BILKENT UNIVERSITY

CS492 - DETAILED DESIGN REPORT

Berkin Kağan Ateş	22102519
Burak Baştuğ	22102266
Umut Başar Demir	22102376
Berin Su İyici	22102342
Arda Öztürk	22103616

Table of Contents

1. Introduction

The logistics industry is a dynamic, continually expanding sector that encompasses warehouses, trucks, loads, routes, time constraints, and cost considerations. Due to the inherent complexity of this business, technology may be needed to help manage all of these components. Based on this concept, TırGöz offers a user-friendly system for companies in the logistics sector, leveraging computer vision technology and route-optimization algorithms to enhance the efficiency of logistical operations. TırGöz proposes a system that provides a holistic view of logistics operations by merging perception and planning layers. TırGöz provides operational efficiency by reducing manual inspection time and optimizing workforce usage. TırGöz offers cost reduction by minimizing late deliveries and unnecessary travel through adaptive route planning. TırGöz ensures work quality by preventing shipment errors through real-time alerts. TırGöz also considers the scalability by ensuring low latency and dynamic route optimization even with a large number of trucks and warehouses.

1.1 Purpose of the System

Since the logistics industry is a complex sector, TırGöz has more than one benefit for more than one aspect. By using its Computer Vision module TırGöz can detect anomalies and count items in the warehouse and during the shipment processes. This increases the work quality of the shipment process and prevents loading/unloading errors. TırGöz also decreases the manual inspection time which can reduce the required workforce.

Another important feature of the TırGöz is the live video stream of the warehouse and loading process, additionally, desired parts of the live stream can be recorded for later uses. During this live stream also the Computer Vision module can actively catch anomalies and alert the anomaly along with the short clip recorded by the stream. Which helps warehouse operators to optimize their workforce and reduce the response time to anomalies occur during the logistics processes.

One of the important steps in the logistics is the physical delivery of the products. TırGöz also aims to ease the delivery process by displaying the optimized route for the drivers based on the loaded products on the trucks and other specified constraints. This also can help with reducing

the deliveries cost by eliminating unnecessary travels, and optimize the branch sales by filling the needs of the branches as fast as possible.

TırGöz also keeps track of the product count and location information in the warehouse with the help of the Computer Vision module, which means TırGöz is not only working for anomaly detection but also normal process tracking. This is how warehouse operators can see the truck and warehouse occupancy percentage. Also, TırGöz client layer allows branch managers to manage the live stream, recording and metric operations with an user-friendly UI.

The last function of the system is product request (order creation) for branch managers. Branch managers can request products from warehouses using the TırGöz application and warehouse operator can see this request. This is how TırGöz also eases the communication aspect of the logistics operations. As a result, TırGöz offers different benefits for the different steps of the logistics operations starting from monitoring of daily processes in the warehouse to efficient delivery to the destination branch by using a well planned and organized system composed of advanced technology and consistently designed architecture.

1.2 Design Goals

In order to create the TırGöz system as an advanced experience for logistics operations some of the design goals are followed through the development process.

1.2.1 Usability

TırGöz focuses on high-efficiency usage for logistics operations. The system can be used with minimal effort due to its simple and user-friendly interface. The core functionalities are intuitive and easy to learn. For instance, clear and immediate vehicle location, optimized path, and real-time alert displays do not require the user to search or interpret complex data. Additionally, to enhance usability, anomaly detection alerts are accompanied by short event clips for easy visual verification, thereby minimizing the time required for manual inspection.

1.2.2 Reliability

TırGöz ensures high operational reliability. Since the Computer Vision and Route Optimization Services are isolated, a failure in one independent module does not affect the other independent modules. Additionally, TırGöz prioritizes rapid recovery from any failure to maintain the integrity

of real-time events and data, particularly during the critical hours of the day. Additionally, the anomaly detection function aims to achieve a high accuracy rate through deterministic validation checks.

1.2.3 Performance

The primary goal of TırGöz is to ensure low-latency processing due to its real-time functional nature. To maintain real-time alerts, TırGöz processes live video streams and detects anomalies with an end-to-end latency of no more than **1000** milliseconds. Also, the Route Optimization Module aims to recalculate the optimal route based on warehouse and truck occupancies within a desired time constraint for the standard operational load.

1.2.4 Supportability

TırGöz is designed to be straightforward, making it easier to understand bugs and problems, thereby reducing the time required to fix errors and install new features. This is why logs are used to record detailed notes in a consistent and easy-to-read format. Additionally, TırGöz aims to minimize complete system restarts, thereby keeping the system operational during minor system changes. To further increase supportability, the system's technical documentation is maintained in a complete and accurate manner.

1.2.5 Scalability

TırGöz must be able to handle an increased number of cameras, trucks, and deliveries. For instance, the Computer Vision Service and Event Processing API should not exceed the expected real-time constraints, even as the number of cameras and streams increases. The Route Optimization Service must perform calculations for both smaller and larger numbers of trucks and deliveries. Additionally, TırGöz must efficiently manage the rapidly growing volume of stored data, including short video clips of detected anomalies, events, and metrics.

1.3 Definitions, acronyms, and abbreviations

This part of the report gives definitions of the commonly used concepts, terms and abbreviations.

Term	**Definition**

RTSP	A networking protocol used for streaming video from security cameras.
CCTV	Also called Closed Circuit Television. A type of surveillance camera.
YOLO	A deep learning model for object detection
Ad hoc	A temporary solution that does not meet the standards of the system
Anomaly Detection	Detecting unusual events and abnormal patterns.
Occupancy Estimation	Process of approximating how full a space is, in our case, a warehouse and a truck load.
Heuristic Algorithms	Fast and rule-based solutions that determine good solutions. Used in our case because route optimisation is computationally expensive.
Meta-Heuristic Algorithms	Problem-agnostic frameworks that guide and improve heuristic rules.
Vehicle Routing Problem	An NP-hard optimisation in the field of logistics. The goal is to find optimal route sets for a fleet.
Dynamic Routing	Real-time adjustments of delivery routes based on new information such as load delays, traffic.

OCR	Abbreviation for Optical Character Recognition, a technology for converting some sort of image into editable and/or machine readable data.

1.4 Overview

This report defines the detailed design specification for TırGöz, an advanced logistics monitoring and optimization platform that integrates computer vision and vehicle routing optimization to enhance logistics efficiency. It provides the technical infrastructure for TırGöz by defining key services, the system architecture, and the technical analysis of non-functional requirements in order to describe the development roadmap. This document is intended to demonstrate that the development of the TırGöz is well-structured, organized, planned, and makes use of best practices.

This report initially describes the overall system architecture, elaborating with the subsystems and services. It explains the roles of the subsystems and services such as client layer, API getaway layer, server layer, and data acquisition layer.. These layers are introduced in a technical way, also, by considering the interactions between the subsystems and services. Additionally, hardware/software mapping is introduced to provide details about the system architecture and the data layer (persistence subsystem) is defined to show how the data is stored and maintained.

Besides the technical structure, this report also introduces test cases, design goals, and engineering constraints. Finally, this report describes the team collaboration aspect of the project and development workflow. In the end, this report indicates that TırGöz is developed as a well designed and planned solution for logistics management processes.

2. Current software architecture

2.1 TırGöz Current Architecture

TırGöz follows a microservices based system design. This design system enables modularity, scalability and independence for ease of deployment of different system modules. The system includes a frontend dashboard interface, and API gateway, backend microservices, data layer, streaming and analytics components. This bundle's goal is to support logistics monitoring and optimisation in real time. The architecture achieves this goal by combining CV supported warehouse monitoring with logistics decision support. This combination enables detection of operational events and their integration into route planning and warehouse analytics in near real time.

The frontend dashboard is a web interface that provides an operational dashboard to logistics operators. This provides the operator with insight into warehouse activity, access to detected anomalies and ability to manage logistics workflows. Communication of frontend and backend services are done through REST APIs with API gateway. This means of communication ensures security and structure for data exchange between subsystems.

The backend services are responsible for core functionalities such as video stream processing, CV inference, anomaly detection, event generation, warehouse state analysis and route optimisation. These services process camera streams incoming from warehouse environments and transform the images into structured operational events such as movement of cargo, occupancy changes and potential anomalies.

The data layer consists of multiple storage systems optimised for different types of data. PostgreSQL is used for structured operational data such as warehouse metadata and logistics events observed with their associated data. Redis supports caching and event messaging, this approach enables low latency communication between components. Object storage systems are used for storing the video clips and other media artifacts generated during anomaly detection.

The architecture also integrates stream processing pipelines for near real time video analysis. In this analysis video are processed by CV models to detect objects, track movements and identify anomalies in warehouse operations. These events are then published to backend services, updating the system state.

This modular architecture allows our system to integrate visual perception, logistics analytics and operational decision support within a unified platform, without the need for cloud storage support. The modular approach also keeps the system flexible for future extensions such as additional vision models, integration with enterprise logistics systems and optimisation algorithms.

2.2 Competitors, Alternatives and Current Solutions

2.2.1 HikVision Intelligent Warehouse Solutions

HıkVision provides solutions that function on AI integrated video monitoring. They provide corporations with warehouse activity monitoring, parcel tracking and operational event tracking [1]. Their systems utilise specialised cameras with edge inference to analyse warehouse activity and provide dashboards to access generated information [2].

Hikvision's focus is on surveillance and visual monitoring. This is very similar to a part of TırGöz. However, TırGöz integrates visual monitoring with decision making and route planning while HikVision stays restrained to processing and informing about warehouse operations. TırGöz aims to achieve this by generating structured events from CV outputs.

HikVision is renowned in video monitoring infrastructure, and their or similar systems are in place for security reasons in many establishments in warehouses [3]. Organisations with existing monitoring systems may be hesitant to adopt additional analytics platforms. Though this may seem as a challenge at first, TırGöz is designed to operate with existing camera infrastructure, this allows organisations to reuse their current systems or upgrade their systems with our analytics tools.

2.2.2 Descartes Systems Group

This competitor provides many solutions including a transportation management platform equipped with cloud-based logistics software designed to optimise transportation operations and supply chain visibility across logistics networks with multiple nodes [4]. The platform supports optimisation of routing, tracking of shipment and coordination of different nodes for operations.

Descartes relies on telematics, manual barcode scans and shipment records but not visual monitoring [5]. TırGöz integrates CV based warehouse monitoring for in house monitoring with logistics planning. TırGöz's approach increases visibility and accountability of operations by providing physical status insights on warehouse operations and cargo handling activities, rather than waiting for manual updates to the system.

2.2.3 Zebra Technologies Warehouse Visibility Platforms

Zebra Technologies develops warehouse technology solutions to improve asset visibility, inventory tracking and operational efficiency. Their solutions integrate barcode scanning, RFID technologies and vision tools to help organisations to improve asset tracking, workflow automation and operational accuracy [6].

Zebra's machine vision software suite enables organisations to deploy smart cameras and scanners[6]. These devices are capable of verifying barcodes, OCR and tracking asset presence. This suite allows operators to automate inspection and enables traceability of such actions and assets.

TırGöz assumes little technical infrastructure, and presents a complement to the likes of Zebra Technologies' suite by providing real time visual monitoring and anomaly detection capabilities that extend beyond traditional scanning based systems.

3. Proposed software architecture

3.1 Overview

3.1.1 Client Layer

3.1.1.1 Role Views

The system has four client-side roles. They are Admin, Manager, Operator, and Driver views. While Admin can access all system functionalities, other roles will be limited to functionalities like stream display and anomaly detection.

3.1.1.2 Shared Views

In addition to role-based views, some pages will be common across all roles. The authentication view is a fully shared view. Map view is also a shared view, but it may present different functionalities and a user interface to different roles.

3.1.2 API Gateway Layer (Gateway Subsystem)

3.1.2.1 Request Router

The system has multiple microservices, each running on a different port on the host machine. The request router is responsible for redirecting client-layer requests to the correct services in other layers.

3.1.2.2 Auth Guard

The auth guard is responsible for validating JWT tokens on every incoming request and enforcing the permissions of the system's roles accordingly.

3.1.2.3 WebRTC Signal Handler

A peer-to-peer structure is used for RTSP streaming on the web application side. The signal handler manages WebRTC signalling between the client layer and MediaMTX streaming server for real-time viewing.

3.1.3 Server Layer

3.1.3.1 Streaming Service

The streaming service contains a recording service. It had contained a media stream server as well; however, the MediaMTX module has been moved to the data acquisition

layer. In that way, it can get the stream without relaying it to the server layer, thereby increasing efficiency. The Recording Service will be responsible for tracking stream chunks in the event of anomalies and for general recording functionality. It will store the recording data as video chunks.

3.1.3.2 Computer Vision Services

This subsystem is the main one that contains the core computer vision logic, including anomaly detection and item counting. It triggers the notification service after finding an anomaly and finishing its other logical processes.

3.1.3.3 Route Planning Services

The route planning layer retrieves the relevant raw data from computer vision services and the data acquisition layer to generate optimal delivery and pickup routes. It combines the raw data with parameters from the map service and outputs the final result to the route optimization service.

3.1.3.4 Core Backend Services

The core backend contains three main services: the auth service, the analysis service, and the event bus. The auth service manages responses from endpoints across different services based on user roles and credentials. The analysis service applies statistical formulations to raw data and generates meaningful statistical observations. The event bus has been added recently. It logs all system events to PostgreSQL for historical analysis and reporting.

3.1.4 Data Acquisition Layer

3.1.4.1 Camera Subsystem

The main point is to retrieve the streaming data and configure the system for the incoming real-time stream, including its connection details (IP address, port, and credentials). Additionally, it tracks the FPS and ping latency for each stream and allows the user to observe bandwidth usage.

3.1.4.2 Sensor Subsystem

It has two services: a sensor polling service and an alert publisher. The sensor polling sends requests to sensors every 2 seconds to read current values and check the critical boundary values. The alert publisher communicates to the event bus if the boundary value is exceeded and notifies the client layer.

3.1.5 Data Layer (Persistence Subsystem)

3.1.2.1 Redis Cache

It provides the system with high-speed data retrieval for frequently accessed data across different parts of the system, such as authentication credentials, sensor health monitoring, and anomaly notifications.

3.1.2.2 PostgreSQL

It is the system's main storage structure. It stores all structured data, including credentials, roles, events, goods, and warehouse attributes.

3.1.2.3 Minio Object Storage

It stores the recorded video chunks generated by the recording service in case of anomalies or user-intended recording requests.

3.2 Subsystem decomposition

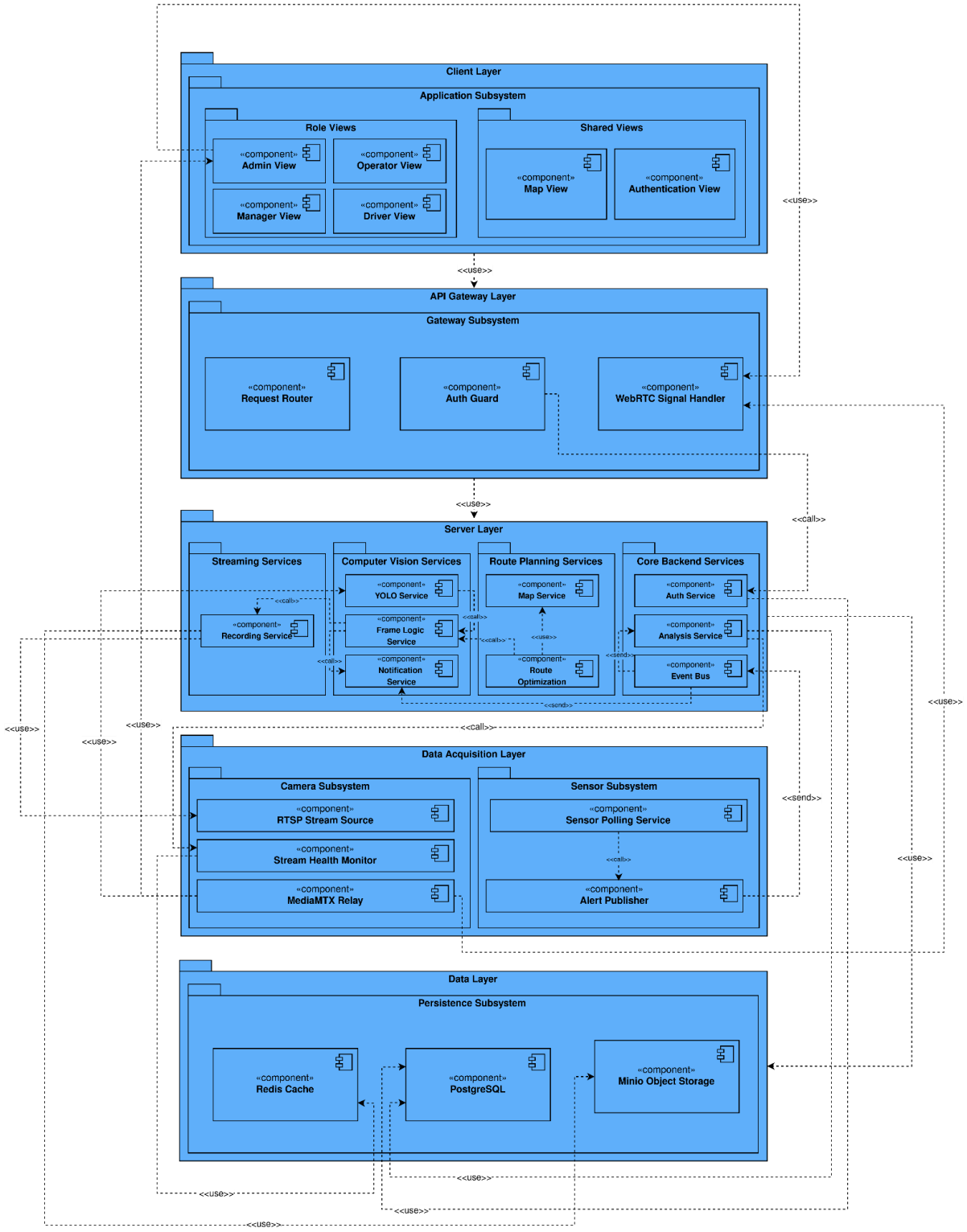

Figure 1: Subsystem Decomposition

3.3 Hardware/Software Mapping

The whole subsystem is deployed on a local host server in warehouses, without any cloud service connection. In the project, there are three hardware components: a local host server, RTSP Cameras, and environmental sensors.

3.3.1 Local Host Server

All backend services, the API gateway, and MediaMTX run on a single local host server.

Hardware	**Software Components**
Local Host Server	API Gateway Layer (Request Router, Auth Guard, WebRTC Signal Handler)
Local Host Server	Server Layer (Recording Service, YOLO Service, Frame Logic Service, Notification Service, Map Service, Route Optimization Auth Service, Analysis Service, Event Bus)
Local Host Server	Data Acquisition Layer (RTSP Stream Source, MediaMTX Relay, Sensor Polling Service, Alert Publisher, Stream Health Monitor)
Local Host Server	Data Layer (Redis Cache, PostgreSQL, Minio Object Storage)

3.3.2 IoT Devices

RTSP cameras are on the same subnet as the host server and are connected to it via a router or switch. Environmental sensors can be connected either directly to the host server using the deployment configuration.

Hardware	**Software Components**

RTSP Camera	Camera Subsystem (RTSP Stream Source, Stream Health Monitor, MediaMTX Relay)
Environmental Sensors	Sensor Subsystem (Sensor Polling Service, Alert Publisher)

3.3.2 Deployment Diagram

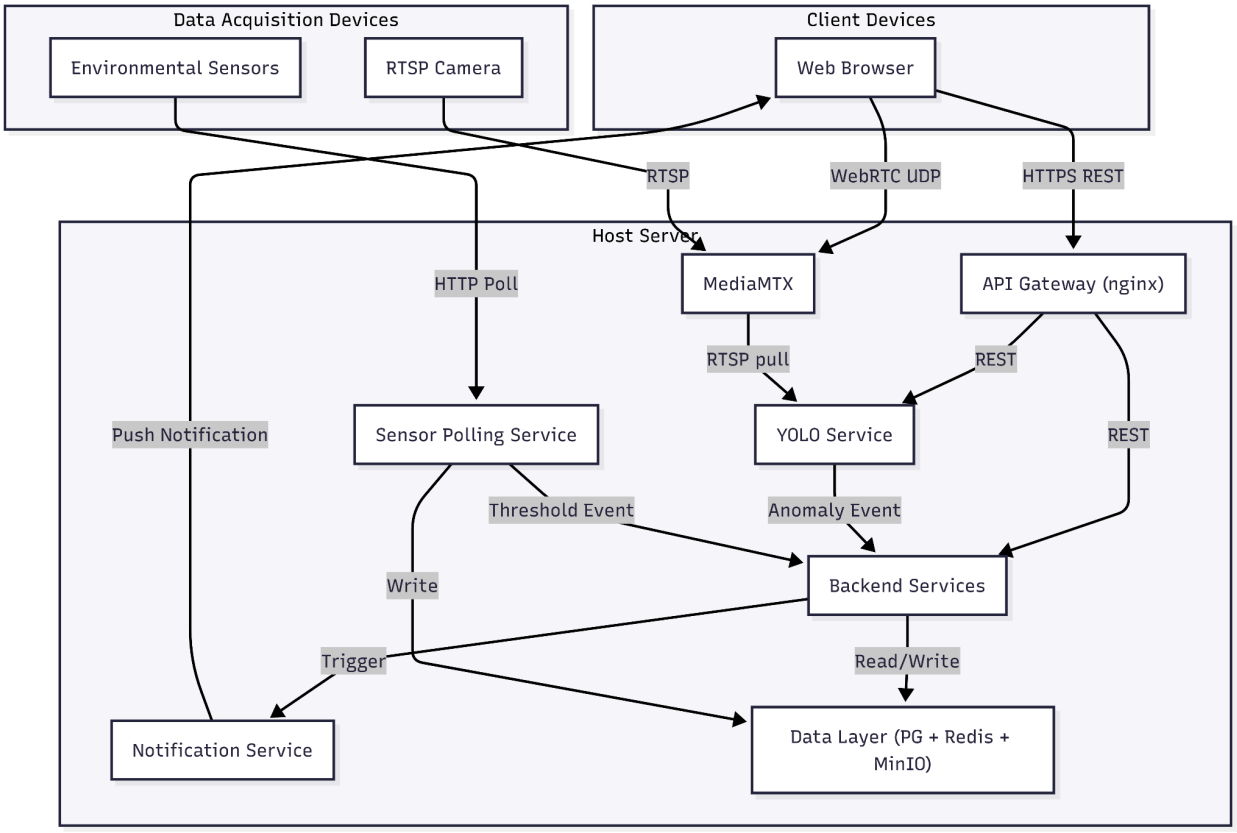

Figure 2: Deployment Diagram

3.4 Persistent Data Management

3.4.1 Relational Data

PostgreSQL is used for the main structured data storage. It stores account credentials, roles, camera-sensor configurations, detected anomaly entries, inventory logs, products, routes, analytic metrics, and warehouse attributes. It provides ACID compliance and a reliable storage

unit. Anomaly event entries can grow rapidly during usage. Therefore, indices are applied to timestamps and camera IDs to provide efficient queries.

3.4.2 Cache Management

Redis is used for cache management. It provides high-speed retrieval for frequently accessed data such as authentication session tokens, aggregated statistical calculations, sensor status pings, and route data. Besides its caching benefits, Redis serves as the messaging backbone for the Event Bus through its built-in Pub/Sub mechanism. This dual use of Redis reduces PostgreSQL load and improves overall system performance.

3.4.3 File Storage

The system provides a client layer for recording real-time streams and saving clips of anomalous scenes. Since storing records can be costly for the server's local storage units and PostgreSQL, Minio provides a scalable solution for this type of data.

3.5 Access Control and Security

3.4.1 Authentication

The system uses four steps for the authentication process. These are JSON Web Token, the auth guard in the gateway, and the auth service in the server layer. The client layer submits its credentials to the auth guard in the gateway, which redirects to the auth service, which authenticates the client and generates a token. This is returned to the browser and stored in its local storage. In addition to credential submissions, the auth guard also checks every incoming request to forward it to the server later. Unauthenticated requests are rejected at the gateway level without reaching the server layer.

3.4.2 Role-Based Structure

The system has a role-based structure, with four roles for the client: Admin, Operator, Manager, and Driver.

- **Admin:** It has full system access. It can view the live stream, perform statistical analysis, and administer accounts.

- **Operator:** It has the same functionalities as the admin role except for user administration.
- **Manager:** It is the role assigned to each branch that warehouses and transports goods. They can insert products into the system or give orders for their own branches. They have very limited access to many features compared to the operator and admin roles.
- **Driver:** The driver role has access only to their assigned routes and real-time navigation.

3.4.3 API Security

All communications between the client and the server layers use HTTPS. It provides the system payloads coming from the client to the server, encrypted. Every endpoint call triggers the auth guard in the gateway, which redirects the request to the corresponding service. Requests with missing, expired, or invalid tokens are rejected at the gateway level and never reach the backend services. This security check also verifies the request's role level and prevents lower-level roles from accessing endpoints that require higher-level roles.

In addition, the API Gateway performs security checks for WebRTC signalling, ensuring that only peers with the permitted user role can connect to the MediaMTX source peer.

4. Subsystem services

4.1 Client Layer Services

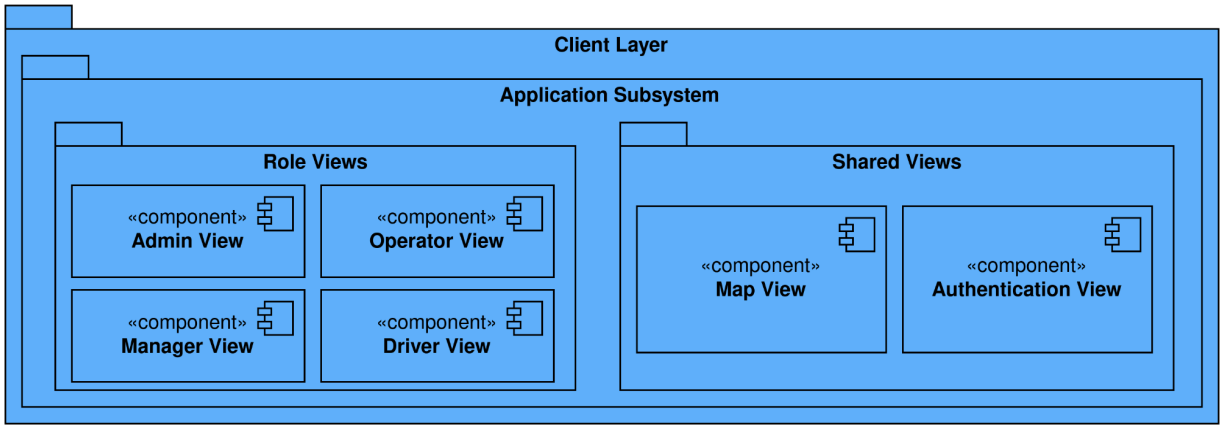

Figure 3: Client Layer

4.1.1 Application Subsystem

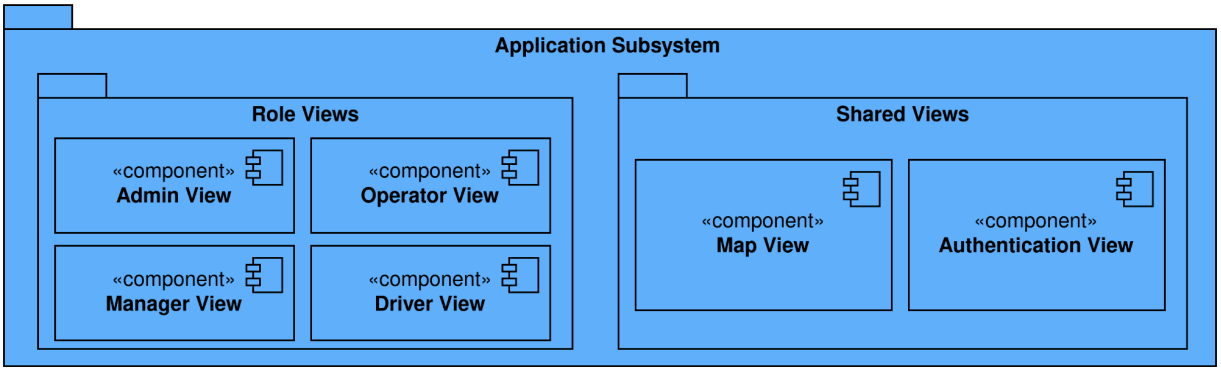

Figure 4: Application Subsystem

4.1.1.1 Role Views

In the TırGöz system there are four roles with different functionalities. This difference in use requires four different Views based on the role.

Admin View: Admin can reach every page, such as warehouse, orders, products, devices and settings pages, and additionally can see user handling areas.

Operator View: Operator can see the pages related to warehouse, orders, products, and devices. This view is the second largest view after the admin, since the operator needs to manage and monitor every step that our system reaches. .

Manager View: Manager has two main functionalities, so the manager can see requesting a product (creating an order) area, and the map view.

Driver View: The driver has the most limited view only consisting of the route on a map view. This is because the only function of the driver is to complete physical delivery of the products.

4.1.1.2 Shared Views

Although there are different roles, some functionalities are in common use for all user types.

Map View: Every user type can see the order map view to monitor it.

Authentication View: Every user type uses the same authentication views such as common login page to authenticate. Also success, fail and unauthorized access messages and views are common for every user. Also every user can update their account information using the same view from their main page after logging in the system.

4.2 API Gateway Layer Services

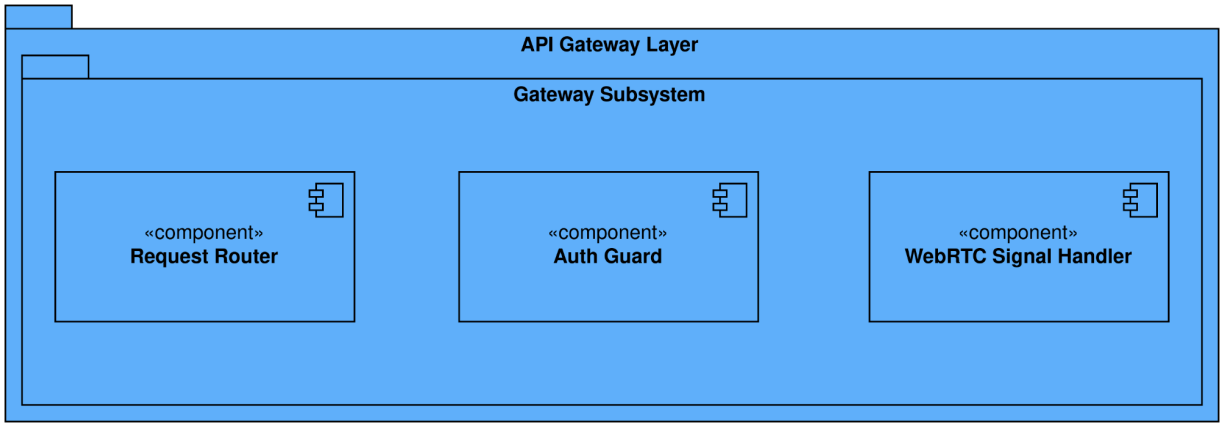

Figure 5: API Gateway Layer

4.2.1 Gateway Subsystem

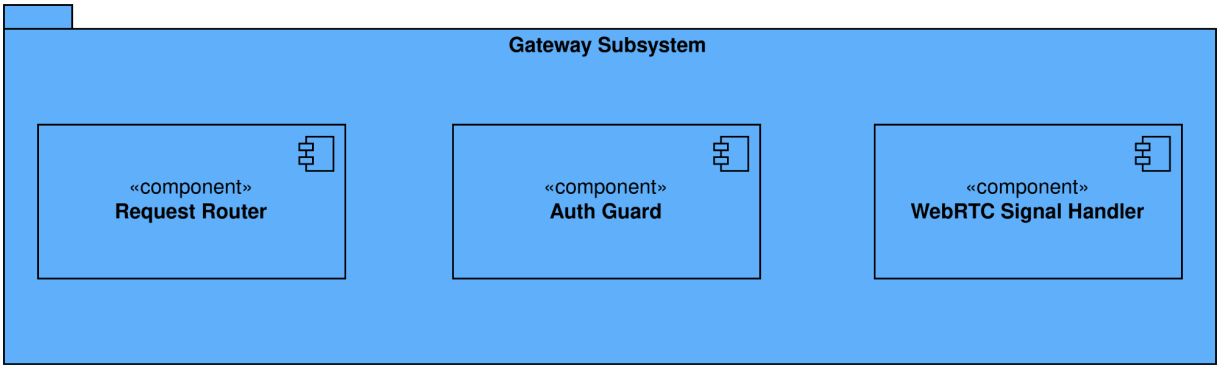

Figure 6: Gateway Subsystem

4.2.1.1 Request Router

Tırgöz project is a multiservice architecture project. It has multiple subsystems and components as microservices, and all of these services are in active communication. To handle the many requests going to different services, an Nginx-based gateway subsystem has been implemented. One of this gateway's functionalities is routing incoming requests.

It receives requests from the client layer, inspects them, and redirects them to the corresponding services. It gets requests on port 80. Requests to '/api/' are redirected to the backbone API service. '/simulation' forwards their requests to the simulation microservice. Besides these gates, '/minio/' are forwarded to the MinIO object storage. It has a 100 MB request size limit for HTTP messaging.

4.2.1.2 Auth Guard

Auth Guard is the system's first security check. It aims to handle security check loads without using auth elements in the server layer. It checks the credentials and role permissions before forwarding the requests.

For credentials, it checks the token validity. If the token is invalid, it returns a 401 Unauthorized response. Additionally, it returns a 403 Forbidden response if there is an inconsistency in role permissions.

4.2.1.3 WebRTC Signal Handler

The system provides real-time streaming functionality via the WebRTC protocol, which uses a peer-to-peer connection and UDP. In a peer-to-peer structure, two peers should find each other and engage in a signalling process to establish a constant stream pipeline between them. The signal handler handles this signalling process by acting as the intermediary between the client browser and the MediaMTX source peer. Both peers share an SDP (Session Description Protocol) that contains each peer's media capabilities and connection parameters. In that way, WebRTC determines the best candidate network path to relay the stream. Since both the host machine and cameras are in the same subnet, no TURN or STUN server is needed for the current implementation.

In addition, it has a peer-to-peer security architecture. It enforces a JWT token validation before beginning the signalling of process. Once all validations are passed, the video stream flows between MediaMTX and the browser, directly bypassing the server layer to reduce the latency and server load.

4.3 Server Layer Services

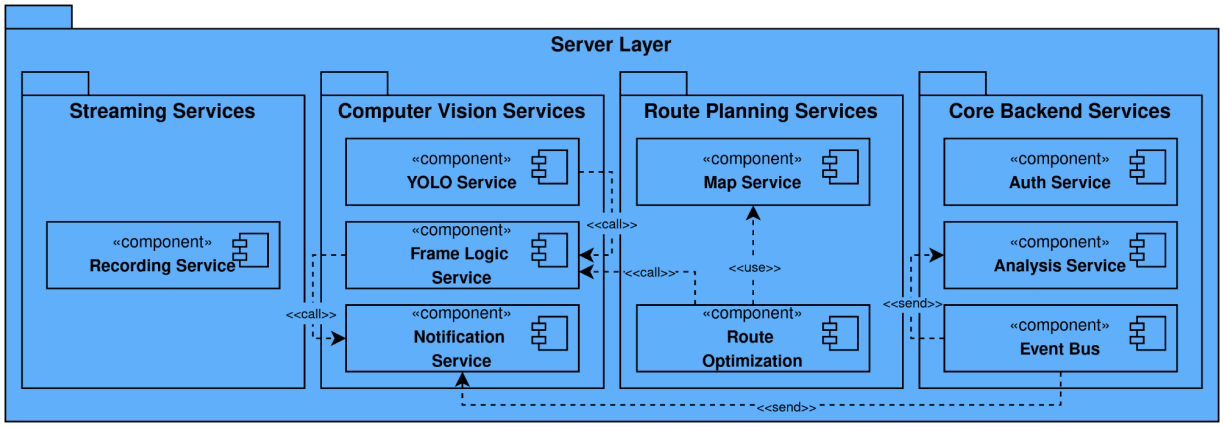

Figure 7: Server Layer

4.3.1 Streaming Services

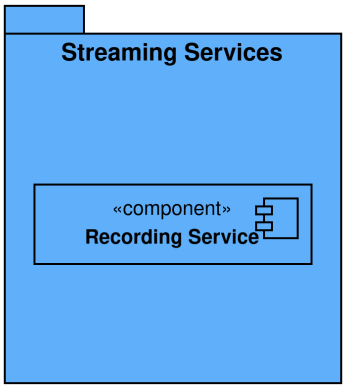

Figure 8: Streaming Services

4.3.1.1 Recording Service

Recording real-time streams has been implemented as a separate service because it uses an OS pipeline to invoke the FFMPEG recording feature. Since the building FFMPEG pipeline can be costly for the server, it is implemented as an atomic structure. It is called in user-intended requests and the Frame Logic Service in the computer vision services. It basically records stream chunks in response to triggers from other services in the system and stores them in the object storage unit.

4.3.2 Computer Vision Services

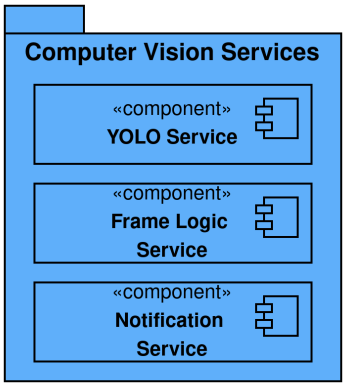

Figure 9: Computer Vision Services

4.3.2.1 YOLO Service

The YOLO Service serves as the primary object detection engine within the Computer Vision Services subsystem. It receives real-time video frames that are relayed directly by the MediaMTX module from the Data Acquisition Layer. By using YOLO deep learning models, this service performs low-latency inference to identify warehouse items, estimate capacity, and detect physical anomalies. Upon processing the frames and identifying any irregular events or anomalies, it generates structured metadata and triggers the Notification Service to alert the relevant users

4.3.2.2 Frame Logic Service

The Frame Logic Service acts as the main decision-making part of the computer vision system. It takes the basic information found by the YOLO Service and uses it to do things like count items or decide if a real problem has happened. When it spots an important event or a broken item, it tells the Recording Service to save a video clip of that exact moment. After finishing these checks, it passes the information along to trigger the Notification Service to alert the users.

4.3.2.3 Notification Service

The Notification Service acts as the messenger for the computer vision system. After the other services finish checking the video and spot a problem, they trigger this service to raise the alarm. It then takes these important alerts and passes them along to the core backend system. This makes sure that warehouse operators and managers get the warnings on their screens right away when something goes wrong.

4.3.3 Route Planning Services

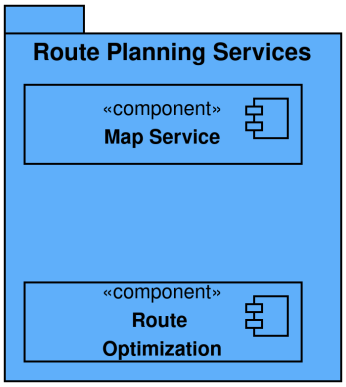

Figure 10: Route Planning Services

4.3.3.1 Map Service

The Map Service acts as the geographic map for the route planning system. It provides the essential map parameters, such as location data and distances, needed to figure out how to get from one place to another. By supplying this information, it helps the Route Optimization service accurately calculate the best and fastest delivery paths for the trucks.

4.3.3.2 Route Optimization

The Route Optimization service acts as the main brain for delivery planning. It takes geographic details from the Map Service and combines them with live data from the warehouse, such as changing truck states and new orders to generate the best delivery and pickup routes. This service relies on rule-based math to update and calculate optimal paths for the drivers.

4.3.4 Core Backend Services

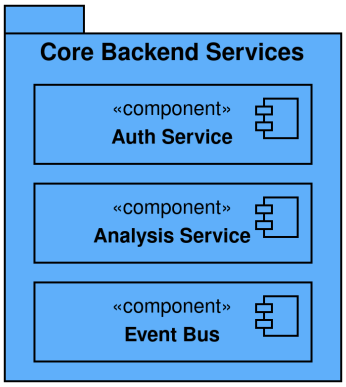

Figure 11: Core Backend Services

4.3.4.1 Auth Service

The Auth Service acts as the main security checker for the backend system. When a user logs in, it takes their credentials from the gateway, verifies their identity, and creates a secure session token. It also manages what each person is allowed to do, making sure that responses from different parts of the system match the permissions of the user's role.

4.3.4.2 Analysis Service

The Analysis Service acts as the main data parser for the platform. It takes the raw data collected from the system and applies statistical formulations to it. By doing this, it generates

meaningful statistical observations, making it easy for administrators and operators to quickly understand the current state and efficiency of their logistics operations on their dashboards.

4.3.4.3 Event Bus

The Event Bus acts as the main message center for the system. It uses Redis and its Pub/Sub feature as the backbone for handling messages. When other parts of the system, like the alert publisher, notice that a specific limit has been crossed, they send a message directly to the Event Bus. Recently added to the core backend, its main job is to safely log all of these system events into the PostgreSQL database so they can be saved for historical analysis and reporting late.

4.4 Data Acquisition Layer Services

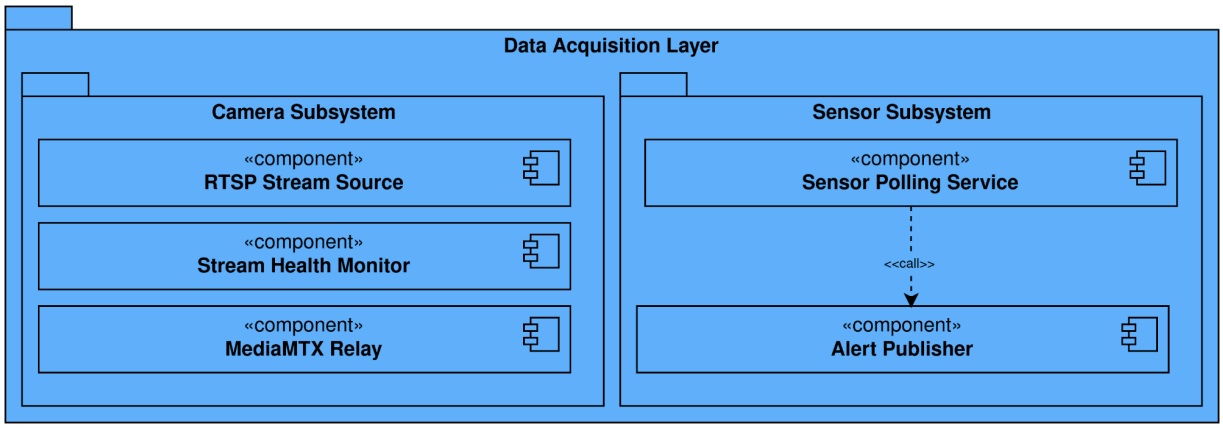

Figure 12: Data Acquisition Layer

4.4.1 Camera Subsystem

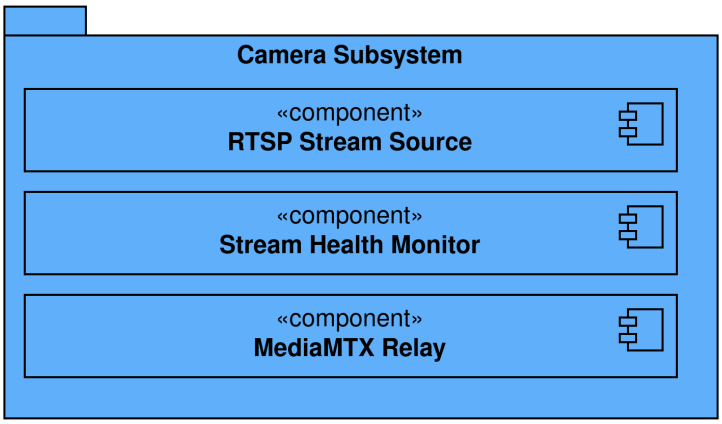

Figure 13: Camera Subsystem

4.4.1.1 RTSP Stream Source

It behaves like a connection point before the MediaMTX Relay. It contains the configurations that identify the physical cameras, which are IP addresses, port numbers, and RTSP URLs. These configurations can be fully editable, and new ones can be added or old ones removed. It provides a flexible deployment environment for different physical implementations.

4.4.1.2 Stream Health Monitor

The health monitor is a diagnostic service that continuously tracks all cameras by retrieving the health and connection status of each registered camera. It informs the system of the cameras' key metrics, including frame rate and ping latency.

In addition to the monitoring feature, it also detects connection losses for each camera and triggers its event listener to communicate with the Event Bus, which then transfers the warning to the client layer and displays the corresponding message to the final user.

4.4.1.3 MediaMTX Relay

Video and audio streams can be published, read, proxyed, recorded, and played back using MediaMTX, a ready-to-use, zero-dependency real-time media server and media proxy. It is considered a "media router" that directs media streams from one end to the other [7]. The main reason MediaMTX is used is that web browsers cannot display the RTPS stream sources

directly using HTML DOM elements. Therefore, the RTSP stream should be converted to a protocol that can be viewed in the browser with low latency.

After the conversion, it relays the stream to the client browser using the signalling process handled by the WebRTC Signal Handler in the Gateway Subsystem. At the same time, it relays the output stream to the YOLO service to process the frames. MediaMTX handles these dual or multiple streams efficiently. Because it relays the same RTSP stream to multiple consumers without duplicating the stream flow between cameras and itself[7]. It is integrated with the gateway subsystem using the ‘stream/’ proxy endpoint.

4.4.2 Sensor Subsystem

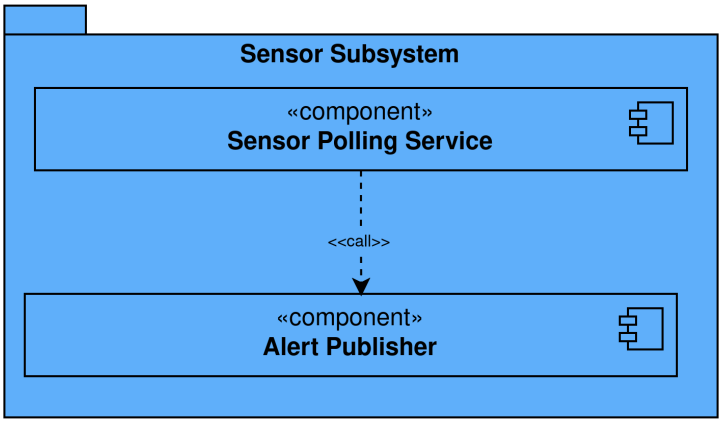

Figure 14: Sensor Subsystem

4.4.2.1 Sensor Polling Service

The Sensor Polling Service acts as the constant checker for the physical environment of the warehouse. It actively sends requests to the connected environmental sensors to read their current values. By constantly collecting this data, it checks to see if any critical boundary values have been crossed, and then works with the Alert Publisher to raise the alarm if a problem is found.

4.4.2.2 Alert Publisher

The Alert Publisher acts as the alarm bell for the sensor system. If the sensor polling service finds that a critical boundary value has been exceeded, this service immediately takes action. It communicates the problem directly to the Event Bus and notifies the client layer so that users can see the warning on their screens right away.

4.5 Data Layer Services

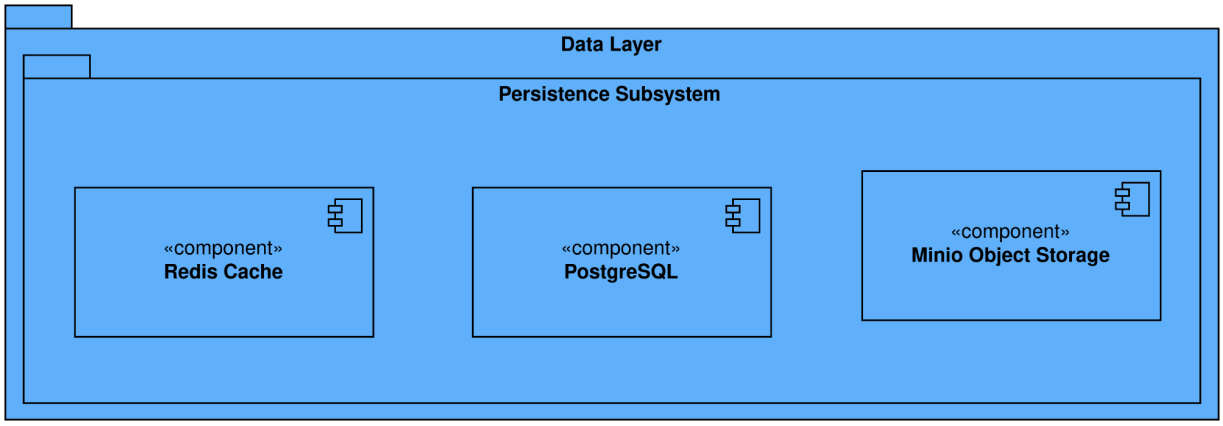

Figure 15: Data Layer

4.5.1 Persistence Subsystem

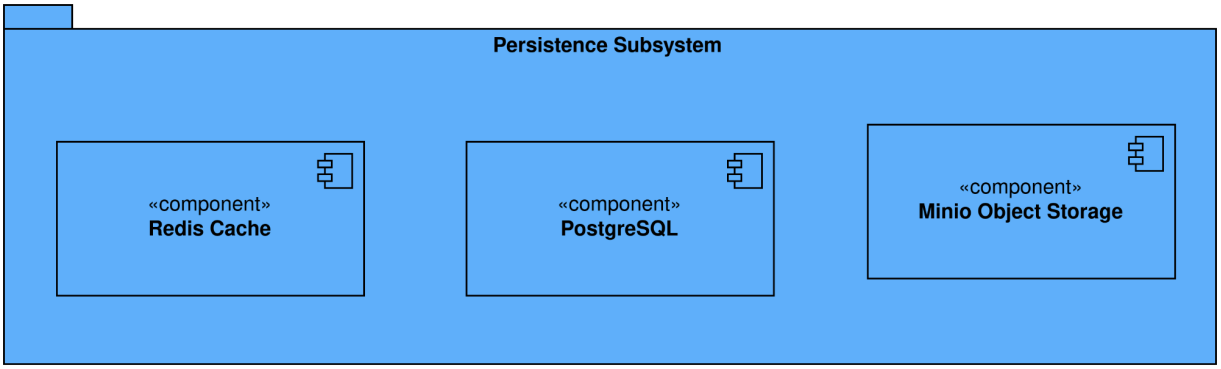

Figure 16: Persistence Subsystem

4.5.1.1 Redis Cache

The Redis Cache acts as the high speed memory for the system. It stores and retrieves data that is used all the time, like login details, anomaly notifications, and sensor status updates. Because it is so fast, it also works as the main messaging system for the Event Bus. By handling these quick tasks, it takes the heavy lifting off the main database and helps the whole platform run much smoother.

4.5.1.2 PostgreSQL

PostgreSQL acts as the main database for the entire system. It stores all the organized information, such as user accounts, camera settings, warehouse items, and logs of any problems found. Because the list of recorded events can grow very quickly, it uses special tracking methods on things like time and camera IDs so that the system can always search and find information fast.

4.5.1.3 MinIO Object Storage

MinIO Object Storage acts as the main video archive for the system. It stores all the recorded video chunks that are created when an anomaly is detected or when a user specifically asks to record a live stream. Because keeping large video files would take up too much space on the local server or the main database, MinIO provides a scalable solution to handle and organize this data.

5. Test Cases

5.1 Test Cases for Functional Requirements

5.1.1 Computer Vision Services Test Cases

Test ID	TC-01
Category	Functional
Objective	Verify YOLO object detection for standard warehouse items.
Procedure	1. Initialize the CV subsystem container. 2. Feed a pre-recorded RTSP test stream of the loading dock containing distinct boxes and pallets. 3. Intercept the JSON output from the YOLO inference engine. 4. Validate the bounding box coordinates against the ground truth. 5. Check the confidence scores for the 'box' and 'pallet' classes.
Expected Results	The object detection module correctly identifies boxes and pallets, drawing accurate bounding boxes with confidence scores exceeding the predefined minimum threshold.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-02
Category	Functional
Objective	Verify the synthetic data generation pipeline outputs.
Procedure	1. Define a JSON/YAML configuration for a specific environment

	template (e.g., truck interior, warehouse shelf layouts). 2. Execute the synthetic data generation script. 3. Inspect the designated output directory. 4. Cross-reference the generated images with their corresponding exported annotation files.
Expected Results	The pipeline successfully outputs generated images alongside correctly formatted annotation files (containing bounding boxes, tracking IDs, and anomaly labels).
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-03
Category	Functional
Objective	Validate data augmentation for model training.
Procedure	1. Input a verified clean image of a truck interior with its annotations. 2. Trigger the real-life scenario augmentation module. 3. Apply sequential transformations: gaussian noise, motion blur, and localized dimming (lighting changes). 4. Programmatically verify that the bounding box coordinates remain correctly aligned with the transformed objects.
Expected Results	The system outputs augmented images simulating scenarios without breaking the bounding box annotations.
Priority	Medium
Date Tested	(To be filled in Final Report)

Test Result	(To be filled in Final Report)

Test ID	TC-04
Category	Functional
Objective	Test persistent object tracking across frames.
Procedure	1. Feed a 30-second video clip where an item moves from the warehouse staging area into the truck region. 2. Introduce a temporary visual occlusion. 3. Log the unique tracking IDs assigned by the tracking module frame by frame.
Expected Results	The tracking module assigns an ID to the item and successfully maintains that exact same ID throughout the movement.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-05
Category	Functional
Objective	Verify anomaly detection for damaged goods.
Procedure	1. Stream a specific test video clip showing a visibly crushed box being loaded. 2. Monitor the Anomaly Detection module's classifier output probabilities. 3. Confirm the 'damage' class probability exceeds the predefined

	condition threshold. 4. Verify that an event payload (containing severity, confidence score, and a short video clip URI) is generated and published to the Alert Service.
Expected Results	The module successfully flags the damage, generates the correct event schema, and extracts the short video clip with timestamps for visual evidence
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-06
Category	Functional
Objective	Evaluate model robustness under stress conditions.
Procedure	1. Load a validation dataset specifically curated with different scenarios. 2. Execute the evaluation script to run inference on the entire dataset. 3. Parse the resulting evaluation logs to calculate the mean Average Precision. 4. Compare the results against the baseline accuracy threshold.
Expected Results	The model completes the evaluation pipeline, and the parsed log files confirm that the accuracy remains within acceptable limits.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-07
Category	Functional
Objective	Verify worker safety equipment detection.
Procedure	1. Input a video frame showing a warehouse worker walking through a designated zone without a required safety vest/helmet. 2. Ensure the CV subsystem is actively tracking humans. 3. Check the anomaly event generation stream results. 4. Verify the event metadata sent to the Alert Service.
Expected Results	The CV subsystem identifies the worker, successfully flags the missing safety equipment, and pushes an event containing the event details, confidence score, and a replay option.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-08
Category	Functional
Objective	Validate barcode detection and decoding.
Procedure	1. Present an item with a clear barcode moving through the defined scanning zone. 2. Trigger the Object Detection module to locate the barcode's surface region.

	3. Pass the cropped barcode region to the decoding library. 4. Cross-reference the decoded string with the test Inventory Database.
Expected Results	The module locates the barcode, successfully decodes the metadata, validates it against the database, and updates the item's location and "count" state
Priority	Low
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-9
Category	Component
Objective	Check that the YOLO service loads the trained model weights successfully at start.
Procedure	1. Start the YOLO Service container. 2. Observe the startup logs of the service. 3. Verify that the configured model weights file is found and loaded. 4. Confirm that the service enters ready state without runtime errors.
Expected Results	The YOLO service loads the model weights successfully and in ready state for inference.
Priority	High
Date Tested	03.02.2026
Test Result	Successful

Test ID	TC-10
Category	Training

Objective	Check that the dataset loader rejects annotation files with missing or malformed labels before training starts.
Procedure	1. Prepare a training dataset containing at least one malformed annotation file. 2. Start the training or fine-tuning pipeline. 3. Observe the dataset validation step. 4. Check whether the problematic file is reported correctly.
Expected Results	The training pipeline stops or skips invalid data according to configuration, reports the malformed annotation clearly, and does not proceed silently with broken labels.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	Partial success, needs better logs and reporting.

Test ID	TC-11
Category	Performance
Objective	Check that inference service keeps the memory usage within acceptable limits during long running stream processing.
Procedure	1. Start the CV subsystem with a continuous RTSP test stream. 2. Let the service run for an extended period (more than 30 minutes) 3. Record the memory usage at regular intervals. 4. Compare the memory usage trend over time. 5. If memory usage increases significantly, test with longer time intervals to stress test the system. 6. Repeat 3-5
Expected Results	Memory usage remains stable after without abnormal growth or memory leak symptoms during continuous inference.
Priority	High

Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

5.1.2 Live Stream Acquisition, Relay, and Delivery Tests

Test ID	TC-12
Category	Functional
Objective	Test that the new camera configuration can be added to the system as an IoT device.
Procedure	1. Log in as an admin. 2. Click on the ‘Devices’ in the sidebar. 3. Add the ‘+’ button to insert a new configuration. 4. Submit device type, name, IP address, and RTSP URL. 5. See the success message in the user interface.
Expected Results	The new configuration is successfully added to the PostgreSQL.
Priority	High
Date Tested	02.12.2025
Test Result	Pass

Test ID	TC-13
Category	Functional
Objective	Check that MediaMTX can receive the stream coming from the RTSP camera.
Procedure	1. After adding camera configuration via the UI, set the RTSP stream URL in the MediaMTX yaml file. 2. Starting the connection between the camera device and the

	MediaMTX server. 3. Check the logs to confirm the active stream.
Expected Results	The stream server receives the stream without errors.
Priority	High
Date Tested	03.12.2025
Test Result	Pass

Test ID	TC-14
Category	Functional
Objective	Check that the stream server can relay the stream to the frontend view.
Procedure	1. Log in as an admin. 2. Click on the ‘Devices’ in the sidebar. 3. Click on the ‘View Stream’ button. 4. Display the real-time stream on the videoplayer component.
Expected Results	The stream can be played in the video player.
Priority	High
Date Tested	03.12.2025
Test Result	Pass

Test ID	TC-15
Category	Functional
Objective	Check that Stream Health Monitor can detect a lost camera connection.
Procedure	1. Log in as an admin. 2. Click on the ‘Devices’ in the sidebar.

	3. Click on the ‘View Stream’ button. 4. Display a real-time line chart showing the current ping and latency values. 5. The line chart can indicate when the connection is lost and notify the user.
Expected Results	Stream Health Monitor reports the camera as disconnected.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-16
Category	Security
Objective	Check that WebRTC signalling is prevented when no valid session token is present.
Procedure	1. Call the ‘start stream’ endpoint via Swagger UI. 2. Observe whether the WebRTC Signal Handler blocks the signalling request at the gateway layer.
Expected Results	The signalling request is rejected by the WebRTC Signal Handler.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-17
Category	Functional
Objective	Check whether MediaMTX can relay the stream to the YOLO service.

Procedure	1. Check that MediaMTX receives the stream and the YOLO service is running. 2. Check the YOLO microservice logs that it can generate metadata using the MediaMTX stream relay. 3. Display the stream with YOLO-drawn boundary boxes on the simulation page.
Expected Results	YOLO service can receive the stream and process it.
Priority	High
Date Tested	03.03.2026
Test Result	Pass

5.1.3 Sensor Polling, Threshold Evaluation, and Alert Publishing Tests

Test ID	TC-18
Category	Functional
Objective	Test that the new sensor configuration can be added to the system as an IoT device.
Procedure	1. Log in as an admin. 2. Click on the ‘Devices’ in the sidebar. 3. Click on the ‘+’ icon under the sensors title. 4. Submit device type, name, and IP address. 5. Seeing the success message box in the user interface.
Expected Results	The new sensor configurations are registered to the PostgreSQL database.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-19
Category	Functional
Objective	Test that the system can read the current value of the registered sensors every 2 seconds.
Procedure	1. After registering a sensor in the system, start the Sensor Polling Service. 2. Observe the incoming data through the simulation page. 3. Verify that reading can be displayed in the UI.
Expected Results	The values can be read successfully from the sensors and displayed on the screen.
Priority	High
Date Tested	12.12.2025
Test Result	Pass

Test ID	TC-20
Category	Functional
Objective	Check that the Sensor Polling Service can detect when a registered sensor loses its connection.
Procedure	1. Start the Sensor Polling Service. 2. Create a case where one of the sensors loses its connection. 3. Observe the Sensor Polling Service response and wait for the warning message in the UI.
Expected Results	The service can detect a lost connection immediately.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

5.1.4 Authentication and Authorization Tests

Test ID	TC-21
Category	Functional
Objective	Check that a user can be added successfully to the system by admin.
Procedure	1. Open the application and login as admin. 2. Click the register user button. 3. Fill the required information. 4. Click the register button
Expected Results	The system successfully registers the new user.
Priority	High
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-22
Category	Functional
Objective	Check that a registered user can log in.
Procedure	1. Open the application login page. 2. Enter the required information. 3. Click the login button.
Expected Results	The system successfully allows the user to log in.
Priority	High
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-23
Category	Functional
Objective	Check that an unregistered user cannot log in.
Procedure	1. Open the application login page. 2. Enter the required information. 3. Click the login button.
Expected Results	The system does not allow the user to log in. Shows the login failed error.
Priority	High
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-24
Category	Functional
Objective	Check that users can only use the authorized endpoints.
Procedure	1. Login the system with different roles (for instance admin). 2. Try to send a POST request to register a user as an admin. 3. Observe that the request is successful. 4. Login the system with different roles (for instance driver). 5. Try to send a POST request to register a user as a driver.
Expected Results	The driver gets unauthorized access as a response and the attempt fails.
Priority	Medium
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-25
Category	Functional
Objective	Check that the admin can delete users.
Procedure	1. Login the system as admin. 2. Find the user management area. 3. Select the user which is wanted to be deleted. 4. Click the delete button.
Expected Results	The user is removed from the system.
Priority	Critical
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-26
Category	Functional
Objective	Check that users can update their accounts.
Procedure	1. Login the system with any role. 2. Find the update account area and click the update account button. 3. Fill the desired fields. 4. Click the confirmation button.
Expected Results	The user information is updated with the new information. .
Priority	Medium
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-27
Category	Functional
Objective	Check that the system logs out the user.
Procedure	1. Login the system with any role. 2. Find the logout button and click.
Expected Results	The system revokes all refresh tokens and access tokens in backend and frontend are revoked, then, the user is led to the login page.
Priority	Critical
Date Tested	21.02.2026
Test Result	Pass

Test ID	TC-28
Category	Functional
Objective	Check that a warehouse operator or admin roles can register a truck into the system.
Procedure	1. Open the application. 2. Login as operator or admin. 3. Find and click the add truck button. 4. Fill the required information. 5. Click the add button..
Expected Results	The system successfully adds a new truck to the system.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

5.1.5 Cross Subsystem Tests

Test ID	TC-29

Category	Integration
Objective	Check that duplicate anomaly events are not sent to the dashboard repeatedly for the identical instance of the anomaly within the short time window.
Procedure	1. Start the CV subsystem and Notification Service 2. Feed a test stream containing the same instance of an anomaly for some determined period. 3. Monitor the event output generated by the CV subsystem. 4. Count how many anomaly notifications are published for the same evene window.
Expected Results	The system suppresses repeated duplicate anomaly events and posts to notification service only the intended number of notifications.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-30
Category	Integration
Objective	Check that the YOLO service rejects frames with unsupported input (these can be caused by corrupted frames because of packet loss, partial frames because of dropped UDP packages …), streaming service handles the failure without interrupting the stream processing.
Procedure	1. Start the Streaming Service, MediaMTX relay and YOLO service. 2. Feed a test RTSP stream into MediaMTX. 3. Inject corrupted or partial frames into the stream. 4. Observe the YOLO service logs and frame processing behaviour. 5. Verify that the streaming service continues processing subsequent valid frames

Expected Results	The YOLO service rejects invalid frames and logs the issue, the feed may be snapshotted for further investigation. The streaming pipeline continues processing the stream without crashing or interrupting the inference pipeline.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-31
Category	Integration
Objective	Check that a model produced by the fine-tuning pipeline can be loaded directly by the deployed YOLO service without manual modification.
Procedure	1. Complete a fine-tuning run and obtain the exported model weights. 2. Replace the previous deployment weights with the newly trained weights. 3. Restart the YOLO service. 4. Run inference on a known validation sample. 5. Verify that the service uses the new model successfully.
Expected Results	The deployed YOLO service loads the newly fine-tuned model directly and performs inference successfully without additional conversion or manual patching.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-32

Category	Compatibility
Objective	Check that inference results remain valid for input streams with different supported resolutions.
Procedure	1. Prepare test streams of the same scene in different supported resolutions. 2. Run the CV subsystem on each stream separately. 3. Record the detected objects and anomaly decisions. 4. Compare the output labels across different test resolutions
Expected Results	The CV subsystem processes all supported resolutions successfully and consistently detects needed data points. Reports with the confidence score if resolution is below a certain threshold.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-33
Category	Usability
Objective	Check that anomaly outputs shown on the interface are understandable and visually verifiable by the user
Procedure	1. Log in as an admin or operator. 2. Open a page where anomaly detections are listed. 3. Trigger or load a known anomaly case. 4. Inspect whether the interface shows ; a. the anomaly type, b. Timestamp, c. confidence score, d. and related clip or frame evidence. 5. Ask the tester to identify the anomaly only from the displayed information.

Expected Results	The user interface presents anomaly results clearly enough for a non-technical user to understand what happened and verify the detection visually.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-44
Category	Integration
Objective	Check that events generated by backend services are correctly published through the Event Bus and stored in the PostgreSQL event log.
Procedure	1. Start the Event Bus service or other subsystem that can generate an event. 2. Trigger a known event 3. Observe the Event Bus logs to verify that the event is received. 4. Query the PostgreSQL event table. 5. Verify event entry is stored with correct metadata such as time, event type etc.
Expected Results	The event generated by the subsystem is successfully received by the Event Bus and stored in PostgreSQL event log with correct metadata.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

5.1.6 Route Planning Tests

Test ID	TC-35
Category	Functional
Objective	Verify that the route planning service recalculates ther route when a new delivery order is added during an operation.
Procedure	1. Start with an active route 2. Add a new delivery order 3. Observe updated route assignments for trucks.
Expected Results	The routing service recalculates routes and integrates the new order into the most optimal route without violating capacity or time constraints.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-36
Category	Functional
Objective	Verify that the routing algorithm respects truck capacity constraints.
Procedure	1. Define truck capacity limits. 2. Create delivery orders whose total load exceeds the truck capacity. 3. Run the route planning service. 4. Observe generated routes and assigned deliveries.
Expected Results	The route planning system assigns deliveries without exceeding the truck capacity. Excess deliveries are assigned to another vehicle or marked as pending.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-37
Category	Integration
Objective	Verify that damaged item detection from the CV subsystem affects routing decisions.
Procedure	1. Load items into a truck. 2. Simulate the CV subsystem detecting damaged cargo. 3. Check routing decision.
Expected Results	The shipment is flagged and excluded from delivery routes until resolved.
Priority	Low
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-38
Category	Functional
Objective	Verify that deliveries are distributed across multiple trucks.
Procedure	1. Add orders exceeding one truck's capacity. 2. Define multiple trucks with different capacities. 3. Run route planning. 4. Observe route assignments.
Expected Results	Deliveries are distributed among trucks while minimizing distance and respecting capacity constraints.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-39

Category	Integration
Objective	Verify that the routing system updates routes when a delivery order is cancelled.
Procedure	1. Generate a route with multiple stops 2. Cancel an order 3. Check the route plan
Expected Results	The cancelled order is removed and routes are recalculated to maintain optimal delivery sequences.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-40
Category	Performance
Objective	Verify that the system performs efficiently under heavy workload conditions.
Procedure	1. Simulate high numbers of route planning requests. 2. Generate multiple CV events simultaneously. 3. Measure response time and system latency.
Expected Results	System maintains acceptable response times and route planning completes within the defined performance limits.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-41
Category	Functional

Objective	Verify that route planning respects delivery time window constraints.
Procedure	1. Create delivery orders with specific time windows. 2. Run the route planning service. 3. Check assigned delivery order sequence.
Expected Results	Routes satisfy all time window constraints. If a route cannot meet the time window, the order is flagged or scheduled for another route.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-42
Category	Functional
Objective	Verify system behavior when the routing algorithm cannot find a feasible solution.
Procedure	1. Define delivery orders with strict time windows and large load requirements. 2. Configure trucks with insufficient capacity or conflicting schedules. 3. Run the route planning service. 4. Check results.
Expected Results	The routing service identifies that no feasible solution exists and returns a clear error message or flags the affected orders for manual handling.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-43
Category	Integration

Objective	Verify that generated routes are correctly displayed on the frontend interface.
Procedure	1. Generate a route plan through the backend routing service. 2. Open the web interface. 3. Navigate to the route visualization page. 4. Load the generated route plan. 5. Inspect the displayed route information.
Expected Results	The interface correctly displays route paths, assigned trucks, delivery order sequence, and estimated arrival times without visual or data inconsistencies.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-44
Category	Functional
Objective	Verify that routes are recalculated when truck capacity is reduced due to new occupancy information.
Procedure	1. Generate a route plan with deliveries assigned to a truck. 2. Simulate CV subsystem reporting that additional items occupy space in the truck. 3. Update the truck's remaining capacity in the backend. 4. Observe new delivery assignments.
Expected Results	The routing system removes excess deliveries from the truck and reassigns them to other trucks or future routes.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

5.2 Test Cases for Non Functional Requirements

Test ID	TC-45
Category	Non-functional
Objective	Validate real-time inference latency constraints.
Procedure	1. Inject a precise timestamp payload into a continuous 60 FPS video frame at the streaming gateway. 2. Stream the video to the CV Subsystem. 3. Capture the resulting event payload at the backend integration layer. 4. Calculate the delta between the initial frame injection timestamp and the event generation timestamp.
Expected Results	The latency from video frame ingestion to event detection is strictly <= 1000 milliseconds.
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-46
Category	Non-functional
Objective	Test CV inference failure recovery.

Procedure	1. While the CV subsystem is actively processing an RTSP stream, manually terminate the inference engine process. 2. Monitor the system monitor's heartbeat logs. 3. Wait for the required millisecond window to pass without a "Success" signal. 4. Observe the system's automated restart and fallback protocols.
Expected Results	The system detects the loss of heartbeat, attempts an automatic restart, and flags the stream as "Interrupted" while sending an alert log to the administration
Priority	High
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-47
Category	Check that MediaMTX can handle multiple stream connections.
Objective	Non-functional
Procedure	1. Add at least 5 camera configurations as admin. 2. Open the live streams in Warehouse>Streams tab. 3. Display the streams switching among the camera checkboxes.
Expected Results	All streams can be displayed without bandwidth and delay problems.
Priority	High
Date Tested	03.12.2025
Test Result	Pass

Test ID	TC-48
Category	Non-functional
Objective	Check that the Sensor Polling Service can handle consecutive HTTP requests at 2-second intervals under system load.
Procedure	1. Start the Sensor Polling Service. 2. Preparing a high-load environment for the running services. 3. Observe the latency and interval timestamps.
Expected Results	Polling shows no latency even under high load.
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-49
Category	Non-Functional
Objective	Check that users can only see the determined features on their main page.
Procedure	1. Login the system with different roles. 2. Check the features provided on the page.
Expected Results	Users can only see the features related to their roles, for instance, the driver can only see the map and account handling but the admin can see warehouse, map, streams and every other features. .
Priority	Medium
Date Tested	(To be filled in Final Report)
Test Result	(To be filled in Final Report)

Test ID	TC-50
Category	Non-Functional
Objective	Check that users can only reach their authorized pages.
Procedure	1. Login the system with manager or operator role. 2. They can access the order handling page. 3. Then logout and login as driver. 4. The driver tries to route to the order handling or any monitoring page other than route monitoring.
Expected Results	The driver is led to the 403 no permission page.
Priority	Critical
Date Tested	12.03.2026
Test Result	Pass

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

Technology and Platform Constraints: The system's architecture and development workflow are shaped by its reliance on RTSP-compatible security cameras, limiting input sources to devices capable of delivering real-time video streams. This requirement influences hardware selection and ensures compliance with streaming standards. The Computer Vision pipeline must also rely on real-time inference frameworks, such as YOLO variants, that support video decoding and low-latency execution. On the backend, a containerized Docker deployment is required, and the frontend must rely on cross-platform web technologies, thereby preventing the use of platform-specific UI toolkits.

Real Time Processing Constraints: The Computer Vision module, including decoding, detection, and event generation, must operate within a latency of 1,000 ms. Any computationally heavy approaches that exceed this limit cannot be used. Continuous real time streaming is required; paused or delayed frames are unacceptable because operators must react instantly to operational issues. The Route Optimization module must also compute updated routes within a tight time budget as warehouse and truck states change. Since vehicle routing is NP-hard, exact solvers (e.g., full linear or integer programming) are not feasible for real-time use, making heuristic or approximation algorithms mandatory.

Integration Constraints: All modules must communicate through standardized REST/HTTP interfaces to retain consistency across components. Data must be stored in the designated storage layers: a relational database for structured truck, warehouse, and configuration data; Redis for cached real-time states and temporary results; and object storage for videos and images. This separation prevents ad hoc storage patterns and ensures predictable access. The Computer Vision module must also adhere to a predefined event logging schema to maintain traceability and ensure that event data is consumable by other services.

Resource Constraints: The entire pipeline must be optimized for efficient memory and compute use. Video parameters, including clip duration, resolution, compression, bitrate, and

FPS, must be controlled to avoid unnecessary storage consumption. The Route Optimization module must also utilize algorithms that remain efficient as the problem size increases, since memory usage in distance matrices, cost tables, and constraint sets grows rapidly. As a result, industry-standard optimization techniques are required.

Security and Compliance Constraints: All communication, both video streams and API traffic, must use secure protocols such as HTTPS/TLS. GDPR and KVKK regulations impose strict retention limits on video clips, event logs, and operational records, mandating the implementation of automatic deletion mechanisms. Role-based authentication and authorization ensure that each user type (operators, security staff, administrators) only accesses data and functions appropriate to their responsibilities.

Testing and Deployment Constraints: Because live camera access is not always available during development, the system must support testing with recorded videos to enable repeatable evaluations. Testing primarily focuses on modular validation, controlled simulations, and small-scale experiments, rather than full industrial deployment scenarios.

Economic Constraints: Since the project must operate without enterprise-grade resources, such as GPU servers, specialized networking systems, or high resolution cameras. All components must run on standard workstations and low-cost development servers. This limits model complexity and prevents the use of computationally expensive solvers. To control long-term operational costs, the system cannot rely on paid cloud services, commercial APIs, or licensed optimization software; instead, it must use open source libraries and cost efficient storage solutions. Video storage must be managed through clip duration limits, compression, and automated retention policies. Camera acquisition is also constrained to affordable RTSP compatible devices already deployed in warehouses.

Ethical Constraints: TırGöz must comply with ethical standards related to privacy, transparency, and appropriate use of surveillance technologies. Video processing must remain strictly limited to logistics related tasks, and the system must avoid collecting or storing unnecessary personal data. The Computer Vision module must not be repurposed for employee monitoring beyond project scope. Ethical design also requires transparency regarding detections and recommendations: operators should understand why alerts are generated,

allowing the system to support rather than replace their decisions. Automated outputs must be clear, interpretable, and aligned with responsible use guidelines.

6.2 Standards

UML 2.5.1: Used for all modeling artifacts (use case, class, sequence, state, activity diagrams). Ensures a consistent and standardized representation of system structure and behavior.

IEEE 29148: Defines structure and quality criteria for functional and non-functional requirements, ensuring clarity, correctness, and traceability.

IEEE 1058: Provides guidelines for project management documentation such as work packages, schedules, team roles, and deliverables.

IEEE 1016: Specifies how software design descriptions should be structured, supporting architectural clarity, interface definitions, and design documentation.

ISO/IEC/IEEE 12207: Defines the software lifecycle processes across planning, development, testing, deployment, and maintenance.

ISO/IEC/IEEE 15288: Covers system level lifecycle processes, including stakeholder interactions, architectural definition, and integration across services.

ISO/IEC 27001: Guides information security practices, including access control, secure storage, authentication, authorization, and logging, ensuring confidentiality, integrity, and availability.

OWASP ASVS: Used informally to strengthen backend and API security through guidelines on input validation, authentication, session handling, and protection against common vulnerabilities.

ISO/IEC/IEEE 29119: Provides a structured approach to testing, including test planning, test design, execution, and reporting.

IEEE 829: Used conceptually to structure test cases and test documentation.

REST Principles & JSON Schema Standards: Define how microservices communicate, specifying consistent request/response structures and data validation rules.

7. Teamwork Details

7.1 Contributing And Functioning Effectively On The Team

During development, each member contributed to different technical areas of the system. Burak Baştuğ primarily focused on authorization, authentication, and testing processes to ensure that system requirements and deliverables were clearly defined and validated. Arda Öztürk and Berin Su İyici worked on the computer vision inference pipeline, developing mechanisms that convert camera data into operational information such as item counts, occupancy estimates, and anomaly detections. Umut Başar Demir focused on the route planning component to generate and dynamically update delivery routes based on operational constraints and real-time data. Berkin Kağan Ateş focused on system infrastructure, implementing the streaming pipeline, backend services, and the web interface used to visualize system outputs and interact with the application. While each member had primary responsibilities, the team collaborated closely throughout development by reviewing each other's work, assisting with integration tasks, and jointly solving technical challenges to ensure the overall system functioned effectively.

7.2 Helping Creating A Collaborative And Inclusive Environment

We emphasized open communication and transparency throughout the development process. Collaboration was supported through the use of structured tools and processes, including Jira for project management and task tracking and GitHub for version control and collaborative development. Jira allowed the team to maintain a clear backlog and monitor progress on work packages. GitHub enabled multiple members to contribute simultaneously through development branhes, ensuring that all contributions are reviewed before being merged into the main codebase.

We held regular meetings to discuss our progress and future objectives. These meetings helped maintain a shared understanding of the system architecture and current project status.

We also made a conscious effort to maintain an inclusive and respectful working environment. Members supported each other in learning unfamiliar technologies such as computer vision pipelines, microservices architecture, and real-time data processing. Spontaneous discussions

allowed experienced members to guide others in specific technical areas. This approach ensured that everyone could actively participate in the development process and improve their technical skills throughout the project.

Additionally, we considered ethical and social factors during the design process, particularly with respect to privacy concerns related to continuous video monitoring in warehouse environments. By discussing these concerns collectively and incorporating ethical safeguards into the system design, we ensured that the project aligned with professional engineering responsibilities and societal expectations.

7.3 Taking Lead Role And Sharing Leadership On The Team

Leadership in the project is distributed among team members, while the task designations are formed along the microservice architecture. Each subsystem has a designated member responsible for development tasks, and ensuring that implementation and report deliverables are both completed on time. This approach ensures that decisionmaking responsibilities are shared rather than concentrated in a single individual.

For example, Berkin Kağan Ateş led the implementation of the frontend and backend infrastructure, ensuring the integration of multiple system components into a cohesive architecture. All members had decisionmaking capabilities on the parts they were designated to lead, while also carrying the responsibility to notify other members on their decisions and taking feedback, if necessary.

Despite these designated leadership roles, decision-making remained collaborative. Important architectural decisions, such as the selection of microservices architecture, event-based communication, and model latency constraints, were discussed collectively before implementation. Overall, our distribution of responsibilities, combined with continuous communication, enables the us to successfully progress toward the project objectives.

9. References

[1] Hikvision – Smart Logistics / Warehouse Solutions
https://www.hikvision.com/en/solutions/solutions-by-industry/logistics/
Accessed: Mar. 11, 2026.

[2] Hikvision – AI-Powered Logistics Monitoring Systems
https://www.hikvision.com/en/solutions/solutions-by-scenario/warehouses/
Accessed: Mar. 13, 2026.

[3] Hikvision – Company Overview and Global Video Surveillance Solutions
https://www.hikvision.com/en/about-us/

[4] Descartes Systems Group. *Dock Scheduling and Yard Management Solutions.*
https://www.descartes.com/solutions/transportation-management/dock-scheduling-and-yard-management

[5] Descartes Systems Group, "Fleet Analytics and AI," Descartes Routing, Mobile & Telematics Solutions. Available:
https://www.descartes.com/solutions/routing-mobile-and-telematics/route-execution-and-fleet-performance-management/fleet-analytics-and-ai
Accessed: Mar. 11, 2026.

[6] Zebra Technologies. "Machine Vision and Fixed Industrial Scanning Software."
https://www.zebra.com/us/en/software/machine-vision-and-fixed-industrial-scanning-software.html
Accessed: Mar. 11, 2026.

[7]Bluenviron, "Bluenviron/mediamtx: Ready-to-use SRT / webrtc / RTSP / RTMP / LL-hls / MPEG-TS / RTP media server and media proxy that allows to read, publish, proxy, record and playback video and audio streams.," GitHub, https://github.com/bluenviron/mediamtx(accessed Mar. 13, 2026).