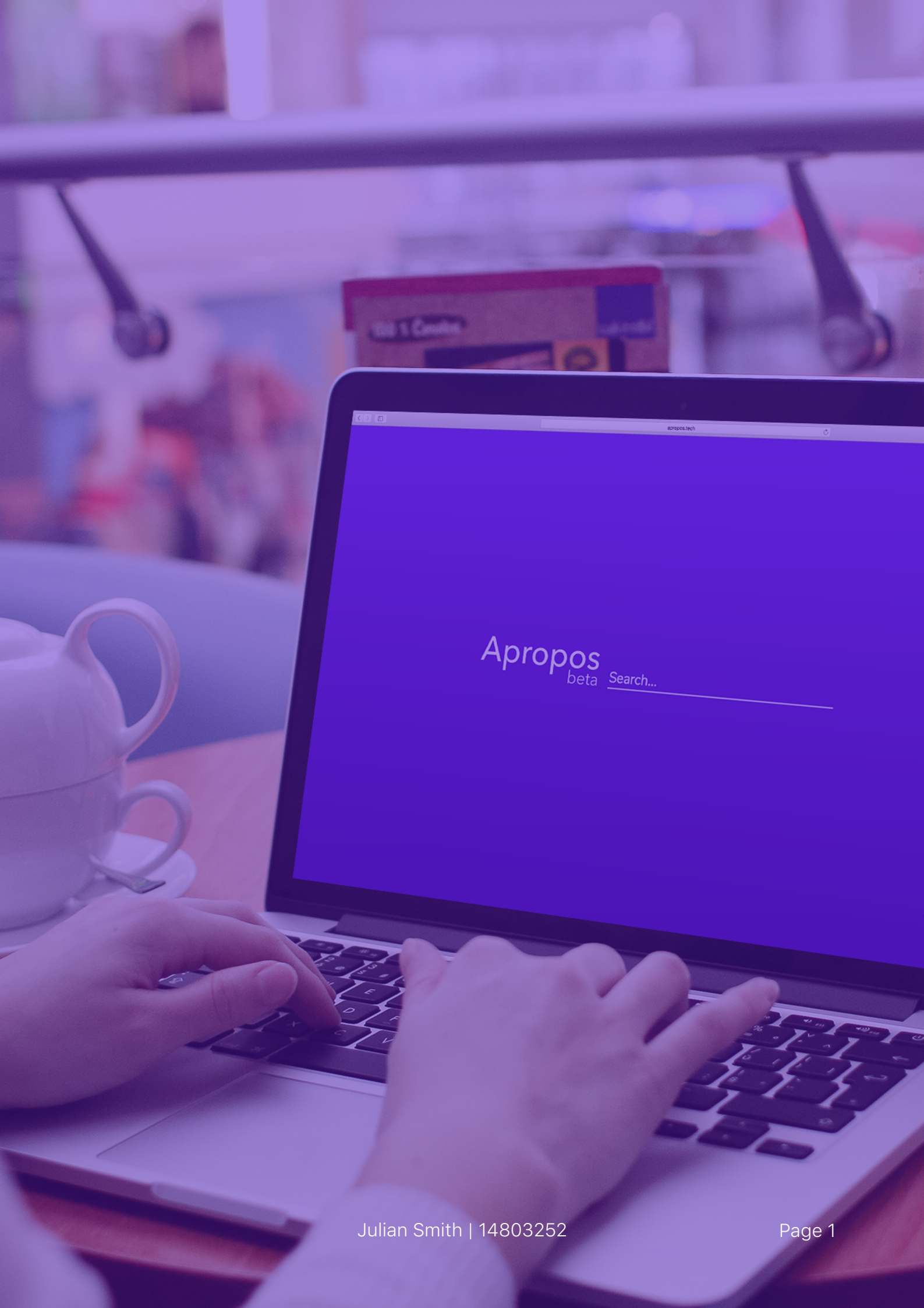
Report To The Examiners

"A Definition First Search Engine"

Julian Richard Smith
BSc Hons Computer Science (Games)
jrs38@uni.brighton.ac.uk
14803252

Contents


apropos.tech
Apropos
beta
Search...

Project Scope

Aims and Objectives

The aim and objective of the project is to produce a web based search engine geared towards programmers, that will display search results generated previously from a web indexer. Put simply, a search engine for programmers. What makes this search engine unique, however, is that it will display a definition, if appropriate, of the searched item as well as an example piece of code, if appropriate, before the search results.

Project Background

My initial interest in producing this search engine first came from studying the "Intelligent systems" module in my second year, whereby we were tasked as part of the coursework to create a simple web crawler in the Python programming language. I very much so enjoyed producing this piece of coursework, predominately due to taking a keen interest in the gathering of data. As such, I inevitably wanted to produce a project based around data gathering having enjoyed doing so in this module. Furthermore, I excelled in my "Web Development " and "Human Computer Interaction" modules in my first year, so I also wanted to produce an internet based and a consumer focused project. After a large range of research prior to the start of the project, I decided to produce the web indexer and crawler in python despite not having a vast experience in using this programming language. My reasoning to do so was for Python's simplicity, but more significantly, it has built in libraries that are best suited to my project. Nonetheless, I was able to profoundly use the knowledge gained from studying the Java and C++ programming languages in a variety of modules learnt throughout my university career, and applying this knowledge to Python. As such, the use of Python in my project was not a limiting factor at all.

Scale Of Project

Due to the inevitable limited time constraints and the overall large scope of the project, I decided to scale the project down and present it as a "Beta" software in that the project will be less feature complete. For instance, only a few queries will display search results and thus definitions and example code. Nonetheless, the tools I have used, created, and implemented to enable these limited results can still be scaled up and used in the future product life cycle to generate a greater number of search query results.

Project Stakeholders

- **Project Team:** Julian Smith
- **Project Supervisor:** Jane Challenger Gillitt
- **Project Second Reader:** Gulden Uchyigit
- **Users:** The project will be available to all users, regardless of their programming ability, but more aimed towards programmers
- **Development Tools:** JetBrains S.R.O's "PyCharm" Python IDE (used to program the back-end of project) and their "PhpStorm" HTML, PHP, Javascript and CSS IDE (used to program the front-end of the project). Oracle Corporation's "MySQL Workbench" was used to create and maintain the database

Methods Of Communication

The project required communication with the project supervisor Jane Challenger Gillitt, whereby areas of progress and challenges that occurred were communicated to ensure that the aims and objectives of the project were completed. Throughout the course of the project, consolations occurred at least once a month as I managed the project efficiently and met my proposed deadlines.

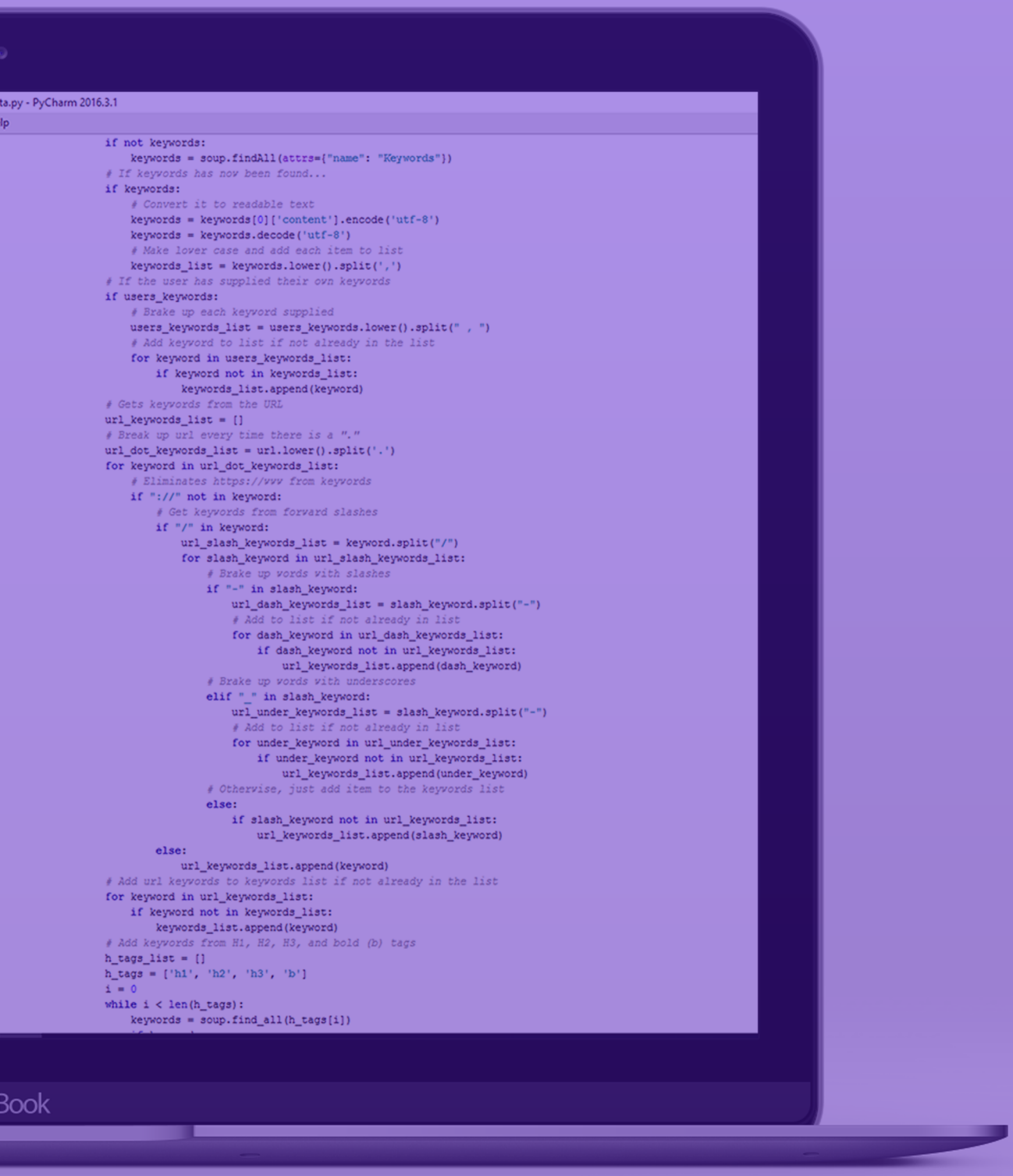
Project Methodology

The project ran as an agile development, predominantly due to the time constraints occurred on the project. By running following agile development principles, it ensured that as many of the prioritised requirements were completed prior to the project deadline. More significantly, this agile approach allowed for some of the requirements to change and even become a higher priority whilst the project was still in development. For example, the issue of the large scope of the project become notable as the project was underway and thus caused the situation were requirements were needed to be adapted. Fortunately, due to the project being agile, this change was easily implemented and did not negatively affected the project as a whole. More so, as a result of the feedback that was received via the user testing, a greater number of requirements and areas of focus arose. These changes were quickly and readily implemented directly due to the project running as an agile development.

Project Requirements List

- **The Project Must Have:**
  - Web Crawler and Indexer: A web crawler and indexer that can successfully visit a website that the user inputs and will then visit every page within the given website. It must also have the capacity to then index (save) all the relevant data from each web page into a purpose built database.
  - Data Gathering: The web crawler and indexer must autogenerate the title, description , and keywords for each page that the crawler will visit. The user should also be able to input to the web crawler the “class” or “id” name of the element that contains a definition or example code.
  - Website: A website where users can search for queries, with the search results generated from the database containing the data gathered from the web crawler and indexer displayed to the user. A definition of the searched query as well as an example piece of code before the search results must also be displayed if relevant to the search query.
  - Usability: The website must be easy to use, visually pleasing, and sufficiently intuitive so that the user will require no instructions. The success of this requirement is measured by the findings gathered from the user testing.
- **The Project Should Have:**
  - Search Results Ranking: Search query results should be displayed and thus ranked in an order that prioritises the most relevant search results.
  - Pagination: The search query results should be displayed on multiple pages (pagination) if there is a large volume of them generated.
  - Update Indexed Results : There should be the ability to check for any changes and update the corresponding indexed webpage in order to keep the content up to date

- **The Project Could Have:**
  - Website Animations: JavaScript or jQuery animations in the loading of the search query results to improve the usability and overall look and feel of the website.
  - Search By Programming Language: Enable the user to be able to limit results relevant to a chosen programming language.
  - Recommendations: If the user incorrectly spells a query or no results are displayed, then the website could for example display “Did you mean…?” followed by the suggested query.
  - Search Autocomplete: When the user searches for a query, a suggested queries that would aim to predict what the user would inevitably search for would appear.
- **The Project Would Like:**
  - Commerciality: The ability for third parties to submit their website for indexing, sponsored search results, or even “trusted” definitions and code examples
  - Greater Search Variety: Provide the web crawler and indexer, database, and thus the website the capability display images and video search results.
  - Internationality: The website to be displayed in multiple languages depending on where the user accessing the website is located.


ta.py - PyCharm 2016.3.1
lp
if not keywords:
keywords = soup.findAll(attrs={"name": "Keywords"})
# If keywords has nov been found...
if keywords:
# Convert it to readable text
keywords = keywords[0]['content'].encode('utf-8')
keywords = keywords.decode('utf-8')
# Make lover case and add each item to list
keywords_list = keywords.lower().split(',')
# If the user has supplied their own keywords
if users_keywords:
# Brake up each keyvord supplied
users_keywords_list = users_keywords.lower().split(" , ")
# Add keyword to list if not already in the list
for keyword in users_keywords_list:
if keyword not in keywords_list:
keywords_list.append(keyword)
# Gets keywords from the URL
url_keywords_list = []
# Break up url every time there is a "."
url_dot_keywords_list = url.lower().split('.')
for keyword in url_dot_keywords_list:
# Eliminates https://vvv from keyvords
if "://" not in keyword:
# Get keywords from forvard slashes
if "/" in keyword:
url_slash_keywords_list = keyword.split("/")
for slash_keyword in url_slash_keywords_list:
# Brake up vords vith slashes
if "-" in slash_keyword:
url_dash_keywords_list = slash_keyword.split("-")
# Add to list if not already in list
for dash_keyword in url_dash_keywords_list:
if dash_keyword not in url_keywords_list:
url_keywords_list.append(dash_keyword)
# Brake up vords with underscores
elif "_" in slash_keyword:
url_under_keywords_list = slash_keyword.split("-")
# Add to list if not already in list
for under_keyword in url_under_keywords_list:
if under_keyword not in url_keywords_list:
url_keywords_list.append(under_keyword)
# Othervise, just add item to the keyvords list
else:
if slash_keyword not in url_keywords_list:
url_keywords_list.append(slash_keyword)
else:
url_keywords_list.append(keyword)
# Add url keywords to keywords list if not already in the list
for keyword in url_keywords_list:
if keyword not in keywords_list:
keywords_list.append(keyword)
# Add keyvords from H1, H2, H3, and bold (b) tags
h_tags_list = []
h_tags = ['h1', 'h2', 'h3', 'b']
i = 0
while i < len(h_tags):
keywords = soup.find_all(h_tags[i])
Book

Deliverables

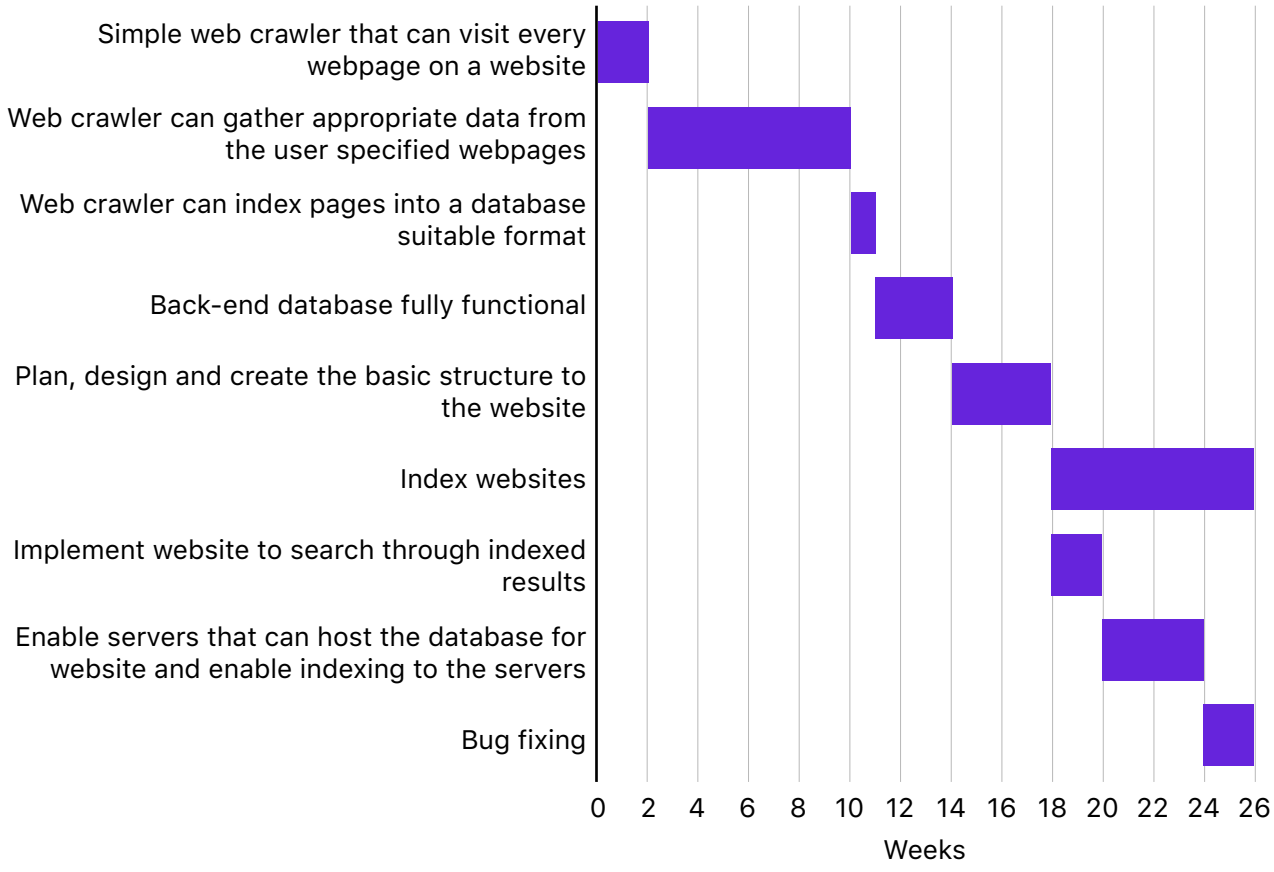
Project Deliverables

- A front-end website (Code Languages: HTML, CSS, PHP, and JavaScript)
- A back-end web indexer and crawler (Code Language: Python)
- A back-end database containing indexed results (Code Language: MySQL)

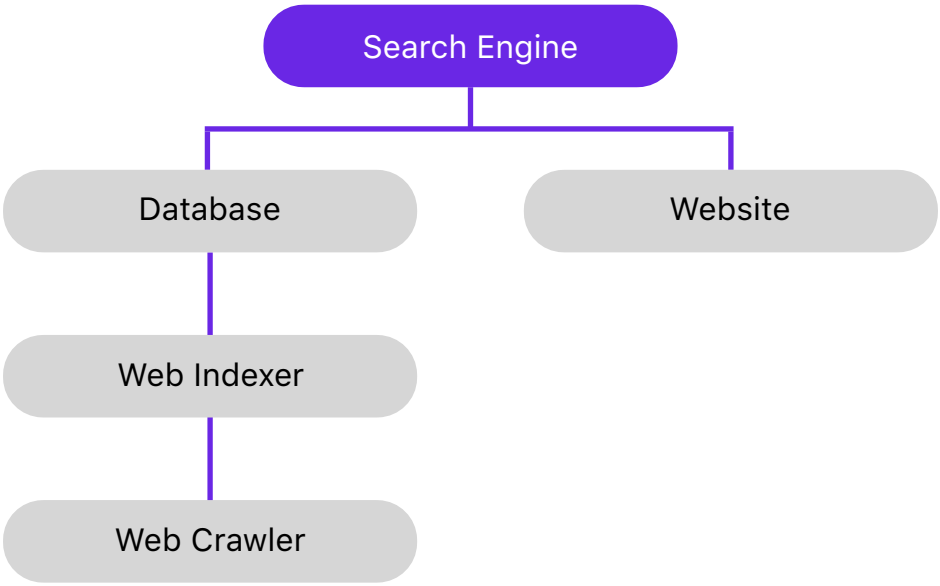
Schedule of Activities

- 14th December 2016: The web crawler is successfully implemented whereby it can gather relevant data on each website. Furthermore, it can also index pages into a database suitable format.
- 7th January 2017: Back-end database is fully functional.
- 31st January 2017: Front-end website is created and is functional.
- 1st February 2017: First instance of user testing for participants to test elements of the website and give a variety of qualitative and quantitive feedback.
- 2nd February 2017: At this stage bug fixing and any improvements, new features, design changes, due to the findings gathered via the user testing will be implemented into the system during this time.
- 1st March 2017: Second instance of user testing is conducted, where the system is tested again using the same criteria as before.
- 2nd March 2017 Onwards: Bug fixing the system and any more improvements needed which were highlighted from the second user testing are implemented.

Gantt Chart



Product Breakdown Structure



Risk Analysis

The nature of the project itself bought about a number of of risks. For instance, heavy time constraints, areas in which I had little practical experience in, and other commitments outside of the project, created a large volume of risks that ultimately could of lead to the project failing. As such, this risk analysis will focus on the three most concerning risks that the project faced.

The possibility of running out of time and failing to complete stages of the project was highly possible throughout the course of the project due to heavy time constraints that were in place. This is exacerbated by the fact that I was the only person working on this project, in conjunction with other responsibilities to my studies, and so the probability of this risk occurring was high. To mitigate the occurrence this risk, I decided to initially focus solely on completing the must have requirements of the project as a priority, with any 'secondary' requirements being completed afterwards. More significantly, as mentioned previously, I took the decision to scale the project back and present it as “beta” software due to the adverse time constraints. Furthermore, due the use of agile development in the running of the project, I was able to quickly and efficiently change the priority of more challenging, time-consuming requirements of the project as the project neared completion. Thus, ensuring that as many requirements as possible were completed before the project timeline ended.

Another related risk that occurred was the possibility of certain areas of the project taking far longer to complete than previously estimated. For example, it took me a greater amount of time than anticipated to successfully gather all the required data, as, quite simply, I've never created a search engine before. Thus, the areas within the project that I had little experience in previously exercising, and concepts that were not totally familiar to me such as setting up a virtual private server, took far longer than anticipated. To reduce the possibility of this risk having a greater detriment in the production of my project, prior to starting the project, and throughout its course, I undertook extensive research into unfamiliar areas of the project before taking them on, to inevitably avoid the smaller problems compounding and in tern creating much larger problems in the long run. One key area where this

was the case was prior to the production of the front-end website, where I researched heavily into PHP and how to navigate and present database results.

The third significant risk that the project faced concerned the creation in and appending data to the database of indexed websites, where I struggled in how I practically went about producing this. This is because the resources available in this particular matter were rather limited, and thus bought about some uncertainty concerning the success of this project as a whole. Fortunately, after many days of research and experimentation, I discovered the third party Python module named “PyMySQL” thanks to the fantastic book “Web Scraping in Python” by Ryan Mitchell (see Annotated Bibliography). Using this module that I discovered and using the resource previously mentioned, I was successfully able to implement this part of the project with some ease, mitigating the chance of further risks in this area occurring. Furthermore, to further reduce the risk imposed on the content by this risk, prior to the projects production, I designated more time than previously anticipated to this area of the project, to ensure its completion especially as it is a fundamental requirement of the project.

```
--------------------------------------------------------------------------------
|                              Welcome to Apropos                              |
--------------------------------------------------------------------------------
|          To get started, type the website you want to crawl's URL below      |
--------------------------------------------------------------------------------
Website URL = www.brighton.ac.uk
WARNING | URL does not contain HTTP or HTTPS request
Add HTTP [1] , add HTTPS [2] , continue [3] = 2
--------------------------------------------------------------------------------
Is https://www.brighton.ac.uk the website you want to index?
Enter Y for Yes or N for No = y
--------------------------------------------------------------------------------
Maximum number of pages to index (0 = No Limit) = 500
--------------------------------------------------------------------------------
Enter Advanced Options?
Enter Y for Yes or N for No = y
--------------------------------------------------------------------------------
|                              Advanced Options                                |
--------------------------------------------------------------------------------
|    In advanced options you can set specific class names to items Apropos indexes   |
|       If you don't want to make changes to one of the options, leave it blank      |
--------------------------------------------------------------------------------
DEFINITION & CODE SETTINGS | Enter settings to enable definition and code gathering,
press Y to enter = y
--------------------------------------------------------------------------------
|                           DEFINITION & CODE SETTINGS                         |
--------------------------------------------------------------------------------
DEFINITION DIV ID NAME | Enter the ID name for the DIV which contains the definition =
DEFINITION DIV CLASS NAME | Enter the CLASS name for the DIV which contains the
definition =
CODE CLASS NAME | Enter the class name for the DIV which contains the code section on
each web page =
TITLE CLASS NAME | Enter the class name for the class which contains the title on
each web page =
DESCRIPTION CLASS NAME | Enter the class name for the class which contains the
description on each web page =
Keywords to apply to whole website (separated by " , ") = brighton , university
INDEX CRITERIA | Only index the web page if  it contains the following text on the
web page. Separate different items by adding " , " between each item = brighton
Turn Indexing Off? Enter Y for Yes = y
--------------------------------------------------------------------------------
Apply changes? Enter Y for Yes or N for No = y
--------------------------------------------------------------------------------
Visiting https://www.brighton.ac.uk
Downloading = https://www.brighton.ac.uk
--------------------------------| SUCCESS |-------------------------------------
TITLE       = University of Brighton
URL         = https://www.brighton.ac.uk
DESCRIPTION = The University of Brighton is a thriving, multi-campus university on the so
KEYWORDS    = university of brighton , brighton university , university , brighton , east
 graduate , london , britain , design , arts , uk , ac , of , why , choose , brighton , u
loyer , pledge , contact , us , quick , links , information , for...
CODE        = NONE
DEFINITION  = NONE
--------------------------------------------------------------------------------
Links Left  = 1 / 500 | 0.20%
Downloading = https://www.brighton.ac.uk/siteinfo/accessibility/index.aspx
--------------------------------| SUCCESS |-------------------------------------
TITLE       = Accessibility
URL         = https://www.brighton.ac.uk/siteinfo/accessibility/index.aspx
DESCRIPTION = Accessibility information for the website
KEYWORDS    = brighton , university , ac , uk , siteinfo , accessibility , index , high/l
contact , us , quick , links , information , for...
CODE        = NONE
```

Project Implementation

Project Naming

After careful consideration, I inevitably decided to name the project "Apropos". The word apropos has a connection to Unix terminal code, where it is used as a command to display a list of items relating to the query which follows the command. Furthermore, in the British English language, the word apropos means to "reference to" and "to concern", and is synonymous with the word "appropriate". As such, the name Apropos is befitting to my project as, like in Unix terminal code, my project displays appropriate items related to the users query.

Design

The aim of the overall design of Apropos is to look modern and minimalistic. Furthermore, and most significantly, it must be easy to use to the point where no instructions are required, in essence, the design must be intuitive.

To start the design of Apropos, I first created a simple colour scheme of five colour that each complement one another (see figure 3.1). For example, I used a light blue colour contrasted with white or an off-white colour to draw attention to the content within. More significantly, I created an elegant gradient using the two blue toned colours to create a modern and clean background for use as the homepage.

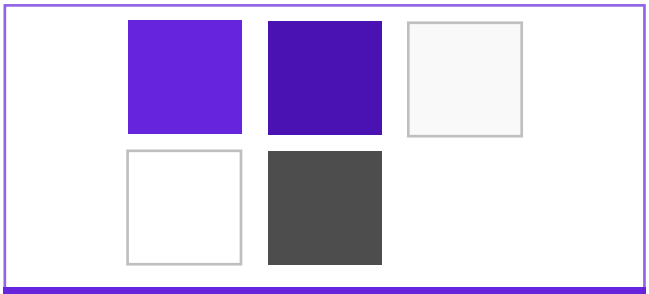


Figure 3.1

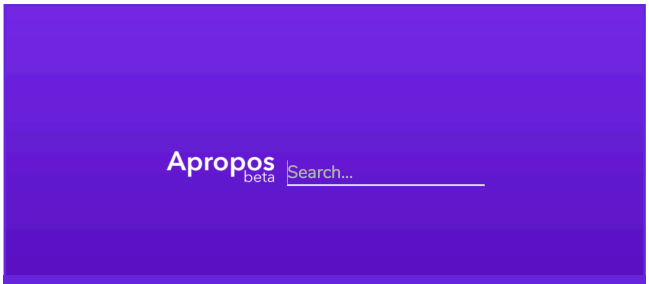


Figure 3.2

I then began to design the main elements of the website such as the search bar, search query results, and the logo locations. Here I aimed to use simple, clean, straight lines in order to achieve the modern and minimalistic view. This is evident in the design of the search bar, where I simply added a white line at the bottom to denote it (see figure 3.2). Furthermore, via the use of Javascript, I implemented a piece of code which automatically has the user inside the search bar on the home screen when they load the page in order to continue the sense of simplicity.

Lastly, I decided to design elements that provided feedback to the user, such as hover animations on the search query results. For example, I added a darker drop shadow around the search query results box to denote that they are clickable, and intern, do an action (see figure 3.3).

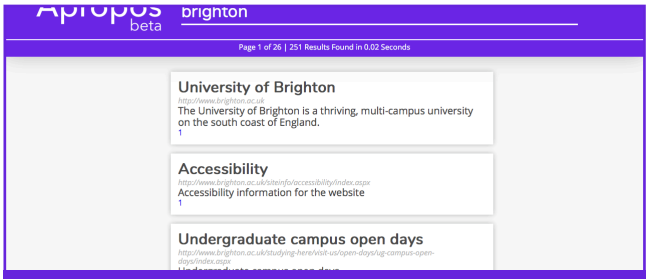


Figure 3.3

Progress Made

Over the course of the project timeline, I made great progress in completing all the required elements of my project. For instance, I successfully created the back-end web crawler and indexer which can visit a chosen website, find every link on that website, and then index the relevant data found to a database. Furthermore, it can also autogenerates keywords words, a description, and a title for each page via content found on the page itself and in the URL. Additionally, I managed to design and create the front-end website whereby users can search for queries, with the search results of these search queries displayed to the user via accessing the database. Most significantly however, a definition of the searched query as well as an example piece of code is displayed if relevant on a number of queries. More significantly however, the front-end website is intuitive and easy to use due to the design of the website, and also from the findings of the user testing that was conducted (see Project Evaluation section).

I also managed to successfully complete a number of not strictly needed requirements to the project, but features that should be present in the project. For example, I implemented pagination into the front end website whereby if there are more than ten search query results returned, then these remaining results will be displayed on another page. This requirement was mentioned by many on the first instance of user testing, and thus was made more important that other similar requirements. As a result, this vastly improved the usability of the website, an improvement which can be seen in the second instance of user testing (see Project Evaluation section).

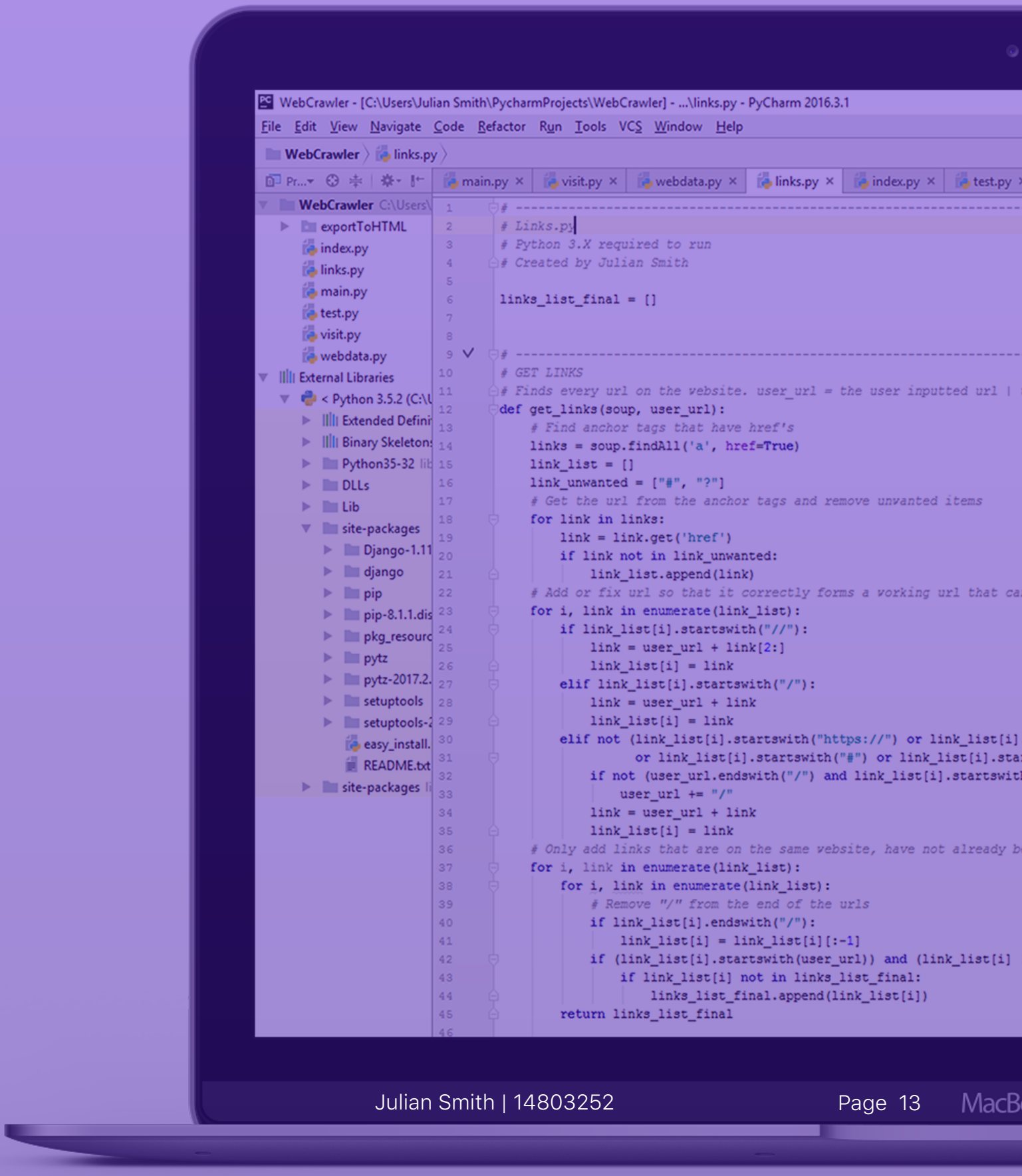
Challenges

One area of the project that produced a significant challenge to overcome arose prior to successfully implementing the website where the project was only functioning in its entirety on my local machine. Thus, this arises the scenario where I cannot submit my project due to this limitation of the system. After a discussion with my project supervisor about this challenge, I decided to purchase a domain name and hosting package via a third party in order to publish my project onto the internet. I managed to successfully upload my website to the hosting package, where it was correctly working online, however I had problems with implementing the database. After spending several days trying to establish the cause of this issue, I discovered this challenge arose over the limitation within the hosting package's MySQL, which the database runs on, where it does not allow for "root user" access. Consequently, this meant that the back-end web crawler and indexer could not index the data it gathers directly to this remote database due to the limited access it had in place, a vital part of the project. As such, I inevitably had to purchase a virtual private server (VPS) where after conducting research into them, I discovered that they do not limit the "root user" access in the MySQL databases. Purchasing the VPS resulted in my database and thus enabled the website to function remotely.

Another challenge occurred when the back-end web crawler visited certain websites, in that sometimes the servers that host websites and their links were sometimes down, or the web page simply doesn't exists anymore. This bought about the problem where the program would terminate due to this error occurring, thus stopping the system from continuing to crawl and index the website. Consequently, this could have adverse affects in the production of indexing websites, especially websites which are exceeding large and contain many links. In order to overcome this challenge, I implemented the "try" and "except" clauses into my program, whereby

the program will “try” to implement the following code, where if it were to throw an error (“except”), it would implement that corresponding code in place. Furthermore, I linked these error handling clauses to the HTTP module that I used in order to provide the user with a more meaningful error code. Consequently, by using these clause statements, I was successfully able to overcome the challenge of the program terminating if an error occurs.

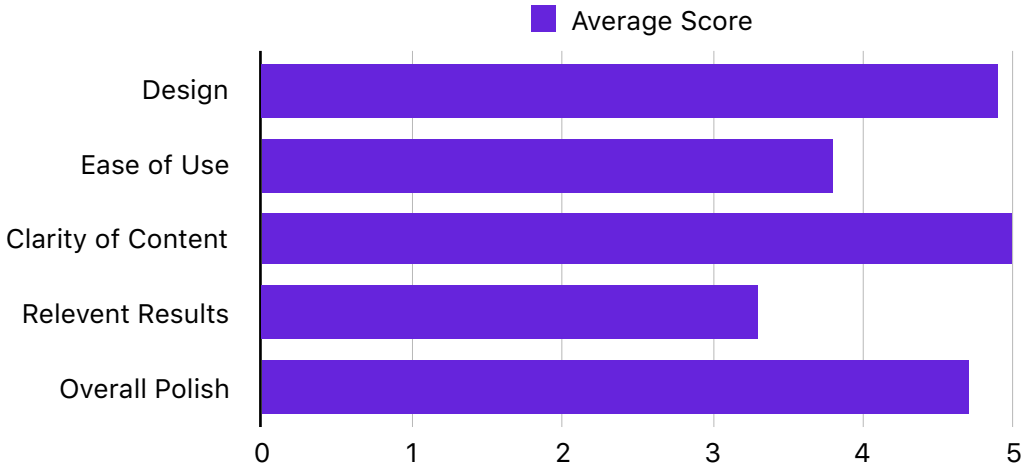
Generating keywords for the website bought about many challenges, mainly due to the them being used by the website to display search results based off these keywords. For example, as well as gathering keywords provided by the website itself, the system also needed to generate keywords from the content of the webpage so that the web page’s keywords are accurate to it’s content. Challenges arose in efficiently and accurately gathering keywords from the URL, as URLs vary in semantics website to website. For instance, one website’s url might be written as “http://www.keywords.co.uk/key_words/can_be_found_here" or even “http://keywords.website.com/here-are-some-keywords". Consequently, it is vital to ensure that the gathering of keywords from the URL is universal in its techniques. My first attempts at achieving this goal were successful initially, but after testing it on a variety of websites, the system was providing illogical keywords. For example, sometimes the system would ignore certain blocks of “/“ and not return the keywords inside of them. To overcome this challenge, I successfully created a technique to extract keywords from the URL via the use of “for” and “if” clauses. For example in short, the system will first break up the url by the occurrence of “.”, and then break these up by “/“, then once more by the occurrence of “-“ or “_”. These are then added to the list of keywords if not already in the list, or are keywords that are unwanted such as “?”, “com” or “index” etc.


WebCrawler - [C:\Users\Julian Smith\PycharmProjects\WebCrawler] - ...\links.py - PyCharm 2016.3.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
WebCrawler
links.py
main.py
visit.py
webdata.py
links.py
index.py
test.py
WebCrawler C:\Users\
exportToHTML
index.py
links.py
main.py
test.py
visit.py
webdata.py
External Libraries
< Python 3.5.2 (C:\
Extended Defini
Binary Skeleton
Python35-32
DLLs
Lib
site-packages
Django-1.11
django
pip
pip-8.1.1.dis
pkg_resourc
pytz
pytz-2017.2.
setuptools
setuptools-
easy_install.
README.txt
site-packages
# Links.py
# Python 3.X required to run
# Created by Julian Smith
links_list_final = []
# GET LINKS
# Finds every url on the website. user_url = the user inputted url
def get_links(soup, user_url):
# Find anchor tags that have href's
links = soup.findAll('a', href=True)
link_list = []
link_unwanted = ["#", "?"]
# Get the url from the anchor tags and remove unwanted items
for link in links:
link = link.get('href')
if link not in link_unwanted:
link_list.append(link)
# Add or fix url so that it correctly forms a working url that ca
for i, link in enumerate(link_list):
if link_list[i].startswith("//"):
link = user_url + link[2:]
link_list[i] = link
elif link_list[i].startswith("/"):
link = user_url + link
link_list[i] = link
elif not (link_list[i].startswith("https://") or link_list[i]
or link_list[i].startswith("#") or link_list[i].sta
if not (user_url.endswith("/") and link_list[i].startswit
user_url += "/"
link = user_url + link
link_list[i] = link
# Only add links that are on the same website, have not already b
for i, link in enumerate(link_list):
for i, link in enumerate(link_list):
# Remove "/" from the end of the urls
if link_list[i].endswith("/"):
link_list[i] = link_list[i][:-1]
if (link_list[i].startswith(user_url)) and (link_list[i]
if link_list[i] not in links_list_final:
links_list_final.append(link_list[i])
return links_list_final

Project Evaluation

User Testing

The user testing involved ten participants aged between twenty and twenty three, all of which attended the University Of Brighton. Within the testing, the participants performed the same activity twice, one month apart. The activity involved performing tasks that enable the participant to navigate throughout the website, evaluating different key areas, gathering quantitive and qualitative feedback. For example, participants had to search for certain queries, and then rate key areas of the website out of a score from one to five. These key areas include: the design, ease of use, clarity of content, relevant results returned, and the overall polish of the website. The participants could also provide an optional comment about an area of the website that might require improving.

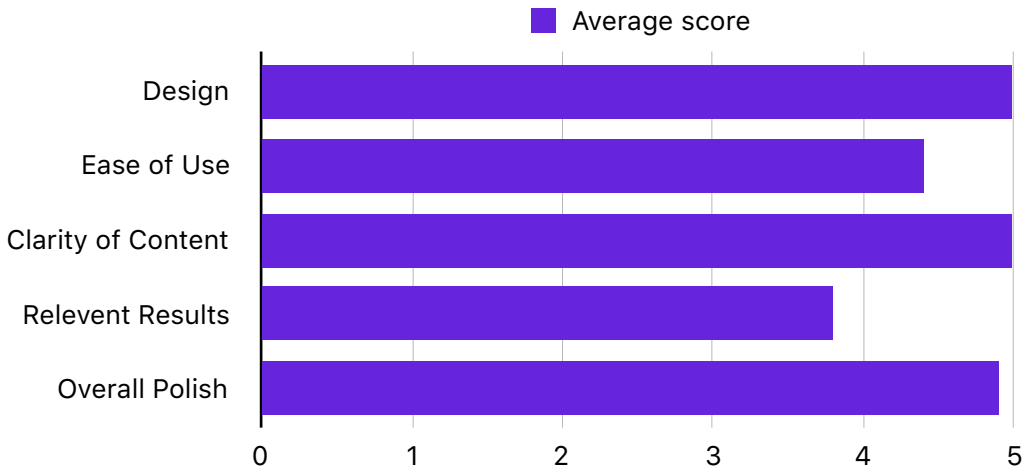


Areas Of Testing	Participants										Average Score
	1	2	3	4	5	6	7	8	9	10	
Look of Website	5	5	5	4	5	5	5	5	5	5	4.9
Ease of Use	4	3	4	4	5	3	3	4	4	4	3.8
Clarity of Content	5	5	5	5	5	5	5	5	5	5	5
Relevent Results Returned	3	3	3	3	3	4	3	3	4	4	3.3
Overall Polish of Website	5	4	4	5	5	5	5	4	5	5	4.7

Participants		Participants	
1	N/A	6	Lots of results
2	Too many results on the page	7	N/A
3	N/A	8	N/A
4	N/A	9	Text quite dim on homepage
5	N/A	10	N/A

Figure 4.1

The results of the first user testing can be found in figure 4.1. Here, the findings suggest that the design and clarity of the website are of particularly good quality, with both of these areas receiving an average score of 4.9 and greater. However, one key area that is notable as a result of the user testing that requires significant improvement is the relevance of search query results returned, having only received an average score of 3.3. Another improvement that was found arose in the optional comment section of the testing, where a small but notable number of users requested pagination to be implemented into the system. Consequently, after the first instance of user testing, I decided to focus on returning relevant queries to the user and also to implement pagination into the system.



Areas Of Testing	Participants										Average Score
	1	2	3	4	5	6	7	8	9	10	
Look of Website	5	5	5	5	5	5	5	5	5	5	5
Ease of Use	4	4	5	4	5	4	4	5	5	4	4.4
Clarity of Content	5	5	5	5	5	5	5	5	5	5	5
Relevent Results Returned	4	4	3	4	4	5	3	4	4	5	4
Overall Polish of Website	5	4	5	5	5	5	5	5	5	5	4.9

Participants		Participants	
1	Some repeated results	6	N/A
2	N/A	7	N/A
3	N/A	8	N/A
4	N/A	9	N/A
5	N/A	10	N/A

Figure 4.2

Findings from the second instance of user testing are shown in figure 4.2, where the average score on all areas of testing positively increased. More significantly, relevant results returned had the largest on average increase in score

with a rise of 0.7, an area of the system highlighted in the first user testing that required attention. Thus, this demonstrates that the actions taken to improve the relevance of such results has been successful, however the average score is still rather low compared to the other areas of the system. Furthermore, this area also received the lowest score by individual participants with a score of 3, consequently indicated that this area still requires some improvement despite its improvement. More so, participant one left a comment that there are some repeated results, an issue or bug that could have attributed to this score not being higher. The average score in the ease of use of the website also saw a substantial increase, indicting that improvements in these areas also were successful. These improvements could be inferred to have caused the small but notable increase in the average score of the overall polish of the website.

Areas Of Improvement

An area of the project that would require further improvement would be in the display of the returned results, where the most relevant page will be shown first. As mentioned previously, the relevance of returned search query results remained the lowest averagely scoring area of the system by participants in the user testing, thus suggesting that this area still requires improvement. In the current system, the search results are found and displayed using MySQL's built in "match()" function which uses natural language processing to compare two strings together. While using this method is far more in depth than using the "like" clauses which the system originally used, it does not provide the same level of page ranking as more well known and widely used browsers do. For example, amongst a variety of other techniques, Alphabet Inc.'s Google uses a technique named PageRank to rank pages based on their hierarchy in the website. Whilst vastly technically challenging, a page ranking technique like this in my project would improve the quality of returned results tremendously.

Areas Of Success

One area of the project that I believed to be a profound success was the design and implementation of the website. I personally feel that the design of the website looks exceeding good and I believe that I have achieved my aims for it to have a modern and professional appearance. More significantly, the design of the website in the first user testing received the highest score, with every participant in the testing scoring this area the highest mark possible. As such, this data further adds to my appreciation of this area, and highlights how great the appearance of the website is.

Another area of the project that I am particularly proud of is in the gathering of the required data such as the title, description, keywords, definitions, and programming code sources. This is the area of the project which I spent the most amount of time on, and thus have the most admiration for to see it all functioning correctly. More specifically though, it successfully gathers all the required data from a large variety of different websites, meeting the exact requirements that were set in place prior to the projects implementation.

One more area of the project which was a success was the web indexer and crawler being able to connect to a remote server and index the search results to it. As mentioned previously, I encountered many challenges in implementing this feature into the project due to third parties supplying servers that limit the access and privileges available to the server. Furthermore, it was rather technically

challenging to set up the server and to ensure that it can infect be access remotely. Consequently, I am delighted to see this part of the project working as planned and in harmony with the web crawler and indexer, but also the website as well. More significantly however, without this area working as intended, the project as a whole will not function and ultimately useless.

Is The Project A Success Or Failure?

I would argue that the project was in fact a success, as I was successfully able to implement all the elements that my project required. For example, as previously mentioned, I was able to implement a web crawler and indexer that can visit a website and gather required data, where it can index this gathered data for display on a front-end website. More significantly though, I believe the project was also a success in that it has enabled me to use and refine the vast array of skills that I have learnt whilst studying at university. For instance, I was able to use my acquired skills in programming, human computer interaction, databases, creating websites, artificial intelligence algorithms. More so, since undergoing implementing the project, I have also seen an increase in the efficiency and my own understanding of programming languages in general. However, it could be argued that the project was a failure in that the smaller scope in place has meant that not as many technical aspects as previously planned has been implemented. Furthermore, some requirements of the project that should of been in the system such as a page ranking system were not implemented due to the adverse time constraints in place. Nonetheless, these early planed requirements were inevitably unfeasible, and the project which I have completed still represents as vast array of advanced technical skills. Therefore the project, despite its smaller scope, it still a success.

n 0.23 Seconds

putation for stability. In order to maintain this reputation,
on. It can be sometimes faster to fix bugs yourself and
s less people. Learn how to contribute. If you find a bug in

ocumentation

established a reputation for stability. In
ld like to...

ocumentation

established a reputation for stability. In
ld like to...

ocumentation

established a reputation for stability. In
ld like to...

Background Research

Impact Of Research On The Project

Due to the nature of my project, I conducted extensive research into a number of areas prior, and throughout the course of my project. For instance, I undertook in the study of large volumes of information concerning web crawling and the process of web indexing, looking also at the documentation of the programming languages that I used.

In the creation of the web crawler and indexer, the documentation for the third party modules that I used as well as for the Python programming language itself, were extremely valuable tools. For example, the documentation for the BeautifulSoup4 module was of great use, which without such a resource, I would of struggled immensely in its implementation. Furthermore, as mentioned previously, the knowledge I gained from studying in depth the book "Web Scraping in Python", written by Ryan Mitchell, provided me with a great understanding in how to successfully crawl and index a website. More so, this resource was of great help in teaching me how to allow and prepare for situations were the web crawler and indexer might throw an error. Consequently, the impact of this research on my project was extensive, where it made it far more technologically advance in a myriad of areas.

The impact of research in the creation of the website was tremendous. For example, prior to its implementation, I found a fantastic video tutorial on the video sharing site YouTube by user "Derek Banas" which described in detail about how to implement the database of indexed results into the website. This resource was further elaborated on by another video tutorial on YouTube by Adam Khoury which explains how to implement pagination into my website, a requirement that was shown to be needed after the first instance of user testing. Having no history of using PHP prior to the creation of this project, these resources were extremely valuable, which without them, the project could of possibly been infeasible within the timeframe in place. Furthermore, due to the breadth of content discussed in these resources, they caused this part of the project to be completed far quicker than previously anticipated, thus enabling more time to be spent on other areas.

I also conducted areas of research into the not so required elements of the project. For instance, I found great resources written by the founders of the world's largest search engine Google, where it describes in great detail how it crawls and index webpages, and more interestingly, how their page ranking (PageRank) and keywords (Hit List) algorithms work. Consequently, the information provided within these resources were very useful, as it provided me with valuable information on how to do several key elements of my project I need to include. Furthermore, this is made more significant considering that this research comes from the biggest search engine in the world. Unfortunately, due to the time constraints of the project, I was unable to include the knowledge concerning page ranking that I gained from reading these resources. Nonetheless, these particular sources still provided information on how exactly a web crawled and indexer works, which was incredibly valuable.

Bibliography

I. Brin, S. and Page, L. (2012) Reprint of: The anatomy of a large-scale hypertextual web search engine. Computer networks, 56(18), pp.3825-3833.

II. Page, L., Brin, S., Motwani, R. and Winograd, T. (1999) The PageRank citation ranking: bringing order to the web.

III. Oudinet, J. (2006) Search Engine Ranking.

IV. Mitchell, R. (2015) Web scraping with python: A comprehensive guide to data collection solutions. United States: O'Reilly Media, Inc, USA.

V. 2016 (2001) Overview — python 3.5.2 documentation. Available at: https://docs.python.org/3/ (Accessed: 20 November 2016).

VI. Richardson, L. (2015) Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation. Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (Accessed: 20 November 2016).

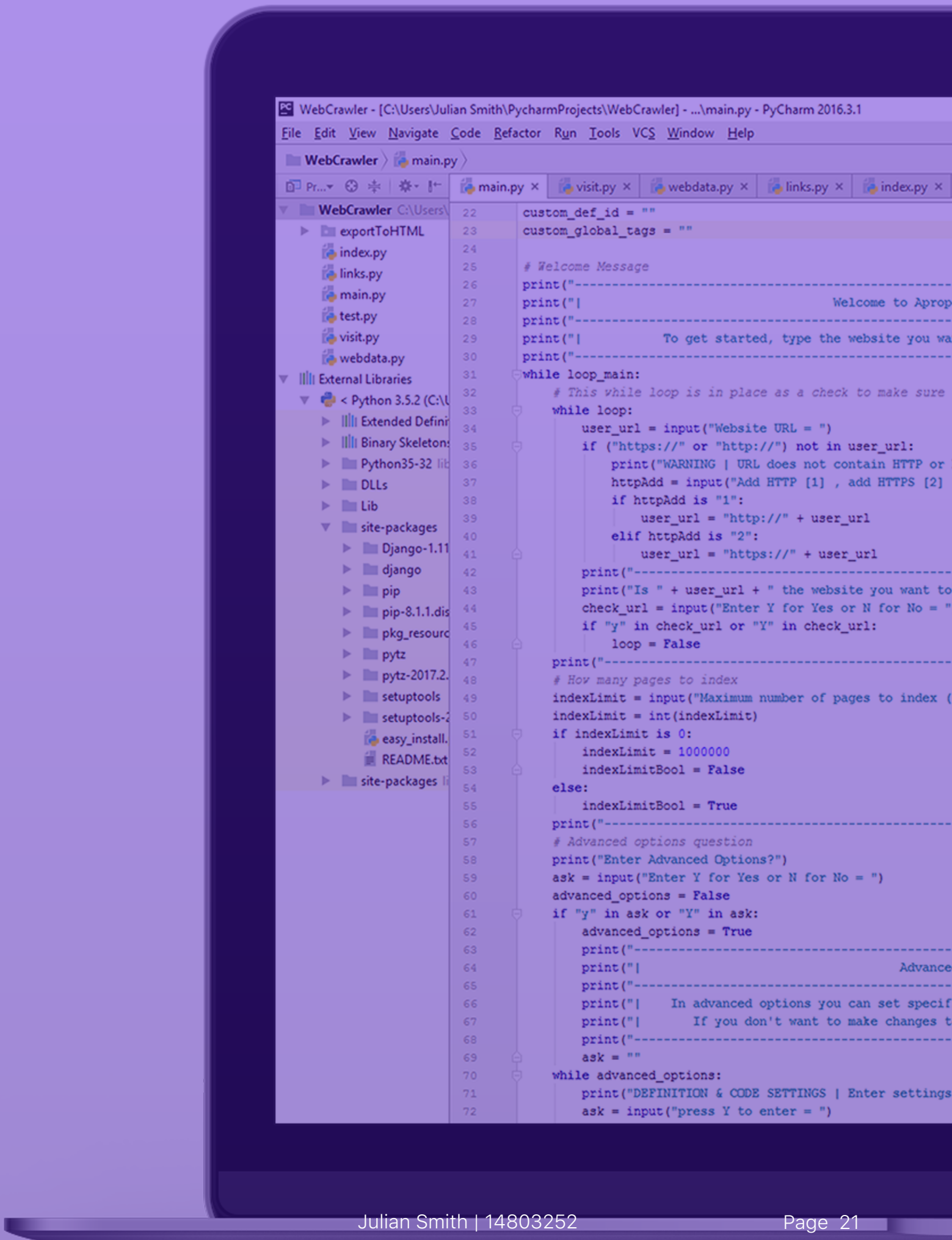
VII. W3Schools online web tutorials (no date) Available at: http://www.w3schools.com (Accessed: 21 November 2016).

VIII. van Rossum, G., Warsaw, B. and Coghlan, N. (2013) PEP 8 -- style guide for python code. Available at: https://www.python.org/dev/peps/pep-0008/ (Accessed: 21 November 2016).

IX. Derek Banas. (2014). PHP MySQL Tutorial. [Online Video]. 5 September 2014. Available from: https://www.youtube.com/watch?v=mpQts3ezPVg. (Accessed: 1 January 2017).

X. MySQL :: MySQL 5.7 Reference Manual :: 13.9.1 Natural Language Full-Text Searches. 2017. MySQL :: MySQL 5.7 Reference Manual :: 13.9.1 Natural Language Full-Text Searches. [ONLINE] Available at: https://dev.mysql.com/doc/refman/5.7/en/fulltext-natural-language.html. (Accessed 17 January 2017).

XI. Adam Khoury. (2013). PHP Pagination Tutorial MySQLi Google Style Paged Results Programming. [Online Video]. 31 July 2013. Available from: https://www.youtube.com/watch?v=T2QFNu_mivw&t=509s. (Accessed: 2 March 2017).


WebCrawler - [C:\Users\Julian Smith\PycharmProjects\WebCrawler] - ...\main.py - PyCharm 2016.3.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
WebCrawler main.py
main.py visit.py webdata.py links.py index.py
WebCrawler
exportToHTML
index.py
links.py
main.py
test.py
visit.py
webdata.py
External Libraries
Python 3.5.2
Extended Defini
Binary Skeleton
Python35-32
DLLs
Lib
site-packages
Django-1.11
django
pip
pip-8.1.1.dis
pkg_resourc
pytz
pytz-2017.2.
setuptools
setuptools-2
easy_install.
README.txt
site-packages
custom_def_id = ""
custom_global_tags = ""
# Welcome Message
print("----------
print("| Welcome to Aprop
print("----------
print("| To get started, type the website you wa
print("----------
while loop_main:
# This while loop is in place as a check to make sure
while loop:
user_url = input("Website URL = ")
if ("https://" or "http://") not in user_url:
print("WARNING | URL does not contain HTTP or
httpAdd = input("Add HTTP [1] , add HTTPS [2]
if httpAdd is "1":
user_url = "http://" + user_url
elif httpAdd is "2":
user_url = "https://" + user_url
print("----------
print("Is " + user_url + " the website you want to
check_url = input("Enter Y for Yes or N for No = "
if "y" in check_url or "Y" in check_url:
loop = False
print("----------
# How many pages to index
indexLimit = input("Maximum number of pages to index (
indexLimit = int(indexLimit)
if indexLimit is 0:
indexLimit = 1000000
indexLimitBool = False
else:
indexLimitBool = True
print("----------
# Advanced options question
print("Enter Advanced Options?")
ask = input("Enter Y for Yes or N for No = ")
advanced_options = False
if "y" in ask or "Y" in ask:
advanced_options = True
print("----------
print("| Advance
print("----------
print("| In advanced options you can set specif
print("| If you don't want to make changes t
print("----------
ask = ""
while advanced_options:
print("DEFINITION & CODE SETTINGS | Enter settings
ask = input("press Y to enter = ")

Documentation

Prerequisites

- Python 3.0 or higher (https://goo.gl/xpldVP)
- BeautifulSoup4 module (https://goo.gl/2OXs3u)
- PyMySQL module (https://goo.gl/fl6hJt)
- A web browser of your choice (latest version of Google Chrome is recommended)
- A constant internet connection
- A text editor or IDE of your choice

Running & Operating The Web Crawler & Indexer

To run the program:

1. Open the command prompt. On Windows: Start -> Run -> type “cmd” in the input box. *On macOS/OS X: Finder -> Applications -> Utilities -> Terminal*
2. Navigate to the location of the main.py file on the USB. In Windows and macOS / OSX: *cd [USB LOCATION]:\Apropos\Web Crawler and Indexer*
3. Run the program using the following command: “main.py” - If a problem occurs, see https://docs.python.org/3/faq/windows.html

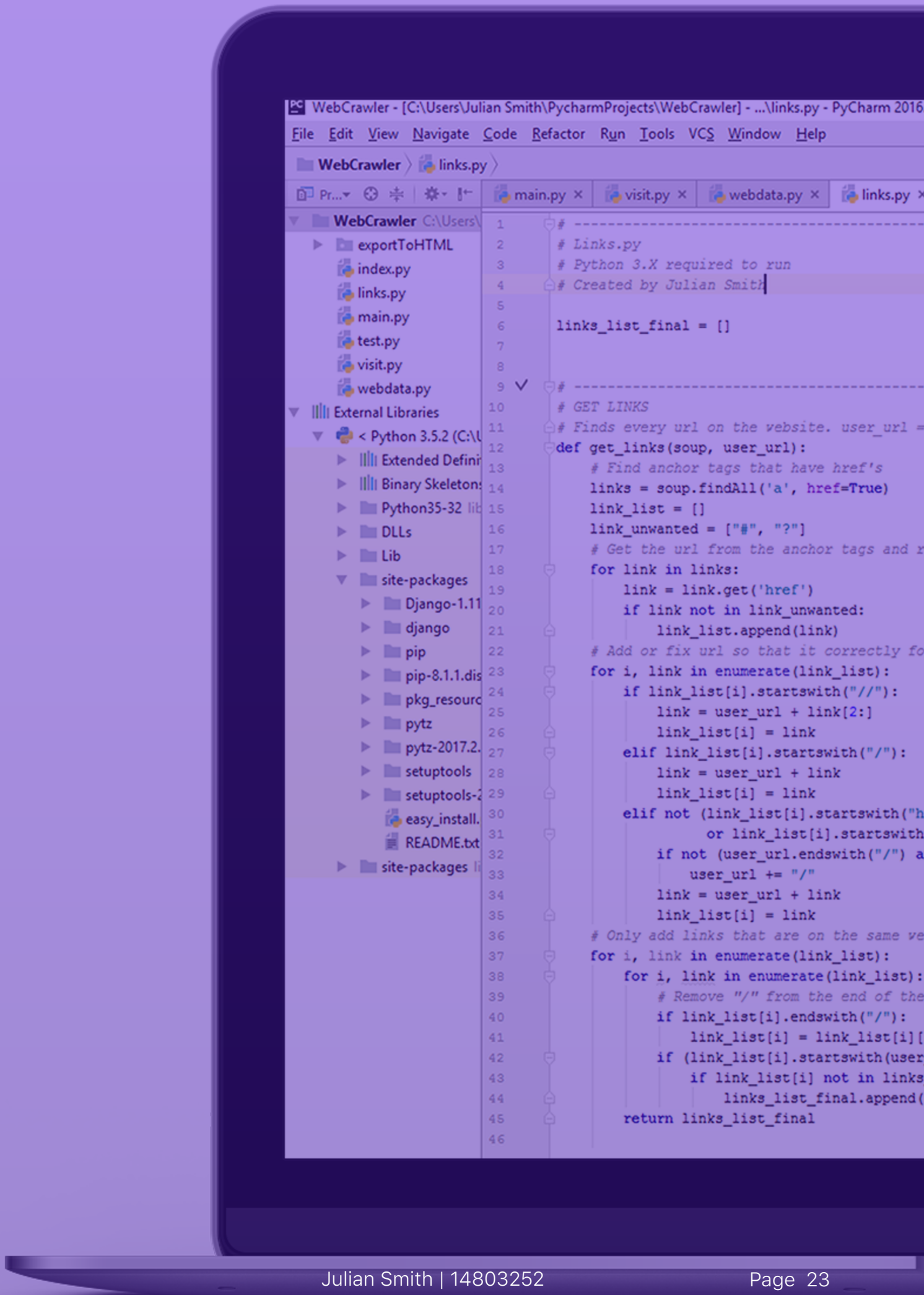
To operate the program:

1. First type in the website you want to index, for example “https://www.example.com"
2. The program will check if the website you inputed is the correct website to index, press “y” if it is, or “n” if it isn’t, and “enter”.
3. It will then ask you how many pages to index, if you don't want a limit to the number of pages, simply type “0” and “enter”.
4. The program will then ask you if you want to enter advanced options, if you do “y” to do so, or “n” if not, and “enter”.
5. If you chose to enter advanced options, you would be given the option to enter “Definition and Code settings”, type “y” to do so, or “n” if not, and “enter”. Here you can input the class name of the div that contain the definition and the code. You will also be presented with options such as the title and description class names of the divs that contain them. Also in advanced options you can add keywords that will be applicable to the whole website. For example, inputing “test , this is , a test” will apply “test”, “this is”, and “a test” as the keywords to the website. There is also the option to only index a page if it contains certain words, which is applied in the same way the keywords are applied. Lastly in advanced options you can turn indexing to the database off by typing “y” to do so, or “n” if not, and “enter”. Turning the index off is recommend for testing as results of which will appear on the front end website.
6. The system will then begin to index every page on the website until the index limit is reached or there are no more pages to visit.

Running The Website

To run the website:

1. On a browser of your choice, visit www.apropos.tech . To see the source code, open any file in the “Website” folder of the USB using a text editor or IDE of your choice.


WebCrawler - [C:\Users\Julian Smith\PycharmProjects\WebCrawler] - ...\links.py - PyCharm 2016
File Edit View Navigate Code Refactor Run Tools VCS Window Help
WebCrawler
links.py
main.py
visit.py
webdata.py
links.py
WebCrawler C:\Users\
exportToHTML
index.py
links.py
main.py
test.py
visit.py
webdata.py
External Libraries
< Python 3.5.2 (C:\
Extended Defini
Binary Skeleton
Python35-32
DLLs
Lib
site-packages
Django-1.11
django
pip
pip-8.1.1.dis
pkg_resourc
pytz
pytz-2017.2.
setuptools
setuptools-2
easy_install.
README.txt
site-packages
# Links.py
# Python 3.X required to run
# Created by Julian Smith
links_list_final = []
# GET LINKS
# Finds every url on the vebsite. user_url =
def get_links(soup, user_url):
# Find anchor tags that have href's
links = soup.findAll('a', href=True)
link_list = []
link_unwanted = ["#", "?"]
# Get the url from the anchor tags and r
for link in links:
link = link.get('href')
if link not in link_unwanted:
link_list.append(link)
# Add or fix url so that it correctly fo
for i, link in enumerate(link_list):
if link_list[i].startswith("//"):
link = user_url + link[2:]
link_list[i] = link
elif link_list[i].startswith("/"):
link = user_url + link
link_list[i] = link
elif not (link_list[i].startswith("h
or link_list[i].startswith
if not (user_url.endswith("/") a
user_url += "/"
link = user_url + link
link_list[i] = link
# Only add links that are on the same ve
for i, link in enumerate(link_list):
for i, link in enumerate(link_list):
# Remove "/" from the end of the
if link_list[i].endswith("/"):
link_list[i] = link_list[i][
if (link_list[i].startswith(user
if link_list[i] not in links
links_list_final.append(
return links_list_final

Project Log

Week 1	Started the project, made a simple program in python that can visit a user inputed website and download the content of the webpage.
Week 2	Created a for loop that can count the number of url links on a webpage, displayed an array list, and then visit the link thats the first item in the list and do the same. However, some of the links on the test pages are currently causing problems as the way they're coded, in that they don't exist.
Week 3	This week I created a method that fixes some of the problem links. For example, I created a system which adds the prefix url to lines that begin with "/" and also fixed links that begin with "//". I also made it so it was only able to find links on that webpage to avoid searching the whole web
Week 4	Using the BeautifulSoup 4 documentation, I was able to implement the web crawler to gather the title from the title tags of the website. Furthermore, I was also able to get other content such as the websites description and also keywords from the meta tags. I encountered a problem of the description formatting incorrectly for certain websites, with a large amount of leading "white space". To combat this, I discovered the python function ".trim()" which successfully fixes this problem. I also had my first supervisor meeting this week, where I spoke about how much progress I've made over the past couple of weeks. At the moment, everything is going to plan. My supervisor also provided me with tips about what to put in my interim planning report
Week 5	This week I began to generate my own keywords the URL of the website. For example, I discovered the split method in which I was able to break up the URL into key parts. Then using a collection of while and if statements, I trimmed each keyword block down and was left with a resulting string that displayed keywords from the URL.

Week 6	I was successfully able to implement into my system a way for the system to automatically generate a description based on content within the paragraph tags, if one had not been supplied. Furthermore, I also added a clause which would crop the description to only 160 characters and add "..." at the end in order for the descriptions to be of a uniformed length. I also had my viva meeting this week, where due to the recommendations of my project supervisor and the second reader, I decided to reduce the scope of the project and present it as a "beta software".
Week 7	I created an "Advanced Options" section where the user can edit certain areas of the crawler, and provide it with more custom information. For instance, I set it up so that the user could input content such as the class name of the div element that contained the definitions or the example code on the website. I also added "try" and "except" causes in the "visitSite" method in order to eliminate the issue of the program throwing errors when visiting a webpage.
Week 8	N/A - Christmas Holidays
Week 9	This week I created methods that are capable of returning definitions and example code, based on the customer information provided by the user. I did struggle with its implementation as the documentation provided by BeautifulSoup was not too clear on this area. Fortunately, I found resources that helped me to successfully implement this part of the project. I also reworked the code on the keywords generation section, where using far more efficient if statements, I was able to reduce the number of lines of code used by this section to produce the same outcome.
Week 10	N/A - Medical Issue
Week 11	I implemented the system to be able to index the data gathered to a database on my local machine, in which using the python module PyMySQL, I was successfully able to export this information. Setting up the database wasn't too challenging, as the install for MySQL pretty much did all the work to do so.

Week 12	This week I designed the front-end website, ensuring that it looked modern and professional. As well as designing all of the elements of my website and the colour scheme, I also designed the logo for my project.
Week 13	I made great progress in implementing the website this week. I was able to successfully translate my designs to HTML and CSS code, having previous experience doing so. Furthermore, I was also able to implement PHP code that is can return somewhat relevant results from the database to a search query.
Week 14	N/A - No progress made this week
Week 15	I formatted and styled the look of the returned results this week. I was also able, using PHP, to output the search query into the title of the page and also into the search bar itself. Furthermore, I also found a resource which describes in detail how to start the website so that the user is already inside of the search box on the homepage, using JavaScript code.
Week 16	After another meeting with my project supervisor this week, it came to my attention that my project was only able to run on my local machine. As such, this week I have purchased the domain name “www.apropos.tech" and a corresponding hosting package in order to enable my project to be presented remotely. However, I have encountered trouble in accessing the database supplied in my hosting package via the web crawler and indexer.
Week 17	The reason why my database was not working was due to the third party software limiting the access to the database. Consequently, I’ve purchased a virtual private server (VPS) which does not have these restrictions in place. As a result, I was able to connect to the VPS’s database via the web crawler and indexer, and also the website. This means that my website it fully functional on a remote machine.

Week 18	The first user testing took place this week and was a great success as I managed to get very meaningful data from the participants. One area that was suggested by the participants that was required was pagination on the search query results. Consequently, this week I implemented this feature into my website thanks to an extremely valuable resource on YouTube. I did not encounter too many problems with its implementation, due to how thorough the tutorial was.
Week 19	After using python for a considerable amount of time now, my skills in the language have vastly improved. As a result, I deleted and redid the keywords generation method, as I was not to happy with it's current state. This time around I used far more efficient and clean "for" clauses to navigate through the URL to generate keywords. Furthermore, I also enabled keywords via content in the H1, H2, H3, and bold tags of the website. I also implemented an advance option that enabled the user to input custom keywords that would be applicable across the whole website.
Week 20	Like in the week before, I totally reworked from the ground up the method that finds the links on each page, using the skills I have learnt through the course of the project. I used advanced for loops and list comparison techniques in order to make sure each link was unique, and also enable relative links to be visited by the crawler.
Week 21	The relevance of queries returned was an area assessed by the participants in the user testing that was highlighted to be a problem. To combat this, I discovered the built in MySQL method "MATCH()", which analyses the query and returns results based on natural language processing. Using this technique, I was able to find and rank page results far more correctly and efficiently then before. I did encounter some issues in its implementation however, with syntax error occurring causing the website to not return any data. Fortunately, the documentation on this method was of great help, and provided me with the information needed to overcome this problem.
Week 22	This week the second user testing phase took place. The data gathered from which highlights that the improvements to the system that I have made over the past month have been successful. This is especially so in the case of the relevance of query results which saw a tremendous increase.

Week 23	N/A - No progress made this week
Week 24	I mostly continued to index more websites to the database and also organise the database itself as it contained many repeated and old results. I also improved the method that gets the example piece of code to be formatted correctly.
Week 25	As the project nears completion, I performed some bug fixing on the system itself. For example, I fixed an issue of the footer of the website being displayed incorrectly and also removed any unused variables from the system itself.
Week 26	In the last week of the project I also focussed more on improving the overall polish of the system. For instance, I improved the wording of the web crawler, making it far more intuitive. Furthermore, I indexed a couple more websites to increase the total number of entries in the database.