PetFinder

Team Members

- Jia Li
- Sophia Raji
- Theerarun Tubnonghee
- Girishkumar Ramkumar

Project Title

PetFinder

Vision Statement

Come and give your future companion a home!

Automated Tests

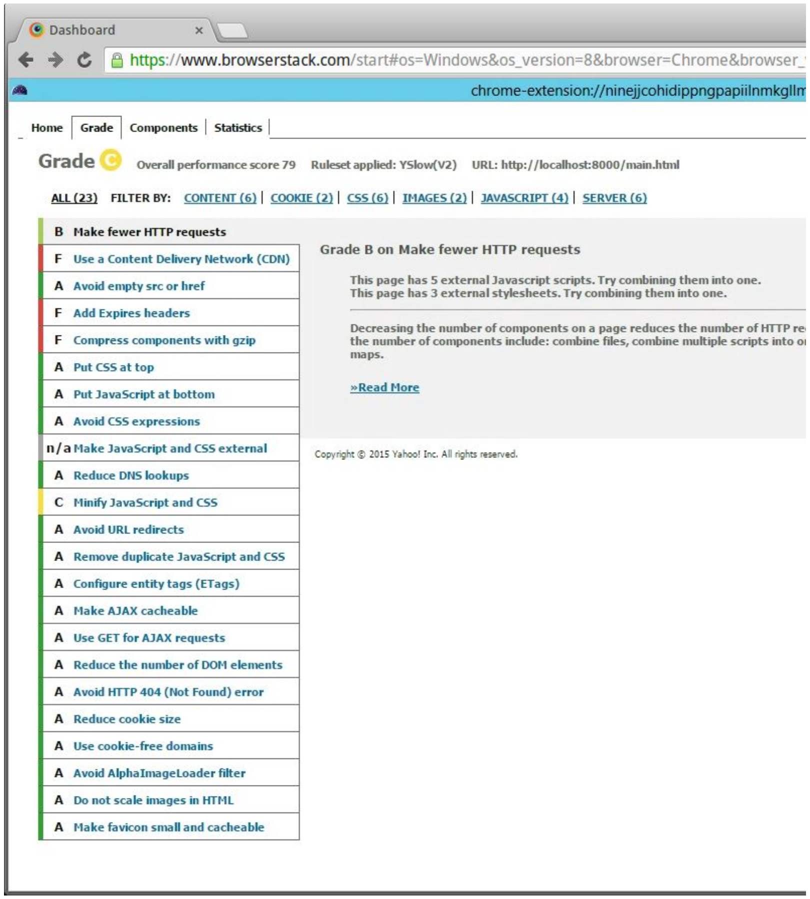
- **Tools used for testing**

The primary tool we used for testing was BrowserStack. BrowserStack is an online website/chrome extension that allows us to test our locally hosted website. BrowserStack has developer tools that allow us to test the efficiency, workflow, layouts and interactivity of our website.

- **Automated Testing Explanation**

In order to run tests with BrowserStack, one must register themselves using their email to create a free account. Once the account is created we can open the Google Chrome Browser and navigate the the BrowserStack website. Once the extension is up and running we can begin by selecting what kind of system and browser we can test our website with. For our purposes we used Windows 8 with the most recent edition of Chrome(46) to test our website. Once BrowserStack finishes launching the browser we can locally host our website using the command “python -m SimpleHTTPServer 8000”. Once the local server is up, we can navigate to our website by typing in “localhost:8000” in the URL bar of the browser launched by BrowserStack. The browser launched by BrowserStack has built in developer tools that we can use to test our website. BrowserStack also has a general test case that it runs the website through and grades our website as well as identifying the areas of the website that can be improved.This was our primary use of BrowserStack.

- **Output of Automated Test Cases**


Dashboard
https://www.browserstack.com/start#os=Windows&os_version=8&browser=Chrome&browser_
chrome-extension://ninejjcohidippngpapiilnmkgllm
Home
Grade
Components
Statistics

Grade C Overall performance score 79 Ruleset applied: YSlow(V2) URL: http://localhost:8000/main.html

ALL (23) FILTER BY: CONTENT (6) | COOKIE (2) | CSS (6) | IMAGES (2) | JAVASCRIPT (4) | SERVER (6)

B	Make fewer HTTP requests
F	Use a Content Delivery Network (CDN)
A	Avoid empty src or href
F	Add Expires headers
F	Compress components with gzip
A	Put CSS at top
A	Put JavaScript at bottom
A	Avoid CSS expressions
n/a	Make JavaScript and CSS external
A	Reduce DNS lookups
C	Minify JavaScript and CSS
A	Avoid URL redirects
A	Remove duplicate JavaScript and CSS
A	Configure entity tags (ETags)
A	Make AJAX cacheable
A	Use GET for AJAX requests
A	Reduce the number of DOM elements
A	Avoid HTTP 404 (Not Found) error
A	Reduce cookie size
A	Use cookie-free domains
A	Avoid AlphaImageLoader filter
A	Do not scale images in HTML
A	Make favicon small and cacheable

Grade B on Make fewer HTTP requests

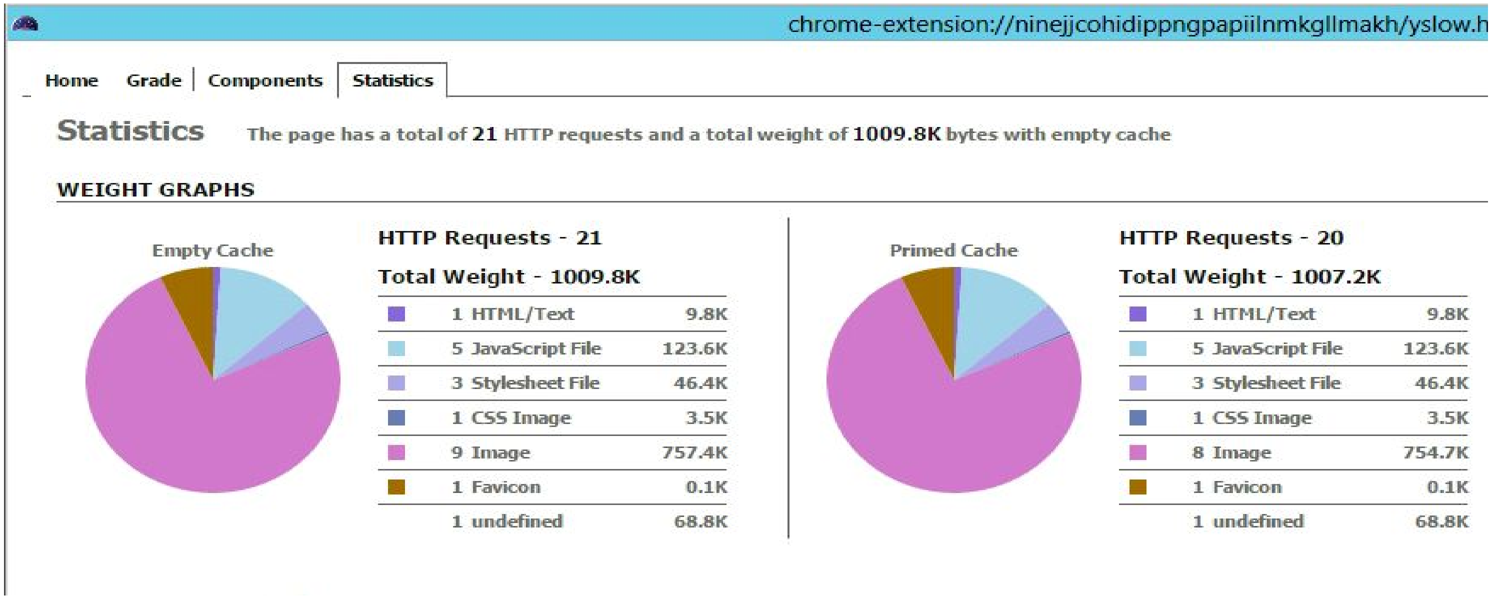
This page has 5 external Javascript scripts. Try combining them into one.
This page has 3 external stylesheets. Try combining them into one.

Decreasing the number of components on a page reduces the number of HTTP re
the number of components include: combine files, combine multiple scripts into o
maps.

»Read More

Copyright © 2015 Yahoo! Inc. All rights reserved.

The few F’s that we got when we ran the automated test tool were due to the fact that our database has not been linked to our website yet as well as a few optimization tricks we were not aware of including compressing components of the javascript and html files to reduce the size of the HTTP response.



Statistics The page has a total of **21** HTTP requests and a total weight of **1009.8K** bytes with empty cache

WEIGHT GRAPHS

HTTP Requests - 21

Total Weight - 1009.8K

■	1 HTML/Text	9.8K
■	5 JavaScript File	123.6K
■	3 Stylesheet File	46.4K
■	1 CSS Image	3.5K
■	9 Image	757.4K
■	1 Favicon	0.1K
	1 undefined	68.8K

HTTP Requests - 20

Total Weight - 1007.2K

■	1 HTML/Text	9.8K
■	5 JavaScript File	123.6K
■	3 Stylesheet File	46.4K
■	1 CSS Image	3.5K
■	8 Image	754.7K
■	1 Favicon	0.1K
	1 undefined	68.8K

The development tools also include a statistics page where we can view how many HTTP requests we are making and where the majority of the weight of the website lies. These tools are really effective for increasing the efficiency and responsiveness of our website as well as identifying the problem areas of our website.

Link To Tool: https://www.browserstack.com

User Acceptance Tests Plans

Project Name: PetFinder	
Test Case for User Requirements	
Test Case ID: User-01	**Test Designed by:** Jia Li
Test Priority (Low/Medium/High): High	**Test Designed date:** <Nov.11th.2015>
Module Name: PetFinder User Registeration	**Test Executed by:** <Name>
Test Title: Verify user account registration	**Test Executed date:** <Date>
Description: Test the registration function for PetFinder users	
Pre-conditions: User has username and password	
Dependencies: N/A	

Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**	**Notes**
1	Navigate to register page	*Username: petfinder@gmail.com*	User should be able to register			Test data is only an example.
2	Provide valid username	*Password: 3308pet*	an account and			
3	Provide valid password		link to the account			
4	Click on Register button		page.			

Post-Conditions:
User has successfully registered an account in website. The account session details are inserted into database user table.

Project Name: PetFinder

Test Case for User Requirements

Test Case ID: User-02	**Test Designed by:** Jia Li
Test Priority (Low/Medium/High): High	**Test Designed date:** <Nov.11th.2015>
Module Name: PetFinder User Login	**Test Executed by:** <Name>
Test Title: Verify account login with valid username and password	**Test Executed date:** <Date>

Description: Test the login function for PetFinder users

Pre-conditions: User has valid username and password

Dependencies: N/A

Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**	**Notes**
1	Navigate to login page	*Username: petfinder@gmail.com*	User should be able to login			Test data is only an example.
2	Provide valid username	*Password: 3308pet*	to the account and			
3	Provide valid password		navigate to account.			
4	Click on Login button					

Post-Conditions:

User is validated with database user table and successfully login to the account. The account login session are marked logged in.

Project Name: PetFinder

Test Case for User Requirements

Test Case ID: User-03	**Test Designed by:** Jia Li
Test Priority (Low/Medium/High): High	**Test Designed date:** <Nov.11th.2015>
Module Name: PetFinder User Pets Search	**Test Executed by:** <Name>
Test Title: Verify Search Pet function	**Test Executed date:** <Date>

Description: Test the search function for PetFinder users

Pre-conditions: User has valid search keyword

Dependencies: User should has an account on website and successfully login

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to search box	*Search Keyword: small dog*	User should be able to see a list of			Test data is only an example.
2	Type in search keyword		available dogs			
3	Click on Search button					

Post-Conditions:
User has successfully got a list of pets fulfill the search keyword shown on website.

Link To VCS: https://github.com/bettylj/PetFinder