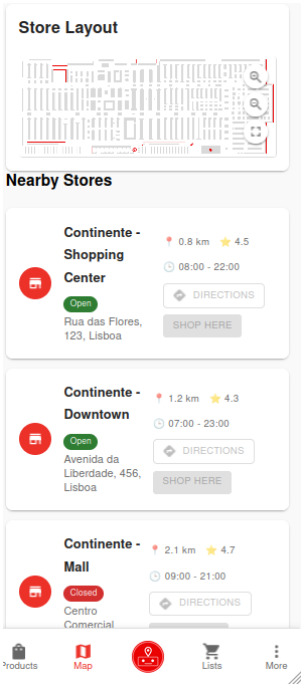


Figure 6: Store Layout

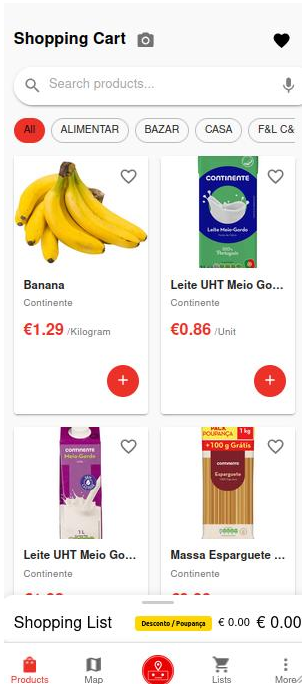


Figure 7: Shopping Cart Layout

The Sonae use-case architecture is designed to support multiple stores with different layouts. However, for this MVP, a single store has been selected for testing purposes. Once the data loading process is complete, the user is presented with a welcome pop-up as shown in Figure 5: Welcome

Popup for only the first time, where they can choose to either continue browsing products or view the store map is depicted in Figure 6: Store Layout.

On the “Shopping Cart” page, which is represented in Figure 7: Shopping Cart Layout, users can add products to their current shopping list using the “+” and “–” buttons, mark items as favourites, and search for products using text or voice input. Products can also be discovered by category selection on the top bar or by scanning barcodes by clicking the camera button.

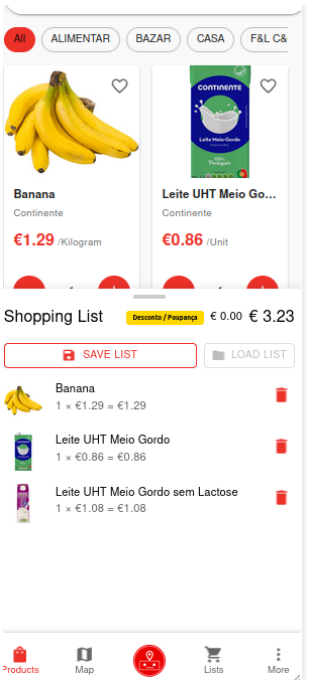


Figure 8: Save Shopping List in Shopping Cart Page

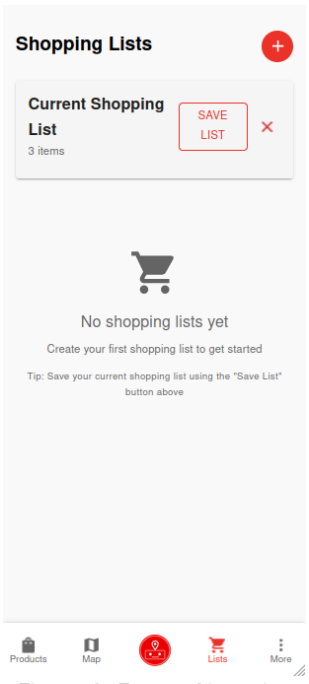


Figure 9: Empty Shopping List Page

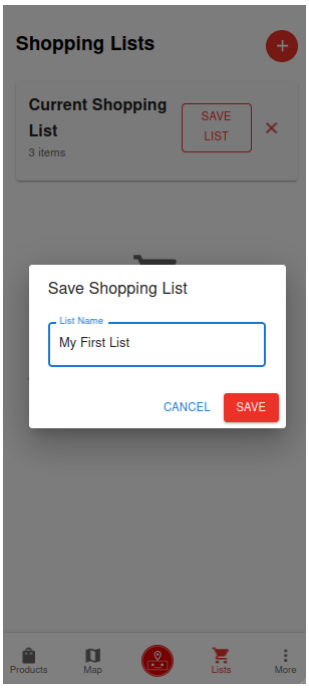


Figure 10: Save Shopping List Popup

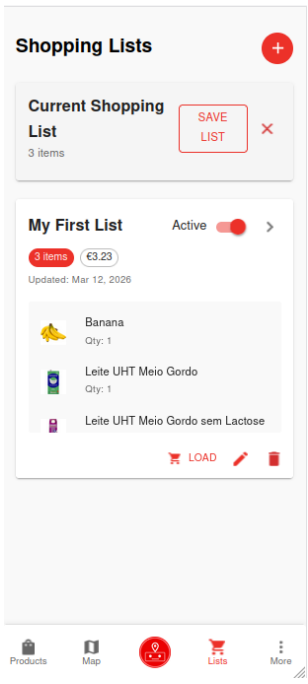


Figure 11: Shopping Lists Page

Users can save their shopping list either by selecting “Save List” as in Figure 8: Save Shopping List on the “Shopping Cart” page or by navigating to the “Lists” section as illustrated in Figure 9: Empty Shopping List Page via the bottom navigation bar. On the “Lists” page, users can create an empty list (Figure 10: Save Shopping List Popup), set lists as active or inactive, view completed lists, and edit or delete existing lists as shown in Figure 11: Shopping List.

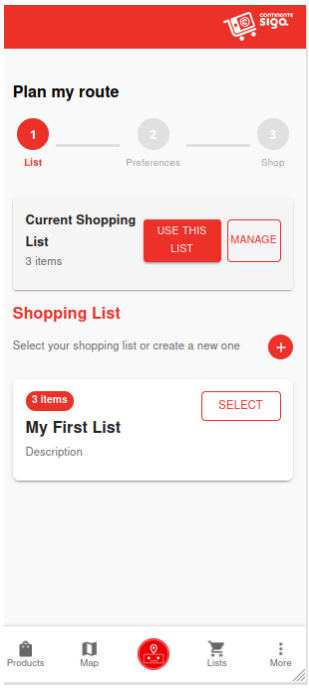


Figure 12: Plan my Route Page – Select Shopping List

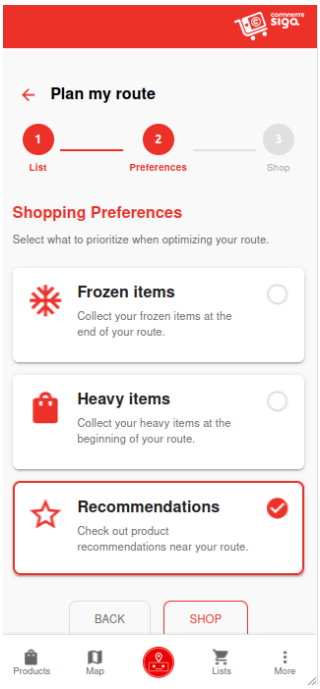


Figure 13: Plan my Route Page – Preferences Screen

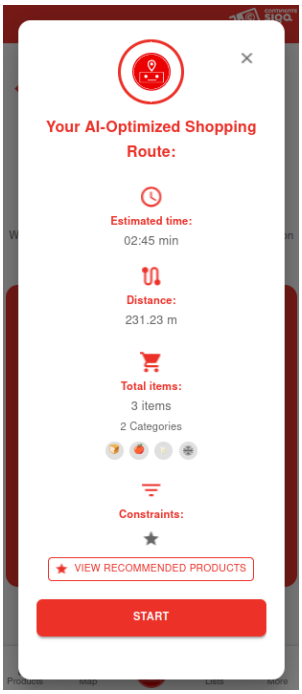


Figure 14: Route Optimization Result

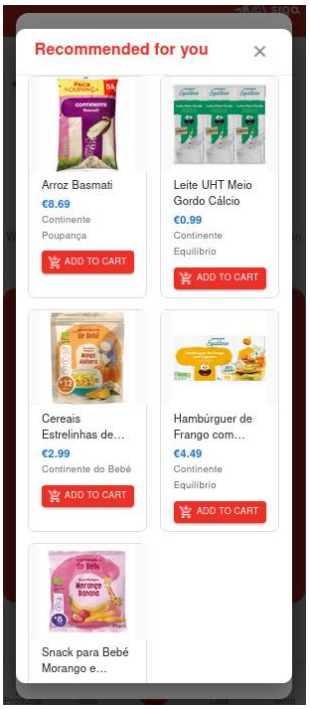


Figure 15: Recommended Products

The “Plan My Route” page can be accessed by selecting the AI logo from the bottom navigation bar is depicted in Figure 12: Plan my Route Page – Select Shopping List. This page follows a stepper-based flow consisting of three steps.

In the first step, users can choose from pre-created shopping lists or continue with their current shopping list. In the next step as presented in Figure 13: Plan my Route Page – Preferences Screen, users can select their route planning preferences, such as “Frozen Items”, “Heavy Items”, and “Recommendations”. Since the AI-based solver cannot generate an optimal route when frozen or heavy item constraints are applied for now, users are required to select the “Recommendations” option to get product recommendations.

After clicking the “Shop” button, the frontend sends the selected shopping list items along with the current store layout to the AI-Solver. The solver calculates the shortest possible route as shown in Figure 14: Route Optimization Result, while the recommendation engine provides five suggested products based on the user’s shopping history persona. Users can view these suggestions by selecting “View Recommended Products” and optionally add them to their current shopping cart as in Figure 15: Recommended Products. If the user adds these items to their current cart, optimal route is re-calculated.

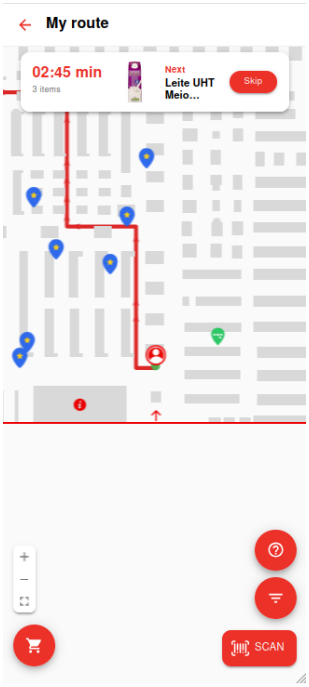


Figure 16: My Route Page Initial Layout State

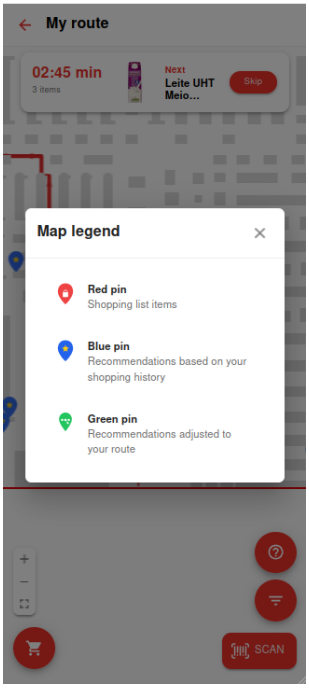


Figure 17: Route Map Legends

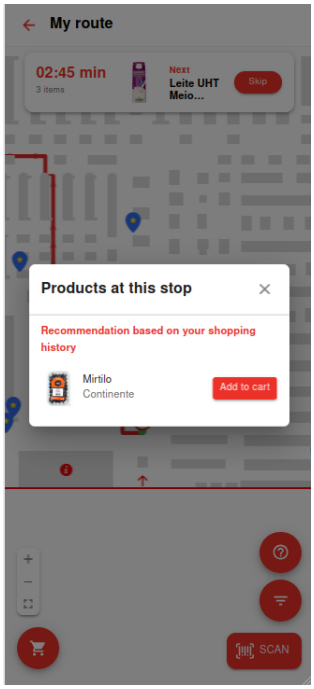


Figure 17: Recommended Product Popup

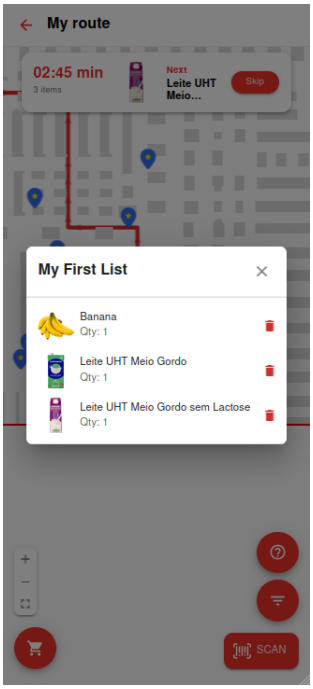


Figure 18: Shopping List Initial Status

Once the route is generated, users are guided through the store using the optimized path displayed on the map as demonstrated in Figure 16: My Route Page Initial Layout State. The active route dynamically highlights the next section to visit, helping users navigate efficiently through the store. Any changes to the shopping list during the journey - such as adding or removing items - trigger an automatic route update to ensure the navigation remains optimal throughout the shopping experience.

Users can view the map legend by selecting the question mark icon as outlined in Figure 17: Route Map Legends. The application currently supports three pin types: red pins, which represent items from the shopping list; blue pins, which indicate recommended products based on the selected persona; and green pins, is the closest recommended product adjusted of the users' route.

Users can select any pin to view detailed product information. Recommended products can be added directly to the shopping list, and once a recommended item is added, the route is automatically recalculated to reflect the updated list.

Users can view the status of their current shopping list by selecting the shopping list icon shown as Figure 18: Shopping List Initial Status. Items can be skipped either by selecting the delete option in the shopping list pop-up or by using the "Skip" button at the top of the screen to skip the next product. In both cases, the item is automatically removed from the shopping list, and the route is recalculated based on the user's current location.

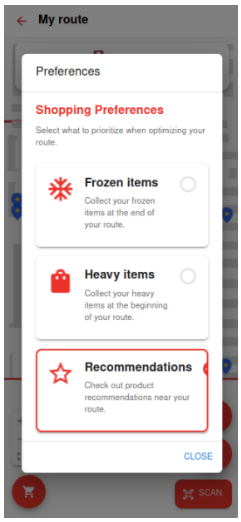


Figure 19: User Preferences Popup

Users can also review and manage their selected preferences highlighted in Figure 19: User Preferences Popup by selecting the filter button on the right side of the screen. This view provides an overview of the active preference settings - such as product handling constraints and recommendation options - allowing users to quickly verify or adjust their choices at any point during the shopping journey. Any updates to these preferences are applied immediately and may trigger a recalculation of the route to ensure the navigation remains within the user's selection.

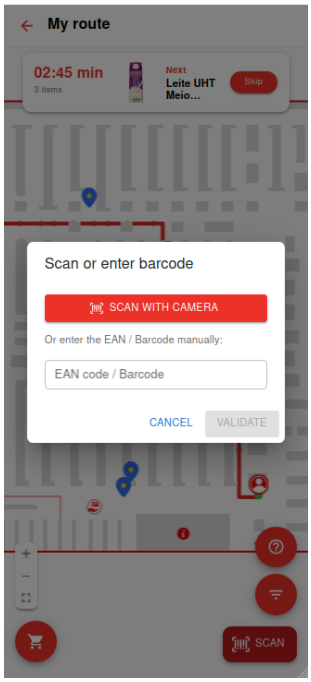


Figure 20: Scan Popup

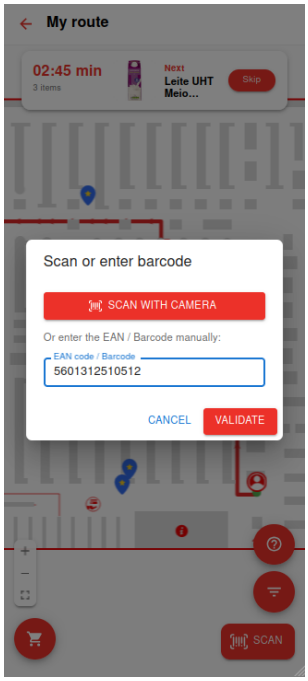


Figure 21: EAN/ Barcode Input

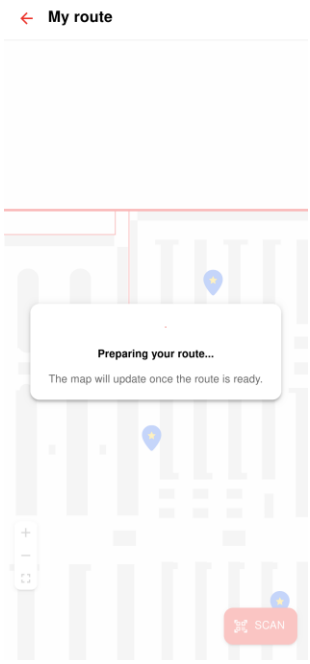


Figure 22: Preparing route popup

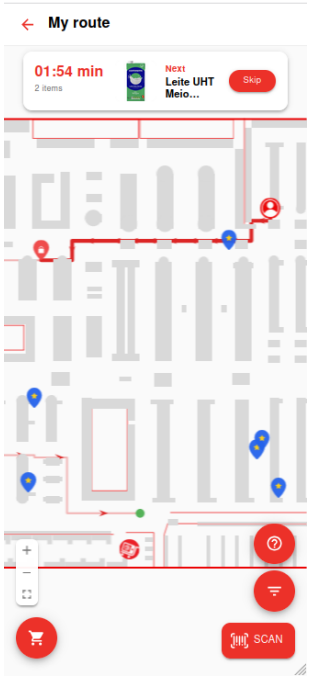


Figure 23: Updated User Location and Route

To update the user's current location and recalculate the route accordingly, users must either scan a product barcode as illustrated in Figure 20: Scan Popup or manually enter the barcode (Figure 21: EAN/ Barcode Input) into the text field. After user clicks the validate button, the route is recalculates based on the products and user's location that can be seen in Figure 22: Preparing route popup. This action allows the system to determine the user's position within the store and adjust the navigation path in real time simulated as shown in Figure 23: Updated User Location and Route.

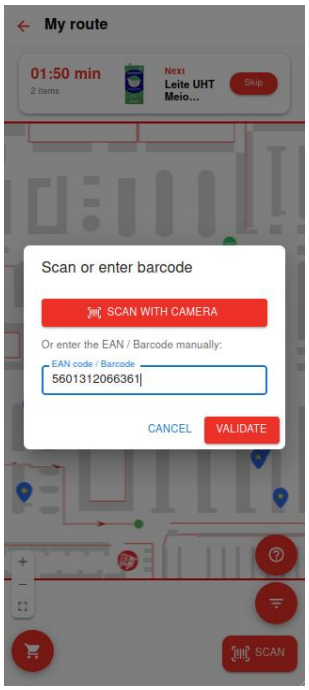


Figure 24: EAN input of a non-listed product

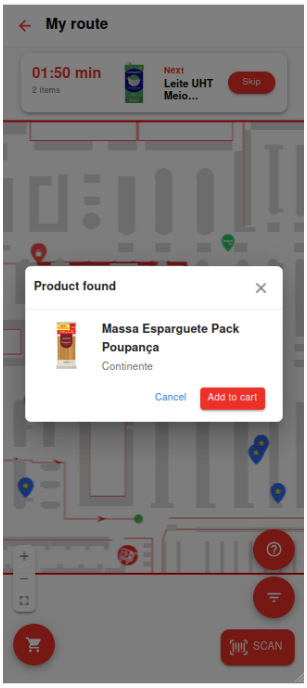


Figure 25: Product Found Popup

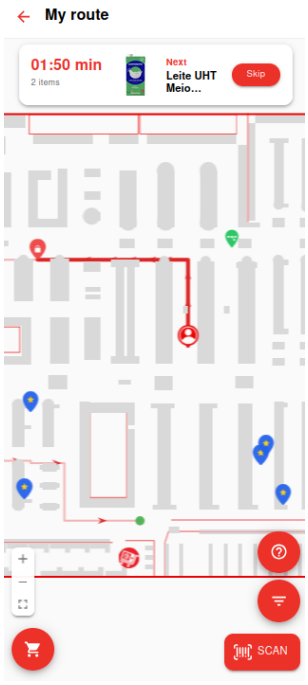


Figure 26: User's Suggested Route

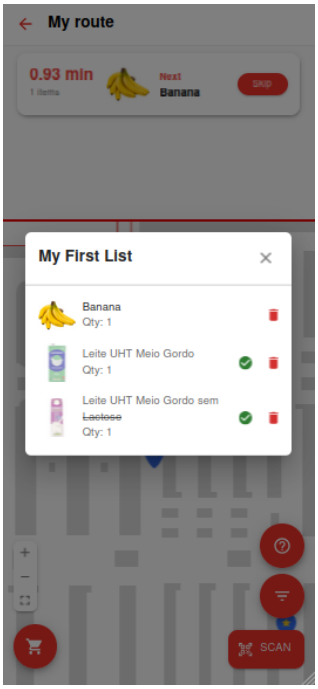


Figure 27: Updated Shopping List

If the scanned or entered product (Figure 24: EAN input of a non-listed product) is not already included in the shopping list, a "Product Found" pop-up as displayed Figure 25: Product Found Popup, giving users the option to add the item to their shopping cart. Once the product is added, the route is automatically recalculated to include the new item.

When users scan the next product in the suggested route, the item's status in the shopping list is updated to "scanned" (Figure 27: Updated Shopping List), indicating it has been collected. The route is then recalculated based on the remaining items in the list (Figure 26: User's Suggested Route , ensuring that navigation continues to follow the most efficient path for the rest of the shopping journey.



Figure 28: Shopping Complete Layout

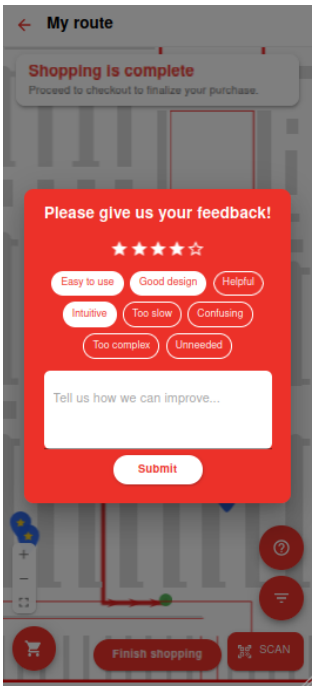


Figure 29: User Feedback Popup

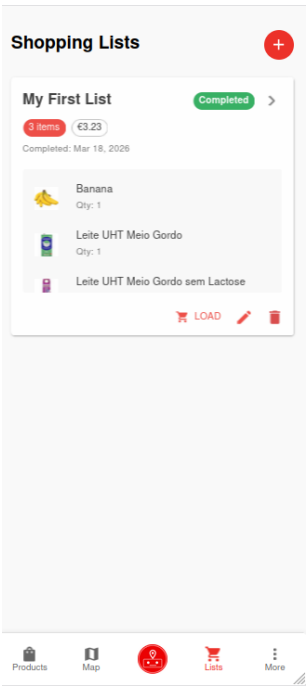


Figure 30: Completed Shopping List Layout

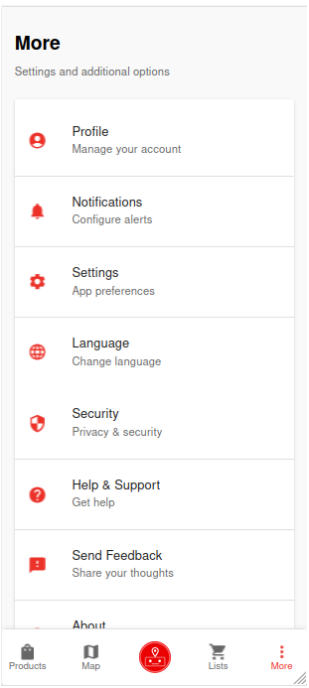


Figure 31: More Menu to restart the journey

Once all items have been collected, a final route is calculated and displayed to guide the user from their current location to the checkout area that can be seen in Figure 28: Shopping Complete Layout. By selecting the “Finish Shopping” button, the user indicates that the shopping process is complete and that they have reached the checkout zone. At this stage, a feedback pop-up form is presented to collect the user’s experience and impressions highlighted as Figure 29: User Feedback Popup.

After the user submits the feedback by clicking “Submit”, the shopping session for that list is finalized. The corresponding shopping list is marked as completed and can no longer be reused or modified illustrated as Figure 30: Completed Shopping List Layout.

If the user wishes to restart the application flow with a different persona, they can do so via the “More section” in the bottom navigation bar provided as the Figure 31: More Menu to restart the journey. By selecting “Exit”, the application is reset, allowing the user to begin a new session from the persona selection step.

2.1.6 Design of the MVP

The implemented MVP2 architecture for Sonae operates as a highly decoupled microservice environment adhering strictly to the project's Common Blueprint, hosted on a dedicated Linux Azure Virtual Machine.

2.1.6.1 Architecture

The system is built upon a highly distributed microservice architecture, strictly implementing the multi-tier blueprint defined in Deliverable D5.2. It features a ReactJS Frontend that communicates exclusively via REST APIs with a central Python-based FastAPI Backend (the Orchestrator). The Orchestrator is served asynchronously using Uvicorn, allowing it to handle multiple concurrent shopping sessions efficiently.

In accordance with the D5.2 specification, every computational or domain-specific component – collectively referred to as "Solvers" (e.g., the Routing Engine and the SDM process) – has been rigorously decoupled. Fujitsu successfully realized this conceptual separation by integrating each of these Solvers as its own distinct, self-contained microservice, running in an isolated virtual environment and exposing a complete, standardized RESTful API. This API-driven isolation ensures independent scalability, fault tolerance, and allows the Orchestrator to remain a lightweight state-manager.

2.1.6.2 Data Integration & State Management

The Orchestrator functions as the central hub for managing the application state, utilizing a local SQLite database for robust, relational data storage of user profiles, personas, and dynamic shopping lists.

To adhere strictly to the decoupled microservice architecture defined in Deliverable D5.2, all static and external data sources have been systematically isolated. Specifically, the Store-Layout data (topological graphs) and the comprehensive Product Database have been shifted exclusively into the scope of the "Routing" microservice. This ensures a clear separation of concerns: the Orchestrator exclusively maintains the transactional state (e.g. current cart status and scanned items), while the dedicated Routing service acts as the strict "Single Source of Truth" for spatial coordinates, distance matrices, and external product metadata.

Consequently, redundant mapping files from MVP1 were entirely eliminated, preventing data synchronization issues and accurately simulating a real-world integration where external enterprise systems feed directly into the specialized routing engine.

2.1.6.3 Routing & SDM Recommendation Service

To preserve the integrity of the operations research code provided by WP3, the mathematical routing algorithms and the Sequential Decision Making (SDM) policy are encapsulated in a separate FastAPI microservice.

The Orchestrator intelligently pre-filters persona-based product recommendations (ensuring no items already on the shopping list are suggested) before passing the valid candidates to the solver. The solver then computes the shortest path and applies the “closest” heuristic to return the final specific recommendation.

To bridge the gap between academic research and enterprise software, Fujitsu developed this dedicated API wrapper and fully integrated it into the automated CI/CD-Pipeline, deploying directly to the Azure VM environment. A key architectural decision was to maintain a strict 1:1 copy of the original WP3 codebase within the wrapper's internal structure. This decoupling ensures that any future algorithm updates, modifications, or enhancements provided by WP3 can be seamlessly integrated by simply replacing the internal files, requiring zero to minimal adjustments to the surrounding integration layer.

2.1.6.4 Auditability & Safety

An immutable Decision Log (using SQL hash chains) is implemented via local SQLite to capture user feedback (e.g. accepted/rejected recommendations) for future model training.

Additionally, a "Safety Net" functionality utilizing static fallback JSONs has been implemented to ensure robust and uninterrupted demonstrations during testing. This Safety Net was specifically designed to prevent critical application failures caused by data inconsistencies, such as mismatches between outdated product IDs in the historical data and newly introduced test products in the current catalog.

Instead of relying dynamically on the live WP3 recommendation API – which might fail if product mappings are out of sync – the system can utilize these statically generated "persona-based top-20 recommendations". To provide maximum flexibility during evaluations, this entire fallback mechanism is controlled by a single Boolean switch within the global configuration file, allowing administrators to instantly toggle between the live AI recommendation process and the static safety net at any time.

2.1.7 Responsibilities

The technical implementation is distributed across the consortium as follows:

- **SON (Sonae):** Lead on requirements definition, UI/UX mockups, provision of the product catalog and Store-Graph data, and execution of internal user tests.
- **INESC TEC & CU:** Development of the preference-oriented routing mathematical models, the static routing solution, and the foundational simulation environment.
- **VUB (AI):** Development of the product recommendation models and the Sequential Decision Making (SDM) policy.
- **FUJ (Fujitsu):** Development of the ReactJS Frontend and FastAPI Orchestrator, integration of the AI-Solver REST APIs, and implementation of the immutable database schema.
- **CATIE & VUB (SMIT):** Support in defining evaluation scenarios, KPIs, and measuring user trust through the AIA index.

2.2 City of Amsterdam

2.2.1 Motivation

People with reduced mobility (e.g. wheelchair users) face significant and persistent challenges when navigating urban environments. Even in a city like Amsterdam, where accessibility is an explicit policy priority, citizens regularly encounter obstacles such as narrow sidewalks, uneven surfaces, steep curbs, inaccessible crossings, or temporary disruptions due to construction. These barriers reduce autonomy, undermine confidence, and can make even short trips unpredictable or unsafe.

This use case aims to empower wheelchair users by providing personalized, accessible navigation that reflects their individual mobility needs, using the city's rich 3D accessibility data. It addresses urgent societal challenges around autonomy, safety, and inclusion, while serving as an ideal testbed for explainable, human-centric AI in sequential decision-making. Through interactive feedback and transparent explanations, the system is supposed to build trust and support more equitable urban mobility.

2.2.2 Goals

The primary goal for MVP2 is to transition from the "Wizard-of-Oz" Figma mock-ups used in MVP1 to a functional, interactive prototype (front- and back-end web application). The technical focus is on generating accessible routes dynamically using real geographic data and providing "computed explanations." Rather than using complex separate XAI models, the system will explain its choices by directly comparing alternative routes and their trade-offs (e.g., highlighting that a suggested route has fewer crossings but takes slightly longer).

2.2.3 Limitations

Live GPS tracking and real-time on-the-road re-routing are excluded from MVP2 and strictly deferred to MVP3. Furthermore, the implicit memory function - where the AI automatically learns and updates user preferences based on unprompted route deviations - is also deferred. Finally, to ensure rapid iteration and testing of the AI algorithms, MVP2 is developed as a responsive web application rather than a native mobile app.

For this use case, the application operates on a restricted geographic dataset that has been prepared specifically for evaluation and testing purposes. The dataset represents the Osdorp-Midden neighbourhood in Amsterdam and includes detailed accessibility and environmental information only for this area. As a result, the system does not support city-wide address coverage in this version.

2.2.4 User Journey

Users interact with the application primarily in a pre-journey planning phase (e.g., at home). During onboarding, they input their specific mobility profile (wheelchair dimensions, manual vs. motorized). For a specific trip, they enter an origin and destination and set their accessibility preferences, such as a desire for fewer crossings, avoidance of steep inclines, or a preference for bike lanes over sidewalks.

The system calculates and presents an optimal route alongside alternatives. The user can interactively adjust preference sliders to explore different trade-offs, prompting the AI to dynamically generate new route variants. After reviewing or completing a route, the user can provide explicit feedback (e.g., a star rating or flagging a blocked sidewalk), which the system logs for future learning.

2.2.5 Evaluation Journey

Testing will take place in a controlled laboratory environment using actual wheelchair users recruited in collaboration with Cliëntenbelang Amsterdam. The evaluation will focus on the usability of the interface, how well users understand the computed explanations (trade-offs between different route attributes), and their overall trust in the system's recommendations.

The application is developed with multi-language support, allowing users to change the language at any point within the app. An example is provided in Figure 32: Amsterdam Use Case - Dutch Layout and Figure 33: Amsterdam Use Case - English Layout.



Figure 32: Amsterdam Use Case - Dutch Layout

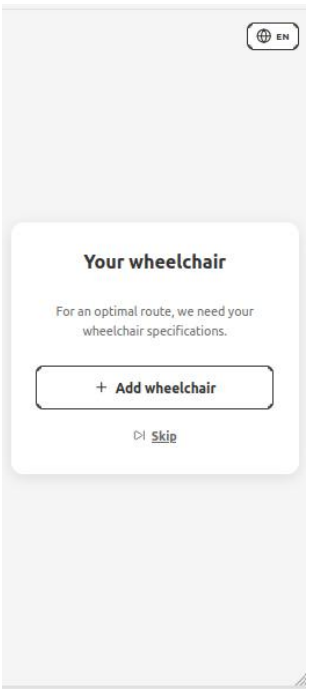


Figure 33: Amsterdam Use Case - English Layout

User journey begins with the configuration of wheelchair preferences. Users can select their wheelchair type and brand, as well as define the overall width and length of the wheelchair.

Once the wheelchair specifications are configured, the application uses this information to tailor the navigation experience to the user's mobility needs. These parameters play a crucial role in ensuring that all calculated routes account for the user's physical constraints, such as required turning radius and accessible path widths. By personalizing the routing engine with these details, the system can avoid unsuitable pathways and prioritize safe, comfortable navigation throughout the environment.

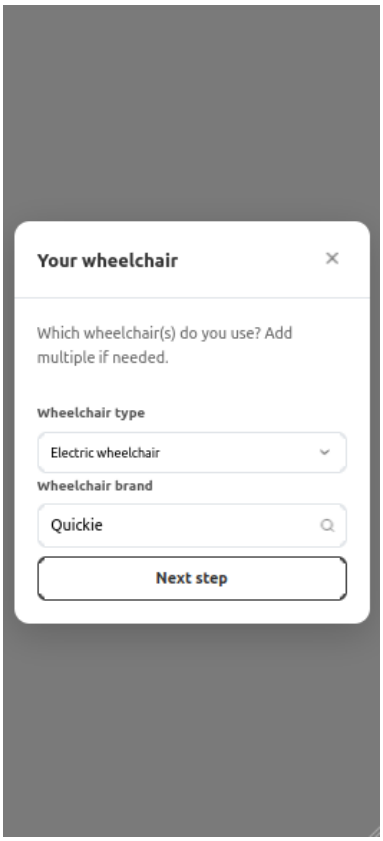


Figure 34: Wheelchair Preferences -1

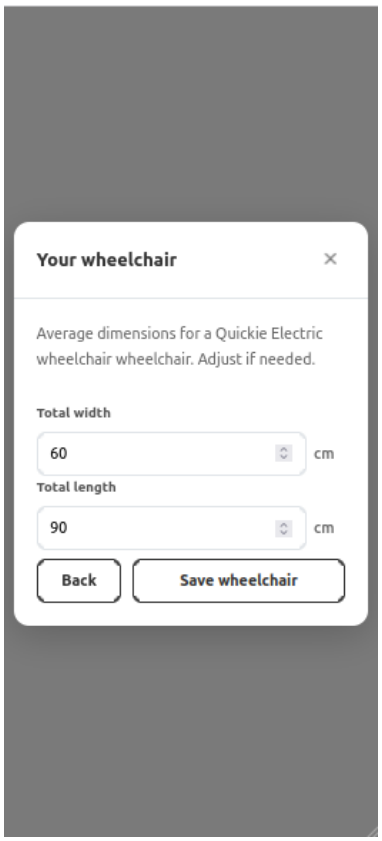


Figure 35: Wheelchair Preferences -2

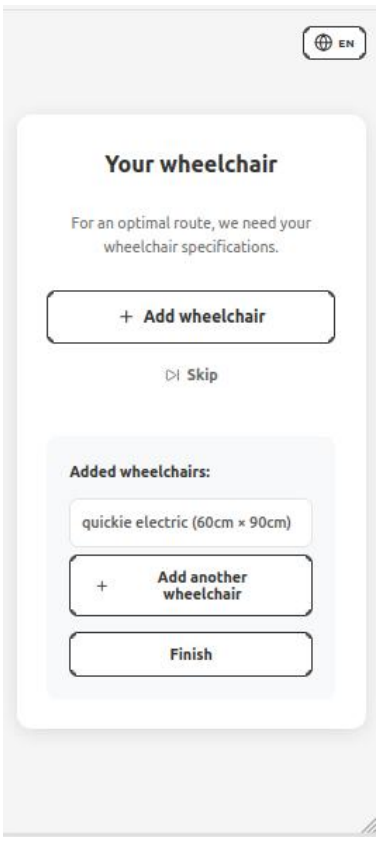


Figure 36: Wheelchair Preferences -3

After saving the wheelchair configuration, users proceed to the preference setup step, where they can indicate their individual needs and priorities. Figures 34–36 provide an overview wheelchair preferences help refine route suggestions by considering factors such as surface smoothness, curb accessibility, or avoidance of steep inclines. This step ensures that the generated routes are not only technically feasible but also aligned with the user's comfort level and mobility requirements.

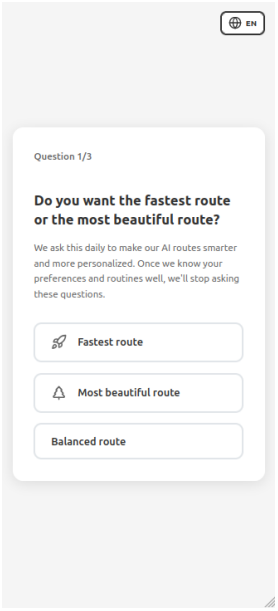


Figure 37: Route Preferences -1

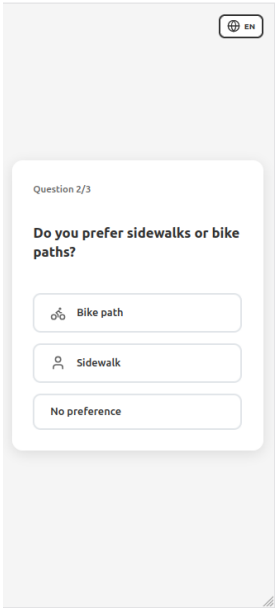


Figure 38: Route Preferences -2

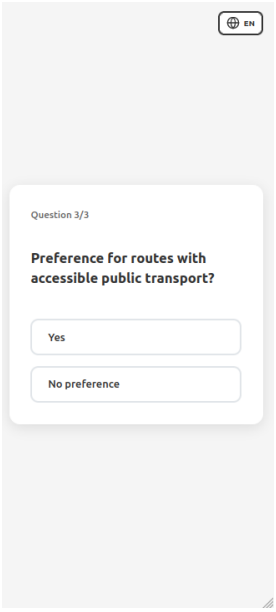


Figure 39: Route Preferences -3

Once the user's wheelchair details and routing preferences (Figure 37-39) have been saved, the user can proceed to select their current starting point and define their desired destination. This step allows the system to generate an accessible route that reflects both the user's mobility constraints and personal preferences.

As mentioned in chapter 2.2.2, the dataset used in this use-case is based on the "Osdorp-Midden" neighbourhood in Amsterdam. Therefore, the address search functionality is limited to streets and locations within this specific area. This scoped dataset ensures consistent and reliable routing results while allowing the system to apply accurate accessibility constraints derived from the underlying environment model.

When the user selects both the starting point and the destination, the interface provides visual confirmation of the chosen locations on the map. This ensures that users can verify their selections before proceeding, reducing the likelihood of errors - especially important given the limited geographic scope of the dataset. The system also validates that both points fall within the supported Osdorp-Midden area, helping maintain data consistency and preventing route calculations outside the defined boundaries.

After the location selection is confirmed, the application prepares the necessary routing parameters in the background. These parameters include the user's wheelchair dimensions, personal routing preferences, and any accessibility constraints defined earlier. By combining this personalized information with the structured neighbourhood dataset, the system ensures that the calculated routes are not only accurate but also tailored to the user's individual mobility needs. This approach enables a navigation experience that is both reliable and inclusive, even in a constrained dataset environment.

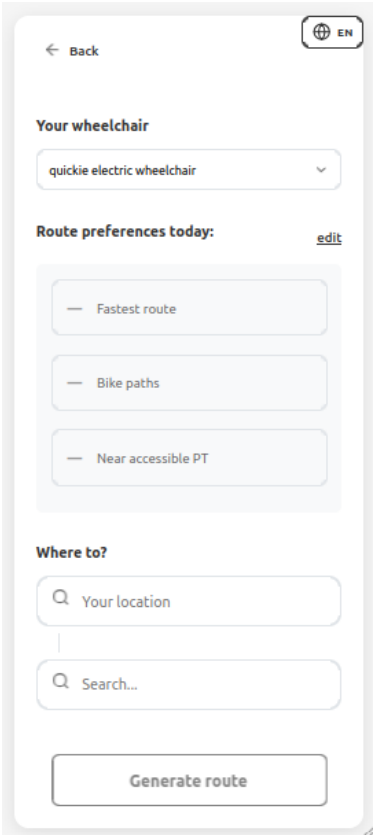


Figure 40: Generate Route Page

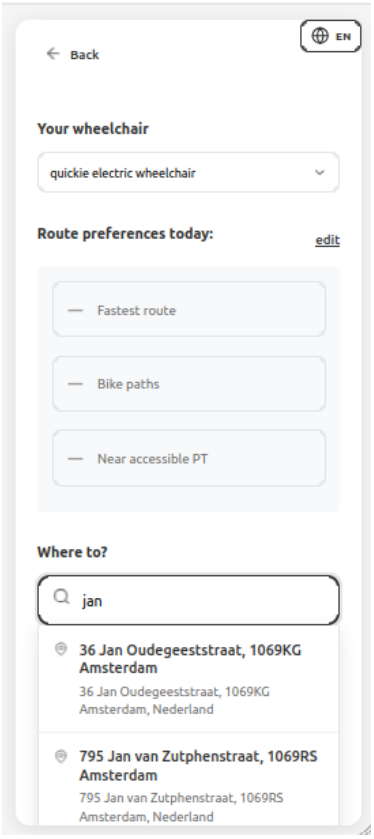


Figure 41: Address Search Input

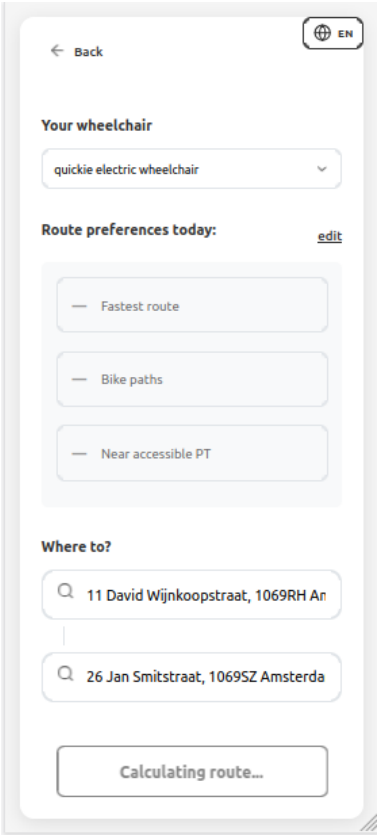


Figure 42: Calculating Route

After the preferences, user will be navigated to Generate Route Page as represented as Figure 40: Generate Route Page. As the user begins typing an address, the system displays a list of filtered addresses (Figure 41Figure 40) within the supported neighbourhood, allowing the user to quickly select a destination without entering the full address manually. This improves usability while ensuring that all selected locations remain within the scope of the available dataset. Once both the starting location and destination are selected, the user can initiate route generation by clicking the Calculate Route button. This action triggers a backend request as shown in Figure 42: Calculating Route that calculates an accessible route between the two selected coordinates, considering the user's wheelchair configuration and routing preferences.

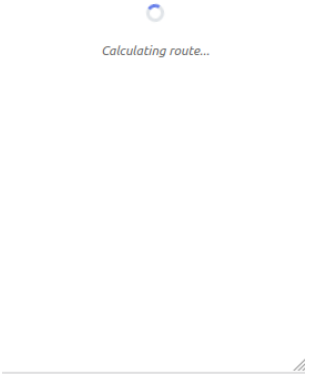


Figure 43: Calculating Route View

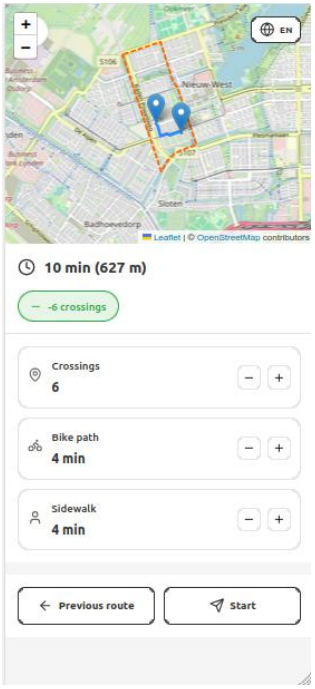


Figure 44: Recommended Route Page

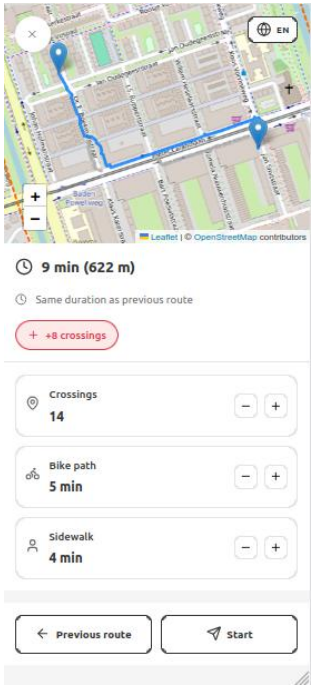


Figure 45: Optimization Sample -1

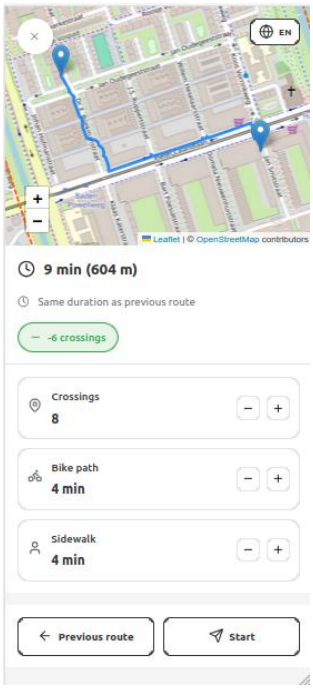


Figure 46: Optimization Sample -2

User will see the calculating route view (Figure 43: Calculating Route View) till the calculation is done. Once the backend returns the calculated route, it is displayed in the frontend (Figure 44: Recommended Route Page) for the user to follow. The current route is visualized as a blue line, and the interface also highlights how many crossings the user must navigate to reach their destination. In addition, the backend provides several detailed metrics, including the total distance in meters, the estimated travel time in minutes, and the breakdown of how much of the route consists of bike paths versus sidewalks. These metrics help users better understand the accessibility and structure of their chosen route.

Users can further refine their experience by adjusting their routing preferences directly in the interface. By increasing or decreasing preference values - such as prioritizing smoother surfaces, avoiding steep paths, or preferring wider sidewalks - an updated request is sent to the backend. The application then recalculates the route based on these new parameters shown as Figure 45: Optimization Sample -1 and Figure 46: Optimization Sample -2. To support comparison, both the newly generated route and its associated metrics can be viewed alongside the previous ones shown in grey enabling users to evaluate how their preference changes impact accessibility, distance, and overall travel experience.

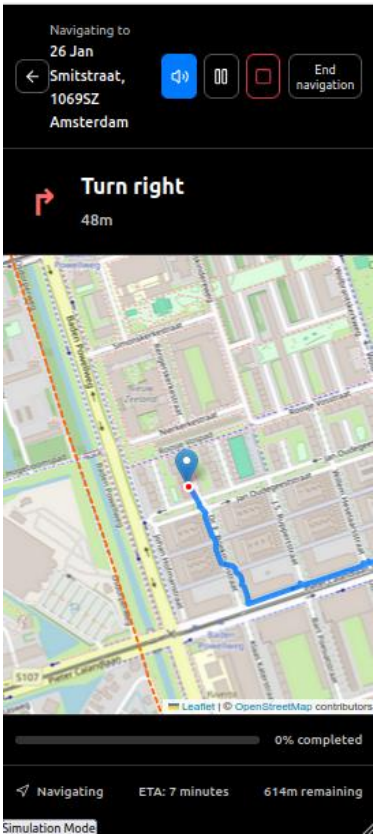


Figure 47: Navigation Mode

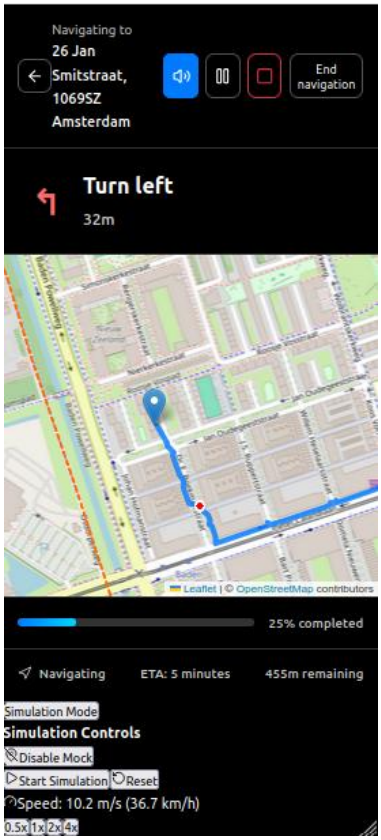


Figure 48: Simulation Mode

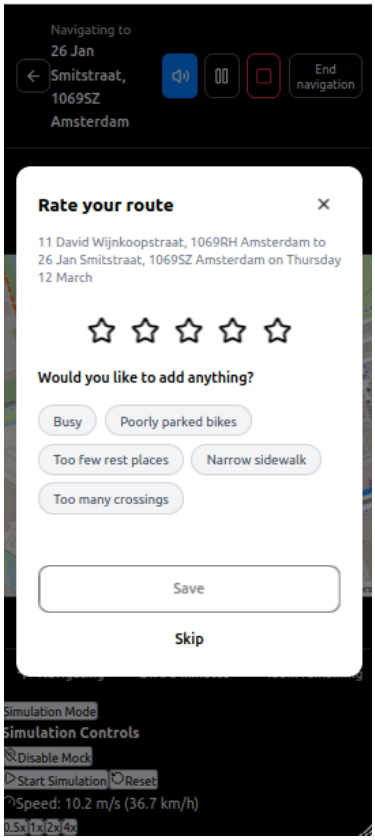


Figure 49: Feedback Modal

Once the user selects the “Start” button, the “Navigation Viewer” page is launched as depicted in Figure 47: Navigation Mode. At this stage, the application requests the user’s current location to provide a more accurate estimation of the remaining travel time. After the system detects that the user has reached their starting position, they can begin following the visualized route step by step. When the user reaches the destination and the route is successfully completed, a feedback pop-up appears, inviting them to share their experience and rate the navigation quality as shown in Figure 49: Feedback Modal.

For users who are not physically located within the supported Amsterdam neighbourhood and therefore cannot test the navigation in real conditions, the application offers a Simulation Mode as shown in Figure 48: Simulation Mode. By enabling the mock-location option, users can set a virtual starting point within the Osdorp-Midden area and initiate a simulated navigation session. Additionally, the simulation provides controls to adjust the walking speed, allowing users to observe how the route progression and estimated time adapt to different movement speeds. This feature ensures that all users - including testers, developers, and evaluators - can experience and validate the routing behaviour regardless of their actual geographic location.

2.2.6 Design of the MVP

The MVP2 for the City of Amsterdam (AMS) transitions the initial "Wizard-of-Oz" concepts into a fully functional, event-driven web application. Operating within a highly decoupled microservice environment, it strictly adheres to the project's Common Blueprint and is hosted on a dedicated Linux Azure Virtual Machine.

2.2.6.1 Architecture

Building upon the overarching multi-tier microservice architecture defined in D5.2, the City of Amsterdam (AMS) implementation is specifically designed to handle highly dynamic user states.

The ReactJS frontend utilizes Redux Toolkit to manage mobility profiles (e.g. wheelchair dimensions, type of mechanism) and continuously adjustable accessibility preferences. It communicates asynchronously via REST APIs with the Python-based FastAPI Orchestrator.

While sharing the same foundational blueprint as the retail Use-Case (SON), the critical architectural focus here is on offloading geographic computations. By isolating the WP3 multi-objective routing engine into its own dedicated Solver microservice, the Orchestrator remains a highly responsive, lightweight state-manager that never blocks the user interface during intensive spatial calculations.

2.2.6.2 Data Integration & Environment Model

The application enforces a strict separation between visual rendering and computational routing data. For real-time based map visualization and address resolution, the frontend interacts directly with the external OpenStreetMap (OSM) API.

Conversely, the isolated AI-Solver microservice acts as the sole administrator of the "Environment Model." This model is a specialized, computationally topological graph enriched with highly granular municipal accessibility metrics (e.g. precise path widths, surface conditions, and crossing layouts provided by the City of Amsterdam). For this reason, the address selection is also limited to this specially defined “Neighborhood-Data”, the scope and accuracy of which are derived exclusively from the topological model provided and therefore do not reflect the complete OSM address space.

2.2.6.3 Routing Service & Computed Explanations

Encapsulating the WP3 algorithms within a dedicated FastAPI wrapper allowed us to seamlessly integrate the routing logic into the automated CI/CD-Pipeline while maintaining a strict 1:1 copy of the original research code.

For the generation of routes and explanations, the MVP implements an interactive mechanism based on the WP3 “Preference Guided Iterated Pareto Referent Optimisation” (PG-IPRO) algorithm. Instead of generating multiple routing alternatives concurrently or utilizing a separate post-hoc Explainable AI (XAI) model, the system employs an iterative, user-steered approach.

Following the generation of an initial route based on hard spatial constraints (e.g. minimum path width, maximum curb height), the user can sequentially adjust specific routing objectives, such as increasing (+) or decreasing (-) the number of street crossings or the overall path length. The Orchestrator queries the Routing-Solver for the updated route and serves both the previous and the newly generated route to the frontend.

The ReactJS interface then renders a direct visual comparison of these competing routes on the map, programmatically calculating and displaying the trade-offs (e.g. “3 fewer crossings” versus “30 meters longer”). This iterative, comparative visualization functions as a “computed explanation”, enabling the user to directly observe the cause-and-effect relationship of their parameter adjustments on the routing outcome.

2.2.6.4 Auditability & Safety

An immutable Decision Log, utilizing cryptographic SQL hash chains, is implemented within the central SQLite database to securely capture all user interactions. Tailored specifically to urban navigation scenarios, this logging mechanism systematically records every dynamic adjustment made to the user's accessibility constraints, the iterative parameter shifts submitted to the Routing algorithm, and the user's final route selection. Furthermore, it captures all explicit in-app feedback, such as route ratings. This tracking establishes a fully transparent audit trail that is critical for facilitating the upcoming laboratory evaluations and building a reliable empirical dataset for continuous algorithmic refinement by WP3.

From a safety and data privacy perspective, the decoupled architecture ensures that sensitive user data, such as specific mobility limitations or wheelchair dimensions, is processed exclusively within the isolated backend environment. When the frontend or Orchestrator queries the external OpenStreetMap (OSM) API for address resolution or base map rendering, only generic geographic coordinates and bounding boxes are transmitted. This strict data segregation prevents the exposure of personalized health or mobility profiles to third-party services, ensuring compliance with strict data protection requirements during the evaluation phase.

2.2.7 Responsibilities

The technical implementation is distributed across the consortium as follows:

- **AMS (City of Amsterdam):** Lead on UI/UX design and provision of the specialized accessibility datasets and routing graphs.
- **FUJ (Fujitsu):** Development of the ReactJS Frontend, the FastAPI Orchestrator, data modeling, and implementation of the immutable audit logs.
- **VUB-AI, TU/e & CU:** Development of the multi-objective routing algorithms, the AI-Solver microservices, and the explanation logic.
- **CATIE & VUB-SMIT:** Support in defining evaluation scenarios, executing lab tests with end-users, and measuring acceptance through the AIA index.

2.3 Proditec

2.3.1 Motivation

Proditec's automatic inspection machines for the pharmaceutical and minting industries require complex, time-consuming manual setup ("recipe setting") by skilled operators. The overarching motivation for this use-case is to design a self-learning AI system using anomaly detection algorithms that makes the defect detection process simpler, faster, and more efficient. The system aims to bridge the gap between operator knowledge and advanced technology by providing an intuitive AI model trainer companion that increases operator trust, reduces setup time, and ensures a high level of system acceptance.

2.3.2 Goals

The primary goal for MVP2 is to transition from the mock-ups evaluated in MVP1 to a functional, "local-first" software implementation that operationalizes a simplified user experience. A key objective is to significantly improve comprehensibility by replacing technical jargon (such as "hyperparameters") with mainstream terms and explicit guidance (using metrics like “False Positives” and “False Negatives”). Technical goals include integrating a semi-supervised labelling module based on the PatchCore algorithm and visualizing the consequences of parameter changes through heatmaps.

2.3.3 Limitations

To meet strict industrial data privacy and latency requirements, the solution is strictly limited to a standalone, offline environment without any cloud dependencies. While the semi-supervised labelling module is fully integrated, the advanced AutoML hyperparameter optimization module, threshold adjustment and heatmap generation remains simulated for MVP2 to ensure the stability of the core human-in-the-loop workflow.

2.3.4 User Journey

Once the operator launches the application, the system guides them into the model-development workflow. The first phase focuses on building the initial dataset required for training. To achieve this, the operator simply runs the machine using an appropriate data set of products.

With the raw dataset collected, the application shifts toward creating high-quality training labels through a human-AI collaboration process. Rather than expecting the operator to manually classify thousands of samples, the system requests only a small set of around twenty labeled examples for each class (good, bad). These labels serve as the foundation for the semi-supervised learning module, which then automatically annotates the remainder of the dataset. The operator is subsequently invited to review a selection of these auto-generated labels, adjusting any mistakes. This ensures that the data set achieves both coverage and reliability while keeping the user’s workload minimal.

At this stage, the application introduces the operator to hyperparameter interpretation. The system presents key hyperparameters and their influence on model behavior, supporting the operator’s understanding without requiring expert knowledge. Automated machine learning techniques then take over, exploring the hyperparameter space and identifying configurations most likely to improve model performance. Throughout this process, visual tools offer the operator transparency regarding why certain configurations are selected.

Once the optimized configuration is established, the operator initiates the training of the model. Following training, the system provides an in-depth visualization of the model’s behavior. Using SHAP-based explanations, the application highlights the specific regions of each product that contributed to the model’s classification decision.

To evaluate the model’s performance, the system generates a confusion matrix summarizing accuracy, false positives, and false negatives. The operator reviews these results to determine whether the performance is sufficient for operational use. If accuracy does not meet expectations, the workflow transitions into an active learning phase. Here, the operator reviews misclassified samples, corrects the labels, and reintegrates these refined examples into the dataset. This iterative cycle, potentially combined with threshold adjustments, allows the model’s performance to gradually improve.

Once the operator is satisfied with the model's accuracy, the system finalizes the workflow by saving both the trained model and the selected threshold in the database. The model is exported as an ONNX file and transferred from the main workstation to the production server. At this point, the model is ready for deployment in real inspection operations.

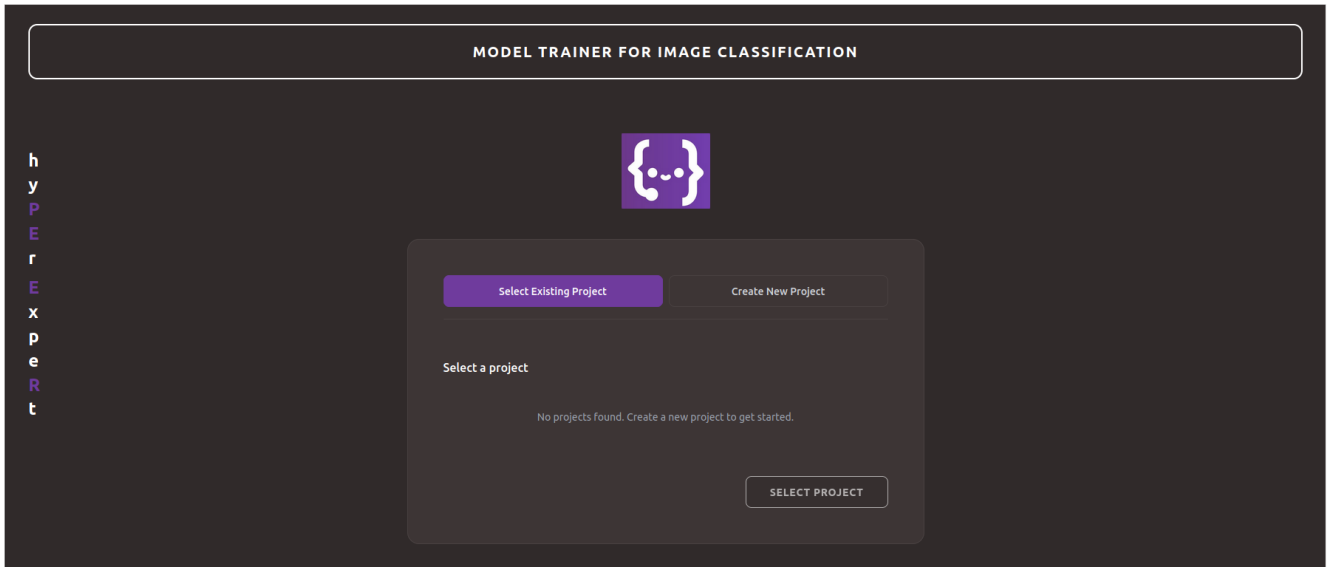


Figure 50: Initial Page

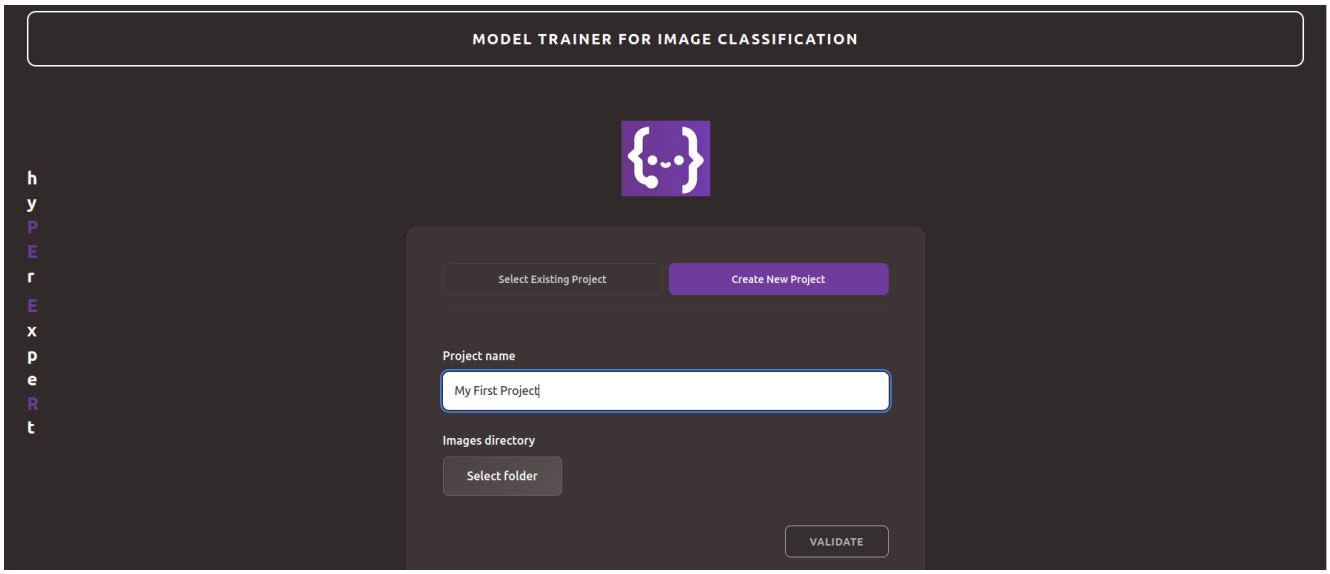


Figure 51: Project Creation Page

The user journey begins with the creation of a project. On the initial screen represented Figure 50: Initial Page, users can either select an existing project from the list or create a new one as designed as in Figure 51: Project Creation Page. To create a new project, users simply enter a project name and choose the directory that contains the images they want to work with.

This step establishes the foundation for the workflow, ensuring that each project is organized with its own dedicated name and image source. Once the project is created or selected, the application prepares the workspace and loads the required files, allowing the user to proceed seamlessly to the next stage of the process.

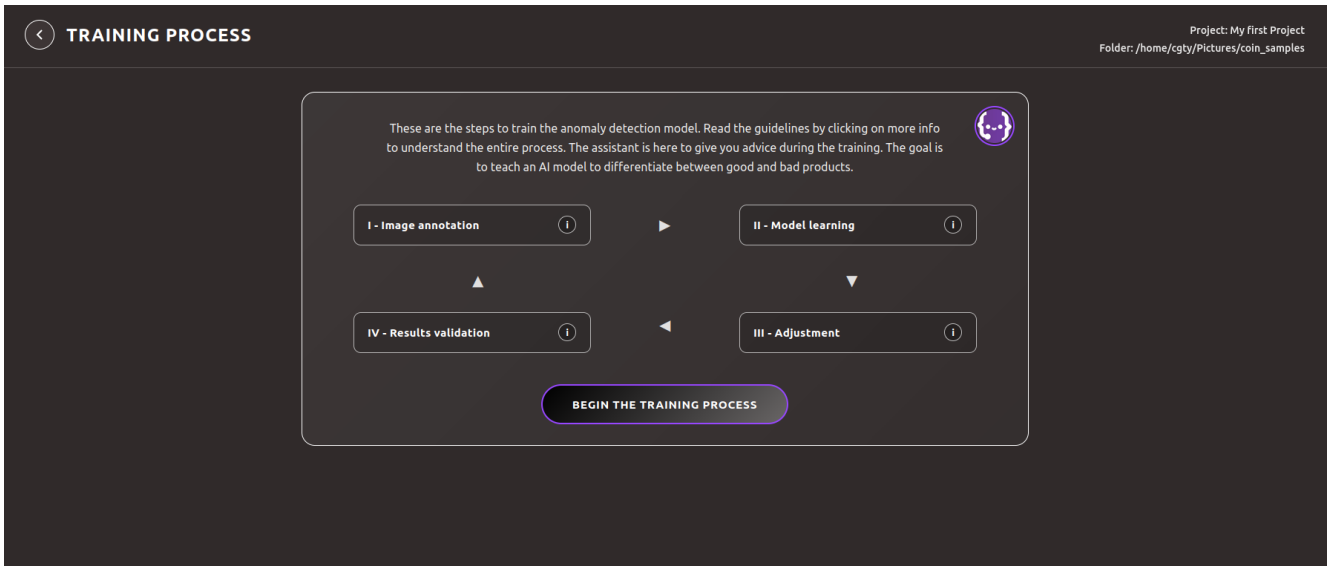


Figure 52: Information Page

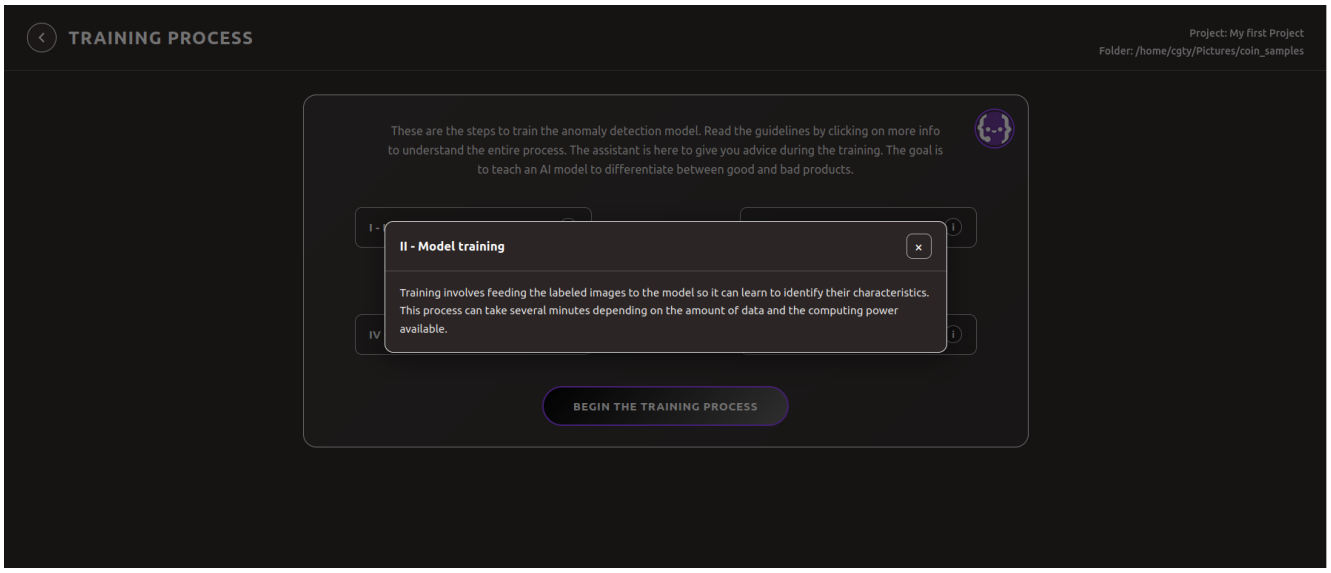


Figure 53: Information Page Details

Once the project is created, the user is presented with an information page (Figure 52: Information Page and Figure 53: Information Page Details) that explains the overall workflow of the application. This introductory screen provides a high-level overview of how the system operates and what the user can expect throughout the process.

By selecting the Info button at any time, users can access detailed explanations for each stage of the workflow, including I – Image Annotation, II – Model Learning, III – Adjustment, and IV – Results Validation. This ensures that users can familiarize themselves with the purpose and functionality of every step, helping them navigate the application with confidence and clarity.

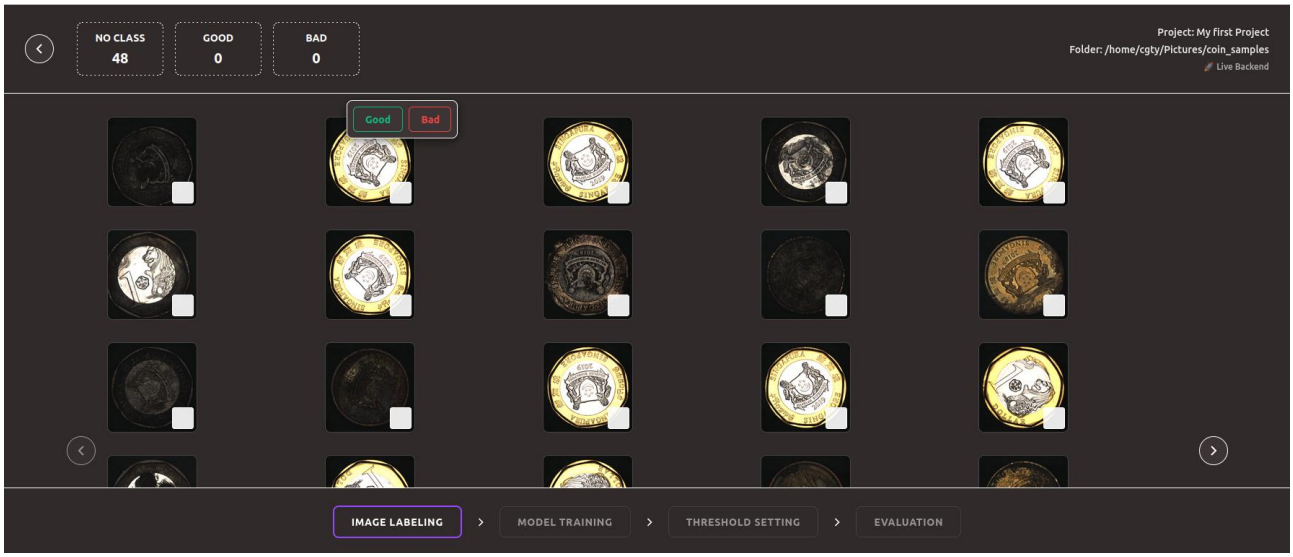


Figure 54: Labelling Page

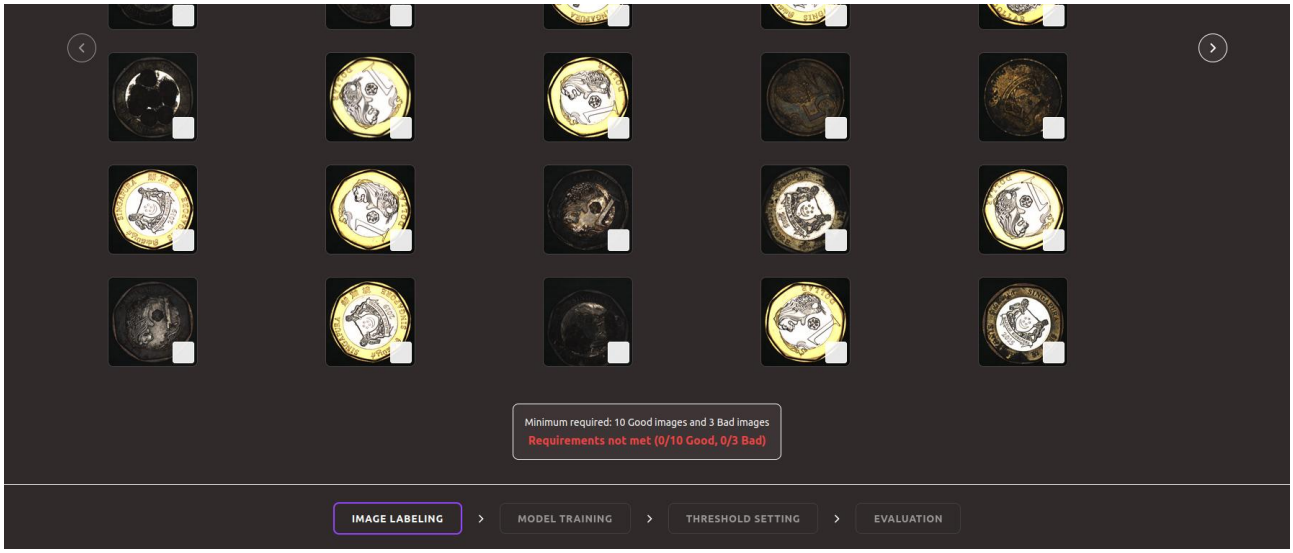


Figure 55: Labelling Page – Not filled

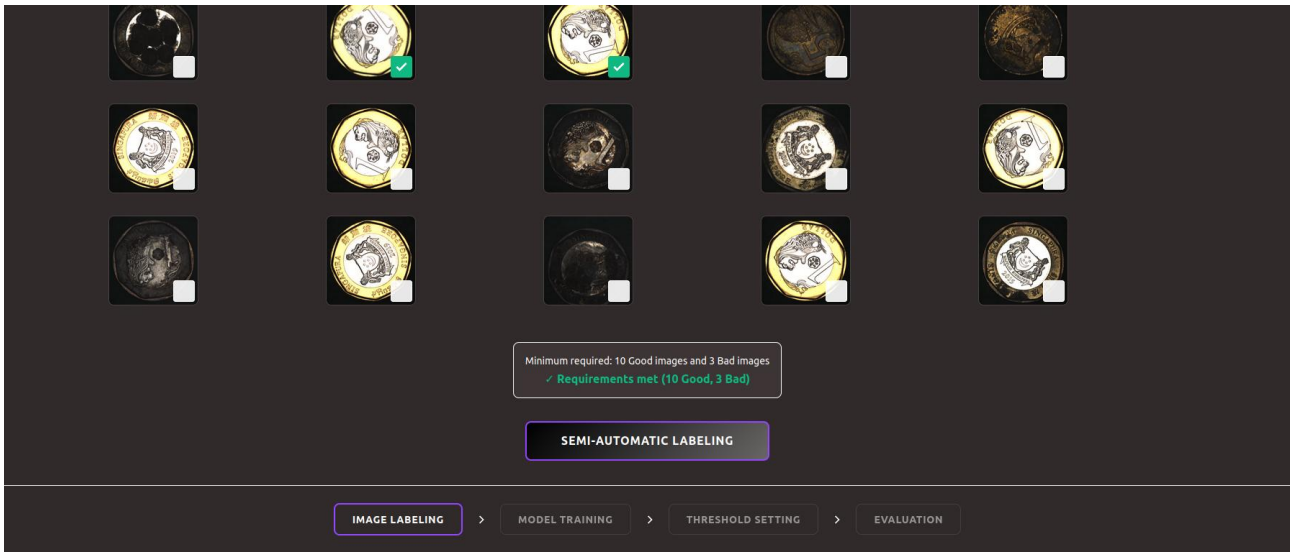


Figure 56: Labelling Page - Requirements met

Once users navigate to the Labelling Page (Figure 55), they are presented with all the images contained in the directory they selected during project setup. From here, users begin the manual annotation phase by assigning labels to the images. Each image includes a checkbox area in the bottom-right corner where users can mark it as “Good” or “Bad.” To ensure sufficient data quality for the semi-automatic labelling process, users must label at least 10 good images and 3 bad images (Figure 56). These minimum requirements help the system learn meaningful distinctions between image classes.

After the required number of labelled images is reached, the interface enables the option to start the semi-automatic labelling process (Figure 57). During this step, the application uses the manually provided labels as reference examples to automatically classify the remaining images in the dataset. This accelerates the annotation workflow and reduces the amount of manual effort needed, while still allowing the user to review or adjust the results later in the process.

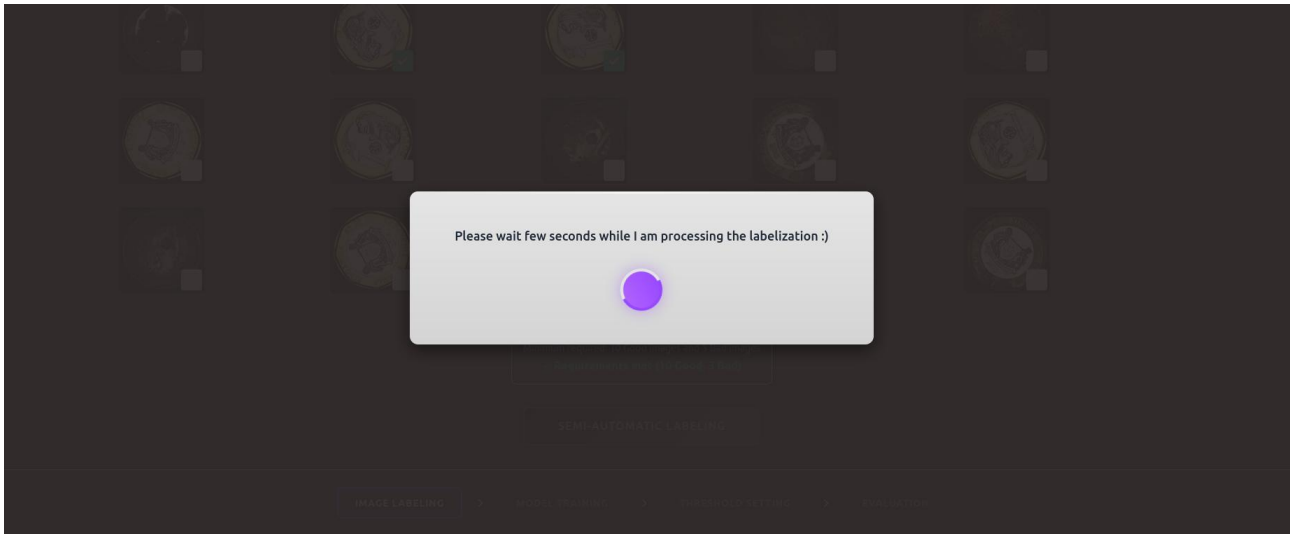


Figure 57: Auto Labelling Processing Screen

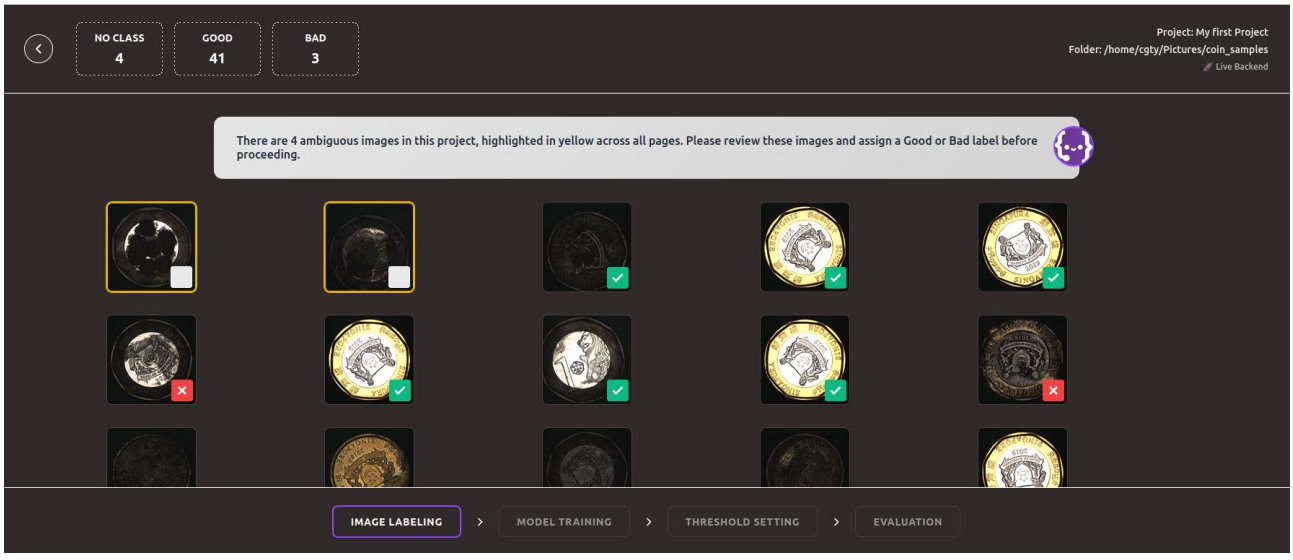


Figure 58: Ambigous Pictures State

Once the semi-automatic labelling process is complete, some images may remain ambiguous, meaning the model was not able to confidently classify them during the automated step that can be seen in Figure 58. These images are collected and presented to the user for further review. At this

stage, the user must manually assign the correct labels to each ambiguous image to ensure the dataset is fully and accurately annotated.

After the user completes this final round of manual labelling as in Figure 59, they can confirm the annotations and proceed to train the model. This additional verification step helps improve the quality of the training data, ensuring the model learns from clearly defined examples and ultimately produces more reliable results.

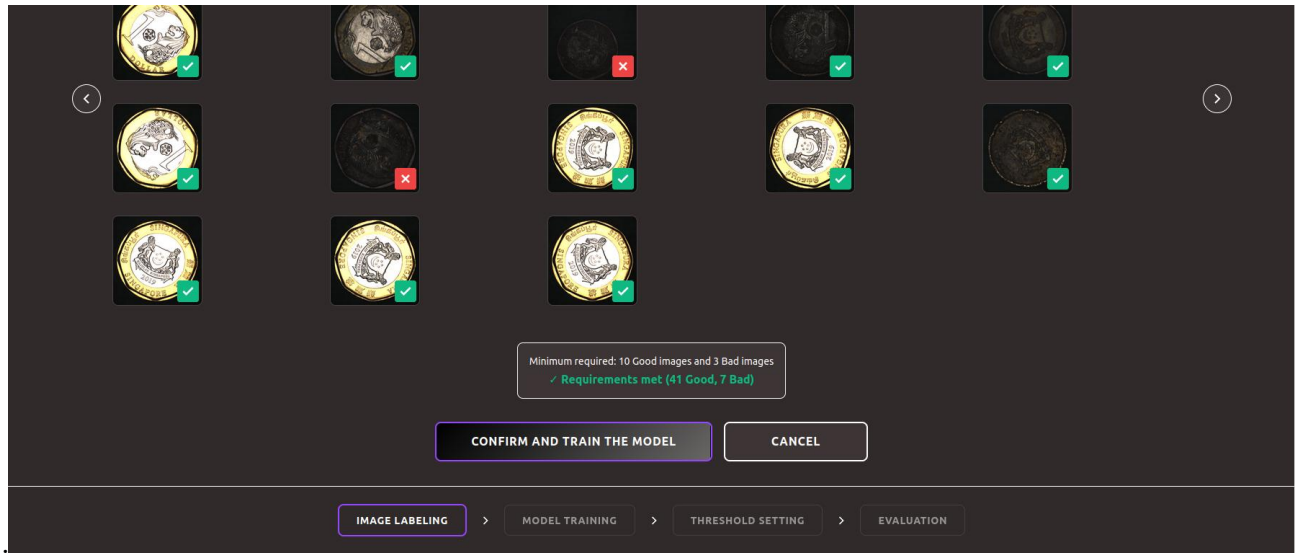


Figure 59: Auto labelling Process Completed

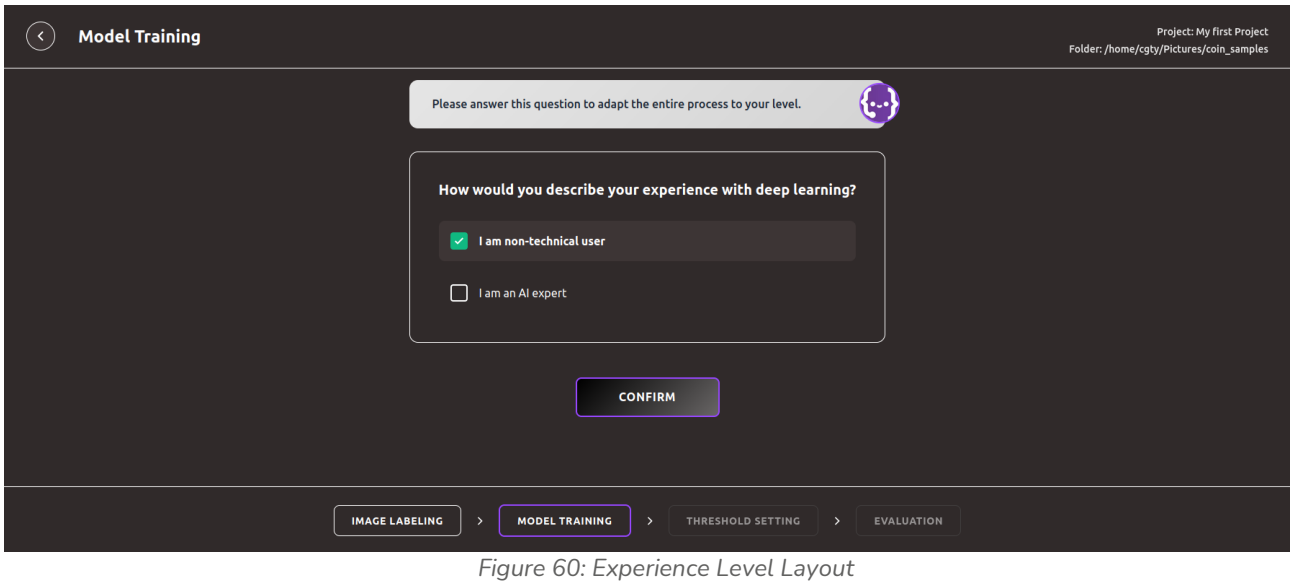


Figure 60: Experience Level Layout

After confirming the labels, users choose their AI experience level to configure the training workflow as depicted in Figure 60:

- I am a non-technical user — The application applies a set of pre-selected (recommended) parameters for model training. These defaults are optimized for general use and require no additional configuration. Users can proceed directly to training with minimal setup and users can only adjust the thresholds, that option can be seen in Figure 61.

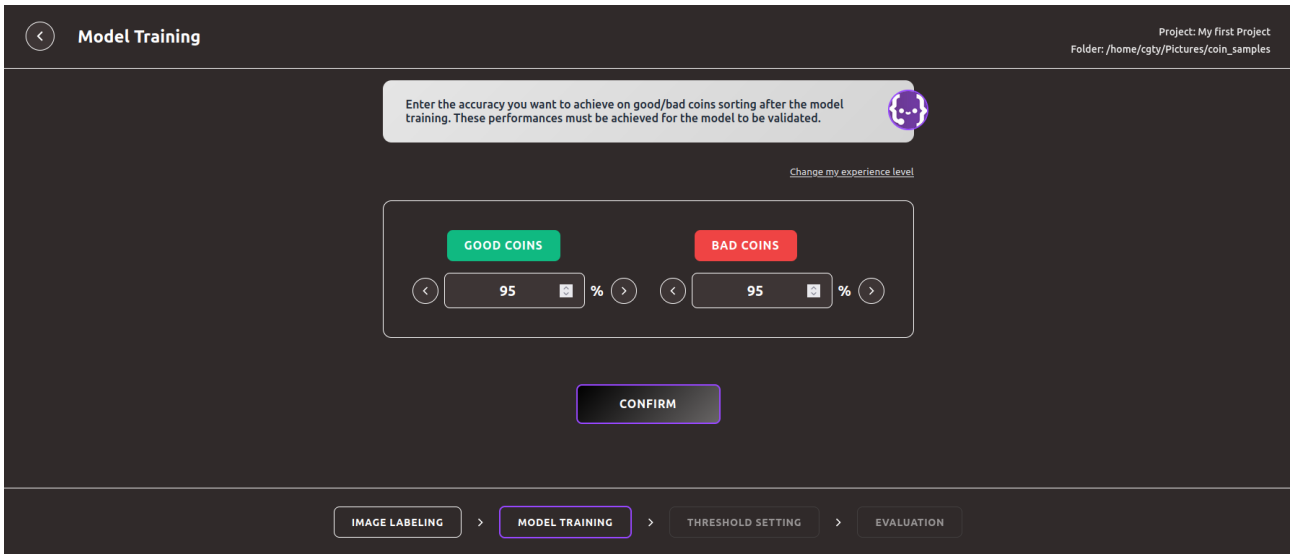


Figure 61: Non-Technical User Threshold Settings

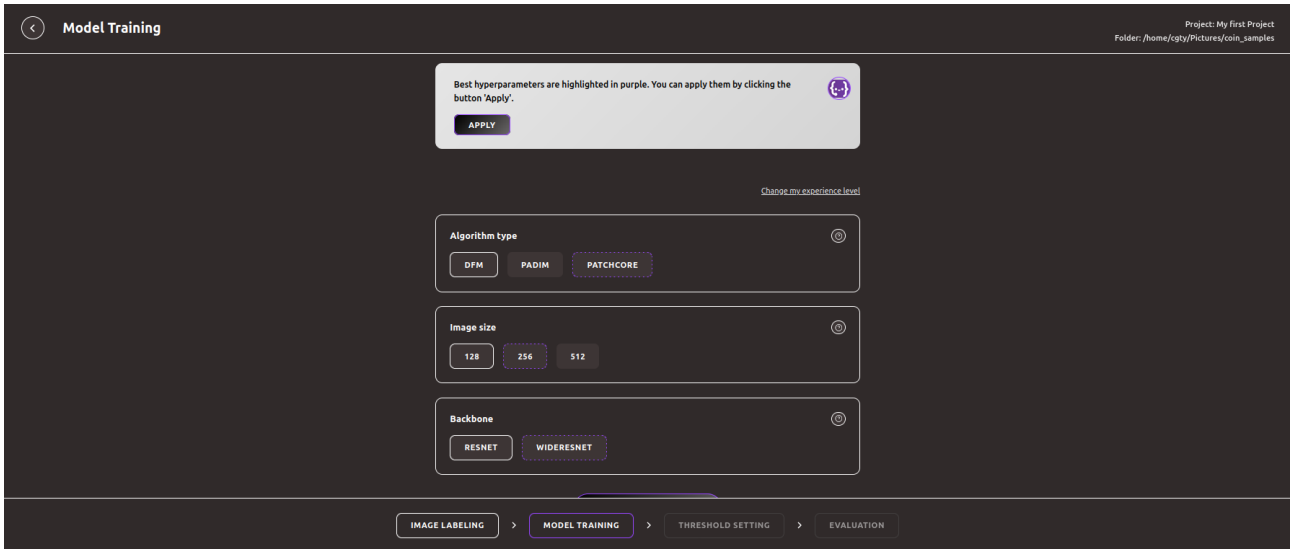


Figure 62: AI Expert Settings

- I am an AI expert — Advanced configuration becomes available, allowing users to select key training parameters such as Algorithm type, Image size, and Backbone. Each option includes a short description and expected trade-offs (e.g., accuracy vs. speed, memory footprint, and training time) to support informed decisions as can be seen in Figure 62.

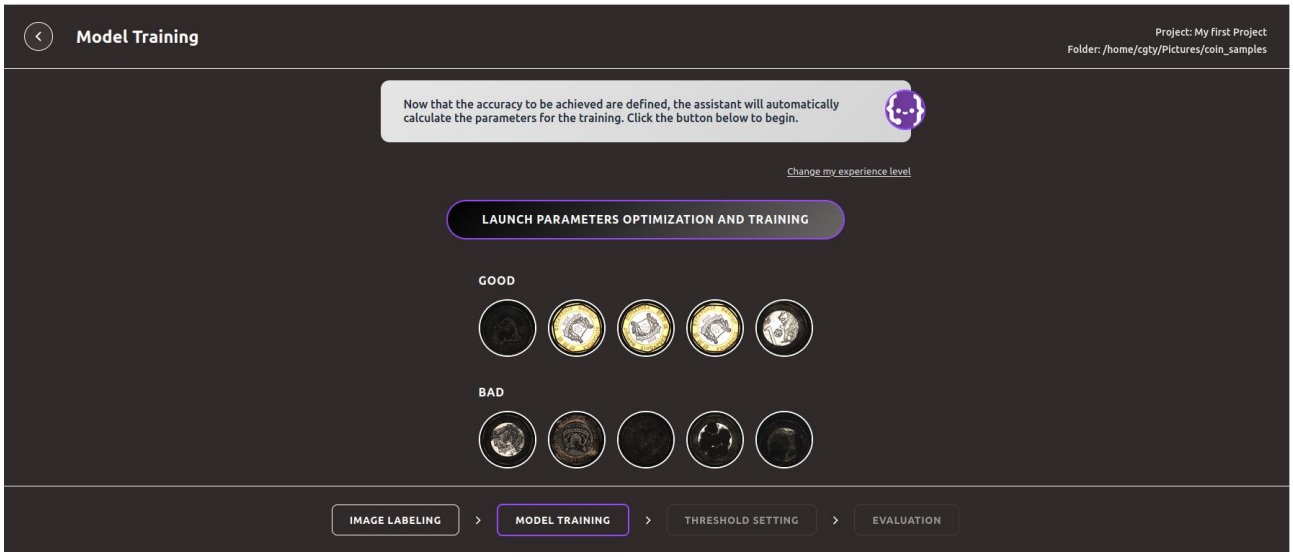


Figure 63: Launch Parameters Page

Once the desired experience level is selected and any optional parameters are reviewed, both user experience level ends up at Launch Parameter Page which allow users start the optimization process and start training can be seen in Figure 63.

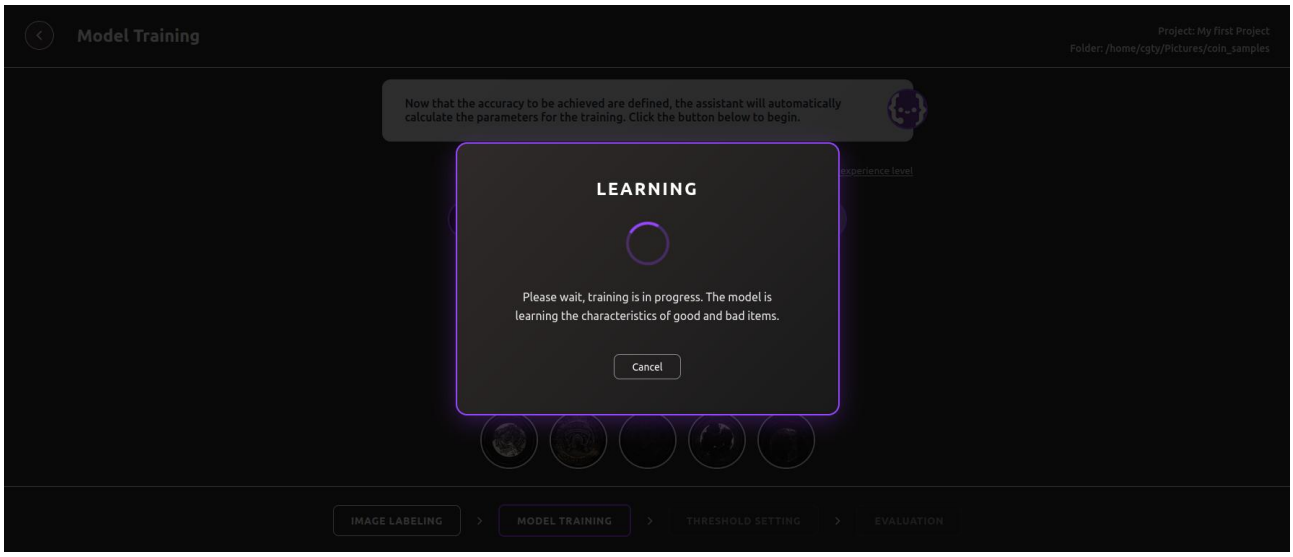


Figure 64: Start Training Process

This action kicks off the pipeline that can be visualized as Figure 64. First, dataset readiness is validated, then the chosen or recommended settings is applied, after optionally lightweight hyper-parameter search is run and finally model training is started. Progress indicators show the current stage (e.g., initializing, optimizing parameters, training, saving checkpoints) and surface any warnings or required actions (for example, insufficient labelled samples or incompatible image sizes).

For expert users, the Advanced Settings panel also provides guidance and guardrails:

- Algorithm type: brief notes on suitability for binary vs. multi-class problems and expected convergence behaviour.
- Image size: recommended ranges based on GPU/CPU memory; automatic resizing behaviour for out-of-range inputs.

- Backbone: a short capability summary (feature richness, inference speed), with hints on when to choose lightweight vs. deeper architectures.

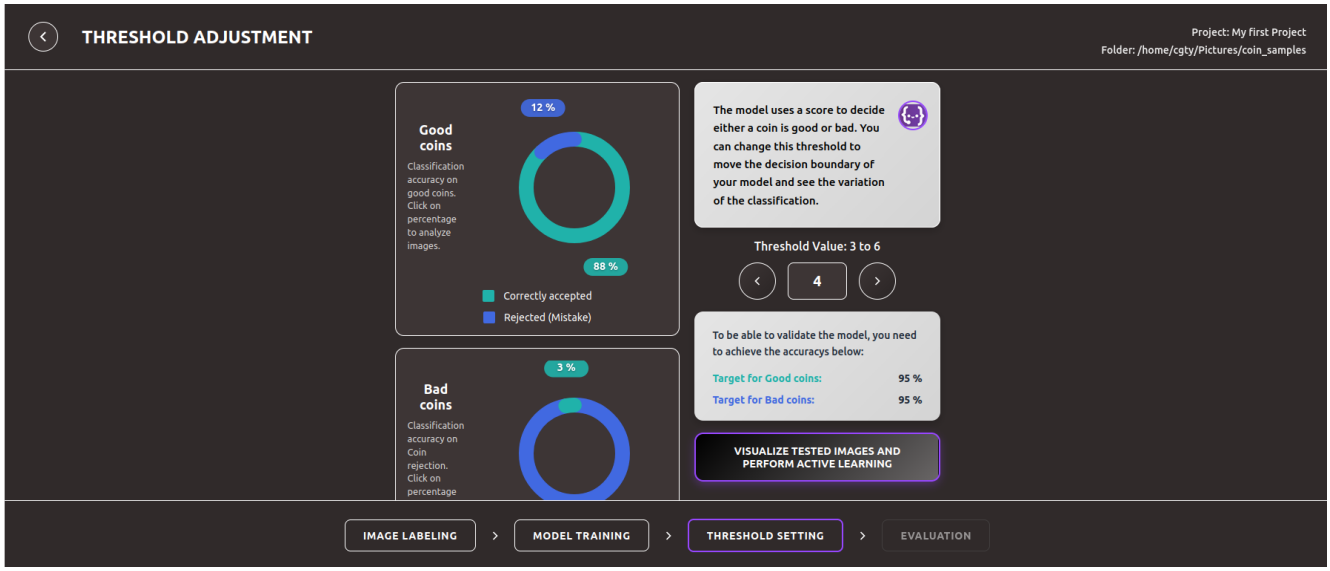


Figure 65: Threshold Settings

After the training process is completed, the system automatically generates a new. onnx model file and saves it in the project directory previously selected during project creation. Each exported model is stored as a separate version, ensuring traceability and reproducibility across training cycles. This follows the project's design principle of creating immutable model artifacts rather than overwriting existing ones.

Once the model is successfully generated, users are directed to the threshold adjustment page highlighted as Figure 65. Although this view is currently a mock interface - because AI model support for threshold modification is not yet implemented - it is designed to eventually allow operators to fine-tune the decision threshold and immediately observe updated metrics, following the architecture's goal of operator-friendly threshold tuning.

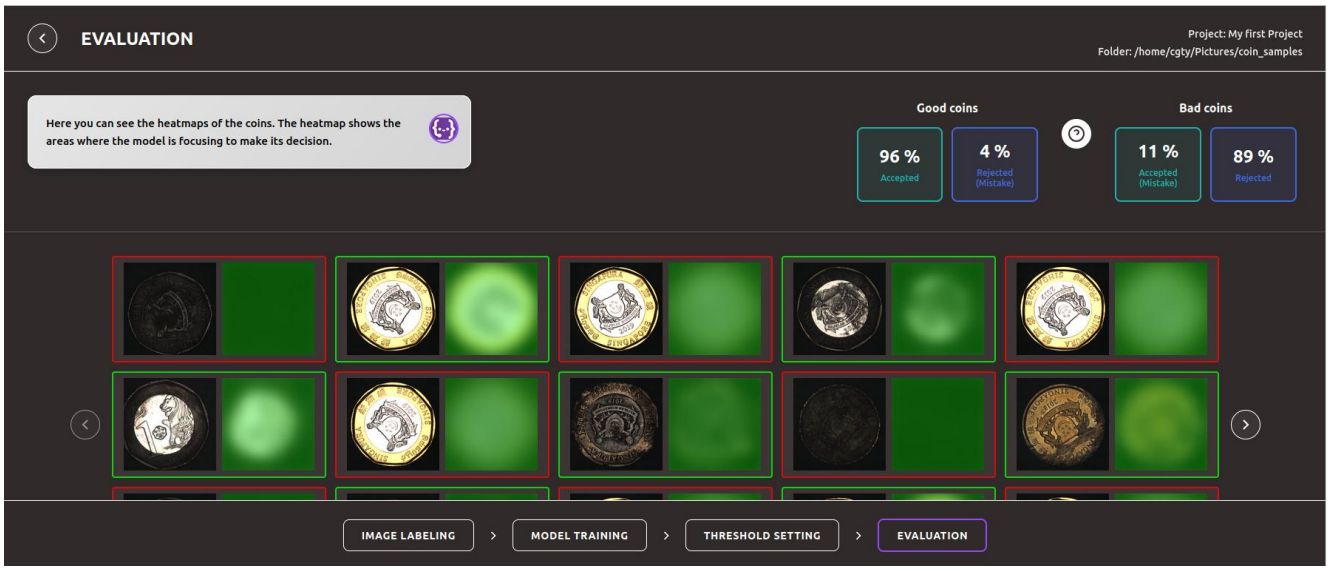


Figure 66: Heatmap Page

After reviewing the threshold settings, users proceed by selecting "Visualize tested images and perform active learning." This step displays heatmaps that illustrate how the model interprets various

image regions. As with the threshold page, the heatmap visualization (Figure 66) is presently a mock placeholder until backend integration for explainability is complete. In future iterations, this page will also serve as an entry point for active-learning workflows, where uncertain or low-confidence images are highlighted for user review.

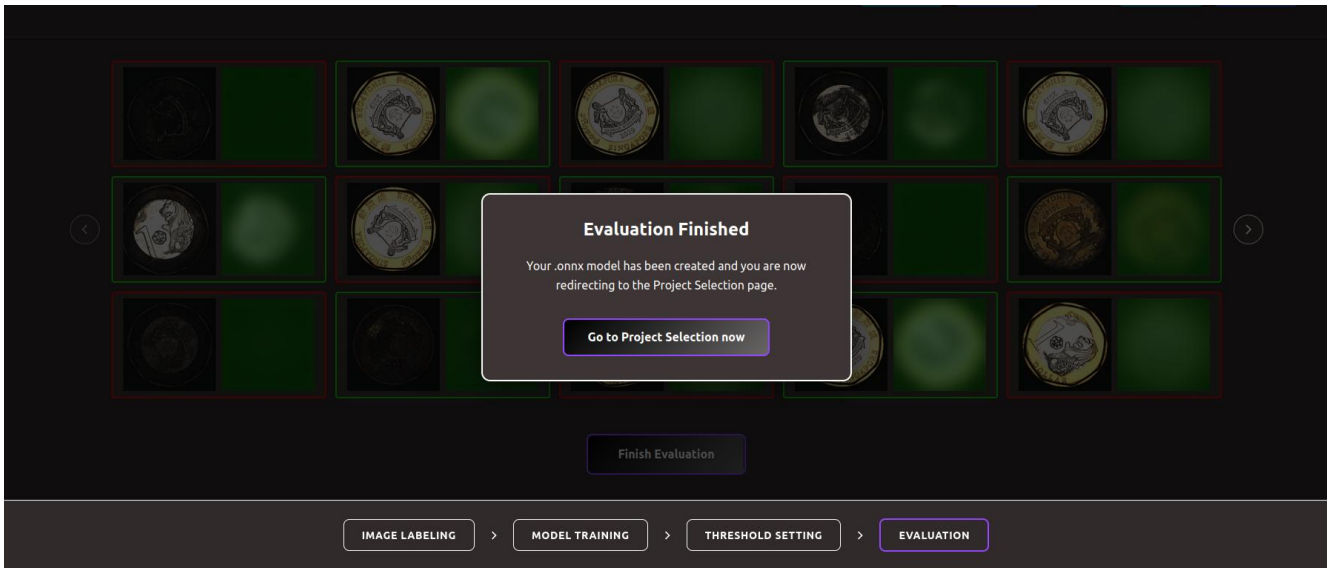


Figure 67: Finish Evaluation Popup

To complete the workflow, users click “Finish evaluation.” This final step closes the project’s evaluation phase, signalling that training, inspection, and preliminary validation are finished that can be seen Figure 67. At this point, all model outputs, decisions, and accompanying metadata - such as model version, thresholds, and evaluation logs - are recorded in the project directory, preparing the project for export, deployment, or further retraining if necessary.

2.3.5 Evaluation Journey

Validation for MVP2 will be conducted in a controlled laboratory environment using internal Proditec employees (acting as proxy users, including both expert customer engineers and novices). The evaluation will utilize well-structured open-source datasets, such as the MVTec Anomaly Detection dataset. The assessment will focus on the speed of the anomaly detection setup, the ability to reach a predefined accuracy target (e.g., >= 95%), usability, and the operators' comprehension of the AI's explanations and heatmaps, measured via the AIA index and semi-structured interview.

2.3.6 Design of the MVP

The MVP2 for Proditec (PDT) transitions the initial UI mock-ups and concepts into a fully functional, standalone application. Operating within a decoupled microservice environment, it strictly adheres to the project's Common Blueprint but is uniquely engineered for a strict "local-first" execution on an industrial Windows workstation to meet stringent latency and data privacy requirements.

2.3.6.1 Architecture

The system implements the multi-tier microservice blueprint defined in Deliverable D5.2 for localized execution without external cloud connectivity. A ReactJS frontend communicates asynchronously via REST APIs with a Python FastAPI backend, the Orchestrator. The AI-driven components, collectively

referred to as “Solvers” (such as the Anomaly Detection model and the Active Learning Loop), are separated from the Orchestrator to avoid monolithic background tasks.

Fujitsu implemented this separation of concerns by integrating the AI-Components into a single dedicated microservice. This service runs in an isolated virtual environment on the local host and exposes distinct RESTful API endpoints for specific tasks, namely an Autolabel API and a Training API. This API-driven isolation ensures that inference and training workloads during image processing do not block the Orchestrator's event loop, maintaining UI responsiveness for the machine operator.

2.3.6.2 Data Integration & Image Processing Model

The application enforces a strict separation between the transactional state of the user interface and the computational image data. The Orchestrator functions as the central hub for managing the application state, maintaining the configuration of inspection recipes, and executing the complete pre-processing and structuring of the file system.

To ensure that the Labelling Process always follows a standardized schema, the Orchestrator programmatically manages the underlying directory structures. A key technical feature is the automated creation of dedicated “Staging” folders. This staging mechanism isolates the operational workflow from the raw input, allowing the semi-supervised labelling and model training processes to be executed iteratively or multiple times on the exact same baseline dataset without risking data corruption.

The isolated Solver microservice acts solely as the administrator of the image processing model. It ingests and processes the prepared image datasets directly from these localized staging directories. By decoupling the visualization state and folder management from the heavy computations, the architecture ensures a highly stable, reproducible, and file-based environment for dataset preparation and model training.

2.3.6.3 Anomaly Detection Service & Computed Explanations

The WP3 anomaly detection algorithms are wrapped within the dedicated AI FastAPI microservice. The internal codebase remains a 1:1 copy of the original research code to support straightforward integration via the CI/CD pipeline. The service utilizes the PatchCore algorithm to execute semi-supervised auto-labelling on the staged image datasets.

2.3.6.4 Auditability & Safety

System auditability is maintained via a local SQLite database implementing an append-only Decision Log, secured by cryptographic SQL hash chains. The database records the operator interactions required for traceability, including manual label corrections, threshold parameter adjustments, and iteration metadata for each trained model. Data security is enforced by the local-first architecture, ensuring that proprietary image datasets are not transmitted over external or cloud networks.

2.3.7 Deviations and Corrective Actions

At the current stage of development, the explainability capabilities provided by the underlying AI-Solver are strictly limited. Currently, the system's explainability is restricted to identifying and isolating

images that the semi-supervised labelling process categorizes as “ambiguous”. Advanced explainability artifacts, such as visual heatmaps (anomaly maps) or exact percentage-based confidence scores, are presently not returned by the model. Until these features are implemented and exposed by the core Solver in future iterations, the Orchestrator and Frontend rely exclusively on the ambiguous-flagging mechanism to prompt the operator for manual review.

To illustrate the future target status and prepare for the application, only mock data and mock heatmaps are currently displayed.

Regarding software deployment, the Azure DevOps CI-Pipeline automatically handles code compilation, dependency resolution, and artifact packaging. However, the automated Continuous Delivery (CD) stage is currently suspended. It must be noted that at the time of writing, final network access credentials, VPN policies, and hardware specifications for the designated industrial Windows workstations were not accessible. Consequently, the exact automated mechanism for the final deployment of release artifacts remains speculative.

2.3.8 Responsibilities

The technical implementation is distributed as follows:

- **PDT (Proditec):** Lead on requirements definition, overall UI/UX software design, provision of the standalone hardware, and supplying the image datasets.
- **JU & VUB (AI):** Development of the PatchCore-based semi-supervised anomaly detection module, the active learning research, and the Explainability layer (heatmap generation and simplified terminology).
- **FUJ (Fujitsu):** Development of the ReactJS Frontend, the FastAPI Orchestrator, integration of the AI-Solver REST APIs, and implementation of the local SQLite database for the immutable audit logs.
- **CATIE & VUB-SMIT:** Execution of laboratory tests, definition of validation scenarios, and measurement of KPIs using the AIA index.

2.4 Datacation

2.4.1 Motivation

The Datacation use-case is driven by the need to optimize warehouse operations in increasingly complex and dynamic logistics environments. Current methods for configuring warehouse zones and allocating storage can be rigid and may not adapt efficiently to future demands. The core motivation is to develop an advanced AI system that assists logistics engineers in optimizing the allocation of location types within a flexible digital warehouse configuration, ensuring compliance with strict safety regulations regarding hazardous materials while meeting operational efficiency objectives.

2.4.2 Goals

The primary goal for MVP2 is to deliver a functional, interactive prototype web application. The technical focus is on implementing a global optimization solver (Integer Programming) that treats both product allocation and warehouse configuration as a unified problem. A key objective is to establish a collaborative human-AI interaction model where the user can actively influence the optimization algorithm by modifying a subset of constraints and weights (e.g., using binary switches) and receive computed explanations justifying the proposed warehouse configurations.

2.4.3 Limitations

Integration with a live, production Warehouse Management System (WMS) is strictly out of scope for MVP2; instead, the system interfaces with a dedicated WMS Digital Twin. Furthermore, to minimize GDPR risks and protect sensitive business data, the MVP2 prototype relies on simulated operational data rather than real, live warehouse datasets. Fully automated, real-time dynamic re-routing of pallets based on live continuous forecasting is deferred to future iterations.

2.4.4 User Journey

The logistics engineer logs into the web application and reviews the current warehouse state retrieved from the WMS Digital Twin. When incoming orders cannot be efficiently stored due to conflicting safety constraints or space limitations, the AI-Assistant presents alternative optimized configuration plans. The user can review the trade-offs of these plans (e.g., “Plan A improves capacity but requires moving hazardous goods”). Through the user interface, the user can interactively enable, disable, or relax certain specific constraints to steer the AI-Solver. Once a satisfactory configuration is found, the user explicitly accepts the plan, and the system logs the decision.

When starting in the initial state the warehouse is set up of multiple articles. Each one being stored in one of total 50 cells. One article belongs to one order, and an order can consist of grouping multiple articles. To facilitate control of incoming orders and traceability of order handling the application provides two control buttons (Next Order, Reset Simulation). Upon ‘Next Order’ new orders are accepted and the according articles of each new order need to be placed in the warehouse. If, however, the warehouse current setup does not provide sufficient storage for all articles, it is possible to generate a solution of a new storage layout (Solve Problems), analyse the solution and apply (Apply Solution) or reject it (Reject Solution).

In addition to the possibility of creating a new storage solution it is also possible to interact with an AI-Assistant to get a better understanding of the current warehouse state and existing orders. Example of useful questions about the warehouse state are:

- Summarize the clients and number of articles of all orders?
- How many orders will leave tomorrow?
- What is the current state of the warehouse?
- What are the current solver parameters?
- What locations are orders of client X stored at?

2.4.5 Evaluation Journey

Testing will take place in a controlled laboratory setting involving actual logistics engineers and warehouse systems experts. The evaluation will focus on measuring the time savings in warehouse planning, the acceptance rate of the AI's proposed configurations, the understandability of the computed explanations provided by the system, and the overall trust in the collaborative assistant, measured through the AIA index.

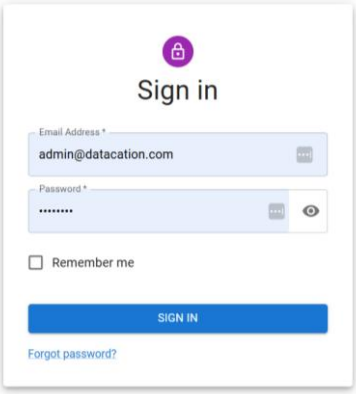


Figure 68: Datacation Login Form

The journey begins with a Login Page, which serves as the entry point to the Datacation Warehouse Allocation Assistant. For the purpose of the MVP, no real user management or authentication mechanism is implemented; instead, the system uses predefined mock credentials to simulate access control and user roles as shown in Figure 68. This approach allows testers and stakeholders to quickly access the application and explore its functionality without the overhead of a fully integrated identity management system. Once the user enters the provided credentials, they are granted access to the main application interface, where they can proceed with creating fit-check scenarios and interacting with the allocation assistant. This simplified login mechanism ensures a smooth onboarding experience while still reflecting the structure of a production-ready system.

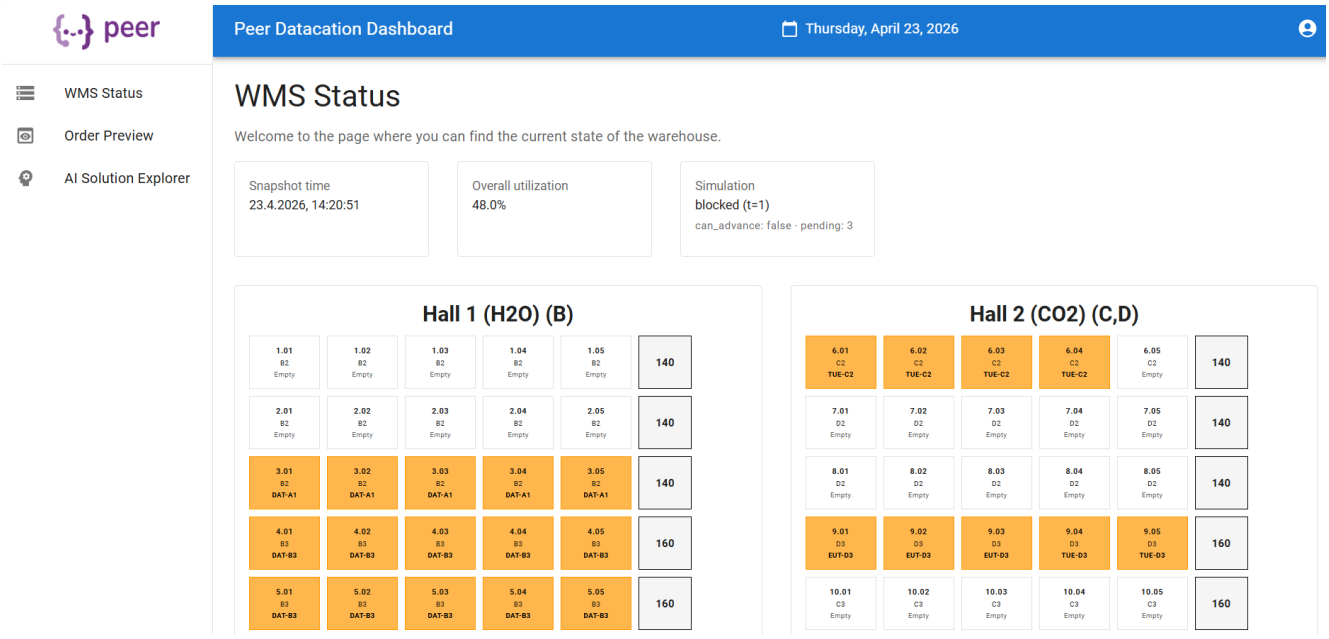


Figure 69: WMS Status Page

After successful login, the user is redirected to the WMS Status Page as represented in Figure 69, which provides a real-time overview of the warehouse environment. On this page, the user can monitor the current layout of the warehouse halls, including a visual representation of storage cells and occupancy. The interface displays key operational indicators such as the snapshot timestamp (indicating when the data was last updated), the overall warehouse utilization rate, and the current simulation status (e.g., active, blocked). Additionally, the page lists current and historical orders which shown in Figure 70, where today's active orders are highlighted in yellow, while previously processed orders are shown in green, enabling quick visual differentiation. This dashboard serves as the main situational awareness screen, allowing users to understand the current state of the warehouse before initiating a fit-check or allocation scenario.

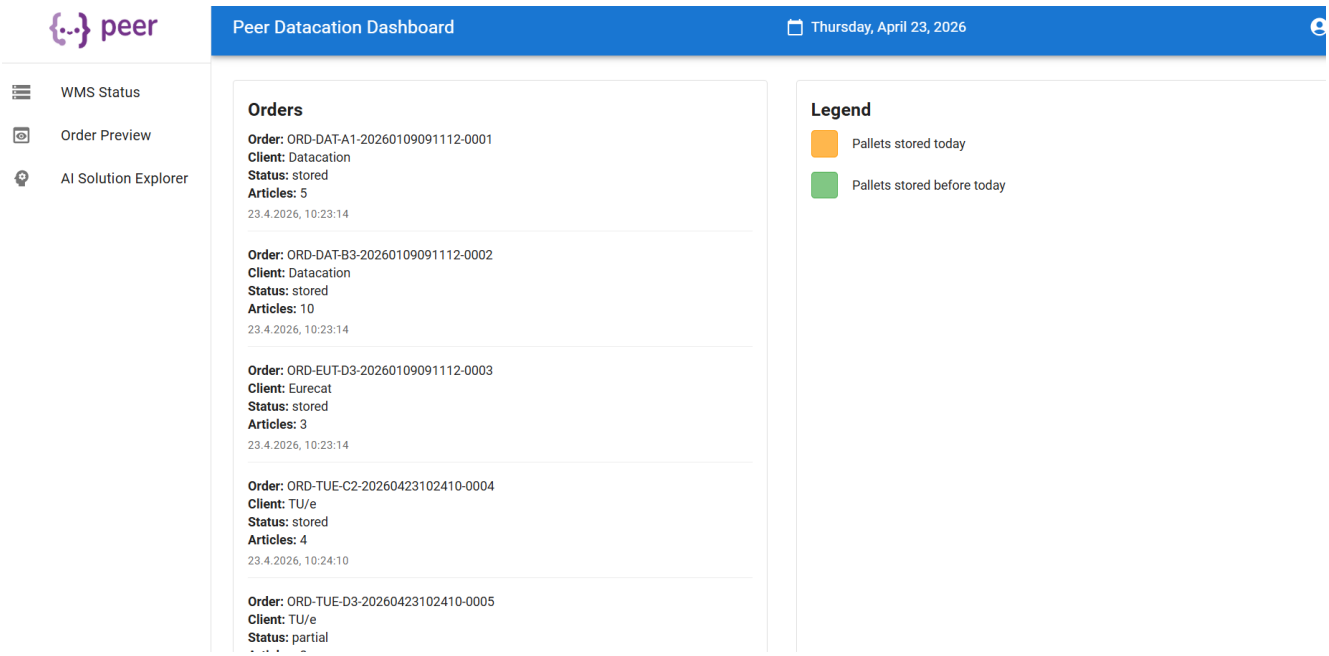


Figure 70: WMS Orders List

After reviewing the WMS status, the user navigates to the Order Preview Page, which is shown in Figure 71, provides a detailed overview of all warehouse orders and their current processing state. This page allows users to view stored orders, monitor their attributes (such as client, number of articles, arrival and departure times, and timestamps), and quickly evaluate their status. Orders that are successfully processed are displayed in a structured table, while problematic or errored orders are highlighted separately (in a red-bordered section), enabling users to immediately identify cases that require attention or further analysis.

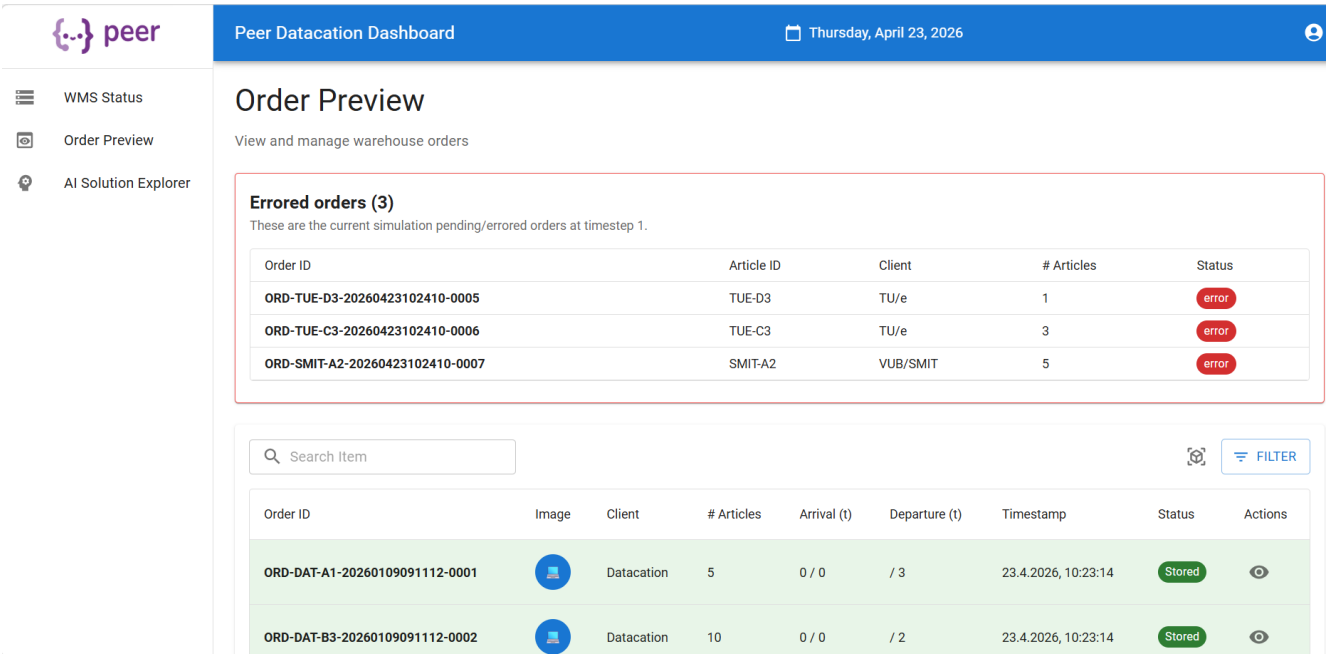


Figure 71: Order Preview Page

The interface also includes filtering and search capabilities, allowing users to narrow down orders based on specific criteria. For each order, users can access more detailed information by clicking the "eye" icon, which opens a detailed view as in Figure 72 of the order, including its allocation status and underlying issues. This page serves as a bridge between high-level warehouse monitoring and detailed problem-solving, guiding users toward selecting relevant orders for further analysis in the AI Solution Explorer.

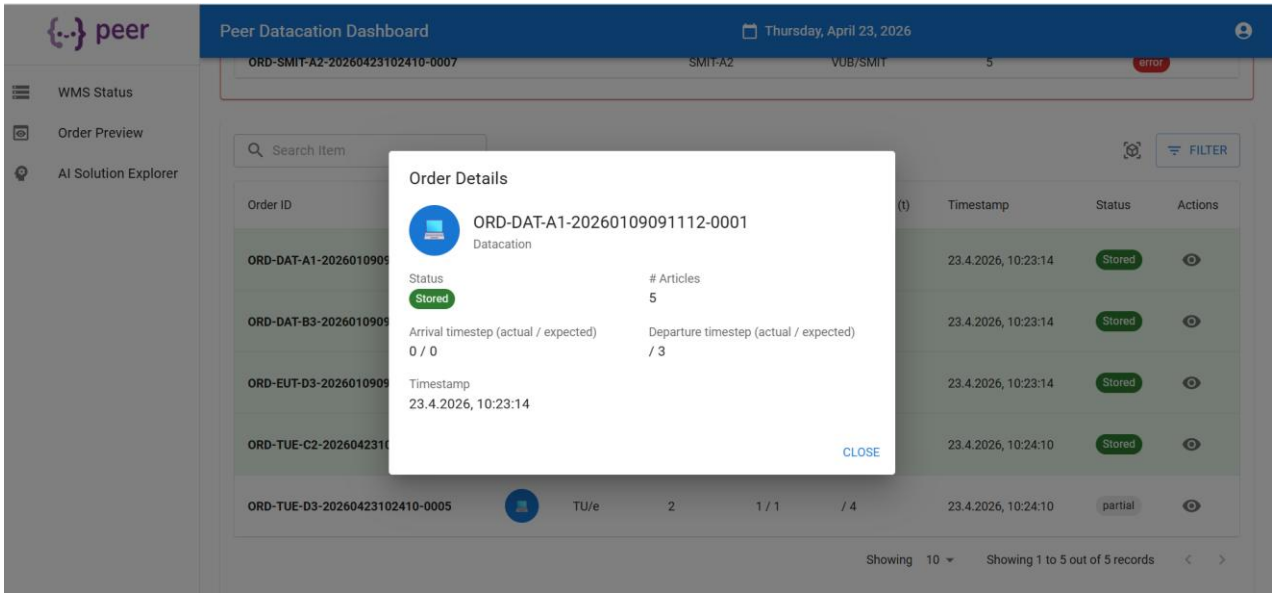


Figure 72: Order Details Popup

After identifying problematic orders in the Order Preview, the user can proceed to the AI Solution Explorer (Figure 73), where detailed analysis and resolution of errored orders take place. On this page, the user can clearly see the list of errored or blocked orders, along with their associated details and current status. By clicking the “Solve Problems” button, the user triggers the optimization process, where the solver computes a feasible warehouse reallocation strategy that resolves the detected issues.

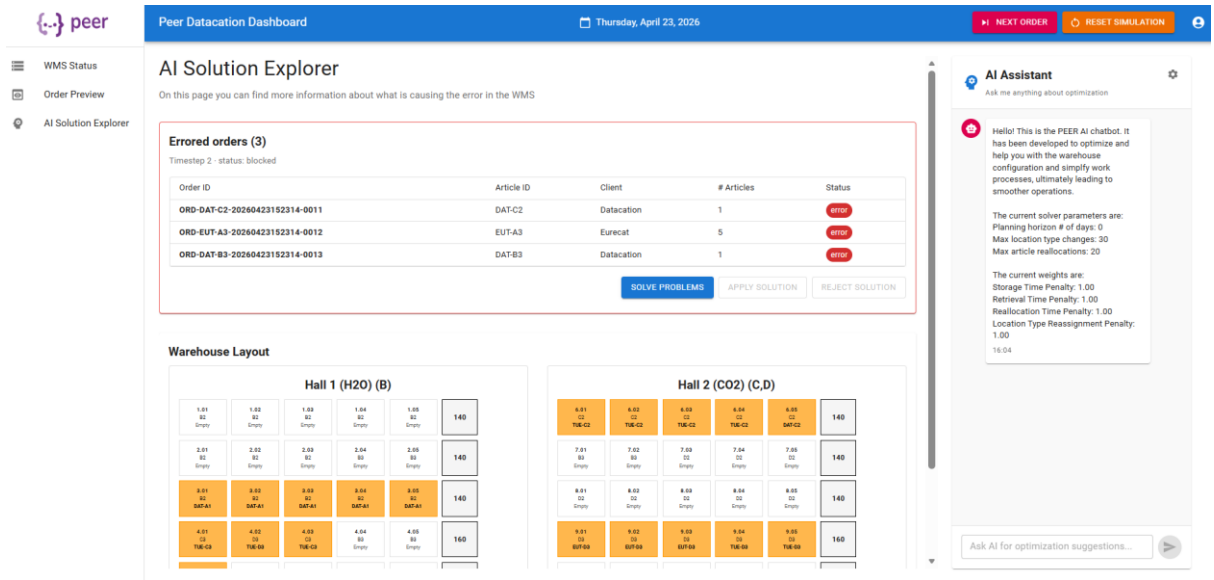


Figure 73: AI Solution Explorer Page

Once the computation is complete, the system presents a proposed solution as represented Figure 74, indicating how the warehouse layout can be adjusted to successfully store the previously errored orders. This includes suggested storage locations and changes in allocation. The user is able to see the changes of the location and the new locations of the errored articles by clicking the highlighted

cell of the layout. After the review, user is then given full control to either accept or reject the proposed solution.

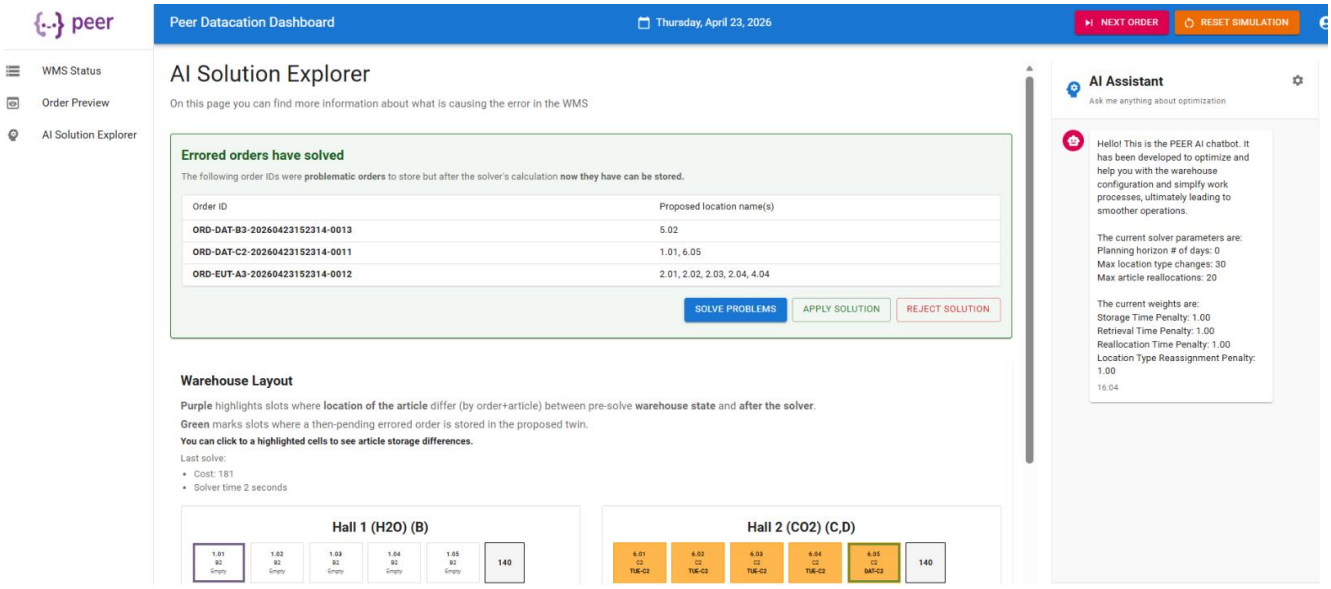


Figure 74: AI-Solver's Solution Proposal State

If the solution is accepted, a detailed popup view is displayed (Figure 75), summarizing all layout changes, such as article reallocations, location type changes, and newly assigned storage positions. This ensures full transparency and traceability of the solver's actions.

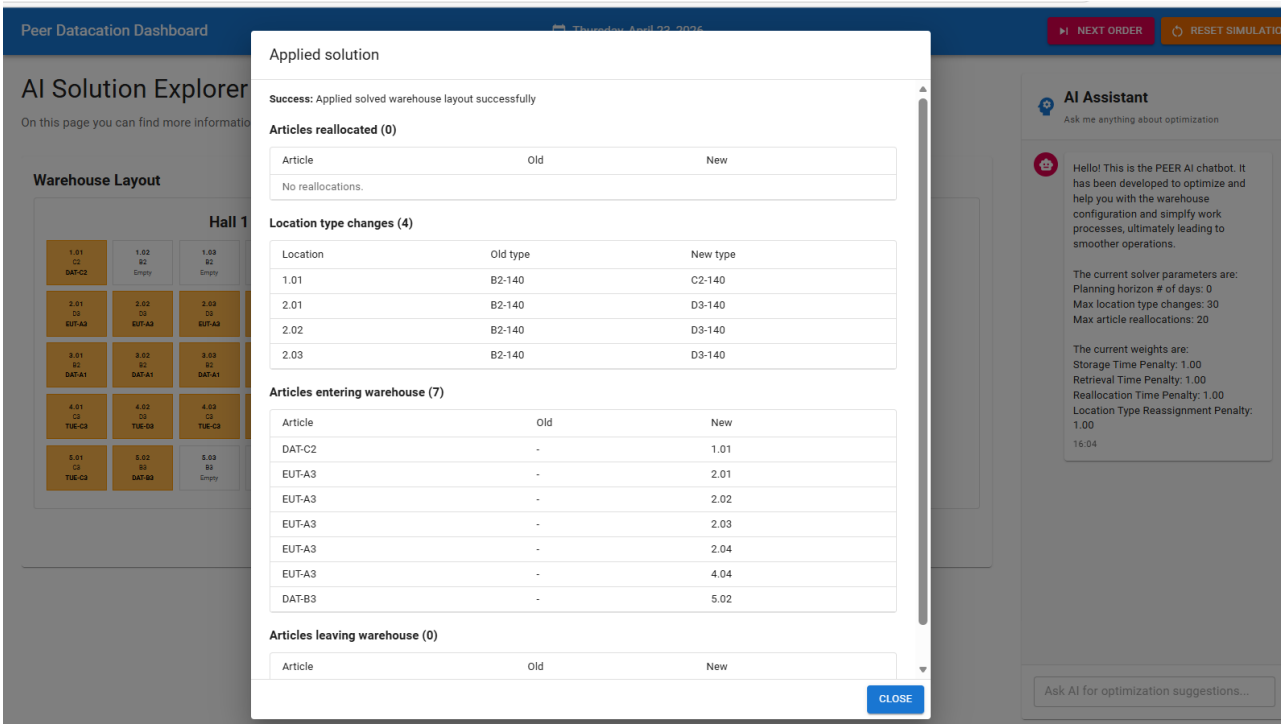


Figure 75: Applied Solution Popup

In addition to executing the optimization, the user is also able to configure solver parameters directly from the interface by clicking the settings (cog) button at the header of the AI-Assistant Layout as shown in Figure 76 . This opens a configuration panel displaying the current solver constraints and weights, such as the maximum number of location type changes, allowed article reallocations, and

penalty factors (e.g., storage time, retrieval time, and reassignment penalties). The user can manually adjust these parameters and apply the changes, enabling fine-grained control over how the optimization behaves.

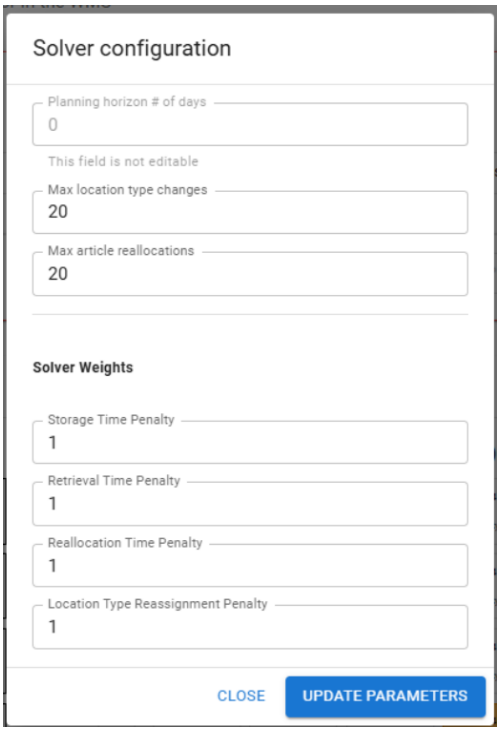


Figure 76: Solver Configuration Popup

Alternatively, the system provides an integrated AI-Assistant as shown in Figure 77, powered by a Large Language Model (LLM), which allows users to interact with the solver in a conversational manner. Through a free-text input field, users can request adjustments (e.g., “increase max location type changes to 30”, “get current parameters”, “reduce storage penalty”), and the assistant interprets these requests, updates the parameters accordingly, and provides a clear rationale for the changes. The assistant also summarizes the current configuration and explains how parameter adjustments impact the optimization outcome.

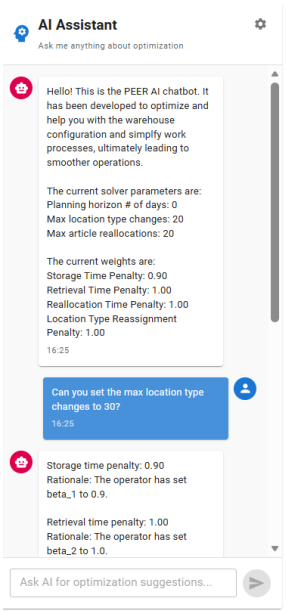


Figure 77: AI Assistant Chat Layout

This dual interaction model—manual configuration and conversational guidance—enables both expert users and non-technical operators to efficiently steer the optimization process, improving usability while maintaining transparency and control over the solver's behaviour.

Finally, the interface provides control actions at the top level are shown in Figure 78, allowing the user to proceed to the next order using the "Next Order" button when there is no pending/errored orders, to restart the entire simulation via the "Reset Simulation" option or logout the entire application by clicking the user button on top right with "Logout". This iterative workflow enables users to step through scenarios, validate solver decisions, and refine warehouse configurations in a controlled and interactive manner.

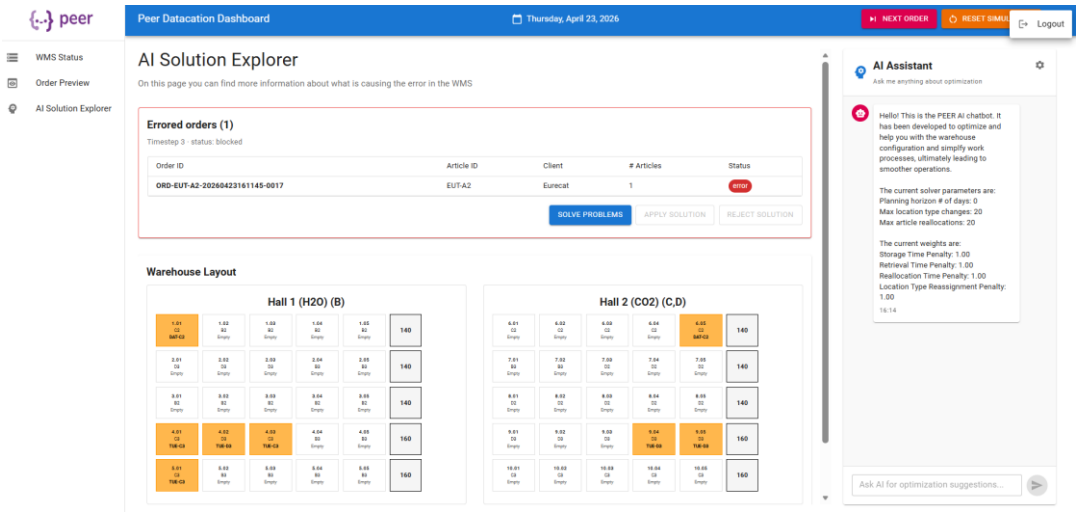


Figure 78: Additional Options of the Application

2.4.6 Design of the MVP

The Datacation MVP2 utilizes a centralized, computationally intensive client-server architecture to accommodate the heavy resource requirements of local Large Language Models (LLMs) and mathematical solvers.

2.4.6.1 Architecture

The system is built upon a highly distributed microservice architecture, strictly implementing the multi-tier blueprint defined in Deliverable D5.2. It features a ReactJS Frontend that communicates exclusively via REST APIs with a central Python-based FastAPI Backend (the Orchestrator). The Orchestrator is served asynchronously using Uvicorn service. It connects securely to all components via REST API to fetch current warehouse states and incoming orders on-demand, ensuring a single source of truth without permanently duplicating sensitive data.

2.4.6.2 Warehouse State Representation

Digital Twin provides state information about the warehouse, as well as current and future orders. It is furthermore possible to control the warehouse state as a simulation by iterating to the next day's order or resetting the warehouse to its initial state.

2.4.6.3 AI-Solution Provider

The AI-Solver is an intelligence layer consisting of two main components: an LLM-based Abstraction Module and an Optimization Core. The Abstraction Module acts as a human-AI interface and translates user inputs and constraints into formal mathematical parameters. The Optimization Core (Mixed Integer Programming) then calculates the optimal warehouse layout and product allocation based on these parameters. Each of these components runs independently and provides a REST interface for its main functionality. For the LLM, a model hosted via the Ollama inference engine is used, which is deployed and operated as a Docker container (qwen2.5:32b-instruct).

2.4.7 Responsibilities

The technical implementation is distributed across the consortium as follows:

- **DAT (Datacation):** Lead on use-case requirements, UI design, and provision of the WMS Digital-Twin (API) that delivers the warehouse state and incoming orders.
- **EUT (Eurecat):** Development of the LLM-based Abstraction Module to translate user interactions into solver constraints.
- **TU/e & INESC TEC:** Development of the mathematical Optimization Core (Solver) and generation of computed explanations for the proposed solutions.
- **FUJ (Fujitsu):** Development of the ReactJS Frontend and FastAPI Orchestrator, the overall API integration (all data flows and interactions are routed and controlled by the Orchestrator), and implementation of the immutable audit logging database.
- **CATIE & VUB-SMIT:** Support in defining validation scenarios, executing laboratory tests with end-users, and measuring KPIs using the AIA index.

3. Conclusions

This report details the second release of the Minimum Viable Product (MVP) for the PEER collaborative AI assistant, showcasing its instantiation across four key use-cases: Sonae, City of Amsterdam, Proditec, and Datacation.

The development and implementation of these MVPs are core activities within WP5, contributing to the overall goal of integrating and validating the AI assistant in real-world scenarios.

As such, the MVP presented here will need to be evaluated with the AIA index developed in the WP4 to assess its acceptability through user studies and gather insights on system usability and acceptance.

The AI depth of the MVP, including planning algorithms, XAI, user modelling, and the human-AI conversation loop, is described in detail in Deliverable 3.2, which defines the PEER building blocks and modular architecture.

PEER will focus on how to systematically put the user at the centre of the entire AI design, development, deployment, and evaluation pipeline, allowing for truly mixed human-AI initiatives on complex sequential decision-making problems. The central idea is to enable a two-way communication flow with enhanced feedback loops between users and AI, leading to improved human-AI collaboration, mutual learning and reasoning, and thus increased user trust and acceptance. As an interdisciplinary project between social sciences and artificial intelligence, PEER will facilitate novel ways of engagement by end-users with AI in the design phase; will create novel AI planning methods for sequential settings which support bidirectional conversation and collaboration between users and AI; will develop an AI acceptance index for the evaluation of AI systems from a human-centric perspective; and will conduct an integration and evaluation of these novel approaches in several real-world use cases.




peer