Pablo Arvizu, Ramit Malhotra, Jenny Lo
INFO 253 - Final Group Project Technical Report
12/12/2013

INFO 253 - Lui.gi Technical Report

Introduction:

- Our team envisioned a fun-to-use URL shortener designed with a Nintendo Mario theme. The core functionality of this application is to support users by allowing them to create a shorter URL but directed to the required page. Upon selecting the long URL that the user is interested in shortening, the user can select one of the two options: create their own URL shortener or have our system auto-generate a shortened URL to the user.
- In addition to serving our user with the main functionality, our team wanted to extend this function by creating 3 additional features:
    - Mobile responsive URL shortener functionality
    - Fun-feature: Our team created a fun Meme generator (or called MIMS generator) for I-School students to send/share Memes through our shorten URL functionality
    - Browser extension for quick URL shortening functionality

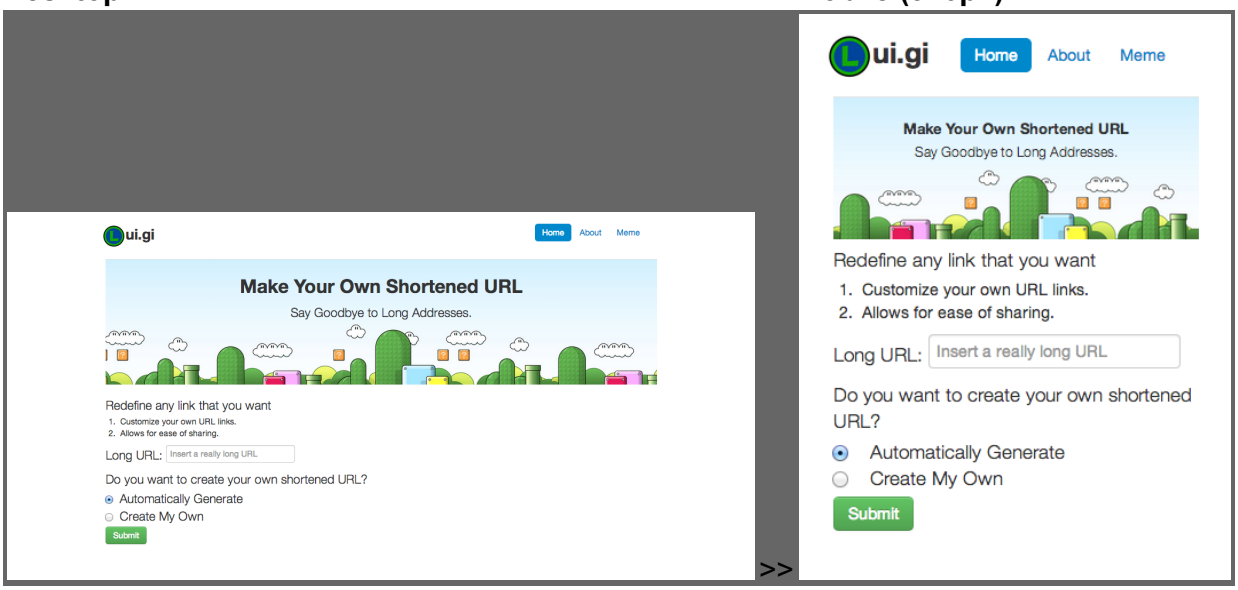
1. Mobile Responsive:

Problem:

- Users should not get penalized for using a smaller device and in today's world, there are more mobile users than ever before. In order cater to users with mobile devices, there is a greater need to make a website flexible, fluid and adaptive.
- Making websites responsive is about making layouts adaptive to viewport sizes.[1] Adaptive layout allows the webpage to respond automatically to users' preferences to view the site on a mobile device. This brings us closer to meeting users' needs.

Example:

Desktop **Mobile (320px)**



[1] http://www.creativebloq.com/responsive-web-design/build-basic-responsive-site-css-1132756

Technology:

- To create mobile responsive screens, we had to make appropriate HTML and CSS edits, add additional mobile responsive CSS stylesheets, re-design pages in HTML and edit CSS selectors to better serve mobile users.
- The core development of mobile responsive screens include the following:
    - HTML meta tag: <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
    - The width property allows us to control the size of the viewport.
    - Device screen dimensions and its pixel ratio: @media screen and (max-width: 480px)
    - The initial-scale property is set to control the zoom level.

Challenges:

- Identifying and prioritizing the appropriate screen sizes for select devices was a complicated process. To cater CSS stylesheets for each and every brand and model of mobile devices would not logical, therefore, our team decided to use the general screen dimensions of mobile devices (320px.)
- Some team members are beginners to CSS and rendering of stylesheets on mobile devices was a steep learning curve for some, especially when multiple stylesheets cascade and cause confusion.
- Huge images are hard to optimize for mobile responsive web design. On our *about.html* page, our team had to redesign the large team image into 3 parts to best serve for mobile device users.

Future:

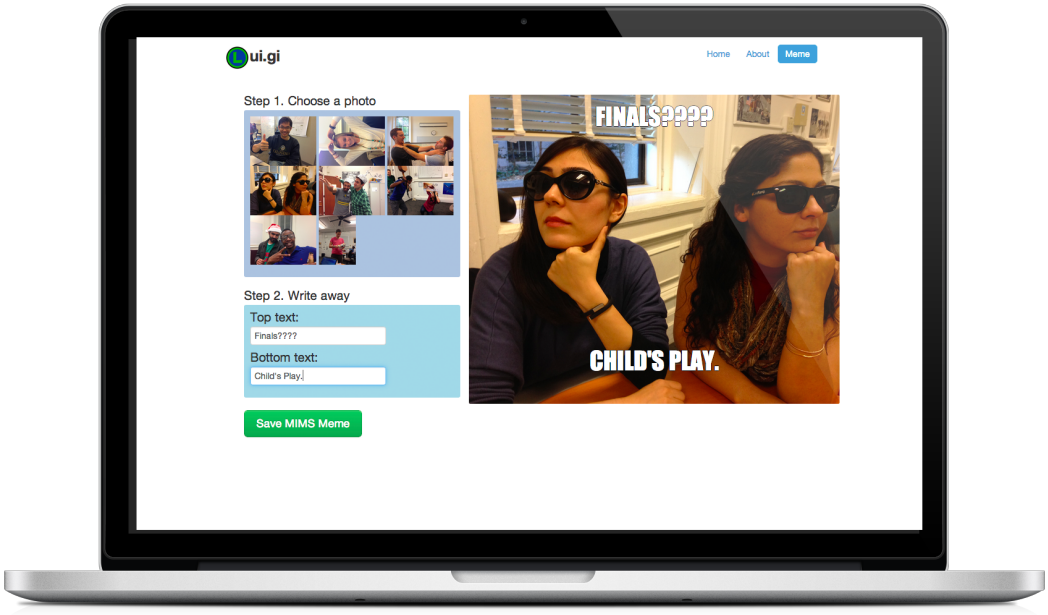
- CSS stylesheets can be better optimized for browsers to render. The mobile responsive stylesheets can be developed and catered toward devices of other sizes.
- The browser extension and meme generator cannot be mobile ready because these functionalities are not catered towards non-mobile users. However, in the future, these functionalities should all be included so that mobile users can take part in this.

2. Meme (MIMS) Generator

Problem:

- Keeping in mind that our URL shortener is to create a fun-to use URL shortener, our team wanted to keep within that theme by making a Meme generator with pictures of I-School's MIMS students. The intent is for the user to create Memes and using our URL shortener, the user can share the meme with friends. This is a side feature to encourage students to utilize our URL shortener.

Example:



Technology:

- The bulk of the development work was done in Javascript. It created the results from the javascript and passed the information to the server on Flask.
- The back-end of the generator was done with python and using Flask that received the input from the user to the database, saving the image selected, the text at the top and the text at the bottom.
- To display the input from the user we created an html template that renders the information from a saved meme.

Challenges:

- The meme generator had to produce an immediate display of the user's input. In addition, the text had to fit appropriately to the meme picture. This was technically challenging because there can be a lot of variations in terms of user's input which might not fit appropriately on to our meme.
- This problem was solved by researching online about jQuery for the appropriate event function for the button to use.
- After generating the meme, the meme content or image has to be saved for later retrieval. The team had to find a way for users to save the meme and share it with friends at a later date.
- As a solution, our team decided to keep the content of the meme in the meme database attached with an id. This id is used to render the image of the meme.
- On the other hand, the url of the rendered meme is then stored on the shorts database linked with the absolute path of the short url.
- The meme id is saved on the memes database which is used to render the meme template when referenced. This reference is saved in the short url database.

- The styling of the text and caption was another challenge as we had to make sure that the meme represented how a meme should look like.
- On the social side, our team had some debate about the idea. While the idea of the Meme generator was meant for fun and for our student community, we weren't sure whether the application might encourage social bullying. However, after some consultation with a lot of students and our TA's comments, we decided to go forward with this idea because similar tools were created in the past and the results were mostly positive rather than negative.

Alternatives:

- Instead of a meme generator, our team considered user login features or link analytics features. However, after some assessment, our team felt that we should not incorporate analytics because this information seems to only be valuable when there are a lot of data for analysis purposes.

Future:

- Future developments for the meme generator include advanced photo and meme editing features, such as: photo filtering and editing functionality. The generator should provide more flexibility to the user, such as user's ability to upload photos, multiple photo grids and incorporating more social sharing functionalities.
- In terms of styling, the meme generator can be made to become mobile responsive as well.

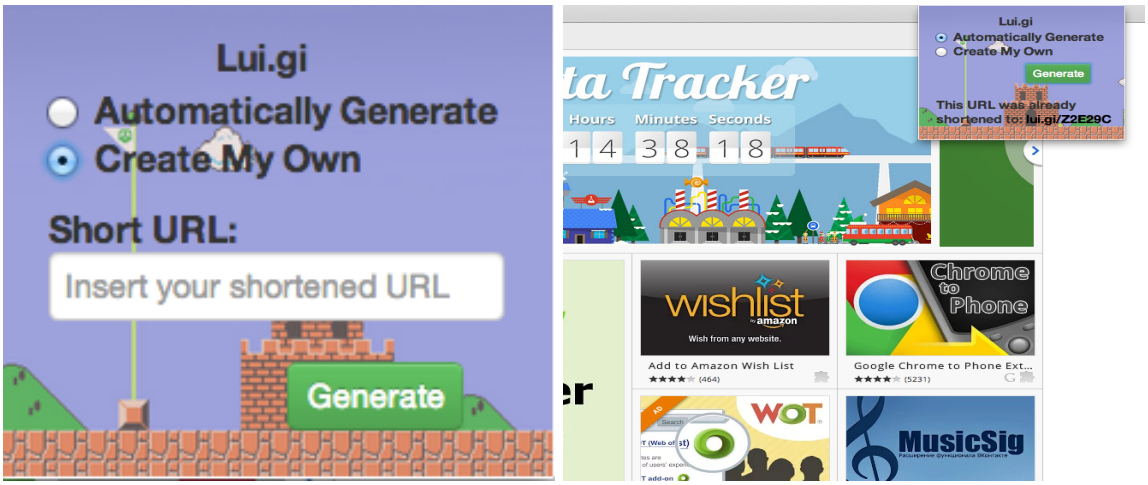
3. Chrome Extension:

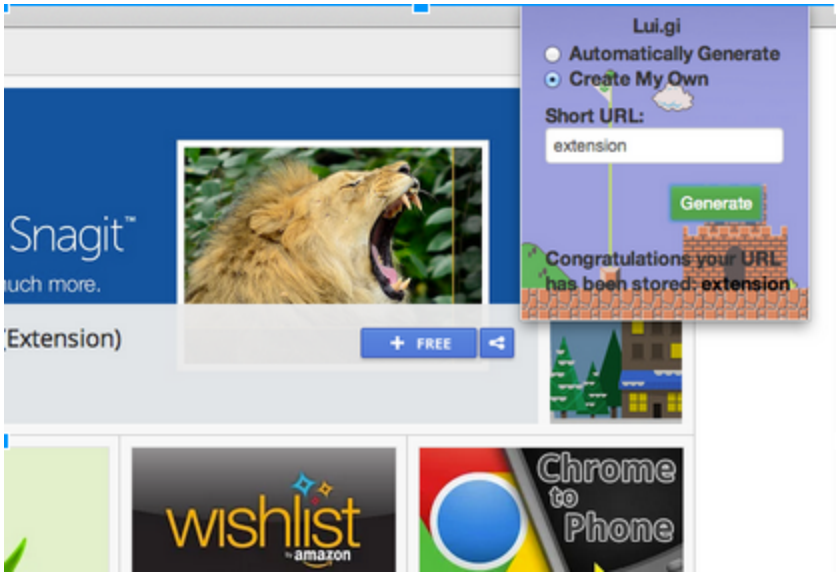
Problem:

- A user has to go to our web page with an existing long URL copied to the clipboard. If the user does not want to put effort into going to the site and pasting the long URL, then the user may never use our service. To cater to such users, we decided to create a Chrome Browser extension so that users can shorten their URLs on the fly if they find valuable URLs of their interest. The location is a prime spot for user's to immediately gain access and shorten URLs.

Example:

(Close up view of the Chrome Extension)





Technology:

- This feature was completed by using the Chrome API. The Chrome API gives developers the ability to create ‘browser actions’ or ‘extensions’. This was used to place a clickable button on the top right of the Chrome browser.
- The basic idea is to create a ‘manifest.json’ file which communicates with the Chrome API, allowing the javascript to use chrome events.
- Jquery, Javascript, HTML and CSS was used to make this extension.

Challenges:

- The major challenge was to understand what ‘permissions’ are needed to enable the extension to communicate outside of its context.

Future:

- Future implementations of the extension could involve incorporation of instant sharing features, and a ‘copy’ button to enable one to instantly get the shortened url on the clipboard for pasting.