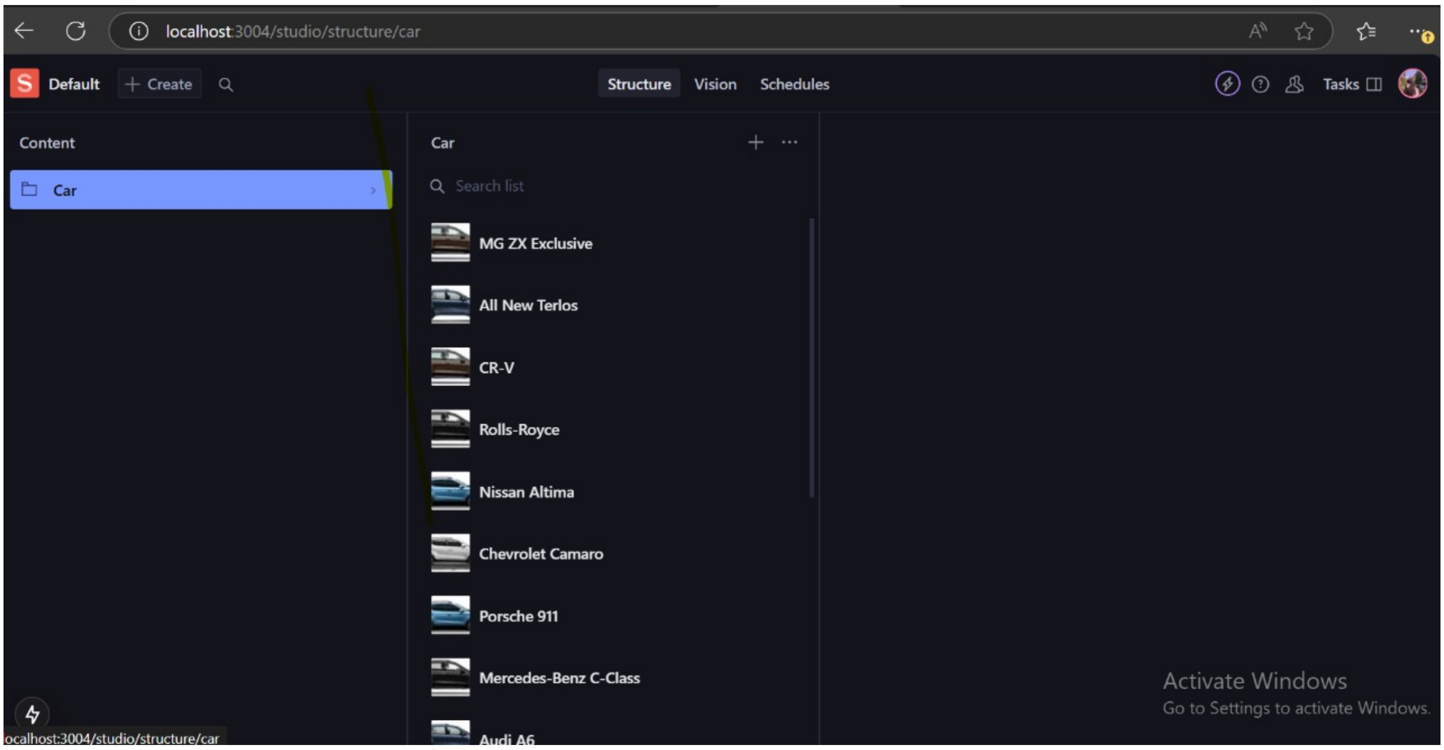
Rental Car E-commerce: Integration Documentation

This document provides a comprehensive overview of the Rental Car E-commerce project, focusing on tasks such as integrating Sanity CMS, creating schemas, importing car data via APIs, and setting up the environment. The screenshots below outline the process.

Sanity Studio Car Data Structure:

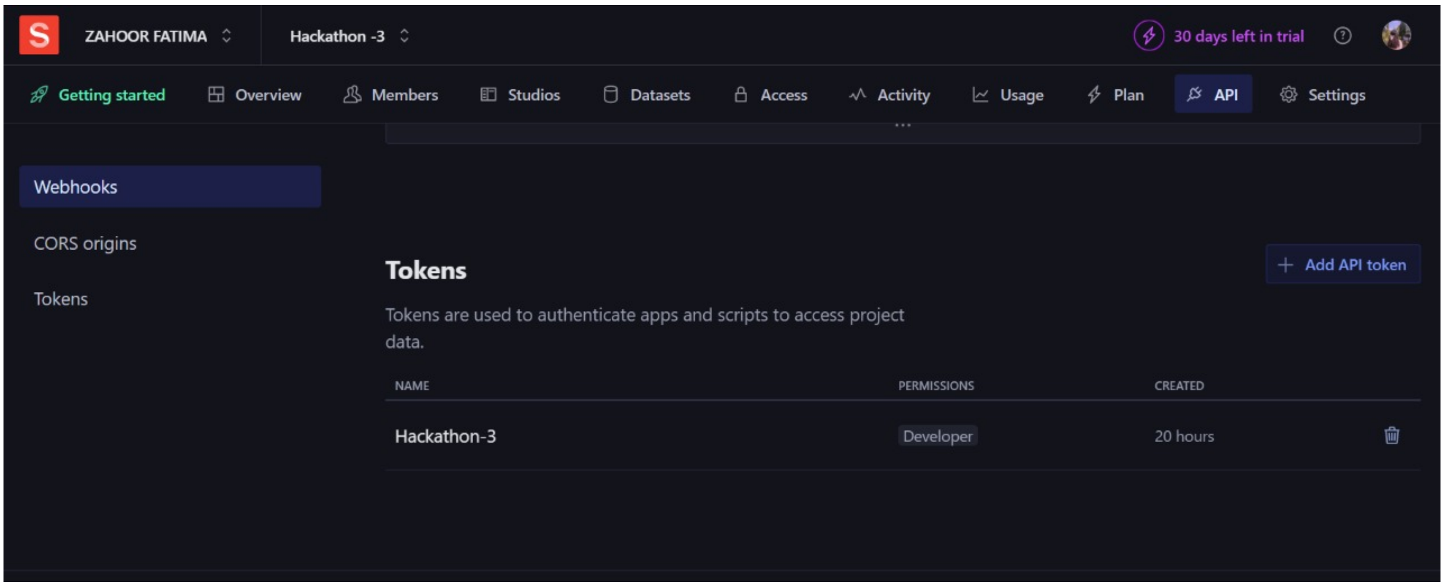
This screenshot shows car data structured in Sanity CMS, including models like MG ZX Exclusive and Rolls-Royce.



MG ZX Exclusive
All New Terlos
CR-V
Rolls-Royce
Nissan Altima
Chevrolet Camaro
Porsche 911
Mercedes-Benz C-Class
Audi A6

API Token Setup in Sanity:

This image demonstrates how API tokens are configured in Sanity to enable secure data access for the application.



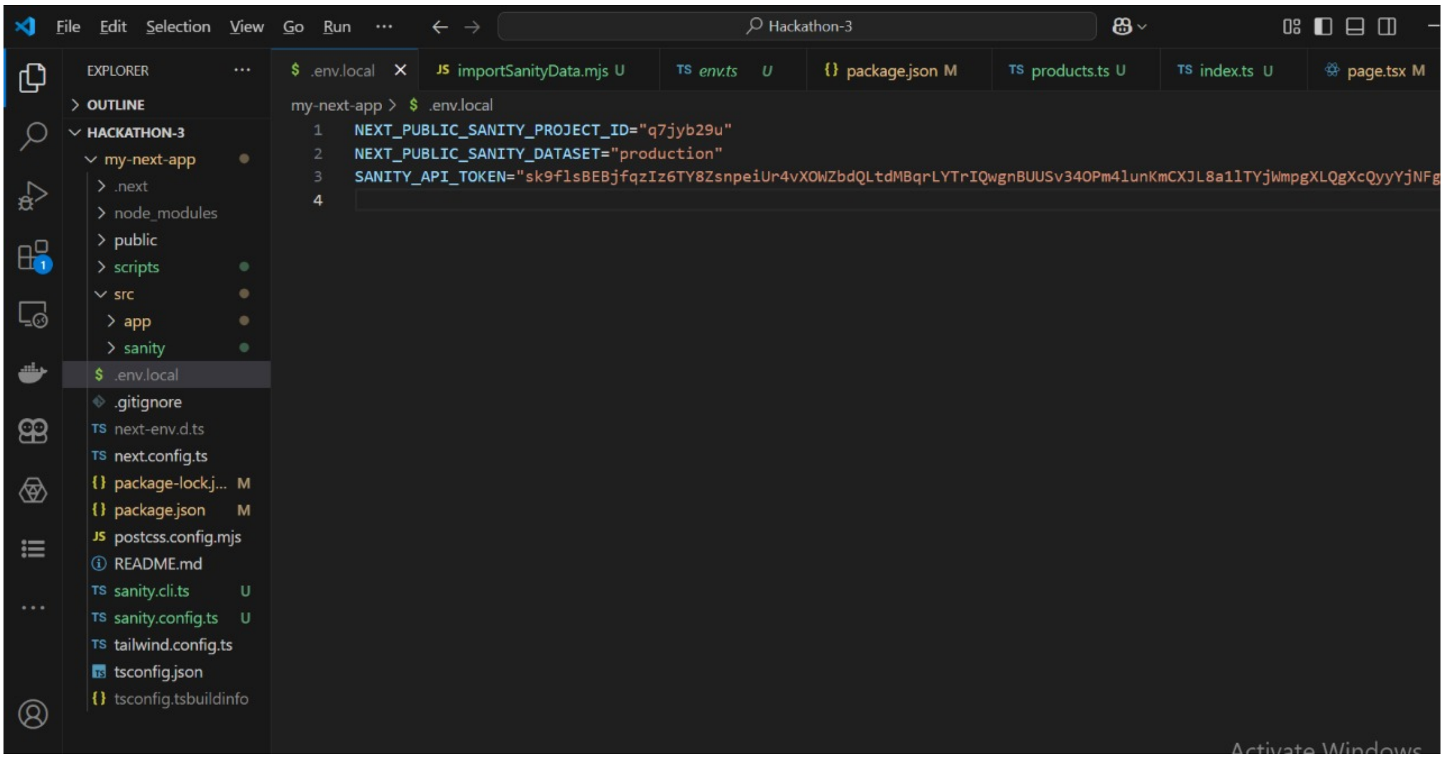
Tokens

Tokens are used to authenticate apps and scripts to access project data.

NAME	PERMISSIONS	CREATED
Hackathon-3	Developer	20 hours

Environment Variable Setup:

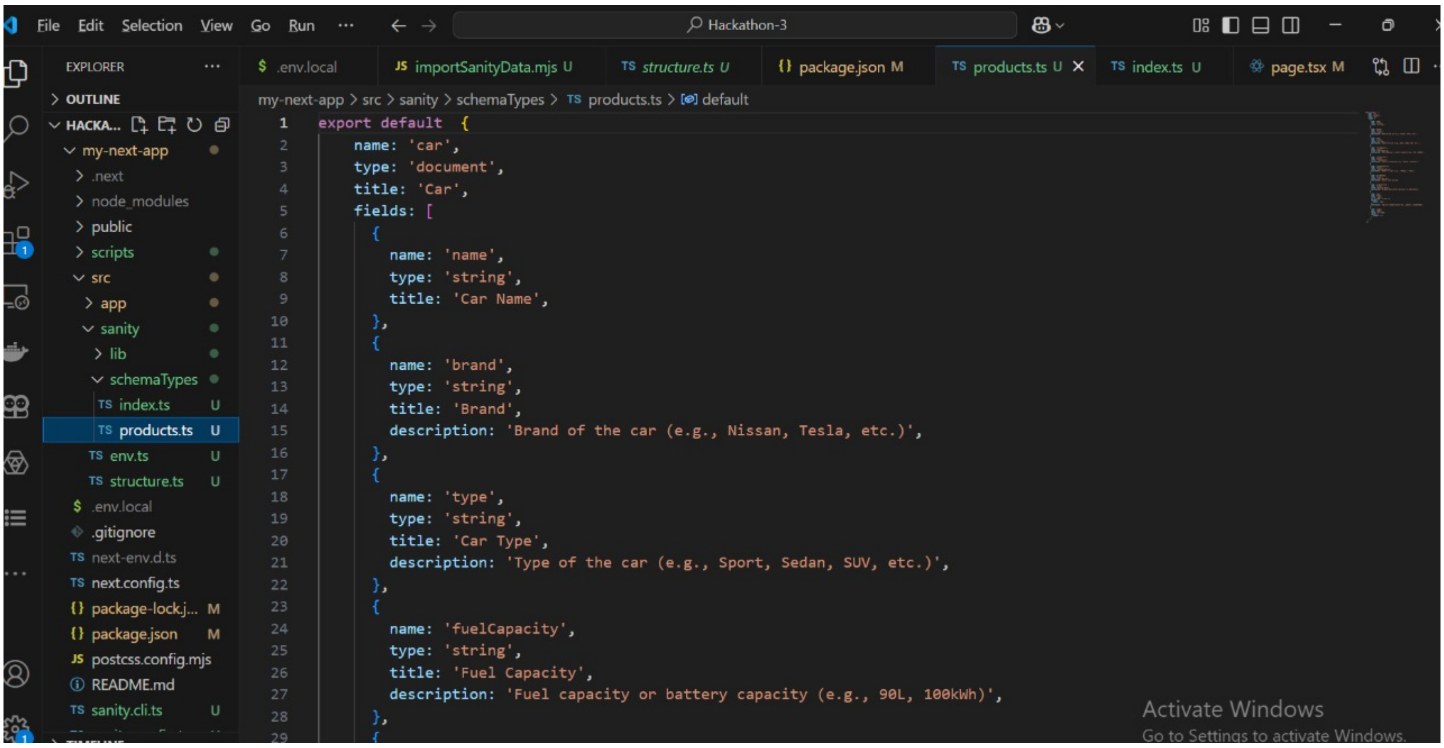
Sensitive credentials such as the Sanity project ID, dataset, and token are securely stored in a .env file.



```
NEXT_PUBLIC_SANITY_PROJECT_ID="q7jyb29u"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="sk9flsBEBjfqzIz6TY8ZsnpeiUr4vXOWZbdQLtdMBqrLYTrIQwgnBUUSv34OPm4lunKmCXJL8a1lTYjWmpgXLQgXcQyyYjNFg
```

Car Schema Definition:

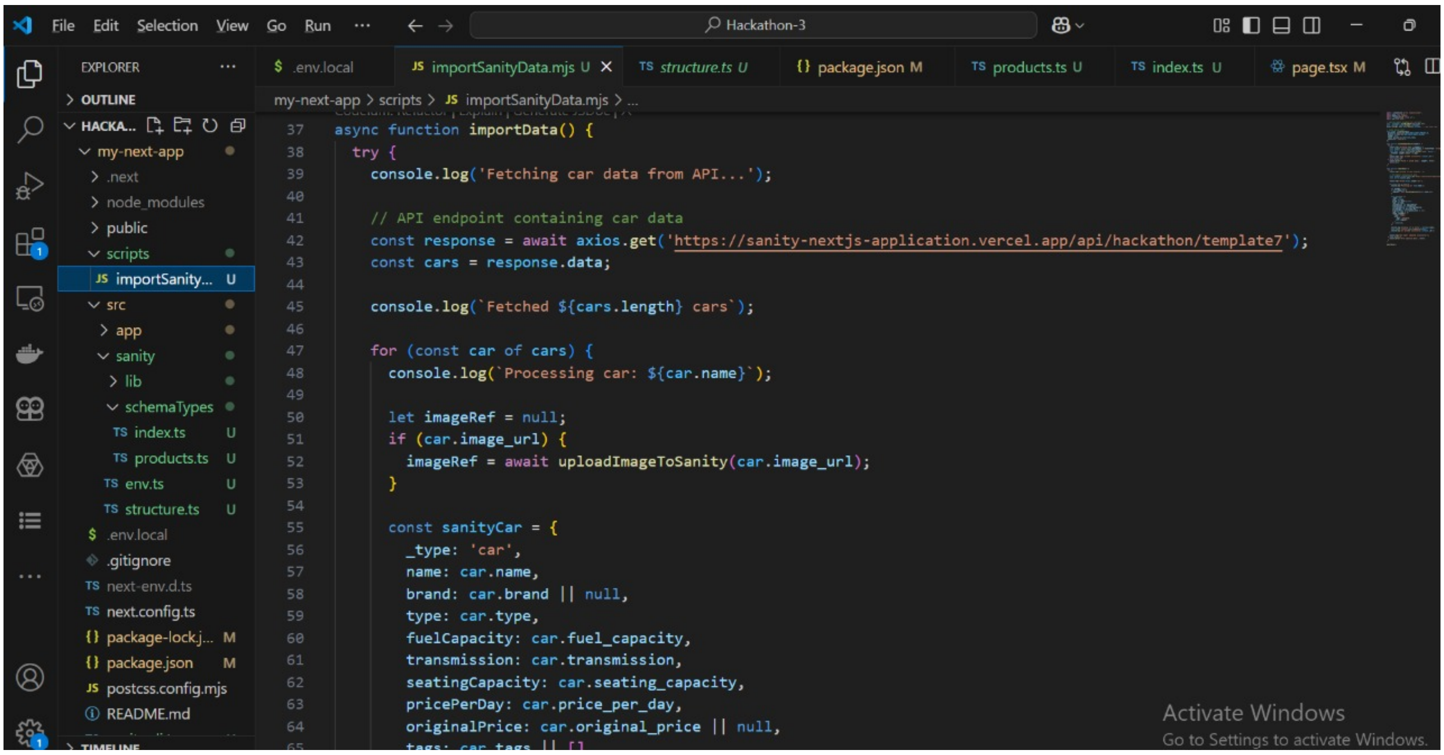
Defines fields in Sanity CMS, such as car name, brand, type, and fuel capacity, ensuring structured data storage.



```
export default  {
    name: 'car',
    type: 'document',
    title: 'Car',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Car Name',
      },
      {
        name: 'brand',
        type: 'string',
        title: 'Brand',
        description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
      },
      {
        name: 'type',
        type: 'string',
        title: 'Car Type',
        description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
      },
      {
        name: 'fuelCapacity',
        type: 'string',
        title: 'Fuel Capacity',
        description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
      },
      {
```

Importing Car Data via API:

This code demonstrates the script used to fetch car data from an external API and upload it into Sanity.



```
async function importData() {
  try {
    console.log('Fetching car data from API...');

    // API endpoint containing car data
    const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
    const cars = response.data;

    console.log(`Fetched ${cars.length} cars`);

    for (const car of cars) {
      console.log(`Processing car: ${car.name}`);

      let imageRef = null;
      if (car.image_url) {
        imageRef = await uploadImageToSanity(car.image_url);
      }

      const sanityCar = {
        _type: 'car',
        name: car.name,
        brand: car.brand || null,
        type: car.type,
        fuelCapacity: car.fuel_capacity,
        transmission: car.transmission,
        seatingCapacity: car.seating_capacity,
        pricePerDay: car.price_per_day,
        originalPrice: car.original_price || null,
        tags: car.tags || []
```

Conclusion

This documentation consolidates the tasks achieved in the Rental Car E-commerce project, including:

- Creating schemas in Sanity CMS for structured car data.
- Configuring API tokens for secure interactions.
- Writing scripts to import car data from external sources into Sanity.
- Setting up environment variables for secure configuration.

These steps establish a scalable and efficient backend architecture for managing car rental data.