

Title: Enhancing sentence understanding by using machine learning algorithms and natural language algorithms aka a poor man's Google.

Description: Through use of building a web browser and parenting tools for filtering explicit material, based on natural language key rules. The information from the these key rules results (ie trying understand if the sentence is explicit) will be stored online to greater enhance the key rules and build a better understanding of what is explicit and what isn't. Aka a adult site vs a sexual education site.

Prepared for: Seamus Langford, Project Supervisor1, Tim Horgan Second Reader

Prepared by: Robert Gabriel

4th of December 2015

Github Orzigation : <https://github.com/Projectbird>

Demo Browser : http://projectbird.github.io/Robin/

Demo Admin : <http://projectbird.github.io/robin-chrome-addon/popup.html>

Splash Demo : http://projectbird.github.io/robin-chrome-addon/popup.html

# Table of Contents

# 1 Glossary of Terms

## 1.1 Whitelist

A whitelist is a list of entities approved for authorized access or privileged membership to enter a specific area in the computing world. These entities could include electronic groups or organizations, privileged websites or even email addresses.

## 1.2 Blacklist

In Internet terminology, a generic name for a list of email addresses or IP addresses that are originating with known spammers. Individuals and enterprises can use blacklists to filter out unwanted e-mails, as most email applications today have filtering capabilities.

## 1.3 Web browser

A web browser (commonly referred to as a browser) is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI/URL) and may be a web page, image, video or other piece of content.

## 1.4 Filtering System

On the Internet, content filtering (also known as information filtering) is the use of a program to screen and exclude from access or availability Web pages or e-mail that is deemed objectionable. Content filtering is used by corporations as part of Internet firewall computers and also by home computer owners, especially by parents to screen the content their children have access to from a computer.

## 1.5 Urls

URL is an acronym for Uniform Resource Locator and is a reference (an address) to a resource on the Internet. A URL has two main components: Protocol identifier: For the URL http://example.com , the protocol identifier is http .

## 1.6 Time Limit

A limit of time within which something must be done.

## 1.7 Browser

is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.

# 2 Assumptions

1. All users have an email address
2. They have used a web browser before.
3. Other assumptions are stated during the document.

# 3 INTRODUCTION

This report contains the project research which I have carried out for the purpose of creating an innovative new way of filtering children’s internet access for parents and teachers (Machine Learning, through a chrome extension.) The main reason for including a chrome extension is in order for it to serve as a reminder and to provide easier access for the parent so that they can monitor and control the settings. Users will be able to access the information on the filtering settings on the children's access on the internet from anywhere in the world. Both applications will be developed with the intention of being used by parents and teachers along with children who would use the application to access the internet. Originally, I had intended to create the application for the one machine only.

Before I decided to include the chrome extension and firebase cross-syncing in the system, my inspiration for the original project was due to a growing, every day problem, the question of how parents should monitor or protect their children online. As the web develops and expands, so too does this problem, and the current systems are complex and hard to use and the parent does not normally know how to set it up.

For this reason, I decided to create to build a web browser using node WebKit and an chrome extension whereby users (parents) could easily adjust settings to block and set time limits on 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 web browser and that the web browser on the children's laptop would sync the information and block facebook in real time. If the child was to access the information it will redirect them back to google and alert the parent about the attempt through email and text message. I wanted the web browser to change color based on the user's behavior, an example they try to access the block site the colors of the web browser will turn to black. So in this way, they will know what they did wrong and learn from the experience.

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

A huge area of my research is machine learning, so 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 sentence in its context, for example it would be able to tell the difference between a porn website and a sexual education website.

# 4 OVERVIEW

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

The parent/teacher can download the to 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 robin browser on the child's computer. Once the user is signed up to the system, they can access any settings and change the settings for the robin browser from the admin panel in the chrome extension.

Firstly the parents / teachers can access the admin panel from the google chrome extension and set rules such as blocked sites, time limits on the internet from here. The settings will be synced to the web browser on the child's laptop or tablet into the Robin Browser.

Secondly the chrome extension will also give text alerts when the child tries to access blocked content, and the system will also email them.

On the child's laptop on the robin browser the web browser will look and feel the same as other browsers in terms of 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. This is so the child will learn that was a bad choice. The color will stay black until the parent resets it.

The second major thing is that there will be a limit on the time they can access the web. The admin sets this time in minutes from the google chrome extension. They will see a grayed out screen saying to ask their parents for more time. The system will award them for learning and going onto approved sites.

From an algorithm blocking point of few, I plan to develop a node module that will scrap and run tests to see what the words on the webpage are classified as profanity and try to build a context to the sentence so to better understand what should be blocked and what shouldn't be.

# 5 Project Drivers

## 5.1 The purpose of the project

Create an application for creating and improving on existing filtering algorithms and allowing the parent to view what there childing is on cross platform browser.

## 5.2 Overall Project Goal

The objective of my project is to develop a web browser with Node.js Webkit (cross platform) with a built in custom filtering system to block websites/content that the parent doesn't want the child to access while, allowing for settings to be synced from one machine to another. There will be two parts to this programme, a google chrome add-on which will be for the admins and the standalone browser which is for the children to use. The filtering system should be learning and collecting data and begin to hopefully understand the context of a sentence.

## 5.3 Project Objectives

* Create a web browser user node.js + WebKit
* Develop it for cross-platform devices (mac and windows).
* Allow for an unlimited amount of new tabs in the browser.
* Block websites based on the terms the parent inputs from the google extension.
* Disable other browsers from running.
* Develop basic features of a web browser.
* Send email to the parent when the child tries to access the blocked content.
* Redirect the child to google.com homepage.
* Save all settings to firebase, so they can sync settings to different machines.
* Change theme of the software to dark to alert the user as well.
* Learn to develop a system to stop other software from running.
* Create a google chrome addon for the teacher/parents to set settings which will sync to the browser.
* Add timed limits to the internet access, so the parent can decide how much time the child can spend online.
* 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 to the firebase website.
* Build System to try and understand context of a sentence using natural learning and machine learning algorithms .

## 5.4 The Stakeholders

### Admin:

A person who takes part in an undertaking with another or others, especially in a 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 below 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 ability of test performing will be included. Any conflicts or major solution issues will be addressed during ad-hoc meetings of stakeholders involved.

### Project supervisor:

A person designated to provide intellectual supervision over all of the project’s stages.

# 

## 5.5 Project Tools

The project tools which I intend to use when developing my web browser and the google chrome plugin in the implementation phase are as follows:

* Gulp
* Node.js
* Github
* Node webkit
* Javascript
* Angularjs
* Html5
* Css3
* Less
* Javascript 5
* Photoshop
* Material Design
* Firebase
* Teamwork
* Npm

### Learning Outcomes

When I have completed the research and implementation phase, I expect to have the following learning outcomes as by that stage, I will have developed my web browser and filtering application and hopefully continued my research into machine learning;

* Learn and discover how to allow for quick cross sync of information for a database with speed in mind.
* Learn 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 rating of 10.
* Code up to the web standards on strict settings.
* Learn and develop in Angularjs.
* Use and 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.
* Develop tools to understand sentences the context.

# 

# Project Constraints

## 6.1 Solution Constraints

The application must be able to work the same in Firefox and Chrome browsers, as these are major the players of the market and will serve most critical amount of users.

Multiple devices with different screen sizes must be fully supported.

Users will connect to the application via wide range of devices.

It should not limit the usage of the application.

## 6.2 Implementation Environment Constraints

The application must be implemented to make use of the selected database solution.

Database will facilitate use of storage and also have positive impact on performance.

## 6.4 Partner or Collaborative Applications

The application will utilise third party systems. It will connect to the APIs of firebase in order to provide enhanced information. The operational details for each remote system will need to be set.

## 6.4 Schedule Constraints

How fast the system is and how quickly I need to get it done ( The 12 week semester period)

The time for project completion is 1st of March 2016.

## 6.5 Budget Constraints

As a student, I do not have the means to fund this project, therefore my budget is non existent.

Areas for growth:

* allow for non student members
* record additional training details
* Integrate the login system

## 6.6 Relevant Facts and assumptions

There is no currently existing system.

## 6.7 Business Rules

1. Email address and password must be provided to access system
2. Must have a computer or a tablet
3. Must have Chrome browser

# Project Detail

## 7.1 Web Browser

Upon initially installing, the users (admin) will have to input their email. This is for the firebase so I can sync information from firebase using the google chrome extension.

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

The user can be awarded ‘points’ for visiting website with extremely low profanity rates, these points can be rewards set by the parents (new toys for example) so that the system will educate them. They will lose points for accessing websites with high profanity rates.

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

## 7.2 Google Chrome Extension

On first install, the user (Admin) will be asked to signup or login to their account using email and password combination.

After this the user (Admin) can easily see what the user (child) is viewing in real time and set black and white listed sites. They can see if the user tried to access a high profanity site or a blacklisted website. The user(admin) can reset the color theme on the web browser. They can see the current user's points.

## 7.3 Profanity Algorithm

The profanity algorithm I’m planning to develop will take the sentences and words from a web page and check each one to see if they’re classified as profanity, in particular as a verb, as verbs are often the decisive element of the meaning of a sentence. Using this information and storing it in the firebase database allows me to update and change the key rules often, as words, especially slang, develop different meanings over time. Hence allowing me to to solve the false positive problem by creating a programme capable of understanding the context of a sentence.

# 

# Competition in the market

## 8.1 Competitive Analysis

### Introduction

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

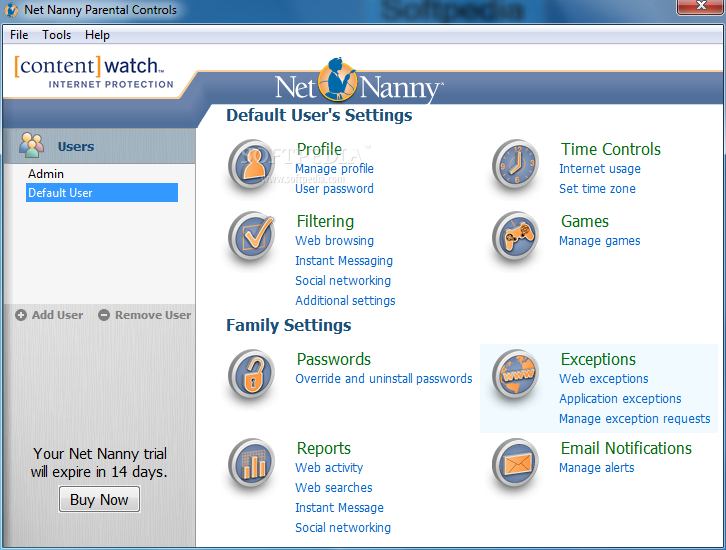
Note: I tested these browsers on a Apple Mac 2015 with 16GB of RAM and an Intel Core-i5 CPU.

### 

### 

### Features

|  |  |  |  |
| --- | --- | --- | --- |
| Title | [SpyAgent](http://internet-filter-review.toptenreviews.com/spyagent-review.html) | [Net Nanny](http://internet-filter-review.toptenreviews.com/netnanny-review.html) , | [Qustodio](http://internet-filter-review.toptenreviews.com/qustodio-review.html) |
| 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 |



### Conclusion

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

On the other hand, Net Nanny gains points for having the most comprehensive profanity filter available, including the ability to perform profanity masking. This is a feature not available with any other program I reviewed. All others take the approach of fully blocking sites with even marginally foul language, which may or may not be the methodology you want to explore.

In short, picking an internet filter software program depends greatly on what is ideal for you. Although I found Net Nanny, SpyAgent and Qustodio 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.

### 

### 

### K9 Web Protection

A very small application that packs quite a punch, K9 Web Protection provides a layer of safety for your children online and runs from right inside your browser.

On the first startup you have to set up a new license to use it and from then on it is simply a case of setting up the filters to the exact specification required. Blocking is primarily done by category with various levels of blocking open to the user ranging from monitor (allows all categories and just logs traffic) to high (protects all default categories as well as social interaction and unrated sites).

Beyond the main category restrictions you can also set up time restrictions to place blocks on when web access is allowed with a custom option allowing you to impose an online curfew for anyone using the internet on the family PC.

The customary web site exception setting permits you to always block or allow specific sites and there is even the chance to block various URL keywords that will automatically bar access to a site should it contain certain words. It goes even deeper than even that with safe search redirecting to K9's own Safe Search and, even though it may concern some that this is turned on as default, it's very easy to turn it off straight from the setup page.

K9 Web Protection performed well in my testing, as I was blocked from accessing various adult sites and gambling services, even when clicking on links from other sites to arrive at them. There was very little I could find wrong with K9 Web Protection and knowing that all of this is completely free meant it gains top marks.

### 

### 

### Kidlogger

Another of the monitoring fraternity, Kidlogger is there to keep an eye on what is happening on your PC and then report back the findings with trademark accuracy.

Kidlogger is a simple to use program that allows you to monitor everything from keystrokes and clipboard content to Skype chats, USB media insertions and website URLs, and everything can be password protected.

Its monitoring capabilities don't end there, as it also offers the chance to capture video calls, screenshots at periodic intervals and even sound from around the PC itself just so long as a mic is present. This is all collected in log files that are stored locally and can only be accessed by the password holder.

One big drawback, like other keylogging programs, is that there is no facility to block access to certain websites and any parents that want to do that will need another application to supplement it. As a keylogger it does the job it sets out to do and in that sense Kidlogger will not disappoint.

### What I will do differently

The problem with preexisting software is that the algorithms only judge sites as a whole, and indiscriminately block entire sites, which might only have one page with dubious content.

For example, reddit, a site whose content ranges from adult only to appropriate for all ages depending on the section, should not be blocked as a whole. This universal blocking will be avoided through the application of a priority code base and the api.

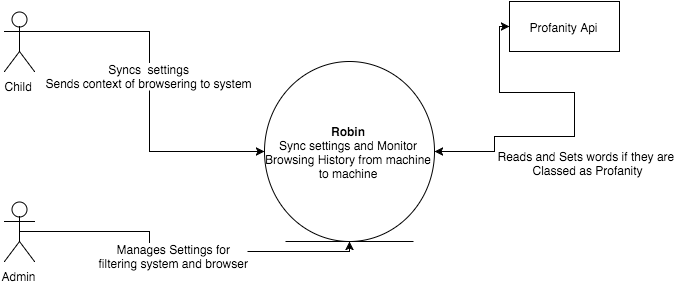
I will also develop the system to be used across multiple devices using up to date programmes, such as Node, Firebase and WebKit. The system will be available on Mac, Windows, Android and iOS. Another feature which the system lacks is the possibility of using cross syncing, which will allow the settings to be saved between devices.

# Functional Requirements

## 9.1 The Scope of the Work

This application will automate existing business processes for accessing the internet .It will provide 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.

## 9.2 Context Diagram



## 

## 9.3 Work Partitioning

I decided to undertake a task of building my a web browser in part (one to see how it will work) along with the use of building npm packages so that the code can be reused for the community as well.

Npm makes it easy for JavaScript developers to share and reuse code, and it makes it easy to update the code that you're sharing.

## 9.4 Business Use Cases

### Typical business events:

**Children should be able to**

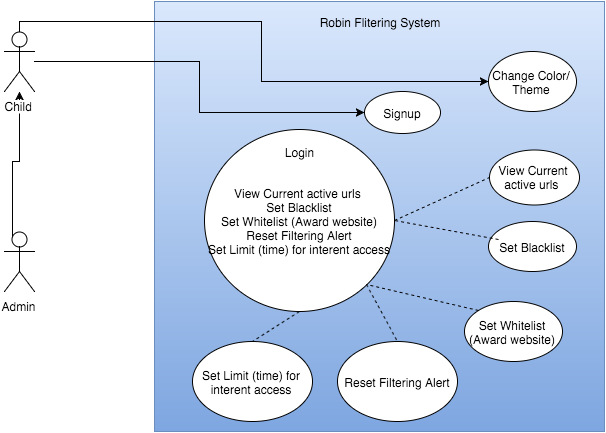
* Browse the web
* Go Back in history
* Go Forward
* Refresh Page
* Receive points for accessing good websites set by administrators.
* Change Color of Browser

**Administrator (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

## 9.5 Business Data Model

### Use case diagram:



## 9.6 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 | 1. Person inputs required information (email 2. System verifies information 3. System creates user account 4. 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 | 1. User inputs his email address and password into login form. 2. System checks if they match 3. 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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 |  |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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 |  |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #11 | Add Time limit |
| Actor | Admin |
| Precondition | Is a registered member. |
| Postcondition | System sets a time limit on child's laptop |
| Main Path | * User selects Set Time Limit ’. * System brings up ‘Time Limit ’ page. * User selects add Time Limit * System brings up a form. * User inputs time limit * User selects ‘Submit’. * System adds info to database. * System updates Set Time Limit’ page. |
| Alternative Path 1 | @8. User leaves a field blank and selects ‘Submit’.  System highlights the blank field. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #12 | Remove Time limit |
| Actor | Admin |
| Precondition | Is a registered member. |
| Postcondition | System removes Time limit |
| Main Path | * User selects Time limit ’. * System brings up ‘Time limit ’ page. * User selects Time limit from list * User selects remove * User selects ‘Submit’. * System removes info to database. * System updates ‘Time limit’ page. |
| Alternative Path 1 | @8. User leaves a field blank and selects ‘Submit’.  System highlights the blank field. |
| Alternative Path 2 |  |

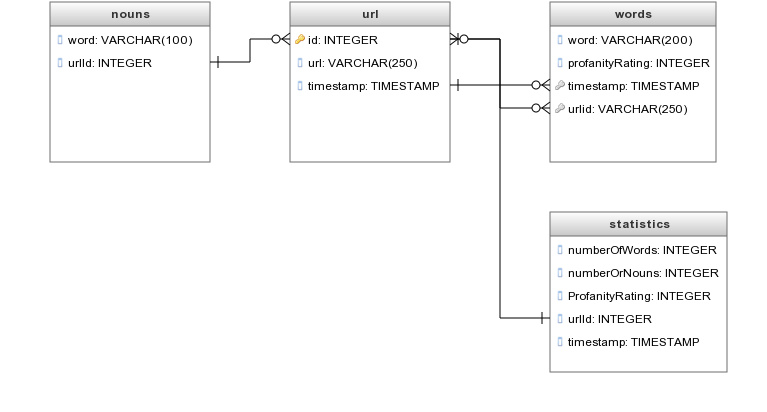
|  |  |
| --- | --- |
| Use Case #13 | Update Time limit |
| Actor | Admin |
| Precondition | Is a registered member. |
| Postcondition | System sets a time limit on child's laptop |
| Main Path | * User selects Time limit Page. * System brings up ‘Time limit ’ page. * User selects update Time limit * System brings up a form. * User inputs time * User selects ‘Submit’. * System adds info to database. * System updates ‘Time limit page. |
| Alternative Path 1 | @8. User leaves a field blank and selects ‘Submit’.  System highlights the blank field. |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| 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 |
| Alternative Path 2 |  |

|  |  |
| --- | --- |
| Use Case #15 | Reset Filtering Alert |
| Actor | Admin |
| Precondition | Is a registered member. |
| Postcondition | Reset color theme back to default, not black |
| Main Path | * User selects Reset filter * System brings up Reset filter ’ page. * User selects reset Reset filter * User selects ‘Submit’. * System adds info to database. * System updates ‘ Reset filter page. |
| Alternative Path 1 |  |
| Alternative Path 2 |  |

## 9.7 Data Model

## 



## 9.8 The Scope of the Product

Scope of product identifies boundaries between users and the product. It separates system automated processing from activities that must be provided by the user.

|  |  |
| --- | --- |
| **Actor** | **System** |
| Provide personal details | Store and maintain the information |
| Provide availability blacklist | Store information and display in schedule system |
| Provide limits and time | Store information and update appropriate user |
| Manage stored information | Provide managing capability for the stored information |

# 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 deleting of children browser accounts.

Rationale: Browser is no longer used that no longer need or child does not need filtering

Originator: Admin.

Fit Criteria: Accounts that are deleted should not leave any remaining details related to them.

Dependencies: Req#1

Priority: High.

History: Created: 10/10/2015

### Requirement # 8

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 # 9

Description: Admins should be able to set a child’s session time.

Rationale: So the child only has access to 30mins once per day with the admin’s permission.

Originator: Admin

Fit Criteria: The results shall have a countdown on the admins panel.

Dependencies: Priority: Low

History: Created: 10/10/2015

### Requirement # 10

Description: Send an email with a message when the child does on profanity site.

Rationale: So the parent admin can see the information what time and url.

Originator: Admin

Fit Criteria: The results will be emailed

Dependencies: Priority: Low

History: Created: 10/10/2015

### Requirement # 11

### 

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: Priority: Low

History: Created: 10/10/2015

### Requirement # 2

Description: The application run profanity checks on all webpages (not websites)

Rationale:

Originator: Admins and Children

Fit Criteria: Stores the information into the profanity database.

Dependencies: None

Priority: Very high

History: Created: 10/10/2015

# Non-functional Requirements & Background Research

## 11.1 Design: Material Design

Material Design is a design language developed by Google. It makes liberal use of grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. Google states that this new design language is based on paper and ink.

Material Design will gradually be extended throughout Google's array of web and mobile products, providing a consistent experience across all platforms and applications. All good applications on the Google Play Store have now based their designs around the Material Design Guidelines. To aid designers, Google has released typefaces, icons and interface layouts that should be used when designing apps for their app store. I downloaded these useful resources and used them in my design.

As many popular apps on the Google Play Store have already begun using the Material Design layouts, people have already become familiar with the way the navigation works. By using the Material Design guidelines, my users will find my app design instantly familiar and easy to navigate.

## 11.2 Nielsen’s Usability Heuristics

While researching how to design for the user, I discovered Jakob Nielsen's 10 general principles for interaction design. They are called "heuristics" because they are broad rules of thumb and not specific usability guidelines:

1. Visibility of System Status
2. Match Between System and the Real World
3. User Control and Freedom
4. Consistency and Standards
5. Error Prevention
6. Recognition Rather Than Recall
7. Flexibility and Efficiency of Use
8. Aesthetic and Minimalist Design
9. Help Users Recognize, Diagnose, and Recover from Errors
10. Help and Documentation

I felt that following these guidelines allowed me to create an app that was easy for the user to navigate and understand.

## 11.3 Personalisation and Internationalisation

The application shall allow the user to customize basic appearance, such as the colour scheme.

It has also potential for expansion to other countries by implementing a range of different languages.

## 11.4 Accessibility

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

## 11.5 Speed and latency (Performance)

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

## 11.6 Reliability and availability

The application shall be available online 24 hours/day and 365 days/year. Short period of time might be scheduled for maintenance with prior notice given.

## 11.7 Capacity

The application should be able to serve 100 simultaneous users. The volume of database should be able to accommodate 1000000 different records per table.

## 11.8 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.

## 11.9 Ease of use

The goal of the application is to allow the user to perform his main tasks and activities within 4 clicks. 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.

## 

## 11.10 Learning

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

## 11.11 Security

Firebase is secure. However, 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 dubbed "Security Rules".

Since these rules are on the server they cannot be overwritten by the client. Security rules are structured just like data in Firebase.

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

Security Rules are even more granular than this. I can write rules against a specific location in Firebase. In this case I am validating that all new sparks have an author and content node.

{

".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 a really easy to configure ruleset to prevent this.

In Terms of encryption firebase uses bcrypt. 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** \gets **InitState()  
 state** \gets **ExpandKey(state, salt, key)  
 repeat (2cost)  
 state** \gets **ExpandKey(state, 0, key)  
 state** \gets **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** \gets **EksBlowfishSetup(cost, salt, input)  
 ctext** \gets **"OrpheanBeholderScryDoubt" //three 64-bit blocks  
 repeat (64)  
 ctext** \gets **EncryptECB(state, ctext) //encrypt using standard Blowfish in ECB mode  
 return Concatenate(cost, salt, ctext)**

Form and controls provide validation services, so that the user can be notified of invalid input before submitting a form. This provides 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.

# Background Research : Tools

## 12.1 Node js

Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i

**Pros**

1. I already know JavaScript

2. It's fast

Node.js is a JavaScript runtime that uses the V8 engine developed by Google for use in Chrome. V8 compiles and executes JavaScript at lightning speeds mainly due to the fact that V8 compiles JavaScript into native machine code.

3. Tooling

Npm is the Node.js package manager and it is excellent. It does, of course, resemble package managers from other ecosystems, but npm is fast, robust, and consistent. It does a great job at specifying and installing project dependencies. It keeps packages isolated from other projects, avoiding version conflicts. But it also handles global installs of shell commands and platform-dependent binaries. I can't remember a time with npm where I've had to ask myself, "Why are those modules conflicting? Where is that module installed? Why is it picking up this version and not that one?"

4. Real-time Made Easy

If Node.js excels at many concurrent connections, then it makes sense that it excels at multi-user, real-time web applications like chat and games. Node's event loop takes care of the multi-user requirement. The real-time power comes thru use of the websocket protocol. Websockets are simply two-way communications channels between the client and server. So the server can push data to the client just as easily as the client can. Websockets run over TCP, avoiding the overhead of HTTP.

5. One Codebase And Your Real-time For Free

If you've made it this far, you may ask yourself, "If Node.js allows me to write JavaScript on the client and server, and makes it easy to send data between the client and server, can I write a web app that runs a single codebase on both client and server, and automatically synchronizes data between the two?"

## 12.2 Node Webkit

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.

It was created in the Intel Open Source Technology Center.

* 120,000+ npm packages. Granted, a lot are duplicates. But that's still a huge number and the community is awesome.
* Cross-platform - this is not Node.js specific, but it's one benefit.

By cross-platform, I don't mean just that you can use it across OS-es, I mean that you can build modules which you'll use both in your desktop application and in your possible web tier.

JavaScript - node-webkit is familiar ground for web developers - There are estimates that say that 50% of the world's devs know JavaScript. Again, most of them only know basics (or only know jQuery), but it is still a lot of people.

Also, a few technical points:

event-driven -Node.js is event-driven environment. Depending on the type of app, you can take use of this (user clicks - event. user types - event. user hit's Save - event.)

async - Node.js and it's async I/O mean the UI will be less blocking than some might be (granted, probably apps in most languages/platforms can be made this way, but you did not ask about most languages/platforms)).

## 12.3 Firebase

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

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

### Automatically scales with your app

When your app is a breakout hit, you don't have to worry about scaling your server code or provisioning extra capacity — Firebase handles that automatically for you. Fireabses servers manage millions of concurrent connections and billions of operations per month.

### First-class security features

All of your data is transferred over a secure SSL connection with a 2048-bit certificate. Database access and validation is controlled at a granular level using firebases flexible security rules language. All of your data security logic is centralized in one place making it easy to update and verify.

### Works offline

Your Firebase app will remain responsive regardless of network latency or 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.

### Pros

* If your app does run off a centralized DB, and is updated by a lot of users - then it's more than capable of handling the Real-Time data updates between devices.
* Stored in the cloud so readily available everywhere.
* Cross Platform API (If you are using this DB with an App)
* They host the data. Meaning that if you are storing a lot of data, you don't have to worry about hardware!

### Cons:

* Unless your app runs of one centralized database updated by a vast quantity of users, it's a major overkill.
* Storage format is entirely different to that of SQL, (Firebase uses JSON) so you wouldn't be able to migrate that easily.
* Reporting tools won't be anywhere near the ones of standard SQL.
* Limited to 50 Connections and 100mb of Storage.
* You don't host the data, Firebase does. And depending on which server you get put on, viewing there up time there seems to be a lot of disruption lately.

# Literature Review : Filtering Systems

For nearly ten years, the issue of Internet filtering has consumed legislators, educators, advocates, study committees, and courts. Despite the well-documented problem of over-blocking (censoring material that is non-pornographic and intellectually valuable), filters are now widely used in schools, libraries, and other centers of learning. In the interest of helping the public understand the issue, I have addressed major issues in this report on overblocking, in particular, how to improve on this issue.

In the late 1990s, rating and filtering systems were developed in response to concerns about pornography and other controversial material on the Internet. Companies began marketing the software to schools and libraries.

The Clinton Administration encouraged filtering as a response to a 1997 Supreme Court decision striking down the Communications Decency Act (CDA), which, in an attempt to block minors from Internet pornography, criminalized virtually all "indecent" or "patently offensive" communications online.[1](http://www.fepproject.org/factsheets/filtering.html#N_1_)

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

Groups such as Peacefire and The Censorware Project began documenting the problem of erroneous blocking, with examples ranging 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 conduct a study on "Tools and Strategies for Protecting Kids From Pornography and Their Applicability to Other Inappropriate Internet Content." The NRC established a committee that held hearings and conducted extensive research.

In May 2002, the NRC released a 402-page report, Youth, Pornography, and the Internet, which noted that because filters rely "on machine-executable rules abstracted from human judgments," they necessarily identify "a large volume of appropriate material as inappropriate.”5

Its to be noted that all internet filters are created by private companies and not CIPA (Children’s Internet Protection Act). So it is the private companies who decide what content is to be blocked and what should be allowed. Instead it should be the CIPA to decide what content is appropriate for students and what is not. This leads to sites being blocked or filtered despite the fact that they do not fall under the criteria set by CIPA, whicher greatly limits the web’s learning and education possibilities. 6

## Body

At the start of the new millennium, some major filter manufacturers claimed to have corrected the problem of over-blocking and to have abandoned reliance on keywords in favor of "artificial intelligence." "Artificial intelligence," however, is simply a more revealing form of keyword blocking. Studies and reports continued to document the erroneous blocking of thousands of educational sites, in particular of sex education sites.8

Along with over-blocking, some filtering software also under-blocks - that is, they fail to identify and block many pornographic sites. Filters initially operate by searching the World Wide Web, or "harvesting," for possibly inappropriate sites, largely relying on keywords and phrases. There follows a process of "winnowing," which also relies largely on these mechanical techniques.9

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 small staffs (between eight and a few dozen people) can give at most a cursory review to a fraction of these sites, and human error is inevitable.

Filtering company employees' judgments are also necessarily subjective, and reflect their employer's' social and political views. Some filtering systems reflect conservative religious views .Filters frequently 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 weapons, including the American Museum of Natural History; and other educational sites such as Planned Parenthood, CNN, and sites discussing anorexia and bulimia.

Despite these well-known problems, 75% of public schools adopted some form of Internet filtering even before Congress required them to do so as part of 2000 "Children's Internet Protection Act."

## 13.2 Conclusion

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

Another 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 profane 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 educating youth about media literacy and sexual values, along with frustrating and restricting research into health, science, politics, arts, and many other educational areas.

https://github.com/Projectbird/profanity

# Literature Review : Machine Learning Algorithms

Machine learning algorithms are organized into taxonomy, based on the desired outcome of the algorithm. Common 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 in 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 own 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 machine 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, I will create a key rule based algorithm. I will manually write down rules that will be matched against an input, and if a match is found, you fire a corresponding action, in this case, checking for profanity and nouns.

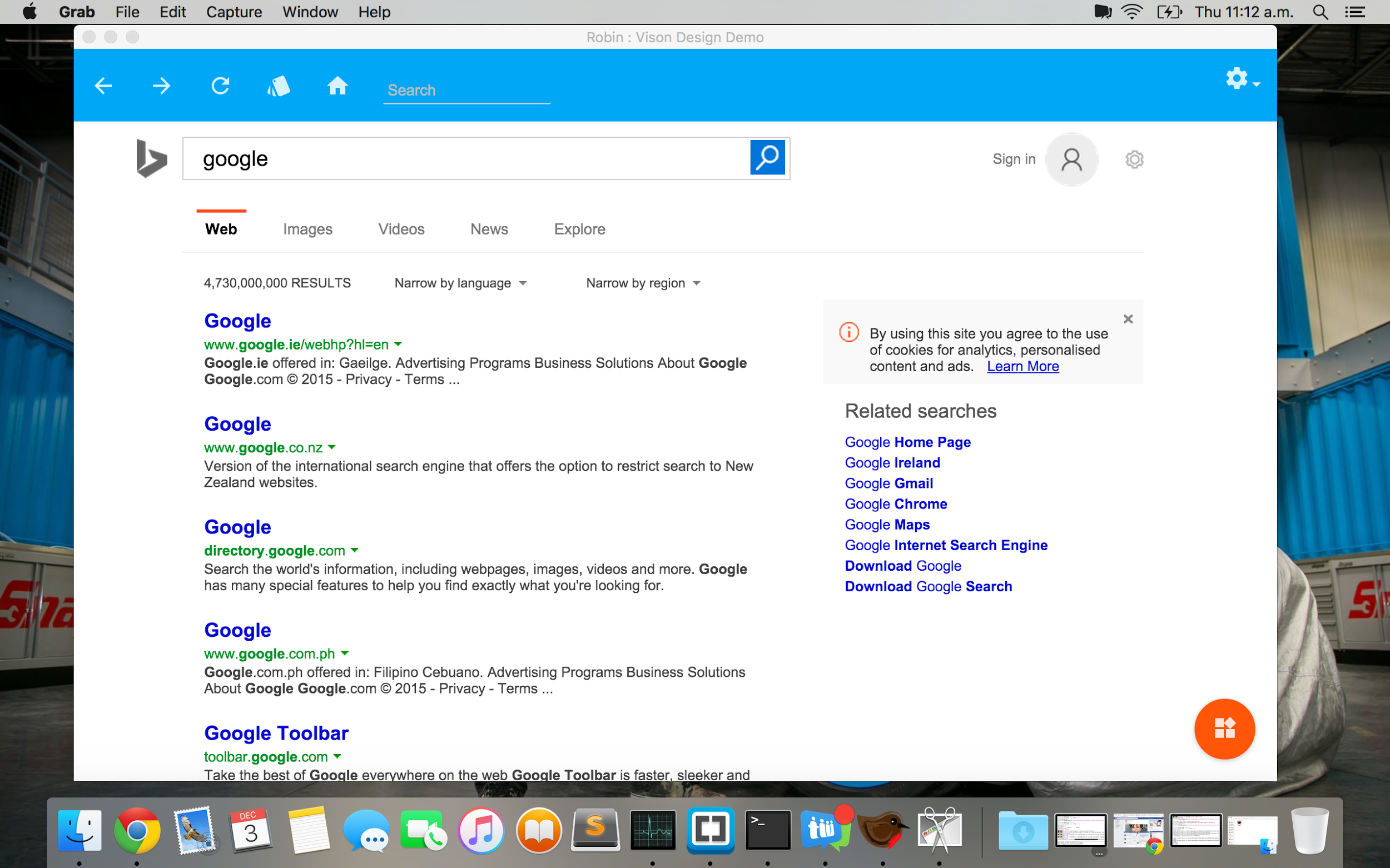
Using this should help not to restrict the format of the users’ input and to come up with rules which, while remaining as general as possible, should still change over time as more information and data sets come to a label.

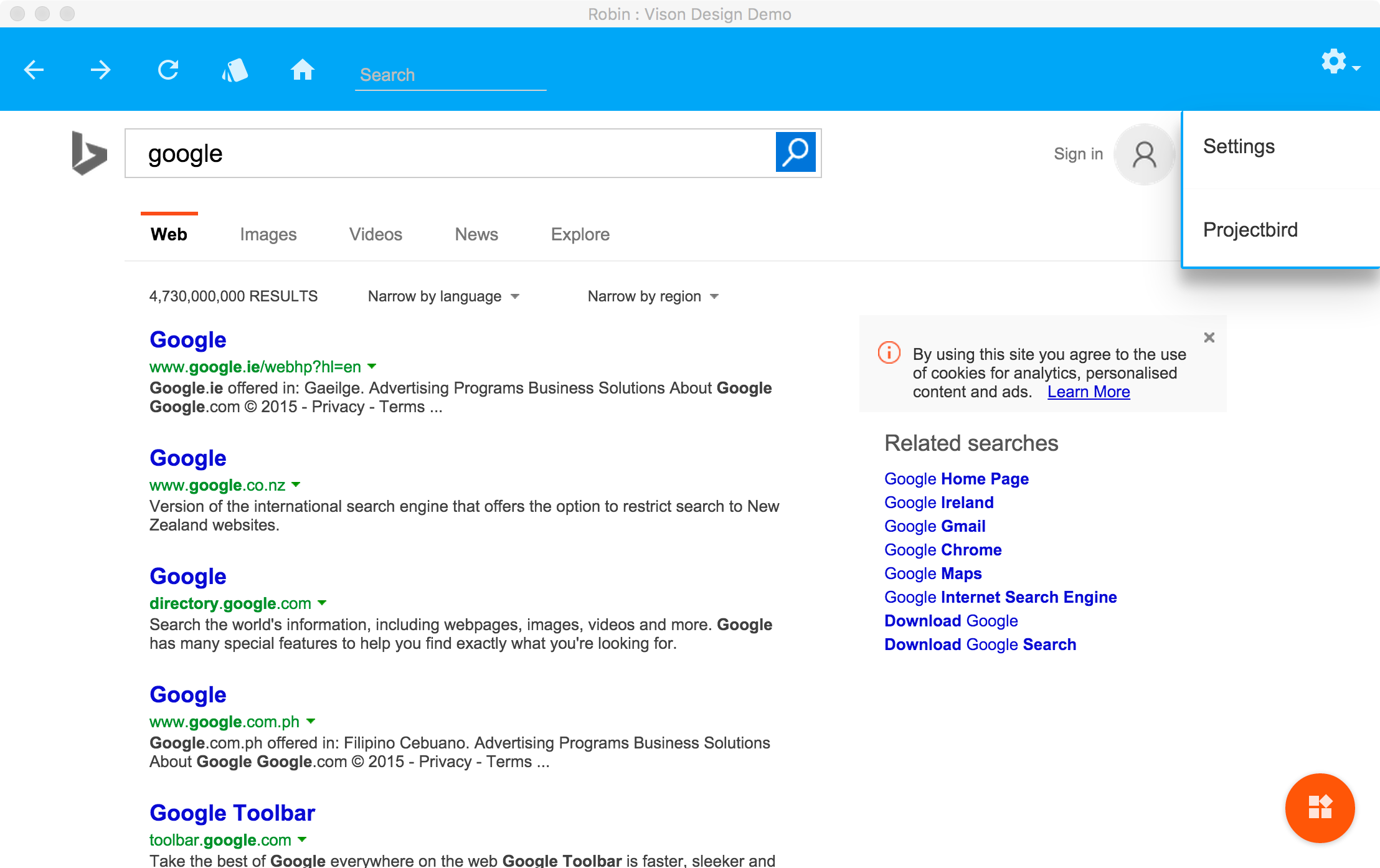
For example, instead of blocking the word "murder,” which can have a double meaning (i.e. a murder of crows vs to murder somebody), you can have a rule such as: unless the word "murder" occurs in the command, don't start the program, OR ignore every sentence unless it contains "murder" which is counted as profanity. Then, I will combine my rules to develop more complex "understanding". How to write/represent rules is another tough problem. This will be done using Regular Expressions.

Therefore my rules will be based on profanity classed words and the use of verbs in the sentence.

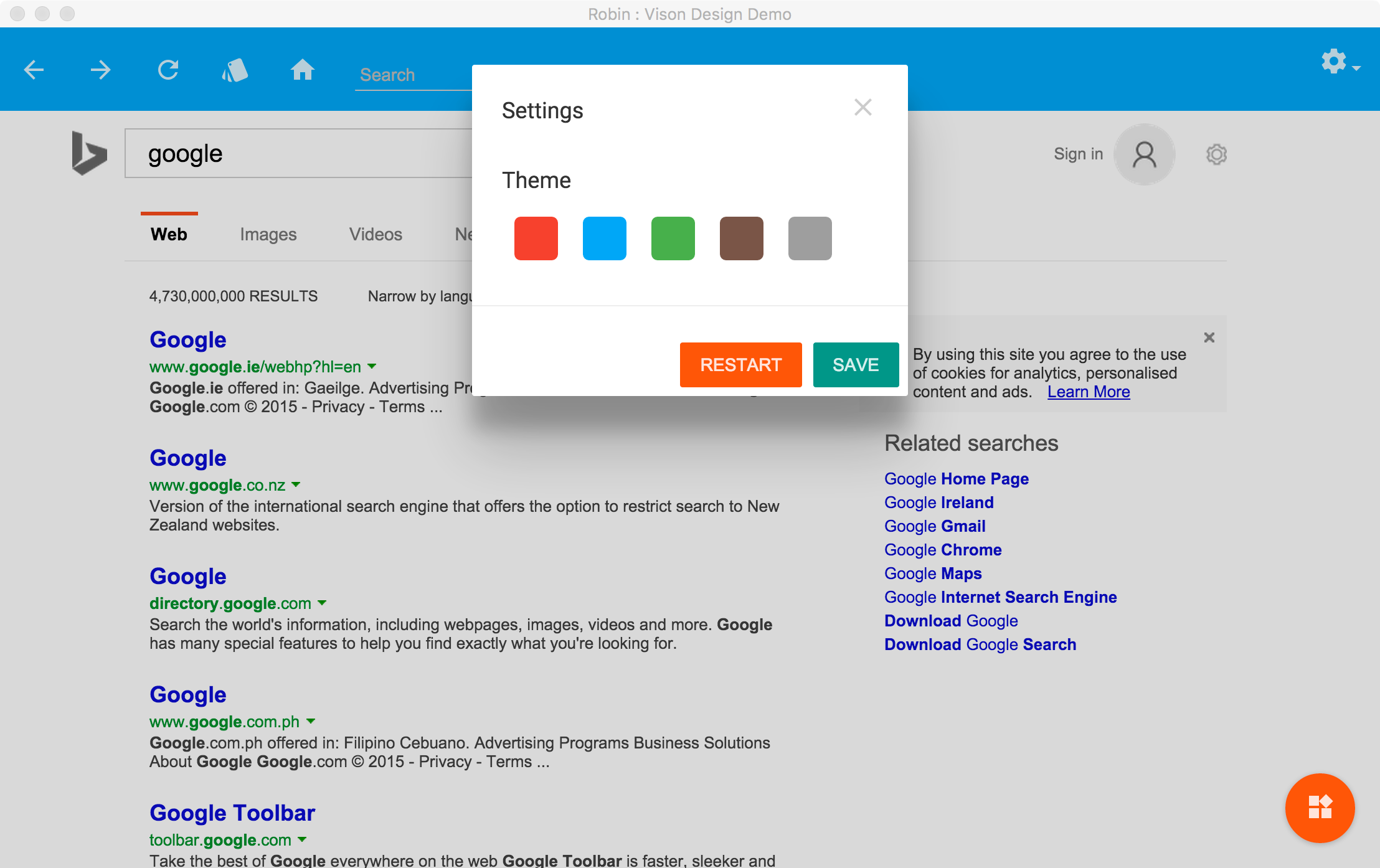
# Initial Design & prototype

Below are all the major screens for the design, to view other designs before the usebabity testing was done please visit, such as focus groups and more. : http://www.projectbird.com/robin.html

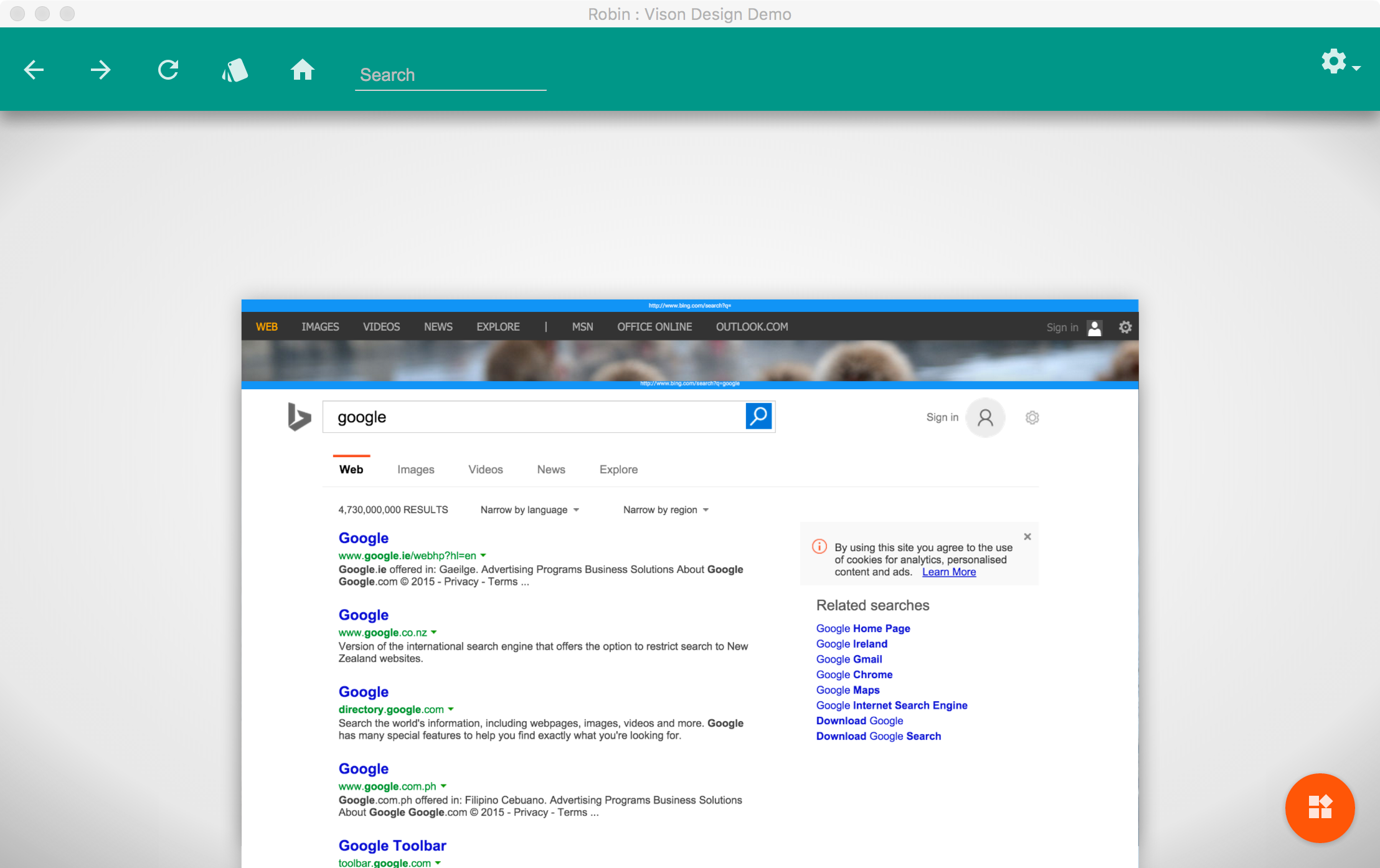
Browser Windows for Child



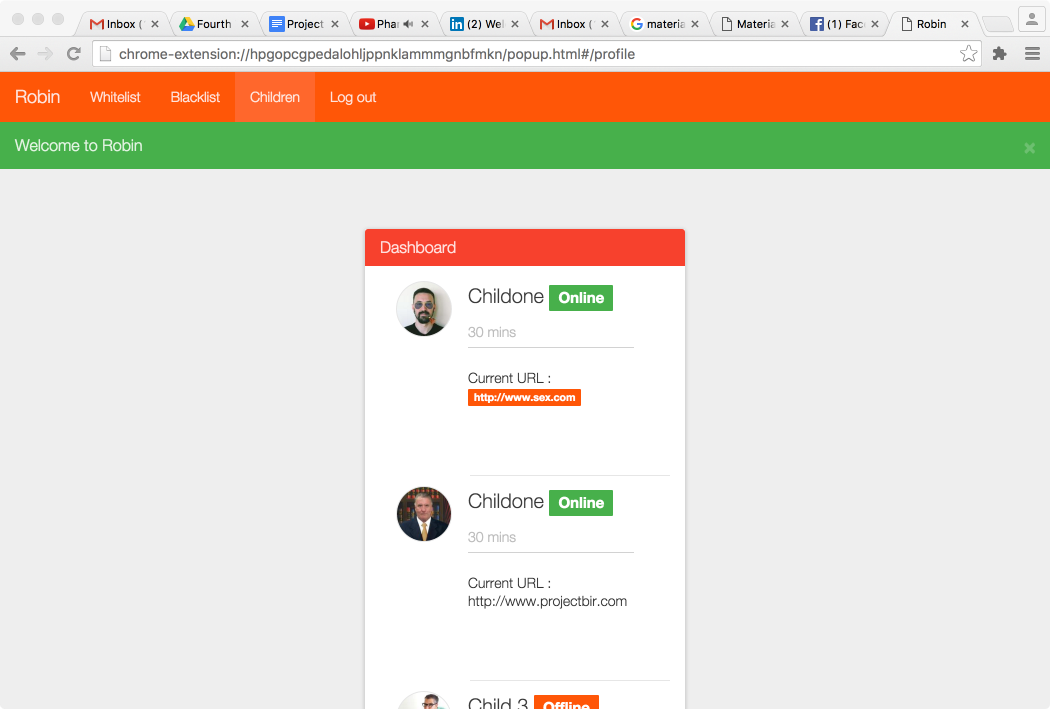
Settings Tab for browser



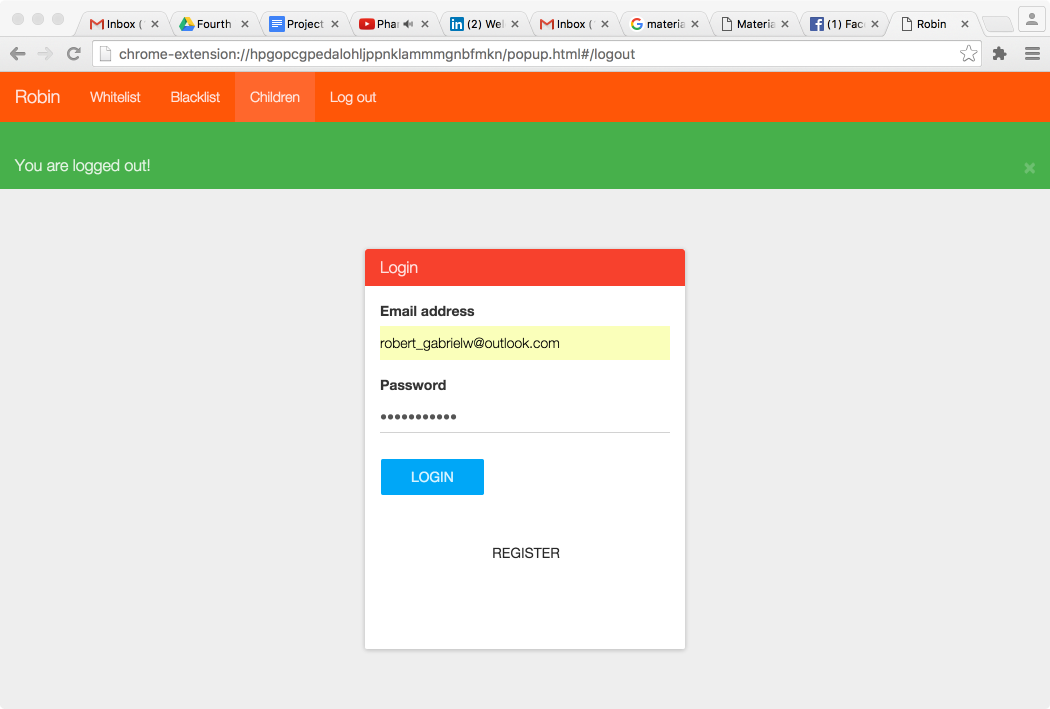
Theme Page



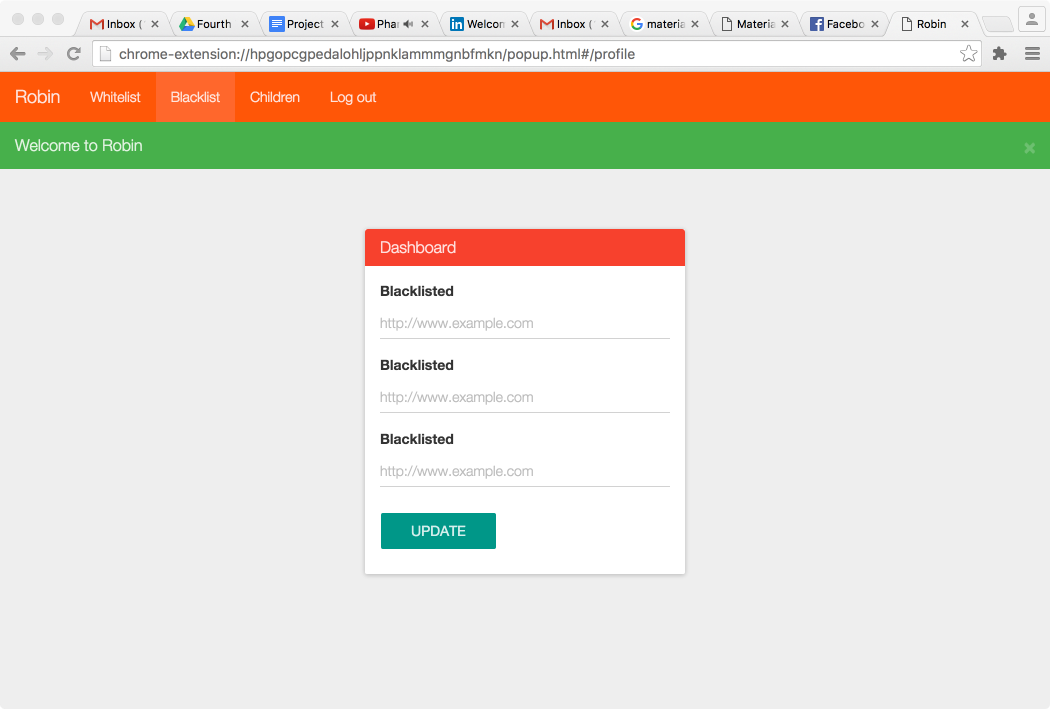
Tabs : View Tabs on a browser



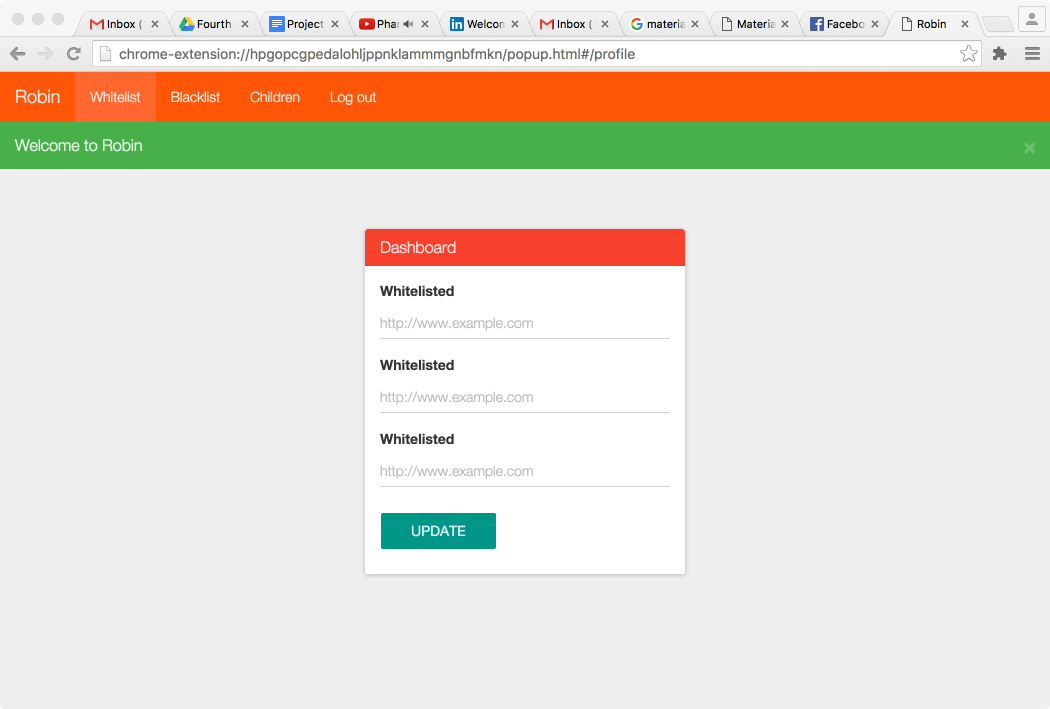
Child Page, so the admin can set limits and see the current url of the child's page.



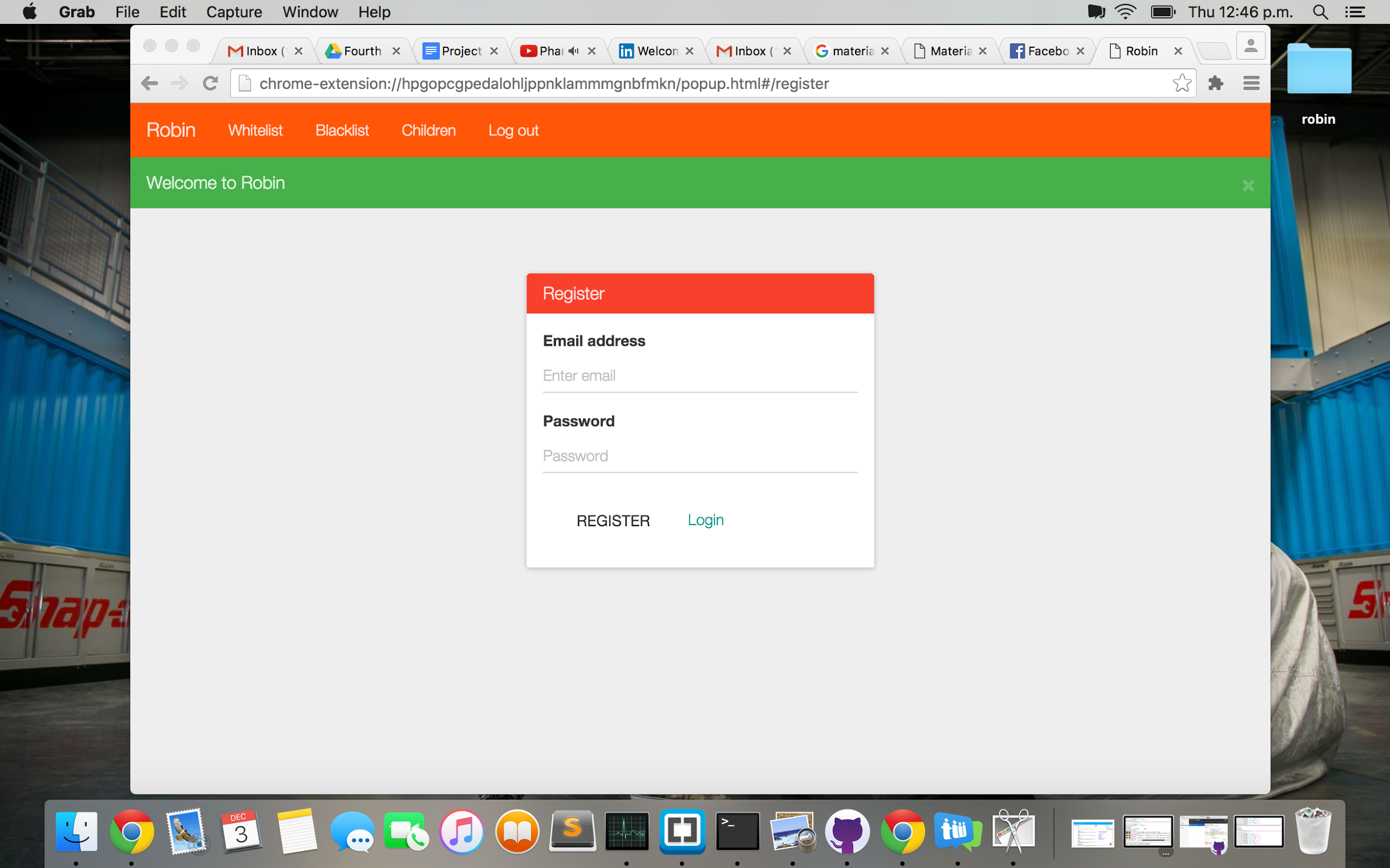
Logout page.



Blacklist. the user can block WebKit completely even if the algorithm disagrees.



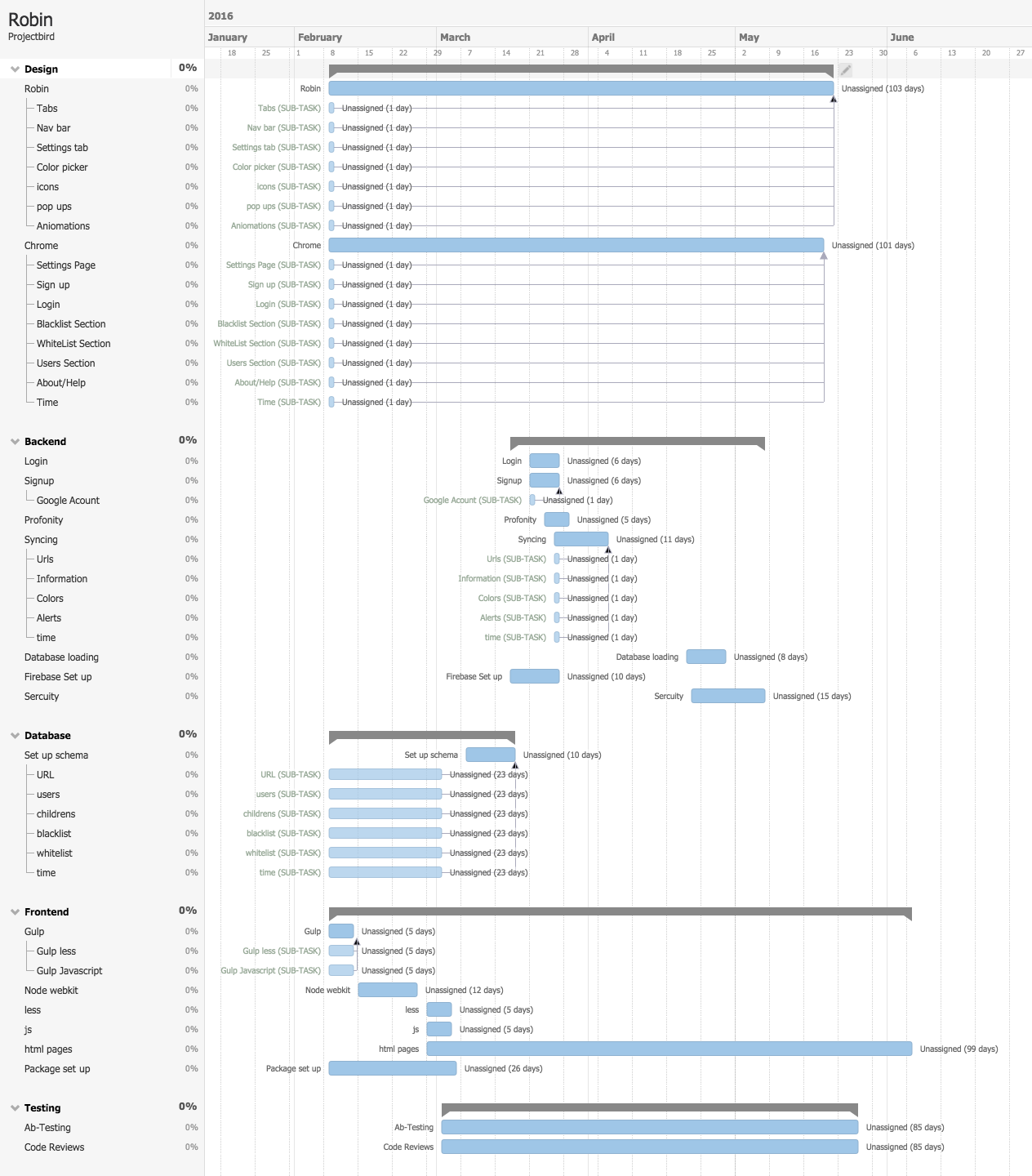
Whitelist where the user can input a good site, this will alert the user.



Login/Register page.

.

# Workflow/ Project Plan for the implementation phase

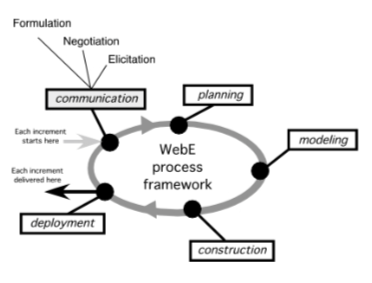


For code testing , I will be using a method used a lot in Teamwork, the agile form of testing which is when you test the software as you go. As I will build a feature, I test it, making sure performance and sql is created before moving on.

Agile testing is a huge benefit in my case as it will allow more bugs to be detected and published, 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 lets me fix the bug 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.



## Milestones and associated tasks

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Milestone | Description | Date Due | Responsible | Status | Days Late |
|  | Version 1 |  | 27 Feb (2016) | Robert G. | Upcoming |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Milestone | Description | Date Due | Responsible | Status | Days Late |
|  | Version 2 |  | 25 Mar (2016) | Robert G. | Upcoming |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Milestone | Description | Date Due | Responsible | Status | Days Late |
|  | Version 3 |  | 23 Apr (2016) | Robert G. | Upcoming |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Milestone | Description | Date Due | Responsible | Status | Days Late |
|  | Final Version |  | 24 May (2016) | Robert G. | Upcoming |  |

## 

## Design (Tasklist)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task** | **Description** | **Start Date** | **Date Due** | **Est.** | **Time** |
|  | Chrome |  | 08 Feb (2016) | 18 May (2016) | 2 hrs | None |
|  | • Settings Page |  | 08 Feb (2016) |  | None | None |
|  | • Sign up |  | 08 Feb (2016) |  | None | None |
|  | • Login |  | 08 Feb (2016) |  | None | None |
|  | • Blacklist Section |  | 08 Feb (2016) |  | None | None |
|  | • WhiteList Section |  | 08 Feb (2016) |  | None | None |
|  | • Users Section |  | 08 Feb (2016) |  | None | None |
|  | • About/Help |  | 08 Feb (2016) |  | None | None |
|  | • Time |  | 08 Feb (2016) |  | None | None |
|  | Robin |  | 08 Feb (2016) | 20 May (2016) | 2 hrs | None |
|  | • Tabs |  | 08 Feb (2016) |  | None | None |
|  | • Nav bar |  | 08 Feb (2016) |  | None | None |
|  | • Settings tab |  | 08 Feb (2016) |  | None | None |
|  | • Color picker |  | 08 Feb (2016) |  | None | None |
|  | • icons |  | 08 Feb (2016) |  | None | None |
|  | • pop ups |  | 08 Feb (2016) |  | None | None |
|  | • Animations |  | 08 Feb (2016) |  | None | None |
|  | | | | | **4 hrs** | **None** |

### Backend (Tasklist)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task** | **Description** | **Start Date** | **Date Due** | **Est.** | **Time** |
|  | Login |  | 20 Mar (2016) | 25 Mar (2016) | 2 hrs 37 mins | None |
|  | Signup |  | 20 Mar (2016) | 25 Mar (2016) | 46 mins | None |
|  | • Google Acount |  | 20 Mar (2016) |  | None | None |
|  | Firebase Set up |  | 16 Mar (2016) | 25 Mar (2016) | 1 hr | None |
|  | Profanity |  | 23 Mar (2016) | 27 Mar (2016) | 10 hrs | None |
|  | Syncing |  | 25 Mar (2016) | 04 Apr (2016) | 1 hr 55 mins | None |
|  | • Urls |  | 25 Mar (2016) |  | None | None |
|  | • Information |  | 25 Mar (2016) |  | None | None |
|  | • Colors |  | 25 Mar (2016) |  | None | None |
|  | • Alerts |  | 25 Mar (2016) |  | None | None |
|  | • time |  | 25 Mar (2016) |  | None | None |
|  | Database loading |  | 21 Apr (2016) | 28 Apr (2016) | 48 mins | None |
|  | Security |  | 22 Apr (2016) | 06 May (2016) | 3 hr | None |
|  | | | | | **20 hrs 6 mins** | **None** |

### Database (Tasklist)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task** | **Description** | **Start Date** | **Date Due** | **Est.** | **Time** |
|  | Set up schema |  | 07 Mar (2016) | 16 Mar (2016) | 1 hr 35 mins | None |
|  | • URL |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | • users |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | • childrens |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | • blacklist |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | • whitelist |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | • time |  | 08 Feb (2016) | 01 Mar (2016) | None | None |
|  | | | | | **1 hr 35 mins** | **None** |

### Frontend (Tasklist)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task** | **Description** | **Start Date** | **Date Due** | **Est.** | **Time** |
|  | Gulp |  | 08 Feb (2016) | 12 Feb (2016) | 43 mins | None |
|  | • Gulp less |  | 08 Feb (2016) | 12 Feb (2016) | None | None |
|  | • Gulp Javascript |  | 08 Feb (2016) | 12 Feb (2016) | None | None |
|  | Node webkit |  | 14 Feb (2016) | 25 Feb (2016) | 1 hr | None |
|  | less |  | 28 Feb (2016) | 03 Mar (2016) | 1 hr 12 mins | None |
|  | js |  | 28 Feb (2016) | 03 Mar (2016) | 1 hr 44 mins | None |
|  | Package set up |  | 08 Feb (2016) | 04 Mar (2016) | 54 mins | None |
|  | html pages |  | 28 Feb (2016) | 05 Jun (2016) | 1 hr 5 mins | None |
|  | | | | | **6 hrs 38 mins** | **None** |

### Testing (Tasklist)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Task** | **Description** | **Start Date** | **Date Due** | **Est.** | **Time** |
|  | Ab-Testing |  | 02 Mar (2016) | 25 May (2016) | 5 hrs | None |
|  | Code Reviews |  | 02 Mar (2016) | 25 May (2016) | 6 hrs | None |
|  | | | | | **11 hrs** | **None** |

# Conclusion

From my research into both natural language algorithms and machine learning, I believe that through my use of ever evolving personal key rules, I can create a programme which will be able to keep up with the ever changing as 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, acceptable in context, and a more pejorative meaning. My software on the other hand, will have the ability to interpret language based on context and as such, it has the capacity to revolutionise child protection on the internet.

Through outline detailed in my report, it is clear that I'm intending to use the the most up to date and advanced tools currently available such as node.js and Firebase and that I have planned it in every detail, leading to a cutting edge project, which I am confident will be published in a paper by the end of the next semester. There is a lot of work ahead but though the use modern technologies recently developed and opportunities offered by CIT, I am certain that it is achievable.

# REFERENCES

## 16.1 Tools

### Gulp

There’s no point in investing your time in learning a new tool if you don’t even know what problem it solves. 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 creates more time to do non repetitive tasks which increases productivity.

Gulp is a javascript task runner that lets you automate tasks such as…

* Bundling and minifying libraries and stylesheets.
* Refreshing your browser when you save a file.
* Quickly running unit tests
* Running code analysis
* Less/Sass to CSS compilation
* Copying modified files to an output directory

<http://brandonclapp.com/what-is-gulp-js-and-why-use-it/>

### Node.js

Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i.

### Node webkit

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.

### Javascript

an object-oriented computer programming language commonly used to create interactive effects within web browsers.

### Angular.js

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets 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.

### Html5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It was finalized, and published, on 28 October 2014 by the World Wide Web Consortium (W3C). This is the fifth revision of the HTML standard since the inception of the World Wide Web.

### Css3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language.

### Less

Less.js is a CSS pre-processor, it means that you will be able to write your in style using the Less preprocessor languages in a \*.less file, then Less.js will compile it into pure CSS. CSS pre-processor comes with a lot of powerful features such as: variables, mixins, nested rules, operations, imports and a lot more

### Firebase

Firebase can power your app's backend, including data storage, user authentication, static hosting, and more. Focus on creating extraordinary user experiences. We'll take care of the rest.

### Material Design

Material Design (codenamed Quantum Paper) is a design language developed by Google. Expanding upon the "card" motifs that debuted in Google Now, Material Design makes more liberal use of grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows.

# Survey

### Name

### Age

### What's your job title if any?

### Which of the Following Do You Use?

Internet Explorer

Chrome

Edge

Firefox

Opera

Others

### Do you use add ons or themes?

Yes

No

### Have you used filtering software on your computer at home?

Yes

No

### Do you have any of the following in the household?

Smart Phone

Tablet

Other

Laptop

Games Console

### Do you monitor what your children access on the web?

Yes

No

Sometimes

### Do you reward your child for behaving on the internet?

Yes

No

### Are you in the same room when your child is on the computer ?

Yes

No

## References

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](http://www.nap.edu/books/0309082749/html)

5. Commission on Child Online Protection (COPA), [Report to Congress](http://www.copacommission.org/report/) (Oct. 20, 2000)

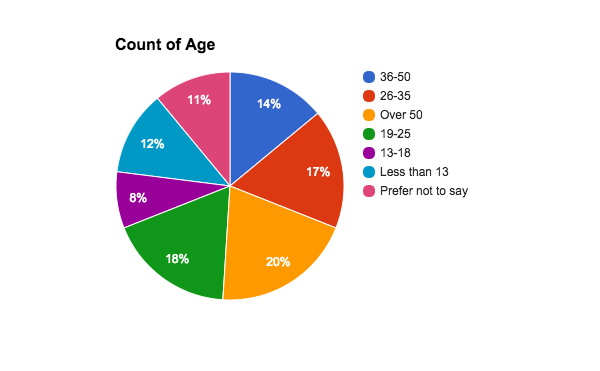
6. See Commission on Child Online Protection (COPA), [Report to Congress](http://www.copacommission.org/report/)(Oct. 20, 2000)

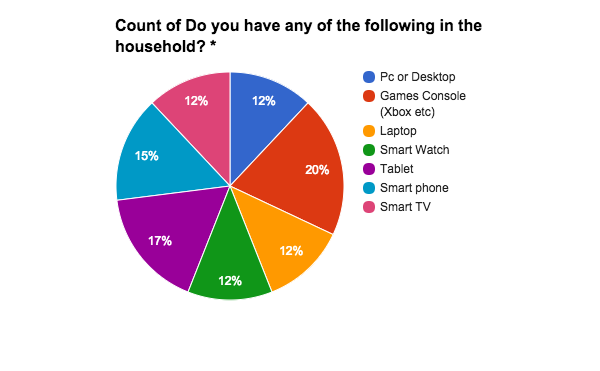
7. Commission on Child Online Protection (COPA), [Report to Congress](http://www.copacommission.org/report/) (Oct. 20, 2000) 8

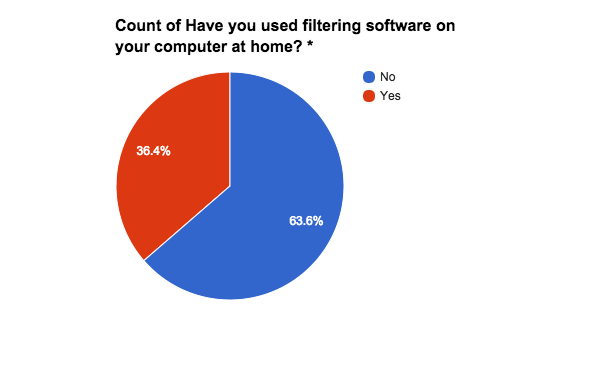
8. http://json.org/example.html

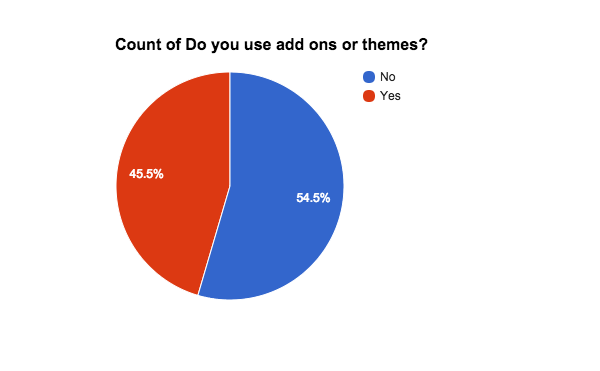
9 https://github.com/Projectbird/profanity

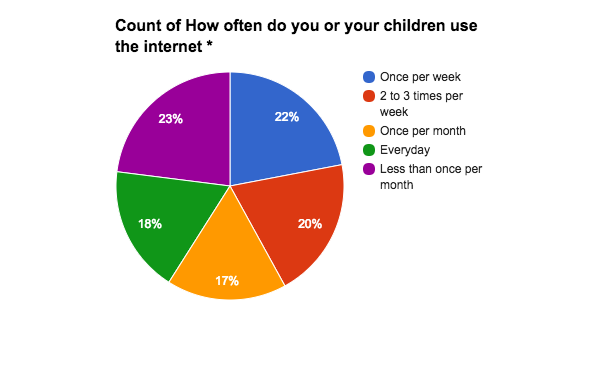
# Survey Results

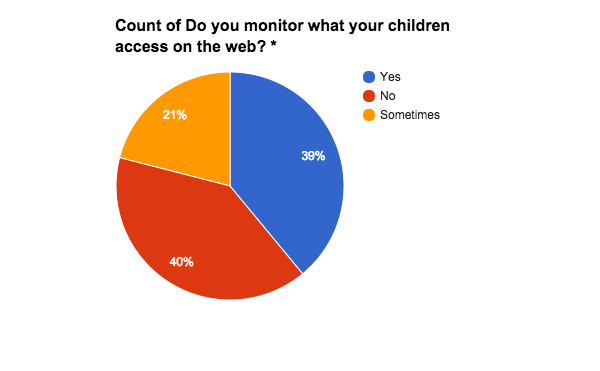


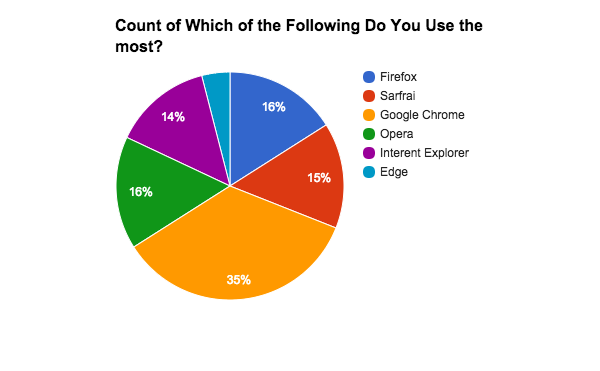


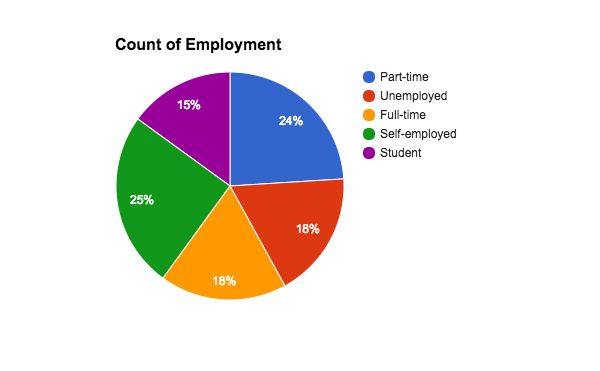












www.fepproject.org/factsheets/filtering.html