**A PROJECT REPORT**

**on**

**“Vehicle Analyzer”**

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfilment of the Requirement for the Award of**

**BACHELOR’S DEGREE IN**

**COMPUTER SCIENCE**

**BY**

**Kumar Gaurav**

**Abhay Singh**

**2105125 4958855**

**2105172 4958001**

****

**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**May 2025**

**ABSTRACT**

The Vehicle Analyzer is a sophisticated web-based application designed to collect, process, and analyze vehicle performance data. This system enables detailed monitoring of fuel consumption patterns, engine performance metrics, and overall vehicle efficiency through an intuitive interface. Built on a robust Spring Boot backend with PostgreSQL database integration, the system offers comprehensive analytics dashboards and complete CRUD functionality for managing vehicle fleets and their associated performance data.

The application adopts the Model-View-Controller (MVC) architectural pattern, ensuring clear separation of concerns and maintainable codebase structure. The backend leverages Hibernate ORM for seamless database interactions, while the frontend employs HTML, CSS, and JavaScript to deliver responsive and interactive user experiences. This report provides a comprehensive examination of the system architecture, database design, API structure, implementation details, and potential future enhancements for the Vehicle Analyzer project.

Contents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Introduction | | |  |
| 2 | Technology Stack & Architecture | | |  |
|  | 2.1 | | Technology Stack Overview |  |
|  | 2.2 | | Architectural Overview |  |
|  | 2.3 | | System Integration Flow |  |
| 3 | DATABASE DESIGN & IMPLEMENTATION | | |  |
|  | 3.1 | | Database Schema |  |
|  | 3.2 | | Entity Relationships |  |
|  | 3.3 | | Database Design Diagram |  |
|  | 3.4 | | Database Design Considerations |  |
| 4 | BACKEND Implementation | | |  |
|  | 4.1 | | Project Structure |  |
|  | 4.2 | | Application Components |  |
|  | 4.3 | | API Documentation |  |
| 5 | Dashboard Implementation and Visualization Results | | |  |
|  | 5.1 | | Main Dashboard Overview |  |
|  | 5.2 | | Individual Vehicle Dashboard |  |
| 6. | Key Takeaways and Advantages | | |  |
|  | 6.1 | Data-Driven Decision Making | |  |
|  | 6.2 | Operational Efficiency Improvements | |  |
|  | 6.3 | Environmental Impact Reduction | |  |
|  | 6.4 | Scalability and Extensibility | |  |
|  | | | |  |

**Chapter 1**

**Introduction**

### **1.1 Project Scope & Objectives**

**Primary Objective:** Develop a web-based system capable of collecting, storing, and analyzing vehicle performance data to optimize fleet management and maintenance schedules.

**Key Project Goals:**

### **1.2 Target Users**

* Fleet managers seeking to optimize vehicle performance
* Maintenance personnel monitoring engine health indicators
* Financial analysts tracking fuel costs and consumption patterns
* Environmental compliance officers monitoring emissions data
* Individual vehicle owners interested in optimizing performance

### **1.3 Project Background**

The Vehicle Analyzer project emerged from the growing need for sophisticated data analytics in vehicle fleet management. Traditional approaches relied on manual record-keeping and basic spreadsheet analysis, while modern fleets generate substantial performance data that can yield valuable insights when properly analyzed. This project addresses this gap by providing a comprehensive platform for centralized vehicle data management with advanced analytical capabilities.

## **2. TECHNOLOGY STACK & ARCHITECTURE**

### **2.1 Technology Stack Overview**

The Vehicle Analyzer system employs a comprehensive technology stack designed for scalability, performance, and maintainability:

|  |  |  |
| --- | --- | --- |
| **Component** | **Technology** | **Purpose** |
| Backend Framework | Spring Boot 3.4.5 | Application infrastructure, dependency management, RESTful API development |
| Programming Language | Java 11 | Backend implementation |
| Database | PostgreSQL 14 | Persistent data storage |
| ORM Tool | Hibernate / Spring Data JPA | Object-relational mapping, database interaction |
| Frontend | HTML5, CSS3, JavaScript | User interface and dashboard visualization |
| API Documentation | Javadocs | Automated API documentation |
| Version Control | Git | Source code management |
| Build Tool | Maven | Dependency management, build automation |
| Development IDE | Visual Studio Code | Development environment |
| API Testing | Postman | API endpoint validation |

### **2.2 Architectural Overview**

The Vehicle Analyzer system follows the Model-View-Controller (MVC) architectural pattern, promoting separation of concerns and modularity:

**Model Layer:**

* Represents the application's data domain and business logic
* Includes entity classes for Vehicle, FuelData, and EngineData
* Implements data validation and business rules

**View Layer:**

* HTML templates and static resources for the frontend
* JavaScript for interactive dashboards and data visualization
* CSS stylesheets for visual presentation

**Controller Layer:**

* REST controllers handling HTTP requests and responses
* Request mapping and parameter validation
* Coordination between the frontend and service layer

**Additional Architectural Components:**

**Service Layer:**

* Contains business logic and transaction management
* Implements data processing and analysis algorithms
* Enforces security constraints and business rules

**Repository Layer:**

* Provides data access abstractions
* Implements database operations through JPA repositories
* Ensures efficient database interactions

**Configuration Layer:**

* Manages application properties and settings
* Configures database connections and security settings
* Defines bean definitions and application context

### **2.3 System Integration Flow**

The Vehicle Analyzer employs a three-tier architecture with clear separation between presentation, business logic, and data persistence layers. Data flows from client browsers through the Spring Boot backend and ultimately to the PostgreSQL database, with each tier handling specific responsibilities in the overall system.

User interactions with the dashboard interface trigger HTTP requests to the RESTful API endpoints defined in the controller layer. These controllers delegate processing to specialized service components which apply business rules and perform data analysis operations. The service layer interacts with the repository layer to retrieve or persist data as needed, with the repository layer handling all direct database operations through Hibernate/JPA abstractions.

This multilayered approach enhances maintainability, scalability, and security by isolating concerns and allowing independent evolution of different system components.

## **3. DATABASE DESIGN & IMPLEMENTATION**

### **3.1 Database Schema**

The Vehicle Analyzer database consists of three primary entities with defined relationships:

**Vehicle Entity:** The central entity representing individual vehicles with their specifications including manufacturer, model, vehicle type, fuel type, engine type, and year of manufacture.

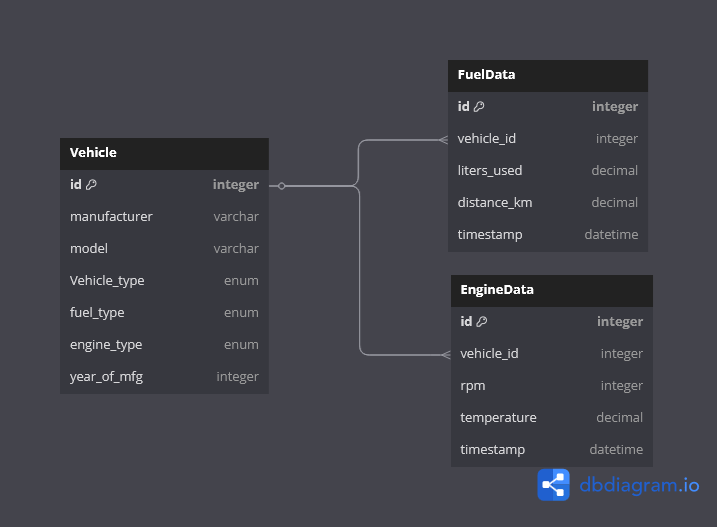
**FuelData Entity:** Records individual fuel consumption entries linked to specific vehicles, tracking liters used, distance traveled in kilometers, and timestamp of the entry.

**EngineData Entity:** Captures engine performance metrics linked to specific vehicles, including RPM measurements, temperature readings, and timestamp of the recording.

### **3.2 Entity Relationships**

* One Vehicle can have multiple FuelData entries (one-to-many relationship)
* One Vehicle can have multiple EngineData entries (one-to-many relationship)
* FuelData and EngineData are independent of each other but relate to the same Vehicle

### **3.3 Database Design Diagram**



### **3.4 Database Design Considerations**

**Performance Optimization:**

* Indices on frequently queried fields (vehicle\_id, timestamp)
* Appropriate data types for efficient storage and retrieval
* Optimized query patterns for analytical operations

**Data Integrity:**

* Foreign key constraints to maintain referential integrity
* Not-null constraints for required fields
* Cascade delete operations to maintain data consistency

**Scalability:**

* Normalized schema to minimize redundancy
* Timestamp fields for time-series analysis
* Separation of concerns between different data types

**Query Efficiency:**

* Structured for common analytical queries
* Support for aggregation functions
* Optimized for dashboard data retrieval

## **4. BACKEND IMPLEMENTATION**

### **4.1 Project Structure**

The backend code follows a structured package organization:

src ──main

├───java

│ └───com

│ └───Vehicleanalyzer

│ │ AnalyzerApplication.java

│ ├───Controller

│ │ EngineDataController.java

│ │ FuelDataController.java

│ │ VehicleController.java

│ ├───Models

│ │ │ DBDesign.png

│ │ │ EngineData.java

│ │ │ FuelData.java

│ │ │ Vehicle.java

│ │ └───enums

│ │ EngineType.java

│ │ FuelType.java

│ │ VehicleTypes.java

│ ├───Repository

│ │ EngineRepository.java

│ │ FuelDataRepository.java

│ │ VehicleRepository.java

│ └───Services

│ │ EngineDataService.java

│ │ FuelDataService.java

│ │ VehicleService.java

│ └───Impl

│ EngineDataServiceImp.java

│ FuelDataServiceImp.java

│ VehicleServiceImp.java

└───resources

│ application.properties

└───static

│ index.html

│ vehicle-details.html

│ vehicles.html

├───css

│ style.css

└───js

dashboard.js

vehicle-details.js

vehicles.js

### **4.2 Application Components**

#### ***4.2.1 Main Application Class***

The AnalyzerApplication class serves as the entry point for the Spring Boot application, initializing the Spring context and bootstrapping the application.

#### ***4.2.2 Controller Layer***

The controller layer handles incoming HTTP requests and delegates processing to the service layer. The Vehicle Analyzer system includes three main controllers:

**VehicleController:**

* Manages all operations related to vehicle entities
* Handles CRUD operations for vehicles
* Maps to the base URL path /api/vehicles

**FuelDataController:**

* Manages fuel consumption data associated with vehicles
* Provides both CRUD operations and analytical endpoints
* Maps to the URL path /api/vehicles/{id}/fuel-data
* Includes specialized endpoints for fuel analysis metrics

**EngineDataController:**

* Manages engine performance data associated with vehicles
* Provides both CRUD operations and analytical endpoints
* Maps to the URL path /api/vehicles/{id}/engine-data
* Includes specialized endpoints for engine performance metrics

#### ***4.2.3 Service Layer***

The service layer contains business logic for processing and analyzing vehicle data:

**VehicleService:**

* Core business logic for vehicle management
* Validation of vehicle data
* Implementation of vehicle lifecycle operations

**FuelDataService:**

* Management of fuel consumption records
* Calculation of fuel efficiency metrics
* Analysis of consumption patterns over time
* Implementation of aggregation operations for dashboards

**EngineDataService:**

* Management of engine performance metrics
* Analysis of engine behavior patterns
* Calculation of performance indicators
* Implementation of statistical operations on engine data

#### ***4.2.4 Repository Layer***

The repository layer provides data access abstractions:

**VehicleRepository:**

* Data access operations for vehicle entities
* Custom query methods for vehicle retrieval

**FuelDataRepository:**

* Data access operations for fuel consumption records
* Custom query methods for analytical operations

**EngineDataRepository:**

* Data access operations for engine performance records
* Custom query methods for statistical analysis

#### ***4.2.5 Model Layer***

The model layer defines the core domain entities:

**Vehicle:**

* Represents a vehicle with its specifications
* Contains manufacturer, model, vehicle type, fuel type, engine type, and year of manufacture

**FuelData:**

* Represents a fuel consumption record
* Contains liters used, distance traveled, and timestamp

**EngineData:**

* Represents an engine performance record
* Contains RPM, temperature, and timestamp

#### ***4.2.6 Exception Handling***

The application includes centralized exception handling:

**ResourceNotFoundException:**

* Custom exception for when requested resources are not found
* Used across all controllers to provide consistent error responses

**GlobalExceptionHandler:**

* Centralized exception handling component
* Provides consistent error responses across the application
* Maps exceptions to appropriate HTTP status codes

### **4.3 API Documentation**

The Vehicle Analyzer system provides a comprehensive set of RESTful API endpoints for managing vehicles and their associated data:

#### ***4.3.1 Vehicle APIs***

**Base Path: /api/vehicles**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /api/vehicles | Retrieve all vehicles |
| GET | /api/vehicles/{id} | Retrieve a specific vehicle by ID |
| POST | /api/vehicles | Create a new vehicle |
| PUT | /api/vehicles/{id} | Update an existing vehicle |
| DELETE | /api/vehicles/{id} | Delete a vehicle |

#### **4.3.2 Fuel Data APIs**

**Base Path: /api/vehicles/{id}/fuel-data**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /api/vehicles/{id}/fuel-data | Retrieve all fuel data for a vehicle |
| GET | /api/vehicles/{id}/fuel-data/{fuel\_id} | Retrieve specific fuel data entry |
| GET | /api/vehicles/{id}/fuel-data/totalfuel | Get total fuel usage for a vehicle |
| GET | /api/vehicles/{id}/fuel-data/totaldistance | Get total distance for a vehicle |
| GET | /api/vehicles/{id}/fuel-data/fuelefficiency | Calculate fuel efficiency for a vehicle |
| POST | /api/vehicles/{id}/fuel-data | Create new fuel data for a vehicle |
| PUT | /api/vehicles/{id}/fuel-data/{fuel\_id} | Update fuel data entry |
| DELETE | /api/vehicle/{id}/fuel-data/{fuel\_id} | Delete Fuel data entry |

**4.3.2 Engine Data APIs**

**Base Path: /api/vehicles/{id}/engine-data**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| GET | /api/vehicles/{id}/engine-data | Retrieve all engine data for a vehicle |
| GET | /api/vehicles/{id}/engine-data/{engine\_id} | Retrieve specific engine data entry |
| GET | /api/vehicles/{id}/engine-data/Avg\_RPM | Get Average RPM engine usage for a vehicle |
| GET | /api/vehicles/{id}/fuel-data/Avg\_Temp | Get Average Temperature for a vehicle |
| POST | /api/vehicles/{id}/engine-data | Create new fuel data for a vehicle |
| PUT | /api/vehicles/{id}/engine-data/{engine\_id} | Update fuel data entry |
| DELETE | /api/vehicle/{id}/engine-data/{engine\_id} | Delete Engine entry |

## **5. Dashboard Implementation and Visualization Results**

The Vehicle Analyzer system provides comprehensive dashboards that transform raw vehicle data into actionable insights. These dashboards serve as the primary interface for users to interact with the system's analytical capabilities, offering both aggregate and individual vehicle analysis.

### **5.1 Main Dashboard Overview**

The main dashboard presents a high-level overview of the entire vehicle fleet, serving as the central hub for system-wide analytics. This dashboard is designed with intuitive navigation and clear visual hierarchy to ensure users can quickly access the information they need.

#### ***5.1.1 Vehicle Type Distribution***

The vehicle type distribution visualization presents the composition of the vehicle fleet categorized by vehicle types (sedan, SUV, truck, van, etc.). This visualization employs a dynamic pie chart with the following features:

* Interactive segments that display exact counts on hover
* Color-coded segments for immediate visual differentiation
* Legend of each vehicle type

This visualization enables fleet managers to understand the diversity of their vehicle portfolio at a glance and make informed decisions about fleet composition and future acquisitions.

#### ***5.1.2 Engine Type Distribution***

The engine type distribution visualization categorizes vehicles by their engine configurations (gasoline, diesel, hybrid, electric, etc.). Implemented as an interactive donut chart, this visualization offers:

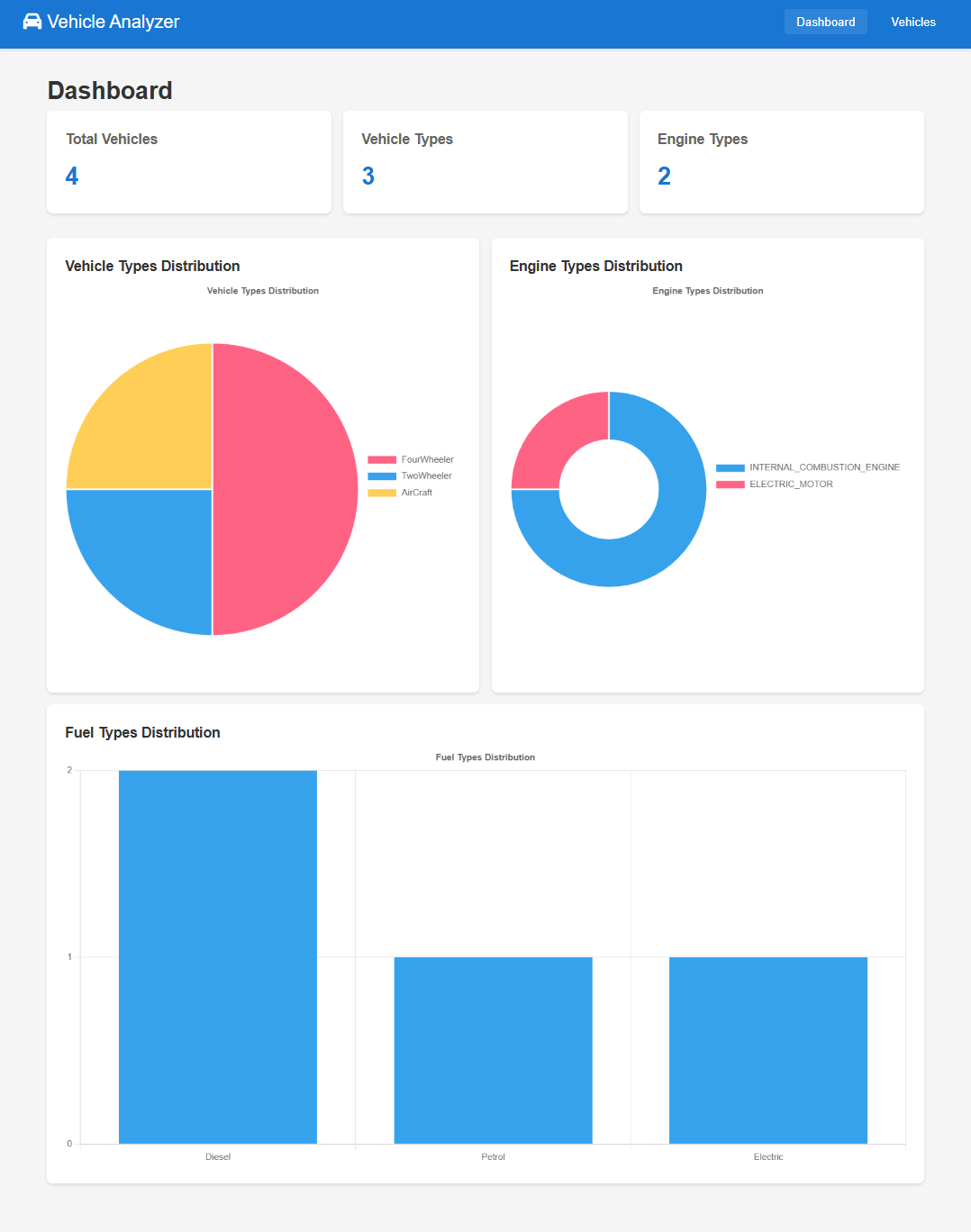
* Segments representing each engine type
* Length proportional to the count of vehicles
* Display exact counts indicating the proportion of each engine type
* Color-coding aligned with organizational standards for engine classification

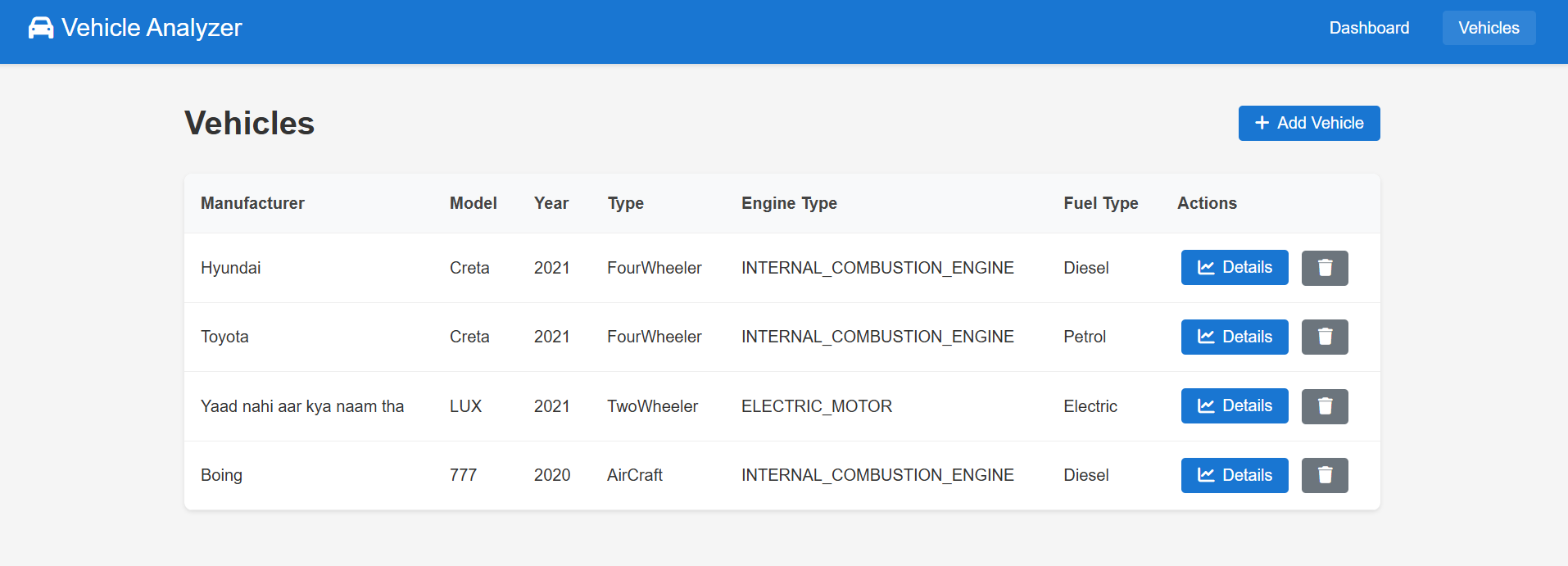
This visualization provides critical insights into the technological distribution of the fleet, supporting strategic decisions about maintenance resources, fuel procurement, and environmental impact assessment.

### **5.1.3 Aggregate Fuel Analysis**

The aggregate fuel analysis dashboard component provides a high-level overview of fuel-related metrics across the entire vehicle fleet:

* **Fuel Type Distribution**: A simple bar chart that categorizes and displays the number of vehicles based on their fuel type (e.g., Petrol, Diesel, Electric).
* **Monthly Fuel Consumption Trend** *(optional)*: *(To be included only if implemented)* Shows total liters consumed across all vehicles over time.
* **Average Fuel Efficiency by Vehicle Type** *(optional)*: *(To be included only if implemented)* Represents how different vehicle types compare in fuel efficiency (e.g., km/L).
* **Cost Analysis Panel** *(optional)*: *(To be included only if implemented)* Displays key cost indicators such as total fuel cost, cost per km, and projected fuel expenditure.





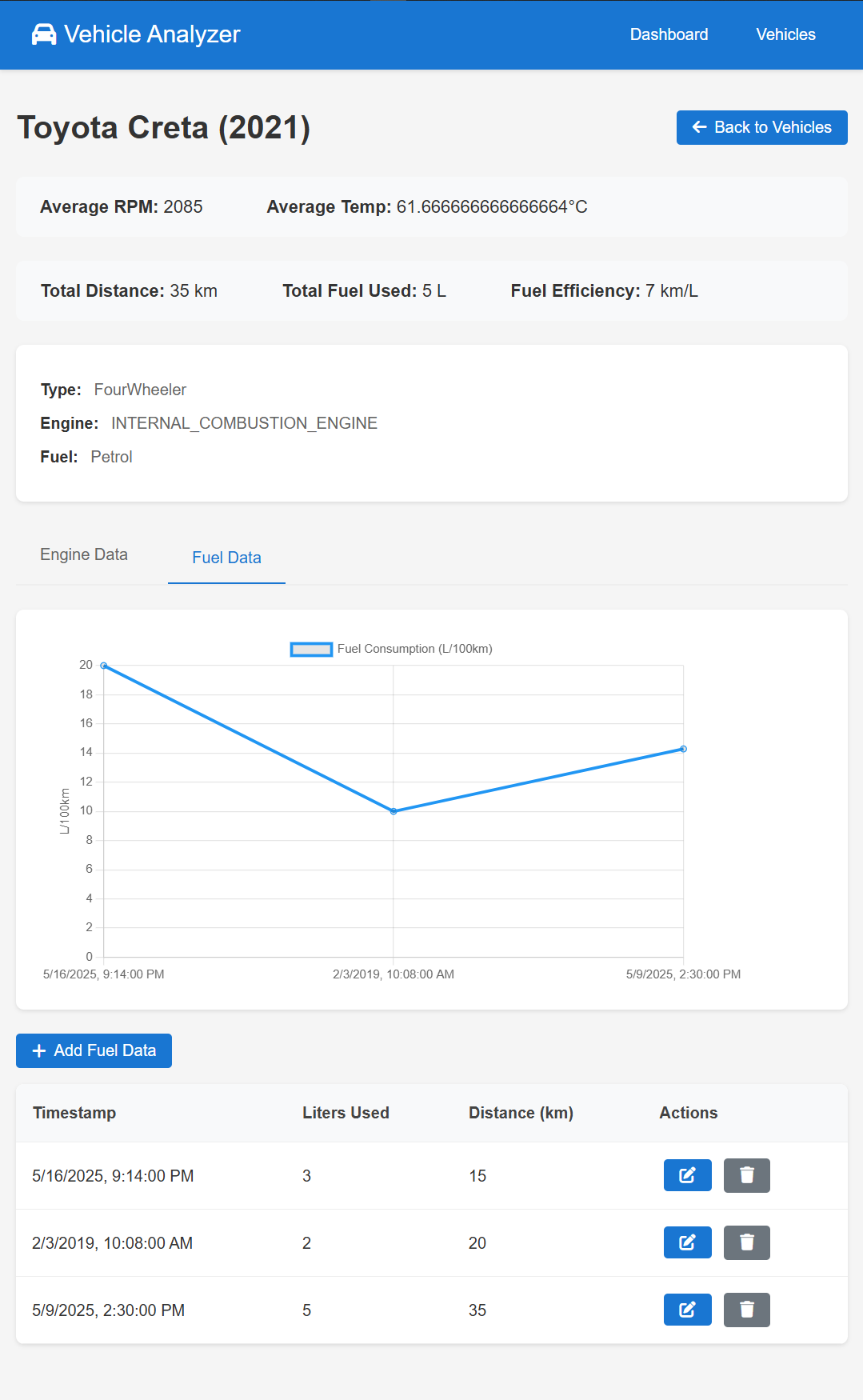
### **5.2 Individual Vehicle Dashboard**

The individual vehicle dashboard provides detailed analysis for each specific vehicle in the system, accessed through the vehicle listing or search functionality. This dashboard combines historical data with real-time analytics to offer comprehensive performance monitoring.

#### ***8.2.1 Vehicle Overview Panel***

The vehicle overview panel serves as a digital passport for each vehicle, displaying:

* Vehicle specifications (manufacturer, model, year, type, engine type)
* Performance summary (average fuel efficiency, total distance traveled, maintenance status)



#### ***5.2.2 Fuel Consumption Analysis***

The fuel consumption analysis for individual vehicles offers deep insights into fuel usage patterns through several integrated visualizations:

* **Consumption vs. Distance**: A scatter plot correlating fuel consumption with distance traveled for each refueling event

These visualizations help identify factors affecting fuel efficiency, such as driving conditions, maintenance needs, or seasonal variations.

#### ***5.2.3 Engine Performance Metrics***

The engine performance visualization provides critical insights into engine health and performance through:

* **RPM Analysis**: A time-series chart showing RPM patterns during operation, with threshold indicators for optimal operating ranges
* **Temperature Monitoring**: A heat map visualization showing engine temperature patterns over time, with color-coded zones indicating normal and concerning temperature ranges

enables proactive maintenance and operation optimization, potentially extending engine life and reducing repair costs.

## **6. Key Takeaways and Advantages**

### **6.1 Data-Driven Decision Making**

The Vehicle Analyzer system transforms vehicle management from a reactive to a proactive discipline through:

* **Evidence-Based Fleet Management**: Replacing intuition-based decisions with data-driven insights derived from comprehensive vehicle performance analytics
* **Predictive Maintenance Capabilities**: Leveraging engine performance data to anticipate maintenance needs before failures occur
* **Resource Optimization**: Enabling precise allocation of resources based on actual usage patterns and performance metrics
* **Trend Identification**: Surfacing long-term patterns that may not be apparent through manual monitoring

This data-driven approach minimizes guesswork and maximizes the efficacy of management decisions, resulting in measurable improvements in fleet performance and cost-effectiveness.

### **6.2 Operational Efficiency Improvements**

The system drives significant operational efficiency improvements across multiple dimensions:

* **Fuel Economy Enhancement**: Analysis tools identify opportunities for fuel efficiency improvements through vehicle selection, route optimization, and driving behavior modification
* **Maintenance Cost Reduction**: Proactive maintenance based on actual engine performance reduces both scheduled and emergency maintenance costs
* **Vehicle Lifecycle Optimization**: Data-driven insights into vehicle performance over time enable optimal replacement timing
* **Administrative Efficiency**: Automated data collection and analysis reduce the administrative burden of fleet management

These efficiency improvements translate directly to reduced operational costs and improved resource utilization.

### **6.3 Environmental Impact Reduction**

The Vehicle Analyzer system supports environmental responsibility through:

* **Emissions Monitoring**: Indirect tracking of emissions through fuel consumption analysis
* **Efficiency Improvement**: Tools to identify and address inefficient vehicle operation that increases environmental impact
* **Fleet Composition Optimization**: Data to support strategic decisions about integrating more fuel-efficient or alternative-fuel vehicles
* **Environmental Reporting**: Metrics that can be used for environmental compliance and corporate sustainability reporting

These environmental benefits align fleet operations with broader sustainability goals while often generating cost savings through improved efficiency.

### **6.4 Scalability and Extensibility**

The Vehicle Analyzer system is designed for growth and adaptation:

* **Scalable Architecture**: The system architecture supports growth from small fleets to enterprise-scale deployments
* **Extensible Data Model**: The core data model can accommodate additional vehicle metrics and parameters as needs evolve
* **API-First Design**: RESTful APIs enable integration with other systems and development of complementary applications
* **Customizable Dashboards**: Visual analytics can be tailored to specific organizational roles and priorities

This flexibility ensures that the system can evolve alongside organizational needs and technological advancements.