D3.1 & D3.2 Mid Version of Use Case Applications

Secure Open Collaboration Framework powered by Artificial Intelligence

31 January 2025

Abstract

This deliverable includes the mid version of use case applications in SOCFAI here

Contents

1 . Introduction

This document provides a comprehensive overview of several use case applications of the SOCFAI project, detailing their purposes, scopes, developments, and data management strategies. It systematically examines each application, from its initial conceptualization and defined objectives to the intricate aspects of its system architecture and functional implementation performed by SOCFAI partners. Furthermore, the document delves into the crucial area of data management, outlining the data sources, integration methods, and usage patterns specific to each use case. Finally, it presents the data analysis and results derived from the applications, concluding with insights into future works and potential enhancements. This structured approach allows for a thorough understanding of the design, implementation, and outcomes of diverse use case scenarios of the SOCFAI project.

2 . Purposes and Scopes of Use Case Applications

2.1 Use Case Application #1 (INOSENS)

In Use Case 1, the main purpose is to perform passenger flow monitoring and passenger density estimation in an airport environment using LiDAR sensor technology. Due to privacy concerns, there are restrictions on using cameras. 2D Camera-based crowd detection methods pose certain challenges especially when cameras are located from the top to see people on the ground. Lidar sensor provides 3D point cloud data which doesn't include personal information on the contrary of camera or CCTV but includes location information for each individual in a certain range. Therefore, the plan is to provide a LiDAR-based solution for human detection to support security purposes such as people detection and crowd estimation.

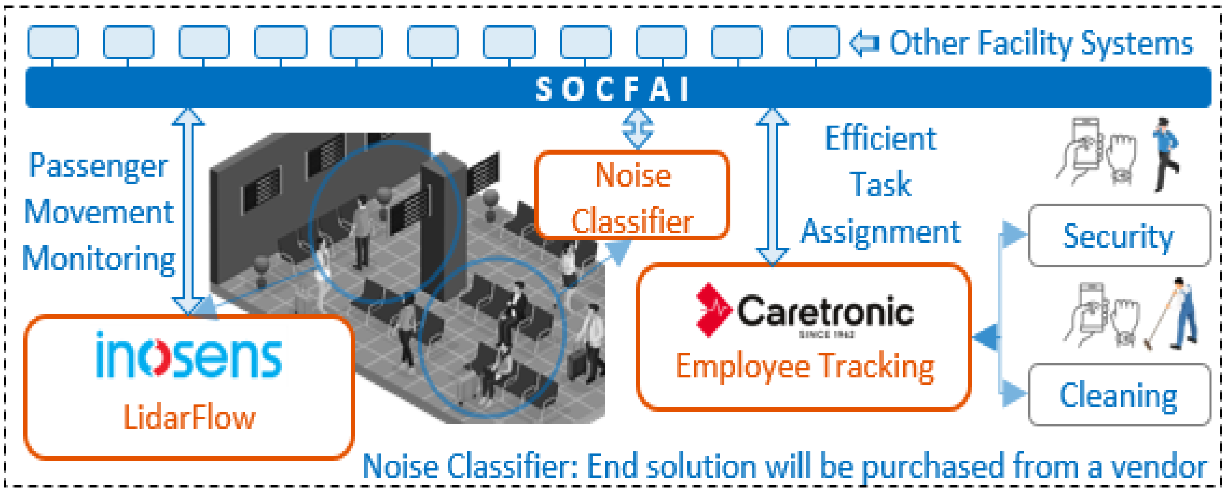

Other Facility Systems
SOCFAI
Passenger Movement Monitoring
Noise Classifier
Efficient Task Assignment
Security
inosens
LidarFlow
Caretronic
SINCE 1962
Employee Tracking
Cleaning
Noise Classifier: End solution will be purchased from a vendor

Figure. General View of Use Case 1

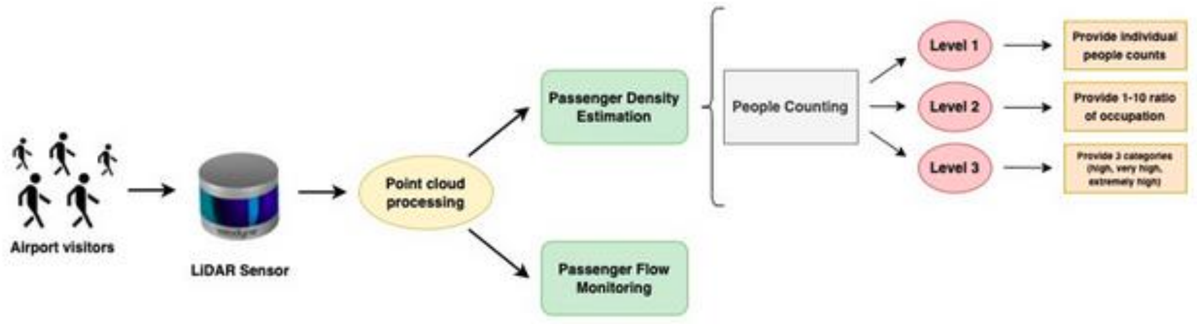

Airport visitors
LiDAR Sensor
Point cloud processing
Passenger Density Estimation
Passenger Flow Monitoring
People Counting
Level 1
Level 2
Level 3
Provide individual people counts
Provide 1-10 ratio of occupation
Provide 3 categories (high, very high, extremely high)

Figure. Purposes of Use Case 1

2.2 Use Case Application #2 (Siemens, TAV)

Use case 2 (Siemens use case) focuses on the baggage analytics and predictions to optimize the baggage operations in an airport setting, specifically within the baggage handling system. It includes a real-time dashboard to provide real-time statistics for international & national flights and predictions per daily, hourly and pattern of baggage counts. TAV Technologies supplies baggage, flight and passenger data to Siemens which is collected from İzmir Adnan Menderes Airport. Within this use case, the developed solution incorporates AI services for predictive analytics, daily, hourly, 15 minutes and flight-specific forecasting, delay prediction and an optimization recommendation system. The solution is developed as docker containerization, platform-independent architecture with SQL server for data storage.

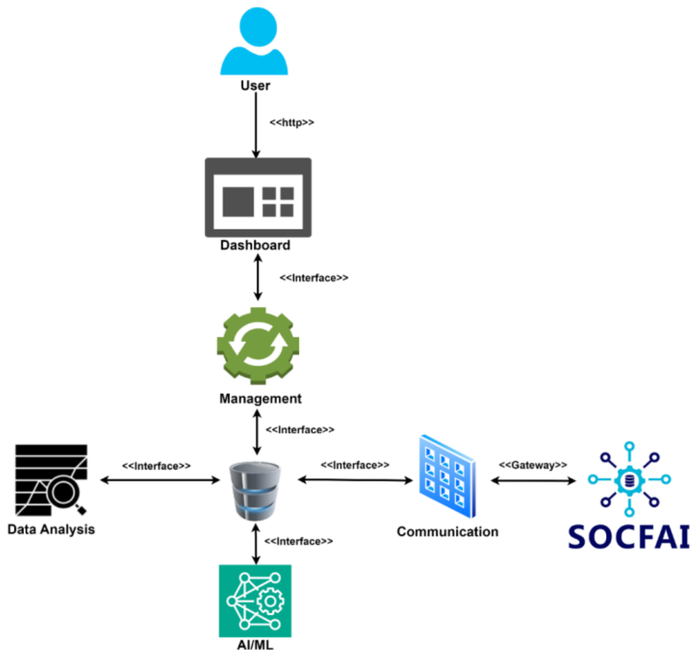


Figure. General View of Use Case 2 (AI-Based Baggage Analytics and Optimization)

2.3 Use Case Application #3 (Enverse, NETAŞ, TAV)

Use Case 3 focuses on enhancing energy efficiency and operational intelligence at İzmir Airport through collaborative efforts among Enverse, NETAŞ, and TAV Technologies.

NETAŞ will develop an IoT-based indoor air quality monitoring system to ensure optimal environmental conditions within the airport. Enverse will deliver real-time energy analytics, predictive models, and an AI-driven control system to optimize energy consumption. TAV Technologies supplies air quality sensor data to NETAŞ which is collected from İzmir Adnan Menderes Airport and all together work on dashboards.

Key components of this use case include:

- Energy Consumption Forecasting – AI-based prediction of energy demand using factors such as passenger traffic, weather data, and SCADA inputs.
- Real-time Energy Monitoring – A cloud-based solution for tracking energy usage.
- HVAC Control System – A digital twin-powered optimization tool to regulate heating, ventilation, and air conditioning efficiently.

A major challenge addressed in this scenario is avoiding energy overconsumption penalties by accurately forecasting demand. The solutions developed under Use Case 3 will be implemented and validated at İzmir Airport, demonstrating their effectiveness in real-world airport operations.

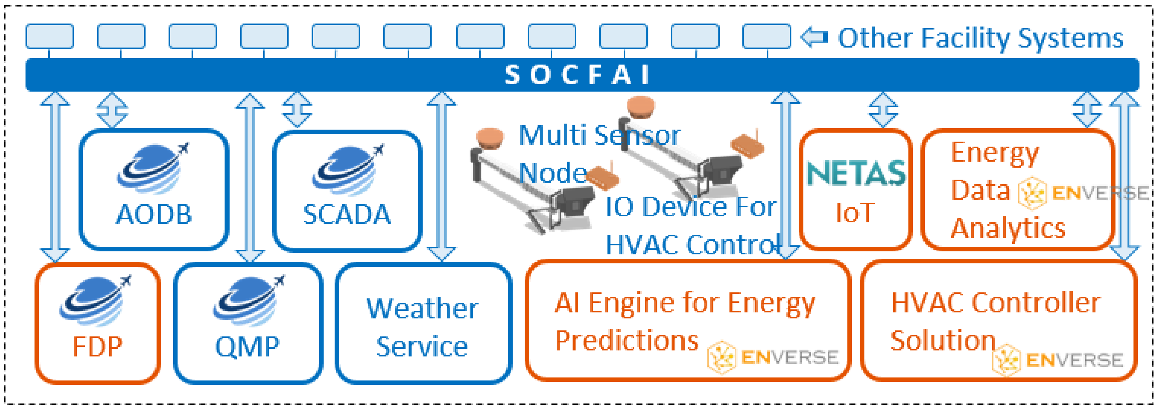
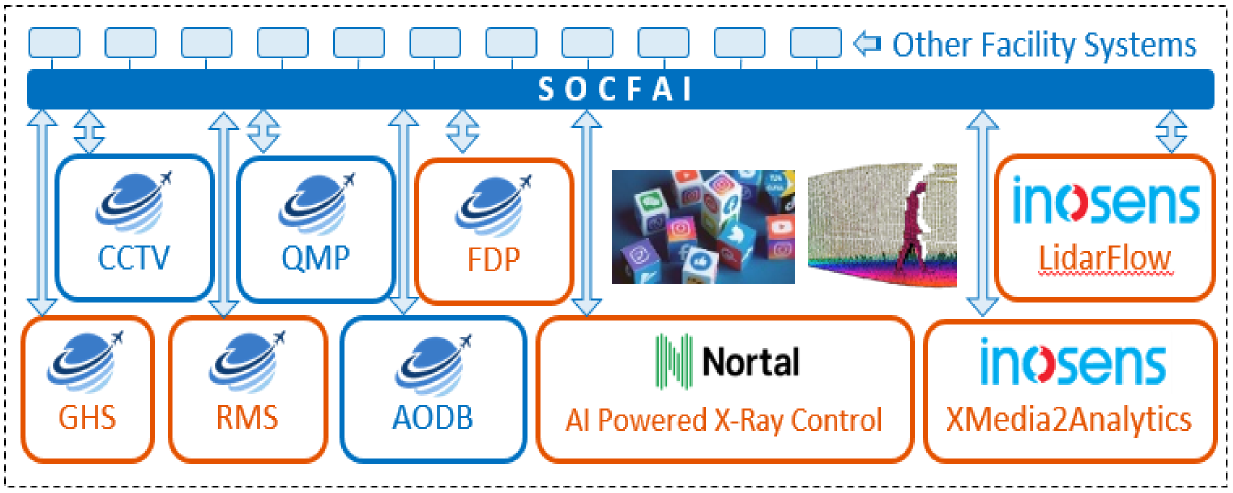


Figure. Data exchange map of use-case 3

2.4 Use Case Application #4 (INOSENS)

In Use Case 4, the main task is to perform sentiment analysis on text-based social media posts, comments, and reviews. The goal is to evaluate passenger satisfaction based on what people are saying online about Izmir airport. The use case also involves analyzing the correlation between time and sentiment and identifying the change in passenger satisfaction after flight delays. XMedia2Analytics is the proposed framework. It provides analytics for businesses using social media and other online sources to help organizations with making intelligent decisions and meeting their goals.



F.gure. General View of Use Case 4

2.5 Use Case Application #5

In the AI powered Port Logistics Management scenario of Use Case Application #5, we implement the transformative solutions for intelligent container yard management and inter-terminal logistics flow within Port environments in terms of Use Case Application #5. This use case application solution employs state-of-the-art technologies such as automated scheduling, real-time tracking, predictive analytics, and digital twins to optimize container placement, data analysis, and shipment dispatching. By leveraging artificial intelligence, these systems not only improve the efficiency of current operations but also proactively address future demands and unexpected operational challenges.

Partnerships with esteemed institutions like KAIST, INJE University and eINS S&C have been instrumental in integrating cutting-edge research with practical applications to overcome traditional constraints of distance, traffic, and operational bottlenecks. The collaboration has also emphasized the importance of data confidentiality and utilization control, which is crucial in a multi-party computation environment.

The Port Logistics Service Platform collects and manages legacy logistics information and multiple types of IoT data from logistics port, warehouse and transport systems, and these are connected with wired and wireless communications. As shown in the below figure, the functional framework to support "AI powered Intelligent Logistic Service Platform" will provide an optimized approach to combine these separately working functions with required operation, security, monitoring, management and logging functionality of port logistic data.

In the scenario of use case 5, data trust solution in alliance with blockchain technology will allow various types of logistics data on the "Intelligent Logistics Service Platform" safe and reliable. And an AI based digital twin service solution will work on prediction and analysis in logistics processes.

Moreover, the inclusion of digital twin technology paves the way for sophisticated scenario creation and simulation-based optimization, providing a dynamic and responsive framework for decision-making processes. The creation and maintenance of these systems signify a monumental shift in Port logistics management, setting new standards for efficiency, reliability, and trust in the industry.

The AI powered Port Logistics Management Platform collects and manages legacy logistics information and multiple types of data from Port Operation, Port Logistics, and Transportation Data sources connected with wired and wireless communications as shown the below figure. These will be in accordance with the aviation SOCFAI framework.

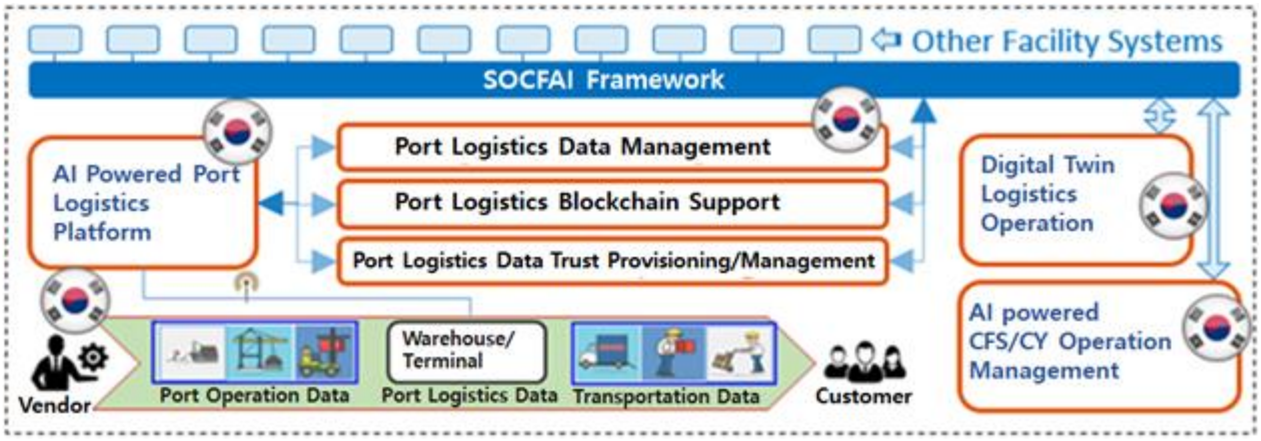

Other Facility Systems
SOCFAI Framework
AI Powered Port Logistics Platform
Port Logistics Data Management
Port Logistics Blockchain Support
Port Logistics Data Trust Provisioning/Management
Digital Twin Logistics Operation
AI powered CFS/CY Operation Management
Vendor
Port Operation Data
Warehouse/ Terminal
Port Logistics Data
Transportation Data
Customer

Figure. Implementation Scope of Use Case Application #5

2.11 Use Case Application #11 (TAV)

Use Case #11 focuses on enhancing airport operational efficiency through the development of three key AI-driven solutions by TAV Technologies: (1) **Flight Delay Prediction**, (2) **Advanced Resource Management System (RMS)** for fixed airport assets, and (3) **Advanced Ground Handling Suite (GHS)** for mobile resources.

- The **Flight Delay Prediction module** will use machine learning and rule-based methods to predict delays for arriving and departing flights using data from weather, city traffic, passengers, baggage, ADS-B signals, and real-time operations. Predictions will support all airport optimization systems.

- The **Advanced RMS** will improve fixed resource allocation (e.g., gates, stands, counters) by incorporating real-time operational and predictive data rather than relying solely on flight schedules from AODB.

- The **Advanced GHS** will optimize deployment of mobile resources (e.g., staff, vehicles) using real-time data on flights, passengers, baggage, and personnel capabilities. It will include multi-KPI optimization and integration with security systems for emergency alerts.

These developments aim to significantly improve the efficiency, accuracy, and responsiveness of airport operations.

3 . Developments of Use Case Applications

3.1 System Architecture

3.1.1 Use Case Application #1 (INOSENS)

In this project, Velodyne VLP-16 is used as the LiDAR sensor. Velodyne VLP-16, commonly known as the "Puck," is a compact and lightweight 3D LiDAR sensor designed for real-time environmental mapping. It features 16 laser channels and provides a full 360° horizontal field of view with a ±15° vertical field of view. The sensor generates up to 300,000 points per second, offering high-resolution point cloud data with a typical range of up to 100 meters and a distance accuracy of ±3 cm.

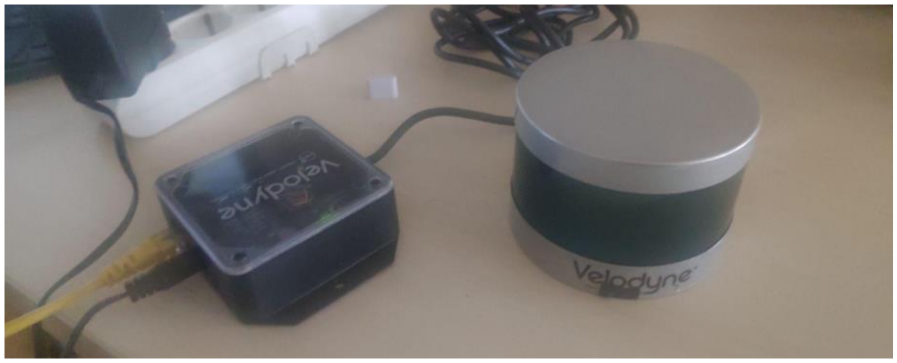


Figure. Velodyne VLP-16 LiDAR Sensor

In the system planned for Use Case 1, it is planned to transfer 3D point cloud data obtained from the lidar sensor located at the airport to the server, perform data pre-processing on the server, estimate passenger locations from a deep learning-based model and perform density analysis, and transfer these inferences to the platform via an API.

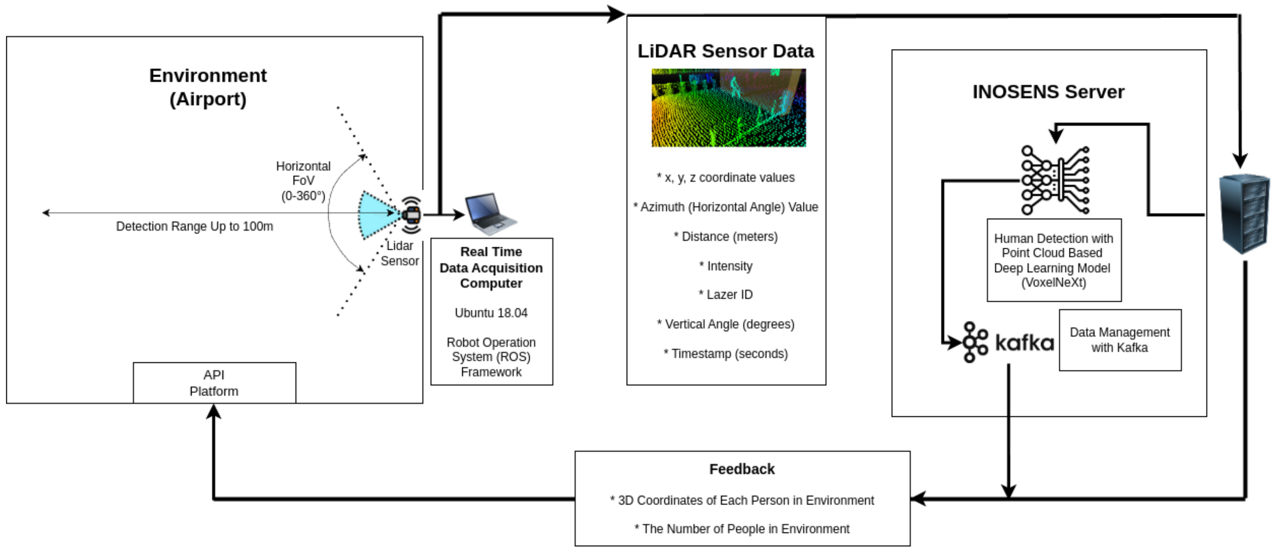

Environment (Airport)
Horizontal FoV (0-360°)
Detection Range Up to 100m
Lidar Sensor
API Platform
Real Time Data Acquisition Computer
Ubuntu 18.04
Robot Operation System (ROS) Framework
LiDAR Sensor Data
* x, y, z coordinate values
* Azimuth (Horizontal Angle) Value
* Distance (meters)
* Intensity
* Lazer ID
* Vertical Angle (degrees)
* Timestamp (seconds)
INOSENS Server
Human Detection with Point Cloud Based Deep Learning Model (VoxelNeXt)
kafka
Data Management with Kafka
Feedback
* 3D Coordinates of Each Person in Environment
* The Number of People in Environment

Figure. Architecture of Use Case 1

LiDAR point cloud data is collected with a computer that has Linux Ubuntu 18.04 operation system and Robot Operating System (ROS) framework. ROS framework provides some useful tools for collecting, visualizing and storing point cloud data in both data collection and real-time data acquisition stages.

After acquiring point cloud data, it is sent to the INOSENS server for analyzing the density of people in the airport. In the server, each human's coordinates in the point cloud data is predicted with a point cloud based deep learning model.

The human coordinates predicted by a deep learning model, along with the derived crowd density information, are transmitted to the airport system via Apache Kafka — a distributed event streaming platform designed for high-throughput, real-time data pipelines and communication between systems.

3.1.2 Use Case Application #2 (Siemens)

The overview of the high-level architecture, system components and their interactions for the solution addressing use-case 2 are illustrated in the following figure. The main building blocks are User Interface part for real-time dashboard, Data Analysis service for data processing, Management service, Database for storage operations, Communication service for real-time data communication with the SOCFAI platform and AI services for daily, hourly, pattern and flight delay predictions.

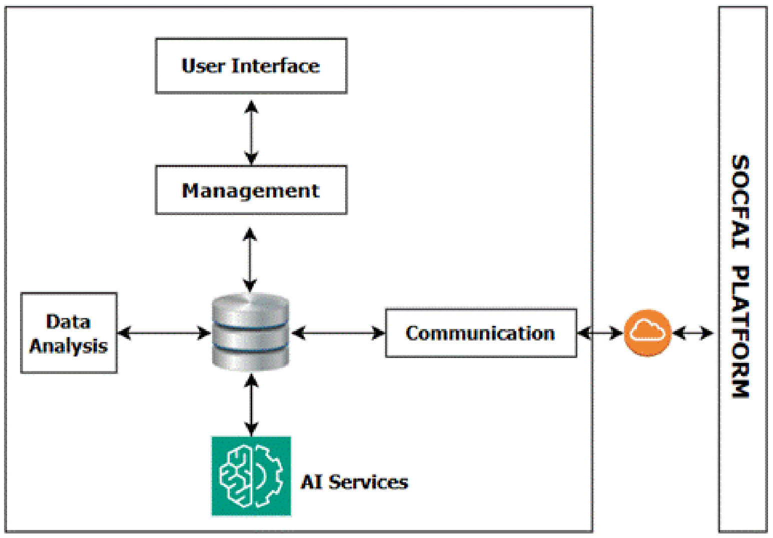


Figure. High-Level Architecture of AI-Based Baggage Analysis and Prediction Application

A real time dashboard is developed for the user interface and interactions with the end-users. The real-time dashboard represents a web page including real-time statistics and predictions about flight and baggage. Management handles requests from Dashboard and response. Communication gets real-time data from the external SOCFAI platform and populates into a database. Data Analysis analyzes populated data in the database to serve Management requests and AI/ML related tasks. AI/ML performs AI and ML algorithms and methods regarding provided data. Users interact with Dashboard via web pages like HTTP. SOCFAI provides real-time flight and baggage data.

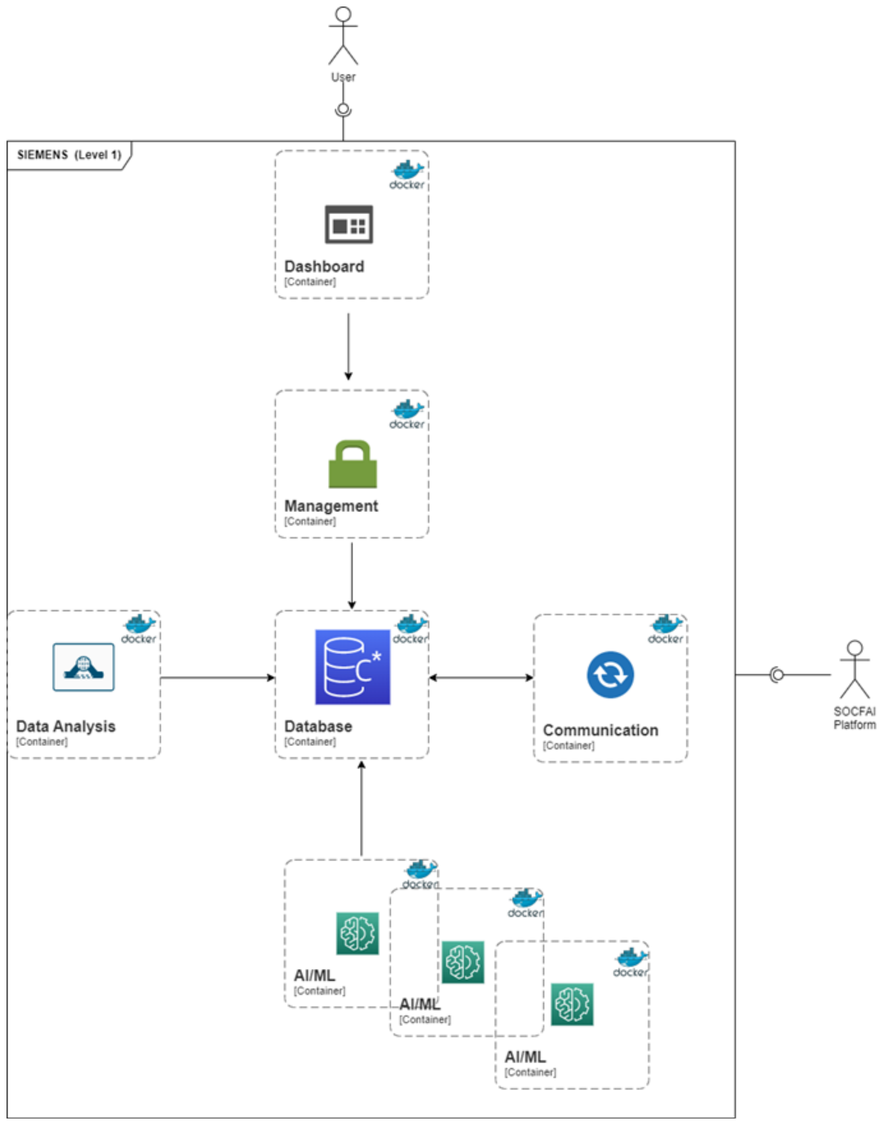


Figure. Whitebox Blocks View

Overview of the white box blocks and their interactions are illustrated in the above figure. AI/ML services represent a container group consisting of specialized containers for specific tasks. Communication service represents a container which communicates with an external network and feeds a Database container. Real-time dashboard represents a container which provides a web page as frontend. Database represents a container for storing real-time data. Data Analysis represents a container analyzing data in the database. Management service represents a container which provides data to a web page as a backend.

3.1.3 Use Case Application #3

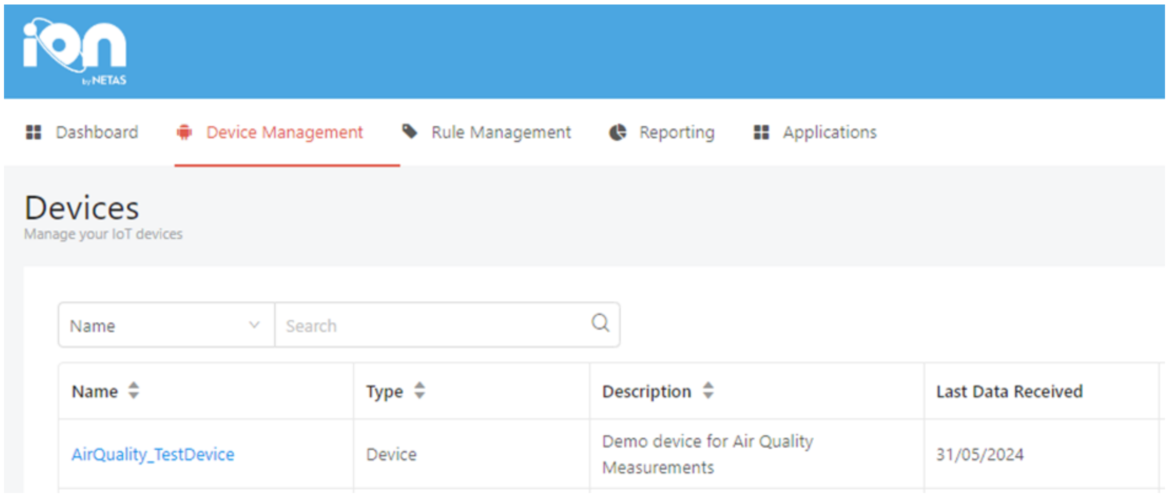
The main purpose of Use Case 3 is to measure the air quality level in airports and transmit this information to other systems to ensure energy efficiency. The European Environment Agency provides air quality data that encompasses a range of measurements. These measurements are classified into distinct categories, namely "good," "fair," "moderate," "poor," "very poor," and "extremely poor." These classifications will serve as indicators to assess the quality of air in different locations and aid in understanding the current conditions of air pollution.

The air quality index is calculated for the real time readings of sensor data received from the different locations in the selected pilot airport Adnan Menderes Airport (ADB). The received data includes the following data and the measurement values:

- Location information: deviceName, devEui, terminal, floor, zone
- Air Quality data: humidity, pressure, temperature, hcho, pm2.5, pm10, co2, tvoc
- Other: light_level, pir, crowdness, timestamp

Based on the data collected, the air quality index is calculated and the current status of each location is determined. The collaborative services of IoT platform – ION, is capable of generating alarms depending on the status of the air quality. These alarms will also be able to trigger some HVAC operations beyond.

The device descriptions are identified on Iot Platform - ION representing each sensor located in Adnan Menderes Airport.

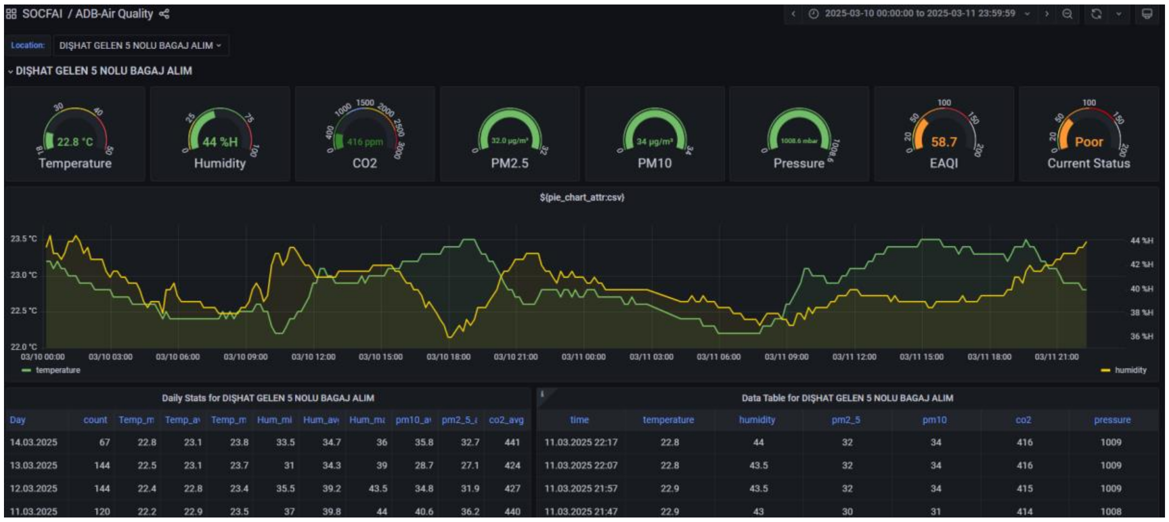


Devices
Manage your IoT devices

Name	Type	Description	Last Data Received
AirQuality_TestDevice	Device	Demo device for Air Quality Measurements	31/05/2024

Until the integration between the SOCFAI platform, the real data will be transferred to ION within the specified periods. For the current situation, manually transferred data are used for dashboards and visualization. The user can select a specific location in the airport and display the current and historical data measurements.

Some examples from the dashboard design are as follows:



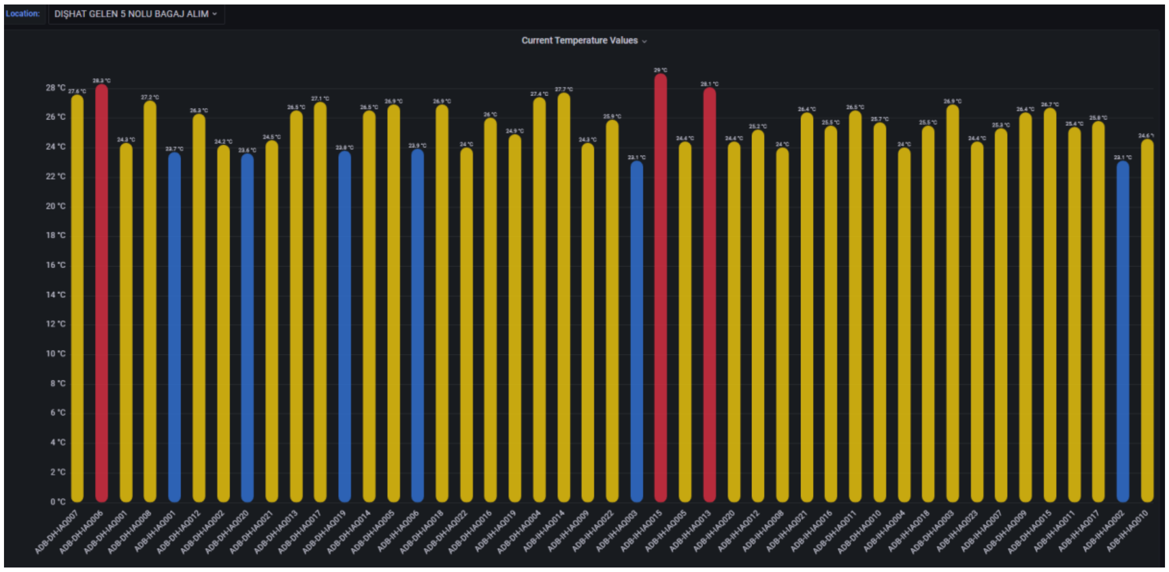
Daily Stats for DIŞHAT GELEN 5 NOLU BAGAJ ALIM

Day	count	Temp_m	Temp_a	Temp_m	Hum_mi	Hum_av	Hum_ma	pm10_a	pm2_5_	co2_avg
14.03.2025	67	22.8	23.1	23.8	33.5	34.7	36	35.8	32.7	441
13.03.2025	144	22.5	23.1	23.7	31	34.3	39	28.7	27.1	424
12.03.2025	144	22.4	22.8	23.4	35.5	39.2	43.5	34.8	31.9	427
11.03.2025	120	22.2	22.9	23.5	37	39.8	44	40.6	36.2	440

Data Table for DIŞHAT GELEN 5 NOLU BAGAJ ALIM

time	temperature	humidity	pm2_5	pm10	co2	pressure
11.03.2025 22:17	22.8	44	32	34	416	1009
11.03.2025 22:07	22.8	43.5	32	34	416	1009
11.03.2025 21:57	22.9	43.5	32	34	415	1009
11.03.2025 21:47	22.9	43	30	31	414	1008

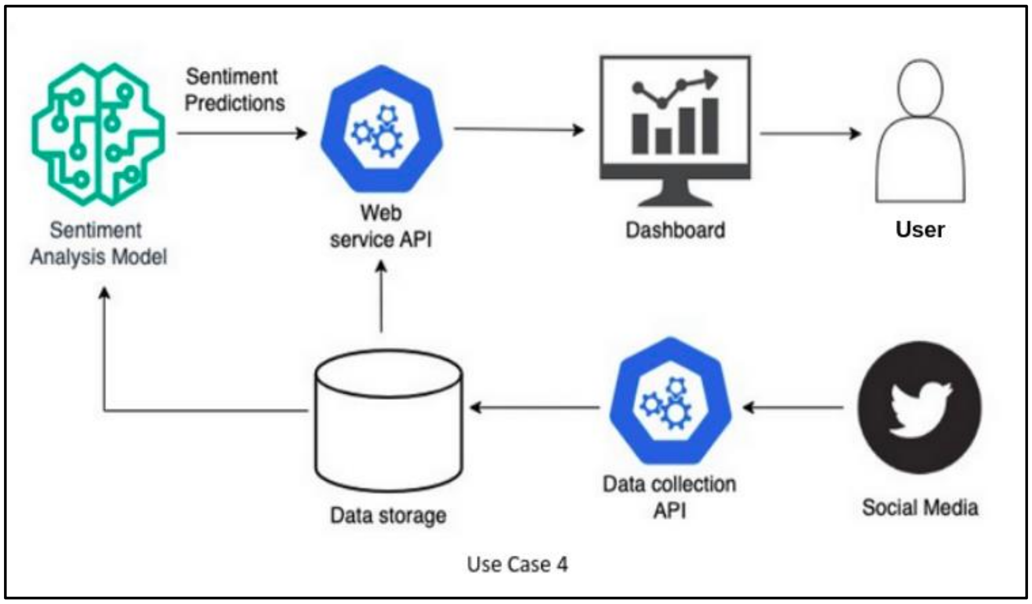
Sensor variables for location “DIŞHAT GELEN 5 NOLU BAGAJ ALIM”



Current temperatures for the location in the airport

3.1.4 Use Case Application #4 (INOSENS)

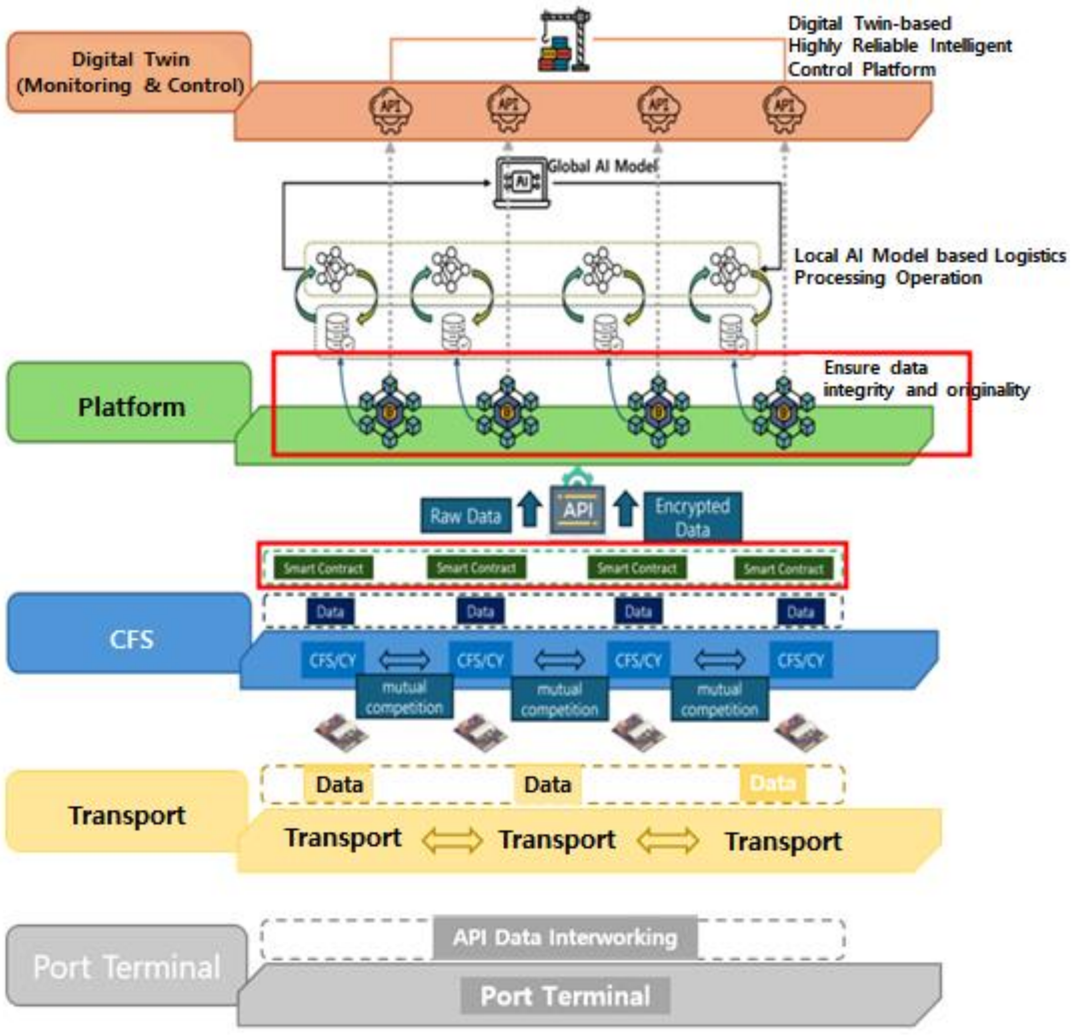
The XMedia2Analytics framework is the proposed solution which will be integrated into the SOCFAI platform. The goal of this framework is to provide businesses with analytics using text-based content from multiple online sources. Specifically, the framework will include AI-based solutions for sentiment analysis of text-based content about the organization (Izmir airport) and data visualizations for the user to identify the correlation between time and passenger satisfaction and how changes in passenger satisfaction are affected by flight arrival and departure delays.



The system consists of several components, including a dashboard for the user to view the results of the sentiment analysis model for each input text (comment, review, or social media post). The FastAPI framework will be used to develop the web service API, which can send the relevant information to the SOCFAI platform. The AI-based sentiment analysis model is necessary for predicting the sentiment. All the text-based content and relevant information collected from online sources will be stored in the data storage. The data collection API, as the name suggests, collects data (including text for sentiment analysis) from online sources.

3.1.5 Use Case Application #5

1) Overall Configuration


Digital Twin (Monitoring & Control)
Digital Twin-based Highly Reliable Intelligent Control Platform
API
Global AI Model
Local AI Model based Logistics Processing Operation
Platform
Ensure data integrity and originality
Raw Data
API
Encrypted Data
Smart Contract
Data
CFS
CFS/CY
mutual competition
Transport
Data
Port Terminal
API Data Interworking

In the Use Case Application #5, the main service scenario on "AI enabled Port Logistics Management" is implemented, and it focuses on exploring innovative solutions for intelligent container yard management and logistics flow between terminals within the port environment.

Particularly, in the AI powered port Logistics Management Platform, Advances in intelligent container yard management solutions and cross-terminal logistics flow management solutions are leading the way in modernizing port operations, collecting real-time data, quality control, and ensuring efficient logistics coordination.

These solutions use state-of-the-art technologies such as automatic scheduling, real-time tracking, predictive analytics, and digital twins to optimize container placement, data analytics, and shipping, as detailed in the requirements.

These systems leverage artificial intelligence to improve the efficiency of current operations, as well as preemptively address future demand and unexpected operational challenges.

Partnerships with KAIST, Inje University, and eINS S&C have been instrumental in integrating state-of-the-art research and practical applications to overcome traditional constraints such as distance, transportation, and operational bottlenecks. This collaboration has also highlighted the importance of data confidentiality and utilization control, which is critical in multilateral computing environments.

The logistics service platform collects and manages existing logistics information and various types of IoT data from logistics ports, warehouses, and transportation systems, and connects them by wired and wireless communication. As shown in the figure below, the functional framework supporting the 'AI-based intelligent logistics service platform' provides an optimized approach that combines these separate work functions with the necessary operational, security, monitoring, management, and logging functions of the port logistics data.

For use case 5, data trust solutions in partnership with blockchain technology make various types of logistics data safe and reliable in "intelligent logistics service platforms." And digital twin service solutions based on AI will be used for prediction and analysis in logistics processes.

In addition, the introduction of digital twin technologies paves the way for sophisticated scenario generation and simulation-based optimization, providing a dynamic and responsive framework for the decision-making process. The creation and maintenance of these systems represents a breakthrough in port logistics management, setting new standards for efficiency, reliability, and reliability in the industry.

2) Partners' Roles in the Development of Use Case Applications #5

A. KULS

Mid Version of Use Case Applications System Architecture are featured in the following capabilities.

- Data Pipeline System Architecture
- Zookeepers, Kafka, and Devezium open sources are used to build a data pipeline.
- initial versions of the system architecture used Java-based Zookeper to build a data pipeline.

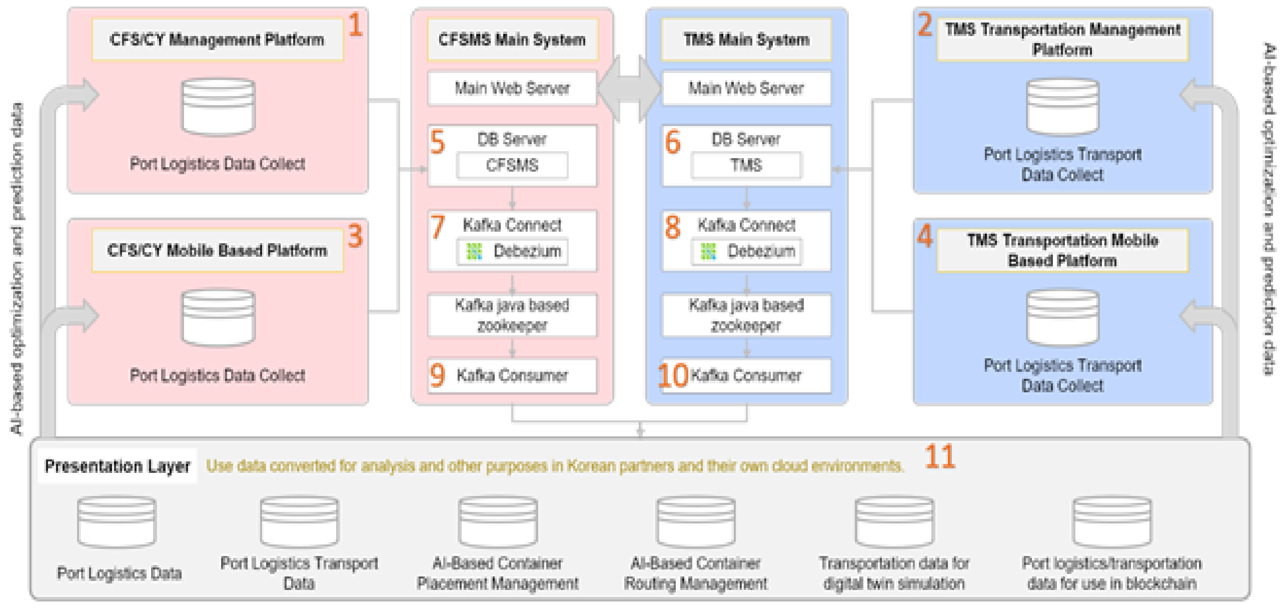
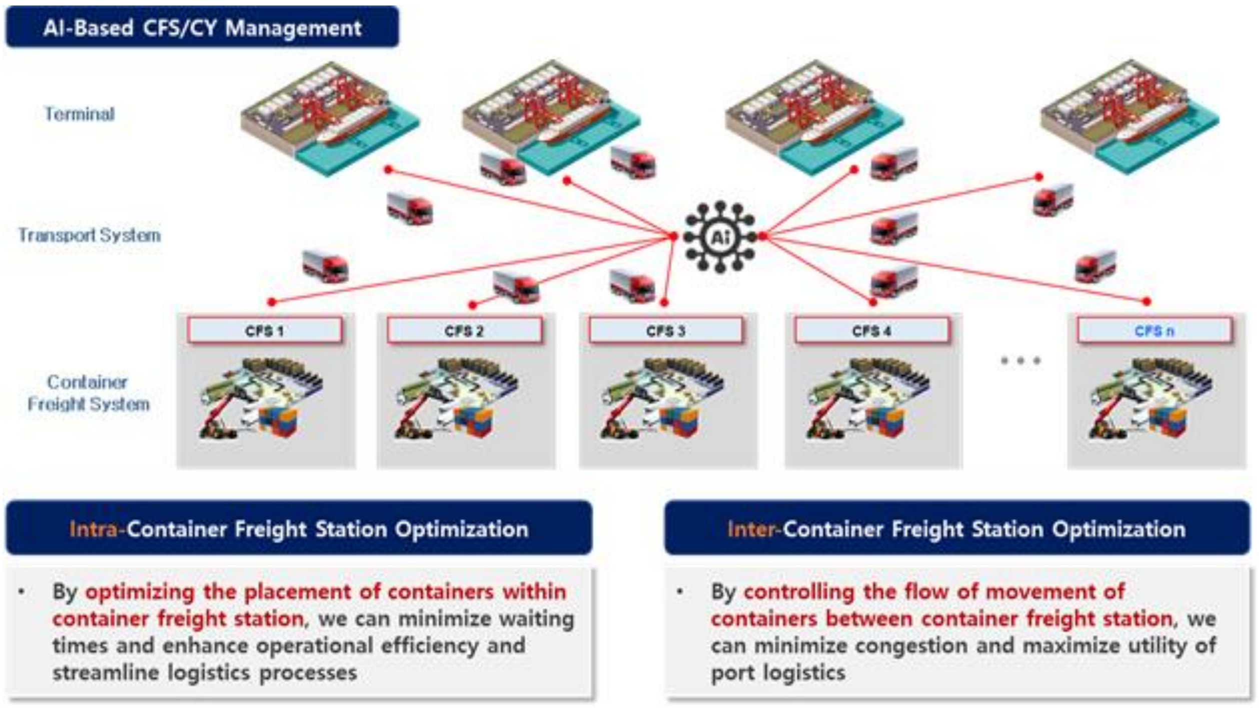

CFS/CY Management Platform
1
Port Logistics Data Collect
CFS/CY Mobile Based Platform
3
Port Logistics Data Collect
CFSMS Main System
Main Web Server
5 DB Server
CFSMS
7 Kafka Connect
Debezium
Kafka java based zookeeper
9 Kafka Consumer
TMS Main System
Main Web Server
6 DB Server
TMS
8 Kafka Connect
Debezium
Kafka java based zookeeper
10 Kafka Consumer
2 TMS Transportation Management Platform
Port Logistics Transport Data Collect
4 TMS Transportation Mobile Based Platform
Port Logistics Transport Data Collect
AI-based optimization and prediction data
AI-based optimization and prediction data
Presentation Layer
Use data converted for analysis and other purposes in Korean partners and their own cloud environments.
11
Port Logistics Data
Port Logistics Transport Data
AI-Based Container Placement Management
AI-Based Container Routing Management
Transportation data for digital twin simulation
Port logistics/transportation data for use in blockchain

Figure. Initial Version of Use Case Applications

B. KAIST

a) AI-based CFS/CY Management System Architecture


AI-Based CFS/CY Management
Terminal
Transport System
Container Freight System
AI
CFS 1
CFS 2
CFS 3
CFS 4
CFS n
Intra-Container Freight Station Optimization
Inter-Container Freight Station Optimization

- By optimizing the placement of containers within container freight station, we can minimize waiting times and enhance operational efficiency and streamline logistics processes

- By controlling the flow of movement of containers between container freight station, we can minimize congestion and maximize utility of port logistics

An AI-based CFS/CY (Container Freight Station / Container Yard) management architecture has been designed and implemented with the objective of optimizing container logistics within and across multiple container freight stations. This system is developed to enhance both intra- and inter-station operations through data-driven decision making and has been developed to support scenario-based analysis and operational optimization through simulation-driven approaches.

At the core of the architecture lies an AI coordination engine, which receives real-time data streams from terminal facilities, transport systems, and various CFSs. Based on these inputs, two levels of optimization are performed:

• **Inter-Container Freight Station Optimization**: Container flows between freight stations are controlled to prevent operational bottlenecks and distribute workloads evenly. Real-time decisions are made to redirect shipments to underutilized stations, thereby minimizing congestion and maximizing system-wide capacity utilization.

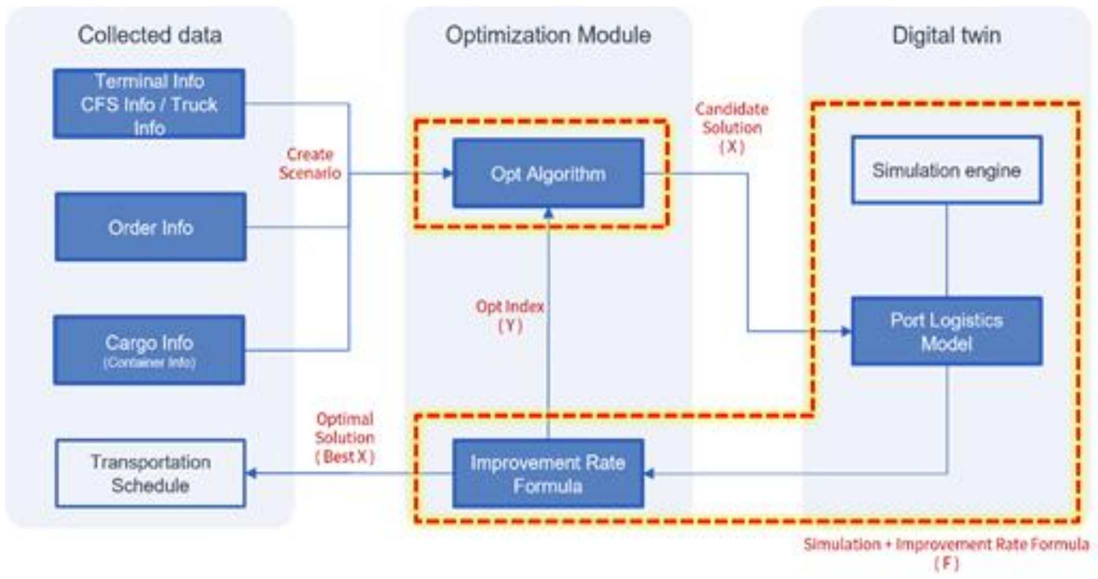
• **Intra-Container Freight Station Optimization**: Within each station, container stacking and relocation plans are optimized using AI algorithms. By considering departure priorities, spatial constraints, and equipment availability, the system minimizes waiting time and reduces inefficient internal movements.

The architecture has been designed to support intelligent decision-making by continuously analyzing operational data and logistics conditions. Optimization processes are executed in a distributed manner to enhance responsiveness and adaptability across various freight station environments.

C. eins S&C

a) Digital Twin Based Optimal Dispatch System Architecture

The Digital Twin-based optimal dispatch system is designed to provide optimal dispatch data based on transportation contracts registered on the port logistics platform. The system mainly consists of two parts: the Simulation Model and the Optimization Module.

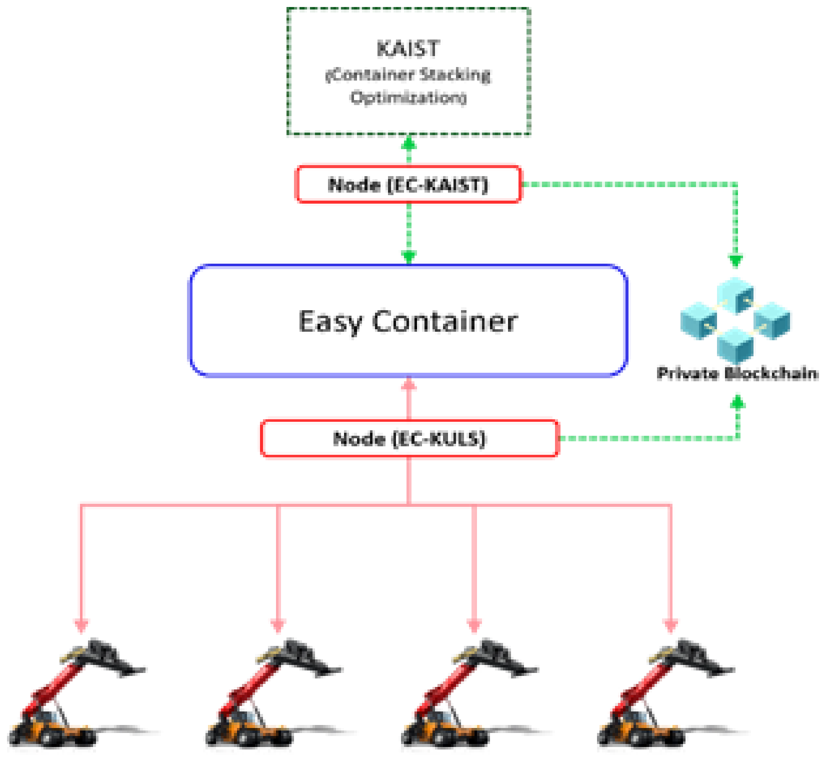


D. INJE University

Blockchain Recording Points

- At every API request/response, a hash (timestamp + who + source + destination + actual
- data) is recorded on the blockchain.
- If disputes arise, API transmission logs can be compared against blockchain records for verification.
- Access to blockchain nodes and monitoring UI is available for authorized members.
- Blockchain entries enable integrity checks and serve as audit logs during disputes or anomalies.

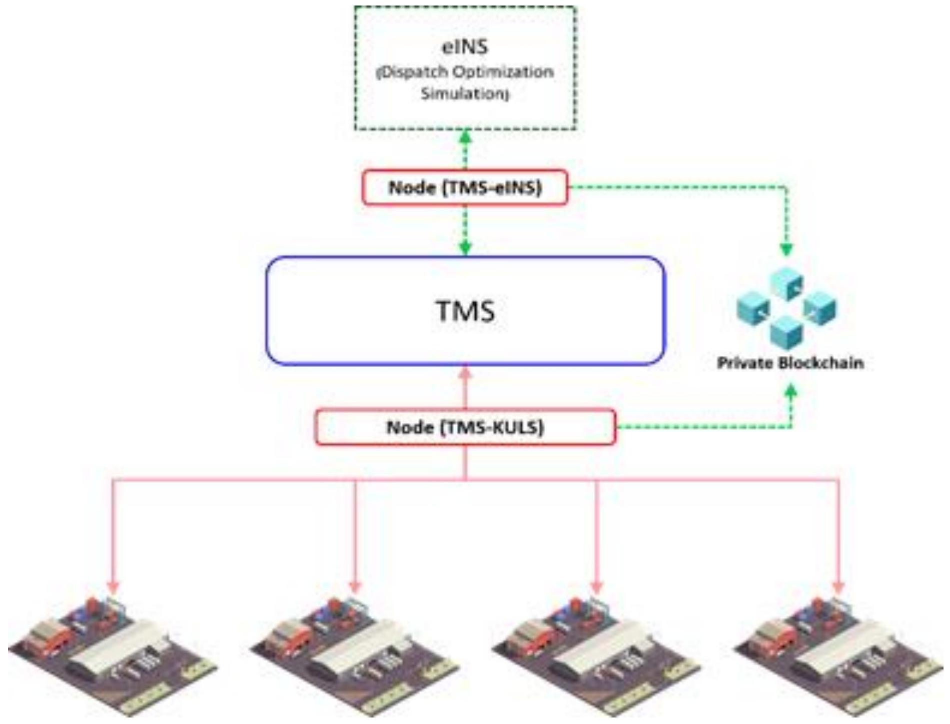
Container Data Sharing between Easy Container and KAIST



Data Flow Steps:

1. **[Step 1]** Request to KAIST:
   Easy Container requests information about containers scheduled for inbound movement to edge computing (used for optimization & AI processing).
2. **[Step 2]** Response from KAIST:
   KAIST responds with optimized stacking layout information for the inbound containers.
3. **[Step 1 again]** Request for outbound containers:
   Easy Container requests information about containers scheduled for outbound movement to edge computing.
4. **[Step 2 again]** Response from KAIST:
   KAIST provides priority layout optimization results for outbound containers.

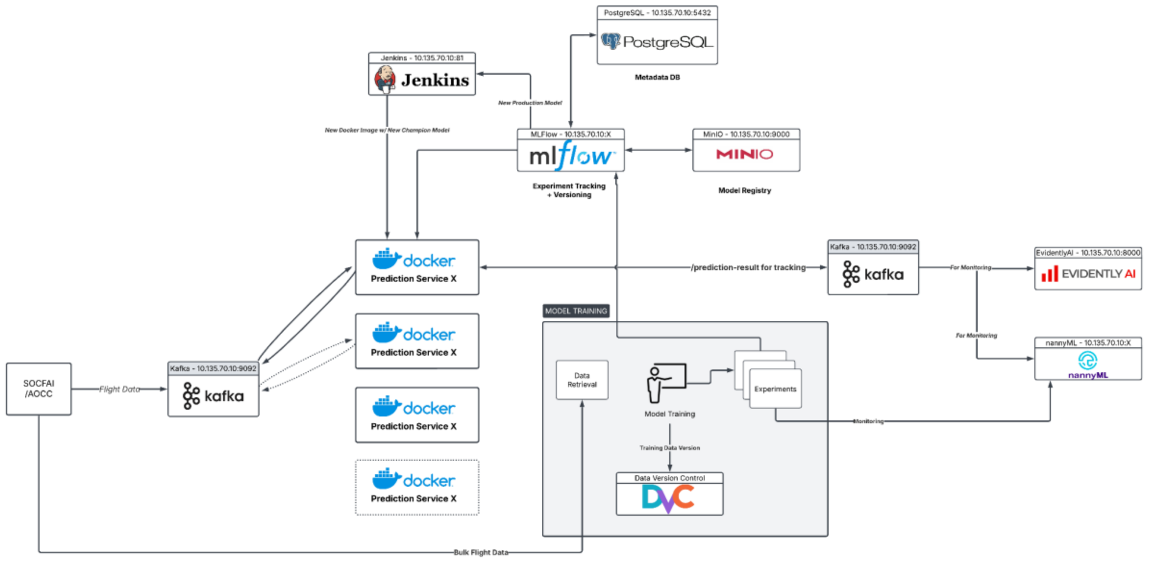
Transportation Data Sharing between TMS and eINS



Data Flow Steps:

1. **[Step 1]** Request to eINS:
   TMS sends data about inbound containers to edge computing nodes at eINS for optimization and digital twin simulation.
2. **[Step 2]** Response from eINS:
   eINS returns dispatch optimization results for inbound containers based on simulation.
3. **[Step 1 again]** Request for outbound containers:
   TMS sends data on outbound containers for digital twin-based dispatch optimization.
4. **[Step 2 again]** Response from eINS:
   eINS replies with dispatch simulation and optimization data for outbound containers.

3.1.11 Use Case Application #11



3.1.11.1 Data Ingestion and Versioning

The foundation of the pipeline begins with data ingestion. Bulk flight data is sourced from SOCFAI (AOCC), serving as the primary input for model training. This raw data, along with all subsequent transformed datasets, is meticulously managed by Data Version Control (DVC). DVC ensures that every iteration of the data used in experiments and training is versioned, providing crucial reproducibility and traceability. This allows developers to pinpoint the exact dataset configuration used for any given model, enabling effective debugging and model rollback if necessary.

3.1.11.2 Model Training and Experimentation

The core of the model development process resides within the "Model Training" block. Here, the versioned data is utilized for model training and experimentation. All outputs, metrics, and parameters from these experiments are meticulously logged and managed by MLflow. MLflow serves as a centralized platform for Experiment Tracking & Versioning, allowing data scientists to compare different model runs, track hyperparameters, and record performance metrics effectively.

3.1.11.3 Model and Metadata Storage

MLflow integrates seamlessly with backend storage solutions:

- PostgreSQL acts as the Metadata DB, storing all the run metadata, parameters, and metrics tracked by MLflow. This relational database ensures persistent and queryable storage of experiment details.

- MinIO functions as the Model Registry and Artifact Store, where the actual trained model files (e.g., serialized model weights, pipelines) are stored. MinIO, being an S3-compatible object storage, provides a scalable and reliable repository for all model artifacts.

3.1.11.4 Continuous Monitoring and Drift Detection

Post-deployment, continuous monitoring is critical for maintaining model performance.

- Kafka serves as the message broker, ingesting prediction results for tracking from the deployed prediction services.
- Evidently AI is employed for monitoring data and concept drift. By analyzing the prediction results and potentially input features from Kafka, Evidently AI helps detect shifts in data distributions or model behavior that could degrade performance.
- NannyML complements this by focusing on performance monitoring, particularly for the model's actual performance over time, often comparing it against baseline performance or detecting unseen data patterns. Both Evidently AI and NannyML are crucial for proactively identifying when a model needs retraining or recalibration due to changing real-world conditions.

3.1.11.5 Model Deployment

The deployment phase is orchestrated through Jenkins.

- Jenkins acts as the CI/CD orchestration tool. When a new production model is deemed ready (identified through MLflow's Experiment Tracking and potentially registered in its Model Registry), Jenkins is triggered.
- Jenkins then initiates the build and deployment of Docker containers (labeled as "Prediction Service X"). Each Docker container encapsulates a specific model, enabling containerized and isolated deployments. This ensures consistency across different environments and simplifies scaling.
- These Prediction Service X Docker containers expose the model for inference. Incoming flight data from SOCFAI (AOCC) is streamed via Kafka, which is then consumed by the prediction services to generate real-time forecasts.

3.1.11.6. Feedback Loop and Iteration

The entire architecture implicitly supports a feedback loop: insights from continuous monitoring (Evidently AI, NannyML) can inform data scientists about the need for new experiments or model retraining. This iterative process, facilitated by the robust infrastructure, ensures that the AI models remain performant and relevant over time.

In summary, this MLOps architecture provides a robust framework for managing the entire machine learning lifecycle, from data to deployment and continuous improvement, leveraging industry-standard tools for efficiency, reliability, and scalability.

3.2 Functional Implementation

3.2.1 Use Case Application #1 (INOSENS)

For use case 1, the Velodyne Puck sensors are installed in Adnan Menderes Airport terminal and they gather data from the passengers who pass through the terminal. The Velodyne sensor uses point cloud processing to gather visual information from the physical environment. For passenger density estimation, we count people in three different levels. Level 1 provides individual people counts. Level 2 provides a 1 to 10 ratio of occupation. Level 3 provides three categories: high, very high, and extremely high. We also use point cloud processing data and deep learning for passenger flow monitoring.

Type of Requirement	Description
Integration	The Lidar sensor is Velodyne VP-16.
Integration	Lidar sensors are installed in Adnan Menderes Airport.
Integration	The Lidar sensor should be connected to a computer for collecting data, which is necessary when using the deep learning model for people detection, people counting, and people density tracking.
Infrastructure	People detection, people counting, and people density tracking will be performed using a deep learning model.
AI Requirements	People detection will be performed on data from the Lidar sensors using a computer vision deep learning model.
AI Requirements	People counting will be performed, and it is dependent on the results from the people detection task.
AI Requirements	People density tracking will be performed, and it is dependent on the results from the people counting task.
AI Requirements	Passenger flow monitoring will be performed, and it is dependent on the results from the

	people detection and the people density tracking tasks.
Other Functional Requirements	The data collected from the Lidar sensors will be stored in a local server in INOSENS or cloud storage server.
Other Functional Requirements	The computer vision deep learning models will be trained and tested using a local or cloud computing server.
Other Functional Requirements	There will be a system for passing on the people detection, people count, people density tracking, and people flow monitoring information to TAV Tech.

3.2.2 Use Case Application #2

In this section, the features and services which are implemented in *AI-Based Baggage Analytics and Optimization Application* are given in the following tables.

Table. Software Service and Modules of Use Case #2 Application

#	Modules & Services	Description
1	User Interface Design	This part is regarding to the development of graphical user interface (GUI) modules of the application for monitoring of Real-Time Statistics and Prediction pages
2	AI/ML services	These services aim to develop AI algorithms for predictions regarding flight and baggage data.
3	Management service	This service is responsible for interfacing between real-time Dashboard and other services.
4	Communication service	Transform raw baggage-flight data into clean data and integrate into database
5	Data Analysis service	Responsible real time statistical SQL calculations over database
6	Database	This part is responsible for data storage operations.

Table. Feature List and Supported Functionalities of Use Case #2 Application

#	Main Features	Supported Functionalities	Description
1	Real-time statistics page	Deleted Bag	Removed baggage count
		Transfer Passenger Ratio	Transfer passenger count over total passenger with baggage
		Rejected Bag	Unauthorized baggage count
		Total Bag Weight	Total weight of delivered baggage
		Average Bag Weight	Mean weight of each delivered baggage
		Waiting Bag Ratio	Unprocessed baggage count over total baggage
		Average Bag	Mean delivered bag count of each passenger
		Average Process Time	Mean process time of each delivered baggage
		Multi Bim Ratio	Changed baggage count over total baggage
		Flight Base Dissimilarity	Baggage count over average baggage from previous flights
		Transfer Bag Ratio	Transfer baggage count over total baggage
		Failed Transfer Bag	Unauthorized transfer baggage count
		Baggage Count	Total number of delivered baggage
2	Predictions page	Minute Baggage Prediction	Expected baggage count in fifteen minutes
		Hourly Baggage Prediction	Expected baggage count in one hour

		Daily Bagge Prediction	Expected baggage count in one day
		Flight Delay Prediction	Expected flight delay
		Mean - Standard Deviation Calculation	Average baggage count of previous flights – Difference between expected and average

Real-Time Dashboard:

The real-time dashboard of AI-based baggage analysis and optimization solution consists of two main pages: Real-Time Statistics and Predictions.

- **Real-Time Statistics Page:** The Real-Time Statistics page features a prominent central time series line chart that displays the temporal distribution of baggage counts, allowing operators to monitor baggage flow patterns over different time periods (1 Day, 1 Week, or 1 Month). This dynamic visualization is surrounded by eight key performance indicators showing critical metrics such as deleted bags, rejected bags, transfer bag ratio, and average process time. Below the main chart, four detailed bar charts provide deeper insights into multi-BIM ratio, flight-based dissimilarity, transfer bag ratio, and failed transfer bags across different airports.

- **Prediction Page:** The Predictions page incorporates four sophisticated AI/ML models that provide baggage handling forecasts at different time intervals - minute-level, hourly, and daily baggage predictions, along with flight delay predictions. The end-users can customize their view through various filters including time range, flight categories, arrivals, flight status, and airlines, with all visualizations updating dynamically based on the selected parameters.

The system's real-time monitoring capabilities and predictive analytics make it an essential tool for optimizing airport baggage operations and flight management. Real Time Statistics and Prediction pages are illustrated in the following figures respectively.

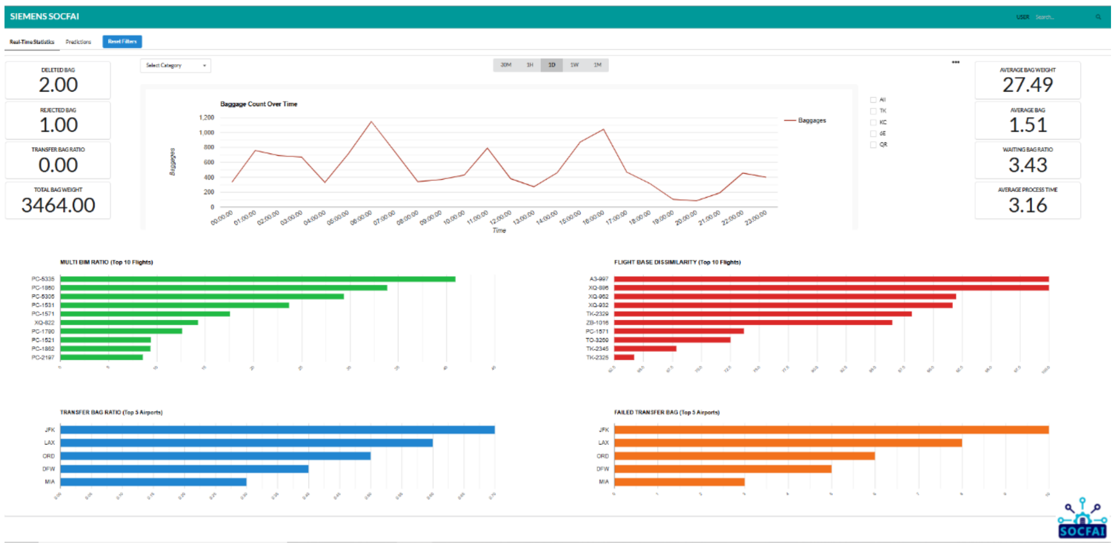
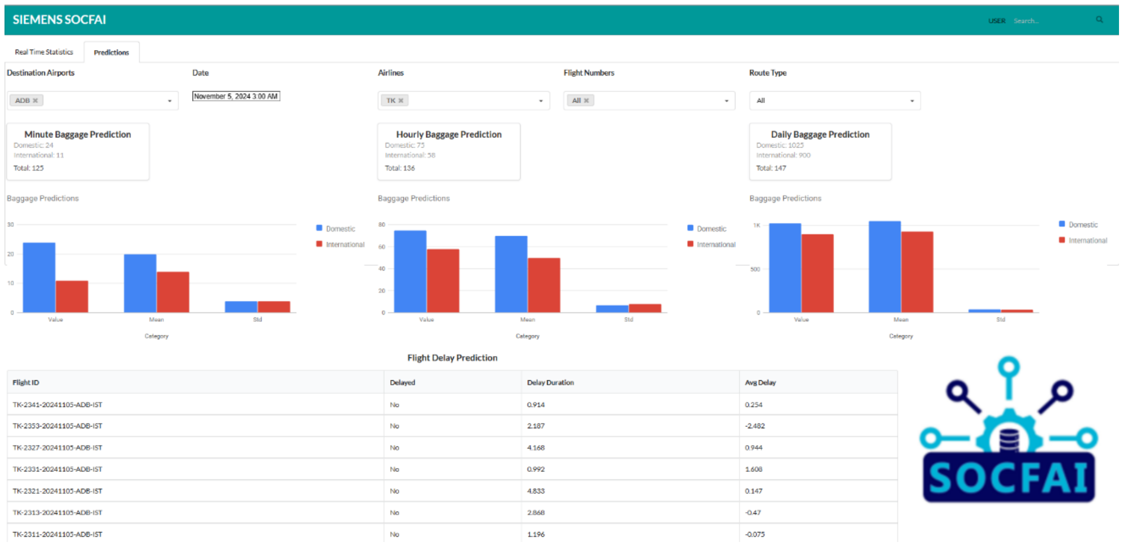


Figure. Real Time Statistics Page



Destination Airports: ADB
Date: November 5, 2024 3:00 AM
Airlines: TK
Flight Numbers: All
Route Type: All

Minute Baggage Prediction
Domestic: 24
International: 11
Total: 125

Hourly Baggage Prediction
Domestic: 73
International: 58
Total: 136

Daily Baggage Prediction
Domestic: 1025
International: 900
Total: 147

Flight Delay Prediction

Flight ID	Delayed	Delay Duration	Avg Delay
TK-2341-20241105-ADB-IST	No	0.914	0.254
TK-2353-20241105-ADB-IST	No	2.187	-2.482
TK-2327-20241105-ADB-IST	No	4.168	0.944
TK-2331-20241105-ADB-IST	No	0.992	1.608
TK-2321-20241105-ADB-IST	No	4.833	0.147
TK-2313-20241105-ADB-IST	No	2.868	-0.47
TK-2311-20241105-ADB-IST	No	1.196	-0.075

Figure. Predictions page

AI/ML services:

In this subsection, AI/ML services and algorithms which are implemented for baggage (daily, hourly and pattern) and delay (flight delay) predictions are explained in detail.

- **Daily prediction:**

One of the challenges in predicting daily baggage numbers is the presence of many categorical features alongside numerical ones. This requires preprocessing operations before feeding the data into the prediction model. As the number of operations increases in machine learning models, time and computational power requirements also increase proportionally. Therefore, achieving higher accuracy with fewer operations is a critical goal for these models. The CatBoost model offers improvements that reduce the number of operations required for handling categorical features through its innovative ordered target encoding method. Unlike most models, CatBoost can directly use non-numerical features in both training and inference phases, utilizing automatic feature combination generation and built-in parameter tuning capabilities. This is one of the main reasons for choosing CatBoost as the prediction model for this task. CatBoost is a gradient boosting model based on decision trees, supporting both GPU acceleration and multiprocessing for efficient handling of large datasets. Trees are built sequentially, with each tree focusing on correcting the errors made by previous trees, while implementing ordered boosting to prevent target leakage. This process continues iteratively until the final tree produces the most accurate result. Additionally, its algorithm handles missing values better than other models while reducing overfitting through multiple techniques including random permutations and fine-grained optimization. CatBoost often requires minimal hyperparameter tuning and provides robust performance out of the box.

These advantages of CatBoost with the highest accuracy among many machine learning and deep learning models makes it a prediction model for the number of daily baggage. It is also a lightweight model with low size. Its training time for large sizes of data is significantly short. It demonstrated robust performance with interpretable feature importance metrics, offering an optimal balance between accuracy, training speed, and generalization.

- **Hourly prediction:**

Dataset that is used for hourly model training includes one year (2023/08/24 – 2024/07/18) baggage (v_bag) and flight records(brs_flight) for all TAV airports. The data is transformed into clean data v7.2 (28 columns and 10918954 rows) by removing outliers, repetitions and performing morphological operations. The main features that are focused on hourly prediction are: FLIGHT_CODE, FLIGHT_DATE, FLIGHT_DELAY, HOUR_BEFORE_FLIGHT, FLIGHT_CLASS, AIRLINE_CODE, DEP_AP_CODE, ARR_AP_CODE, IS_HOLIDAY, IS_WEEKEND, YEAR_SEASON, DAY_PART, BAG_NUMBER. Flight ticket time (SCHEDULED_TIME) is used for FLIGHT_DATE. FLIGHT_DELAY is the time difference between ACTUAL_TIME and SCHEDULED_TIME in minutes. IS_HOLIDAY, IS_WEEKEND, YEAR_SEASON, DAY_PART features are extracted from FLIGHT_DATE. HOUR_BEFORE_FLIGHT is used to divide baggage records into one-hour windows so that hourly baggage pattern is extracted.

The ideal baggage pattern of the flight should have zero flight delays with high baggage. So that the training data set is filtered into 3 sets: 1-low flight delay, high total baggage, 2- mid delay, mid baggage, high delay low baggage. If the first set has enough records (at least 25 percent of the total set) it is used during the training. If the first set is not enough, the second set is used. If this is also not enough the last set is used.

Three steps of machine learning are used for hourly prediction: 1- unsupervised classification(mean-shift) 2- supervised classification (decision tree) 3-regression (polynomial regression). The baggage patterns are transformed into FLIGHT_CLASS by using mean shift unsupervised classification. 60% bandwidth similarity is used during training. FLIGHT_CLASS that comes from unsupervised classification is used as target label 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 AIRLINE_CODE, DEP_AP_CODE, ARR_AP_CODE. After each baggage pattern is classified into the flight classes the polynomial regression is used to model each baggage pattern as a polynomial expression.

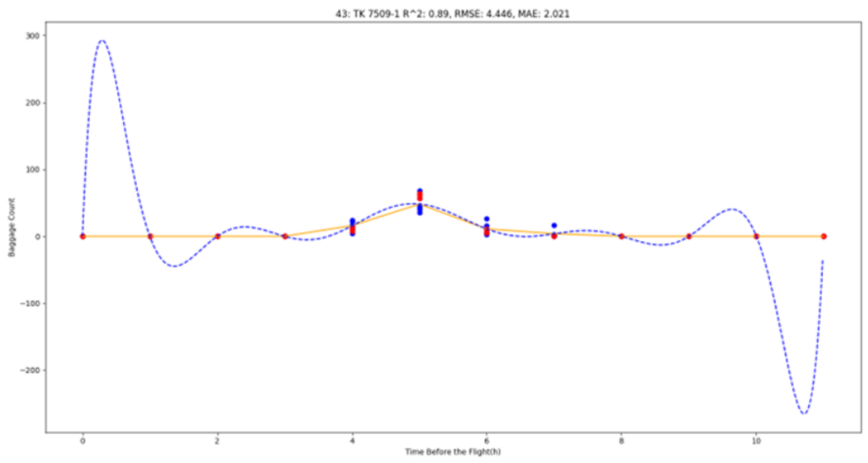


Figure. Hourly Baggage Prediction using Polynomial Regression Model

For example, some TK 7509 flights have baggage patterns that is classified as Class-43 by mean-shift are linked to that flight class by decision tree and modeled as the blue polynomial expression by polynomial regression and normalized to orange expression which can be used to predict the baggage arrival up to 12 hours before the flight.

So, the basic idea for hourly baggage prediction is flight info (i.e. TK 7509, TK-ADB-SAW) and flight date (i.e. holiday, weekend, morning, summer) is transformed into flight class by using decision tree model. After that the normalized polynomial regression model of this flight class is used to predict baggage count that will happen in one hour.

70% of the filtered data (low flight delay, high baggage as much as possible) is used for training and 30% of the overall set is used for testing the models. As a result, R^2: 0.94, RMSE: 8.243, MAE: 2.598 is achieved for hourly baggage prediction of 2563 distinct flight codes, from 15 different airports.

- **Pattern prediction:**

The dataset used for baggage pattern prediction originally consisted of ~11M records with 25 features. A multi-step filtering process was applied to narrow down the scope: first by setting

SYSTEM_AIRPORT = "ADB", then FLIGHT_STATUS = "DEPARTED", which resulted in 2,3M records. A new feature called TIME_WINDOW was generated to aggregate baggage counts in 15-minute intervals between BIM_CREATE_DATE and SCHEDULED_TIME, grouped by FLIGHT_CODE and SCHEDULED_TIME. After this aggregation, the dataset was reduced to 68K records with 15 features. Feature types include numerical values such as TIME_MONTH, TIME_DAY, and WEEKDAY, and categorical variables such as SEASON, TIME_OF_DAY, and CATEGORY. The target variable is BAG_NUMBER. Preprocessing involved scaling of numerical features and label encoding for categorical variables, ensuring compatibility across all model types. Three distinct AI model views have been explored: Time-Series, Deep Learning, and Machine Learning.

The time-series modeling approach focuses on capturing temporal trends in baggage data using historical patterns. Classical models such as ARIMA, SARIMA, and Prophet were implemented alongside neural time-series models like LSTM (Long Short-Term Memory). These models were trained using sequences of time-ordered data, taking into account seasonal, weekly, and hourly fluctuations. Feature engineering remained consistent across views to allow fair model evaluation. Based on performance benchmarks—including accuracy and forecasting stability, LSTM emerged as the most reliable model for this context. Its ability to model long-term dependencies and adapt to sequential dynamics made it ideal for predicting short-term baggage flow in high-traffic environments.

In the deep learning perspective, three model architectures were explored: Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), and a hybrid CNN+DNN architecture. The DNN model, applied on the cleaned and engineered dataset, demonstrated superior performance. The final DNN architecture was able to learn from both linear and non-linear interactions among the 15-minute interval data, offering high flexibility in adapting to operational variations.

The machine learning approach tested a wide variety of regression algorithms including Random Forest, Gradient Boosting, LightGBM, AdaBoost, Bayesian Ridge, and Elastic Net, among others. All models used the same dataset version and feature transformations to ensure consistent evaluation. Through hyperparameter tuning and cross-validation, Random Forest Regressor was identified as the most effective model. It demonstrated robust performance with interpretable feature importance metrics, offering an optimal balance between accuracy, training speed, and generalization.

- **Flight delay prediction:**

Flight delay prediction presents unique challenges due to its complex feature landscape, combining diverse data types including temporal, meteorological, operational, and categorical variables. This multifaceted nature of flight data necessitates sophisticated preprocessing and feature engineering approaches to capture intricate relationships between variables effectively. The challenge lies not only in handling this diverse feature set but also in developing models that can process large-scale aviation data efficiently while maintaining high prediction accuracy. Our comprehensive analysis evaluated seven distinct machine learning approaches, including tree-based ensembles, deep neural networks, and instance-based learning methods.

XGBoost's architecture proves especially suitable for flight delay prediction through its advanced handling of both numerical and categorical features, coupled with its ability to capture complex interactions through gradient boosting. The model's tree-based structure naturally accommodates the hierarchical nature of aviation data, while its regularization techniques help prevent overfitting - a common challenge when dealing with highly variable flight delays. Our implementation achieved a test R^2 Score of 0.5375 with the integration of the novel dissimilarity ratio, representing a significant improvement over traditional approach.

LightGBM emerged as another powerful contender, demonstrating exceptional balance between performance and generalization. With a test R^2 Score of 0.5126, it showed a particular strength in maintaining stable performance across different operational conditions. The model's efficiency in handling large-scale datasets and robust performance with minimal parameter tuning makes it an excellent choice for production environments where reliability is crucial.

Traditional tree-based ensembles, including Random Forest and Gradient Boosting, showed consistent but more moderate performance improvements. Random Forest achieved a test R^2 Score of 0.4461, while Gradient Boosting reached 0.4434, both demonstrating reliable prediction capabilities with better interpretability compared to more complex models. AdaBoost, while showing modest improvements with the dissimilarity ratio integration, achieved a test R^2 Score of 0.3149, highlighting its utility in specific operational contexts.

The Deep Neural Network implementation, despite its sophisticated architecture, achieved a test R^2 Score of 0.2742, revealing limitations in handling structured aviation data compared to tree-based approaches. The K-Nearest Neighbors model, with a test R^2 Score of 0.1774, demonstrated the challenges of instance-based learning in the complex domain of flight delay prediction.

The models' effectiveness is further enhanced by their ability to handle missing values - a common occurrence in flight data - and their robust feature importance calculation capabilities, which provide valuable insights into delay factors. The parallel processing capabilities and optimization techniques, particularly in XGBoost and LightGBM, make them suitable for real-time applications in airport operations, where quick and accurate predictions are essential. These models demonstrated exceptional performance in capturing both seasonal patterns and unexpected delay factors, while maintaining computational efficiency during both training and inference phases.

These comprehensive results, combined with the models' ability to process large-scale aviation datasets efficiently and provide interpretable results, establish a robust framework for flight delay prediction. The success in balancing accuracy with computational efficiency, coupled with the ability to handle the complex, multi-dimensional nature of flight data, makes these approaches particularly valuable for practical applications in aviation operations management.

3.2.3 Use Case Application #3

The Use Case 3 - IoT Portal requirements are:

Type of Requirement	Requirement
UX/UI	**Data Visualization:** All sensor data should be visualized on IoT platform and accessible from web in a user-friendly fashion. Visualizations should reflect data fusion and advanced data analytics outcome.
UX/UI	**Detail Visibility:** Sensor Data details (json, xml content details) should be visible to IoT operators for real time control and troubleshooting purposes
UX/UI	**Custom Dashboard Creation:** End-users or customer should create customized dashboards on IoT platform serving to different use-cases.
Integration	**IoT Protocol Support:** Industry standard IoT protocols should be supported on IoT platform. Initially MQTT and HTTP should be supported. COAP support should be a nice to have feature.
Integration	**E2E Security:** Sensors and GWs should communicate with IoT platform in a secure way. For HTTP protocols, HTTPS should be supported. For MQTT, TLS v1.2 should be supported.
Integration	**3rd Party Data Storage System Integration:** The data processed at IoT platform should be transmitted to different types of DBs (BigQuery, Maria10, MSSQL, mongo, etc..). RabbitMQ or Apache Nifi like solutions should be implemented on IoT platform.
Functional	**Rule and Event Based Triggering:** The system should be able to send out notification to other system when a pre-defined certain threshold is exceeded on the incoming data. The notification might be through SMS, E-mail or webhook methods
Functional	**Device Management:** All data sources (sensors, Gateways or other 3rd party system) should be configured on IoT platform. The configuration should include at least QoS level, IP/non-IP connection type, IP address, Definition
Functional	**Multi-Tenancy:** IoT platform should support different customers, use-cases on the same system isolated from each other.
Functional	**3rd Party Application Support:** IoT platform should allow 3rd party service deployment through dockerized micro-services

Functional	**Data Analytics:** Incoming data should be pre-processed to make it ready for advanced data analytics through additional scripts or applications deployed on top of existing microservices
Infrastructure	**Cloud and On-Prem Support:** The system should serve from both Cloud based systems (like Azure, AWS or private clouds) and on-premise.
Infrastructure	**Scalability:** The system should be scaled up based on the number of request and triggered APIs
Infrastructure	**Docker Based Cluster Architecture:** IoT platform services should be dockerized and orchestrated by Kubernetes. All services should be deployed as microservices.
Non-Functional	**Creation of Different Domains:** Different domains should be created to handle different use-case or different types of sensors/GWs
Non-Functional	**Data Traffic Optimization:** IoT platform should schedule device updates or data transmission times to avoid traffic congestion on the network.

3.2.4 Use Case Application #4 (INOSENS)

Required Features and Services for Framework

The goal of the XMedia2Analytics framework is to enable the implementation of the following features and services.

UX/UI Functional Requirements

Type of Requirement	**Description**
UX/UI	For each social media post, the sentiment analysis model returns a prediction.
UX/UI	On the dashboard, there is a list containing each social media post, date and time it was posted, language, and predicted sentiment.
UX/UI	On the dashboard, there is a line graph showing the number of negative posts, positive posts, and total posts per day. This can also be done for another time period (such as month). Line graph can show the past

	X number of days (or weeks or months).
UX/UI	Underneath the previous line graph is another line graph that uses information from AODB. It plots the number of delayed flights per day. This can also be done for hours. Line graph can show the past X number of days and it can show the upcoming 24 hours.
UX/UI	The user is able to modify each sentiment if the model detected the wrong sentiment.

Integration and Infrastructure Requirements

Type of Requirement	**Description**
Integration	Data is collected from social media platforms such as Twitter.
Integration	The web application will use AODB data to show information on the frontend of the sentiment analysis platform.
Infrastructure	UI dashboard will display social media information, predictions, and analytics to the user.
Infrastructure	Web service API will display information on the frontend.
Infrastructure	Data collection API will retrieve the social media posts and other similar text such as comments and reviews.
Infrastructure	The sentiment analysis model will use deep learning to classify the social media posts.

AI and Other Functional Requirements

Type of Requirement	**Description**
AI	Sentiment analysis will be performed on text-based social media posts using an NLP deep learning model.

AI	Language detection will be performed using an advanced deep learning model.
Other	The social media posts will be stored in a local or cloud storage server.
Other	The sentiment analysis deep learning models will be trained and tested using a local or cloud computing server.
Other	The UI dashboard must have a login and logout authentication system.

Real-Time Dashboard

The dashboard is projected to have three different tabs. They will show the results of the sentiment analysis model on our data as well as visualizations involving these results.

Prediction of Sentiment Analysis Results

A page that shows the results of the sentiment analysis results in the form of an interactive table. Includes text, predicted sentiment, date and time, detected language. Also includes a feature that will allow the user to change the sentiment if the predicted sentiment is wrong.

Visualization of DateTime vs. Sentiment and Language

Data visualizations that show the correlation between DateTime and predicted sentiments and the correlation between DateTime and language appear in this page. Y-axis represents the number of comments per time period with a given prediction (for sentiment or language), X-axis shows the time period (such as day, week, or month), and the color-coded lines represent each sentiment or language.

Correlation between Changes in Sentiment and Delays

This page shows the recent flight delays in the form of a table and a data visualization indicating the changes in sentiment corresponding to recent delays.

Data Collection Features

Uploading Datasets

Inosens already has a relatively small dataset which contains data extracted from various online sources. The user of the platform might also want to upload existing data. Thus, this application will also require an option to upload existing data into the system. The user should then be able to view the results in the dashboard, including sentiment analysis prediction results and the relevant data visualizations.

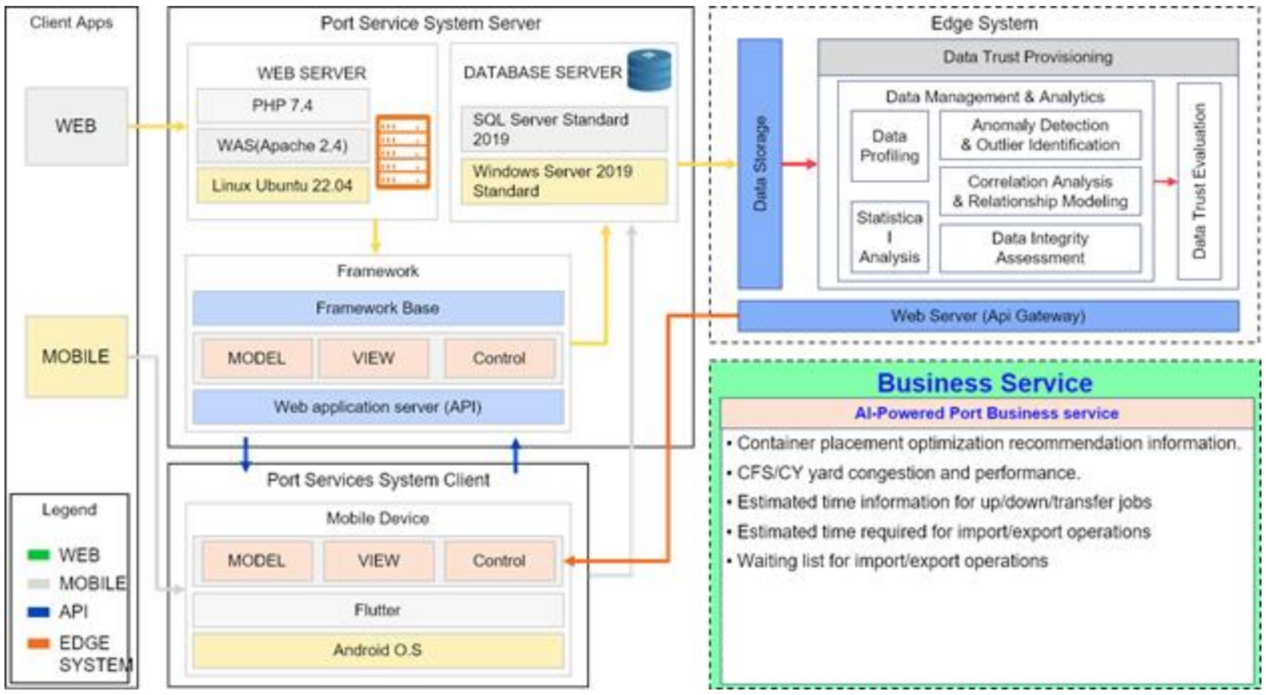
Batch Processing

Because the data will be mainly extracted from online sources, we should constantly be uploading data from the designated sources. This can be done in batch processing or stream processing. For practical purposes, batch processing seems to be the preferable option.

3.2.5 Use Case Application #5

1) Implementation of CFS/CY Port Logistics Data Platform (KULS)

a) port logistics data platform Architecture


Client Apps
WEB
MOBILE
Legend
WEB
MOBILE
API
EDGE SYSTEM
Port Service System Server
WEB SERVER
PHP 7.4
WAS(Apache 2.4)
Linux Ubuntu 22.04
DATABASE SERVER
SQL Server Standard 2019
Windows Server 2019 Standard
Framework
Framework Base
MODEL
VIEW
Control
Web application server (API)
Port Services System Client
Mobile Device
MODEL
VIEW
Control
Flutter
Android O.S
Edge System
Data Storage
Data Trust Provisioning
Data Management & Analytics
Data Profiling
Anomaly Detection & Outlier Identification
Correlation Analysis & Relationship Modeling
Statistical Analysis
Data Integrity Assessment
Data Trust Evaluation
Web Server (Api Gateway)
Business Service
AI-Powered Port Business service
• Container placement optimization recommendation information.
• CFS/CY yard congestion and performance.
• Estimated time information for up/down/transfer jobs
• Estimated time required for import/export operations
• Waiting list for import/export operations

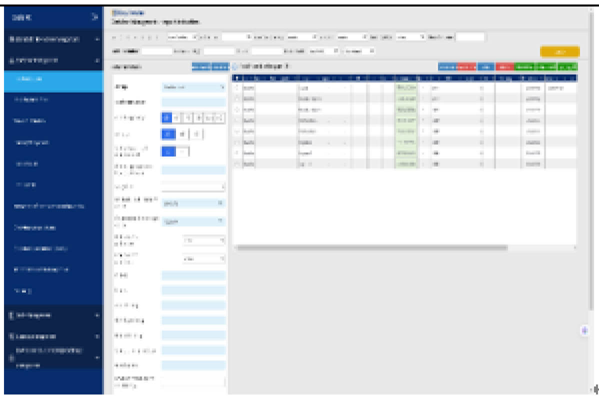
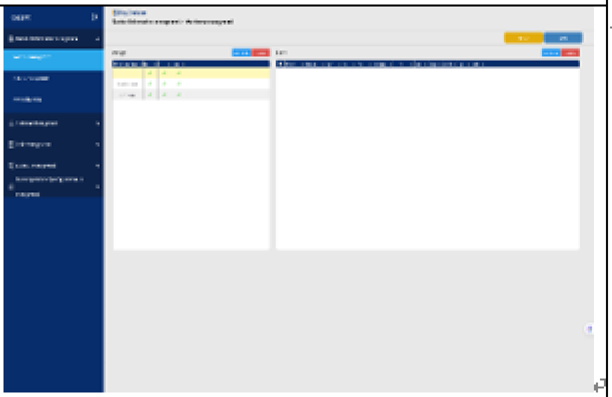
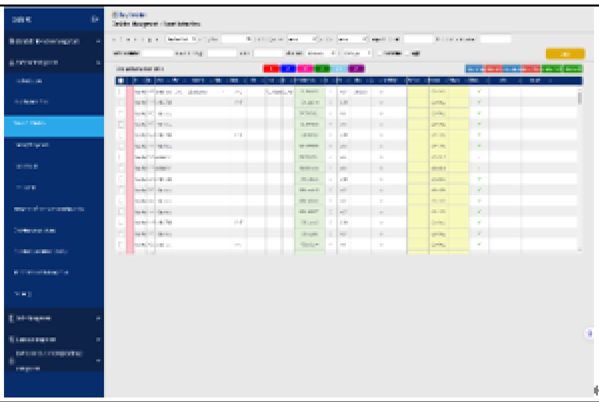
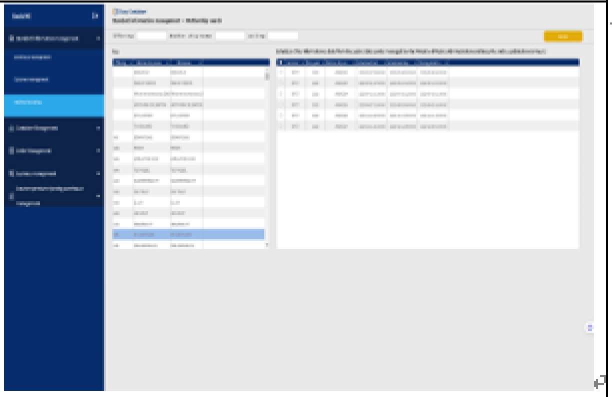
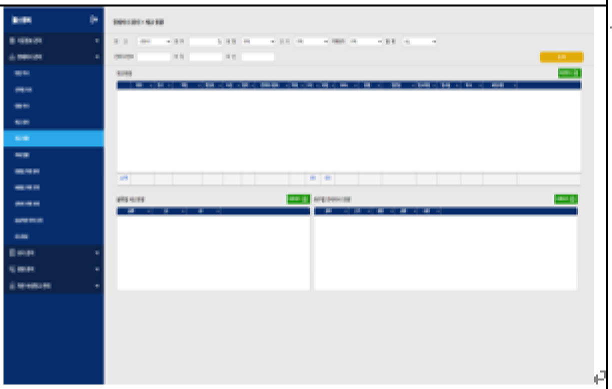
b) The CFS/CY management system that collects and manages port logistics information is as follows.

The CFS/CY management system that collects and manages port logistics information is as follows.

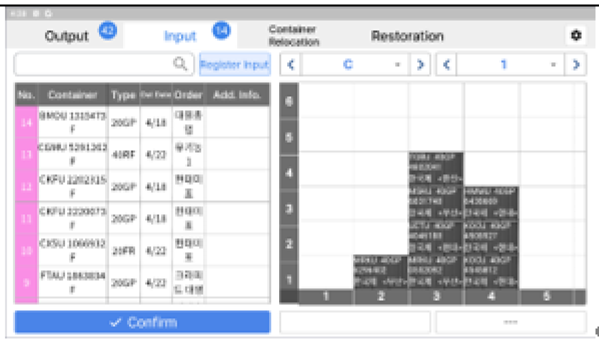
- Register and manage container import/export orders
- Manage container loading/ unloading/transfer processing details.
- Web and mobile device-based environments
- Management of container import/export performance,

Container real-time inventory management,

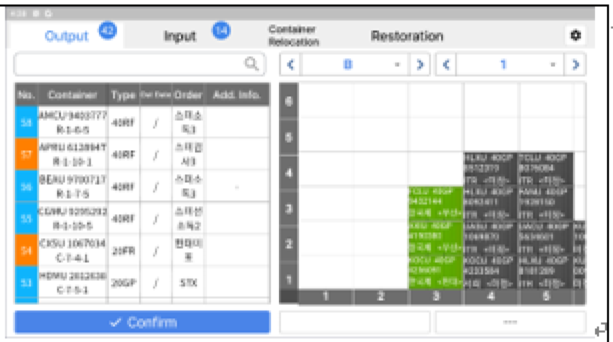
Web device-based environments

Web-device-based environments	
	
	
	

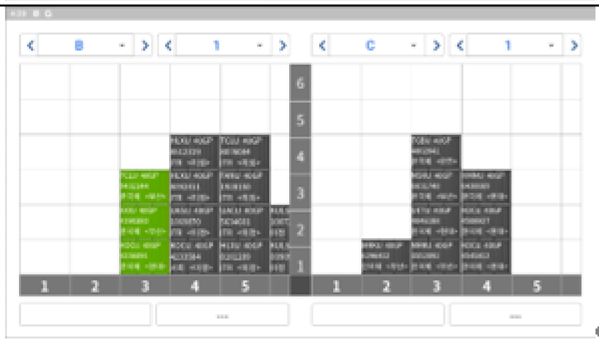
mobile device-based environments

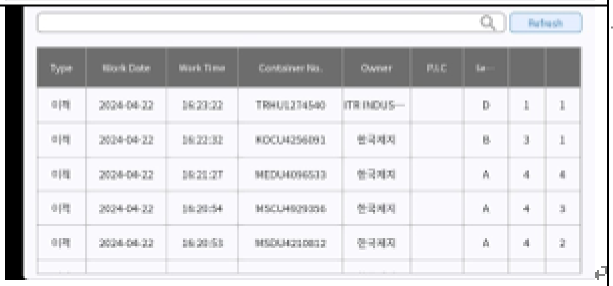


No.	Container	Type	[illegible]	Order	Add. Info.
14	BMOU 1315473 F	20GP	4/18	[illegible]	
13	CGMU 5291262 F	40RF	4/22	[illegible]	
12	CKFU 2202315 F	20GP	4/18	현대미포	
11	CKFU 2220073 F	20GP	4/18	현대미포	
10	CKSU 1066912 F	20FR	4/22	현대미포	
9	FTAU 1863834 F	20GP	4/22	[illegible]	



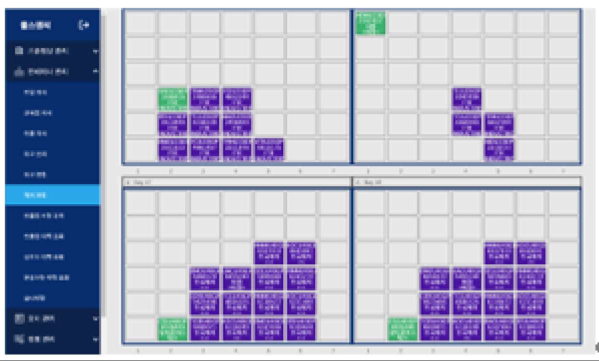
No.	Container	Type	[illegible]	Order	Add. Info.
58	AMCU 9403777 R-1-6-5	40RF	/	[illegible]	
57	APRU 6128847 R-1-10-1	40RF	/	[illegible]	
56	BEAU 9700717 R-1-7-5	40RF	/	[illegible]	
55	CGMU 9295292 R-1-10-5	40RF	/	[illegible]	
54	CKSU 1067034 C-7-4-1	20FR	/	현대미포	
53	HDMU 2612636 C-7-5-1	20GP	/	STX	



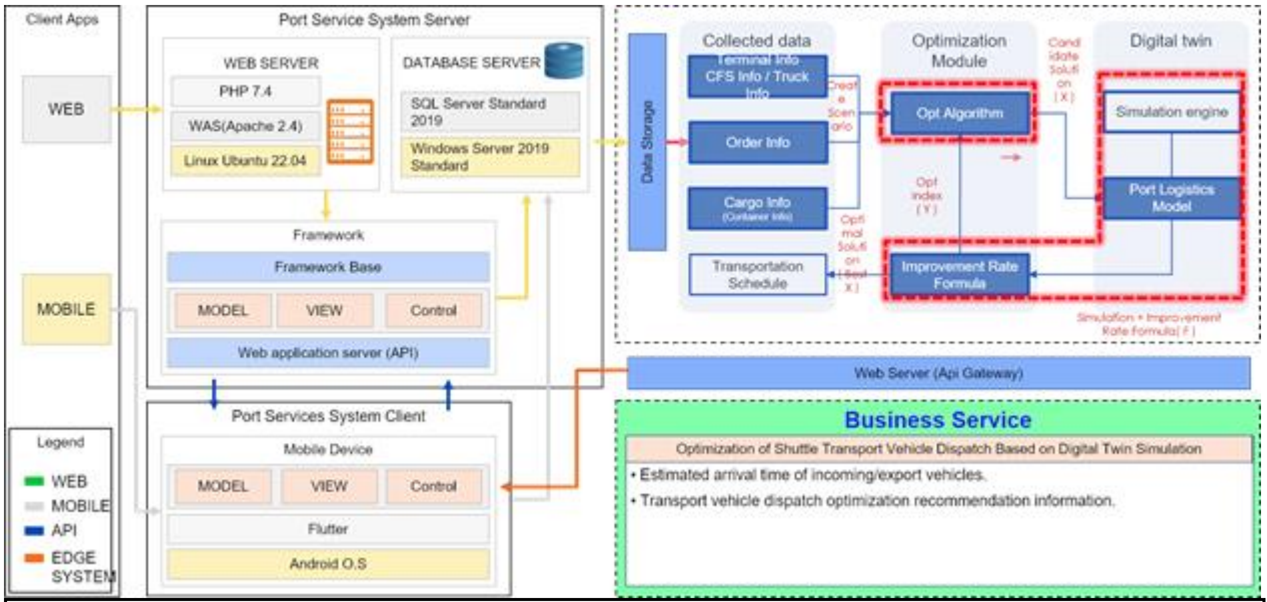


Type	Work Date	Work Time	Container No.	Owner	P.L.C	[illegible]		
이적	2024-04-22	16:23:22	TRHU1274540	ITR INDUS—		D	1	1
이적	2024-04-22	16:22:32	KOCU4256091	한국제지		B	3	1
이적	2024-04-22	16:21:27	MEDU4096533	한국제지		A	4	4
이적	2024-04-22	16:20:54	MSCU4029356	한국제지		A	4	3
이적	2024-04-22	16:20:53	MSDU4210812	한국제지		A	4	2

d



b) Transportation data platform Architecture



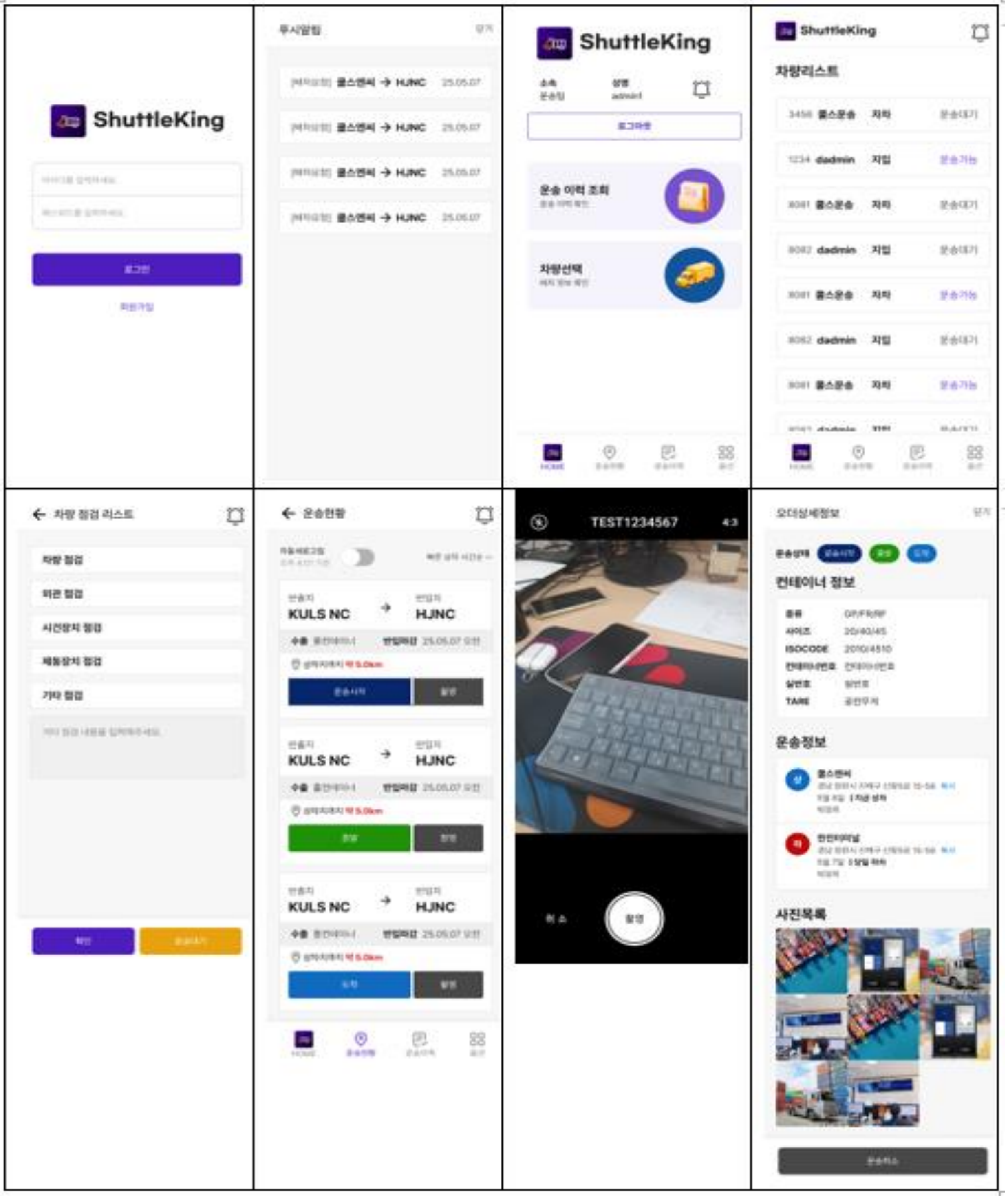
The following systems collect and manage port logistics transport data.

- Register and manage import and export container shipping orders.
- Manage container import/export shipping details.
- Manage container transport vehicle information.
- Manage real-time transport status information for containers.
- Manage container import and export performance.
- Web and mobile device-based environments

Web device-based environments

mobile device-based environments

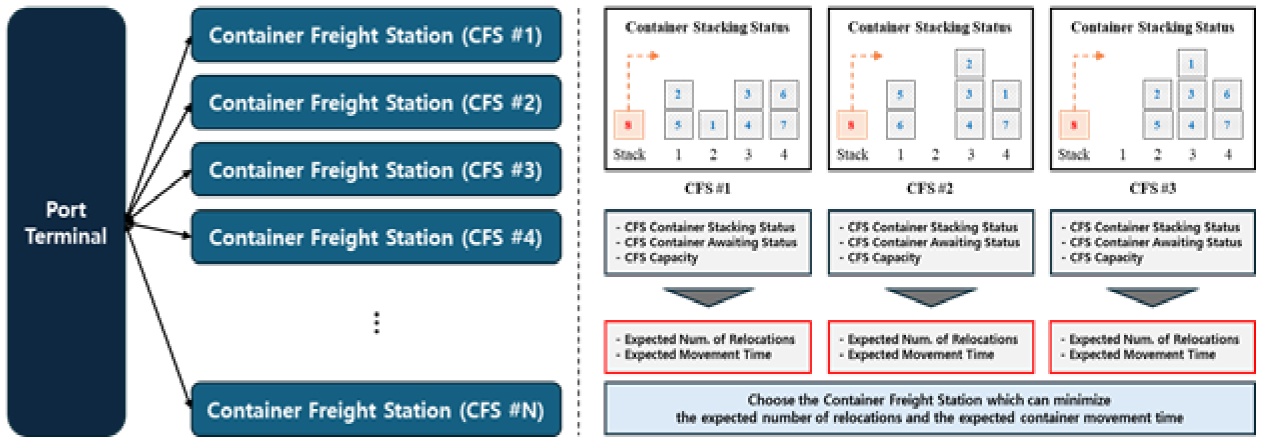
mobile device-based environments


ShuttleKing
로그인
회원가입
푸시알림
[버스도착] 콜스센씨 → HJNC 25.05.07
ShuttleKing
로그아웃
운송 이력 조회
차량선택
ShuttleKing
차량리스트
dadmin
HOME
차량 점검 리스트
차량 점검
외관 점검
시건장치 점검
제동장치 점검
기타 점검
확인
운송현황
KULS NC
HJNC
TEST1234567
취 소
촬영
오더상세정보
컨테이너 정보
TARE
운송정보
사진목록

2) Implementation of CFS/CY Optimization (KAIST)

a) Inter-Container Freight Station Optimization

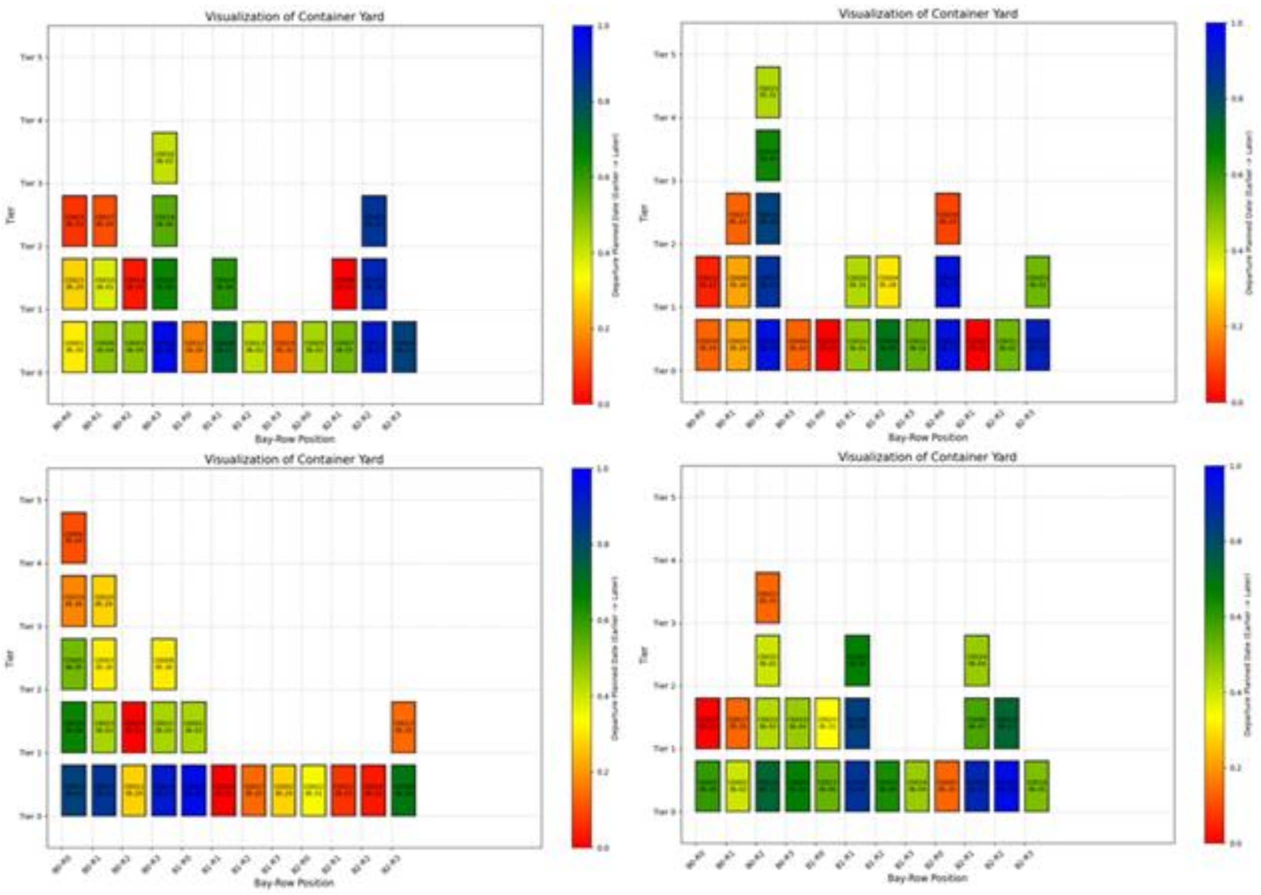
Inter-Container Freight Station Optimization

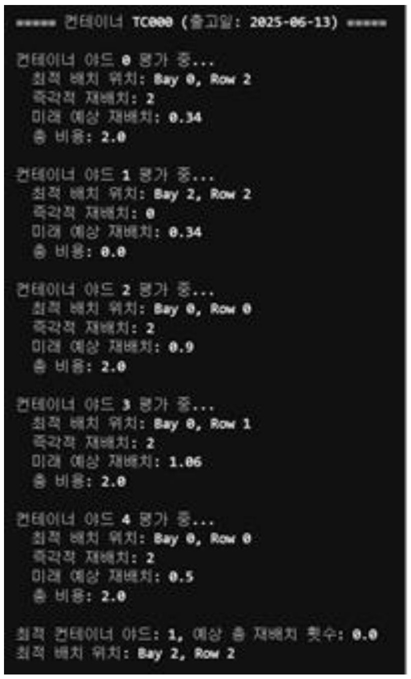

Port Terminal
Container Freight Station (CFS #1)
Container Freight Station (CFS #2)
Container Freight Station (CFS #3)
Container Freight Station (CFS #4)
Container Freight Station (CFS #N)
Container Stacking Status
Stack 1 2 3 4
CFS #1
CFS #2
CFS #3
- CFS Container Stacking Status
- CFS Container Awaiting Status
- CFS Capacity
- Expected Num. of Relocations
- Expected Movement Time
Choose the Container Freight Station which can minimize the expected number of relocations and the expected container movement time

To optimize the flow of containers between multiple container freight stations (CFSs), a decision-support module has been developed. This module enables dynamic selection of the most suitable CFS based on operational data collected from each station.

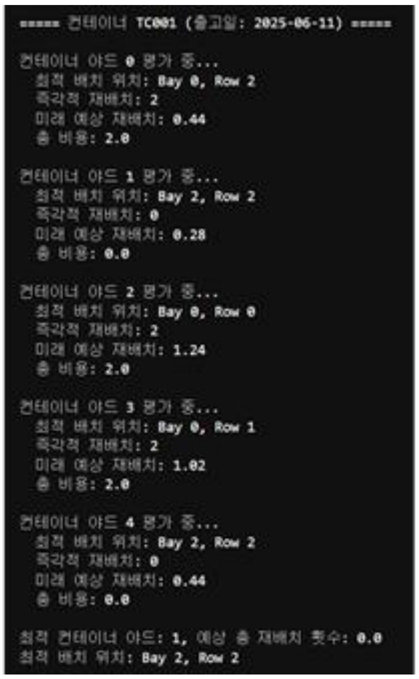
In the proposed framework, container stacking status, container awaiting status, and available capacity are monitored in real time across all candidate stations. For each newly arriving container, the expected number of relocations and the anticipated movement time (ex. future retrieval time) are estimated for each CFS.

Based on these criteria, a station is selected that minimizes total expected operational cost. The selection algorithm has been designed to reduce yard congestion, avoid inefficient distribution, and maximize system-wide throughput.

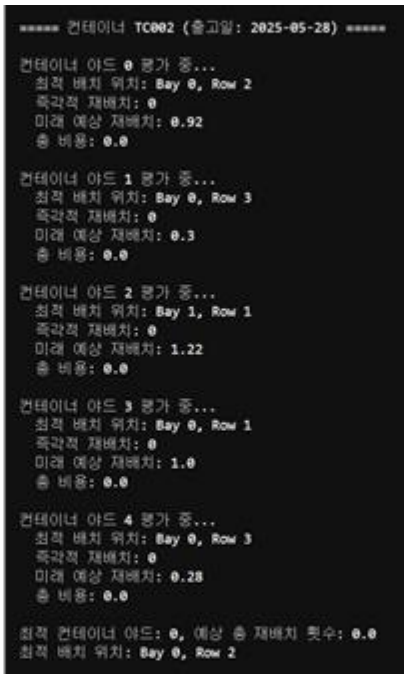




```
===== 컨테이너 TC000 (출고일: 2025-06-13) =====

컨테이너 야드 0 평가 중...
  최적 배치 위치: Bay 0, Row 2
  즉각적 재배치: 2
  미래 예상 재배치: 0.34
  총 비용: 2.0

컨테이너 야드 1 평가 중...
  최적 배치 위치: Bay 2, Row 2
  즉각적 재배치: 0
  미래 예상 재배치: 0.34
  총 비용: 0.0

컨테이너 야드 2 평가 중...
  최적 배치 위치: Bay 0, Row 0
  즉각적 재배치: 2
  미래 예상 재배치: 0.9
  총 비용: 2.0

컨테이너 야드 3 평가 중...
  최적 배치 위치: Bay 0, Row 1
  즉각적 재배치: 2
  미래 예상 재배치: 1.06
  총 비용: 2.0

컨테이너 야드 4 평가 중...
  최적 배치 위치: Bay 0, Row 0
  즉각적 재배치: 2
  미래 예상 재배치: 0.5
  총 비용: 2.0

최적 컨테이너 야드: 1, 예상 총 재배치 횟수: 0.0
최적 배치 위치: Bay 2, Row 2
```



```
===== 컨테이너 TC001 (출고일: 2025-06-11) =====

컨테이너 야드 0 평가 중...
  최적 배치 위치: Bay 0, Row 2
  즉각적 재배치: 2
  미래 예상 재배치: 0.44
  총 비용: 2.0

컨테이너 야드 1 평가 중...
  최적 배치 위치: Bay 2, Row 2
  즉각적 재배치: 0
  미래 예상 재배치: 0.28
  총 비용: 0.0

컨테이너 야드 2 평가 중...
  최적 배치 위치: Bay 0, Row 0
  즉각적 재배치: 2
  미래 예상 재배치: 1.24
  총 비용: 2.0

컨테이너 야드 3 평가 중...
  최적 배치 위치: Bay 0, Row 1
  즉각적 재배치: 2
  미래 예상 재배치: 1.02
  총 비용: 2.0

컨테이너 야드 4 평가 중...
  최적 배치 위치: Bay 2, Row 2
  즉각적 재배치: 0
  미래 예상 재배치: 0.44
  총 비용: 0.0

최적 컨테이너 야드: 1, 예상 총 재배치 횟수: 0.0
최적 배치 위치: Bay 2, Row 2
```



```
===== 컨테이너 TC002 (출고일: 2025-05-28) =====

컨테이너 야드 0 평가 중...
  최적 배치 위치: Bay 0, Row 2
  즉각적 재배치: 0
  미래 예상 재배치: 0.92
  총 비용: 0.0

컨테이너 야드 1 평가 중...
  최적 배치 위치: Bay 0, Row 3
  즉각적 재배치: 0
  미래 예상 재배치: 0.3
  총 비용: 0.0

컨테이너 야드 2 평가 중...
  최적 배치 위치: Bay 1, Row 1
  즉각적 재배치: 0
  미래 예상 재배치: 1.22
  총 비용: 0.0

컨테이너 야드 3 평가 중...
  최적 배치 위치: Bay 0, Row 1
  즉각적 재배치: 0
  미래 예상 재배치: 1.0
  총 비용: 0.0

컨테이너 야드 4 평가 중...
  최적 배치 위치: Bay 0, Row 3
  즉각적 재배치: 0
  미래 예상 재배치: 0.28
  총 비용: 0.0

최적 컨테이너 야드: 0, 예상 총 재배치 횟수: 0.0
최적 배치 위치: Bay 0, Row 2
```

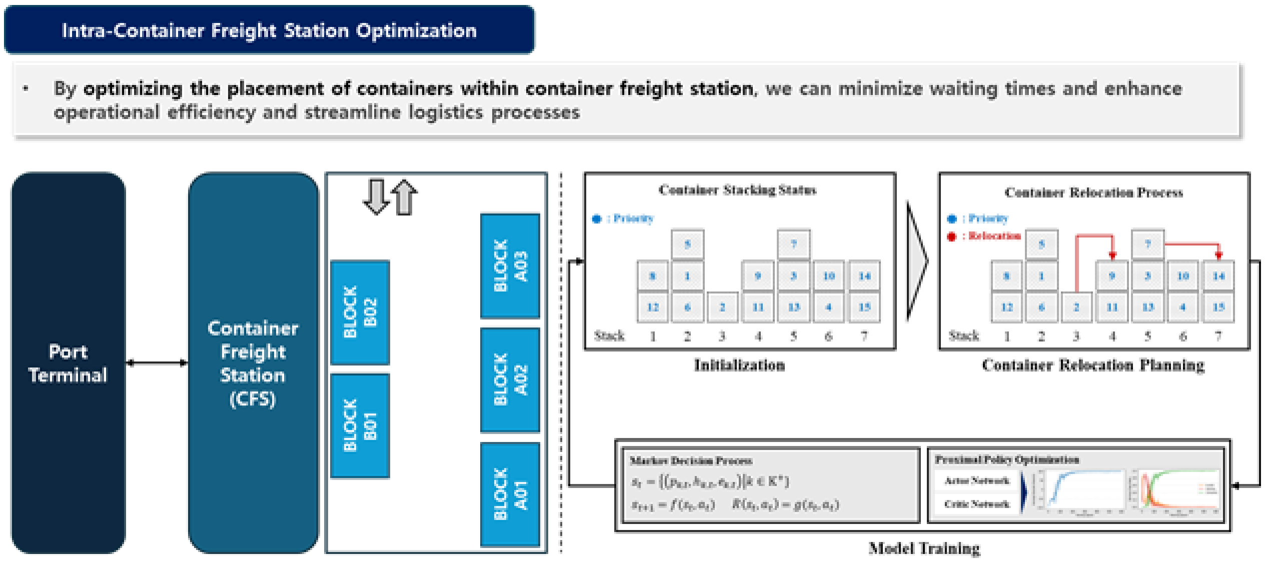
To validate the effectiveness of inter-station optimization, container stacking scenarios were simulated across multiple container freight stations (CFSs). For each target container, the expected operational cost—composed of immediate relocation and future retrieval actions—was evaluated based on current stacking configurations.

The container yard visualizations illustrate how different container positions and priority levels influence potential relocation paths. Containers are color-coded according to their expected departure date, providing an intuitive view of stacking priority.

Additionally, detailed logs were generated for selected containers across multiple CFSs, providing information such as optimal placement suggestions, the estimated number of both immediate and future relocations, and the final cost associated with each placement option.

By comparing these results across stations, the system was able to identify the CFS that offered the lowest total handling cost for each container. In several cases, cost savings were achieved by diverting containers to less congested yards, validating the core premise of dynamic inter-CFS optimization.

b) Intra-Container Freight Station Optimization



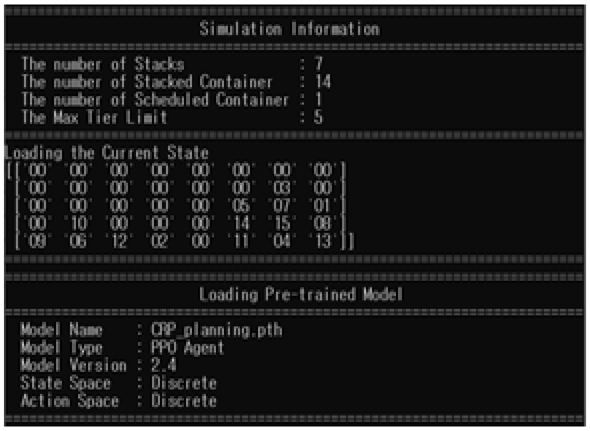
Intra-Container Freight Station Optimization

- By **optimizing the placement of containers within container freight station**, we can minimize waiting times and enhance operational efficiency and streamline logistics processes

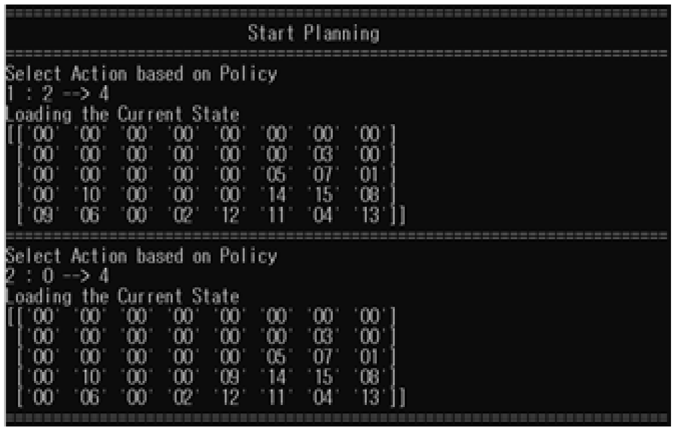
For intra-station optimization, a container stacking and relocation planning module has been implemented. This module determines the optimal placement of containers within each CFS in order to reduce internal inefficiencies.

The problem is formulated as a Markov Decision Process (MDP), where the state includes container positions, retrieval priorities, and equipment availability. A reinforcement learning approach has been employed using the Proximal Policy Optimization (PPO) algorithm.

By learning optimal placement strategies based on synthetic and historical data, the system has been shown to reduce the number of unnecessary relocations and minimize average container retrieval time. The learned policy assigns high-priority containers to more accessible positions, while low-priority ones are placed in deeper stacks, thus improving both space utilization and handling efficiency.



```
                    Simulation Information

  The number of Stacks               : 7
  The number of Stacked Container    : 14
  The number of Scheduled Container  : 1
  The Max Tier Limit                 : 5

Loading the Current State
[['00' '00' '00' '00' '00' '00' '00' '00']
 ['00' '00' '00' '00' '00' '00' '03' '00']
 ['00' '00' '00' '00' '00' '05' '07' '01']
 ['00' '10' '00' '00' '00' '14' '15' '08']
 ['09' '06' '12' '02' '00' '11' '04' '13']]

                  Loading Pre-trained Model

  Model Name    : CRP_planning.pth
  Model Type    : PPO Agent
  Model Version : 2.4
  State Space   : Discrete
  Action Space  : Discrete
```



```
                         Start Planning

Select Action based on Policy
1 : 2 --> 4
Loading the Current State
[['00' '00' '00' '00' '00' '00' '00' '00']
 ['00' '00' '00' '00' '00' '00' '03' '00']
 ['00' '00' '00' '00' '00' '05' '07' '01']
 ['00' '10' '00' '00' '00' '14' '15' '08']
 ['09' '06' '00' '02' '12' '11' '04' '13']]

Select Action based on Policy
2 : 0 --> 4
Loading the Current State
[['00' '00' '00' '00' '00' '00' '00' '00']
 ['00' '00' '00' '00' '00' '00' '03' '00']
 ['00' '00' '00' '00' '00' '05' '07' '01']
 ['00' '10' '00' '00' '09' '14' '15' '08']
 ['00' '06' '00' '02' '12' '11' '04' '13']]
```

To further validate the effectiveness of intra-station optimization, a policy-based simulation was conducted using a pre-trained reinforcement learning agent. The model was trained using the Proximal Policy Optimization (PPO) algorithm and deployed to evaluate container placement strategies under predefined yard configurations.

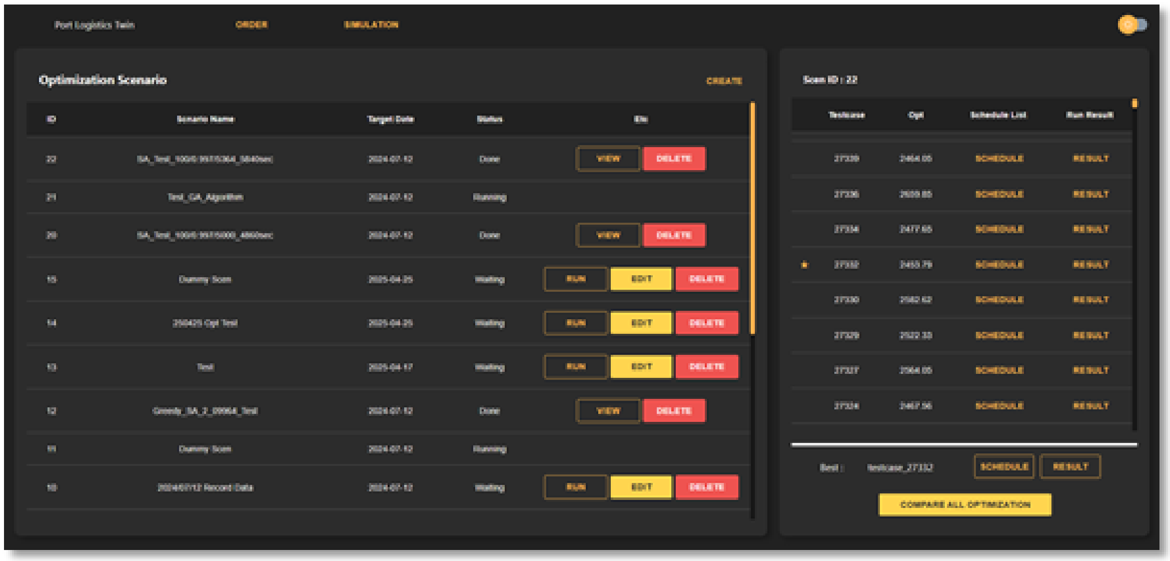
The simulation environment was configured with the number of stacks, maximum stack height, the number of stacked containers, and the number of scheduled containers. For example, in the illustrated scenario, 7 stacks were defined with a maximum height of 5 tiers, 14 containers were initially placed, and 1 container was scheduled for insertion.

At each decision point, the current yard state was assessed, and an action was selected based on the learned policy. The action corresponds to the stack index where the incoming container should be placed. This decision is made to minimize both immediate and future relocation costs.

The log captures the sequential process in which the model interprets the current state of the container yard, determines the optimal placement action by referencing its learned policy within a discrete action space, and subsequently updates the yard configuration based on the selected action.

This process demonstrates the agent's ability to generalize learned placement strategies across diverse yard states, while adhering to operational constraints such as stack height limits and spatial distribution.

3) Implementation of Digital Twin Simulation Platform (eins S&C)



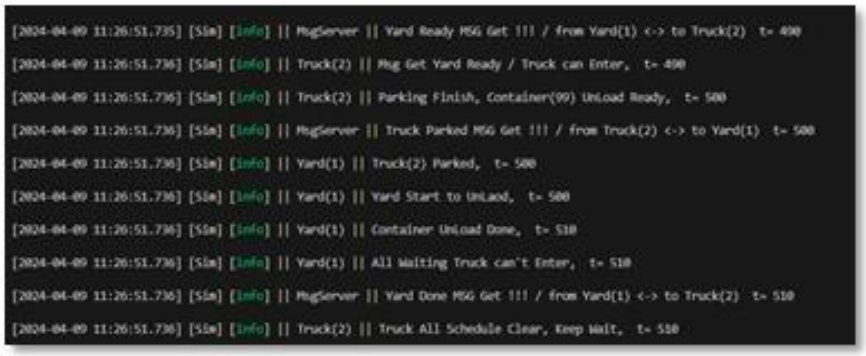
Optimization Scenario

ID	Scenario Name	Target Date	Status	Etc
22	SA_Test_100/0.997/5364_5840sec	2024-07-12	Done	VIEW DELETE
21	Test_GA_Algorithm	2024-07-12	Running	
20	SA_Test_100/0.997/5000_4860sec	2024-07-12	Done	VIEW DELETE
15	Dummy Scen	2025-04-25	Waiting	RUN EDIT DELETE
14	250425 Opt Test	2025-04-25	Waiting	RUN EDIT DELETE
13	Test	2025-04-17	Waiting	RUN EDIT DELETE
12	Greedy_SA_2_09964_Test	2024-07-12	Done	VIEW DELETE
11	Dummy Scen	2024-07-12	Running	
10	20240712 Record Data	2024-07-12	Waiting	RUN EDIT DELETE

Scen ID : 22

	Testcase	Opt	Schedule List	Run Result
	27339	2464.05	SCHEDULE	RESULT
	27336	2609.85	SCHEDULE	RESULT
	27334	2477.65	SCHEDULE	RESULT
★	27332	2455.79	SCHEDULE	RESULT
	27330	2582.62	SCHEDULE	RESULT
	27329	2522.33	SCHEDULE	RESULT
	27327	2564.05	SCHEDULE	RESULT
	27324	2467.56	SCHEDULE	RESULT

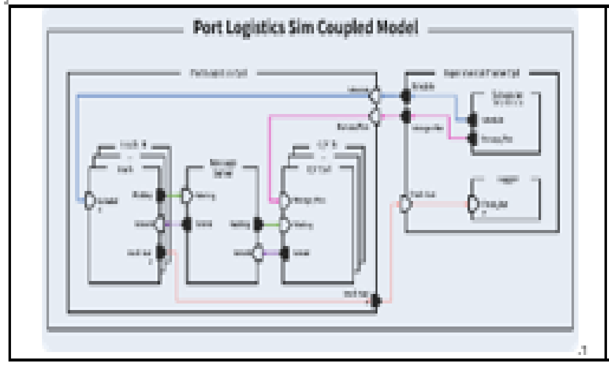
Best : testcase_27332 SCHEDULE RESULT

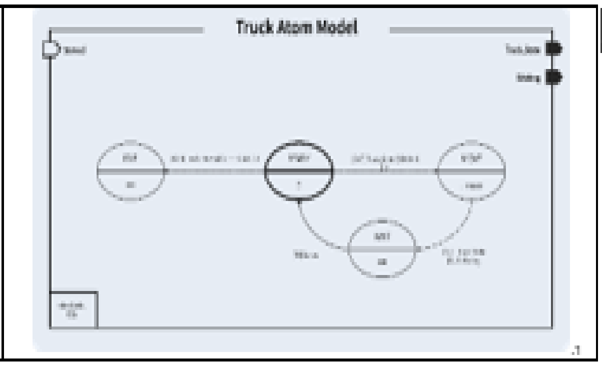


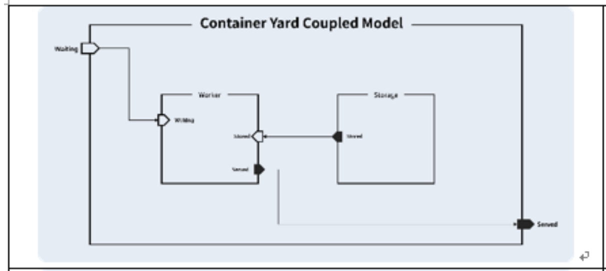
```
[2024-04-09 11:26:51.735] [Sim] [info] || MsgServer || Yard Ready MSG Get !!! / from Yard(1) <-> to Truck(2)  t= 490
[2024-04-09 11:26:51.736] [Sim] [info] || Truck(2) || Msg Get Yard Ready / Truck can Enter,  t= 490
[2024-04-09 11:26:51.736] [Sim] [info] || Truck(2) || Parking Finish, Container(99) UnLoad Ready,  t= 500
[2024-04-09 11:26:51.736] [Sim] [info] || MsgServer || Truck Parked MSG Get !!! / from Truck(2) <-> to Yard(1)  t= 500
[2024-04-09 11:26:51.736] [Sim] [info] || Yard(1) || Truck(2) Parked,  t= 500
[2024-04-09 11:26:51.736] [Sim] [info] || Yard(1) || Yard Start to UnLoad,  t= 500
[2024-04-09 11:26:51.736] [Sim] [info] || Yard(1) || Container UnLoad Done,  t= 510
[2024-04-09 11:26:51.736] [Sim] [info] || Yard(1) || All Waiting Truck can't Enter,  t= 510
[2024-04-09 11:26:51.736] [Sim] [info] || MsgServer || Yard Done MSG Get !!! / from Yard(1) <-> to Truck(2)  t= 510
[2024-04-09 11:26:51.736] [Sim] [info] || Truck(2) || Truck All Schedule Clear, Keep Wait,  t= 510
```

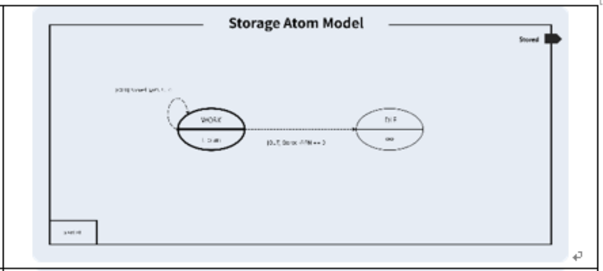
· The Simulation Model is modeled based on the actual port logistics transportation system and derives the simulated operation results for each truck according to the input schedule list.

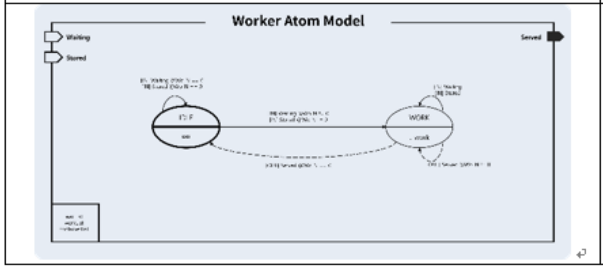
· The Optimization Module retrieves the transportation contract data for the target optimization date from the platform, explores various dispatch combinations using that data, and evaluates each combination's operation result through the simulation model.

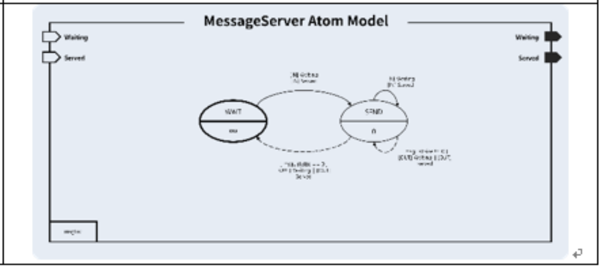








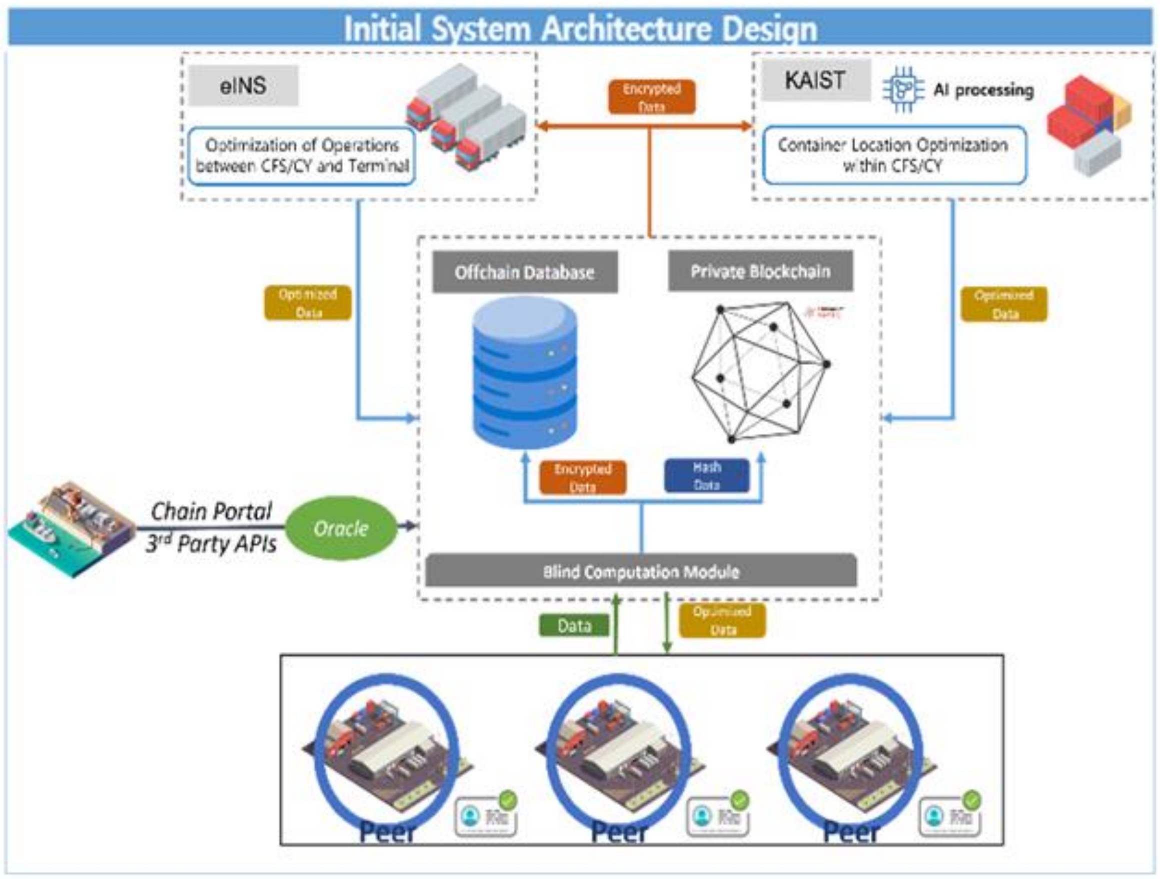




- The Simulation Model part is designed and developed based on the Digital Twin Simulation platform using BAS (Big Data, AI, Simulation) technologies. In particular, for this project, a DEVS-based model was used to model dynamic systems, where Truck, Container Yard(CFS, and Terminal) objects send and receive events as time progresses to simulate the system. The platform is developed in C++, which ensures fast simulation speed.

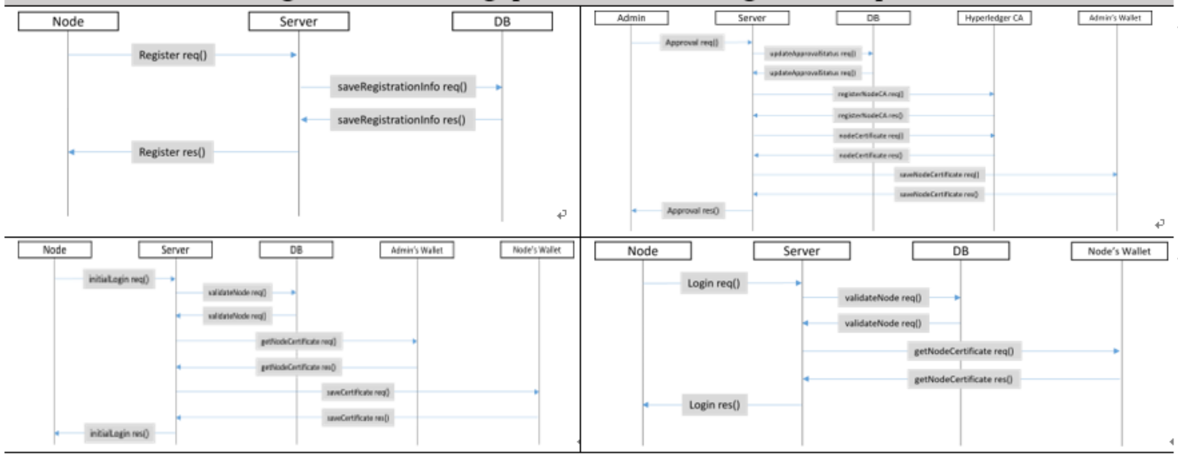
- The Optimization Module mainly uses the Python programming language to implement search and evaluation algorithms. Currently, the module generates an initial solution using the Greedy algorithm and performs search and evaluation using the SA (Simulated Annealing) algorithm. The algorithms are continuously being modified and tuned to achieve better results through ongoing testing

4) Implementation of Data Trust Provisioning Module and Blockchain Capability (INJE Univ.)

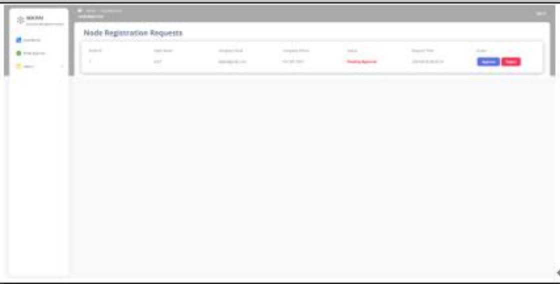
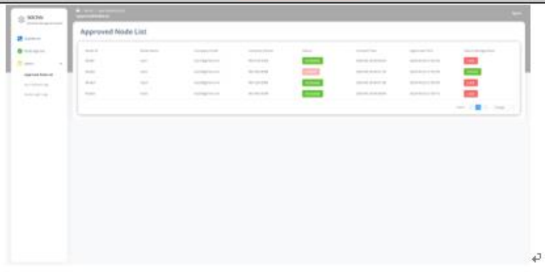
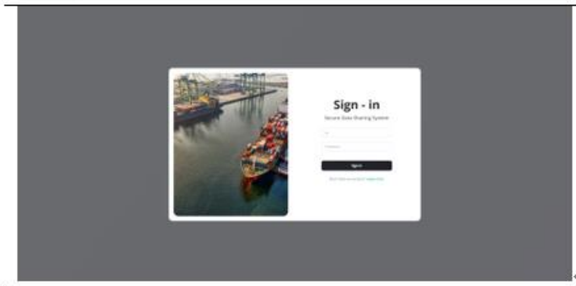
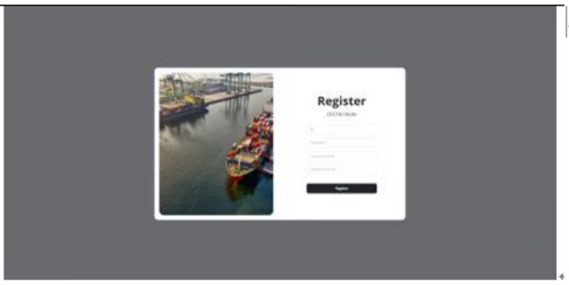


Description: This system represents an initial architecture designed to securely collect and process logistics data. Data from eINS and KAIST is encrypted and stored in an off-chain database, while the hash of the data is recorded on a private blockchain to ensure integrity. The Blind Computation Module performs secure computations on the data, and the optimized results are shared back with the participating institutions. Real-world logistics information is connected through an oracle, and the final optimized data is distributed to each peer node. The entire architecture is designed to enhance security, preserve data integrity, and support efficient logistics operations.

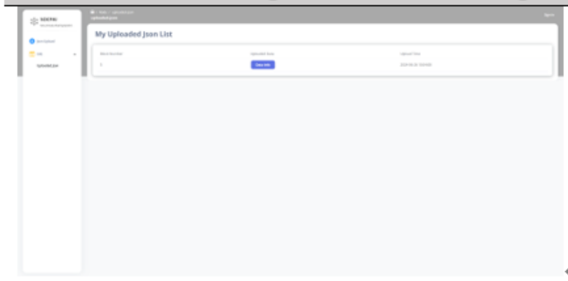
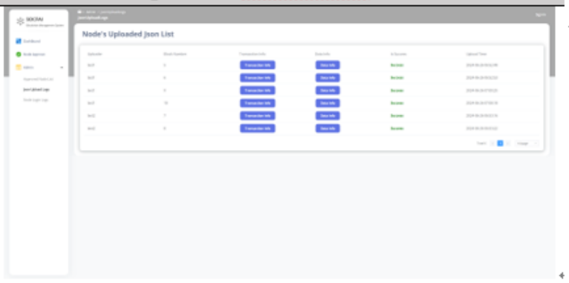
Designed Data sharing, private Node Management Sequence



Description: This system is designed for private node-based data sharing and authentication management. First, when a user sends a registration request, the system stores the registration information and responds accordingly. After administrator approval, the node is issued a certificate, which is stored in the system—allowing it to officially join the network. During the initial login, the system validates the node and its certificate, then stores the certificate for future use. For regular logins, the system again verifies the node and certificate before granting access. The overall sequence establishes the foundation for secure data sharing and trust among participating nodes.

Membership management function based on private blockchain	
	
	

Description: This system is a demo version of the first-stage private member management page, designed to enable secure registration, verification, and data sharing of blockchain nodes. The process begins with the registration page, where a user submits node-related information such as email and company phone number. These registration requests are listed on the "Node Registration Requests" admin page, where an administrator can either approve or reject them. Once approved, the node appears on the "Approved Node List" page, which displays key information such as node status and registration date. After approval, users can access the system via the Sign-in page. Only verified nodes are granted access to the secure data sharing environment. This entire flow ensures a secure, authenticated process for onboarding and managing private nodes in the network.

Information management(secure sharing data) function of private blockchain for member	
	

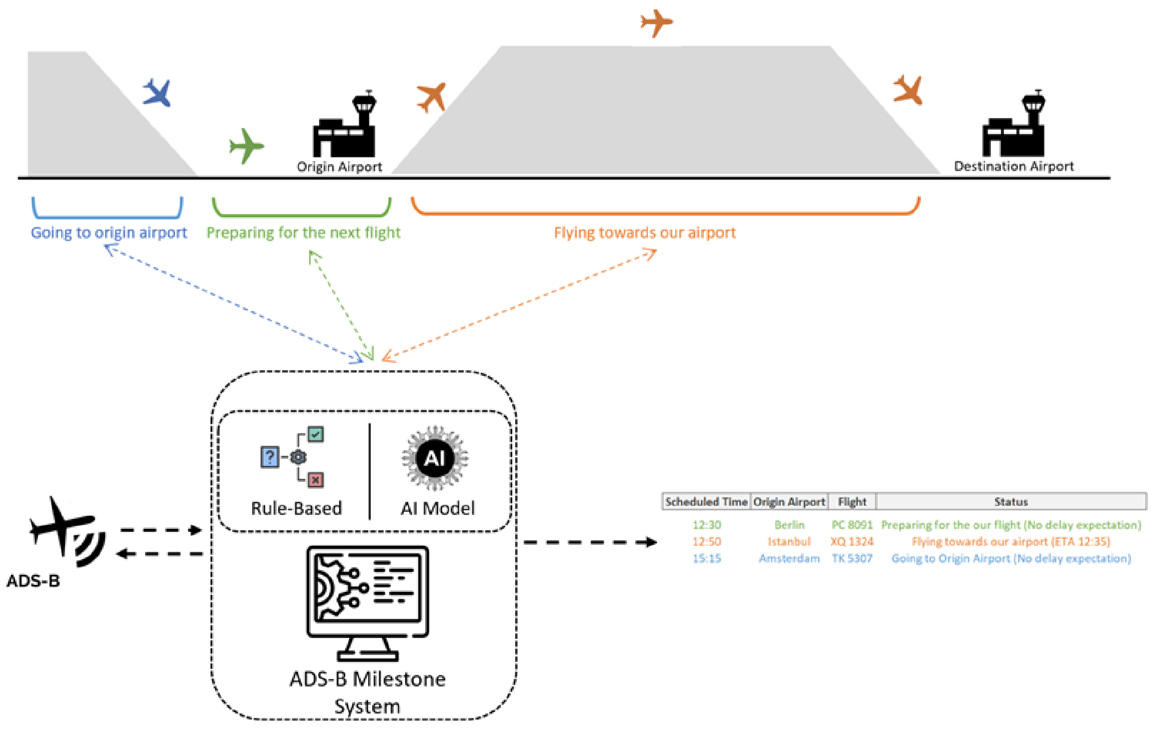
Description: These screens are part of a first-stage demo version of the data management interface designed for a private blockchain system. The first screen allows each node to view a list of their own uploaded JSON data, including file names and upload times, along with the ability to submit that data for blockchain processing. The second screen is for administrators to monitor the full list of uploaded JSON data across all nodes. It displays which block number each file is recorded under, associated transaction information, verification results, and status. This demo interface allows both users and administrators to intuitively track the data upload and verification status, providing a basic but functional test environment for blockchain-based data integrity tracking.

3.2.11 Use Case Application #11

3.2.11.1. Real-time Flight Status and ATAD Prediction

The primary functional objective is to provide highly accurate, near real-time predictions for the Actual Time of Arrival at Destination (ATAD) for flights arriving at İzmir Adnan Menderes Airport (ADB).

- **Data Ingestion:** The system continuously ingests high-frequency ADS-B positional data for specific flights and routes (currently 115 routes, 6,965 flights for ADB) sourced from Flightradar24 (FR24). This data is complemented by scheduled flight information from the Flight Management System (FMS).
- **Data Transformation:** Raw ADS-B data undergoes immediate preprocessing, including parsing timestamped location information and calculating time-to-arrival metrics.
- **Machine Learning Prediction:** A GPU-accelerated Random Forest model, trained on historical ADS-B and FMS data, is utilized to predict ATAD. This model is served via containerized prediction services (Docker) which consume incoming flight data streams from Kafka. The model is optimized for low latency, enabling near real-time predictions with a low Mean Absolute Error (MAE) of approximately 109 seconds.
- **Outlier Handling:** To maintain the integrity of ATAD predictions for "normal" flight operations, the system employs a rule-based mechanism to identify and filter out anomalous flight behaviors such as holding patterns (in-air) or holding positions (on-ground) from the training data. These specific scenarios, driven by ATC directives or ground operations, are handled by distinct operational logic rather than the core predictive model.



Scheduled Time	Origin Airport	Flight	Status
12:30	Berlin	PC 8091	Preparing for the our flight (No delay expectation)
12:50	Istanbul	XQ 1324	Flying towards our airport (ETA 12:35)
15:15	Amsterdam	TK 5307	Going to Origin Airport (No delay expectation)

3.2.11.2 Comprehensive Flight Operational State Detection

Beyond ATAD prediction, the system provides real-time situational awareness by identifying critical flight operational states. This functionality is primarily driven by a robust Rule-Based System, leveraging various data points to make contextual determinations.

- **Departure Status Monitoring:** The system monitors and infers:
    - **Estimated Airport Arrival Status:** Based on distance-based calculations relative to the estimated time of departure.
    - **ADS-B Transmitter Activation:** Verifying if the pilot has activated the ADS-B transmitter at the scheduled or estimated departure time.
    - **On-Time Performance:** Determining if the flight departed on schedule or quantifying any associated delays.
- **Destination Validation:** The system can verify whether an aircraft's movement is indeed destined for the specific airport under observation, preventing misidentification of transiting aircraft.
- **Congestion-Based Holding Detection:** For arriving aircraft, the system attempts to detect instances of in-air holding patterns, which are often indicative of airport congestion or

adverse weather during approach. This provides early indicators of potential arrival delays not directly captured by ATAD.

- **Cost Optimization:** This rule-based approach, integrated with the predictive model, enables cost optimization by strategically limiting the acquisition of high-frequency ADS-B data only to critical points or segments, rather than continuous full-flight acquisition.

3.2.11.3. Flight Services Time Prediction

This functional area focuses on optimizing ground handling operations by predicting the precise timing of various airport services.

- **Service-Specific Predictions:** Utilizing historical FMS data, the system develops and deploys models to predict the start and end times for a wide array of flight services. These services include, but are not limited to:
  - Boarding
  - Bridge Connection/Disconnection
  - Baggage Carousel Activation/Deactivation
  - Check-in Operations
  - Chute Operations
  - Counter Availability
  - Ground Electricity Connection/Disconnection
  - Gate Occupancy
  - In-Block
  - Off-Block
- **Resource Optimization:** By providing accurate start and end time predictions for each service, the system enables more efficient allocation of ground staff, equipment, and gate resources, ultimately minimizing aircraft turnaround times and improving overall airport operational efficiency.

3.2.11.4. Robust MLOps Enablement

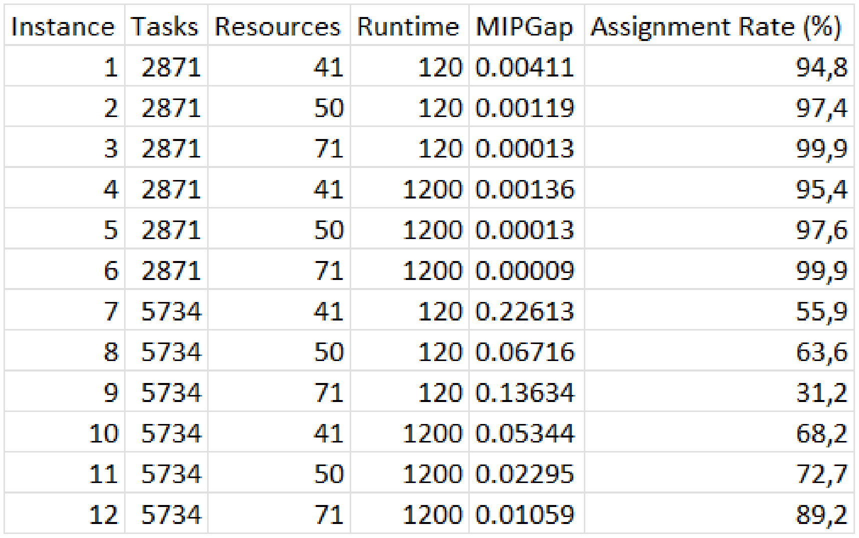
The underlying MLOps architecture ensures the continuous, reliable, and scalable operation of all predictive and rule-based functionalities.

- **Data Versioning (DVC):** All datasets, from raw input to preprocessed features, are versioned using DVC. This guarantees data reproducibility for every model training run and facilitates seamless rollback to previous data states if required.
- **Experiment Tracking (MLflow):** MLflow is central to managing the model development lifecycle. It tracks every experiment run, recording parameters, metrics, and model artifacts, enabling systematic comparison and selection of optimal models.
- **Model Registry (MLflow & MinIO):** Trained models are registered in MLflow's Model Registry, with the actual model binaries stored in MinIO. This provides a centralized, versioned repository for all production-ready models.
- **Automated Deployment (Jenkins & Docker):** Jenkins orchestrates the automated deployment of models. Upon approval, Jenkins builds and deploys new or updated models as Docker containers (Prediction Service X), ensuring consistent and scalable inference environments.
- **Continuous Monitoring (Evidently AI & NannyML):** Post-deployment, the system continuously monitors model performance and data integrity. Evidently AI detects data drift and concept drift within the live prediction data (streamed via Kafka), while NannyML focuses on performance monitoring, identifying any degradation in model accuracy. This proactive monitoring ensures the models remain relevant and effective in dynamic operational environments.
- **Metadata Management (PostgreSQL):** A PostgreSQL database serves as the backend for MLflow, storing all experiment metadata, model versioning information, and performance logs, providing a persistent and queryable record of the entire MLOps process.

This integrated functional implementation provides a powerful toolset for optimizing airport operations, enhancing prediction accuracy, and ensuring the long-term reliability and adaptability of machine learning solutions.

3.2.11.5. Analysis Results of RMS Optimization Engine

Our analysis on the Resource Management System (RMS) optimization engine examines how the engine performs across different task-resource assignment scenarios, focusing on runtime, optimality gap (MIPGap), and the number of planned versus unplanned tasks. The study compares twelve instances by varying the number of resources (41, 50, 71), the number of tasks (2871 and 5734), and the time allowed for optimization (120s and 1200s) (as seen below).

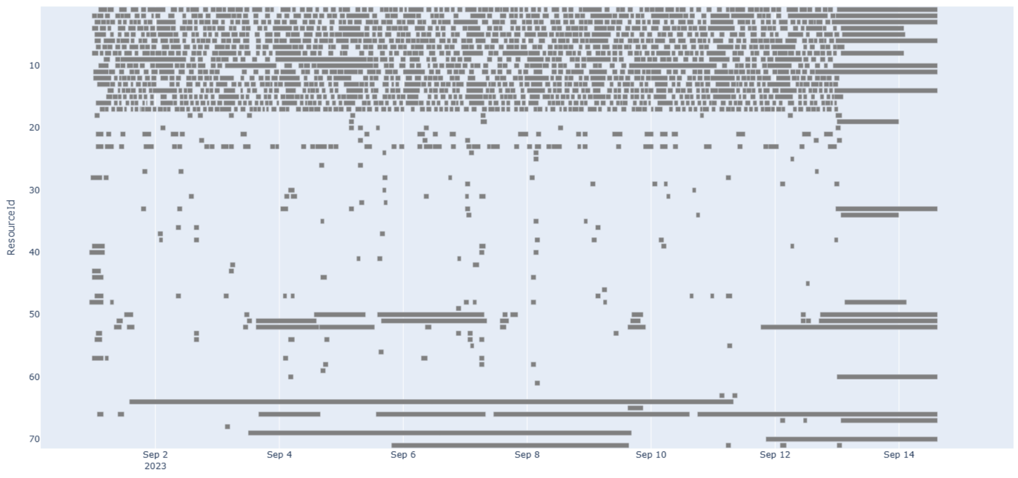


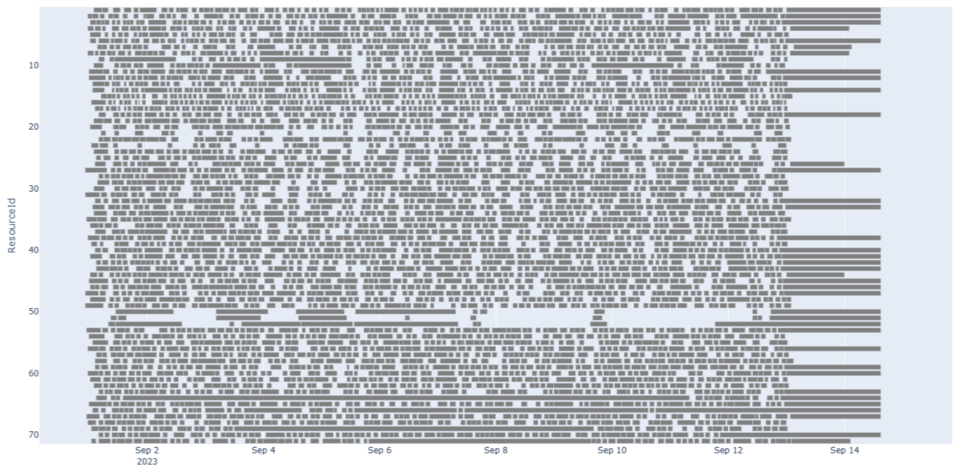
Instance	Tasks	Resources	Runtime	MIPGap	Assignment Rate (%)
1	2871	41	120	0.00411	94,8
2	2871	50	120	0.00119	97,4
3	2871	71	120	0.00013	99,9
4	2871	41	1200	0.00136	95,4
5	2871	50	1200	0.00013	97,6
6	2871	71	1200	0.00009	99,9
7	5734	41	120	0.22613	55,9
8	5734	50	120	0.06716	63,6
9	5734	71	120	0.13634	31,2
10	5734	41	1200	0.05344	68,2
11	5734	50	1200	0.02295	72,7
12	5734	71	1200	0.01059	89,2

For smaller instances (2871 tasks), results show that increasing the number of resources consistently improves task planning and reduces MIPGap. For example, with 120 seconds, raising resources from 41 to 71 increased planned tasks from 2721 to 2867 and reduced MIPGap from 0.00411 to 0.00013. Extending the runtime to 1200 seconds offered only marginal gains, indicating that small problems reach near-optimal solutions quickly.

Larger instances (5734 tasks), however, show more complex behavior. While more resources generally help, the solver sometimes underperforms under tight time constraints. Notably, with 71 resources and only 120 seconds, performance dropped sharply—only 1787 tasks were planned, far less than with 50 resources (3645 tasks). This suggests that a high number of resources increases model complexity, requiring more time to exploit effectively. When the time limit was extended to 1200 seconds, this same configuration (71 resources) achieved the best results, planning 5115 tasks with a low MIPGap of 0.01059. The difference between shorter and longer runtimes are shown in the following figures.

Figure. Task assignment results for 5734 tasks, 71 resources, 120s runtime



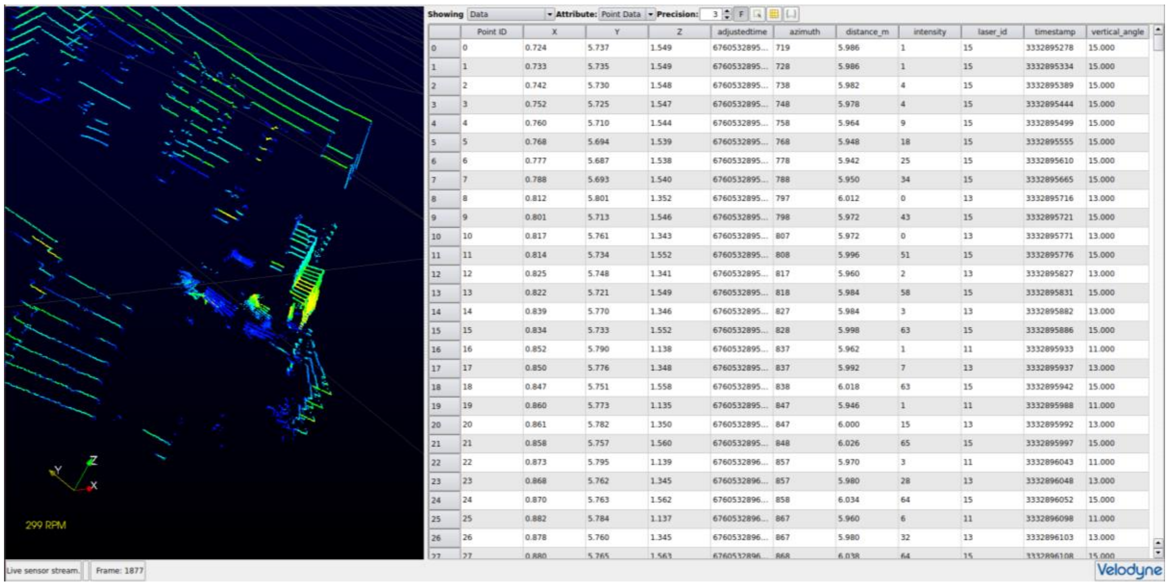


The results demonstrate that the optimization engine is highly effective at solving small to moderately sized assignment problems within a short time frame, delivering near-optimal solutions with very low MIP gaps. For larger-scale problems, however, the engine's performance becomes more sensitive to the interplay between runtime and the number of available resources. While increasing either parameter generally improves outcomes, their effectiveness is not independent: more resources require more computational time to be fully utilized. Moreover, certain configurations (e.g., high resource count with tight runtime) may paradoxically lead to degraded performance due to solver overhead and limited convergence. This analysis highlights the importance of careful parameter tuning based on problem scale and supports the development of adaptive strategies that allocate solver resources dynamically in response to instance characteristics.

4 . Data Management

4.1 Use Case Application #1 (INOSENS)

4.1.1 Data Sources:



	Point ID	X	Y	Z	adjustedtime	azimuth	distance_m	intensity	laser_id	timestamp	vertical_angle
0	0	0.724	5.737	1.549	6760532895...	719	5.986	1	15	3332895278	15.000
1	1	0.733	5.735	1.549	6760532895...	728	5.986	1	15	3332895334	15.000
2	2	0.742	5.730	1.548	6760532895...	738	5.982	4	15	3332895389	15.000
3	3	0.752	5.725	1.547	6760532895...	748	5.978	4	15	3332895444	15.000
4	4	0.760	5.710	1.544	6760532895...	758	5.964	9	15	3332895499	15.000
5	5	0.768	5.694	1.539	6760532895...	768	5.948	18	15	3332895555	15.000
6	6	0.777	5.687	1.538	6760532895...	778	5.942	25	15	3332895610	15.000
7	7	0.788	5.693	1.540	6760532895...	788	5.950	34	15	3332895665	15.000
8	8	0.812	5.801	1.352	6760532895...	797	6.012	0	13	3332895716	13.000
9	9	0.801	5.713	1.546	6760532895...	798	5.972	43	15	3332895721	15.000
10	10	0.817	5.761	1.343	6760532895...	807	5.972	0	13	3332895771	13.000
11	11	0.814	5.734	1.552	6760532895...	808	5.996	51	15	3332895776	15.000
12	12	0.825	5.748	1.341	6760532895...	817	5.960	2	13	3332895827	13.000
13	13	0.822	5.721	1.549	6760532895...	818	5.984	58	15	3332895831	15.000
14	14	0.839	5.770	1.346	6760532895...	827	5.984	3	13	3332895882	13.000
15	15	0.834	5.733	1.552	6760532895...	828	5.998	63	15	3332895886	15.000
16	16	0.852	5.790	1.138	6760532895...	837	5.962	1	11	3332895933	11.000
17	17	0.850	5.776	1.348	6760532895...	837	5.992	7	13	3332895937	13.000
18	18	0.847	5.751	1.558	6760532895...	838	6.018	63	15	3332895942	15.000
19	19	0.860	5.773	1.135	6760532895...	847	5.946	1	11	3332895988	11.000
20	20	0.861	5.782	1.350	6760532895...	847	6.000	15	13	3332895992	13.000
21	21	0.858	5.757	1.560	6760532895...	848	6.026	65	15	3332895997	15.000
22	22	0.873	5.795	1.139	6760532896...	857	5.970	3	11	3332896043	11.000
23	23	0.868	5.762	1.345	6760532896...	857	5.980	28	13	3332896048	13.000
24	24	0.870	5.763	1.562	6760532896...	858	6.034	64	15	3332896052	15.000
25	25	0.882	5.784	1.137	6760532896...	867	5.960	6	11	3332896098	11.000
26	26	0.878	5.760	1.345	6760532896...	867	5.980	32	13	3332896103	13.000
27	27	0.880	5.765	1.563	6760532896...	868	6.038	64	15	3332896108	15.000

LiDAR sensors operating in any environment provide point cloud data of the environment. This point cloud data includes three-dimensional coordinate information (x, y, z) and density (magnitude of signal reflection power) for each point. Azimuth/vertical angle and distance information for each point can also be obtained indirectly from three-dimensional coordinate data.

Table. Structure of LiDAR Point Cloud Data

Column Name	Description
Point ID	Unique identifier for each point (sequential number)
X	X coordinate of the point (in meters)
Y	Y coordinate of the point
Z	Z coordinate of the point
adjustedtime	Adjusted timestamp when the point was captured
azimuth	Horizontal angle of the laser beam (in degrees)

distance_m	Distance from the sensor to the point (in meters)
intensity	Reflectivity or strength of the returned laser signal
laser_id	ID of the laser that captured the point (e.g., 0–15 for 16-beam LiDAR)
timestamp	Timestamp when the point was recorded
vertical_angle	Vertical angle of the laser beam (in degrees)

As a preliminary step, an initial data collection study was carried out at Gebze Technical University. During this phase, the Robot Operating System (ROS) framework was utilized to establish the connection between the computer and the Velodyne VLP-16 LiDAR sensor, and to explore methods for data acquisition and recording.

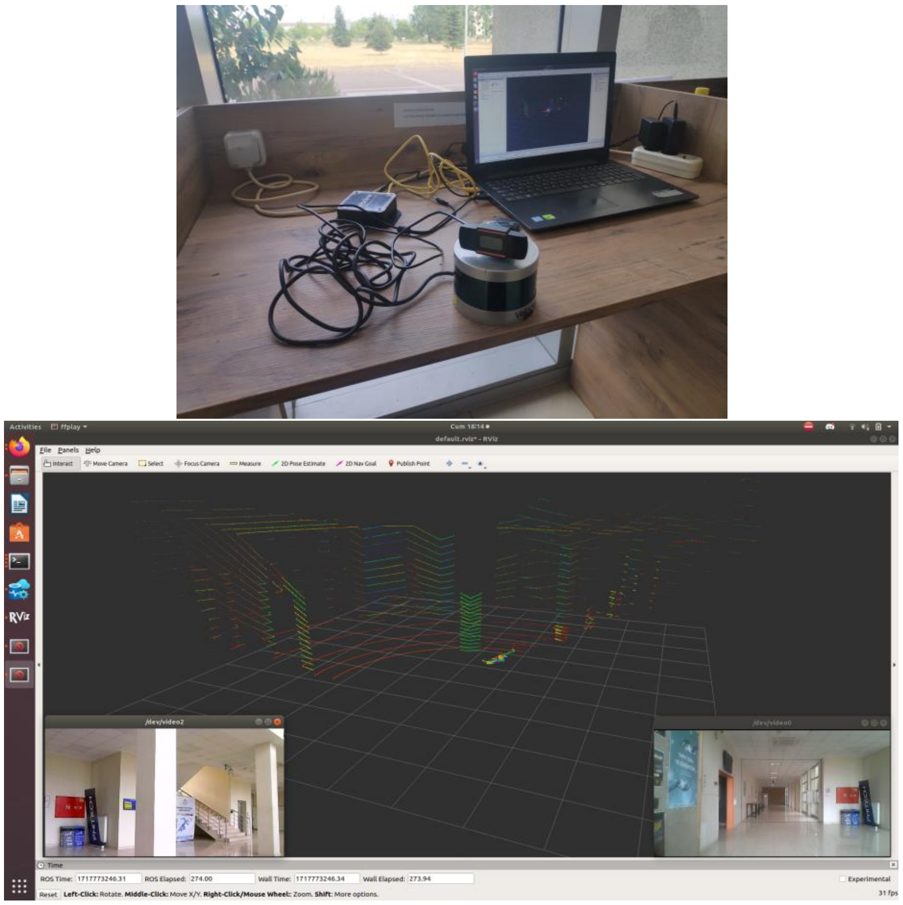


Figure. Data Collection Study in Gebze Technical University

Subsequently, the main data collection effort took place at Adnan Menderes Airport, specifically in the X-ray checkpoint area where passengers hand over their luggage and undergo security screening. Approximately 4 to 5 hours of LiDAR point cloud data were collected in this environment.

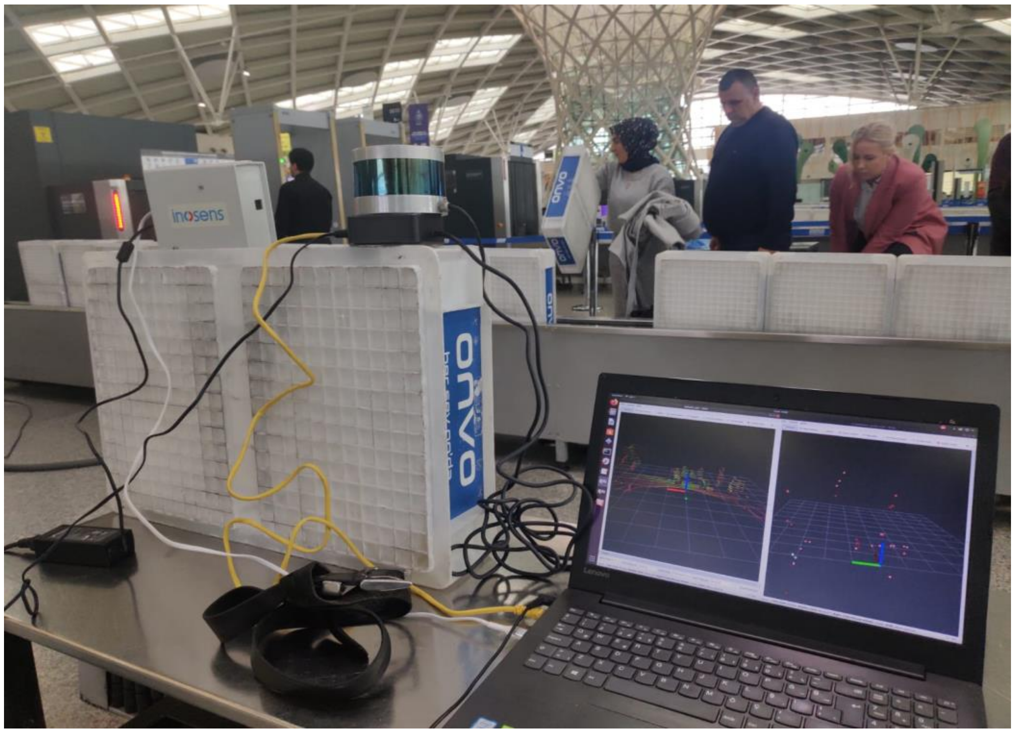


Figure. Data Collection Study in Adnan Menderes Airport

4.2 Use Case Application #2

4.2.1 Data Sources:

Two different datasets regarding flight and baggage information are analyzed in this use case. The datasets are collected from ADB airport and have been shared in this publication in an anonymized form. The authors do not have permission to share data. Initially, the data is collected for a one month period. Details about the dataset are described in the following paragraphs with emphasizing major properties:

- **Flight Dataset:** This dataset contains a substantial amount of data, with around 30K rows, related to flight operations over a month. This dataset includes detailed information about flight properties, aircraft features, arrival and departure times, airport codes, flight status, and various timestamps. The large volume of data enables extensive analysis of flight performance, on-time operations, and airport/airline efficiency. It can support decision-making processes, identify trends, and detect anomalies in the complex world of flight management.

- **Baggage Dataset:** This comprehensive dataset contains detailed information about baggage handling and tracking within an airport or airline system. With approximately 11M rows, it provides a complete record of baggage events, statuses, arrival details, security-related information, passenger profiles, and timestamps for various handling stages throughout the entire month. The scale of this dataset allows for in-depth analysis of baggage management processes, identification of trends and patterns, and optimization of operational efficiency across the entire baggage handling system.

Major features regarding flight and baggage dataset are summarized in the following table with feature name and feature description pair details. These features are mainly interpreted in the upcoming sections.

Table. Major features of dataset

Feature Name	Feature Description
ACTUAL_TIME	Actual time of flight
BAG_NUMBER	Number of baggage in a single baggage record
BAGGAGE_EVENT	Information about baggage operation itself
BAGGAGE_STATUS_ALL	Status of baggage
BIM_CREATE_DATE	First timestamp when baggage is entered to the system
BIM_ID	ID of a baggage record
CATEGORY	Identifies flight type (Domestic, International)
ESTIMATED_TIME	Estimated time of flight
FLIGHT_CODE	Combination of Airline Code and Flight Number
FLIGHT_STATUS	Status of flight
ID	Flight ID (also connection with Flight Dataset)
SCHEDULED_TIME	Scheduled time of flight
SYSTEM_AIRPORT	Identify airport from the point of flight direction view

4.3 Use Case Application #3

4.3.1 Data Sources

Use Case 3 relies on multiple data streams to enable intelligent air quality monitoring, energy consumption analysis, and automated HVAC control at Adnan Menderes Airport (İzmir). The primary data sources include:

1. IoT Air Quality Sensor Data (NETAŞ)

- Real-time environmental measurements collected from distributed IoT sensors across the airport, including:
  - Temperature (°C)
  - Relative Humidity (%)
  - Particulate Matter (PM2.5 & PM10) (µg/m³)
  - Carbon Dioxide (CO_2) Levels (ppm)
- Air Quality Index (AQI) – A computed metric aggregating sensor data to assess overall air quality.
- Sensor metadata (location, timestamp, device health status).

2. Energy Consumption Data (SOCFAI Platform)

- Historical and real-time energy usage from airport systems (HVAC, lighting, etc.).
- SCADA system logs providing granular energy demand patterns.
- Peak load and penalty thresholds from electricity providers to optimize consumption.

3. Passenger & Flight Data (TAV Technologies / SOCFAI Platform)

- Flight schedules & delays (for occupancy prediction).
- Passenger traffic analytics (real-time and forecasted footfall).

4. External Contextual Data

- Weather forecasts (temperature, humidity, wind speed) for predictive adjustments.
- Electricity pricing & demand-response signals (for cost-efficient energy use).

4.3.2 Data Integration & Usage

Air quality and passenger data will be cross-analyzed with energy consumption trends to identify inefficiencies.

AI models will process these datasets to:

- Predict HVAC demand based on occupancy and air quality.
- Trigger automated ventilation adjustments when thresholds are breached.
- Optimize energy use while maintaining passenger comfort.

This multi-source data framework ensures real-time responsiveness, energy savings, and improved indoor air quality across the airport.

4.4 Use Case Application #4 (INOSENS)

In this use case, the text-based content will be gathered from online sources. This includes social media posts, comments, and reviews about the organization (Izmir airport). As a result, the data will include text as well as relevant information including the date and time on which it was posted. Also, AI models will be used to detect information about the posts, specifically, the language of the post (e.g. Turkish) and the predicted sentiment. This information will be stored into the system.

Data Collected from Online Sources	
Text	The text extracted about the organization from an online source. It could be a social media post, a comment, or a review.
DateTime	The date and time on which the user or passenger posted the relevant text.
Author	An ID or username identifying the author of the text.
Information Acquired via AI Models	
Sentiment	The predicted sentiment of the text.
Language	The language of the text, as predicted by an AI model.

4.5 Use Case Application #5

For the data management of the CFS/CY Port Logistics operation, the following procedures are formalized and applied to all modules in the AI powered Logistics Management Platform .

- **Data Sources:** Description of the data sources used by the application.

    ① Port logistics data collection and processing.

    ② Port logistics transport data collection and processing.

    ③ Logistics data needed to optimize AI-based container deployment.

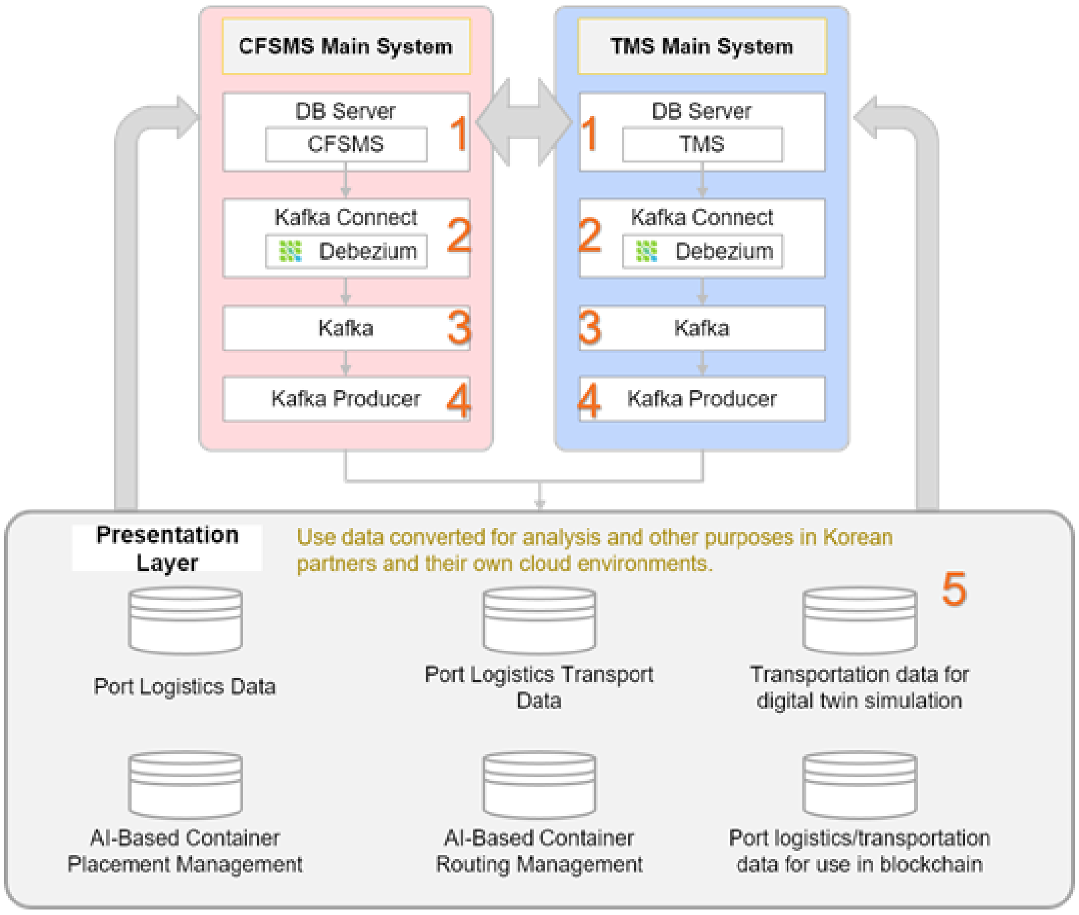
④ Transportation data needed to optimize shuttle transport with digital twin simulation.

⑤ Secure data sharing using private blockchain for logistics optimization.

⑥ Data Standardization Column Definition Form.

- **Data Model (if applicable):** Overview of the data structure and relationships.
- **Data Processing Pipelines (if applicable):** Description of how data is processed and transformed within the application.

1) Data Collection and Flow


CFSMS Main System
DB Server
CFSMS
1
Kafka Connect
Debezium
2
Kafka
3
Kafka Producer
4
TMS Main System
DB Server
TMS
1
Kafka Connect
Debezium
2
Kafka
3
Kafka Producer
4
Presentation Layer
Use data converted for analysis and other purposes in Korean partners and their own cloud environments.
5
Port Logistics Data
Port Logistics Transport Data
Transportation data for digital twin simulation
AI-Based Container Placement Management
AI-Based Container Routing Management
Port logistics/transportation data for use in blockchain

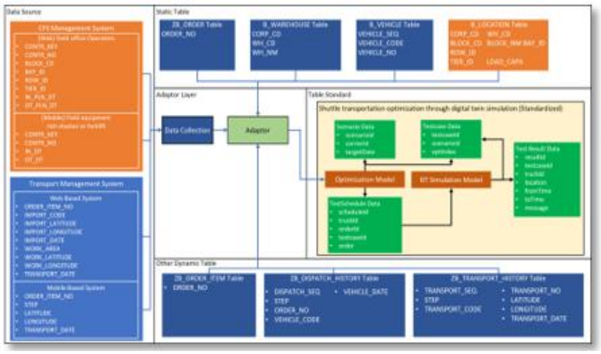
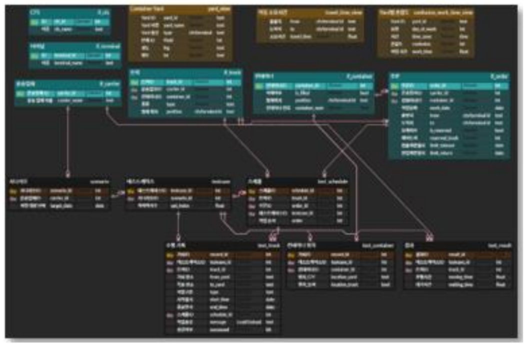
① Kafka System Configuration Chart

Sqlserver -> debezium -> debezium source connect -> kafka topic -> kafka sink connect

-> sink database

② Data conversion: Converting data to a single-message conversion (SMT), such as NewRecordState, in Kafka Sync Connect settingsKafka System Configuration Chart

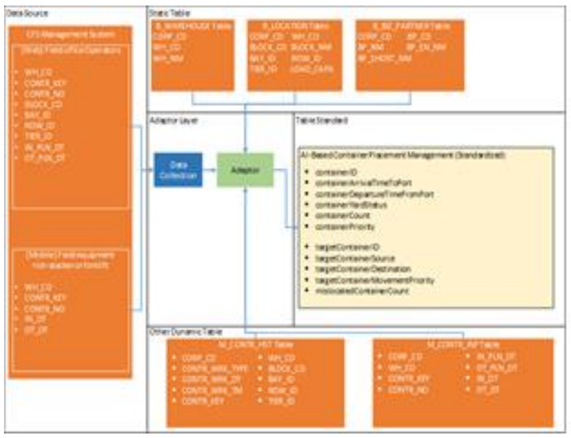
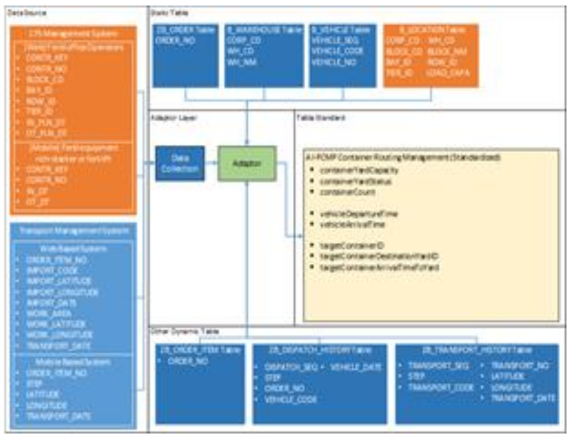
③ Data exchange of digital twin-based optimal dispatch system services.

· The Digital Twin-based optimal dispatch system service plans to exchange data with the platform through a data pipeline. Currently, to complete the pipeline development, we are actively communicating with the leading organization to finalize the data pipeline.

· To facilitate this, we have completed the database (DB) design based on data standardization in advance and conducted application tests by integrating sample data into the system. Once the pipeline is completed, we plan to finalize the integration between the platform and the optimal dispatch system service.

① Data exchange of AI-based CFS/CY management services

· For container placement management, data are collected from CFS management systems and field equipment used by on-site operators. Information such as container identifiers, block and bay positions, tier levels, and planned in/out times is retrieved from both web-based and mobile interfaces. These raw data are processed through an adaptor layer and mapped to a unified structure that includes container yard status, placement priority, and mislocation tracking. Supporting metadata such as warehouse and block information is linked through reference tables, and dynamic work logs are recorded for historical analysis.

· For container routing management, data related to container distribution and yard capacity across freight stations are collected from CFS systems and transport management systems. Each container is assigned a destination yard based on standardized fields such as current yard status, container count, and destination yard assignment. All collected data are integrated through the adaptor layer and transformed into a consistent format that supports decision-making for inter-station operations.

4.11 Use Case Application #11

In all our projects, the Flight Management System (FMS) served as the core dataset, providing fundamental flight information. For each specific project, we actively sought out and integrated additional variables that could significantly enhance predictive accuracy and model relevance.

For the Flight Delay Prediction project, a crucial addition was the METAR (Meteorological Aerodrome Report) data. These reports, issued by meteorological observation offices at airports, provide critical atmospheric conditions for aviation. Published at 30-minute intervals for airlines, METAR data allowed us to extract detailed runway weather conditions at or very close to the aircraft's landing time. This was particularly vital, as adverse weather events are identified by EUROCONTROL as major contributors to flight delays. Consequently, specific variables derived from these METAR reports, directly linked to common weather-induced delays, were integrated into our modeling phase. This enriched dataset was instrumental in building a more comprehensive and accurate delay prediction model.

Flight Management System			
Flight Features	**Aircraft Features**	**Calculated Features**	**METAR Features**
origin_airport	aircraft_seatCapacity	pair_flight_time	wind_dir
stad	aircraft_type_icao	pair_flight_route_change	wind_speed
atad		pair_flight_seat_change	visibility
airline_iatacode		pair_flight_service_change	present_weather
flightNumber		flight_distance	pressure
serviceType_code		next_flight_distance	
routeType		last_1_days_avg_flights	
origin_publicHoliday		last_2_days_avg_flights	
destination_publicHoliday		last_3_days_avg_flights	
		last_7_days_avg_flights	
		last_14_days_avg_flights	
		last_21_days_avg_flights	
		last_30_days_avg_delays_of_airports	
		last_60_days_avg_delays_of_airports	
		last_30_days_avg_delays_of_airline	
		last_60_days_avg_delays_of_airline	
		regular_flight	

For the ADS-B Milestone project, we strategically acquired a comprehensive dataset of ADS-B transmissions specifically for İzmir Adnan Menderes Airport (ADB). This dataset encompasses 115 distinct routes and a total of 6,965 flights. The high-frequency positional data obtained from these ADS-B transmissions, captured from the moment an aircraft commences movement, proved instrumental. This rich detail allowed us to achieve Actual Time of Arrival at Destination (ATAD) predictions with a remarkably low margin of error, significantly enhancing the precision of our short-term forecasts.

ADSB Data	timestamp, lat, lon, alt, gspeed, vspeed, tracks, squawk, callsign, source, fr24_id

For the Flight Services project, our focus was on optimizing ground operations by predicting the precise start and end times for each specific task associated with a flight. Leveraging data from the FMS (Flight Management System), we developed models that provide granular estimations for various services. This includes everything from boarding and baggage handling to security and aircraft pushback, ensuring more efficient resource allocation and smoother turnarounds at the gate.

5 . Data Analysis and Results (Specific to the Use Case Applications)

5.1 Use case #1 (INOSENS)

5.1.1 Data Labeling

Following the data collection, the human point clouds within the recorded LiDAR data were manually annotated. In total, around 100 point cloud samples were labeled and prepared for model training.

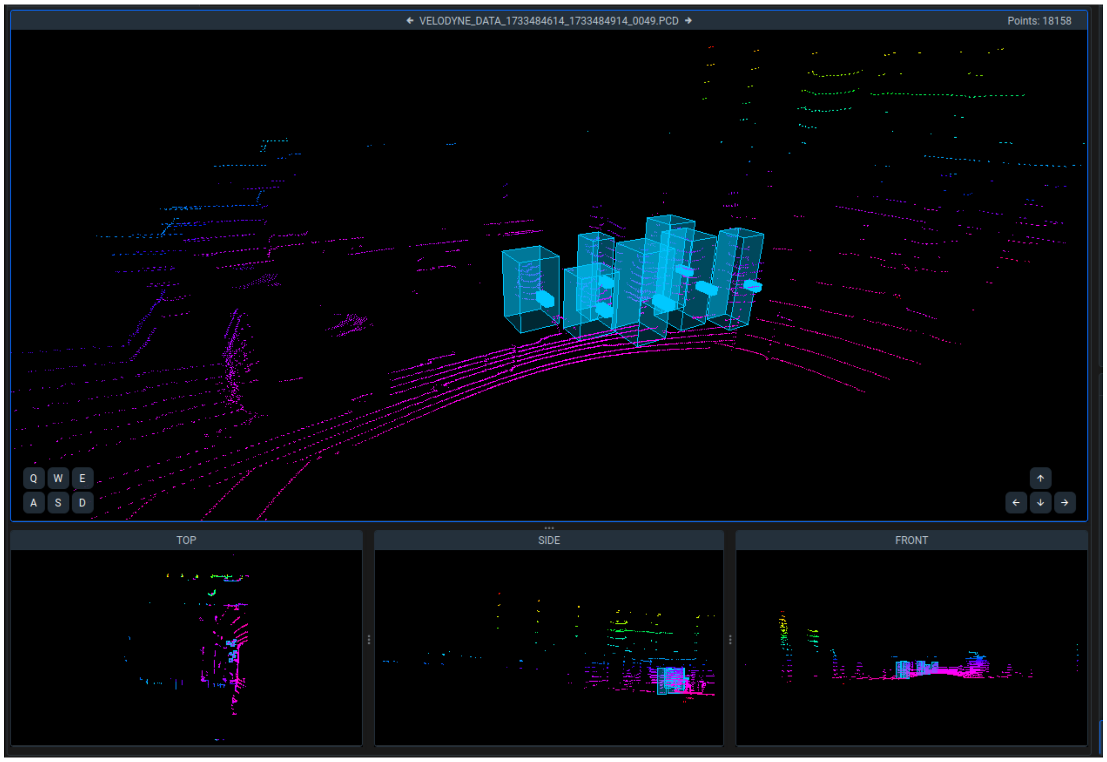


Figure. Data Labeling of LiDAR Point Cloud Data Collected in Adnan Menderes Airport

5.1.2 Model Training and Testing

Following the annotation process, model development was carried out for 3D human detection. In this phase, several model architectures such as PV-RCNN and SECOND were trained and evaluated. The performance of each model was compared using the Average Precision (AP) score — a common evaluation metric in object detection that reflects the area under the precision-recall curve, providing a measure of both detection accuracy and consistency. These initial models yielded very low performance on the airport lidar dataset. Subsequently, transfer learning was applied to the VoxelNeXt model, which had been previously trained on the nuScenes dataset. The model was fine-tuned using the labeled LiDAR point cloud data collected from the airport, and the following results were obtained.

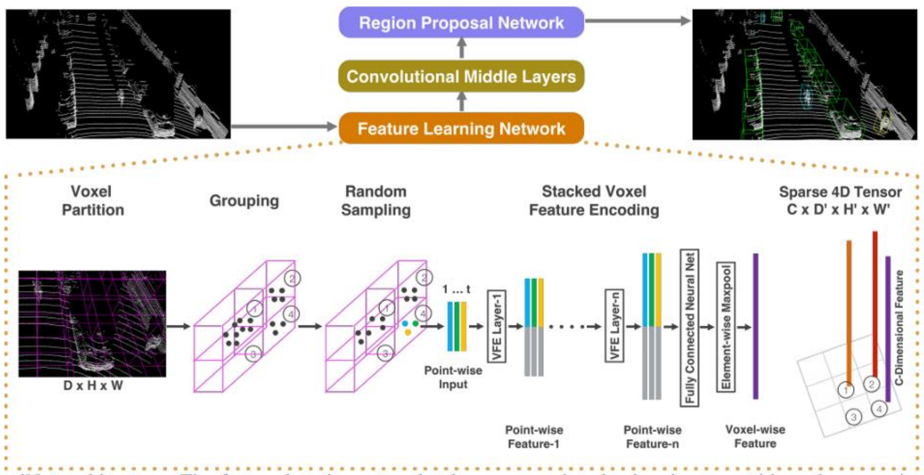


Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

Figure. Architecture of VoxelNeXt Model

(Source Paper: https://arxiv.org/pdf/1711.06396)

VoxelNeXt, specialized version of VoxelNet architecture and a state-of-the-art 3D object detection model, utilizes sparse voxel-based feature encoding and a center-based detection head to efficiently detect objects in LiDAR point cloud data. It builds upon voxel-based architectures like VoxelNet and SECOND-Net, introducing improved kernel designs and a simplified pipeline for better generalization and computational efficiency. In this study, VoxelNeXt was fine-tuned using transfer learning on a custom airport dataset consisting of 100 labeled LiDAR point clouds. The dataset was collected within an airport environment to enable human detection, crowd flow monitoring, and density estimation.

Table. Evaluation Scores of VoxelNeXt After Training Phase

Metric Type	AP Score (%)
BBox (Bounding Box)	95.77
BEV (Bird's Eye View)	81.95

3D	78.92

The fine-tuned VoxelNeXt model demonstrated strong performance in detecting pedestrians within the airport LiDAR data. The 2D bounding box accuracy (BBox AP) reached 95.77%, indicating that the model was highly capable of localizing pedestrians in image space. The Bird's Eye View (BEV) AP scored 81.95%, showing that spatial placement from a top-down perspective was also reliable. The 3D detection AP, which reflects full object localization in three-dimensional space, reached 78.92%, suggesting a solid performance in estimating object size, position, and depth. To improve model performance, future work will focus on continued data annotation efforts and increasing the number of labeled samples.

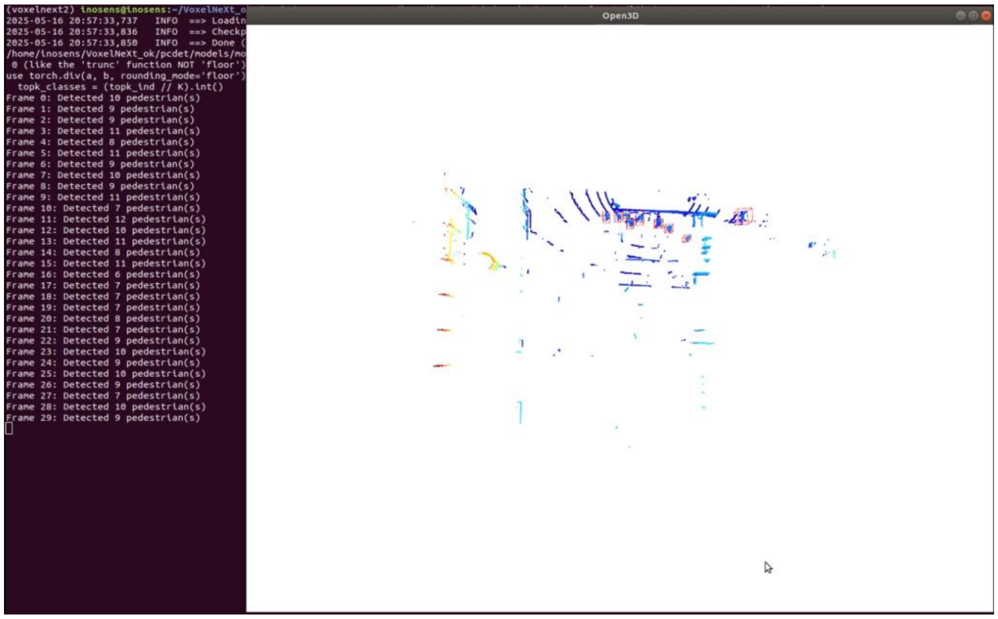

```
(voxelnext2) inosens@inosens:~/VoxelNeXt_o
2025-05-16 20:57:33,737   INFO  ==> Loadin
2025-05-16 20:57:33,836   INFO  ==> Checkp
2025-05-16 20:57:33,850   INFO  ==> Done (
/home/inosens/VoxelNeXt_ok/pcdet/models/mo
 0 (like the 'trunc' function NOT 'floor')
use torch.div(a, b, rounding_mode='floor')
  topk_classes = (topk_ind // K).int()
Frame 0: Detected 10 pedestrian(s)
Frame 1: Detected 9 pedestrian(s)
Frame 2: Detected 9 pedestrian(s)
Frame 3: Detected 11 pedestrian(s)
Frame 4: Detected 8 pedestrian(s)
Frame 5: Detected 11 pedestrian(s)
Frame 6: Detected 9 pedestrian(s)
Frame 7: Detected 10 pedestrian(s)
Frame 8: Detected 9 pedestrian(s)
Frame 9: Detected 11 pedestrian(s)
Frame 10: Detected 7 pedestrian(s)
Frame 11: Detected 12 pedestrian(s)
Frame 12: Detected 10 pedestrian(s)
Frame 13: Detected 11 pedestrian(s)
Frame 14: Detected 8 pedestrian(s)
Frame 15: Detected 11 pedestrian(s)
Frame 16: Detected 6 pedestrian(s)
Frame 17: Detected 7 pedestrian(s)
Frame 18: Detected 7 pedestrian(s)
Frame 19: Detected 7 pedestrian(s)
Frame 20: Detected 8 pedestrian(s)
Frame 21: Detected 7 pedestrian(s)
Frame 22: Detected 9 pedestrian(s)
Frame 23: Detected 10 pedestrian(s)
Frame 24: Detected 9 pedestrian(s)
Frame 25: Detected 10 pedestrian(s)
Frame 26: Detected 9 pedestrian(s)
Frame 27: Detected 7 pedestrian(s)
Frame 28: Detected 10 pedestrian(s)
Frame 29: Detected 9 pedestrian(s)
```

Figure. An Example of Trained VoxelNeXt Model's 3D Box Predictions on a Test Point Cloud

5.2 Use case #2 (Siemens)

5.2.1 Data Preprocessing:

The raw data is transformed into clean data via several preprocessing steps. These steps are divided into three major parts as data pre-processing, outlier analysis and repetition check. Data preprocessing operations are done through data simplification and data compression. For each baggage data, feature column occupancy rate is calculated. The columns with occupancy rates below 100% are identified. The columns with empty cells are joined to achieve 100% occupancy.

- 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

- The rest of the columns with the empty cell are excluded.

After completing the pre-processing steps, the following data analysis is conducted to help with the rest of the preprocessing steps. For each flight code with departure status, hourly baggage arrival pattern before flight time is extracted in the following table. For example, flight code 1 is repeated 13 times for the one month period and 100 baggage has arrived when there is more than 1 hour and less than 2 hours before flight time.

Table: Hourly Baggage Arrival Pattern per Flight Code.

Flight Code	**Total Baggage Count**	**Flight Count**	**Hourly Flight Pattern (1 Hour, 2 Hours, 3 Hours, ...)**
1	308	13	[0,100,0, …]
2	2656	22	[0,2,867, …]
3	5325	23	[0,27,33, …]

For each flight code, average baggage arrival versus time before flight is plotted as in the following figure. Each different color represents a distinct flight code. Excessive baggage count is observed as 1750. Also, it is found that there can be baggage arrivals even 12 hours before the flight. In this initial representation before the data cleaning and preprocessing steps, there are many outliers regarding the arrival pattern of baggage records in the following figure. Many baggage records seem processed before too early hours than flight time.

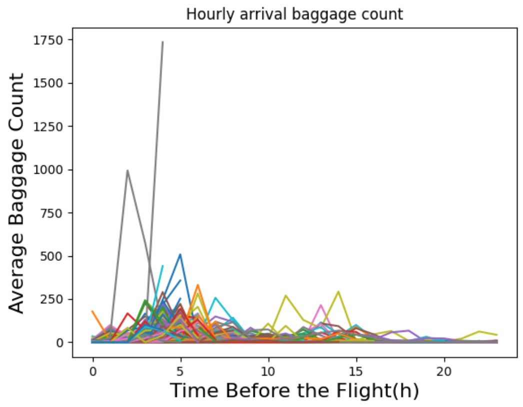


Figure. Hourly Baggage Arrival Pattern per Flight Code (Initial)

Outlier baggage records are excluded based on rules such as empty PASSENGER_ID, BIM_CREATE_DATE > ACTUAL_TIME and earlier time stamps do not exist in a predetermined 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 the following figure. Furthermore, it can be seen that the arrival pattern of baggage records is distributed in a reasonable way. The following figure indicates the most activity concentrated in the few hours leading up to the flight. Finally, the dimensionality reduction is obtained like from (858K, 82) to (685K, 23) after data cleaning, feature selection and merge operations.

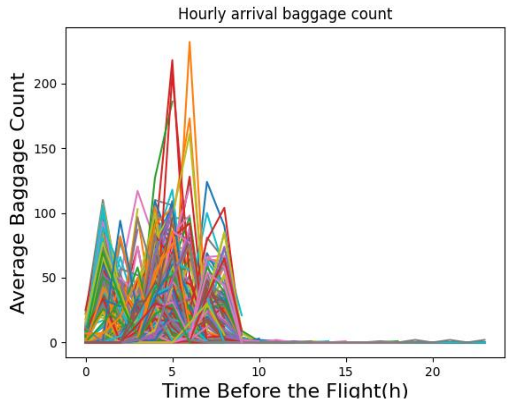


Figure. Hourly Baggage Arrival Pattern per Flight Code (Final)

5.2.2 Data Storage:

The data can be classified as structured or unstructured. Structured data fits neatly into tables, while unstructured data cannot be easily mapped. The traditional Relational Database Management Systems (RDBMS) excel at handling structured data, but struggle with large volumes of structured or unstructured data. To address this, NoSQL databases have emerged as a more scalable, flexible, and distributed solution capable of efficiently managing both structured and unstructured big data. These advantages make NoSQL systems preferable for big data projects compared to the traditional RDBMS technologies.

This study area deals with big data regarding millions of flights and baggage data with future projection. Additionally, it requires it to be expanded with real-time updates. Thousands of new rows will be added daily from the airport system with each flight, and new columns may be added as well. This will result in a high volume of data and transactions. Given these requirements, a NoSQL data management system is the most suitable solution for this big data study.

There are many NoSQL database technologies used in the software field. After the research and requirement analysis process, it is decided that Cassandra has been chosen as the best option for this study. It is a wide-column store database, which is a type of NoSQL database that can be used for relational types of big data. Cassandra is an open-source database that can be distributed across multiple machines. It also provides a scalable, maintainable system with high performance and fault tolerance.

- **Daily Baggage Prediction:**

Throughout the project's lifecycle, the dataset has evolved significantly through continuous data contributions from TAV Technology. With each dataset update, both machine learning and deep learning models undergo retraining and testing to evaluate potential improvements in accuracy. This iterative process involves training models from scratch when the dataset size increases, allowing us to identify any shifts in model performance and accuracy patterns. Analysis of the second-largest dataset (v7) revealed that the Random Forest model achieved the highest accuracy. During feature engineering with this dataset, we discovered that while some features had a significant positive impact on model accuracy, others showed negative effects.

The following table provides detailed results showing the impact of both selected and excluded features on model performance. However, when testing with the latest and largest dataset (v8), the model performance hierarchy changed. The CatBoost model surpassed the Random Forest model, achieving a lower Mean Absolute Error (MAE). As a result, CatBoost is currently implemented as the primary model for daily baggage prediction in the project.

Table. Results of feature selection in the clean dataset v7

Selected Features	Mean Absolute Error (MAE)	Excluded Features
is_holiday, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	159.11	flight_code, dep_ap_code, arr_ap_code, category
flight_code, is_holiday, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	18.57	dep_ap_code, arr_ap_code, category
category, dep_ap_code, arr_ap_code, is_holiday, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	25.97	flight_code
flight_code, category, is_holiday, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	17.92	dep_ap_code, arr_ap_code
flight_code, category, dep_ap_code, arr_ap_code, is_holiday, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	17.55	
flight_code, category, dep_ap_code, arr_ap_code, is_weekend, hour_of_day, day_of_week, day_of_month, month, season	17.49	is_holiday
flight_code, category, dep_ap_code, arr_ap_code, hour_of_day, day_of_week, day_of_month, month, season	17.51	is_holiday, is_weekend
flight_code, category, dep_ap_code, arr_ap_code, is_holiday, hour_of_day, day_of_week, day_of_month, month, season	17.54	is_weekend
flight_code, category, dep_ap_code, arr_ap_code, day_of_week, day_of_month, month, season	19.37	is_holiday, is_weekend, hour_of_day
flight_code, category, dep_ap_code, arr_ap_code, hour_of_day, day_of_month, month, season	17.98	is_holiday, is_weekend, day_of_week
flight_code, category, dep_ap_code, arr_ap_code, hour_of_day, day_of_week, month, season	96.19	is_holiday, is_weekend, day_of_month
flight_code, category, dep_ap_code, arr_ap_code, hour_of_day, day_of_week, day_of_month, season	21.00	is_holiday, is_weekend, month
flight_code, category, dep_ap_code, arr_ap_code, hour_of_day, day_of_week, day_of_month, month	17.62	is_holiday, is_weekend, season

Table. AI model comparison for the latest and the largest clean dataset v8

AI Model	Mean Absolute Error (MAE)
Random Forest	17.51
XG-Boost	17.85
Light-GBM	17.06
CatBoost	16.81
CNN-LSTM Hybrid Model	24.75

- **Pattern prediction:**

In this analysis, we evaluated two versions of the clean dataset (v7 and v8) using various machine learning, deep learning, and time-series approaches. The performance assessment incorporated multiple evaluation metrics, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R² scores. Summary of the results are displayed in the following table. Based on the Validation Mean MAE and Test MAE scores, the ExtraTreeRegressor demonstrated superior performance. However, its significant model size of 1.2GB presents a practical limitation. The BaggingRegressor emerged as the second-best performer with a MAE score of 0.020, while both Deep Neural Networks (DNN) and Random Forest algorithms also showed promising results. Support Vector Machine (SVM) exhibited the poorest performance, notably due to its negative R² score, indicating its inadequacy for this task.

Table. Performance Evaluation of AI Models

se	ataset	odel	alidation Mean MSE	alidation Mean RMSE	alidation Mean MAE	alidation Mean R2	st MSE	st RMSE	st MAE	st R2
1	bag_clean_v7	NN	0,001	0,033	0,022	0,382	0,001	0,032	0,022	0,440
2	bag_clean_v8	NN	0,001	0,035	0,024	0,406	0,001	0,034	0,024	0,428
3	bag_clean_v8	ndomForest	0,001	0,031	0,022	0,519	0,001	0,031	0,022	0,519
4	bag_clean_v8	radient Boosting	0,001	0,034	0,025	0,416	0,001	0,035	0,025	0,411
5	bag_clean_v8	traTreeRegressor	0,001	0,028	0,019	0,597	0,001	0,028	0,019	0,607
6	bag_clean_v8	ecisionTree	0,001	0,033	0,023	0,472	0,001	0,033	0,023	0,476
7	bag_clean_v8	nearRegression	0,002	0,044	0,033	0,046	0,002	0,044	0,033	0,051
8	bag_clean_v8	M	0,004	0,065	0,058	-1,094	0,004	0,065	0,058	-1,056
9	bag_clean_v8	uberRegressor	0,002	0,045	0,032	0,006	0,002	0,045	0,032	0,008
10	bag_clean_v8	ssiveAggressiveRegressor	0,004	0,063	0,056	-0,981	0,004	0,063	0,055	-0,922
11	bag_clean_v8	uantileRegressor	0,002	0,046	0,032	-0,077	0,002	0,047	0,033	-0,077
12	bag_clean_v8	DRegressor	0,002	0,044	0,033	0,043	0,002	0,044	0,033	0,042
13	bag_clean_v8	oissonRegressor	0,002	0,045	0,034	0,034	0,002	0,045	0,034	0,002
14	bag_clean_v8	asticNet	0,002	0,045	0,034	0,000	0,002	0,045	0,034	0,000
15	bag_clean_v8	dge	0,002	0,044	0,033	0,045	0,002	0,044	0,033	0,049
16	bag_clean_v8	sso	0,002	0,045	0,034	0,000	0,002	0,045	0,034	0,000

17	bag_clean_v8	ggingRegressor	0,001	0,029	0,020	0,578	0,001	0,029	0,020	0,590
18	bag_clean_v8	laBoostRegressor	0,002	0,049	0,041	-0,215	0,004	0,066	0,057	-1,099
19	bag_clean_v8	traTreeRegressor	0,001	0,032	0,023	0,477	0,001	0,033	0,023	0,472
20	bag_clean_v8	TM	0,003	0,050	0,035	0,126	0,001	0,037	0,027	0,195
21	bag_clean_v8	TM	0,003	0,050	0,035	0,137	0,001	0,034	0,024	0,319
22	bag_clean_v8	TM	0,003	0,050	0,035	0,125	0,001	0,033	0,023	0,364
23	bag_clean_v8	TM	0,002	0,045	0,031	0,296	0,001	0,033	0,023	0,356
24	bag_clean_v8	TM	0,002	0,048	0,034	0,189	0,001	0,033	0,023	0,366
25	bag_clean_v8	TM	0,002	0,045	0,032	0,295	0,001	0,033	0,023	0,348
26	bag_clean_v8	.inear	0,284	N/A	N/A	N/A	0,216	N/A	0,212	N/A

SHAP (SHapley Additive exPlanations) values are a way to explain the output of any machine learning model. It uses a game theoretic approach that measures each player's contribution to the final outcome. In machine learning, each feature is assigned an important value representing its contribution to the model's output. This approach is applied to the RandomForest model and the result is displayed in Figure9. The most influential feature is TIME_HOUR, which shows a wide distribution of SHAP values, implying a strong effect on the prediction, especially for lower and higher hours of the day. TIME_MINUTE, DAY_OF_YEAR, and TIME_OF_DAY also show significant influence, suggesting that minute-level and seasonal timing factors contribute meaningfully to the model's decision. Other features like BUSY_PERIOD, CATEGORY, and WEEKDAY have more modest yet non-negligible impacts. Features like IS_HOLIDAY and QUARTER appear to contribute the least, as their SHAP values are tightly clustered near zero. Overall, the model seems highly sensitive to fine-grained temporal attributes, which aligns with scenarios requiring precise time-based predictions.

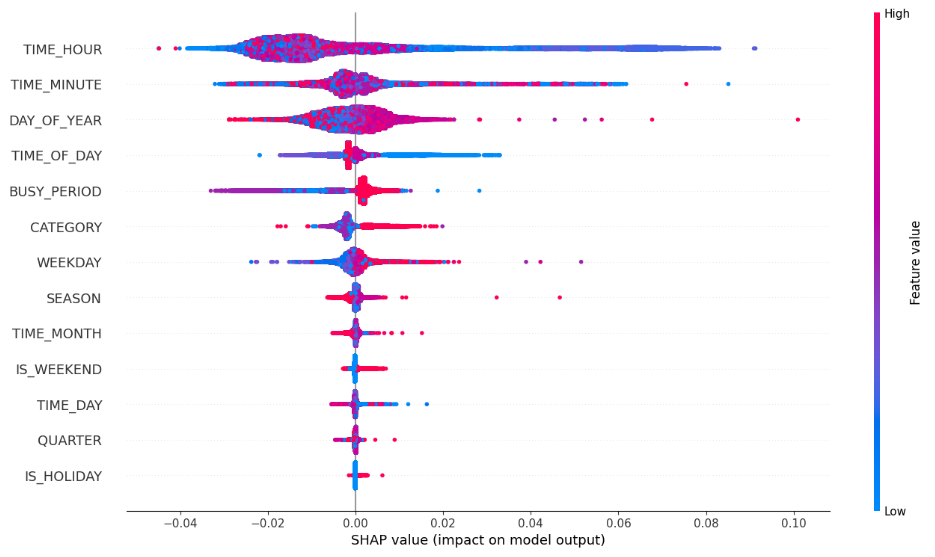


Figure. SHAP Analysis of RandomForest Model

The following chart illustrates the distribution of airport baggage data by flight category. Most baggage records, accounting for 56.4%, are associated with international flights, while domestic flights make up 43.5% of the observations. Negligible 0.0639% of the data falls under the "EMPTY" category, indicating either missing or unclassified entries. Overall, the data shows a slightly higher volume of baggage activity linked to international travel, with minimal data quality issues in the CATEGORY field.

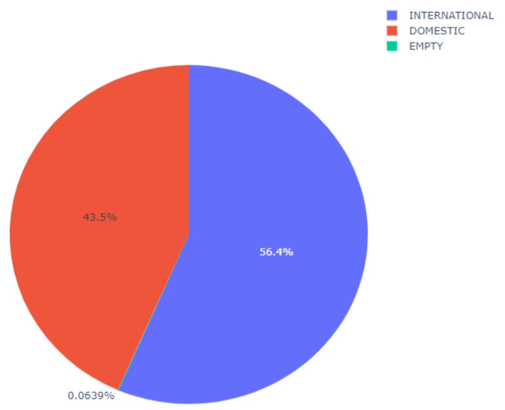
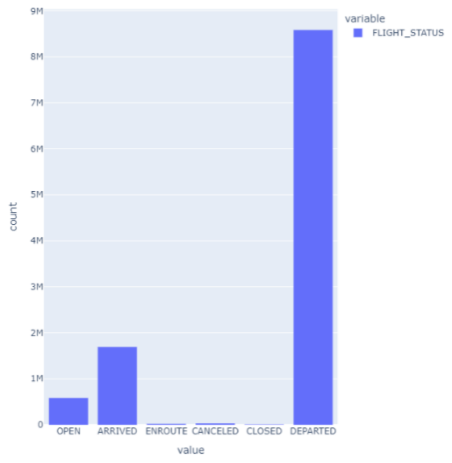


Figure. Distribution of Flight Category

That chart presents the distribution of baggage records based on the flight status across all observations. The majority of the data, by a substantial margin, corresponds to the **DEPARTED** status, with approximately 8.6 million entries, indicating that most baggage records are linked to flights that have already left. **ARRIVED** flights follow with a significantly smaller count, around 1.7 million, while **OPEN** flights account for just under 700,000 records. The statuses **ENROUTE**, **CANCELED**, and **CLOSED** represent only a negligible fraction of the dataset. This distribution suggests that the dataset primarily captures baggage activity tied to completed departures, which is valuable for analyzing operational throughput and historical performance.



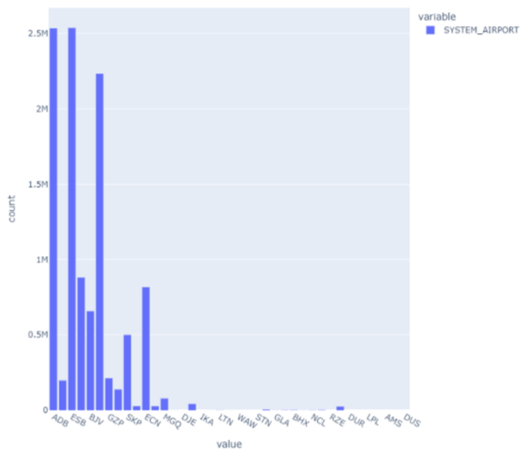


Figure. Distribution of Flight Status

Figure. Distribution of Flight System Airport

This last chart shows the distribution of baggage records across different system airports. The highest bag counts are observed for three airports—**ADB**, **ESB**, and **SAW**—each with over 2 million observations, indicating that these are the primary hubs in the dataset. Other airports such as **BJV**, **GZP**, **ECN**, and **SKP** also show considerable activity, ranging from several hundred thousand to just under a million records. The remaining airports have significantly lower counts, with many showing only minimal baggage activity.

- **Flight Delay Prediction:**

XGBoost demonstrates the best performance with the lowest Mean Absolute Error (MAE) of 7.622, followed closely by LightGBM with an MAE of 7.8514. These two-gradient boosting-based algorithms significantly outperform the other models in the comparison. Random Forest and Gradient Boosting show similar performance levels, with MAE scores of 8.4194 and 8.4234 respectively, placing them in the middle range of our comparison. However, they're notably less accurate than XGBoost and LightGBM. The poorest performing models are AdaBoost and DNN, with considerably higher MAE scores of 10.1882 and 10.31 respectively. This indicates that these models have, on average, larger prediction errors compared to the other approaches.

If we prioritize MAE as our primary evaluation metric, the clear recommendation would be to use XGBoost for this problem, as it shows approximately 2.9% better performance than LightGBM and about 26% better performance than the worst-performing model (DNN). The superior performance of XGBoost is also supported by its better scores across other metrics (MSE, RMSE, and R²), suggesting it provides the most reliable and accurate predictions overall.

Table. Model Comparison

Model	Test MAE	Test MSE	Test RMSE	Test R^2
(GBoost	7,622	112,1498	10,5901	0,5375
ightGBM	7,8514	118,188	10,8714	0,5126
andom Forest	8,4194	134,2991	11,5887	0,4461
radient Boosting	8,4234	134,9552	11,617	0,4434
daBoost	10,1882	166,121	12,8888	0,3149
NN	10,31	258,11	16,07	0,2742

5.4 Use case #4 (INOSENS)

For the data analysis component of our study, experiments were performed on data from two sources. The first is a set of comments the Inosens team obtained from TAV Technologies about Izmir Airport, and the second is Ekşi Sözlük (a popular Turkish-language website where users discuss a variety of topics). The first set of data visualizations shows the number of comments posted in a given language in each month within a specific timeline. The second set of data visualizations shows the number of comments for each predicted sentiment in each month within the same timeline for each set of comments.

First, the language of each comment was predicted using an open source RoBERTa-based model.

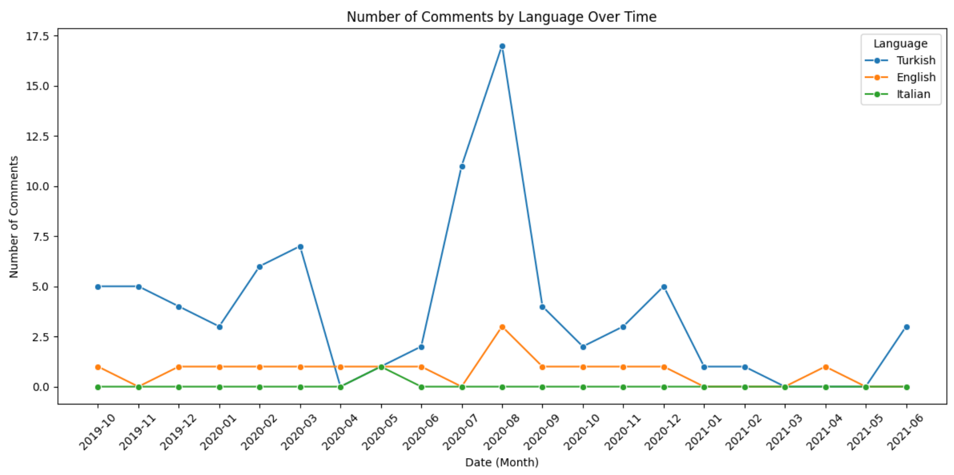


Figure. Detected language for each of the comments from TAV Technologies

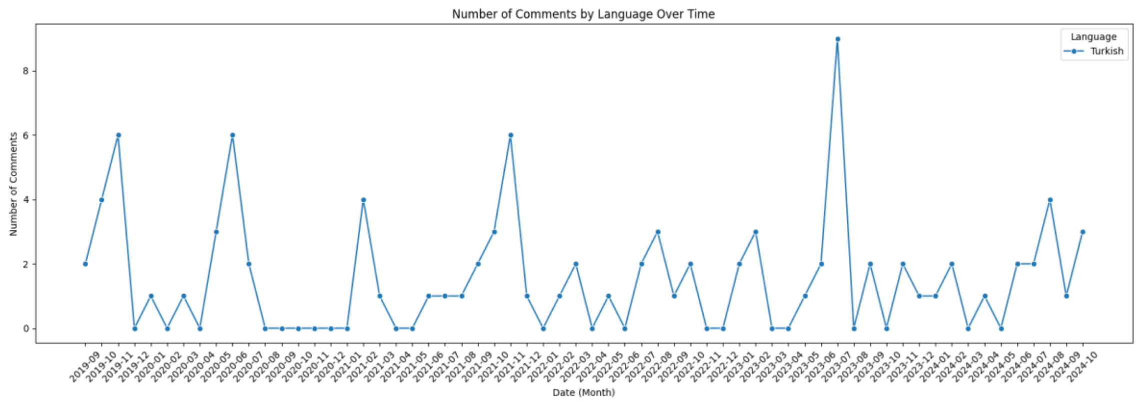


Figure. All comments from Ekşi Sözlük are in Turkish, as detected by the model

Then the sentiment of each comment was predicted using an open source BERT-based model. The model was specifically trained on Turkish text, so in this study, the model was only implemented on the Turkish comments.

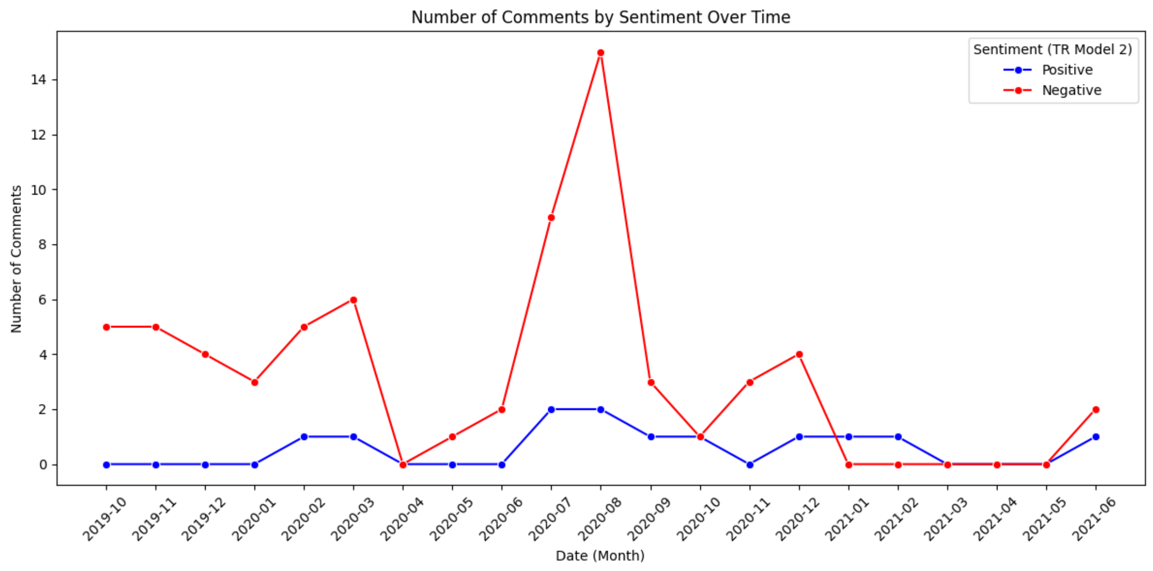


Figure. Detected sentiment for each of the comments from TAV Technologies

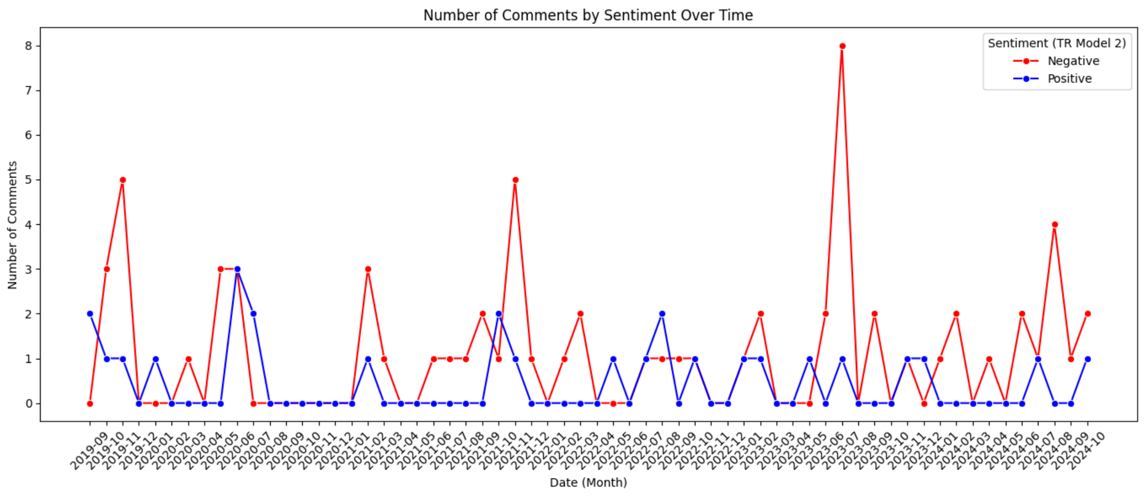


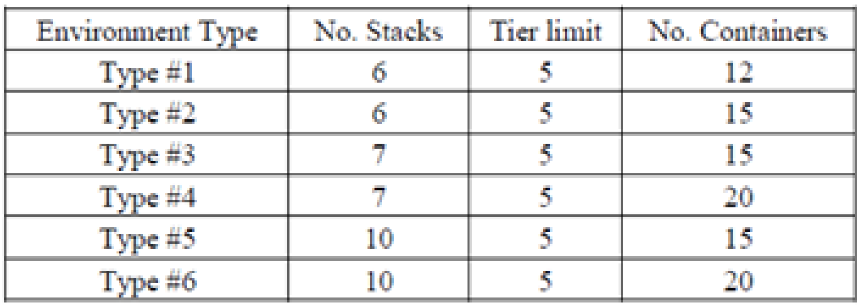
Figure. Detected sentiment for each of the comments from Ekşi Sözlük

5.5 Use case Application #5 (Korean Consortium)

- AI-based CFS/CY Management Data Analysis and Results

  - To develop container relocation planning technology for AI-based CFS/CY management system, container placement information and schedule information within the CFS/CY were utilized. Each container data consists of a unique container ID, placement information consisting of bay, row, and tier, and retrieval schedule information.

- In order to evaluate the performance of the container relocation planning model, a simulation environment was configured with the following six different container yard placement configurations.

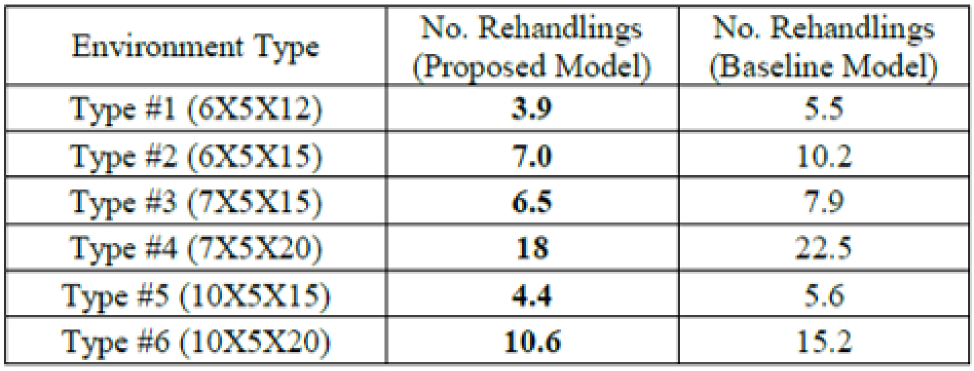


Environment Type	No. Stacks	Tier limit	No. Containers
Type #1	6	5	12
Type #2	6	5	15
Type #3	7	5	15
Type #4	7	5	20
Type #5	10	5	15
Type #6	10	5	20

- The target state was set to a state where no more container rehandling was required, and the number of container rehandlings required to reach the target state was set as an evaluation index, and the average performance was measured by executing 1,000 episodes for each environment.

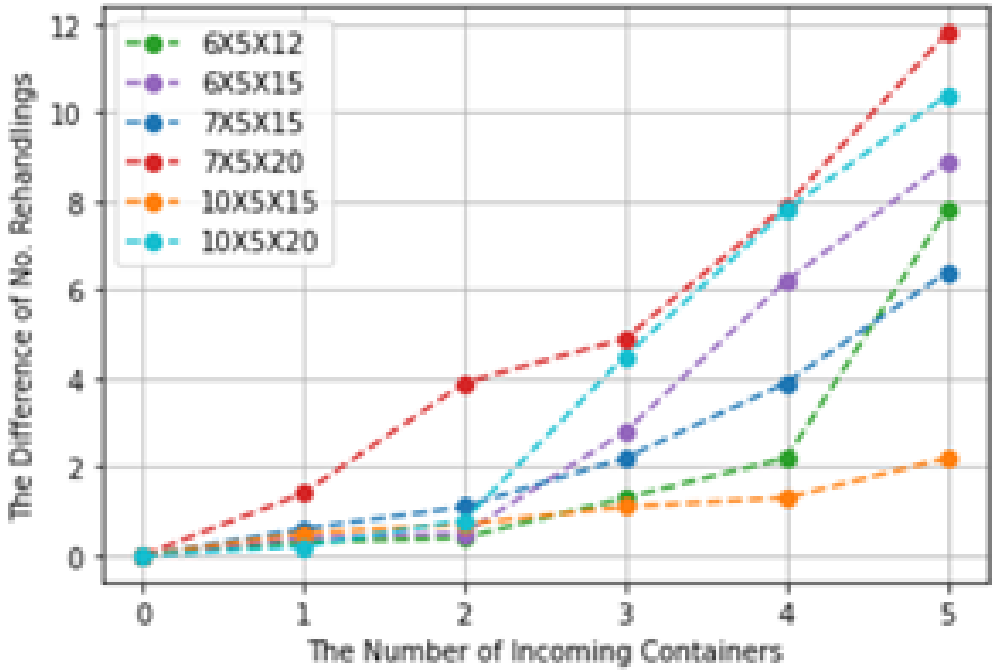
- Performance evaluations were conducted on the baseline model, which performs optimal placement by considering only the current status without considering scheduled incoming containers, and the proposed (developed) model, which performs optimal placement by considering the information on the scheduled incoming containers together.

- The experimental results confirmed that the optimal placement state could be reached with fewer containers rehandlings in all environment settings when the information on containers scheduled to be brought in was utilized together.



Environment Type	No. Rehandlings (Proposed Model)	No. Rehandlings (Baseline Model)
Type #1 (6X5X12)	**3.9**	5.5
Type #2 (6X5X15)	**7.0**	10.2
Type #3 (7X5X15)	**6.5**	7.9
Type #4 (7X5X20)	**18**	22.5
Type #5 (10X5X15)	**4.4**	5.6
Type #6 (10X5X20)	**10.6**	15.2

- As a result of analyzing the performance difference according to the number of scheduled incoming containers, it can be confirmed that the performance advantage of developed model becomes more evident as the number of scheduled incoming containers, which shows that the developed model is effective for long-term optimization that takes into account future situations.



- Digital Twin-Based Optimized Dispatch Data Analysis and Results

- To develop and enhance the Digital Twin-based optimal dispatch system, actual dispatch operation records from previous periods were obtained. The acquired dispatch data was input into the developed simulation model to generate simulation results, which were then compared and analyzed with the simulation results of the dispatch data generated by the optimization module.

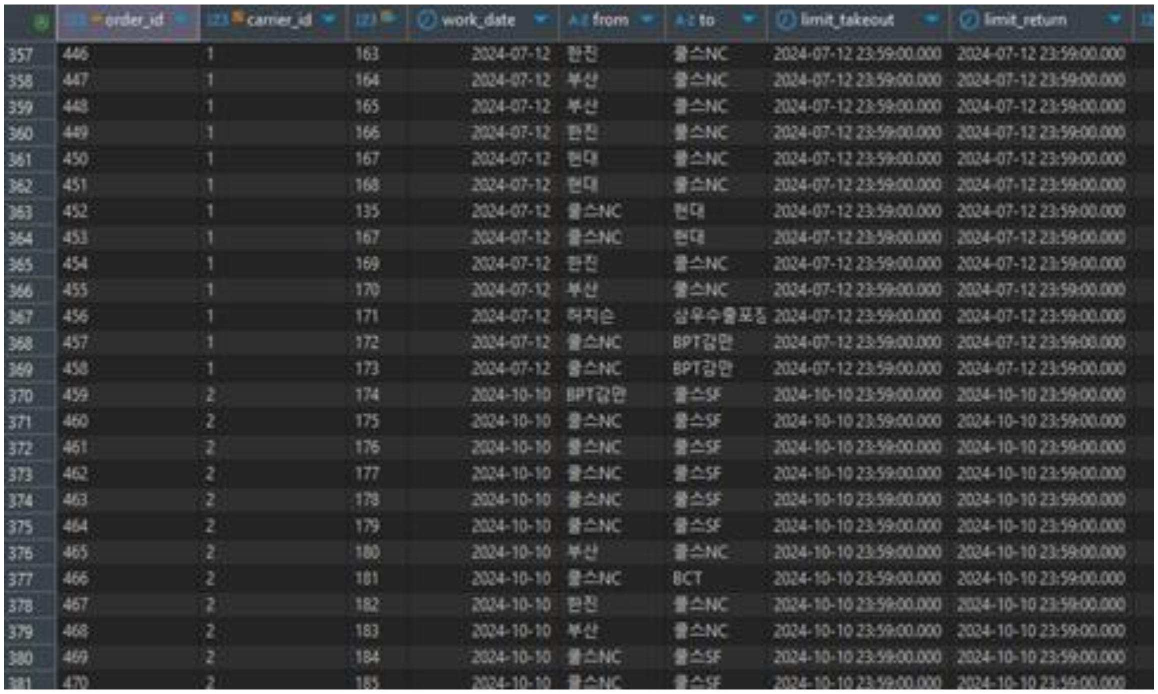
- During this process, data preprocessing was performed to address inconsistencies between the recorded data and the standardized DB schema. Additionally, unreliable data was identified and cleansed.

- When the optimal dispatch system is activated, it records various data, providing valuable business insights and allowing for monitoring of optimization results. For example, when the optimization module runs, it goes through several optimization stages driven by the algorithm. By recording the objective index (total operation time of all trucks) at each stage, it becomes possible to analyze and compare the optimization results for each stage.

- The graph above shows an example of the optimization module test results obtained by using both the Greedy algorithm and the SA algorithm. By comparing the total operation time of the simulated dispatch schedule with the total operation time of each optimization algorithm, it is possible to evaluate the performance of the optimization module. Based on these results, performance improvements can be achieved through parameter tuning of the algorithm or by setting a target completion time.

- Currently, it is expected that using the optimal dispatch system will exceed the target improvement rate of 10% in total operation time of trucks. Further research is ongoing to continuously enhance performance. This improvement is expected to reduce congestion in port logistics and decrease both operation time and costs.

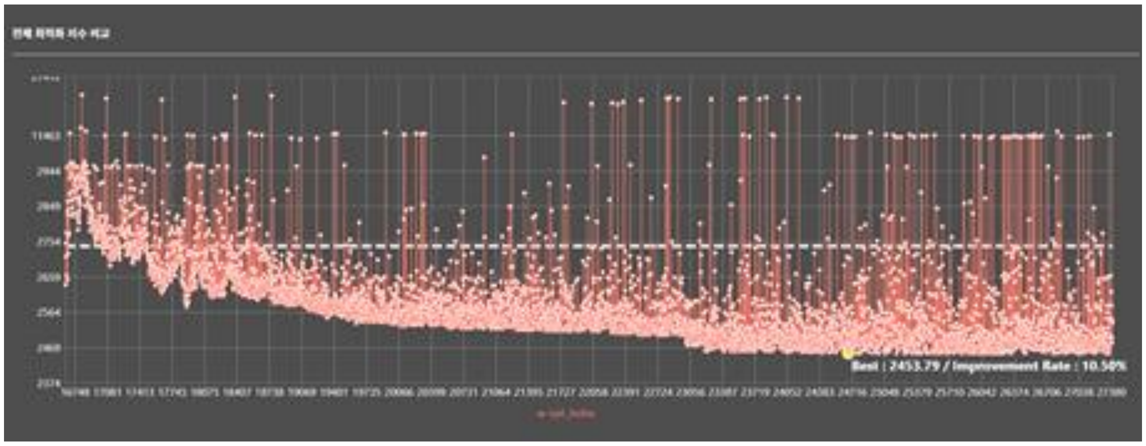
- Digital Twin-Based Optimized Dispatch Data Analysis and Results



	order_id	carrier_id		work_date	from	to	limit_takeout	limit_return
357	446	1	163	2024-07-12	한진	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
358	447	1	164	2024-07-12	부산	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
359	448	1	165	2024-07-12	부산	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
360	449	1	166	2024-07-12	한진	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
361	450	1	167	2024-07-12	현대	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
362	451	1	168	2024-07-12	현대	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
363	452	1	135	2024-07-12	울스NC	현대	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
364	453	1	167	2024-07-12	울스NC	현대	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
365	454	1	169	2024-07-12	한진	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
366	455	1	170	2024-07-12	부산	울스NC	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
367	456	1	171	2024-07-12	허치슨	삼우수출포장	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
368	457	1	172	2024-07-12	울스NC	BPT감만	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
369	458	1	173	2024-07-12	울스NC	BPT감만	2024-07-12 23:59:00.000	2024-07-12 23:59:00.000
370	459	2	174	2024-10-10	BPT감만	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
371	460	2	175	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
372	461	2	176	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
373	462	2	177	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
374	463	2	178	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
375	464	2	179	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
376	465	2	180	2024-10-10	부산	울스NC	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
377	466	2	181	2024-10-10	울스NC	BCT	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
378	467	2	182	2024-10-10	한진	울스NC	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
379	468	2	183	2024-10-10	부산	울스NC	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
380	469	2	184	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000
381	470	2	185	2024-10-10	울스NC	울스SF	2024-10-10 23:59:00.000	2024-10-10 23:59:00.000

- To develop and enhance the Digital Twin-based optimal dispatch system, actual dispatch operation records from previous periods were obtained. The acquired dispatch data was input into the developed simulation model to generate simulation results, which were then compared and analyzed with the simulation results of the dispatch data generated by the optimization module.

- During this process, data preprocessing was performed to address inconsistencies between the recorded data and the standardized DB schema. Additionally, unreliable data was identified and cleansed.



- When the optimal dispatch system is activated, it records various data, providing valuable business insights and allowing for monitoring of optimization results. For example, when the optimization module runs, it goes through several optimization stages driven by the algorithm. By recording the objective index (total operation time of all trucks) at each stage, it becomes possible to analyze and compare the optimization results for each stage.
- The graph above shows an example of the optimization module test results obtained by using both the Greedy algorithm and the SA algorithm. By comparing the total operation time of the simulated dispatch schedule with the total operation time of each optimization algorithm, it is possible to evaluate the performance of the optimization module. Based on these results, performance improvements can be achieved through parameter tuning of the algorithm or by setting a target completion time.
- Currently, it is expected that using the optimal dispatch system will exceed the target improvement rate of 10% in total operation time of trucks. Further research is ongoing to continuously enhance performance. This improvement is expected to reduce congestion in port logistics and decrease both operation time and costs.

5.11 Use case Application #11 (TAV)

5.11.1 Data Preprocessing

Raw ADS-B positional data, sourced from Flightradar24 (FR24), underwent a series of meticulous preprocessing steps. Each flight's ADS-B trajectory, encompassing timestamped location information, was transformed to enable the prediction of Actual Time of Arrival at Destination (ATAD). For every data point within a flight's trajectory, the time remaining until arrival was calculated. Additionally, comprehensive flight-specific information, such as scheduled times and aircraft details, was extracted from the Flight Management System (FMS) and integrated with the ADS-B data.

A critical aspect of our preprocessing involved the identification and removal of anomalous flight behaviors, such as holding patterns (aircraft circling in the air) and holding positions (aircraft waiting on the ground). These outlier scenarios were excluded from the training dataset. The rationale behind this exclusion is that such events are typically governed by real-time Air Traffic Control (ATC) directives and airport ground operations, making them more amenable to a rule-based approach rather than being accurately predicted by a generalized machine learning model trained on typical flight behavior.

5.11.2 Rule Based System

Beyond the primary objective of ATAD prediction, this project also aimed to optimize data acquisition costs. Instead of continuously purchasing high-volume ADS-B data for an entire flight duration, our strategy focuses on acquiring data only at key points where it is most critical. While ATAD prediction is feasible from a limited number of data points, accurately identifying various operational states of a flight is another key expectation.

To address these requirements, a Rule-Based System was developed and integrated alongside the AI model. This system is designed to provide real-time situational awareness by performing crucial state checks. These states include:

- Estimated Airport Arrival Status: Assessment of the aircraft's estimated arrival time at the destination airport, calculated using distance-based methods relative to its estimated time of departure.
- ADS-B Transmitter Activation: Verification of whether the pilot has activated the ADS-B radio transmitter at the estimated time of departure.
- On-Time Performance: Determination of whether the flight commenced on schedule or experienced a delay, quantifying the extent of any delay.
- Destination Check: Confirmation of whether the aircraft's movement is indeed destined for our specific airport.
- In-Flight Holding Detection: Identification of instances where aircraft are observed holding in the air due to airport congestion during the approach or landing phase.

This hybrid approach, combining predictive modeling with rule-based logic, allows for both accurate forecasting and precise operational monitoring, contributing to a more comprehensive and cost-effective solution.

5.11.3 Modelling

A robust machine learning model was developed to predict ATAD based on the aircraft's real-time position. Following extensive preprocessing and data transformation, the resulting voluminous dataset necessitated an efficient modeling approach. The Random Forest algorithm, implemented using the rapids-cuML library, a GPU-accelerated machine learning framework, demonstrated superior performance in our evaluations.

Model training and evaluation were performed using a 5-fold cross-validation strategy to ensure robust and generalizable performance across diverse flight conditions. Subsequent hyperparameter optimization, efficiently conducted using the Optuna framework, further refined the Random Forest model. This optimized model achieved a Mean Absolute Error (MAE) of approximately 109 seconds on the test set.

Model	Train Score (MAE)	Test Score (MAE)
XGBoost	6.705160	5.800887
Catboost	6.028014	5.914732
Random Forest	3.564341	3.612258
Random Forest Tuned	2.412423	2.229232

An in-depth analysis of the model's predictive accuracy, correlated with the aircraft's proximity to the destination airport, revealed a comparatively higher error margin during the descent phase. This increased error is primarily attributed to the complex and highly dynamic maneuvers aircraft undertake in response to prevailing air traffic conditions near the airport, including varying queue lengths for landing. These variables are inherently challenging to predict with high precision. Feature importance analysis of the final model underscored the significance of variables such as the aircraft's distance to both the departure and arrival airports, and its angular position relative to them, highlighting their crucial role in accurate prediction.

In conclusion, this ADS-B-based short-term delay prediction framework provides near real-time forecasts, significantly complementing existing long-term predictive models used for strategic resource allocation and maintenance scheduling. Its primary contribution lies in facilitating the dynamic optimization of critical airport operations, such as gate assignments and ground handling coordination, and enhancing overall air traffic flow management.

6. Conclusion and Future Works

6.1 Use Case Application #1 (INOSENS):

This work in this use case presents a preliminary investigation into the use of LiDAR sensors for intelligent monitoring applications in airport environments, with a focus on Passenger Flow Management, Passenger Density Estimation, and People Counting. A series of point cloud recordings were captured using a LiDAR sensor installed in a terminal area to observe and analyze the movement patterns of individuals. Due to the labor-intensive nature of manual annotation, only a small portion of the collected data—specifically 100 point cloud frames—was labeled. These labeled samples were structured in KITTI format, which allowed for compatibility with existing 3D object detection frameworks.

To evaluate the feasibility and effectiveness of using state-of-the-art deep learning models in this context, transfer learning was performed on a pre-trained VoxelNeXt model. The training process was conducted using the labeled subset, and visual evaluation methods were employed to examine the model’s performance in detecting pedestrians within the 3D space. Despite the limited size of the dataset, the initial results demonstrated the potential of applying such models to real-world airport monitoring scenarios.

In future work, the primary objective will be to increase the size of the labeled dataset by annotating a significantly larger portion of the collected LiDAR data. This is expected to improve the model’s accuracy and robustness, particularly in dense and dynamic environments. Final installation of the LiDAR sensor is planned at Adnan Menderes Airport. At the sensor's finalized location, a continuous data collection process is scheduled over a period of days. This new dataset will then be manually labeled and used for retraining the model, which is expected to mark the final stage of development before deployment.

Additionally, deployment of the trained model in a real-time pipeline will be a key focus. To achieve this, optimized methods for real-time data acquisition from the LiDAR sensor will be developed, allowing for immediate inference on incoming point clouds. The final goal is to create an end-to-end system capable of delivering real-time estimations of passenger flow and density to the airport’s operational platforms. This will be accomplished through Kafka-based integration, where model predictions are streamed in a structured and scalable format suitable for downstream consumption. Through these efforts, the study aims to contribute to the development of intelligent infrastructure solutions that enhance the efficiency and safety of airport operations.

6.2 Use Case Application #2 (Siemens):

Maintaining baggage requests of passengers in an airport may have direct or indirect effect on other operations such as ground services, flight etc. In the use case #2 application, temporal pattern analysis of baggage operations is investigated to determine if there is a correlation for flight delays in a holistic perspective. Several pre-processing and cleaning strategies are applied on the given dataset while also considering the cross selection between the baggage and flight features. The results reveal two major findings. Firstly, creating multiple baggage records per passenger has a negative impact on the related departed flight operation. Secondly, the increase in the pattern dissimilarity ratio for the baggage arrival correlates with the flight delay possibility.

Various machine learning and deep learning models were evaluated for daily baggage prediction using datasets of different sizes. The analysis revealed that Random Forest performed optimally with smaller datasets, while CatBoost achieved superior performance when tested on the largest dataset. Through comprehensive feature engineering and selection processes, the optimal set of features was identified to enhance the prediction accuracy.

Concerning pattern prediction, the machine learning models—particularly ensemble methods such as Tree Regressor and Bagging Regressor—demonstrate superior performance compared to deep learning and traditional time-series models. Among these, tree-based algorithms like Random Forest and Decision Tree consistently yield more accurate and robust results. Their ability to effectively handle nonlinear relationships, mixed data types, and outliers makes them especially well-suited for complex predictive tasks involving structured data. In future, we would like to improve results by the new candidate features to be involved from flight data.

There may be missing features that are affecting baggage arrival but those are not supplied by TAV systems due to limitations and complexity. For example, if there is a planned flight delay and passengers are informed about it then the baggage arrival will shift towards the new flight date or if a flight is used as transfer to another flight by most of the passengers, then the baggage arrival of the first flight will greatly rise. Because of unpredicted scenarios like these, the baggage arrival pattern may stretch, shrink or amplify. Adaptive prediction is the future work that the arrived baggage up to current time will be used as an adaptive feature to predict the non regular pattern changes mentioned above. So that the baggage arrival classification and regression will not be performed only by flight parameters but the baggage itself up to prediction time.

6.4 Use Case Application #4 (INOSENS)

Sentiment analysis of passengers is an effective strategy for identifying passenger satisfaction, the same way it can help identify customer satisfaction in other scenarios. Passengers are often likely to broadcast their candid opinions, and effectively their levels of satisfaction, on online sources. This includes social media, forum websites, and other platforms for reviewing businesses.

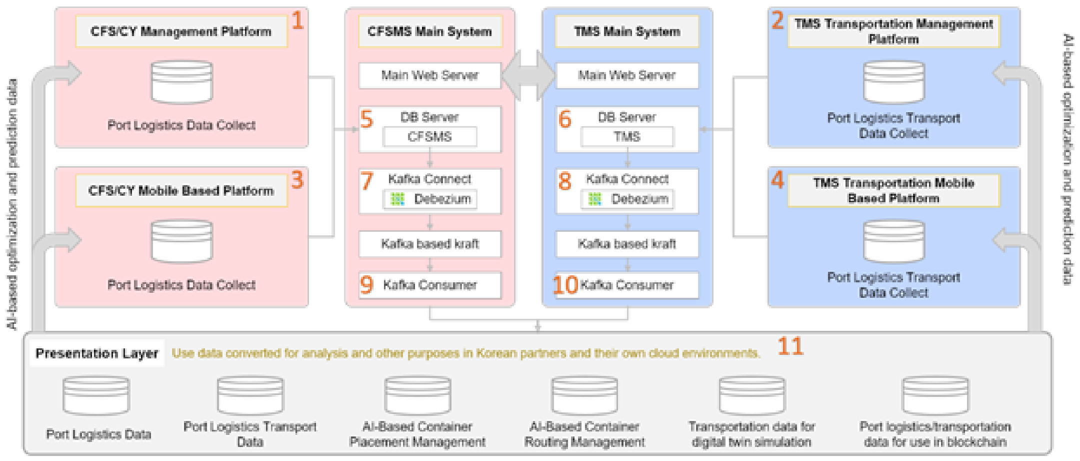
This use case leverages sentiment analysis to identify overall passenger satisfaction within any given time frame. A major emphasis of this proposed application is to enable the user to analyze whether there is a correlation between time and sentiment as well as volume of posts, comments, and reviews. The data analysis component of this study showed that there was a higher volume of posts and a disproportionately high volume of negative posts during the summer months. That corresponds to when people are more likely to travel.

Another major emphasis of this use case is to identify whether there is a change in satisfaction after flight delays. Although this feature has not yet been implemented, sentiment analysis of online text-based content is no doubt a highly effective method for identifying passengers' satisfaction levels related to flight arrival and departure delays.

6.5 Use case Application #5 (Korean Consortium)

1) Implementation of CFS/CY Port Logistics Data Platform (KULS)

- Summary of the development progress and key findings of the use case application are as follows.
    - Data pipeline construction is underway utilizing new technologies such as Kraft, Kafka, and Devezium open source.
    - In the middle version, we configured the Java-based Zookeper mode to Kafka-based KRaft mode to improve performance and reflect the latest technologies. This eliminated Zookeper dependencies, simplified installation, configuration, and maintenance, and improved Kafka's performance and reliability. We also leveraged cache memory to improve UI speed and data processing speed for KafkaStreams and Kafka Connect.
    - Mobile device-based mobile apps are being developed to collect vehicle location control and transportation status data.
- Future works are described.
    - New technologies such as Kraft, Kafka, and Devezium open source build data pipelines.

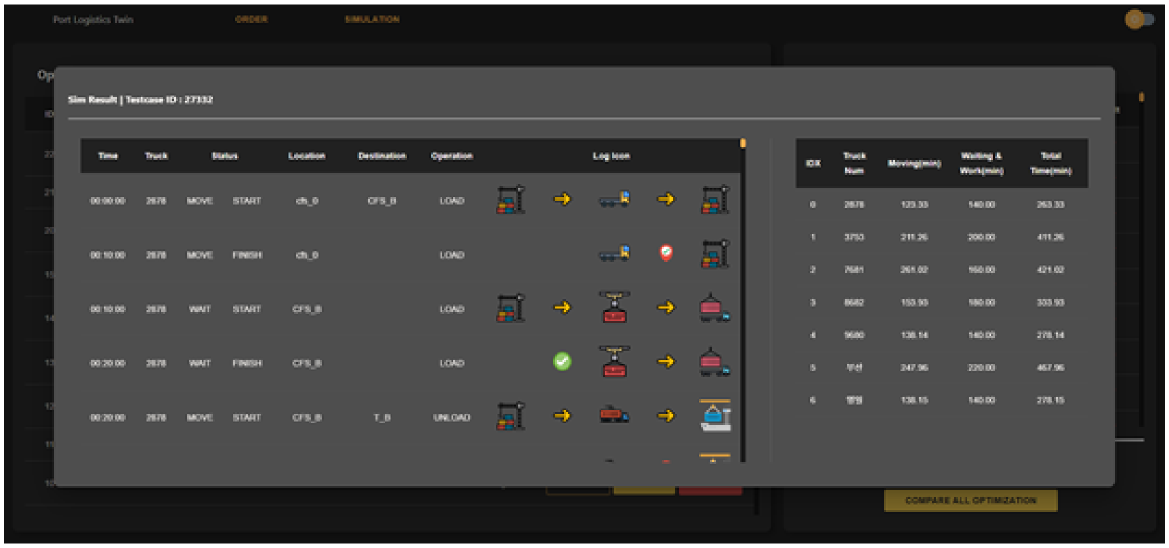

AI-based optimization and prediction data
CFS/CY Management Platform
1
Port Logistics Data Collect
CFS/CY Mobile Based Platform
3
Port Logistics Data Collect
CFSMS Main System
Main Web Server
5
DB Server
CFSMS
7
Kafka Connect
Debezium
Kafka based kraft
9
Kafka Consumer
TMS Main System
Main Web Server
6
DB Server
TMS
8
Kafka Connect
Debezium
Kafka based kraft
10
Kafka Consumer
2
TMS Transportation Management Platform
Port Logistics Transport Data Collect
4
TMS Transportation Mobile Based Platform
Port Logistics Transport Data Collect
AI-based optimization and prediction data
Presentation Layer
Use data converted for analysis and other purposes in Korean partners and their own cloud environments.
11
Port Logistics Data
Port Logistics Transport Data
AI-Based Container Placement Management
AI-Based Container Routing Management
Transportation data for digital twin simulation
Port logistics/transportation data for use in blockchain

2) Implementation of CFS/CY Optimization (KAIST)

- Summary of the development progress and key findings of the use case application.
    - A mid-version of the AI-based CFS/CY management system has been designed and implemented.
    - For intra-CFS optimization, a reinforcement learning model based on the Proximal Policy Optimization (PPO) algorithm was applied to determine efficient container stacking and relocation strategies, which were validated through simulation.
    - Experiments conducted under various yard configurations confirmed a reduction in unnecessary relocations and improved accessibility for high-priority containers.
    - Initial development for inter-CFS optimization has been completed, including the design of evaluation metrics, cost models, and visualization components.
    - Standardized data pipelines have been constructed to collect and process operational information to support both optimization modules.
- Future works are described.
    - Scenario-based simulations will be conducted to evaluate the performance of the inter-CFS optimization module under varying yard congestion and capacity conditions.
    - Quantitative analysis will be performed to validate the effectiveness of dynamic CFS selection strategies based on expected relocation costs and operational efficiency.
    - Further refinement of the decision-support logic and simulation fidelity is planned to enhance real-time applicability.
    - Additional efforts will focus on optimizing the deployment environment to support future integration with the logistics platform.

3) Implementation of Digital Twin Simulation Platform (eins S&C)

- Summary on Twin and Simulation



Sim Result | Testcase ID : 27332

Time	Truck	Status		Location	Destination	Operation	Log Icon
00:00:00	2678	MOVE	START	ch_0	CFS_B	LOAD	
00:10:00	2678	MOVE	FINISH	ch_0		LOAD	
00:10:00	2678	WAIT	START	CFS_B		LOAD	
00:20:00	2678	WAIT	FINISH	CFS_B		LOAD	
00:20:00	2678	MOVE	START	CFS_B	T_B	UNLOAD	

IDX	Truck Num	Moving(min)	Waiting & Work(min)	Total Time(min)
0	2678	123.33	140.00	263.33
1	3750	211.26	200.00	411.26
2	7681	261.02	160.00	421.02
3	8682	153.93	180.00	333.93
4	9680	138.14	140.00	278.14
5	[illegible]	247.96	220.00	467.96
6	[illegible]	138.15	140.00	278.15

- This system is expected not only to reduce congestion in port logistics transportation but also to actively support platform users' decision-making during the dispatch planning stage, significantly reducing work time.

- Future works

  - The Digital Twin-based optimal dispatch system has currently been completed up to the Early Prototype stage. Ongoing research is being conducted to further enhance the accuracy and performance of the optimization module. Additionally, we are continuously collaborating to complete the data pipeline and are conducting research to provide additional business insights to users through the optimal dispatch system service.
  - We plan to improve fidelity by receiving congestion data required for loading/unloading operations at CFS and terminals and enhancing the simulation model.
  - In the event that a specific transportation contract requires urgent handling, we will explore ways to reflect this in the system to derive a new optimal dispatch plan.
  - We are in discussions to enable Kuls to develop the UI for the optimal dispatch system, which will be mounted on the platform in the future. Once this task is completed, we will proceed with the integration and testing of the platform and the system.

6.11 Use case Application #11 (TAV)

The complexity and intricate interrelations of airport operations highlight the importance of each activity happening accurately and on time. All processes, from an aircraft's landing to its preparation for the next flight, directly influence one another on both the airside and landside. This operational integrity is clearly demonstrated when any delay in an incoming aircraft directly impacts numerous subsequent processes, such as bridge operations, chute deployment, boarding, and carousel baggage retrieval. Therefore, accurately predicting all these processes with a high degree of precision is crucial for enhancing passenger satisfaction and optimizing resource allocation.

Extensive data analysis and business discussions revealed that flight information frequently changes for various reasons leading up to the aircraft's departure. Our FDP (Flight Departure Prediction) model, trained on data obtained during the flight planning stage, proved insufficient in achieving the desired accuracy levels due to these uncertainties. To address this deficiency, the ADS-B (Automatic Dependent Surveillance-Broadcast) Milestone project was initiated, aiming to provide more accurate and sensitive predictions closer to the actual flight time. Initial results indicate that the ADS-B model can predict delays with an error margin of just 2 minutes. We aim to measure the true success of this system, which is not yet live, by comparing its predictions with those from the FDP project made a month prior. This comparison will concretely demonstrate the new model's contribution to operational efficiency.

In another significant project, alongside developing models to predict when various flight services (including boarding, bridge, carousel, check-in, chute, counter, electricity, gate, inblock, and offblock) will begin and how long they will last relative to an aircraft's arrival or departure, we've also commenced MLOps (Machine Learning Operations) architectural work. This integration involves the implementation of tools such as MLflow for experiment tracking and model management, MinIO for scalable object storage, and NannyML for data drift detection. This will ensure the sustainability, easy updateability, and scalability of the developed models. Consequently, this will enable the models to adapt quickly to evolving operational needs and generate more reliable predictions. Ultimately, these projects will significantly contribute to more effective management of airport operations, minimizing delays, and improving the overall passenger experience.