Software Design Document (SDD) for Car Services

TEAM-36

SRINIVAS SIRIGIRI

SE22UARI167

Project Title: A website for car service, car resale and car insurance.

1. Introduction

The Car Services Website is designed to provide users with a platform to manage car-related services, including car servicing, resale, and insurance. The system is built using HTML/CSS for the frontend, Java Servlets for the backend, and a local SQL Server for database management. It does not include a user login system but uses a username to track user history.

1.1 Purpose

This document serves as a comprehensive guide for designing, developing, and maintaining the Car Services Website. It outlines the system's architecture, components, and design considerations.

1.2 Scope

This document applies to all aspects of the Car Services Website, including the frontend, backend, database, and interactions between them.

1.3 Definitions, Acronyms, and Abbreviations

- **SDD:** Software Design Document
- **HTML:** HyperText Markup Language
- **CSS:** Cascading Style Sheets
- **JSON:** JavaScript Object Notation

1.4 References

Project-Software-Design-Specification.docx

2. Use Case View

The Car Services Website includes the following key use cases:

- **Book Car Service:** Users can book car service appointments.
- **List Car for Resale:** Users can list cars for sale.
- **Inquire about Car Insurance:** Users can inquire about car insurance policies.

2.1 Use Case: Book Car Service

- **Description:** Users can book car service appointments by providing their username, car details, and preferred service date.
- **Usage Steps:**
    1. User navigates to the "Book Car Service" page.
    2. User fills out the car service form.
    3. User submits the form.
    4. The server validates the input and stores the data in the database.

3. Design Overview

The Car Services Website follows a client-server architecture.

3.1 Design Goals and Constraints

- **Goal:** Provide a user-friendly platform for managing car-related services.
- **Constraint:** The system must use HTML/CSS for the frontend, Java Servlets for the backend, and a local SQL Server for database management.

3.2 Design Assumptions

- The server is running on the local machine.
- The database is properly configured and running.

3.3 Significant Design Packages

The system is divided into the following packages:

- **Frontend:** Contains HTML, CSS, and JavaScript files.
- **Backend:** Contains Java Servlets and database interaction logic.
- **Database:** Contains SQL Server database schema and data.

3.4 Dependent External Interfaces

The system depends on the following external interfaces:

- SQL Server database
- Web browser (client)

4. Logical View

The logical view describes the detailed design of the system.

4.1 Design Model

The system comprises the following classes:

- **User:** Represents a user with a username.
- **ServiceBooking:** Represents a car service booking.
- **CarListing:** Represents a car listed for resale.
- **InsuranceInquiry:** Represents an insurance inquiry.

4.2 Use Case Realization

- **Book Car Service Use Case:**
    1. User fills out the car service form on the frontend.
    2. JavaScript sends a POST request to the server.
    3. The `ServiceBookingServlet` receives the request, validates the input, and interacts with the database to store the data.

5. Data View

The data view describes the persistent data storage perspective of the system.

5.1 Domain Model

The domain model consists of entities such as User, ServiceBooking, CarListing, and InsuranceInquiry.

5.2 Data Model (Persistent Data View)

5.2.1 Data Dictionary

- **Users Table:**
    - Username (VARCHAR, Primary Key)
- **ServiceBookings Table:**
    - BookingID (INT, Primary Key)
    - Username (VARCHAR, Foreign Key referencing Users)
    - CarDetails (VARCHAR)
    - ServiceDate (DATE)
- **CarListings Table:**
    - ListingID (INT, Primary Key)
    - Username (VARCHAR, Foreign Key referencing Users)
    - CarMake (VARCHAR)
    - CarModel (VARCHAR)
    - Price (DECIMAL)
- **InsuranceInquiries Table:**
    - InquiryID (INT, Primary Key)
    - Username (VARCHAR, Foreign Key referencing Users)
    - CarMake (VARCHAR)
    - CarModel (VARCHAR)
    - InquiryDate (DATE)

6. Exception Handling

- **DatabaseConnectionException:** Thrown when a database connection cannot be established.
- **InvalidInputException:** Thrown when user input is invalid.

7. Configurable Parameters

- **Database URL:** The URL for connecting to the SQL Server database.
- **Database Username:** The username for accessing the database.
- **Database Password:** The password for accessing the database.

8. Quality of Service

8.1 Availability

The system should be available 24/7, with minimal downtime for maintenance.

8.2 Security and Authorization

Ensure that user data is securely stored and protected from unauthorized access.

8.3 Load and Performance Implications

The system should be able to handle a large number of concurrent users without performance degradation (Currently not possible).

8.4 Monitoring and Control

Implement monitoring and logging to track system performance and errors.

9. Testing Strategy

- **Unit Testing:** Test individual servlets and database interactions.
- **Integration Testing:** Test the interaction between the frontend and backend.
- **User Acceptance Testing (UAT):** Conduct UAT to ensure the system meets user requirements.

10. Deployment Strategy

1. Set up the server environment (e.g., Tomcat).
2. Configure the database.
3. Deploy the application.

11. Maintenance and Updates

1. Regularly back up the database.
2. Apply security patches and updates.
3. Plan for future enhancements based on user feedback.

12. Glossary

- **Car Servicing:** Maintenance and repair services for vehicles.
- **Car Resale:** Listing and selling cars through the platform.
- **Car Insurance:** Policies for vehicle insurance.