Software Engineering Project:

Car Dealership Management System

Made by: Tristan Beason, Prohor Muchev, Jovan Tone & Erol Balkan

Mentors: Goce Gavrilov & Viktor Denkovski

Date: 28.04.2024

Table of Contents:

Introduction:

-The Car Dealership Management System (CDMS) is a comprehensive software solution designed to streamline vehicle inventory management processes within a car dealership. The system aims to improve vehicle inventory management, streamline sales processes and optimize service operations. In this project document, we delve into the key features, functionalities and benefits of our system, highlighting its role in enhancing dealership efficiency, customer satisfaction and business performance.

Goals:

-The primary objective of CDMS is to revolutionize the way automotive dealerships manage their inventory, interact with customers, track sales performance and maintain service operations. By providing a centralized platform with robust features and intuitive interfaces, CDMS aims to empower dealership staff with the tools and insights needed to drive success in a competitive market landscape.

User Personas:

-To ensure that CDMS meets the diverse needs of dealerships stakeholders, we have identified several user personas representing different roles within the organization. These personas include:

- Sales Managers: Responsible for overseeing sales operations, tracking performance metrics and analyzing sales trends.
- Service Technicians: Tasked with performing vehicle maintenance, inspections and repairs to ensure optimal performance and customer satisfaction.
- Finance Managers: Involved in managing financial transactions, processing sales contacts and facilitating financing options for customers.
- Customers: Individuals interested in purchasing or leasing vehicles from the dealership, seeking information, scheduling test drives and making inquiries about available inventory.

-By understanding the unique perspectives, requirements and preferences for each user persona, CDMS can be tailored to deliver a personalized and seamless user experience.

Functional Requirements:

-CDMS encompasses a wide range of functional requirements designed to address the diverse needs and priorities of dealership stakeholders. These requirements are documented in the form of use cases and user stories, providing a clear understanding of the system's capabilities and functionalities. Examples of functional requirements include (these are also our key-features):

1. Inventory Management: The system provides real-time visibility into the dealership's vehicle inventory, allowing users to view, update and track vehicle details such as model, brand, price and specifications like license plate numbers. Its functions include: adding, updating and deleting vehicle listings, categorizing vehicles based on attributes and facilitating search and retrieval.

2. User Access Control: Role-based access control ensures that users only have access to the information and functionalities relevant to their roles within the dealership, such as: sales managers, finance managers, service technicians and customers.

3. Sales Tracking: Sales managers can monitor sales performance, track vehicle sales and generate sales reports to analyze trends and identify opportunities for improvement.

4. Customer Interaction: Customers can browse the dealership's inventory online, view detailed vehicle information, schedule test drives and make online inquiries about vehicle availability and pricing. These features enable communication with the dealership staff.

5. Maintenance Management: Service technicians can access vehicle service history and can create maintenance schedules to efficiently perform vehicle inspections, maintenance and repairs.

-By prioritizing these functional requirements and aligning them with dealership objectives, CDMS aims to deliver tangible value and measurable outcomes for dealership operations.

Quality Requirements:

-Quality attributes play a crucial role in determining the effectiveness, reliability and usability of CDMS. To ensure that the system needs the highest standards of quality, we have identified and prioritized the following quality attributes:

- Reliability: Ensuring that CDMS operates consistently and accurately, minimizing downtime and disruptions to dealership operations.
- Usability: Designing CDMS with intuitive interfaces and navigation paths, enabling users to perform tasks efficiently and effectively.
- Performance: Optimizing system performance to support high volumes of concurrent users, rapid data processing and real-time updates.
- Security: Implementing robust security measures to protect sensitive dealership data, prevent unauthorized access and mitigate cybersecurity threats.
- Scalability: Designing CDMS to scale seamlessly with growing dealership operations, accommodating increased user demand and expanding inventory sizes.

-By focusing on these quality attributes and incorporating best practices in software development and quality assurance, CDMS aims to exceed user expectations and deliver superior user experience.

The Code:

Java Spring Boot Application Overview:

-The Vehicle Application is developed in IntelliJ using Java Spring Boot which is a popular framework for building robust and scalable Java applications. Spring Boot simplifies the development process by providing pre-configured dependencies and auto-configuration capabilities, allowing developers to focus on application logic rather than infrastructure setup, Key components of the Spring Boot application include controllers, services, repositories and configuration classes.

Java Components:

1. Main Vehicle Class: The ‘Vehicle’ class represents the entity model for vehicles in the application. It is annotated with ‘@Entity’ to indicate that it is a JPA entity, and it defines attributes such as mode, brand, color, price and license plate. Additionally, the class includes methods for generating a random license plate and year for each vehicle, This class serves as the foundation for storing and manipulating vehicle data within the application.

2. Controllers: The ‘VehicleController’ class defines RESTful endpoints for handling HTTP requests related to vehicle operations, such as retrieving all vehicles, adding new vehicles, updating existing vehicles and deleting vehicles. These endpoints are mapped to specific URL paths and HTTP methods using annotations such as ‘@RestController’, ’@GetMapping’, ‘@PostMapping’, ‘@PutMapping’, ‘@PatchMapping’ and ‘@DeleteMapping’.

3. Services: The ‘VehicleService’ class contains business logic for performing various operations on vehicles, such as retrieving vehicles from the database, adding new vehicles, updating existing vehicles and deleting vehicles. This class encapsulates the interaction with the ‘VehicleRepository’ and implements functionalities such as error handling and data validation.

4. Repositories: The ‘VehicleRepository’ interface extends the ‘JpaRepository’ interface provided by Spring Data JPA and defines methods for querying and manipulating vehicle data in the underlying database. It inherits common CRUD (Create, Read, Update and Delete) operations such as ‘findAll’, ‘findById’, ‘findByColor’, ‘save’, ‘update’ and ‘delete’, eliminating the need for boilerplate SQL queries.

5. Configuration Classes: The ‘VehicleConfig’ class serves as a configuration class for the Vehicle module, defining bean configurations and initialization logic using Spring’s ‘@Configuration’ and ‘@Bean’ annotations. It initializes sample vehicle data during application startup using the ‘CommandLineRunner’ interface, populating the database with predefined vehicle entities.

6. Application Properties: The ‘application.properties’ file contains the configuration properties for the Spring Boot application.

7. Commands HTTP File: The ‘COMMANDS.http’ file contains the HTTP requests that can be executed. It also contains sample requests for interacting with the RESTful endpoints of the application. We can use these commands to test out the functions of our CDMS project.

Database:

-The application uses an H2 in-memory database, configured via Spring Boot’s default database properties in the ‘application.properties’ file. The H2 database provides a lightweight, embedded database solution that simplifies development and testing without requiring an external database setup. The database schema is automatically created based on the entity mappings defined in the ‘Vehicle’ class, allowing seamless persistence of vehicle data.

Dependency Management:

-The project utilizes Apache Maven as a build automation tool and dependency management system. The ‘pom.xml’ file defines the project metadata, dependencies, plugins and build configurations.

Conclusion:

- In summary, the development of our Car Dealership Management System using Java Spring Boot has been a meticulous process aimed at delivering a comprehensive solution for modern car dealerships. Leveraging the robustness of Spring Boot and the efficiency of Spring Data JPA, we have crafted a scalable and efficient application architecture. The inclusion of user personas, functional requirements, and quality attributes in our documentation has provided a clear roadmap for development, ensuring alignment with stakeholder expectations. Through diligent coding practices and adherence to industry standards, we have created a system that not only manages inventory and facilitates sales but also prioritizes user experience and system performance. Moving forward, we are excited about the potential for further enhancements and expansions, driving continued innovation in the automotive industry.

Citations:

1. Spring Boot Official Documentation: https://spring.io/projects/spring-boot

2. Spring Data JPA Reference Documentation: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

3. Baeldung - Spring Boot Tutorials: https://www.baeldung.com/spring-boot

4. Spring Framework Reference Documentation: https://docs.spring.io/spring-framework/docs/current/reference/html/