

Web Application in Django Report

Mykola Nadlonok

January 2022

1 Introduction

I have created a web application in Django framework, which is connected with a relational database in MySQL. That simple application allows to rent a car in one of the biggest 60 cities in the USA. The motivation came from the topic of my master thesis. That topic is strongly connected with car models databases, so, having already some data, I decided to connect it with some other data and generate a web application for renting cars.

2 Databases

As the start point, I was learning Django framework from YouTube tutorial, where a blog web application was created. I was following those videos and created a similar blog web application, which was modernised to the rent web application later. So I had the basic web application in the beginning. After that, I had to create a database and connect it with Django framework. Tables have been created during the process of Django migrations. In my database, I have 6 tables. They are:

1. **auth_user** (login, password, name, surname and other info about application user). Table with all registered users.
2. **blog_cars** (manufacturer, model, engine capacity, fuel type and other information)
3. **blog_city** (city, state, population)
4. **blog_rent** (date of renting, city and car)
5. **blog_booking** (booking date, price, rent and customer information)
6. **blog_post** (title, content, date of post, author). Admin can add some articles, which can be interesting for customers.

The next step was to import data, which was stored in .csv files. Python script was used to perform that. Then I have connected my database to Django framework, using settings.py file.

9 DESCRIBE blog_cities;					
COLUMNS (4r x 6c)					
Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	(NULL)	auto_increment
city	varchar(100)	NO		(NULL)	
state	varchar(100)	NO		(NULL)	
population	int(11)	NO		(NULL)	

9 DESCRIBE blog_bookings;					
COLUMNS (6r x 6c)					
Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	(NULL)	auto_increment
booking_date	datetime(6)	NO		(NULL)	
rent_duration	int(11)	NO		(NULL)	
price	int(11)	NO		(NULL)	
customer_id	int(11)	NO	MUL	(NULL)	
rent_id	bigint(20)	NO	MUL	(NULL)	

9 DESCRIBE blog_post;					
COLUMNS (5r x 6c)					
Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	(NULL)	auto_increment
title	varchar(100)	NO		(NULL)	
content	longtext	NO		(NULL)	
date_posted	datetime(6)	NO		(NULL)	
author_id	int(11)	NO	MUL	(NULL)	

9 DESCRIBE blog_rents;					
COLUMNS (5r x 6c)					
Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	(NULL)	auto_increment
from_date	datetime(6)	NO		(NULL)	
to_date	datetime(6)	NO		(NULL)	
car_id	bigint(20)	NO	MUL	(NULL)	
city_id	bigint(20)	NO	MUL	(NULL)	

9 DESCRIBE auth_user;					
COLUMNS (11r x 6c)					
Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	(NULL)	auto_increment
password	varchar(128)	NO		(NULL)	
last_login	datetime(6)	YES		(NULL)	
is_superuser	tinyint(1)	NO		(NULL)	
username	varchar(150)	NO	UNI	(NULL)	
first_name	varchar(150)	NO		(NULL)	
last_name	varchar(150)	NO		(NULL)	
email	varchar(254)	NO		(NULL)	
is_staff	tinyint(1)	NO		(NULL)	
is_active	tinyint(1)	NO		(NULL)	
date_joined	datetime(6)	NO		(NULL)	

9 DESCRIBE blog_cars;					
COLUMNS (20r x 6c)					
Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	(NULL)	auto_increment
manufacturer	varchar(100)	NO		(NULL)	
model	varchar(100)	NO		(NULL)	
description	varchar(100)	NO		(NULL)	
transmission	varchar(100)	NO		(NULL)	
engine_capacity	int(11)	NO		(NULL)	
fuel_type	varchar(100)	NO		(NULL)	
metric_urban	double	NO		(NULL)	
metric_extra_urban	double	NO		(NULL)	
metric_combined	double	NO		(NULL)	
imperial_urban	double	NO		(NULL)	
imperial_extra_urban	double	NO		(NULL)	
imperial_combined	double	NO		(NULL)	
co2	double	NO		(NULL)	
fuel_cost	double	NO		(NULL)	
euro_standard	varchar(100)	NO		(NULL)	
noise_level	double	NO		(NULL)	
emissions_co	double	NO		(NULL)	
emissions_hc	double	NO		(NULL)	
emissions_nox	double	NO		(NULL)	

Figure 1: Description of generated tables in database

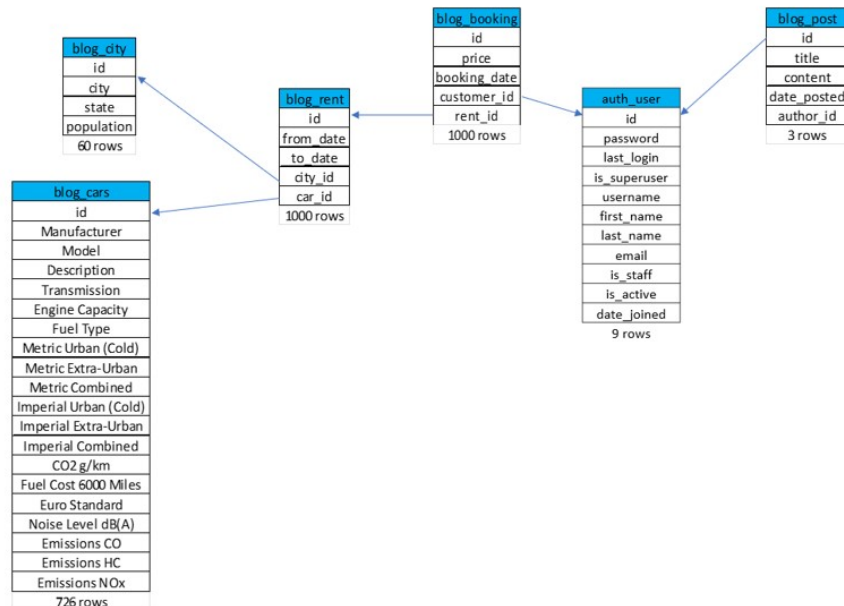


Figure 2: ERD Diagram

3 Functionality

The html files are responsible for the visual aspect of the application. My home views for logged in and logged-out users look in the following way:

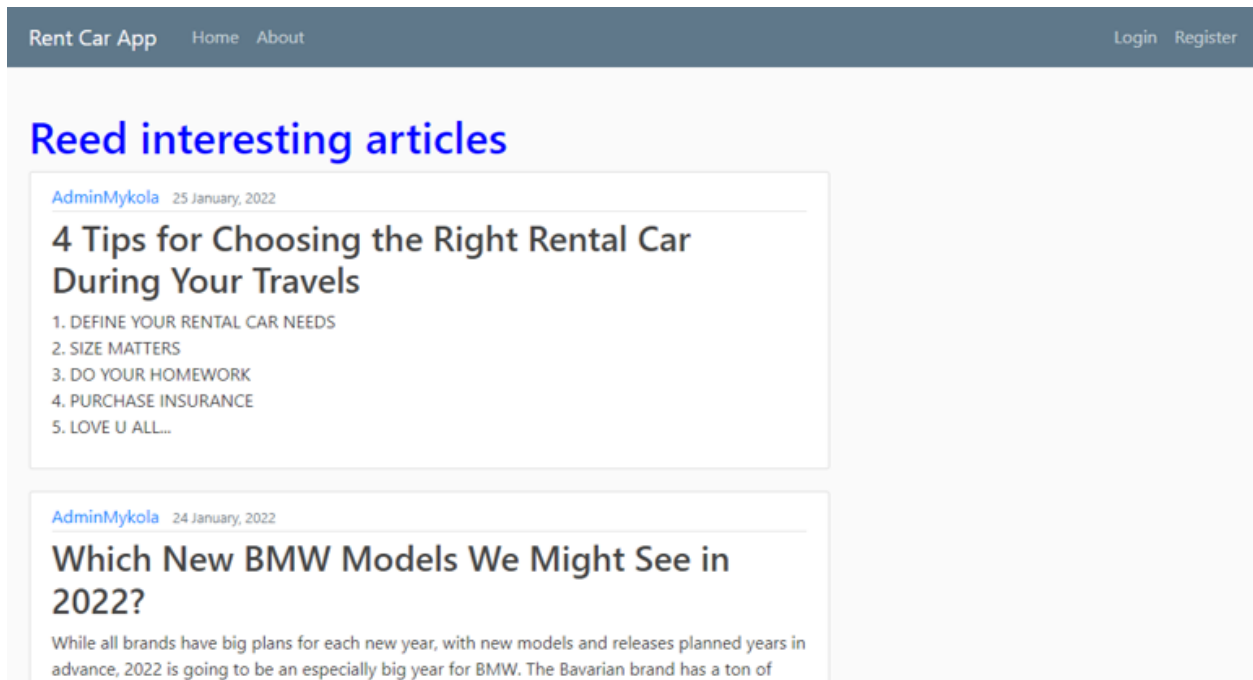


Figure 3: Screenshot of home page for logged out user

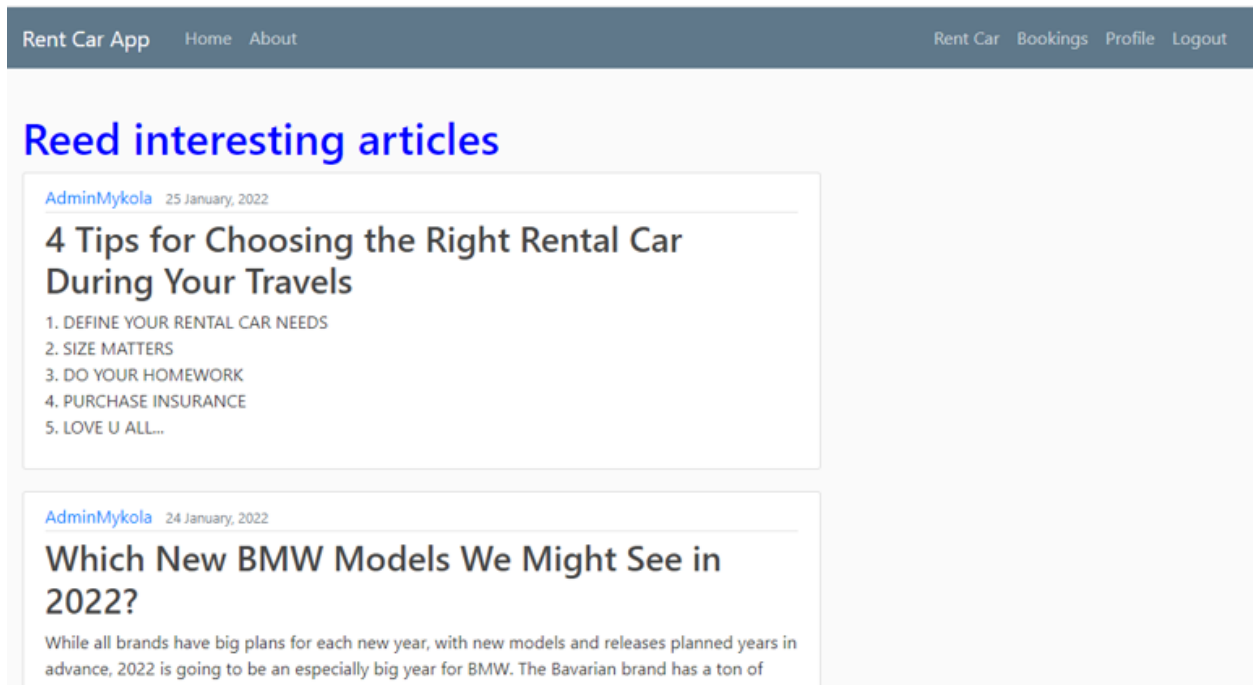


Figure 4: Screenshot of Home Page for logged in user

If the user is not logged in, it is possible to login or register. I have used crispy forms for security goals. Created password has to be quite strong and meet several conditions. Django also keep the password encrypted in the database.

As it is possible to notice on the screenshots above, bookmarks Home and About are available for all.

Home Page. On Home bookmark, there are articles about cars, which were posted by Admin. Posts can be added, updated and deleted only by Admin.

About Page. On the About Page everyone can read short description about Rent Car App.



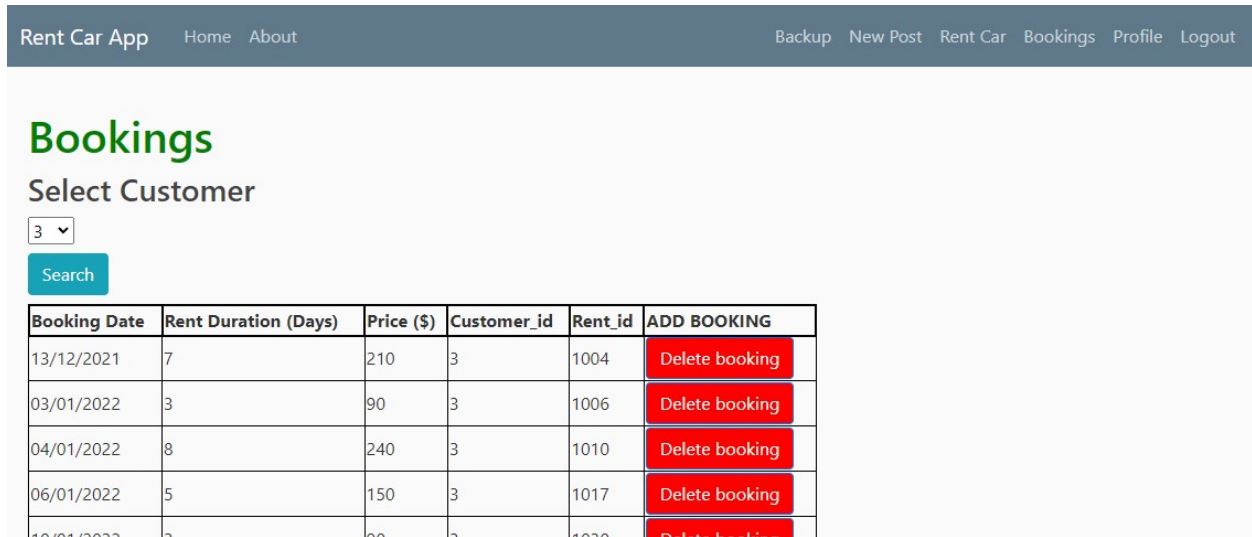
Figure 5: Screenshot of About Page

Rent Car Page. Every logged-in user (Admin, Accountant, or Customer) have access to Rent Car Bookmark. User can choose options of renting a car (choose city, car brand and dates). After clicking Search, only models of chosen brand will be shown and user can book the concrete car. That page looks in the following way:

Brand	Model	Model description	Fuel	Engine Vol (cm3)	Transmission	Euro Standard	ADD BOOKING
ALFA ROMEO	147 Range	GTA	Petrol	3179	M6	IV	Book car
ALFA ROMEO	166 (2004) Range	3.2 V6 24v	Petrol	3179	M6	IV	Book car
ALFA ROMEO	Spider/GTV (2003) Range	2.0 JTS	Petrol	1970	M5	IV	Book car
ALFA ROMEO	GT Range	2.0 JTS Selespeed	Petrol	1970	SAT5	IV	Book car

Figure 6: Screenshot of Rent Page

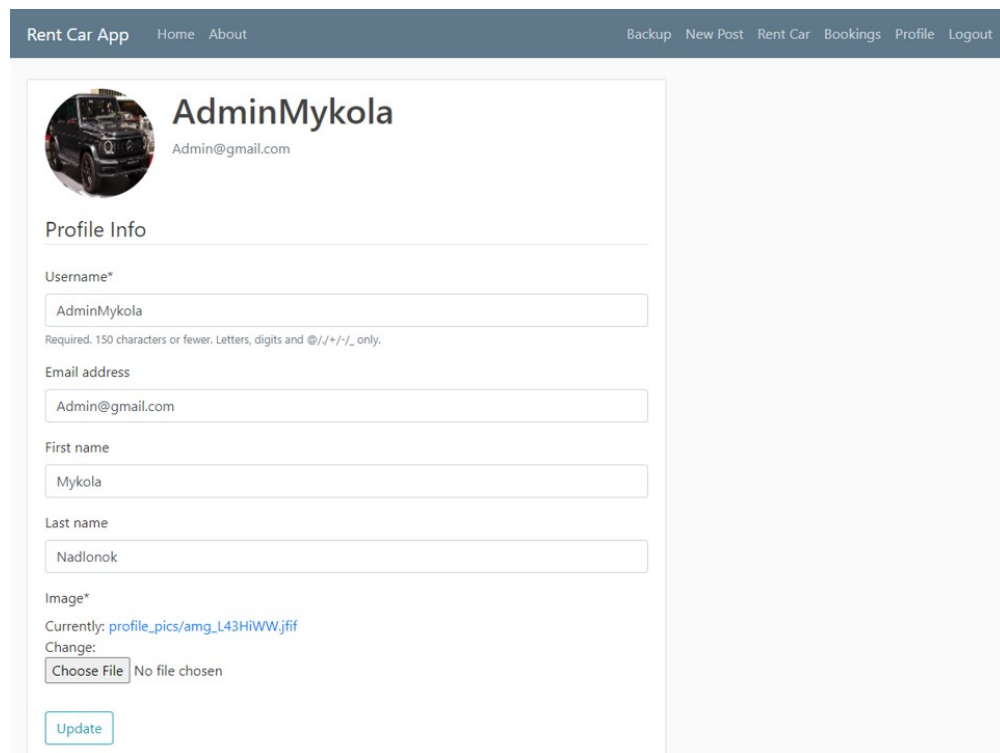
Booking Page. Admin and Accountant have access to the whole bookings table. They can search bookings made by concrete customer and they have possibility to delete any booking. However, customer can see only his own bookings and have possibility to delete only them.



Booking Date	Rent Duration (Days)	Price (\$)	Customer_id	Rent_id	ADD BOOKING
13/12/2021	7	210	3	1004	Delete booking
03/01/2022	3	90	3	1006	Delete booking
04/01/2022	8	240	3	1010	Delete booking
06/01/2022	5	150	3	1017	Delete booking
10/01/2022	7	210	3	1020	Delete booking

Figure 7: Screenshot of Booking Page (Admin, Accountant)

Profile Page. Profile Page is available for every logged-in user. Such user can update information in his profile and add profile images. An example of such Profile Page is illustrated below.



AdminMykola
Admin@gmail.com

Profile Info

Username*
AdminMykola
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address
Admin@gmail.com

First name
Mykola

Last name
Nadlonok

Image*
Currently: [profile_pics/amg_L43HiWW.jfif](#)
Change:
Choose File No file chosen

Update

Figure 8: Screenshot of Profile Page

On figure 9 it is possible to notice that Admin has one additional bookmark, which is not available for other users.

BackUp Page. Admin can create backup and restore data from backup.

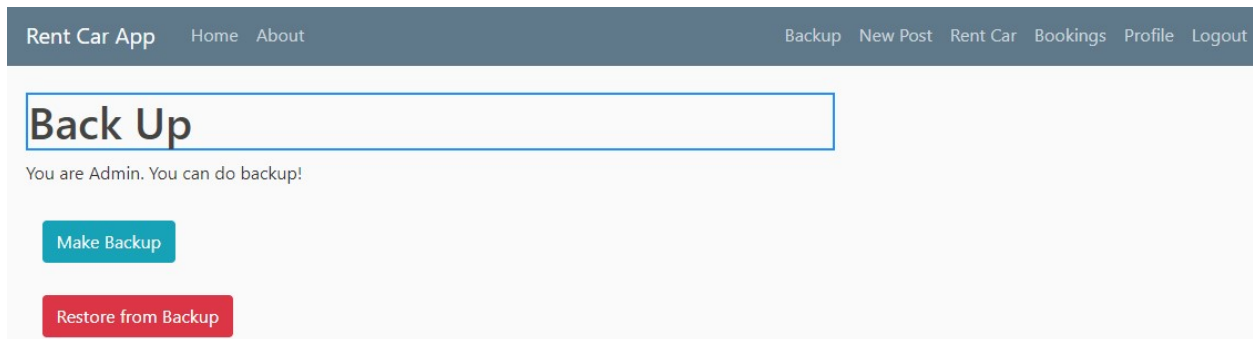


Figure 9: Screenshot of BackUp Page

My project has 3 folders (django_project, blog, users) and the file manage.py.

Folder django_project - responsible for application's working and connecting to database;

Folder user - responsible for registration process and manipulation with user profile;

Folder blog_app - responsible for the main logic of renting car application.

To run the created application, it is necessary to enter the command "python manage.py run-server" in the project folder.