Software Design - CSC 312

Status Report 1

SEO Analyzer

Group 4

Delario Nance, Jr., Sky Luo, Donald Lin, Alp Niksarli

March 21st, 2024

1 Introduction

1.1 Highlights

• What was the plan for this iteration?
- Development
    - Build a minimal Flask app whose frontend lets users input a URL which the backend receives
    - Write a Python scraper which grabs HTML contents from a URL's corresponding web page
    - Write a Python broken link detector which detects if a URL corresponds to a non-existent web page
    - Write a meta tag analyzer which checks if common keywords in a web page's content match the meta tag keywords
- Research
    - Research how to score a website's SEO performance
- Customers
    - Schedule and meet with Liz Brigham to discuss customer needs

• Highlight what the team accomplished.
- Development
    - We built a minimal Flask app whose home page lets users input a URL which the backend receives and sends back to the frontend to display on the home page
    - We wrote a Beautiful Soup scraper which grabs each link and meta tag from a URL's corresponding HTML
    - We wrote a Python program which, given a URL, returns True if requests to the URL lead to a 200 status code (success)
    - We wrote a Python program which, given a URL, returns common keywords in the web page's content which also appear as meta tag keywords
- Research
    - We learned that Google does not take meta tags into consideration when ranking search results

1.2 Changes

• Summarize any major changes since the proposal.
- We decided that we need to look at more features than just meta tags and broken links when giving SEO advice to the user

• Include each change's date, motivation, description, and implications.
- Change: Analyze more features than just meta tags and broken links when creating SEO advice
    - Date: March 18, 2024
    - Motivation: Google ignores meta tags when ranking searches and most websites seem to not have broken links, so we need to find features of webpages which have a stronger influence on SEO
    - Description: Previously, we decided to give users SEO advice by displaying content-relevant meta tags which they are not using and links on the webpages

which do not lead to existing webpages. Now, we have decided to look for more features with a more prominent impact on a webpage's Google search ranking.
    - Implications: In order to ensure that we do not waste large amounts of time by writing code for features with small impact on Google search rankings, we will likely need to define in a more stable set of features to use
- If there were none, note that there were no changes.

2 Customer-Iteration Goals

- Describe the product backlog of this iteration.
    - Implement code which offers SEO advice to the user, given their web page URL
    - Implement code which gives an SEO score to the user, given their web page URL
    - Implement a dashboard which displays SEO advice to the user
    - Create database of user profiles
    - Set up AWS server
- Describe the customer's desired overall experience.
    - Our proposed customer's experience is that they register an account for their given website or webpage. Then, after logging in, they can request personalized SEO feedback in such a way that using the tool does not require advanced SEO knowledge and the tool does not leak their web page's private information to other companies.
- Describe the sprint backlog of this iteration.
    - Development
        - Build a minimal Flask app whose frontend lets users input a URL which the backend receives
        - Write a Python scraper which grabs HTML contents from a URL's corresponding web page
        - Write a Python broken link detector which detects if a URL corresponds to a non-existent web page
        - Write a meta tag analyzer which checks if common keywords in a web page's content match the meta tag keywords
    - Research
        - Research how to score a website's SEO performance
    - Customers
        - Schedule and meet with Liz Brigham (Hurt Hub) to discuss customer needs
- Explain why you have selected the work items in the sprint backlog for this sprint (or iteration).
    - Development
        - We planned to build the minimal Flask app so that we have an interactive UI for testing code and later extending into a dashboard which displays SEO advice in an organized format
        - We planned to write the Python scraper so that we can extract SEO-related data such as meta tags and broken links from HTML code

    - We planned to write a Python broken link detector so that the user can remove broken links from their website, which is commonly believed to improve a web page's SEO
- Research
    - We planned to research how to score a website's SEO performance so that obtain an objective, numerical metric which our software can use to rate an input website's SEO
- Customers
    - We planned to schedule and meet with Liz Brigham so that we can find a customer and gain their feedback on the project idea, which we can use to improve or sharpen our project requirements

2.1 Use Cases

• Write a use case for each main user goal for a primary or secondary customer.
• Show each use case's title, user goal, and full basic flow. Choose meaningful titles.
• Each use case should have at least one specific alternative flow and one bounded alternative flow.

Name: Obtain Advice for Improving Website's SEO	
Goal: A startup employee requests personalized advice for improving their company website's SEO	
Actors: Startup Employee, SEO analyzer software	
Basic Flow: 1. **{ Visit login page}** 2. The user visits the login page 3. The user inputs information to create an account 4. The system registers the user for an account 5. The system redirects the user to the login page 6. The user inputs their account information 7. The system verifies that the account has been registered **{ Verify account existence }** 8. The system logs the user into their account 9. The user inputs their company website's home page 10. The user presses a "Get SEO	Alternative Flow(s): *Specific Alternative Flow:* At **{ Verify account existence }**, if the system determines that the account has not been registered, The system displays "Account does not exist." Resume the basic flow at **{ Return to login page }** *Bounded Alternative Flow:* At any point between **{ Visit login page }** and **{ Log user out }**, if the Internet connection is lost Display "The network connection has been lost. Please log in again after reconnecting to the Intenet." Resume the basic flow at **{ Log user out }**

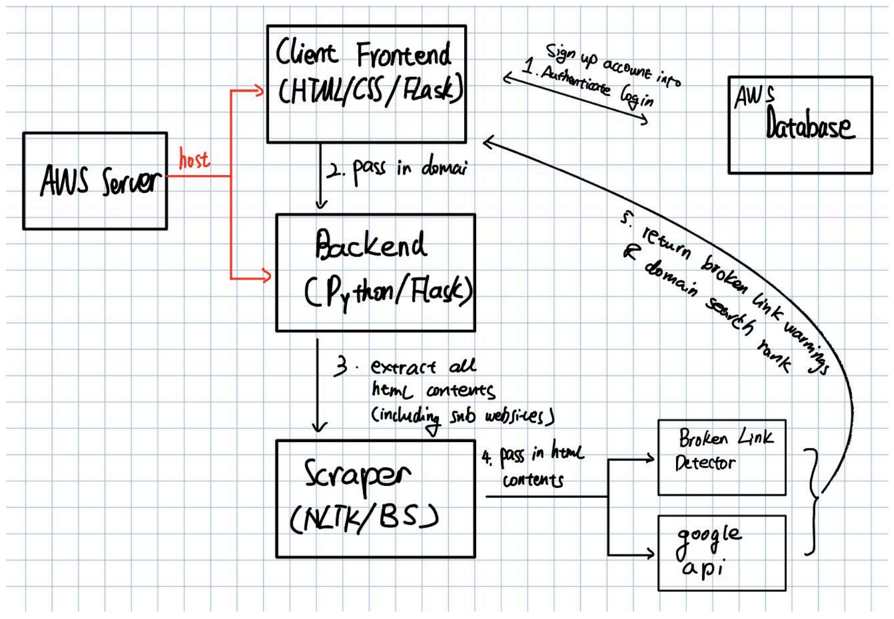
Advice” button 11. The system generates personalized advice for improving the company website’s SEO 12. The system displays the generated personalized advice 13. The user requests to log out of their account 14. **{ Log user out }** 15. The system logs the user out of their account **{ Return to login page }** 16. The system redirects the user to the login page	

Name: Obtain Score of Website’s SEO	
Goal: A startup employee requests a score of their company website’s SEO	
Actors: Startup Employee, SEO analyzer software	
Basic Flow: 1. **{ Visit login page}** 2. The user visits the login page 3. The user inputs information to create an account 4. The system registers the user for an account 5. The system redirects the user to the login page 6. The user inputs their account information 7. The system verifies that the account has been registered **{ Verify account existence }** 8. The system logs the user into their account 9. The user inputs their company website’s home page 10. The user presses a “Get SEO Scores” button 11. The system generates a score for	Alternative Flow(s): *Specific Alternative Flow:* At **{ Verify account existence }**, if the system determines that the account has not been registered, The system displays “Account does not exist.” Resume the basic flow at **{ Return to login page }** *Bounded Alternative Flow:* At any point between **{ Visit login page }** and **{ Log user out }**, if the Internet connection is lost Display “The network connection has been lost. Please log in again after reconnecting to the Intenet.” Resume the basic flow at **{ Log user out }**

the company website's SEO 12. The system displays the generated score 13. The user requests to log out of their account 14. **{ Log user out }** 15. The system logs the user out of their account **{ Return to login page }** 16. The system redirects the user to the login page	

3 System Description

The diagram and descriptions should be more precise and detailed.
• Draw a block diagram to show how the proposed system will interact with external services, databases, etc. Clearly mark the boundaries of the system.



• Use the above diagram to introduce the system.

- Our system, as of so far developed, will be hosted by an AWS EB server and will use an AWS RDS database. There are two main modules, the frontend HTML/CSS/Flask codes and the backend Python/Flask codes that are divided into the following functions: scraper, broken link detector, and Google API interactor. The steps of using the system are: user signup/login, domain name input, scraping html contents, passing html contents to the broken link detector and the Google API interactor, and eventually returning the broken link warnings & domain name search rank to the client.

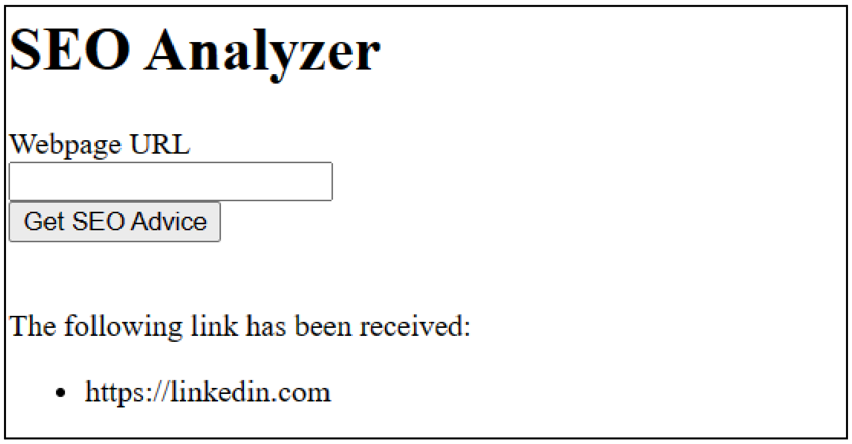
• What are the main elements of the proposed system?
- Frontend
- Backend
- Scraper
- Broken Link Detector
- Google API Interactor
- AWS RDS Database
- AWS EB Server

4 Current Status

Summarize the current implementation status of your system.
- Our system contains a minimal Flask app whose home page lets users input a URL and display it back to the user
- Our system contains a scraper which grabs links and meta tags from a given web page
- Our system contains have a broken link detector which checks if a URL corresponds to a valid web page
- Our system contains a meta tag analyzer which returns common keywords in a given web page's content which also appear as that page's meta tag keywords

4.1 Screenshots

Add the screenshot(s) of the system's working part(s). Explain the screenshot, i.e., explain (1) what you are trying to show in the screenshot, (2) which part in the system description diagram the part belongs to, (3) what is the behavior of the system, and (4) why this system is important.

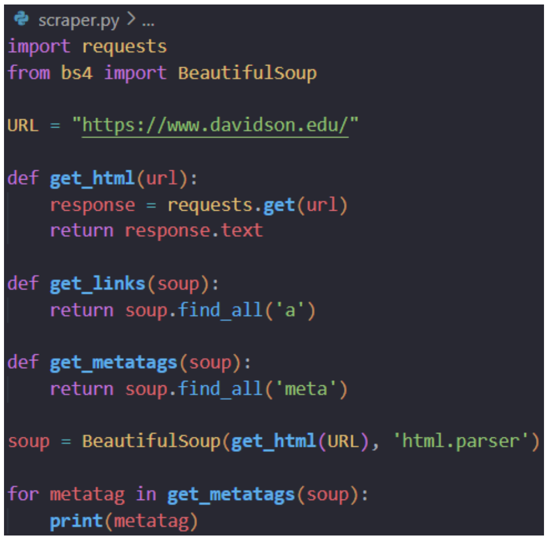
- 
SEO Analyzer

Webpage URL

The following link has been received:

• https://linkedin.com
- Description: Build a minimal Flask app which lets users input a URL that the backend will receive
- Motivation: To obtain a UI that we can use to test code and later extend into a dashboard
- Results: We developed a Flask app whose home page lets users input a URL and see it

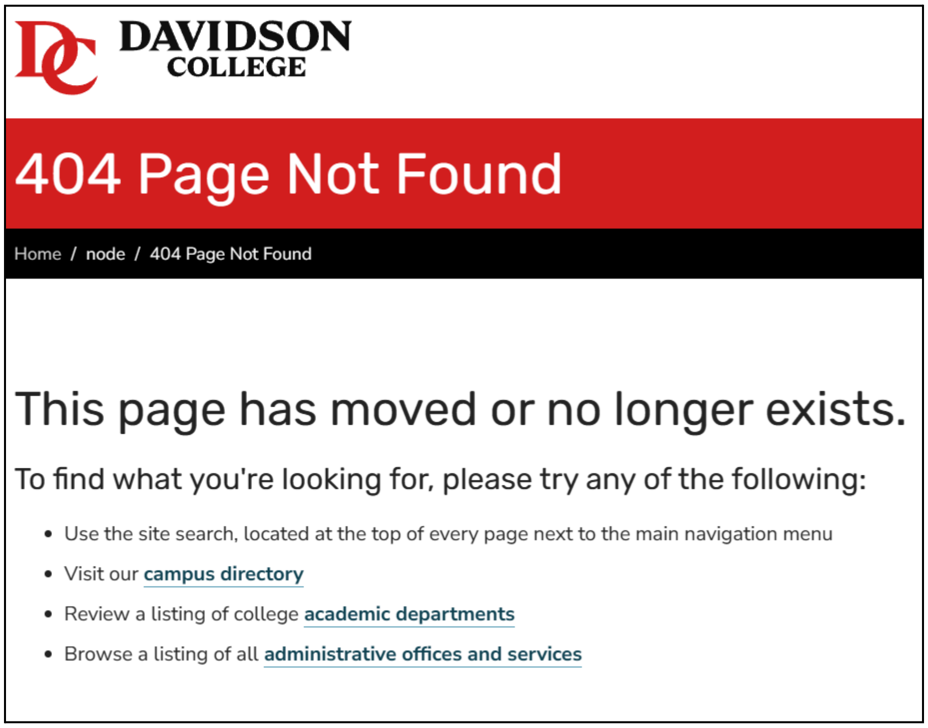
displayed back on the page

- 

scraper.py > ...

```python
import requests
from bs4 import BeautifulSoup

URL = "https://www.davidson.edu/"

def get_html(url):
    response = requests.get(url)
    return response.text

def get_links(soup):
    return soup.find_all('a')

def get_metatags(soup):
    return soup.find_all('meta')

soup = BeautifulSoup(get_html(URL), 'html.parser')

for metatag in get_metatags(soup):
    print(metatag)
```

- Description: Write a Python scraper which grabs HTML contents from a URL
- Motivation: For obtaining SEO-related data such as meta tags, we will parse HTML files
- Results: We wrote a Beautiful Soup scraper which grabs each link and meta tag from a URL's corresponding HTML

- 

DAVIDSON COLLEGE

404 Page Not Found

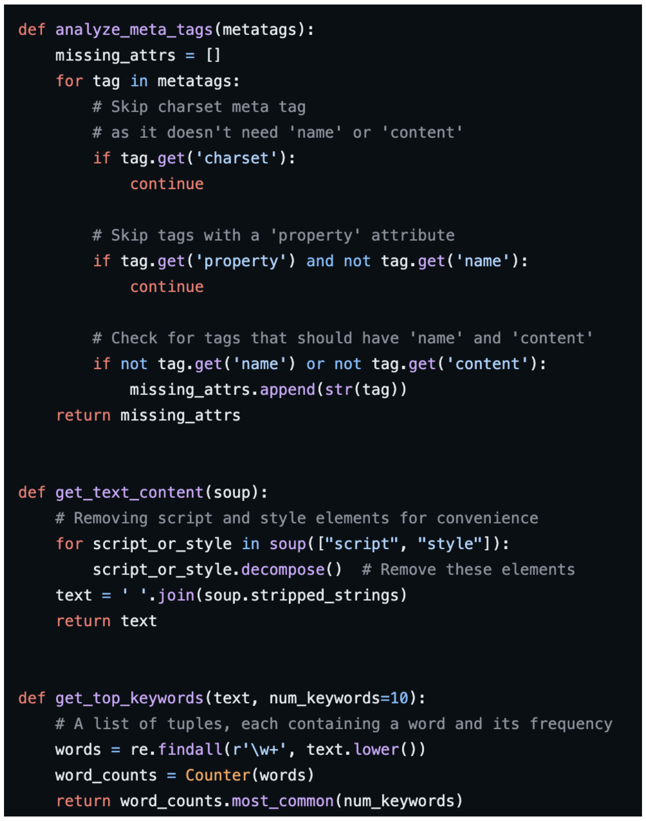
Home / node / 404 Page Not Found

This page has moved or no longer exists.

To find what you're looking for, please try any of the following:

- Use the site search, located at the top of every page next to the main navigation menu
- Visit our campus directory
- Review a listing of college academic departments
- Browse a listing of all administrative offices and services

- Description: Write a Python program which detects if a URL corresponds to a non-existent web page
- Motivation: There is a popular belief that broken links hurt a web page's ranking in search results
- Results: We wrote a Python program which, given a URL, returns True if requests to the URL lead to a 200 status code (success)

- 

```python
def analyze_meta_tags(metatags):
    missing_attrs = []
    for tag in metatags:
        # Skip charset meta tag
        # as it doesn't need 'name' or 'content'
        if tag.get('charset'):
            continue

        # Skip tags with a 'property' attribute
        if tag.get('property') and not tag.get('name'):
            continue

        # Check for tags that should have 'name' and 'content'
        if not tag.get('name') or not tag.get('content'):
            missing_attrs.append(str(tag))
    return missing_attrs


def get_text_content(soup):
    # Removing script and style elements for convenience
    for script_or_style in soup(["script", "style"]):
        script_or_style.decompose()  # Remove these elements
    text = ' '.join(soup.stripped_strings)
    return text


def get_top_keywords(text, num_keywords=10):
    # A list of tuples, each containing a word and its frequency
    words = re.findall(r'\w+', text.lower())
    word_counts = Counter(words)
    return word_counts.most_common(num_keywords)
```

- Description: Write a program which checks if common keywords in a web page’s content match the meta tag keywords
- Motivation: There is a popular belief that meta tags of common keywords helps a web page’s ranking in search results
- Results: We wrote a Python program which, given a URL, returns common keywords in the page which appear as meta tag keywords

4.2 Tests

• List the tests you performed for this iteration’s implemented parts.

- Broken Link Detector: used several real-life working and broken links test the codes
- Meta Tag Analysis: used several Davidson’s websites (Davidson’s home page, hurthub etc.)
- Frontend Interface: used Davidson’s home page to test if Flask successfully receives and displays the input link

• From the acceptance tests in the proposal, explain your plan to test them technically. In other words, how would you automatically test your acceptance tests? What are the inputs to the tests and outputs from the tests?

- As the product has not been finalized yet, most acceptance tests from the proposal do not have a passing implementation currently.
- In order to test these in an automatic manner, we can use Unit Testing through an automation library like Selenium in Python.
- As inputs our testing functions would receive domain names and keywords to search for while in the output we would receive the data that can be extracted from our dashboard which would most likely have certain suggestions about keywords or tags about the site.

How would you determine whether the test was passed or failed?

- We would create certain expected outcomes for each input we would be testing. As the unit testing software runs, it would check if the outcome of the product matches the expected outcomes defined by our developer team.

5 Project Management

Continue to maintain the Change Log. Add any new changes to the project, tracking the date and description of each change.

Date	Description
March. 10th	In contact with the HurtHub, in process of scheduling meeting
March. 12th	Implemented basic Python scraper function to retrieve HTML content from provided URLs
March. 12th	Developed basic front-end HTML form for URL input
March. 14th	Meta tag analyzer development completed, including keyword analysis feature
March. 15th	Added broken links detection functionality in the Python codebase

6 Review and Retrospective

• What went well?
- The implementation of the work items has been successful. We were also able to get in contact with Liz from the HurtHub, who consented to the utilization of our tool on the HurtHub website and has offered to facilitate introductions to other small businesses that could benefit from our services.

• What didn't go well?
- We were unable to meet with Liz. This has delayed our understanding of the customer-specific requirements.

• For the goals that were not met, what were the issues?
- The main issue was schedule conflicts, but we foresee that we will be able to find a suitable meeting time and talk with Liz very soon.

• How do you plan to overcome the issues?
- We have prioritized the meeting with Liz and our customer for the next sprint. We have already initiated contact and will be using more flexible scheduling tools to ensure that we can align our availability.

• What do you plan to do differently in the next iteration?

- For the next iteration, we aim for clearer and more direct communication by using a shared calendar and routine updates to keep everyone informed. We'll aim to anticipate and plan for potential obstacles earlier, and set aside extra time for any unplanned work. Additionally, each team member will sharpen their focus on their roles.

7 Team Management

• What were the team roles for this iteration?
- Developers: all team members
- Scrum Master: Sky
- Product Owner: Alp

• What did each team member contribute?
- **Alp Niksarli** A Python scraper of HTML contents (input is an HTML link)
- **Delario Nance** A front-end (HTML) interface that prompts the user for HTML link input
- **Sky Luo** A Python Broken links detection function (input is HTML link)
- **Donald Lin** Meta tag analyzer (find missing attributes; doing keyword analysis of the web content and see if it matches the meta tag keyword content)

• What were the challenges regarding team management, e.g., regular meeting, etc.?
- Role confusion: although we defined the roles for each member, we ended up sharing the work of different roles.

• What are the plans to overcome the challenges?
- Strictly follow the defined roles in the next sprint.

• If you were the third party who knows very well about your team, what suggestions would you give to your team?
- Put into more time and effort connecting and communicating with clients.

8 Goals for the Next Iteration

• Write the next iteration's product log.
- Creating a simple draft dashboard for our users where they will be able to see the results of multiple functionalities such as meta tag analyzer or broken link detector
- Starting out with analyzing the textual content of the scraped pages with the help of Natural Language Processing.
- Writing recursive scrapers in order to analyze in-depth content of each website

• Write the next iteration's sprint log.
- Database
    - Set up a database in AWS RDS & write SQL Codes for the Database & Flask codes to connect the Database
- Server
    - Function: analysis of HTML visible text - return the analyzed keywords from HTML. Compare with the user's target keywords.
    - Function (for broken link detector) that recursively scrape links, visit links, scrape links… Stop when the domain ≠ user input domain.

    - Structure the data on the backend so that it can be displayed on the dashboard
- Frontend
    - Front-end forms for register & login
    - Design dashboard
    - Implement dashboard
    - Improve styles of frontend for the demo
    - Find websites to demo on in presentation
- Customers
    - Schedule meeting with Liz (3 pm Friday)
    - Meet with Liz Brigham (3 pm Friday)
- Research
    - Research Google Search Console
    - Research Page Rank

• Other than the issues discussed in Section 6, i.e., Review and Retrospective, what potential challenges do you see in the next iteration?

- Connecting with our customers who are small business owner in the Hurt Hub and don't already use third party services or the SEO plugins of these services

• Briefly explain how your team would overcome each of the mentioned challenges.

- Implementing more features such as sitemap creation or advanced broken link detector in order to benefit any type of website even the non-coded ones by the businesses.