Justin Gallagher, Ted Li, Jacob Zimmerman, Howard Chen
Group 20
November 14, 2014

15-221
Fall 2014
Section A

ExplainShell for Chrome

Progress Report

Submitted to
Thomas M. Keating
Assistant Teaching Professor
School of Computer Science
Carnegie Mellon University

Prepared by
Justin Gallagher
Ted Li
Jacob Zimmerman
Howard Chen

School of Computer Science
Carnegie Mellon University

November 14, 2014
(Professor Keating granted us a two-day extension)

Contents

1 Overview

1.1 Purpose

This report intends to inform the client of our current progress in building ExplainShell for Chrome. Specifically, we will cover additional information that we have found relevant to our project, the goals we have accomplished, what we still need to complete, required changes to our original procedure, and our plan for finishing the project at the deadline. We will conclude with a discussion of what exactly we expect to produce when finished.

1.2 Project

ExplainShell for Chrome is intended to help the user understand Unix shell commands found online. With the aid of our Chrome extension, users will be able to click on commands and be directed to the relevant `explainshell.com` page. Our application eliminates the need to copy and paste the command into `explainshell.com` and places a friendly reminder of ExplainShell's utility in the reader's view. Combined, these two aspects eliminate the largest barriers to people using `explainshell.com` freely, thus promoting its use.

As a complement to this utility, each click that users generate by following links to `explainshell.com` from our Chrome extension will be sent off anonymously to our ExplainShell Trends site, which will compile a database of clicks which we will be able to query for statistics such as most recently clicked commands, top referring sites and pages, most popular commands, and more. Through this site, ExplainShell for Chrome users *and others* will be able to learn about new, helpful shell commands.

Thus, this system will both help bring attention to commands that users don't understand as well as introduce them to new content generated by their peers. It solves issues related to malicious code execution (because potential victims will be more wary about running unknown commands), Unix shell education, and community involvement.

2 Literature Review

We have found a number of reference materials useful along the development process, mostly relating to the actual implementation of Chrome extensions and web apps in Node.js. Of the sites we've visited, those that we document here have been the most crucial to our current progress.

2.1 Chrome Extension Developer Docs

To get the ball rolling, we consulted the Chrome Extension Developer documentation. There was a convenient starter extension[1] that in the end we only had to minimally tweak to get the first version

[1] `https://developer.chrome.com/extensions/samples#search:contextmenus`

of our app working. Even though it was small, the importance of this reference cannot be understated, because through it we learned some best practices about writing Chrome extensions. These include things like what pieces of metadata to include in the `manifest.json` file and how best to structure our files and assets.

2.2 Setting Up a Node.js Stack with Express & MongoDB

Setting up the backend ExplainShell Trends site turned out to be relatively painless due to the comprehensiveness of this tutorial[2]. It went through the process of getting Node.js installed and getting a simple app up and running. From here, we were able to adapt the site to our needs, including setting up boilerplate pages for the landing and about pages.

This tutorial also covered the process of setting up a local MongoDB instance, which turned out to be incredibly straightforward. Using MongoDB, we were able to begin populating the database with some sample data points for the purpose of debugging.

3 Progress

We now present a general assessment of our progress so far. We will also propose potential changes to our original schedule.

3.1 General Assessment

Up until this week, we have completed the development of Phase 1, 2, and 4. So far, we have built a Chrome extension that parses and searches any web page to identify any shell commands. As the extension finds any potential commands using a set of heuristics, it will turn the original command text into a clickable link, which will redirect the user to the corresponding usage documentation at the ExplainShell website.

We have also completed the development of a back-end server that tracks the commands that users search for using our extension. The server compiles a database of clicks, which we will use to present various analytics, including most recently visited commands, trending commands and top referring sites.

The next step is to complete Phase 3, during which we will integrate the front-end interface with the back-end server. To do this, we will likely need to contribute to the existing ExplainShell code base to add in relevant APIs.

3.2 Status

Below is a general list of tasks that we have completed so far. Note that the items listed below correspond to our original Gantt chart we included in our initial proposal.

[2] http://cwbuecheler.com/web/tutorials/2013/node-express-mongo/

- Create initial Chrome extension
- Test initial Chrome extension
- Parse page, add in external link
- Create in-page popup on hover (partially completed, see below for details)
- Setup basic analytics / trends site
- Add analytics support to back-end

Note that we have only partially completed the development of the in-page popup. Our original proposal was to create a custom-styled popup to display relevant information. This however would require us to first contribute to the ExplainShell API, since we need to add in relevant API that allows us to retrieve relevant reference documents from ExplainShell through web requests. Therefore, the development of the in-page popup cannot be fully completed before the ExplainShell API is available.

Therefore as of now, we have to temporarily compromise with an in-page popup that uses an `iframe` web element to display the reference. This falls short of our original proposal, since an `iframe` web element does not allow us to customize the look and feel of the popup, and usually suffers from a slower loading time.

As a reference of the work we have completed so far, please refer to our GitHub repository[3]. A rudimentary version of our Chrome extension is also available on the Chrome Web Store[4].

3.3 Projections

Our project is slightly falling behind. This is partly due to the fact that the time cycle of contributing to the ExplainShell open-source took longer than we thought. We underestimated how long it would take to research the technologies being used in the ExplainShell website (like Flask), and the length of time required to gain a deep enough understanding of how the ExplainShell app works. Since part of our project is dependent upon the integration of our code into the ExplainShell code base, we may have to make a few adjustments to our original schedule to deliver our final product on time.

To this end, we be making some adjustments to our original plan. Particularly, if we cannot contribute to the ExplainShell code base by our project deadline, we may need to compromise with our existing `iframe` solution for the in-page popup, instead of using a custom-styled popup as proposed originally.

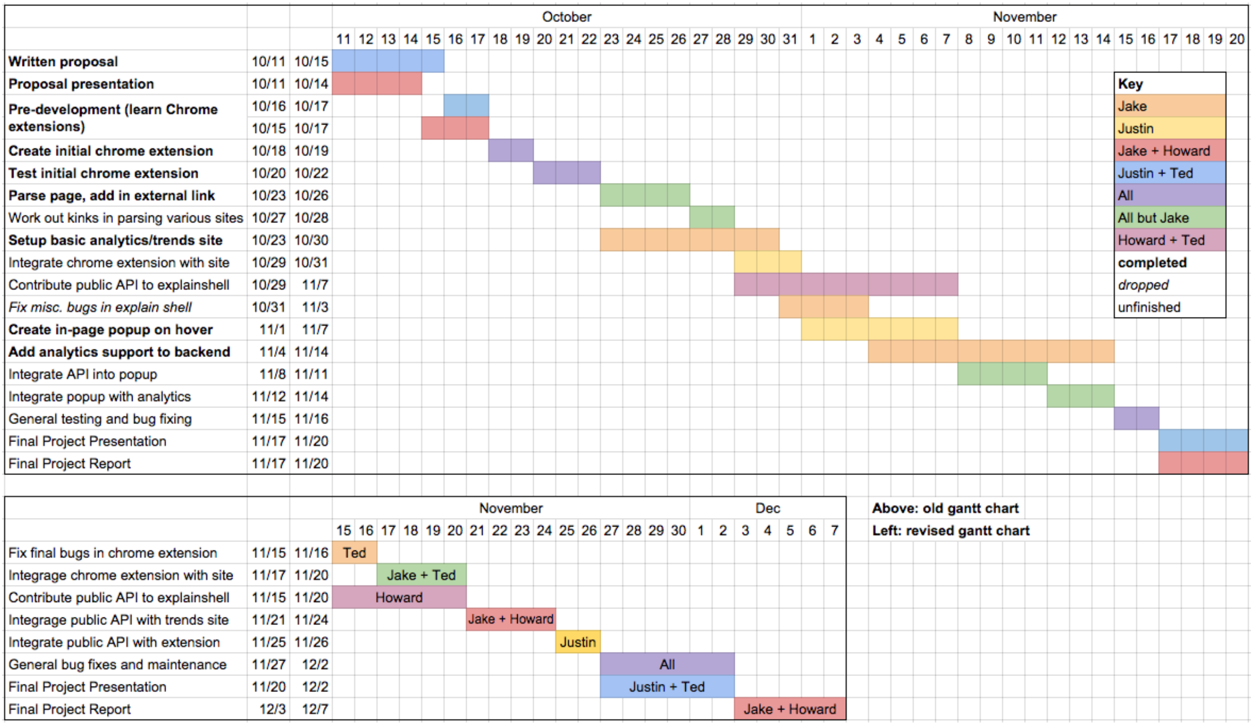
Our next major step would be to integrate the back-end logic with the front-end. After the integration, our back-end server would be able to track what users are searching for and provide relevant analytics. This also would be the final portion of our project. We expect it to take one to two weeks to complete.

[3] https://github.com/justingallagher/explainshell-for-chrome

[4] https://chrome.google.com/webstore/detail/explainshell-for-chrome/mhjgkgmolboemmbempmaapihlgcpomfi

4 Recommendations

Considering the above projections, we recommend the revised Gantt chart seen in Figure 4. It is given in context with a modified version of the old Gantt chart that shows which tasks have been completed, dropped, and not finished.



Written proposal	10/11	10/15
Proposal presentation	10/11	10/14
Pre-development (learn Chrome extensions)	10/16	10/17
	10/15	10/17
Create initial chrome extension	10/18	10/19
Test initial chrome extension	10/20	10/22
Parse page, add in external link	10/23	10/26
Work out kinks in parsing various sites	10/27	10/28
Setup basic analytics/trends site	10/23	10/30
Integrate chrome extension with site	10/29	10/31
Contribute public API to explainshell	10/29	11/7
Fix misc. bugs in explain shell	10/31	11/3
Create in-page popup on hover	11/1	11/7
Add analytics support to backend	11/4	11/14
Integrate API into popup	11/8	11/11
Integrate popup with analytics	11/12	11/14
General testing and bug fixing	11/15	11/16
Final Project Presentation	11/17	11/20
Final Project Report	11/17	11/20

Key

- Jake
- Justin
- Jake + Howard
- Justin + Ted
- All
- All but Jake
- Howard + Ted
- **completed**
- *dropped*
- unfinished

Fix final bugs in chrome extension	11/15	11/16	Ted
Integrage chrome extension with site	11/17	11/20	Jake + Ted
Contribute public API to explainshell	11/15	11/20	Howard
Integrage public API with trends site	11/21	11/24	Jake + Howard
Integrate public API with extension	11/25	11/26	Justin
General bug fixes and maintenance	11/27	12/2	All
Final Project Presentation	11/20	12/2	Justin + Ted
Final Project Report	12/3	12/7	Jake + Howard

Above: old gantt chart
Left: revised gantt chart

Figure 1: Revised Gantt chart, in context with progress on old Gantt chart

As discussed earlier, these tasks are centered around resolving the two bottlenecks to our current development: contributing to a public ExplainShell API, and integrating it with the popup in our Chrome extension.

Once these tasks are done, we will be well on our way to finishing up the project. The only thing that will need to be done after this is to collaborate on the final project report and presentation, but by that time the only ongoing development will be general maintenance.

5 Discussion

Unfortunately, we were unable to use our excess time to contribute back to the ExplainShell repository fixing bugs and tidying up the code base. We had allocated time to fix things like general bugs, but now we will only have enough time to contribute a public API. This does not, however, interfere with the main goals of the ExplainShell for Chrome project. It would have merely been a nice gesture, considering that the ExplainShell app provides the core basis of functionality to our project.