1. Project Proposal

This section presents the project idea, problem statement, target users, and key features of the AutoImport KZ platform. The goal of the project is to design a system that addresses real-world challenges related to importing cars to Kazakhstan.

1.1 Project Overview

Project Title:
AutoImport KZ – Smart Car Import Platform for Kazakhstan

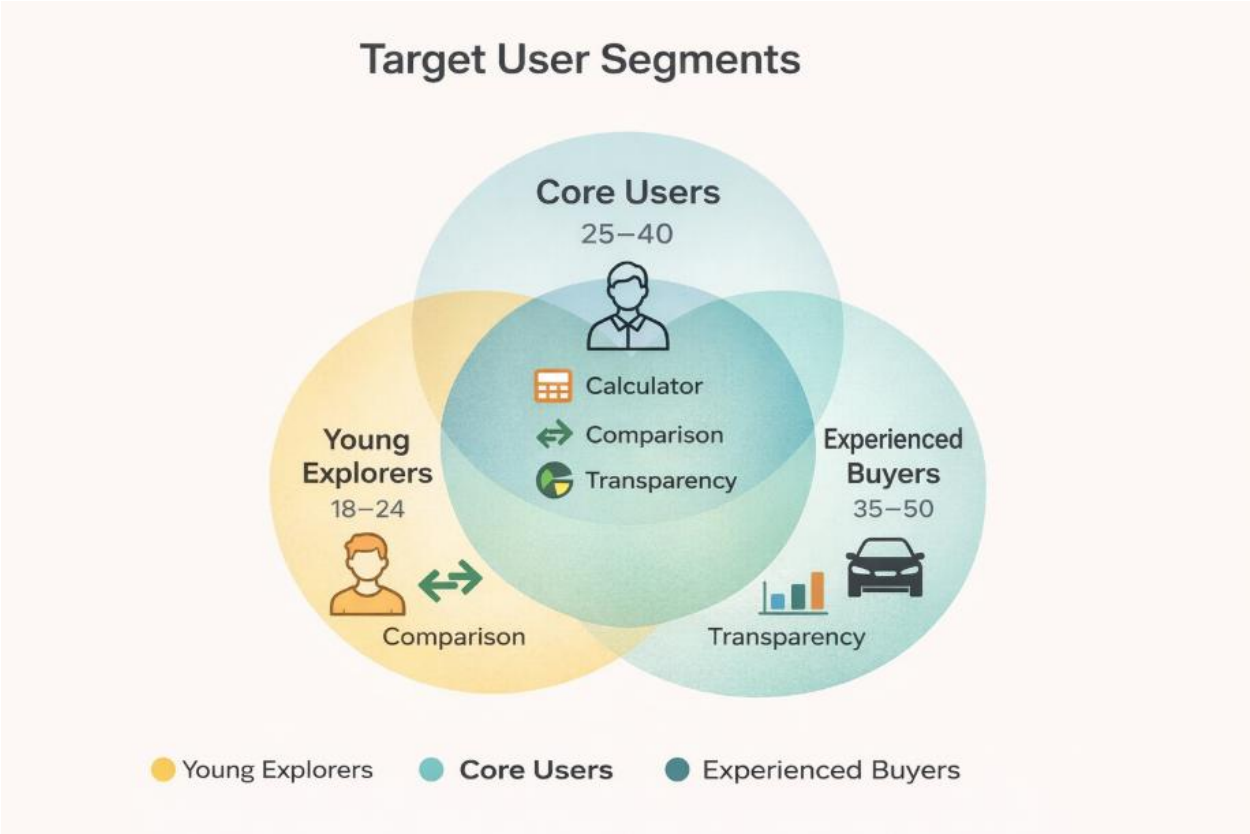
AutoImport KZ is a web-based platform designed to help users estimate the total cost of importing cars from foreign countries to Kazakhstan. Instead of focusing on direct car sales, the platform provides analytical tools such as import cost calculation, country comparison, and import recommendations. The system aims to make the car import process more transparent and understandable for users.

1.2 Problem Statement

In Kazakhstan, imported cars are often significantly more expensive due to customs duties, logistics costs, and additional fees. Many users are interested in purchasing cars from countries such as Japan, South Korea, the USA, or Europe, but face difficulties in calculating the final price and understanding import regulations. Existing platforms do not provide clear cost breakdowns or comparisons between countries, which leads to financial risks and poor decision-making.

1.3 Target Users

The platform is designed for users who are interested in importing cars and want to understand real import costs before making a decision.



1.4 Competitor Analysis

Existing car marketplaces mainly focus on local listings and advertisements. They lack tools for import cost calculation, country comparison, and warning systems. AutoImport KZ differs by providing analytical and decision-support features specifically for car import scenarios.

Comparative Analysis of Car Import Platforms

Criteria	OLX.kz	Kolesa.kz	Local Car Dealers	AutoImport KZ (Our Project)
Focus on car import	No	Partial	No	**Yes**
International cars	Limited	Limited	No	**Yes**
Import cost calculator	No	No	No	**Yes**
Customs & fees breakdown	No	No	No	**Yes**
Country comparison	No	No	No	**Yes**
Import warnings (engine volume, age)	No	No	No	**Yes**
Cost transparency	Low	Medium	Low	**High**
Import recommendations	No	No	No	**Yes**
User calculation history	No	No	No	**Yes**
Platform type	Marketplace	Marketplace	Offline	**Analytical platform**
Architecture transparency	No	No	No	**Yes (Go monolith)**

1.5 Planned Features

The system includes the following features:

- User registration and authentication
- International car catalog
- Import cost calculator
- Comparison of import costs between countries
- Import recommendations
- Warning system for expensive imports
- Saving calculation history

1.6 Project Scope

The project focuses on system design, architecture, and core business logic. Features such as real payment processing, external VIN verification, and live customs APIs are considered out of scope for the current milestone.

1.7 Proposal Summary

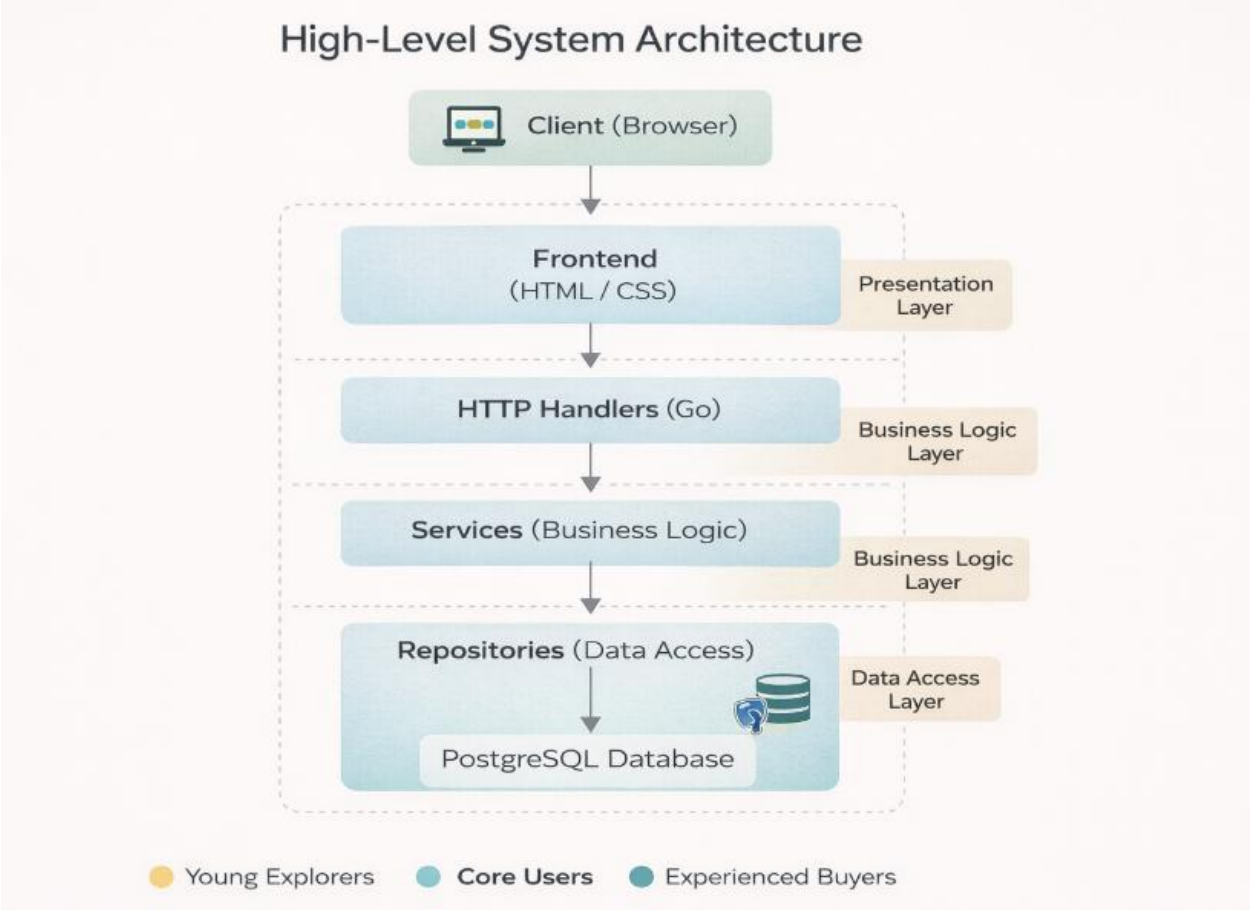
AutoImport KZ is a decision-support platform that helps users make informed choices when importing cars to Kazakhstan. The project demonstrates practical application of backend development, database design, and system architecture concepts.

2. Architecture & Design

This section describes the overall system architecture, main components, and design approach used in the AutoImport KZ project. The goal of the architecture is to ensure clear separation of responsibilities, simplicity of implementation, and support for core business logic such as import cost calculation, country comparison, and recommendation features.

2.1 System Architecture Overview

AutoImport KZ is developed using a **monolithic architecture**. The backend is implemented in **Go (Golang)** and exposes a **REST API** for client communication. The system follows a layered structure that separates presentation, business logic, and data access layers, which improves maintainability and clarity of the system design.



2.3 Modules and Responsibilities

Handlers

- Handle incoming HTTP requests and responses
- Route user actions to appropriate services
- Return JSON responses or render HTML pages

Services

- Implement core business logic
- Perform import cost calculations
- Handle country comparison, recommendations, and warnings

Repositories

- Communicate with the database
- Perform CRUD operations
- Isolate data access logic from business logic

Models

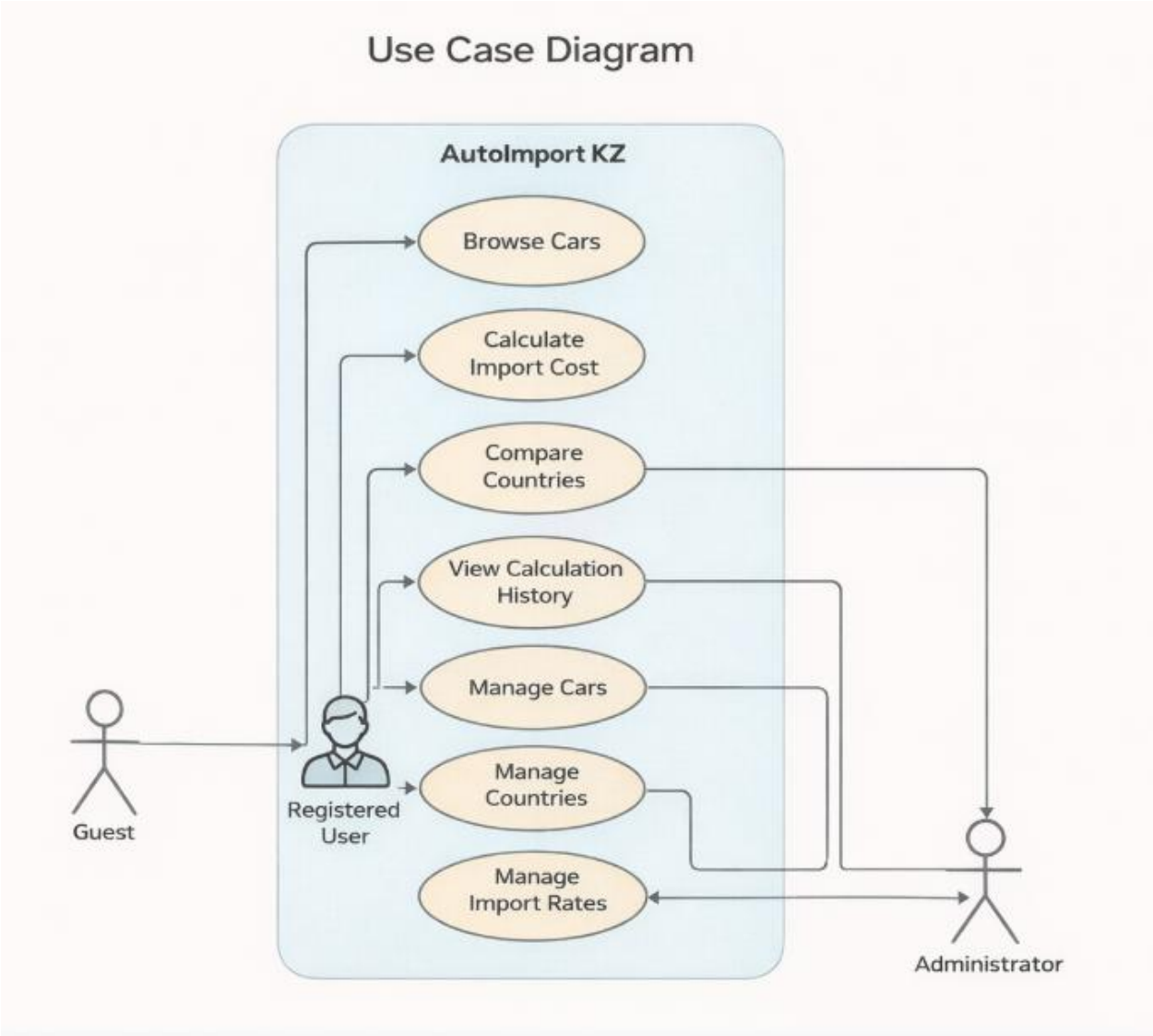
- Define domain entities (User, Car, Country, Calculation, Warning)

Frontend

- Provide user interface
- Collect user input for calculations
- Display system results and recommendations

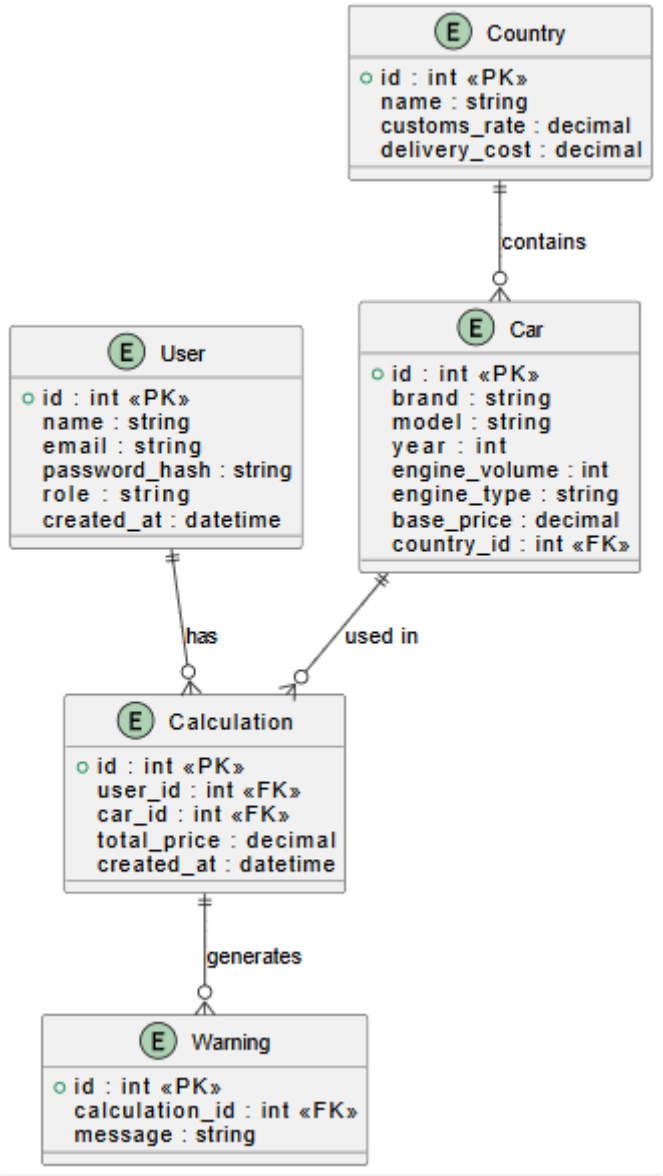
2.5 Use Case Overview

The system supports multiple user roles. Guests can browse available cars, registered users can calculate import costs and compare countries, and administrators can manage system data.



2.6 Database Design

The database design includes core entities such as users, cars, countries, calculations, and warnings. These entities support storing calculation history and generating recommendations and warnings.



2.5 ERD (Entity Relationship Diagram – Description)

The Entity Relationship Diagram (ERD) describes the main data entities of the AutoImport KZ system and the relationships between them. The database is designed to store user information, vehicle data, import calculations, and system-generated warnings.

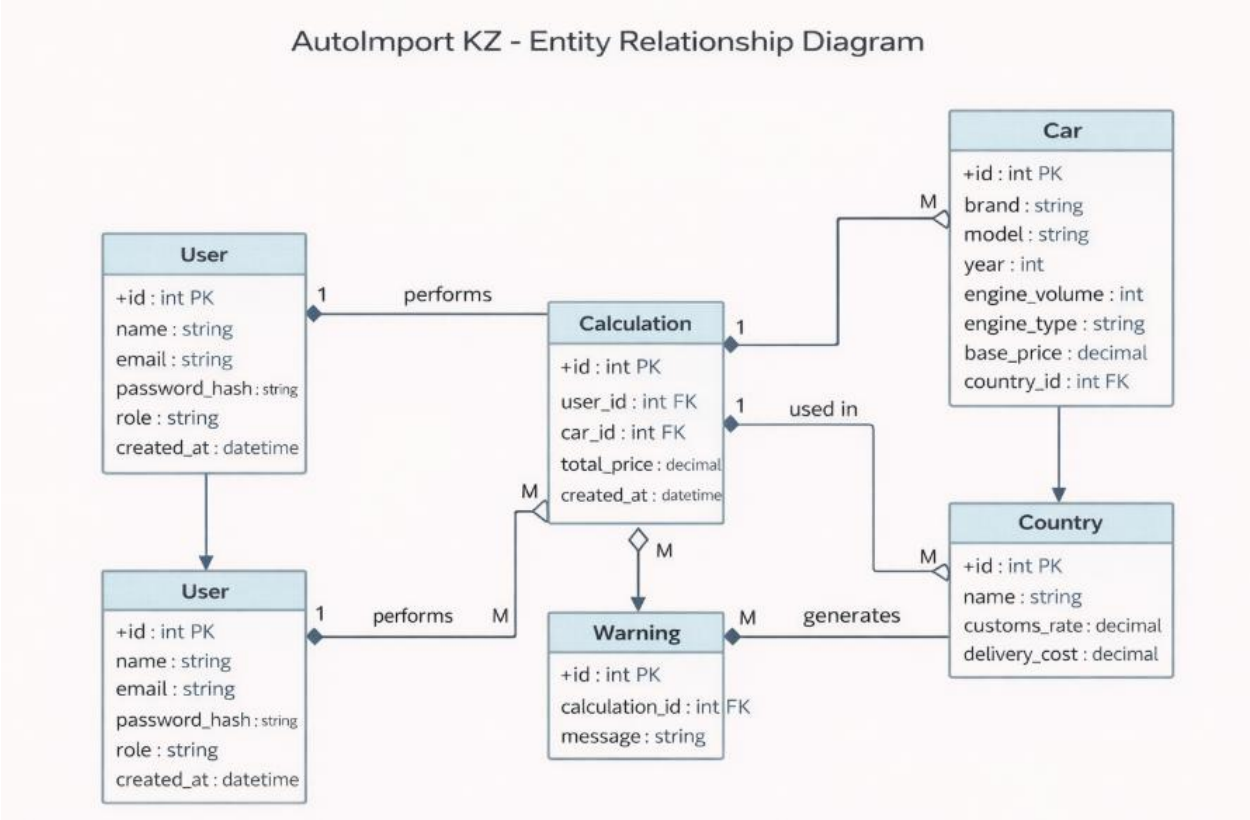
Entities

- **User**
- **Car**
- **Country**
- **Calculation**
- **Warning**

Relationships

- One **User** can have many **Calculations**
- One **Car** can be used in many **Calculations**
- One **Country** can be associated with many **Cars**
- One **Calculation** can generate multiple **Warnings**
- **Warning** is dependent on a specific **Calculation**

This structure allows the system to store calculation history, perform comparisons, and generate import warnings.

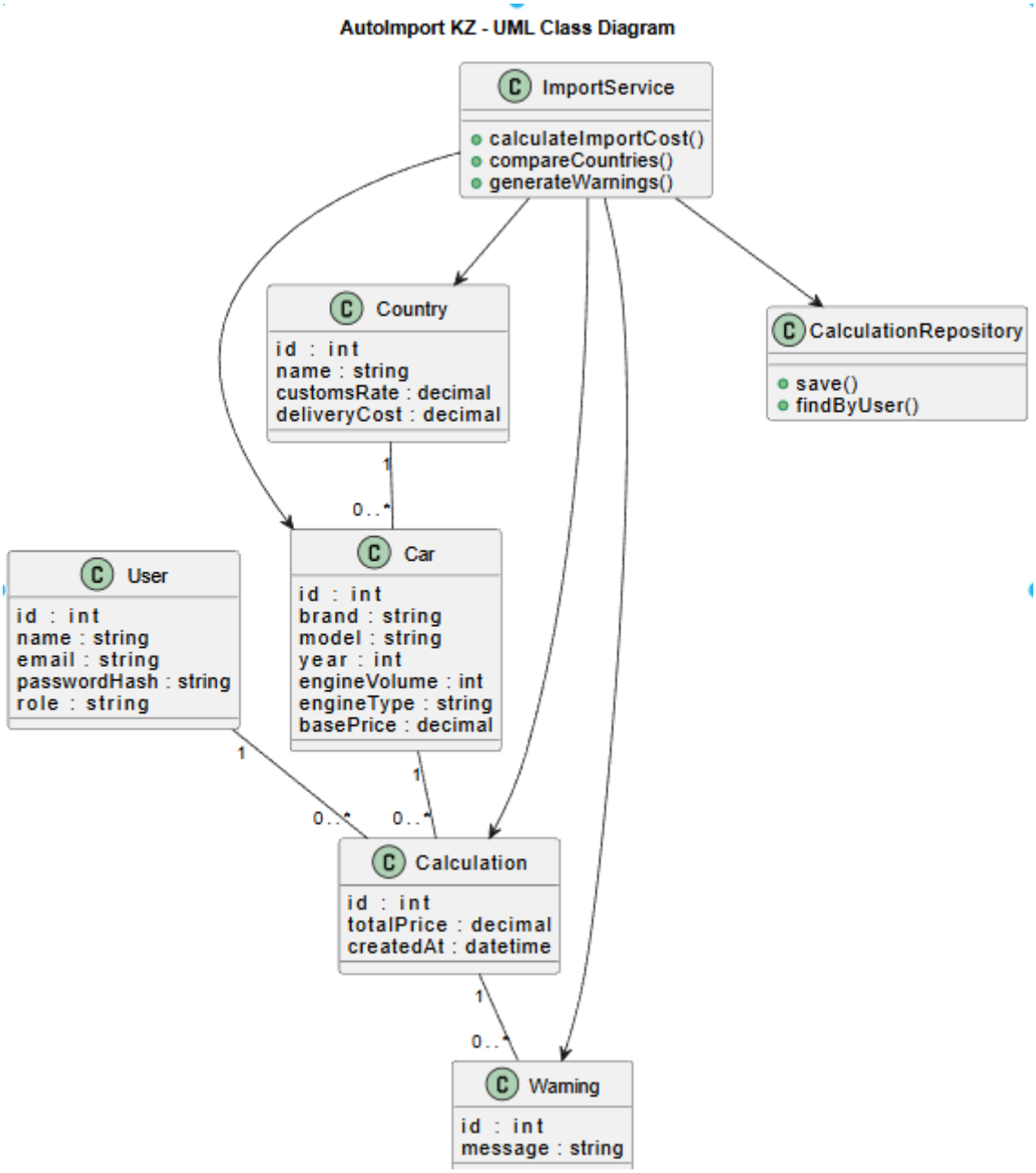


2.6 UML Diagram (Class-Level Description)

The UML class diagram represents the main domain classes of the AutoImport KZ system and their responsibilities.

- **User** contains personal and authentication data required for accessing the system
- **Car** represents a vehicle available for import, including technical parameters
- **Country** stores import-related parameters such as customs rates and delivery costs
- **Calculation** represents an import cost estimation performed by a user
- **Warning** stores system-generated alerts related to high import costs or risks

The classes interact through service layers that handle business logic and data access.



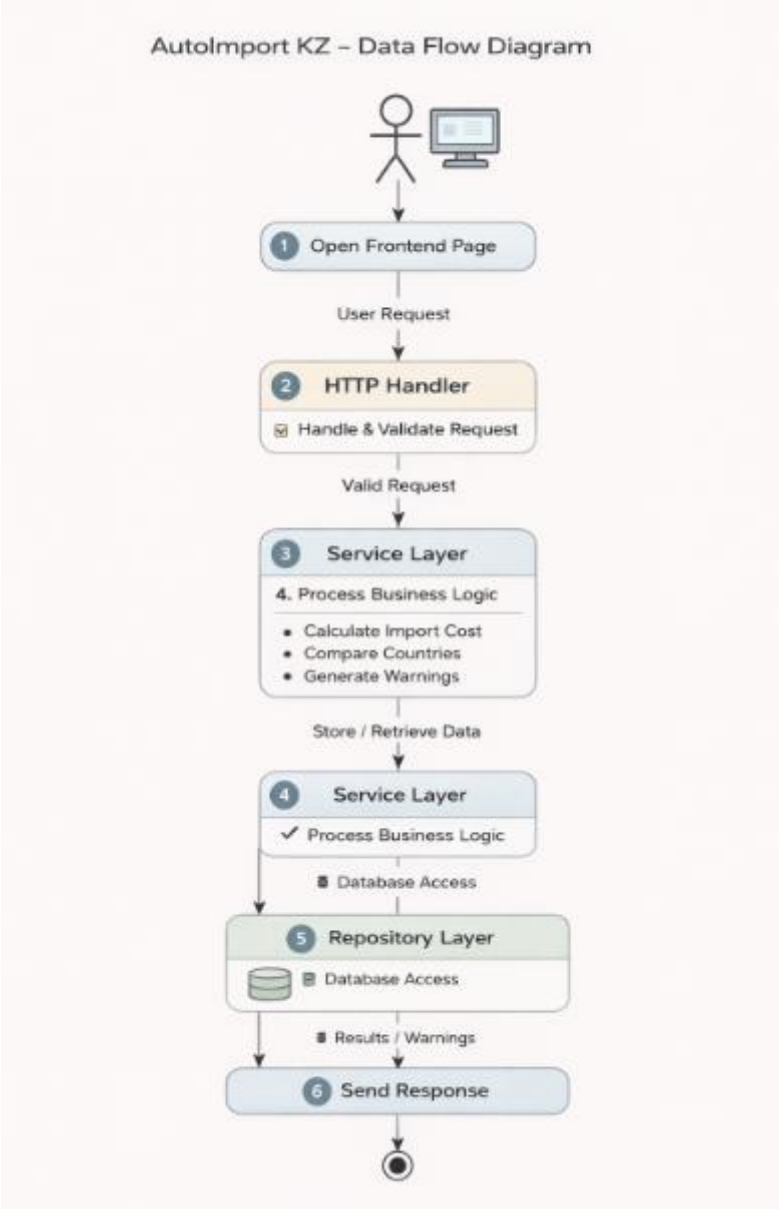
2.7 Data Flow Description

The data flow in the AutoImport KZ system follows a clear and structured sequence:

1. The user opens a frontend page in the browser
2. The user submits input data (car details and country selection)
3. The HTTP handler receives and validates the request
4. The service layer processes business logic such as cost calculation and comparison
5. The repository layer retrieves or stores data in the database

6. The system sends the calculated result, warnings, or recommendations back to the user

This flow ensures separation of responsibilities and reliable processing of user requests.



2.7 Summary

The proposed architecture ensures a clear and structured foundation for implementing the AutoImport KZ platform. The monolithic approach and modular design meet the project requirements and support future feature extensions.

---

3. Project Plan – Gantt

This section describes the planned development stages of the AutoImport KZ project for Weeks 7–10. The plan focuses on system design, architecture preparation, and core structural implementation. Final polishing and advanced features are intentionally excluded, as required by the assignment.

3.1 Week 7 – Project Planning and Analysis

During Week 7, the team focuses on defining the project foundation. The main goal is to finalize the project idea and clearly understand system requirements related to car import cost calculation and comparison.

Planned tasks:

- Finalization of the AutoImport KZ project concept
- Requirement analysis for import calculations and user roles
- High-level system architecture design
- Git repository creation and initial setup

3.2 Week 8 – Architecture and Design

Week 8 is dedicated to detailed system design and architectural planning. The team prepares all required diagrams and defines the overall system structure.

Planned tasks:

- Detailed architecture design
- Creation of ERD, UML, and Use Case diagrams
- Frontend structure and page flow planning
- Skeleton backend setup using Go (Golang)

3.3 Week 9 – Core Structure Development

In Week 9, the focus shifts to implementing the core project structure. The goal is not full functionality, but a clear and working skeleton of the system.

Planned tasks:

- Service and repository layer skeleton implementation

- Frontend page templates for main user interactions
- Planning integration between frontend and backend layers

3.4 Week 10 – Testing and Preparation

Week 10 is reserved for verification and preparation for project defense. Only basic testing and documentation tasks are included.

Planned tasks:

- Basic system testing
- Documentation preparation
- Defense preparation

Final polish tasks are intentionally not included, as required.

4. Git Repository Setup

One Git repository for the project:

https://github.com/daikipeek/DRT-CAR-STORE.git