# RegEx Search & Replace Extension for Chrome and Firefox

*Dalimil Hajek*

BSc Hons Project Report
Computer Science
School of Informatics
University of Edinburgh

2018

# Abstract

The aim of this project was to build a browser extension to allow users to search and replace text with regular expressions in editable text input fields of web pages.

After evaluating existing extensions that were unsuccessfully attempting to implement this functionality, the new extension has been carefully designed, developed, and finally successfully released for Chrome and Firefox browsers.

In addition to the future-rich search and replace function, this plugin also adds the ability to save favorite patterns, store search history, or predefine text templates that can be inserted into the editable area of a page.

The software followed an iterative development process, where user feedback was collected via several means, including Google Analytics, which was used to track user interaction, and a support website used to collect user feedback comments.

After the initial release, about twenty updates have been subsequently released over the span of a few months. This iteration was further supported by automated tests of several kinds.

The extension has received excellent reviews and at the time of writing has over 3000 weekly users (users from both browsers combined).

4

## Acknowledgements

Thanks to

- Boris Grot - for supervising the whole project, and making important feature suggestions leading up to the first official releases of the extension on the Chrome and Firefox web stores

- Michael O'Boyle - for making suggestions, especially regarding Google Analytics

- Christoph Metze - for finding several important bugs that subsequently led to releases `1.3.2`, `1.3.3`, and `1.3.4`

- Daniel Tomberlin - for pointing out a use case when trying to search across multiple single-line inputs, and for updating his web store rating and review after I implemented it in `1.3.6`

- GitHub user MarkRH - for finding a bug that was later fixed in `1.1.3`

- StackOverflow user wOxxOm - for suggesting `Document.execCommand` API that I used to fix issues with templates in `1.2.0`

And also thanks to all those people who submitted user feedback or reviews.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Dalimil Hajek*)

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

Modern web browsers allow users to find text in a web page, but when it comes to editable text areas that are often used on blogging platforms, online forums, social media, and email web clients, as a form of user input, none of the existing browsers allow users to also replace the found occurrences.

The aim of this project was to build an extension that adds this browser functionality. The tool allows users to find and replace text in editable text input fields of web pages, and includes a large set of features and search options, including regular expression support and occurrence highlighting.

## 1.1 Motivation

Search and replace functionality can be extremely useful when composing long emails, writing posts on social media, online forums, or blogging platforms, as well as in any email web clients.

The most frequent use cases are:

- Fixing a typographical error – A word or a phrase may have been used several times with the wrong spelling. This can reoccur several times in a forum post or an email, and it would be convenient to replace everything at once.

  This also includes fixing unreadable characters in blog entries such as $Â€?$, due to a change in encoding or some unintended text handling.

- Normalizing incorrectly formatted text – Regular expressions can be used to detect formatting errors such as multiple spaces before a period, missing upper-case letter, various metric unit formatting errors, and similar. Search and replace extension supporting regular expressions can quickly find and fix these.

- Renaming a phrase – Often a word or a phrase that occurs several times throughout a text needs to be corrected or substituted (perhaps using a synonym or a wording that sounds better)

Without having a browser extension for search and replace, one could imagine a solution where all text is copied and pasted into an advanced text editor, fixed using the built-in search and replace function, and copied back into the web page input field.

In addition to being a lengthy and time consuming process, this method would in many cases lose all text formatting, because more advanced editable text elements on the web may contain images, emojis, and text containing many formatting tags, which would not be preserved during the copy-pasting.

Additional motivation behind the development of this project was to add search-and-replace related features that are missing even from the more advanced text editors. One of them is storing the search history, and also being able to save favorite search patterns, that can later be quickly accessed. Both of these would save time and increase user productivity.

## 1.2   Existing Extensions

Web browsers support standard search functionality for any text on a page but no browsers have the find & replace functionality. Users have asked for this feature on Google Chrome forums[1], but the decision of browser developers was to leave the implementation of this functionality to potential text-processing web applications, rather than implementing it as a part of the browser.

Figure 1.1: Google Chrome browser find tool

Figure 1.2: Mozilla Firefox browser find tool

_____

[1]Google Chrome Forum link: https://productforums.google.com/forum/#!topic/chrome/Y4UORlpdYfo

There have been several attempts to implement this functionality via an extension. Most of them either don't work, are missing functionality (particularly support for regular expressions), are limited to certain websites, or are counter-intuitive and hard to use in general.

### 1.2.1 Chrome

- **Search and Replace**[2]

    - This is the dominant one with over 52,000 users (as of Dec 2017)
    - It has only 3.1/5 stars and many reviews are complaining it doesn't work or that it destroys other content of the current page
    - It has usability issues (control UI is partially hidden) and it doesn't work in many places such as Blogger or Facebook
    - This is a very bad and simplistic solution that is mostly broken but benefits from a larger user base because it appeared in the web store in 2013

- **Find Replace**[3]

    - This extension requires copy-pasting your desired text into the provided box (it is plain-text only, so formatting is lost).
    - It has almost 1000 users (as of Dec 2017), but it has been around since 2013
    - It is unable to perform the search and replace in browser. This is no better than copy pasting text into a text editor and performing the operation there

- **FindR**[4]

    - It has almost 2000 users and 3.9/5 stars. It has been in the web store since April 2016.
    - This extension is used for replacing any HTML text in the page, rather than text in input fields – for this purpose it seems to work, but when one tries to use it only for text input fields, things break and the extension stops working (highlighting and match indicator both disappear and replace button stops working)
    - It also requires permission to *Read and change all your data on the websites that you visit*, which might be a privacy issue (the extension can read everything even when the user isn't using it)

- **Easy Replace**[5]

    - It has over 3000 users but only 2.6/5 stars
    - Most of them report it doesn't work because it only focuses on plain-text text areas and completely ignores more advanced editable HTML elements that most sites use these days

---

[2]https://chrome.google.com/webstore/detail/search-and-replace/bldchfkhmnkoimaciljpilanilmbnofo
[3]https://chrome.google.com/webstore/detail/find-replace/cfjmfciolkikfodjfdmdpdmpfbjdofek
[4]https://chrome.google.com/webstore/detail/findr/bidnaaogcagbdidehabnjfedabckhdgc
[5]https://chrome.google.com/webstore/detail/easy-replace/ojoeejfegihohnkjlfoonbnailkohkce

### 1.2.2 Firefox

- **Find and Replace for Firefox**[6]

    - Old add-on, not compatible with the latest Firefox 57.
    - It has almost 3000 users but only 3.5/5 stars
    - It's not working for most users, has almost no options (no RegEx, no high-lighting, etc.)
    - It was last updated in 2012 and is not maintained

- **FoxReplace**[7]

    - This extension provides different functionality – it asks the user to prede-fine a list of substitutions and then it automatically applies them globally across the text in newly loaded web pages
    - FoxReplace has over 7000 users (Dec 2017) but targets a different audience


### 1.2.3 Other Browsers

Supporting more browsers add extra work, so it only makes sense if these browsers represent a significant portion of the market. The usage share of browsers is measured by various methods and is often quite inaccurate (see [19]). Approximate values for usage share for desktop browsers are (taken from [16]): Chrome 65%, Firefox 12%, IE 8%, Safari 6%, Edge 4%.

This project focused on Chrome and Firefox, which together represent a large portion of the overall market, and which both mostly follow the same Extension API (there are only a few differences [8] so we didn't need to have separate code-bases).

Safari, although widely used by Mac users, has its own extension API and is in general much more involved as it requires dealing with Apple's developer libraries and licenses, so we decided not to develop an extension for this browser.

---

[6]https://addons.mozilla.org/en-US/firefox/addon/find-and-replace-for-firefox
[7]https://addons.mozilla.org/en-US/firefox/addon/foxreplace/

# Chapter 2

# Background

A browser extension is a bundled collection of files - HTML, CSS, JavaScript, images and other assets, that together add functionality to the browser. They have access to the standard DOM API (same as a regular website) but in addition to that they can access the WebExtensions API, which is provided by the browser. At the time of writing, the WebExtensions API is to a large extent (only with slight deviations in implementations) supported by Mozilla Firefox, Google Chrome, Microsoft Edge, and Opera browsers (see detailed explanation in [4], [3], [6] and [7]).

A typical extension consists of a background page, which holds the main logic of the extension and has full access to the APIs, pop-up widget, which presents the extension's user interface, and one or more content scripts, which are pieces of JavaScript code that are injected into the current web page so that we can interact with its content.

After installing a browser extension in Chrome or Firefox, a small icon representing the extension is added to the top toolbar. When users click this icon, the extension displays its pop-up widget. The standard way to present an interface to the user is via this widget. One could come up with alternative ways to present extension's UI but this design pattern using a pop-up is the preferred way and something that users expect by default. Since a pop-up widget works well in our scenario, we decided to go with that.

# Chapter 3

# Design

## 3.1 Naming and Search Engine Optimization

Based on the competition research mentioned in the introduction chapter above, the following extension names already exist: *Easy Replace*, *Search and Replace*, *FindR*, *Find Replace*, *FoxReplace*. Using any of these existing names would be bad for the SEO and discoverability, and would most likely lead to confusions.

At the same time, we want to clearly indicate that the extension is used for input fields and editable content rather than HTML source code.

People are likely to search for browser extensions by typing in the functionality that they need. In our case, that might look something like "find and replace in text input fields extension". Stating the extensions purpose in its title and description should help us do better in search results. Therefore, we avoided any newly-invented words and named it *Find & Replace for Text Editing*. We are not trying to trademark a new brand name, we are simply trying to match what people might search for, so that's why this name was chosen.

The naming idea was largely successful. In January 2018 (a few months after the first release), the extension ranked 2nd in the Google search results for queries *search and replace in chrome*, and *search and replace browser*, 1st for the query *search and replace in firefox*, 7th for the query *find and replace text*, and 8th for the query *search and replace text*.[1]

## 3.2 User Interface Inspiration

To some extent, we wanted to follow the current standard of find & replace toolbars. Many of these can be seen in more advanced text editors. The user interfaces of some of the popular ones were examined and used as an inspiration.

---

[1]Tested using incognito modes in both Firefox and Chrome, and searching via www.google.co.uk on 25th January 2018

Figure 3.1: Google Search Results Ranking

We wanted our extension to follow some of the design patterns that the existing text editors use, so that users are presented with a user interface that they can easily understand and start using quickly.

At the same time, it should not be assumed that average users are familiar with regular expression or more advanced search functions.  Therefore, the UI design of some of these editor widgets should only be used as an inspiration - average users are not developers and the number of options in this extension must not feel overwhelming.

## 3.3   UI Components

To support all the functionality that we specified, the extension widget should have the following UI elements:

- *Find* input field
- *Replace* input field
- **Action buttons**
    - Replace
    - Replace All
    - Find next
    - Find previous
    - Save to favorites
    - Expand/Collapse advanced search options
- **Options**
    - Match Case (Aa)
    - Whole Word (Ab|)
    - Use Regex (.*)
    - Limit to Text Selection
- *X of Y* or *No Results* indicator
- Regex groups indicator (for regex search only)

Figure 3.2: Android Studio find and replace tool



Figure 3.3: Google Docs find and replace tool



Figure 3.4: Visual Studio Code find and replace tool



Figure 3.5: Sublime Text 3 find and replace tool

- **Panel tabs**
  - Favorites
  - History
  - Templates
  - Help/Info/Feedback

We split the user interface layout into two types - simple and advanced. Because displaying all search options in one widget might feel overwhelming for average users, there is a way of switching the search UI to the *advanced* state that includes regex options and helpful previews of matched regex groups etc.
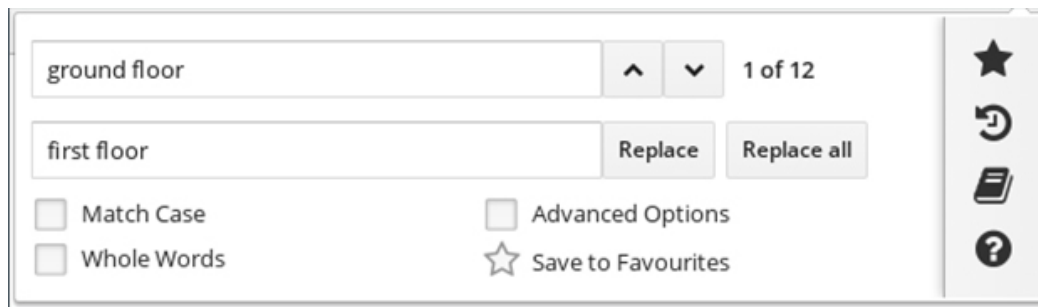


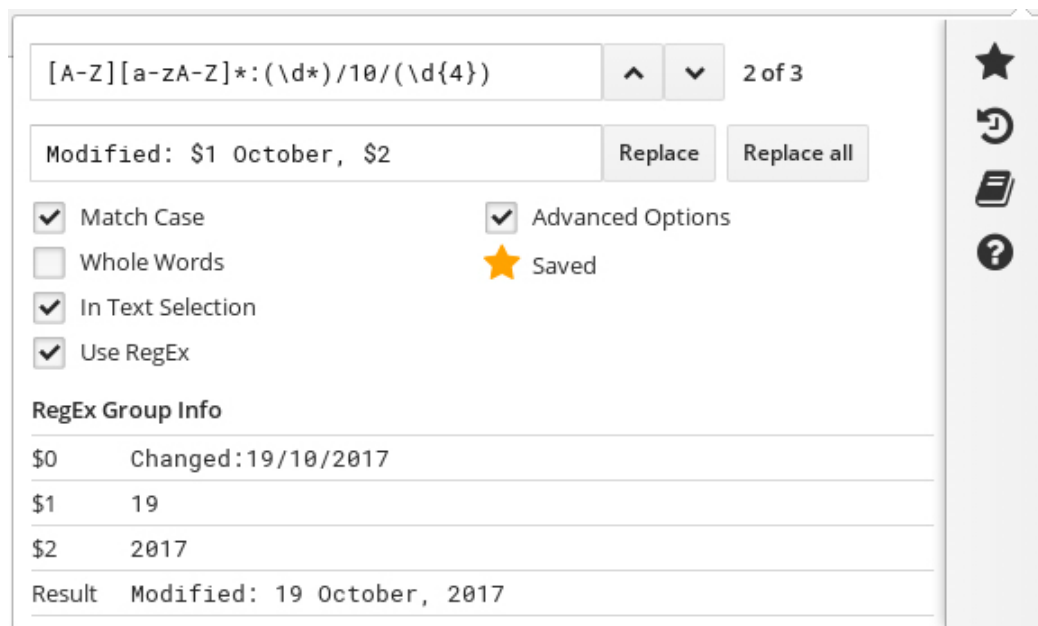Figure 3.6: Search widget UI – simple view



Figure 3.7: Search widget UI – advanced view

## 3.4   Why not Material Design

For the general look and feel, it was decided not to use Google's increasingly popular Material Design [5] for several reasons. First, Material Design works well when

there is a lot of space and all the elements can be spread out. Unfortunately, this extension's user interface is a small widget with very limited space and many condensed components.

Secondly, Material Design likes to add its *ripple effect* to most interactions (such as clicking a button). This looks great for actions that have large impact (e.g. navigating to a new page, or submitting a form), but for our purposes we need something less flashy, as most buttons are going to be pressed very often (*Find next* and *Replace* buttons) and too many effects or animations would present too much distraction for the user.

An extension (a pop-up widget) like ours should, to a large extent, feel like it is part of the browser by perhaps matching some of the user interface styles. We don't want to create an extension that gives users the feeling that it completely doesn't belong because of its wild user interface and flashy styles.

## 3.5  Scope of Search

Instead of searching in any visible text or simply the HTML source code, we want to limit our search only to elements that contain editable text. There are several ways to create an editable area on web. These are described in [12] and [20].

### 3.5.1  <input type="text">

For a short single line of text, HTML <input> element is often used. There are other types of input fields (many new were added with HTML5), such as date, email, number, tel, time, and similar, but text is the standard one (see [10]).

However, due to the short length (20 characters by default), this is not the kind of input that needs the find & replace functionality on its own. The primary use case of our extension are large multi-line text areas, but we also include the short single-line input kind to allow users to search and replace text in a page across many single-line inputs simultaneously.

### 3.5.2  <textarea></textarea>

Multi-line plain-text input space (see [11]). This should be a common target for find & replace. It is used by many sites to allow users compose longer pieces of text, one of them is a new post creation process on Reddit.

### 3.5.3  <**div contenteditable="true"**></**div**>

Enabling rich text formatting by allowing HTML tags inside the text area, `contenteditable` elements are used in Gmail, Facebook posts, Facebook Messenger, GitHub editor, Twitter, and many other sites.

Note that `contenteditable` is a global attribute and is therefore not limited to `div` tags (see [9]).

### 3.5.4  **contenteditable tag inside** <**iframe**></**iframe**>

Blogger.com is an example of a site that isolates the main contenteditable area in an iframe. When performing find & replace we must consider the scenario where we are dealing with elements inside an <iframe> on the page.

### 3.5.5  **Other DOM**

This extension is not meant to modify (find & replace) the raw HTML text of the page's source. It is limited to finding occurrences in text areas that are modifiable by users.

There are certainly sites that might try to avoid all the options discussed above and implement their own text editor functionality. One notable example is Google Docs, which is using static DOM but listen to user's keyboard events to modify it internally in JavaScript. Implementing an online text editor from scratch without using contenteditable or textareas involves a lot of programming work, and such editor should probably include its own find & replace functionality anyway, which is exactly what Google Docs do.

At the time when the project was first designed, it seemed reasonable to exclude single-line inputs from the scope, as their small editable input space didn't seem suitable for our search and replace functionality. After receiving a lot of user feedback which revealed new use cases, single-line inputs were included in the search scope via an optional checkbox option in the search widget (to ensure backward compatibility and no unexpected behavior for existing users).

# Chapter 4

# Implementation

## 4.1  Search Widget

To implement the search UI widget, we could simply create DOM elements for all the input components and listen to any changes as the user interacts with the UI. Unfortunately, all input components manage their own state - a better approach would be to store the state of the search parameters in one central place/datastore and have the UI inputs reflect this data. Therefore, we used the React.js library to implement the user interface.

The React library [15] has become very popular in recent years - one reason is that it enforces the pattern of always reflecting the current application state in the UI. React's internal state is the single source of truth (see React input handling [13]).

Without it, we would have to manage all the inputs separately and this could create many UI inconsistencies - incorrect update of our internal data might create a state of the application where our search parameters are set to certain values internally but display different state externally via our UI. As we are dealing with a lot of different inputs (many search parameters as well as the simple and advanced modes of the search layout), using React seemed to be a wise choice.

We set up the project so that we can test the UI widget components in isolation, without creating an actual browser extension and calling the relevant APIs. During development we simply mount our root UI component on a regular website. In production, the same component is mounted in the place of the pop-up widget, which also behaves as a web page.

To also support React's JSX syntax (see [14]) and newest ECMAScripts language standards, we used Babel [17] for compiling our JavaScript sources. We used Webpack [18] to bundle all our source code into a single file, which is obfuscated and small in size. Both of these libraries are widely used and often used in conjunction with React.

## 4.2   Accessibility

## 4.3   Favorites Panel

## 4.4   History Panel

## 4.5   Templates Panel

## 4.6   Help Panel

# Chapter 5

# Evaluation

## 5.1 Distribution and Marketing

The extension was released on Chrome and Firefox web stores, where users can download it. It was then shared on HackerNews, Quora, Facebook, and several subreddits on Reddit. These sources constituted only a minority of the overall web store visits however, and the major acquisition source remained to be organic Google search (people who specifically search for this extension on Google rather than learning about it via social media or forum posts).
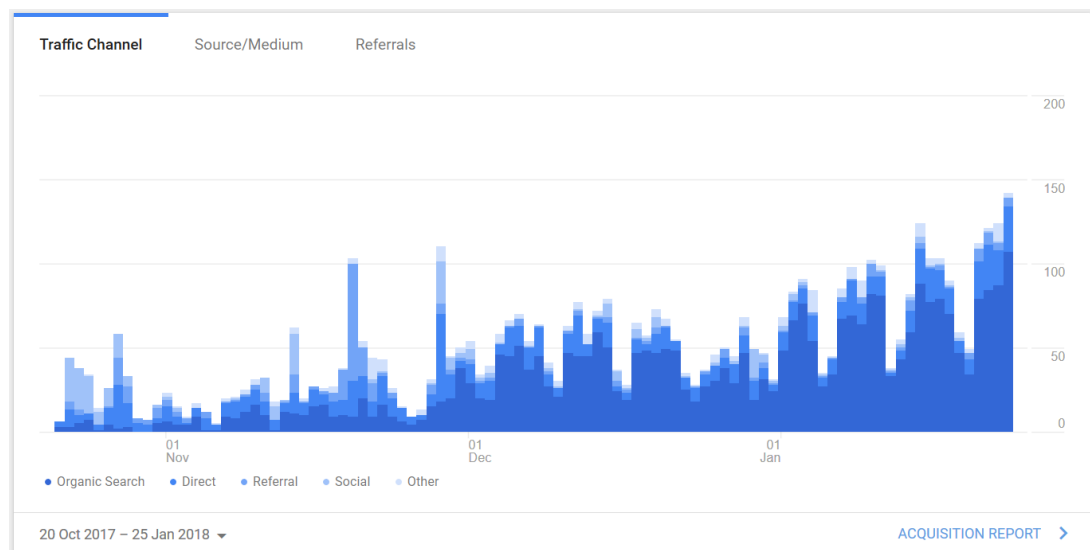


Figure 5.1: Chrome web store page - user traffic sources

## 5.2 Google Analytics

In order to collect statistics on how users use our extension and how they interact with different UI elements, we implemented Google Analytics into our extension.

Collecting this data requires us to inform users of the data collection and provide a way for the users to opt-out from collecting these analytics. This might dissuade a small percentage of potential users from installing the extension in the first place, but the belief was that this percentage would me small, which turned out to be true.

## 5.3  Testing

## 5.4  Support Website

There are various ways we could collect user-submitted feedback. We could have a website that provides a contact email address or asks users for a feedback message and sends the email automatically. This email inbox could get very cluttered and hard to manage, especially since we want to collect various forms of feedback (e.g. did the user recently uninstall the extension, or are they happy and simply want to propose new features?).

We decided to proceed with an alternative solution of creating a support website and storing all submitted feedback messages in a database. This database can later be queried and filtered based on various criteria (time of submission, feedback type - general feedback vs complaint after uninstalling, ...), which simplified feedback evaluation.

We created a static landing page website that presents the extension with relevant links to the browser web stores, and also allows users to submit feedback via a simple form.
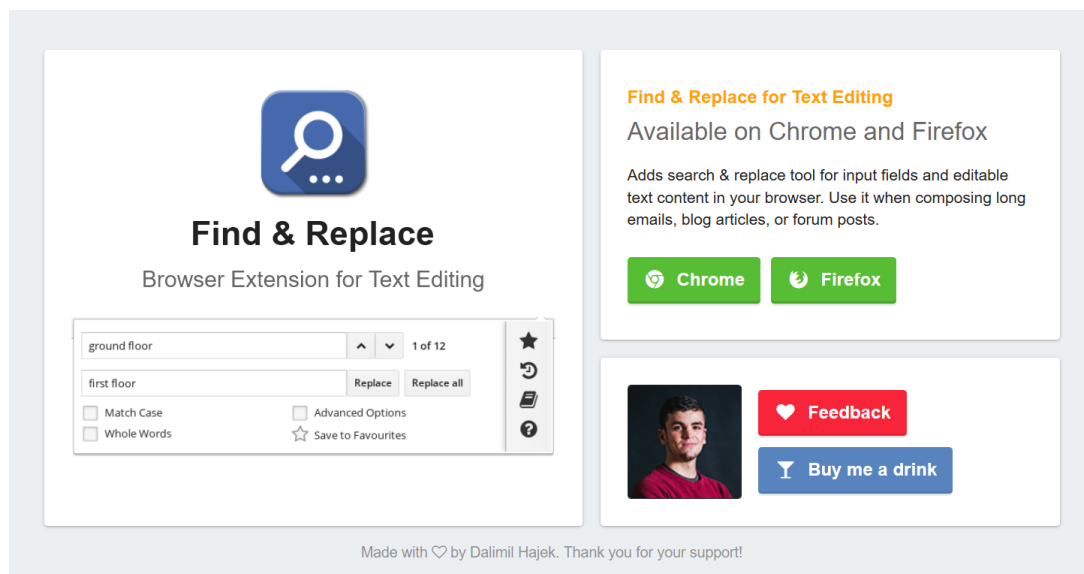


Figure 5.2: Support website landing page

The website was deployed using Firebase Hosting [1], which is Google's hosting for deploying static content to its content-delivery network (CDN).

The database used to store the feedback comments is Firebase Realtime Database [2], which is a NoSQL cloud database from the same family of Google products. This allowed us to have a secure place to store our data with minimal setup time. It also allowed us to have an optimized static website with only the required JavaScript SDKs without having to implement a complex back-end server.

# Chapter 6

# Conclusions

The extension has received excellent reviews and at the time of writing has over 3000 weekly users (users from both browsers combined), 4.7/5 stars (14 reviews) on the Chrome Web Store and 4.9/5 stars (7 reviews) on the Firefox Web Store.

## 6.1 Further Work

# Bibliography

[1] Google Developers. Firebase Hosting. https://firebase.google.com/docs/hosting/. [Online; accessed 26-January-2018].

[2] Google Developers. Firebase Realtime Database. https://firebase.google.com/docs/database/. [Online; accessed 26-January-2018].

[3] Google Chrome Developer Docs. Extensions Overview. https://developer.chrome.com/extensions/overview. [Online; accessed 25-January-2018].

[4] Google Chrome Developer Docs. What are extensions? https://developer.chrome.com/extensions. [Online; accessed 25-January-2018].

[5] Google. Material Design. https://material.io/. [Online; accessed 25-January-2018].

[6] Mozilla Developer Network. Anatomy of an extension. https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Anatomy_of_a_WebExtension. [Online; accessed 25-January-2018].

[7] Mozilla Developer Network. Browser Extensions. https://developer.mozilla.org/en-US/Add-ons/WebExtensions. [Online; accessed 25-January-2018].

[8] Mozilla Developer Network. Chrome incompatibilities. https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Chrome_incompatibilities. [Online; accessed 25-January-2018].

[9] Mozilla Developer Network. contenteditable. https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/contenteditable. [Online; accessed 25-January-2018].

[10] Mozilla Developer Network. HTML <input>. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input. [Online; accessed 25-January-2018].

[11] Mozilla Developer Network. HTML <textarea>. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea. [Online; accessed 25-January-2018].

[12] Mozilla Developer Network. Making content editable. https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Editable_content. [Online; accessed 25-January-2018].

[13] Facebook Open Source. Forms - React. https://reactjs.org/docs/forms.html. [Online; accessed 26-January-2018].

[14] Facebook Open Source. JSX in Depth. https://reactjs.org/docs/jsx-in-depth.html. [Online; accessed 26-January-2018].

[15] Facebook Open Source. React - A JavaScript library for building user interfaces. https://reactjs.org/. [Online; accessed 26-January-2018].

[16] StatCounter. Desktop Browser Market Share Worldwide. http://gs.statcounter. com/browser-market-share/desktop/worldwide/#monthly-201712-201712-bar, 2017. [Online; accessed 25-January-2018].

[17] Babel Open Source Team. Babel - The compiler for writing next generation JavaScript. https://babeljs.io/. [Online; accessed 26-January-2018].

[18] Webpack Open Source Team. Webpack - JavaScript module bundler. https: //webpack.js.org/. [Online; accessed 26-January-2018].

[19] Wikipedia. Usage share of web browsers. https://en.wikipedia.org/wiki/Usage_ share_of_web_browsers. [Online; accessed 26-January-2018].

[20] Johannes Wilm. The Editing taskforce. https://w3c.github.io/editing/, 2017. [Online; accessed 25-January-2018].