Vehicle Classified Ads API using Django REST Framework

Overview

This project is a RESTful API for a vehicle classified ads platform, allowing users to post and browse listings for cars and motorcycles. It provides a backend-only solution suitable for mobile or frontend integration. The goal was to simulate a basic version of platforms like Divar.ir, focusing on backend functionality.

Features Implemented

- User Registration and Authentication (Token based)
- User profile information
- Create, Update, Delete Ads, Reviews, and Images (CRUD)
- Upload and manage multiple images for each ad
- Used `Pillow` to restrict maximum image size and ensure uploaded images meet dimension requirements
- Implemented a one-to-one relationship between the built-in User model and a custom `Profile` model to store additional user information.
- Used Django signals to automate tasks (auto-creating profiles, cleaning up related data)
- Filter, search, and order ads by fields
- Permissions
- Pagination for listing ads
- Custom validators to prevent inappropriate content
- Admin content check phase for ads, images, and reviews

Technical Stack

- **Backend Framework:** Django, Django Rest Framework
- **Authentication:** Token Authentication Or JSON Web Tokens (JWT)
- **Database:** SQLite
- **Image Handling:** ImageField with media_root and media_url and Pillow library
- **Pagination:** Page number pagination and Limit Offset pagination
- **Filtering and Searching:** django-filters library and DRF filters (SearchFilter and OrderingFilter)
- **Development tools:** postman for testing APIs, Git for version control

Project Structure

- **ads_app:** provides full CRUD functionality (Create, Read, Update, Delete) for car ads, motorcycle ads, uploaded images, and user reviews.
- **content_check_app:** handles moderation, where admins review submitted car ads, motorcycle ads, images, and reviews, and decide whether to approve or reject them.
- **user_app:** provides user authentication features including registration, login, logout, password change, and access to a personal dashboard showing pending and rejected submissions.

Challenges & Learning

Throughout this project, I encountered and overcame several technical challenges that deepened my understanding of Django and the Django REST Framework:

- **Manual Migration Management:** I learned how to manage migration files manually to handle model instance changes more precisely and avoid migration conflicts.

- **Signals:** I explored and implemented Django signals to automate certain actions, such as post-save logic and data cleanup.

- **Generic Relationships:** I challenged myself to design a flexible review system by using Django's `ContentType` framework and `GenericForeignKey` to allow a single `Review` model to support both `CarAd` and `MotorcycleAd` models.

- **User Profiles:** I extended the built-in `User` model by creating a custom `Profile` model using a one-to-one relationship to store additional user information.

- **Request Lifecycle:** I studied the lifecycle of an incoming HTTP request in Django to better understand how to handle complex behaviors and customize the request-response process.

- **Permissions:** I learned to implement and manage permissions effectively, including custom permissions at both the view level and the object level, to control access to various resources.