[Date]

Maksim Livingstone

SQA AH COURSEWORK

URL path finder

Contents

[Description of the Problem 2](#_Toc21428588)

[Outline of problem. 2](#_Toc21428589)

[Scope 2](#_Toc21428590)

[Boundaries 3](#_Toc21428591)

[Constraints 3](#_Toc21428592)

[UML 4](#_Toc21428593)

[Requirements 5](#_Toc21428594)

[End user requirements: 5](#_Toc21428595)

[Survey 5](#_Toc21428596)

[Functional requirements: 5](#_Toc21428597)

[Project plan 6](#_Toc21428598)

[Identified tasks 6](#_Toc21428599)

[Resources required 6](#_Toc21428600)

[Estimate of timings 6](#_Toc21428601)

# Analysis

## Description of the Problem

I intend to design a program to find paths between webpages. The program will include the following features: a help screen, text-based UI to help user find their way, a database of webpages crawled that will be created using SQL, with a linked database of pages they have linked to and a pathfinding system.

The end users of my program will be people willing to find orphan links in webpages if a search system is not used, therefore will probably be tech literate.

My project meets the advanced higher computing requirements as it will have a UI suitable for tech literate users with validation for if the pages have valid URLs by using a try catch with a get() procedure and checking the code sent from the sever is not 404 and that the domain exists. My project will interface with an SQL database, creating a database and writing and reading URL’s from it. My project will also have a sorting algorithm for a verbose mode to print the node maps.

## Requirements

### End user requirements

* User must be able to use text-based UI to input a starting website, and an end website to find a path to.
* User must be able to input the number of moves they wish it to be done in.
* The user must be able to view a help page from the text based UI.
* The user must be able to view the pages that the spider finds without the pathfinder if they wish.
* The user must be able to view the requested path, or receive an error message that there is no path

### Functional requirements

* The program must display a text-based UI that can take in a starting page, end page(optional), number of moves and mode (input).
* The program will be able to crawl a URL and find all links on the URL, follow them, and repeat the process until the maximum jumps is achieved.
* The webpage’s URL is to be stored then all the links leading off also need to be stored in a database using SQL.
  + This is done by making the first field the current URL and the second field will be a link that leads off.
* URLs need to be stored and retrieved in a database using SQL from within the program
* Depending on the mode that the user requested, the program will either display the contents of the database (aka the results of the spider) sorted alphabetically or show the path between the requested URLS:
  + The program will then call the separate path finding algorithm.
  + SQL queries then need to be written into a 2d array.
  + (possible: sort 2d array so that easier to read.)
  + Then a node map object is created using the data in the 2d array.
  + Then using a pathfinding algorithm, find A path (not always shortest) from first link to second link in the maximum number of jumps given by the user.

Start Up

|  |  |
| --- | --- |
| Inputs |  |
| Processes | Check if program is being started in terminal or with text based UI |
| Outputs | Start-up state |

If start up state is with UI

|  |  |
| --- | --- |
| Inputs | Start-up state |
| Processes | Generate UI |
| Outputs | print UI to screen. |

Once UI is displayed right after start up

|  |  |
| --- | --- |
| Inputs | User inputs mode |
| Processes | Checks if mode exists in dictionary of commands and If it does, it calls the associated function. |
| Outputs |  |

If the path finder is called (even if UI isn’t specified):

|  |  |
| --- | --- |
| Inputs | Start web page and end webpage and number of jumps and whether reindexing is required |
| Processes | If reindexing is required:   * Call web scraper * initialise connection to database * Write output of scraper to database. * Load database into csv * Close connection to database * Execute pathfinder with start point and end point * Generate UI with pathfinder results. * Call UI object with contents   If reindexing is not required by the user:   * initialise connection to database * Load existing database into csv * Close connection to database. * Execute pathfinder with start point and end point * Generate UI with pathfinder results. * Call UI generator method with contents |
| Outputs |  |

When the results of scraping are called

|  |  |
| --- | --- |
| Inputs |  |
| Processes | If reindexing is required:   * Call web scraper * initialise connection to database * Write output of scraper to database. * Load database into csv * Close connection to database * Sort the webpages alphabetically * Generate UI with sorted map results. * Call UI generator method with contents   If reindexing is not required by the user:   * initialise connection to database * Load existing database into csv * Close connection to database. * Sort the results alphabetically if required by user (defaults to sorting it) * Generate UI with sorted map results. * Call UI generator method with contents |
| Outputs |  |

Generate UI

|  |  |
| --- | --- |
| Inputs |  |
| Processes |  |
| Outputs | * Print stored Text * Print prompt if required. |

Get prompts from user

|  |  |
| --- | --- |
| Inputs | Get user input |
| Processes | Check if user input is valid  Call related function from dictionary |
| Outputs |  |

### Scope

* A full design including pseudocode and data dictionary and query design for creating databases.
* A working data scraper and path finder, both being called from one system
* A help file that can be displayed in command prompt
* A test plan including test persona, test cases and outputs when different data is inputted and the results of the testing.
* End user survey to design functional requirements.
* An evaluation of the solution.
* The program will be tested and made for python 3.7 on a windows 10 machine
* This program will largely be tested on [www.wikipedia.com](http://www.wikipedia.com) as [www.wikipedia.com/robots.txt](http://www.wikipedia.com/robots.txt) says “# Friendly, low-speed bots are welcome viewing article pages, but not dynamically-generated pages please.” Which gives permission for bots to scrape [www.wikipedia.com](http://www.wikipedia.com) as long as the time between pings is reasonable.
* ~~Scope the clearly defined outline of what the solution will deliver in terms of functionality~~

### Boundaries

* Will not follow links onto a different domain.
* Max number of jumps in any direction will be 300 as too not slow down the computer.

UI will be text based only and not a graphical interface.

* It is not going to be designed nor tested for macOS or any windows installation prior to windows 10 or Linux installation.
* There will be no graphical representation of the possible paths due to time constraints.
* The program will not be packaged up into an installer so modules will need to be added manually on
* The path will not always be the shortest as paths may be reused from previous scraping.
* The program will not be tested with URLs that contain spaces as [RFC 1738](http://www.ietf.org/rfc/rfc1738.txt) (which has now been superseded by RFC  3986) states that a space is an unsafe character.

### Constraints

Some technical, legal and time constraints apply to this project.

* This project will need to be completed between 1st of October and 9th of March 2020 due to SQA requirements.
* This project can only use open source modules for data scraping and no commercial ones.
* I will be using primarily windows 10 and Linux Fedora for development.
* The project will be primarily written in python 3.7 as it is the language I am most proficient in, with the exception of the SQL that is being used to store and fetch the URL’s from the database and any external modules that are programmed in another language.
* This program has only been tested on websites with prior permission, due to the relatively large quantity of pings resulting from the scraping. Doing so without prior permission of the website holder may break the Computer Misuse Act.

~~Constraints the restrictions that apply to the development.~~

### Survey

An example survey that was used for the creation of the end user requirements.



A common request was for a help function with the text based UI as it’s not the most common method of using a system for many people. This was one of the examples that helped me create my end user requirements.

## UML

<https://www.lucidchart.com/invitations/accept/fcc0b091-b583-4fe0-82f1-93118e9e6ccf>

## Project plan

### Identified tasks

### Resources required

The resources that will be required at each stage of development are listed here:

|  |  |
| --- | --- |
| Analysis | * Gantt project 2.8.10 (windows) * Google chrome (windows) * Firefox (Debian) * Microsoft word (windows) * Survey monkey. * Internet connection. * Balsamiq mock-ups 3 |
| Design | * Microsoft word (windows) |
| Implementation | * Visual studio code 1.38.1 * Python 3.7 with scrapy and MySQL installed * Anaconda environment * Google chrome * Git 2.20.1.windows.1 for backup and version control * GitHub account * Access to official scrapy documentation and tutorial * Access to MySQL documentation. |
| Testing |  |
| Evaluation |  |

### Estimate of timings

# Design

## Top level

## Data structures.

|  |
| --- |
| UI |
| +sectionName  -contents  -prompt  -commands |
| +UI  +setContents(contentText)  +setCommands(prompt, \*\*kwargs)  +ShowUI(acceptCommands) |

|  |
| --- |
| NoodleMap |
| +edges  -Matrix |
| + NoodleMap()  - add\_edge()  + dijkstra(startnode, endnode)  + returnMap(sort) |

# Note to self

Show inputs etc on wireframe.