International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 04 | Apr-2018 www.irjet.net p-ISSN: 2395-0072

A Personalized Web Browser

D.VIGNESH KUMAR[1], D.KARTHIK KUMAR[2], S.RAJAPRIYAN[3], MS. J. SHANTHALAKSHMI REVATHY [4]

[123], *Computer Science and Engineering, Velammal College of Engineering and Technology, Madurai, Tamil Nadu, India*
[4]*Assistant Professor, Computer Science and Engineering, Velammal College of Engineering and Technology, Madurai, Tamil Nadu, India*

Abstract - *In a normal web browser the user used to browse in single full screen and switches over tabs for browsing to browse two different pages also normal browser are personalized based upon the search engines and regular browser are less interactive. Our proposed browser can be used to browse multiple sites in a single screen that is in split mode. This model supports more Interactiveness by supporting shortcut features and voice control, this is done without adding any hazardous extensions. Data analysis is made on data which is fetched from search engine of the browser to make more personalized browser for the users. User needs to create an account in browser server. While creating an account user need to give his unique id like mobile number for authentication, also account name and password for sign in. The data is sent to the server and data analysis will take place in server for the particular user.*

Key Words: Browser, Light weight browser, Personalization, Data analysis, Electronjs, Voice Control.

1. INTRODUCTION

The present browsers are less interactive where we need to use mouse to scroll up and down and to navigate through pages we need cursor. In some browser the interaction is enhanced using some extensions but extensions may contain malware and causes security issues. The browser also don't support multiple view user needs to switch tabs for viewing multiple web pages. Then if we turn towards data analysis, personalisation is done for only specific search engine. There is no any common personalization for a browser. For example if user has Google account the user data is analysed which is given in Google search engine. The size of browser is also too large and consumes more RAM of our system. Browsers like chrome consumes more RAM of our system when many tabs are opened also it occupies more space in our system. Normal browsers are cross platform but for different operating systems they need to modify some codes or structure of browser to support different operating system. Our personalized browser is designed in way to find remedy for all above mentioned issues. Our browser is made using web technologies so it is light weighted and design is made easier using HTML. Interaction is enhanced using voice and shortcuts inbuildly without adding any extensions.

Data analysis made common for all the search engines. It supports cross platform without modifying anything in code and structure using electronjs.

1.1 Objective

- To develop common analysis for all search engines.
- To enhance interaction without adding extensions.
- To develop light weight browser.
- To support multiple views of webpage in single tab.

2. ARCHITECTURE

There are a lot of web browsers available in the market. All of them interpret and display information on the screen however their capabilities and structure varies depending upon implementation. But the most basic component that all web browsers must exhibit is listed below:

- Controller/Dispatcher
- Interpreter
- Client Programs

Controller works as a control unit in CPU. It takes input from the keyboard or mouse, interpret it and make other services to work on the basis of input it receives.

Interpreter receives information from the controller and executes the instruction line by line. Some interpreters are mandatory while some are optional. For example, HTML interpreter program is mandatory and java interpreter is optional.

Client Program describes the specific protocol that will be used to access a particular service. Following are the client programs that are commonly used:

- HTTP
- SMTP
- FTP
- POP

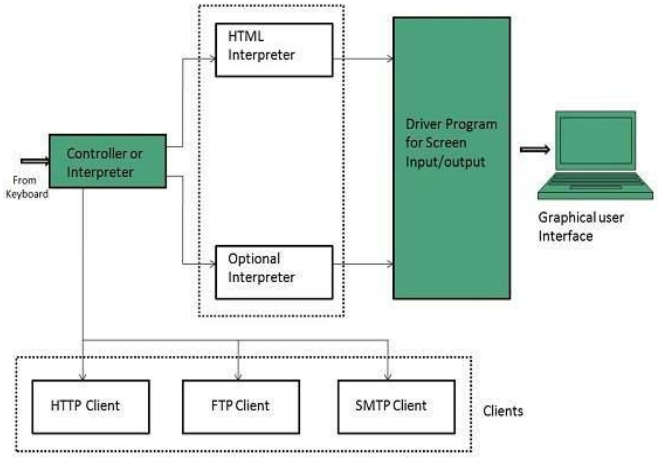

Fig -1: Browser Architecture

3. SYSTEM DESIGN

The first thing to do is step back and build a model for how browsers are used. User research and design analysis allows for complex models, which, depending on the kinds of questions you want answers to, may be entirely worthwhile. In this little model everything people do with browsers falls into three categories:

Reading. It's the ability to read information off the screen, through the browser, that is the largest percentage of human activity with browsers. Think about it: the reason for all of that searching, clicking and bookmaking is to find things worth reading (or perhaps skimming, sending to friends, or printing out). If you were to take a stopwatch and measure how much time most people spend doing various things with their browser, reading or skimming pages would be at the top of the list. There is a reason that the page itself is the largest part of the UI of any web browser. Scroll bars get more usage than most other browser features combined.

Navigating: Since reading doesn't require much interaction from the browser, most browsers build their structure around moving between pages. Hypertext systems are based on the ability to navigate, both through clicking links, and using other browser tools (Bookmarks, searches). A good browser makes it easy not only to move from one page or site to another, but also makes it easy to return to things the user has already seen (e.g. History) and things they have deemed as important or interesting (Bookmarks).

Interacting: Any time the user has to put information into the browser, URLs, passwords, credit card numbers, naked pictures of their neighbors, they are interacting with the browser or a website. Within the browser itself, this could be setting preferences, or adding/managing bookmarks. With websites it could be creating email, chatting with friends, or doing any of the other amazing things websites do.

4. IMPLEMENTATION

The development software used for the project is HTML, CSS, JAVASCRIPT and ELECTRON JS to convert them into execution files. For voice recognition implementation we are using Pocketsphinx, offline speech recognition engine.

HTML HyperText Markup Language (**HTML**) is a markup language for creating webpages. Webpages are usually viewed in a web browser. They can include writing, links, pictures, and even sound and video. HTML is used to mark and describe each of these kinds of content so the web browser can show them correctly.HTML can also be used to add Meta information to a web page. Meta information is information about the web page. For example, the name of the person who made it. Meta information is not usually shown by web browsers.

CSS Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

JavaScript JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content engineering. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMAScript specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA. Normal browser is done with event handling for forward and backward navigation of web page in the browser using java script. Now we know convert them into execution file. This can be done using Electronjs.

Electron Electron is an open-source framework created and maintained by GitHub. It allows for the development of desktop GUI applications using front and back end

components originally developed for web applications: Node.js runtime for the backend and Chromium for the frontend. Electron is the main GUI framework behind several notable open-source projects including GitHub's Atom and Microsoft's Visual Studio Code source code editors, the Tidal music streaming service desktop application and the Light Table IDE, in addition to the freeware desktop client for the Discord chat service.

NPM npm package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc. Using this npm we are installing the electron and other required runtime environments. A basic Electron app consists of three files: Package.json (metadata), main.js (code) and index.html (graphical user interface). The framework is provided by the Electron executable file (electron.exe on Windows, electron.app on MacOS, and electron on Linux). Developers wishing to add branding and custom icons can rename and/or edit the Electron executable file.

The most important file in the Electron file is Package.json. It keeps information about the package. The most common information in Package.json is:

- "name", the application name
- "version", the application version string
- "main", the name of the main script file of the application

Package.json is an npm file. Using this electron execution file can be created using electron by converting the HTML CSS files. iframes in HTML are used to split the screen for multiple web page in a single tab. To generalize the analysis common for all search engine user needs to create account in browser server. The data which is searched is passed to through server and server will analysis the data. The result is provided as insights about the data provided and filters the content according to the user's previous searches and event occurrence. To analysis data at server side hadoop is used.

Hadoop Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the Map Reduce programming model. Originally designed for computer clusters built from commodity hardware—still the common use—it has also found use on clusters of higher-end hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework. Using this data analysis is made on data sent to the server and provides insights about the user.

Interactiveness is enhanced using speech and shortcuts without adding any extensions that is both these features are done inbuilt.

PocketSphinx For Voice Recognition we are using PocketSphinx .It is a offline voice recognition. PocketSphinx is one of Carnegie Mellon University's open source large vocabulary, speaker-independent continuous speech recognition engine. This is a Research System.

The words are identified based on the pronunciation for a word or a phrase from "The CMU Pronouncing Dictionary" The Carnegie Mellon University Pronouncing Dictionary is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations. CMUdict is being actively maintained and expanded. We are open to suggestions, corrections and other input. Its entries are particularly useful for speech recognition and synthesis, as it has mappings from words to their pronunciations in the ARPAbet phoneme set, a standard for English pronunciation. The current phoneme set contains 39 phonemes, vowels carry a lexical stress marker:

0 — No stress
1 — Primary stress
2 — Secondary stress

The Pocketsphinx API is designed to ease the use of speech recognizer functionality in your applications:

- It is very likely to remain stable both in terms of source and binary compatibility, due to use of abstract types.
- It is fully re-entrant, so there is no problem having multiple decoders in the same process.
- It allows a drastic reduction in code footprint and a modest but significant reduction in memory consumption.

There are multiple possible search modes:

- Keyword: efficiently looks for a key phrase and ignores other speech. It allows configuring the detection threshold.
- Grammar: recognizes speech according to the JSGF grammar. Unlike key phrase search, grammar search doesn't ignore words which are not in the grammar but tries to recognize them.
- ngram/lm: recognizes natural speech with a language model.
- allphone: recognizes phonemes with a phonetic language model.

5. EXPERIMENTAL RESULTS

We conducted experiments on time taken to load a webpage in our electron web browser in comparison with other web browsers like Google Chrome, Mozilla Firefox, Internet Explorer, Microsoft Edge, Safari. The results showed faster retrieval of a web page for our electron web browser.

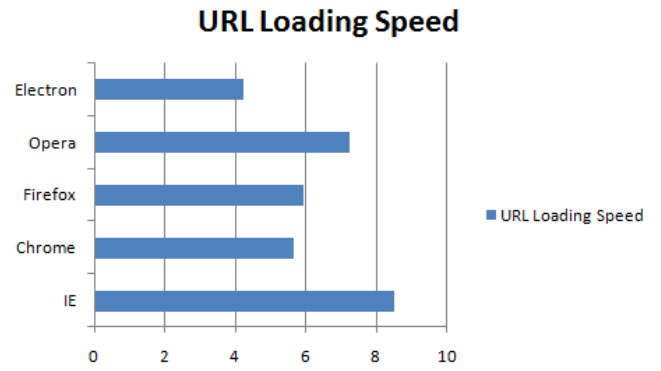

Fig -2: Experimental Results

6. CONCLUSION

This browser provides more generalized personalization with richer interaction and multiple views. All these features were added without adding any extension which is more vulnerable in cybercrime. Also our browsers are fast light, weighted, cross platform and consumes less RAM compared to other browsers leading in the market.

Future scope Now days operating systems memory size are reducing. An operating system can be build under few mega bytes. Thus it increases compact ability of computer devices. To suite with this environment the browser should be light weighted cross platform that is supports all operating system without changing any structure of the browser.

Also Gestures can be included for interaction with browsers for physically aided peoples. Data analysis can be proceeded with machine learning and predicts what the user going to search before he types. These features will moves the browsing experience to next level of generation. The following are future scopes of the browser.

- Gesture controlled browser for physically aided peoples.
- Machine learning Browser for predicting the future searches of users.
- Fully Voice controlled web browser

REFERENCES

[1] A. Grosskurth, M. W. Godfrey, "A reference architecture for Web browsers", 21st IEEE International Conference on Software Maintenance (ICSM'05)

[2] X. Qiao, G. Nan, Y. Peng, L. Guo, J. Chen, Y. Sun, and J. Chen. Ndnbrowser: An extended web browser for named data networking. Journal of Network and Computer Applications, 50:134–147, 2015.

[3] Chris Grier, Shuo Tang, Samuel T. King,"Web Browsing with the OP Web Browser", 2008 IEEE Symposium on Security and Privacy (sp 2008)

[4] Tingshao Zhu, Xinguo Xu, Guohua Liu,"Identifying machine query for an intelligent web browser system", 2010 IEEE 2nd Symposium on Web Society

[5] Chris Grier, Shuo Tang, Samuel T. King, "Secure Web Browsing with the OP Web Browser" 2008 IEEE Symposium on Security and Privacy (sp 2008)

[6] Sean Pretorius, Adeyemi R. Ikuesan, Hein S. Venter, "Attributing users based on web browser history", 2017 IEEE Conference on Application, Information and Network Security (AINS)

[7] Ankit Kumar, Joy Bose, Divya Bansal, "A web browser responsive to the user interest level", 2015 Annual IEEE India Conference (INDICON)

[8] Joy Bose, Amit Singhai, Ankur Trisal, Vinod Keshav, Utkarsh Dubey, "A hands free browser using EEG and voice inputs", 2015 International Joint Conference on Neural Networks (IJCNN)