INFO 253 • Fall 2013

*Final Project Technical Report*

## *Petite URL Shortener*

## *Group Members*

Kiki Liu

Shaohan Chen

Sophia Lay

Rahul Verma

## 

**Introduction**Petite is a website provides with URL shortener service. Built on the main service, we add on six features: short URL automated generation, social network sharing, history tracking, personal annotation, searching existing petite URL, responsive mobile experience. In this project, we apply HTML, CSS and Bootstrap as the basis to build the layout. We also learn python FLASK shelves, Javascript/JQuery, Ajax to enable dynamic page. We practiced github to cooperate.

**Short URL Automated Generation**

* **The problem it solves**This feature provides user with automated alias with length of 7 digits. Before having this feature, users have to name their own alias. If the entry is without alias, the website will prompt users to enter alias.
* **An example walk through by a user**User will first paste the long url in the first box, and then they can either fill in their own alias or skip this part. If the user does not enter his own alias, after clicking on the submit button, the server will send back a short url with an automated alias. Once a particular automated alias is paired with the long url, everytime the same long url, the user will get the same alias.
* **The technology details for how we solved it**

Random generate 7 bytes with python library

* **Problems we overcame**

We want to randomly generate URL alias only when user doesn’t specify the alias or the alias user defines has been taken for other URL. We took some time for the logic.

* **Alternatives we considered with trade-offs**

We considered using MD5 or other algorithm to generate URL alias, but the computing itself is less efficient than random algorithm.

* **Ideas for improving the feature in the future**  
  We may want to manage the URL alias for major or frequently-used websites like Google, Facebook.

**Social Network Sharing**

* **The problem it solves**The social network sharing feature provides users with an easy way to share the short url with their followers on Twitter and Facebook.
* **An example walk through by a user**1. The user will first submit the long url.  
  2. The server will reply the user with short url. Once the short url shows up, it will   
   be showed up with the Facebook and Twitter sharing button.   
  3. If the user has logged in their social platforms already, once the user clicks on   
   the button, it will pop out another small window with short url and hashtag.   
  4. The user can edit the message to add more comment. After editing the   
   comment, the user can broadcast the short url on Twitter and Facebook.
* **The technology details for how we solved it**

We studied different API in Facebook and Twitter and chose the best one to fit our scenario. There is not much technology we tackle excepting understanding how the complex JS and html code renders from Facebook and Twitter, especially the problem from Facebook API talking below:

* **Problems we overcame**When we tested the API on our local machine, the facebook button would automatically disappear after 45 secs, we spent some time to figure out it is actually the default setting of the API. And this problem does not show up when we test the running on server.
* **Alternatives we considered with trade-offs**Our designer would like to unify the icons from different social network services. Then we realized if we adopted that approach, the user would have to jump out to another page. Therefore we didn’t modify the icon and instead keep the users in our service.
* **Ideas for improving the feature in the future**1.Provide with more social platform options for user to select. In case users   
   might not want to have too many share buttons, the system should allow user   
   to customize which social platform to show up in setting.  
    
  2.Design a better share button for different social platforms to make it look more   
   consistent in style.

**History Tracking**

* **The problem it solves**The tracking history feature will remind user the urls they recently shortened, providing the users an easy way to review the urls they would want to browse again.
* **An example walk through by a user**The history will be tracked by user’s cookie, so the user does not need to login to get the recent history. The user will get the ten most recent shortened URLs with the header of URL and the annotations. If this is the first time for user to shorten URL, the website will reply to users that there is currently no history records.
* **The technology details for how we solved it**

We use cookies to record user’s id and store the information in our server. Since   
 user generated the second url, his/her previous URL will be shown in the   
 history section at the bottom of the page along with the notes, timestamp, web   
 page title and the short URL.

* **Problems we overcame**History section looks similar with the search section, but it turns out that we have to render the page by different approaches. Using FLASK to render html tag and variables is completely new to us. We spent some time figuring out how to escape the tag with “|safe”.
* **Alternatives we considered with trade-offs**We was thinking to implement user login feature to track history. However, we later realized that using cookie can actually do history tracking and users don’t have to login each time to see his/her url history. User login system might be too much in this case since it might increase the network loading as well as complexity of operation.
* **Ideas for improving the feature in the future**Since history will indicate one’s preference, we can suggest from the other users with close history.

**Personal Annotations**

* **The problem it solves**In the previous version, user can only have the short URL without additional descriptive information. In this version, we provide an option for user to annotate about the URL. With this feature, users can record the his own comments and have more understandings when browsing his own recent history.
* **An example walk through by a user**1. Enter annotation while submitting the long url.  
  2. After the long url is submitted, our system returns the shorten url as well as   
   storing the original url and annotation into our database  
  3. User can then search the websites he or she has created shorten urls with   
   annotation by using the search box on the top-right of the page. The matched   
   result will be shown on the history url box on below.
* **The technology details for how we solved it**We add one more text entry in the HTML form for user to enter the annotation. And then we use JQuery to store the annotation into the database.
* **Problems we overcame**

The layout design is very challenging in term of information display. Our designer took some effort to enable a neat and friendly interface.

* **Alternatives we considered with trade-offs**We came up with the idea of providing users to annotate their own URL during brainstorming the new features, we came up with several different forms of annotation, including tag, category, and unlimited text. We decided to adopt the form of text since we think it makes more sense for users to comprehensively record whatever he/she likes to put in, instead of having default category. Also, with the new feature of searching, the text will provide more data for searching function to interact with.
* **Ideas for improving the feature in the future**1. Allow the annotations to be opened to other users.

2. Recommend similar websites based on annotations of other users.

**Searching Existing Petite URL**

* **The problem it solves**In the event that a user has entered a lot of url’s to shorten and would like to find out how many of them were specifically for a particular activity, like ‘coupon’ or ‘shopping’, a search feature is very useful in saving time and increasing user satisfaction.
* **An example walk through by a user**Suppose a user would like to search for multiple words in the comments section of a shortened url. By just entering the words that might have been entered in the comments section while creating the short url, a list of links with that comment show up.
* **The technology details for how we solved it**

Having a means of searching multiple words was important in order to not constrain the user with only one-word queries. We have created regex expression dynamically to match any number of words in the comments, one by one. We have used a jquery/Ajax call in order to prevent the natural behaviour of the form which is to POST data directly to flask. Instead we use AJAX to control how data is sent and how received information is manipulated. We use jquery to pass information to the flask server and render the json response object as an HTML addition to the existing DOM.

* **Problems we overcame**

The issue with dynamically adding the ZeroClipboard copy button functionality to newly added search results was particularly challenging. We used jquery to add <li> elements to the DOM and then again glue the ZeroClipboard function to all the new list buttons. Apart from that, sending data to flask server using POST and then receiving data from flask using json took a bit of trial and error. I would like to advise myself is to go through the documentation of flask and ZeroClipboard a bit more thoroughly so that there is less amount of time wasted in trying stuff that will hit a dead end.

* **Alternatives you considered with trade-offs**

Instead of using shelve db , we were thinking of using php and mysql database to store the history which would help us reuse some part of code that we had developed for I257 Database management course. But that would unnecessarily increase the size of the static files and also introduce complexity that was not necessary. We used Shelve to store the cookie id and related url information

* **Ideas for improving the feature in the future**

Instead of using cookie id for identifying users, it would be great to have a userid which is independent of session time or client-side conditions. It would mean that the history is truly maintained for a user and not just a session.

**Responsive Mobile Experience**

* **Feature description**  
  The responsive mobile design feature provides users a more comfortable environment to browse and use our service on mobile devices.
* **The problem it solves**Before implementing responsive mobile design, the way users see the website on mobile devices is the same as on Desktop. As a result, to type in each form, users have to zoom in/out and slide back and forth to complete each task due to small screen. By adopting responsive mobile design, users are able to start typing all the required information once when they get to the landing page. No more annoying zoom-in and sliding!
* **An example walk through by a user**  
  1. User accesses to our landing page through mobile web browser.  
  2. User types url they want to get shorten.  
  3. A box with shortened url appears.  
  4. User can then double tap on the shorten url to copy. Besides, user can also see what shorten urls he/she has created in history url section. User can also double tap on the urls to copy.
* **The technology details for how we solved it**We used viewport meta tag to detect user screen size and assign different stylesheets for presentation. In mobile.css file, we first adjusted the location of logo and search box on the top, then removed copy buttons on both shorten url & history url box and replaced it with text box wrapped url that user can simply double tap to copy.
* **Problems we overcame, and what advice would you give your past-self before implementing the feature?**1. We first tried to test changes each time on mobile device but realized it was hard and efficient. We then used Safari Develop tool to simulate mobile browser and felt that this really works.  
    
  2. In history url section, the original design wrapped all title, url, notes and time stamp in one same box for layout consistency. However, when we moved to mobile, we realized that all the information inside the text box will be copied when user double clicked it. As a result, we modified the layout design and made url to be wrapped in another box and this solution fulfilled the copy feature really well in the end.
* **Alternative you considered with trade-offs**It is hard for user to click small buttons on the mobile web. As a result, we have to remove copy button in the history section (which being used in PC version) and figure out another way to fulfill the copy feature. We first thought about just making each button bigger, but later realized that the layout became really messy that we decided to give up this idea and to seek other solutions.
* **Ideas for improving the feature in the future**Implement responsive design for other devices. (ex: tablets)