Team Name:	Hercules
Team Members:	Ian Dalrymple (dalrympi) Adeline Harcourt (harcoura) Nigel Robbins (robbinni)
Deliverable:	Project Plan
Due Date:	10/02/2017

Introduction:

The Hercules group will be creating a graphical web crawler that consists of a front-end site for user input and a back-end crawler running on a server. Our project will be based primarily off of JavaScript frameworks and libraries, which our group members have past experience with. Any deviations to this plan will be documented in the final report and associated presentation of the project.

Description (User's Perspective):

This project allows the end user to visualize the "World Wide Web" as a literal web of connected web pages. Their entrypoint will be a single web page with a simple form which accepts a URL, a search mode, search count, and optional keyword. When the form is submitted, they will see an interactive web of the sites connected to their provided URL. If the keyword is entered and encountered during the search the crawl will halt and the results immediately sent back to the client. Further, the user can see previous searches in the form's autocomplete facilitated by the use of persistent cookies. If we have time we will also allow them to link searches together so that larger and larger webs can be viewed. The graphic below has been included to illustrate a possible front end for the system. If the user clicks on any one of the nodes in the tree a new tab will open in the browser for that URL.

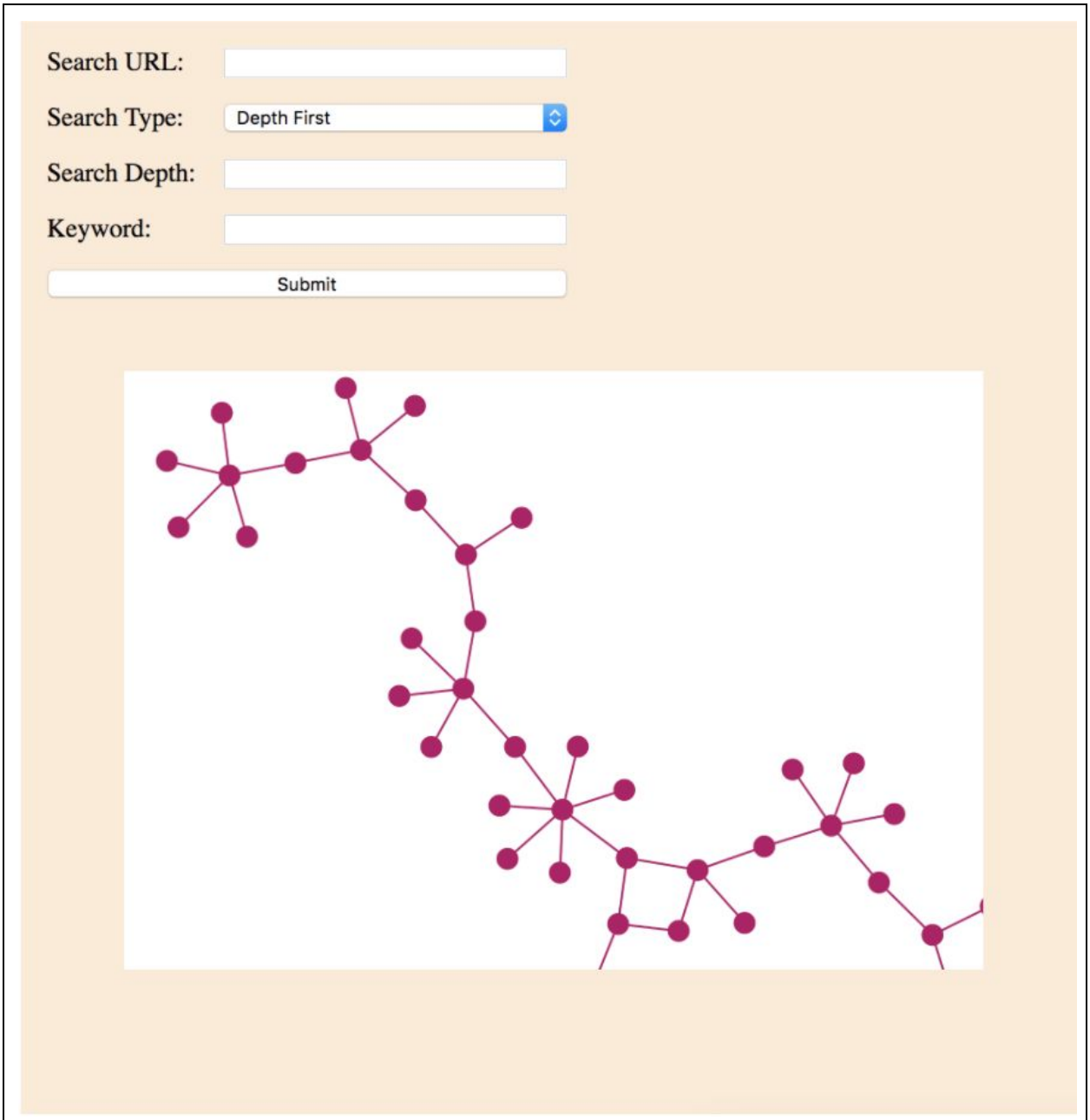

Figure 1: User Interface w/ Resulting Graphic

High Level Software Description:

An Express server will serve the single web page which the user interacts with. When a user starts a client (by going to the web page) a persistent cookie is generated (first time) or otherwise extended. When the client requests a crawl, the Express server will start a crawler that uses Cheerio. As the crawler crawls the web, it stores links in a database or structured flat file. When the crawl is complete, the server turns the database / flat file records into a JSON tree and sends it back to the client. The JSON tree is then used to create a graphic for the user. The figure below has been included to illustrate this architecture.

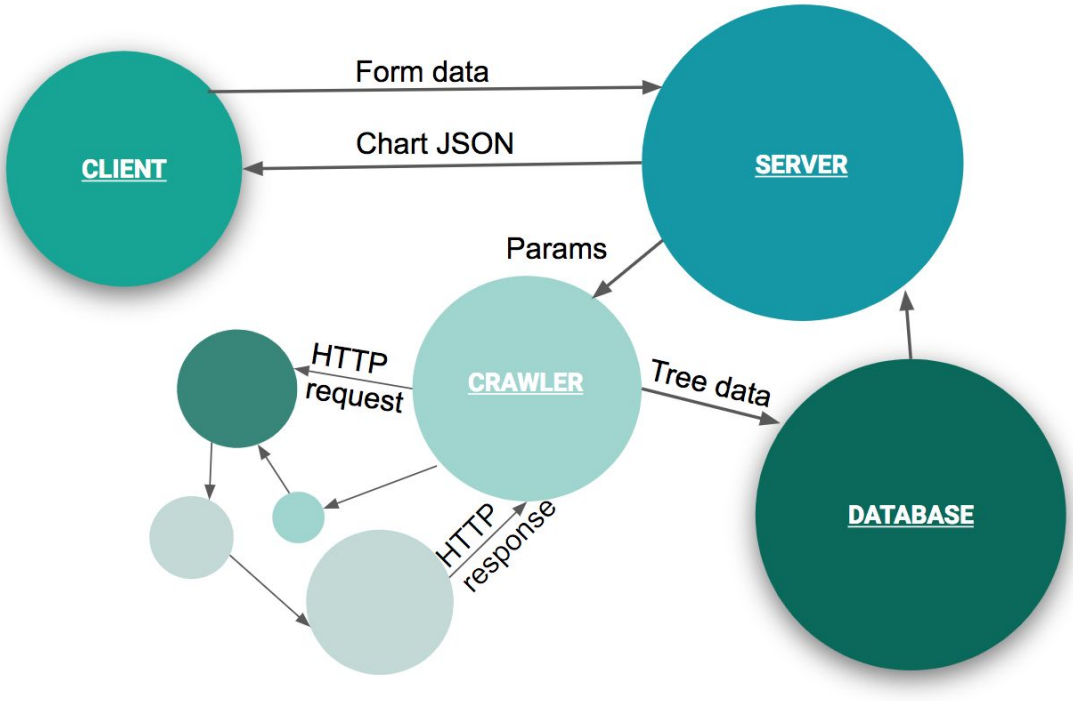

Figure 2: High Level Architecture

System I/O for Common Use Cases:

The table below has been included to provide additional detail for various use cases of the system. For the cases where a cookie already exists the form input fields will somehow be prepopulated with the respective values from the cookie.

URL	Type	Depth	Keyword	Cookie	Output
Valid	BFS	1	Null	Never visited	1) Client check to ensure depth equals 1 or 2 since BFS has been chosen. 2) Form data rolled up and sent to server with POST. 3) Cookie will be generated on the server and sent back to client. The cookie will contain name of crawler, expiration of x days and attribute key value pairs for URL, type, depth, and keyword. 4) The server will begin the crawling process saving connected sites to a database / flat file. 5) When the crawl is complete the results will be packaged in JSON format and sent back to client. 6) Client browser presents results to user. Each node is clickable resulting in new window with the web page opened.
Valid	DFS	1	Null	Never visited	Same as above but client validate depth between 1 and X. The upper limit will be determined after some testing has taken place.
Valid	BFS	3	Null	Never visited	Client validation of depth fails. Message posted on UI indicating issue. All input fields are left as is.
Invalid	DFS	1	Null	Never visited	1) Client check to ensure depth equals 1 or 2 since BFS has been chosen. 2) Cookie will be generated on the server and sent back to client. The cookie will contain name of crawler, expiration of x days and attribute key value pairs for URL, type, depth, and keyword. 3) Server validation of URL fails and error code sent back to client. 4) Client posts the results on message bar NOT alert popup.
Valid	BFS	1	Null	Visited	1) Client check to ensure depth equals 1 or 2 since BFS has been chosen. 2) Form data rolled up and sent to server with POST. 3) Cookie updated with new input from form and expiration bumped out x days. 4) The server will begin the crawling process saving connected sites to a database / flat file. 5) When the crawl is complete the results will be packaged in JSON format and sent back to client. 6) Client browser presents results to user. Each node is clickable resulting in new window with the web page opened.

Valid	BFS	1	Null	Cookie Expired	1) Client check to ensure depth equals 1 or 2 since BFS has been chosen. 2) Form data rolled up and sent to server with POST. 3) Cookie expiration bumped out x days. 4) The server will begin the crawling process saving connected sites to a database / flat file. 5) When the crawl is complete the results will be packaged in JSON format and sent back to client. 6) Client browser presents results to user. Each node is clickable resulting in new window with the web page opened.
Valid	BFS	1	String with length > 0	Visited	1) Client check to ensure depth equals 1 or 2 since BFS has been chosen. 2) Form data rolled up and sent to server with POST. 3) Cookie updated with new input from form and expiration bumped out x days. 4) The server will begin the crawling process saving connected sites to a database / flat file. 5) Crawler encounters the keyword and immediately stops crawl. 6) Results packaged will be packaged in JSON format and sent back to client with indication the keyword has been reached. 7) Client browser presents results to user. Each node is clickable resulting in new window with the web page opened. Indication that keyword encountered included.

Table 1: Use Case Inputs and Outputs

Leveraged Technologies:

The table below outlines the software and hardware technologies being used for the project.

Technology	**Purpose**
GitHub	Version control
Cytoscape	User interface graphical construction
Node.js	Web server and crawling
Express.js, Request.js, Cheerio.js	Web server and crawling
JSON	Data format for transmission of tree from server to client
MySQL or Flat Structured Files	Backend tree caching database or potentially flat structured files containing the tree structure
OSU FLIP(s)	Web and DB hosting
Web Browser (Chrome and FireFox)	User will access the crawler through any browser. Testing will be performed against the two listed and if time permits Internet Explorer will be included as well.
Table 2: Leveraged Technologies	

Responsibilities w/ Work Time:

The tables below outline the weekly work to be completed by each team member.

Ian Dalrymple	
Task	**Time Estimate (hrs)**
Week 3 - Research DB or flat file storage of trees	10
Week 4 - Implement schema Begin investigating transmission of JSON tree results	15
Week 5 - Implement client transfer of URL and keyword to server	15
Week 6 - Implement JSON result transmission (Part 1) Midpoint check	15
Week 7 - Implement JSON result transmission (Part 2)	15
Week 8 - Begin development of testing protocols	15
Week 9 - Implement testing Assist Adeline	15
Week 10 - Assist Adeline Final Report	10
Total Time	**110**
Table 3: Ian Dalrymple Task List	

Adeline Harcourt	
Task	**Time Estimate (hrs)**
Week 3 - Create express web framework skeleton Create HTTP route Create error routing	15
Week 4 - Implement breadth first search crawl	12
Week 5 - Implement depth first search crawl	12
Week 6 - Implement returning JSON to web page Midpoint check	12
Week 7 - Implement keyword halt	10
Week 8 - Implement crawl caching in mySQL database or simple writing to flat file	15
Week 9 - Add measures to crawl dynamically loaded web content Finish other week's tasks as needed	13
Week 10 - Implement streaming JSON back to web page Test and tweak program Finish other week's tasks as needed Final Report	13
Total Time	**102**
Table 4: Adeline Harcourt Task List	

Nigel Robbins	
Task	**Time Estimate (hrs)**
Week 3 - Research creating Cytoscape charts with dummy data Progress Video	15
Week 4 - Add the main UI route to the express server Progress Video	12
Week 5 - Develop integration and unit tests Progress Video	15
Week 6 - Enhance UX (chart legend, filters, colors, etc.) Midpoint check	15
Week 7 - Add records of previous searches in Cookies Progress Video	10
Week 8 - Begin work on stretch goal to combine searches Progress Video	14
Week 9 - Finish work on stretch goal Progress Video	10
Week 10 - Test stretch goal and assist Adeline Final Report	14
Total Time	**105**
Table 5: Nigel Robbins Task List	

Conclusion:

The Hercules team plans to create a graphical web crawler that is intuitive, efficient, and easy to use with elegantly-displayed results using JavaScript libraries Cheerio.js and Cytoscape.js. The project will take approximately 300 hours to complete and will be finished on schedule. If time permits addition functionality will be added as noted in the body of the report.