Auto Service Web App

Design Document

Table of Contents

[Introduction 3](#_Toc186766798)

[Problem Statement 3](#_Toc186766799)

[Goals 3](#_Toc186766800)

[Non-Goals 4](#_Toc186766801)

[Technology 4](#_Toc186766802)

[Design 4](#_Toc186766803)

[Database 4](#_Toc186766804)

[API 6](#_Toc186766805)

[Dependencies 7](#_Toc186766806)

[Testing 7](#_Toc186766807)

[Functional Testing 7](#_Toc186766808)

[Performance Testing 7](#_Toc186766809)

[Scale Testing 7](#_Toc186766810)

# Introduction

Auto Service is a web portal that facilitates car service bookings. It will be used by any person that owns a car and has basic computer skills. The amount of time they spend on the portal depends on what their goal is.

# Problem Statement

Booking your car in for service usually involves manually entering certain details every time your car is due for service. To make this process autonomous, Auto Service will collect these details once and use it for future services to save you time.

# Goals

1. Customer should be able to create an account and access it anytime.
2. Customer should be able to edit their account details.
3. Customer should be able to add vehicles to their fleet.
4. Customer should be able to book their vehicles for servicing.
5. Customer should be able to opt-in for service reminders.
6. Customer should be able to select the dealership that will perform the service.

# Non-Goals

1. Customer will not be able to receive detailed updates on their car service.
2. Customer will not be able to communicate with car technicians.

# Technology

**Frontend**

* **Framework:**
  + React + Vite
* **Styling:**
  + Tailwind

**Backend**

* **Framework:**
  + Nodejs

**Database:**

* Amazon RDS with Oracle DB

# Design

# Database

The Customer database will be used for storing customer personal and vehicle details. It will also be used to create service bookings.

Contains three tables:

* CUSTOMER: Stores customer details
  + Customer\_ID
  + Name
  + Surname
  + Email
  + Address
  + City
  + Province
  + Postal
  + Password(encrypted)
  + Service\_Reminders
  + Title
* VEHICLE: Stores details of customer vehicles
  + Car\_ID
  + Customer\_ID
  + Make
  + Model
  + Registration\_Number
  + VIN
  + Customer\_ID(FK)
* BOOKING: Stores details of customer service bookings
  + Booking\_ID
  + Customer\_ID(FK)
  + Vehicle\_ID(FK)
  + Date
  + Type

Relationships:

* CUSTOMER has **One-to-Many** relationship with VEHICLE
* CUSTOMER has **One-to-Many** relationship with BOOKING
* VEHICLEhas **One-to-One** relationship with CUSTOMER
* VEHICLEhas **One-to-Many** relationship with BOOKING
* BOOKINGhas **One-to-One** relationship with CUSTOMER
* BOOKINGhas **Many-to-One** relationship with VEHICLE

# API

**GET**

/api/bookings

/api/bookings/{id}

/api/vehicles

/api/vehicles/{id}

**POST**

/api/signup

/api/login

/api/new-booking

/api/new-vehicle

**PUT**

/update-personal-details/{id}

/update-security-details/{id}

/update-vehicle-details/{id}

/update-booking-details/{id}

**DELETE**

/remove-booking/{id}

/remove-vehicle/{id}

# Dependencies

# Testing

**Frontend**

* **Framework:**
  + **t**
* **Styling:**
  + T

**Backend**

* **Framework:**

# Functional Testing

# Performance Testing

# Scale Testing