Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ № 2

по теме: **РЕАЛИЗАЦИЯ ПРОСТОГО САЙТА СРЕДСТВАМИ DJANGO.**

по дисциплине: Web-программирование

Специальность:

45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверил:	Выполнила:
Говоров А.И. __________	студентка
Дата: «22» декабря 2021г.	группы К33422
Оценка ______________	Редичкина А.М

Санкт-Петербург

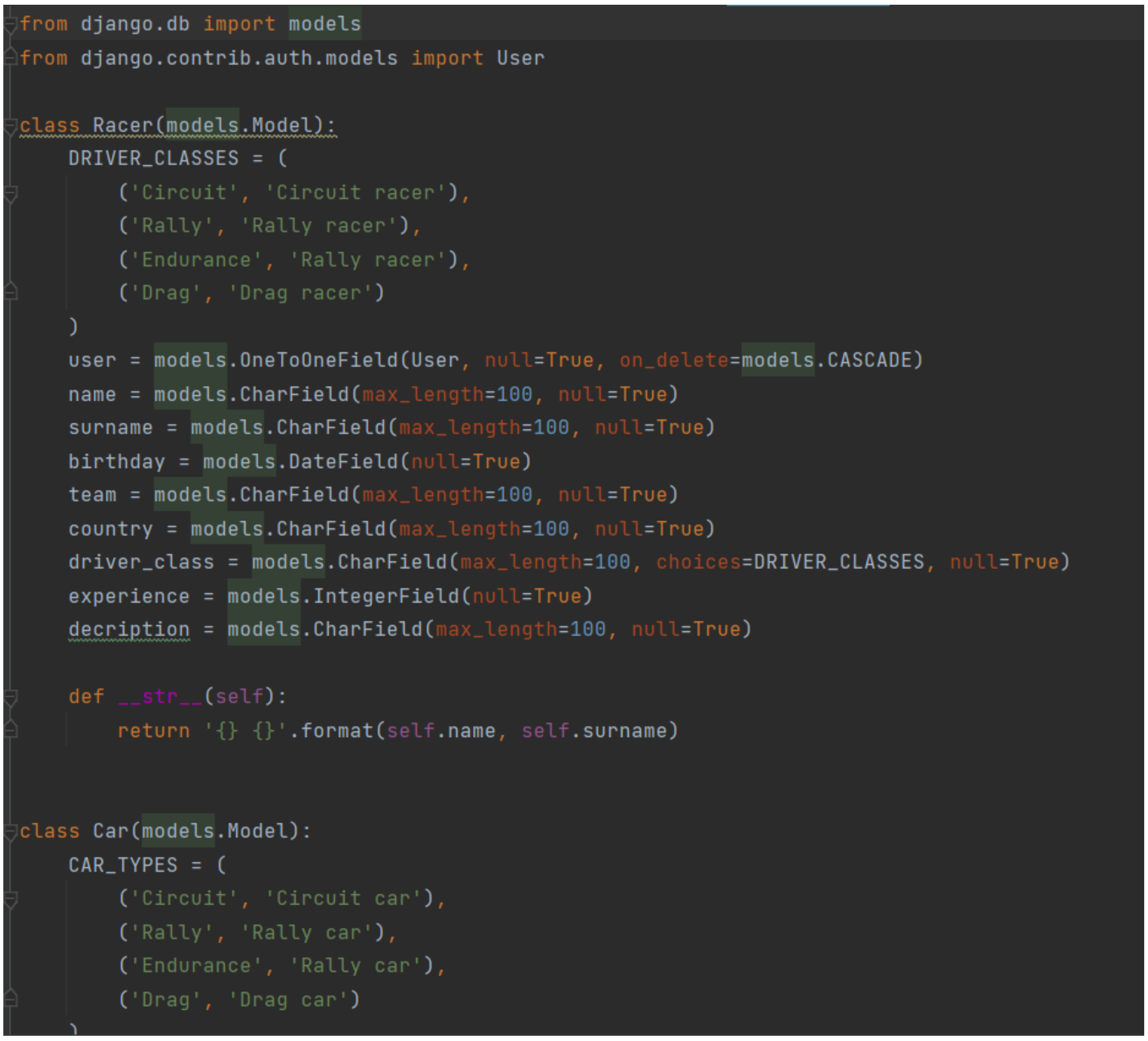
2021

ЦЕЛЬ РАБОТЫ

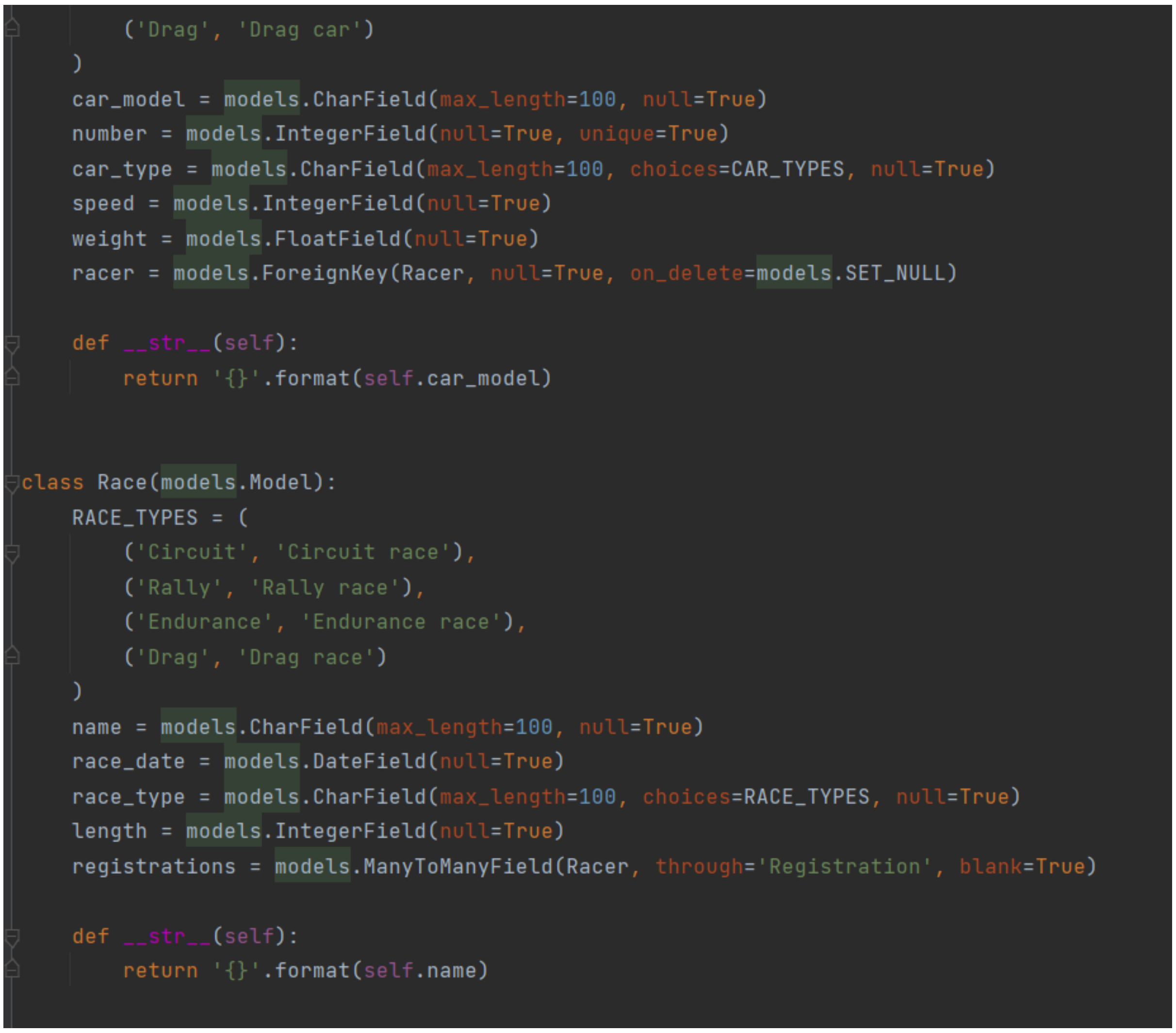
Овладеть практическими навыками и умениями реализации web-серверов средствами Django 2.2

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

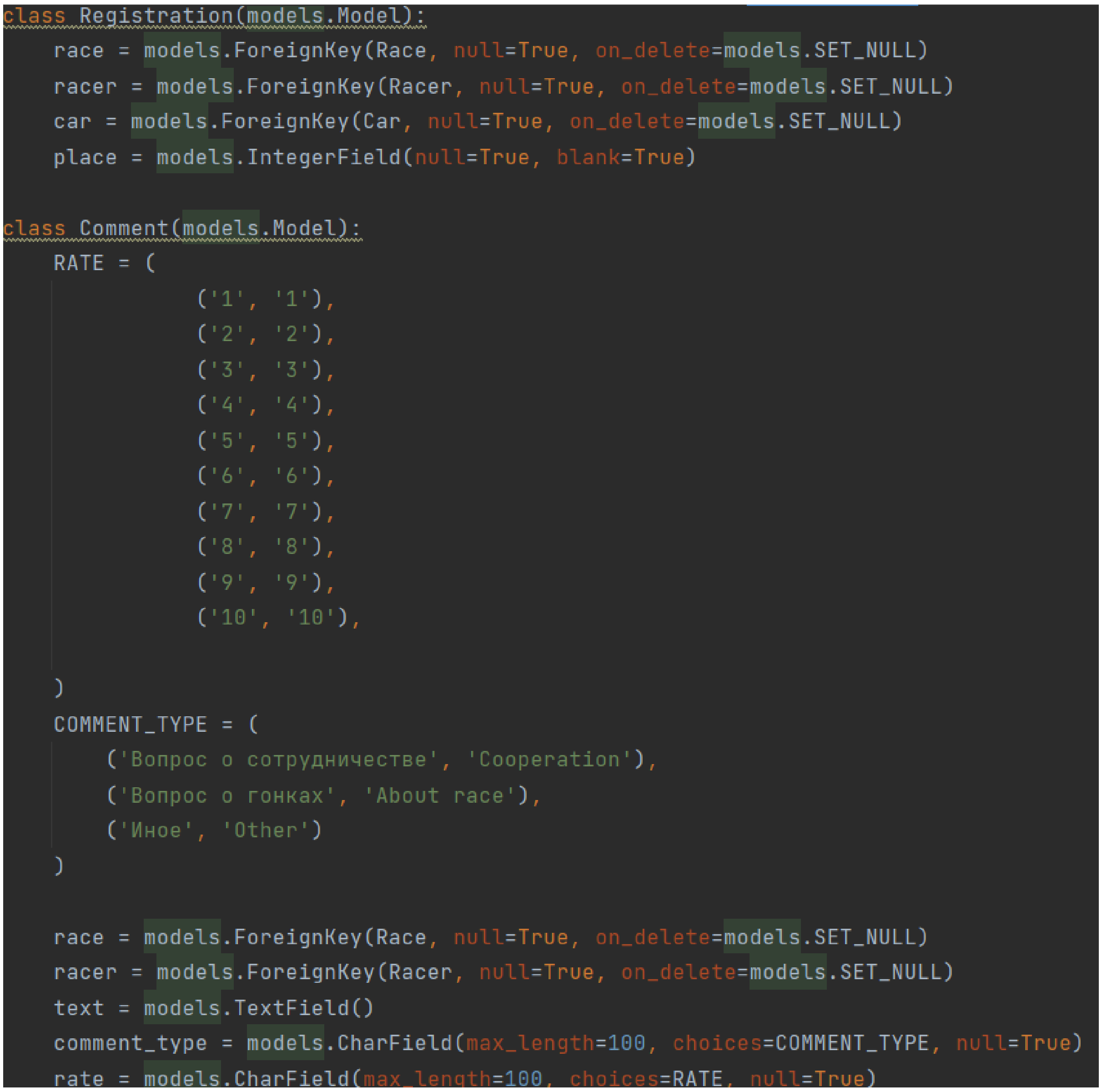
1. Описание модели данных (models.py)



```python
from django.db import models
from django.contrib.auth.models import User

class Racer(models.Model):
    DRIVER_CLASSES = (
        ('Circuit', 'Circuit racer'),
        ('Rally', 'Rally racer'),
        ('Endurance', 'Rally racer'),
        ('Drag', 'Drag racer')
    )
    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    name = models.CharField(max_length=100, null=True)
    surname = models.CharField(max_length=100, null=True)
    birthday = models.DateField(null=True)
    team = models.CharField(max_length=100, null=True)
    country = models.CharField(max_length=100, null=True)
    driver_class = models.CharField(max_length=100, choices=DRIVER_CLASSES, null=True)
    experience = models.IntegerField(null=True)
    decription = models.CharField(max_length=100, null=True)

    def __str__(self):
        return '{} {}'.format(self.name, self.surname)


class Car(models.Model):
    CAR_TYPES = (
        ('Circuit', 'Circuit car'),
        ('Rally', 'Rally car'),
        ('Endurance', 'Rally car'),
        ('Drag', 'Drag car')
    )
```

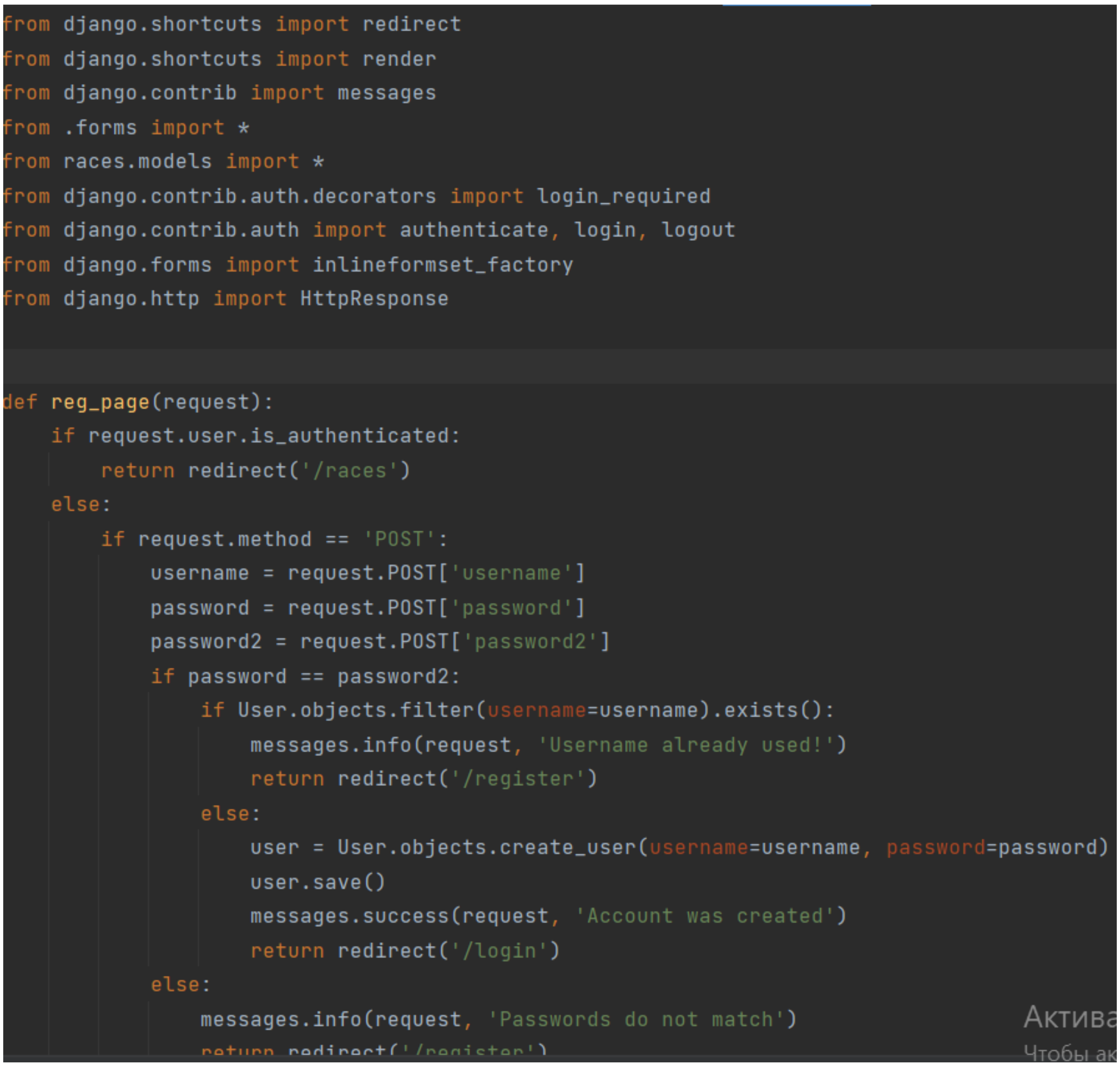


```
        ('Drag', 'Drag car')
    )
    car_model = models.CharField(max_length=100, null=True)
    number = models.IntegerField(null=True, unique=True)
    car_type = models.CharField(max_length=100, choices=CAR_TYPES, null=True)
    speed = models.IntegerField(null=True)
    weight = models.FloatField(null=True)
    racer = models.ForeignKey(Racer, null=True, on_delete=models.SET_NULL)

    def __str__(self):
        return '{}'.format(self.car_model)


class Race(models.Model):
    RACE_TYPES = (
        ('Circuit', 'Circuit race'),
        ('Rally', 'Rally race'),
        ('Endurance', 'Endurance race'),
        ('Drag', 'Drag race')
    )
    name = models.CharField(max_length=100, null=True)
    race_date = models.DateField(null=True)
    race_type = models.CharField(max_length=100, choices=RACE_TYPES, null=True)
    length = models.IntegerField(null=True)
    registrations = models.ManyToManyField(Racer, through='Registration', blank=True)

    def __str__(self):
        return '{}'.format(self.name)
```

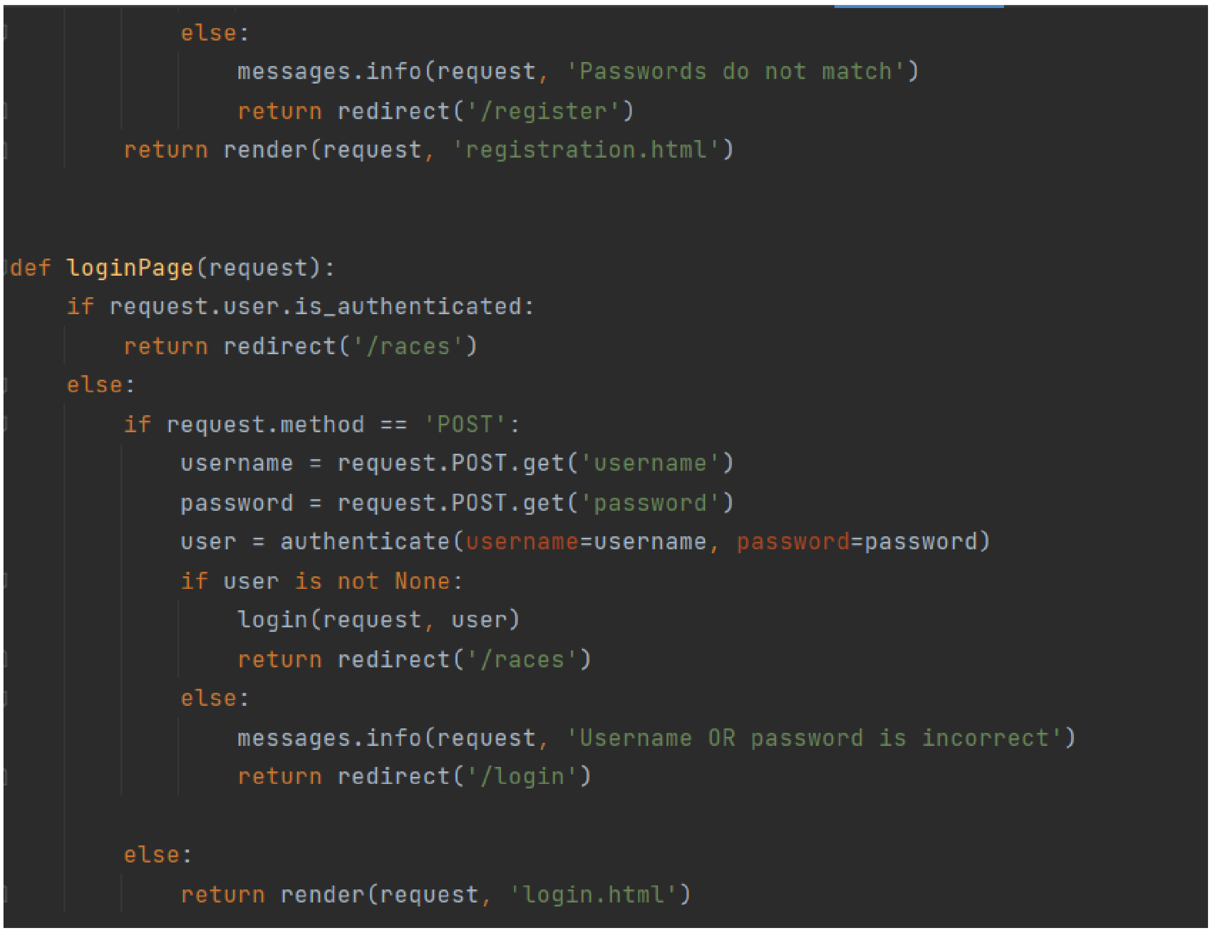


```python
class Registration(models.Model):
    race = models.ForeignKey(Race, null=True, on_delete=models.SET_NULL)
    racer = models.ForeignKey(Racer, null=True, on_delete=models.SET_NULL)
    car = models.ForeignKey(Car, null=True, on_delete=models.SET_NULL)
    place = models.IntegerField(null=True, blank=True)

class Comment(models.Model):
    RATE = (
                ('1', '1'),
                ('2', '2'),
                ('3', '3'),
                ('4', '4'),
                ('5', '5'),
                ('6', '6'),
                ('7', '7'),
                ('8', '8'),
                ('9', '9'),
                ('10', '10'),

    )
    COMMENT_TYPE = (
        ('Вопрос о сотрудничестве', 'Cooperation'),
        ('Вопрос о гонках', 'About race'),
        ('Иное', 'Other')
    )

    race = models.ForeignKey(Race, null=True, on_delete=models.SET_NULL)
    racer = models.ForeignKey(Racer, null=True, on_delete=models.SET_NULL)
    text = models.TextField()
    comment_type = models.CharField(max_length=100, choices=COMMENT_TYPE, null=True)
    rate = models.CharField(max_length=100, choices=RATE, null=True)
```

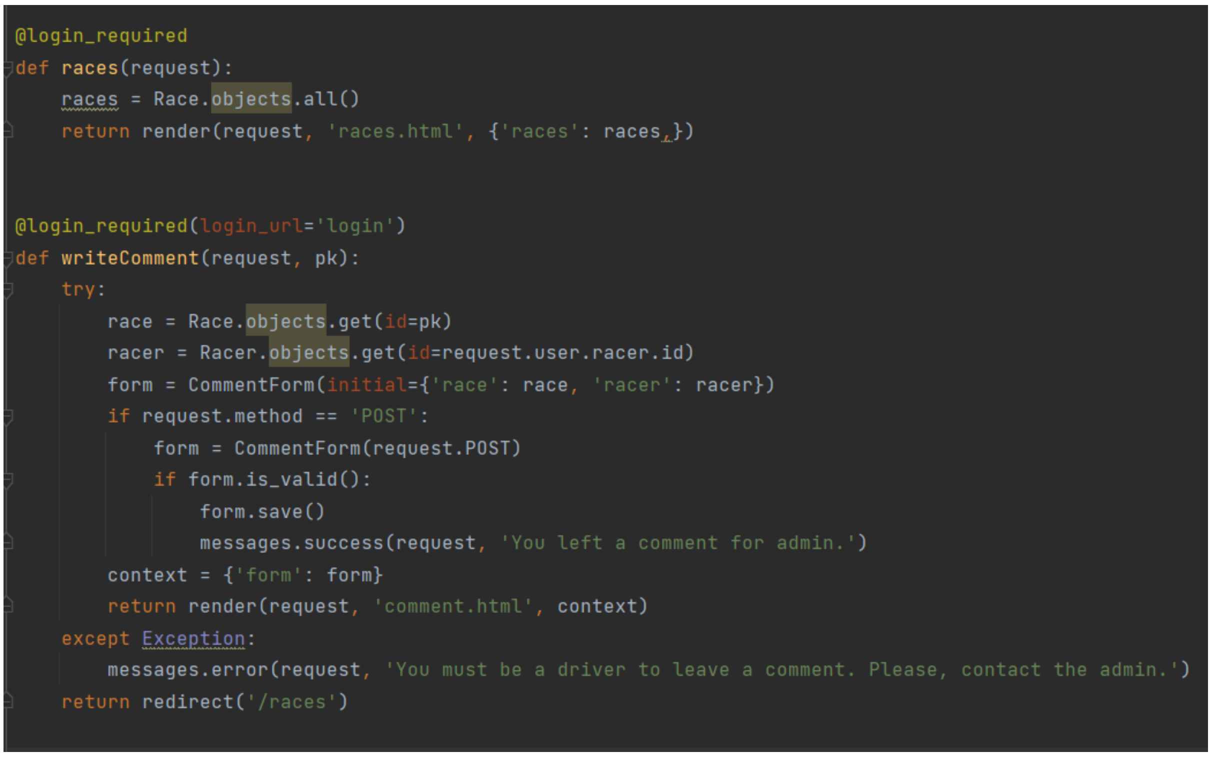
2. Содержание файла view.py



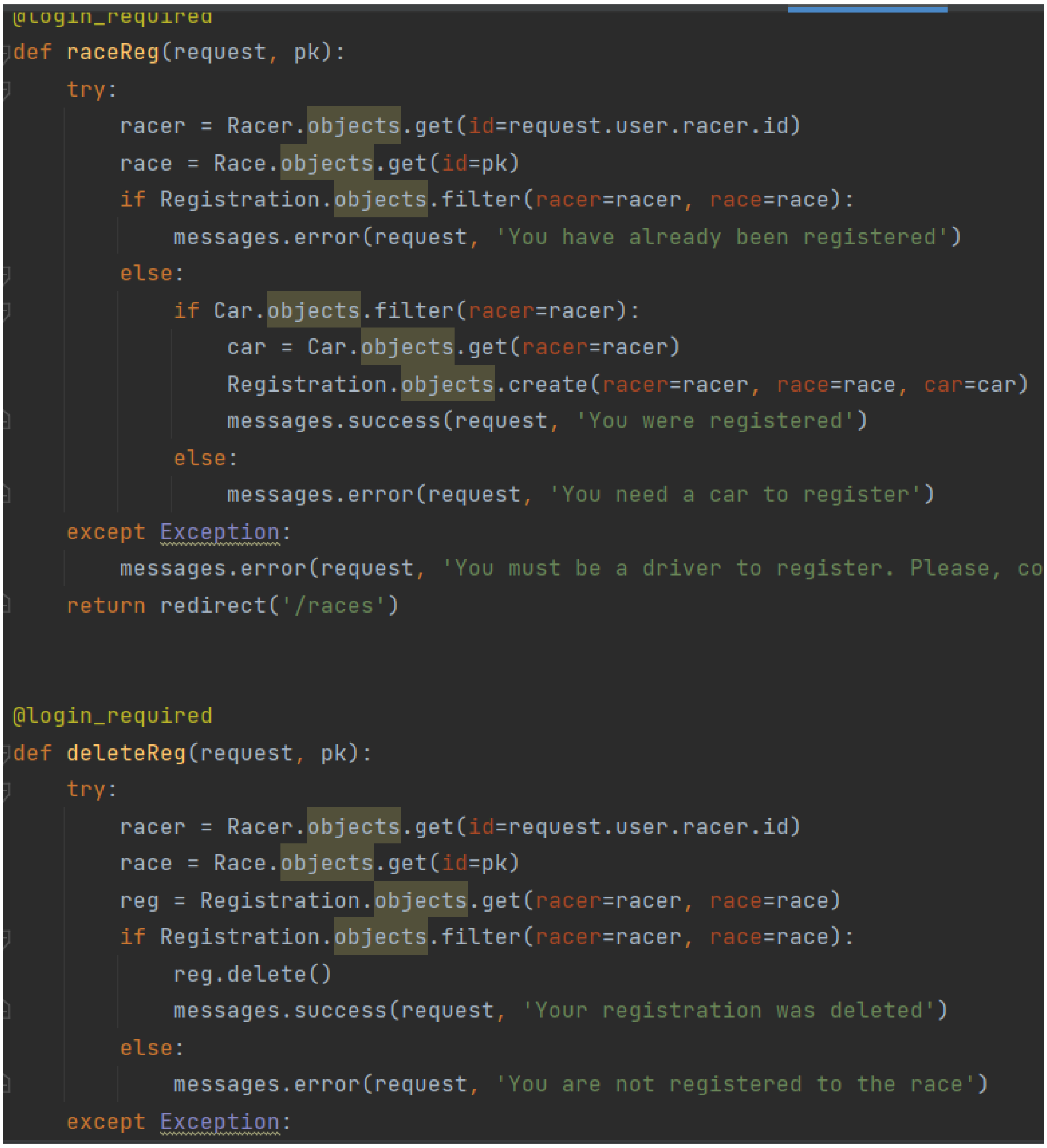
```python
from django.shortcuts import redirect
from django.shortcuts import render
from django.contrib import messages
from .forms import *
from races.models import *
from django.contrib.auth.decorators import login_required
from django.contrib.auth import authenticate, login, logout
from django.forms import inlineformset_factory
from django.http import HttpResponse


def reg_page(request):
    if request.user.is_authenticated:
        return redirect('/races')
    else:
        if request.method == 'POST':
            username = request.POST['username']
            password = request.POST['password']
            password2 = request.POST['password2']
            if password == password2:
                if User.objects.filter(username=username).exists():
                    messages.info(request, 'Username already used!')
                    return redirect('/register')
                else:
                    user = User.objects.create_user(username=username, password=password)
                    user.save()
                    messages.success(request, 'Account was created')
                    return redirect('/login')
            else:
                messages.info(request, 'Passwords do not match')
                return redirect('/register')
```



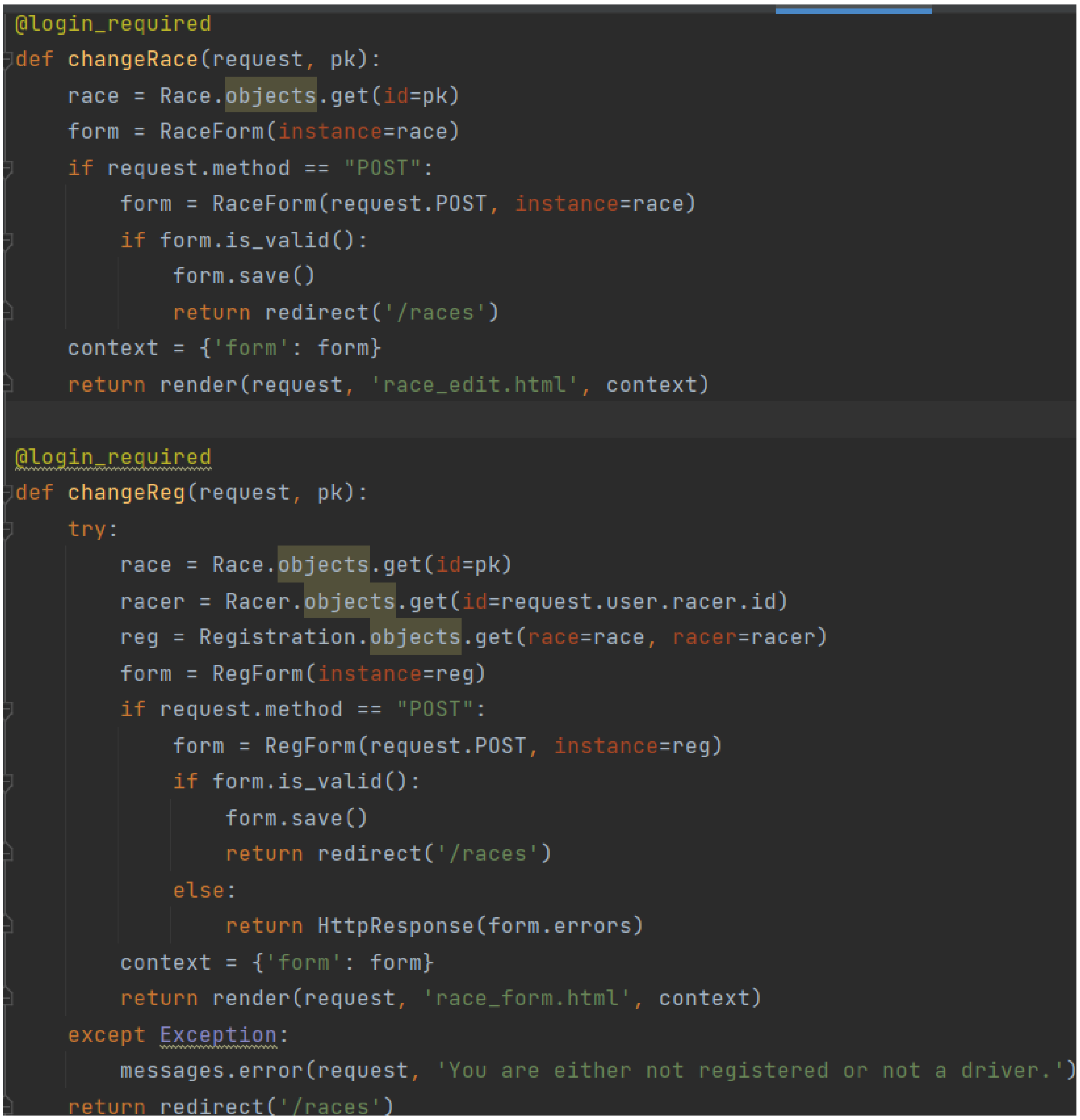
```python
            else:
                messages.info(request, 'Passwords do not match')
                return redirect('/register')
        return render(request, 'registration.html')


def loginPage(request):
    if request.user.is_authenticated:
        return redirect('/races')
    else:
        if request.method == 'POST':
            username = request.POST.get('username')
            password = request.POST.get('password')
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('/races')
            else:
                messages.info(request, 'Username OR password is incorrect')
                return redirect('/login')

        else:
            return render(request, 'login.html')
```



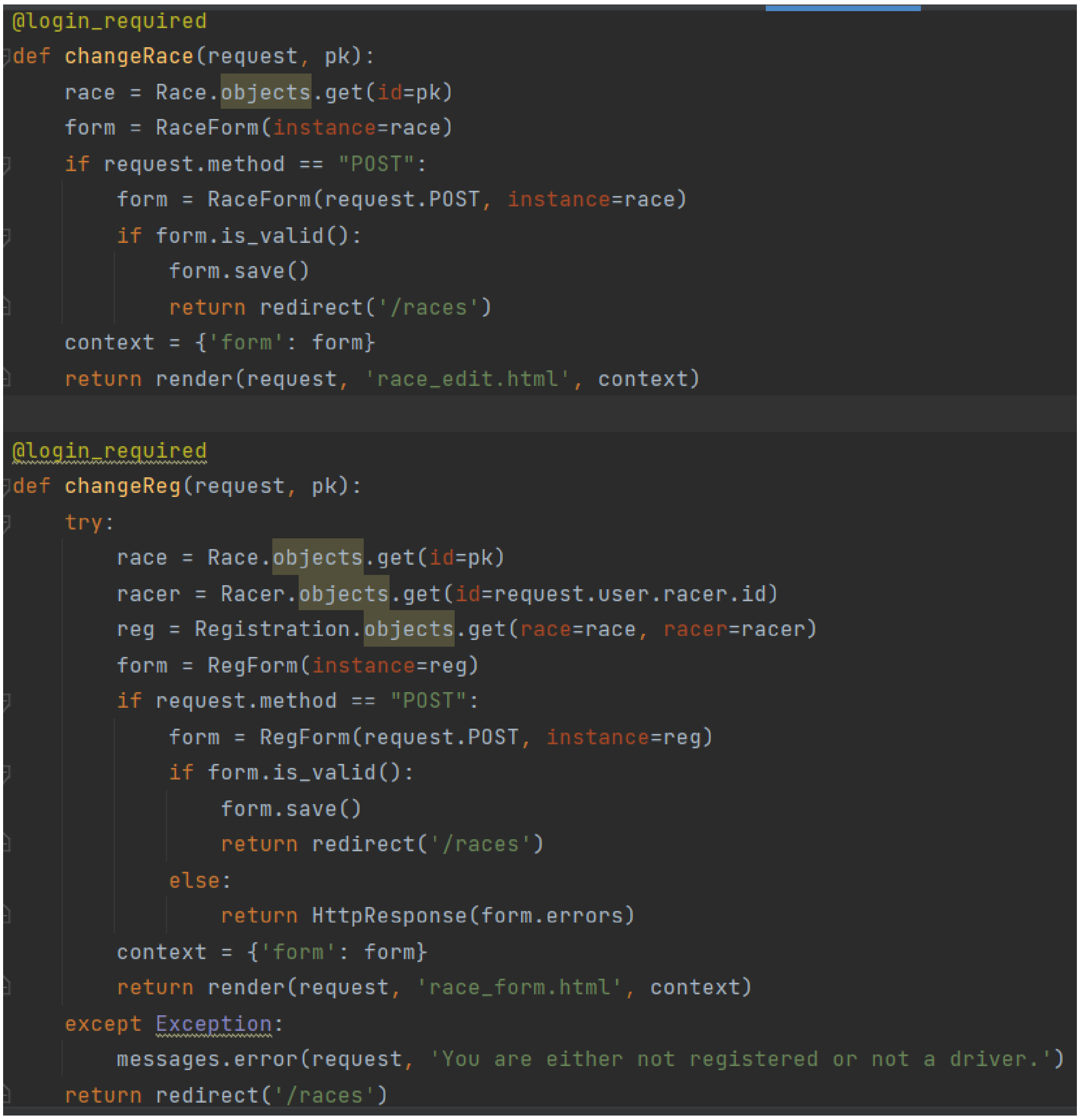
```python
@login_required
def races(request):
    races = Race.objects.all()
    return render(request, 'races.html', {'races': races,})


@login_required(login_url='login')
def writeComment(request, pk):
    try:
        race = Race.objects.get(id=pk)
        racer = Racer.objects.get(id=request.user.racer.id)
        form = CommentForm(initial={'race': race, 'racer': racer})
        if request.method == 'POST':
            form = CommentForm(request.POST)
            if form.is_valid():
                form.save()
                messages.success(request, 'You left a comment for admin.')
        context = {'form': form}
        return render(request, 'comment.html', context)
    except Exception:
        messages.error(request, 'You must be a driver to leave a comment. Please, contact the admin.')
    return redirect('/races')
```

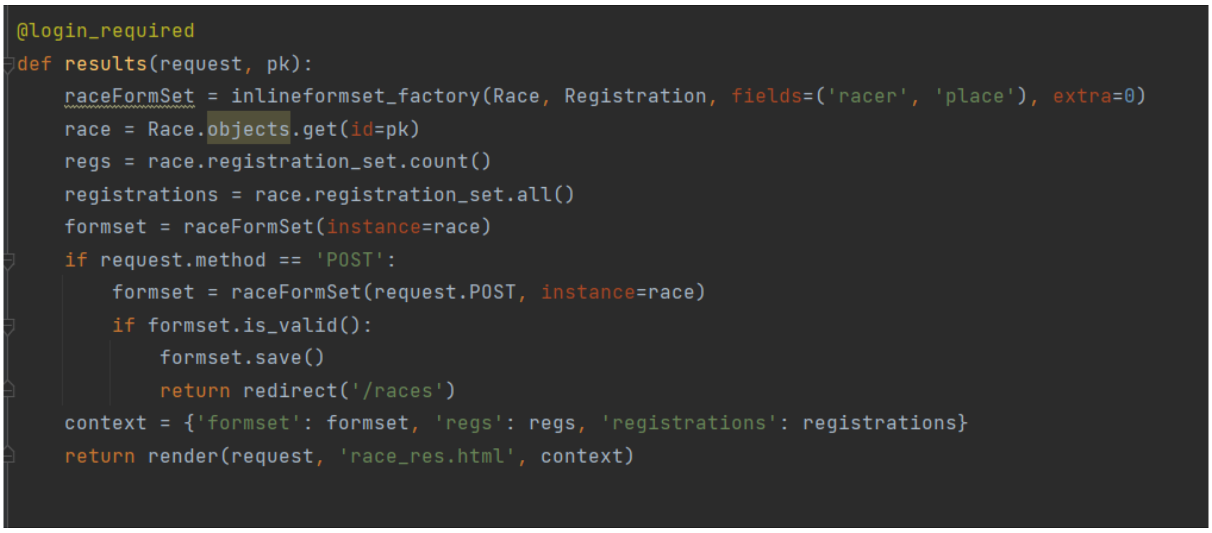


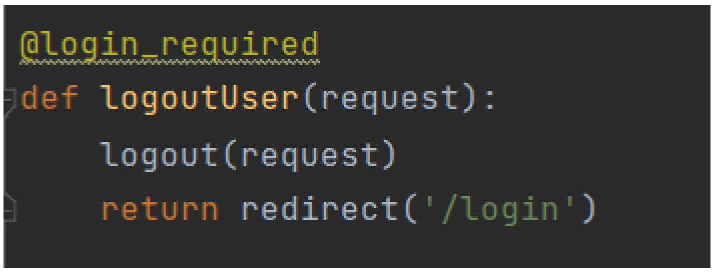
```
@login_required
def raceReg(request, pk):
    try:
        racer = Racer.objects.get(id=request.user.racer.id)
        race = Race.objects.get(id=pk)
        if Registration.objects.filter(racer=racer, race=race):
            messages.error(request, 'You have already been registered')
        else:
            if Car.objects.filter(racer=racer):
                car = Car.objects.get(racer=racer)
                Registration.objects.create(racer=racer, race=race, car=car)
                messages.success(request, 'You were registered')
            else:
                messages.error(request, 'You need a car to register')
    except Exception:
        messages.error(request, 'You must be a driver to register. Please, co
    return redirect('/races')


@login_required
def deleteReg(request, pk):
    try:
        racer = Racer.objects.get(id=request.user.racer.id)
        race = Race.objects.get(id=pk)
        reg = Registration.objects.get(racer=racer, race=race)
        if Registration.objects.filter(racer=racer, race=race):
            reg.delete()
            messages.success(request, 'Your registration was deleted')
        else:
            messages.error(request, 'You are not registered to the race')
    except Exception:
```



```python
@login_required
def changeRace(request, pk):
    race = Race.objects.get(id=pk)
    form = RaceForm(instance=race)
    if request.method == "POST":
        form = RaceForm(request.POST, instance=race)
        if form.is_valid():
            form.save()
            return redirect('/races')
    context = {'form': form}
    return render(request, 'race_edit.html', context)

@login_required
def changeReg(request, pk):
    try:
        race = Race.objects.get(id=pk)
        racer = Racer.objects.get(id=request.user.racer.id)
        reg = Registration.objects.get(race=race, racer=racer)
        form = RegForm(instance=reg)
        if request.method == "POST":
            form = RegForm(request.POST, instance=reg)
            if form.is_valid():
                form.save()
                return redirect('/races')
            else:
                return HttpResponse(form.errors)
        context = {'form': form}
        return render(request, 'race_form.html', context)
    except Exception:
        messages.error(request, 'You are either not registered or not a driver.')
    return redirect('/races')
```

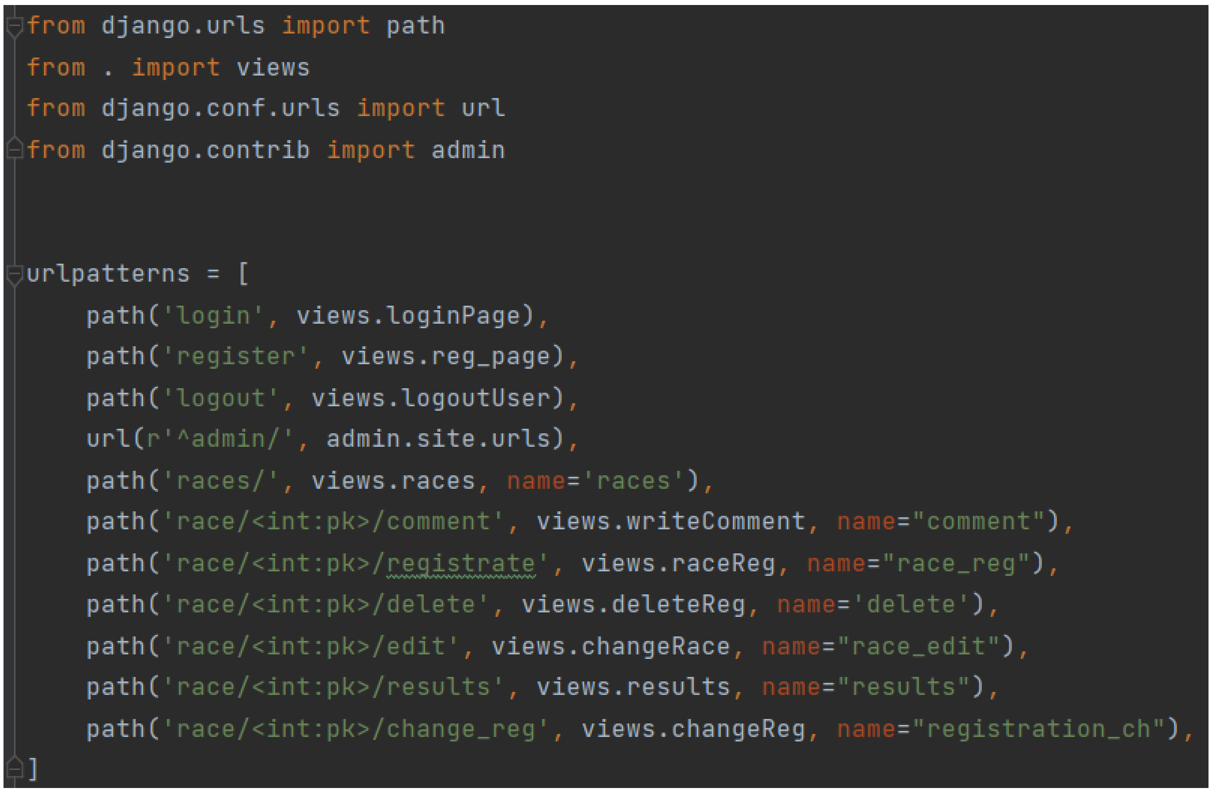

```python
@login_required
def changeRace(request, pk):
    race = Race.objects.get(id=pk)
    form = RaceForm(instance=race)
    if request.method == "POST":
        form = RaceForm(request.POST, instance=race)
        if form.is_valid():
            form.save()
            return redirect('/races')
    context = {'form': form}
    return render(request, 'race_edit.html', context)

@login_required
def changeReg(request, pk):
    try:
        race = Race.objects.get(id=pk)
        racer = Racer.objects.get(id=request.user.racer.id)
        reg = Registration.objects.get(race=race, racer=racer)
        form = RegForm(instance=reg)
        if request.method == "POST":
            form = RegForm(request.POST, instance=reg)
            if form.is_valid():
                form.save()
                return redirect('/races')
            else:
                return HttpResponse(form.errors)
        context = {'form': form}
        return render(request, 'race_form.html', context)
    except Exception:
        messages.error(request, 'You are either not registered or not a driver.')
    return redirect('/races')
```


```python
@login_required
def results(request, pk):
    raceFormSet = inlineformset_factory(Race, Registration, fields=('racer', 'place'), extra=0)
    race = Race.objects.get(id=pk)
    regs = race.registration_set.count()
    registrations = race.registration_set.all()
    formset = raceFormSet(instance=race)
    if request.method == 'POST':
        formset = raceFormSet(request.POST, instance=race)
        if formset.is_valid():
            formset.save()
            return redirect('/races')
    context = {'formset': formset, 'regs': regs, 'registrations': registrations}
    return render(request, 'race_res.html', context)
```



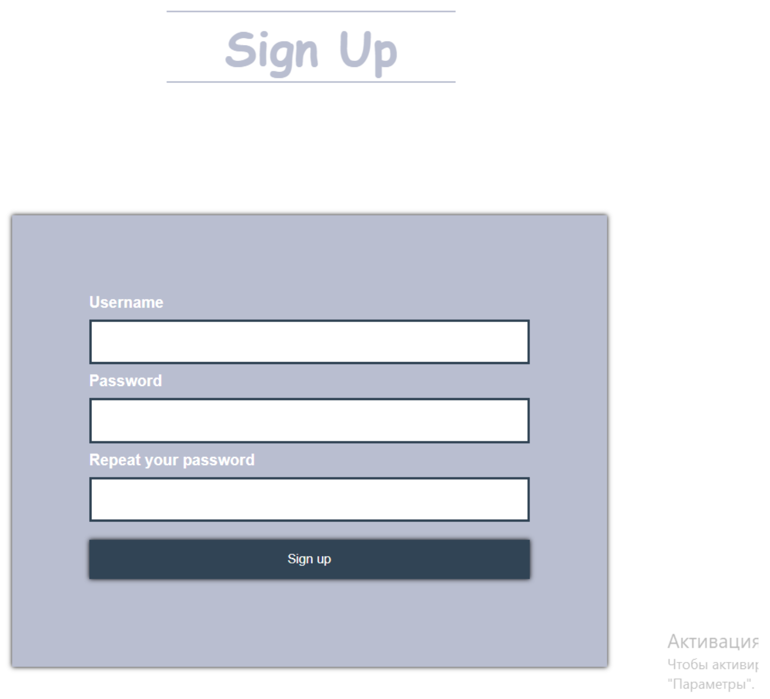
```python
@login_required
def logoutUser(request):
    logout(request)
    return redirect('/login')
```

3. Содержание файла url.py



```python
from django.urls import path
from . import views
from django.conf.urls import url
from django.contrib import admin


urlpatterns = [
    path('login', views.loginPage),
    path('register', views.reg_page),
    path('logout', views.logoutUser),
    url(r'^admin/', admin.site.urls),
    path('races/', views.races, name='races'),
    path('race/<int:pk>/comment', views.writeComment, name="comment"),
    path('race/<int:pk>/registrate', views.raceReg, name="race_reg"),
    path('race/<int:pk>/delete', views.deleteReg, name='delete'),
    path('race/<int:pk>/edit', views.changeRace, name="race_edit"),
    path('race/<int:pk>/results', views.results, name="results"),
    path('race/<int:pk>/change_reg', views.changeReg, name="registration_ch"),
]
```

4. Интерфейс



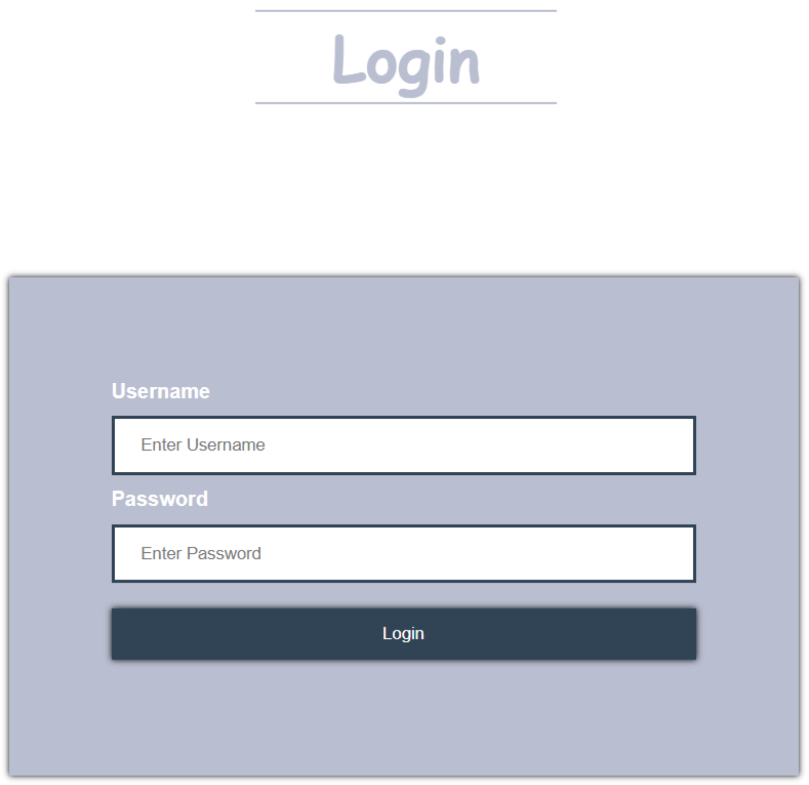
Sign Up

Username

Password

Repeat your password

Sign up



Login

Username

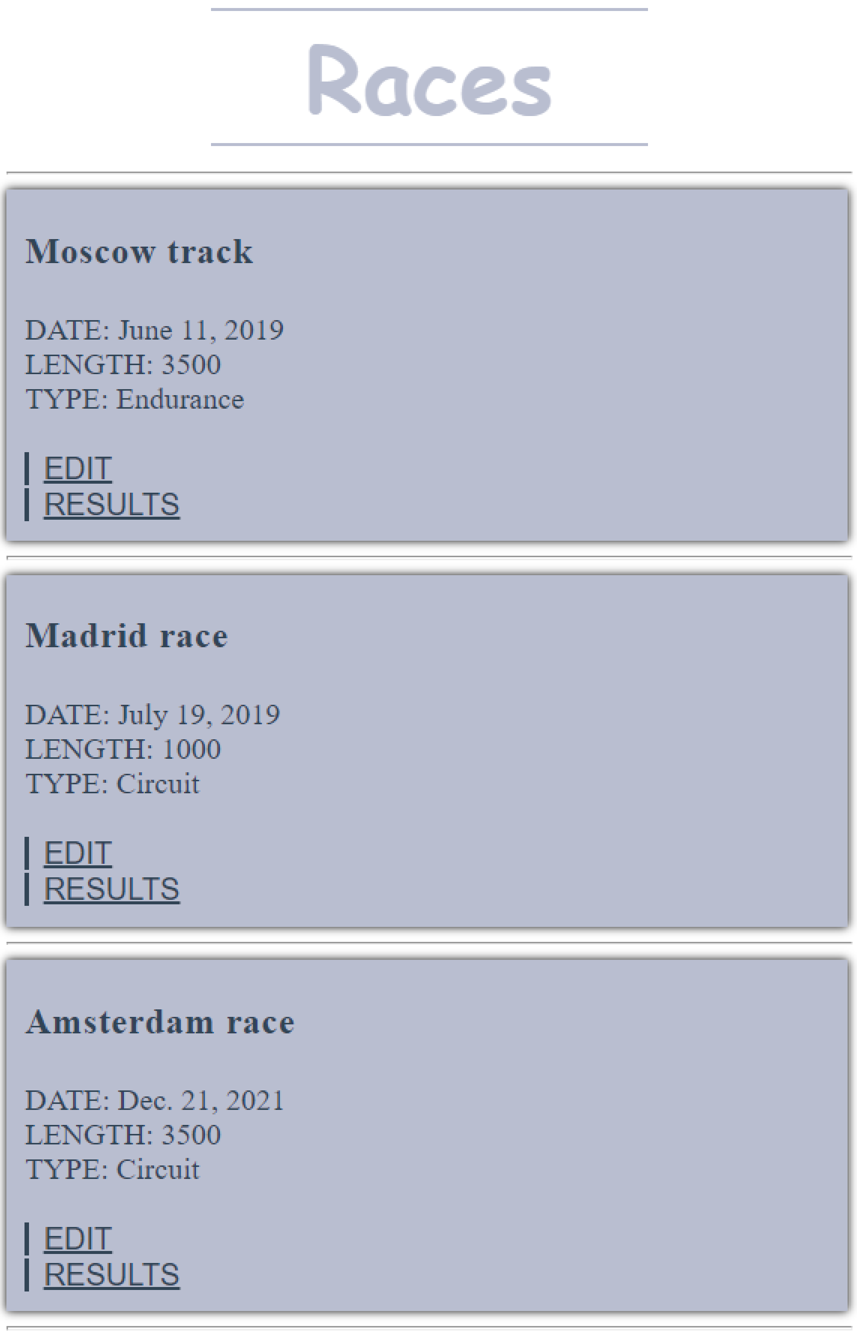
Enter Username

Password

Enter Password

Login

Табло с гонками для администратора



Races

Moscow track

DATE: June 11, 2019
LENGTH: 3500
TYPE: Endurance

| EDIT
| RESULTS

Madrid race

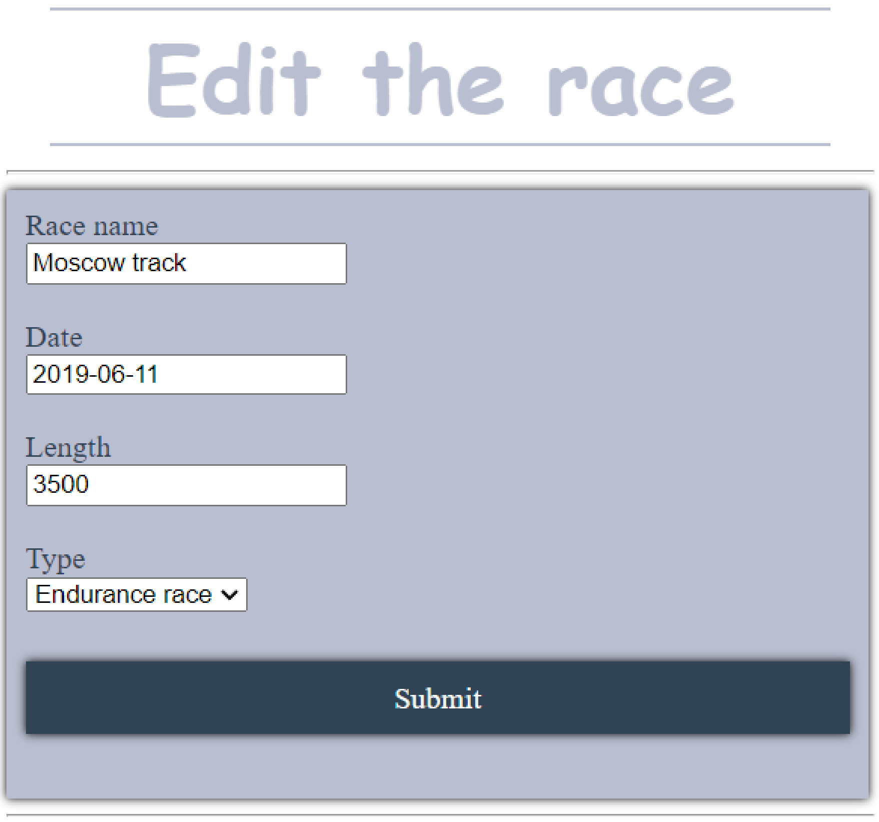
DATE: July 19, 2019
LENGTH: 1000
TYPE: Circuit

| EDIT
| RESULTS

Amsterdam race

DATE: Dec. 21, 2021
LENGTH: 3500
TYPE: Circuit

| EDIT
| RESULTS



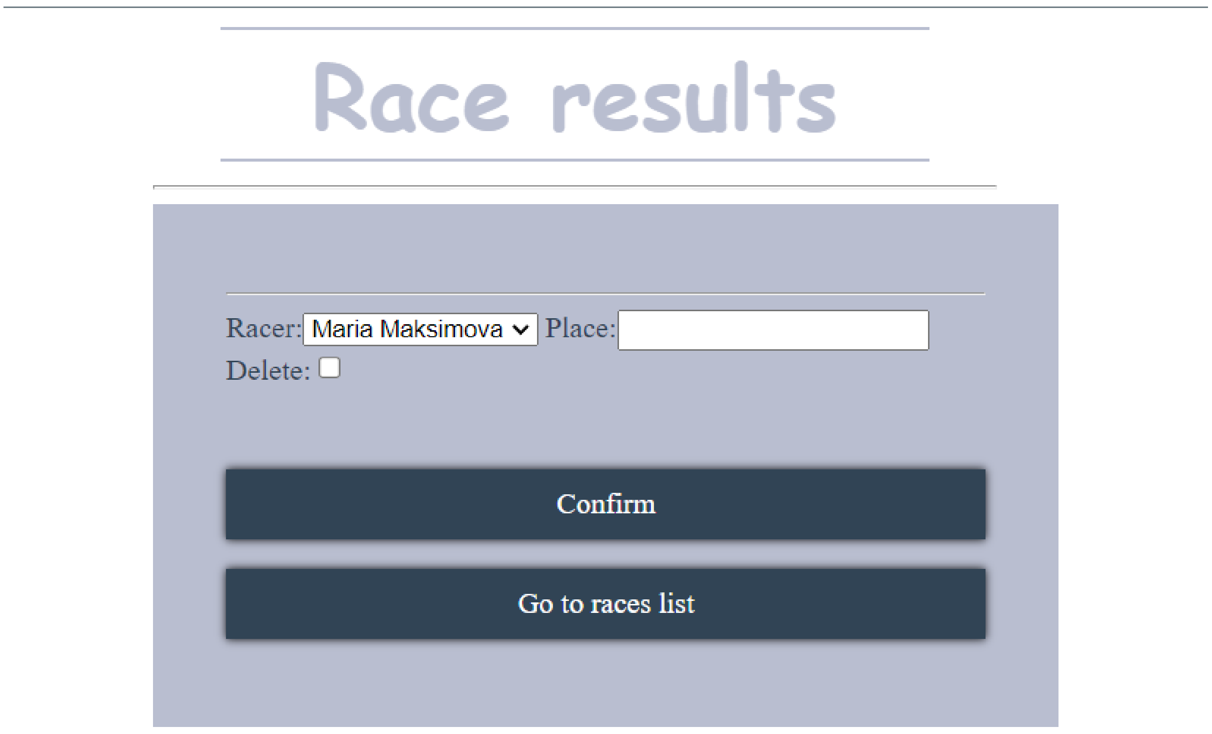
Edit the race

Race name
Moscow track

Date
2019-06-11

Length
3500

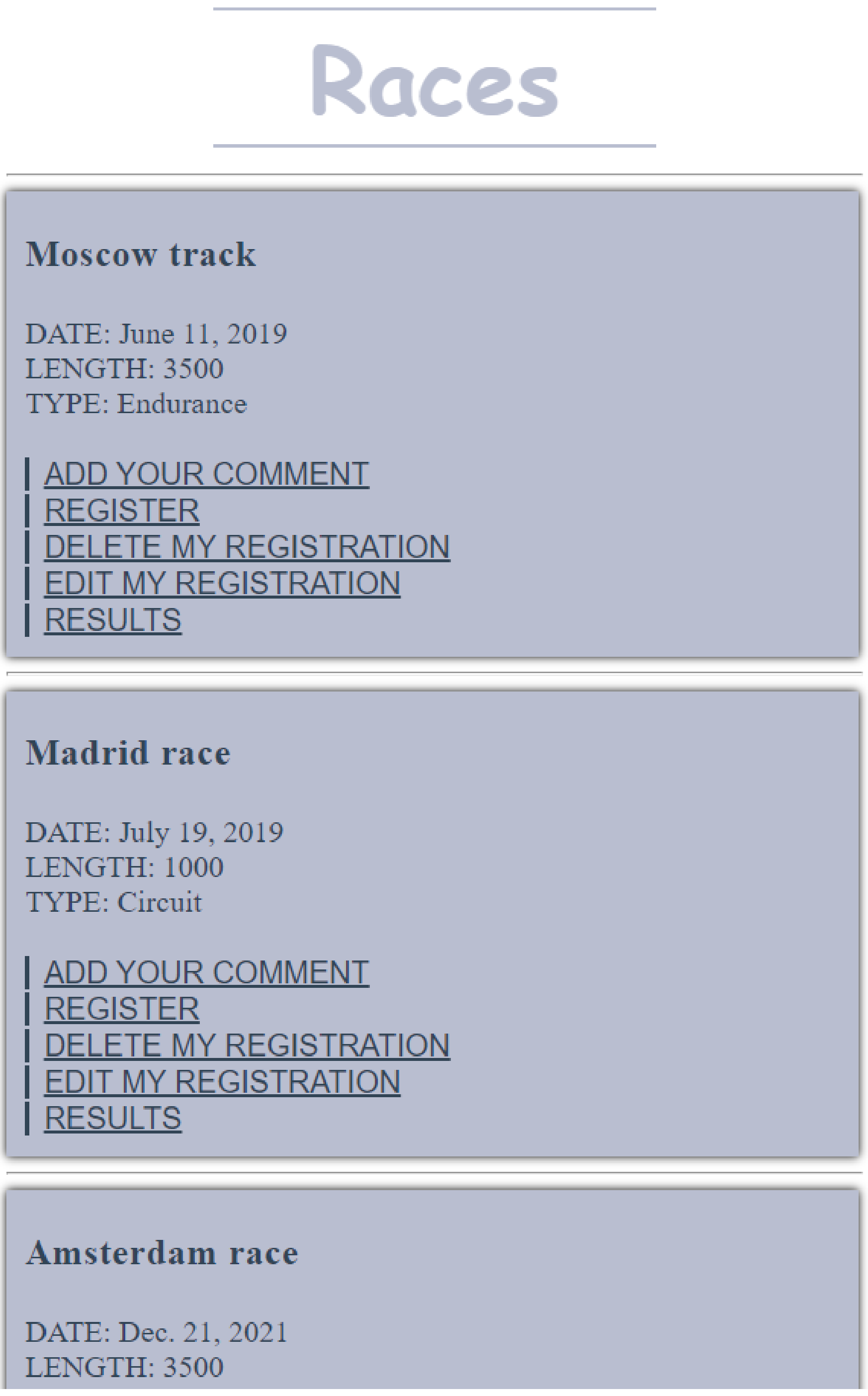
Type
Endurance race



Race results

Racer: Maria Maksimova Place:
Delete: ☐

Табло с гонщиками для гонщиков/обычных пользователей



Moscow track

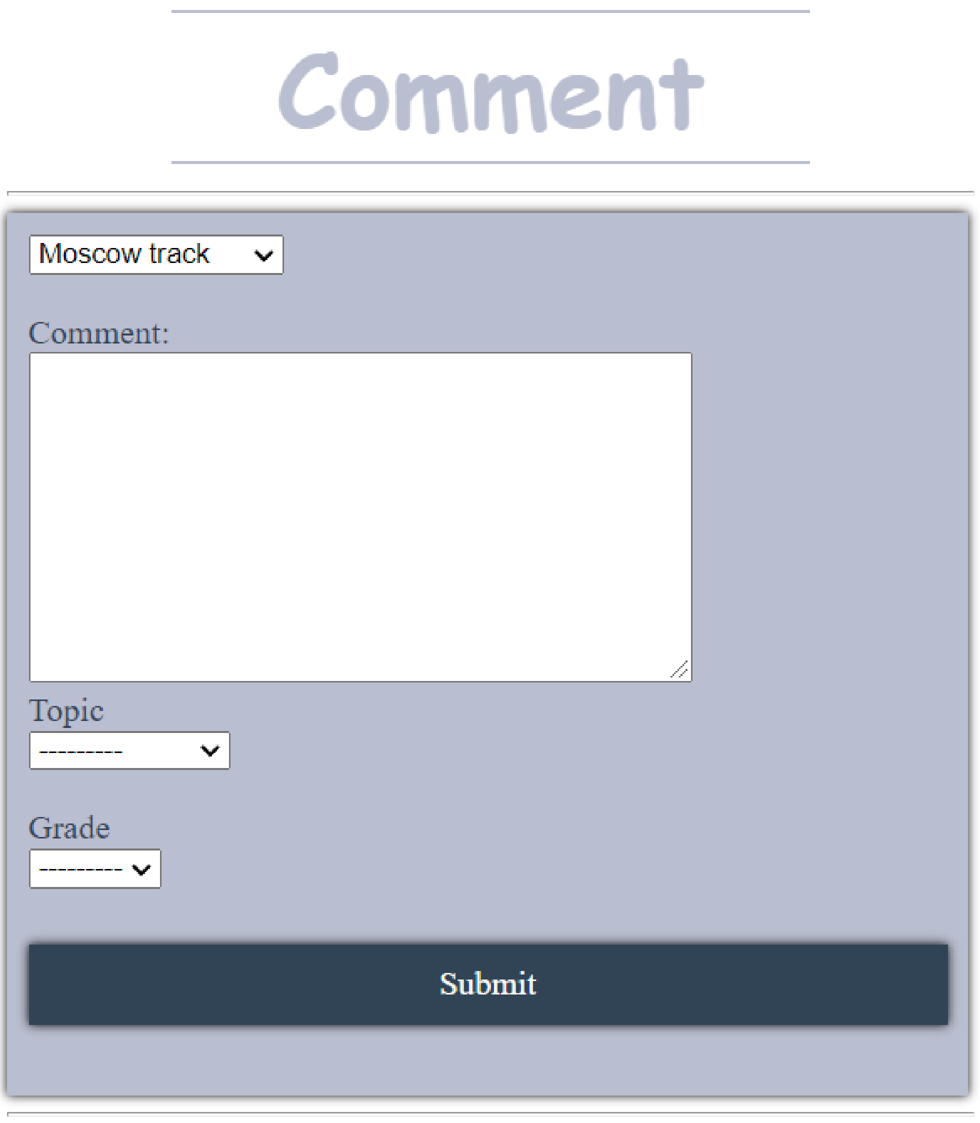
DATE: June 11, 2019
LENGTH: 3500
TYPE: Endurance

Madrid race

DATE: July 19, 2019
LENGTH: 1000
TYPE: Circuit

Amsterdam race

DATE: Dec. 21, 2021
LENGTH: 3500



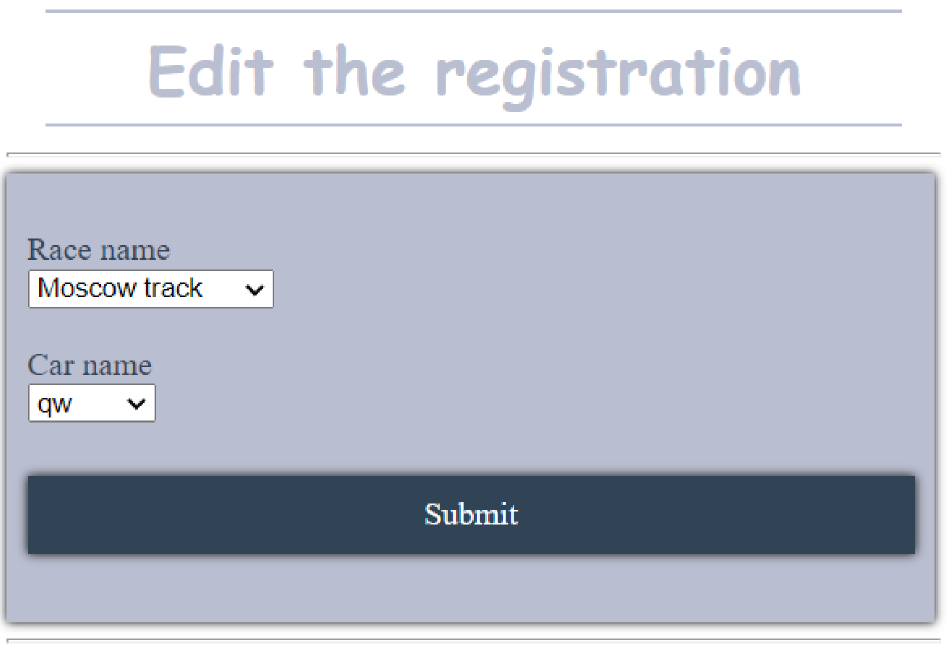
Comment

Moscow track

Comment:

Topic

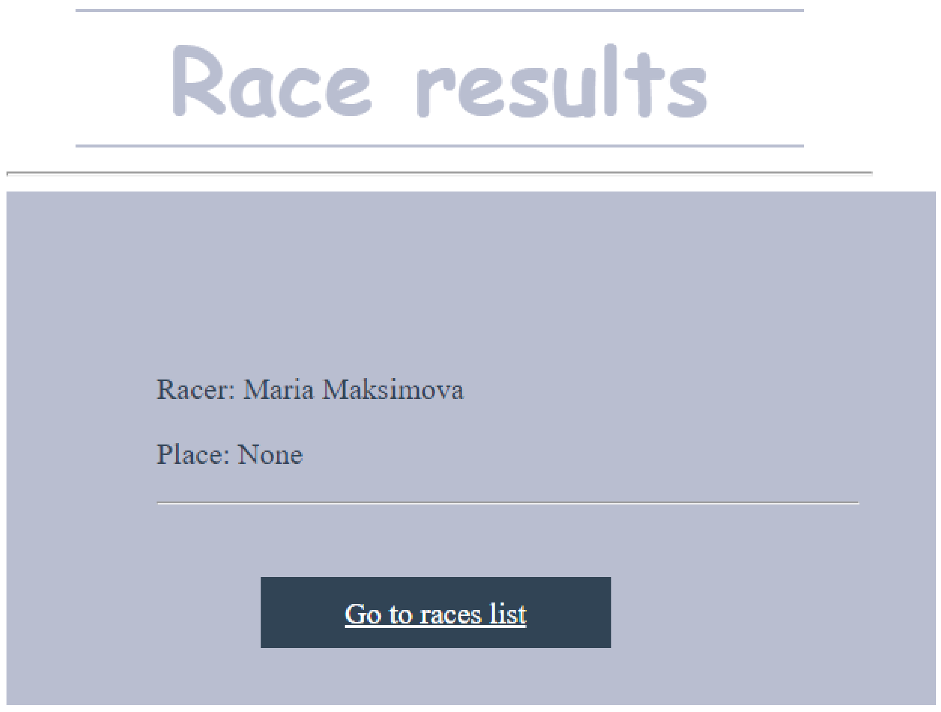
Grade



Edit the registration

Race name
Moscow track

Car name
qw



Race results

Racer: Maria Maksimova

Place: None