**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

**Prepared by:** Robert James Gabriel

Student Id: R00102430

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

**Visit :**<http://www.projectbird.com/robin> for more

Table of Contents

Introduction. 4

Overview.. 5

Project Drivers. 6

The purpose of the project. 6

Web Browser. 6

Chrome Extension. 6

Overall Project Goal 7

The Stakeholders. 7

Project Objectives. 8

Project Tools. 8

Learning Outcomes. 9

Literature review.. 10

Filtering Systems. 10

Body. 11

Conclusion. 12

Angular vs Knockout.js. 13

Data Binding. 13

Routing. 14

Browser Support. 14

Ecosystem.. 14

Other Features. 15

Conclusion. 16

Machine Learning Algorithms. 17

Nw.js Vs Election. 19

Features List. 19

Conclusion. 20

Competition in the Market. 21

Introduction. 21

Spy Agent: 21

Net Nanny: 21

Qustodio: 21

Features. 22

Conclusion. 22

Firebase. 23

Automatically scales with your app. 23

First-class security features. 23

Works offline. 23

Concussions. 24

ANALYSIS AND DESIGN.. 25

The Scope of the Work. 25

Context Diagram.. 25

Business Use Cases. 25

Business Data Model 26

Use case diagram.. 26

Use Cases. 27

Design and Implementation. 31

Design. 32

Material design. 32

Motion. 32

Layout 32

.. 32

Code. 32

**Nw.js and Gulp**. 33

**Angular.js, L.js and JavaScript**. 36

Problems. 37

Naive Bayes Classifier Algorithm.. 38

Adding Extra Features and Tweaks. 39

Datasets/Training. 39

Database / Firebase. 40

Sequence diagrams. 41

Architecture diagrams. 42

Security. 43

Results & TESTING.. 45

Syncing Database Bandwidth. 45

Performance. 45

Blocking/Algorithm results. 45

Survey Results. 45

Conclusion. 46

# **Introduction**

This report contains the project research and execution which I have carried out for the purpose of creating an innovative new way of filtering children’s internet access for parents and teachers. 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. Users 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, I had intended to create the application for the one machine only, but the surveys and research proved this was annoying.

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, 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 Web node Kit and a chrome extension whereby users (parents) could quickly 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 were to access the information, it would redirect them back to Google and alert the parent about the attempt though the chrome extension. 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 about this problem.

A vast area of my research and development is machine learning, the Baylor classifier algorithm, 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 context of a page, for 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, its 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 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. 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.

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.

From an algorithm blocking point of few, I developed 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. The algorithm used in this node module is based on Naive Bayes Classifier.

## **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 users (admin) will have to input their email. This is for the firebase so the application can sync information from firebase 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 website 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 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 user (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 users (child) selection.

**​Google Chrome Extension**: Set Blacklist sites, Set White list 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 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 collect data and begin to understand hopefully the context of a sentence, 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 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 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.

· 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 add-on 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 the context of a sentence using natural learning and machine learning algorithms.

### **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

· Angular’s

· 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 percentage rating

· Build a node module for parsing texts

· Build a node module for checking text profanity,

· Code up to the web standards on strict settings.

· Learn and develop in Angular.js.

· 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.

# **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 over blocking, 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

The over-blocking tendencies of Internet filters soon became 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 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.”

It's 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, which significantly limits the web’s learning and education possibilities.

### **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 for "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.

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, mostly relying on keywords and phrases. There follows 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 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."

### **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 judgment can be largely attributed to the fact that they do not follow the Children's Internet Protection Act but rather filter by their 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.

## **Angular vs Knockout.js**

Angular.JS and Knockouts are JavaScript libraries/frameworks that help create rich and responsive web UI interactions. Knockouts is a library that connects parts of the UI to a data model using declarative bindings. The same can be said about AngularJS, which is where the confusion comes from. The fundamental difference between the two solutions is that AngularJS manages the whole application and defines guidelines on how the application code should be structured, whereas with KnockoutJS the application structure is entirely up to you

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

### **Data Binding**

Data Binding is the process of establishing a connection between the application UI and the business logic. If the settings and notifications are configured correctly, the data reflects changes made in the UI. This also means that when the data is changed, the UI will represent that change.

In the HTML syntax for Knockout everything is done using the data-bind attribute and appropriate binding type; however, the need to specify all properties as observable requires additional effort. Mappings need to be performed when loading JSON data from the server to convert properties to observables. There is a mapping plugin that can be used to make this easier, but it is your responsibility to manage the mapping when retrieving data and when sending data back to the server.

Angular syntax for outputting values is much simpler and compact. It makes it easier to read and compose. The major difference is the binding methodology. Where Knockout binds to the provided model, Angular binds to the special $scopeobject.

Additionally, Knockout can only apply bindings once. If you try to apply bindings again it will throw an error. In Angular, however, scopes can be nested. This approach 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 management of application states. and back/forward browser history navigation. Knockout does not support routing.

[AngularUI Router](https://github.com/angular-ui/ui-router) is a routing framework for Angular.JS, which allows you to organize the parts of your interface into a state machine. Unlike the $route service in the Angular ngRoute module, which is organized around URL routes, UI-Router is organized around states.

States are bound to named, nested and parallel views allowing you to manage your application's interfaces with ease. Additionally, 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

AngularJS 2.0 will include capabilities from the ECMAScript 6 JavaScript specification, including an improved syntax for classes, modular loading system for code, and annotations for declaratively describing the purpose of a class.

### **Ecosystem**

Angular seems to be more widely adopted with a broader user base. The table below shows some interesting ecosystem statistics:

|  |  |  |
| --- | --- | --- |
|  | Knockout | Angular |
| GitHub Stars | 5,587 | 30,567 |
| GitHub Forks | 925 | 11,617 |
| Stack Overflow Questions | 12,126 | 60,119 |

### **Other Features.**

It would be fair to say that Knockout can only be compared to a subset of Angular features. Angular is a more extensive framework, and includes the following additional features:

· Modules - a module is a container for a set of components. Those components can be controllers, services, filters, directives and etc. From .NET or Java perspective, a module is more like a namespace. It enables 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 a number of built in services including:

o $http - used to make AJAX requests to the remote servers

o $q - promise/deferred implementation inspired by Kris Koala’s Q

o $log - service for logging. By default, writes to browser console if present

· Dependency Injection - the Angular injector subsystem is in charge of creating components, resolving their dependencies, and providing them to other components as requested.

· Scopes - arranged in hierarchical structure which mimic 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 own filter. Filters can be applied to expressions in view templates using the following syntax:

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

· Internationalization and Localization

· When working with Knockout, this functionality is not available in the core library, but can be added by using external libraries or custom logic.

### **Conclusion**

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

Angular’s ability to bind directly to plain objects, modular structure, and strict development guidelines prevent many issues right from the start, and provide a strong architectural foundation for the application.

Think of it this way: Knockout is primarily used to control UI representation in lower complexity applications, whereas Angular is a JavaScript framework that is much better suited for large, complex enterprise applications. It provides not only UI binding, but also best practices for application structure, development and testing.

## **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 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, 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.

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

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 actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn’t hold, a NB classifier still often does a great job in practice. 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 movies with Brad Pitt and Tom Cruise, you hate movies where they’re together).

## **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 (formerly atom-shell). Deciding which one to go with is not so obvious. That is precisely why I made a small comparison chart to show why I choose 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 used 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** | [Chromium](https://www.chromium.org/Home) | [libchromiumcontent](https://github.com/atom/libchromiumcontent) |
| **Layout Engine** | Blink / Webkit 537 | Webkit 537 |
| **JavaScript Engine** | [V8](https://code.google.com/p/v8/) | |
| **Node.js Engine** | [io.js](https://iojs.org/) v1.2.0 | [io.js](https://iojs.org/) 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 |
| **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](https://github.com/edjafarov/node-webkit-updater)) | Mac/Win (thru Squirrel) |
| **Kiosk Mode** | Partial (Buggy on Mac6) | |
| **Windows Installer** | Through [nw-builder](https://github.com/nwjs/nw-builder) | Yes ([external module](https://github.com/atom/grunt-electron-installer)) |
| **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](https://github.com/nwjs/nw.js/wiki/Using-MP3-&-MP4-(H.264)-using-the--video--&--audio--tags.)) | Yes |

#### 

### **Conclusion**

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

## **Competition in the Market**

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

### **Spy Agent:**

Spytech SpyAgent is our [award winning](http://www.spytech-web.com/spyagent-awards.shtml), 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 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](https://en.wikipedia.org/wiki/Content-control_software) marketed primarily towards [parents](https://en.wikipedia.org/wiki/Parent) as a way to control a child's computer activity. The flagship product allows a computer owner to block and filter [Internet](https://en.wikipedia.org/wiki/Internet) content, place time limits on use, and block desktop PC games

### **Qustodio:**

is a brand of [content-control software](https://en.wikipedia.org/wiki/Content-control_software) marketed primarily towards [parents](https://en.wikipedia.org/wiki/Parent) as a way to control a child's computer activity.

### **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.

## **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. Firebases 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.

### 

### **Concussions**

If your going to be linking to something such as a web or mobile application where the data is constantly changing by multiple users (all accessing the same database stored in the cloud) then Firebase is the way to go.

**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.

# **ANALYSIS AND DESIGN**

## **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.

## **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. |
| **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. |

|  |  |
| --- | --- |
| **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. |
| **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** | **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** |  |

### 

### 

# **Design and Implementation**

## **Design**

### **Material design.**

**Material** provides context in design, the surface and edge of a "material" provides us visual cues. Let's compare this to real life. We understand the dimensions of a room, because we see walls. At the same time, the interior provides us an understanding of the context of the room. Your kitchen looks very different than your bathroom for example.

The same is applied in Material Design. The combination of style and content provides **context** 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 very similar story. Motion provides context in a design through the **flow** of an application, especially when it comes to the **continuity** of a product, a user has the feeling of being uninterrupted. There are no obstacles, such as inconsistency in design or a confusing navigation.

How exactly does motion work? Here's an example. There's a home feed that consists of a list of cards. When you tap a single card, the material of the card expands to become the full width and height of the screen instead of the dimensions of a single card.

### **Layout**

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

## 

## 

## 

## **Web Browser/ Chrome Extension**

### **Code**

Development of the frontend of the browser is broken up into several major tools and complements. These include the nw.js framework which allows to develop the browser, angular.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 classifier alogthirm, interact with the database layer and the apis for the several projects.

### **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 creates more time to do non repetitive tasks which increases 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

**|-assets**

**| |-css**

**| | |-less**

**| | | |-bootstrap**

**| | | | |-mixins**

**| | | |-material**

**| | | |-robin**

**| |-fonts**

**| |-img**

**| | |-banner**

**| | |-banners**

**| | |-icons**

**| |-js**

**| | |-app**

**| | |-lib**

**| |-view**

**|-tests**

**|-index.html**

**|-package.json**

**|-gulpfile.js**

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 Robin looks like this.

As you can, see the title for the programme 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 Chromes APIs. In the devDependencies, you can include other Npm packages to the bundle when developing the programme 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 programme itself along with turning the less files to CSS, minifying the JavaScript, moving files around.

To use project, you run the following

1. Npm install (this installs all the node packages we are dependent on)

2. Then we run gulp build.

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.

### **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 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. And it all happens within the browser, making it an ideal partner with any server technology.

To implete Angular.js , I first and to added 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.

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.

The use of detecting when the web browser tab has changed the following angular directive was built to use

**app.directive('iframeOnload', [function() {**

**return {**

**scope: {**

**callBack: '&iframeOnload'**

**},**

**link: function(scope, element, attrs) {**

**element.on('load', function() {**

**return scope.callBack();**

**})**

**}**

**}**

**}]);**

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

**Work horse +Syncing**

### **Problems**

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

## **Naive Bayes Classifier Algorithm**

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 we needed to develop an AI to make the filtering system work.

In Robin's AI for understanding what to block and what not to block, the Naive Bayes Classifier model algorithm to start. For people who aren't sure what the Naive Bayes Classifier model is​

​The Naive Bayesian classifier is based on Bayes’ theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large data sets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods

So 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 Robin case, what are positive words and what are negative words. Robin itself has a built in system where as you search, 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 acurate, as more people use it. By training the AI, it means to educate it 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 AI 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 Normal 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 Normal 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, we can predict which should be blocked and what shouldn't be blocked. This is done by looking at the scores and comparing the amount of profanity of other websites that were searched for using the similar score as our testing.

### **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 we used which helped a lot in improving accuracy.

Negation handling is quite important in sentiment analysis, A word/phrase followed by a "fucked" 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 will not be 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 will add a lot of spurious and noisy features and they need to be eliminated by doing some feature selection. This can be 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 the robin browser and chrome extension.

### **Datasets/Training**

The Training sets are taking from scraping the following the websites, removing dulpetes and training them against the google profainity api.

**Good Words:**

**Negative Words:**

Problems

One of the problems I in counted 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.

## **Database / Firebase**

## 

## **Sequence diagrams**

\

## **Architecture diagrams**

## **Security**

The security in the application is within the user of tranfering data inside firebase, Firbase uses https only for its protocols. Within the set of firebase you can configure almost everything.

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.

{ ".read": true, // everyone can read ".write": "auth !=== null" // only authenticated users can write}

Security Rules are even more granular than this. The application has rules against a specific location in Firebase. In this case The application validatie 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 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)**

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.

# **Results & Testing**

There was two types of testing, one for the ui. The other was the for the code and mostly focused around the applications handling of parsing, syncing and rendering pages.

## **Syncing**

## **Database Bandwidth**

## **Performance**

## **Blocking/Algorithm results**

## **Survey Results**

# **Conclusion**

EVALUATION AND FURTHER WORK

First 3 in module discributor

4 questions do 2

1 a

1 b

1 c

Question + sections + all refference papers

1 {1,2} + Guidllines

2 {2} Humen physcialy

3 { 2} Humen physcialy

4 { 3 - 6

fUNC => EFFECT

uNTILTU

uSEABITILYT

Addordances = IMPORTANT FOR EXAMS

cONNITUVE LOAD THEORY