Title: Developing a web browser and Chrome extension with on the fly filtering and parent-child monitoring using machine learning algorithms, Node.js and Nw.js.

Description: The object of the project is to investigate and develop a method of solving a common problem in existing filtering systems. Trying to understand better what websites to block and what not to block. An example of this is a difference between an adult site vs. a sexual education site, which currently the majority of filtering systems block both.

Prepared for: Dr. Donna O'Shea & Gerard MacSweeney
Prepared by: Robert James Gabriel
Student Id: R00102430
GitHub organizations: Please note all set ups are in the readme of the GitHub repo

Browser: https://github.com/Projectbird/Robin
Chrome Extension: https://github.com/Projectbird/Robin-Chrome
Parser/Scraper: https://github.com/RobertJGabriel/Text-Scraping
Testing Data: https://github.com/RobertJGabriel/List-Of-Explicit-Words
Profanity Check: https://github.com/RobertJGabriel/profanity
Baylor Classifier Algorithm: https://github.com/RobertJGabriel/natural
Google Chrome link: https://chrome.google.com/webstore/detail/robin/bdikhnciedglconafjhhlommcnpajhjn?authuser=2

Landing Page : http://www.projectbird.com/robin

Showcased: http://www.uplabs.com/posts/material-design-web-browser
http://www.uplabs.com/posts/matieal-design-chrome-extension

Introduction

The following report contains the impregnation of the application outlined in semester one which was developing a web browser and Chrome extension with on the fly filtering and parent-child monitoring using machine learning algorithms, Node.js and Nw.js

The main reason for including a chrome extension is for it to serve as a reminder and to provide easier access to the parent so that they can monitor and control the settings of the web browser the children is using. Parents will be able to access the information on the filtering settings on the children's access to the internet from anywhere in the world. Both applications have been developed with the intention of being used by parents and teachers along with kids who would use the application to access the internet. Originally, the application was intended to create the web browser for the one machine only, but the surveys and research proved this was annoying.

Before the inclusion of the Chrome extension and Firebase cross-syncing in the system, the inspiration for the original project was due to a growing, everyday problem, the question of how parents should monitor or protect their children online. As the web develops and expands, so too does this issue, and the current systems are complex and hard to use, and the parent does not typically know how to set it up.

For this reason, I decided to create to build a web browser using Nw.js and a chrome extension whereby parents can quickly adjust settings to block and disable the web browser, which will then sync the information and store it locally and run based on that information. The idea was that, the parent could input to block “facebook.com” from the Chrome extension and that the web browser on the children's laptop would sync the information and block Facebook in real time. If the child were to access the information, it would redirect them back to Google and alert the parent about the attempt though the chrome extension. The goal was for the web browser to change color based on the child's behaviour. An example is that if they try to access the blacklisted site the colors of the web browser will turn to black. They will know what they did wrong and learned from the experience.

After further investigation into the idea application , I decided that although this application could be used by many people, in the majority, it is parents and teachers who have to worry about this problem.

A vast area of the research and development in this application is towards machine learning and the Baylor classifier algorithm. That the system core will be able to tell when a word is classed as profanity and build up a database on these terms and try to understand the if a page is an adult website or not and stop false positives. An example, it would be able to tell the difference between a porn website and a sexual education website.

Overview

This application is a filtering system for teachers and parents to keep an eye on what their children and students are doing online, the application is designed for educators and individuals with basic computer knowledge to enable them to keep track of their children's internet activity and time on the web.

The parent/teacher can download the system via the internet onto the family laptop or children laptop and login into the admin panel using the Chrome extension to set the settings to the web browser on the child's computer. Once the parent is signed up to the system, they can access any settings and change the settings for the web browser from the admin panel in the Chrome extension.

On the first install, the parent will be asked to signup or login to their account using email and password combination.
After this, the parent can easily see what the child is viewing in real time and set black and white listed sites and block other browsers from being opened.
The parent can see if the child tried to access a high profanity site or a blacklisted website and can reset the color theme on the web browser back to the child selection.

On the child's laptop, the web browser will look and feel the same as other browsers regarding layouts. When the child tries to access blocked information, they will be redirected to Google's homepage. They color of the browser will change to black. The child will learn that was a bad choice. The color will stay black until the parent resets it.

A major feature is that the parent will be able to disable internet access on the browser. The admin can disable the child's internet access by clicking a button and re-clicking to enable the internet access from the Google Chrome extension. They will see a grayed out screen and unable to do anything in the browser.

The algorithm for filtering negative and positive sites used for the application is based on the Naive Bayes Classifier and developed into a node module using node.js. The node module will scrap and parse words on the web page and check if each word is classified as profanity or not. It also saves any new negative words to firebase as it is used for training sets for the Naive Bayes Classifier. So as each person uses the web browser, it gets more words for training. Training involves giving the Naive Bayes Classifier negative and positive words. So when a child goes on a new site, it will use the training data to determine to block the website of not and save these results to firebase.

Project Drivers

The purpose of the project

Create an application for creating and improving on existing filtering algorithms and allowing the parent to view what their children is on using the cross-platform browser.

Web Browser

Upon initially installing, the parent will have to input their email. This is for firebase so the application can sync information from the database using the google chrome extension and link the account to the browser.

The child can open the app and browse the internet which includes going back and forward in their internet history, refreshing, stopping the page from loading, going home to your homepage and having multiple tabs for easy tasks.

The child can change the browser theme to their favorite color. Their information is saved. The web browser will sync every 1 min or on load for new settings for blocked websites from the Firebase database. The system will sync the information of the current URL into the firebase.

When a child tries to access a blocked website or URL they will be redirected to the homepage and the color/ theme of the web browser will be changed to black and cannot be changed back until the admin(parent) resets it from the google chrome extension.

It will block other browsers from opening if the setting is checked in the chrome extension.

Browser features: Refresh page, Create new tab, View tabs, Go Forward, Go Back, Go Home, Search the web, Change the theme

Chrome Extension

On the first install, the parent will be asked to signup or login to their account using email and password combination.
After this, the parent can easily see what the child is viewing in real time and set black and white listed sites and block other browsers from being opened.
The parent can see if the child tried to access a high profanity site or a blacklisted website and can reset the color theme on the web browser back to the child selection.

Google Chrome Extension: Set Blacklist sites, Set Whitelist Sites, Personal settings, On the Fly smart understanding of what to ban, See what your children are on, Stop access to the web, Alert you when your child goes on banned site.

Overall Project Goal

The objective of the application is to develop a web browser with Node.js Webkit with a built in filtering system to block websites/content that the parent doesn't want the child to access while, allowing them to monitor them from a Chrome extension. I will also allow for settings to be synced from one machine to another.

There are two main parts to this application the web browser and the Google Chrome extension which is for admins to monitor over there children who use the standalone browser. The filtering system learns and collects data used for negative and positive words for training. It begins to understand which website to block based on the text. The filtering system is based around the Naive Bayes Classifier and classifies a page as negative or positive.

The Stakeholders

Admin
A person who takes part in an undertaking with another or others, especially in business or firm with shared risks and profits, in this case, the parents or teachers.

Child
A young human being below the age of puberty or the legal age of majority.

System Designers & Developers
Group of persons involved in design and implementation of:

- User Interfaces
- Backend code
- Database structure
- Testing

The individuals need to possess knowledge of web app usability evaluation, user experience aspects; technologies used to develop interaction between user and background mechanisms and also be able to create data structures that support required functionality. Additionally, the ability of test performing will be included. Any conflicts or major solution issues was addressed during ad-hoc meetings of stakeholders involved.

Project Objectives

- Create a web browser using node.js + Webkit
- Develop it for cross-platform devices (mac and windows).
- Block websites based on the terms the parent inputs from the Google extension.
- Disable other browsers from running.
- Develop essential features of a web browser.
- Redirect the child to the homepage.
- Save all settings to firebase, so they can sync settings to different machines.
- Change the theme of the software to dark.
- Learn to develop a system to stop other software from running.
- Create a google chrome add-on for the teacher/parents to set settings which will sync to the browser.
- Disable/Enabled web browser access.
- Build a system to scrap website text and process to the Firebase database.
- Build code to check if a word is classified as profanity and process it to the Firebase database.
- Build code to try and understand the context the page using machine learning algorithms to build a filtering system.

Project Tools

The project tools used in the application for the web browser and the Google Chrome extension are the follows:

- Gulp
- Node.js
- Github
- Node-Webkit (NW.js)
- JavaScript
- Angular.js
- Html5
- NPR
- Css3
- Less
- JavaScript 5
- Photoshop
- Material Design
- Firebase
- Teamwork Projects
- NPM

Learning Outcomes

The application is now completed. The learning outcomes finished and included the developing of web browser and filtering system.

- Learn and discover how to allow for quick cross sync of information for a database with speed in mind.
- Leart how to use Node-Webkit to the fullest.
- Develop and Publish a Chrome Extension.
- Develop and maintain the a cross-platform application (Mac and Windows)
- Learn how to create a rendering engine using Chromium (Chrome).
- Develop an algorithm to help with what should be blocked with a based on the percentage rating
- Build a node module for parsing texts
- Build a node module for checking verbs
- Build a node module for checking text profanity,
- Code up to the web standards on strict settings.
- Learn and develop in Angular.js.
- Learn different APIs aka firebase.
- Follow the design standards of Material Design + visual design principles of a web browser.
- Understand develop and publish natural language based on the algorithm.
- Understand how to compile Chrome.
- Learn the basics of Algorithms and data structures.
- Develop tools to understand sentences and web pages.

Literature review

Below is a brief history and facts about the history of filtering systems.

History of Filtering Systems

For nearly ten years, the issue of Internet filtering has employed legislators, educators, advocates, study committees, and courts. Despite the well-documented problem of over-blocking, filters are now generally used in schools, libraries and households. In the case of helping the public understand the issue, the application has addressed major issues in this report on over blocking, in particular, how to improve on this issue.

In the late 1990s, rating and filtering systems were developed in answer to concerns about pornography and other questionable material on the Internet. Companies began selling the software to schools and libraries in the states.

The Clinton Administration encouraged filtering as a response to a 1997 Supreme Court decision striking down the Communications Decency Act, which, in an effort to block minors from Internet pornography, criminalized essentially all "indecent" or "patently offensive" communications online.

The over-blocking tendencies of Internet filters soon grew known. With a rapidly expanding Web (approaching a billion sites by the early] 2000), screens relied on ‘keywords’ and phrases to identify sites that might be thought inappropriate for minors.

Groups such as Peace Fire and The Censorware Project began documenting the problem of false blocking, with examples covering from information on breast cancer to the Website of Congressman Dick Armey.

In 1998, Congress asked the National Research Council (part of the National Academy of Sciences) to lead a study on "Tools and Strategies for Protecting Kids from Pornography and Their Applicability to Other Inappropriate Internet Content." The NRC organized a committee that held hearings and conducted comprehensive research.

In May 2002, the NRC released a 402-page report, Youth, Pornography, and the Internet, which noted that since filters rely "on machine-executable rules abstracted from human judgments," they necessarily identify "a large volume of appropriate material as inappropriate.”

Conflicts of Filtering Systems

At the start of the new millennium, some major filter makers claimed to have fixed the problem of over-blocking and to have broken reliance on keywords for "artificial intelligence." "Artificial intelligence," however, is simply a more revealing form of keyword blocking. Investigations and reports continued to document the erroneous blocking of thousands of educational sites, in particular of sex education sites.

Along with over-blocking, some filtering software also under-blocks - that is, they fail to distinguish and block many pornographic sites. Filters originally operate by searching the World Wide Web, or "harvesting," for possibly inappropriate sites, mostly relying on keywords and phrases. There happens a process of "winnowing," which also relies primarily on these mechanical techniques.

Most filtering companies also use some form of human review. But because 30,000 - 50,000 new Web pages enter the "work queue" each day, the company's' relatively short staffs can give at most a cursory review to a fraction of these sites, and human error is determined.

Filtering company employees' judgments are also fundamentally subjective, and reflect their employer's' social and political views. Some filtering systems reflect traditional religious views. Filters often block all pages on a site, no matter how innocent, based on a "root URL."

Likewise, one item of disapproved content often results in blockage of the entire site. For example, a sexuality column on Rte.ie 12 or schools in New York City in 1999, where filters barred students studying the Middle Ages from Web sites about medieval weaponry, including the American Museum of Natural History; and other informative sites such as Planned Parenthood, CNN, and sites discussing anorexia and bulimia.

Summary

From the above research, it is evident that filters operate by censoring large amounts of expression in advance rather than correcting unlawful speech after the fact. Their poor judgment can be mainly credited to the fact that they do not follow the Children's Internet Protection Act but rather filter by their systems and databases.

A huge flaw in the filtering and software engineering is that it will never fully reflect a reductive view of human expression, i.e. the human brain would never reduce context and its value and meaning to decontextualized keywords and phrases or broad subject-matter labels (e.g., "violence," "drugs," "alternative lifestyles"), the processes which result in the false positive of blocking sex education websites.

One possible method of fixing this, would be to rate the page based on rating system where amount of words on a page is noted and each word is assessed for profanity based on the criteria set by the Children's Internet Protection Act, returning then a percentage of profanity words, as most adult themed sites use more profanity than that of a regular site.

This could allow filtering software to advance from the current set up, whereby filters set barriers and taboos rather than training youth about media literacy and sexual values, along with frustrating and restricting research into health, science, politics, arts, and many other educational areas.

Angular vs Knockout.js

Angular.JS and Knockouts are JavaScript libraries/frameworks that help create healthy and responsive web UI interactions.
Knockouts' is a library that connects parts of the UI to a data model using declarative bindings. For the most part for AngularJS is the same, which is where the confusion comes from.

The core difference between the two frameworks is that AngularJS controls the entire application and establishes guidelines on how the application code should be structured while with KnockoutJS the application structure is entirely up to you

When comparing any frontend framework, it's necessary to look at the most import factors, when comparing both Knockout and Angular.js which are the two most popular JavaScript frameworks. It's important to consider the following areas Data Binding, Templating, Extensibility, Variable Observation, Routing, Browser Support, Ecosystem and Other Features.

Data Binding

Data Binding is the method of establishing a relationship between the UI and the logic. If the settings are configured correctly, the data reflects changes made in the UI. This also means changes in the data are represent in the UI.

HTML syntax for Knockout. Everything is arranged using the data-bind attribute and appropriate binding type. The need to specify all properties as observable requires additional effort. Mappings need to be done when loading JSON data from the server to convert properties to observables. It is the developers responsibility to maintain the mapping when retrieving data and when sending data back to the server.

The angular syntax for outputting values is simpler and more compact. It makes it easier to read and understand. The major difference is the binding setting. Where Knockout binds to the given model, Angular binds to the special $scopeobject.

Additionally, Knockout can only apply bindings once. If you try to use bindings again, it will throw an error. In Angular, yet, scopes can be nested. This way makes Angular views and controllers independent and reusable, and as a result, they can be tested independently.

Routing

Routing is a great feature that allows control of the application states. Such as back/forward in browser history navigation. Knockout does not support routing.

AngularUI Router is a routing framework for Angular.JS, which enables you to assemble the parts of your interface into a casing machine. Unlike the $route setting in the Angular ngRoute module, which is created around URL routes, UI-Router is organized around states.

States are bound to name, nested and parallel views enabling you to manage the application's interfaces. States can be nested within each other.

Browser Support

KnockoutJS:

- Mozilla Firefox (versions 3.5 - current)
- Google Chrome (current)
- Microsoft Internet Explorer (versions 6 - 11)
- Apple Safari for Mac OS (current)
- Apple Safari for iOS (versions 6 - 8)
- Opera (current version)

AngularJS

- Safari, Chrome, Firefox, Opera, IE9 and mobile browsers (Android, Chrome Mobile, iOS Safari)
- Versions 1.2 and later of AngularJS do not support Internet Explorer versions 6 or 7
- Versions 1.3 and later of AngularJS drop support for Internet Explorer 8

Ecosystem

Angular is more widely adopted with a broader user base.
The table below shows ecosystem statistics:

	Knockout	Angular
GitHub Stars	7,587	30,567
GitHub Forks	1,275	11,617
Stack Overflow Questions	21,126	60,119

Other Features.

Knockout can only compare to a small part of Angular features. Angular is a more extensive framework, and includes:

Modules: a module is a container for a set of components. Those components can be controllers, services, filters, directives, etc. It allows you to package code into reusable components. Modules can reference other modules. They serve as application building blocks.

Services: a service can be used to organize and share code across your app. Each service is a singleton, and all components reference a single service instance. Angular contains some built-in services including:

- **$http:** used to make AJAX requests to the remote servers
- **$q:** promise/deferred implementation inspired by Kris Koala's Q
- **$log:** service for logging. By default, writes to browser console if present

Scopes: arranged in hierarchical structure which mimics the DOM structure of the application, Scopes can watch expressions and propagate events.

Filters: A filter formats the value of an expression for display to the user. They can be used in view templates, controllers or services. It is easy to define your filter. Filters can be applied to expressions in the light templates using the following syntax.

Form Validation: form and controls provide validation services so that the user can be notified of invalid input.

Internationalisation and Localisation: Working with Knockout, this functionality is not possible in the core library. Can be added by using external libraries or custom logic.

Summary

Knockout has a low barrier to entry but is also harder to maintain when code base and complexity grows. It is not easy to build the required infrastructure correctly, and poor choicesmade in structuring code may cost a lot to fix in the future.

Angular's capacity to bind straight to plain objects, modular structure and strict development guidelines prevent many issues right from the start and give a strong architectural framework for the application.

Knockout is essentially used to control UI design in lower complexity applications, whereas Angular is a JavaScript framework that is greatly better suited for large, complicated enterprise applications. It gives not only UI binding but also best practices for application structure, development, and testing which is why I choose it for my application.

Machine Learning Algorithms

Below is the selection of different machine learning and some natural learning algorithms and the uses for all and the possible idea used for filtering systems

Machine learning algorithms are organised into a taxonomy, based on the desired outcome of the algorithm. Typical algorithm types include:

Supervised learning: where the algorithm generates a function that maps inputs to desired outputs. One standard formulation of the supervised learning task is the classification problem: the learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input-output examples of the function.

Unsupervised learning: which models a set of inputs.

Semi-supervised learning: which combines both labeled and unlabeled examples to generate an appropriate function or classifier.

Reinforcement learning: where the algorithm learns a policy of how to act given an observation of the world. Every action has some impact on the environment, and the environment provides feedback that guides the learning algorithm.

Transduction: similar to supervised learning, but does not explicitly construct a function: instead, tries to predict new outputs based on training inputs, training outputs, and new inputs.

Learning to learn: where the algorithm learns its inductive bias based on previous experience.

The performance and computational analysis of machine learning algorithms is a branch of statistics known as computational learning theory.

Machine learning is about designing algorithms that allow a computer to learn. Learning does not necessarily involve consciousness but rather it is a matter of finding statistical regularities or other patterns in the data. Thus, many machines learning algorithms will barely resemble how human might approach a learning task. However, learning algorithms can give insight into the relative difficulty of learning in different environments.

Which brings us to the Natural Language, understanding is a very hard problem, and many researchers are working on it.

To begin with, the application uses a key rule-based algorithm and the Baylor classifier algorithm.The application will set rules that will be matched against new text, and if a match is found, you fire a corresponding action, in this case, checking for profanity and nouns and if matches with the negative side it will block the website and alert the parent.

Using this should help not to restrict the format of the users' input and to come up with rules which while remaining as general as possible.

For example, instead of blocking the word "murder," which can have a double meaning (i.e. a murder of crow's vs to murder somebody), you can have a rule such as: unless the word "murder" occurs in the command, don't start the program, OR ignore every sentence unless it contains "murder" which is counted as profanity. Then, the application combines the set of rules to develop more complex "understanding".

Therefore, the applications rules are based on profanity classed words and the use of verbs in the sentence along with the Baylor classier algorithm.

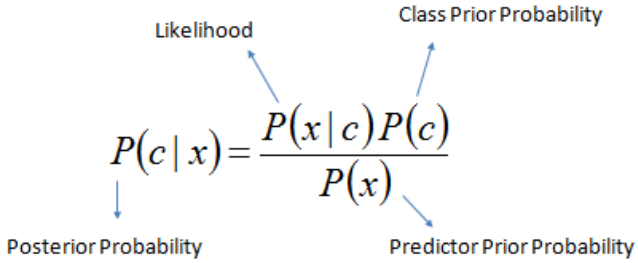
A meeting with Dr. Ted Scully explained that the theory was semi-correct, but I would be better looking into the Naive Baylor classifier as it is commonly used for this and that with the use of the firebase parsing information mine storing it will have an enormous great set of example training data.

So in the applications use of the algorithm, it will use negative and positive words to train the system. Negative words being cruise words and those found on porn and adult sites. When visiting a random site ti will compare the new text with the actual words and classify the page as positive of negative.

The advantage of using the Naive Bayes classifier are the following

Advantages: Super simple, you're just doing a bunch of counts. If the NB conditional independence assumption holds, a Naive Bayes classifier will gather quicker than discriminative models like logistic regression, so you need less training data. And even if the NB theory doesn't hold, an NB classifier still often does a great job in training. A good bet if want something fast and easy that performs pretty well.

Disadvantages: Its main disadvantage is that it can't learn interactions between features (e.g., it can't learn that although you love music with Bona and Greenday, you hate movies where they're together).



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Nw.js Vs Election

If you wish to create a native desktop application from web technologies, the open source world offers two main choices: NW.js (formerly node-Webkit) and Electron (once atom-shell). Deciding which one to go with is not so obvious. That is precisely why the below comparison chart quickly shows why the application was developed using Nw.js over election.

Nw.js: node-Webkit is an app runtime based on Chromium and node.js. You can write native apps in HTML and JavaScript with node-Webkit. It also lets you call Node.js modules directly from the DOM and enables a new way of writing native applications with all Web technologies. It's created and developed in the Intel Open Source Technology Center.

Election: The Electron framework lets you write cross-platform desktop applications using JavaScript, HTML, and CSS. It is based on Node.js and Chromium and is utilized in the Atom editor.

Features List

	NW.js 0.12.3	Electron 0.34.1
Project inception	2011	2014
Sponsor	Intel	GitHub
Platform Support	Mac, Linux & Windows	
Browser Runtime	Chromiumhttps://www.chromium.org/Home	libchromiumcontenthttps://github.com/atom/libchromium-content
Layout Engine	Blink / Webkit 537	Webkit 537
JavaScript Engine	V8https://code.google.com/p/v8/	
Node.js Engine	io.js v1.2.0	io.js v?
ES6/2015 Support	Yes (all from V8 v4.1)	Yes (all from V8 v4.4)
Chrome-Equivalent Version	41	44
Development Model	Open Source	
Licensing	MIT License	
Entry Point	HTML or JavaScript5	JavaScript
Bare Distribution Size	75MB – 100MB4 (30MB – 34MB zipped)	To confirm. Anybody ?
Chrome Apps Support	Yes (in beta)	No

	NW.js 0.12.3	Electron 0.34.1
Support of chrome.* APIs	Yes (in beta)	No
Adobe Flash Support	Full NPAPI Plugin	Pepper Plugin
Mac App Store Support	Yes	
Windows App Store Support	Yes	?
Support for Windows XP	Yes	No
Source Code Protection	V8 Snapshot1	ASAR Archive Support2
Auto-update	Mac/Linux/Win (module)	Mac/Win (thru Squirrel)
Kiosk Mode	Partial (Buggy on Mac6)	
Windows Installer	Through nw-builderhttps://github.com/nwjs/nw-builder	Yes (external module)
html5test.com Score3	515	520
Browser Mark 2.1 Score3	5294	5643
Octane 2.0 Score3	27619	28346
GitHub Trends		
Open Codecs/Containers	Vorbis, Theora, Opus, VP8, VP9, PCM, Ogg, WebM, WAV	
Licensed Codecs: MP3, MP4, H.264, AAC	Yes (with some effort)	Yes

Summary

It was hard to draw a conclusion at this stage, but most of the comments received so far seem to favour NW.js. Some of the reasons cited to choose NW.js are the longer track record, overall development philosophy, cross-platform auto-updater, and Windows XP support.

Competition in the Market

Below is the section of popular filtering systems, the benefits of each.

Introduction

The following text describes a competitive analysis of the Internet Filter Software, in particular, Net Nanny, Spy Agent and Qustodio Important in this analysis are the installation, DE installation, update, filtering, etc.

Note: Tests were performed on an Apple Mac 2015 with 16GB of RAM and an Intel Core-i5 CPU.

Spy Agent

Spytech SpyAgent is our award winning, powerful computer spy software that allows you to monitor EVERYTHING users do on your computer - in total stealth. SpyAgent provides a large array of essential computer monitoring features, as well as the website, application, and chat client blocking, logging scheduling, and remote delivery of logs via email or FTP. SpyAgent will put your mind at ease with its innovative and unmatched, yet easy to use feature-set that provides the ultimate all-in-one computer monitoring software package.

Net Nanny

Net Nanny is a brand of content-control software marketed primarily towards parents as a way to control a child's computer activity. The flagship product allows a computer owner to block and filter Internet content, place time limits on use, and block desktop PC games

Qustodio

is a brand of content-control software marketed primarily towards parents as a way to control a child's computer activity.

Features

Title	SpyAgent	Net Nanny	Qustodio
Website Monitoring	YES	YES	YES
URL Based Website Blocking	yes	YES	YES
Content / Category Based Website Blocking	YES	YES	YES
Social Media Monitoring	YES	NO	YES
Search History Monitoring	YES	YES	YES
Chat/IM Recording	YES	YES	YES
Email Recording	YES	NO	NO
Email Attachment Recording	YES	NO	NO
Software Keylogger	YES	YES	NO
Automatic Screenshots	YES	NO	YES
Program Activity Monitoring	YES	YES	YES
Application Stealth/ Invisibility	YES	NO	NO
Remote Monitoring	YES	NO	NO
Website Whitelisting	NO	YES	YES
Enforce Program Time Limits	NO	NO	NO
More Expensive than Some Competitors	NO	YES	NO

Summary

Some performed extremely well is certain areas but lacked the diversity of features necessary to carry out a thorough job of keeping children safe online. For example, although Net Nanny ranked high on the list for offering the greater number of features, the conventional software program provides only basic monitoring for social media websites. If you want a more expansive view of what your child is doing across a variety of social networks including Twitter and Google +, you have to pay for Net Nanny Social.

Net Nanny gains points for having the most comprehensive profanity filter available, including the ability to perform profanity masking. All others take the approach of fully blocking sites. Which may or may not be the method you want to use.

Picking an internet filter software application depends hugely on what is ideal for you. Although I found Net Nanny to be the overall best, they can be annoying and can interfere with the internet usage of the whole family, which isn't the point.

Firebase Vs MySql

Firebase can power any app's backend, including data storage, user authentication and more. They provide these services so you can focus on creating extraordinary user experiences.

Data in Firebase is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with Firebase, all of your customers share one Firebase database and automatically receive updates with the newest data.

Automatically scales with your app
The great thing about Firebase is you don't need to worry about scaling your server code. Firebase handles that automatically for you. Firebase servers manage millions of concurrent connections and billions of operations per month.

First-class security features
All of the data is transferred over a secure SSL connection with a 2048-bit certificate. The database access and validation are controlled by using Firebase's flexible security rules language. All of your data security logic is centralised in one place making it easy to update and verify.

Works offline
The Firebase database will remain responsive regardless of internet connectivity. All writes to a Firebase database will trigger local events immediately, before any data has been written to the server. Once connectivity is reestablished, the client will receive any changes it missed, synchronizing it with the current server state.

Summary

If you're going to be linking to something such as the web or mobile application where the data is constantly growing by various users (all accessing the same database saved in the cloud). Firebase is the way to go. The major reason for the choice of firebase over mysql is cause of the cross platform api which allows for the development on more devices.

Pros

- It's capable of handling the Real-Time data updates between devices.
- Stored in the cloud so readily available everywhere and scaleable.
- Cross-Platform API
- encryption
- They host the data.

Cons

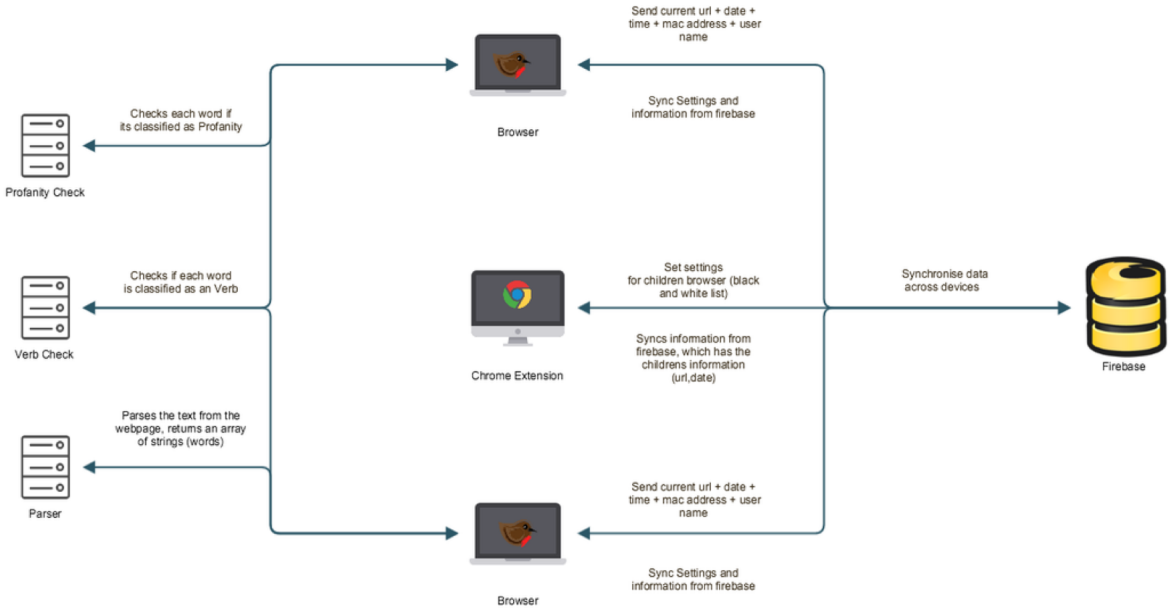
- Unless your app runs on one centralised database updated by a vast quantity of users, it's a major overkill.
- The storage format is entirely different to that of SQL, (Firebase uses JSON) so you wouldn't be able to migrate that quickly.
- Reporting tools aren’t near the ones of standard SQL.
- Limited to 50 Connections and 100mb of Storage (free version).
- You don't host the data, Firebase does. Depending on which server you get put on, viewing their up time there seems to be a lot of disruption lately.

ANALYSIS AND DESIGN

The Scope of the Work

This application will automate existing business processes for accessing the internet. It will provide the self contained remote access to procedures necessary for blocking and viewing internet filtering and history on children's pcs. It will integrate external sources to enhance user experience.

Context Diagram



Business Use Cases

Children should be able to

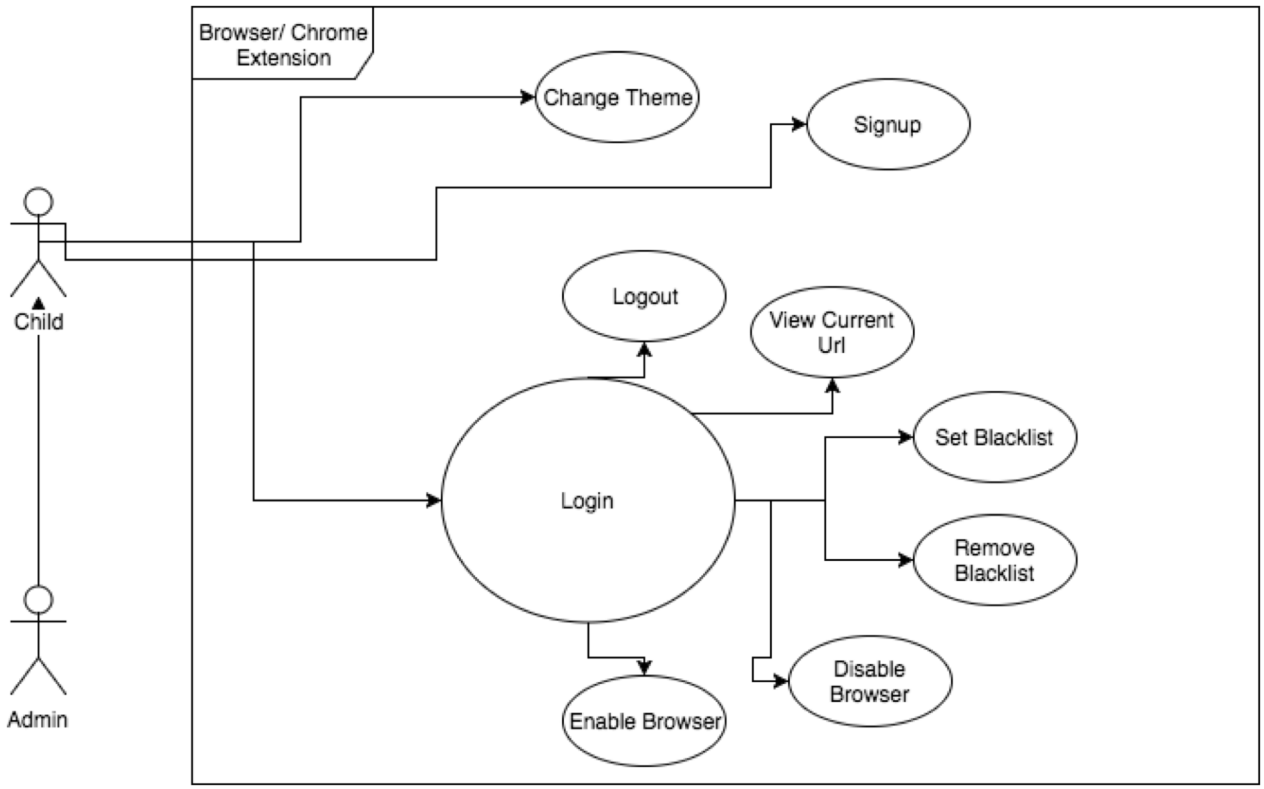
Browse the web
Go Back in history
Go Forward
Refresh Page
Receive points for accessing good websites set by administrators.
Change Color of Browser

Teachers / Parents should be able to do above and also from the chrome extension

Sign into Browser on set up.
Set settings for filtering system
Unblock filtering when the child accesses banned websites.
See what site the child is currently on.
Set sites which the child will be rewarded for visiting

Business Data Model

Use case diagram



Use Cases

Use Case #1	Sign up
Actor	Admin
Precondition	Must have email address
Postcondition	Admins email is now set up to sync information from chrome app
Main Path	Person inputs required information (email System verifies information System creates user account System is now ready to sync settings from cloud
Alternative Path 1	@2. System verifies user already exists System displays message and allows user to reenter details or switch to log in page @2. User input is incomplete System displays message and allows user to continue
Exception	@2. System rejects user sign-up if user does not provide valid email address

Use Case #2	Login
Actor	Admin
Precondition	Must be a member
Postcondition	They are now logged into Chrome Extension
Main Path	User inputs his email address and password into login form. System checks if they match System brings them to the account dashboard on chrome app.
Alternative Path 1	@2. User inputs incorrect email address or password details System display that the user information is incorrect, and allows for re entering information
Alternative Path 2	@2. User inputs not existing email System alerts user that there is no account registered

Use Case #3	Add Whitelist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System now knows what a good website is
Main Path	User selects Good websites ’. System brings up ‘Good website ’ page. User selects add good website System brings up a form. User inputs website url User selects ‘Submit’. System adds info to database. System updates ‘Good Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #4	Remove Whitelist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System removes good website is
Main Path	• User selects Good websites ’. • System brings up ‘Good website ’ page. • User selects good website from list • User selects remove • User selects ‘Submit’. • System removes info to database. • System updates ‘Good Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #5	Update Whitelist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System now knows what a good website is
Main Path	• User selects Good websites ’. • System brings up ‘Good website ’ page. • User selects update good website • System brings up a form. • User inputs website url • User selects ‘Submit’. • System adds info to database. • System updates ‘Good Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #6	Add Blacklist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System now knows what a bad website is
Main Path	• User selects Bad websites ’. • System brings up Bad website ’ page. • User selects add Bad website • System brings up a form. • User inputs website url • User selects ‘Submit’. • System adds info to database. • System updates Bad Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #7	Remove Blacklist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System removes Bad website is
Main Path	• User selects Bad websites ’. • System brings up Bad website ’ page. • User selects bad website from list • User selects remove • User selects ‘Submit’. • System removes info to database. • System updates Bad Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #8	Update Blacklist
Actor	Admin
Precondition	Is a registered member.
Postcondition	System now knows what a bad website is
Main Path	• User selects Bad websites ’. • System brings up Bad website ’ page. • User selects update Bad website • System brings up a form. • User inputs website url • User selects ‘Submit’. • System adds info to database. • System updates Bad Website List’ page.
Alternative Path 1	@8. User leaves a field blank and selects ‘Submit’. System highlights the blank field.

Use Case #9	Change Theme / Color .
Actor	Child or Admin
Precondition	
Postcondition	Browser has a new theme or Color
Main Path	1. User selects ‘Settings’. 2. System brings up settings page. 3. User selects prefered color / theme. 4. System changes browser color / theme
Alternative Path 1	

Use Case #10	View current urls of children actively.
Actor	Admin
Precondition	Is a registered member.
Postcondition	Current Urls is displayed
Main Path	1. User clicks settings tab 2. System bring up settings tab 3. User clicks current URLs on Child's browser 4. System displays schedule information
Alternative Path 1	

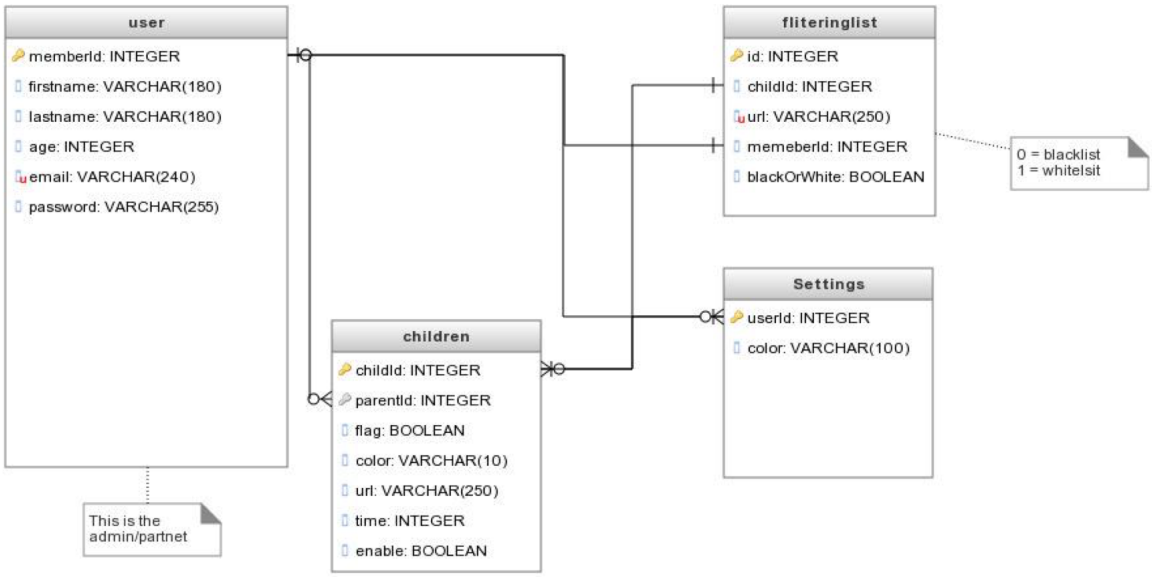
Use Case #11	Disable Browser
Actor	Admin
Precondition	Is a registered member.
Postcondition	System disables internet access on the childs browser
Main Path	• User selects the child '. • User clicks disable internet button • System adds info to database. • System disables internet.
Alternative Path 1	

Use Case #13	Enable Browser
Actor	Admin
Precondition	Is a registered member.
Postcondition	System disables internet access on the childs browser
Main Path	• User selects the child '. • User clicks enables internet button • System updates to database. • System enables internet.
Alternative Path 1	

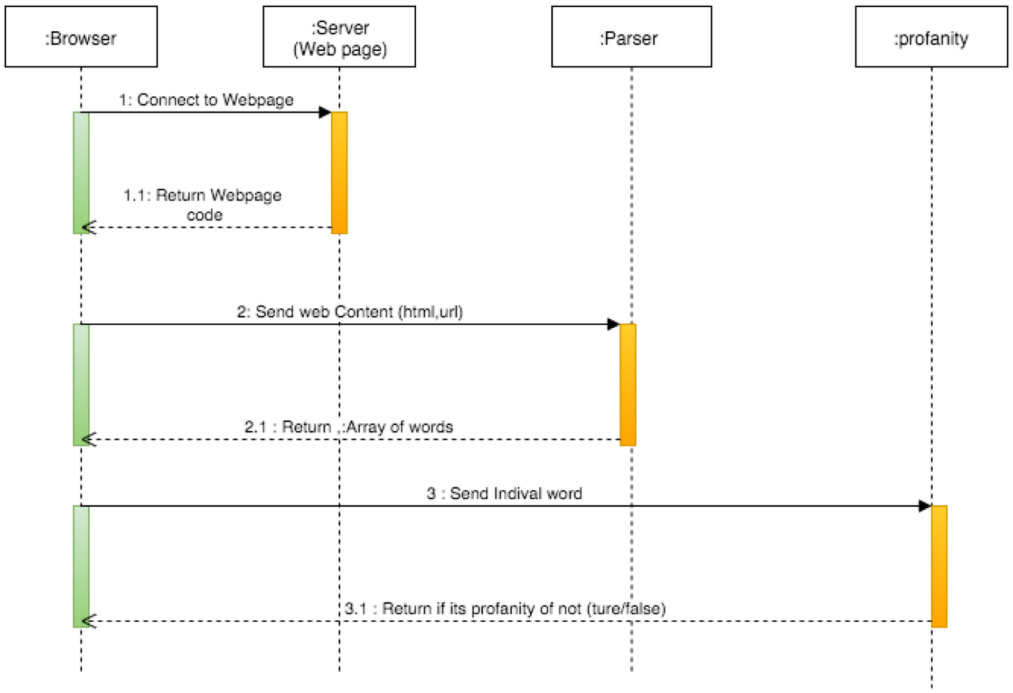
Use Case #14	Child Views banned or high profanity website
Actor	Child
Precondition	Has internet connect
Postcondition	The theme/color of website is now black
Main Path	• User ties to visit a pornographic site • System brings up Checks for a profanity rating • System changes theme to black • System redirect user to homepage url • System alerts the admin though google chrome app
Alternative Path 1	@2 System gets a low profanity and lets them visits

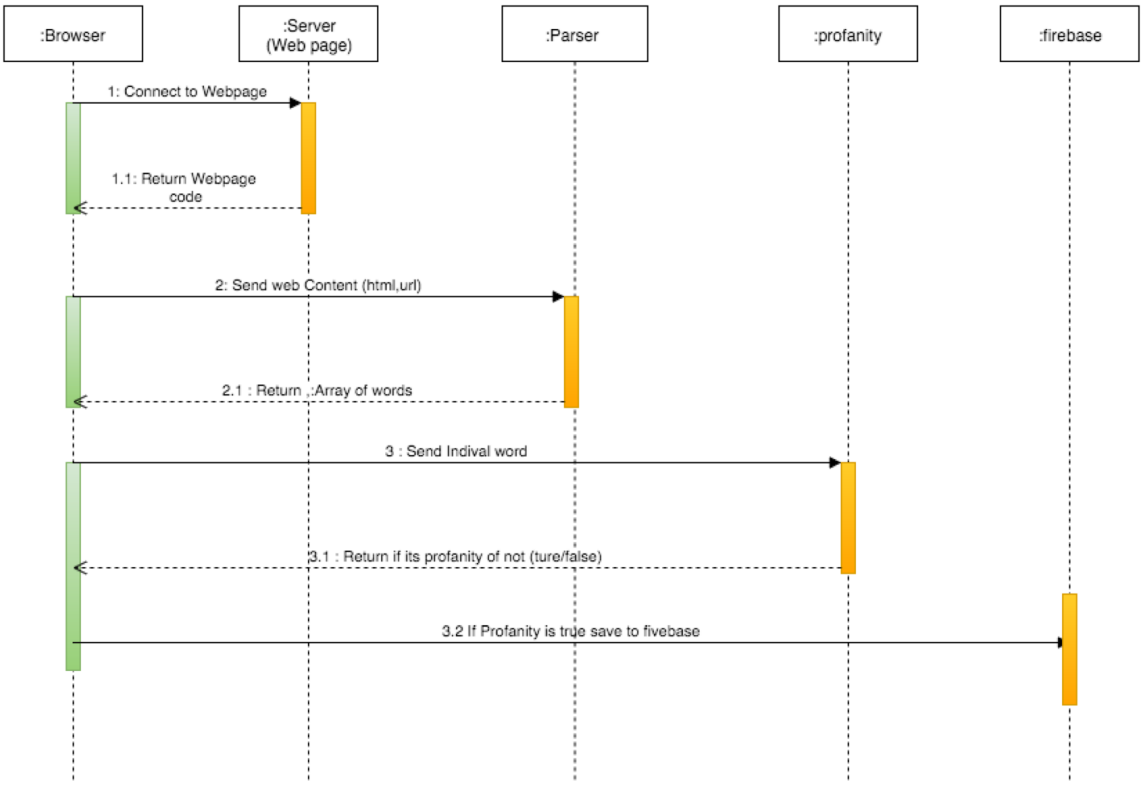
Use Case #15	Logout
Actor	Admin
Precondition	Must be a member
Postcondition	They are now logged out
Main Path	User clicks logout button User inputs his password into login form. System checks if they match The system logs them out
Alternative Path 1	@2. User inputs incorrect password details System display that the user information is incorrect, and allows for re entering information
Alternative Path 2	@2. User inputs not existing email System alerts user that there is no account registered

Data Model

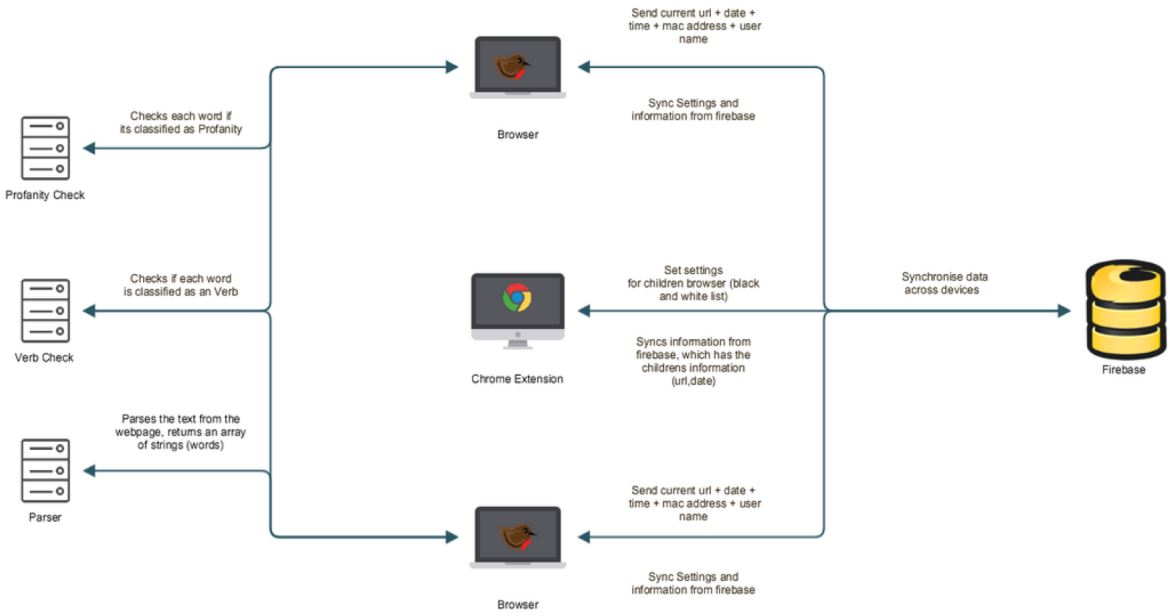


Sequence diagrams





Architecture diagrams



Functional Requirements

Requirement	#1
Description:	The application runs profanity checks on all webpages (not websites)
Rationale:	To search the web
Originator:	Admins and Children
Fit Criteria:	Decides if the page should be blocked and store the information line.
Dependencies:	None
Priority:	Very high
History:	Created: 10/10/2015

Requirement	#2
Description:	Admins should be able to view their children's current URL.
Rationale:	This has a goal of recording and tracking children's web use so that the admin can take action and supervise it.
Originator:	Members
Fit Criteria:	The results shall be displayed in time progress chart on chrome extension.
Dependencies: Priority:	Low
History:	Created: 10/10/2015

Requirement	#3
Description:	Member should be able to change his profile details/settings.
Rationale:	This is part of customization provided for members to ensure individual preferences and also to take into account any change in the member’s recorded personal details.
Originator:	Common stakeholders agreement.
Fit Criteria:	This brings a value from usability point of view.
Dependencies:	Req#1
Priority:	Medium
History:	Created: 11/10/2015

Requirement	# 4
Description:	The application should record member details and history
Rationale:	To be able to login and manage the children's history
Originator:	Admins
Fit Criteria:	Created account should be instantly ready to use
Dependencies:	None
Priority:	Very high
History:	Created: 10/10/2015

Requirement	# 5
Description:	The application should block and redirect to url
Rationale:	When a site is counted as bad and blocked it should redirect them to google
Originator:	Admins
Fit Criteria:	Created account should be instantly ready to use
Dependencies:	None
Priority:	Very high
History:	Created: 10/10/2015

Requirement	# 6
Description:	Change browser color to black when accessed block website
Rationale:	When a site is counted as bad it will change the theme to black and alert the user.
Originator:	Admins
Fit Criteria:	Created account should be instantly ready to use
Dependencies:	None
Priority:	Very high
History:	Created: 10/10/2015

Requirement	# 7
Description:	The application should allow for viewing all children's details by admin user.
Rationale:	As an admin, he must be able to have an access to all children account's details.
Originator:	Admin
Fit Criteria:	The application will have to identify the admin user when he logs in and display appropriate menu options.
Dependencies:	Req#1
Priority:	High
History:	Created: 11/10/2015

Requirement	#8
Description:	Admins should be able to enable/disable the childs browser
Rationale:	So the child only has access the internet when the parent wants
Originator:	Admin
Fit Criteria:	The results shall been shown in the admins panel.
Dependencies:	none
Priority:	Low
History:	Created: 10/10/2015

Requirement	#9
Description:	Sync Information from account to account
Rationale:	So the information is passed when logged in
Originator:	Admin, child
Fit Criteria:	The information is saved and displayed on devices.
Dependencies:	none
Priority:	Low
History:	Created: 10/10/2015

Requirement	#10
Description:	Users should be able to do basic web browser functions
Rationale:	The user will be able to go back/forword/refresh/home and search
Originator:	Admin
Fit Criteria:	The results are shown in the browser
Dependencies:	none
Priority:	Low
History:	Created: 10/03/2016

Requirement	#11
Description:	Users should be able to manage tabs
Rationale:	The user will be able to go delete and create tabs
Originator:	Admin
Fit Criteria:	New tabs will be created of tabs will be deleted
Dependencies:	none
Priority:	Low
History:	Created: 10/03/2016

Non-functional Requirements

Accessibility

Application future growth would include support for different forms of disability aspects (visual, aural aids).

Speed and latency (Performance)

The interface shall have the maximum response time of 1 second. Within that time user must be presented with the result of his actions or its confirmation.

Reliability and availability

The application is available online 24 hours/day and 365 days/year. A short period can be scheduled for maintenance with prior notice given.

Capacity

The application can serve 10,000 simultaneous users at the moment. The volume of the database should be able to accommodate 1000000 different records per table per minute.

Maintainability and support

Working application should take no to little need for support. Initial setup expects for the database to be created and the admin account set. Post deployment maintainability could provide assistance for any emergency situations or new version release.

Ease of use

The goal of the application is to allow the user to perform his primary tasks and activities within three clicks or less. The navigational labels should be self-explanatory and guide the user through each level of the website. It should also provide accurate feedback to make him confident that application works as expected. When it comes to error handling, it should stop the user from continuing if any occurs by giving proper feedback along with the choice of alternate route.

Learning

The application is ready to be used by anyone who has little technical knowledge of using the internet. Any user can use the core functionality of the application within few minutes and without prior training provided.

Security

The security in the application is the user of transferring data inside firebase, Firebase uses https only for its protocols. Within the settings of Firebase, you can configure almost everything, from validation to the configuring write and read access.

Firebase is secure. It is only as secure as you make it. The component that is missing by default is Firebase Security Rules. Therefore, a way must be provided to write server enforced rules with a language they have named "Security Rules."

Since these rules are on the server, they cannot be edited by the client. Security rules are constructed just like data in Firebase.

In this case, I'm allowing everyone to read from the Firebase while only permitting authenticated users to write access.

```
{   ".read": true}

{

  ".read": true,

  ".write": "auth !=== null", // only authenticated users can write

  "sparks": {

    // location's with $ are wildcards that will apply to all children of

the location

    "$sparkid": {

      ".validate": "newData.hasChildren(['author', 'content'])" // only

post sparks that have author and content nodes

    }
```

So with Firebase, you can write validation rules in the security section of your Firebase that force the shape of the data the user is allowed to put in the database. I've written several public facing apps before that I've also hammered on trying to put in junk data. They give you an easy to configure ruleset to prevent this.

Regarding encryption, Firebase uses crypt. The bcrypt function is the default password hash algorithm for BSD and other systems including some Linux distributions such as SUSE Linux, which firebase is based on.The prefix "$2a$" in a hash string in a shadow password file indicates that hash string is a bcrypt hash in modular crypt format. The rest of the hash string includes the cost parameter, a 128-bit salt (base-64 encoded as 22 characters), and 184 bits of the resulting hash value (base64 encoded as 31 characters).

The bcrypt algorithm depends heavily on its "Eksblowfish" key setup algorithm, which runs as follows:

```
EksBlowfishSetup(cost, salt, key)
  state InitState()
  state ExpandKey(state, salt, key)
  repeat (2cost)
     state ExpandKey(state, 0, key)
     state ExpandKey(state, 0, salt)
  return state
```

Hence, ExpandKey (state, 0, key) is the same as regular Blowfish key schedule since all XORs with the all-zero salt value are ineffectual.ExpandKey (state, 0, salt) is similar, but uses the salt as a 128-bit key.

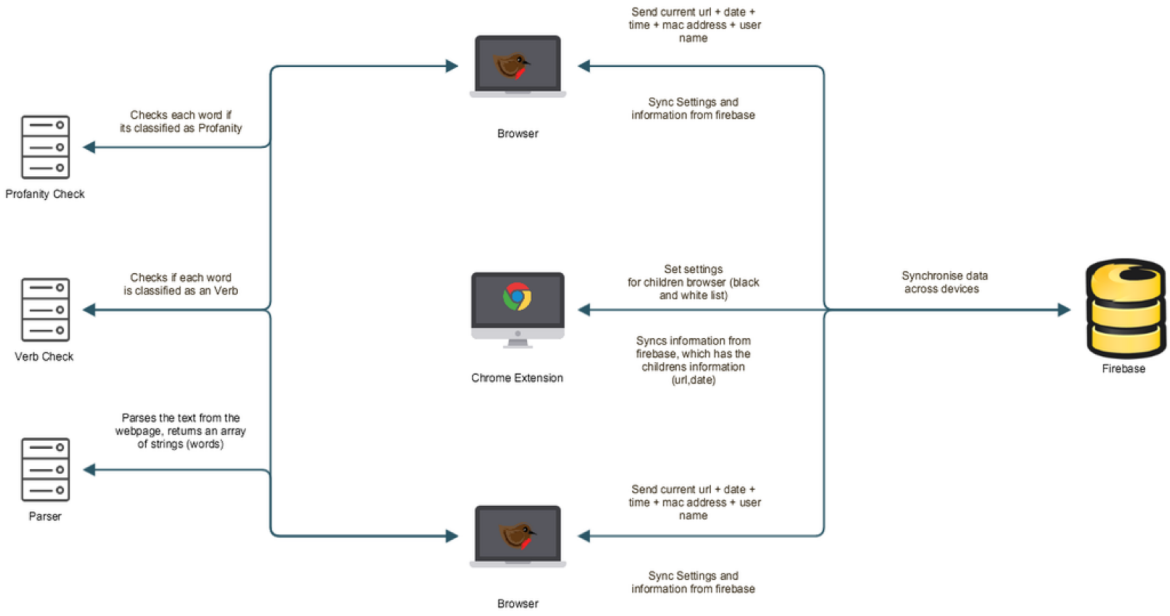
The full bcrypt algorithm utilizes these functions to compute a hash from a given input derived from the password, as follows:

```
bcrypt(cost, salt, input)
  state EksBlowfishSetup(cost, salt, input)
  ctext "OrpheanBeholderScryDoubt" //three 64-bit blocks
  repeat (64)
     ctext EncryptECB(state, ctext) //encrypt using standard Blowfish in ECB mode
  return Concatenate(cost, salt, ctext)
```

Firebase Form and controls present validation services so that the user can be notified of invalid input before submitting a form. This gives a better user experience than server-side validation alone because the user gets instant feedback on how to correct the error. Keep in mind that while client-side validation plays an important role in providing good user experience, it can easily be circumvented and thus can not be trusted. Server-side validation is still necessary for a secure application.

Design and Implementation

System Design



The overall system is designed with the fact and taking advantage of Angulars MVC pattern. MVC is more like a traditional web app style, where you navigate from page to page. With Angular you have only one page and the transition from page to page is seamless. You can think AnglularJS as MVC + Ajax.

The code is spilt into six major packages with there own repos, the web browser, the Chrome extension, parser, verb checker, the profanity checker, the filtering system. With Firebase being the data layer. The reason for the spilt is to allow for it be more maintainability, and if some of them go down it won't affect the rest of the application.

One of the biggest benefits of the of doing this pattern is that it does not need to rewrite the application from scratch. Instead, what you can do is to add new features, and plug them into the existing application and further enhance the filtering system.

System Resilience
As the application is composed of multiple packages, if one of them goes down only some features from the application will go down, not the entire application.

Visual Design: Material design

The overacting design guidelines followed for the design for the web browser and Chrome extension is built following the guidelines set out by Google for material design.

Material provides the context in design, the surface and edge of a "material" provide us visual cues. We know the dimensions of a room because we see walls. The interior provides us an understanding of the meaning of the room. Your kitchen looks very different than your bathroom for example.

The same is applied in Material Design. The mixture of style and content provides connection to the user in a digital space, much like physical walls and interiors. A user has a better understanding of the user interface because the designed material provides context for the interface.

Motion

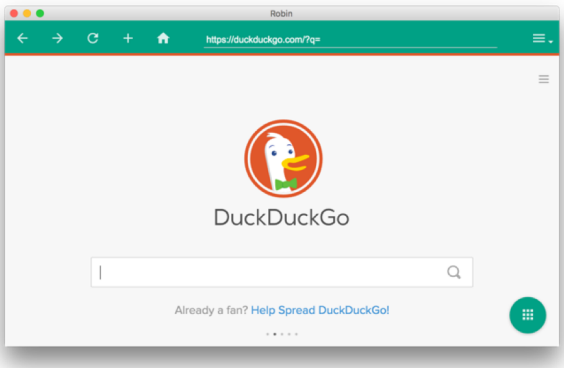
The concept of Motion in Material Design has a similar story. Motion provides the context in design through the flow of an application, appeal when it comes to the continuity of a product; a user has the feeling of being unbroken. There are no obstacles, such as inconsistency in design or a confusing navigation.

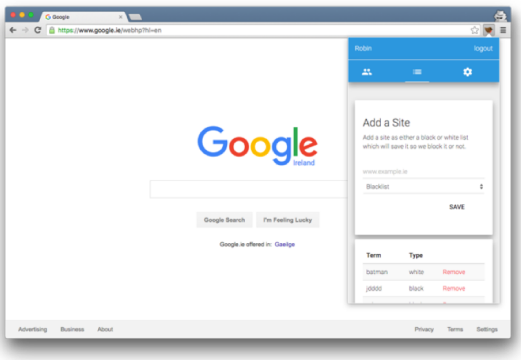
How exactly does motion work? Here's an example. When you tap a single card, the material of the card extends to become the full width and height of the screen instead of the dimensions of a card.

Layout

Designing a layout in Material Design uses some of the core principles of print design, which Google shows as a source of inspiration for Material Design. There's a strong importance on building user interfaces that scale well between different types of devices. As you're informed, scalability has become crucial for designing products that are successful on multiple devices.

Below are the browser and chrome extension. More screenshots can be found in the appendix





Web Browser/ Chrome Extension

It's important to note several things that include, the codebase for both the browser and the Chrome extension share the same code base in a file called core.js. The application was built using node.js, github and code revisions to make sure any unneeded code was removed. In the end, there were 1,694 lines of java-script code, 800 lines of HTML code and 4,000 lines of LESS files. Visit the Github repos for more information and set up guides along with style guides. The code is professional com-mented and easy to follow. Below is some snippets of code that is not standard.

```
{
  "name": "Robin",
  "main": "index.html",
  "icon": "assets/img/icons/icon.icns",
  "version": "0.6.1",
  "window": {
    "title": "Robin",
    "toolbar": false,
    "width": 850,
    "height": 500,
    "position": "mouse",
    "min_width": 850,
    "min_height": 500
  },
  "webkit": {
    "plugin": true
  },
  "devDependencies": {
    "gulp": "*",
    "gulp-less": "*",
    "gulp-minify-css": "*",
    "scraper-web": "*",
    "chalk": "*",
    "natural":"*",
  },
  "scripts": {
    "start": "nodewebkit",
  },
  "dependencies": {
    "gulp-install": "^0.4.0"
  }
}
```

Code

Development of the browser and the Chrome extension is broken up into several major tools and complements. These include the nw.js framework which allows development of the browser, angu-lar.js core.js file which works for interacting with the HTML to the backend node.js core, along with rending of elements. The node.js which runs the Baylor classi-fier algorithms, interact with the database layer and the API's for the several components of the code make it flow smoothly.Such as the verb checker etc.

Nw.js and Gulp

Gulp: Gulp solves the problem of repetition. Many of the tasks that web developers find themselves doing over and over on a daily basis can be simplified by becoming automated. Automating repetitive tasks create more time to do nonrepetitive tasks which increase productivity.

Node-Webkit (Nw.js): NW.js is an app runtime based on Chromium and node.js. You can write native apps in HTML and JavaScript with NW.js. It also lets you call Node.js modules directly from the DOM and enables a new way of writing native applications with all Web technologies.

Knowing what each tool does, having to create a folder structure like the following structure

Dist: where compressed and optimized files go

Fonts: Where the fonts are located

Img: where the images are located

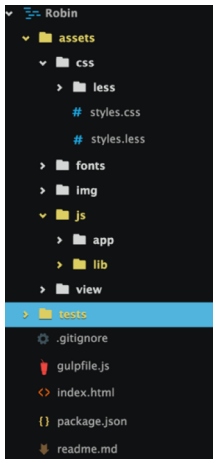
Js: where the javascript files are located

Less: where the less files are located

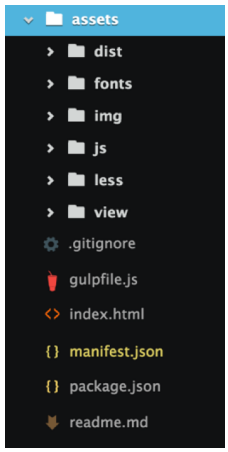
Views: where the HTML files are located

App: where the system javascript files are located

Lib: where the dependencies such as jquery is saved



Browser



Chrome

First exampling the set of the Node-Webkit, it's build around both the package.json and gulp.js file. With the package.JSON file you have a JSON file which you can set rules and dependencies for other npm packages to use. Within the package file for the web browser looks like this.

As you can, see the title for the program is set here, also with the minimum height and width of the application can reach along with the use of chromium by enabling the WebKit engine, which also allows for the access to Chrome's APIs. In the devDependencies, you can include other Npm packages to the bundle when developing the program and the version number you need.

The gulp file then auto helps build and automate the creation of the exe and dmg in creating the browser program itself along with turning the less files to CSS, minifying the JavaScript, moving files around.

To use, you run the following

1. Npm install (this installs all the node packages we are dependent on)
2. Then we run gulp build.

```
var gulp = require('gulp');
var less = require('gulp-less');
var gutil = require('gulp-util');
var minifyCSS = require('gulp-minify-css');
var uglify = require('gulp-uglify');
var chalk = require('chalk');
var logger = require('gulp-logger');
var rename = require('gulp-rename');

gulp.task('build', ['less', 'clean', 'scripts'], function() {
  console.log(chalk.red('Starting.....'));
  var nw = new NwBuilder({
    files: ['*', 'assets/css/**', 'assets/js/**', 'assets/view/**', 'assets/img/**', 'assets/fonts/**', 'node_modules/**'
    platforms: ['osx32', 'osx64', 'win64'],
    macIcns: "assets/img/icons/logo.icns",
    version: "0.12.0",
    quiet: true,
    zip: false

  });

  // Build returns a promise
  nw.build().then(function() {
    console.log(chalk.green('All Done.... ') + chalk.red("  <3"));
  }).catch(function(error) {
    console.error(error);
  });

});
```

Within the above code, we have two snippets of the build process that moves the files and compress it into a folder and runs the nw.js code to build the web browser itself in a folder called build which creates the web browser as a native app.

Angular.js, L.js and JavaScript

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and allows you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

To add Angular.js, first had to add the following to my L.js. It's important to note that for the loading of JavaScript files, Lazy.js is used that allows for parallel script / CSS loading instead of one after each other.

Inside the file Main.js which is located under assets/js/app/main.js.

Looks like this

```
ljs.load(['http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js', 'assets/js/lib/jquery-1.11.3.js','/js/lib/firebase.js'], 'assets/js/lib/core.js', 'assets/js/lib/cookie.js',
 function() {
   $.material.init();
 });
}
```

As you can see it loads in the order you want it, anything within the [] will be loaded first then the rest after. Then in the callback function, you can call anything functions you wish. The materiel.init() is called.

For the Angular.js core file, Snippets will be used to example major or important factors.
Inside the core.js file we can include other projects and classes by adding it using require(name of module). This is done for the verb checker, profanity checker and the parser along with others. The use of detecting when the web browser tab has changed the following angular directive was built.

Then within the angular tempting engine for creating new tabs, we add the function as follow i**frame-onload="iframe-LoadedCallBack()" .**

```
app.directive('iframeOnload', [function() {
  return {
    scope: {
      callBack: '&iframeOnload'
    },
    link: function(scope, element, attrs) {
      element.on('load', function() {
        return scope.callBack();
      })
    }
  }
}]);
```

To allow for the the angular directive to load on the iframe. You send a function to inherit the directive and bind it to load event in the iframe.

```
$scope.iframeLoadedCallBack = function() {
  balance();
  checkForBannedUrl();
};
```

This allows for all i-frames on load the following function. Balance and checkforbannedurls. This takes the blacklist and whitelists and checks if the current url of the tab that the user is viewing matches the banned url or term the parent has set. It then calls the smart catch function which changes the color to black of the web browser.

```
/**
 * Banned Urls, redirects if its a banned url
 * @param {none} none
 * @return {none} none
 */
function checkForBannedUrl() {
  for (i = 0; i < $scope.blackList.length; i++) {
    var currentUrlNow = $('.iframe.active').contents().get(0).location.href,
      bannedUrl = $scope.blackList[i]["url"];
    if (currentUrlNow.indexOf(bannedUrl) > -1) {
      smartCaught();
      break;
    }
    $scope.searchTerm = currentUrlNow;
    $scope.apply;

  }
}
```

All other buttons and clicks such as the refresh and click back all use the $scope elements that bind them to functions to perform
that action.

```
' ng-click="goBack()">
```

```
/**
 * Go Back in iframe
 * @param {none} none
 * @return {none} none
 */
$scope.goBack = function() {
  document.getElementById($('.iframe.active').attr('id')).contentWindow.history.back();

};
```

To add angular to your HTML you need to state it in the body of the HTML skeleton.

```
<body ng-app="robin" ng-controller="controller" data-ng-init="init()">
```

Which then is states to use our app called Robin in the core.js file. Under the controller called our controller called controller. Where the variables under the $scope + variable name can be accessed

```
var app = angular.module('robin', []).filter('trustUrl', function($sce) {
  return function(url) {
    return $sce.trustAsResourceUrl(url);
  };
});
```

```
app.controller('controller', function($scope) {

  $scope.listOfProfanityWords = [];
  $scope.listOfProfanity = [];
  $scope.listOfGoodWords = [];
  $scope.browser = [];
  $scope.words = [];
  $scope.loggedin = null;
  $scope.tabsLimit = 6;
  $scope.caughtColor = "#7B1FA2";
```

Work Horse

```
function workHorse() {
  console.log('Robin Running.....');

  if (typeof $scope.listOfProfanity !== 'undefined' && $scope.listOfProfanity.length > -1) {
    for (var i = 0; i < $scope.listOfProfanity.length; i++) {
      profanityToFirebase($scope.listOfProfanity[i]);
    }
  }
  if ($scope.loggedin) {
    runIsItDisabled(); //Disable the web browser
  }
  if ($scope.words.length !== 0) {

    var temp = classifier.classify(unique($scope.words).toString());
    var k = classifier.getClassifications(unique($scope.words).toString());
    var positiveScore = k[0]["value"].toFixed(20);
    var negativeScore = k[1]["value"].toFixed(20);
    var higher = Math.max(negativeScore, positiveScore);
    console.log(positiveScore + " " + negativeScore + " " + higher.toFixed(20));
    if (higher.toFixed(20) === negativeScore) {
      type = "negative"
      smartCaught();
    } else {
      type = "positive";
    }
    setWebsiteScore($('.iframe.active').contents().get(0).location.href, higher.toFixed(20), type);
    console.log("Current Page is " + type);
  }
  $scope.words = []; //clears it
}
```

The above code is the workhorse module. A workhorse is designed to deal with large amounts of data and deal with it so the system doest freeze like the problem the application had and results of performance increase can be seen. The workhorse is running every 5 seconds where most of the filtering and checks is done. Within here several things are done, where we train the classifier for negative and positive words, and we run the scraper and parser to get the text of the current website the application is viewing. It is built into the core.js in both the chrome extension and browser. It also came because of if the child switches between websites, the workhorse will work though all the text and process on it.

As is visible in the screenshot above, we store a positive and negative score for the text from the website and depending on if negative or positive is the outcome it will print to the console and if negative it will call a function called smartCaught which will redirect the user to the homepage, change the color to black as the theme and alert the parent.

At the end of the example above is the setWebsiteScore function, this will get the Baylor classifier result score and store it onto the Firebase database so we can track results and graph what is going on and if results clash in the future and we can get an average for one URL.

Another important point to say is the if statement just under the console.log. This if statement checks if the words parsed from the website are classed as profanity and if it's already in are testing data sets for training. It is stored on firebase and synced across all machines.

Also in here , fro every new negative word is added to an object and looped though one at a time and added to firebase for our training data.

Parser/Scrapper

```
var request = require("request");
var sanitizeHtml = require('sanitize-html');
var chalk = require('chalk');

/**
 * Send text to parse and return just text
 * @param  {String} Url
 * @return {String} Text from website
 */
module.exports = function(urls, callback) {
  request({
    uri: urls,
  }, function(error, response, body) {
    var removeJavascript = sanitizeHtml(body);
    var test = removeJavascript.replace(/<(?:.|\n)*?>/gm, '');
    var removespacing = test.replace(/\s\s+/g, ' ');
    callback(removespacing.split(" "));
  });

}

```

As mention in this report and the research report parsing and scraping is a huge part.
The applications use of parser was custom built as existing parsers and scrappers were poor and didn't person to the need for the applications spec.

In the above code. It's used in the browser application to by passing the URL, then the application uses the request function that's built into node to make a URL request which returns the HTML, with this back the code sanitizes the HTML, removing any CSS or HTML tags then using regex codes, the applications removes whitespace and line break along with javascript. This then returns the text of websites in an array. It's straightforward and efficient. This code can be stored online, and applications can return the responses.

Profanity Checker

```
/**
 * Checks for profanity
 * @param {object} callback
 * @param {String} word
 * @return {profanity} returns true or false if the word is classed.
 */
function profanityCheck(word, callback) {
  $.ajax({
    url: "http://www.wdyl.com/profanity?q=" + word,
    async: true,
    type: "GET",
    dataType: "json",
    success: function(data) {

      data.response === "true" ? $scope.listOfProfanity.push(word) : null;
      callback(data.response);
    },
    error: function(e) {
      // alert('error, try again');
    }
  });
}
```

As described earlier in the report, every time the workhorse function and the parser returns the text from a URL; it checks each word if it exists in the local array of negative words. Then after if it doesn't exist. It runs the following code which is a hidden API by Google. The API is a profanity checker, you pass a string and it will return a JSON responsive of true or false if its classified as profanity or not and if it is it w ill add it to the to the array of list of profanity which in turn stores it on firebase as well for my training data. One thing to note is the use of Async loading here, as many people may be using it. Below can be seen when a new item is added to a table in fire-base, we can call child_added sync, which downloads the new enter, then the application checks whatever its classed as good or not and adds it the training data.

```
try {
  ref.child("profanity").on('child_added', function(snapshot,
    prevChildKey) {
    for (var q in snapshot.val()) {
      if (snapshot.val()["type"] === "good") {
        $scope.listOfGoodWords.push(snapshot.val()["word"]);
        classifier.addDocument(snapshot.val()["word"].toLowerCase(),
          'positive');
        classifier.train();
      } else {
        $scope.listOfProfanityWords.push(snapshot.val()["word"]);
        classifier.addDocument(snapshot.val()["word"].toLowerCase(), 'negative');
        classifier.train();
      }
    }
  }, function(errorObject) {
    console.log("The read failed: " + errorObject.code);
  });
} catch (e) {}
```

How it links the child to parent

To keep the system simple, the user only needs to register once and they sign into the admin (Chrome extension) and then to link and tell us which web browser to monitor, the parent signs in with the same account and the application will link the accounts and say that the this is a child, and the parent can control it.

Below is the code that is used to make the entry in the firebase table or firebase terms, the object.
That each browser is identified by the mac address of the device they are login with. This information is placed and used as identifying children and parent accounts.

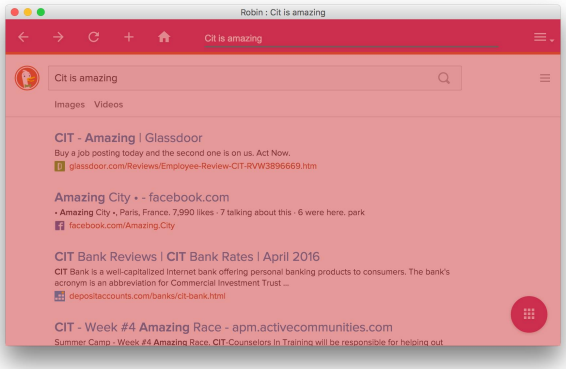
```
/**
 * Set the current userId in the database.
 * @param {String} id
 * @return {none} none
 */
function setIpAddress(id) {
  var usersRef = ref.child(id).child("children").child(removeRegexForMac(usersMacAddress));
  usersRef.set({
    name: user,
    status: "active",
    currentUrl: "none",
    time: getCurrentTime(),
    date: getCurrentDate(),
    platform: navigator.platform
  });
}
```

Disabling/Enabling the Browser

```
/**
 * Attach an asynchronous callback to read the data at our posts reference
 * @param {none} none
 * @param {none} none
 * @return {none} none
 */
function runIsItDisabled() {
  var usersRef = ref.child($scope.loggedin).child("children").child(removeRegexForMac(usersMacAddress)).on("value", function(snaps
    $scope.stop = snapshot2.val()["stop"];
    if ($scope.stop == "yes") {
      document.getElementById("blank").style.display = "block";
    } else {
      document.getElementById("blank").style.display = "none";
    }
    console.log($scope.stop);
  });
}
```

Another option and feature that changed in the application were the use of limiting the child from having internet access. Original it was set at a time set by the parent. This was received poorly by the user testing. It was changed to a disable button.

In the Chrome extension, there is a button where they option to turn off the internet for the child. When the user those this it sets a boolean flag in the Firebase database, and a handy option in Firebase is it will sync changes. So this information is alerted and synced down to the client. As you can see above. If the flag is set to true it will add a block effect to the browser disabling the use of the browser. And If no it will remove the block.




```
  /**
   * Stop Innet
   * @param {String} userData
   * @param {String} email
   * @param {String} password
   * @return {none} none
   */
  $scope.startInternet = function(id, event) {

    console.log($scope.stopped);
    var usersRef = ref.child(authData.uid).child("children").child(event.target.id);
    usersRef.update({
      stop: "no"
    });
    $scope.stopped = "yes";
    $scope.$apply;
  }

});
```

Firebase structure

As Firebase is new to most people and the benefits are endless, the applications use it to great effect in many areas. One of the main advantages is the ease of use and the fact rendering JSON objects rendering time is more significant for updating and syncing.

The mac address identifies the child, this is unique and allows to identify individual machines. In the list object, there is the list of objects for the words to block example Batman and its type is white and that is allowed where jddd is black and will be stopped by the system.
Information is created when a user signs up they get assigned a random id. Within the object we have a child object which where all the child's information is stored, an information object which stores the username and password . There then is a list object is where the blacklist and whitelist settings are stored there.

The same is for the profanity training data. Its all stored within a another object within a object called profanity and within there is either

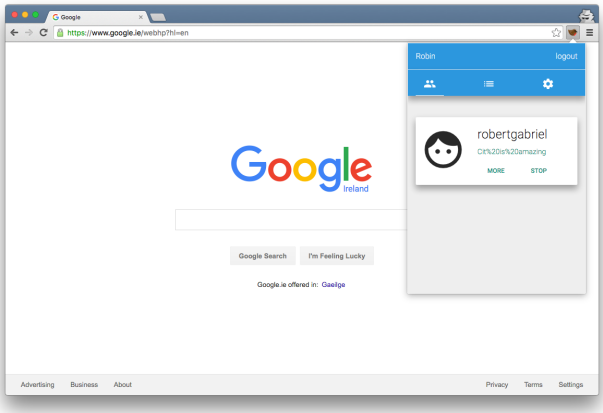
Unique

In Angular there's another option when rendering for loops but where to want to remove any dualities that are caught in the buffer.

An example of this is when the applications render the list of children.
As you can see the application uses the data-ng-repeat to render the child objects but also is the unique detective, which we pass in the object to have unique and in this examples case is the name key in the child object.

```
1   <div data-ng-repeat="banndedUrls2  in child |  unique:'name'">
```

Then to the code example to the right, it's a piece of code that will remove duplets from the list of objects.
Below is the example of the loop working.




```
app.filter('unique', function() {
  return function(input, key) {
    var unique = {};
    var uniqueList = [];
    for (var i = 0; i < input.length; i++) {
      if (typeof unique[input[i][key]] == "undefined") {
        unique[input[i][key]] = "";
        uniqueList.push(input[i]);
      }
    }
    return uniqueList;
  };
});
```

Chrome Extension Notable parts

Most of the code base is the same for both the Chrome extension and the web browser, besides a few things

Within the security features of building a Chrome extension is default not allowing of links within the app (when a parent wants to click on the link), along with the default allowing of images stored on the app. Below the application is told on compile to allow links and images from “https,FTP,mailto,could and within the Chrome extension”.

```
$compileProvider.aHrefSanitizationWhitelist(/^\s*(https?|ftp|mailto|coui|chrome-extension):/);
$compileProvider.imgSrcSanitizationWhitelist(/^\s*(https?|ftp|file|blob|assets|chrome-extension):|data:image\//);
```

The chrome extension uses the route function of angular to switch between pages. This is done by the routing path. So when they user clicks the link to a selected child. The application has a a base index.html file has has the header and footer but no body. It has <div ng-view> tag. Depending on the route the html will injected into there. For the link below is the path child followed by the childs id. Then in the router file

```
'><a ng:href="child/{{banndedUrls2.id}}">More</a></button>
```

Then in angular file all links flow though the path. There is a $routerProvider. Which allows to control the path and location of each link. Then when it comes to the child/:id which in the example was clicked. it stops and loads the following inflation to load the profile.html and inject it into the index.html. To use the controls child for the access of built in function, not to cache the page and delay the loading by one second as to allow for syncing though firebase.

```
$routeProvider.when('/index.html', {
}).when('/child/:id', {
  templateUrl: './assets/view/profile.html',
  controller: 'child',
  cache: false,
  resolve: {
    // I will cause a 1 second delay
    delay: function($q, $timeout) {
      var delay = $q.defer();
      $timeout(delay.resolve, 1000);
      return delay.promise;
    }
  }
```

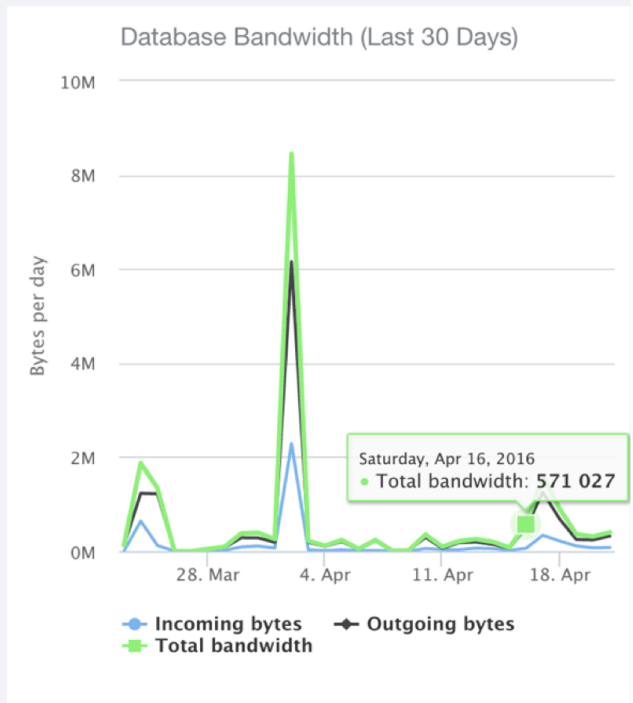
Similar to the browser index.JSON file, When developing for a chrome extension you have to have any 3rd party scripts defined as safe under the web_accessable_resouces. These include Jquery, bootstrap along others. Otherwise that chrome wont allow these to load. You also have to state which permissions you require. In our case we need to allow reading of tabs, local storage ,http and https. This allows us to save personal settings such as colors.

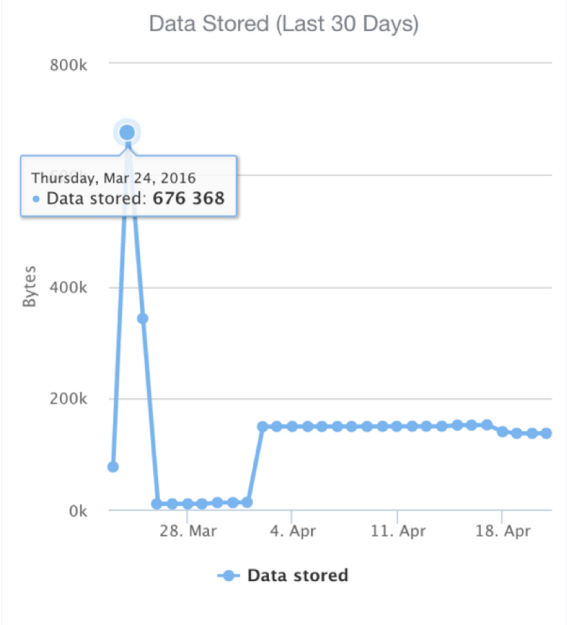
```
{
  "name": "Robin",
  "version": "0.12.0",
  "manifest_version": 2,
  "description": "See what your child is doing on web, Made easy with Robin",
  "content_security_policy": "script-src 'self' https://cdn.firebase.com  https://*.firebaseio.com ; object-src 'self'",
  "browser_action": {
    "default_popup": "index.html"
  },
  "permissions": ["tabs", "storage", "http://*/*", "https://*/*", "fontSettings"],
  "icons": {
    "128": "assets/img/icons/icon-128.png",
    "16": "assets/img/icons/icon-16.png",
    "48": "assets/img/icons/icon-48.png"
  },
  "web_accessible_resources": [
    "https://*.firebaseio-demo.com",
    "https://*.firebaseio.com",
    "https://cdn.firebase.com",
    "http://octicons.github.com/components/octicons/octicons/octicons.css",
    "http://www.justinaguilar.com/animations/css/animations.css",
    "http://jsonip.com",
    "http://ajax.googleapis.com/ajax/libs/angularjs/1.5.0-rc.0/angular-route.js",
    "http://ajax.googleapis.com/ajax/libs/angularjs/1.5.0-rc.0/angular.min.js",
    "http://fonts.googleapis.com/css?family=Roboto:300,400,500,700",
    "https://fonts.googleapis.com/icon?family=Material+Icons",
    "http://cdnjs.cloudflare.com/ajax/libs/jquery-serialize-object/2.8.0/jquery.serialize-object.compiled.js",
    "https://cdn.firebase.com/js/client/2.3.2/firebase.js",
    "assets/css/bootstrap/*.css",
    "assets/css/fonts/*",
    "assets/js/*.js"
  ]
}
```

Problems

The biggest problem came down to the effect of having 14,000 individual words used for training the Baylor classifier algorithm. What the problem came down to the fact of processing 14,000 results on load was locking up the application (both the web browser and Chrome extension) this is known as bottlenecking. This took several code revisions to get it working.

The solution was breaking the 14,000 words in the data set into objects of 1,000 words (500, negative and 500 positives) and downloading another 1,000 every 15 seconds using the work-horse module. This reduced the bottle necking to zero. Along with less bandwidth and spikes of data used on the server side. The workhorse is also used for scrapping and looping though the text of a website that is processed into the filtering system.





Naive Bayes Classifier

```
try {
  ref.child("profanity").on('child_added', function(snapshot,
    prevChildKey) {
    for (var q in snapshot.val()) {
      if (snapshot.val()["type"] === "good") {
        $scope.listOfGoodWords.push(snapshot.val()["word"]);
        classifier.addDocument(snapshot.val()["word"].toLowerCase(),'positive');
        classifier.train();
      } else {
        $scope.listOfProfanityWords.push(snapshot.val()["word"]);
        classifier.addDocument(snapshot.val()["word"].toLowerCase(), 'negative');
        classifier.train();
      }
    }
  }, function(errorObject) {
    console.log("The read failed: " + errorObject.code);
  });
} catch (e) {}
```

As you can see above the datasets are downloaded from firebase on the startup of the application and whenever a new set is added. It downloads and trains the classier which can be seen above. It lowers each word to lowercase. This is done for all words on a new webpage too.
As above we get an object of words (data set) and loop though it and add each word to the classifier. By the addDoucment function. Then we tell it to train.
Its worth noting that the naive classifier code is from an existing module that has been updated as it was out of date and broken. It is credited in the git repo.

So then when a child access a new site it passes in an array of words into the profanity nam module and will return either negative or positive.

Being able to understand sentence context and the words on a page is an important aspect of the system, as well as the need for our system to know what to block and what not to block. So the application needed to develop an AI to make the filtering system work.
In the applications' filtering system for understanding what to block and what not to block, the Naive Bayes Classifier model algorithm to start.

For the Naive Bayesian classifier algorithm to be any good, you need to train it in what is good and what is bad. So in the applications case, what are positive words and what are negative words. The web browser filtering system is scrapping and parsing words as the users searches. It scrapes the web pages and pushes profanity based words and non-profanity based words into two lists. Profanity and not profanity words. We use these two lists for our training data. This is ever evolving and changing training data, which becomes more accurate, as more people use it. By training the the algorithm. It starts to be become more educated on particular inputs that we know the output of, e.g. profanity or not profanity, so that later on we may test them with unknown inputs (which the filtering system has never seen before) for which they may classify or predict, etc. (in the case of supervised learning) based on its knowledge.

The objects(text on a page) can be classified as either Profanity or Normal Words.
Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently existing objects of Profanity.

Since there are normally twice or three times as many Normal Word objects as Profanity, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership towards positive words rather than profanity.

In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case, the percentage of positive words and Profanity objects and are often used to predict outcomes before they happen.

In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule.

Using this information, the application can predict which should be blocked and what shouldn't be blocked.

Adding Extra Features and Tweaks

By directly using any learning model on a simple bag of features might not lead to a very accurate model. Here are some pre and post processing steps the application used which helped a lot in improving accuracy.

Negation handling is quite important in sentiment analysis, A word/phrase followed by a “ed" or any other term that negates the meaning of what follows should be handled appropriately. A technique that works well is treating negated terms by adding a prefix like "_ed" to the actual word and then using the counts. Another technique that works is updating the counts of negated terms in the opposite class

Using individual words was not enough to capture information about sentiment. Word n-grams or a sequence of n consecutive words are a very useful addition to the feature space as they can be used to capture combinations of adjectives with other parts of speech.

Adding n-grams added a lot of spurious and noisy features, and they need to be eliminated by doing some feature selection. This was done by measuring the mutual information of individual features with the training set and selecting the top few thousand features. Again using the lists generated from the web browser.

Datasets/Training

The Training sets are taking from scraping the following the websites, removing duplets and training them against the profanity API. To get the base training data. I complied two lists of 3500 of the most popular negative websites (porn and gambling) and another of positive websites which was made up the world most popular websites from the Alexa world rankings. These can be found in the GitHub repo.
Then within the browser source code, there is an npm command called “npm rechecks” which checks if the training data is empty and if so will scrap these 3500 websites for all their words and classify it off the profanity API and store it on firebase for the applications training data.

Good Words:["the","of","and","to","a","in""]
Negative Words: ["4r5e","5h1t","5hit","a55","shit","anus","ar5e"]

As you can see the system syncs and trains the algorithm based on the classified words either positive or negative words and training the system so that when a user visits a website it will re-load and have no information to make the choice on if to block the web page or not.

Problems

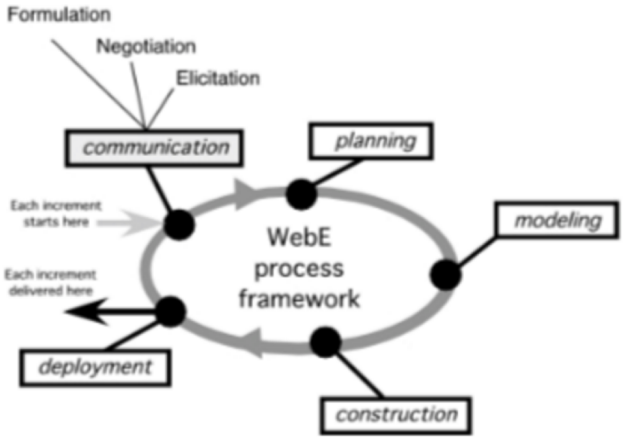
One of the problems encountered from creating and training the Naive Bayes Classifier Algorithm was the use of the of negative words and positive words. Because the use of negative and positive words both appear in porn sites. To counteract this and the results can be seen later, the positive words were halved that of the negative words.

Results & Testing

For testing for the application, a method used a lot in modern software development, the agile form of testing which is when you test the software as you go. As each feature was built it was reviewed and tested by myself first though a method of dog feeding, I test it, making sure performance and issues were annoying before moving on.

Agile testing is a huge benefit in the development of the browser and filtering system as it allowed more bugs to be detected and fixed, instead of rushing and not making clean, reusable code.

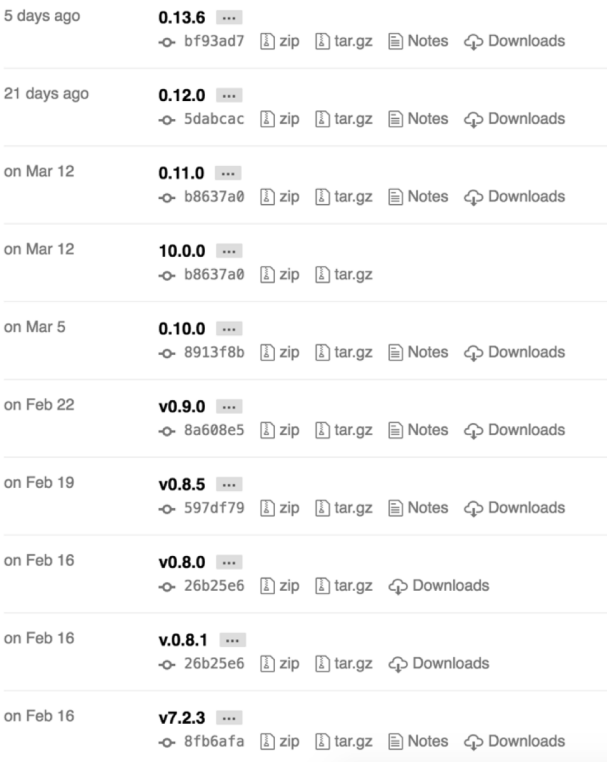
Along with that, I will be able to note and take feedback from myself and the results of testing, which allows for bugs to be fixed while the code is still fresh in my mind.
As you can see below, I will use the Web-E process framework, to build and deal with problems as they arise.



There were 13 releases, tested by 45 live parents and 61 of their children where usability testing monitored though a-b testing in the firebase logs.

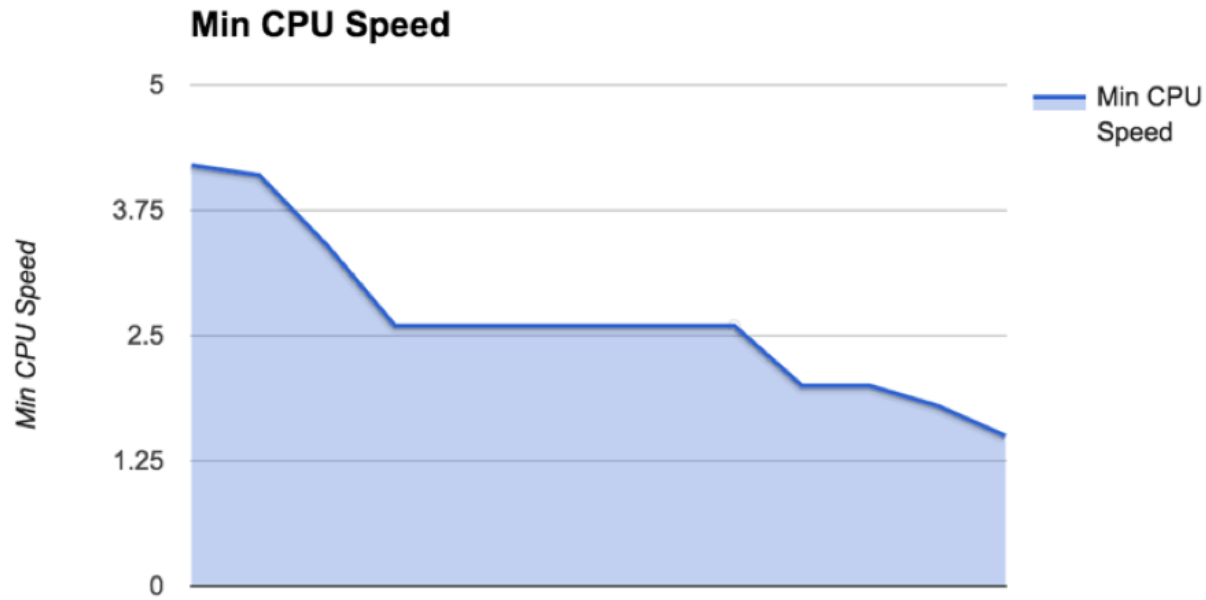
Performance

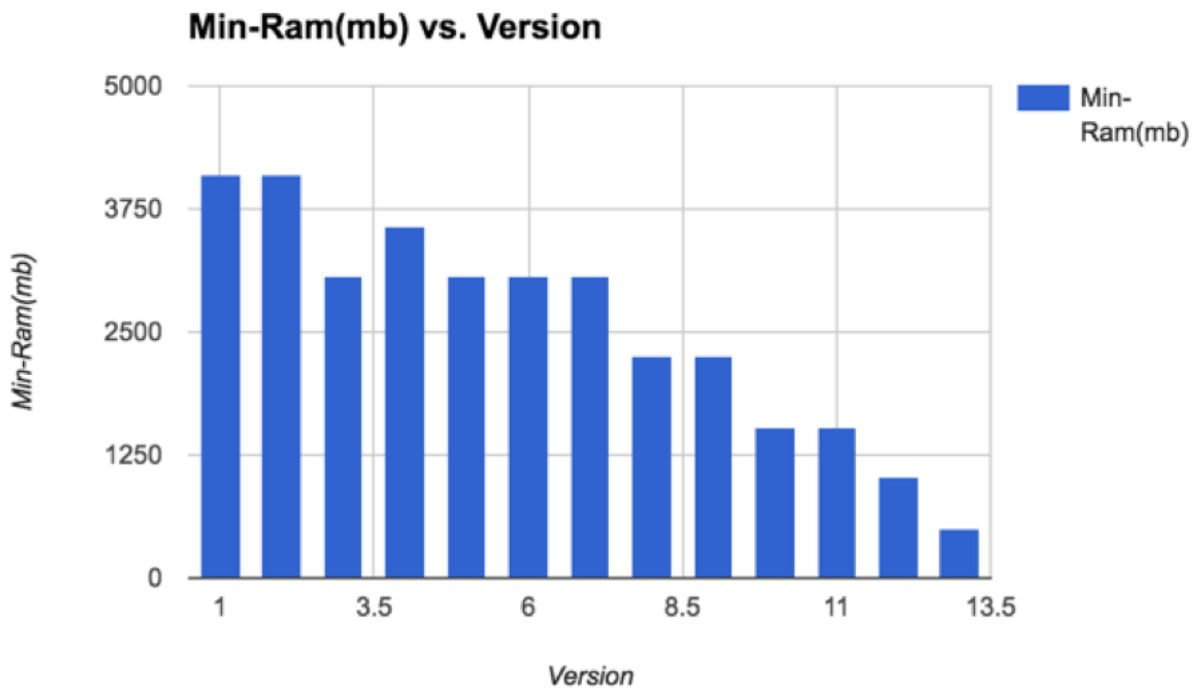
There were 13 releases in total for the versions of the web browser and the Chrome extension. When mention performance it was important to look at areas to improve on. With the method of using the webE process framework, it became apparent with each code revision that could be written less. From minifying files (removing whitespace), using advantages in local storage to making sure it was clearing the buffer.



As seen below you can see the improvements over time as I started to remove unneeded code during syncing moments and boot times.
This lead from 4gbs of ram to 512mbs of ram to be used.

The measurement was done when the average running time of the system running and the ram and CPU usage over the active time of usage from the live testers. Below is graphs showing improvements over time with the revisions with the decrease in cpu usage and ram.



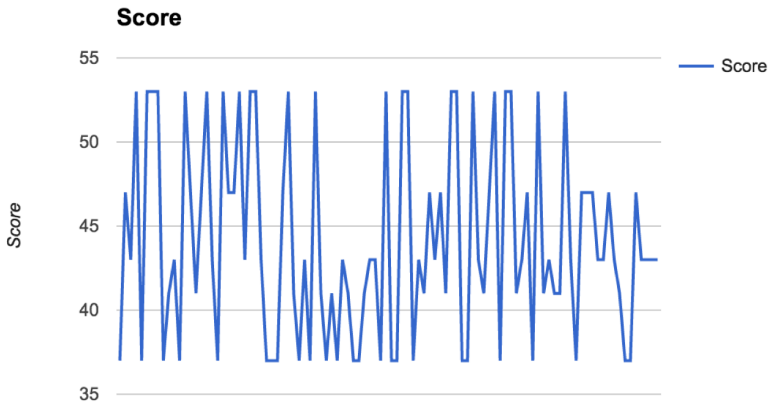

Min-Ram(mb) vs. Version
5000
3750
2500
1250
0
Min-Ram(mb)
1
3.5
6
8.5
11
13.5
Version
Min-Ram(mb)

Blocking/Algorithm Results

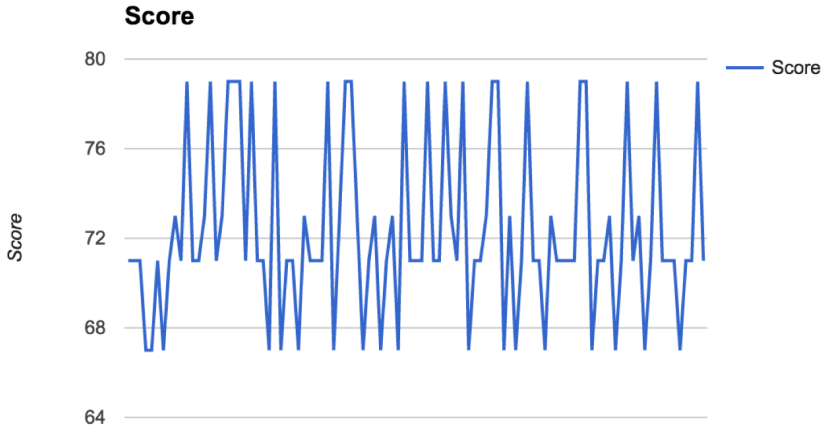
To test this correctly became a problem. When developing the system and discussing the system with Doctor Ted Scully and Doctor Donna O'Shea. There needed to be a ramp up of data and sites.
As discussed earlier in the report that the use of scraping results from 3,000 negative sites and 3,000 positive sites using the parser to generate the training data.

The methods here were manual to make sure it was working and reviewing it to a point. I then built an npm package to automate these tests(using gulp), to see the responses of where it was negative or not of I personally knowing the outcome of the result before hand. These are stored online in firebase.
It's also important to note that each time a person was using the browser and navigating to a negative site, it was locking the system.

Below is before I halved the amount of positive to negative words for the training data, as I was getting false positive effects. It was tested on 100 websites which can be found in the git repo of positive and negative sites.



Below is the results for the same 100 websites tested over and over but with half the positive training to that of the negative.



Conclusion

Now that the project is completed, I can say that I learnt a lot about natural-language algorithms, machine learning and development of the tools to create a web browser and the Chrome extension which I didn't before. It became clear that when it came to the implementing stage, it would be hard to achieve 90% or higher in detecting. As time went by and I came to understand the algorithm better and after having talked to both Dr. Donna O'Shea and Dr. Ted Scully, achieving 70% became the ideal goal to get the boyar classifier algorithm working. I did have problems in getting that score, but I discovered that halving the positive words over the negative words allowed me to achieve a result of 70% -80% as seen earlier in the document. This worked as the positive words were appearing in adult themed websites as well and sometimes over powering the negative words. Based on this, a future study into text behaviour on adult websites might be possible to improve the existing work.

What I've learned

Building the application and developing the algorithm helped to increase the standard of my coding and algorithm knowledge which I thought was my weak point. I learnt a lot about that in my time on work placement in Teamwork.com. I did achieve all my outcomes besides the disabling of the other browsers, the reason for this being a performance issue. It isn't ideal to kill process on windows like that and on OS X (Apple computers) you cannot access the commands to kill out programmes and stop them from running, so it's up to the parent to disable the browsers by going though the child protection settings in OS X settings, however that is not particularly difficult or time consuming. One of the areas of which I am proud is that I created the start of a live database which keeps track of the ever-changing slang used by adult websites. For example, a dog breeding site could refer to a female dog as a 'bitch,' with perfect innocence. However, most child protection software cannot differentiate between this use. The application I developed, on the other hand, has been able to interpret language and slang based on on the collection of terms collected such as _ed, and so it protects children on the web more effectively than existing systems, according to the tests. The only other regret I have, is that I wish I had had more time to investigate text patterns and tends in websites and peoples web useage.

As per the outline detailed in my report, I intend to use the most up to date and advanced tools currently available such as node.js, gulp, npm and Firebase, which the college doesn't teach, leading to a cutting edge project, which had been awarded by being showcased on mateialup.com which showcases beautify crafted software. I was extremely happy with the project and my work overall, as I kept to a high level of coding and attention to detail in both engineering of the services and the material design interface aswell as it blocking porn sites but not sexual education site.

Sorry for any spelling mistakes :)

How the application compares to others

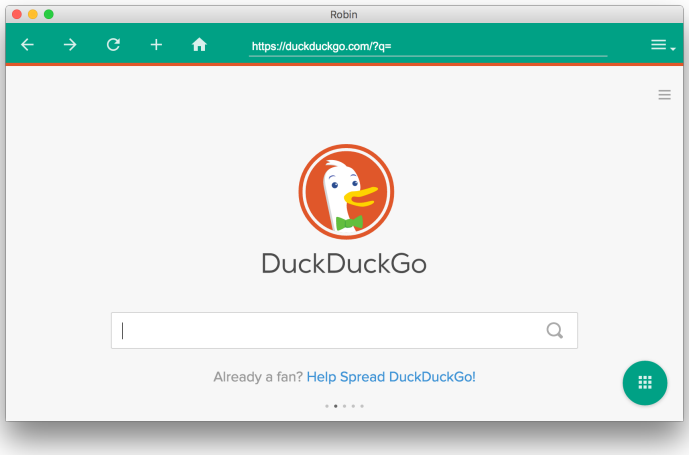
Title	SpyAgent	Net Nanny	Qustodio	Robin
Website Monitoring	YES	YES	YES	Yes
URL Based Website Blocking	yes	YES	YES	Yes
Content / Category Based Website Blocking	YES	YES	YES	Yes
Social Media Monitoring	YES	NO	YES	Yes
Search History Monitoring	YES	YES	YES	Yes
Chat/IM Recording	YES	YES	YES	No
Email Recording	YES	NO	NO	No
Email Attachment Recording	YES	NO	NO	No
Software Keylogger	YES	YES	NO	No
Automatic Screenshots	YES	NO	YES	No
Program Activity Monitoring	YES	YES	YES	Yes
Application Stealth/Invisibility	YES	NO	NO	Yes
Remote Monitoring	YES	NO	NO	Yes
Website Whitelisting	NO	YES	YES	Yes
Enforce Program Time Limits	NO	NO	NO	Yes
More Expensive than Some Competitors	NO	YES	NO	No

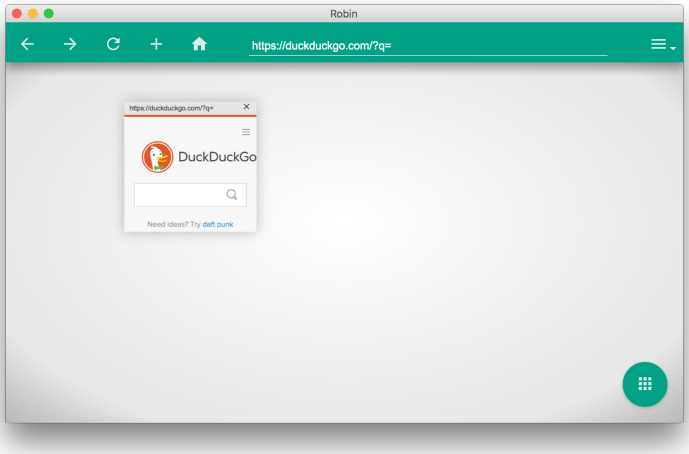
Bibliography

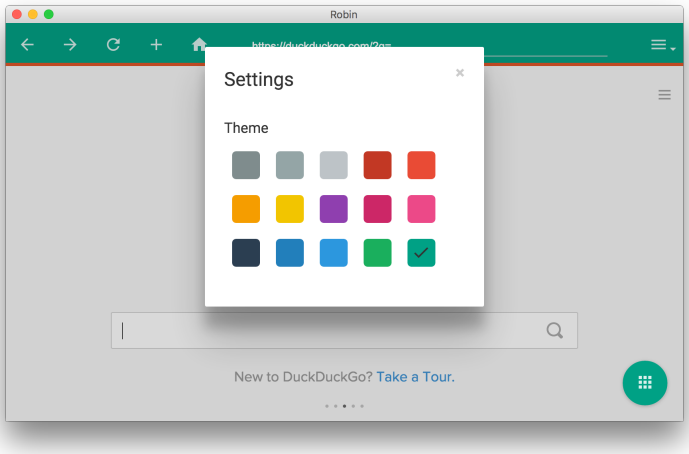
1. For a description of these events, see Marjorie Heins, Not in Front of the Children: "Indecency," Censorship, and the Innocence of Youth" (Rutgers U. Press, 2007)
2. Mainstream Loudoun v. Board of Trustees of the Loudoun County Library, 2 F. Supp.2d 783, 24 F. Supp.2d 552 (E.D.Va. 1998)
3. National Research Council, Computer Science and Telecommunications Board, Youth, Pornography, and the Internet
4. Commission on Child Online Protection (COPA), Report to Congress (Oct. 20, 2000)
5. See Commission on Child Online Protection (COPA), Report to Congress(Oct. 20, 2000)
6. Commission on Child Online Protection (COPA), Report to Congress (Oct. 20, 2000) 8
7. https://github.com/Projectbird/profanity
8. **Browser:** https://github.com/Projectbird/Robin
9. **Chrome Extension:** https://github.com/Projectbird/Robin-Chrome
10. **Parser/Scraper:** https://github.com/RobertJGabriel/Text-Scraping
11. **Testing Data:** https://github.com/RobertJGabriel/List-Of-Explicit-Words
12. **Profanity Check:** https://github.com/RobertJGabriel/profanity
13. **Website Lists:** https://github.com/RobertJGabriel/List-Of-Explicit-Words
14. **Baylor Classifier Algorithm**: https://github.com/RobertJGabriel/natural

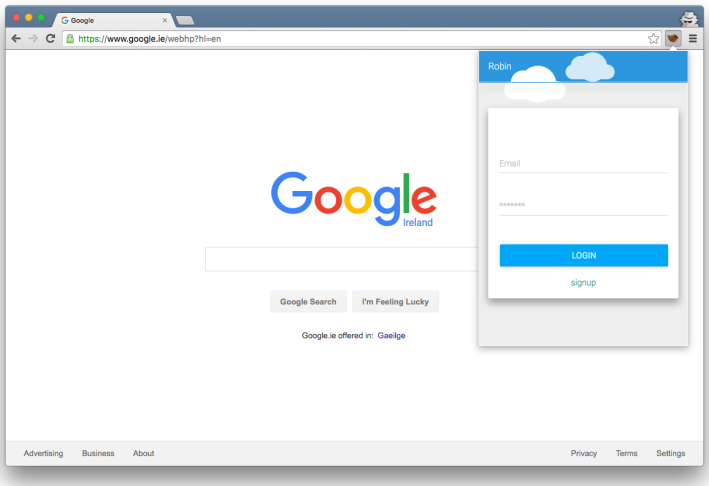
Appendix

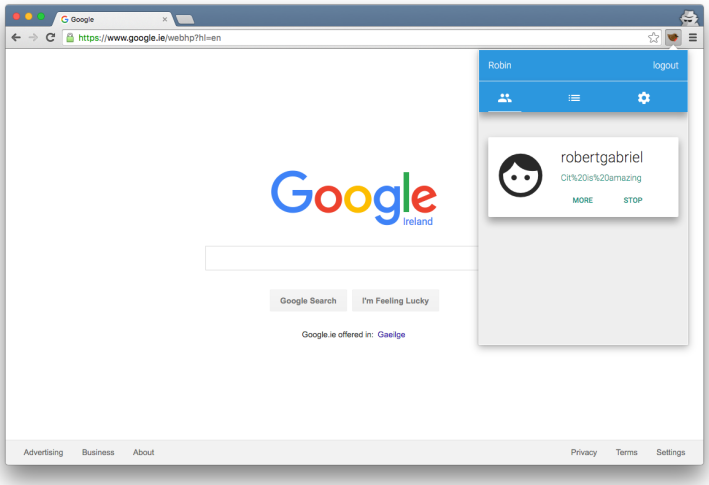
Screenshots

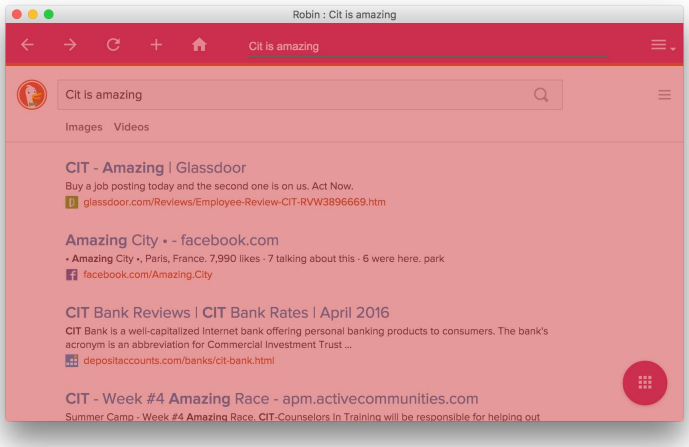


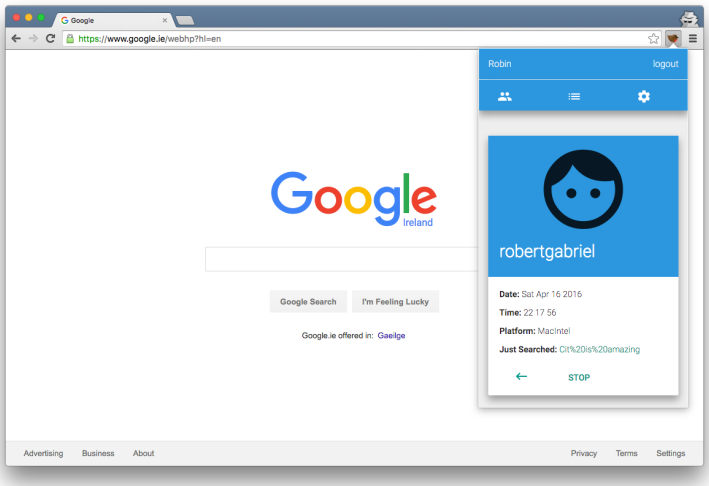


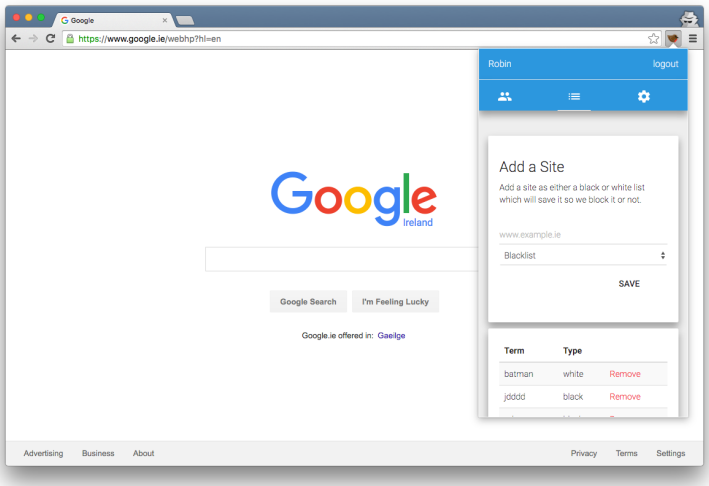


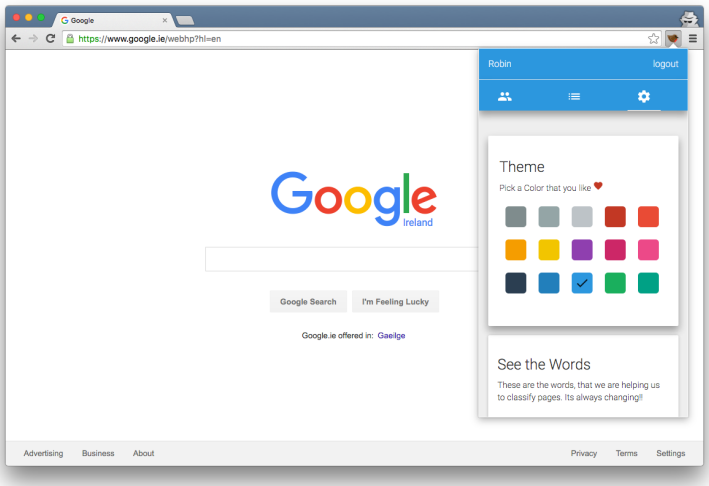

Google
https://www.google.ie/webhp?hl=en
Robin
Email
LOGIN
signup
Google
Ireland
Google Search
I'm Feeling Lucky
Google.ie offered in: Gaeilge
Advertising
Business
About
Privacy
Terms
Settings


Google
https://www.google.ie/webhp?hl=en
Robin
logout
robertgabriel
Cit%20is%20amazing
MORE
STOP
Google
Ireland
Google Search
I'm Feeling Lucky
Google.ie offered in: Gaeilge
Advertising
Business
About
Privacy
Terms
Settings


Robin : Cit is amazing
Cit is amazing
Cit is amazing
Images
Videos
CIT - Amazing | Glassdoor
Buy a job posting today and the second one is on us. Act Now.
glassdoor.com/Reviews/Employee-Review-CIT-RVW3896669.htm
Amazing City • - facebook.com
• Amazing City •, Paris, France. 7,990 likes · 7 talking about this · 6 were here. park
facebook.com/Amazing.City
CIT Bank Reviews | CIT Bank Rates | April 2016
CIT Bank is a well-capitalized Internet bank offering personal banking products to consumers. The bank's acronym is an abbreviation for Commercial Investment Trust ...
depositaccounts.com/banks/cit-bank.html
CIT - Week #4 Amazing Race - apm.activecommunities.com
Summer Camp - Week #4 Amazing Race. CIT-Counselors In Training will be responsible for helping out


Google
https://www.google.ie/webhp?hl=en
Robin
logout
Google
Ireland
Google Search
I'm Feeling Lucky
Google.ie offered in: Gaeilge
robertgabriel
Date: Sat Apr 16 2016
Time: 22 17 56
Platform: MacIntel
Just Searched: Clt%20is%20amazing
STOP
Advertising
Business
About
Privacy
Terms
Settings


Google
https://www.google.ie/webhp?hl=en
Robin
logout
Google
Ireland
Google Search
I'm Feeling Lucky
Google.ie offered in: Gaeilge
Add a Site
Add a site as either a black or white list which will save it so we block it or not.
www.example.ie
Blacklist
SAVE
Term
Type
batman
white
Remove
jdddd
black
Remove
Advertising
Business
About
Privacy
Terms
Settings


Google
https://www.google.ie/webhp?hl=en
Robin
logout
Google
Ireland
Google Search
I'm Feeling Lucky
Google.ie offered in: Gaeilge
Theme
Pick a Color that you like
See the Words
These are the words, that we are helping us to classify pages. Its always changing!!
Advertising
Business
About
Privacy
Terms
Settings

User Manual

How do I signup

You click signup from the dropdown. Enter your email and password and that's it you're all signed up

How do I Login

You click the login in button and Enter your email and password and that's it your all logged in.

How do I connect my child's browser to my account, so I can see what there doing?

Once you have signed up and logged in to the admin panel (chrome app). You simply go to eh robin browser and log in using the same details. That's it you can now see what the child is doing from the chrome app.

What information do you collect?

We collect the following information

The mac address of the machine (so we can tell which child it is)
The type of device (is it an iPad or a Dell laptop)
The Url your currently visiting
The time your searching and from where
The Current date
What Devices are supported?

Browser

Well 64bit windows 7
64bit windows 8
64bit windows 8.1
64bit windows 10
Mac Yosemite 64
64bit Yosemite 32
Chrome Extension

How do I take a device off my account?

You simply go to the browser and click log out.

How are the machine learning and filtering working?

To keep this simple, we collect information based on profanity words from what the child is searching and we store those online in our online database. Based on these ever-changing words of profanity we update the profanity algorithm. So as the more people search the better it gets.

The URL is sent to use, we parse out all the words, and check each word for is it classed as profanity and is its a verb. If it's either of them we update our database and use the new words we added to adjust the filtering algorithm.

Each URL is based on the following base formula.

A number of words on a page.
The amount or profanity words
Number of verbs.

Using this information, we will start to see a trend where adult sites will have a higher profanity over a number of words. This is a great indicator of what to block.

Can I add a site to be allowed?

You sure can, these are called whitelists, which is a set of allowed sites even if our system thinks it should be blocked. To add a site open up the robin chrome extension and click lists on the navigation. From here you can input the base URL so if you wanted to block pornhub.com. Just enter Pornhub and select black to block it or select white to allow it

How can I see the list of Profanity words

Open up the robin chrome extension and click profanity on the navigation? You can see all the profanity words from our database here.

Github states

Number of download (total)	**6202**
Number of Stars (total)	13
Number of Forks (Total)	2
Favioute code	Javascript