

Source Code of App: myapp

admin.py

```
from django.contrib import admin
from .models import Car, CarType

admin.site.register(Car)
admin.site.register(CarType)
```

apps.py

```
from django.apps import AppConfig

class MyAppConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "myapp"
```

forms.py

```
from django import forms
from .models import CustomUser
from .models import Schedule, Car
from .models import CarReview
from .models import Review

class CarForm(forms.ModelForm):
    class Meta:
        model = Car
        fields = '__all__'

class UserUpdateForm(forms.ModelForm):
    class Meta:
        model = CustomUser
        fields = ['first_name', 'last_name', 'nickname', 'age', 'phone_number', 'email', 'address', 'profile_picture']
        widgets = {
            'profile_picture': forms.FileInput(attrs={'class': 'form-control'}),
        }

class ScheduleForm(forms.ModelForm):
    date = forms.DateField(input_formats=('%Y-%m-%d'))

    class Meta:
        model = Schedule
        fields = ['car', 'date', 'time']

class CarReviewForm(forms.ModelForm):
    class Meta:
        model = CarReview
        fields = ['rating', 'review_text']

class ReviewForm(forms.ModelForm):
    class Meta:
        model = CarReview
        fields = ['review_text', 'rating']
```

models.py

```
from django.db import models
from django import forms
from django.contrib.auth.models import AbstractUser, Group, Permission
from django.conf import settings
from django.contrib.auth.models import User

class CarType(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Car(models.Model):
    STATUS_CHOICES = [
        ('Pending', 'Pending'),
        ('Approved', 'Approved'),
    ]
    name = models.CharField(max_length=100)
    description = models.TextField(blank=True, null=True)
    horsepower = models.IntegerField(verbose_name="แรงแม่", default=0)
    car_type = models.ForeignKey(CarType, related_name='cars', on_delete=models.CASCADE)
    image = models.ImageField(upload_to='cars/', blank=True, null=True, default='default_car.jpg')
    status = models.CharField(max_length=20, choices=STATUS_CHOICES)

    owner = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        null=True,
        blank=True,
        related_name='cars'
    )
    is_available = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True, null=True)

    def __str__(self):
        return self.name

class Schedule(models.Model):
    car = models.ForeignKey(Car, on_delete=models.CASCADE, related_name='schedules')
    date = models.DateField() # วันที่จะ
    time = models.CharField()
```

```

        max_length=20,
        choices=[('morning', 'ช่วงเช้า'), ('afternoon', 'ช่วงบ่าย')] # เลือกช่วงเวลา
    )
    is_booked = models.BooleanField(default=False) # ตรวจสอบว่าถูกจองแล้วหรือยัง
    booked_by = models.ForeignKey(
        settings.AUTH_USER_MODEL, # เชื่อมกับผู้ใช้ที่ทำการจอง
        on_delete=models.CASCADE,
        related_name='bookings',
        null=True, blank=True # อนุญาตให้ว่างได้สำหรับการจองที่ยังไม่มีเจ้าของ
    )

    def __str__(self):
        return f"{self.car.name} - {self.date} ({self.get_time_display()})"

class CarForm(forms.ModelForm):
    class Meta:
        model = Car
        fields = ['name', 'description', 'horsepower', 'image']

class CustomUser(AbstractUser):
    nickname = models.CharField(max_length=50, null=True, blank=True)
    age = models.PositiveIntegerField(null=True, blank=True)
    phone_number = models.CharField(max_length=15, null=True, blank=True)
    address = models.TextField(null=True, blank=True)
    profile_picture = models.ImageField(upload_to='profile_pics/', null=True, blank=True)

    # Add related_name to resolve clashes
    groups = models.ManyToManyField(
        Group,
        related_name='customuser_groups',
        blank=True,
        help_text='The groups this user belongs to.',
        verbose_name='groups'
    )
    user_permissions = models.ManyToManyField(
        Permission,
        related_name='customuser_permissions',
        blank=True,
        help_text='Specific permissions for this user.',
        verbose_name='user permissions'
    )

    def __str__(self):
        return self.username

class Notification(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE) # ผู้ที่ได้รับการแจ้งเตือน
    borrower = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='borrower_notifications', null=True, blank=True)
    message = models.TextField() # ข้อความแจ้งเตือน
    timestamp = models.DateTimeField(auto_now_add=True) # เวลาที่แจ้งเตือนถูกสร้าง
    schedule = models.ForeignKey(Schedule, null=True, blank=True, on_delete=models.SET_NULL) # เชื่อมโยงกับการจอง
    is_confirmed = models.BooleanField(default=False) # สถานะการยืนยันการจอง
    is_approved = models.BooleanField(default=False) # สถานะการอนุมัติจอง

    def __str__(self):
        return f"Notification for {self.user.username} at {self.timestamp}"

class CarReview(models.Model):
    car = models.ForeignKey('Car', on_delete=models.CASCADE, related_name='reviews')
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE) # แกะไขที่นี้
    rating = models.IntegerField(choices=[(i, i) for i in range(1, 6)]) # Rating 1-5
    review_text = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Review for {self.car.name} by {self.user.username}"

class Review(models.Model):
    car = models.ForeignKey(Car, on_delete=models.CASCADE)
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE) # แกะไขที่นี้
    comment = models.TextField()

    def __str__(self):
        return f"Review for {self.car.name} by {self.user.username}"

```

tests.py

```

from django.test import TestCase

# Create your tests here.

```

urls.py

```

from django.urls import path
from . import views
from .views import notification_list
from .views import user_management, delete_user
from .views import delete_car
from .views import user_reservations
from .views import cancel_reservation

urlpatterns = [

    path('export-code/', views.app_to_pdf, name='export_code'),

    path('', views.welcome_view, name='welcome'), # หน้าแรก
    path('login/', views.login_view, name='login'), # หน้าเข้าสู่ระบบ
    path('register/', views.register_view, name='register'), # หน้าไปยังหน้าลงทะเบียน
    path('logout/', views.logout_view, name='logout'), # ล็อกเอาท์
    path('cars/types/', views.car_type_list_view, name='car_type_list'), # แสดงประเภทของรถ
    path('cars/type/', views.car_list_by_type_view, name='car_list_by_type'), # แสดงรถในแต่ละประเภท
    path('cars/', views.car_detail_view, name='car_detail'), # แสดงรายละเอียดของรถ
    path('edicar/', views.edicar, name='edicar'), # แกะไขข้อมูลรถ

```

```

path('delcar/', views.delcar, name='delcar'), # ลบรถ
path('delete_car/', delete_car, name='delete_car'), # ลบรถ
path('car/schedule/', views.car_schedule, name='car_schedule'), # ตารางงานของรถ
path('approval_list/', views.car_approval_list, name='car_approval_list'), # แสดงรายการรถที่รออนุมัติ
path('approve_car/', views.approve_car, name='approve_car'), # อนุมัติรถ
path('addcar/', views.add_car, name='addcar'), # เพิ่มรถใหม่
path('pending-cars/', views.pending_car_list, name='pending_car_list'), # รถที่รอการอนุมัติ
path('delete-car/', views.delete_car, name='delete_car'), # ลบรถ
path('profile/', views.profile_view, name='profile'), # ดูโปรไฟล์
path('edit-profile/', views.edit_profile_view, name='edit_profile'), # แก้ไขโปรไฟล์
path('cars/toggle_status/', views.toggle_car_status, name='toggle_car_status'), # เปลี่ยนสถานะของรถ
path('my-cars/', views.my_cars, name='my_cars'), # เส้นทางสำหรับแสดงข้อมูลรถของผู้ใช้
path('confirm_selection/', views.confirm_selection, name='confirm_selection'), # ยืนยันการเลือกเวลา
path('edit_car/', views.edicar, name='edit_car'), # แก้ไขรถ
path('create_booking/', views.create_booking, name='create_booking'), # สร้างการจอง
path('confirm_reservation/', views.confirm_reservation, name='confirm_reservation'), # ยืนยันการจอง
path('notification_list/', views.notification_list, name='notification_list'), # แสดงรายการการจองเดือน
path('car_list/', views.car_list_view, name='car_list'), # แสดงรายการรถทั้งหมด
path('cars/', views.car_list_view, name='car_list'), # เส้นทางแสดงรถในแต่ละประเภท

path('cars/reviews/', views.car_review_view, name='car_review'),

path('notification_list/', notification_list, name='notification_list'),

path('users/', user_management, name='user_management'),
path('users/delete/', delete_user, name='delete_user'),

path('cancel_reservation/', views.cancel_reservation, name='cancel_reservation'), # เส้นทางสำหรับยกเลิกการจอง
path('my_reservations/', views.user_reservations, name='user_reservations'), # เส้นทางสำหรับดูการจองของผู้ใช้

```

]

views.py

```

from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .models import *
from django.http import HttpResponseRedirect
from django.urls import reverse
from .forms import CarForm, UserUpdateForm
from django.contrib.auth.decorators import user_passes_test, login_required
from django.http import JsonResponse
from .models import Car, Notification, Schedule
from datetime import datetime
from django.contrib.admin.views.decorators import staff_member_required
from myapp.models import CustomUser
from django.db.models import Q
from .models import Car, Review
from .forms import ReviewForm

import os
import pdfkit
from django.http import HttpResponseRedirect

# กำหนด path ของ wkhtmltopdf (Windows ใช้ path นี้, Linux/Mac อาจไม่ต้อง)
PDFKIT_CONFIG = pdfkit.configuration(wkhtmltopdf=r"C:\Program Files\wkhtmltopdf\bin\wkhtmltopdf.exe")

def app_to_pdf(request, app_name):
    app_path = os.path.join(settings.BASE_DIR, app_name)
    pdf_filename = f"{app_name}.pdf"

    if not os.path.exists(app_path):
        return HttpResponseRedirect("App not found", status=404)

    html_content = f"

```

Source Code of App: {app_name}

```

"

for root, dirs, files in os.walk(app_path):
    for file in files:
        if file.endswith((".py", ".html", ".css", ".js")): # แปลงเฉพาะไฟล์ที่ต้องการ
            file_path = os.path.join(root, file)
            with open(file_path, "r", encoding="utf-8") as f: # อ่านไฟล์เป็น UTF-8
                html_content += f"

{file}

{f.read()}

"

# ใช้ pdfkit พร้อมกำหนดให้ใช้ encoding UTF-8
options = {'encoding': 'UTF-8'}
pdfkit.from_string(html_content, pdf_filename, configuration=PDFKIT_CONFIG, options=options)

with open(pdf_filename, "rb") as pdf:
    response = HttpResponseRedirect(pdf.read(), content_type="application/pdf")
    response["Content-Disposition"] = f'attachment; filename="{pdf_filename}"'
    return response

@staff_member_required # ให้เฉพาะแอดมินเข้าถึงได้
def user_management(request):
    users = User.objects.all()
    return render(request, 'user_management.html', {'users': users})

@staff_member_required
def delete_user(request, user_id):
    user = get_object_or_404(User, id=user_id)

    if user.is_staff:

```

```

        messages.error(request, "ไม่สามารถลบผู้ดูแลระบบได้")
    else:
        user.delete()
        messages.success(request, "ลบผู้ใช้งานเรียบร้อยแล้ว")

    return redirect('user_management')

# ฟังก์ชันสำหรับหน้าแรก
def welcome_view(request):
    """
    แสดงหน้า Welcome (หน้าแรกของระบบ)
    """
    return render(request, 'welcome.html')

# ฟังก์ชันสำหรับหน้าล็อกอิน
def login_view(request):
    """
    จัดการการเข้าสู่ระบบของผู้ใช้:
    - รับ username และ password
    - ตรวจสอบว่าถูกต้องหรือไม่
    - หากสำเร็จ จะเปลี่ยนไปยังหน้า car_type_list
    """
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('car_type_list')
        else:
            messages.error(request, 'Invalid username or password')

    return render(request, 'login.html')

# ฟังก์ชันสำหรับหน้าสมัคร
def register_view(request):
    """
    สมัครสมาชิกใหม่:
    - ตรวจสอบความถูกต้องของข้อมูล
    - สร้างผู้ใช้ใหม่
    - ส่งข้อความแจ้งเตือนความสำเร็จหรือข้อผิดพลาด
    """
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']
        if password == confirm_password:
            if CustomUser.objects.filter(username=username).exists():
                messages.error(request, 'Username already exists')
            elif CustomUser.objects.filter(email=email).exists():
                messages.error(request, 'Email already exists')
            else:
                user = CustomUser.objects.create_user(username=username, email=email, password=password)
                user.save()
                messages.success(request, 'Account created successfully')
                return redirect('login')
        else:
            messages.error(request, 'Passwords do not match')

    return render(request, 'register.html')

# ฟังก์ชันสำหรับการล็อกเอาท์
def logout_view(request):
    """
    ออกจากระบบ:
    - เคลียร์ session ของผู้ใช้
    """
    logout(request)
    return redirect('login')

# ฟังก์ชันแสดงรายการประเภทของรถ
def car_type_list_view(request):
    """
    แสดงรายการประเภทของรถ
    """
    car_types = CarType.objects.all()
    return render(request, 'car_type_list.html', {'car_types': car_types})

# ฟังก์ชันแสดงรายการรถ
def car_list_view(request):
    cars = Car.objects.all()
    return render(request, 'car_list.html', {'cars': cars})

# ฟังก์ชันแสดงรถในแต่ละประเภท
def car_list_by_type(request, type_id):
    """
    แสดงรายการรถที่อยู่ในประเภทที่กำหนด
    """
    cars = Car.objects.filter(
        car_type_id=type_id,
        status='Approved'
    ).exclude(image_isnull=True).exclude(image='') # กรองรถที่ไม่มีไฟล์ภาพ
    return render(request, 'car_list.html', {'cars': cars, 'car_type': get_object_or_404(CarType, id=type_id)})

# ฟังก์ชันแสดงรายละเอียดของรถแต่ละคัน
def car_detail_view(request, car_id):
    """
    แสดงรายละเอียดของรถแต่ละคัน
    """
    car = get_object_or_404(Car, pk=car_id)
    return render(request, 'car_detail.html', {'car': car})

# ฟังก์ชันเพิ่มรถ
def add_car(request, type_id):
    car_type = get_object_or_404(CarType, id=type_id) # ดึงข้อมูลประเภทของรถ
    if request.method == 'POST':
        form = CarForm(request.POST, request.FILES)
        if form.is_valid():
            car = form.save(commit=False)

```

```

        car.car_type = CarType.objects.get(id=request.POST.get('car_type')) # ใช้ค่า car_type ที่มาจากฟอร์ม
        car.status = 'Pending' # ตั้งสถานะเป็น 'Pending' สำหรับการอนุมัติ
        car.save()
        return redirect('car_list_by_type', type_id=car.car_type.id) # รีไดเรกไปยังหน้ารายการของประเภทนั้น
    else:
        form = CarForm()

    return render(request, 'addcar.html', {'form': form, 'car_type': car_type})

# ฟังก์ชันแก้ไขรถ
from django.shortcuts import redirect

@login_required
def edicar(request, id, type_id):
    car = get_object_or_404(Car, id=id)

    if request.user != car.owner and not request.user.is_staff:
        return HttpResponseForbidden("คุณไม่มีสิทธิ์ในการแก้ไขรถคันนี้")

    if request.method == 'POST':
        form = CarForm(request.POST, instance=car)
        if form.is_valid():
            form.save()
            messages.success(request, "แก้ไขข้อมูลรถสำเร็จ!")
            return redirect('car_list_by_type', type_id=type_id)
    else:
        form = CarForm(instance=car)

    return render(request, 'edit_car.html', {'form': form, 'car': car})

# ฟังก์ชันลบรถ
@login_required
def delcar(request, car_id, type_id):
    car = get_object_or_404(Car, id=car_id)

    # ตรวจสอบว่าเจ้าของรถหรือแอดมินหรือไม่
    if request.user != car.owner and not request.user.is_staff:
        return HttpResponseForbidden("คุณไม่มีสิทธิ์ในการลบรถคันนี้")

    # ถ้าเจ้าของหรือแอดมิน ยืนยันการลบ
    car.delete()
    return redirect('car_list') # เปลี่ยนไปหน้าแสดงรายการรถ

# ฟังก์ชันแสดงตารางงานของรถ
def car_schedule(request, car_id):
    # กรองรายการที่ได้รับการยืนยันแล้ว
    schedules = Schedule.objects.filter(car_id=car_id, is_booked=True)

    return render(request, 'car_schedule.html', {'schedules': schedules})

def confirm_selection(request):
    selected_date = request.GET.get('date')
    selected_time = request.GET.get('time')
    car_id = request.GET.get('car_id')

    if selected_date and selected_time and car_id:
        car = Car.objects.get(id=car_id)
        return render(request, 'confirm_selection.html', {
            'selected_date': selected_date,
            'selected_time': selected_time,
            'car': car,
        })
    else:
        return render(request, 'error.html', {'message': 'ข้อมูลไม่ครบถ้วน'})

# แสดงรายการรถที่รอการอนุมัติ
def car_approval_list(request):
    # ดึงเฉพาะรถที่รอการอนุมัติ
    cars_pending = Car.objects.filter(status='Pending')

    # ส่งข้อมูล car_type ไปพร้อมกับ cars
    return render(request, 'car_approval_list.html', {'cars': cars_pending})

# ฟังก์ชันสำหรับอนุมัติรถ
def approve_car(request, car_id):
    if not request.user.is_staff:
        return redirect('car_type_list')

    car = get_object_or_404(Car, id=car_id)
    car.status = 'Approved' # อัปเดตสถานะเป็น Approved
    car.save() # บันทึกการเปลี่ยนแปลง
    return redirect('car_list_by_type', type_id=car.car_type.id) # กลับไปหน้ารายการรถ

# แสดงรายการรถที่อนุมัติแล้ว
def car_list_by_type_view(request, type_id):
    # ดึงข้อมูล CarType โดยใช้ type_id
    car_type = get_object_or_404(CarType, id=type_id)

    # ใช้ car_type.cars.all() เพื่อดึงทั้งหมดที่เกี่ยวข้องกับ car_type นี้
    cars = car_type.cars.filter(status='Approved').exclude(image__isnull=True).exclude(image='')

    return render(request, 'car_list.html', {'cars': cars, 'car_type': car_type})

def pending_car_list(request):
    cars = Car.objects.filter(status='Pending').exclude(image__isnull=True).exclude(image='')
    return render(request, 'pending_cars.html', {'cars': cars})

# เช็คค่า user เป็น staff (admin) หรือไม่
def is_admin(user):
    return user.is_staff

@user_passes_test(is_admin)
def pending_car_list(request):
    pending_cars = Car.objects.filter(status='Pending') # รถที่ยังไม่ได้อนุมัติ
    return render(request, 'pending_car_list.html', {'cars': pending_cars})

```

```
@login_required
def delete_car(request, car_id):
    car = get_object_or_404(Car, id=car_id)

    # ตรวจสอบว่าเป็นเจ้าของรถหรือเป็นแอดมิน
    if request.user != car.owner and not request.user.is_staff:
        return JsonResponse({'success': False, 'error': "คุณไม่มีสิทธิ์ลบรถคันนี้"}, status=403)

    car_type_id = car.car_type.id # ดึงประเภทของรถไว้ก่อนลบ
    car.delete()

    return JsonResponse({'success': True, 'redirect_url': f'/cars/type/{car_type_id}/'})

@login_required
def profile_view(request):
    user = CustomUser.objects.filter(id=request.user.id)
    return render(request, 'profile.html', {'user': request.user})

@login_required
def edit_profile_view(request):
    if request.method == 'POST':
        form = UserUpdateForm(request.POST, request.FILES, instance=request.user)
        if form.is_valid():
            form.save()
            messages.success(request, 'โปรไฟล์ถูกอัปเดตเรียบร้อยแล้ว!')
            return redirect('profile')
    else:
        form = UserUpdateForm(instance=request.user)
    return render(request, 'edit_profile.html', {'form': form})

def car_detail_view(request, car_id):
    # ดึงข้อมูลรถตาม car_id
    car = get_object_or_404(Car, id=car_id)

    # ดึงข้อมูลเจ้าของรถ
    owner = car.owner # เจ้าของรถ (ผู้ใช้)

    # ส่งข้อมูลไปที่เทมเพลต
    context = {
        'car': car,
        'owner': owner, # ส่งข้อมูลเจ้าของรถไปที่เทมเพลต
    }

    return render(request, 'car_detail.html', context)

# ฟังก์ชันสำหรับเปลี่ยนสถานะของรถ
def toggle_car_status(request, car_id):
    # ดึงข้อมูลรถจากฐานข้อมูลโดยใช้ car_id
    car = get_object_or_404(Car, id=car_id)

    # ตรวจสอบว่าเจ้าของรถเป็นผู้ที่ล็อกอินอยู่หรือไม่
    if car.owner == request.user:
        # เปลี่ยนสถานะของรถ (พร้อมใช้งาน -> ไม่พร้อมใช้งาน)
        car.is_available = not car.is_available
        car.save()

    # หลังจากเปลี่ยนสถานะแล้ว ให้ย้อนกลับไปยังหน้ารายละเอียดของรถ
    return redirect('car_detail', car_id=car.id)

def my_cars(request):
    # ดึงข้อมูลรถที่เจ้าของเป็นผู้ใช้ปัจจุบัน
    cars = Car.objects.filter(owner=request.user)

    # ส่งข้อมูลรถไปยัง template
    return render(request, 'my_cars.html', {'cars': cars})

# ฟังก์ชันสำหรับการยืนยันการจอง
@login_required
def confirm_reservation(request, reservation_id):
    schedule = get_object_or_404(Schedule, id=reservation_id)

    if request.method == 'POST':
        action = request.POST.get('action')

        if action == 'approve':
            # อัปเดตสถานะให้เป็นยืนยัน
            schedule.is_booked = True
            schedule.save()

            # แจ้งเตือนผู้จองว่าการจองได้รับการยืนยัน
            Notification.objects.create(
                user=schedule.booked_by,
                message=f"เจ้าของรถ {schedule.car.name} ได้ยืนยันการจองของคุณ วันที่ {schedule.date} ช่วง {schedule.get_time_display()}",
                schedule=schedule,
                is_confirmed=True,
                is_approved=True
            )

        elif action == 'reject':
            # ลบการจองหากถูกปฏิเสธ
            schedule.delete()

            # แจ้งเตือนผู้จองว่าการจองถูกปฏิเสธ
            Notification.objects.create(
                user=schedule.booked_by,
                message=f"เจ้าของรถ {schedule.car.name} ปฏิเสธการจองของคุณ วันที่ {schedule.date} ช่วง {schedule.get_time_display()}",
                is_confirmed=True,
                is_approved=False
            )

    return redirect('notification_list') # กลับไปที่แจ้งเตือน

# ฟังก์ชันการสร้างการจอง
@login_required
def create_booking(request, car_id):
    car = get_object_or_404(Car, id=car_id)
```

```

if request.method == 'POST':
    selected_date = request.POST.get('date')
    selected_time = request.POST.get('time')

    if selected_date and selected_time:
        # ตรวจสอบว่าช่วงเวลาที่ถูกจองไปแล้วหรือไม่
        existing_booking = Schedule.objects.filter(car=car, date=selected_date, time=selected_time).exists()
        if existing_booking:
            messages.error(request, "ช่วงเวลาที่ถูกจองแล้ว กรุณาเลือกช่วงเวลาอื่น")
            return redirect('car_detail', car_id=car.id)

        # สร้างการจองใหม่
        schedule = Schedule.objects.create(
            car=car,
            date=selected_date,
            time=selected_time,
            is_booked=False, # ต้องให้เจ้าของรถกดยืนยันก่อน
            booked_by=request.user
        )

        Notification.objects.create(
            user=car.owner, # เจ้าของรถ (ผู้รับการแจ้งเตือน)
            borrower=request.user, # ผู้ยืม
            message=f"คุณมีการจองรถ {car.name} จาก {request.user.username} สำหรับวันที่ {selected_date} ช่วง {schedule.get_time_display()} โปรดตรวจสอบและอนุมัติ",
            schedule=schedule
        )

        messages.success(request, "ส่งค่าการจองไปยังเจ้าของรถแล้ว และคุณจะได้รับการแจ้งเตือนเมื่อมีการตอบกลับ")
        return redirect('notification_list') # ไปยังหน้าการแจ้งเตือน
    else:
        messages.error(request, "กรุณาเลือกวันที่และช่วงเวลาให้ครบถ้วน")

return render(request, 'book_time.html', {'car': car})

@login_required
def notification_list(request):
    # ดึงการแจ้งเตือนที่ผู้ใช้ได้รับ
    notifications_as_user = Notification.objects.filter(user=request.user).order_by('-timestamp')

    # ดึงการแจ้งเตือนที่ผู้ใช้เป็นผู้ยืม
    notifications_as_borrower = Notification.objects.filter(borrower=request.user).order_by('-timestamp')

    # รวมทั้งสองรายการ
    notifications = Notification.objects.filter(
        Q(user=request.user) | # เจ้าของรถ
        Q(borrower=request.user) # ผู้ยืม
    ).order_by('-timestamp')

    return render(request, 'notification_list.html', {
        'notifications': notifications,
        'is_owner': Car.objects.filter(owner=request.user).exists() # ตรวจสอบว่าเป็นเจ้าของรถหรือไม่
    })

@login_required
def cancel_reservation(request, reservation_id):
    reservation = get_object_or_404(Schedule, id=reservation_id, booked_by=request.user)

    if reservation.is_booked:
        # ตั้งสถานะให้ว่าง
        reservation.is_booked = False
        reservation.booked_by = None
        reservation.save()

        # แจ้งเตือนเจ้าของรถ
        Notification.objects.create(
            user=reservation.car.owner,
            message=f"การจองรถ {reservation.car.name} วันที่ {reservation.date} ของ {request.user.username} ถูกยกเลิกแล้ว",
            schedule=reservation,
            is_confirmed=True, # ถือว่าได้รับการยืนยันแล้ว
            is_approved=False # แสดงว่าการจองถูกยกเลิก
        )

        # แจ้งเตือนผู้จองว่าการจองของพวกเขาถูกยกเลิก
        Notification.objects.create(
            user=request.user,
            message=f"การจองรถ {reservation.car.name} วันที่ {reservation.date} ถูกยกเลิกแล้ว",
            schedule=reservation,
            is_confirmed=True,
            is_approved=False
        )

        messages.success(request, "ยกเลิกการจองสำเร็จ")
    else:
        messages.error(request, "ไม่สามารถยกเลิกการจองนี้ได้")

    return redirect('user_reservations') # กลับไปหน้ารายการจองของผู้ใช้

def car_review_view(request, car_id):
    car = get_object_or_404(Car, id=car_id)
    reviews = CarReview.objects.filter(car=car)

    if request.method == "POST":
        form = ReviewForm(request.POST)
        if form.is_valid():
            review = form.save(commit=False)
            review.car = car
            review.user = request.user # กำหนดผู้ใช้ที่ส่งรีวิว
            review.save()
            return redirect('car_review', car_id=car.id) # รีไดเร็กไปหน้ารีวิวหลังจากบันทึกเสร็จ
    else:
        form = ReviewForm()

    return render(request, 'car_review.html', {'car': car, 'reviews': reviews, 'form': form})

@staff_member_required # ให้เฉพาะแอดมินเข้าถึงได้
def user_management(request):

```

```

users = CustomUser.objects.all() # ๒ เปลี่ยนจาก User.objects.all() เป็น CustomUser.objects.all()
return render(request, 'user_management.html', {'users': users})

@staff_member_required
def delete_user(request, user_id):
    user = get_object_or_404(CustomUser, id=user_id) # ๒ ใช้ CustomUser แทน User

    if user.is_staff:
        messages.error(request, "ไม่สามารถลบผู้ดูแลระบบได้")
    else:
        user.delete()
        messages.success(request, "ลบผู้ใช้งานเรียบร้อยแล้ว")

    return redirect('user_management')

from .models import Schedule # เปลี่ยนจาก Reservation เป็น Schedule

@login_required
def user_reservations(request):
    reservations = Schedule.objects.filter(booked_by=request.user)
    return render(request, 'user_reservations.html', {'reservations': reservations})

```

__init__.py

load_data.py

```

# your_app/management/commands/generate_dbml.py
from django.core.management.base import BaseCommand
from django.apps import apps
from django.db import models

class Command(BaseCommand):
    help = 'Generate DBML file from Django models'

    def handle(self, *args, **kwargs):
        # สร้างตัวแปรสำหรับไฟล์ DBML
        dbml_content = ""

        # ดึงโมเดลทั้งหมดจากทุกแอปในโปรเจกต์
        models_list = []
        for app_config in apps.get_app_configs():
            models_list.extend(app_config.get_models())

        # สร้าง DBML content สำหรับแต่ละตาราง
        for model in models_list:
            table_name = model._meta.model_name
            dbml_content += f"Table {table_name} {\n"

            # สร้างคอมเมนต์ตามฟิลด์ในโมเดล
            for field in model._meta.fields:
                field_name = field.name
                field_type = field.get_internal_type().lower()
                if isinstance(field, models.CharField):
                    dbml_content += f"    {field_name} varchar\n"
                elif isinstance(field, models.TextField):
                    dbml_content += f"    {field_name} text\n"
                elif isinstance(field, models.IntegerField):
                    dbml_content += f"    {field_name} integer\n"
                elif isinstance(field, models.DateTimeField):
                    dbml_content += f"    {field_name} timestamp\n"
                elif isinstance(field, models.ForeignKey):
                    related_model = field.related_model
                    dbml_content += f"    {field_name}_id integer [note: 'foreign key to {related_model._meta.model_name}']\n"

            dbml_content += "}\n\n"

        # สร้างความสัมพันธ์ระหว่างตาราง
        for model in models_list:
            for field in model._meta.fields:
                if isinstance(field, models.ForeignKey):
                    from_table = model._meta.model_name
                    to_table = field.related_model._meta.model_name
                    dbml_content += f"Ref: {from_table}.{field.name}_id > {to_table}.id // many-to-one\n"

        # เขียนข้อมูล DBML ลงไฟล์
        with open('generated_schema.dbml', 'w') as file:
            file.write(dbml_content)

        self.stdout.write(self.style.SUCCESS('Successfully generated DBML file'))

```

0001_initial.py

Generated by Django 5.1.1 on 2024-10-17 08:43

```

import django.db.models.deletion
from django.db import migrations, models

```

```

class Migration(migrations.Migration):

    initial = True

    dependencies = []

    operations = [
        migrations.CreateModel(
            name="CarType",

```



```

        fields=[
            (
                "id",
                models.BigAutoField(
                    auto_created=True,
                    primary_key=True,
                    serialize=False,
                    verbose_name="ID",
                ),
            ),
            ("name", models.CharField(max_length=100)),
        ],
    ),
    migrations.CreateModel(
        name="Car",
        fields=[
            (
                "id",
                models.BigAutoField(
                    auto_created=True,
                    primary_key=True,
                    serialize=False,
                    verbose_name="ID",
                ),
            ),
            ("name", models.CharField(max_length=100)),
            ("description", models.TextField()),
            ("horsepower", models.IntegerField()),
            (
                "image",
                models.ImageField(blank=True, null=True, upload_to="car_images/"),
            ),
            (
                "car_type",
                models.ForeignKey(
                    on_delete=django.db.models.deletion.CASCADE, to="myapp.cartype"
                ),
            ),
        ],
    ),
]

```

0002_alter_car_image.py

Generated by Django 5.1.1 on 2024-10-17 10:43

from django.db import migrations, models

```

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0001_initial"),
    ]

    operations = [
        migrations.AlterField(
            model_name="car",
            name="image",
            field=models.ImageField(blank=True, null=True, upload_to="car_images"),
        ),
    ]

```

0003_alter_car_image_schedule.py

Generated by Django 5.1.1 on 2024-12-09 09:15

import django.db.models.deletion
from django.db import migrations, models

```

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0002_alter_car_image"),
    ]

    operations = [
        migrations.AlterField(
            model_name="car",
            name="image",
            field=models.ImageField(blank=True, null=True, upload_to="car_images/"),
        ),
        migrations.CreateModel(
            name="Schedule",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("date", models.DateField()),
                ("detail", models.TextField()),
                (
                    "car",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        related_name="schedules",
                        to="myapp.car",
                    ),
                ),
            ],
        ),
    ]

```

0004_remove_car_horsepower_remove_schedule_detail_and_more.py

Generated by Django 5.1.1 on 2024-12-09 10:09

from django.db import migrations, models

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0003_alter_car_image_schedule"),
    ]

    operations = [
        migrations.RemoveField(
            model_name="car",
            name="horsepower",
        ),
        migrations.RemoveField(
            model_name="schedule",
            name="detail",
        ),
        migrations.AddField(
            model_name="schedule",
            name="is_booked",
            field=models.BooleanField(default=False),
        ),
        migrations.AlterField(
            model_name="car",
            name="description",
            field=models.TextField(blank=True, null=True),
        ),
        migrations.AlterField(
            model_name="car",
            name="image",
            field=models.ImageField(blank=True, null=True, upload_to="cars/"),
        ),
    ]
```

0005_car_horsepower.py

Generated by Django 5.1.1 on 2024-12-11 07:44

from django.db import migrations, models

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0004_remove_car_horsepower_remove_schedule_detail_and_more"),
    ]

    operations = [
        migrations.AddField(
            model_name="car",
            name="horsepower",
            field=models.IntegerField(default=0, verbose_name="แรงแม่"),
        ),
    ]
```

0006_schedule_time.py

Generated by Django 5.1.1 on 2024-12-12 07:54

from django.db import migrations, models

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0005_car_horsepower"),
    ]

    operations = [
        migrations.AddField(
            model_name="schedule",
            name="time",
            field=models.CharField(
                choices=[("morning", "ช่วงเช้า"), ("afternoon", "ช่วงบ่าย")],
                default=0,
                max_length=10,
            ),
        ),
    ]
```

0007_car_status_alter_schedule_time.py

Generated by Django 5.1.1 on 2024-12-15 16:42

from django.db import migrations, models

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0006_schedule_time"),
    ]

    operations = [
        migrations.AddField(
            model_name="car",
            name="status",
            field=models.CharField(
                choices=[("Pending", "Pending"), ("Approved", "Approved")],
                default="Pending",
                max_length=10,
            ),
        ),
    ]
```

```

    ),
    migrations.AlterField(
        model_name="schedule",
        name="time",
        field=models.CharField(
            choices=[("morning", "เช้านี้"), ("afternoon", "บ่ายนี้")],
            default=0,
            max_length=20,
        ),
    ),
]

```

0008_alter_car_image.py

Generated by Django 5.1.1 on 2024-12-16 08:01

from django.db import migrations, models

```

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0007_car_status_alter_schedule_time"),
    ]

    operations = [
        migrations.AlterField(
            model_name="car",
            name="image",
            field=models.ImageField(
                blank=True, default="default_car.jpg", null=True, upload_to="cars/"
            ),
        ),
    ]

```

0009_customuser.py

Generated by Django 5.1.1 on 2024-12-17 12:57

import django.contrib.auth.models
import django.contrib.auth.validators
import django.utils.timezone
from django.db import migrations, models

```

class Migration(migrations.Migration):

    dependencies = [
        ("auth", "0012_alter_user_first_name_max_length"),
        ("myapp", "0008_alter_car_image"),
    ]

    operations = [
        migrations.CreateModel(
            name="CustomUser",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("password", models.CharField(max_length=128, verbose_name="password")),
                (
                    "last_login",
                    models.DateTimeField(
                        blank=True, null=True, verbose_name="last login"
                    ),
                ),
                (
                    "is_superuser",
                    models.BooleanField(
                        default=False,
                        help_text="Designates that this user has all permissions without explicitly assigning them.",
                        verbose_name="superuser status",
                    ),
                ),
                (
                    "username",
                    models.CharField(
                        error_messages={
                            "unique": "A user with that username already exists."
                        },
                        help_text="Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.",
                        max_length=150,
                        unique=True,
                        validators=[
                            django.contrib.auth.validators.UnicodeUsernameValidator()
                        ],
                        verbose_name="username",
                    ),
                ),
                (
                    "first_name",
                    models.CharField(
                        blank=True, max_length=150, verbose_name="first name"
                    ),
                ),
                (
                    "last_name",
                    models.CharField(
                        blank=True, max_length=150, verbose_name="last name"
                    ),
                ),
                (
                    "email",

```

```

        models.EmailField(
            blank=True, max_length=254, verbose_name="email address"
        ),
    ),
    (
        "is_staff",
        models.BooleanField(
            default=False,
            help_text="Designates whether the user can log into this admin site.",
            verbose_name="staff status",
        ),
    ),
    (
        "is_active",
        models.BooleanField(
            default=True,
            help_text="Designates whether this user should be treated as active. Unselect this instead of deleting accounts.",
            verbose_name="active",
        ),
    ),
    (
        "date_joined",
        models.DateTimeField(
            default=django.utils.timezone.now, verbose_name="date joined"
        ),
    ),
    ("nickname", models.CharField(blank=True, max_length=50, null=True)),
    ("age", models.PositiveIntegerField(blank=True, null=True)),
    (
        "phone_number",
        models.CharField(blank=True, max_length=15, null=True),
    ),
    ("address", models.TextField(blank=True, null=True)),
    (
        "profile_picture",
        models.ImageField(blank=True, null=True, upload_to="profile_pics/"),
    ),
    (
        "groups",
        models.ManyToManyField(
            blank=True,
            help_text="The groups this user belongs to.",
            related_name="customuser_groups",
            to="auth.group",
            verbose_name="groups",
        ),
    ),
    (
        "user_permissions",
        models.ManyToManyField(
            blank=True,
            help_text="Specific permissions for this user.",
            related_name="customuser_permissions",
            to="auth.permission",
            verbose_name="user permissions",
        ),
    ),
],
options={
    "verbose_name": "user",
    "verbose_name_plural": "users",
    "abstract": False,
},
managers=[
    ("objects", django.contrib.auth.models.UserManager()),
],
),
]

```

0010_add_owner_field.py

Generated by Django 5.1.1 on 2025-01-14 19:36

from django.db import migrations

class Migration(migrations.Migration):

```

    dependencies = [
        ("myapp", "0009_customuser"),
    ]

```

```

    operations = []

```

0011_car_owner.py

Generated by Django 5.1.1 on 2025-01-14 19:44

```

import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models

```

class Migration(migrations.Migration):

```

    dependencies = [
        ("myapp", "0010_add_owner_field"),
    ]

```

```

    operations = [
        migrations.AddField(
            model_name="car",
            name="owner",
            field=models.ForeignKey(
                blank=True,
                null=True,
                on_delete=django.db.models.deletion.CASCADE,
                to=settings.AUTH_USER_MODEL,
            ),
        ),
    ]

```

```
    ),
]
```

0012_car_is_available_alter_car_car_type_alter_car_status.py

Generated by Django 5.1.1 on 2025-01-17 09:37

```
import django.db.models.deletion
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0011_car_owner"),
    ]

    operations = [
        migrations.AddField(
            model_name="car",
            name="is_available",
            field=models.BooleanField(default=True),
        ),
        migrations.AlterField(
            model_name="car",
            name="car_type",
            field=models.ForeignKey(
                on_delete=django.db.models.deletion.CASCADE,
                related_name="cars",
                to="myapp.cartype",
            ),
        ),
        migrations.AlterField(
            model_name="car",
            name="status",
            field=models.CharField(
                choices=[("Approved", "Approved"), ("Pending", "Pending")],
                max_length=20,
            ),
        ),
    ]
```

0013_car_created_at.py

Generated by Django 5.1.1 on 2025-01-24 10:02

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0012_car_is_available_alter_car_car_type_alter_car_status"),
    ]

    operations = [
        migrations.AddField(
            model_name="car",
            name="created_at",
            field=models.DateTimeField(auto_now_add=True, null=True),
        ),
    ]
```

0014_notification.py

Generated by Django 5.1.1 on 2025-01-26 11:05

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0013_car_created_at"),
    ]

    operations = [
        migrations.CreateModel(
            name="Notification",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("message", models.TextField()),
                ("timestamp", models.DateTimeField(auto_now_add=True)),
                (
                    "user",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        to=settings.AUTH_USER_MODEL,
                    ),
                ),
            ],
        ),
    ]
```

0015_notification_is_confirmed_notification_schedule_and_more.py

Generated by Django 5.1.1 on 2025-01-29 05:04

```

import django.db.models.deletion
from django.db import migrations, models

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0014_notification"),
    ]

    operations = [
        migrations.AddField(
            model_name="notification",
            name="is_confirmed",
            field=models.BooleanField(default=False),
        ),
        migrations.AddField(
            model_name="notification",
            name="schedule",
            field=models.ForeignKey(
                blank=True,
                null=True,
                on_delete=django.db.models.deletion.SET_NULL,
                to="myapp.schedule",
            ),
        ),
        migrations.AlterField(
            model_name="schedule",
            name="time",
            field=models.CharField(
                choices=[("morning", "เช้านี้"), ("afternoon", "บ่ายนี้")],
                max_length=20,
            ),
        ),
    ]

```

0016_notification_is_approved.py

```

# Generated by Django 5.1.1 on 2025-01-29 08:50

from django.db import migrations, models

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0015_notification_is_confirmed_notification_schedule_and_more"),
    ]

    operations = [
        migrations.AddField(
            model_name="notification",
            name="is_approved",
            field=models.BooleanField(default=False),
        ),
    ]

```

0017_carreview.py

```

# Generated by Django 5.1.1 on 2025-01-30 09:16

import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models

class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0016_notification_is_approved"),
    ]

    operations = [
        migrations.CreateModel(
            name="CarReview",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                (
                    "rating",
                    models.IntegerField(
                        choices=[(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]
                    ),
                ),
                ("review_text", models.TextField()),
                ("created_at", models.DateTimeField(auto_now_add=True)),
                (
                    "car",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        related_name="reviews",
                        to="myapp.car",
                    ),
                ),
                (
                    "user",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        to=settings.AUTH_USER_MODEL,
                    ),
                ),
            ],
        ),
    ]

```

```
    ],
    ),
]
```

0018_review.py

Generated by Django 5.1.1 on 2025-01-30 09:58

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0017_carreview"),
    ]

    operations = [
        migrations.CreateModel(
            name="Review",
            fields=[
                (
                    "id",
                    models.BigAutoField(
                        auto_created=True,
                        primary_key=True,
                        serialize=False,
                        verbose_name="ID",
                    ),
                ),
                ("comment", models.TextField()),
                (
                    "car",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE, to="myapp.car"
                    ),
                ),
                (
                    "user",
                    models.ForeignKey(
                        on_delete=django.db.models.deletion.CASCADE,
                        to=settings.AUTH_USER_MODEL,
                    ),
                ),
            ],
        ),
    ]
```

0019_schedule_booked_by.py

Generated by Django 5.1.1 on 2025-02-03 08:51

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0018_review"),
    ]

    operations = [
        migrations.AddField(
            model_name="schedule",
            name="booked_by",
            field=models.ForeignKey(
                blank=True,
                null=True,
                on_delete=django.db.models.deletion.CASCADE,
                related_name="bookings",
                to=settings.AUTH_USER_MODEL,
            ),
        ),
    ]
```

0020_notification_borrower.py

Generated by Django 5.1.1 on 2025-02-03 10:18

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0019_schedule_booked_by"),
    ]

    operations = [
        migrations.AddField(
            model_name="notification",
            name="borrower",
            field=models.ForeignKey(
                blank=True,
                null=True,
                on_delete=django.db.models.deletion.CASCADE,
                related_name="borrower_notifications",
                to=settings.AUTH_USER_MODEL,
            ),
        ),
    ]
```

0021_alter_car_owner_alter_car_status.py

Generated by Django 5.1.1 on 2025-02-11 04:24

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ("myapp", "0020_notification_borrower"),
    ]

    operations = [
        migrations.AlterField(
            model_name="car",
            name="owner",
            field=models.ForeignKey(
                blank=True,
                null=True,
                on_delete=django.db.models.deletion.CASCADE,
                related_name="cars",
                to=settings.AUTH_USER_MODEL,
            ),
        ),
        migrations.AlterField(
            model_name="car",
            name="status",
            field=models.CharField(
                choices=[("Pending", "Pending"), ("Approved", "Approved")],
                max_length=20,
            ),
        ),
    ]
```

__init__.py