Ryan O’Connor

Computer Science Project

Dr. Cynthia Howard

February 8, 2015

Requirement Specification

Scope

This software system will provide mobile device users with an environment where they can manually aggregate content from user specified web pages into customizable feeds. The interface will be designed so that a user can specify content of interest from an in-app web browser by touch.

The client application will require an internet connection to connect to both the internet and a web server. It will feature an in-app web browser, a list style interface to organize and display links, and a user control panel to modify settings and feeds. The web crawling to accumulate content will be done on a web server that will store all user data and crawled content in a database.

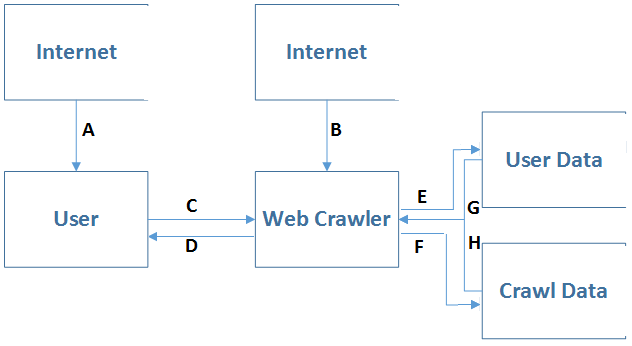
Some use cases include aggregating links from multiple news sources into one feed for news articles, forum viewing, combining job search results from multiple job sites, or creating a feed for up-to-date posts from some favorite blogs/bloggers. If a user so chooses, they can also receive notifications on whatever content they specify when something changes or something new is found.

System Environment



The software system will consist of two segments, the front-end mobile application and the back-end web server. To provide fast computation and save mobile device battery, the heavy computation and consistent web requests will be done on the web server. The server-side will consist of at least two machines: the web server will consist of either one or many nodes that act as workers and communicate directly with the database server. The mobile application will communicate only with the web server where incoming traffic and data will be handled and web crawling will be executed.

**Data Flow**



User Creation:

C. Username / Password

E. Username / Password

Add URL:

A. HTML

C. Username, URL, HTML tags, Keywords

E. Username, URL, HTML tags, Keywords

B. HTML

F. URL, HTML

D. Links

Crawl Scheduled Link:

H. URL, HTML

B. HTML

F. URL, HTML

G. Username, URL, HTML tags, Keywords

D. Links

**Functional Requirements**

ID = Identification

DESC = description

DEP = dependencies

ID: FR1

TITLE: Download mobile application

DESC: The user should be able to download the mobile application from an application store for free.

DEP: None

ID: FR2

TITLE: Notify user of updates

DESC: The user should be notified when updates to the application are pushed. They should also be able to choose whether or not the update should occur.

DEP: FR1

ID: FR3

TITLE: User registration

DESC: The user needs to be able to register a new account with the application by providing a username and password. This will save the users settings and their data so that it can be used across multiple devices.

DEP: FR1

ID: FR4

TITLE: User log-in

DESC: Log in capability must exist so that a user can access their account data from any device running the application.

DEP: FR1, FR3

ID: FR5

TITLE: Change Forgotten Password

DESC: If a user forgets their password she must be able to change it by email authentication.

DEP: FR1

ID: FR6

TITLE: Create Feed

DESC: While the user is logged in, the user can create a feed that will contain aggregated content from URL’s specified by the user. These should be found in a user control panel section of the UI.

DEP: FR4

ID: FR7

TITLE: Provide URL - Paste

DESC: While the user is logged in, the user will be prompted to add a URL and to specify the content in which they would like to receive updates on by loading the web page in a browser. Keywords of interest can also be entered at the end of this process, as well as update frequency.

DEP: FR4

ID: FR8

TITLE: Provide URL - Share

DESC: While the user is logged in, he can add a URL via the “Share…” option in their stock mobile browser.

DEP: FR4

ID: FR9

TITLE: Manual Request

DESC: While viewing a feed, pulling down on a list of links immediately updates the list in a manual fashion as opposed to waiting for the next update scheduled by the server.

DEP: FR4, FR7/FR8

**Performance Requirements**

ID: PR1

TITLE: Quick response time

DESC: The time between requesting an update and receiving one must have little latency.

ID: PR2

TITLE: Intuitive In-app Web Browser

DESC: The web browser must have an intuitive UI that is good at interpreting then visually highlighting the HTML element that verifies the content tapped on by the user.

ID: PR3

TITLE: High error tolerance

DESC: The web crawler must be able to notice garbage input and relay the proper error message to the user. It must also be able to detect when a page has been recoded with new attribute values for the sought after HTML elements.

ID: PR4

TITLE: Proper web request’s

DESC: The web crawler must be able to respond correctly to a servers’ status code. It must also look for “robots.txt” documents and other indicators that the site owners don’t want it to be crawled.

ID: PR5

TITLE: High scalability

DESC: Scale must be kept in mind to handle many concurrent users, so utilizing a load balancer that can add more worker nodes must be accounted for in the design. Concurrent requests to the database must be possible as well.

ID: PR6

TITLE: Dependability

DESC: Communication between the application and server must be highly dependable and fault tolerant.

ID: PR7

TITLE: Extendibility

DESC: The source code must be written so that features can be easily implemented without having to modify other functions very much.

**Entity and Attributes**

Each user has one password and many URL’s. Each URL entered has “0 to many” keywords, “1 to many” tags, and “1” frequency option that specifies how often updates should be scheduled for.

“Tags” example: <a class=”stuff” href=”google.com”></a>

“Frequency” example: 15 minutes



Each unique URL will have its’ HTML data stored with a timestamp specifying when the data was collected. This portion of the database acts as a work queue to ensure that each link is checked consistently and compared with its’ previous results to check for changes.



