



# INTERNSHIP PROJECT

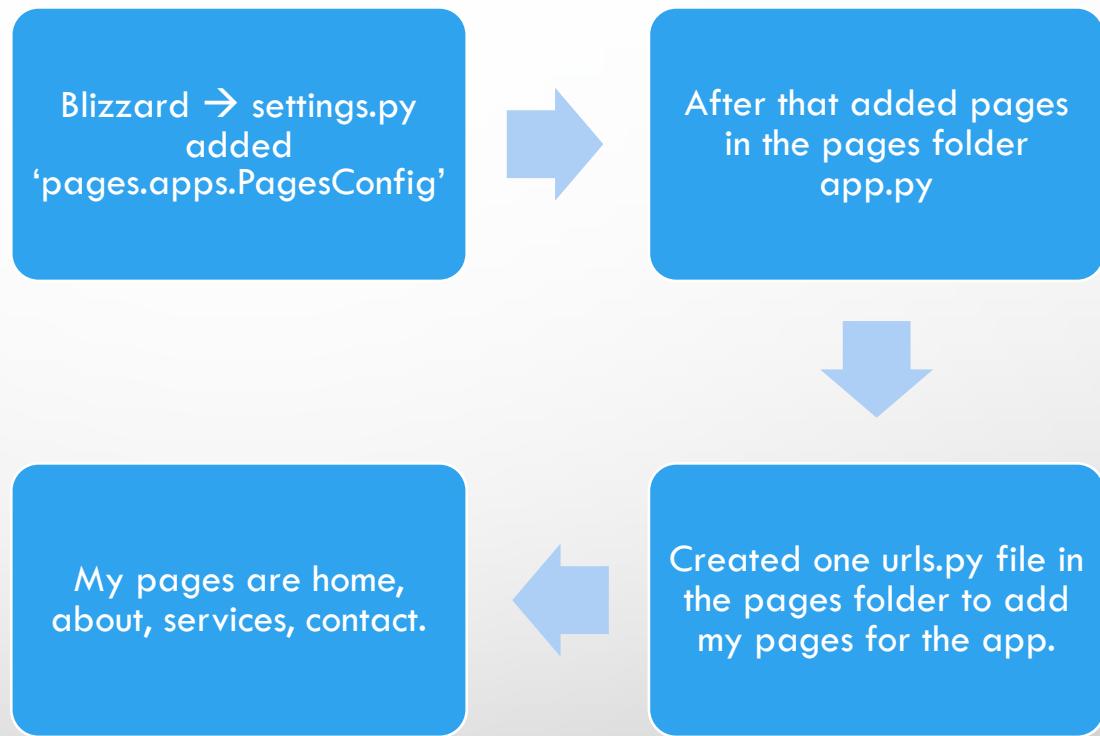
TECHNICAL PRESENTATION

BIZZARD AUTO

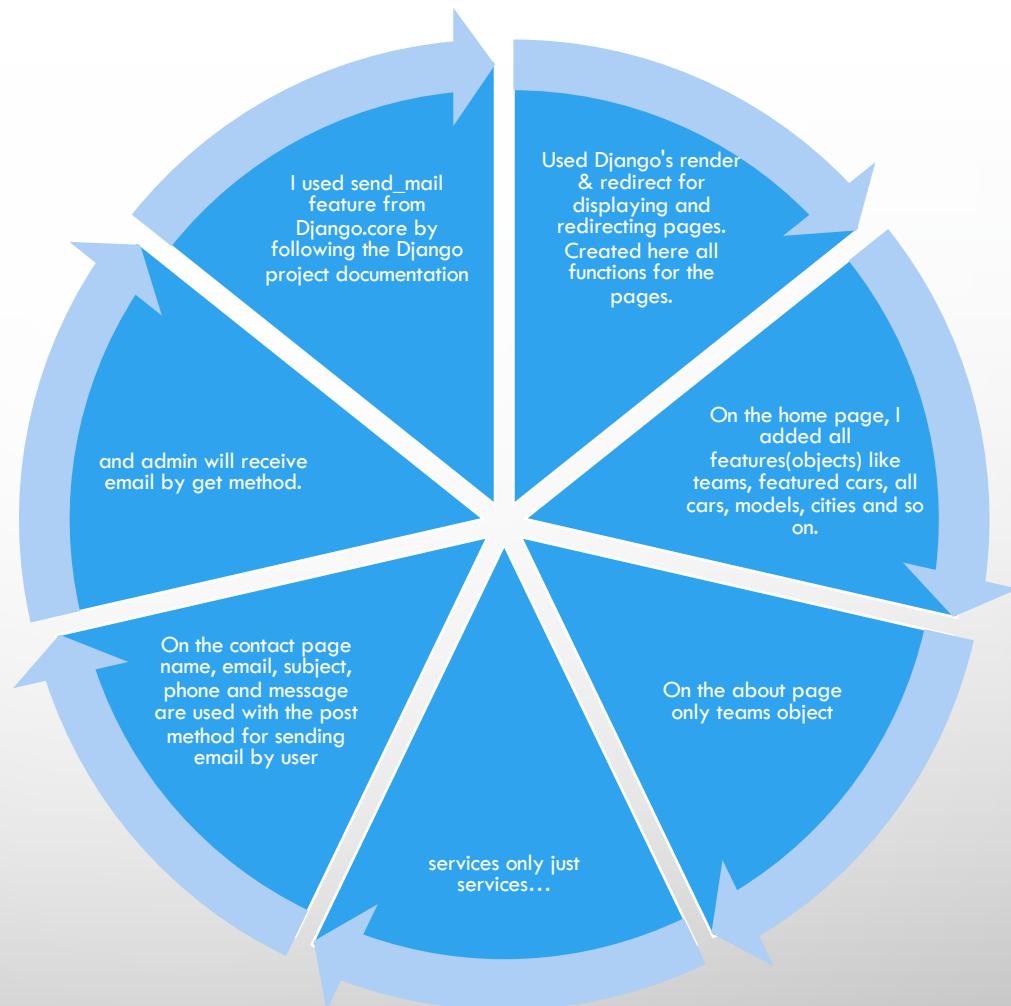


**Created app by running  
the command:**

`python manage.py startapp blizzard`



## Pages → views.py



blizzard →  
settings.py

Added:

```
TEMPLATES = [
{
    'DIRS': ['templates'],
}
]>>>>>>
```

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'blizzard/static'),
]
```

# root→templates

Here I created all pages and sub-pages folders

Main:	base.html
Accounts:	dashboard.html, login.html, register.html
Admin:	base_site.html
Cars:	car_detail.html, cars.html, search.html
Includes:	footer.html, messages.html, navbar.html, topbar.html
Pages:	home.html, about.html, services.html, contact.html



## In this part, I installed using pip

PostgreSQL

Psycopg2-binary (PostgreSQL DB adapter for Python)

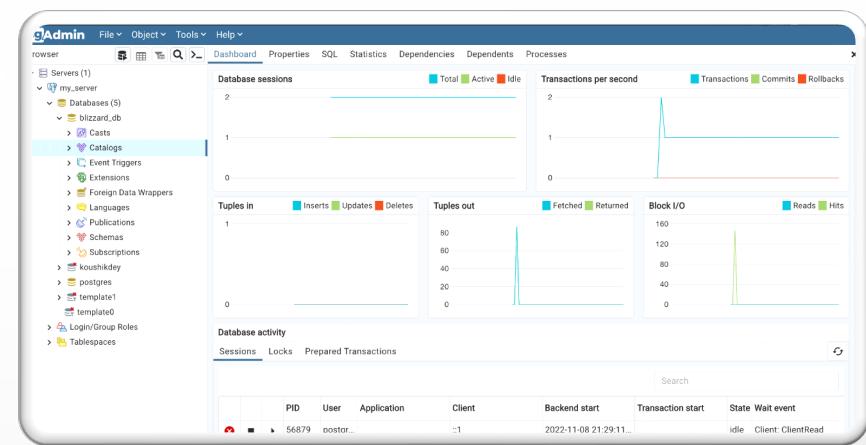
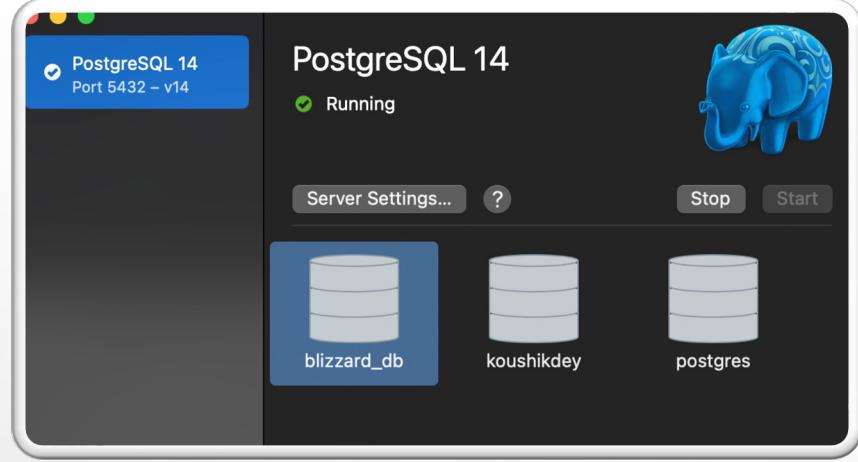
Pillow (For processing basic image functionality)

After that I created admin/superuser by running the command:

*python manage.py createsuperuser*

# PostgreSQL

```
settings.py M ×  
blizzard > settings.py > ...  
100  
101 # Database  
102 # https://docs.djangoproject.com/en/3.0/ref/settings/#databases  
103  
104 DATABASES = {  
105     'default': {  
106         'ENGINE': 'django.db.backends.postgresql',  
107         'NAME': 'blizzard_db',  
108         'USER': 'postgres',  
109         'PASSWORD': config('DB_PASSWORD'),  
110         'HOST': 'localhost',  
111     }  
112 }  
113
```



# PostgreSQL & pgAdmin panel

# CREATED DB MODEL FOR TEAM AND CLIENTS

THEN RUN COMMANDS TO  
SAVE MODELS IN  
POSTGRESQL:

- **python manage.py makemigrations**
- **python manage.py migrate**

Then registered these models to admin.py

```
koushik, last month | 1 author (koushik)
class Client(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    subject = models.CharField(max_length=250)
    phone = models.CharField(max_length=100)
    message = models.TextField(blank=True)

    def __str__(self):
        return self.name
```

```
100, 2 minutes ago | 2 authors (koushik and others)
# from distutils.command.upload import upload
from django.db import models

# Created models for Team and Clients.      Find related code in blizzard-django-project

koushik, 2 months ago | 1 author (koushik)
class Team(models.Model):
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    designation = models.CharField(max_length=255)
    photo = models.ImageField(upload_to='photos/%Y/%m/%d/')
    facebook_link = models.URLField(max_length=100)
    twitter_link = models.URLField(max_length=100)
    google_plus_link = models.URLField(max_length=100)
    created_date = models.DateTimeField(auto_now_add=True)
```

```
models.py in admin.py
pages > admin.py > ...
  1 # Register your models here.
  2 koushik, last month | 1 author (koushik)
  3 class TeamAdmin(admin.ModelAdmin):
  4     def thumbnail(self, object):
  5         return format_html('<img src={} width="40" style="border-radius: 10px;" />'.format(
  6             object.thumbnail.short_description = 'Photo'
  7
  8         list_display = ('id', 'thumbnail', 'first_name', 'last_name', 'designation', 'created_date')
  9         list_display_links = ('id', 'thumbnail', 'first_name')
 10         search_fields = ('first_name', 'last_name', 'designation')
 11         list_filter = ('designation')
 12
 13     admin.site.register(Team, TeamAdmin)
 14
 15 koushik, last month | 1 author (koushik)
 16 class ClientAdmin(admin.ModelAdmin):
 17     list_display = ('name', 'email', 'subject', 'phone', 'message')
 18     list_display_links = ('name', 'email', 'subject')
 19     search_fields = ('name', 'subject', 'email')
 20     list_per_page = 10
 21     list_filter = ('name', 'email', 'subject')
 22
 23     admin.site.register(Client, ClientAdmin)      Find related code in blizzard-django-project
```

```
models.py M urls.py X
blizzard > urls.py > ...
14     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18 from django.conf.urls.static import static
19 from django.conf import settings
20
21
22 urlpatterns = [  Find related code in blizzard-django-project
23     path('admin/', admin.site.urls),
24     path('', include('pages.urls')),
25     path('cars/', include('cars.urls')),
26     path('accounts/', include('accounts.urls')),
27     path('socialaccounts/', include('allauth.urls')),
28     path('contacts/', include('contacts.urls')),
29 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
30
```

```
STATIC_ROOT = static/
158 STATIC_ROOT = os.path.join(BASE_DIR, 'static')
159 STATICFILES_DIRS = [
160     os.path.join(BASE_DIR, 'blizzard/static'),
161 ]
162
163 # Media Settings
164
165 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
166 MEDIA_URL = '/media/'
167
168 # Messages
169
170 from django.contrib.messages import constants as messages
171 MESSAGE_TAGS = {
172     messages.ERROR: 'danger',
173 }
174
```

# MEDIA SETTINGS BLIZZARD

settings.py & urls.py

# HTML PAGES

HOME, ABOUT, SERVICES  
& CONTACT

```
> dj home.html
<!-- Search box 3 start -->
<div class="search-box-4 sb-8">
    <form action="{% url 'search' %}" method="">
        <div class="form-group">
            <input type="text" name="keyword" placeholder="Search" />
        </div>
        <div class="form-group">
            <select class="form-control search-fields" name="model">
                <option selected="true" disabled="disabled">Model
                {% for model in model_search %}
                    <option value="{{model}}>{{model}}</option>
                {% endfor %}
            </select>
        </div>
        <div class="form-group">
            <select class="form-control search-fields" name="city">
                <option selected="true" disabled="disabled">Location
                {% for city in city_search %}
                    <option value="{{city}}>{{city}}</option>
                {% endfor %}
            </select>
        </div>
        <div class="form-group">
            <select class="form-control search-fields" name="year">
                <option selected="true" disabled="disabled">Year
            </option>
        </div>
    </form>
</div>

class="carbox-overlap-wrapper">
<div class="overlap-box">
    <div class="overlap-btns-area">
        <div class="car-magnify-gallery">
            <a href="{{car.car_photo.url}}" class="overlap-btn">
                <i class="fa fa-expand"></i>
                
                
                
                
12              <div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
13                  <div class="carousel-inner banner-slider-inner text-center">
14                      <div class="carousel-item banner-max-height active item-bg">
15                          
16                          <div class="carousel-content container banner-info-2 bi-2 text-left">
17                              <% include 'includes/messages.html' %>
18                              <h3>Imagine the unimaginable</h3>
19                              <h5>Come and visit our high-tech performance cars that are made only for</h5>
20                              <a href="{% url 'services' %}" class="btn btn-lg btn-theme">Read more</a>
21                      </div>
22                  </div>
23                  <div class="carousel-item banner-max-height item-bg">
24                      
```

```
<h3>Create an account</h3>
{%
    include 'includes/messages.html'
%}
<form action="{% url 'register' %}" method="POST">
    {% csrf_token %}
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="text" name="firstname" class="input-text" value="Koushik" required="required">
        <i class="fa fa-user"></i>
        <input type="password" name="password1" class="input-text" value="Koushik123" required="required">
        <i class="fa fa-lock"></i>
    </div>
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="password" name="password2" class="input-text" value="Koushik123" required="required">
        <i class="fa fa-lock"></i>
    </div>
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="text" name="email" class="input-text" value="koushik@blizz.com" required="required">
        <i class="fa fa-envelope"></i>
    </div>
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="checkbox" name="terms" value="1" checked="checked" required="required">
        I accept the Terms of Service
    </div>
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="submit" value="Create Account" class="button">
    </div>
</form>
```

When I use the post method I need to use

{% CSRF\_TOKEN %}

```
<form action="{% url 'contact' %}" method="POST" enctype="multipart/form-data">
    {% include 'includes/messages.html' %}
    {% csrf_token %}      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
    <div class="row">
        <div class="col-lg-7">
            <div class="row">
                <div class="col-md-6 text-left">
                    <div class="form-group name">
                        <input type="text" name="name" value="Koushik" required="required" class="input-text">
                        <i class="fa fa-user"></i>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-lg-7">
            <div class="form-group message">
                <input type="text" name="message" value="Hello World" required="required" class="input-text" style="height: 100px;">
                <i class="fa fa-pencil"></i>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-lg-7">
            <div class="form-group file">
                <input type="file" name="attachment" class="input-file" style="width: 100%; height: 100px;">
                <i class="fa fa-paperclip"></i>
            </div>
        </div>
    </div>
    <div class="form-group form-box">      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="submit" value="Send Message" class="button">
    </div>
</form>
```

To prevent cross-site forgery

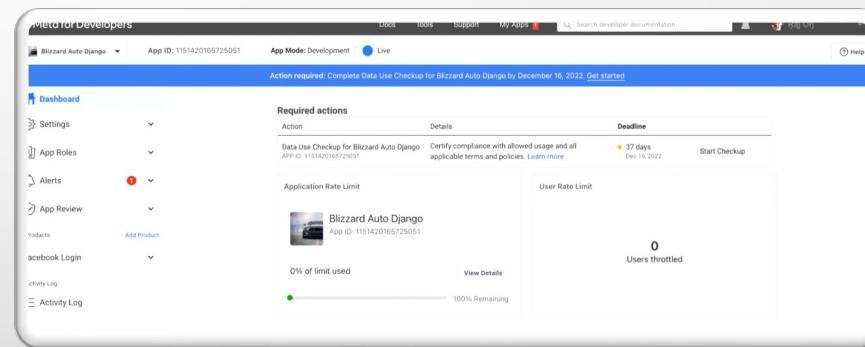
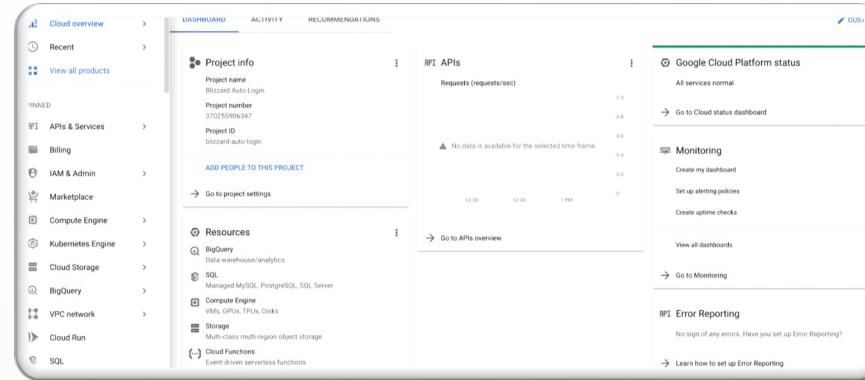
```
<li>
    <a href="javascript:{document.getElementById('logout').submit()}" class="sign-in"><i class="fa fa-sign-in"></i> Sign In</a>
    <form action="{% url 'logout' %}" id="logout" method="POST">
        {% csrf_token %}      Find related code in blizzard-django-project      koushik, last updated 2018-01-10
        <input type="hidden">
    </form>
</li>
{% else %}
```

# USER REGISTRATION

I didn't make a model here as Django comes with a built-in user model. and I didn't use hashing technique as Django is using sha256 hashing technique.



# SOCIAL LOGIN: GOOGLE & FB CONSOLE FOR DEVELOPERS



# LOGOUT FUNCTIONALITY

User authentication is by default  
enabled in Django, so it wasn't to call it  
from anywhere.



## INSTALLED DECOUPLE

`pip install python-decouple`

**Procedures in settings.py**

`from decouple import config`

**Created an env file in the root folder.**

`DEBUG = config('DEBUG', cast=bool)` which I mentioned as True in the .env file.

All other secret keys like APP\_SECRET\_KEY, DB\_PASSWORD and SMTP EMAIL\_HOST\_PASSWORD are saved in the .env file.

# **DATABASE BACKUP/DUMPDATA & REQUIREMENTS**

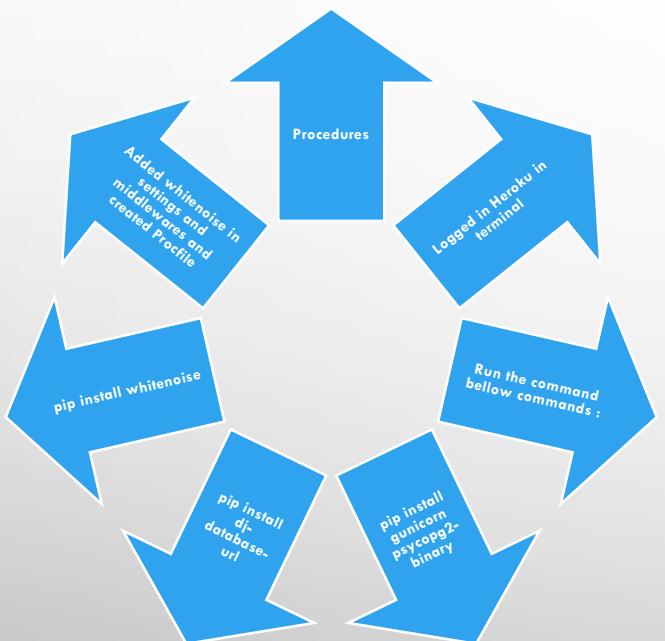
**Run these commands below to create a back-up in JSON file and all requirements for blizzard app**

```
python manage.py dumpdata --natural-foreign --natural-primary -e  
contenttypes -e auth.permission --indent 4 > project_dump.json
```

```
pip freeze > requirements.txt
```

# DEPLOY TO HEROKU

<https://blizzard-car.herokuapp.com/>



```
import os
import dj_database_url

from pathlib import Path
from decouple import config
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
]
```

```
tests.py      dj car_detail.html  Procfile  x
Procfile
koushik, last month | 1 author (koushik)
1  release: python manage.py migrate
2  web: gunicorn blizzard.wsgi
```

```
198 # whitenoise settings
199 STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
200
201 DEFAULT_AUTO_FIELD='django.db.models.AutoField'
```

## AFTER DEPLOYMENT PROCEDURES ACTIVATED ADMIN IN THE HEROKU SERVER

```
Command : heroku run python manage.py shell
```

```
>>> from django.contrib.sites.models import site
```

```
>>> site = site()
```

```
>>> site.domain = 'blizzard-car.herokuapp.com'
```

```
>>> site.name = 'blizzard-car.herokuapp.com'
```

```
>>> site.save()
```

```
>>> print(site.objects.get(name='blizzard-car.herokuapp.com').id)
```

# TESTING

The image shows two terminal windows side-by-side, both displaying Python test code. The left window is titled 'ContactModelTest' and the right window is titled 'ClientModelTest'. Both windows show code snippets for testing templates and database models.

```
projects / tests.py [1] ContactModelTest : ① test_string_car_id
16     def test_template_name_correct(self):
17         response = self.client.get(reverse("contact"))
18         self.assertTemplateUsed(response, "pages/contact.html")
19
20     def test_template_content(self):
21         response = self.client.get(reverse("contact"))
22         self.assertContains(response, "<h3>Email:</h3>")
23         self.assertNotContains(response, "Should not be here!")
24
25     # Contact model test
26
27     You, 45 minutes ago | author (You)
28     class ContactModelTest(TestCase):
29
30         def test_string_email(self):
31             contact = Contact(email = "Users email address", user_id = "Users ID")
```

```
tests > ② ClientModelTest : ③ test_string_car_id
60     def test_template_name_correct(self):
61         response = self.client.get(reverse("services"))
62         self.assertEqual(response.status_code, 200)
63
64     def test_template_content(self):
65         response = self.client.get(reverse("services"))
66         self.assertTemplateUsed(response, "pages/services.html")
67
68     def test_string_representation(self):
69         response = self.client.get(reverse("services"))
70         self.assertContains(response, "<h3>Super Fast</h3>")
71         self.assertNotContains(response, "Should not be here!")
72
73     # Model tests
74     You, 55 minutes ago | author (You)
75     class TeamModelTest(TestCase):
76
77         def test_string_representation(self):
78             team = Team(first_name = "first name", last_name = "last name")
```

- **python manage.py test**

# BLIZZARD AUTO

The image displays five screenshots of the Blizzard Auto software interface, arranged in a grid-like layout. The top row shows 'User Management' and 'Car Management'. The middle row shows 'Team Management' and 'Social Account Management'. The bottom row shows a search interface and a 'Latest Cars' section.

**User Management:**

- Shows a list of users categorized by role: Accounts, Authentication and Authorization, Cars, Contracts, Pages, and others.
- Includes a search bar and filter options.

**Car Management:**

- Shows a list of cars categorized by model: Accounts, Authentication and Authorization, Cars, Contracts, Pages, and others.
- Includes a search bar and filter options.

**Team Management:**

- Shows a list of teams categorized by role: Accounts, Authentication and Authorization, Cars, Contracts, Pages, and others.
- Includes a search bar and filter options.

**Social Account Management:**

- Shows a list of social accounts categorized by provider: Accounts, Authentication and Authorization, Cars, Contracts, Pages, and others.
- Includes a search bar and filter options.

**Search Interface:**

- A modal window titled 'Select car to change' with fields for Model, Location, Year, and Type.
- Includes a search bar and filter options.

**Latest Cars:**

- A section titled 'Latest Cars' featuring a grid of six car models: Porsche Taycan Sport Turismo, Volvo V60, Audi e-tron 5-line quattro, Ford Mustang Mach-E, Volvo XC90, and BMW X1.
- Each car card includes details like price, mileage, year, and body style.