

## # CarMarket - Car Selling Online Market

A full-stack car buying and selling website with **\*\*10 pages\*\***, built with:

- **\*\*Frontend:\*\*** Vite + React.js + Bootstrap 5
- **\*\*Backend:\*\*** Node.js + Express.js + MongoDB

### ## 10 Pages

1. **\*\*Home\*\*** – Landing page with hero, featured listings, and CTAs
2. **\*\*Browse Cars\*\*** – Listings with search, filters (price, fuel, transmission), and pagination
3. **\*\*Car Detail\*\*** – Single car view with specs and seller contact
4. **\*\*Add Car\*\*** – Sell your car with **\*\*image upload\*\*** (up to 5 images; protected, login required)
5. **\*\*Edit Car\*\*** – Edit your listing and add/remove images (protected, owner only)
6. **\*\*About\*\*** – About the platform
7. **\*\*Contact\*\*** – Contact form and support info
8. **\*\*Login\*\*** – User login
9. **\*\*Register\*\*** – User registration
10. **\*\*Profile\*\*** – Account info and “My Listings” (protected)

### ## Prerequisites

- **\*\*Node.js\*\*** (v18+)
- **\*\*MongoDB\*\*** (local or Atlas)

## ## Setup

### ### 1. Backend (server)

```
```bash
cd server
npm install
...`
```

Create a `.env` file (or copy from `.env.example`):

```
...
PORT=5000
MONGODB_URI=mongodb://localhost:27017/car-market
JWT_SECRET=your-secret-key-change-in-production
...`
```

Start the server:

```
```bash
npm run dev
...`
```

Server runs at **`http://localhost:5000`**.

**\*\*Seed the database with 20 sample cars\*\*** (and a demo user):

```
```bash
cd server
npm run seed
```
```

This creates user `demo@carmarket.com` / `demo123` and 20 pre-loaded car listings.

### ### 2. Frontend (client)

```
```bash
cd client
npm install
```
```

Start the dev server:

```
```bash
npm run dev
```
```

Frontend runs at **\*\*http://localhost:5173\*\***. API requests are proxied to the backend during development.

### ### 3. MongoDB

- **Local:** Start MongoDB and use `MONGODB\_URI=mongodb://localhost:27017/car-market`.
- **Atlas:** Use your connection string in `MONGODB\_URI`.

## ## API Overview

- **Auth:** `POST /api/auth/register`, `POST /api/auth/login`, `GET /api/auth/me` (Bearer token)
- **Cars:** `GET /api/cars` (list with query params), `GET /api/cars/featured`, `GET /api/cars/:id`, `POST /api/cars` (protected; JSON or multipart with images), `PUT /api/cars/:id` (protected; JSON or multipart), `GET /api/cars/my/listings` (protected). Uploaded images are stored in `server/uploads/cars/` and served at `/uploads/cars/...`.

...

## Cars-selling Online Market

```
├─ client
|   ├── .env.example
|   ├── eslint.config.js
|   ├── index.html
|   ├── package-lock.json
|   ├── package.json
|   ├── public
|   │   └─ vite.svg
|   └─ README.md
|   └─ src
|       ├── api.js
|       └─ App.css
```

- | | ├── App.jsx
- | | ├── assets
- | | | └── react.svg
- | | ├── components
- | | | ├── AdminRoute.jsx
- | | | ├── Footer.jsx
- | | | ├── Navbar.css
- | | | ├── Navbar.jsx
- | | | └── PrivateRoute.jsx
- | | ├── context
- | | | └── AuthContext.jsx
- | | ├── index.css
- | | ├── main.jsx
- | | └── pages
- | | ├── About.css
- | | ├── About.jsx
- | | ├── AddCar.jsx
- | | ├── Admin.jsx
- | | ├── Auth.css
- | | ├── CarDetail.jsx
- | | ├── Cars.jsx
- | | ├── Contact.css
- | | ├── Contact.jsx
- | | ├── EditCar.jsx
- | | └── Home.jsx

- | | | |└─ Login.jsx
- | | | |└─ Profile.jsx
- | | | |└─ Register.jsx
- | |└─ vite.config.js
- | |└─ Documentation
- | |└─ About.pdf
- | |└─ README.md
- |└─ server
  - | | | |└─ .env
  - | | | |└─ .env.example
  - | | | |└─ config
  - | | | |└─ db.js
  - | | | |└─ middleware
  - | | | |└─ auth.js
  - | | | |└─ upload.js
  - | | | |└─ models
  - | | | |└─ Car.js
  - | | | |└─ User.js
  - | | | |└─ package-lock.json
  - | | | |└─ package.json
  - | | | |└─ routes
  - | | | |└─ admin.js
  - | | | |└─ auth.js
  - | | | |└─ cars.js
  - | | | |└─ seed.js

└─ server.js

└─ uploads

└─ cars

...

## ## Production

- **Backend:** Set `PORT`, `MONGODB_URI`, and a strong `JWT_SECRET`. Run `node server.js`.
- **Frontend:** Set `VITE_API_URL` to your backend URL (e.g. `https://api.yoursite.com/api`). Run `npm run build` and serve the `dist` folder.