D3.4 SOCFAI PLATFORM

Secure Open Collaboration Framework powered by Artificial Intelligence

31 December 2025

Abstract

This deliverable includes the SOCFAI Platform specified in two domains, Airport and Port domain.

Contents

1. Introduction

The following structure clearly separates the document into the common technological foundation and the specialized AI/ML functions and data flows unique to the Airport and Port domains.

- **Objectives of the SOCFAI Project**

The **SOCFAI** (Secure Open Collaboration Framework powered by Artificial Intelligence) project is a comprehensive initiative dedicated to leveraging **big data and Artificial Intelligence (AI)** to resolve complex, systemic operational challenges inherent in multi-stakeholder environments, specifically in **airports and ports**. A core objective of the project is to provide a unified approach to manage the operations cycle, moving beyond individual process management to integrated, predictive operations. This involves deploying AI models to achieve optimization, such as optimizing container placement in ports and resource management in airports. Ultimately, the project aims to establish a framework that supports real-time common situational awareness and predictive operations management, utilizing advanced technologies for a holistic perspective. Furthermore, the project emphasizes maintaining data confidentiality and utilization control, which is crucial for secure collaboration in multi-party computing environments. In the summary of the objectives of the SOCFA Platform, the following features are considered.

- Integration and Standardization: It acts as the single point of entry for integrating disparate data sources (AODB, ADS-B, LiDAR, CFSMS, TMS) and standardizing this raw data into a format suitable for advanced analysis.
- AI Service Provision: It hosts and manages the shared infrastructure for AI/ML Services, ensuring the efficient training, deployment, and real-time inference of all Use Case-specific models (e.g., VoxelNeXt, CatBoost, PPO).
- Operationalization (MLOps): It implements the essential MLOps tools (MLflow, MinIO, NannyML) required for the sustainability, scalability, and maintainability of the predictive models in a live operational environment.
- Security and Trust: It embeds the necessary mechanisms—including a private blockchain in the Port domain—to guarantee the security, integrity, and non-repudiation of shared operational data, thereby fostering trust among collaborating partners.

- **Role and Scope of the SOCFAI Platform (Common Platform Perspective)**

The **SOCFAI Platform** functions as the **central technological backbone** of the entire project, providing a cohesive framework that supports the full lifecycle of AI services and enables collaborative decision-making across the Airport and Port domains. Its primary role is

multifaceted, encompassing integration, AI service provision, operationalization, security, and trust. The platform acts as the point of integration for diverse data sources, standardizing this raw data into a format suitable for advanced analysis. It hosts and manages the shared infrastructure for **AI/ML Services**, ensuring the efficient training, deployment, and real-time inference of all Use Case-specific models. The platform's scope extends to implementing essential MLOps tools, which are required for the **sustainability, scalability, and maintainability** of the predictive models within a live operational setting. Crucially, it embeds mechanisms, such as private blockchain technology in the Port domain, to guarantee the **integrity, security, and trustworthiness** of shared operational data, thereby fostering reliance among collaborating partners. By integrating these functions, the SOCFAI Platform transforms disparate AI/ML applications into an integrated, secure, and predictive operational system. In summary, the roles of the SOCFAI Platform are as follows.

- Integration and Standardization: It acts as the single point of entry for integrating disparate data sources (AODB, ADS-B, LiDAR, CFSMS, TMS) and standardizing this raw data into a format suitable for advanced analysis.
- AI Service Provision: It hosts and manages the shared infrastructure for AI/ML Services, ensuring the efficient training, deployment, and real-time inference of all Use Case-specific models (e.g., VoxelNeXt, CatBoost, PPO).
- Operationalization (MLOps): It implements the essential MLOps tools (MLflow, MinIO, NannyML) required for the sustainability, scalability, and maintainability of the predictive models in a live operational environment.
- Security and Trust: It embeds the necessary mechanisms—including a private blockchain in the Port domain—to guarantee the security, integrity, and non-repudiation of shared operational data, thereby fostering trust among collaborating partners.

And the platform's scope extends beyond simple data visualization to cover end-to-end data and operational management across two distinct sectors:

- Airport Domain: Focused on optimizing passenger flow, baggage handling, flight service times, and energy consumption.
- Port Domain: Focused on optimizing container yard management (stacking and rehandlings) and inter-CFS/TMS transportation logistics.

By providing this common, robust, and secure infrastructure, the SOCFAI Platform transforms individual AI/ML solutions into an integrated, collaborative, and predictive operational system.

2. SOCFAI Platform Common Architecture & Core Technology

2.1 High-Level System Architecture and Components (General Architecture & Components)

The SOCFAI Platform is the core foundation of the Secure Open Collaboration Framework powered by Artificial Intelligence, designed to support the various Use Case applications in airport and port operations. The platform provides an integrated environment that covers the entire lifecycle of intelligent services, from data collection and AI model operation to the final end-user applications.

2.1.1. Definition of Platform Layers

The SOCFAI Platform architecture is broadly divided into four layers based on their function and role.

Layer	Main Role	Key Technology/Components
1. Data Collection & Integration	Real-time collection and standardization of data from various external and internal systems for integration into the platform.	Apache Kafka, ROS Framework, API Gateway, Debezium
2. Processing & Analytics	Execution of complex predictive, optimization, and classification AI/ML algorithms based on the collected data. Supports data analysis and preprocessing tasks.	AI/ML Services (Dockerized), Data Analysis Service, PostgreSQL/Cassandra DB
3. Data Governance & Trust	Ensures data integrity and confidentiality, and secures data trustworthiness through blockchain-based trust mechanisms.	Private Blockchain (Port), Encryption/Access Control Module
4. Application & User Interface (UI)	The final point of interaction, visualizing the results of AI/ML services, providing real-time operational information to users, and collecting operational feedback.	User Interface (Real-time Dashboard), Management Service

2.1.2. Container-Based Microservices Architecture

The SOCFAI Platform adopts a Microservices Architecture and implements it on a container-based (Docker) foundation to maximize scalability, independent deployment, and flexibility.

A. Key Components and Roles

Each Use Case operates as an independent service while leveraging the platform's common components (e.g., based on the system architecture of Use Case #2 - Siemens).

- User Interface (UI): The web page (dashboard) where end-users view real-time statistics, prediction information, and interact with the system.
- Management Service: The central control module that handles requests from the dashboard and manages communication with backend services like the Data Analysis Service.
- Communication Service: Responsible for real-time data exchange between the SOCFAI Platform and external systems, and handles ingesting received data into the database.
- Data Analysis Service: Supports the analysis of data stored in the database and aids in data preparation (preprocessing) and related tasks for AI/ML models.
- AI/ML Services: A group of independent containers that execute AI/ML algorithms specialized for specific Use Cases, such as CatBoost, XGBoost, VoxelNeXt, and PPO. (E.g., VoxelNeXt for Use Case #1, Daily Baggage Prediction for Use Case #2, and ATAD Prediction model for Use Case #11 exist as separate Docker containers).
- Database: Stores the data required for platform operation. For Use Case #2, Cassandra, a NoSQL database, was mentioned as a suitable solution for high throughput and scalability.

B. MLOps Framework Integration

The platform integrates MLOps (Machine Learning Operations) tools to ensure the continuous operation and management of AI models.

- Model Management: MLflow is used to track AI model training experiments and perform model version control.
- Object Storage: MinIO, a scalable object storage solution, is used to store trained model artifacts and large-volume data.
- Continuous Monitoring: Tools like NannyML are utilized to detect and monitor model performance degradation (Data Drift) in the operational environment, ensuring model stability and reliability.

Through this, the SOCFAI Platform provides a common technological base for deploying, serving, and managing the diverse AI models derived from each Use Case in a sustainable and scalable manner.

2.2. Data Integration & Streaming Engine

The SOCFAI Platform is designed to handle high-volume, high-velocity data from diverse operational and sensor sources across both airport and port environments. This requires a robust and scalable data integration and streaming engine to ensure real-time data availability for the AI/ML services.

2.2.1. Kafka-Based Real-time Data Pipeline Configuration

Apache Kafka is employed as the core Distributed Event Streaming Platform to manage the real-time data pipelines. Kafka ensures reliable, high-throughput, and low-latency communication between various systems and the SOCFAI Platform's processing services.

- Real-time Data Flow: For Use Case Applications, Kafka acts as the message broker, facilitating the transfer of real-time operational data.
    - In Use Case #2 (Baggage Prediction), the Communication Service receives real-time flight and baggage data and populates it into the database. This communication layer often relies on technologies like Kafka to handle the data stream.
    - In the MLOps Pipeline, real-time flight operation data from the SOCFAI (AOCC) system is streamed via Kafka topics to the Prediction Service Docker containers for inference. Kafka is also used to stream the prediction results back to the tracking systems.
- LiDAR Sensor Integration: For Use Case #1 (Passenger Monitoring), data is initially processed on an INOSENS server after acquisition. The coordinates of detected humans and derived crowd density information are then transmitted to the airport system via Apache Kafka for high-throughput, real-time communication.
- Port Logistics Integration: In Use Case #5 (Port Optimization), the data pipeline utilizes Debezium and Kafka Connect to capture Change Data Capture (CDC) events from Port Logistics DB servers (CFSMS and TMS). This ensures that database changes are ingested and converted into Kafka topics efficiently and reliably.

2.2.2. External System Interworking Mechanism (API Gateway, SOCFAI Platform Connection)

The platform must seamlessly connect with a variety of proprietary and standard operational systems to access the necessary data.

- Diverse Data Sources: The platform integrates with critical systems:
    - Airport Operational Systems (AODB/FMS): Providing flight schedules, service times (like inblock, offblock, gate, check-in), and aircraft characteristics.
    - Real-time Tracking Systems (ADS-B): The ADS-B-based short-term delay prediction framework (Use Case #11) relies on real-time data from sources like Flightradar24.
    - IoT/Sensor Data: The system connects with LiDAR sensors (e.g., Velodyne VLP-16) for 3D point cloud data and other IoT platforms (like ION in Use Case #3) for air quality/HVAC sensor data.
- Partner Access Platform (PaP): While not explicitly detailed in the provided documents, a Partner Access Platform (PaP) is typically used in such collaborative frameworks (as mentioned in public project descriptions) to allow stakeholders to connect, perform tests, and synchronize data with external cloud environments, acting as a controlled API Gateway for data exchange.
- Data Standardization: Crucially, data received from these disparate systems must be standardized and normalized before being stored or fed into the AI/ML models. This standardization is a fundamental prerequisite for reliable data processing across all Use Cases.

2.3 AI/ML Model Operations (MLOps) and Management

The SOCFAI Platform establishes a robust MLOps (Machine Learning Operations) architecture to ensure the developed AI models are not only accurate but also sustainable, scalable, and manageable throughout their entire lifecycle in live operational environments. This framework allows the platform to adapt quickly to evolving operational needs, generating reliable predictions across all Use Cases.

2.3.1. Experiment Tracking and Version Management (MLflow)

The MLOps architecture incorporates mechanisms for systematic tracking and management of the diverse AI models developed across the project.

- Experiment Tracking: MLflow is integrated into the architecture for comprehensive experiment tracking and model management. This tool allows developers to record and compare various model training runs, including parameters, metrics (like MAE or accuracy), and resulting model artifacts.

- Version Control: The platform implements a Version Control and Model Registry Strategy (as defined in the Use Case AI Model deliverable) to meticulously track changes to model code, data, and configurations. This ensures reproducibility and facilitates the promotion of approved model versions to production.

2.3.2. Model Deployment and Service Provision (Jenkins, Docker)

The models must be deployed efficiently and securely to provide real-time inference results to the operational systems.

- Containerization: All AI/ML models are containerized using Docker. This ensures that each model (e.g., the CatBoost baggage predictor or the VoxelNeXt human detector) runs consistently across different environments, from development to production.
- Deployment Architecture: The deployment plan involves models residing within the platform's container-based microservices architecture, often acting as independent Prediction Service modules.
- Service Provision: These Dockerized services provide real-time inference via an API server to be integrated with existing applications and systems. For example, the ADS-B model's output is designed to facilitate the dynamic optimization of critical airport operations, such as gate assignments and ground handling coordination. While specific CI/CD tools like Jenkins were not detailed in the provided snippets, the commitment to an MLOps architecture implies the use of automated pipelines for building, testing, and deploying these Docker containers.

2.3.3. Model and Metadata Repository (MinIO, PostgreSQL)

Centralized repositories are essential for storing and managing model artifacts and their associated metadata.

- Model Artifact Storage: MinIO is utilized for scalable object storage of large assets, including the trained model files themselves and potentially large datasets or artifacts generated during the training phase.
- Metadata Storage: A relational database like PostgreSQL is typically used as the Model Registry backend to store metadata associated with each model version—such as training parameters, performance metrics, and deployment status—providing a single source of truth for model governance.

2.3.4. Continuous Monitoring and Drift Detection (Evidently AI, NannyML)

Once deployed, models are subject to continuous monitoring to maintain their predictive reliability.

- Continuous Monitoring: The platform is equipped with tools to ensure the sustainability and reliability of the deployed models.
- Data Drift Detection: NannyML is specifically incorporated into the MLOps architecture for data drift detection. This is crucial because operational data (e.g., flight patterns, passenger density) can change over time. Detecting drift ensures that the models adapt quickly to these evolving operational needs and that model output remains reliable, minimizing the need for manual intervention.
- Performance Monitoring: The architecture also includes provisions for monitoring model performance metrics (e.g., test score, error margins) during training and continually comparing them to live performance, flagging when a model may need retraining.

3. Airport Data Sources and Data Processing

3.1. Key Data Sources

3.1.1. Flight Operation Information: FMS, AODB (Flight Management System, Airport Operational DB)

The data consists of operational details for scheduled and active flights at the airport. Sources include the Flight Management System (FMS) and the Airport Operational Database (AODB).

- **FMS** provides flight plan data such as route, estimated times, and performance parameters, which are essential for navigation and fuel optimization.
- **AODB** contains airport-level operational data, including flight schedules, gate assignments, and resource allocation. This ensures coordination between airlines, ground handling, and airport services.

Each flight record includes attributes such as flight number, origin, destination, scheduled and actual times, and aircraft type. These attributes are used to monitor operational performance and predict turnaround efficiency. Data quality checks are applied to handle inconsistencies between scheduled and actual times before integration into analytics pipelines.

3.1.2. Real-time Location/Flight Data: ADS-B (Flightradar24)

This dataset consists of real-time positional and status information for aircraft in operation. The primary source is ADS-B signals, aggregated by platforms like Flightradar24.
Each record includes latitude, longitude, altitude, ground speed, and heading, along with timestamps for continuous tracking. This data is critical for situational awareness and live monitoring of flight progress.

To ensure reliability, missing or delayed ADS-B signals are interpolated using previous positions and estimated trajectories. The processed data supports applications such as congestion analysis, delay prediction, and integration with airport operational systems for dynamic resource planning.

3.1.3. Environment/Weather Data: LiDAR Point Cloud, IoT Sensors (Velodyne VLP-16, METAR)

The dataset consists of environmental and atmospheric measurements collected through advanced sensing technologies. Sources include **LiDAR point clouds**, **IoT sensors** such as **Velodyne VLP-16**, and meteorological reports like **METAR**.

- **LiDAR Point Cloud (Velodyne VLP-16):**
  Provides high-resolution 3D spatial data of the airport environment. Each point in the cloud represents a precise location with attributes such as elevation and intensity, enabling accurate modeling of terrain, obstacles, and infrastructure. This data is essential for applications like autonomous vehicle navigation and surface condition analysis.

- **IoT Sensors:**
  Velodyne VLP-16 sensors capture real-time spatial and distance measurements, while additional IoT devices monitor parameters such as temperature, humidity, and air quality. These sensors enable continuous environmental monitoring and integration with operational systems for safety and efficiency.

- **METAR Reports:**
  Standardized meteorological data from aviation weather stations, including wind speed, visibility, temperature, and pressure. METAR data is critical for flight planning and operational decision-making under varying weather conditions.

Data preprocessing includes filtering noise from LiDAR scans and synchronizing sensor readings with METAR timestamps. Combined, these sources support predictive models for runway conditions, visibility analysis, and dynamic resource allocation during adverse weather.

3.1.4. Social Media and Text Data (Online Sources)

The training data consists of reviews of various international airports, including Izmir Airport. The training data is in the English and Turkish languages.

Each review from the training data has a review score from a scale of 1 to 5. This review score indicates the sentiment. A function that converts the review score to the corresponding sentiment was incorporated into the model training pipeline. A score of 4 or 5 is positive, a score of 3 is neutral, and a score of 1 or 2 is negative.

The neutral class had noticeably less reviews compared to the other two classes. Thus, the nlpaug Python library was used to generate reviews similar to the already existing neutral reviews to handle the imbalance in the training data.

3.2. Data Preprocessing and Refinement

3.2.1. LiDAR Point Cloud Processing and 3D Data Labeling (KITTI format)

Recording Raw Data and Converting to PCD Format:

Raw LiDAR data collected from the Velodyne VLP-16 sensor were first recorded in ROS as .bag files, preserving the full timestamped message structure of each frame. These ROS bag files were then converted into PCD (Point Cloud Data) format to obtain a frame-based, file-structured representation of the point cloud data suitable for offline processing.

Region of Interest (ROI) Filtering:

The generated PCD files were processed using a custom ROI filtering script to retain only the spatial region relevant to human detection. This step removed unnecessary points, significantly reduced data size, and focused the dataset on the target area. The cropped point clouds formed the basis for the subsequent annotation procedure.

3D Annotation Using Supervisely:

The cropped PCD files were uploaded to the Supervisely platform, where each scene was manually annotated with 3D bounding boxes corresponding to Pedestrian instances. Bounding boxes were adjusted according to realistic human dimensions and aligned with the point cloud geometry.

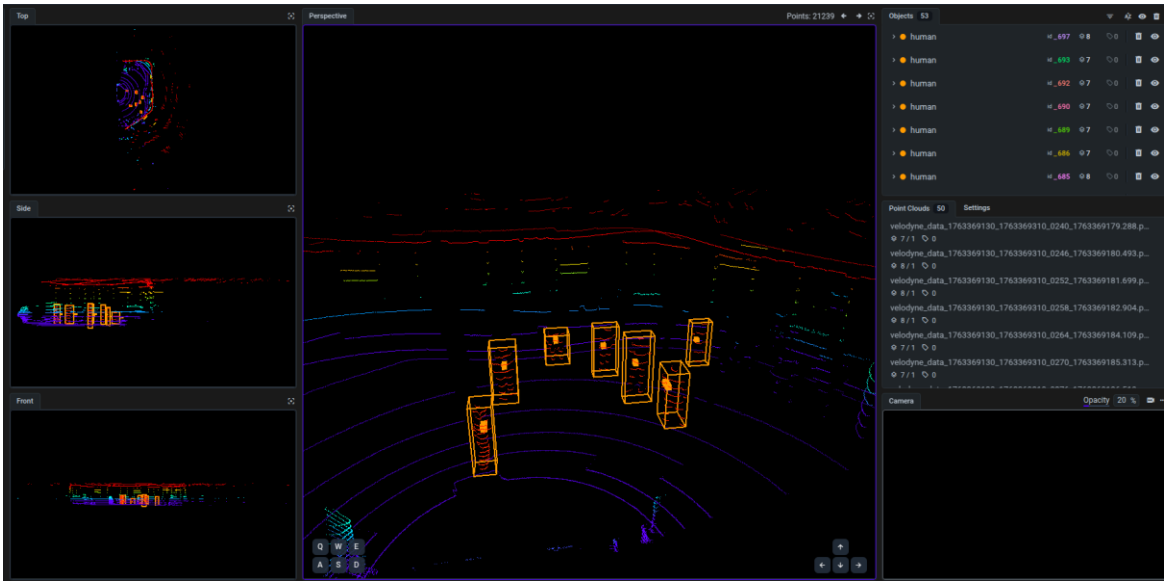


Figure 1. *Supervisely 3D annotation interface*

Cleaning JSON Labels and Converting to TXT Format:

Supervisely exports annotations in JSON format, which includes various unnecessary metadata such as camera calibration fields, object IDs, and visualization parameters. To prepare these labels for the VoxelNeXt model, the JSON files were parsed and cleaned, preserving only the essential 3D bounding box parameters:
(x, y, z, w, l, h, yaw, class).
The cleaned annotations were saved in a compact TXT format, with one bounding box per line.

Converting PCD Files to NPY Format:

Since VoxelNeXt operates directly on NumPy arrays, all PCD point cloud files were converted into .npy format. During this conversion, each point was stored in the standard LiDAR representation:
(x, y, z, intensity).
The resulting NPY files provide efficient and model-ready inputs for training and evaluation.

Matching File Names for NPY–TXT Consistency:

To maintain dataset integrity, each .npy file was paired with a .txt annotation file using identical filenames. For example:

```
000153.npy  → point cloud
000153.txt  → corresponding labels
```

This one-to-one naming strategy ensures that the model can reliably associate each point cloud with its corresponding annotations, enabling a seamless VoxelNeXt training workflow.

3.2.2. Baggage Data Refinement and Feature Engineering (Dimensionality Reduction)

Tables with baggage and flight info are used for data refinement. Old format v_bag table contains baggage info for all TAV airports while new format baggage table contains baggage info for only ADB airport with additional features like baggage life cycle. Old format brs_flight contains flight info for all TAV airports while the new format flight table contains flight info for only ADB airport with additional features like transfer flight connections. Both old formats (v_bag, brs_flight) and new formats (baggage, flight) data are supported at the same time so that a generic solution including all TAV airports with new data features is targeted. The new format is transformed into the old format with additional columns so that both can be processed at the same time. Topics with missing records, transfer info, bag route info, record status, local & utc mixed timing are handled during the transformation process so that matching and lossless data is achieved.

Clean data is formed from baggage and flight tables after three preprocessing steps. The steps are morphological operations, baggage pattern analysis, repetition-outlier exclusion. Cells are

removed, filled, simplified and combined during morphological operations. Columns with occupancy rate below 100% are identified. Columns with empty cells are joined to achieve 100% occupancy rate. Some sample logics are described in following bullet points:

- ATD (Actual Time of Departure) and ATA (Actual Time of Arrival) are combined into ACTUAL_TIME; STD (Scheduled Time of Departure) and STA (Scheduled Time of Arrival) are combined into SCHEDULED_TIME; ETD (Estimated Time of Departure) and ETA (Estimated Time of Arrival) are combined into ESTIMATED_TIME regarding arrival or departure flight.
- SEAT_NUMBER, SECURITY_NUMBER, SEQUENCE_NUMBER, FILE_NAME is combined into PASSENGER_ID. i.e., 04B-33-33.
- BAG_NUMBER with empty cell is filled with the expected average bag number.

Regarding pattern analysis of baggage records, two major approaches are studied. Firstly, average hourly baggage arrival pattern for each flight code is calculated for several flights in distinct days. Pattern Dissimilarity (PD) for a single flight here is defined as shown in formula 1. Numerator b_{ij} is the absolute difference for flight i at hour j and equals to |Hourly Baggage for Flight i at Hour j−Mean Hourly Baggage for Flight Code at Hour j|.

Denominator a_j is the mean hourly baggage for the flight code at hour j. Please note that hour j here refers to delta before flight time.

$$PD_i = \frac{\sum_{j=0}^{T}(b_{ij})}{\sum_{j=0}^{T}(a_j)} \qquad (1)$$

Let us there are 3 distinct flights for a flight code. Baggage count for 1 hour and 2 hours before flight time is examined. Flight 1: 10 baggage at hour 1, 30 baggage at hour 2; Flight 2: 5 baggage at hour 1, 20 baggage at hour 2; Flight 3: 15 baggage at hour 1, 10 baggage at hour 2. Mean baggage count: (10 + 5 + 15) / 3 at hour 1 equals to 10; (30 + 20 + 10) at hour 2 equals to 20. PD for Flight 1 is calculated as (|10 – 10| + |30 -20|) / (10 + 20) and equals to 0.33 (33%).

The measure of PD is needed to demonstrate correlation with flight delay possibility. If there is an increase in PD of a flight, it directly impacts flight delay. For each calculated PD information, relation between flights is analyzed based on a threshold approach. A flight is marked delayed if the time difference between ACTUAL_TIME and SCHEDULTED_TIME is greater than 15 minutes. Overall number of delayed flights are counted and the delay ratio for each threshold is calculated. So that the correlation between PD ratio and flight delay is shown in Table 1.

Table 1. Correlation Between Pattern Dissimilarity Ratio and Flight Delay

Pattern Dissimilarity Ratio Threshold	Flight Delay Ratio	Delayed Flight Count	Total Flight Count

30	57.30%	2199	3838
40	60.36%	1457	2414
50	61.75%	980	1587
60	66.98%	720	1075
70	68.61%	518	755
80	69.96%	396	566
90	71.87%	327	455
100	73.68%	266	361

Flight delay ratio over PD ratio threshold in given sample data in Table 3.1 is also displayed in Figure 3.1. Results indicate that the percentage of unexpected baggage records affect flight operations negatively such as delay in our case.

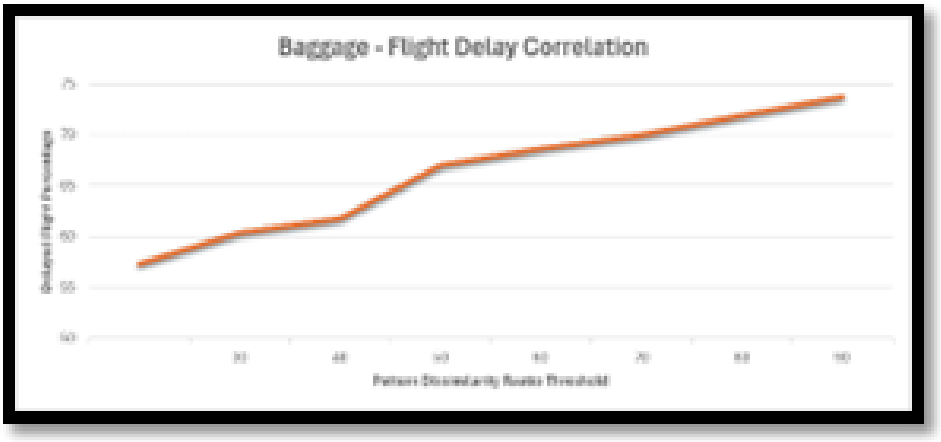


Figure 2: Pattern Dissimilarity Ratio – Flight Delay Correlation.

As a second pattern analysis of baggage records, a new feature is generated with the name of MULTI_BIM_CREATE. It is calculated for each baggage record as the number of repetition baggage records with the same ID, BIM_CREATE_DATE, PASSENGER_ID having STATUS_INDICATOR changed or deleted. Repetition records may appear in baggage dataset due to several reasons such

as baggage rehandling, integration from multiple sources. For each flight, the percentage of baggage records that has repetition (MULTI_BIM_CREATE is greater than 1) over total baggage records is computed as shown in Table 2. For example, Flight ID 4 has a total 154 baggage records. 42 records have repetition. So, MULTI_BIM_CREATE_RATIO is calculated as 27.27%.

Table 2: Multi BIM Create Ratio per Flight ID.

ID	MULTI_BIM_CREATE_RATIO
1	0.5 (1/199)
2	3.33 (2/60)
3	13.7 (10/73)
4	27.27 (42/154)

Correlation between MULTI_BIM CREATE_RATIO and flight delays is displayed in Table 3. A flight is marked as delayed if the time difference between ACTUAL_TIME and SCHEDULTED_TIME is greater than 15 minutes. For example, there are 464 flight records that have MULTI_BIM_CREATE_RATIO >= 15% and 66.16% of flights have delays in the first row.

Table 3: Correlation Between Multi BIM Create Ratio and Flight Delay.

MULTI_BIM_CREATE_RATIO	Delay Ratio	Delayed Flight Count	Total Flight Count
15	66.16%	307	464
16	66.83%	268	401
17	69.01%	236	342
18	70.3%	213	303
…	…	…	…
59	90.0%	9	10
60	100.0%	7	7

Flight delay percentage over MULTI_BIM CREATE_RATIO threshold is displayed in Figure 3. It indicates that the percentage of repetitive baggage records affect flight operations negatively such as delay.

Figure 3: MULTI_BIM CREATE_RATIO – Flight Delay Correlation.

Outlier baggage records are excluded based on rules such as empty PASSENGER_ID, BIM_CREATE_DATE > ACTUAL_TIME and earlier timestamps that do not exist in the pre-determined date range. Repetition records are eliminated based on evaluating cases of BIM_CREATE_DATE, BIM_ID, FILE_NAME, ID, STATUS_INDICATOR and PASSENGER_ID feature relations. After pre-processing, hourly baggage arrival patterns for each flight code are again visualized in Figure 4. Furthermore, the arrival pattern of baggage records is distributed in a reasonable way. Figure 3.3 indicates most activity concentrated in the few hours leading up to the flight time. Finally, because of data cleaning process, feature selection and feature merge, dimensionality reduction is also obtained from 82 to 23 features.

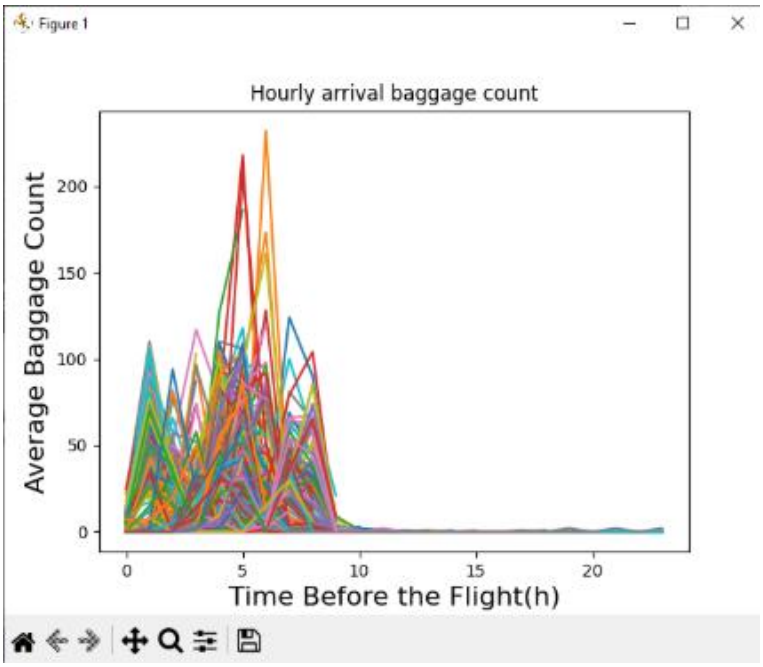


Figure 4: Hourly Baggage Arrival Pattern per Flight Code (Final).

4. Airport Specialized AI Models and Services

4.1 Passenger Flow and Density Management

Efficient passenger flow and density management is a critical component of modern airport operations, directly influencing passenger experience, safety, queue stability, and overall terminal efficiency. In this project, LiDAR-based 3D human detection is integrated with an AI-powered analytics service to provide real-time insights on crowd density and movement dynamics. By leveraging high-resolution 3D point cloud data from the Velodyne VLP-16 sensor and the VoxelNeXt detection model, the system continuously monitors passenger distribution, flow direction, congestion patterns, and high-traffic zones within airport terminals.

4.1.1. AI Model: VoxelNeXt (3D Human Detection)

The core AI component of the system is the VoxelNeXt 3D object detection model, which performs real-time pedestrian detection directly on LiDAR point clouds. VoxelNeXt is a sparse-convolutional neural network optimized for high-accuracy human detection in complex indoor and semi-indoor environments such as airport waiting areas, boarding gates, check-in halls, and security checkpoints.

Key characteristics of the VoxelNeXt model within this project include:

- 3D Human Detection:
  Accurately identifies individual passengers in cluttered environments using LiDAR point clouds.
- Robustness to Occlusions:
  Performs reliably even when passengers occlude one another within dense crowds, thanks to its voxel-based spatial understanding.
- High Precision in Varying Lighting Conditions:
  Since LiDAR is independent of ambient light, the model provides consistent performance in day, night, shadowed, or backlit environments.
- Real-Time Inference:
  Optimized on CUDA-accelerated architectures, enabling continuous monitoring without delay.
  Integration with ROI-based Cropping:
  Only the predefined regions of interest in the terminal are processed to maximize efficiency and reduce false positives.

The output of VoxelNeXt consists of 3D bounding boxes representing detected pedestrians. These detections serve as the primary input for flow and density analytics.

4.1.2. Service: Passenger Density Estimation (3-Level Classification), Passenger Flow Monitoring

Once the VoxelNeXt model detects passengers in the scene, the system processes these detections to provide two critical analytical services:

a) Passenger Density Estimation

Once the VoxelNeXt model detects passengers in the LiDAR point cloud, all 3D bounding boxes are filtered by a predefined Region of Interest (ROI) corresponding to the area of interest in the terminal (e.g., queueing/counter zone). For each frame:

- The system counts the number of active passenger tracks inside the ROI.
- In parallel, it computes a spatial occupancy metric, defined as:

$$\text{Occupancy}(\%) = \frac{\sum_i (dx_i \cdot dy_i)}{\text{ROI\_area}} \times 100$$

Where dx_i and dy_i are the ground-plane dimensions of each detected passenger box, and ROI_area is the area of the monitored zone in square meters.

Based on this occupancy value, the system classifies the density into three semantic levels:

- Low Density
  - Sparse passenger distribution inside the ROI

- Occupancy remains below an operational comfort threshold
- No congestion risk; normal operating conditions

- Medium Density
  - Noticeable increase in passenger count and occupied floor area
  - Potential queue formation in front of service points
  - Recommended to start monitoring for bottleneck development and prepare countermeasures

- High Density
  - Crowd accumulation beyond predefined safety/comfort limits
  - Elevated risk of congestion, delays, and local safety concerns
  - Automatically flagged as an alert state, which can trigger operational responses such as opening additional counters, redistributing staff, or temporarily redirecting passenger flow

This LiDAR-based spatial occupancy estimation provides a camera-agnostic, privacy-preserving view of how intensely a terminal zone is being used at any given time.

b) Passenger Flow and Queue Dynamics Monitoring

Beyond static density estimation, the system continuously tracks passengers over time to characterize flow dynamics and queue formation inside the ROI.

Using frame-to-frame 3D tracking:

- Each detected passenger is assigned a persistent track ID based on 2D ground-plane distances (x–y), ensuring robust tracking despite height changes or minor detection noise.
- For each track, the system maintains a trajectory history and instantaneous speed estimate (m/s), smoothed over time to reduce jitter.

On top of this tracking layer, several flow-related indicators are computed:

- Direction of Movement & Ingress/Egress
  - A virtual vertical line (flow line) is defined in the ROI (e.g., x=0 m).
  - When a passenger's trajectory crosses this line from left to right, it is counted as Ingress (In); from right to left as Egress (Out).
  - Using a sliding time window (e.g., 60 seconds), the system estimates flow rates in passengers per minute for both directions:
    1. FlowIn (/min)
    2. FlowOut (/min)

- Queue Zone Monitoring
  - A dedicated Queue Zone is defined as a sub-region within the ROI (e.g., in front of check-in or security counters).
  - For each frame, the system counts how many tracked passengers are inside this zone.
  - If the number of passengers in the Queue Zone exceeds a configurable threshold (e.g., ≥ 3 people), the system flags "Queue: YES" and visually highlights the zone.

- Bottleneck Detection (Slow/Stopped Flow)

- The average speed of all tracks inside the ROI is computed per frame.
- When High Density coincides with low average speed (below a minimum movement threshold), the system classifies the situation as a bottleneck:
    1. Many people present
    2. Movement is slow or nearly stationary
- This condition is logged and can be used to trigger alerts or staff interventions.

- Queue Build-up Visualization
    - Tracks that remain within a small spatial radius (e.g., 1–1.5 m) for longer than a predefined time (e.g., ≥ 5–10 s) are marked as queue candidates.
    - Additionally, all per-frame analytics (density, flow, bottleneck flags, queue indicators, queue-zone counts) are stored in an analysis.jsonl file, enabling offline plotting and statistical analysis (e.g., occupancy vs. time, cumulative in/out flow, queue zone population over time).

These flow and queue analytics provide airport operators with a temporal view of how crowds enter, occupy, and leave critical service areas. This helps identify not only where congestion occurs, but also when and under which conditions, supporting data-driven decisions on terminal layout, signage, and staffing strategies.

4.2. Flight and Baggage Prediction

4.2.1. Short-Term Arrival Delay Prediction (ATAD Prediction): GPU-accelerated Random Forest (ADS-B data based)

4.2.1.1. Model Details

a. Data Pipeline & Feature Engineering

The ATAD prediction model relies on a rich flow of real-time information that reflects how an aircraft behaves during its final phase of flight. At the heart of this system is ADS-B data, which streams in every few seconds and captures the aircraft's exact position, speed, altitude, and heading. On its own, this data already paints a detailed picture of the flight, but it becomes far more meaningful when combined with information from the Flight Management System (FMS), such as the planned schedule, the type of aircraft, and the intended route. When weather reports are available—things like wind, visibility, or sudden changes in conditions—they also help the model understand why an aircraft may be slowing down, speeding up, or adjusting its descent pattern.

Before any machine learning begins, the system puts this data through a structured preprocessing stage. All data streams are aligned at 10-second intervals so that every snapshot of the aircraft is synchronized across all available sources. This alignment allows us to build features that truly reflect how the aircraft is moving in real time. We look at classic kinematic indicators such as altitude, ground speed, vertical rate, and distance to the airport, but we also create more

insightful features like how much the aircraft is deviating from its expected route or how its current behavior compares to typical descent profiles.

Some flights behave differently—not because of normal physics, but because of operational events such as holding, vectoring, or extended sequencing. These cases often create noise rather than insight for the model. For that reason, the training process automatically identifies such situations and sets them aside for rule-based evaluation. This keeps the training dataset "clean" and ensures that the model learns natural arrival patterns rather than exceptional or operationally driven ones.

b. Model Selection & Training

For the prediction task, we selected a GPU-accelerated Random Forest Regressor. Random Forests are known for being stable, robust, and particularly effective when working with varied data sources—exactly the environment of a real airport. By running the model with NVIDIA's cuML library, we dramatically reduce training time and gain the ability to run predictions in real time without delay.

Training the model involves much more than simply feeding data into an algorithm. Using 5-fold cross-validation, we check that the model performs consistently across different subsets of the data, ensuring that the predictions are reliable and not overly tuned to specific scenarios. Then, using Bayesian hyperparameter optimization with Optuna, we fine-tune key parameters such as the depth of the trees or how many features each tree considers. This method allows the algorithm to find its best configuration efficiently and intelligently.

The target we aim to predict is the time remaining until arrival, expressed in seconds. By predicting this continuously throughout the entire inbound segment, the system becomes progressively more accurate as the aircraft gets closer, giving airport teams a dynamic and early awareness of expected landing times.

c. Model Performance & Results

To measure success, we use the Mean Absolute Error (MAE), which tells us how far off, on average, our predictions are from the actual arrival times. The final tuned model achieves an MAE of about 109 seconds, meaning it predicts arrivals with an accuracy better than two minutes—a valuable margin for airport operations that often work under strict time constraints.

We tested several algorithms along the way, and the results speak clearly:

Table 3:Tunes Model Test Results.

Model	Train MAE	Test MAE
XGBoost	6.71	5.80
CatBoost	6.03	5.91
Random Forest	3.56	3.61
Random Forest Tuned	2.41	2.23

The tuned GPU-accelerated Random Forest outperformed other models both in speed and accuracy, making it the natural choice for real-time deployment.

When we examine which features matter most, factors such as distance to the airport, the aircraft's angular approach, speed, and recent trajectory changes rise to the top—no surprise, as these are the elements that reflect the aircraft's physics most strongly. The outcome is a prediction engine that genuinely understands how an inbound flight behaves.

More importantly, these predictions translate into real operational value. Airport teams can reassign gates earlier, adjust ground handling plans with more confidence, and manage staffing in a more efficient and predictable way. Even small improvements in timing can ripple through the airport ecosystem, reducing delays and improving passenger flow.

4.2.1.2. System Architecture & Deployment

a. Architecture Overview

The ATAD prediction model does not stand alone—it lives inside a modern, flexible system designed to work smoothly in a live airport environment. The core prediction service runs in a Docker container, making it easy to deploy, replicate, or scale. Real-time data from ADS-B and FMS sources flows into the system through Kafka, which ensures that even high-volume data arrives reliably and quickly.

Once the data reaches the prediction service, the model generates updated arrival-time estimates almost instantly. Alongside this, a rule-based engine operates in parallel, looking for special conditions that the machine learning model should not interpret on its own, such as holding patterns or unusual transmitter behavior. Together, the ML model and rule engine offer

a balanced approach: intelligent predictions when patterns are normal, and deterministic logic when operations require special handling.

b. Deployment Pipeline

The operational pipeline is designed for clarity and reliability from end to end. Kafka acts as the entry point, collecting incoming flight data and distributing it to the prediction service. The Dockerized model then processes each update and calculates the ATAD, sending the results to the dashboard and downstream systems that depend on them. Meanwhile, the rule-based engine monitors the same stream and adds contextual flags to help operators understand why certain predictions may change.

Because the entire system is optimized for low latency, it keeps pace with real-time flight operations, allowing controllers and coordination teams to make decisions based on fresh, continuously updated insights.

c. Monitoring & MLOps

Behind the scenes, a full MLOps ecosystem keeps the model healthy and reliable. MLflow tracks every trained model, experiment, and parameter set, ensuring full transparency. Tools like Evidently AI and NannyML monitor the system continuously, watching for signs that the data has changed or that the model's accuracy is shifting over time.

If the environment evolves—as flight patterns change, seasons shift, or airlines adjust operations—the system supports automated retraining. This keeps the model aligned with real-world behavior so that it stays useful even as airport dynamics evolve.

d. Integration with Airport Operations

The predictions generated by the system appear directly on the airport's operational dashboard, where they are easy to follow and act upon. RESTful APIs allow other systems—such as gate planners or baggage teams—to access the predictions in real time and incorporate them into their workflows.

Because the architecture is modular and container-based, scaling to other airports or higher data loads is straightforward. New nodes can be added, new Kafka topics spun up, and the system continues to operate seamlessly.

4. Key Takeaways

The ATAD prediction system blends advanced machine learning with real-time operational data to give airports a clearer, earlier, and more accurate view of when flights will arrive. By relying on

a tuned GPU-accelerated Random Forest model, enriched with well-crafted features and supported by a robust data pipeline, the system can predict arrival times within a two-minute margin—enough to make a meaningful difference in airport decision-making.

Under the hood, containerized microservices, continuous monitoring, and automated retraining ensure that the solution remains both reliable and adaptable. And on the operational side, the predictions support smoother gate usage, more efficient ground handling, and better overall flow across the airport.

In short, the system turns raw data into a practical advantage, helping airports shift from reacting to delays to planning ahead with confidence.

4.2.2. Daily Baggage Prediction: CatBoost (Best-Performing Model)

Prediction using AI models requires choosing the most suitable method for the problem. This is a time related prediction and regression problem, therefore time-series models are the most probable ones to solve this. There may be candidates from both machine learning (ML) and deep learning (DL) models for this prediction. Hence, many different models were trained to find the model that achieved the best performance for daily prediction. Random forest, XGBoost, linear regression, and CatBoost are ML models and LSTM, BiLSTM, CNN, and CNN-LSTM hybrid models are DL models that were trained.

Model development with all experiments were completed as the same environment and resources with the ones in the Pattern Prediction section that you can see below. While training models two different metrics are used as loss functions. One of them is mean squared error (MSE) and the other one is root mean squared error (RMSE). Models aim to minimize these two errors while learning the general pattern in the data. In the testing phase, some different metrics are also used to better understand the accuracy of models. These are mean absolute error (MAE) and R^2 Score. CatBoost gives the best test score with minimum MAE among many trained models. Hyperparameter optimization requires trying many values for different kinds of parameters to make a model with the best accuracy. For this purpose, Grid Search and Optuna methods were applied in every model experiment to find the optimized valued of following parameters:

Parameter	Description
depth	Depth of trees
l2_leaf_reg	Coefficient at the L2 regularization term of the cost function
learning_rate	The learning rate used for training

The data is grouped by individual flights, as each flight possesses unique characteristics such as hour, day, month, and arrival and departure airports. These flight groups are then split into training and test sets internally. An 80/20 split strategy is applied, allocating 80% of the data to training and 20% to testing. To implement this strategy, a rule was established requiring each flight to have occurred at least 4 times. If a flight has fewer than 4 occurrences, all instances are assigned to the training set. Otherwise, they are divided according to the specified 80/20 ratio.

A 5-fold cross-validation approach was implemented within the training subset to assess model robustness and optimize hyperparameters. A similar splitting strategy was applied here, ensuring that samples from every flight group were represented in each fold. In each iteration, 80% of the training data (four folds) was allocated for training and 20% (one fold) for validation. This rotation created five distinct train-validation splits, enabling evaluation of the model's generalization ability across various data configurations.

4.2.3. Hourly Baggage Pattern Prediction: Mean-Shift, Decision Tree, Polynomial Regression Hybrid Approach

Three steps of machine learning are used for hourly prediction as mean-shift unsupervised classification, decision tree supervised classification, and polynomial regression. The baggage patterns are classified by using mean shift unsupervised classification with 60% bandwidth similarity. Pattern classes that come from unsupervised classification are used as target labels at the decision tree classification. Two different models are used to achieve class prediction with and without flight code so that if it is the first flight of this flight route (means a new flight code) the model can still predict the flight class by using destination route and airplane company code. Finally polynomial regression is used to model each baggage pattern as a polynomial expression.

Mean-shift is used for labeling the patterns because its arrangeable similarity parameter fits perfectly the classification requirement during the tests. The decision tree is the best model that leads to the most accurate regression matching. Due to the baggage arrival pattern showing polynomial behavior, polynomial regression is chosen for most accurately modeling the pattern.

During training models mean squared error (MSE) and root mean squared error (RMSE) metrics are used as loss function. These parameters are tried to be minimized during training. Mean absolute error (MAE) and R^2 Score are also used in the testing stage to improve model accuracy.

70% of the filtered clean data with low flight delays, high baggage volume as much as possible is used for training and 30% of the overall set is used for testing the models. As a result, R2: 0.94, RMSE: 8.243, MAE: 2.598 is achieved for hourly baggage prediction of 2563 distinct flight codes, from 15 different airports.

4.2.4. Minutely Baggage Pattern Prediction: RandomForest (Best-Performing Model)

Employed AI models here are systematically grouped under three main methodological categories: machine learning, deep learning, and time-series modeling. The machine learning category includes ensemble and linear models such as Random Forest, Gradient Boosting, Bagging Regressor, ExtraTree Regressor, Decision Tree etc. These models are good for handling structured tabular data. The deep learning category is represented by the Deep Neural Network (DNN), which excels in modeling high-dimensional data through layered abstractions and non-linear transformations. In the time-series modeling category, the Long Short-Term Memory (LSTM) network is utilized, as it is specifically designed to capture temporal dependencies and sequential patterns within historical baggage data.

The proposed models were ensured through k-fold cross-validation with k = 5, allowing for balanced performance evaluation across different data partitions. Hyperparameter optimization was performed using a systematic grid search strategy. All computational experiments were executed on a high-performance workstation equipped with a dual-GPU setup featuring two NVIDIA GeForce GTX 1080 Ti units (each with 11 GB GDDR5X memory, supporting CUDA version 12.2).

Overall performance comparison is shown in Table 4.1. The regression results indicate that tree-based ensemble models outperform other model families across all evaluation metrics. The ExtraTreeRegressor demonstrates the best overall performance, achieving the lowest RMSE (0.028) and MAE (0.019), alongside the highest R^2 score (0.607). This suggests its strong capability to model complex, non-linear relationships. However, both ExtraTree and Bagging are known to produce very large model sizes, which can be a significant consideration in deployment scenarios where memory or latency constraints exist. Memory consumption or occupied disk size may affect edge deployment in airport terminal systems used for real-time decision-making. Additionally, such ensemble models can bring out substantial inference latency due to the need to traverse deep trees per prediction. This delay may be unacceptable in latency-sensitive operations.

Table 4 Test Set Performance

Model	MSE	RMSE	MAE	R2
BaggingRegressor	0.001	0.029	0.020	0.590
DecisionTree	0.001	0.033	0.023	0.476
DNN	0.001	0.032	0.022	0.440
ExtraTreeRegressor	0.001	0.028	0.019	0.607
Gradient Boosting	0.001	0.035	0.025	0.411
HuberRegressor	0.002	0.045	0.032	0.008
LinearRegression	0.002	0.044	0.033	0.051
LSTM	0.001	0.037	0.027	0.262
PoissonRegressor	0.002	0.045	0.034	0.002

RandomForest	0.001	0.031	0.022	0.519
Ridge	0.002	0.044	0.033	0.049
SGDRegressor	0.002	0.044	0.033	0.042

The RandomForest model also yields strong results, with RMSE = 0.031, MAE = 0.022, and R^2 = 0.519, representing a good trade-off between predictive power and model complexity. In contrast, the DecisionTree regressor. Deep learning models show mixed performance: the DNN achieves moderate error rates (RMSE = 0.032, MAE = 0.022, R^2 = 0.440), while the LSTM underperforms significantly (R^2 = 0.262, MAE = 0.027), which may indicate insufficient sequence dependencies in the dataset or suboptimal hyperparameter settings.

Linear and generalized linear models, including LinearRegression, Ridge, SGDRegressor, PoissonRegressor, and HuberRegressor, consistently yield the worst results. This highlights their inability to capture non-linearities present in the data and further reinforces the suitability of non-linear ensemble methods for this task.

To evaluate the contribution of individual features, Shapley Additive Explanations (SHAP) analysis was conducted. The results are visualized in Fig. 4.1, providing insights into the relative importance and impact of each feature on the model's predictions. Among all features, TIME_HOUR shows the highest influence, followed by TIME_MINUTE, DAY_OF_YEAR, and TIME_OF_DAY, indicating that fine-grained temporal features play a crucial role in the prediction task. In contrast, variables such as IS_HOLIDAY, QUARTER, and TIME_DAY exhibit minimal contribution, as reflected by their narrow SHAP value distributions near zero.

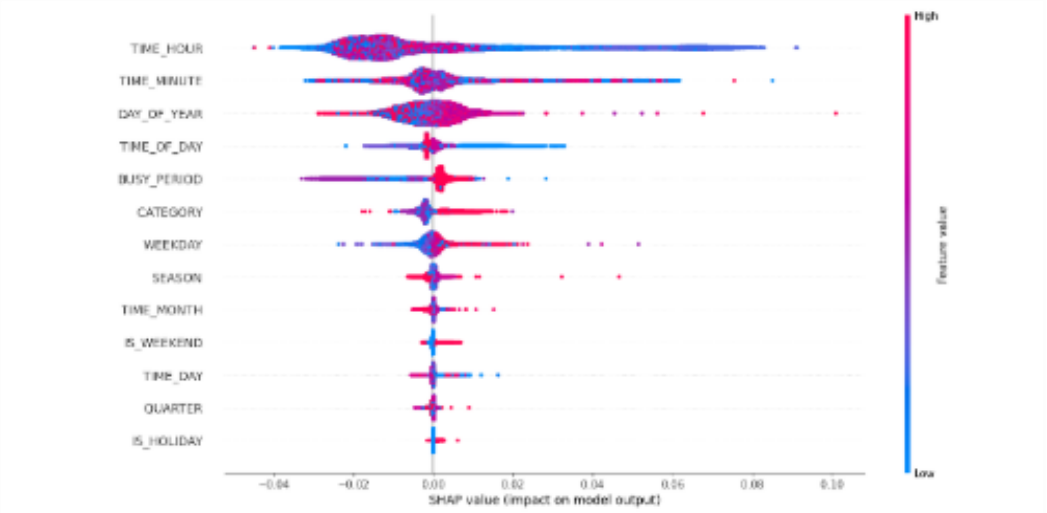


Figure 5. SHAP Analysis for RandomForest Model

4.2.5. Flight Delay Prediction: XGBoost, LightGBM (Best-Performing Models)

To develop an effective and reliable framework for predicting flight delays, this study incorporates a range of AI models drawn from two key categories: machine learning (XGBoost, LightGBM, Gradient Boosting AdaBoost, Random Forest) and deep learning (Deep Neural Network - DNN). Each of these approaches offers unique strengths in capturing the complex relationships between input features and delay outcomes, and their complementary use contributes to the robustness and generalization of the proposed predictive system.

The model's robustness was ensured through k-fold cross-validation (k=5) with random splitting. Hyperparameter optimization was conducted using a systematic grid search mechanism with MAE as the scoring metric. The hyperparameter optimization strategy balanced performance requirements with computational efficiency. Rather than exhaustively searching vast parameter spaces, we leveraged domain knowledge and preliminary experiments to focus on promising regions of the hyperparameter space. Optimized hyperparameters regarding AI models are summarized in Table 5.

Table 5. Hyperparameter Optimization

Model	Selected Parameters
XGBoost	N_estimators: 1000 Learning_rate: 0.1 Max_depth: 13
LightGBM	N_estimators: 1500 Learning_rate: 0.1 Max_depth: 20
Random Forest	N_estimators: 500 Min_sample_split: 2 Max_depth: 15
Gradient Boosting	N_estimators: 300 Learning_rate: 0.1 Max_depth: 5
AdaBoost	N_estimators: 100 Learning_rate: 0.1
DNN	Hidden Layers: 3 Neurons: (128,64,32) Activation: ReLU Dropout Rate: 0.2

Overall performance comparison is shown in Table 6. We would like to separate results with and without Dissimilarity Ratio where it is a cross-feature providing insight from baggage dataset. XGBoost emerged as the superior performer among all tested models, demonstrating exceptional predictive capabilities with the highest absolute performance metrics especially via minimum Test MAE score as 7.6 minutes. LightGBM demonstrated the most balanced and robust performance profile among all tested models. The relatively poor performance of the DNN (R^2 Score: 0.2742) can be attributed to several factors such as feature properties and representations. The conventional tree-based models, including Random Forest and Gradient Boosting, demonstrated consistent but moderate performance improvements with the integration of the Dissimilarity Ratio, achieving R^2 scores of 0.4461 and 0.4434 respectively.

Table 6. Performance comparison of models

Model	Metric	With Dissimilarity	Without Dissimilarity	Improvement
XGBoost	Train MAE	**4.2320**	4.8787	+13.26%
	Test MAE	**7.6220**	7.8828	+3.31%
	Train MSE	33.1772	44.2155	**+24.97%**
	Test MSE	112.1498	119.7797	+6.37%
	Train RMSE	5.7600	6.6495	+13.38%
	Test RMSE	10.5901	10.9444	+3.24%
	Train R^2	0.8635	0.8251	+4.65%
	Test R^2	0.5375	0.5258	+2.22%
LightGBM	Train MAE	6.4034	6.8377	+6.35%
	Test MAE	7.8514	8.1218	+3.33%
	Train MSE	76.2445	87.2403	+12.60%
	Test MSE	118.188	126.6679	+6.69%
	Train RMSE	8.7318	9.3403	+6.51%

	Test RMSE	10.8714	11.2547	+3.41%
	Train R^2	0.6862	0.6550	+4.76%
	Test R^2	0.5126	0.4985	+2.83%
Random Forest	Train MAE	8.0472	8.3596	+3.74%
	Test MAE	8.4194	8.6475	+2.64%
	Train MSE	121.186	131.2059	+7.64%
	Test MSE	134.2991	142.223	+5.57%
	Train RMSE	11.0085	11.4545	+3.89%
	Test RMSE	11.5887	11.9257	+2.83%
	Train R^2	0.5013	0.4811	+4.20%
	Test R^2	0.4461	0.4370	+2.08%
Gradient Boosting	Train MAE	8.3291	8.5411	+2.48%
	Test MAE	8.4234	8.6562	+2.69%
	Train MSE	132.308	138.946	+4.78%
	Test MSE	134.9552	143.0988	+5.69%
	Train RMSE	11.5025	11.7875	+2.42%
	Test RMSE	11.617	11.9624	+2.89%
	Train R^2	0.4555	0.4505	+1.11%
	Test R^2	0.4434	0.4335	+2.28%
	Train MAE	10.1921	10.4827	+2.77%
	Test MAE	10.1882	10.4230	+2.25%

AdaBoost	Train MSE	167.3839	175.697	+4.73%
	Test MSE	166.121	174.3044	+4.70%
	Train RMSE	12.9377	13.2551	+2.39%
	Test RMSE	12.8888	13.2024	+2.38%
	Train R^2	0.3112	0.3051	+2.00%
	Test R^2	0.3149	0.3100	+1.58%
DNN	Train MAE	9.4300	9.6000	+1.77%
	Test MAE	10.3100	10.5000	+1.81%
	Train MSE	223.1600	234.0900	+4.67%
	Test MSE	258.1100	263.2000	+1.93%
	Train RMSE	14.9400	15.3000	+2.35%
	Test RMSE	16.0700	16.2200	+0.92%
	Train R^2	0.3590	0.3658	-1.86%
	Test R^2	0.2742	0.2718	+0.88%

To enhance the interpretability of our predictive models and identify key delay drivers, we employed SHAP analysis (for top 20 features) on the best-performing XGBoost model for departure delay prediction as shown in Figure 6. SHAP values provide a unified framework for understanding feature contributions to individual predictions and overall model behavior.

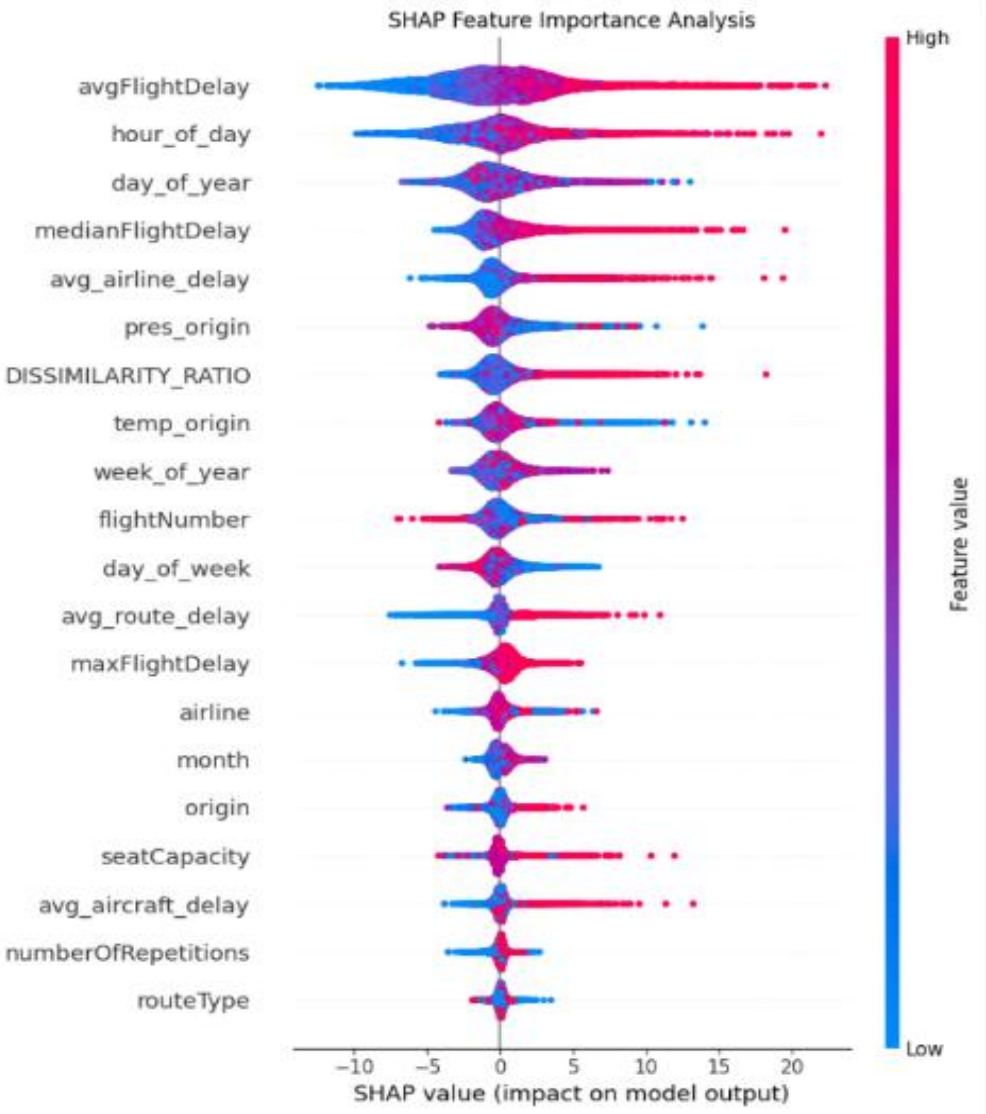


Figure 6. SHAP feature importance analysis

The SHAP analysis reveals several critical insights into the factors driving departure flight delays. Some of the major features are described within the following bullet points.

- avgFlightDelay: Historical average flight delays emerge as the strongest predictor. This indicates that flights with historically poor punctuality performance are significantly more likely to experience delays.

- hour_of_day: Time of day shows substantial impact, with peak hours (morning and evening rush periods) contributing positively to delay predictions, while off-peak hours show negative contributions.
- day_of_year: Seasonal patterns demonstrate clear influence, with certain periods (likely holiday seasons and summer months) showing higher delay propensity.
- DISSIMILARITY_RATIO: Novel introduced feature ranks in 7th place, demonstrating significant predictive power. Higher dissimilarity ratios (indicating deviation from normal baggage processing patterns) consistently contribute to increased delay predictions.
- medianFlightDelay and avg_airline_delay: Airline-specific performance metrics show strong predictive capability, highlighting the importance of carrier operational efficiency.
- pres_origin and temp_origin: Weather conditions at origin airports contribute moderately to delay predictions, with extreme values in either direction increasing delay likelihood.

4.3. Operational Optimization & Situational Awareness

Operational optimization and situational awareness constitute one of the core functional domains of the SOCFAI Platform. These capabilities enable airports to improve operational resilience by anticipating disruptions, optimizing the allocation of scarce resources, and reducing the overall operational overhead associated with real-time decision making. Airports operate in highly dynamic environments characterized by fluctuating passenger demand, congestion, irregular operations, and weather-induced uncertainties. To address these challenges, SOCFAI integrates machine learning techniques, rule-based intelligence, and mathematical optimization to support data-driven decision making within Airport Operations Control Centers (AOCC).

The platform supports both predictive and prescriptive operational intelligence. Predictive intelligence anticipates events such as delays, conflicts, or bottlenecks, while prescriptive intelligence recommends optimal courses of action considering operational constraints. These components are deployed as modular microservices, ensuring scalability and interoperability with existing airport systems such as AODB, RMS, and ADS-B sources. This modular service architecture not only enables ease of deployment but also supports the integration of additional airport-specific services in subsequent development phases.

4.3.1. Resource Management System (RMS) Optimization: Optimization Engine Performance Analysis (MIPGap, Assignment Rate)

The Resource Management System (RMS) provides an automated, data-driven resource allocation framework for operational areas such as gates, stands, and counters. Traditional airport allocation systems typically rely on fixed rules and schedule-based planning, which are limited in their capacity to respond to real-time operational disruptions. SOCFAI enhances this process by integrating predictive analytics and optimization to proactively address such uncertainties.

Predictive delay models trained using historical schedules, METAR weather data, and ADS-B telemetry provide probabilistic estimates of arrival and departure deviations. These predictions are incorporated directly into the optimization engine to improve allocation decisions. The optimization engine itself utilizes Mixed-Integer Programming (MIP) to compute optimal allocations under both hard and soft constraints, including compatibility, turnaround requirements, operational efficiency, and safety considerations.

Performance of the RMS optimization engine is evaluated using industry-standard Key Performance Indicators. The most relevant metrics include:

- **MIPGap**, which quantifies the proximity of a computed solution to the global optimum. Lower values indicate more accurate and efficient optimization.
- **Assignment Rate**, which measures the proportion of successful resource assignments relative to total assignment requests.
- **Conflict Resolution Time**, indicating the responsiveness of the system in irregular operational conditions.

In addition to MIP optimization, reinforcement learning (RL) techniques are incorporated to improve adaptability during high-uncertainty operational scenarios such as diversions or ground handling delays. The RL agent is exposed to simulated operational environments and learns optimal reallocation policies through repeated interaction. RMS results are disseminated through the SOCFAI eventing pipeline using Kafka streams and made available for visualization and operator intervention.

4.3.2. Rule-Based System: Real-time Flight Status Detection, Data Acquisition Strategy for Cost Optimization

Real-time flight status detection plays a central role in situational awareness within the SOCFAI Platform. The system combines machine learning classifiers and rule-based logic to determine

operational states such as On Approach, Holding, Landed, and Delayed. The classification models utilize operational telemetry and aviation data sources, including:

- ADS-B surveillance data,
- AODB and flight schedule feeds,
- contextual operational parameters such as congestion.

These classification results serve as input to downstream operational systems, including ETA prediction, delay risk assessment, and resource planning.

A secondary challenge associated with continuous flight monitoring is the cost of high-frequency data acquisition. To reduce operational and computational overhead, SOCFAI implements a Bayesian optimization-based sampling model. Rather than ingesting telemetry at a fixed rate, the model adjusts the sampling interval dynamically based on operational conditions. This helps reduce the data acquisition cost without compromising situational awareness or prediction accuracy.

Rule-based logic complements the learning-based models by providing deterministic reasoning for safety-critical scenarios. Rules are triggered in situations such as:

- unexpected deviations from flight trajectory,
- telemetry signal loss or degradation,
- sudden schedule conflicts or cancellation events.

This hybrid architecture ensures that operational reliability and interpretability are preserved even when AI-based modules operate under uncertainty. Outputs produced by the real-time flight monitoring pipeline are integrated into SOCFAI visualization components and decision-support modules, enabling early detection of irregular operations and proactive mitigation actions.

4.4. Passenger Experience and Sentiment Analysis

4.4.1. AI Model: BERT-based Sentiment Analysis Model, RoBERTa-based Language Detection

BERT-based Sentiment Analysis Model

The core AI component of the system is a custom DistilBERT sentiment analysis model which was fine-tuned on airport review. DistilBERT is a BERT-based model which is 40% smaller and 60% faster while

maintaining 97% of BERT's performance. Sentiment analysis is one of the applications for which this model is highly effective.

Key characteristics of the DistilBERT model include:

- Knowledge distillation

  A smaller "student" model learns from the "teacher" model by mimicking the output probabilities of the teacher and learning from the "dark knowledge" or uncertainty present in the teacher's output instead of relying solely on hard targets.

- Tokenization

  Each text input is broken down into sub-words which enables the model to represent unfamiliar words in addition to more common words.

- Positional embeddings

  The position of each token is represented, and this provides information about the order of each word in the input sequence.

- Encoder

  Extracts information about the dependencies between various words and captures subtleties in natural language within the text.

- Contextual embeddings

  Contain information about the context for each token and are used for classification after the encoder layers.

The output of DistilBERT for the text classification task is a logits vector containing a score for each of the classes (i.e., sentiments). The model's prediction is the class (or sentiment) with the highest score.

RoBERTa-based Language Detection

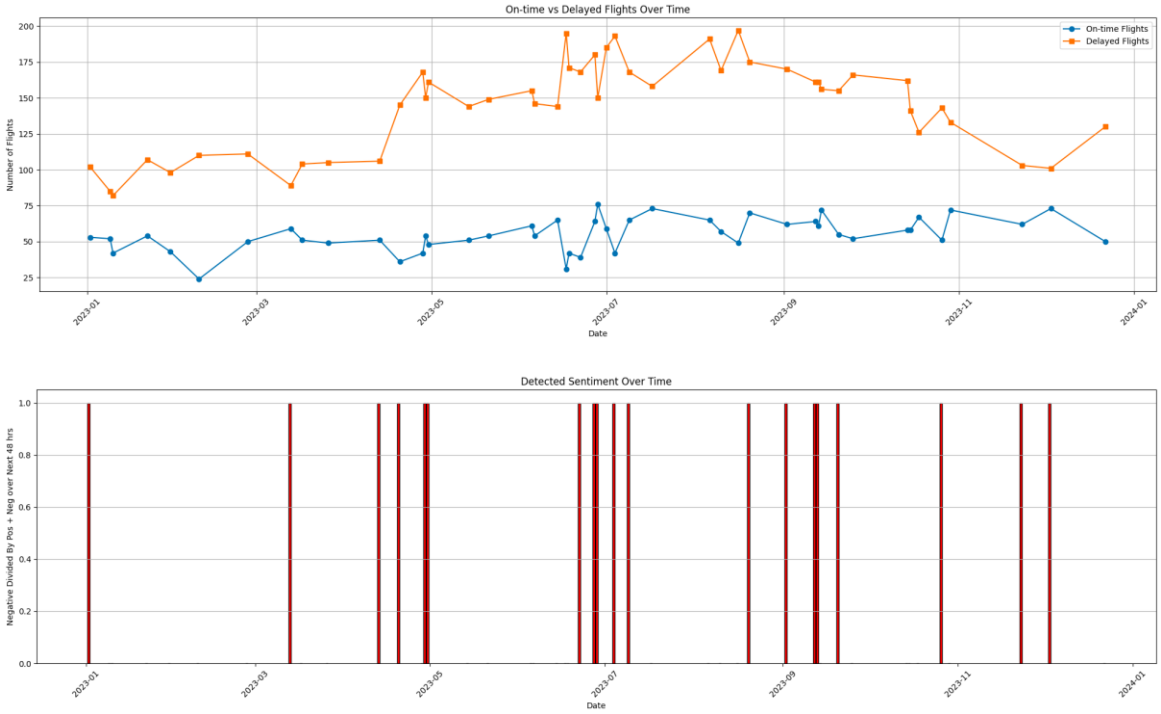
Another AI model leveraged by the system is an open source RobERTa-based language detection model. The model was trained on 20 languages and the average accuracy for detecting languages on the test set is 99.6%. Meanwhile, the average accuracy for detecting languages on the test set when using the Python langid library was 98.5%.

Similar to BERT-based models, the output for the RoBERTa model is also a logits vector.

4.4.2. Service: Time-based Sentiment/Language Visualization, Sentiment Change Analysis after Delays

a) Time-based Sentiment/Language Visualization

The time-based graph showing the number of delays per unit of time (e.g. day) is compared to the graph showing the sentiment change.



Because the use case focuses on negative comments that are often associated with flight delays, this is determined by the rate of the increase of negative comments. The increase rate of negative comments is determined by an expression that is dependent on the number of negative comments and the number of positive comments on a given day (the day for which the increase rate value is plotted) and the day after that.

b) Sentiment Change Analysis after Delays

The rate of increase of negative comments is given by a value derived from the result of a mathematical expression containing the variables n (negative), p (positive), and d (day).

The expression represents the number of negative comments within a span of two days (48 hours) divided by the sum of the number of negative comments and positive comments within the same time period. The variable n represents the number of negative comments on a given day and p represents the number of positive comments on a given day. The variable d represents the day. Thus, the subscripts indicate whether the number of negative or positive comments is for the given day (d) or the day after (d+1).

$$\frac{n_d + n_{d+1}}{n_d + n_{d+1} + p_d + p_{d+1}}$$

5. Airport Specialized Data Utilization and Functions

To support real-time indoor air-quality assessment at İzmir Adnan Menderes Airport, Milesight AM308L environmental sensors were deployed at multiple terminal locations. These devices continuously measure fine particulate matter (PM2.5, PM10), volatile organic compounds (TVOC), carbon dioxide (CO_2), temperature, humidity, atmospheric pressure, light level, and occupancy-related PIR activity.

Each sensor publishes its measurements into the SOCFAI data pipeline, where raw telemetry is forwarded to an **Amazon SQS queue**. A Python-based ingestion service retrieves sensor messages via the **boto3** library, parses the JSON payload, and stores only the relevant ADB data into a temporary **PostgreSQL** database. This preprocessing step filters out unrelated metadata (device brand, model, external airport sensors) and ensures that downstream components receive a uniform and lightweight dataset.

Once stored in PostgreSQL, the data becomes available for both real-time AQI computation and time-series visualization.

The **AQI (Air Quality Index)** is a standardized numerical representation of air pollutant concentration levels. It converts physical pollutant measurements into a normalized scale (typically **0–500**) indicating the potential impact on human health. The AQI score is computed individually for each pollutant and the **worst pollutant defines the final AQI**:

$$AQI = \max\left(AQI_{PM2.5}, AQI_{PM10}, AQI_{CO2}, AQI_{TVOC}\right)$$

Although AQI traditionally focuses on outdoor pollutants, an adapted indoor version is used for airports, emphasizing:

- **PM2.5**
- **PM10**
- **CO_2 concentration** (proxy for ventilation and occupancy load)
- **TVOC** (indoor chemical pollutant level)

Milesight AM308L sensors directly provide raw PM2.5, PM10, CO_2, and TVOC values, which are transformed into AQI scores using pollutant-specific breakpoints.

The Air Quality Index used in this project follows the **European Air Quality Index (EAQI)** defined by the **European Environment Agency (EEA)**. EAQI categorizes air quality into **five classes** based on pollutant concentrations, providing an intuitive color-coded representation of air-health impacts.

AQI is calculated using the U.S. EPA breakpoint formula, widely adopted for indoor AQI systems as well:

$$AQI_p = \frac{(I_{\text{high}} - I_{\text{low}})}{(C_{\text{high}} - C_{\text{low}})} \times (C_p - C_{\text{low}}) + I_{\text{low}}$$

where:

C_p = measured pollutant concentration
C_{low} , C_{high} = breakpoint concentration range for that pollutant
I_{low}, I_{high} = AQI scale corresponding to breakpoint interval
AQI_p = AQI value for pollutant *p*

European AQI does **not** use a per-pollutant formula like the U.S. EPA system; instead, it assigns pollutant values into fixed concentration ranges that map directly to **air quality classes**.

Table 7 European AQI Categories and Breakpoints for PM2.5 & PM10

EAQI Class	Air Quality	PM2.5 Range (µg/m³)	PM10 Range (µg/m³)
1	Good	0 – 10	0 – 20
2	Fair	10 – 20	20 – 35
3	Moderate	20 – 25	35 – 50
4	Poor	25 – 50	50 – 100
5	Very Poor	50 – 800	100 – 1200

Since CO_2 and TVOC strongly influence indoor comfort and ventilation quality, SOCFAI extends EAQI using widely accepted indoor air hygiene ranges:

EAQI Class	CO_2 Level (ppm)	Interpretation
1	400 – 800	Good ventilation
2	800 – 1000	Acceptable
3	1000 – 1500	Moderate ventilation load
4	1500 – 2000	Poor ventilation
5	> 2000	Very poor / ventilation failure

EAQI Class	TVOC Level (ppb)	Interpretation
1	0 – 220	Good indoor air
2	220 – 660	Mild pollution
3	660 – 2200	Noticeable contamination
4	2200 – 5500	Poor indoor air
5	> 5500	Very poor; unsafe buildup

This ensures consistency with European regulatory standards while providing additional insight into **indoor air quality**, which is essential in enclosed airport terminal spaces.

Real-time AQI Computation and Delivery

The air-quality monitoring pipeline operates on a scheduled retrieval mechanism rather than event-based ingestion. A background service polls the **Amazon SQS queue at fixed intervals (typically every 1–5 minutes)** using the boto3 library. During each cycle, all available sensor messages are fetched, parsed, and written into the PostgreSQL database. This periodic batching approach reduces API load, ensures stable processing, and provides a consistent update rhythm suitable for terminal-level environmental monitoring.

After each polling cycle, the AQI computation service processes the newly added records. For every sensor reading, the system determines the EAQI class for PM2.5, PM10, CO_2, and TVOC, then computes the final AQI based on the highest pollutant category. The resulting AQI values, along with pollutant contributions and sensor metadata, are stored in dedicated AQI tables.

Grafana dashboards query PostgreSQL at short refresh intervals to visualize real-time and historical air-quality trends. The platform displays per-location AQI values, pollutant-specific charts, anomaly flags, and threshold exceedances, enabling airport operations teams to monitor indoor environmental conditions continuously and react promptly to ventilation or occupancy-related issues.

5.1 Prediction Dashboard and User Interface (UI)

5.1.1. Real-time Statistics Page and Prediction Page Configuration

The SOCFAI Platform provides two complementary interface modules for operational intelligence: the Real-time Statistics Page and the Prediction Page. The Real-time Statistics Page is designed for continuous situational monitoring, whereas the Prediction Page supports proactive decision-making through AI-based forecasting and prescriptive analytics. Together, they enable operators to interpret current operational conditions, forecast upcoming disruptions, and derive optimal interventions.

Real-time Statistics Page

The Real-time Statistics Page consolidates live operational data streams and presents them through a unified visualization layer. It integrates information from multiple operational systems, including AODB and schedule data, aircraft telemetry from ADS-B, and contextual data from IoT and environmental sensors. Its primary purpose is to provide a consolidated operational view across key airport performance dimensions such as flights, resources, and environmental conditions.

The Real-time Statistics Page supports several core visualization functions:

- **Flight Status Dashboard**, which displays current flight positions, estimated arrival and departure times, and disruption alerts using ADS-B and AODB data.
- **Resource Utilization Metrics**, presenting gate occupancy, stand assignments, and counter availability based on resource allocation outcomes from the RMS module.
- **Dynamic KPI Visualizations**, where operational indicators such as turnaround time, assignment rate, and congestion metrics are updated continuously as new information becomes available.

From a technical standpoint, the Real-time Statistics Page relies on a low-latency data pipeline developed on a stream-processing architecture using Kafka. This architecture supports high-frequency data ingestion and processing, enabling real-time synchronization between SOCFAI services and visualization components. The user interface is deployed as a modular dashboard framework that allows AOCC operators to configure and customize widgets and display elements for specific operational needs. In addition, the alerting subsystem enables configurable thresholds for anomalies such as resource conflicts or elevated passenger density, automatically generating notifications and operational prompts.

Prediction Page

The Prediction Page complements the Real-time Statistics Page by providing forecasting and prescriptive decision support. The platform leverages AI models developed within the SOCFAI framework to estimate likely operational outcomes for upcoming time periods. The prediction capabilities are based on multiple model families, including delay prediction, dynamic resource allocation models, and passenger flow forecasting.

The core predictive services include:

- **Flight Delay Prediction**, where Gradient Boosted Decision Trees estimate delay probabilities based on historical schedules, ADS-B telemetry, and weather data.
- **Resource Allocation Forecasting**, which uses reinforcement learning agents and RMS outputs to simulate future allocation scenarios under changing demand conditions.
- **Passenger Flow Forecasting**, where time-series analysis and sensor-based occupancy trends are used to anticipate terminal congestion and identify potential bottlenecks.

The Prediction Page also incorporates a simulation engine that allows users to evaluate hypothetical scenarios and operational contingencies. Operators may simulate conditions such as sudden weather deterioration, peak-hour demand surges, or flight diversions to assess the impact on downstream operational processes. Each predictive output is accompanied by

confidence metrics to support risk-based decision-making and provide transparency regarding model reliability.

Prediction outputs are integrated directly into SOCFAI's optimization services and rule-based components, enabling automated transition from forecasting to operational recommendations. In this way, the predictive layer feeds into resource management and anomaly detection workflows, supporting both proactive and reactive operational control.

Benefits

The dual-interface architecture provides operational benefits across several dimensions:

- **Unified Operational View**, combining real-time monitoring and predictive analytics within a single operational environment.
- **Data-Driven Decision Support**, enabling AOCC teams to anticipate operational disruptions and execute mitigation measures efficiently.
- **Scalability and Extensibility**, supporting integration of additional data sources, sensors, and predictive models as airport operational needs evolve.

The combination of real-time situational monitoring and predictive forecasting therefore enables SOCFAI to support decision-making under uncertainty and enhance the overall resilience of airport operations.

6. Port Data Sources and Data Governance

The import/export port logistics platform manages container cargo activities such as inbound, outbound, transfer, storage, inventory control, container location tracking, dispatch requests, and vehicle movement data. It also handles transportation requests and transport management for import/export containers moving between port terminals and CY/CFS logistics warehouses within the port hinterland, including vehicle and driver dispatch information.

The platform integrates a wide range of logistics data from external entities such as Customs, port terminals, shipping lines, and the National Tax Service, with the goal of enabling real-time, data-driven operational management.

6.1 Key Data Sources

6.1.1 Port Logistics Data (CFSMS Main System, TMS Main System)

6.1.1.1 CFMS Main System(CFS/CY)

(1) IN, OUT information between port terminals of import and export containers:

The platform collects and manages data related to the movement of container cargo between the import and export port terminals and the CFS/CY warehouse within the hinterland. This includes cargo information, expected date of entry and exit, vessel arrival/exit schedule, and major vessel and vessel information. For imported containers, detailed tracking is carried out based on DO_NO (Handover Order) and BL_NO (Lading Bill) to understand the history of entry and exit, operational status, and progress of cargo handling. Export containers are managed in the same way using the reservation number (BK_NO). In addition, the essential properties of each container such as container number, size, seal number, ISO code, delivery vessel, terminal information, and planning schedule are maintained to increase traceability and operational efficiency throughout the port logistics.

(2) Import and Export Container Operation Plan Information:

Based on the expected gate-in/gate-out date for each container, the work schedule, gate-in schedule, and ship departure schedule are planned. For imported containers, we integrate bonded warehouse information and customs clearance status and also manage work status, cargo weight, cargo specification details, HS code, IMDG code, UNNO code. It also integrates gate-in/gateout reporting data, work schedule information, container specifications, seal numbers, and cargo descriptions.

(3) Filling and Unfilling Job Data:

The system collects and manages schedule and status information related to filling operations (loading cargoes into containers) and non-charging operations (removing cargoes from containers). Related logistics data such as transport requests, loading and gateout information, unloading and gate-in information are recorded according to the operating schedule. The warehouse location tracking system is used to monitor and manage the yard movement history and container movement history at the level of BLOK, BAY and ROW in the CFS/CY warehouse.

(4) Inventory and bonded warehouse status information:

The platform manages detailed bonded warehouse information such as container inventory status, loading/loading results, transshipment activity records, container damage information, and abnormal conditions (survey data). This enables precise monitoring of container status and increased visibility into warehouse operations.

6.1.1.2 TMS Main System

(1) Transport Order & Transport Status Data:

Transport order data includes transport order numbers, order detail numbers, shipping line, shipper/consignor, transport type (inbound/outbound/round-trip), origin and destination, operation site (warehouse), container and cargo details, dispatch status, and transport status. This information is collected and managed in real time, with a standardized workflow that enables end-to-end monitoring from order creation to completion.

(2) Dispatch Request & Dispatch Status Data

Dispatch request data includes dispatch numbers, transport companies, drivers, and vehicle information required for transport execution.
Vehicle data consists of vehicle codes, plate numbers, transport company, tonnage, and equipment type.

Transport progress is managed through the stages of Dispatch Registration → Dispatch Request → Transport Start → Departure Report → Arrival Report → Completion Report. GPS-based location data collected via mobile devices enables real-time tracking of vehicle status and movement during transit.

(3) Integrated Logistics Data

Transport rate table integration based on distance, container size, and transport type Automated calculation of sales/purchase freight and settlement data upon transport completion. Integration with the National Tax Service for electronic tax invoice issuance/cancellation e-POD (Electronic Proof of Delivery) API integration with automated gate systems at port terminals

Terminal Gate IN/OUT event and status information API. Exception event data such as transport delays, pending/idle status, or container not released Automatic accumulation of logistics KPIs such as lead time, utilization rate, and turn-around rate.

(4) Customer, Shipper, Carrier & Partner Data

Master data is managed for customers, shippers, shipping lines, terminals, transport companies, forwarders, warehouses (including bonded warehouses), vehicles, and customer-specific settlement rules.

Through EDI integration with shipping lines and terminals, the system collects and updates shipping line information, terminal information, B/L data, inbound/outbound schedules, shipper details, and forwarder data.

(5) External Agencies & Service Integration Data Sources:

Customs: import declaration, export declaration, bonded transport declaration and inquiry APIs
National Tax Service: electronic tax invoice issuance/cancellation API
Shipping Lines / Terminals: COPINO EDI/API (container in/out), B/L data, schedules, yard plans
Partner Warehouses: transport request and dispatch linkage APIs
AI logistics platforms, monitoring solutions, and IoT sensors (temperature, humidity, shock, etc.) for extended integration Collection of transport exception events (delays, waiting times)
Expandability for further IoT, sensor-based, and AI-driven modules

6.1.1.3 CFMS Main System(WMS)

(1) IN, OUT information between port terminals of import and export containers

The platform collects and manages data related to the movement of container cargo between the import and export port terminals and the CFS/CY warehouse within the hinterland. This includes cargo information, expected date of entry and exit, vessel arrival/exit schedule, and major vessel and vessel information. For imported containers, detailed tracking is carried out based on DO_NO (Handover Order) and BL_NO (Lading Bill) to understand the history of entry and exit, operational status, and progress of cargo handling. Export containers are managed in the same way using the reservation number (BK_NO). In addition, the essential properties of each container such as container number, size, seal number, ISO code, delivery vessel, terminal information, and planning schedule are maintained to increase traceability and operational efficiency throughout the port logistics.

(2) Import and Export Container Operation Plan Information:

Based on the expected gate-in/gate-out date for each container, the work schedule, gate-in schedule, and ship departure schedule are planned. For imported containers, we integrate bonded warehouse information and customs clearance status and also manage work status, cargo weight, cargo specification details, HS code, IMDG code, UNNO code. It also integrates gate-in/gateout reporting data, work schedule information, container specifications, seal numbers, and cargo descriptions.

(3) Warehouse internal operations (WMS Operation data):

Collect and manage schedule and status information related to loading operations (loading cargo into container) and unloading operations (removing cargo from container). Related logistics data such as transport requests, loading and gateout information, unloading and gate-in information are recorded according to the operating schedule. Use the warehouse location tracking system to monitor and manage the yard movement history and container movement history at the level of BLOK, BAY and ROW in the CFS/CY warehouse.

(4) Inventory and bonded warehouse status information:

The platform manages detailed bonded warehouse information such as container inventory status, loading/loading results, transshipment activity records, container damage information,

and abnormal conditions (survey data). This enables precise monitoring of container status and increased visibility into warehouse operations.

6.1.1.4 AI based Container Freight Station Management System(CFSMS)

One of the most critical components of import/export port logistics data is the accurate location information of container cargo.

In CFS/CY warehouses within the port hinterland, container storage areas generally follow predefined zones and placement rules. However, achieving optimal placement and high operational efficiency requires extensive field know-how and consideration of numerous variables.

Therefore, the focus is on leveraging AI to analyze and apply these diverse operational variables and accumulated expertise, enabling smarter decision-making for optimal container placement and logistics operations.

(1) Inbound and Outbound Information for Import/Export Containers:

Data is managed for import and export containers moving between seaport terminals and CFS/CY logistics warehouses located in the port hinterland. This includes a wide range of information such as container-level cargo details, planned inbound and outbound dates, vessel arrival and departure schedules, and key carrier and vessel information.

In addition, the system collects and manages core operational data required for transportation, handling, and status tracking, including container gate-in/gate-out history, work progress status, cargo handling milestones, container number, size, seal number, ISO code, shipping line information, inbound/outbound terminal details, and terminal schedule forecasts.

(2) Work Planning for Import/Export Containers:

Work planning data is managed for import and export containers based on their scheduled gate-in and gate-out dates. This includes container work reservations, inbound schedules, bonded warehouse information for import cargo, and customs clearance status linked with the Korea Customs Service.

Additional data such as gate-out schedules, work status, cargo weight, cargo specifications, vessel departure dates, HS Code, IMDG Code, UN Number, import/export gate-in and gate-out declaration records, container work reservations, inbound/outbound schedules, container size, seal number, cargo description, and HS Code are also systematically managed.

(3) Gateout and Cargo Handling Operational Data:

Operational data related to gate-out and cargo handling is collected and managed for each container according to a planned work schedule. This includes transport requests sent to trucking companies, loading and gate-out information, unloading and gate-in information, and location tracking data for handling containers in CFS/CY warehouses.
Detailed operational records include loading cargo into containers, pulling cargo out of

containers, container relocation and relocation activities, and storage location data at BLOCK, BAY, ROW levels in yards.

(4) Inventory and bonded warehouse status information:

The system monitors the inventory status of each container, including loading, unloading, and transfer history, as well as the results of special notes or surveys.
This allows accurate supervision of warehouse inventory and bonded cargo status

6.1.1.5 Digital Twin based Transport Management System(TMS)

Key components of digital twin simulation-based port logistics transportation optimization system and simulation model, grid and simulation
The optimization module, port logistics container trailer operating system, aims to improve the total transport time by more than 10%.

(1) Transport Order and Transport Status Data:

Transport order data includes the transport order number, order line number, shipper/consignee, transport type (inbound/outbound/round-trip), origin and destination, work location (warehouse), container information, cargo details, dispatch status, and transport status.

(2) Dispatch Request and Dispatch Status Data

Dispatch request data consists of the dispatch ID, trucking company, driver information, and vehicle details. Vehicle information includes the vehicle code, license plate number, carrier company, vehicle tonnage, and equipment type. Transport progress is managed through the following lifecycle:

Dispatch Created → Dispatch Requested → Transport Started → Departure Reported → Arrival Reported → Transport Completed.

GPS-based mobile tracking is used to monitor the vehicle's real-time status and location during operations.

(3) Customer, Shipper, and Carrier Master Data:

Master data is managed for various logistics partners, including shippers, terminal operators, trucking companies, warehouses (including bonded warehouses), and vehicles. Additional key data includes terminal information, bills of lading (B/L), and gate-in/gate-out scheduling information.

6.1.2. Mobile/Field Equipment Data

6.1.2.1 AI based Container Freight Station Management System(CFSMS)

In mobile and tablet-based logistics operation environments, information is delivered concisely, focusing on the key data that must be checked and processed immediately on-site. This enables

rapid identification of container flows and work status, while also improving collaboration efficiency among shipping lines, terminals, and bonded warehouses.

(1) Import/Export Container Gate-In/Gate-Out Information:

On mobile screens, essential container details—such as container number, size, ISO code, and seal number—are clearly displayed along with the scheduled gate-in/gate-out dates and current status.

Major vessel information, including carrier details and vessel arrival/departure schedules, is also provided to support prompt decision-making in the field.

(2) Import/Export Container Work Planning Information:

Mobile devices allow intuitive access to operational schedules, such as gate-in/gate-out plans and container work reservation status.

(3) Loading and Unloading Work Data:

On-site stuffing and unstuffing operations can be recorded in real time using mobile devices, including work status, start/end times, and progress updates.
Transport requests, loading/unloading details, and gate-in/gate-out confirmations are updated instantly, while container positions can be viewed at the BLOCK, BAY, and ROW level, enabling structured management of yard and warehouse movements.

(4) Inventory and Bonded Warehouse Status Information

Mobile-based inventory screens provide a simple view of current container inventory and the results of various operations (loading, unloading, transfer, etc.).
Damage information and special inspection notes (Survey Data) can be quickly entered by field personnel, supported by photo capture features, enhancing accuracy and reliability in bonded warehouse container management.

6.1.2.2 DigitalTwin based Transport Dispatch Management System(TMS)

The mobile interface provides all necessary transport-related information in an intuitive format and allows users to track each stage of the process from order creation to completion.

(1) Transport Orders and Transport Status Information

Key information required for field operations is clearly presented, including: Transport order number, order detail number, Origin and destination, work site (warehouse), Container details (container number, size, seal number, etc.), Transport type (inbound/outbound/round-trip), Cargo information (item name, quantity, weight, etc.), Dispatch status and transport progress (real-time tracking of each stage)

(2) Dispatch Requests and Dispatch Status Information

A mobile dashboard provides real-time visibility into dispatch and transport progress. Managed data includes: Dispatch number, vehicle number, transport company, Driver information (name, contact details), Vehicle specifications (tonnage, equipment type, etc.), Dispatch request and approval status, Transport progress flow: *Dispatch Registered → Dispatch Requested → Transport Started → Departure Report → Arrival Report → Completed,* GPS-based vehicle tracking and movement monitoring

(3) On-Site Transport and Operation-Linked Data

Mobile devices support real-time input and confirmation of operational data, including: Terminal Gate IN/OUT timestamps and status, Electronic Proof of Delivery (e-POD) upload and verification, Delay alerts, gate waiting notifications, and other exception events

(4) Customer and Partner Master Data Inquiry

Customer and cargo owner profiles, Shipping line and terminal information (operating hours, gate locations, etc.), B/L information and container schedule data, Warehouse and bonded warehouse locations and instructions

6.2 Data Pipeline and Model

In ZooKeeper-based Kafka, metadata is managed between brokers and ZooKeeper, which can lead to metadata inconsistency issues. Additionally, the architecture introduces increased costs due to additional server systems, a higher number of Java processes, increased resource consumption, and greater operational and monitoring complexity for multiple daemons in failure scenarios. To overcome these limitations, the data pipeline adopted the latest Kafka version supporting KRaft, a modern architecture designed to replace ZooKeeper and provide more efficient metadata management.

In relation to the data pipeline and data model, Kafka with KRaft enhances the reliability and scalability of real-time data streaming by integrating metadata management within the Kafka cluster itself. This contributes to a more streamlined architecture for data pipelines, improves operational efficiency, and enables faster and more stable data synchronization across systems such as port logistics warehouse management (CFS/CY) and port logistics transportation management systems. The data model can now better support real-time event processing, schema evolution, and standardized data governance aligned with modern data engineering practices.

6.2.1. Kafka/Debezium-Based CFSMS/TMS Data Pipeline

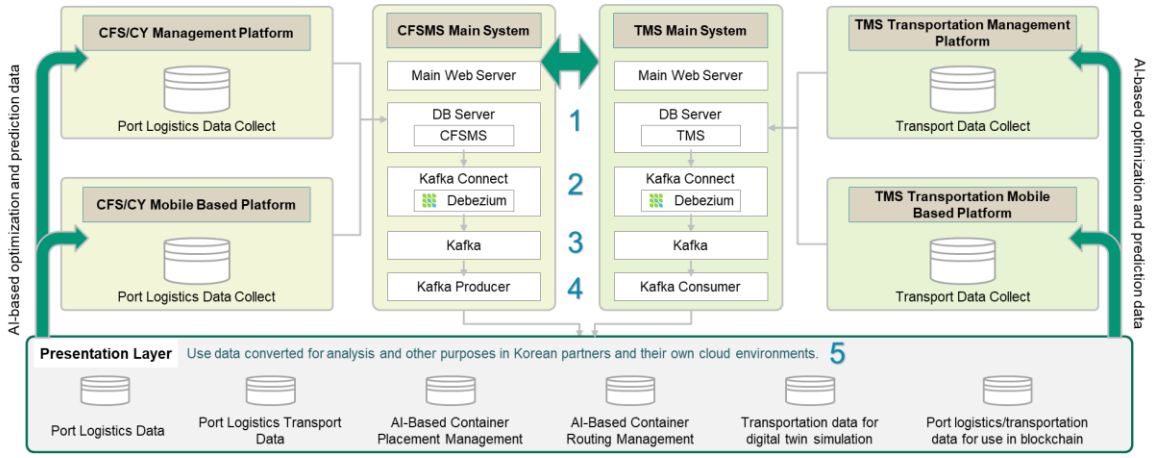


Figure 7 AI-Based services for data optimization, digital twin simulation-based services, and data interworking between transportation management systems and CFS/CY distribution warehouses

6.2.2. Data Standardization and DB Schema Design

Through the standardized data dictionary, unified message schema, and CDC-enabled database design, the SOCFAI-driven port logistics and transport management platforms established a highly interoperable and scalable data architecture. This foundation enables reliable real-time integration, facilitates AI analytics and digital twin simulations, and supports future expansion to government, customs, and external enterprise systems.

In the SOCFAI-based port logistics and transportation management platforms, data exchange among heterogeneous systems such as port terminals, CFS/CY warehouses, transportation companies, customs, and the National Tax Service is essential. To enable this, an integrated data management framework based on data standardization has been established. Terminology, codes, formats, and message structures have been standardized to ensure that all systems and platform users can correctly understand the meaning of data. In particular, terminology standardization—covering field names, business terms, and abbreviations—ensures consistent interpretation, while code standardization for status and location codes enables accurate determination and analysis of process states. In addition, format standardization for date/time, coordinates, and currency ensures an optimized structure for real-time transmission and traceability in the data pipeline, and Kafka JSON Schema-based message structure standardization supports system-wide structural compatibility and CDC integration.

Furthermore, a column definition registry has been established to manage the structure, definitions, data types, and sample values of actual data entered into the system, thereby enhancing the clarity and usability of data structures. This improves data comprehension across the platform and supports efficient system integration, data consistency validation, and quality management. Personal data standardization applies encryption and masking rules to strengthen

privacy protection and compliance with security policies, while data quality management—covering integrity, validity, and deduplication—ensures reliable data for AI analytics and digital twin simulation environments. The standardized data is streamed in real time through the Kafka-based data pipeline and is actively used for intelligent analytics and optimization in digital twin–based logistics operations.

The port logistics platform operates a collaborative data governance organization composed of business domain managers, the IT operations team, and settlement/accounting personnel to maintain and manage data standards.

This organization regularly reviews the standardized data column definition documents and carries out the approval and management procedures whenever new fields are added or message schemas are modified. The platform's database schema is designed based on an entity-driven business domain model, separating normalized transaction tables from analytical summary tables (OLAP).

In addition, to support Kafka–Debezium–based CDC integration, all tables include PK, Updated_Timestamp, and Event_Type fields, enabling real-time data tracking.

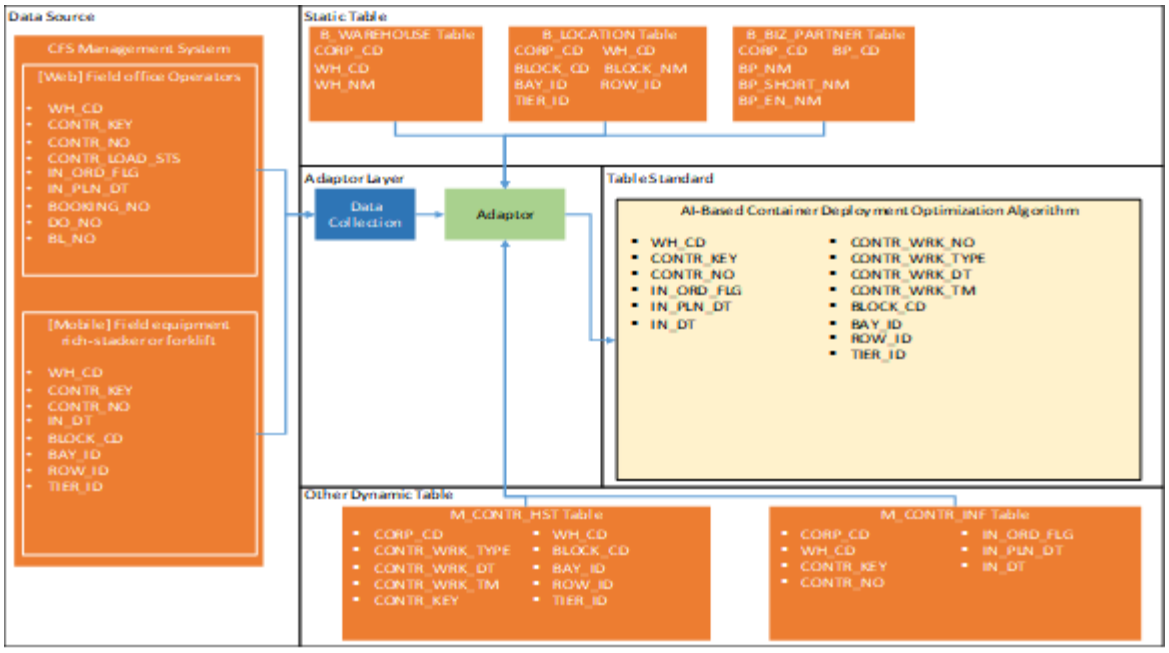


Figure 8. Port logistics data collection and processing

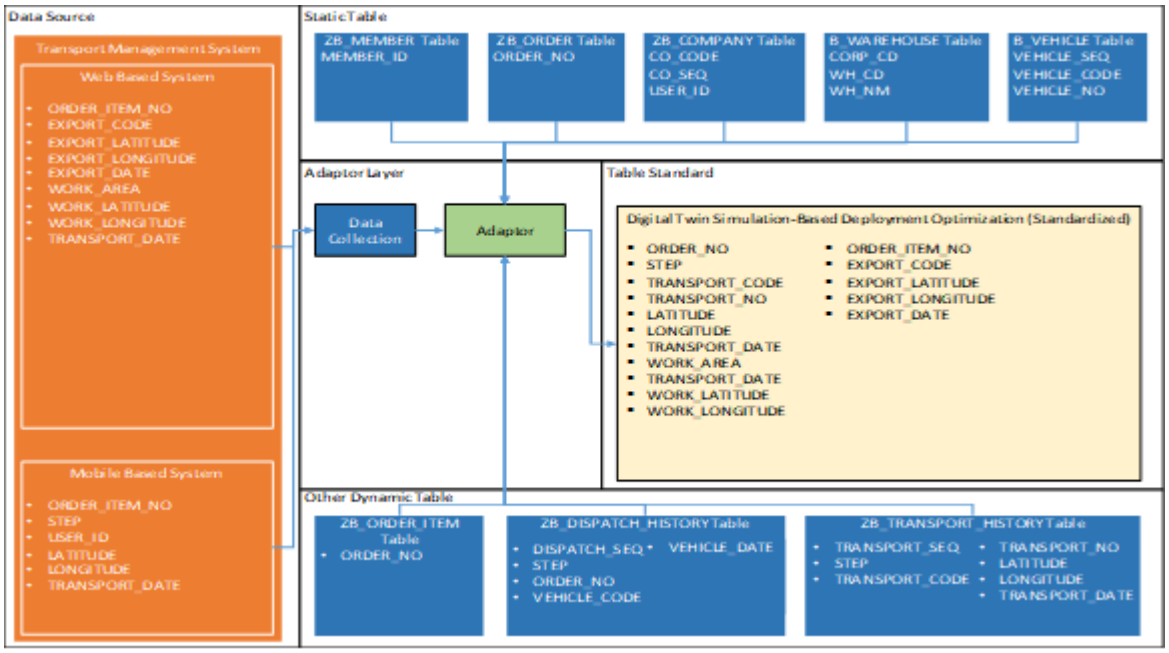


Figure 9. Port logistics transportation data collection and processing

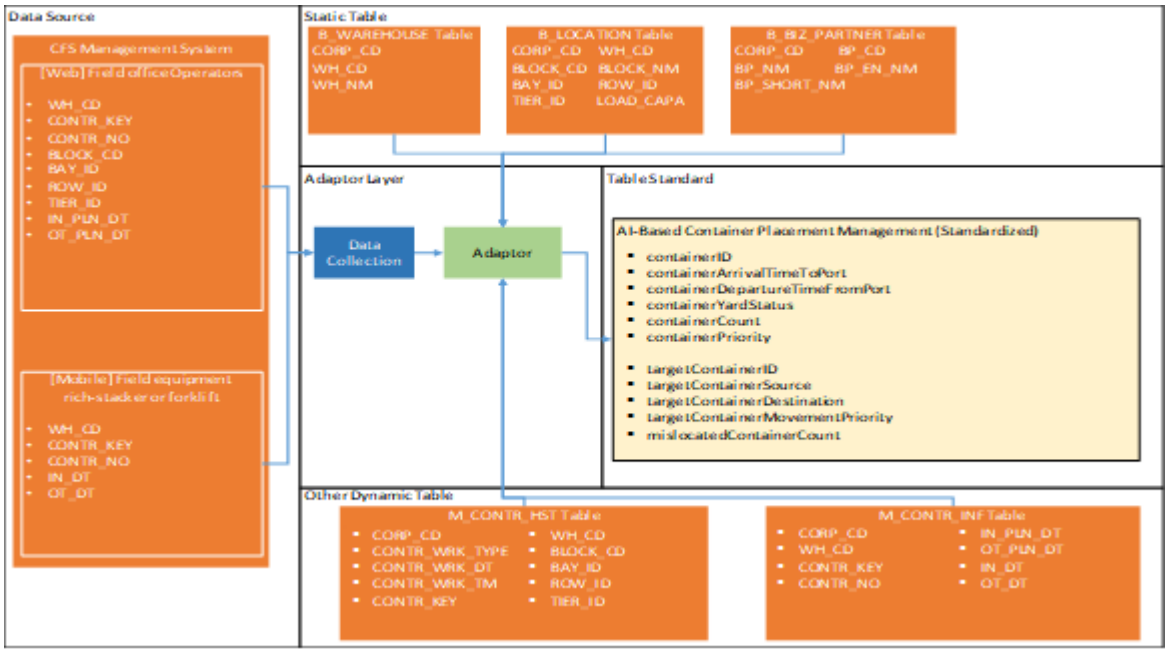


Figure 10. AI-powered container deployment optimization

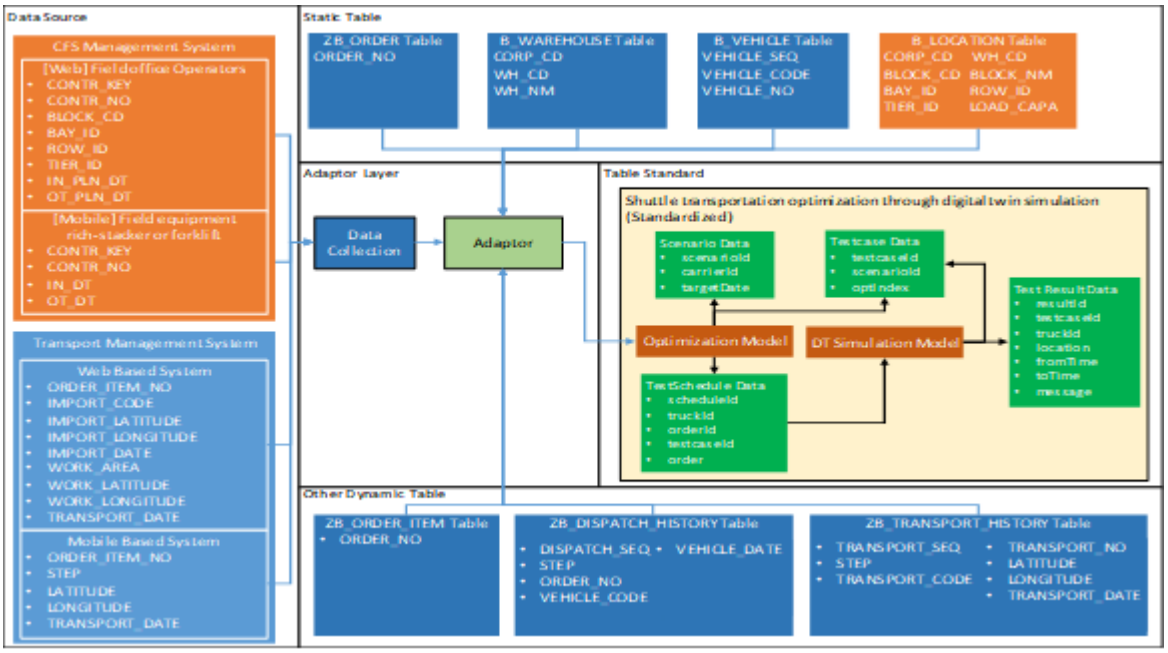


Figure 11. Shuttle transport optimization data with digital twin simulation

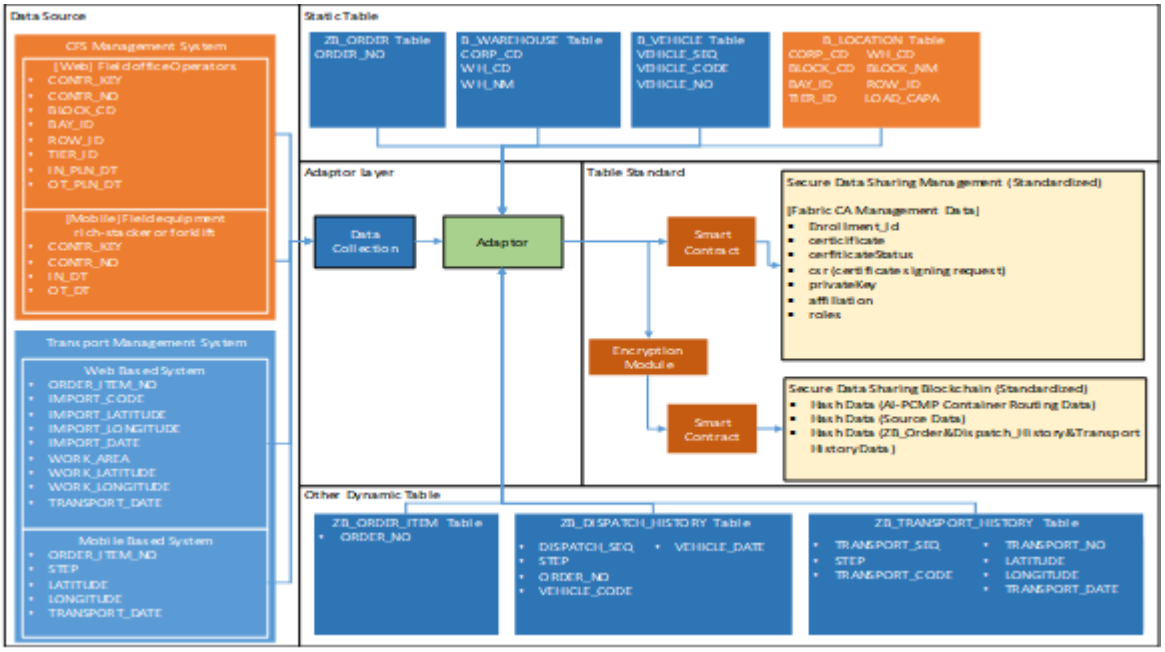


Figure 12. Secure Data Sharing using Private Blockchain for Logistics Optimization Data

Column Definition (컬럼 정의서)

no	DATABASE	TABLE	Field	Field (Initials Case)	DESCRIPTION (ENG)	DESCRIPTION (KOR)	DATA TYPE	DATA SIZE	DATA FORMAT	Not Null Y/N	PK Y/N	FK Y/N	AK Y/N	Constraints terms	USE Y/N	PERSONAL Y/N	ENCRYPT Y/N	OPEN/CLOSE Y/N
1	CFSMS	M_CONTR_INF	CORP_CD	CORP_CD	Company classification code	업체 구분코드	varchar	10	KULSINC	Y	Y							
2	CFSMS	M_CONTR_INF	WH_CD	WH_CD	Warehouse code	창고코드	varchar	10	KNC10									
3	CFSMS	M_CONTR_INF	CONTR_KEY	CONTR_KEY	Classification number	구분번호	int	4	[4073]	Y	Y							
4	CFSMS	M_CONTR_INF	STOCK_FLG	STOCK_FLG	Inventory status	재고상태	varchar	1	Y									
5	CFSMS	M_CONTR_INF	CONTR_NO	CONTR_NO	Container Number	컨테이너 번호	varchar	20	MRSU0219470									
6	CFSMS	M_CONTR_INF	CONTR_SIZE	CONTR_SIZE	Container Specification	컨테이너 규격	varchar	3	20	Y								
7	CFSMS	M_CONTR_INF	CONTR_TYPE_CD	CONTR_TYPE_CD	Container Type	컨테이너 타입	varchar	10	GP									
8	CFSMS	M_CONTR_INF	CONTR_LOAD_STS	CONTR_LOAD_STS	Loaded status	적재상태	varchar	1	F					보관 컨테이너, 이용불가적 입출 컨테이너				

Item	Item Definition and Creation Guidelines
DATABASE	Write the name of the database to which the column belongs
TABLE	Write the name of the table to which the column belongs
Field	The physical English name of the column
COLUMN (Initials Case)	Use abbreviations as the physical English name of the column
DESCRIPTION (ENG)	Additional description of column
RELATED ENTITY (ENG)	Write the logical data element 'entity name' that the column represents
DATA TYPE	Write the physical DBMS datatype of the column value
DATA SIZE	Write the length of the physical DBMS data of the column value
DATA FORMAT	The format of the data to represent the values in that column
Not Null Y/N	Indicates whether a column value must exist at the time the data is generated (insert)
Primary Key Y/N	If a column participates in a PK (default key), mark it using the "PK" and numeric order of participation and omit it if you do not participate in a PK

Item	Item Definition and Creation Guidelines
Alternate Key Y/N	If you are a column participating in an AK (alternative key), mark it using the "AK" and numeric order of participation, and if you are not participating in an AK, skip it
Foreign Key Y/N	Write the relevant table name and column name by a period (.) only if the column is one that participates in an FK (external key) constraint
Constraints terms	In addition to the characteristics of the column values described in the corresponding value area, additional constraints to be specified for the column (such as allowable range, delimitation, and default values) are described
USE Y/N	Write whether the column is used or not
PERSONAL Y/N	Whether the column values contain personal information ("Personal Information De-identification Action Guidelines" Identifier Action Criteria)
ENCRYPT Y/N	Write whether the column is encrypted for privacy reasons, etc
OPEN/CLOSE Y/N	Whether the metadata and source data information in the column are disclosed or not disclosed (in the case of non-disclosure, the reason for non-disclosure is stated)

Figure 13. Column Definition Form

Column Definition Form Used for Data Standardization in CFS/CY Port Logistics Warehouse Management Systems and Port Logistics Transportation Management Systems

The form used in import/export container transportation and transport order management systems between export/import port terminals and logistics warehouses within the port hinterland follows the guidelines shown in Figure 7: Column Definition Form. The column definition form is based on an Excel document and utilizes filtering functions for ease of use. It serves as a column management register for adding and modifying columns by database and table, and includes information such as table name, column name, column name abbreviation, description, data type, data size, data format, usage status, personal data protection, and encryption requirements.

6.3 Data Trust and Security

6.3.1. Private Blockchain-Based Data Integrity Assurance

Inje University develops and operates the Trust Layer that ensures data integrity across the SOCFAI port-logistics ecosystem.

A private blockchain network is deployed among participating institutions (EC-KAIST, TMS-eINS, EC-KULS, TMS-KULS), where each organization maintains an independent node. The blockchain serves as a tamper-proof audit substrate, recording cryptographic hash values of key logistics events without storing raw operational data.

The system guarantees immutability and cross-institution verifiability by storing SHA-256 digest values of uploaded CFSMS/TMS data, optimization data accesses, and result distribution events. Any unauthorized modification in upstream systems can be detected through hash mismatch verification.

This architecture eliminates trust gaps between institutions and provides a shared, immutable history of all critical data-handling operations within the SOCFAI platform.

6.3.2. API Request/Response Hash Recording and Audit Log Function

Inje University implements an API-level audit logging mechanism that captures both request and response payloads in hashed form and writes them to the private blockchain. For every API invocation (data upload, retrieval, optimization request, or result dispatch), the system computes:

- Request Payload Hash
- Response Payload Hash
- Timestamp and Caller Identification

Only hash digests are recorded on-chain to avoid exposing sensitive port-logistics or personal data. This approach ensures traceability, non-repudiation, and provable integrity of all API interactions between SOCFAI subsystems.

The audit log function provides a foundation for cross-institution dispute resolution, post-hoc verification, and compliance monitoring. Through this mechanism, Inje University strengthens transparency and accountability within the SOCFAI data pipeline.

7. Port Specialized AI Models and Services

7.1 Container Yard Optimization (KAIST)

Within the Port Domain, the SOCFAI Platform provides container yard optimization functions that enhance operational efficiency at both the individual CFS level and the network level across multiple CFS facilities. These functions combine reinforcement learning-based intra-yard stacking optimization with cost-driven inter-yard assignment strategies, enabling reduced rehandling, balanced yard utilization, and improved long-term operational stability throughout the logistics workflow.

7.1.1 Intra-CFS Optimization (Within Yard): PPO (Proximal Policy Optimization)-Based Reinforcement Learning Model

- The intra-yard optimization module supports automated decision-making within a single CFS by using a Proximal Policy Optimization (PPO) reinforcement learning model. The model is trained in a simulation environment that reflects realistic yard conditions, including stack geometry, capacity constraints, and the sequential flow of container handling tasks. Through repeated interaction with this environment, the model learns operational strategies that reduce overall handling workload.
- The state representation reflects the current configuration of all stacks in the yard, the occupancy status of each position, and the attributes of the container involved in the next operation.
- At every decision step, the agent selects an operational action. These actions include placing an arriving container into a designated stack or relocating a container that blocks an ongoing retrieval. All actions are defined by the physical layout and available capacity of the yard.
- The reward function is designed to reduce cumulative handling effort. The model receives higher rewards when the selected actions are expected to lower the total number of relocations required throughout the handling process.
- As learning progresses, the PPO agent develops yard-handling patterns that lead to more efficient operations and reduced workload compared with heuristic or rule-driven strategies.
- The trained model is deployed as a microservice within the SOCFAI Platform and supports real-time decision-making by integrating directly with standardized port logistics data streams.

7.1.2. Inter-CFS Optimization (Between CFSs): Optimal CFS Selection Algorithm Based on Minimizing Expected Operational Cost

The inter-yard optimization module supports container assignment across multiple CFS facilities by estimating the expected operational cost of sending an inbound container to each available yard. This function enables a more balanced distribution of workload within the Port Domain and prevents excessive concentration of operations at a single location. The module operates within the SOCFAI Platform and uses standardized logistics data to support real-time decision-making.

- The selection process evaluates each candidate CFS using an expected cost metric that reflects the overall handling effort required after the container is stored. This metric

considers the current occupancy, stack availability, and the predicted workload associated with future handling tasks.

- For every inbound container, the decision engine compares the expected cost values of all eligible CFSs and selects the yard that is anticipated to require the least total handling effort.
- Several assignment options can be referenced for comparison, including random assignment, assignment focused only on immediate handling needs, and assignment based on longer-term workload estimates. The balanced cost-based method provides a more stable operational outcome by balancing these factors.
- By distributing inbound containers across CFSs according to anticipated workload, the module reduces the likelihood of local congestion and improves the overall efficiency of yard operations across the network.

7.2 Digital Twin-Based Dispatch Optimization (eINS S&C)

This section describes the key components and data flow architecture of the Digital Twin–based Port Logistics Dispatch Optimization System developed by eINS S&C. The system is designed to enhance the efficiency of container truck operations within port logistics by integrating a DEVS-based simulation model, an optimization module utilizing Greedy and Simulated Annealing algorithms, and a real-time data integration environment. Through this configuration, the system aims to achieve more than a 10% improvement in total transportation time.

7.2.1 Simulation Model: DEVS (Discrete Event System Specification)-Based Modeling (Truck Atom, Container Yard Coupled Model)

The simulation engine is designed based on the Discrete Event System Specification (DEVS) formalism. Individual behaviors of trucks and container yards are defined as Atomic Models, accurately representing real-world port logistics processes.

When a dispatch schedule is provided as input, each truck and yard model performs its respective operations through the simulation engine. Upon completion, the simulated results for each truck are stored in the database.

- Truck Atomic Model: Models the state transitions of trucks—including movement, loading/unloading, waiting, and operation completion—as discrete events. Each event is simulated according to defined time and state transition rules.

- Container Yard Atomic Model: Reflects the operational environment within terminals and Cargo Freight Stations (CFS), incorporating factors such as container in/out operations, equipment usage time, and work schedules.
- Model Coupling Structure: The truck and yard models exchange messages to dynamically process loading and unloading events, reproducing the interactive behavior of the actual logistics environment. Simulation results are integrated with the database to quantitatively calculate key performance indicators (KPIs) such as transportation time and waiting time for each scenario.
- Simulation Execution : The implemented simulation models are executed using WAiSER, a proprietary Digital Twin-based simulation engine developed by eINS S&C. The WAiSER engine manages time steps and event scheduling, accurately reproducing the behaviors of transport vehicles and CFS/Terminal operations as specified in the models. This enables precise simulation of dynamic behaviors observed in real port operations, including task interactions, waiting times, and vehicle movements. The simulation outputs are structured as event logs and KPIs, which are stored in the database and utilized for performance evaluation and feedback within the optimization module.

7.2.2. Optimization Module: Greedy Algorithm and SA (Simulated Annealing) Algorithm

The Optimization Module of the Dispatch Optimization System consists of two main stages: a Greedy-based initial solution search and a Simulated Annealing-based global optimization process. Within the SOCFAI platform, the module operates as an independent service container, receiving input data from the platform and delivering optimized results through the database and blockchain API.

- Dispatch Optimization System Architecture

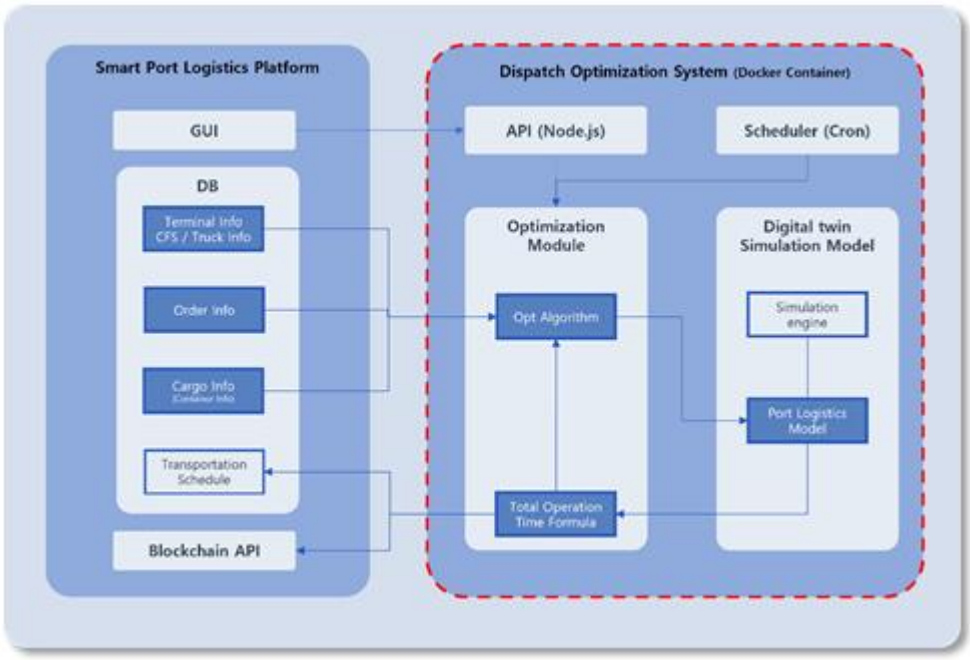


Figure 14. Dispatch Optimization System Architectural Model

- Data Input/Output Structure
  - Input: Transport order data, truck availability, CFS/Terminal information, and other operational parameters.
  - Output: Optimized dispatch plan (scheduling results)
  - Communication: Platform database and blockchain API for secure data exchange.
- Initial Solution Generation (Greedy Algorithm): Based on the origin and destination of each transport order, truck availability, distance, and estimated travel time, the algorithm greedily constructs an initial dispatch schedule that minimizes operational time in a straightforward and efficient manner.
- Global Search (Simulated Annealing Algorithm): Starting from the initial solution, the algorithm iteratively explores neighboring solutions through probabilistic exchanges to minimize the Total Operation Time. By occasionally accepting less optimal solutions, the algorithm avoids local optima and converges toward a global optimum.
- Simulation Coupling: Each candidate schedule generated during the optimization process is validated through the Digital Twin–based Port Logistics Simulation Model. The · simulation reflects real-world variables such as processing time, travel

duration, and waiting delays, ensuring the practicality and reliability of the optimized solution.

- Containerization and Deployment : The Dispatch Optimization System, including both the optimization module and simulation model, is containerized using Docker. This approach ensures consistent execution across development, testing, and deployment environments, minimizes inter-module dependencies, and enhances system scalability and reliability within the SOCFAI platform.
- System Operation Modes
    - Daily Optimization Mode : The system automatically executes the optimization module one day prior to scheduled operations. A Cron-based scheduling service triggers the optimization process at a predefined time each day. The module runs for approximately one hour, explores the solution space, and registers the optimized dispatch plan into the platform database for user access.
    - Emergency Optimization Mode : When operational change, for example due to updated transport orders, the system can be executed on demand through REST API calls to the containerized service. In this mode, the optimization process typically runs for about 10 minutes to produce a revised dispatch plan, allowing the platform to respond quickly to real-time operational updates.

7.2.3 Performance Target: Over 10% Improvement in Total Transportation Time

The performance of the system is evaluated using real port logistics transportation data. Compared to the baseline schedule, applying the optimization system results in an average reduction of more than 10% in total transportation time (Total Transportation Time).

- Evaluation Criteria:
    - The optimization algorithm is applied under identical order conditions to those of the baseline schedule.
    - Truck operation and waiting times are compared across multiple test scenarios.
- Results Summary:

- Achieved an average transportation efficiency improvement of over 10%.
- Expected to contribute to reduced transportation costs, improved equipment utilization, and overall operational efficiency in real-world port logistics environments.

8. Port Applications and Mobile Environment

8.1. CFS/CY and TMS Platform Functions

8.1.1. Container Order Registration/Management, Loading/Unloading/Transfer Process Detail Management

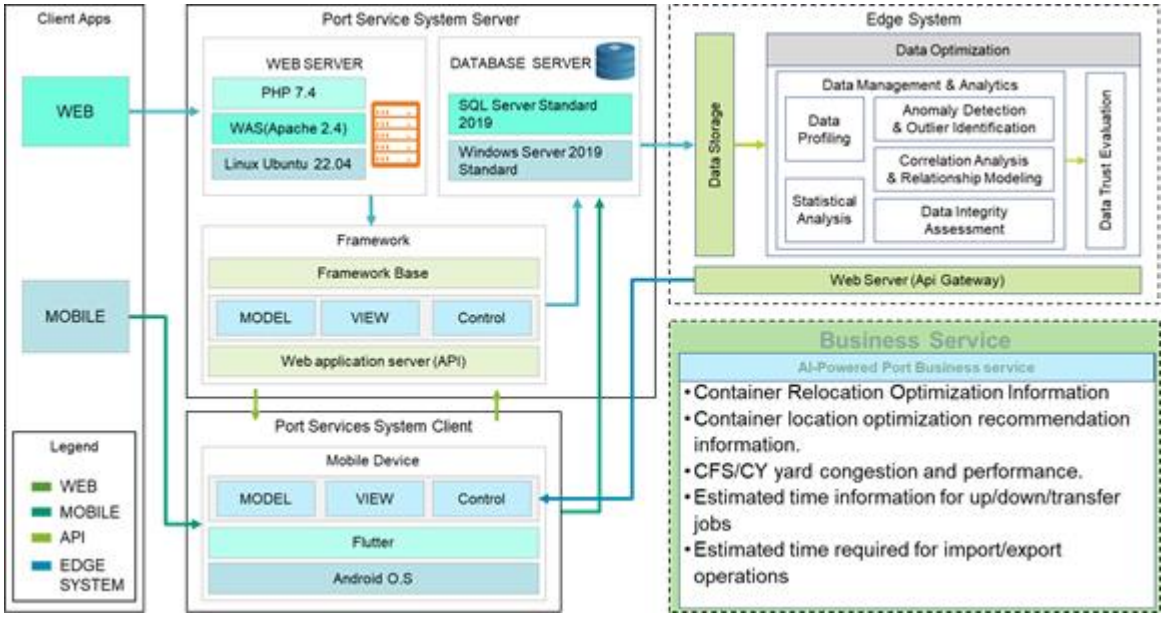


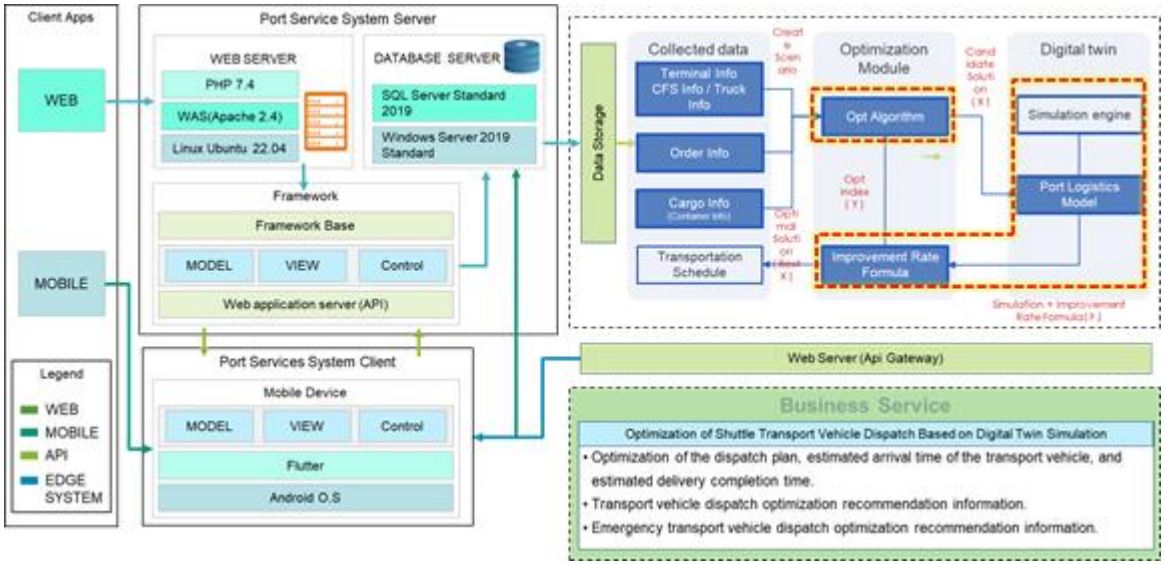
Figure 15 CFS/CY and TMS Platform Functional Model

The system enables real-time monitoring of the location and operational status of import/export containers, systematically manages inbound and outbound schedules, and supports the registration and management of container work plans and progress status. It also allows pre-registration of inbound and outbound schedules to link with transportation planning.

Additionally, it continuously monitors real-time container inventory and status, efficiently manages loading, unloading, and transshipment work history, as well as logistics performance,

and provides real-time mother vessel information to support work planning and transportation scheduling. Furthermore, it allows users to register transport requests, track status changes, and systematically manage transport progress, while supporting seamless information sharing and collaboration between field workers and office staff through web and mobile device–based environments.

8.1.2. Real-time Inventory Management and Transport Status Information Management



The system collects and centrally manages import/export port logistics and transportation information, enables registration and management of import/export container transport orders, and systematically manages detailed transport information and vehicle dispatch planning. It also allows real-time monitoring of container transportation status for quick and accurate tracking of operational progress, supports performance analysis based on import/export container transport activities, and enables registration, management, printing, downloading, and issuance of transaction statements. Furthermore, it provides features for issuing electronic tax invoices, transmitting them to the National Tax Service, and managing cancellations efficiently.

8.2. Mobile-Based Field Operations

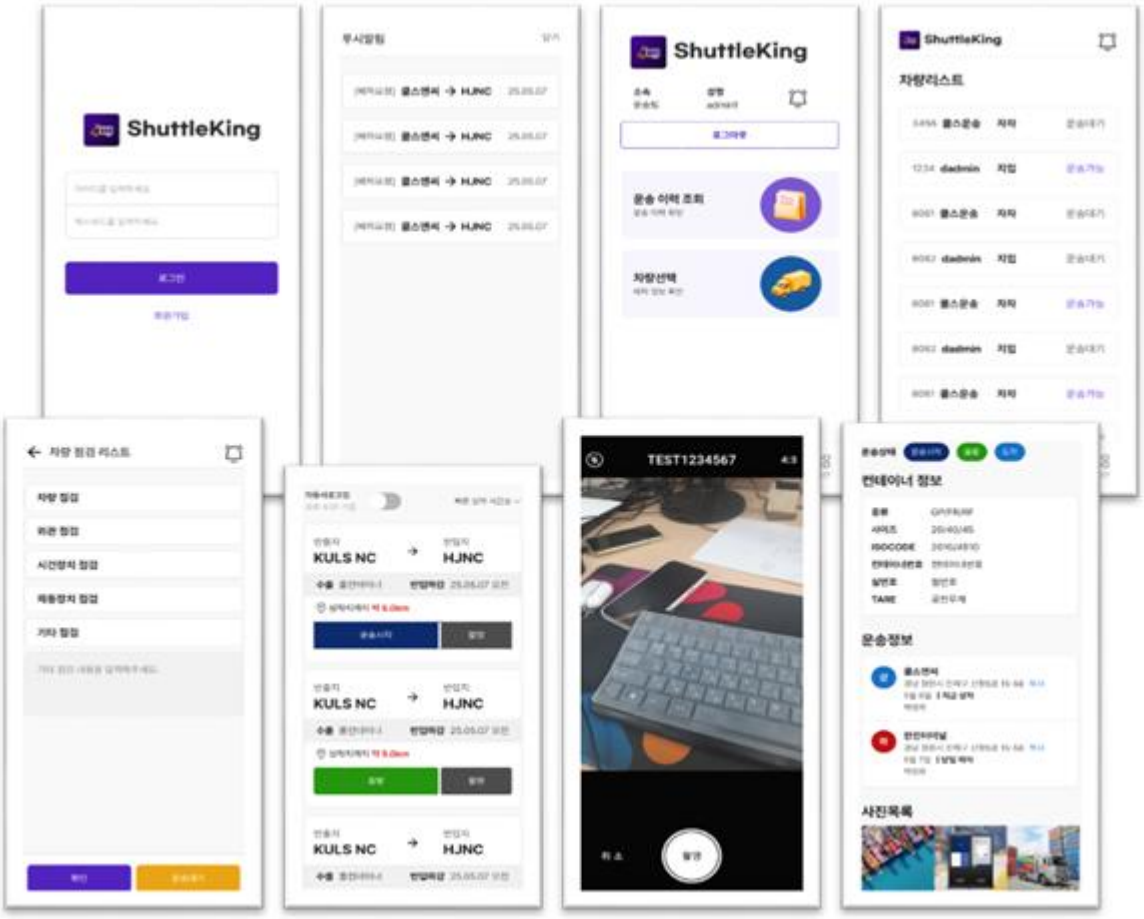
8.2.1. Mobile App Development (Flutter, Android O.S.)

The system enables real-time monitoring of the location and operational status of import/export containers, systematically manages inbound and outbound schedules, and tracks transportation requests and transportation progress while integrating and managing port logistics information. It also efficiently manages the history and progress of loading, unloading, and transshipment operations. Moreover, by providing a web and mobile device–based environment, it supports seamless communication between field workers and office staff and enables the registration and management of import/export container work plans.

8.2.2. Vehicle Location Control and Transport Status Data Collection

The system collects and centrally manages import/export port logistics and transportation information while systematically managing detailed container transport data and vehicle dispatch information. It continuously monitors real-time transport status and vehicle locations to accurately track operational progress and supports performance analysis and management based on import/export container transport operations. Additionally, it provides a web and mobile-enabled environment that facilitates seamless communication between transport

company offices and drivers, enabling real-time updates on transport status, vehicle management, and container operations to support rapid and informed decision-making.



9. Conclusions

9.1. SOCFAI Platform Implementation Achievements and Implications for Airport Domain

The implementation of the SOCFAI Platform in the airport domain has successfully transformed fragmented operational processes into an integrated, predictive, and data-driven management system.

(1) Key Achievements in Technical Infrastructure

- Unified Data Integration: The platform serves as a single entry point for integrating disparate data sources, including AODB (flight schedules), ADS-B (real-time flight tracking), LiDAR (3D spatial data), and IoT/Environmental sensors.
- Scalable Architecture: By adopting a container-based microservices architecture (Docker) and a Kafka-based real-time data pipeline, the platform ensures high-throughput, low-latency data processing essential for real-time airport operations.
- Robust MLOps Framework: The integration of tools such as MLflow for experiment tracking, MinIO for artifact storage, and NannyML for drift detection ensures the sustainability and reliability of AI models in a live environment.

(2) Specialized AI Model Performance

- Flight Arrival Prediction (ATAD): Developed a GPU-accelerated Random Forest model that predicts arrival times within a two-minute margin (MAE: 109 seconds), significantly improving gate and ground handling planning.
- Passenger Flow Management: Implemented the VoxelNeXt 3D detection model using LiDAR to monitor crowd density and flow dynamics in real-time while preserving passenger privacy.
- Baggage & Delay Analytics: * Achieved an R^2 score of 0.94 for hourly baggage pattern prediction.
    - Identified a strong correlation between Pattern Dissimilarity (PD) in baggage and flight delays, with higher PD ratios corresponding to a higher likelihood of operational delays.
- Passenger Sentiment Analysis: Fine-tuned a DistilBERT model to analyze airport reviews, allowing operators to visualize sentiment changes in relation to flight delays.

(3) Operational Implications

- Proactive Decision Making: The platform enables a shift from reactive to predictive operations, allowing Airport Operations Control Centers (AOCC) to anticipate bottlenecks and optimize resource allocation (gates, counters) before disruptions occur.
- Enhanced Situational Awareness: Through the Real-time Statistics and Prediction Pages, operators gain a consolidated view of flights, resource utilization, and environmental conditions (such as the Air Quality Index).
- Resource Optimization: The integration of mathematical optimization (Mixed-Integer Programming) and Reinforcement Learning within the Resource Management System (RMS) improves the efficiency of assignment rates and conflict resolution.

9.2. SOCFAI Platform Implementation Achievements and Implications for Port Domain

Through the development of the port logistics platform and port logistics transport management platform, we successfully established a digital-based integrated operating system that manages complex business processes related to the transportation of import and export containers. We also performed transport order request, order registration, vehicle dispatch, real-time location-based monitoring, transport status tracking, CFS/CY warehouse inbound/outbound management, performance tracking, settlement closing and carryover management, transaction statement issuance, e-tax invoice processing, etc. Through this implementation, stakeholders can move away from manual planning, task status management, container allocation and location control, and fragmented system environments. Instead, we can collaborate efficiently through real-time information sharing using web and mobile-based environments tailored to various businesses and user roles such as carriers, warehouse operators, field workers, drivers, and shippers. In addition, real-time operation data enables accurate transportation performance analysis and automated cost settlement, which greatly improves business efficiency and accuracy.

In addition, using open-source technologies such as Kafka, Devezium, Prometheus, and Grafana, we have established a robust data pipeline and real-time monitoring system to achieve standardization and real-time data integration between systems. This ensures data consistency and reliability among various stakeholders within the port logistics ecosystem, minimizing transport delays, missing data, and information disconnection. It has also laid the foundation for advanced service expansion such as AI-based container placement optimization, digital twin simulation for optimizing vehicle dispatch, container location tracking, and predictive logistics operation models.

The impact on import and export port logistics is as follows:

- First, port logistics and transportation management systems need to evolve beyond simple information delivery into a data-driven decision-making support platform. With the support of real-time data collection and standardized data management frameworks, AI predictive analysis, digital twin-based simulation, and expansion to smart logistics operational models are already underway.
- Second, smooth data linkage between various entities such as CFS/CY warehouses, port terminals, transport operators, customs administration systems, and the IRS is essential. Preparations are underway for the establishment of a standardized data interface and open API strategy, including the integration of the API for issuing transaction statements and electronic tax invoices with the IRS, as well as linking EDI-based data between port terminals and transport operators. The integration with the customs administration system is also planned in the future.

In the Port Domain in terms of SOCFAI Platform, the following considerations are requested to promote more advanced capability implication.

- The developed Digital Twin-based optimal dispatching system has demonstrated its capability to enhance truck scheduling efficiency within port logistics operations. In the future, this system can be extended in several directions to maximize its applicability and overall value.
- First, the system can be expanded into a multimodal logistics framework, integrating air, road, rail, and maritime transport for unified scheduling and optimization. This expansion would enable a comprehensive understanding of cargo flows across different transport modes, facilitating seamless intermodal coordination and overall cost reduction in logistics networks.
- Second, by integrating real-time IoT sensor data and AI-based predictive models, the platform can evolve into a real-time decision support system. Through this integration, the system can dynamically adjust schedules in response to delays, congestion, and unexpected disruptions, achieving more intelligent and adaptive logistics management.
- Third, the developed technologies can be applied to other industrial domains as well. For instance, similar optimization principles can be utilized in factory AGV (Automated Guided Vehicle) scheduling, warehouse operations, and smart city urban logistics systems, where resource allocation and dynamic routing problems also exist, making technical adaptation feasible.Lastly, as the system continues to mature, it can contribute to the full-scale establishment of a Digital Twin for overall port logistics.

- Such a system would provide valuable data-driven insights to support policy-making, infrastructure planning, and sustainability assessment.

- **Port logistics platform and port logistics transport platform AI module and digital twin module service integration**
  - Build a data pipeline and connect data with each institution.
  - Build a blockchain network on an intelligent integrated logistics platform and prepare integrated smart contract API and blockchain services.
  - Prepare for integration with AI-based container placement optimization modules and blockchain-based smart contract API and blockchain serviPrepare digital twin simulation-based container dispatch plan optimization module, smart contract API, and blockchain service integration.

- **Future Applications and Deployment Scenarios for AI-based CFS/CY Optimization**
  - The Inter-CFS optimization technology can be applied as an automated recommendation mechanism that continuously evaluates workload imbalance across multiple CFSs and suggests the most suitable station for each incoming container. This would help prevent unnecessary concentration in congested yards and maintain stable system-wide operational efficiency.
  - The Intra-CFS optimization model can evolve into a container placement recommendation system that incorporates retrieval schedules, equipment availability, and container priority. Such an extension would improve the reliability of handling time-critical or sensitive cargo by ensuring that high-priority containers are placed in more accessible positions.
  - By integrating the outputs of both Inter- and Intra-CFS optimization, the system can be used to support an end-to-end decision-making workflow covering the entire process. This would enable the prevention of redundant rehandling and minimize the total operational cost across overall container management.

- **Data Trust Provisioning and Blockchain Capabilities**
  - Expansion of Smart Contract Utilization: The smart contracts, which currently focus on serving as an immutable 'Trust Layer' for transaction logging and auditing, can be applied in the future as an automated process execution mechanism that triggers associated follow-up tasks when a specific logistics event is verified.
  - Application of Blind Computing Technology: To enhance data confidentiality in the multi-party computing environment, 'blind computing' technology, as mentioned in the document, can be applied. This can be utilized as an advanced security feature that allows partners to prove data integrity without exposing sensitive business information.

- Securing Network Scalability and Interoperability: In preparation for the future increase in logistics data, measures for securing network scalability and interoperability with external systems can be explored. This can be used to support future scenarios requiring higher transaction throughput or trusted data linkage with third-party logistics platforms.

Advancement of Trust-Based Data Governance: The established node governance framework can be used to support a more sophisticated data sharing and access control model in the future. This can be leveraged as a foundation for an advanced collaborative environment among consortium partners.