Development and Evaluation of User Interface for Multimedia Search Engine

Ningjing Chen
Abstract

Development and Evaluation of User Interface for Multimedia Search Engine

Ningjing Chen

User interfaces play an important role when users browse different websites. This thesis explores the possibility of re-designing the user interface of an existing multi-media search engine, and asks the question how far one might reach only by re-designing the user interface.

This thesis will show you the process of how a clean and rich-functioned website is created. It starts with benchmarking current popular search engines, choosing features which are valuable, and mimicing scenarios on how people use this website. It continues with a draft design of a prototype, and finally the real implementation.

In the end, after reviewing the design process and the website itself, the thesis gives some suggestions on improvements.
Acknowledgements

Thanks to my supervisor Carl for giving me the chance to work on such a creative and challenging project. Thanks to my three supervisors at picsearch Carl, Thomas and Pablo giving me valuable suggestions about creative ideas and technical skills. Thanks to my colleagues Samuel for teaching me python. Thanks to all other colleagues for this and that help!

Thanks for my supervisor Erik at Uppsala university for giving me really important advice when I was lost.
Index

1 Introduction

1.1 Background

1.2 Purpose

1.3 Disposition

1.3.1 Knowing the trend

1.3.2 Standing in the user’s shoes

1.3.3 Choosing features from the current search engines

1.3.4 Writing Scenarios to mimic user’s behavior

1.3.5 Get early feedback

1.3.6 Implement from easy to hard

2 Preparation

2.1 Literature Review

2.1.1 CSS - the Zen of CSS design

2.1.2 Javascript - DOM scripting

2.1.3 Pagination 101

2.2 Benchmark of popular search engines

2.2.1 Layout

2.2.2 Functionality

2.3 Feature choosing

2.3.1 Popular searches (image/video/audio)

2.3.2 Randomn searches (image/video/audio)

2.3.3 Image directory

2.3.4 My collection (We name it Springio Box)

2.3.5 Email a list to Friends

2.3.6 Result filter by Type/Size/Color/etc

2.3.7 Search suggestion

2.3.8 Different-sized thumbnail view

3 Draft Design

3.1 Draft design of Logo

3.2 draft web page design

3.2.1 Scenarios

3.2.2 web page prototype according to scenarios

3.2.3 Evaluation and Quick feedback

4 Implementation

4.1 Basic Layout
4.2 problems encountered and solution

4.2.1 How to display all result image thumbnails squared and resizable? ........................... 34
4.2.2 How to show popup when user hover a thumbnail? ............................................ 40
4.2.3 How to customize the dropdown menu............................................................... 41
4.2.4 How to implement drag and drop ...................................................................... 42
4.2.5 How to implement Instant Edit ............................................................................ 46
4.2.6 How to email SpringioBox to others ................................................................. 47

5 Brush up .................................................................................................................... 52

5.1 Cross Browser Problem Solving ............................................................................. 52

CSS .............................................................................................................................. 52
Javascript ..................................................................................................................... 52
Other ............................................................................................................................ 52

5.2 Search Engine Optimization(SEO)........................................................................ 52

6 Conclusion, Evaluation and Future work ................................................................. 54

6.1 Conclusion .......................................................................................................... 54

6.2 Evaluation and Future Work .............................................................................. 54
1 Introduction

1.1 Background

There has been an increase in the use of image, video and audio search services recently. PicSearch(http://about.picsearch.com/) is a leading provider of global search services within the field of multimedia search. Over the years, it has become a respected service provider and has gained the trust of many of the most high profile web properties worldwide to power their search functionality.

Meanwhile, when it refers to a website, user interface is the only part user could see which can be seen as the borderline of communication between the user and the core of website. The design of the user interface is therefore a crucial part in determining how good the whole website is in the eyes of the user.

As multimedia content gets more common and competition grows, it gets more important to construct a refreshing and user-friendly interface for a multimedia search website. It would be interesting to investigate what is especially important when it comes to search engine interfaces, and also to study how well Picsearch and other major search portals follow these user-friendliness guidelines.
1.2 Purpose

The trend of website these days is to put more and more effort on the client-side/user interface - we call it rich user interface.

The main purpose of this thesis is therefore -- based on the fundamental functionalities picsearch provides, creating a new website to see how much extra value it could add by only redesigning the user interface.

Concerning that young people are still the main group of internet users, we regard them as our target group, and try to give them a refreshing and brand new experience of using multimedia search service.
1.3 Disposition

1.3.1 Knowing the trend
The thesis will firstly lists some related literature as basis to help us catch the current web design trend and also follow the standard way of implementation.

1.3.2 Standing in the user’s shoes
Then generally there are two starting options to carry out the project.
1) Based on the current website, we will think about how to improve page by page – add more functions, remake the style, rearrange the components and so forth.
2) Regardless of the current website, we will start from scratch. Then “move” the components from the current website if needed.
1 is from detail to general, while 2 is the opposite. We pick 2 since it has not so much limitation as 1, while thinking about design, we’d better jump out of the box and brainstorm freely. In addition to that, it also provide us the possibility to think from the user's perspective. From the start of entering this site, what he/she wants to see and what he/she expects to do at different phases.

1.3.3 Choosing features from the current search engines
To get an overview of how much functions search engine could embrace, firstly we will carry an investigation among several most popular search engines – google, msn, ask, yahoo, youtube and etc – to benchmark worth-including features and also based on these features, devise more related and even “further” functions.

1.3.4 Writing Scenarios to mimic user’s behavior
Writing use case describing the actions the user takes to use this site.

1.3.5 Get early feedback
As the requirements are ready, we can start draft design – make prototype to get feedback on the early stage, which is quite efficient compared to getting feedback at the end of this project.

1.3.6 Implement from easy to hard
When it comes to the implementation phase, we start from the overall layout to specific functions. Following this order makes it possible to break task into subtasks, and overcome them one after another.
2 Preparation

2.1 Literature Review

2.1.1 CSS - the Zen of CSS design [1]

For previous web design projects, I used adobe dreamweaver to generate the layout automatically with tables. Nowadays, as web standards and pure CSS layout become more and more common and compulsory, During the process of this project, we will follow the trend - comply with web standard and pure css layout.

The first book found is <the Zen of CSS design>, it’s a book based on examples from the landmark CSS Zen garden site (http://www.csszengarden.com/) which provides a simple html file with semantic names, the goal is to use a CSS style sheet to apply unique styles and graphics to the page, resulting in some stunning visual displays of web design.

I have already picked up basics of CSS from my previous projects, after reading various case studies in this book
1) I finally understand what CSS could really achieve – not only small details like font-size or color, but also the whole layout. Instead of using tables to put fragments together, CSS separate design and content totally which makes designer and developer cooperation easier.
2) Different Browsers have different understanding of render. The author explains in detail about this within some cases.
3) Get lots of inspirations from graphics, typography, margin and so on.

2.1.2 Javascript - DOM scripting [2]

Javascript looks cool to most of the users. It adds spice to static web pages, which makes web pages change magically without reloading.

<DOM Scripting> introduces the DOM from the very start.

“ What is DOM? In short, Document Object Model(DOM) is a way of conceptualizing the contents of a document. We can refer to any element in the document according to the naming conventions.”

In the view of DOM, it’s more accurate to call a document a node tree. In which every object/element is a node. We have element node (<body>, <p>, <ul> and etc.)
**text node** (all the text content)
**attribute node** (title="a" within `<p title="a">` is an attribute)

We can get the node by using `getElementById` or `getElementsByTagName` and its child nodes by using `childNodes`, `firstChild`, `lastChild`; We can set/get Attribute by `setAttribute/getAttribute`; We can create nodes by `createElement`, `createTextNode` and append them into the document by `appendChild`, `insertBefore` and so on.

After reading this book and trying out some examples, I can easily add/edit/remove any object in the web page.

### 2.1.3 Pagination 101 [3]

Since pagination is a must feature for result page, and there are lots of design issues, too. I refer to this article to give me some hints to find the most user-friendly solution.

It gives us suggestions and also several case studies. I found some of the suggestions are enormously meaningful for me:
- Provide large clickable areas;
- Identify the current page;
- Space out page links

For a), we all have experienced some paginations which are so hard to click, you have to "aim" carefully at the extremely tiny target. c) is kind of similar, some page links are even sort of stuck to one another. About b), it’s always make the user feel secure when they know where they are.

To the author, Flickr’s pagination is the best practice among all. It’s true to me, too. That’s how we will proceed.
2.2 Benchmark of popular search engines

To devise features for the new site, we firstly try to get an overview of what functions other popular multimedia search engines have, we choose Picsearch/ Ask/ Yahoo/ Google/ Live/ YouTube/ Flickr/ Photobucket/ Exalead/ Blinkx/ Truveo as samples.

While browsing the sample sites one by one, we write down all the features for each separately, and after all the browsing is done, based on the record, several features in common and several unique features are extracted which are roughly classified into two main aspects -- layout and functionality.

2.2.1 Layout

1 Homepage

Homepage is the first page users will see when entering Springio, we want to deliver information about what this website is really about and what functionalities it provides, here are some candidates we might have.

A ) Skin Change <Ask>

B ) Spotlight/ Top( Featured/ Popular) Videos/ Popular searches <Picsearch/ YouTube/ Photobucket/ Truveo>

B1) Spotlight <Youtube>

B2) Top/Featured Ones

<Youtube>

<Truveo>
B3) Popular searches

<Cpicsearch>

Popular searches - Click on an image to view the search results

<Picsearch>

<Photobucket>

<Truveo>

Most Popular Videos on the Web Today

C). Videos being watched right now (refresh each 5 secs)  < YouTube>

D). Video Wall < Blinkx>
(i.e. a "wall" composed of 5*5 video windows with short length to make the user get a
general idea of content of each. Moreover, this wall could be embedded in blogger,
myspace, facebook or other applications via some code.)

2 Resultpage

On ResultPage, users will see all the thumbnail version of results, we may need the following
components to give them more customization. A user-friendly pagination is also very important.

A) The window resizes well  < Picsearch/ Ask/ Yahoo/ Google/ Live>
(i.e. fix the display automatically without overlapping or constraint width)

B) Change the result display way (e.g. thumbnail/ list/ thumbnail size)

<Youtube>
C ) Change the result sort way (e.g. Most recent/ Relevancy) <YouTube/Exalead/Blinkx>

D ) Hover effects <Picsearch/Ask/Live>

E ) Good Pagination <YouTube/Exalead/Blinkx>

F ) Classified clearly <Truveo>
(i.e. not all the results together as a list, but still classified in detailed as in the home page, 
e.g. Top Ranked, Featured Channel, Featured Category, Featured Tags)

3 Detailedpage

DetailedPage is about a specific image/video/audio, the components on it is sort of similar among different sites, here they are.

A ) View image/ View page

B ) Back to Result

C ) Previous/ Next

<Picsearch>

<Exalead>

<Flickr>
D ) Remove Frame

E ) Searchbar still display

F ) Send to a Friend < Truveo>

2.2.2 Functionality

1 Search Suggestion

A ) Below the input box before the result display <Ask/Yahoo>
(e.g. dog -- dogs/ dog breeds/ dog names/ dog training/ dog health)

B ) Relate search result after the result display <Ask/Live/YouTube>
(e.g. dog -- Chow Chow/ Jack Russell)

C ) Expand search result after the result display <Ask/Yahoo>
(e.g. dog -- horse/ monkey)

D ) Narrow search result after the result display <Picsearch/Ask>
(e.g. dog -- puppy / dog breeds / dog search)

2 MyStuff/ Shortcuts/ Quicklist/ Scratchpad without login

A ) MyStuff <Ask>
( Like a box where user could do all the operation about the images/webpages )

B ) Shortcuts <Exalead>
( All the shortcuts would display at the homepage)

C ) Scratchpad <Live>
( While browsing other search results, it would occupy an area which could be turned on/off)
D ) Quicklist <Youtube>
( Collection of user's favourite videos, and one can choose if remove it from the quicklist after watching it )

E ) Playlist <Blinkx>
( Collection of user's favourite videos, and one can choose if remove it from the quicklist after watching it )

3 Search Options

A ) Image Type

A1 ) Size <Picsearch/Ask/Yahoo/Google/Live/Exalead>
(e.g. VerySmall/Medium/Large/VeryLarge+BuddyIcon/MyDesktopSize/Wallpapers800*600/1024*768/1600*1200)
*Exalead could specify a certain width or height as threshold

A2 ) Color <Picsearch/Ask/Yahoo/Google/Exalead>
(e.g. Color/BlackWhite/Greyscale)

A3 ) FileType <Picsearch/Ask/Yahoo/Google/Live/Exalead>
(e.g. GIF/PNG/BMP/JPG/Image/Animation)

A4 ) Layout <Exalead>
(e.g. Landscape/Portrait)

A5 ) Content <Google/Exalead>
(e.g. Face/News)

B ) VideoType

B1 ) Source <Exalead>
(e.g. youtube.com/dailymotion.com)

B2 ) Length <Ask/Yahoo/Google/Exalead>
(e.g. Short/Medium/Long)

B3 ) FileType <Ask>
(e.g. Flash/WindowsMedia/MPEG/RealMedia/QuickTime/Other)

B4 ) Source <Exalead>
(e.g. youtube.com/dailymotion.com)

C ) Other Search Options
C1) Domain <Picsearch/Ask/Yahoo/Google/Exalead>  
(e.g. only in.edu domain/only search in this domain)

C2) Family friendly/Safe Search <Picsearch/Ask/Yahoo/Google/Live/Exalead>

C3) Current Location <Live>  
(i.e. knowing your location sometimes helps us provide more relevant information.)

4 Search Setting

A) Searching/Interface Language  
B) Display n result per page  
C) Open result in new window

5 Other

A) Image Directory <Picsearch>  
(all the keywords are put into a multilevel catalog, so user could have an image tour according to his interested fields)

B) Mail image/video/list to a Friend <Yahoo>(need yahoo mailbox)  
<Blinkx>(mail the playlist on behalf of the user, and his friends could visit it via encrypted url)

C) Wall from the playlist <Blinkx>  
(put the videos in the playlist randomly to form a wall)

D) Image Label <Google>  
(a game to make users label the images)

E) Explore last 7 days/this month/popular tags <Flickr>  
(get to know what's popular searches)

F) Viewing History <YouTube/Ask>  
(Ask seems to only have a menu without implementing it)
2.3 Feature choosing

Based on the features benchmarked above as well as the existing/potential features Picsearch has already. The first version of feature collection is here:

2.3.1 Popular searches (image/video/audio)
Suggesting keywords for users who stumble upon “Springio” without specific search target.

2.3.2 Randomn searches (image/video/audio)
Randomn images/videos/audios for users who stumble upon “Springio” without specific search target. Similar to “try my luck” at google. For these who wanna follow the “fate guidance”.

2.3.3 Image directory
Help users take a tour in different area via vivid images, kind of educational and inspiring. This feature is already existing at Picsearch’s site, but only shows as a link which is dim. At Springio we will bring it to the front.

2.3.4 My collection (We name it Springio Box)
Save the favorite images/videos/audios for later use/reference.

2.3.5 Email a list to Friends
Combined with Springio Box, user could always email a list to a friend with some message. This makes the image/video/audio sharing easier between friends (Not have to paste every link manually).

2.3.6 Result filter by Type/Size/Color/etc
Enable users to find image/video/audio with certain properties.

2.3.7 Search suggestion
Similar keywords suggest. (e.g. dog->cat)

2.3.8 Different-sized thumbnail view
Enable users to view thumbnails in their preferred way.
3 Draft Design

3.1 Draft design of Logo
As predefined, there are three main aspects of requirements:

1) The user group is Youth

2) The site name is Springio
After looking it up carefully in the dictionary, I find 3 meanings for the closest word springy:
   a) Elastic; rebounds readily
   b) (Of movements) light and confidently active
   c) (Of land) abounding in or having springs of water

3) The color tune is mainly Green/Blue/Grey/White
According to the above three requirements, I make up two alternative drafts -- with two different thoughts and styles, here is their explanations with sort of pros. (one's pro is also the other's con). All the fonts, shapes and effects could be changed later.

a) Serious One

Explanation
A flower-like "creature" jumps with its "hands" up and spring-like body. The blue background is in the shape of a magnifier which indicates "search".

Pros
- More like a brand
- If the feature of share is not so successful, this one could still work
b) Cute One

![Springio Logo](image)

**Explanation**
Two "i"s turns into two smiling spring-bodily "creature"s which indicates this search service has the feature of share. There is also a horizontal spring.

**Pros**
- More Lively, maybe more to the youth's taste
- Feature indicated clearly

After showing these to my company, I got the suggestion that they prefer the firrst one, but
1) the meaning of the logo isn't self-evident,
2) since it's a site for youth, maybe could add more colors,
3) no need to have a slogan.

Then I tried to mix the above two and add color -- green. I suddenly realized that when the user sees the name "springio", the first reaction to it should be the season "spring" which is green and full of hopes. After that, a childish, happy flower was drawn trying to convey this kind of feeling.
3.2 draft web page design

3.2.1 Scenarios
Now we have all the features we want, can we do the draft design/prototype right away? Let's wait a little bit and try to organize the features in an appropriate way – it's like assembling components. We will use the Scenarios method to illustrate user's activities.

Scenario 1 User who stumbles upon Springio

Adela stumbles upon Springio, she doesn't know what to search for. Luckily, on the homepage, she sees some random thumbnail images, within which a universe image is so fascinating. She checks it and loves it a lot, so she shares it with her friend Qin via Email.

Scenario 2 User who get email from her friend

Qin receives an email from Springio, it's said, “Your friend Adela wanna to share this image with you! Click the link below”. Clicking the link leads her to that image's page on Springio.

Scenario 3 User who is familiar with Springio

Yi loves cat, she wants to find some cute cat images. She comes to Springio, searches “cat” and comes to the result page, with lots of cat images thumbnails on it. She uses different filters to find the images she wants and share them with her boy friend Hao.

Scenario 4 Another user stumbles upon Springio

Zhao comes to Springio, the image dictionary catches her eyes. She chooses category Travel, then goes down to Asia, Southeast Asia, China, Shanghai/Beijing. She finds images she wants and then shares them with her friends.
3.2.2 web page prototype according to scenarios

Before the phase of real implementation. We make some web page prototypes showing what it would look like when it's done. So that we can give feedback early before we start any coding, since it's easy to change when it's still in prototype.

According to Scenario 1, now let's follow Adela to the home page (pic-3.1).

Home page pic-3.1

As you can see below, there are popular image searches on the left which provides the most popular search key words last week – clicking any keyword will lead you to do a search. On the right side, there are several random images, clicking any will lead you to that image's preview page (pic-3.2). Adela clicks an image and then comes to
Here you can find a box with more detailed info on top, beside which there is a previous/next image navigation. A frame is at the bottom with web page where that image comes from. Adela clicks the “add” button on the top thumbnail, then a SpringioBox appears on the right. (You will see that box prototype in next Scenario).

Now let’s switch to Scenario 3, firstly Yi comes to result page (pic-3.3). All cat images (Just imagine...). As we can see, the springio logo is blinking at us via clicking which user can always go back to the home page; at the top right, the word “cat” is in the search box with the menu “image” chosen; Below the header “Search Result”, we could see how many results are available and also related suggestion, in the case of “cat”, we have suggestion “dog, tiger”.

Preview page pic-3.2
Result page pic-3.3

As Yi wanna make the thumbnail bigger, there are four squares (three while implementing), the chosen one is blue and the left are grey, when user hovers, it’s baby-blue.

Yi only want colorful and big images, there are three filter up there. The default setting is “All Image Sizes”, “All Image Colors”, “All Image Types”. Then she hovers “All Image Sizes”, a menu drops down, and she chose “big”. Similar action applied to “All Images Colors” and chose “Colorful”.

Although the thumbnail is already big, Yi still wanna see more, so she hover a thumbnail, a popup appears with image’s info on it, also a bigger thumbnail which is of the original height-width ratio.

She clicks the “add to SpringioBox” link on the popup, then the springioBox opens on the right side automatically and with the chosen image in it. Actually she can also click the “Springio Box” button to open it. (pic-3.4)

Then she keeps on adding more images, 5,6,7... ok! It’s done! Now she clicks the “Folder 1” title, it turns into a text field immediately, and she change the name into “cat”. Here she can create a new folder by clicking “New Folder” button and also she can fold any folder if it’s too long, the leaves on the left to the title would become one if it’s folded.
Then when she hovers the folder title line, there is a link called “email it”, she clicks it and redirected to form with “which email you wanna send” and “what do you wanna say to him/her”. She fills the form and then clicks “send”.

Since it’s not so different among image/video/audio, so there we attach video/audio prototype by the way.

The only difference for video part (pic-3.5A & pic-3.5B) and audio part (pic-3.6) is that instead of having filter for color/type/size, it has source filter - from which site those videos come.
Result video pic-3.5A

Result video pic-3.5B
3.2.3 Evaluation and Quick feedback

After I showed these prototypes to my colleagues with related explanation, some feedback is received.

1) **Suggestion** -- Remove the advanced search/family friendly on the searchBox to make it easier;
   **Solution** -- Removed

2) **Suggestion** -- Change the plain text “add to springioBox”/“new folder” into more intuitive graphical icon.
   **Solution** -- Change to +

3) **Suggestion** -- The image popup info part is too much, just combine them into one simple line.
   **Solution** -- Change to

4) **Suggestion** -- The video/audio info should only appear on the popup.
   **Solution** -- Display only the thumbnail with the time length on it, when the user hover, it will show more.
4 Implementation

4.1 Basic Layout

According to <the Zen of CSS>, we use semantic naming conventions. Here are the main ‘div’s shared by all the pages we will have.

```html
<div id="extra">
</div>

<!-- 1.Branding -->
<div id="branding"> ...

<!-- 2.Navigation & Searchbar -->
<div id="nav_searchbar">...

<!-- 3.Content -->
<div id="content">...

<!-- 4.Site information -->
<div id="site_info">

extra is the option for later use, here we use it to style the top stripe.
branding is the logo part.
nav_searchbar is the one with image/video/audio choices and search box.
content is the main wrapper for different content inside.
site_info is the footer.

Now as the content(html) is there, let's deal with the presentation(css style).

First of all, since different browsers have different default css style, so the first thing before the real work is to override them.

Here is our reset.css

/* Normalizes margin, padding */
body, div, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, form, fieldset, input, p, blockquote, th, td
{ margin: 0;
padding: 0; }

/* Normalizes font-size for headers */
h1, h2, h3, h4, h5, h6
Now we can start by styling the unchanging elements.

We want our site to stay in the center. Instead of using an extra wrapper, we apply the style to <html> and <body> elements directly.

```html
html { text-align:center; }
body { position:relative;
    margin: 0 auto;
    width:80%;
    text-align:left; }
```

A proportional width and auto-right and auto-left margins placed in the <body> element will center the design within <html>.

The top stripe

```html
div#extra { background:transparent url(images/topbar.png) top left repeat-x; height:20px; }
```

Here a background image is repeated along the x-axis.
The logo

```css
div#branding { width:379px; height:103px; left:3%; position:relative;}
```

The logo image is put into the html code directly.

Navigation and search bar

```css
div#nav_searchbar { position:relative; top:-99px; height:103px; width:50%; float:right;}
div#navigation { background:transparent url(images/nav.png) top left no-repeat; margin:0px 0px 0px 10px; width:180px; height:40px; font-size:12px; font-weight:bold; text-transform:uppercase;}
div#navigation ul { position:relative; top:7px; left:6px;}
div#navigation ul li { display:inline;}
div#navigation ul li a { padding:2px 4px; margin:2px 4px; color:#ABF850; text-decoration:none;}
div#navigation ul li a.current { background-color:#ABF850; color:#FFFFFF;}
div#navigation ul li a:hover { color:#ABF850; background-color:#E4FDC6;}
div#searchbar { position:relative; top:-6px; left:10px; width:290px; height:25px; background:transparent url(images/greyline.png) top left repeat; padding:20px 5px 5px 10px;}
```

`float:right` makes searchbar on the top right part. `li.current` is set when it's on that section (image/video/audio) which visually display as green background and white text in contrast to the default white background and green text.

When user hover the text, it's still green text, but with light green background which gives them feedback to make them feel safe.

We use 5px*5px grey-line repeated as the background diagonal.

Site info

```css
div#site_info { font-size:11px; color:#CCCCCC; clear:both;}
div#info_menu { background:url(images/lightgrey.png) top left repeat-x; padding-top:5px;}
div#info_menu li { width:200px; display:inline; border-right:#CCCCCC 1px solid; padding:0px 8px 0px 6px;}
div#info_menu li a:link, a:visited { text-decoration:underline; color:#CCCCCC;}
div#info_menu li a:hover { text-decoration:underline; color:#FFFFFF; background-color:#CCCCCC;}
div#copyright { position:relative; top:-14px; float:right; width:100px;}
```

We use `display:inline` to make the items in the list on one line.
4.2 problems encountered and solution

We encountered problems now and then during our implementation process, here are the some typical ones.

4.2.1 How to display all result image thumbnails squared and resizable?

4.2.1 Problem description

We have thumbnails of different width-height ratio, but now as you can see from the prototype, we want them to be squares, with some white padding and grey border around. In addition to that, when user clicks 3 square switchers on the top, the squares’ size could be changed among Small/Medium/Big.

4.2.1 Method

Firstly let’s see what we want to achieve, we want to cut a square from a not-square image as shown on the following picture – this is the first step, then add paddings and borders.

The grey box is a div we call it “innerimg”, which has the property overflow: hidden which will cut all things outside the div off. Then we need firstly center the original thumbnail, how can we do that? The solution is using the property margin, negative means moving up/left, positive means moving down/right.

There are different width-height ratio and they compared with the defined square size – actually there are 5 situations:

1) height >= width and width > size => scale the width to size, get the height according to the ratio, move the position along the y direction to the center
2) height < width and width > size => scale the height to size, get the width according to the ratio, move the position along the x direction to the center
3) height and width both < size => no scale, move the position along both x and y directions to the center
4) height > size > width => no scale, move the position along both x and y directions to the center
5) width > size > height => no scale, move the position along both x and y directions
Here are the corresponding javascript functions

```javascript
function squareIt(img, size) {
    // outerimg part
    img.parentNode.style.width = size + "px";
    img.parentNode.style.height = size + "px";

    // innerimg part
    img.parentNode.parentNode.style.width = size + "px";
    img.parentNode.parentNode.style.height = size + "px";

    // get the oriheight&oriwidth to caculate the ratio
    height = img.getAttribute("oriheight");
    width = img.getAttribute("oriwidth");

    // caculate the thumbnail width and height according to the original ratio
    // NOTE: since the thumbnail width and height couldn't be got directly
    if ((height-width)>=0 && (width-size)>=0)
    {
        width = 128 * width / height;
        height = 128;
    }
    else if ((height-width)<0 && (height-128)>=0)
    {
        height = 128 * height / width;
        width = 128;
    }

    // S1: height >=width and width>=size
    // then scale the width to size
    // get the height according to the ratio
    // move the position along the y direction to the center
    if ((height-width) >=0 && (width-size)>=0)
    {
```

```
img.height = size * height / width;
img.width = size;
var top = (img.height-size) / 2;
img.style.marginTop = "-" + top + "px";
}  
// S2: height<width and width>size  
// then scale the height to size  
// get the width according to the ratio  
// move the position along the x direction to the center  
else if ((height-width)<0 && (height-size)>=0)
{
  img.width = size * width / height;
  img.height = size;
  var left = (img.width-size) / 2;
  img.style.marginLeft = "-" + left + "px";
}  
// S3: height and width both < size  
// then no scale  
// move the position along both x and y directions to the center  
else if ((height-size)<0 && (width-size)<0)
{
  img.height = height;
  img.width = width;
  var top = (size-img.height) / 2;
  var left = (size-img.width) / 2;
  img.style.marginTop = top + "px";
  img.style.marginLeft = left + "px";
}  
// S4: height > size > width  
// then no scale  
// move the position along both x and y directions to the center  
else if ((height-width)>=0)
{
  img.height = height;
  img.width = width;
  var top = (img.height-size) / 2;
  var left = (size-img.width) / 2;
  img.style.marginTop = "-" + top + "px";
  img.style.marginLeft = left + "px";
}  
// S5: width > size > height  
// then no scale  
// move the position along both x and y directions to the center  
else if ((height-width)<0)
The original image's width and height are passed through parameters oriwidth and oriheight. We need that to calculate the ratio and move the position.

All the code above is the way to crop a thumbnail, since we need to crop all of them, here is the real function we call

```javascript
/*
* crop images thumbnail of specific class to square
* @param size : any integer
*    classname : class name of the thumbnail
* @return null
*/
function squareThumb(size, classname) {

    var imgCollection = document.getElementsByTagName("img");

    for (var i=0; i<imgCollection.length; i++) {
        // search for the specific classname
        if (imgCollection[i].className == classname) {
            squareIt(imgCollection[i], size);
        }
    }
}
```

After we have the ready squared innerimg, we wanna apply paddings and borders to it, we can't use padding and border to it directly, since it will lead to add more room to the innerimg which is not our intention.

So we create a new div called “outerimg” wrapping innerimg, and apply padding and border to it. Here are the css style:

```css
div#result_images .outerimg {background-color:#FFFFFF; float:left; border:#CCCCCC solid 1px;
```
Then how can we change the size of the thumbnail? As you may already see, the squareThumb() function passes “size” and “classname” as parameters, we just need to change the “size”, easy.

We add event on each square switch onclick="switchImage('small','medium','big')" to invoke the javascript function:

```javascript
/*
* switch the image thumbnail
* @param size: "small"/"medium"/"big"
* @return null
*/
function switchImage(size) {
    // clear if the current class is chosen
    document.getElementById("small_square").className = ";"
    document.getElementById("medium_square").className = "";
    document.getElementById("big_square").className = "";

    if (size == "small") {
        document.getElementById("small_square").className = "current";
        squareThumb(SI,"thumbnail");
        setCookie('tsize',SI);
    } else if (size == "medium") {
        document.getElementById("medium_square").className = "current";
        squareThumb(MI,"thumbnail");
        setCookie('tsize',MI);
    } else if (size == "big") {
        document.getElementById("big_square").className = "current";
        squareThumb(BI,"thumbnail");
        setCookie('tsize',BI);
    }
}
```

*SI is the cookie constant for small thumbnail, here it is 50, and MI is 65, BI is 80.

### 4.2.1 Evaluation

This way of chopping doesn’t require any back-end server to really operate on the thumbnail or produce square version. It’s totally on the front-end, we use the original
thumbnail, but user only sees its “square version”. It’s the same as the switch size part, the
thumbnail itself doesn’t change, but user’s view changes.
4.2.2 How to show popup when user hover a thumbnail?

4.2.2 Problem description

As you can see from pic-3.3, when the user hover a thumbnail, a popup will appear with the related info on it. There might 2 ways to do it, one is via javascript and the other is via css. Which is better?

4.2.2 Method

1) using javascript to create a new div on the fly
2) using css to show hidden div

There are some problem with 1, where should we store the info? Or we need to use ajax? How to implement ajax in python?

Let’s just make it simple. When we fetch the image, we fetch the info about that image but hide it within a span, and when user hover the thumbnail, the hidden span will appear.

Here is the code, at first the span visibility:hidden, when hover the thumbnail image, visibility:visible. And also we use z-index:100 to make sure it appear on top of all the thumbnail images.

```html
div#result_images ul.images span { position:absolute; margin:-90px 0px 0px -42px; padding:6px 5px 5px 5px; z-index:100; visibility:hidden; background:url(/I/greyline.png) top left repeat; border:1px dotted #CCCCCC; }
div#result_images ul.images li span img{ background-color:#FFFFFF; padding:4px; margin:2px 5px 2px 5px; }
div#result_images ul.images li:hover span { z-index:100; visibility:visible; }
```

4.2.2 Evaluation

Choosing implementation via css means even the user turns off javascript, it can still work. It’s also good for SEO since the info is rendered there. Moreover, as you can see, it’s not much code but work smoothly.
4.2.3 How to customize the dropdown menu

4.2.3 Problem Description

We try use the predefined component <select> to implement, but no matter what kind of css style we apply, it still remains its default appearance.

4.2.3 Method

Since it’s hard to CSS style the default dropdown box, while we want it look exactly like the prototype, we make it manually.

Every filter, take “size” for example, is actually an unordered list. When users hover the selecting box, the unordered list will show for choosing. When users choose an item, it will display visibly above the list.

For the hover thing, we use pure CSS to achieve

```css
div#opt_size ul { display:none; padding:0px 0px 0px 6px; z-index:15;}
div#opt_source:hover ul { display:block;}
```

For the real choosing function, it just reloads the page with another parameter. We will implement this via python later.

4.2.3 Evaluation

Now it looks just like in the prototype. And the code is easy. It tells us, sometimes, if this way doesn’t work, just give it up totally, and find a brand new way.
4.2.4 How to implement drag and drop

4.2.4 Problem Description

We want user not only can click “add to SpringioBox” to add image, but also drag the thumbnail image and drop it to SpringioBox.

4.2.4 Method

Since it may involve a lot of javascript programming, and it’s also a really popular function that there must be lots of ready library online.

We search on the internet for a ready script about drag and drop, and here is one we found http://www.dhtmlgoodies.com/index.html?whichScript=drag-drop-custom We try out the demos and feel it is the one.

It creates a clone object of the applied object, and then move the applied object, in our case, it’s the thumbnail info span. This object is called “source” in this script. Where does that source go? The springioBox, it’s called “target”.

We put the thumbnails into source collection by calling addSource(), and put the folders within springioBox into target collection by calling addTarget().

Before adding the source and target, we need to create the drag-drop-object first. So here is the code we use.

```javascript
var dragDropObj = new DHTMLgoodies_dragDrop();
...
//Add all the span into Source collection
var liCollection = document.getElementById("result_images").firstChild.getElementsByTagName("li");
for (var i=0; i<liCollection.length; i++){
    dragDropObj.addSource(liCollection[i].childNodes[0].childNodes[0].nextSibling.childNodes[0].childNodes[0].id,true,true,true,false,'onDragFunction');
}
...
//Add Folders in SpringioBox into the Target collection
var folders = document.getElementById("springio_boxinner");
for(var i=0; i<folders.childNodes.length; i++){
    dragDropObj.addTarget(folders.childNodes[i].id,'dropItems');
}
dragDropObj.init();
```
Now we have the source and target, but how should they behave when something happens? It’s time to override two core methods:

```javascript
function onDragFunction(cloneId, origId) {
    var obj = document.getElementById(cloneId);
    obj.style.border = '3px solid #cccccc';
    document.getElementById('content').style.cursor = 'move';
    document.getElementById('springio_box').style.cursor = 'move';
}

function dropItems(idOfDraggedItem, targetId, x, y) {
    document.getElementById('springio_box').style.zIndex = 2;
    var objori = document.getElementById(idOfDraggedItem);
    var obj = document.getElementById('box_' + idOfDraggedItem);
    if ((obj == null) || (obj.parentNode.parentNode.id != targetId)) {
        // check if the object is image/video/audio
        if (idOfDraggedItem.substring(idOfDraggedItem.search(/frompage=/i) + 9, idOfDraggedItem.search(/frompage=/i) + 10) == 'i')
            addSpringio("imageInfo?" + idOfDraggedItem, targetId, "springio_boxinner");
        else if ...
        document.getElementById('content').style.cursor = null;
        document.getElementById('springio_box').style.cursor = null;
        objori.style.visibility = "";
        return;
    }
}
```

OnDragFunction is called when users start drag something, DropItems is called when users release the mouse.

When users start to drag something, the cursor will change to move and the border will be thicker to inform the user it’s chosen.

When users release the mouse, we check the type of the dragging object first, since image/video/audio has different ways of rendering. In this case, it’s image, and all the div creation and appending, styling is encapsulated in the function addSpringio.

Here is the code snippet from addSpringio

```javascript
... // Get the values based on positions
While searching on the internet, we find several ready script to implement drag&drop, choosing this because its function is closer to what we expected about springio, and also its
instruction is concise and easy.

However, it still took us not a short time to figure out how to make some small change. The reason is it’s not easy to customize.

If we could implement it for another time, we will choose jQuery library probably.
### 4.2.5 How to implement Instant Edit

#### 4.2.5 Problem Description

Besides the drag and drop, users can create/delete/edit folders in the springioBox. Here comes another requirement, edit the folder's name instantly.

#### 4.2.5 Method

Same as the drag&drop, firstly we search for ready-to-use script, here it is!


Then followed the step by step instructions:

1) Download this Javascript file: InstantEdit 2.0 JS

2) Create a update file that handles the input. For example this PHP: Update File

3) In your page add the javascript: `<script type="text/javascript" src="instantedit.js"></script>`

4) Set fixed vars (like hidden elements in a field post). These will be posted with the editable field so you can identify a user/session.

```
<script type="text/javascript"> setVarsForm("pageID=profileEdit&userID=11");
</script>
```

5) Last step: in your HTML for any editable field add a SPAN arround it:

```
<span id="userName" class="editText">John Doe</span>
```

Instead of using PHP, we will use Python to handle the input. And here is the code

```
<span id="f1" class="editText" style="cursor: pointer;" title="Click to Rename!"> Folder_1 </span>
```

#### 4.2.5 Evaluation

This script is really well made and easy to apply, Still if we could implement it for another time, we would preferably use jQuery library.
4.2.6 How to email SpringioBox to others

4.2.6 Problem Description

We skip the python part since this thesis mainly focuses on front-end development, we don’t care what back-end programming is used.

One of the important features we want to implement is to email springioBox to others. It involves the following steps 1) Store and pass data; 2) How to send the very long url.

It seems ok even if we send the long url, (not look good though). The real problem is when this long url is rendered in the emails which we send to the user’s friends, it automatically adds line break according to different browser size. That means when the link is clicked, it’s not the original long url we sent, which also means it DOESN’T WORK.

4.2.6 Method

Since we don’t want to create “account” stuff to bother users which might refer to database, while we still want to save all the images/videos/audios in the springioBox. Cookie is chosen as it is small and stored at each user’s browser which means no need for database to relate specific user to specific springioBox set.

But how can we store the info about every image/video/audio?

Firstly, cookie is actually a plain map array{(a:b:expire),(a1:b1:expire1)...}, so we need prepare an easy way to access and edit it. Here is the functions we need:

```javascript
function setCookie(c_name, value, expiredays) {
    var exdate = new Date();
    exdate.setDate(exdate.getDate()+expiredays);
    document.cookie=c_name+"="+escape(value)+((expiredays==null) ? "" : ";expires="+exdate.toGMTString());
}

function getCookie(c_name) {
    if (document.cookie.length > 0) {
        c_start = document.cookie.indexOf(c_name + "=");
        if (c_start != -1) {
            c_start = c_start + c_name.length + 1;
            c_end = document.cookie.indexOf(";", c_start);
            if (c_end == -1) c_end = document.cookie.length;
            return unescape( document.cookie.substring(c_start, c_end));
        }
    }
    return null;
}
```
Then take image as an example. We know there might be more than one folders in the springioBox, and there might be more than one images in a folder, each image has its own information - id, size, url and etc.

We use `join()` to create such pattern:

```
folder-num: [FOLDER-NUM]
```
```
folder-names: [FOLDER1-NAME]||[FOLDER1-SIZE]||[FOLDER1-OPEN/CLOSED]###
[FOLDER2-NAME]||[FOLDER2-SIZE]||[FOLDER2-OPEN/CLOSED]###
[FOLDER3-NAME]||[FOLDER1-SIZE]||[FOLDER3-OPEN/CLOSED]### ...
```
```
folder_images_id: [IMAGE1-ID]### [IMAGE2-ID]### ...
```

By using `split()`, it's easy to convert the strings into arrays which are handler.

Once the page is going to redirect, we store the data within springioBox into cookie;
```javascript
function boxStore() {
  var folder_name = new Array();//Record name || length || fold of each folder
  var img_q_array = new Array();
  .......
  var folders = document.getElementById("springio_boxinner");
  var count = 0;
  //Index of images in img_array
  for (var i=0; i<folders.childNodes.length; i++)
  {
    var fname = folders.childNodes[i].id.replace("Folder_","f");
    //Change from Folder_n to fn
    .......
    // 0 for close
    folder_name[i] += 0;
    else
    folder_name[i] += 1;
    for (var j=0; j<imgCollection.length; j++)
    {
      var id = imgCollection[j].getAttribute("href");
      //Extract Infos from the id(url)
      //1.Get positions
      var qs = id.search(/q=/i);
      ........
      // q || start || offset || frompage
      img_q_array[count] = id.substring(qs, starts-1) + "||";
  }
}
```
if (id.substr(frompages+9) == "image")
{
    img_pageurl_array[count] = id.substring(page_urls+9,image_urls-1);
    ....
}
else if (id.substr(frompages+9) == "video")
    ....
else if (id.substr(frompages+9) == "audio")
    ....
    count++;
}
}
setCookie("folder_num",folders.childNodes.length);
......

Once a new page is rendering, we fetch data from cookie and then render them as springioBox.
function boxRead() {
    var folder_num = getCookie("folder_num");
    var folder_name = getCookie("folder_names").split("###");
    if (folder_num == "")
        return;
    var img_q_array = getCookie("folder_images_q").split("###");
    ....
    var count = 0;
    for (var i=0; i<folder_num; i++)
    {
        fn=createFolderlight("springio_boxinner");
        //foldername like "Folder_n"
        fnSpan = document.getElementById(fn.replace("Folder_","f");
        fnSpan.childNodes[0].nodeValue = folder_name[i].split("||")[0];
        //Modify the Foldername displayed
        ....
        for (var j=0; j<folder_name[i].split("||")[1]; j++)
        {
            if (img_q_array[count].split("||")[3] == "image")
            {
                id = "imageInfo?q=" + img_q_array[count].split("||")[0] + "&start=" +
                img_q_array[count].split("||")[1];
            ....
            }
            else if (img_q_array[count].split("||")[3] == "video")
            ....
            else if (img_q_array[count].split("||")[3] == "audio")
            ....
            addSpringiolight(id, fn, "springio_boxinner");
            count++;
        }
        if (folder_name[i].split("||")[2] == 0)
        {
            fnSpan.previousSibling.firstChild.setAttribute("src", "I/onlygrass_close.png");
        }
Cookie solved the problem that the current user has data about SpringioBox all along, but how can this current user send these data/cookies to friends?

The most straightforward way came to my mind is – storing them in the url according to some pattern. And fetch them back to cookies when someone comes to the url.

Since there might be several folders in SpringioBox, we need to know which folder should be sent. The order number is passed as a parameter in addition to other stored cookies. A long url is returned by calling the following function.

```javascript
function getEmail_box(num) {
    var name_arr = getCookie("folder_names").split("###");
    var name = name_arr[num].split("||")[0];
    var length = parseInt(name_arr[num].split("||")[1]);
    //Calculate the start index of images
    var start = 0;
    for (var i=0; i<num; i++) {
        start += parseInt(name_arr[i].split("||")[1]);
    }
    var qs = "q=";
```
for (i=start; i<(start+length); i++)
{
    if(i != start) {
        qs = qs + "**";
    }
    qs = qs + img_q_array[i].split("||")[0];

    if (img_q_array[i].split("||")[3] == "image")
    {
        widths = widths + img_width_array[i].split("||")[0];
    }
    else if (img_q_array[i].split("||")[3] == "video")
    {
    }
    else if (img_q_array[i].split("||")[3] == "audio")
    {
    }
}

var link = URL + "/shareInfo?ft=1&name=" + name + ";
link = link + qs + ids + page_urls + image_urls + widths + heights +
       filessizes + durations + bitrates + thumb_urls + frompages;

return link;
}

How can we solve the long url thing? Noticed that we have tinyURL kind of service these
days? Luckily we have an open source service called Smurl (http://smurl.name/) which is
also easy to include in our Python code.

```python
>>> import xmlrpclib
>>> api = xmlrpclib.Server('http://smurl.name/

>>> print api.createSmurl('http://www.somelengthydomain.name/path/test')
'http://smurl.name/6r'
```

4.2.6 Evaluation

This is the hardest problem we encountered during the whole process.

The way we pass the data via cookie is not very efficient, each time it use indexOf to
calculate the position manually. A better solution is to use JSON - short for JavaScript
Object Notation, is a lightweight computer data interchange format. It is a text-based,
human-readable format for representing simple data structures and associative arrays
(called objects).

The way we construct the url is not efficient either, it's better to generate some unique
hashcode for each url to email, but it might require database support. but at the same time,
it doesn't need to rely on the smurl server.
5 Brush up

5.1 Cross Browser Problem Solving

We used Firefox (http://www.mozillaonline.com/) while developing, mainly because of its fancy add-on – firebug (http://getfirebug.com/). It makes edit, debug, and monitor CSS, HTML, and JavaScript live in any web page. Now it’s time to compromise to IE who shares the majority of browser market.

IE is notorious around the web developing area for its nonstandard render way yet high market share. After trying to make Springio look and behave ok in IE, we find these differences.

CSS
1 IE doesn’t support z-index in a proper way – always create a new stack context starting with the value of 0;
2 IE6 doesn’t support input[type=text];
3 Content disappears and then reappears after scrolling or window resize(peek-a-boo bug) in IE6;
4 IE6 doesn’t support :hover.

Javascript
1 IE doesn’t support setAttribute, instead use XX.ARRIBUTE = VALUE;
2 IE doesn’t accept variables without initialization;
3 IE doesn’t support removeChild() function;
4 IE doesn’t support the value NaN;
5 When set the style visibility to default, in FF using null, in IE using “ ”.

Other
URL length limitations are different between FF and IE.

5.2 Search Engine Optimization(SEO)

After uploading and configurating of the whole website, now the most important thing is to make it easily found by search engine – which helps springio draw more traffic.

Search engine optimization(SEO) is the process of improving the volume and quality of traffic to a website from search engines via "natural" ("organic" or "algorithmic") search
results for targeted key words. Usually, the earlier a site is presented in the search results or the higher it "ranks", the more searchers will visit that site. (wikipedia)
6 Conclusion, Evaluation and Future work

6.1 Conclusion
Springio is my dear baby and she is better than I could imagine - clean style and rich functionalities. Moreover, I learned a lot from it indeed!

A) Preparation is important
Concluding the process of creating Springio, I would like to say “as you can imagine, you can always make it work!“ That's why the benchmark part is so important within the whole process, although it might feel like doing some dull collecting. We choose features and get inspirations from the comparison of different websites’ different ways.

Literature reading is also important since it gives us the solid basis and confidence to depend on and inspiration as well.

B) “Dream” as wild as possible
When it comes to picking the features we wanna implement, I doubted now and then whether it could really be implemented. I didn't know how far it could go, but I kept “dreaming” on and on.

I encountered overwhelming problems now and then when I even thought about giving up, however, with the help of Search Engines and other people, all of them are solved along the way. “There is always a way out!”

Try to imagine, if I didn't dream from the start, or afraid of unable to implement them. The site would definitely be not so good as now! That’s the power of imagination!

C) Coding is similar no matter which language
This is the first time I code in python, I had concerns before I started it. But with the help of my colleague, I got hold of it quickly. It's really a great chance for me to learn a new programming language during a project. Now I can put Python in my CV!

6.2 Evaluation and Future Work
Springio has refreshing style – cute logo and clean color and layout and rich user interface functionalities – add images to springio box; share images with other users; filter results according to different standards; switch thumbnail sizes. According to alpha user(all colleagues at picsearch) testing, it fullfills the requirements we propose at the start – give users a refreshing and brand new experience of using multimedia search service.
Thanks to each step of the progress, all of them are so necessary and meaningful.
Literature review gives us the theory basis; Benchmark other search engines enriches our ideas; Scenarios writing make it more straightforward and detailed to understand what we wanna achieve; draft prototype presents our thoughts early which brings early feedback.

Springio complies with web standard – which means in different browsers, it could always display and work well. Springio also is rich in user-friendly functionalities.

However, being not satisfied is the force to improvement. Springio still have a lot of enhancement to explore.

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. On one hand, as you can see, we encountered a lot of problems about javascript among different browsers -- jQuery is browser compatible; on the other hand, we used different javascript library within Springio to implement some special effect – jQuery has a rich plugin library to choose from.

2) Instead of depending on cookie, choose another way
The way Springio takes to store the image/video/audio info is depending on cookie by javascript, which means if the user forbids cookie or javascript, it can't work at all.

3) Make a Mobile version
As surfing internet via mobile gets more and more popular, it's reasonable to make a mobile version.

4) Share among Facebook[5]/Twitter[6] and etc
Social media websites have a large user group basis, to promote Springio, sharing among them is a good choice.
References

http://kurafire.net/log/archive/2007/06/22/pagination-101