Janeeb Solutions

SYSTEM REQUIREMENTS SPECIFICATION

Extended Planning Instrument for Unpredictable Spaces and Environments

Introduction

Vision and Objectives

Given the potential associated with South Africa's logistics juggernaut, our project aims to revolutionize the efficiency of goods transportation. With an annual movement of 1.5 billion tonnes of goods, contributing 10% to the nation's GDP, and employing a million-strong workforce, the logistics sector is a cornerstone of the South African economy. Our mission is to design a system that transforms inefficiency into a symphony of precision, ensuring that every truck is a marvel of intelligent packing and optimal space management. This project isn't just an upgrade, we see it as a revolution in efficiency.

Business Need

The logistics sector in South Africa, despite its massive scale, suffers from significant inefficiencies in space utilization within transportation vehicles. These inefficiencies lead to increased operational costs, reduced profit margins, and environmental impacts due to suboptimal usage of fuel and increased trips. There is a clear business need for an advanced system that can dynamically adjust the placement of goods in real-time, optimizing the available space within logistics trucks. By addressing this need, our project will contribute to cost savings, improved profitability, and a reduction in environmental impact.

Project Scope

The scope of this project encompasses the development of a comprehensive system designed to enhance the efficiency of goods transportation within logistics trucks. Key components of the project include:

- **Dynamic Packing Algorithm:** An intelligent algorithm capable of real-time adjustment of goods placement, prioritizing efficient space utilization to maximize the number of goods transported in a single trip.
- **Machine Learning Model:** A robust machine learning model that learns from historical data, user feedback, and real-world outcomes to continuously enhance the packing algorithm's performance.
- **Manager Interface:** A user-friendly interface for logistics managers to input specific constraints, preferences, and priorities, facilitating customized packing solutions.
- **Real-time Dashboard:** A dashboard that displays real-time information about the packing progress and the algorithm's performance, providing valuable insights and facilitating decision-making.

- **Simulation Renderer:** A visual simulation tool to showcase the packing process, enhancing understanding and showcasing the algorithm's effectiveness.

By addressing these components, our project aims to deliver a scalable and high-performance system capable of transforming the logistics sector in South Africa.

User Stories

1. Logistics Managers

1.1 As a logistics manager, I need to input specific constraints and preferences for the packing algorithm to ensure that the loading process aligns with operational requirements.

1.2 As a logistics manager, I need to monitor real-time information about the packing progress and algorithm performance through the dashboard to make informed decisions and adjustments.

2. Truck Drivers

2.1 As a truck driver, I need to understand the sequence for unloading goods at various destinations to ensure a smooth and efficient delivery process.

3. Warehouse Staff

3.1 As a warehouse staff member, I need to prepare goods for loading according to the appropriate delivery to ensure the algorithm can optimize the correct packages to be packed.

3.2 As a warehouse staff member, I need to pack the shipment according to the system's real-time information to ensure accurate and efficient loading.

4. IT and System Administrators

4.1 As an IT and system administrator, I need to perform regular maintenance, updates, and troubleshooting to ensure the system operates smoothly and efficiently.

4.2 As an IT and system administrator, I need to manage user access and permissions to ensure that only authorized personnel can input data and access sensitive information.

Functional Requirements

Requirements

F1. Dynamic Packing Algorithm:
1.1 Real-Time Adjustment:
  1.1.1 The algorithm dynamically adjusts the placement of goods in real-time.
  1.1.2 The algorithm prioritizes efficient space utilization to maximize the number of goods transported in a single trip.
1.2 Optimization:
  1.2.1 The algorithm optimizes the available space within the logistics truck to ensure the smallest amount of wasted space.
  1.2.2 The algorithms optimizes the packing solution and considers the unpacking Solution based on where the current delivery is to.
1.3 Constraints and Preferences:
  1.3.1 The algorithm takes into account specific constraints and preferences input by the logistics manager.

F2. Machine Learning Model:
2.1 Learning from Data:
  2.1.1 Model gathers data from gathered data sets.
  2.1.2 Model normalizes data.
  2.1.3 Model processes data to be used for algorithms.
  2.1.4 Model is trained based on the data that has been processed.
2.2 Continuous Improvement:
  2.2.1 The model updates and enhances the packing algorithm based on previous boxes loaded onto containers.

F3. Manager Interface:
3.1 Input Constraints and Preferences:
  3.1.1 Logistics managers can input specific constraints for the packing algorithm.
  3.1.2 Logistics managers can set preferences and priorities for packing.
3.2 User-Friendly Interface:
  3.2.1 The interface is intuitive and easy to use for logistics managers.
3.3 Real-Time Monitoring:
  3.3.1 Managers can monitor packing progress in real-time.

F4. Real-Time Dashboard:
4.1 Data Display:
  4.1.1 The dashboard displays real-time information about the packing process.

4.2 Algorithm Performance:

    4.2.1 The dashboard provides insights into the algorithm’s performance.

4.3 User Interaction:

    4.3.1 Users can interact with the dashboard to view detailed information.

Subsystems

1. Authentication/Authorization Subsystem

- 1.1 User Registration
- 1.2 User Log-In
- 1.3 Password Reset:

2. Dynamic Packing Algorithm Subsystem

- 2.1 Real-Time Adjustment
- 2.2 Optimization
- 2.3 Constraints and Preferences

3. Machine Learning Model Subsystem

- 3.1 Learning from Data
- 3.2 Continuous Improvement

4. Manager Dashboard Interface Subsystem

- 4.1 Input Constraints and Preferences
- 4.2 Algorithm performance and monitoring.
- 4.3 Real-Time data display.

5. Packer Instructions Subsystem

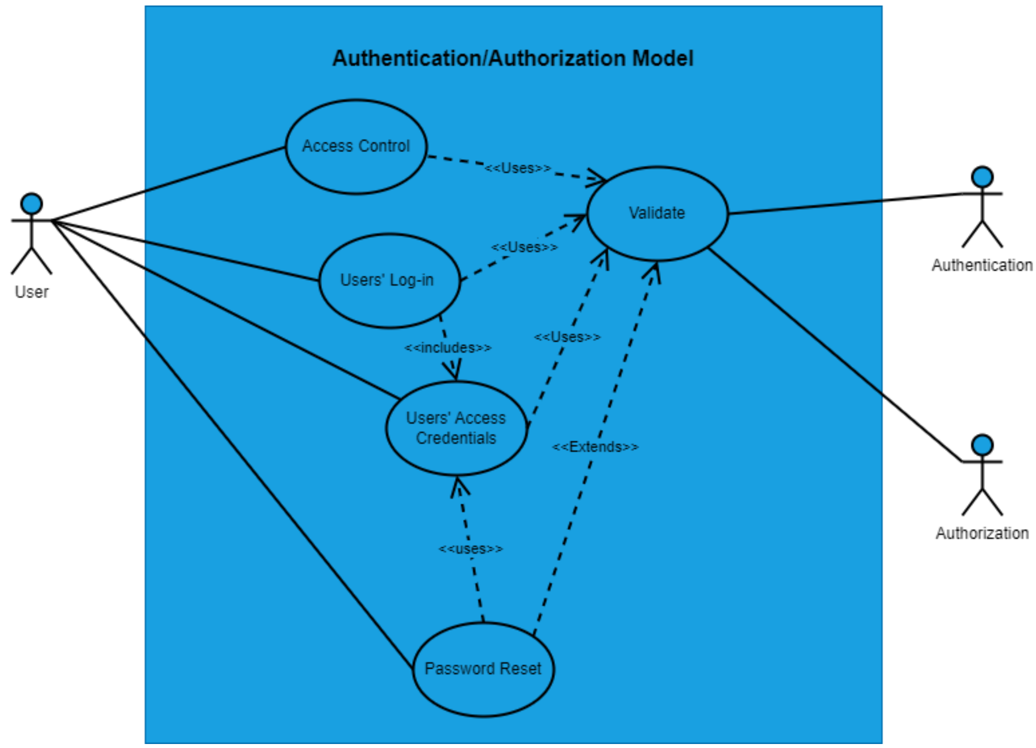
- 5.1 Real-Time packing solution display.
- 5.2 Package QR scanner.

6. Delivery Instructions Subsystem

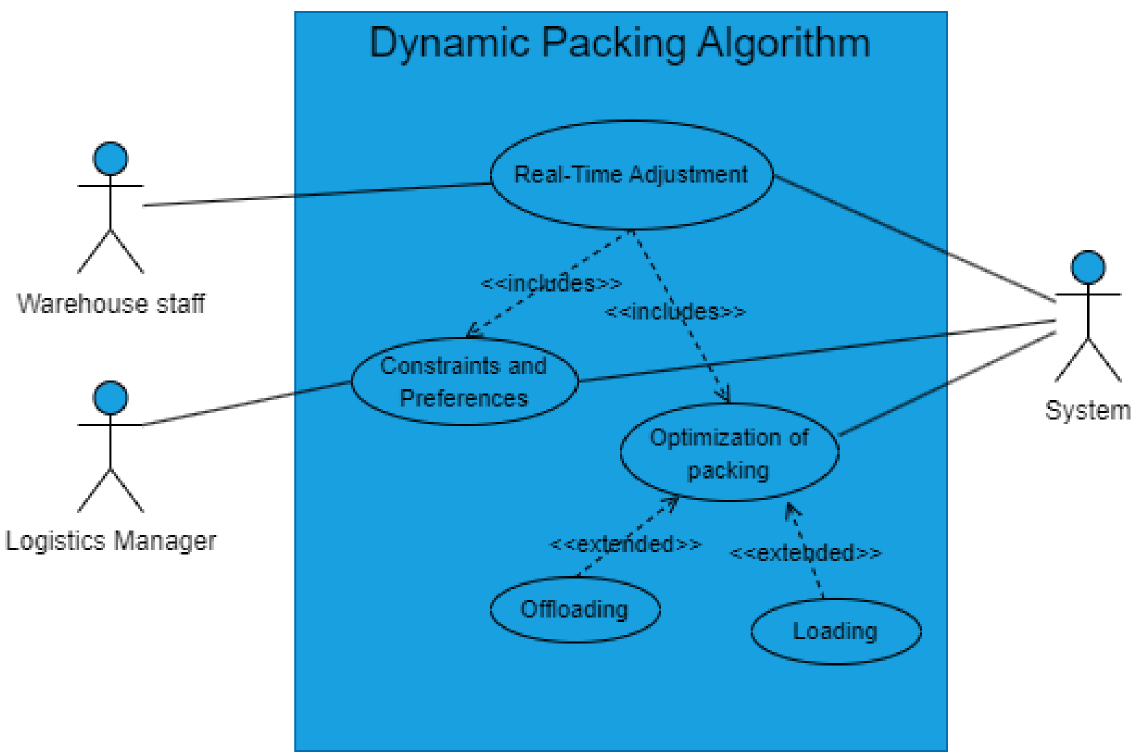
- 6.1. Delivery destination view.
- 6.2. Offloading instructions.

Use Case Diagrams

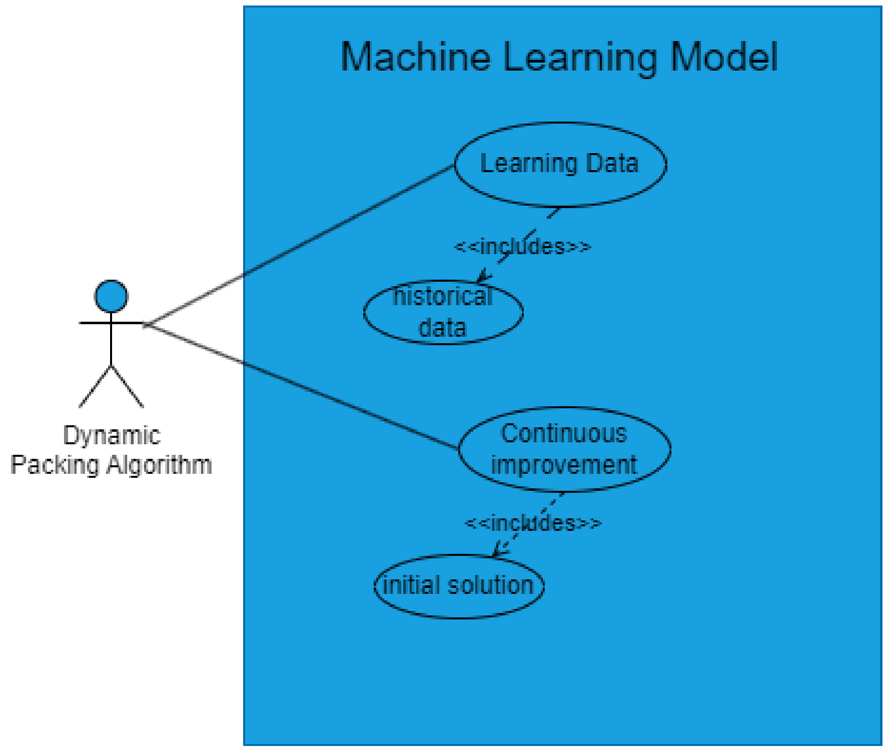
1. Authentication/Authorization



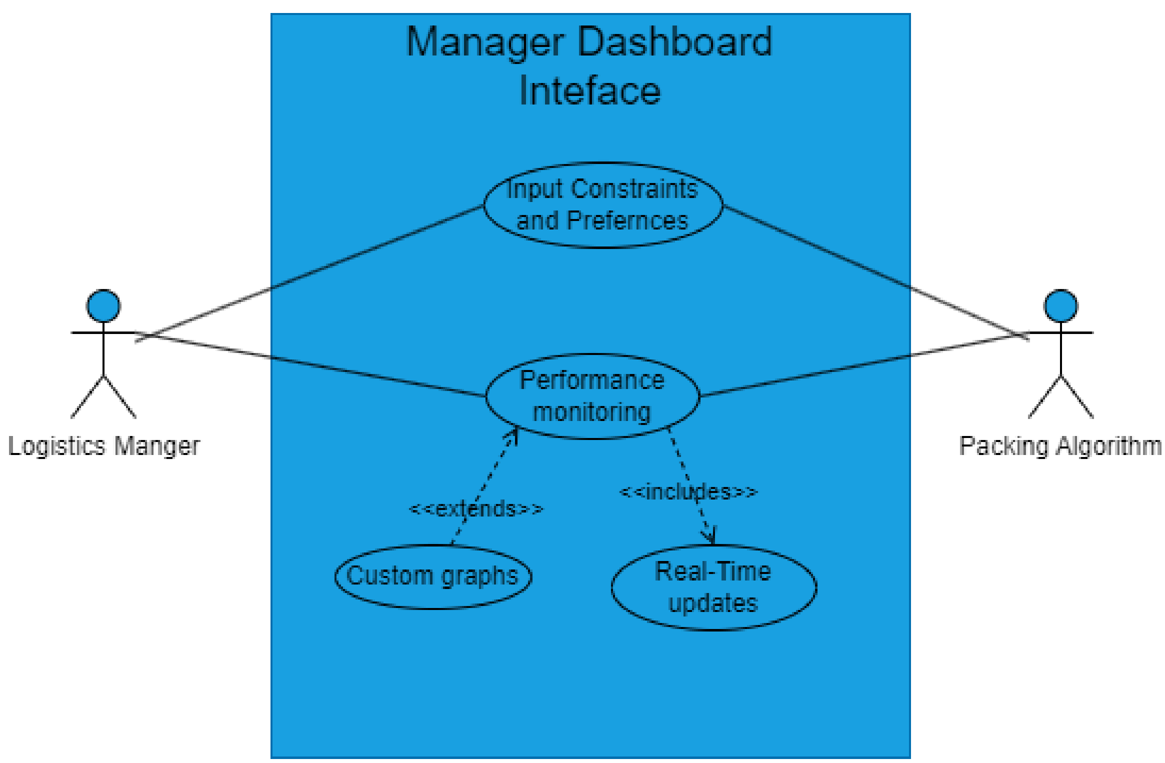
2. Dynamic Packing Algorithm



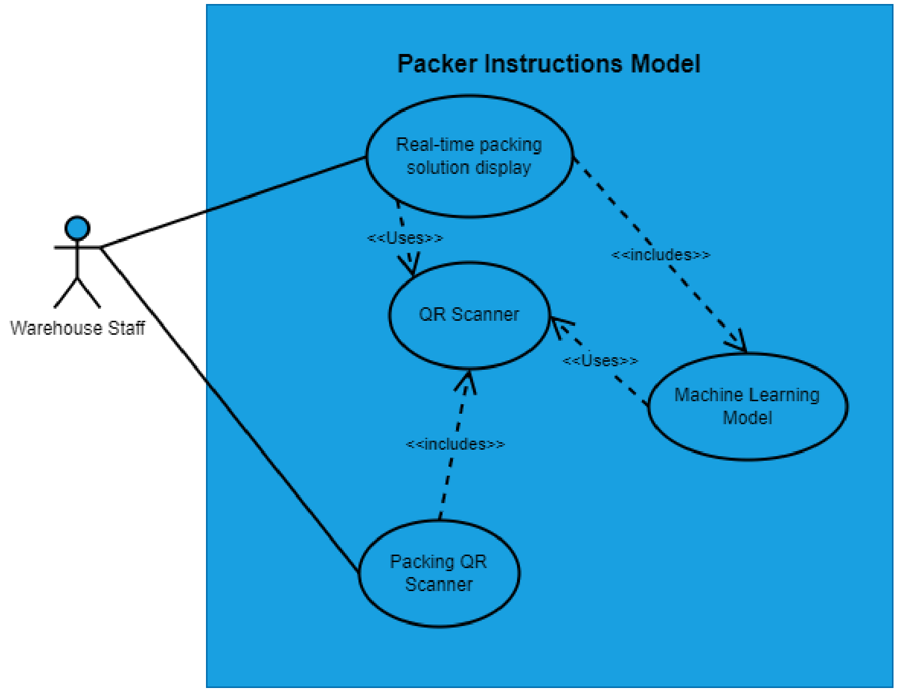
3. Machine Learning Algorithm



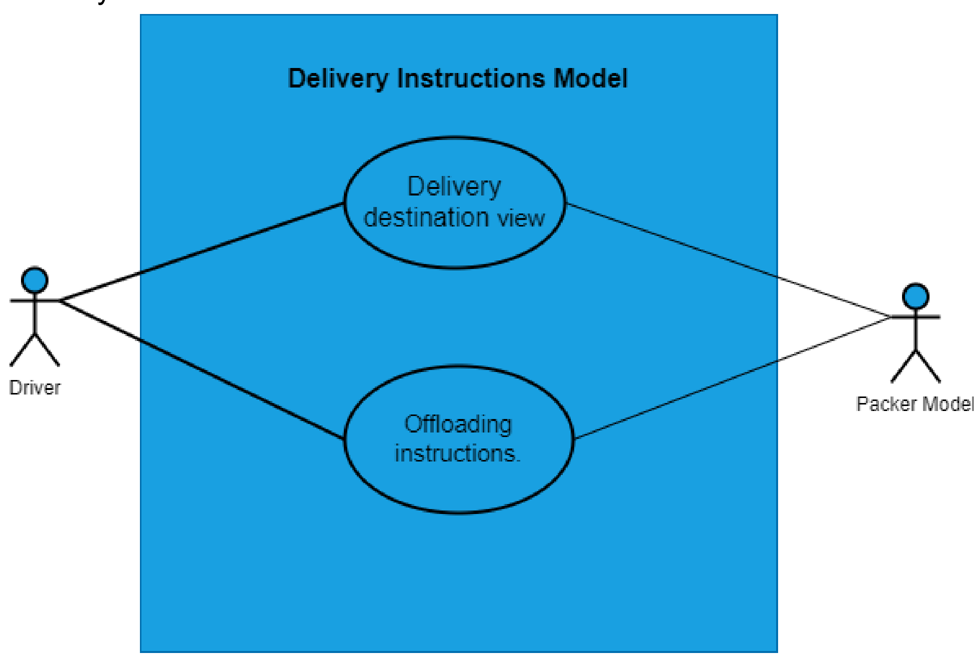
4. Manager Dashboard Interface



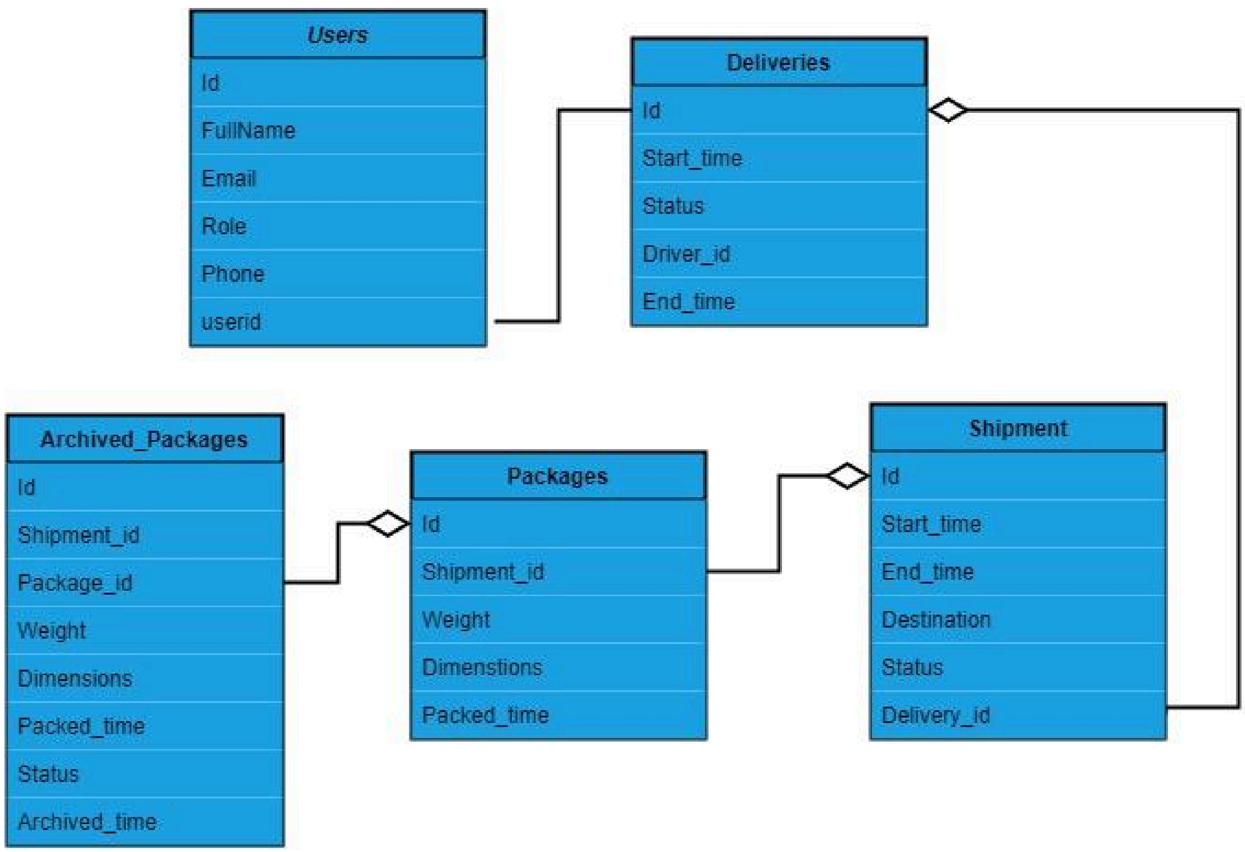
5. Packer Instructions



6. Delivery Instructions



Class Diagram



Non-Functional Requirements

NF1. Performance:

1.1. The application shall have fast response times for user interactions.
1.2. The application's database operations, implemented with Supabase, should have efficient query execution times to ensure quick retrieval and storage of data.
1.3. The system should return the optimal packing route within 2-5 minutes of the provided configuration/input parameters.
1.4. The real time 3D render should reflect optimizations in real time without any delay or buffering.

NF2. Scalability:

2.1. The backend infrastructure, particularly the database layer hosted on Supabase, should be capable of handling increasing loads as the user base expands.
2.2. The system should be scalable in the sense the algorithm could potentially be used for packing cargo trains, or shipping containers

NF3. Security:

3.1. All user authentication and authorization processes, including registration, login, and password reset, must follow the best practices for safe transfer and storage of sensitive user information.
3.2. Data stored in the Supabase database must be encrypted to protect user privacy and comply with relevant data protection regulations.

NF4. Reliability:

4.1. Error handling mechanisms shall be implemented to handle erroneous input and provide helpful feedback to users in case of unexpected errors.
4.2. Continuous integration and deployment pipelines (CI/CD) shall be set up using GitHub Actions to ensure reliable software releases.

NF 5. Usability:

5.1. The application shall have an intuitive and user-friendly interface to ensure ease of use for all user types, including logistics managers, truck drivers, and warehouse staff.
5.2. The interface should be accessible and provide a consistent user experience across various devices and screen sizes.
5.3. The system shall provide clear instructions and feedback to users, guiding them through the process and informing them of any errors or required actions.
5.4 The system shall provide real-time updates to users, ensuring they are always informed of the current status and any changes in the packing process.

Architectural Patterns

For the logistics optimization system described in the project, a Microservices Architecture is a suitable architectural pattern. This pattern breaks down the application into smaller, independently deployable services that communicate over a network. Each service is responsible for a specific piece of functionality and can be developed, deployed, and scaled independently.

1. Authentication/Authorization Service
    - Manages user registration, login, and password reset functionalities to ensure secure access.
2. Dynamic Packing Algorithm Service
    - Adjusts and optimizes the placement of goods in real-time, considering constraints and preferences.
3. Machine Learning Model Service
    - Learns from data and continuously improves the packing algorithm.
4. Manager Dashboard Interface Service
    - Provides an interface for inputting constraints, monitoring algorithm performance, and displaying real-time data.
5. Packer Instructions Service
    - Displays real-time packing solutions and integrates package QR scanning for verification.
6. Delivery Instructions Service
    - Organizes delivery destinations and provides offloading instructions to ensure efficient delivery processes.

In the microservices architecture, services communicate with each other through well-defined APIs. This approach ensures loose coupling, allowing each service to be developed, deployed, and maintained independently. For real-time data updates and event-driven communication, supabase's built-in real-time functionality is used to facilitate efficient and reliable data exchange between services.

The microservices architecture offers several significant benefits. Enhanced flexibility allows teams to develop, deploy, and scale services independently, making it easier to manage and evolve the system. Improved maintainability is achieved through smaller, modular services that are easier to understand, test, and maintain. Greater resilience is another advantage, as the failure of one service does not necessarily impact others, thus improving the system's overall robustness. Additionally, technology agility allows different services to leverage the most suitable technologies for their specific requirements, optimizing performance and development efficiency.

Design Patterns

Facade

The Manager Dashboard Interface will act as the facade, providing a simplified and unified interface for the logistics manager to interact with the underlying subsystems like the Dynamic Packing Algorithm Service, Machine Learning Model Service, Packer Instructions Service, and Delivery Instructions Service.

The facade will expose methods for the manager to input constraints, preferences, and priorities for the packing algorithm. It will communicate with the Dynamic Packing Algorithm Service, translating the user input and retrieving the generated packing solution for presentation.

The facade will also provide an interface for the manager to review and provide feedback on the packing solutions. It will collect this feedback and pass it to the Machine Learning Model Service to improve the algorithm over time. Additionally, it will retrieve and display insights or recommendations from the Machine Learning Model Service.

Furthermore, the facade will allow the manager to access packing instructions for warehouse staff and delivery instructions for truck drivers by communicating with the respective services and presenting the information in a user-friendly format.

By acting as the facade, the Manager Dashboard Interface will handle the complexities of communicating with the subsystems, promoting better code organization, readability, and separation of concerns. It also facilitates easier maintenance and extensibility, as changes to the subsystems won't affect the facade's interface as long as the contracts remain unchanged.

Singleton

The Singleton Pattern ensures that a class has only one instance and provides a global point of access to that instance. This is particularly useful for managing shared resources and maintaining consistent state across the system.

A singleton database connection can efficiently manage database connections, optimizing resource utilization and maintaining connection limits for improved performance and stability.

Security services managing authentication and authorization also benefit from the Singleton Pattern by providing a consistent and universally accessible point of control, enhancing overall security.

Constraints

The primary constraint for our logistics optimization system is the requirement to use specific datasets provided in the linked documents. These datasets will serve as the foundation for developing and training the machine learning models, as well as for testing and validating the packing algorithms. There are no additional technical or operational constraints at this stage.

Data Set 1

Data Set 2

Service Contracts

Register Request and Response

Register Request:
- email (string): The user's email address.
- password (string): The user's chosen password.
- firstName (string): The user's first name.
- lastName (string): The user's last name.
- country (countryEnum): The user's country, represented as an enumeration value.

This information is sent when a user wants to create a new account.

Register Response:
- status (string): The status of the registration request, indicating success or failure.
- message (string): A message providing additional information about the status.
- timestamp (number): The time when the response was generated.

This information is returned after a user attempts to register, indicating the outcome of their registration attempt.

Login Request and Response

Login Request:
- email (string): The user's email address.
- password (string): The user's password.

This information is sent when a user wants to log into their account.

Login Response:
- status (string): The status of the login request, indicating success or failure.
- timestamp (number): The time when the response was generated.

- userWithToken (IUserResponse): An object containing user information along with an authentication token.

This information is returned after a user attempts to log in, indicating the outcome of their login attempt and providing the necessary authentication details if successful.

Technology Requirements

Backend:

- Languages & Frameworks: Python
- Database: PostgreSQL on Supabase
- Machine Learning: TensorFlow
- APIs: RESTful APIs

Frontend:

- Languages & Frameworks: JavaScript, Vue, PrimeVue, Tailwind CSS
- Visualization: Three.js.

DevOps:

- Version Control: Git, GitHub
- CI/CD: GitHub Actions
- Containerization: Docker
- Cloud Platforms: Vercel

Security:

- Auth/Access Control: OAuth 2.0, JWT
- Encryption: SSL/TLS

Collaboration:

- Communication: Discord, WhatsApp, Google Drive
- Project Management: GitHub Project Board
- Documentation: Google Docs.