Тестовое задание: Создание веб-приложения для управления информацией об автомобилях с использованием API

Описание задания:

Вам нужно создать веб-приложение для управления информацией об автомобилях с использованием python (django). Приложение должно включать следующие функции:

1. **Управление автомобилями**:
    - Пользователи могут просматривать список автомобилей.
    - Пользователи могут просматривать информацию о каждом автомобиле.
    - Только зарегистрированные пользователи могут добавлять новые автомобили.
    - Пользователи могут редактировать и удалять только свои записи об автомобилях.
2. **Комментарии**:
    - Зарегистрированные пользователи могут оставлять комментарии к автомобилю.
    - Комментарии должны отображаться под соответствующим автомобилем.
3. **API**:
    - Реализуйте REST API для получения, создания, редактирования и удаления автомобилей.
    - Реализуйте REST API для получения и добавления комментариев к автомобилям.
4. **Регистрация и авторизация**:
    - Реализуйте регистрацию и авторизацию пользователей.
    - Пользователи могут войти в систему и управлять только своими автомобилями.
5. **Административная панель**:
    - Модели автомобилей и комментариев должны быть доступны в административной панели Django.

Технические требования:

1. **Стек технологий**:
    - Python
    - Django (последняя стабильная версия)
    - Django REST Framework (DRF) для реализации API
    - База данных SQLite (по умолчанию) или PostgreSQL.
2. **Модели**:
    - `Car` — модель автомобиля с полями:
        - `make` (марка автомобиля) — например, "Toyota", "Ford"
        - `model` (модель автомобиля) — например, "Camry", "Mustang"
        - `year` (год выпуска) — например, 2020, 2021

        - `description` (описание автомобиля) — например, "Компактный седан с хорошей экономией топлива", "Спортивный автомобиль с мощным двигателем"
        - `created_at` (дата и время создания записи)
        - `updated_at` (дата и время последнего обновления записи)
        - `owner` (внешний ключ на пользователя, который создал запись)
    - `Comment` — модель комментария с полями:
        - `content` (содержание комментария) — например, "Отличный автомобиль, но немного шумный на высоких скоростях."
        - `created_at` (дата и время создания комментария)
        - `car` (внешний ключ на автомобиль)
        - `author` (внешний ключ на пользователя)
3. **Примеры данных о автомобилях**:
    - **Автомобиль 1**:
        - `make`: "Toyota"
        - `model`: "Camry"
        - `year`: 2021
        - `description`: "Компактный седан с отличной экономией топлива и современными технологиями безопасности."
    - **Автомобиль 2**:
        - `make`: "Ford"
        - `model`: "Mustang"
        - `year`:
        - `description`: "Спортивный автомобиль с мощным двигателем и агрессивным дизайном."
    - **Автомобиль 3**:
        - `make`: "Honda"
        - `model`: "Civic"
        - `year`: 2020
        - `description`: "Надежный и экономичный седан, идеален для городской эксплуатации."
4. **Views и шаблоны**:
    - Реализуйте следующие страницы:
        - Список автомобилей (главная страница).
        - Страница с информацией о конкретном автомобиле и комментариями, а также форма для добавления комментариев.
        - Форма для добавления нового автомобиля (доступна только зарегистрированным пользователям).
        - Форма для редактирования и удаления автомобиля (только для владельца автомобиля).
    - Используйте базовый HTML/CSS для минимального оформления страниц.
5. **API**:
    - Используйте Django REST Framework для создания API.
    - Реализуйте следующие конечные точки API:

        - `GET /api/cars/` — получение списка автомобилей.
        - `GET /api/cars/<id>/` — получение информации о конкретном автомобиле.
        - `POST /api/cars/` — создание нового автомобиля.
        - `PUT /api/cars/<id>/` — обновление информации о автомобиле.
        - `DELETE /api/cars/<id>/` — удаление автомобиля.
        - `GET /api/cars/<id>/comments/` — получение комментариев к автомобилю.
        - `POST /api/cars/<id>/comments/` — добавление нового комментария к автомобилю.
6. **Документация**:
    - Напишите краткую документацию по установке и запуску приложения (например, в README.md файл).
    - Укажите, как запустить сервер разработки, как выполнить миграцию базы данных и как использовать API.

Оценочные критерии:

1. **Код**:
    - Чистота и структура кода.
    - Корректное использование Django ORM и Django REST Framework.
    - Наличие адекватных комментариев и документации.
2. **Функциональность**:
    - Полное выполнение всех требований задания.
    - Корректность работы всех функций и API.
3. **Интерфейс**:
    - Минимальный пользовательский интерфейс.
4. **Безопасность**:
    - Правильная настройка прав доступа и авторизации.
    - Безопасное использование API (например, через аутентификацию).
5. **Документация**:
    - Наличие четкой инструкции по запуску проекта и использованию API.

Как сдавать задание:

1. Разработайте проект и разместите его на платформе для управления исходным кодом (например, GitHub).
2. Включите все необходимые файлы для запуска проекта (включая `requirements.txt` и инструкции по установке).
3. Отправьте ссылку на репозиторий и краткое описание выполненного задания нам.

Срок выполнения задания: 5 дней с момента получения тестового задания.