Classic Car Blog

A full-stack web application for **classic cars enthusiasts**, featuring a collection of iconic vehicles. Users can register, login, like cars, and view detailed car information. Built with **Django REST Framework** (backend) and **Angular 20 + PrimeNG** (frontend).

Table of Contents

1. Project Overview
2. Features
3. Technologies Used
4. Project Structure
5. Setup Instructions
6. Backend Setup
7. Frontend Setup
8. API Documentation
9. Frontend Layout Notes
10. Usage
11. Dependencies
12. License

Project Overview

Classic Car Blog provides a platform to showcase classic cars, emphasizing JDM, European, and American legends. The platform allows user registration, authentication, car liking, commenting, and admin car management.

Features

- User registration & login (token-based authentication)
- Like and comment on cars
- Featured car section and car collections
- Responsive UI with PrimeNG components
- Admin panel for car management
- Home page full-width layout; other pages constrained with a container

---

Technologies Used

Backend:

- Python 3.13+
- Django 5.1+
- Django REST Framework
- Django REST Framework Token Authentication
- Django CORS Headers
- SQLite (default) or PostgreSQL

Frontend:

- Angular 20
- PrimeNG & PrimeIcons
- Tailwind CSS
- RxJS

Project Structure

```
classic-car-blog/
│
├─ backend/
│   ├─ carproject/          # Django project settings
│   ├─ cars/                # Cars app (models, views, serializers)
│   ├─ users/               # Users app (authentication)
│   ├─ manage.py
│   └─ requirements.txt     # Python dependencies
│
├─ frontend/
│   ├─ src/app/
│   │   ├─ auth/            # Login & Register components
│   │   ├─ components/      # Home, CarList, CarDetails, LikeButton
│   │   ├─ services/        # AuthService, CarService
│   │   ├─ navbar/          # Navbar component
│   │   └─ app.component.ts
│   ├─ package.json
│   └─ angular.json
│
└─ README.md
```

Setup Instructions

Backend Setup

1. Clone the repository:

```
git clone <repository-url>
cd classic-car-blog/backend
```

1. Create and activate a virtual environment:

```
python -m venv venv
# Linux/Mac
source venv/bin/activate
# Windows
venv\Scripts\activate
```

1. Install dependencies:

```
pip install -r requirements.txt
```

1. Apply database migrations:

```
python manage.py migrate
```

1. Create a superuser for admin access:

```
python manage.py createsuperuser
```

1. Start the backend server:

```
python manage.py runserver
```

Backend API will run at: `http://127.0.0.1:8000/api/`

Frontend Setup

1. Navigate to the frontend directory:

```
cd ../frontend
```

1. Install Node dependencies:

```
npm install
```

1. Start the frontend server:

```
ng serve
```

Frontend will run at: `http://localhost:4200/`

API Documentation

Method	Endpoint	Description
POST	`/api/users/create/`	Register a new user
POST	`/api/users/login/`	Login user (username/email)
GET	`/api/cars/`	List all cars
GET	`/api/cars/<id>/`	Retrieve car details
POST	`/api/cars/<id>/like/`	Toggle like for a car
GET/POST	`/api/cars/<id>/comments/`	List/Create comments
GET/PUT/DELETE	`/api/comments/<id>/`	Manage a single comment

Authentication:

- Token-based auth using Django REST Framework Token
- Include token in `Authorization` header for protected endpoints:

```
Authorization: Token <your_token>
```

Frontend Layout Notes

- **Home Page:** Full-width layout, fills viewport height (`min-h-screen`).
- **Other Pages:** Constrained inside a `container` with padding (`px-6 py-12`) to maintain consistent content width.

- **Navbar & Footer:** Navbar is included in `app.component.html`. Footer is implemented in `home.component.html` for full-width effect.

Angular Example for Page Layout:

```
<!-- Full-width for home -->
<app-home></app-home>

<!-- Container for other pages -->
<div class="container mx-auto px-6 py-12">
  <router-outlet></router-outlet>
</div>
```

Usage

1. **Register** a new account via frontend or API.
2. **Login** with username or email.
3. **Browse cars**, like your favorites, and leave comments.
4. **Admin Panel**: Manage cars at `/admin/`.
5. **Authentication**: Token stored in `localStorage` for persistent sessions.

Dependencies

Backend (`requirements.txt`):

```
Django>=5.1
djangorestframework>=4.14
djangorestframework-authtoken>=1.4
django-cors-headers>=4.0
```

Frontend (`package.json` key deps):

```
{
  "@angular/core": "~20.1.0",
  "rxjs": "~7.8.0",
  "primeng": "^16.0.0",
  "primeicons": "^6.0.0",
  "tailwindcss": "^4.3.0"
}
```

License

This project is licensed under the MIT License.