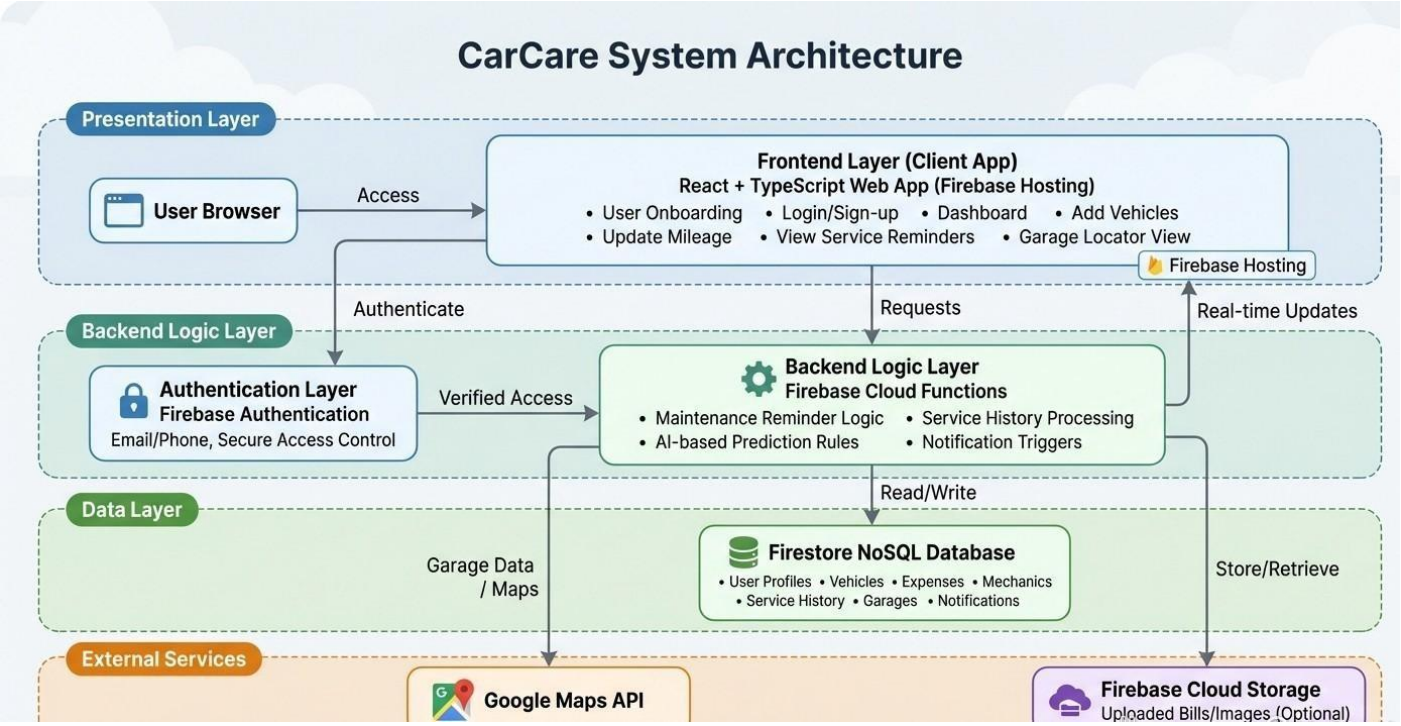
CarCare System Architecture



1. Overview

The CarCare system uses a **cloud-based, serverless architecture** designed for scalability, real-time updates, and secure data handling. The system is built using **React + TypeScript** on the frontend and **Firebase** services for backend logic, authentication, hosting, and data storage.
This architecture ensures fast performance, low maintenance overhead, and seamless synchronization between users and mechanics.

2. Presentation Layer

User Browser / Client App

- The CarCare web application runs in any modern browser.
- Users can complete onboarding, log in, view dashboards, manage vehicles, view reminders, and find garages.

Frontend Layer (React + TypeScript)

- The frontend is built using **React + TypeScript** and hosted on **Firebase Hosting**.
- It manages all UI interactions and communicates securely with Firebase services.

Key Features:

- User Onboarding
- Login / Signup
- Dashboard View
- Add Vehicles
- Update Mileage
- Maintenance Reminders
- Garage Locator View

3. Backend Logic Layer

Firebase Authentication

- Handles secure login/signup using email and phone number.
- Provides access control and generates tokens for protected data operations.
- Ensures only verified users can access personal vehicle and service data.

Firebase Cloud Functions

Cloud Functions act as the backend logic engine and support:

- **Maintenance Reminder Logic**
- **AI-based prediction rules** based on mileage & service history
- **Service history processing**
- **Notification triggers** to keep users updated

This eliminates the need for a traditional backend server.

4. Data Layer

Firestore NoSQL Database

The main data store for CarCare. It
securely stores:

- User Profiles
- Vehicle Details
- Service History
- Reminders
- Expenses
- Mechanic Details
- Garage Listings
- Notifications

Firestore provides:

- Real-time data updates to the frontend
- Scalable cloud-hosted storage
- Secure access rules

5. External Services

Google Maps API

- Integrated to display nearby garages.
- Receives user location & returns distance, coordinates, and map data.

Firebase Cloud Storage

- Optional storage for **uploaded bills, service images, invoices**, etc.
- Used for storing large or unstructured files.

6. Data Flow Explanation

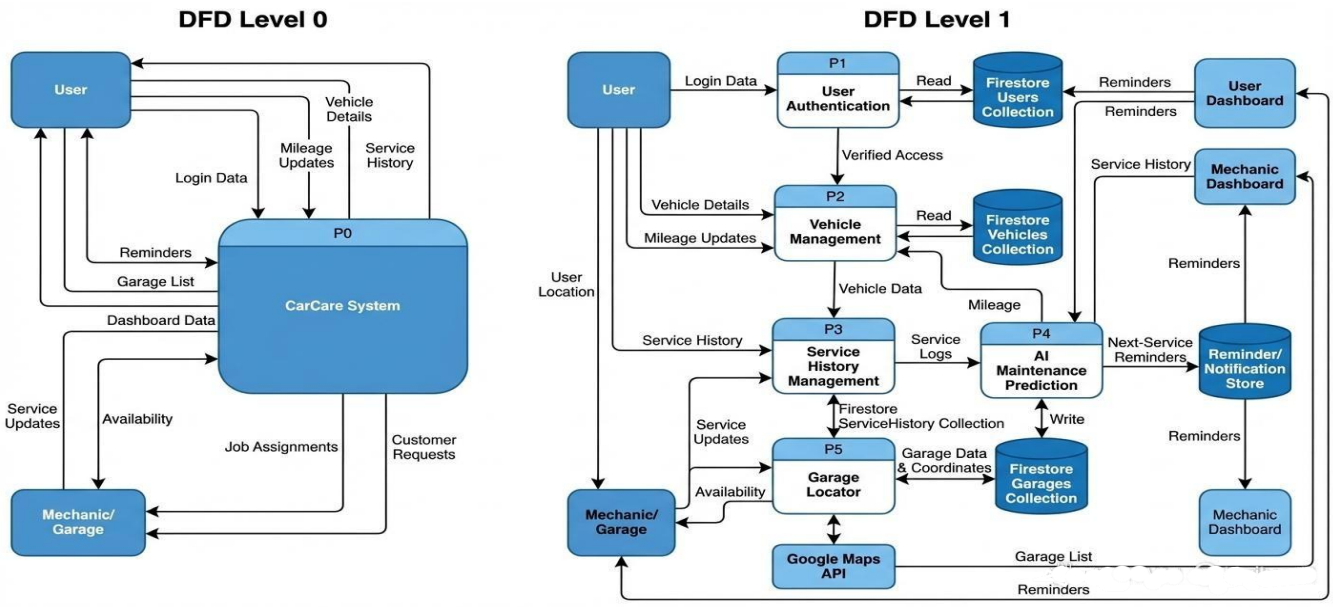
The overall flow of data in the CarCare system works as follows:

1. **User Browser → React Frontend**
    - User interacts with the web app hosted on Firebase.
2. **Frontend → Firebase Authentication**
    - Login and user verification.
3. **Frontend → Cloud Functions**
    - Requests for reminders, service processing, vehicle updates, etc.
4. **Cloud Functions → Firestore** ○ Read/write operations based on user actions.
5. **Cloud Functions → Google Maps API** ○ Fetching garage locations and map data.
6. **Cloud Functions → Cloud Storage**
    - Uploading bills or images from users.
7. **Firestore → Frontend (Real-time)**
    - Users see instant updates on dashboards (service history, reminders, expenses, etc.)

7. Advantages of This Architecture

- **Serverless** — No server maintenance required
- **Scalable** — Firebase handles growing traffic
- **Real-time** — Dashboards update instantly
- **Secure** — Authentication + Firestore rules
- **Cost-efficient** — Pay only for usage
- **Fast development** — Ideal for MVP and student projects

DATA FLOW DIAGRAM



CarCare – Vehicle Maintenance & Service Management System

1. DFD Level 0 – Context Diagram

Overview

DFD Level 0 represents the *entire CarCare system as a single high-level process*. It shows how users and mechanics exchange data with the system. This gives a complete picture of the system's external interactions without going into internal processing details.

External Entities

1. User

The user interacts with the CarCare system by sending:

- Login Data
- Vehicle Details
- Mileage Updates
- Service History

The system responds back with:

- Maintenance Reminders
- Garage List
- Dashboard Data

2. Mechanic / Garage

Mechanics interact with the system by sending:

- Service Updates
- Availability Information

The system sends mechanics:

- Job Assignments
- Customer Requests

Main Process: CarCare System (P0)

Inside the CarCare System, the following actions occur:

- User authentication
- Storing and retrieving vehicle data
- Processing service history
- AI-based generation of reminders
- Fetching and providing garage details
- Sending dashboard updates to the user

DFD Level 0 – Text Explanation of Data Flows

User → CarCare System:
Login Data, Vehicle Details, Mileage Updates, and Service History

CarCare System → User:
Reminders, Garage List, Dashboard Data

Mechanic → CarCare System: Service
Updates, Availability

CarCare System → Mechanic:
Job Assignments, Customer Requests

2. DFD Level 1 – Detailed Process Diagram

Overview

DFD Level 1 expands the CarCare System into **five detailed subprocesses** that show how data moves through the system internally. It also shows how Firestore collections and Google Maps API are used to store and retrieve data.

Processes in DFD Level 1

P1 – User Authentication

- User sends login data.
- System validates via Firebase Authentication.
- Verified access token is returned.

Data Store: Firestore Users Collection

P2 – Vehicle Management

- User adds or edits vehicle details.
- User sends mileage updates.
- Vehicle data is stored and retrieved from Firestore Vehicles Collection.

Data Store: Firestore Vehicles Collection

P3 – Service History Management

- System receives service history inputs from the user or mechanic.
- Service logs are stored in Firestore ServiceHistory Collection.
- Data is shown in user and mechanic dashboards.

Data Store: Firestore ServiceHistory Collection

P4 – AI Maintenance Prediction

- Reads mileage and service logs.
- Computes next service schedule using rule-based AI.
- Stores the next-service reminders in the Reminder/Notification Store.

Output: Maintenance Reminders

P5 – Garage Locator

- Receives user location.
- Fetches garage data and coordinates from Google Maps API.
- Reads/writes garage data from Firestore Garages Collection.
- Sends garage list to User Dashboard and Mechanic Dashboard.

External Entity: Google Maps API

DFD Level 1 – Text Explanation of Data Flows

User → P1 Authentication

- Login Data → Verified Access

User → P2 Vehicle Management

- Vehicle Details → Stored in Firestore
- Mileage Updates → Trigger AI Prediction

User → P3 Service History Management

- Service History → Stored & Used for Prediction

P4 → Users

- Next-Service Reminders → User Dashboard

P5 → Users

- Garage List retrieved using Google Maps API

Mechanic ↔ System

- Service Updates
- Availability
- Reminders and Job Assignments

Data Stores Used in DFD Level 1

Firestore Users Collection

Used for login verification, user profile data

Firestore Vehicles Collection

Stores vehicle details and mileage

Firestore ServiceHistory Collection

Stores logs of all user services

Firestore Garages Collection

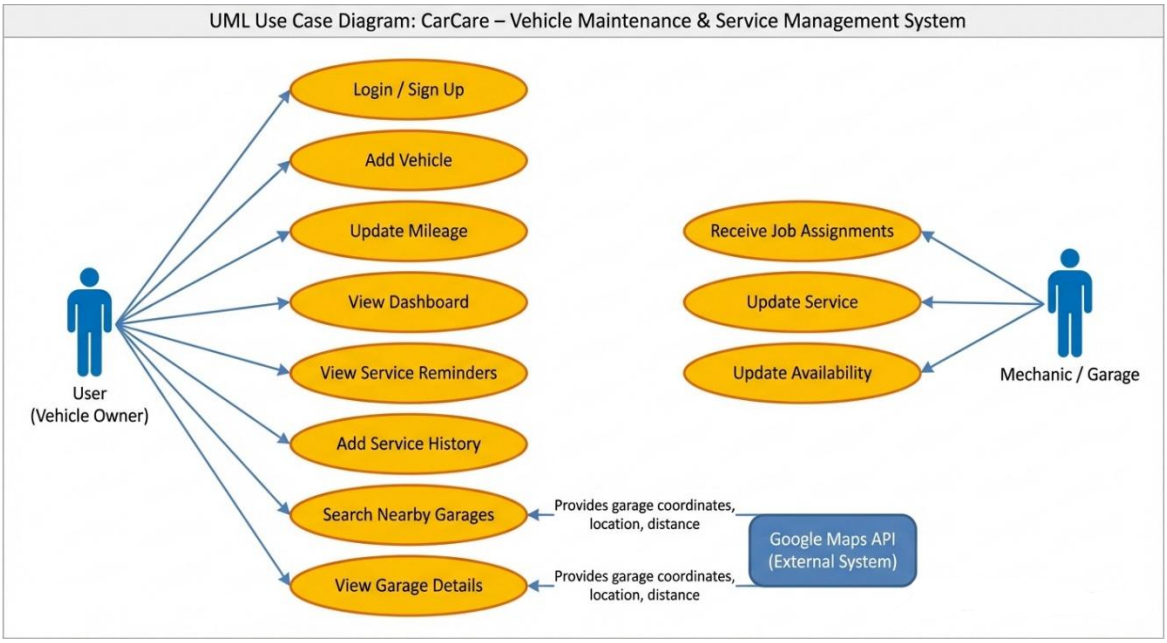
Stores garage details, coordinates, ratings

Reminder / Notification Store

Stores next-service reminders

UML USE CASE DIAGRAM DOCUMENTATION

CarCare – Vehicle Maintenance & Service Management System



1. Overview

The Use Case Diagram represents how different actors interact with the CarCare system.
It identifies the primary functionalities offered by the system from the user's perspective.

CarCare has **two main actors**:

1. **User (Vehicle Owner)**
2. **Mechanic / Garage**

There is also one optional external actor:

3. **Google Maps API** (External System)

2. Actors in the System

1. User (Primary Actor)

The user performs the following actions:

- Create account / login
- Add vehicle
- Update mileage
- View dashboard
- View service reminders
- Add service history
- Search for nearby garages
- View garage details

2. Mechanic / Garage (Secondary Actor)

The mechanic performs:

- View assigned jobs
- Update service status
- Provide availability
- Manage service records

3. Google Maps API (External Actor)

Used for:

- Fetching nearby garage locations
- Getting coordinates and distance

3. Use Cases of CarCare

Below are the primary use cases included in the UML diagram:

UC1 – Login / Sign Up

Actor: User
Description: Authenticates using Firebase Authentication.

UC2 – Add Vehicle

Actor: User
Description: User adds vehicle details (brand, model, year, mileage).

UC3 – Update Mileage

Actor: User
Description: Updates current odometer reading.

UC4 – View Dashboard

Actor: User
Description: Access complete overview including reminders, history, and garage suggestions.

UC5 – View Service Reminders

Actor: User
Description: Receives AI-generated next-service reminders.

UC6 – Add Service History

Actor: User
Description: Enters details about repairs, parts replaced, and service cost.

UC7 – Search Nearby Garages

Actor: User
External: Google Maps API
Description: System fetches garage locations based on user's position.

UC8 – View Garage Details

Actor: User
Description: View distance, address, services, and contact details.

UC9 – Receive Job Assignments

Actor: Mechanic
Description: Mechanic receives customer requests or upcoming jobs.

UC10 – Update Service

Actor: Mechanic
Description: Mechanic updates service status, parts used, and completion details.

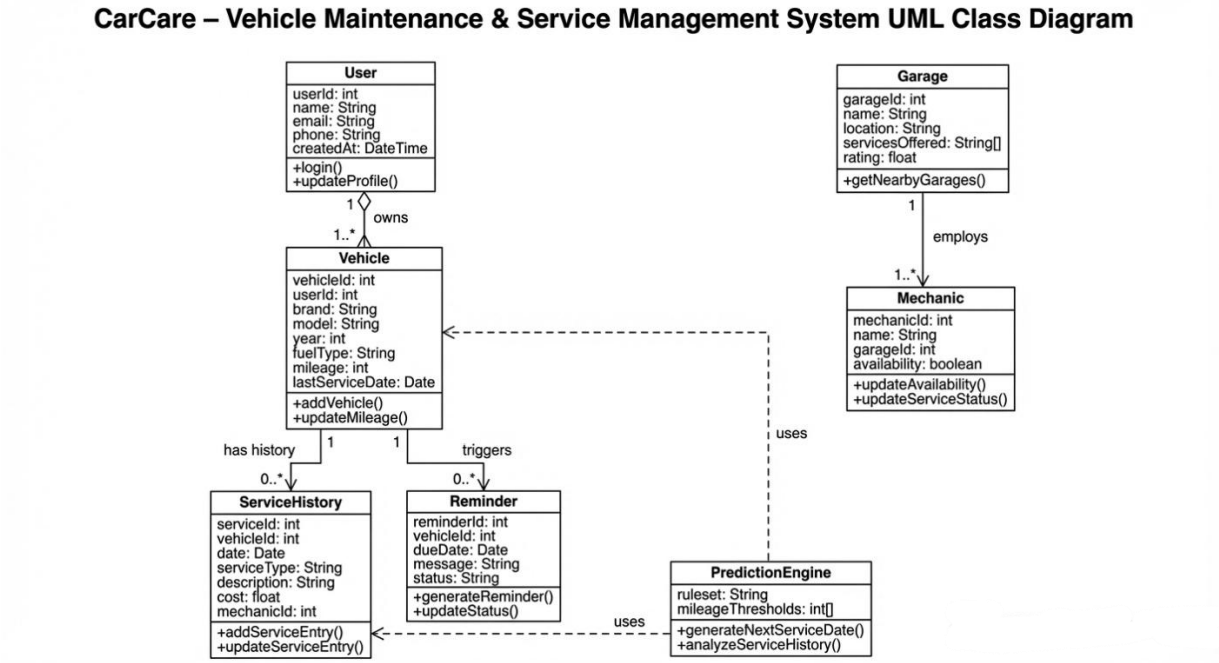
UC11 – Update Availability

Actor: Mechanic
Description: Mechanic updates working hours and availability.

UML CLASS DIAGRAM

Overview

The UML Class Diagram defines the structure of the CarCare application.
It shows classes, attributes, methods, and relationships between core components .CarCare uses a modular component-based architecture with Firestore collections acting as primary data stores. The following classes abstract the main system modules.



Class Descriptions

1. User

Attributes:

- userId
- name
- email
- phone
- createdAt

Methods:

- login()
- updateProfile()

Description:
Represents the authenticated vehicle owner. Stores identity information and connects to the user's vehicles and service history.

2. Vehicle

Attributes:

- vehicleId
- userId
- brand
- model
- year
- fuelType
- mileage
- lastServiceDate

Methods:

- addVehicle()
- updateMileage()

Description:
Stores all vehicle information. Changes in mileage trigger reminder and prediction logic.

3. ServiceHistory

Attributes:

- serviceId
- vehicleId
- date
- serviceType
- description
- cost
- mechanicId

Methods:

- addServiceEntry()
- updateServiceEntry()

Description:
Represents repair/maintenance jobs done on a vehicle.

4. Reminder

Attributes:

- reminderId
- vehicleId
- dueDate
- message
- status

Methods:

- generateReminder()
- updateStatus()

Description:
Generated automatically based on mileage and service logs.

5. Garage

Attributes:

- garageId
- name
- location
- servicesOffered
- rating

Methods:

- getNearbyGarages()

Description:
Fetched using Firestore + Google Maps API to locate nearby service centers.

6. Mechanic

Attributes:

- mechanicId
- name
- garageId
- availability

Methods:

- updateAvailability()
- updateServiceStatus()

Description:
Represents garage personnel managing service jobs.

7. PredictionEngine

Attributes:

- ruleset
- mileageThresholds

Methods:

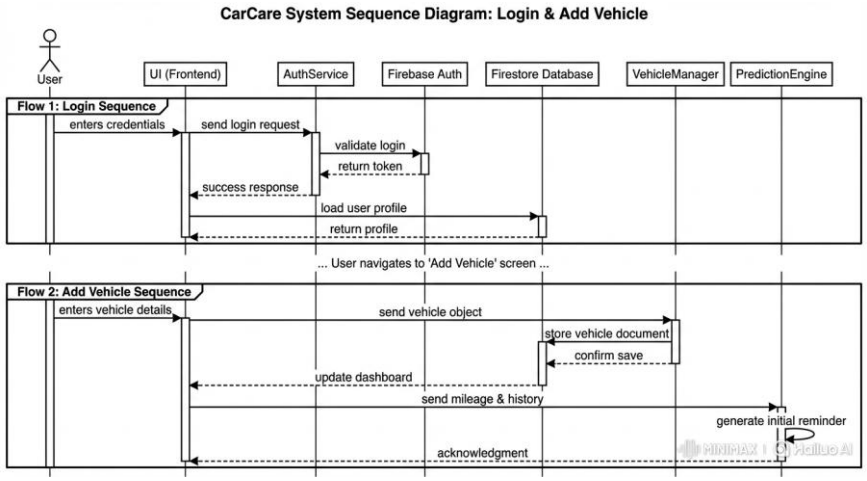
- generateNextServiceDate(vehicle)
- analyzeServiceHistory()

Description:
Implements rule-based AI logic to forecast upcoming maintenance.

Relationships

- User ↔ Vehicle: **1-to-many**
- Vehicle ↔ ServiceHistory: **1-to-many**
- Vehicle ↔ Reminder: **1-to-many**
- Garage ↔ Mechanic: **1-to-many**
- User ↔ Garage (via search): **indirect**
- PredictionEngine ↔ Vehicle & ServiceHistory: **uses**

SEQUENCE DIAGRAM (Login & Add Vehicle)



Overview

Sequence Diagrams explain step-by-step interactions between system components during a feature execution.

A. LOGIN SEQUENCE

Description

This sequence shows how a user logs into the CarCare app using Firebase Authentication.

Steps

1. **User → UI:** Enters email/phone and password.
2. **UI → AuthService:** Sends login credentials.
3. **AuthService → Firebase Authentication:** Validates credentials.
4. **Firebase Auth → AuthService:** Returns authentication token.
5. **AuthService → UI:** Confirms login success.
6. **UI → Firestore:** Requests user profile data.

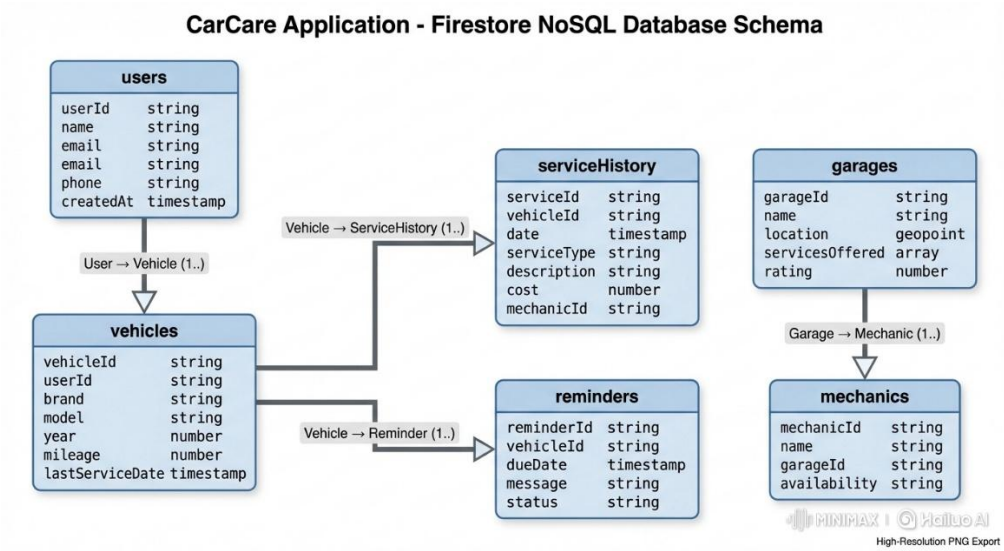
7. **Firestore → UI:** Returns user data for dashboard display.

B. ADD VEHICLE SEQUENCE

Steps

1. **User → UI:** Opens "Add Vehicle" page and submits details.
2. **UI → VehicleManager:** Sends vehicle object.
3. **VehicleManager → Firestore:** Stores vehicle document in vehicles collection.
4. **Firestore → VehicleManager:** Confirms save.
5. **VehicleManager → UI:** Refreshes dashboard with updated vehicle list.
6. **UI → PredictionEngine:** Sends mileage + year + history.
7. **PredictionEngine:** Generates initial prediction rules.

4.3 DATABASE SCHEMA (Firestore)



Overview

CarCare uses Google Firebase Firestore, a NoSQL cloud database that stores data in collections and documents. This structure ensures scalability and lightning-fast reads required for real-time updates.

Collections & Fields

1. users Collection

Field	Type	Description
userId	string	Authenticated user reference
name	string	User full name
email	string	Login email
phone	string	Mobile number
createdAt	timestamp	Account creation time

2. vehicles Collection

Field	Type	Description
vehicleId	string	Unique identifier
userId	string	Foreign key to users collection
brand	string	Vehicle brand
model	string	Vehicle model
year	number	Manufacturing year
mileage	number	Current odometer
lastServiceDate	timestamp	Last service entry

3. serviceHistory Collection

Field	Type	Description
serviceId	string	Unique service identifier
vehicleId	string	Map to vehicle
date	timestamp	Date of service
serviceType	string	Type of maintenance
description	string	Details
cost	number	Service cost
mechanicId	string	Mechanic reference

4. reminders Collection

Field	Type	Description
reminderId	string	
vehicleId	string	
dueDate	timestamp	
message	string	
status	string	

5. garages Collection

Field	Type
garageId	string
name	string

Field	Type
location	geoPoint
servicesOffered	array
rating	number

6. mechanics Collection

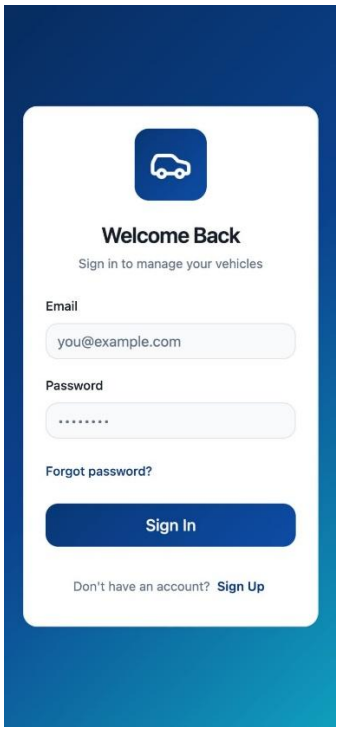
Field	Type
mechanicId	string
name	string
garageId	string
availability	string

WIREFRAMES / UI DESIGN

1.Onboarding Screen

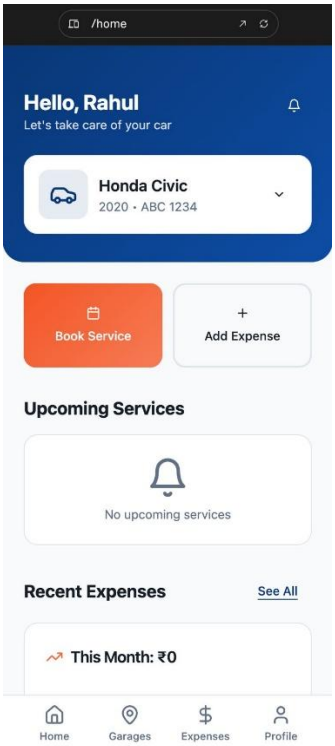
Description:
Shows a welcome message and application purpose. Guides users to login or sign up.

2.Login / Signup Screen



Description:
Allows user authentication using email/phone. Integrated with Firebase Auth.

3. **Dashboard Screen**

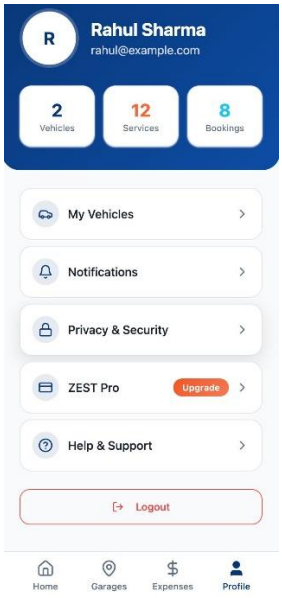


Description:

Displays:

- Vehicles
- Reminders
- Garage suggestions
- Quick actions

Profile / Settings Screen



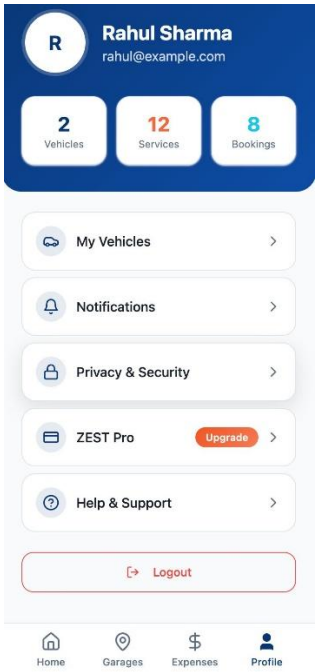
Description

This screen shows the user's personal information and profile options.

Features include:

- Profile stats (Vehicles, Services, Bookings)
- My Vehicles
- Notifications
- Privacy & Security
- Zest Pro upgrade
- Help & Support
- Logout button

Notification Screen

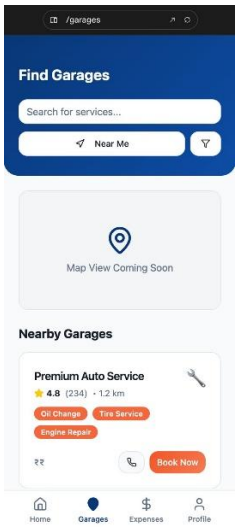


Description

Displays alerts and reminders such as:

- Upcoming service
- Booking confirmations
- Expense alerts

Garage Finder Screen



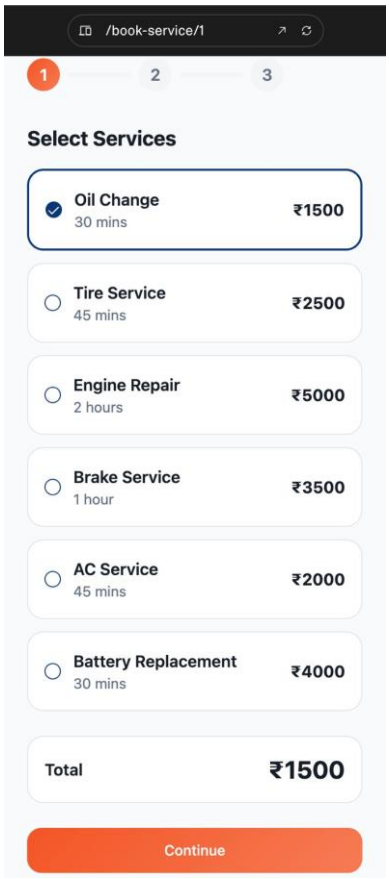
Description

Allows users to:

- Search for nearby garages
- Filter based on services
- View garage list with ratings, distance, and service tags
- Book Now button for selected garage

Uses **Google Maps API** for approximate location (map coming soon).

Book Service – Select Services Screen



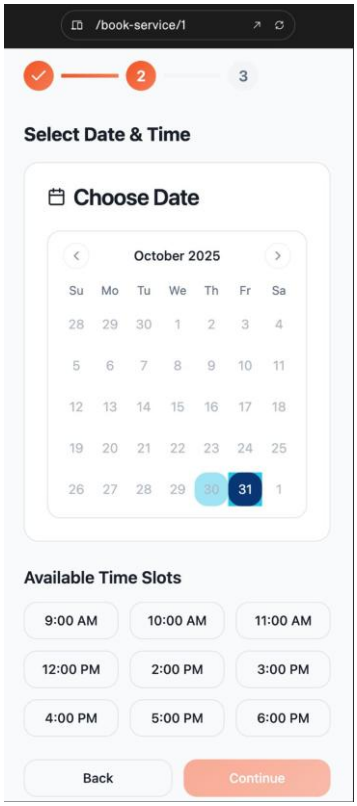
Description

This is step 1 of the service booking workflow.

Users can:

- Select one or more services (Oil Change, Tire Service, Engine Repair, etc.)
- View cost and duration
- Automatically calculate total amount

Book Service – Select Date & Time



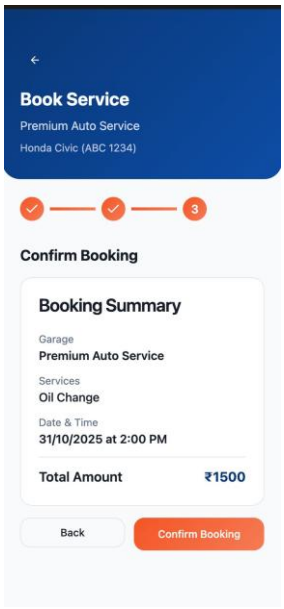
Description

This is step 2 of the booking process.

Users can:

- Choose a date from the calendar
- Select available time slots
- Proceed to confirmation

Book Service – Confirm Booking Screen

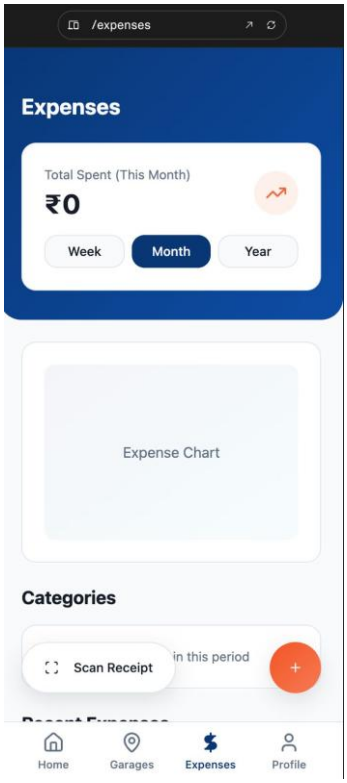


Description

This final step displays a summary:

- Garage name
- Selected service
- Date & time
- Total amount
- Confirm Booking button

Expenses Screen



Description

Allows users to manage vehicle-related expenses.

Features:

- Expense chart (month/week/year toggle)
- Scan Receipt button
- Add Expense button
- Recent expenses list