ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

ỨNG DỤNG PYTHON VÀO LẬP TRÌNH WEBSITE

GVHD: TS. NGUYỄN ĐỨC THÁI
GVPB: ThS. NGUYỄN HỒNG NAM

SVTH: NGUYỄN PHƯỚC THỊNH (1413785)

TP. HỒ CHÍ MINH, THÁNG 5/2019

Mục lục

Danh sách bảng

Danh sách hình vẽ

Chương I

Giới thiệu đề tài

1 Tính cấp thiết

Với sự lớn mạnh không ngừng của Internet và độ phổ biến của nó trên toàn Thế giới, việc các doanh nghiệp và cửa hành kinh doanh sử dụng nó để mang lại doanh thu cho mình là hết sức quan trọng. Tuy nhiên, có rất nhiều đơn vị muốn chiếm lĩnh vị trí cao trên nó dẫn đến việc cạnh tranh rất gay gắt trong công cuộc đưa website của mình đến với đại đa số khách hàng ở khắp mọi nơi, được gọi tắt là SEO (Search Engine Optimization).

Từ đó việc tối ưu trang web để mang lại thứ hạng cao trên các công cụ tìm kiếm ra đời. Người lập trình viên ngoài việc phải đáp ứng các kịch bản của ứng dụng mà còn phải đảm bảo tối ưu website của họ đối với các công cụ tìm kiếm. Để giúp cho công việc của họ được thuận tiện hơn và tuân thủ được các tiêu chí tối ưu mới nhất, chúng tôi cung cấp một giải pháp tự động, phân tích cú pháp trang web và đưa ra những đề nghị sửa chữa, nhằm tối ưu SEO cho trang web của họ.

2 Mục tiêu

Để mang đến sự thuận tiện và hài lòng cho người sử dụng, ứng dụng của chúng tôi sẽ đặt ra những mục tiêu sau đây:

- Cung cấp dịch vụ đánh giá miễn phí! Vì thế chúng tôi sẽ đặt quảng cáo lên trang web để mang lại nguồn thu duy trì cho chúng tôi.
- Cập nhật những tiêu chuẩn về SEO đến người dùng thông qua các bài viết trên website và đảm bảo rằng công cụ của chúng tôi sẽ sử dụng những tiêu chuẩn mới nhất.
- Tự động phân tích cú pháp trang web người dùng, so sánh với các tiêu chuẩn về SEO, đưa ra kết quả, đánh giá và gợi ý cho người dùng cách để họ có thể khắc phục nếu website chưa đạt chuẩn hoặc thiếu các thành phần quan trọng.

3 Phương pháp thực hiện

Chúng tôi sử dụng Python để làm ngôn ngữ lập trình cốt lõi cho ứng dụng của mình. Python được biết đến là ngôn ngữ dành cho tính toán và phân tích nên sẽ thích hợp để xử lý các cú pháp cho trang web chúng tôi cần kiểm tra.

Bên cạnh đó, với kho thư viện đồ sộ được chia sẻ công khai trên `https://pypi.org` sẽ giúp chúng tôi triển khai nhanh chóng ứng dụng nhờ vào các framework mở được chia sẻ miễn phí để sử dụng.

Cụ thể hơn, sau đây là những framework và thư viện chúng tôi sẽ sử dụng cho ứng dụng của mình:

- Django: Framework Python dùng để phát triển ứng dụng web.
- Requests, Lxml: Thư viện Python có nhiệm vụ phân tích cú pháp và lấy từng phần tử của website chúng tôi cần kiểm tra.
- reCAPTCHA: Ứng dụng do Google phát triển, nhằm hạn chế spam và BOT tác động lên website để giữ trang web trở nên an toàn.
- Bootstrap: Thư viện dùng để thiết kế giao diện cho trang web của chúng tôi, nó hỗ trợ tốt cho việc hiển thị website đa nền tảng.
- Font Awesome: Thư viện cung cấp các icon cần thiết cho giao diện website.
- Heroku: Cloud platform miễn phí để chúng tôi triển khai ứng dụng Python của mình lên Internet.

4 Bố cục báo cáo

Tiếp theo, những phần sau chúng tôi sẽ trình bày cách chúng tôi sử dụng những framework và thư viện để xây dựng nên website cùng với những tiêu chí được áp dụng để đánh giá SEO cho một trang web. Phần kết luận đánh giá sẽ được giới thiệu vào mục cuối cùng kèm theo những tài liệu tham khảo mà chúng tôi sử dụng để hoàn thành báo cáo này.

Chương II

Nền tảng lý thuyết

Phần này chúng tôi sẽ giới thiệu cũng như trang bị những kiến thức cơ bản để sử dụng những công cụ được liệt kê ở phần trước.

1 Ngôn ngữ Python

Python hiện đang là một trong những ngôn ngữ lập trình phổ biến. Một phần nhờ vào khả năng dễ tiếp cận, cấu trúc rõ ràng và quan trọng hơn, nó có thể giải quyết tốt các bài toán kỹ thuật với thời gian thực thi nhanh và tiết kiệm dòng code. Python được tạo ra bởi Guido van Rossum và phát hành vào năng 1991.[1]

Phiên bản sử dụng: 3.7.3

- Biến (Variable): Không giống với các ngôn ngữ khác, Python không có câu lệnh riêng biệt để khai báo biến. Biến không cần phải khai báo kiểu giá trị nào và có thể thay đổi dựa vào giá trị mà nó được gán.

```
# x is of type int
x = 5
# x is now of type str
x = "Thinh"
```

- Chuỗi (String): Chuỗi ký tự trong Python được chứa trong cặp dấu nháy đơn hoặc dấu nháy kép. Để hiển thị chuỗi ra màn hình, sử dụng lệnh `print()`.

```
a = "Hello, World!"
print(a)

>>>"Hello, World"
```

- Toán tử (Operator):
  - Số học:
  - So sánh:

+	Cộng
−	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư
**	Lũy thừa
//	Chia lấy phần nguyên

Bảng II.1: Toán tử Số học

==	Bằng
!=	Không bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

Bảng II.2: Toán tử So sánh

– Logic:

and	Trả về `True` nếu 2 điều kiện đều đúng
or	Trả về `True` nếu 1 trong 2 điều kiện là đúng
not	Đảo ngược kết quả của điều kiện

Bảng II.3: Toán tử Logic

– Identity:

is	Trả về `True` nếu 2 biến cùng trỏ tới 1 đối tượng
is not	Trả về `True` nếu 2 biến không trỏ cùng đối tượng

Bảng II.4: Toán tử Identity

– Membership:

in	Trả về `True` nếu biến nằm trong tập hợp các biến
not in	Trả về `True` nếu biến không nằm trong tập hợp các biến

Bảng II.5: Toán tử Membership

- Dictionary: Tập hợp không có thứ tự, có thể thay đổi và lập chỉ mục. Được biểu diễn bằng cặp dấu ngoặc nhọn, bên trong là khóa (key) và giá trị (value) tương ứng.

```
hoten = {
  "ho": "Nguyen Phuoc",
  "ten": "Thinh"
}
```

- Câu điều kiện (If...Else): Dùng để thực thi một hành động sau khi thỏa điều kiện cho trước. Lưu ý trong Python, sử dụng thụt lề dòng để phân biệt các khối lệnh với nhau.

```
a = 1
b = 2
if a > b:
  print("a is greater than b")
elif a == b:
  print("a and b are equal")
else:
  print("b is greater than a")

>>>"b is greater than a"
```

- Vòng lặp (For): Dùng để lặp qua một chuỗi (có thể là list, tuple, dictionary, set hoặc string).

```
hoten = ["Nguyen", "Phuoc", "Thinh"]
for x in hoten:
  print(x)

>>>"Nguyen"
>>>"Phuoc"
>>>"Thinh"
```

- Hàm (Function): Gồm một khối code, được khởi chạy khi được gọi đến. Để truyền dữ liệu vào 1 hàm được gọi là tham số (parameter). Hàm trả về kết quả thông qua lệnh `return`.

```
# a function is defined using the def keyword
def add(n):
  return 1 + n
# calling a function
add(1)

>>>2
```

- Lớp/Đối tượng (Class/Object): Python là ngôn ngữ lập trình hướng đối tượng. Hầu hết mọi thứ trong Python là một đối tượng (object), gồm thuộc tính và phương thức của nó. Sử dụng lớp (class) để khởi tạo 1 đối tượng mới.

```
# create a class
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
  # object method
  def func(self):
    print(f"My name is: {self.name}, {self.age} years old")
# create object
p = Person("Thinh", 20)
```

```
p.func()
# modify object property
p.age = 22
print(p.age)

>>>"My name is: Thinh, 20 years old"
>>>22
```

- Module: Có thể xem module là 1 bộ thư viện mã code, được lưu bởi tệp hoặc thư mục tách biệt với project đang thực thi, được nhúng vào để tái sử dụng những bộ code chứa trong đó.

  Để sử dụng các hàm hoặc lớp trong file `mymodule.py`, ta sử dụng lệnh sau:

```
import mymodule
# import only part from a module
from mymodule import myfunc
```

- PIP: Là trình quản lý gói (package) hoặc module dành cho Python. Gói là nơi chứa tất cả các file cần thiết cho 1 module.

  Để cài đặt 1 gói trong Python, sử dụng lệnh sau:

```
\>pip install Django
```

- Xử lý ngoại lệ (Try...Except): Khi chương trình xảy ra lỗi hoặc ngoại lệ, Python sẽ dừng lại và đưa ra thông báo lỗi cho người dùng. Để tránh ứng dụng bị gián đoạn, sử dụng câu lệnh `try` để bắt và xử lý các ngoại lệ khi chương trình đang được thực thi.

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")

>>>"Variable x is not defined"
```

2 Framework Django

Django là 1 framework Python web cấp cao, thúc đẩy phát triển nhanh chóng, gọn gàng và tiện dụng. Được xây dựng bởi các nhà lập trình viên có kinh nghiệm, xử lý được các vấn đề rắc rối khi phát triển web, do đó người dùng chỉ cần quan tâm hoàn thiện các chức năng cho web mà không cần phải quá lo lắng về nền tảng phía sau. Và quan trọng nó là mã nguồn mở và miễn phí.

Chúng tôi sẽ sử dụng mục tài liệu[2] tại trang chủ của Django framework để trình bày những khái niệm và cách để hiện thực ứng dụng của chúng tôi.

Phiên bản sử dụng: 2.1.4

- Sau khi cài đặt xong Django từ trình quản lý gói PIP của Python, dùng lệnh sau để khởi tạo project mới có tên là `lvtn`:

```
\>django-admin startproject lvtn
```

Một folder mới được tạo ra chứa các thành phần của project, có cấu trúc và chức năng như sau:

```
lvtn\
  lvtn\
    __init__.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

  - `manage.py`: Một CLI giúp tương tác với ứng dụng web.
  - `lvtn\__init__.py`: File rỗng, để chỉ cho Python biết thư mục này nên được xem là một gói.
  - `lvtn\settings.py`: Chứa các tùy chỉnh của project.
  - `lvtn\urls.py`: Các khai báo URL cho trang web.
  - `lvtn\wsgi.py`: Được sử dụng khi deploy project lên Internet.

- Server phát triển: Dùng để khởi chạy ứng dụng web trên máy tính local.

```
\>python manage.py runserver
```

Khi server đang chạy, truy cập vào địa chỉ `http://127.0.0.1:8000` trên trình duyệt web để thấy ứng dụng đang được trình diễn.

- Tạo app mới: Mỗi ứng dụng được viết trong Django bao gồm 1 gói Python tuân thủ theo 1 quy ước nhất định. Django đi kèm với 1 tiện ích tự động tạo cấu trúc thư mục cơ bản của 1 ứng dụng, do đó người lập trình chỉ cần quan tâm đến việc phát triển code bên trong mà thôi.

  Tạo app `checkweb` có nhiệm vụ xử lý chính cho project của chúng tôi.

```
\>python manage.py startapp checkweb
```

Thư mục mới được tạo ra có cấu trúc như sau:

```
checkweb\
  migrations\
    __init__.py
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  views.py
```

- `migrations\`: Thư mục chứa các file được sinh ra khi có thay đổi về cấu trúc cơ sở dữ liệu.
- `admin.py`: Dùng để thiết đặt các thuộc tính được hiển trị trong trang quản trị admin mà Django cung cấp sẵn.
- `apps.py`: Khai báo app được sử dụng trong project, đảm bảo rằng các app không bị trùng lặp trong 1 dự án.
- `models.py`: Django hỗ trợ các phương thức để xử lý cơ sở dữ liệu mà không cần sử dụng đến các câu lệnh truy vấn SQL trực tiếp.
- `tests.py`: Được người dùng sử dụng để triển khai các kịch bản thử nghiệm và rà soát lỗi trước khi phát hành ứng dụng.
- `views.py`: Đóng vai trò xử lý trung tâm của ứng dụng, quản lý việc hiển thị, kết nối đến cơ sở dữ liệu và thực thi các hàm do lập trình viên thêm vào ứng dụng.

  Sau khi tạo xong app, cần phải khai báo trong project bằng các thêm dòng sau vào file `lvtn\settings.py`:

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    # add code below
    "checkweb.apps.CheckwebConfig",
]
```

- Migration: Khi tạo mới hoặc có sự thay đổi về cấu trúc cơ sở dữ liệu, migration có nhiệm vụ lưu lại quá trình ứng dụng thay đổi, do đó có thể truy vết lại và phục hồi lại những cập nhật trước đó 1 cách dễ dàng, nhất là khi ứng dụng gặp lỗi. Để thực hiện, dùng các lệnh sau:

```
\>python manage.py migrate
\>python manage.py makemigrations
\>python manage.py migrate
```

- Trang quản trị admin: Một trong những ưu điểm của Django so với các framework khác là nó cung cấp trang Django administration giúp hiển trị trực quan cơ sở dữ liệu do người dùng thiết lập, cho phép xem, tạo mới, chỉnh sửa, xóa và nhiều tính năng khác nữa.

  Để có thể truy cập vào trang quản trị thì trước tiên cần phải tạo tài khoản `superuser`:

```
\>python manage.py createsuperuser
\>Username: ___
\>Email address: ___
\>Password: ___
\>Password (again): ___
\>Superuser created successfully.
```

Sau khi điền đầy đủ các thông tin trên thì có thể đăng nhập tài khoản để truy cập vào trang quản trị tại địa chỉ: `http://127.0.0.1:8000/admin/`

- Class-based view: Đây là chức năng được Django hỗ trợ, giúp lập trình viên ít phải viết code hơn để hiện thị 1 giao diện lên trình duyệt web. Nó hỗ trợ tốt trong việc truyền tham số, lấy giá trị từ model và có thể dễ dàng tùy chỉnh theo ý muốn.

  Để sử dụng, cần phải thêm module vào file muốn dùng nó. Đoạn code sau có chức năng hiển thị file `about.html` ra đường dẫn `http://127.0.0.1:8000/about/`.

```
from django.urls import path
from django.views.generic import TemplateView

urlpatterns = [
  path("about/", TemplateView.as_view(template_name="about.html")),
]
```

- Django template: Dựa trên file `.html` nhưng có chèn thêm các đoạn code riêng biệt để mỗi khi chạy chương trình, Django sẽ render ra giao diện lên trình duyệt tương ứng.

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
<h1>{{ section.title }}</h1>
{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:100 }}</p>
{% endfor %}
{% endblock %}
```

3 Thư viện Python

3.1 Requests

Tham khảo[3].

Phiên bản sử dụng: 2.21.0

- Cách cài đặt:

```
\>pip install requests
```

- Phương thức `get`: Dùng để lấy toàn bộ nội dung trang web dựa trên tham số url.

```
import requests
page = requests.get(url)
```

3.2 Lxml

Tham khảo[4].

Phiên bản sử dụng: 4.3.3

- Cách cài đặt:

```
\>pip install lxml
```

- Gói `lxml.html`: Dùng để phân tách chuỗi HTML.

```
from lxml import html
content = html.fromstring(page.content)
value = content.xpath("//title/text()")
```

4 Thư viện giao diện

4.1 Bootstrap

Tham khảo[5].

Phiên bản sử dụng: 4.3.1

- CSS: Sao chép và dán dòng code bên dưới vào trong thẻ `<head>` trước tất cả các định dạng khác để tải CSS của Bootstrap.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/
4.1.3/css/bootstrap.min.css">
```

- JS: Đặt trong thẻ `<script>` ở gần cuối trang web, trước khi đóng thẻ `</body>` để kích hoạt chúng. jQuery phải được đặt trước, đến Popper.js và sau cùng là phần JavaScript.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/
popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/
bootstrap.min.js"></script>
```

4.2 Font Awesome

Tham khảo[6].

Phiên bản sử dụng: 5.8.2

Trước khi sử dụng được, cần phải chèn dòng code bên dưới vào thẻ `<head>` nằm ở đầu trang web.

```
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/
all.css">
```

Để chèn icon vào trang web, sử dụng dòng code tương tự cú pháp bên dưới:

```
<i class="fas fa-heart"></i>
```

5 Bảo mật

5.1 reCAPTCHA

Tham khảo[7].

Phiên bản sử dụng: reCAPTCHA v2

Cách đơn giản để sử dụng reCAPTCHA vào trang web bằng cách nhúng mã JavaScript và thẻ `g-recaptcha`. Thẻ `g-recaptcha` là 1 thẻ `DIV` với tên class là `"g-recaptcha"`, có thuộc tính `data-sitekey` chứa Site key được cấp khi đăng ký sử dụng reCAPTCHA.

```
<html>
  <head>
    <title>reCAPTCHA demo: Simple page</title>
    <script src="https://www.google.com/recaptcha/api.js" async defer></script>
  </head>
  <body>
    <form action="?" method="POST">
      <div class="g-recaptcha" data-sitekey="your_site_key"></div>
      <br/>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

Sau khi submit form sử dụng reCAPTCHA, cần phải gửi giá trị có tên là `g-recaptcha-response` bằng phương thức POST về máy chủ của Google để xác minh người dùng đã xác thực bằng reCAPTCHA tại địa chỉ `https://www.google.com/recaptcha/api/siteverify`.

- secret (bắt buộc): Khóa Secret key được cấp đồng thời với Site key khi đăng ký sử

dụng.

- response (bắt buộc): Kết quả trả về của thuộc tính có tên là `g-recaptcha-response`.
- remoteip: Địa chỉ IP của người dùng cuối.

Tiếp theo, Google sẽ trả về kết quả kiểm tra có thỏa reCAPTCHA dưới dạng JSON bằng các giá trị sau:

```
{
  "success": true|false,
  "challenge_ts": timestamp, # timestamp of the challenge load
  "hostname": string, # the hostname of the site where the reCAPTCHA was solved
  "error-codes": [...] # optional
}
```

Dựa vào kết quả này, chúng tôi có thể biết được người dùng đã có giải được captcha hay chưa, sau đó tiến hành các yêu cầu từ người dùng.

6 Triển khai ứng dụng

6.1 Heroku platform

Tham khảo[8].

Đầu tiên và quan trọng nhất, các ứng dụng Heroku yêu cầu 1 file `Procfile` để cài đặt nền tảng sử dụng, được đặt tại thư mục gốc.

`Procfile`

```
web: gunicorn myproject.wsgi
```

File `Procfile` này yêu cầu `Gunicorn`, 1 máy chủ web được khuyến nghị dùng cho ứng dụng Django. Để cài đặt, sử dụng lệnh:

```
\>pip install gunicorn
```

Thay đổi trong file `settings.py` của ứng dụng Django. Khi sử dụng Heroku, các thông tin nhạy cảm sẽ được lưu trữ trong môi trường được gọi là config vars. Nó bao gồm các thông tin để kết nối đến cơ sở dữ liệu, trong khi bình thường sẽ được ghi trong file `settings.py` của Django.

Gói `django-heroku` sẽ tự động cấu hình ứng dụng Django để nó hoạt động trên Heroku. Nó tương thích với các ứng dụng Django 2.0. Để cài đặt, sử dụng lệnh:

```
\>pip install django-heroku
```

Sau khi cài đặt, cần phải `import` câu lệnh sau vào đầu file `settings.py`:

```
import django_heroku
```

Sau đó thêm phần sau vào cuối file `settings.py`:

```
# activate django-heroku.
django_heroku.settings(locals())
```

Triển khai ứng dụng và hoàn tất.

Chương III

Các tiêu chuẩn SEO

Để thu hút khách hàng truy cập và sử dụng webstie của mình, thì phải cần rất nhiều tiêu chí như giao diện đẹp, đáp ứng nhanh, thuận tiện mang đến trải nghiệm tốt cho người dùng. Tuy nhiên, ứng dụng của chúng tôi không đánh giá về mặt người dùng mà nó được dùng tối ưu để triển khai cho các công cụ tìm kiếm, giúp cho trang web được thứ hạng tốt hơn để người dùng có thể dễ dàng nhìn thấy, còn việc giữ chân khách hàng và sử dụng dịch vụ thì nó không là nhiệm vụ hàng đầu ứng dụng của chúng tôi.

Sau đây là các tiêu chuẩn cần thiết cho 1 website chuẩn SEO[9]:

1 Tên miền và hosting

Khi lựa chọn tên miền và hosting, tốt nhất nên lựa chọn tên liên quan đến nội dung website hoặc liên quan đến ngành nghề của cửa hàng hay doanh nghiệp. Tên miền được sử dụng lâu sẽ được đánh giá cao, bên cạnh đó nên chọn tên miền ngắn gọn, dễ nhớ. Song song với tên miền thì hosting là yếu tố đi kèm quan trọng không thể thiếu, hosting có tốc độ nhanh, hoạt động ổn định với băng thông và dung lượng thích hợp, ít khi bị gián đoạn là một trong những yếu tố giúp website thân thiện hơn với công cụ tìm kiếm.

2 Tốc độ tải trang

Tốc độ tải trang của một website là một yếu tố quan trọng trong việc thiết kế một trang web chuẩn SEO, vì vậy việc tối ưu cấu trúc code web giúp việc tải trang nhanh hơn. Thông thường một website được đánh giá cao khi có tốc độ tải trang trung bình từ 0.5 giây đến 2 giây. Một website có tốc độ tải trang chậm có thể sẽ làm cho người dùng cảm thấy khó chịu, đồng thời việc xếp hạng trên bảng công cộng tìm kiếm của Google cũng sẽ gặp khó khăn vì vậy cần phải chú ý vào tốc độ tải của một trang web khi thiết kế.

3 URL website

Bên cạnh tốc độ tải trang thì việc tối ưu URL website là điều quan trọng không kém, địa chỉ URL cần xuất hiện một cách rõ ràng và thân thiện ngay trên các thanh công cụ tìm kiếm. Khi thiết kế website chuẩn SEO thì phải xây dựng một URL có khả năng tùy biến và thân thiện.

Một số yếu tố cần thiết khi xây dựng một URL tốt:

- URL nên mã hóa theo tiêu đề bài viết, có liên quan tới nội dung bài viết, keywords, description.
- Dễ dàng thêm canonical URL cho những trang có nội dung bị trùng lặp nhiều.
- Sử dụng rewrite URL đối với các liên kết và phân tách mỗi từ bằng dấu gạch nối.
- Chỉ có 1 trong 2 có đuôi `www` hoặc không có `www` trước URL để tránh bị phạt nội dung trùng lặp.
- Không dùng các loại URL tự động có số bên trong như `www.example.com/?p=56789`.
- Có 1 file `robots.txt` chuẩn để điều khiển các crawler 1 cách hợp lý.

4 Thẻ tiêu đề và mô tả

Thẻ tiêu đề (title) và thẻ mô tả (meta description) được đánh giá là hai thẻ quan trọng nhất đối với một website.

- Thẻ title cần đảm bảo ngắn gọn, xúc tích, duy nhất và không nên dài quá 65 ký tự.
- Thẻ meta description phải tóm tắt được nội dung chính trên website, hay một bài viết một cách ngắn gọn, đầy đủ thông tin và đặt biệt là không được dài quá 150 ký tự.

5 Sitemap và robots.txt

Khi thiết kế website chuẩn SEO không thể bỏ qua sơ đồ trang (sitemap) giúp phân chia cấu trúc website và cấu trúc URL khi duyệt website tốt hơn. Ngoài sitemap, website còn cần có thêm file `robots.txt` giúp crawler của các bộ máy tìm kiếm có thể thu thập thông tin từ website một cách đầy đủ, nhanh chóng và thuận thiện hơn.

6 Thiết bị di động

Website thân thiện với thiết bị di động, có thể tương tác và hiển thị nội dung tốt, trên mọi trình duyệt hay các thiết bị là một tiêu chuẩn mới cũng có thể xem đó là điểm

cộng với một website chuẩn SEO. Ngày nay, người dùng có xu hướng sử dụng nhiều thiết bị khác nhau để tra cứu thông tin như PC, máy tính bảng hay điện thoại di động, do đó mà việc xây dựng một giao diện phù hợp và thích ứng được với nhiều thiết bị khác nhau là điều mà một website cần có.

7 Các yếu tố khác

Bên cạnh các yếu tố quan trọng đã được nêu thì còn có một số yếu tố cơ bản cần thiết khác khi xây dựng một website chuẩn SEO.

- Website có giao diện độc đáo, bố cục phù hợp với cấu trúc thuận tiện và logic, chuẩn UX/UI giúp người dùng có thể tìm đến bất kỳ nội dung nào trên website một cách thuận tiện.
- Tối ưu các thẻ H1 đến H6 trên website cùng với thuận tiện trong việc tối ưu thẻ mô tả (thuộc tính alt) của hình ảnh.
- Tối ưu hiển thị cho bài viết, hình ảnh, video và các nội dung khác trên website.
- Nội dung được tải về trực tiếp trong code HTML, không phải thông qua JavaScript, AJAX, iframe...
- Tích hợp thêm các nút chia sẻ mạng xã hội như nút Like Faceook, nút G+1 hay nút Tweet giúp việc chia sẻ nội dung website trên cộng đồng mạng trở nên nhanh chóng và thuận tiện hơn.
- Tối ưu cấu trúc website theo chuẩn W3C (Word Wide Web Consortium) đồng thời thân thiện với trải nghiệm người dùng.
- Đảm bảo tính bảo mật của website, hạn chế tối đa các hoạt động tấn công mạng hay virus tấn công website.
- Tạo trang 404 cho các trang không tìm thấy.

Chương IV

Thiết kế giải pháp

1 Thiết lập môi trường

Để ứng dụng được chạy ổn định và tránh ảnh hưởng đến những gói cài đặt mặc định, chúng tôi sẽ tạo một môi trường làm việc riêng cho ứng dụng hoạt động 1 cách độc lập so với hệ thống chung. Để tạo môi trường mới, sử dụng lệnh:

```
\>python -m venv ./venv
```

Tạo file `requirements.txt` để lưu lại các gói cần thiết để cài đặt:

```
Django==2.1.4
lxml==4.2.5
requests==2.21.0
gunicorn==19.9.0
django-heroku==0.3.1
```

Để kích hoạt sử dụng môi trường vừa tạo và cài đặt các gói cần thiết vào môi trường, sử dụng các lệnh sau:

```
\>venv\Scripts\activate
\>pip install -r requirements.txt
```

Như vậy, đã thiết lập xong các gói dùng để triển khai ứng dụng của chúng tôi, nếu cần thêm gói mới, chúng tôi sẽ cập nhật vào file `requirements.txt` và chạy lại lệnh cài đặt.

2 Kịch bản người dùng

Để tạo giao diện đa nền tảng, tối ưu cho các thiết bị di động, chúng tôi sử dụng Bootstrap để quản lý tính responsive cho trang web. Ngoài ra, chúng tôi còn sử dụng thêm vài icon nữa nên sẽ lựa chọn Font Awesome vào website.

2.1 Trang chủ

Chúng tôi hiển thị 1 form để người dùng nhập vào URL cần kiểm tra.

Để tránh kẻ xấu lợi dụng website chúng tôi để liên tục spam request đến các trang web khác. Chúng tôi sử dụng phương thức POST cho form và sử dụng thêm reCAPTCHA nhằm xác thực người dùng.

2.2 Trang kiểm tra

Trang này chúng tôi dùng để hiện thị thông tin được phân tích từ website của người dùng nhập vào. Kết quả sẽ được hiển thị bằng thẻ `<table>` gồm các cột:

- Thành phần: Hiển thị các mục mà chúng tôi xem xét trang web người dùng, như là tiêu đề, mô tả và các thẻ khác.
- Trạng thái: Sử dụng icon từ Font Awesome để cho người dùng biết tiêu chí đó có đạt yêu cầu hay không. Nếu đạt yêu cầu thì sẽ hiển thị icon có dấu tích xanh lá, ngược lại trang web hiển thị dấu x đỏ kèm theo giải thích và hướng dẫn để người dùng khắc phục lỗi đó.
- Chi tiết: Mục này chúng tôi dùng để liệt kê ra nội dung được lấy từ trang web của người dùng, ví dụ như nội dung của thẻ tiêu đề, mô tả, hình ảnh...

Ngoài ra, chúng tôi sử dụng đoạn JavaScript để tính toán điểm cho website dựa trên kết quả kiểm tra, màu sắc sẽ thay đổi theo từng thang điểm:

- $[80, 100]$: Màu xanh lá.
- $[50, 80)$: Màu vàng.
- $[0, 50)$: Màu đỏ.

Nút Trở lại được đặt ở cuối bảng cho phép người dùng quay lại trang chủ để có thể kiểm tra trang web khác.

2.3 Trang thông tin

Đây là nhóm các trang web chúng tôi dùng để hiển thị đến người dùng các trang như Liên hệ, Giới thiệu.

Những thủ thuật và tiêu chuẩn đánh giá của chúng tôi cũng sẽ được thêm vào nhóm này.

3 Xử lý Backend

Các file chúng tôi nhắc đến bên dưới nếu không có giải thích gì thêm sẽ mặc nhiên nằm trong thư mục `checkweb` mà chúng tôi đã tạo ở những phần trước.

Để có thể dễ dàng quản lý các phần lập trình khi thêm vào, chúng tôi tạo thêm file `functions.py`. Tạo file `urls.py` để khai báo và quản lý các URL cho ứng dụng.

Khi người dùng click vào nút submit trên trình duyệt, dữ liệu sẽ được truyền tới file `views.py` để xử lý, tại đây chúng tôi gọi đến hàm kiểm tra tính xác thực của CAPTCHA trong file `functions.py`.

views.py

```
from .functions import reCaptcha

class CheckView(TemplateView):
  def post(self, request):
    if reCaptcha(request):
      pass
      # code
```

functions.py

```
import requests
from django.conf import settings

def reCaptcha(request):
  response = request.POST["g-recaptcha-response"]
  data = {
    "secret": settings.GOOGLE_RECAPTCHA_SECRET_KEY,
    "response": response
  }
  verify = requests.post("https://www.google.com/recaptcha/api/siteverify",
     data=data)
  result = verify.json()
  return result["success"]
```

Nếu người dùng xác thực thành công thì sẽ trả về kết quả là `True` và tiếp tục thực thi bên trong câu lệnh `if`. Tiếp theo, chúng tôi gọi hàm `parsing()` với tham số `domain` là URL website người dùng nhập vào để phân tách dữ liệu.

views.py

```
if reCaptcha(request):
  if(parsing(domain)):
    pass
    # code
```

functions.py

```
from lxml import html

def parsing(domain):
  try:
    page = requests.get(domain)
    content = html.fromstring(page.content.decode("utf-8"))
  except Exception:
    return False

  value = dict()
  elm = {
       "title": "//title/text()",
  }

  for k,v in elm.items():
    try:
      value[k] = content.xpath(v)[0]
    except Exception:
      print(k, "not found!")

  return value
```

Đoạn code sẽ lấy ra tiêu đề của trang web và trả về kết quả vào biến `value`. Chúng tôi quay lại file `views.py` để lấy giá trị này và hiển thị ra giao diện người dùng.

`views.py`

```
if(parsing(domain)):
  context = parsing(domain)
  return render(request, "checkweb/check.html", context)
```

Do đang sử dụng template do Django cung cấp nên file `check.html` sẽ được lưu tại `lvtn\templates\checkweb\check.html`. Chúng ta có thể gọi giá trị ra giao diện bằng biến có tên là key tương ứng của biến `context`.

`check.html`

```
{% extends "base.html" %}

{% block title %}Checking page{% endblock %}
{% block content %}
<!-- get variable -->
<h2>{{ title }}</h2>
{% endblock %}
```

Tới đây, chúng tôi đã cung cấp những cái nhìn cơ bản cách ứng dụng của chúng tôi hoạt động. Chúng tôi sẽ tiếp tục hoàn thiện báo cáo này trong những lần cập nhật sau.

Chương V

Hiện thực

Phần này chúng tôi sẽ trình bày quá trình chúng tôi xây dựng và phát triển để tạo thành sản phẩm hoàn chỉnh. Dựa trên nền tảng lý thuyết được giới thiệu ở phần trước, chúng tôi sẽ áp dụng chúng vào những hướng dẫn bên dưới. Ở phần cuối của chương này sẽ là hướng dẫn về cách triển khai ứng dụng lên Internet bằng việc sử dụng máy chủ do trang Heroku cung cấp.

Lưu ý, trong bài báo cáo này chúng tôi đang sử dụng môi trường lập trình là Windows nên có thể có những câu lệnh sẽ khác với MacOS, Linux hay những môi trường lập trình khác.

1 Khởi tạo project

1.1 Tạo thư mục và thiết lập môi trường

Chúng tôi quyết định đặt tên dự án của mình là `check-seo`, do đó chúng tôi sẽ tạo thư mục mới để lưu trữ code.

Để tạo mới thư mục, bạn có thể click chuột phải → chọn New → chọn Folder → đặt tên `check-seo`.

Ở đây, chúng tôi sẽ tạo thư mục bằng command line trong Windows PowerShell. Để mở PowerShell tại thư mục hiện tại, bấm giữ phím `Shift` → click chuột phải chọn Open PowerShell window here → nhập lệnh sau để tạo thư mục mới:

```
\>mkdir check-seo
```

Di chuyển khung làm việc vào thư mục project vừa tạo.

```
\>cd check-seo
```

Tại đây, chúng tôi sẽ tiến hành cài đặt môi trường để lập trình cho ứng dụng Python của mình. Đảm bảo là bạn đã cài đặt xong Python theo hướng dẫn ở phần ***Nền tảng lý***

thuyết, để tránh dài dòng, chúng tôi sẽ hạn chế nhắc lại những kiến thức đã được trình bày ở phần trước.

Tại khung cửa sổ của PowerShell, nhập lệnh sau để cài đặt môi trường:

```
\>python -m venv ./venv
```

Thư mục mới được tạo ra có tên là `venv` chứa các tập tin hệ thống giúp tạo môi trường ảo cho Python. Để kích hoạt môi trường ảo, sử dụng câu lệnh:

```
\>.\venv\Scripts\activate
```

Khi kích hoạt môi trường thành công, sẽ có phần thông tin `(venv)` hiển thị ở đầu mỗi dòng lệnh, giống như `(venv)\>`

1.2 Cài đặt các thư viện cần thiết

Sau khi tạo xong thư mục và kích hoạt xong môi trường ảo, tiếp theo chúng tôi sẽ tiến hành cài đặt các thư viện phục vụ cho dự án của chúng tôi.

Trong thư mục `check-seo`, tạo tệp mới có tên là `requirements.txt` sẽ chứa thông tin về thư viện và phiên bản sử dụng, để xem thông tin cụ thể của từng thư viện, bạn có thể tìm kiếm chúng trên kho lưu trữ công khai của Python là `https://pypi.org`. Nội dung của file `requirements.txt` như sau:

```
Django==2.2
lxml==4.3.3
requests==2.21.0
```

Để tiến hành cài đặt thư viện được liệt kê trong file `requirements.txt`, sử dụng câu lệnh:

```
(venv)\>pip install -r requirements.txt
```

Trình cài đặt thư viện Python sẽ tiến hành tải về và cài đặt trong môi trường ảo mà chúng tôi đã kích hoạt. Ngoài thư viện chính, trình cài đặt còn tải thêm những thư viện khác bổ trợ đi theo từng thư viện. Để kiểm tra các gói đã cài đặt, sử dụng lệnh:

```
(venv)\>pip freeze list
```

Kết quả trả về có thể như sau:

```
certifi==2019.3.9
chardet==3.0.4
Django==2.2
idna==2.8
lxml==4.3.3
pytz==2019.1
requests==2.21.0
```

```
sqlparse==0.3.0
urllib3==1.24.3
```

1.3 Tạo project và xây dựng app Django

Sau khi cài đặt xong những thư viện cần thiết, tiếp theo, chúng ta sẽ tiến hành tạo mới project Django có tên là `src` trong thư mục `check-seo` bằng câu lệnh:

```
(venv)\>django-admin startproject src .
```

Sau khi thực thi thành công câu lệnh trên thì sẽ tạo ra thư mục `src` chứa các file cài đặt cho project và file `manage.py` giúp quản lý các thao tác command line cho ứng dụng.

Theo thiết kế của ứng dụng, chúng tôi sẽ tạo thêm 2 app cho project là `checkweb` quản lý chính cho việc thu thập, đánh giá SEO cho website và `tips` sẽ đảm nhiệm hiển thị các bài đăng về thủ thuật SEO. Để tạo app, sử dụng lần lượt 2 câu lệnh sau đây:

```
(venv)\>python .\manage.py startapp checkweb
(venv)\>python .\manage.py startapp tips
```

Để quản lý các file giao diện `html`, chúng tôi tạo thêm thư mục `templates` tại thư mục chính của project. Tiếp theo, chúng tôi đi vào thư mục `src` sau đó tạo thêm thư mục có tên là `static_venv` đảm nhiệm việc lưu trữ các file CSS, JavaScript và các thư viện bên ngoài như Bootstrap, jQuery.

Sau khi tạo mới app, thư mục `templates` và `static_venv`, cần phải đăng ký vào cấu hình để project hiểu được cấu trúc của chương trình tại file `settings.py` trong thư mục `src`.

Để khai báo app, tìm đến dòng `INSTALLED_APPS` và thêm đoạn code bên dưới vào hàng cuối cùng, kết quả sẽ tương tự như:

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",

    "checkweb.apps.CheckwebConfig",
    "tips.apps.TipsConfig",
]
```

Cấu hình templates cho project tại khóa `DIRS` của mục `TEMPLATES`:

```
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
```

```
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]
```

Cuối cùng trong phần này, chúng tôi sẽ cấu hình phần `static` để hiển thị các file CSS, JavaScript,... tại mục `STATIC_URL`, chúng tôi sẽ thêm đoạn code vào để được kết quả như dưới đây:

```
STATIC_URL = "/static/"
STATIC_ROOT = os.path.join(BASE_DIR, "static")
STATICFILES_DIRS = [os.path.join(BASE_DIR, "src/static_venv")]
```

2 Cấu trúc giao diện Templates

2.1 Xử lý Frontend

Phần này, chúng tôi sẽ trình bày về cách chúng tôi phân chia các file giao diện `html` trong thư mục `templates` được tạo ở hướng dẫn bên trên.

Đầu tiên, chúng tôi tạo file `base.html` có chức năng là khung sườn cho toàn bộ giao diện với khả năng kết nạp các file khác để giúp chia nhỏ giao diện thành các phần có chức năng riêng biệt. Việc chia nhỏ giao diện thành các file thành phần giúp chúng tôi có thể quản lý code tốt hơn và tránh rối rắm khi nếu lưu quá nhiều dòng code trong một file duy nhất.

Tiếp theo, chúng tôi tạo thêm hai file nữa có tên là `header.html` và `footer.html`. Quay lại file `base.html`, ta có cấu trúc code đơn giản như sau:

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <!-- Header -->
    {% include "header.html" %}
    <main>
        {% block content %}{% endblock %}
    </main>
    <!-- Footer -->
```

```
    {% include "footer.html" %}
    {% block script %}{% endblock %}
</body>
</html>
```

Để có thể thay đổi nội dung theo từng giao diện, chúng tôi đã sử dụng ba block là `title`, `content` và `script`, do đó chúng tôi sẽ thay đổi nội dung ở hai block này tùy theo mục đích mà chúng tôi muốn hướng đến.

Ở file `header.html` và tương tự là file `footer.html` sẽ có nội dung cơ bản như sau:

```
<header>
    <nav>
        <a href="/"><h1>Danh Gia Web</h1></a>
    </nav>
</header>
```

```
<footer>
    <div>DGW &copy; 2018 - {% now "Y" %}</div>
</footer>
```

Sau khi cấu trúc xong bộ khung cho giao diện, tiếp theo chúng tôi sẽ xây dựng giao diện cho từng app dựa trên những gì đã thiết lập.

Tại thư mục `templates` tạo thêm hai thư mục mới có tên trùng với hai app đã tạo là `checkweb` và `tips`. Chúng tôi sẽ đi sâu vào việc tạo giao diện cụ thể cho phần app `checkweb` vì tại đây là trọng tâm chính của ứng dụng đánh giá website của chúng tôi. Phần giao diện `tips` có phần đơn giản hơn nhiều và bạn có thể thiết lập dựa theo hướng dẫn ở phần `checkweb`. Hơn nữa, chúng tôi sẽ cung cấp mã nguồn ở phần ***Kết luận***, do đó bạn có thể tự nghiên cứu dựa theo những đoạn code của chúng tôi.

Mở thư mục `checkweb` vừa tạo, dựa theo kiến trúc ở phần ***Thiết kế giải pháp***, chúng tôi tiến hành tạo thêm các file mới là `index.html`, `about.html`, `contact.html` và `check.html`.

`index.html`

```
{% extends "base.html" %}
{% block title %}Trang Chu{% endblock %}
{% block content %}
<div>Noi dung Trang chu</div>
{% endblock %}
```

Các file còn lại cũng có cấu trúc tương tự, với nội dung ở các block sẽ khác nhau tùy theo mỗi file. Ở đây chúng ta quan tâm đến hai file đó là `index.html` và `check.html` sẽ được nhắc đến ở những phần sau.

2.2 Xử lý Backend

Sau khi tạo xong các file `html`, tiếp theo chúng tôi sẽ tiến hành cấu hình để xử lý phần backend của ứng dụng.

Đầu tiên, chúng tôi sẽ quản lý các url để hiển thị file giao diện trong ứng dụng tại file `urls.py` trong thư mục `src`. File `urls.py` sẽ có nội dung như sau:

```
from django.urls import path, include

urlpatterns = [
    path("", include("checkweb.urls")),
    path("thu-thuat/", include("tips.urls")),
]
```

Tiếp theo, chúng tôi thực hiện việc kết nối giữa truy vấn url và giao diện tại file `views.py` trong thư mục app `checkweb` được tạo ra khi chạy lệnh `startapp` lúc đầu. Django hỗ trợ việc kết nối này đơn giản và tiết kiệm dòng code hơn nhiều bằng chế độ ***Class-based views***. Chúng tôi sẽ sử dụng để tạo giao diện cho trang chủ, giới thiệu, liên hệ và trang kiểm tra.

```
from django.views.generic import TemplateView

class IndexView(TemplateView):
    template_name = "checkweb/index.html"

class AboutView(TemplateView):
    template_name = "checkweb/about.html"

class ContactView(TemplateView):
    template_name = "checkweb/contact.html"

class CheckView(TemplateView):
    template_name = "checkweb/check.html"
```

Trong thư mục `checkweb` hiện tại, tạo file `urls.py` để quản lý các url trong ứng dụng được gọi từ hàm `include` ở file `urls.py` trong thư mục `src`.

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.IndexView.as_view(), name="index"),
    path("gioi-thieu/", views.AboutView.as_view(), name="about"),
    path("lien-he/", views.ContactView.as_view(), name="contact"),
    path("kiem-tra/", views.CheckView.as_view(), name="check"),
]
```

Với cách tương tự, chúng tôi sẽ tiến hành cấu hình đối với app `tips`. Chi tiết, bạn có thể xem trên mã nguồn của chúng tôi.

3 Hiện thực chức năng đánh giá website

Sau khi thiết đặt xong phần giao diện templates, tiếp theo chúng tôi sẽ đi sâu vào xây dựng các hàm thực hiện chức năng chính cho project của mình. Đó là công việc nhận vào url trang web mà người dùng muốn đánh giá, kiểm tra captcha và url, sau đó trả về kết quả cho người dùng.

Theo cách thông thường, các hàm này được viết trong file `views.py` ở thư mục app `checkweb`. Tuy nhiên, để dễ dàng quản lý hơn, chúng tôi sẽ tạo một file mới cùng trong thư mục app và đặt tên là `functions.py` sẽ chứa những đoạn code phục vụ cho việc phân tích và kiểm tra cho ứng dụng của chúng tôi. Do việc tách ra như vậy, tại file `views.py` chúng tôi sẽ nhúng file này bằng dòng code sau được thêm vào ở đầu file:

```
from . import functions
```

3.1 Tạo form nhập và xử lý xác thực reCaptcha

Tạo giao diện nhập url

Trước tiên, chúng tôi sẽ tạo giao diện form nhập để người dùng gửi url muốn kiểm tra vào ứng dụng của chúng tôi. Do đó, chúng tôi sẽ tiến hành thêm thẻ `form` vào file `index.html` trong thư mục `templates/checkweb`.

```
<form action="{% url 'check' %}" method="POST">
    {% csrf_token %}
    <input type="url" name="url" id="url" required>
    <button id="submit">Submit</button>
</form>
```

Form của chúng tôi sẽ sử dụng giao thức `POST`, được xử lý trong backend và xuất về đường dẫn được khai báo trong `urls.py` có tên là `check`, cụ thể ở đây sẽ là file `check.html`.

Sau khi đã tạo xong thẻ `form`, tiếp đến chúng tôi truy cập vào trang `https://www.google.com/recaptcha/admin/` để lấy thông tin về khóa API của reCaptcha. Khóa này được dùng để kích hoạt tính năng của reCaptcha. Theo hướng dẫn trên trang admin, chúng tôi sẽ chèn tiếp thẻ `div` bên trong thẻ `form` và bên dưới thẻ `button`.

```
<div class="g-recaptcha" data-sitekey="<SITE_KEY>" data-callback="onSubmit"
    data-badge="bottomleft" data-size="invisible"></div>
```

Ngoài ra, để reCaptcha có thể hoạt động được, chúng tôi sẽ chèn thêm đoạn code để xử lý JavaScript. Đoạn code này chúng tôi sẽ đặt trong khối `script` được kế thừa từ khung `base.html` mà chúng tôi đã khai báo trước đó.

Đoạn JavaScript bên dưới có nhiệm vụ bắt sự kiện click vào nút Summit và tạo ra đoạn mã xác thực reCaptcha mà chúng tôi sẽ xử lý tiếp ở phần tiếp theo:

```
{% block script %}
<script src="https://www.google.com/recaptcha/api.js" async defer></script>
<script>
var sm = document.getElementById("submit");
sm.onclick = validate;

function validate(event) {
    event.preventDefault();
    grecaptcha.execute()
};

function onSubmit(token) {
    sm.onclick = null;
    sm.click();
}
</script>
{% endblock %}
```

Hàm xác thực reCaptcha

Chúng tôi sẽ viết hàm này trong file `functions.py` có tên là `reCaptcha`.

- Đầu vào sẽ là mã được sinh ra từ việc xác thực của reCaptcha được lưu với tên là `g-recaptcha-response` từ truy vấn `POST` và IP của người dùng truy cập.
- Đầu ra là kết quả sau từ đánh giá của reCaptcha sau khi gửi các giá trị đầu vào lên máy chủ. Sẽ có hai giá trị là `true` và `false` tương ứng với kết quả xác thực người dùng là thành công hay thất bại.

Khi đăng ký reCaptcha, chúng ta còn nhận được ngoài khóa `SITE_KEY` thì còn một khóa nữa là `SECRET_KEY`. Để có thể dễ dàng kiểm soát các khóa trong ứng dụng, chúng tôi sẽ lưu khóa bí mật này trong file `settings.py` ở thư mục `src`.

```
# Google reCAPTCHA secret key
# https://developers.google.com/recaptcha/docs/verify/

GOOGLE_RECAPTCHA_SECRET_KEY = "<SECRET_KEY>"
```

Và để sử dụng biến `GOOGLE_RECAPTCHA_SECRET_KEY` thì Django có hỗ trợ bằng cách thêm dòng code sau vào đầu file `functions.py`:

```
from django.conf import settings
```

Bên cạnh đó, để gửi dữ liệu lên máy chủ của reCaptcha, chúng tôi cần thêm sự hỗ trợ của thư viện Python là `requests` mà chúng tôi đã cài đặt từ lúc mới thiết lập môi trường của ứng dụng.

```
import requests
```

Sau khi thêm các thư viện và ý tưởng xử lý thì hàm xử lý reCaptcha sẽ có nội dung như sau:

```
data = {
    "secret": settings.GOOGLE_RECAPTCHA_SECRET_KEY,
    "response": response,
    "remoteip": userIP,
}
verify = requests.post(
    "https://www.google.com/recaptcha/api/siteverify", data=data)
result = verify.json()
return result["success"]
```

Hàm `reCaptcha` được xây dựng xong thì tiếp theo, chúng tôi sẽ quay lại file `views.py` để gọi lại hàm này và xử lý nó.

Việc xử lý captcha nằm trong class `CheckView` mà chúng tôi đã tạo trước đó. Theo quy ước trong Django thì để xử lý truy vấn `POST`, chúng tôi viết thêm hàm `post` trong class `CheckView`.

```
def post(self, request):
    url = request.POST["url"]
    if reCaptcha(request.POST["g-recaptcha-response"],
        request.META["REMOTE_ADDR"]):
        # code check web here
        return render(request, "checkweb/check.html")
    return redirect("/")
```

Nếu kiểm tra captcha thành công thì ứng dụng sẽ trả về trang giao diện trong file `check.html`. Nếu xác thực thất bại thì sẽ chuyển hướng trở lại trang chủ. Chúng tôi có sử dụng hàm chuyển hướng trang `redirect`. Thêm vào đầu file `views.py` đoạn code sau để chèn hàm này:

```
from django.shortcuts import redirect
```

Đến đây, chúng tôi đã giải quyết xong vấn đề kiểm tra xác thực người dủng bằng reCaptcha.

3.2 Phân tích cấu trúc nội dung website

Lấy source code và phân tách trang web

Theo cách hoạt động ứng dụng của chúng tôi sẽ phân tích trang web bằng cách lấy source code tương đương với tính năng View source trên các trình duyệt web. Để xử lý chức năng này, chúng tôi xây dựng hàm `parsing` trong file `functions.py`.

- Đầu vào sẽ là url của trang web cần kiểm tra
- Đầu ra là biến có kiểu `dict()` chứa nội dung các yếu tố mà chúng tôi phân tách ra

được từ source code.

Ngoài thư viện `requests`, chúng tôi cần sự trợ giúp thêm của thư viện `lxml`.

```
from lxml import html
```

Sau khi thêm vào thư viện cần thiết, bên dưới là đoạn code dùng để lấy source code dựa trên url trang web.

```
def parsing(url):
    try:
        page = requests.get(url, timeout=5)
        content = html.fromstring(page.content.decode("utf-8"))
    except BaseException:
        return False
```

Vì để tránh chương trình bị lỗi khi người dùng nhập vào url mà chúng tôi không thể kiểm tra được nên chúng tôi đã sử dụng cấu trúc `try/except` trong Python để giải quyết vấn đề này. Ngoài ra, nếu truy vấn vượt quá năm giây cũng sẽ bị xem là lỗi không kiểm tra được. Cuối cùng, biến `content` được decode theo chuẩn mã `utf-8` để hiển thị font chữ không bị lỗi.

Tiếp theo, cũng trong hàm `parsing`, chúng tôi khai báo biến `value` có kiểu là `dict()` để lưu các yếu tố được phân tách. Bên cạnh đó là các biến dùng để phục vụ làm input cho các hàm về sau.

```
value = dict()
domain = url.split("/")[2]
urlDomain = url.split("/")[0] + "//" + domain
```

Việc phân tách cấu trúc trong source code, chúng tôi sử dụng phương pháp `xpath`. Để dễ dàng quản lý, chúng tôi chia ra hai loại là thành phần chỉ có một giá trị, ví dụ thẻ `title` sẽ được lưu cấu trúc `xpath` trong biến `elm`, những thành phần có hơn một giá trị, ví dụ thẻ `img` sẽ được lưu cấu trúc `xpath` trong biến `elms`. Cả hai biến `elm` và `elms` đều cò kiểu là `dict()`.

```
elm = {
    "title": "//title/text()",
    "description": "//meta[@name='description']/@content",
    "favicon": "//link[contains(@rel, 'icon')]/@href",
    "robotsMeta": "//meta[@name='robots']/@content",
}
elms = {
    "h1Tags": "//h1//text()",
    "h2Tags": "//h2//text()",
    "aTags": "//a/@href",
    "cssInlines": "//@style/..",
    "imgTags": "//img",
}
```

Sau khi đã có cấu trúc `xpath` của từng thuộc tính, tiếp theo chúng tôi sẽ sử dụng vòng lặp `for` để lấy dữ liệu dựa trên cấu trúc có sẵn. Do có hai cấu trúc khác khau, nên chúng tôi cũng sẽ dùng hai vòng lặp để xử lý.

```
for k, v in elm.items():
    try:
        value[k] = content.xpath(v)[0]
    except BaseException:
        value[k] = None

for k, v in elms.items():
    try:
        value[k] = content.xpath(v)
    except BaseException:
        value[k] = None
```

Sau khi chạy xong hai vòng lặp này thì biến `value` đã lưu được khá nhiều dữ liệu từ việc phân tách source code dựa trên `xpath`. Tuy nhiên, có một số yếu tố sẽ không đúng chuẩn do những website khác nhau dẫn đến có nhiều kết quả không như ý. chúng tôi sẽ giải quyết vấn đề này bằng các hàm bổ sung và được chúng tôi đề cập trong những phần tiếp theo.

Cuối cùng, chúng tôi trả về kết quả của biến `value` và chuyển qua file `views.py` để xử lý dữ liệu trả về.

```
return value
```

Tiếp theo, chúng tôi sẽ xử lý dữ liệu của `value` ở file `views.py`. Đoạn code bên dưới bao gồm luôn phần xử lý xác thực reCaptcha được chúng tôi trình bày ở phần trước.

```
def post(self, request):
    url = request.POST["url"]
    if reCaptcha(request.POST["g-recaptcha-response"],
        request.META["REMOTE_ADDR"]):
        context = parsing(url)
        if context:
            context["url"] = url
            return render(request, "checkweb/check.html", context)
        return redirect("/")
    return redirect("/")
```

Biến `context` sẽ nhận kết quả trả về từ hàm **parsing**. Nếu không có giá trị, nghĩa là việc kiểm tra url gặp lỗi, do đó chúng tôi thay vì trả về truy vấn đến trang giao diện `check.html`, mà sẽ chuyển hướng trở về trang chủ.

3.3 Xử lý nâng cao các cấu trúc trong website

Favicon

Vấn đề xảy ra ở đây là do định dạng cấu trúc đường dẫn đến hình ảnh của các website khác nhau có phần khác biệt, và do `xpath` chỉ lấy nội dung bên trong thẻ nên chúng tôi cần phải định dạng lại cấu trúc đường dẫn này. Ví dụ, liên kết hình ảnh bắt đầu bằng '`/`', để hiển thị đúng thì cần phải gắn thêm phần tên miền vào trước liên kết. Sau đây là hàm xử lý trong file `functions.py`

```
def getLinkImg(elm, urlDomain):
    if elm and elm[:2] not in {"ht", "//"}:
        elm = urlDomain + "/" + elm.lstrip("/")
    return elm
```

Chúng ta sẽ gán lại khóa này trong biến `value`:

```
value["favicon"] = getLinkImg(value["favicon"], urlDomain)
```

Thẻ H1, H2

Trong quá trình phát triển, chúng tôi nhận thấy nội dung của các thẻ này thường xuyên bị thừa các ký tự khoảng trắng ở đầu hoặc cuối dòng. Đôi khi có vài thẻ có nội dung rỗng. Nguyên nhân là vì bên trong các thẻ này còn được lồng thêm các thẻ khác như thẻ `a`, `span`. Do đó chúng tôi cần viết thêm hàm để dọn sạch các khoảng trống bị dư thừa, cũng như các thẻ không chứa nội dung.

```
def cleanElms(elms):
    if elms:
        for idx, _ in enumerate(elms):
            elms[idx] = elms[idx].strip()
        elms = list(filter(None, elms))
    return elms
```

Sau đó, chúng tôi gán lại dữ liệu cho các thẻ này trong hàm `parsing`:

```
value["h1Tags"] = cleanElms(value["h1Tags"])
value["h2Tags"] = cleanElms(value["h2Tags"])
```

Robots.txt

Mặc định, nếu trang web sử dụng file này thì sẽ có cấu trúc đường dẫn là `https://ten-mien/robots.txt`. Do đó, chúng tôi sẽ sử dụng thư viện `requests` để truy vấn đến liên kết này và kiểm tra xem mã trạng thái trả về có phải là 200 không, cũng như có kiểu nội dung là `plain`.

```
def getlinkRobots(urlDomain):
```

```
    try:
        value = requests.get(urlDomain + "/robots.txt")
    except BaseException:
        return None
    if value.status_code != 200 or "plain" not in
        value.headers["Content-Type"]:
        return None
    return urlDomain + "/robots.txt"
```

Đây là yếu tố mới không có trong biến `value` khi xử lý `xpath`, do đó chúng tôi gán giá trị hàm này vào khóa mới:

```
value["robotsTxt"] = getlinkRobots(urlDomain)
```

Sitemap.xml

Thông thường, cũng tương tự như với `robots.txt`, đường dẫn của `sitemap.xml` sẽ là `https://ten-mien/sitemap.xml`. Tuy nhiên, ở một số các trang web, liên kết `sitemap.xml` không giống như mặc định và đường dẫn đó được khai báo trong file `robots.txt` có nội dung tương tự như:

```
Sitemap: https://ten-mien/site-map/sitemap.xml
```

Do đó, để kiểm tra đường dẫn `sitemap.xml` của trang web, chúng tôi sẽ xem xét nội dung trong file `robots.txt` trước, nếu không tìm thấy, chúng tôi sẽ tiếp tục kiểm tra với liên kết mặc định. Đối với mỗi liên kết có được, chúng tôi sẽ kiểm tra mã trạng thái trả về có là 200 hay không, cũng như kiểu nội dung là `xml`.

```
def getlinkSitemap(urlDomain, robots):
    try:
        value = requests.get(urlDomain + "/sitemap.xml")
    except BaseException:
        return None
    if robots:
        txt = requests.get(robots).content.decode("utf-8")
        txt = txt.replace("\n", "")
        sitemap = re.findall(r"Sitemap:.*xml", txt)
        if sitemap:
            sitemap = sitemap[0].split("Sitemap: ")[1:]
            return sitemap
    if value.status_code != 200 or "xml" not in value.headers["Content-Type"]:
        return None
    sitemap = [urlDomain + "/sitemap.xml"]
    return sitemap
```

Cũng tương tự yếu tố `robots.txt`, chúng tôi tiếp tục thêm khóa mới vào biến `value`:

```
value["sitemaps"] = getlinkSitemap(urlDomain, value["robotsTxt"])
```

Lỗi liên kết

Đầu tiên, chúng tôi viết hàm dùng để định dạng lại thẻ liên kết lấy được từ `xpath`. Sau đó lọc ra các liên kết trùng nhau, liên kết `tel`, `mailto` và `javascript` cũng được chúng tôi loại ra trước khi kiểm tra liên kết đó có lỗi hay không.

```
def getlinkA(elms, urlDomain):
    idx = 0
    while idx < len(elms):
        elm = elms[idx]
        if elm in ("#", "/") or elm.split(":")[0] in ("tel", "mailto",
            "javascript"):
            elms.pop(idx)
            idx -= 1
        elif elm[:2] not in ("ht", "//"):
            elms[idx] = urlDomain + "/" + elm.lstrip("/")
        elif elm[:2] == "//":
            elms[idx] = "https:" + elm
        idx += 1
    return set(elms)
```

Tiếp theo, chúng tôi viết hàm để truy vấn đến liên kết bằng thư viện `requests` và để hạn chế thời gian phản hồi, chúng tôi đặt thêm tham số `timeout=5` để bỏ qua liên kết nếu nó phản hồi sau quá năm giây.

```
def checkBrokenLink(elm):
    try:
        value = requests.get(elm, timeout=5)
    except BaseException:
        return elm
    if value.status_code != 200:
        return elm
    return None
```

Hàm ở trên chỉ đang xét ở từng liên kết đơn lẻ. Tuy nhiên trong một website có thể có rất nhiều liên kết dẫn đến nếu xét tuần tự từng liên kết thì dẫn đến thời gian chờ sẽ rất lâu. Do đó, chúng tôi đã áp dụng xử lý đồng thời vào quá trình kiểm tra này. Để sử dụng gói hỗ trợ này trong Python, chèn dòng code sau vào đầu file `functions.py`:

```
from concurrent.futures import ThreadPoolExecutor
```

Sau khi chèn xong gói thư viện hỗ trợ, chúng tôi triển khai hàm đồng thời và trả về mảng các liên kết bị lỗi:

```
def getBrokenLink(elms):
    res = list()
    with ThreadPoolExecutor() as executor:
        for elm in executor.map(checkBrokenLink, elms):
            if elm:
                res.append(elm)
    return res
```

Cuối cùng, chúng tôi thêm các giá trị này vào khóa mới trong biến `value`

```
value["aTags"] = getlinkA(value["aTags"], urlDomain)
value["aBrokens"] = getBrokenLink(value["aTags"])
```

CSS nội tuyến

Chúng tôi cần định dạng lại kết quả trả về để nó ngắn gọn hơn nhưng vẫn đảm bảo đầy đủ nội dung mà chúng tôi muốn hiển thị. Bởi vì `xpath` sẽ trả về kết quả là toàn bộ nội dung của thẻ có CSS nội tuyến, chúng tôi sẽ viết hàm để chỉ trả về phần mở thẻ, nơi chứ nội dung của thuộc tính `style` và loại bỏ phần nội dung cũng như phần code đóng thẻ.

```
def getCSSInlines(elms):
    for idx, value in enumerate(elms):
        tmp = html.tostring(value, encoding="utf-8").decode("utf-8")
        elms[idx] = re.search(r"<.*?>", tmp).group()
    return elms
```

Sau đó, chúng tôi gán lại nội dung lại vào biến `value`:

```
value["cssInlines"] = getCSSInlines(value["cssInlines"])
```

Thuộc tính alt

Việc xử lý yếu tố này cũng gần tương đương với cách làm trên. Chúng tôi sẽ kiểm tra các thẻ `img` xem nếu bị thiếu thành phần thuộc tính `alt` thì sẽ trả về thẻ `img` đó.

```
def checkMissAlts(elms):
    res = list()
    for elm in elms:
        tmp = html.tostring(elm, encoding="utf-8").decode("utf-8")
        if not re.search(r"alt=(\'|\").+(\'|\")", tmp):
            res.append(re.search(r"<img.*?>", tmp).group())
    return res
```

Xếp hạng

Chúng tôi sử dụng dữ liệu về xếp hạng trang web bằng cách lấy API từng kho dữ liệu mở của `Open PageRank`. Sau khi đăng ký tài khoản thành công, bạn sẽ nhận được khóa API dùng để truy cập miễn phí lên kho dữ liệu này. Cũng tương tự như khóa khi đăng ký reCaptcha, chúng tôi sẽ lưu khóa này trong file `settings.py` của ứng dụng:

```
# Open PageRank key
# https://www.domcop.com/openpagerank/
```

```
OPEN_PAGERANK_KEY = "<API_KEY>"
```

Sau đó, chúng tôi quay trở lại file `functions.py` để hoàn thiện hàm để lấy thông tin xếp hạng.

```
def getPageRank(domain):
    url = "https://openpagerank.com/api/v1.0/getPageRank?domains[0]=" + domain
    headers = {
        "API-OPR": settings.OPEN_PAGERANK_KEY
    }
    data = requests.get(url, headers=headers)
    result = data.json()["response"][0]
    return result["rank"]
```

Cuối cùng là gán giá trị trả về này vào biến `value`:

```
value["pageRank"] = getPageRank(domain)
```

Đến đây chúng tôi đã hoàn thành xong giai đoạn lấy nội dung source code từ trang web người dùng muốn kiểm tra, phân tách các nội dung cần đánh giá bằng `xpath` và định dạng lại những nội dung này bằng các hàm nâng cao bổ sung.

3.4 Hiển thị kết quả đánh giá ra giao diện

Đây là phần giao diện được xử lý tại file `check.html` mà chúng tôi đã tạo ở những phần trước. Lần này, chúng tôi sẽ đi sâu vào chi tiết ở file này.

Những yếu tố đánh giá chúng tôi sẽ sử dụng thẻ `table` để hiển thị thông tin. Nội dung này được đặt trong block `content`:

```
{% block content %}
<table id="tbCheck">
    <thead>
        <tr>
            <th scope="col" width="15%">Tieu chi</th>
            <th scope="col" width="15%">Ket qua</th>
            <th scope="col" width="70%">Chi tiet</th>
        </tr>
    </thead>
    <tbody>
        <!-- code here -->
    </tbody>
</table>
{% endblock %}
```

Bảng này chúng tôi sẽ chia thành ba cột lần lượt là:

- Tiêu chí: các yếu tố chúng tôi đánh giá trong website của bạn.

- Kết quả: hiển thị icon tương ứng từ `Font Awesome` với ba trạng thái là thành công, thất bại và thông tin.
- Chi tiết: sẽ là nội dung được lấy từ biến `value` mà chúng tôi xử lý ở phần trước, cùng với phần chú thích của chúng tôi để bạn có thể biết được rõ hơn tiêu chí đánh giá của chúng tôi về nội dung này.

Tiếp theo chúng tôi sẽ hiển thị thông tin đánh giá cho từng mục trong phần thân của thẻ `tbody`. Ví dụ, với mục tiêu đề, đoạn code của chúng tôi sẽ có nội dung như sau:

```
<tbody>
    <tr>
        <th scope="row">Tieu de</th>
        <td>
            {% if title and title|length < 65 %}
            <i class="fas fa-check-circle text-success"></i>
            {% else %}
            <i class="fas fa-times-circle text-danger"></i>
            {% endif %}
            <input type="hidden" class="point" value="5">
        </td>
        <td>
            {% if title %}
            <div>Do dai tieu de trang cua ban la <b>{{ title|length }}</b> ky
                tu. Hau hat cac cong cu tim kiem se cat bot tieu de trang thanh
                65 ky tu.</div>
            <small><i class="fas fa-angle-double-right"></i> {{ title }}</small>
            {% else %}
            <div>Khong tim thay tieu de tren trang cua ban.</div>
            <small><i class="fas fa-angle-double-right"></i><em>
                None</em></small>
            {% endif %}
        </td>
    </tr>
</tbody>
```

Đối với yếu tố đánh giá mà kết quả trả về là mảng danh sách ví dụ như là thẻ `h1` thì chúng tôi sẽ xử lý như sau:

```
<tbody>
    <!-- <tr>...</tr> -->
    <tr>
        <th scope="row">The h1</th>
        <td>
            {% if h1Tags %}
            <i class="fas fa-check-circle text-success"></i>
            {% else %}
            <i class="fas fa-times-circle text-danger"></i>
            {% endif %}
            <input type="hidden" class="point" value="5">
        </td>
        <td>
            {% if h1Tags %}
```

```
            <div>Tim thay <b>{{ h1Tags|length }}</b> the h1 tren trang cua
                ban.</div>
            <small>{% for h1Tag in h1Tags %}<i class="fas
                fa-angle-double-right"></i> {{ h1Tag }}<br>{% endfor %}</small>
            {% else %}
            <div>Khong tim thay the h1 tren trang cua ban.</div>
            <small><i class="fas fa-angle-double-right"></i><em>
                None</em></small>
            {% endif %}
        </td>
    </tr>
</tbody>
```

Chúng tôi đã giới thiệu hai hình thức đánh giá cơ bản để hiển thị ra giao diện cho người dùng. Những tiêu chí khác tương tự, bạn có thể tham khảo tại trang mã nguồn mở của chúng tôi.

Tiếp đến, chúng tôi sẽ trình bày về cách chúng tôi chấm điểm trang web của bạn dựa trên những kết quả này. Chúng tôi sử dụng thư viện jQuery để đọc các class hiển thị icon ở cột Kết quả. Với class `fa-check-circle` xuất hiện ở cột Kết quả có nghĩa là yếu tố đó đã vượt qua bài kiểm tra của chúng tôi. Dựa vào vị trí đó, chúng tôi tìm tiếp đến thẻ `input` nơi chứa trọng số để đánh giá tiêu chí đó được thể hiện ở thuộc tính `value`.

Đoạn code này được chúng tôi đặt trong block `script` để có thể kế thừa giao diện từ bố cục chung trong file `base.html`:

```
{% block script %}
<script>
$(document).ready(function() {
    var total = 0;
    $("#tbCheck .point").each(function() {
        total += parseInt($(this)[0].value)
    });
    var score = 0;
    $("#tbCheck .fa-check-circle").each(function() {
        score += parseInt($(this).parent().children().last()[0].value)
    });
    score = Math.round(score / total * 100);
})
</script>
{% endblock %}
```

Sau cùng, tạo thẻ `div` có `id="score"` để chúng tôi hiển thị kết quả điểm đánh giá sau khi tính toán xong. Để trực quan hơn, chúng tôi sẽ hiển thị thêm màu tương ứng với điểm nhận được.

- Xanh lá nếu điểm lớn hơn hoặc bằng 80/100.
- Vàng nếu điểm lớn hơn hoặc bằng 50/100.
- Đỏ sẽ trong trường hợp còn lại.

```
<script>
$("#score").text("Diem: " + score);
if (score >= 80) {
    $("#score").addClass("btn-success")
} else if (score >= 50) {
    $("#score").addClass("btn-warning")
} else {
    $("#score").addClass("btn-danger")
}
</script>
```

Đến đây, chúng tôi đã trình bày xong tất cả các quá trình mà chúng tôi từ lúc khởi tạo, cài đặt và xây dựng nên ứng dụng của chúng tôi. Tuy nhiên, ứng dụng vẫn chỉ đang nằm ở `localhost`, dẫn đến người dùng chưa thể truy cập vào ứng dụng trên môi trường Internet. Để triển khai ứng dụng của mình, chúng tôi sẽ trình bày ở phần tiếp ngay sau.

3.5 Triển khai ứng dụng lên Heroku

Sau khi tạo tài khoản và cài đặt xong Heroku CLI, chúng tôi sẽ tiến hành các cấu hình để triển khai ứng dụng của mình.

Đầu tiên, chúng tôi cấu hình file `Procfile` để cài đặt nền tảng sử dụng, được đặt tại thư mục gốc `check-seo`.

```
web: gunicorn src.wsgi
```

Để chỉ định phiên bản hỗ trợ của Python, chúng tôi tạo thêm file `runtime.txt` có nội dung là:

```
python-3.7.3
```

Tiếp theo chúng tôi cài đặt thêm thư viện `gunicorn` là môi trường web server cho Django được hướng dẫn bởi Heroku.

```
(venv)\>pip install gunicorn
```

Heroku cũng khuyến khích cài thêm thư viện Python do họ phát triển dành riêng cho ứng dụng Django để giúp ứng dụng hoạt động tốt trên nền tảng của họ.

```
(venv)\>pip install django_heroku
```

Sau khi cài đặt, chúng tôi cần chèn thư viện này vào file `settings.py` trong thư mục `src`:

```
# Configure Django App for Heroku.
# https://devcenter.heroku.com/articles/django-app-configuration

import django_heroku
```

```
django_heroku.settings(locals())
```

Chúng tôi cũng không quên thêm thông tin của hai gói cài đặt này vào cuối file `requirements.txt`:

```
gunicorn==19.9.0
django-heroku==0.3.1
```

Đến đây, chúng tôi đã hoàn tất cấu hình để triển khai ứng dụng lên Heroku. Việc cần làm tiếp theo là gửi code này lên máy chủ của Heroku với tên app của chúng tôi trên tài khoản là `checkseo`.

Tạo `git` trong thư mục gốc:

```
(venv)\>git init
(venv)\>heroku git:remote -a checkseo
```

Xác nhận và tải code lên Heroku:

```
(venv)\>git add .
(venv)\>git commit -m "upload"
(venv)\>git push heroku master
```

Đợi khoảng vài giây để Heroku khởi động và xây dựng ứng dụng từ mã nguồn từ bạn gửi lên. Ứng dụng của chúng tôi sẽ có đường dẫn mặc định là `https://checkseo.herokuapp.com`.

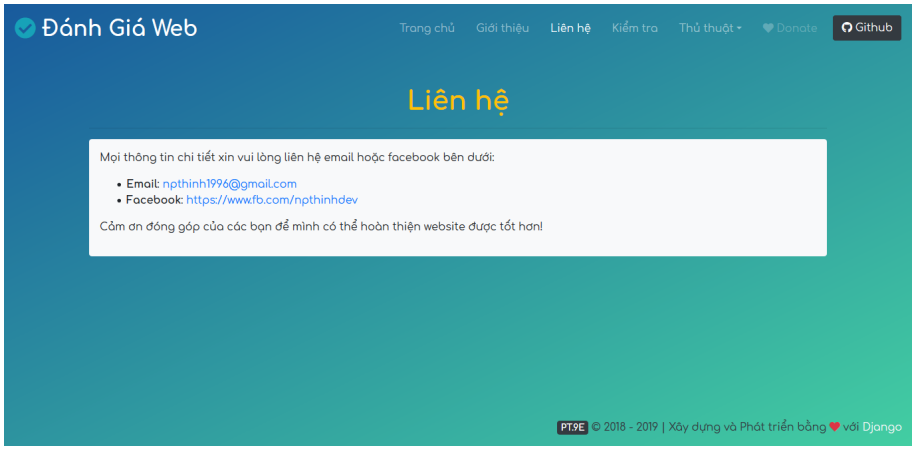
4 Sản phẩm hoàn chỉnh

Sau khi thực hiện xong các bước trong những hướng dẫn bên trên, chúng tôi đã triển khai thành công ứng dụng của mình. Tuy nhiên, để tránh sự phức tạp trong bài báo cáo này, chúng tôi đã lượt bỏ phần cấu hình các thẻ class cho thư viện Bootstrap, bạn có thể xem chi tiết chúng trong mã nguồn được chia sẻ của chúng tôi.

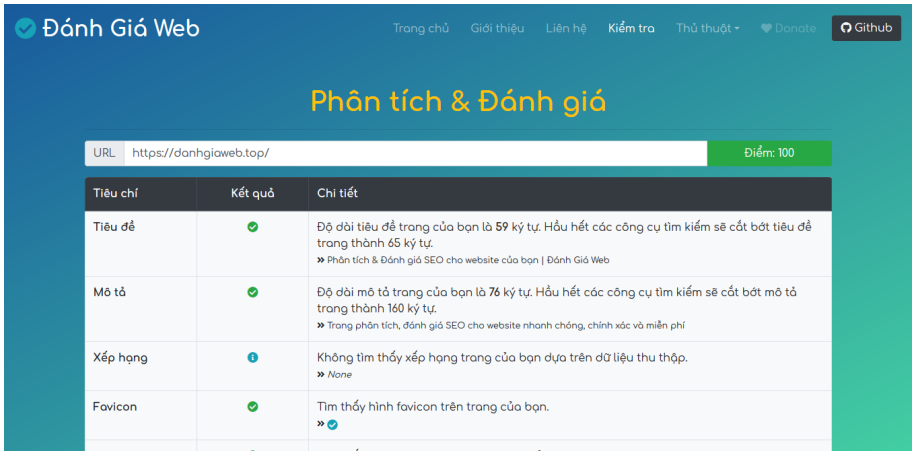
Bên dưới, chúng tôi sẽ đưa ra cho bạn thấy được khi ứng dụng của chúng tôi hoàn chỉnh về giao diện sẽ được hiển thị như thế nào.

Hình V.1: Trang chủ của ứng dụng

Hình V.2: Trang giới thiệu của ứng dụng

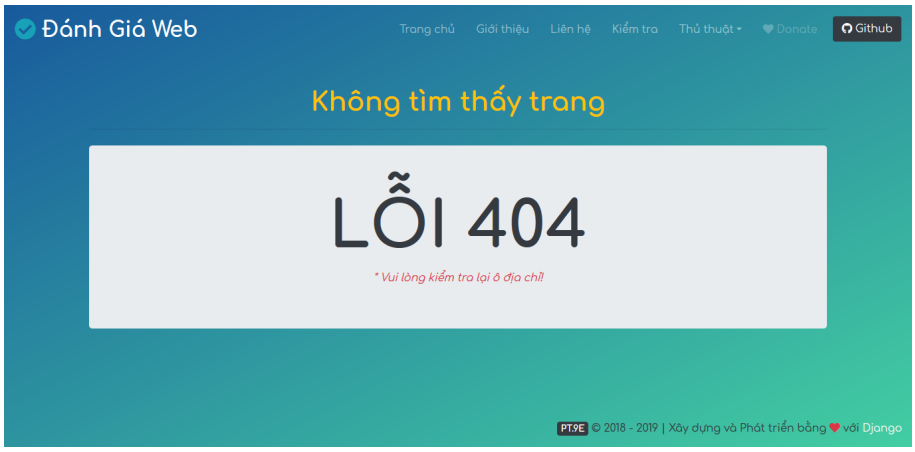


Hình V.3: Trang liên hệ của ứng dụng



Hình V.4: Trang phân tích của ứng dụng

Hình V.5: Trang thủ thuật của ứng dụng



Hình V.6: Trang lỗi 404 của ứng dụng

Chương VI

Kết luận đánh giá

Trong giải pháp được nêu ở phần trước, chúng tôi chưa áp dụng lưu trữ vào cơ sở dữ liệu, tính năng này chúng tôi sẽ cập nhật vào các phiên bản tiếp theo. Ngoài ra, những phần kiến thức nền tảng chúng tôi sẽ sớm bổ sung thêm chi tiết đi kèm với nhiều ví dụ hơn.

Hiện tại, ứng dụng của chúng tôi đã được triển khai với phiên bản thử nghiệm, người dùng có thể truy cập và thử dùng vài tính năng mà chúng tôi đã phát triển.

Website: `https://danhgiaweb.top`

Chúng tôi nhận các ý kiến phản hồi đóng góp qua email: npthinh1996@gmail.com

Tài liệu tham khảo

[1] Python tutorial. https://www.w3schools.com/python/. (truy cập 26/05/2019).

[2] Django documentation. https://docs.djangoproject.com/en/2.2. (truy cập 25/12/2018).

[3] Requests 2.21.0. http://docs.python-requests.org/en/master/user/quickstart. (truy cập 25/12/2018).

[4] lxml.html. https://lxml.de/lxmlhtml.html. (truy cập 25/12/2018).

[5] Introduction. https://getbootstrap.com/docs/4.1. (truy cập 25/12/2018).

[6] Icons. https://fontawesome.com/icons. (truy cập 26/12/2018).

[7] recaptcha v2. https://developers.google.com/recaptcha/docs/display. (truy cập 25/12/2018).

[8] Django. https://devcenter.heroku.com/categories/working-with-django. (truy cập 26/12/2018).

[9] Những tiêu chí cần thiết cho website chuẩn seo. https://seoconghuong.com/kien-thuc-seo/562-nhung-tieu-chi-can-thiet-cho-website-chuan-seo.html. (truy cập 26/12/2018).