THE DESIGN AND EVALUATION OF NOVEL PROTOTYPES

TO VISUALIZE WEB BROWSING HISTORY

By

Anudeep Makkena

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The School of Graduate Studies

Laurentian university

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE

Laurentian Université/Université Laurentienne
School of Graduate Studies/École des études supérieures

Title of Thesis Titre de la thèse	THE DESIGN AND EVALUATION OF NOVEL PROTOTYPES TO VISUALIZE WEB BROWSING HISTORY
Name of Candidate Nom du candidat	Makkena, Anudeep
Degree Diplôme	Master of Science

Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance	February 28, 2014

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Ratvinder Grewal
(Supervisor/Directeur de thèse)

Dr. Kalpdrum Passi
(Committee member/Membre du comité)

Dr. Julia Johnson
(Committee member/Membre du comité)

Dr. Prabhat K. Mahanti
(External Examiner/Examinateur externe)

Approved for the School of Graduate Studies
Approuvé pour l'École des études supérieures
Dr. David Lesbarrères
M. David Lesbarrères
Director, School of Graduate Studies
Directeur, École des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

Mainstream Web browsers support users in revisiting Web Pages by providing them with a history tool. Research shows that this history tool is severely underutilized. One possible reason is the manner in which the pages are displayed: a linear list of textual links. This thesis investigates the redesign of the history tool by introducing visualization to display the visited Web pages. Three distinct visual prototypes were designed ranging from a traditional scientific visualization method to a concrete visualization that incorporates a metaphor and knowledge transfer from the real-world. The low-fidelity prototypes were evaluated by participants and the best performing design was implemented as a high-fidelity prototype. Further evaluation with participants was conducted and the results were compared against the performance of participants using the traditional history tool of linear textual links.

Dedicated to my mom and dad…….

Acknowledgements

I would like to thank my supervisor Dr. Ratvinder S. Grewal for giving me immense support and guidance throughout my master's program.

Dr. Kalpdrum Passi and Dr. Julia Johnson being my committee members, providing their valuable feedback.

Faculty members who taught me in my master's program, participants in my experiments and the administrative staff.

Callum for proof reading and helping in statistics.

My dear friends Sarita Lakra and Jay, for making my road smooth in the master's program…

Table of Contents

Thesis Defence Committee…… (ii)

Abstract…… (III)

Acknowledgements…… (V)

Table of Contents…… (VI)

List of Figures…… (VIII)

List of Tables…… (X)

List of Appendices…… (XI)

1. Introduction…… 1
   - 1.1 Web browsers…… 1
     - 1.1.1 History tools in various browsers…… 2
   - 1.2 Problems addressed in the browsing history…… 2
   - 1.3 Motivation for thesis work…… 3
   - 1.4 Purpose of study…… 3
   - 1.5 Organization of thesis…… 4
2. Related Work…… 5
   - 2.1 Related work in 1995-2011…… 5
     - 2.1.1 Mosaic graphic history…… 5
     - 2.1.2 Pad prints…… 6
     - 2.1.3 Domain tree browser…… 7
     - 2.1.4 Web view…… 8
     - 2.1.5 Browsing icons…… 9
     - 2.1.6 Hyper history…… 10
     - 2.1.7 Pivot bar…… 12
     - 2.1.8 History tree…… 13
     - 2.1.9 Visual bookmark system…… 14
     - 2.1.10 I-Pot…… 15
     - 2.1.11 Eye browse…… 15
     - 2.1.12 Web browsing history grid…… 16
   - 2.2 Relevant work…… 17
     - 2.2.1 Visual and non-visual display…… 18
     - 2.2.2 Metaphors…… 18

2.3 Comparing the existing tools with our approach……. 19
3. Design……20
3.1 Back end work……20
3.1.1 Understanding the places. SQlite database……20
3.1.1.1 Moz_places table……20
3.1.1.2 Moz_history visits…… 21
3.1.1.3 Moz_favicons table…… 22
3.1.1.4 Session store. js…… 23
3.2 Front end work…… 24
3.2.1 Information visualization…… 24
3.3 Paper prototypes…… 26
3.3.1 Design 1…… 27
3.3.2 Design 2…… 28
3.3.3 Design 3…… 30
4. Design Selection…… 33
4.1 Research methods…… 33
4.2 Participants…… 34
4.3 Pilot study…… 34
4.4 Procedure…… 34
4.5 Results…… 35
4.6 Discussion…… 39
5. Prototype implementation…… 40
5.1 Designing process…… 40
5.2 Timeline features…… 42
5.2.1 Days of a week…… 42
5.2.2 Hourly format…… 43
5.2.3 List of visited Webpages…… 43
5.3 Implementation of the final design…… 45
5.3.1 Research methods…… 45
5.3.2 Participants…… 45
5.3.3 Pilot study…… 45
5.3.4 Procedure…… 46
5.3.5 Results…… 46
5.4 Discussion…… 49
6. Conclusions and Future Work…… 51
6.1 Conclusions…… 51
6.2 Future research…… 52
7. References……53

List of Figures

Figure 1.1 An example of Parallel browsing session visualization…… 4
Figure 2.2 Overview of the graphic history view…….. 6
Figure 2.2 The Pad Prints browser Companion…….. 7
Figure 2.3 Screen shot of domain tree browser…… 8
Figure 2.4 Web view's two display organization…… 9
Figure 2.5 Browser icons user interface…… 10
Figure 2.6 Hyper history's episode and search view…… 11
Figure 2.7 Hyper history's auto bookmarks and compact view…… 12
Figure 2.8 Pivot bar recommendations…….. 12
Figure 2.9 History tree: Top to bottom view…….. 13
Figure 2.10 History tree: Grid view…… 13
Figure 2.11 3D Layout modes…… 14
Figure 2.12 Organic graphic representations of the I-Pot…….. 15
Figure 2.13 Eye browse…….. 16
Figure 2.14 web browsing history grid…… 17
Figure 3.1 Relationship schemas between the tables in places .SQlite…… 23
Figure 3.2 days of a week in the timeline prototype…… 27
Figure 3.3 Browsing days shown in 24-hour format…….. 27
Figure 3.4 Tabs and windows information for a browsing session…….. 28
Figure 3.5 Overview of timeline prototype…… 28
Figure 3.6 Days of week in the clock prototype…….. 29
Figure 3.7 Twelve hour period intervals for a browsing day…… 29
Figure 3.8 Browsing sessions details including tabs formation……30
Figure 3.9 Days of week in the table prototype ……..31
Figure 3.10 Browsing sessions for a day……31

Figure 3.11 Browsing sessions details for the table prototype…….…….. 31
Figure 3.12 Overview of the table prototype…….…….. 32
Figure 4.1 Five point Likert scale…….…… 34
Figure 4.2 Mean times for the time completion result charts…….. 35
Figure 4.3 Mean value for the accuracy result chart…….…… 36
Figure 4.4 Mean values for the Likert result chart…….…….. 37
Figure 5.1 Back-end architecture…….…….. 41
Figure 5.2 General screen of the timeline prototype…….. 42
Figure 5.3 Days of week in the timeline prototype……43
Figure 5.4 Hourly-wise browsing details……43
Figure 5.5 Favicon along with the title of the webpage……..43
Figure5.6 Duration of a web page and zooming…….…… 44
Figure 5.7 History information available…….…… 44
Figure 5.8 Summary of mean time in the completion chart……47
Figure 5.9 Mean values chart for the accuracy…….. 48
Figure 5.8 Mean value chart for the Likert scale……48

List of Tables

Table 3.1 Seven different types…… 22

Table 4.1 One-way ANOVA statistics table for time completion……..36

Table 4.2 One-way ANOVA statistics table of accuracy…… 37

Table 4.3 One-way ANOVA statistics table for Likert scale…… 38

Table 4.4 Example of unstructured interview questions…….. 39

List of Appendices

Appendix A: Approved Ethics Certificate……..58

Appendix B: Tasks for Design Selection.....65

Appendix C: T-test results for the Timeline design .....……69

Appendix D: Connecting to database using JDBC driver.....74

Appendix E: Java code used for getting all the sessions from session manager.....75

Appendix F: Code for generating Timeline ……..99

Chapter 1

Introduction

The use of World Wide Web, also known as WWW, has been increasing rapidly in the past few years [1]. The term WWW refers to a set of Internet protocols and software which is used to present information in a format called hypertext to the users [2]. According to Group 2011 [3], it is estimated that 2.2 billion users are using the internet, which is almost 30% of the total population. A recent survey estimates that on average individual in the U.S spends 13 hours per week on the Web, which is almost equal to the amount of time spent watching television [4]. People use the Web for daily activities, banking transactions, social media, and entertainment. In 1993, Marc Andreessen and his team created a program called mosaic which interprets the content of hypertext documents and sends the information to a monitor, which can be viewed by users. This program, later described as the world's first Web browser, became popular and opened the gates of Web for the general public. This program was considered as an open source product which was distributed free of cost [2].

1.1 Web Browsers

According to Dan's Web Tips [5], it is estimated that there are more than 100 Web browsers present all around the world. Some of them are used for specific purposes such as "Albert", which is used as a full screen browser for VM/CMS systems with IBM 3270 terminals. Other browsers use "Hot Java", a Web browser which is written in Java. Yet another example is a "Kiosk browser", which is developed mainly to use in kiosks and it runs under the KDE Unix graphical environment. Though we have many browsers, some of the popular ones are Internet Explorer, Firefox, Chrome, Safari and Opera. The main browser paradigm consisted of a single tab where the user has to visit a different sequence of pages in the same browser window [6]. However, the current browsers provide a different tabs provision, which supports parallel browsing. Due to this tabbing interface, the user can open multiple Web pages in different tabs in the same browser window. During the initial stages of introducing tabs, the tabs facility was supported through different extensions, though this method was later adapted by the browsers themselves. The user can switch between Web pages using multiple windows through a main taskbar of the operating system. Through the multiple tabs, the user can switch between Web pages using the list of tabs that are present inside the browser [6]. However, *"the differences between tabbing and multi-windowing are important from the perspective of the human-computer interaction and user interfaces, in the Web usage mining, we only need to know which resources (pages) are being browsed and how the user travels "[7]*. Current browsers support navigation features in the form of back, forward, Bookmarks, history or by entering URL (Uniform resource locator) in the address bar. The back or forward navigating method returning for recently visited pages, Bookmarks and history are for returning to the pages that have been visited in the past. Most modern browsers store the information about visited pages in the

history. Every browser has its own way of storing the history. History consists of linear textual links which are either sorted by chronological order, visit date or by the frequency of visits. Users can delete the history if they are not willing to let anyone know about their visited Webpages. However, history will not be generated if the browsing is done in a private mode. By browsing in a private mode, the pages will not be recorded in the history. According to Browser Statistics in December 2011 [8], the popular Web browsers usages are as follows:
Firefox (37.7%), Chrome (34.6%), Internet Explorer (20.2%), Safari (4.2%) and Opera (2.5%).

1.1.1 History tools in various browsers

In Internet Explorer, the history tool is located under the tools option. The user can access the visited Web pages in the history by using any of the five options.

- View by date,
- View by site,
- View by most visited,
- View by order visited today,
- View by search option: Using search option the user can use keywords to find the Web page.

In Mozilla Firefox, the user can see the list of visited Web pages in a linear textual links. This includes the title along with a thumbnail image and URL (Uniform resource locator). The user can see the Web page dates and the frequency of visits by activating those options in the browser. Using search option the user can type URL or a title to find the Web page.

In Google Chrome, the history tool consists of a list of visited Web pages, which are aggregated by day, week, or month. Each entry includes a timestamp, a thumbnail image, and a URL associated with a Webpage. There is a search option associated with the browser for finding the Webpage by typing the URL or a title.

In Safari, the browsing tool includes the bookmarks menu and the top site pages [9]. Top site pages include thumbnail images of the pages in a 3D style. Safari provides a drag and move option to rearrange the thumbnail images on the top site pages.

In Opera browser, the history tool lists the visited Web pages, which are aggregated by the day. Each entry includes a timestamp, a thumbnail image, the title, and the URL associated with a Web page. The history tool in this browser looks similar to the google chrome in displaying the Web pages. By using clearing browsing data option the user can delete the entire history from the browser.

1.2 Problems addressed in the browsing history

From the user's perspective, there are two general ways of browsing [10].

1. Search – Finding a Website the user has never visited before.
2. Revisitation – Returning to a Webpage the user has visited in the past.

H. Obendrof et.al [11] differentiated the revisiting of Web pages into four types based on time.

1. Short-term (72.6% of all revisits takes place within an hour),
2. Medium term (12% of revisits occur within a day, while 7.8% of revisits occur within a week),
3. Long term (7.6% of revisits occur after a week).

According to the 6th GVU survey, 13.41% subjects had reported that they could not find pages which were visited recently [12]. Cockburn and Mackenzie in 2000 found that 80% of the Web pages are previously visited [1]. The same survey found that 42% of the pages are visited using the back button, and 0.1% are accessed through the history list. This shows that the history list is barely used. Current browsers have some problems in representing the history of a browser.

1. The browser history does not provide information about the amount of time spent on a particular Web page.
2. They do not show the relationship between the existing Web pages whether the current Web page was visited using the previous Webpage in the list.
3. There is no indication about whether the Web page was reached through a search engine or by entering a URL.
4. The tabs and windows information is not mentioned in the history of a browser.

1.3 Motivation for Thesis work

The previous research findings show that there is scope for improvement in the display and navigation of Web browser history [1, 12]. Current browsers contain a linear list of textual links in the history. After several studies, I came up with a concept of Breadcrumb navigation, which is a secondary navigation aid that is used to locate Web pages within the Web site. The term breadcrumbs came from Grimm's fairy tale. It shows the path taken by user from the start of a page to the current location within the Website. This concept can be used to show the path taken between Web pages in the history of a browser. The current browser does not show any relationship between the existing Web pages. Using visualization methods, the path can be shown along with the tabs and windows information. To accomplish this method, a good visualization tool needs to be constructed by reviewing the existing visualization tools.

1.4 Purpose of study

The aim of this master's thesis is to build a hierarchical relationship between the Web pages by considering the today's browsing habits. The browsing habits of the user have changed a lot from mid–nineties until now i.e. usage of different tabs and using different windows which

leads to parallel browsing patterns. A recent paper [6], suggests that users switch tabs at least 57.4% of the time in a browser, which makes use of tab-based parallel navigation. A survey conducted with 236 experienced Web users by Aula et al. [13] found that the multiple tabs or windows are often used in parallel while searching the Web pages.

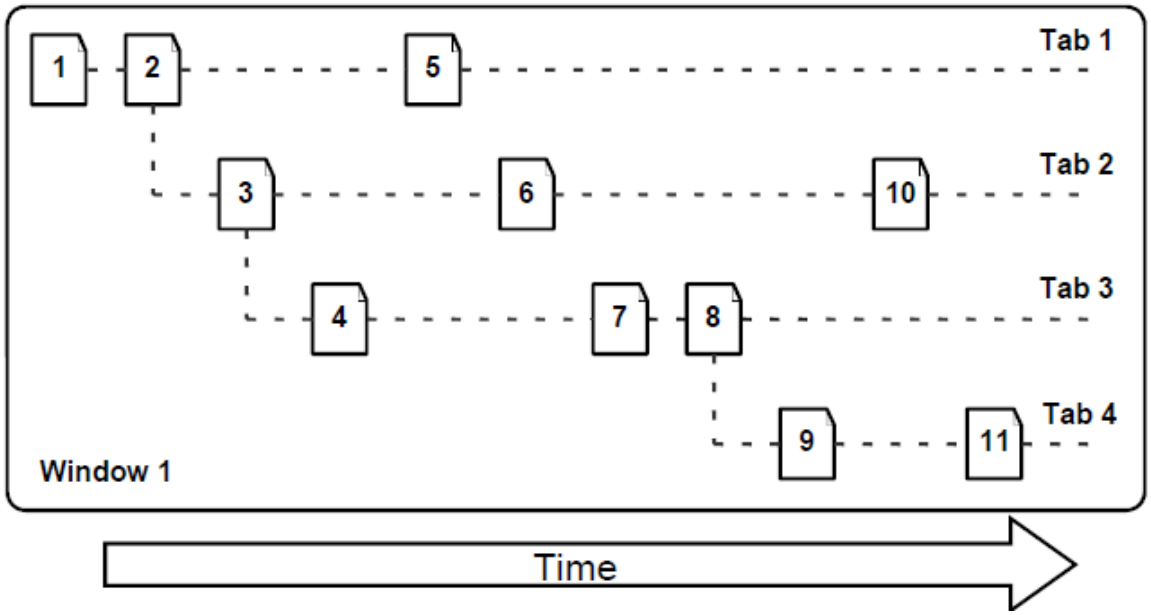


Figure 1.1 An example of parallel browsing session visualization [5]

1.5 Organization of Thesis

The remainder of the thesis is structured with as follows: Chapter 2 presents the related work on various visualization tools which are used in the past for visualizing the history of a browser. Chapter 3 gives an overview about the method of designing a prototype, which consists of back end as well as front end design. Chapter 4 gives details of an experiment conducted to choose an optimal prototype among a variety of designs. In Chapter 5, the preferred prototype was implemented and another experiment was conducted. There is a questionnaire that was given to compare the selected prototype with the existing browser history. The results are also discussed in the same chapter. In chapter 6, we wrote the conclusions and future research work that can be continued. In Appendix section, we provided the code that was used for designing the prototype as well as the questionnaire that was used for the experiments.

Chapter 2

Related Work

Studies have shown that people use the history of a browser to find the revisited Web pages [1]. Current browser lacks graphical visualization techniques for displaying the Web pages in the history. The use of visual aids in the history mechanism is more effective than displaying the Web pages in a textual manner [14]. There is therefore a need to develop the history mechanism by using different visualization techniques, as most of the users use this option to revisit Web pages [15]. In the past many researchers tried to visualize the history of a browser in which some of them are presented in this chapter. This chapter includes the comparison of our approach with the past research work. The snap shot of the previous visualization tools was taken directly from their original work.

2.1 Related work between 1995-2011

2.1.1 Mosaic graphic history

In 1995, Ayers and Stasko [16] created a graphical history view add on and incorporated it into a mosaic browser. This view displays a tree which consists of a single browsing session. A node within the tree consists of a thumbnail and title of the Web page. When a new page is visited it is added as a child node to the previous page. Opening and closing a browser consists of a single session. The relationships between pages are indicated using an arrow, and a special arrow indicates that a Web page is the destination of more than one hypertext link. If a Website is already present in the tree then it does not add a new node instead it highlights the most visited Web page. By using a title shortening algorithm, the graphic history view can shorten the title of the documents as the tree grows to the right side of the screen. A short arrow appears in the left side of the node to show that the Website was the destination to more than one link. The nodes that are connected to this link are highlighted by positioning the cursor on this arrow. The user can save the session in a text file by using the save command. The authors believed that by adding visual cues, the graphic history view will help the users in finding the revisited Web pages easily.

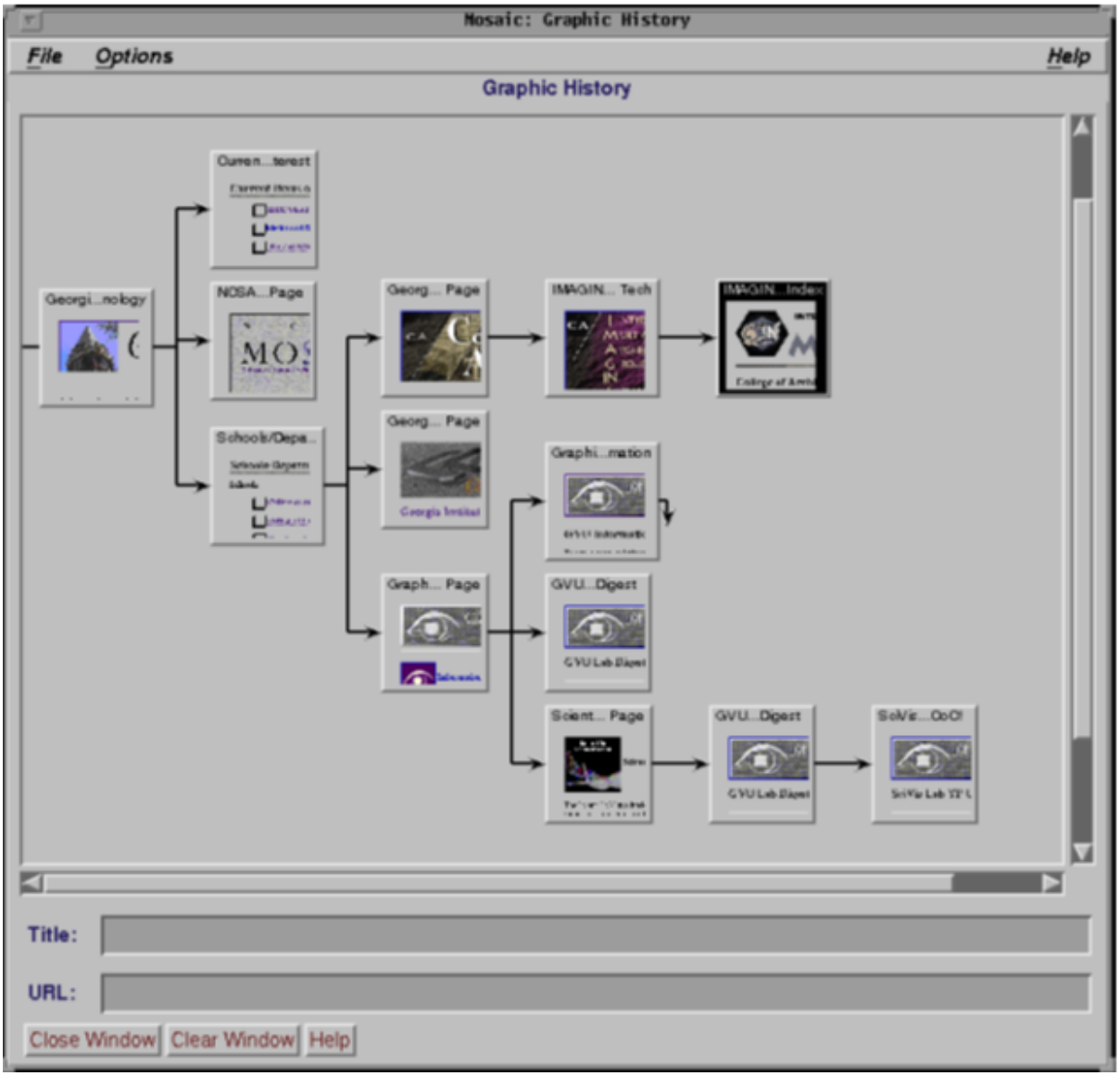


Figure 2.1 Overview of the graphic history view [16]

2.1.2 Pad prints

Pad prints [17], is a tool which visualizes the tree of a visited page in the form of a single history tree. It also depicts a left to right hierarchical structure of visited Web pages, like the graphical history view. Pad prints uses pad++, a Zooming user interface (ZUI) substrate which allows the user to focus on a particular part of the tree. The browsing activity is logged using a proxy server and pad prints do not require html modifications, as a browser always changes due to the addition of new html features. When a user accesses pages from the browser, those pages will be added in the pad prints display. The pages are added as the child nodes to the parent node, unless that page was already present in the hierarchy. In this case, it marks the existing page with a yellow outline to denote it as the current page. The result of a usability test

suggests that, the number of Web pages accessed using pad prints were less when compared to Netscape Navigator 3.0 history mechanism. In addition, the time taken to complete the task with pad prints which involved revisiting a Web page was very less when compared to the Netscape.

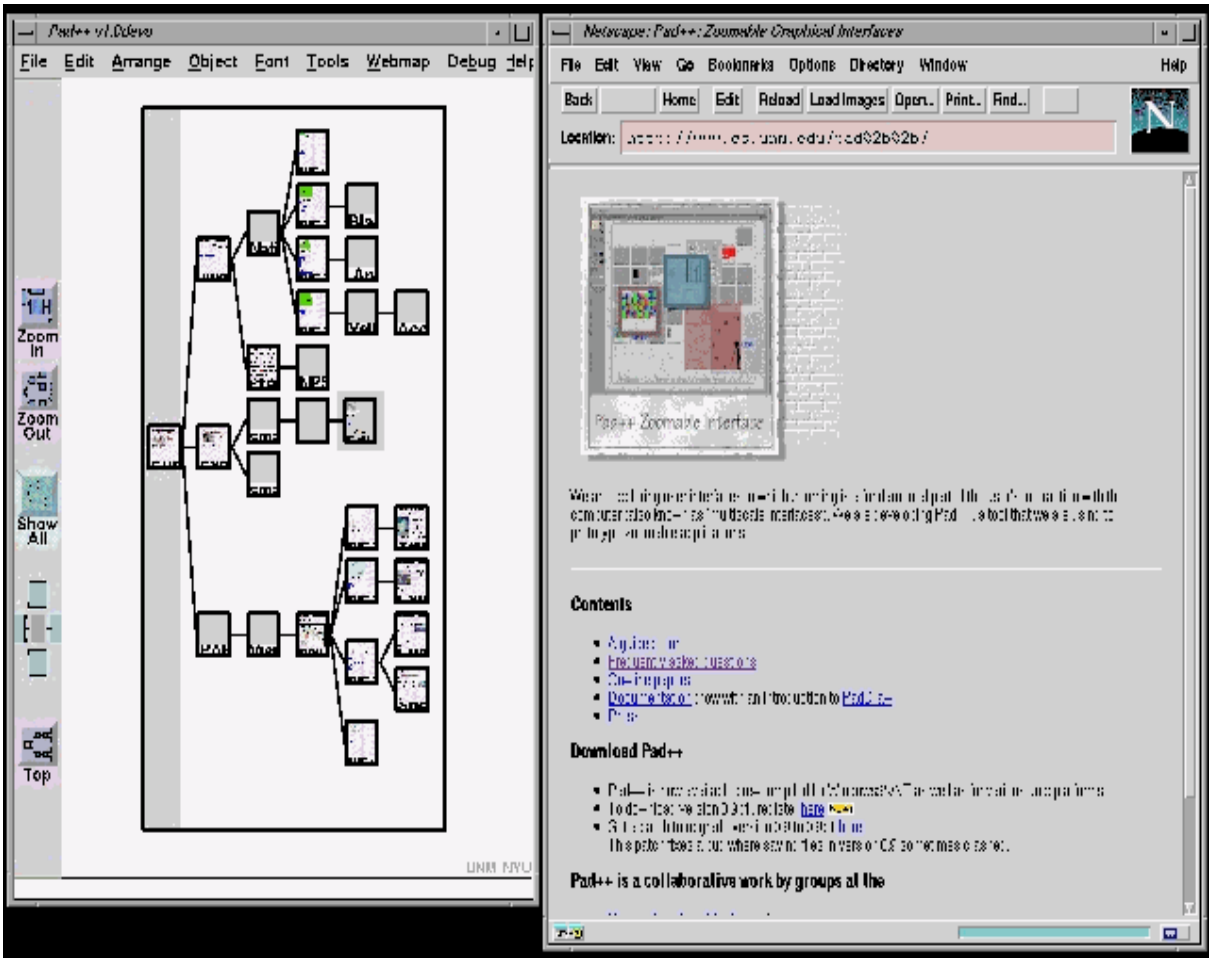


Figure 2.2 The pad prints browser companion [17]

2.1.3 Domain tree browser

Domain tree browser (DTB) [18], a Web history visualization tool which builds a tree structured visual history and contains one tree per server-level domain. DTB was implemented using Java Swing package and Jazz was used for zoomable user interface toolkit. This tool consists of two panels which are known as a domain panel and a tree panel. It creates a node in the tree for every visited page and puts a thumbnail image of the Web page on it. If the URL is already present in any domain it marks the existing page with a green color. The node of a tree increases with the number of visits. Every domain consists of a single tree where the user can

prune the tree if it is not required. Usability testing was done with four participants and results showed that DTB helped in reducing the amount of time for revisiting pages .The problem with this tool is that it doesn't show the relationship existing in two domains. However, user satisfaction was high using with this tool when compared to existing browser history.

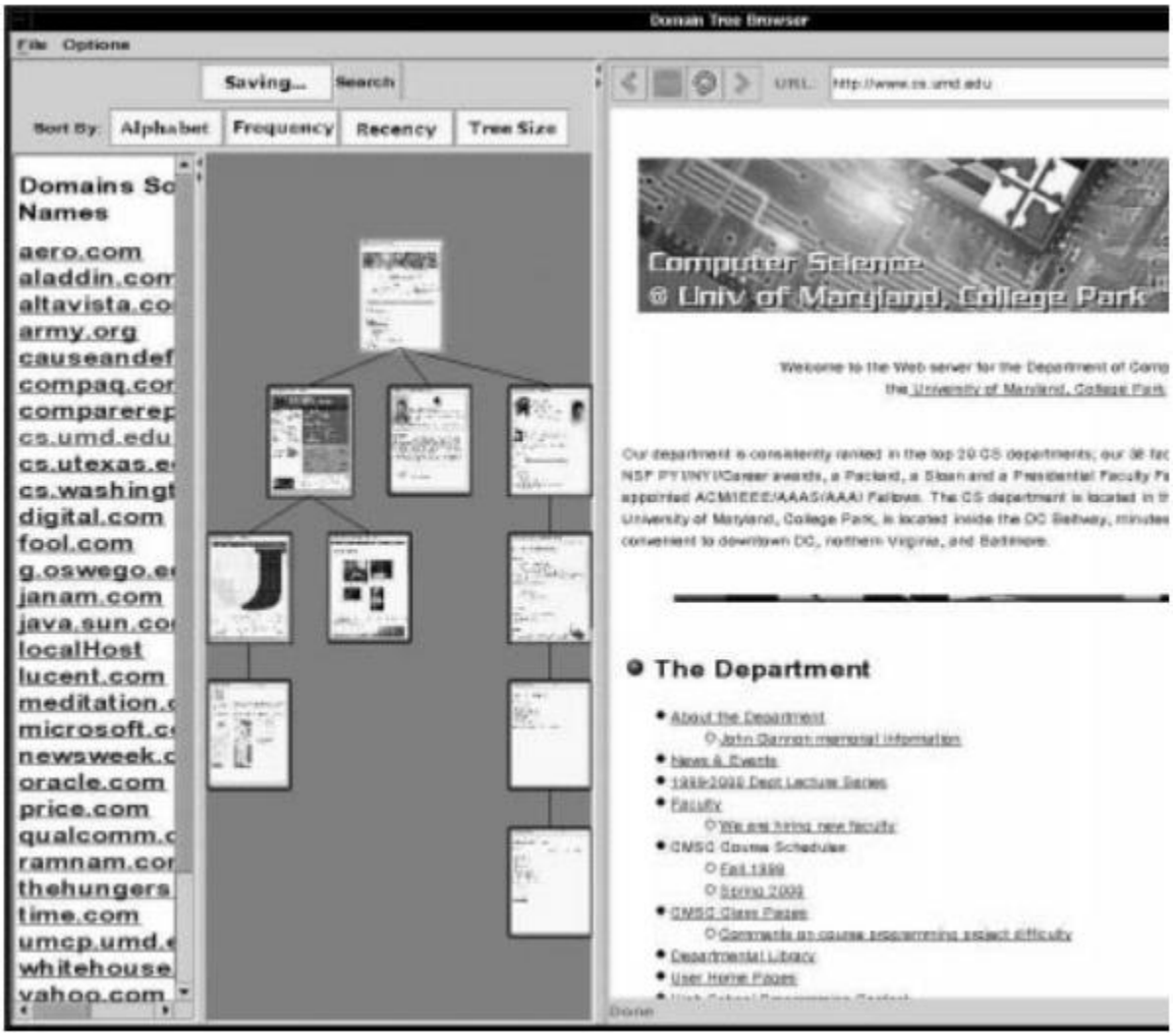


Figure 2.3 Screen shot of Domain tree browser [18]

2.1.4 Web view

Web view [19] is a tool which displays the graphical overview of the user's browsing path. It supports two types of organization display: one is hub and spoke view, and the other is temporal display. Hub and spoke view displays the tree like a nesting relationship between the location of Web pages and the temporal displays recently visited pages in an ordered list. It has a dog-ear which turns into a green color by the increasing number of visits to the same page.

Preliminary evaluation was done with seven subjects and it indicated that Web view provided a statistically significant improvement in terms of efficiency. This tool was designed to improve the efficiency as well as the usability while revisiting the Web page.

Hub and Spoke View **Temporal View**

Figure 2.4 Web view's two display organizations [19]

2.1.5 Browsing icons

Browsing icons [20] is a visual Web history tool which draws animated graphs of the user's path through the Web unlike pad prints, graphical History view and domain tree browser [DTB]. By using a proxy server, it can be attached to any type of a browser. Browsing icons has two methods of visualizing the Web pages: task and session-based. Task view provides small versions of the graph of all sessions that belong to this task. Session view shows an interactive graph of the current Web session. Opening the browser initiates a new session and it is visually displayed as a graph in the session view. When user access pages from the browser, they are added as nodes to the graph in the session view. If the Web page is already present in the graph then the node grows with the number of visits. Spring-based layout algorithm was used for

drawing this graph. Hovering over a node shows thumbnail and further information in a detailed view. The user can reform the shape of the graph and manually change the name of nodes. The results of a usability test suggest that, number of pages accessed and the time taken to complete the tasks were less with the Browsing icons when compared to the Netscape.

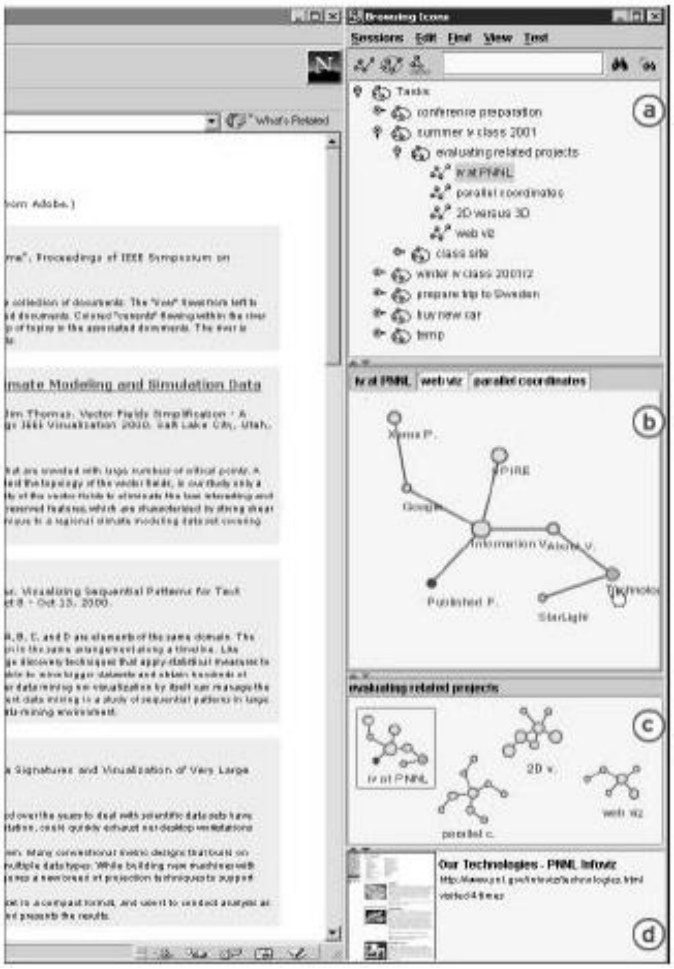


Figure 2.5 Browsing icons user interface [20]

2.1.6 Hyper history

Hyper history [21] is a browser extension which visualizes the user's browsing habits. It has auto bookmarking, the date of last visit, visit counts, visit frequency duration, search and live history. If the URL is already present, then hyper history does not add a duplicate of the resource of the tree, but rather updates the visit date. The tool has an episode view which forms a tree like structure and the recent history which shows the recent pages visited. The personal roadmap displays the most revisited Web pages. The reasons for forming this tool are

to address the browser history's shortcomings and present useful ways to improve the hypertext navigation.

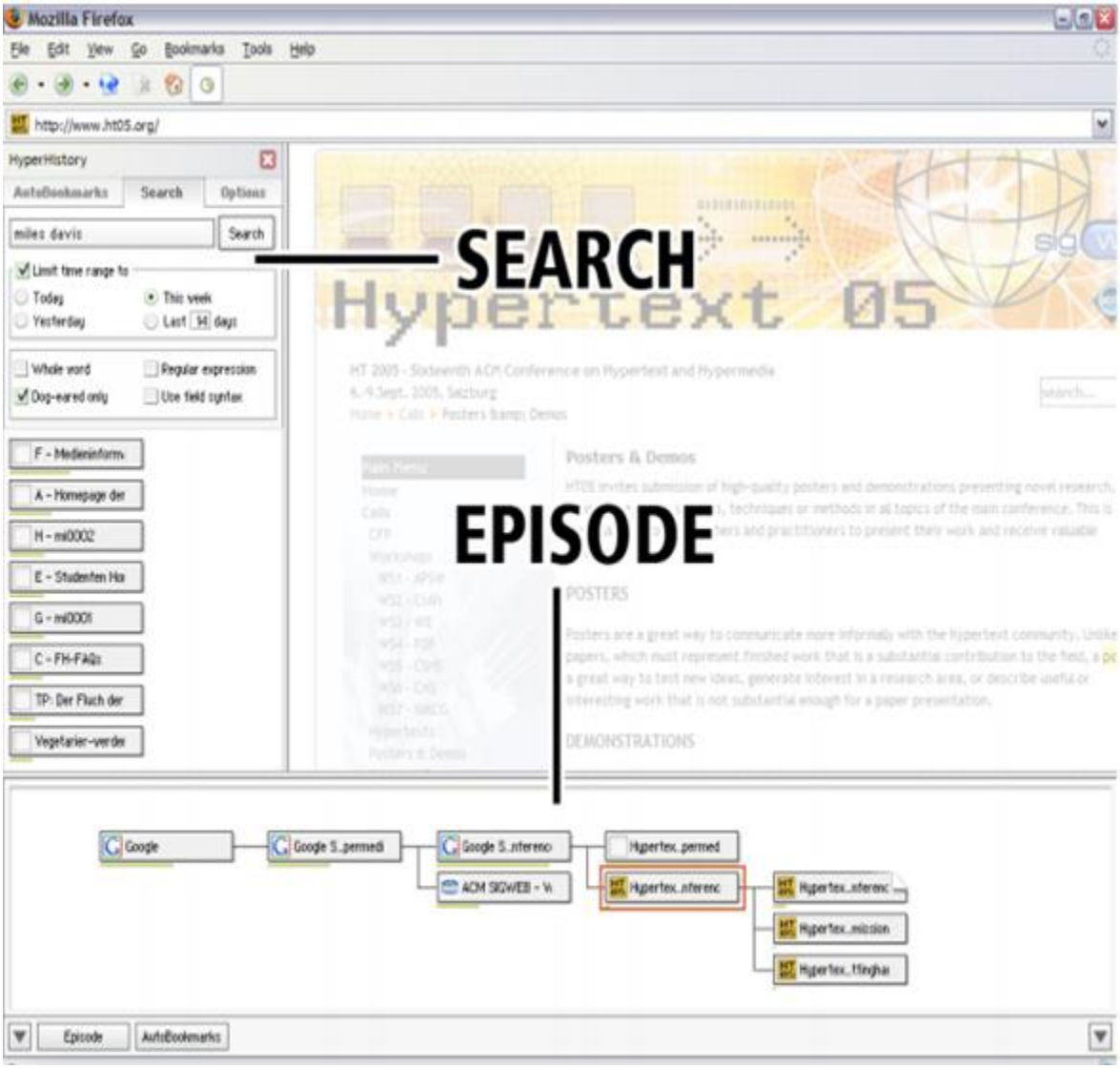


Figure 2.6 Hyper history's Episode and Search view [21]

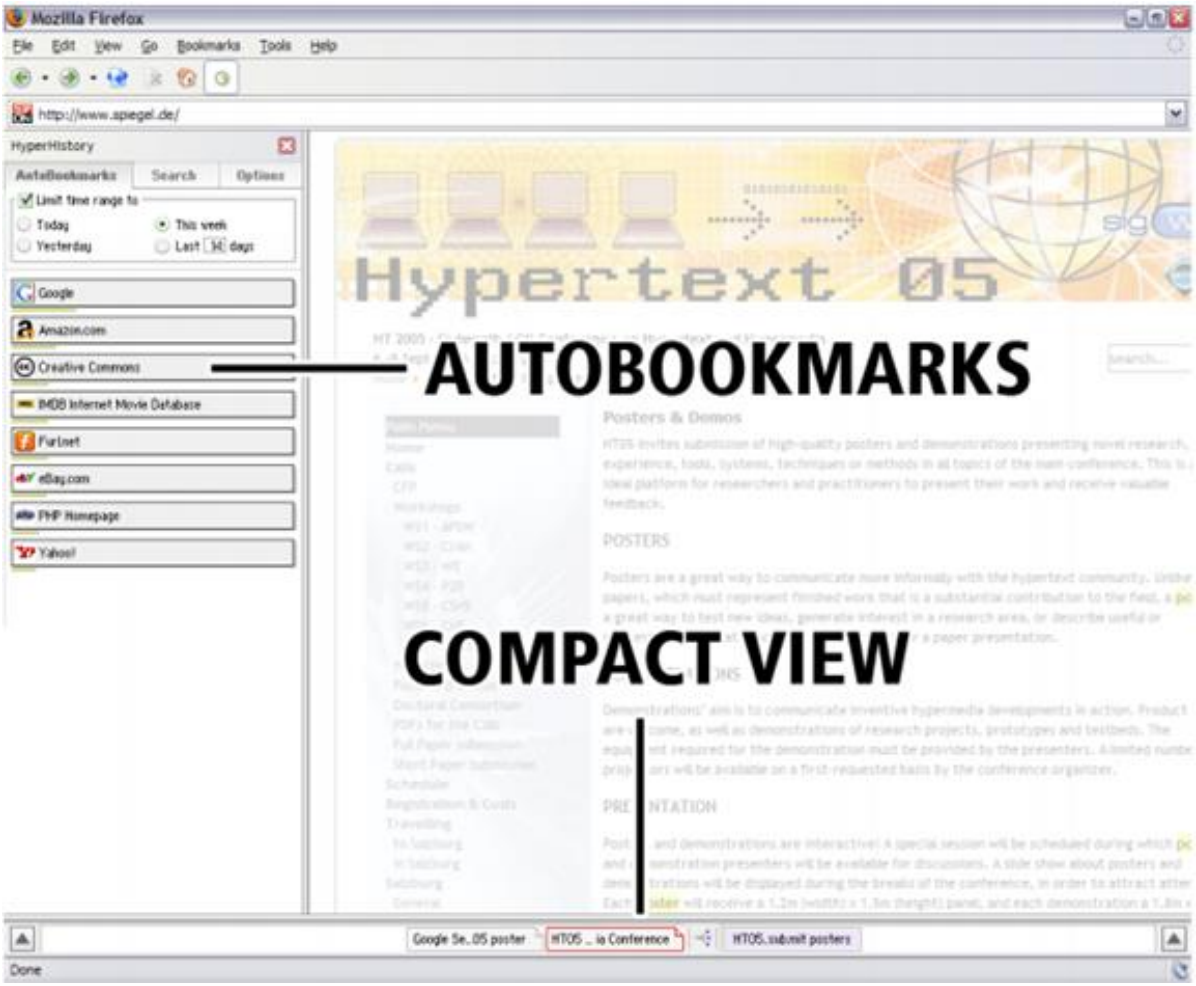


Figure 2.7 Hyper history's Auto bookmarks and Compact view [21]

2.1.7 Pivot bar

Pivot bar [22] is a dynamic browser tool bar which reminds the users of revisited pages that are related to the page they are currently observing. This toolbar uses ranking methods, which estimates for each page what is going to be visited in the next browsing session. It automatically reminds users of past visits which are related to the current Web page. Results show that 22.7% of the revisits were used by this toolbar.

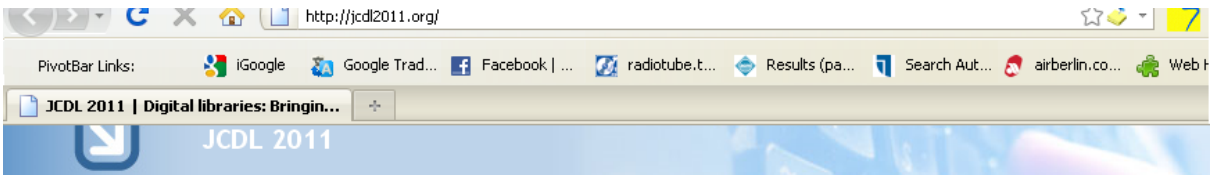


Figure 2.8 pivot bar recommendations [22]

2.1.8 History tree

There are other commercial browser add-ons, such as history tree 1.2 [23], which form a tree structure from top to the bottom. The branching occurs on the tree whenever a new tab was used. Each node in the tree consists of a title and the time of visit. This add-on is best suited for short- term re visitation purposes. The user can simultaneously see their history using this tool while browsing the Web pages. This tool has an option to display the history in a grid view.

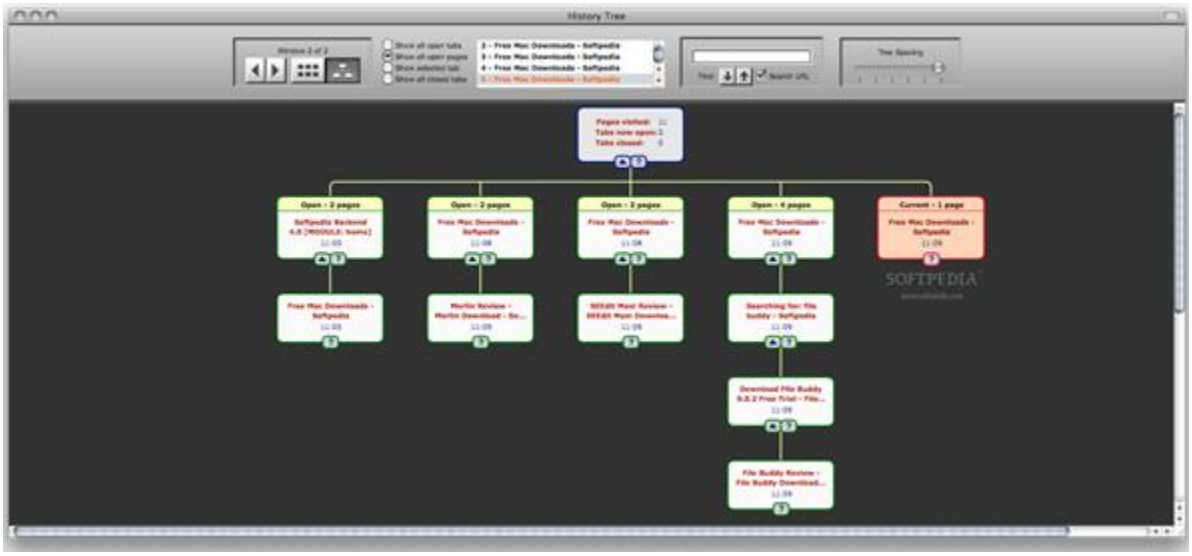


Figure 2.9 History tree: Top to bottom view [23]

Figure 2.10 History tree 1.2: Grid view [23]

2.1.9 Visual bookmark system

T. Yamaguchi [24] created a 3d visualization tool for the browsing history called as a visual bookmark system. It consists of three layout modes: a book mode, a circle mode, and a cube mode. In a book mode, users can see each page as if they are browsing the book. The circle mode displays the images of Web pages around the circumference of a circle. This mode has three pages placed in circle in which center page is bigger than the other pages. In a cube layout mode, images of Web pages are displayed on the surface of the cube, and the browsing history can be viewed by rolling over the cube. The authors believed that by using these functions in the 3d visualization tool, users will browse revisited Web pages effectively.

(a) Book mode

(b) Circle mode

(C) Cube mode

Figure 2.11 3D Layout modes [24].

2.1.10 I-pot

To avoid long textual menus, I-pot [25] a new approach using visual organic and contextual cues which will be implemented in the future. It has two main functions: scanning pages for search terms, and reading page content in a book mark management. The design will be formed on the following basis.

• A plant patch will represent the name of the tree which is used for project description.

• The roots will link different projects to one another.

• A plant pot will represent the primary folder which will be later useful for showing the hierarchies.

• Branches are used for representing the connections.

• Leaves are used as a symbol of data files.

• Flowers or fruits will indicate a special file.

The goal of an I-pot design is to present recognizable visual representations. This tool is expected to be finished within three years of time with good user satisfaction ratings.

Figure 2.12 Organic graphical representation of the I-Pot revisitation function [25]

2.1.11 Eye browse

Eye browse [26] was developed to track the individual browsing activity which can be used for personal as well as social gain. This tool provides quick access to the user's recent browsing activities and generates readable statistical visualizations. Eye browse consists of a Firefox add-

on as well as a Website. The features include Website recommendations, which are based on the past browsing history. Recommendations of Websites are done through previously visited sites, location, frequency of Web sites and duration of browsing activities, daily activities etc. . A survey with 13 users suggests that statistics related to a particular site is very useful while viewing that site.

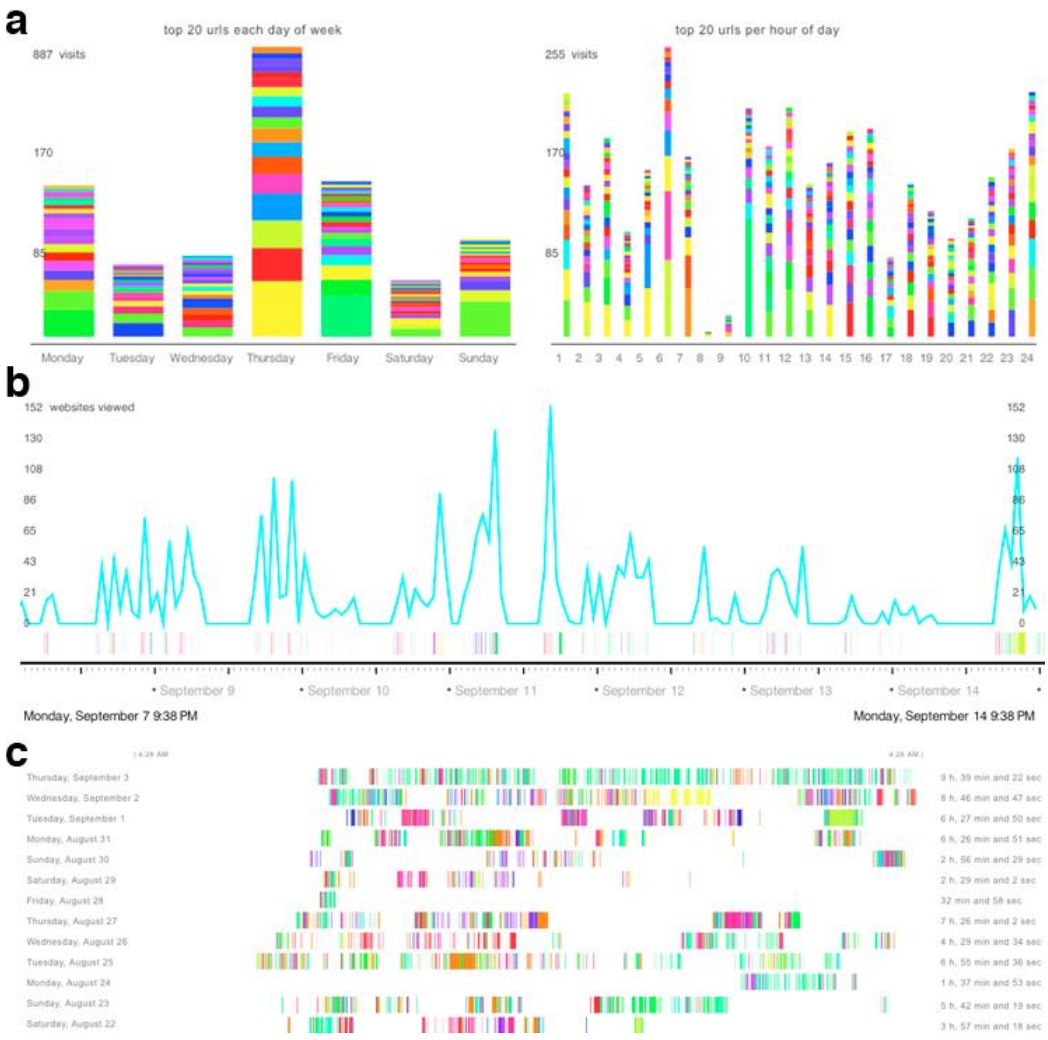


Figure 2.13 Eye browse: (a) Top 20 URLs for day of week and time of day; (b) Timeline of pages visited over the course of 1 week (c) Timeline over 20 days [26]

2.1.12 Web browsing history grid

Gholam Khaksari [27] believes that “an appropriate model for representation of browsing history is a grid, populated with thumbnail images of the captured pages”. The history grid consists of labelled tabs or columns and each tab has its related Web pages. The browsing

history grid stays in the background and it will be activated only after a mouse click on the historian button. By clicking on this button, the history grid comes to the front thus making the browser display blurred out. A user can easily move between the browser view and the history grid easily using historian button. Thumbnail images can be moved to any columns or anywhere within the column using a left mouse click or by dragging them. The other features includes saving thumbnail images, copying and pasting and the zooming. By using this model, the author believes that it will reduce cognitive workload, reduces frustration and creates enjoyable Web browsing experience.

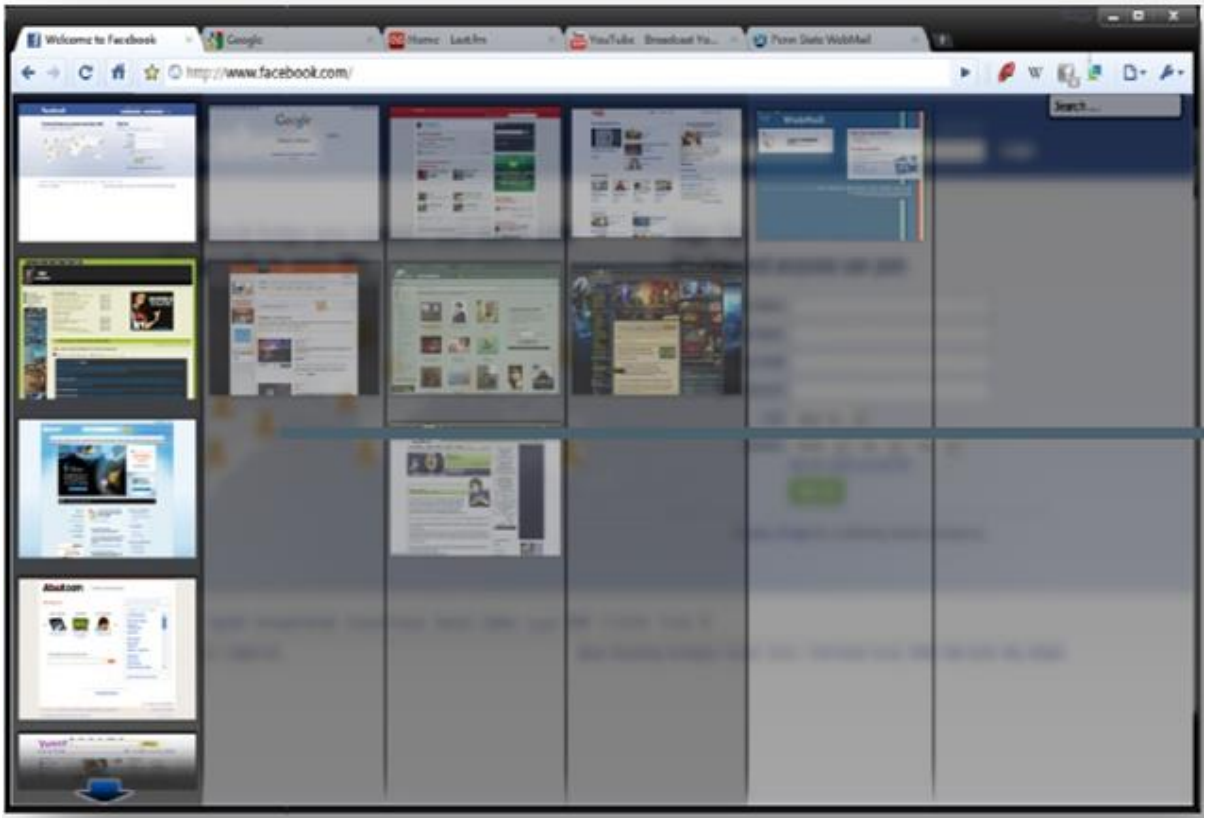


Figure 2.14 Web browsing history grid [27]

2.2. Relevant work

Our research area falls under the category of HCI (Human computer interaction). HCI deals with the interaction between human and the computers and it is also called as a man-machine interaction (MMI). The aim of the HCI is to develop a user-centered interface design where the user can interact with an interface effectively. The poor design can lead to confusion as well as difficult to use [28]. After developing a user-centered design, usability testing is done to evaluate the design by testing with the users. This can be done using either with low-fidelity or

high-fidelity prototypes. A prototype is used for testing the design ideas which is used while developing a working model [29].

- Low-fidelity prototype: This prototype is mainly used in the process of creating a design and it can be presented on a paper or it can be drawn on a computer using software tools. By using low-fidelity prototypes, there is a chance to enable early exploration of alternate designs [30].

- High-fidelity prototype: It is a functional prototype and it is almost similar to the final product with the same interaction techniques. High-fidelity prototypes are mostly expensive and it takes a lot of time to produce when compared to low-fidelity prototypes.

2.2.1 Visual and non-visual display

The representation of a data can be displayed in a visual or a non-visual form. The visual display consists of graphics and different colours for representing the data.
"*Graphics are the visual means of resolving logical problems*" [31]. Tufte (1983) in his book "The visual display of quantitative information" mentioned that the graphical excellence consists of complex ideas communicated with clarity, precision and efficiency [32]. Graphical display should:

- Show the data
- Avoid distorting the implications of the data
- Make large data sets coherent
- Encourage the eye to compare different pieces of data.
- Serve a reasonably clear purpose, description, tabulation or decoration.

The goal of a graphical representation is to show the data in such a way that the viewers can detect and understand the nature, as well as the underlying structure of the data [33]. Graphical representation is also known as visualization, and is defined as:

(i) Form a visual image in the mind.
(ii) To present something which can be visible to the eye [34].

Non-visual display usually consists of table and it does not have graphics in them. This method is mainly used in displaying the large chunks of data in a table format.

2.2.2 Metaphors

A metaphor is defined as an object or a thing which is represented as something else. According to Barr [35], "*a metaphor is a device for explaining a concept or a thing by asserting its similarity to another concept or thing*". Metaphors are known in the field of human computer

interaction and it is often used in the designing process. Three major metaphors in HCI are direct manipulation metaphor, desktop metaphor and the document metaphor.
Examples of direct manipulation metaphors:

- Scroll bars,
- Buttons,
- Check box.

Examples of desktop metaphors:

- Clock,
- Calculator,
- CD player.

Examples of document metaphor:

- Typewriter,
- Microsoft word.

2.3 Comparing the existing tools with our approach

The visualization tools that we saw in the previous research tried to show that a relationship existed between pages, taking the form of a tree [16, 17, 18, and 23] or a graph [20]. This tree reflects the user's path and not the hierarchy of visited Web sites. Most of the visualization tools failed to show the information such as the amount of time spent on a Webpage. The goal of our research is to show the user's hierarchy of visited Web pages. Our belief is that a browser will become more intuitive and efficient by showing the user's path between pages in the history.

The above research was done mainly in the mid–nineties and in the early 2000. Today, the browsing habits of the user have been changed a lot through the usage of different tabs and windows. By considering all the browsing habits of today, we are trying to build a visualization tool for visited Web pages. This visualization tool will generate a graphical overview of the user's path in the form of breadcrumb trails.

Chapter 3

Design

Design is the most important part for building any type of tool [28]. Before finalizing the design there are some important steps that have to be addressed for building a prototype. The front end design can only be formed after having all the required data in the back end. The back end consists of a database where all the necessary data is stored and this is not visible or accessible to the users. The front end is a user interface (UI) where the user can directly interact with it. This chapter was divided into two sections:

1. Back end
2. Front end

3.1 Back end work

For this project the Web browser Mozilla Firefox was used as it is an open source product. In this chapter we will refer Mozilla Firefox as Mozilla. Mozilla stores its history in the places.SQLite database, except for the tabs and windows information. The tabs and windows information are stored in a sessionstore.js file. The information in this file is stored in a Json format. The history information of a Mozilla browser is at the user's profile folder which is located at:
C:\Users\AppData\Roaming\Mozilla\Firefox\Profiles\x2est285.default1358884730935\places.sqlite.

This database includes history as well as information relating to Bookmarks. For our work, we will consider only the information associated with the browser's history. The SQLite database consists of three tables where the history information is stored.

3.1 Understanding the places. SQLite database

3.1.1 moz_places table

The moz_places table consists of seven fields that are used for reconstructing the history of a browser.

(i) Id: It is stored in an integer format, and it represents the number of that particular row in the table.

(ii) URL: It is stored as a LONGVARCHAR, and it represents the entire URL string. The URL does not repeat with the number of visits to the same page. In this case, the count increases in the visit_count field.

(iii) Title: It is stored as a LONGVARCHAR, and it represents the title of the Web page. If there is a blank space in the title, then it means the URL was running in the background.

(iv) rev_host: This is the host name of the URL, which was shown in a reverse order in the table. As with the title, it is also stored in a LONGVARCHAR format.

(v) visit_count: It is stored in integer format. The visit count increases with the number of visits made to the Web page. The usage of back button in the browser does not increase the visit count of a Web page. By refreshing the same Web page, the frequency of visits increases.

(vi) Typed: This is stored in integer format and it flags when the URL is typed in the Web browser.

(vii) favicon_id: This is stored in an integer format which shows the favicon image of the URL.

(viii) last_visit: The time stamp is stored in PR time format. This field stores the time when the URL was last visited including the date.

3.1.2 moz_historyvisits table

The moz_historyvisits table consists of four fields which are useful for generating the history in the browser.

(i) from_visit: It is stored in integer format which represents the previous page from the id of the moz_places table. Google search engine results, as well as typing the URL in the browser will always show 0 in the from_visit field.

(ii) place_id: It is the id column of the moz_places table.

(iii) visit_date: The visit_date stores the time in PR time format, which consists of a 64 bit integer representing the number of microseconds. The PR time format can be

converted into a C Time format dividing date time by 10,00,000 [36]. This SQL syntax can be used to get the URL visited as well as the date and the time of visit.

```
SELECT datetime(moz_historyvisits.visit_date/1000000,'unixepoch'),moz_places.url
FROM moz_places, moz_historyvisits
WHERE moz_places.id = moz_historyvisits.place_id
```

(iv) visit_type: It represents one of the seven types based on the type of visit made to a Web page.

1	TRANSITION_LINK	In this transition type, the user followed a link to get to a browser window.
2	TRANSITION_TYPED	This transition type means the user typed the Web page URL in the URL bar, selected it from URL bar autocomplete results, or either clicked it from a history query.
3	TRANSITION_BOOKMARK	This transition is done only when the user followed a bookmark to get into the Web page.
4	TRANSITION_EMBED	This transition type occurs when some inner content is loaded.
5	TRANSITION_REDIRECT_PERMANENT	This transition occurs when there is a permanent redirecting to the Web page.
6	TRANSITION_REDIRECT_TEMPORARY	This transition occurs when there is a temporary redirecting to the Web page.
7	TRANSITION_DOWNLOAD	In this transition type, the user followed a link from downloads in a browser to get to the Web page.

Table 3.1 Seven different types

3.1.3 moz_favicons table

This table consists of two fields which are useful for this project**.**

(i) Id: It is stored in an integer format, and it represents the number of that particular row in that table.

(ii) Data: The memory of the thumbnail image of the URL is stored.

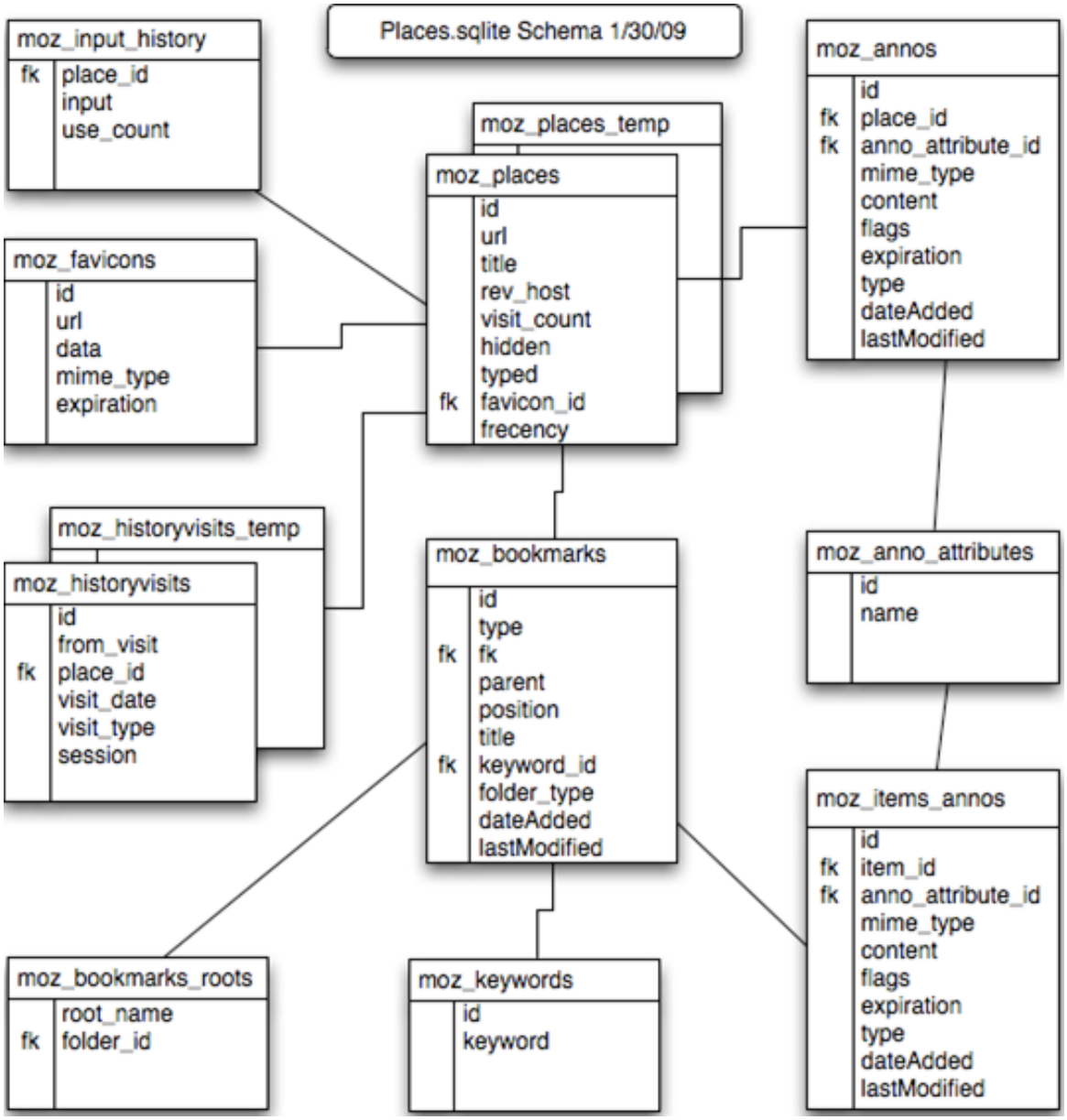


Figure 3.1 Relationship schemas between the tables in the places.SQLite database [37]

3.1.4 Sessionstore.js

Our studies suggest that the entire history information of a Mozilla browser is stored in a places.SQLite database, except the tabs and windows information. The tabs and windows information are stored in a sessionstore.js file. The information in this file is stored in a Json format. The sessionstore.js file is formed only when the browser is closed. Opening and closing a browser consists of a single session. Whenever a new session begins, Mozilla will then erase the previous session. Mozilla stores only one session at a time. The session store.js file has some fields which are essential for the project.

- _closed Windows: This has different windows information which is used while browsing the session. In this type, the current browsing window details in which the initial browsing began will not be displayed.

- _closed Tabs: The tabs that are closed in the middle of a browsing session will be stored.

- Window _closedtabs: The initial browsing window details will be stored including the tabs that are closed during the browsing session.

- Tabs: The tabs that are left opened while closing the browser are stored in this field. This information is mainly used for restoring a session.

- Selected: This indicates the number of tab that that were opened in a session.

- Index: This indicates the number of URL's that were opened in each tab.

The start time and the end time of a session are also stored in the sessionstore.js file. The file is stored in a Json format, which is not a readable format. To make it a human readable format, the file can be opened through any online Json formatter. For my project, we have copied the file and pasted it in www.jsonformat.com to read the information. Now, we have all the information that is necessary to develop a prototype in two different areas.

For the study, we required more sessions however the browser stores only one session. We used a session manager add-on to retrieve more sessions. This add-on retrieves sessions and stores it in a session manager options folder. The sessions are stored based on the time as well as the date when the session was closed. All the necessary data was transferred to a single database. The data from the database is used to display in the visualization tool.

3.2 Front end

To design any prototype, the developer's model should meet the user's mental model [28]. After several studies, three different designs were selected based on the guidelines provided by the past researchers as well as my own ideas. Before implementing the final design, an experiment was conducted using paper prototypes consisting of three designs. This chapter includes the theoretical background that was used for examining the benefits of visualization.

3.2.1 Information Visualization

In 1999, Card et.al [37] defined the term information visualization as "*the use of computer-supported, interactive, visual representations of data to amplify cognition*". *"The purpose of*

visualization is insight not pictures and the main goals of this insight are discovery, decision making and explanatory" [37].

Sachinopoulou [38] has classified the visualization techniques into six groups, which can be represented in a multidimensional space.

- **Geometric technique**: This technique will be useful to visualize the transformations and projections of data in a geometrical space. Some well-known techniques that fall under the geometric technique include scatter plots, plots and matrices, Hyper slice, prosection views, surface plots, volume plots, contours, parallel coordinates, textures and rasters.

- **Icon-based:** Each data item in this technique is represented by an icon, and the values of the attributes are associated with features such as color, shape and orientation of the icon. Some well-known methods that fall under this category include stick figures, chernoff faces, colour icon, Autoglyph, and glyphs.

- **Pixel-oriented**: This technique uses pixels for representing the data. Some of the methods that fall under pixel oriented techniques include space fillings and mosaic plots.

- **Hierarchical displays:** This method includes trees and hierarchies, which are useful for hierarchal relationship as well as network structure. Some of the methods that fall under this technique include hierarchal axes, dimension stacking, trees, world within worlds, and the info cube.

- **Distortion technique**: This method allows distorting the large amount of data, which can be visualized (what does this mean- it can be visualized how?). Some of the methods that fall under this technique include the perspective wall, pivot table and table lens, fish eye view, hyperbolic trees, and the hyper box.

- **Graph based techniques:** This technique is used when there are large graphs which include nodes and edges. This can be presented clearly and quickly. Some well-known methods that fall under graph based techniques include basic graphs, and hyperbolic graphs.

Some well-known statisticians and graphic designers [31, 32, and 33] have offered guidance for building static information via graphics. The dynamic displays need user interface designers [39].
"What can we learn from the visual information seeking mantra" described by Shneiderman [in his seminar paper [39]. The mantra "Overview First, Zoom and Filter, details on demand" describes the presentation of data which can be used as guidelines for building a visualization tool.

- **Overview**: Users should be able to see the data in its entirety.

- **Zoom:** User can examine the interesting items closely.

- **Filter:** Filtering reduces the complexity in the display, without changing the data representation. Users should be able to filter through search input texts or any other GUI component~~s~~.

- **Details on demand:** There should be access to more detailed information about each item than just a detailed view. This can be useful for solving particular tasks such as finding a specific piece of data or relating data points.

- **Relate:** Users can view the relationships between data items. Relationships are determined in detail through similarity between the items.

- **History:** Users should be able to return to the previous displays in an easy manner. The ability to replay a sequence of changes, as well as assist the users in refining the data exploration, is well supported by the history.

- **Extract:** Users should be allowed to save the work, thereby preventing the need to repeat data manipulations to access previously viewed data.

3.3 Paper Prototypes

We formed three different paper prototypes to test the design before implementation. We used the method of low-fidelity prototype for testing these three designs (see chapter 2 in relevant work). The reason for using these three designs was to know whether the user will be able to interact with a design without prior knowledge. We used the method of paper prototyping for selecting the design because of its low cost and time investment (see in chapter 2). Each design had a different look: one was based on a timeline (Figure 3.5), another based on a clock metaphor (Figure 3.6), and the third one was based on a linear list (Figure 3.12). All three paper prototypes possessed identical functionalities, which are used for finding the visited Web pages in the history of a browser. The history logs of the author were used for conducting the experiment and it was used for all the three paper prototypes.

Features of three paper prototypes

The three prototypes have identical features such as tabs and windows information, the day of the week as well as the titles of Web pages.

3.3.1 Design 1

We named the first design as the timeline prototype**.** The timeline is one of the interactive visualization techniques that were used to find information in a compressed space. The timeline is selected based on its features, which can be helpful for finding the visited Web pages based on its time.

Features

We designed the timeline which consists of three stages:

First stage: The seven days of a week will be there in this stage. A rectangle box appears on a day whenever browsing took place. A small chunk appears on the box whenever there is a browsing done on a particular day. Each chunk represents a single browsing session and it is placed on the box based on the time the browsing occurred. The box consists of a 24-hour period format in which half box consists of AM and another half consists of PM. The browsing sessions are highlighted using a dark colour. As shown in Figure 3.2, the user browsed 2 days in a week and that is the reason a rectangle box appeared on those days. The days which are not browsed will not be highlighted. The user can see the details of a browsing day in the second stage.

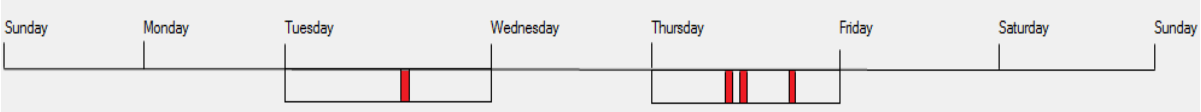


Figure 3.2 Days of a week in the timeline prototype

Second stage: A 24-hour period of a particular browsing day will be shown from the start of the day to the end of a day. In that 24-hour format, the browsing sessions are placed in a small box based on the time. The small chunks that we saw in the first stage can be seen clearly in the second stage. The details of Thursday are shown in Figure 3.3.

Figure 3.3 Browsing day shown in a 24-hour format

Third stage: The third stage consists of a single browsing session details. This stage appears when the user clicks on a browsing session in the second stage. This stage includes the tabs and windows information, as well as the start and the end time of a session. The Web pages are shown in rectangle boxes, and these boxes consist of titles. Each box represents a single Web page.

Figure 3.4 Tabs and windows information for a browsing session

The stages are accessible in numerical order, starting with the first stage and progressing towards the third stage. These stages appear from top to the bottom of the screen which can be seen in Figure 3.5.

Figure 3.5 Overview of the Timeline prototype

3.3.2 Design 2

The second design adapts an analogue clock metaphor. We came up with a new design approach to check whether this technique helps in finding the visited Web pages in the history of a browser. To evaluate this approach we used the paper prototype.

Features

Clock prototype consists of three stages:

First stage: The days of a week are drawn in a clock shape. Each clock represents a day and each day consists of a 24-hour period. Suppose if the user browses on Tuesday between 16:00-17:00 then the timings of that particular browsing session will be highlighted using a dark

colour. The user can click on the highlighted session to see the browsing details. The days that are not browsed will not be highlighted.

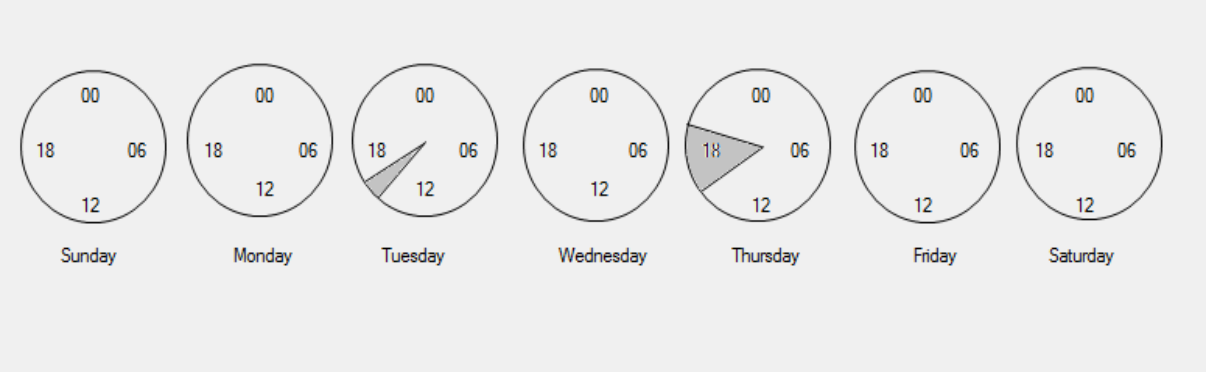


Figure 3.6 Days of a week in the clock prototype

Second stage: A 24-hour time period of a particular day will be shown in two 12-hour time period intervals. Each twelve hour period consists of an AM and PM in the form of a clock shape. The browsing session will be highlighted based on the time the user had browsed. Suppose that the browsing was done in the AM period, and then the particular time in that period will be highlighted.

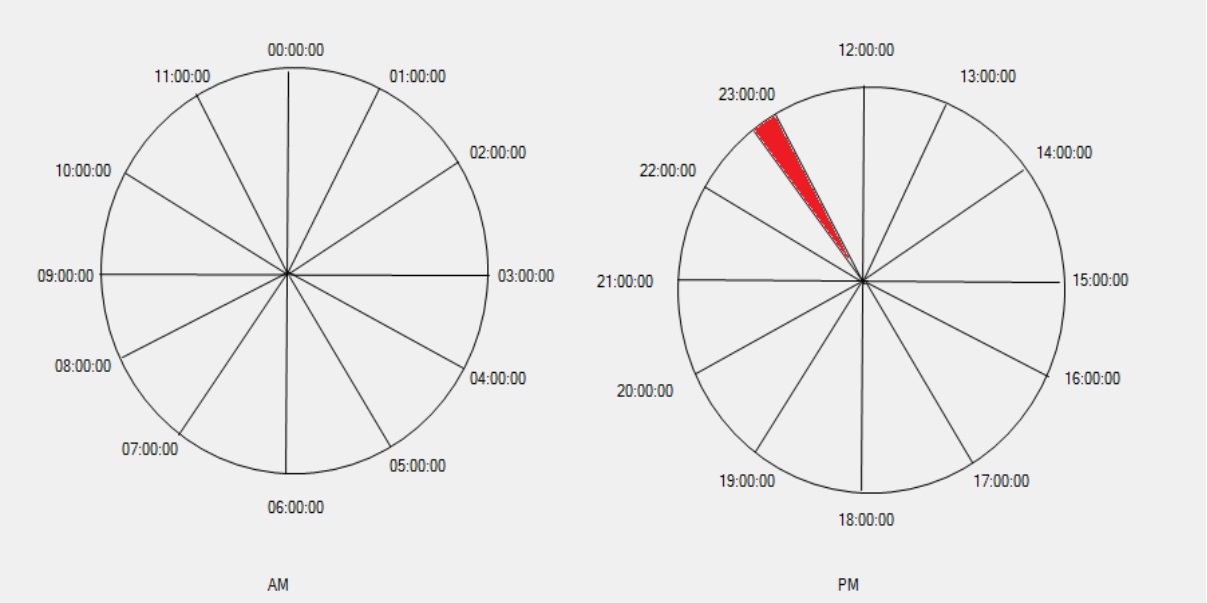


Figure 3.7 Twelve hour period intervals for a browsing day

Third stage: This stage consists of the browsing session details. This stage occurs only when the user clicks on a particular browsing session in the second stage. In this type, the small clock appears based on the number of tabs that are used while browsing. Each tab consists of a single clock, which shows the time spent on a Web page. The title of the Web page is presented inside

the clock based on its time. The quadrants in the clock appear based on the number of Web pages. Suppose if the user opened three tabs while browsing then three different clocks will appear in the third stage.

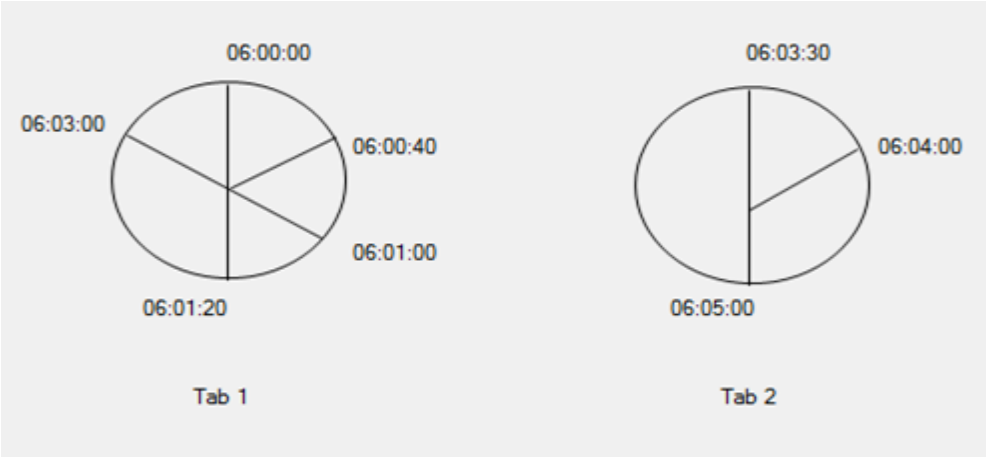


Figure 3.8 Browsing session details including tabs information

3.3.3 Design 3

This design was selected based on its simplicity to represent the data. We named this design as the table because of its non-visual aspect to display the data. Tables are mainly used for keeping all the data in one place. This approach is used mainly in databases for exploratory purposes. The table was selected for its simple look to find the visited Web pages in the history of a browser.

Features

The table prototype consists of three stages:

First stage: The days of the week are shown in this stage. Each day is represented using a rectangle box. The browsing days are highlighted using a horizontal line within the box. The days that are not browsed will not be underlined. As shown in Figure 3.9, two browsing days occurred during the week.

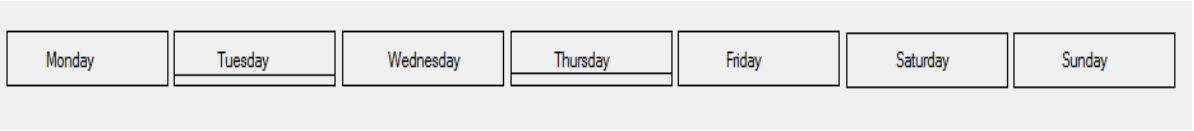


Figure 3.9 Days of the week in the table prototype

Second stage: When the user selects a particular day, browsing sessions for that day will be shown. The browsing times will be shown using a rectangular box on the left side of the main screen. For example, if the user browses on Thursday then the browsing sessions for that particular day will be listed.

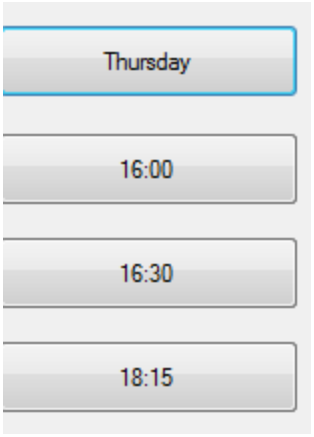


Figure 3.10 Browsing sessions for a day

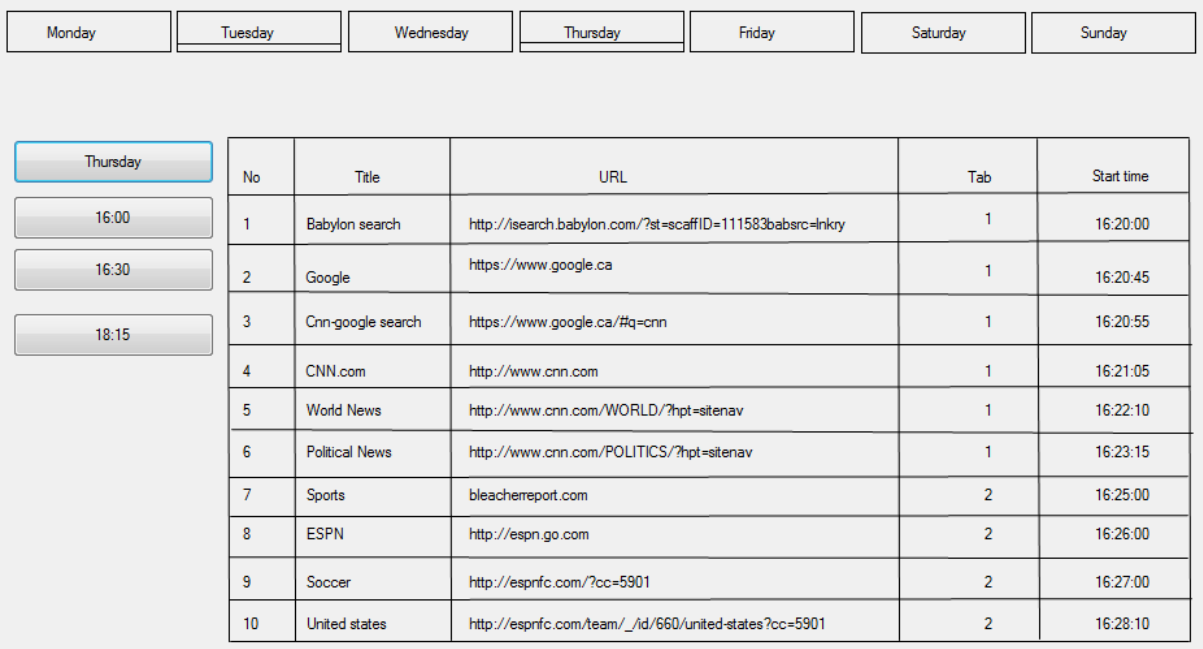
Third stage: This stage consists of the browsing session details. When the user selects a particular time, then the details will be shown in a table format. In the table, the row number will be used for each Web page. Each row consists of a row number, title, URL, tab number, and the start time.

No	Title	URL	Tab	Start time
1	Babylon search	http://isearch.babylon.com/?st=scaffID=111583babsrc=lnkry	1	16:20:00
2	Google	https://www.google.ca	1	16:20:45
3	Cnn-google search	https://www.google.ca/#q=cnn	1	16:20:55
4	CNN.com	http://www.cnn.com	1	16:21:05
5	World News	http://www.cnn.com/WORLD/?hpt=sitenav	1	16:22:10
6	Political News	http://www.cnn.com/POLITICS/?hpt=sitenav	1	16:23:15
7	Sports	bleacherreport.com	2	16:25:00
8	ESPN	http://espn.go.com	2	16:26:00
9	Soccer	http://espnfc.com/?cc=5901	2	16:27:00
10	United states	http://espnfc.com/team/_/id/660/united-states?cc=5901	2	16:28:10

Figure 3.11 Browsing session details for the table prototype

The features include a variable zoom option as well as a search option to find the Web page. In the paper prototype, we did not show these features.

The second stage is formed on the left side of the screen after selecting a day. The third stage will be formed in the center after selecting the times which are displayed in the second stage. All the stages appeared one after another after selecting a particular browsing day which can be seen in figure 3.12.



No	Title	URL	Tab	Start time
1	Babylon search	http://isearch.babylon.com/?st=scaffID=111583babsrc=lnkry	1	16:20:00
2	Google	https://www.google.ca	1	16:20:45
3	Cnn-google search	https://www.google.ca/#q=cnn	1	16:20:55
4	CNN.com	http://www.cnn.com	1	16:21:05
5	World News	http://www.cnn.com/WORLD/?hpt=sitenav	1	16:22:10
6	Political News	http://www.cnn.com/POLITICS/?hpt=sitenav	1	16:23:15
7	Sports	bleacherreport.com	2	16:25:00
8	ESPN	http://espn.go.com	2	16:26:00
9	Soccer	http://espnfc.com/?cc=5901	2	16:27:00
10	United states	http://espnfc.com/team/_/id/660/united-states?cc=5901	2	16:28:10

Figure 3.12 Overview of the Table prototype

Chapter 4

Design Selection

This chapter explains the methodology that was used for the experiment and the results that were gained after running the experiment. Three paper prototypes were tested to find the best design amongst them in regards to the completion time, accuracy and the Likert scale.

4.1 Research Methods

Our approach was to conduct an experiment with three different paper prototypes, after which participants filled out a questionnaire with each prototype to determine the answers. After finishing the questionnaire the participants were asked to rate the difficulty level of each of the three prototypes. Our goal was to find the best design among the three prototypes. The best design was determined based on three variables, which are completion time, accuracy and the Likert scale. The system we chose to design was the history of a browser which has some visualization concepts in displaying them. Since the questions are the same among the three prototypes, we attempted to eliminate effects caused by the order in which the three designs were presented in. This was accomplished using permutations and combination concept in which we followed 3! (3*2*1 = 6). This concept gave us an idea to change the orders in six different ways. The three paper prototypes are named in this chapter as timeline, clock and table. Timeline is referred as P1, clock is referred as P2 and the table is referred as P3.

The six orders are given below:

Order 1: P1 P2 P3

Order 2: P1 P3 P2

Order 3: P2 P1 P3

Order 4: P2 P3 P1

Order 5: P3 P1 P2

Order 6: P3 P2 P1

4.2 Participants

We conducted the experiment with 30 participants, consisting of 17 males and 13 females from the Laurentian University. The 30 participants were divided into six different groups, and each group contained 5 participants. Each group participated in one of the orders from orders 1 to 6. They were from the department of Mathematics and Computer science. The age of participants ranged from 20 to 30 years old. All participants were asked to participate in the study voluntarily. When the participants were asked about their browsing habits, most stated they browse at least five hours a day. After obtaining the ethics approval certificate from the ethics board, the experiment was started (see Appendix A). A consent form was given to each participant to let them know the purpose of the experiment (see Appendix A).

4.3 Pilot study

A pilot study was conducted with 5 participants to test the prototypes, as well as to verify the experimental procedure. The pilot testing helped in refining the questionnaire. It was felt that an unstructured interview was required to get feedback from the users after the completion of experiment.

4.4 Procedure

The experiment started with a short introduction, in which the experimenter explained the study to the participant. The participants were instructed to determine the answers to three questions relating to information contained within each prototype. The questionnaire can be seen in Appendix B. The participants were informed that that time will be noted using a stop watch, and that there was no time limit for answering the questions. After completing the questions, the participants were asked to indicate the level of difficulty for each design based on a five point Likert scale (see figure 4.1). At the end of the experiment, a short unstructured interview was conducted to determine what aspects of the prototype the participant thought were effective and also the possible avenues for improvement. On an average, all the participants finished the experiment within 15 minutes.

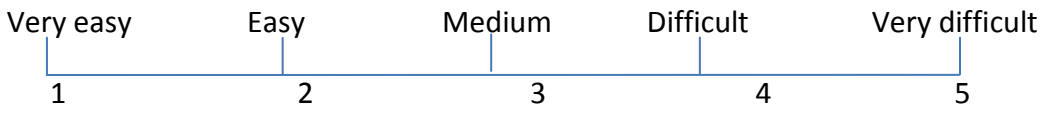


Figure 4.1: Five point Likert scale

4.5 Results

In order to assess the results, we used the ANOVA (Analysis of Variance) to identify the statistical differences between the three prototypes. Alpha level refers to significance level and it's the pre-chosen probability value used for rejecting the null hypothesis. No statistical difference is known as a null hypothesis. The popular alpha level values used in statistics are 0.01 and 0.05. A p-value is the estimated probability to find out if the null hypothesis is true. The p-value is compared with alpha level to find the statistical difference. If the p-value is less than alpha level then it rejects null hypothesis or if it's greater than alpha level then there is no statistical difference. The mean differences were tested using one-way analysis of variance with an alpha level of 0.05. ANOVA is useful in comparing the means when there are more than two groups available. Between columns/groups and within columns/groups are the two variances that are essential while calculating the statistical difference. Sum of squares and mean squares are used for finding the variances. Sum of squares includes SSC, SSE, SST and each of them has a different degree of freedom (df). SSC is calculated between columns/groups, and it has a degree of freedom df= C-1 (C is the column number). SSE is calculated within the column/group, and it has a degree of freedom (df), where df= N-C (N is the total number of participants). SST is the total sum of squares and it has a degree of freedom df= N-1. Mean square has MSC (Mean square columns) and MSE (Mean square error) for finding the variances between column/groups and within the columns/groups. The below are the formulas used for finding the mean squares:

1. MSC = SSC/df
2. MSE = SSE/df

The ratio of two variances are calculated using F =MSC/MSE. The results were analyzed using SPSS software. Completion time, accuracy and the Likert scale are the three variables which form the basis of the results. Each paper prototype consists of these three variables. If any point lies beyond a set range (determined by the upper and lower quartiles) then it is deemed to be an outlier. The outliers from the time variables have been removed for analyzing the results.

Number of outliers found in time variable in all the three prototypes:

1. Timeline – 3 outliers found.
2. Clock – 1 outlier found.
3. Table – 4 outliers found.

After removing the outliers, the mean time to completion for the three prototypes were calculated individually, and are shown in Figure 4.2.

Time completion

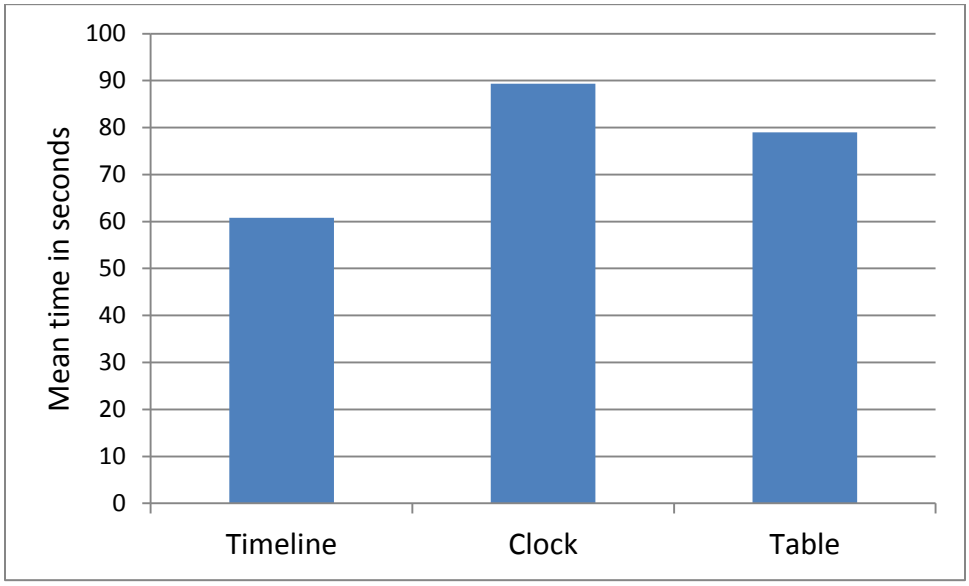


Figure 4.2 Mean times for the time completion results chart

The mean values show that the users took less time to perform the tasks using the timeline prototype, with a mean time μ= 60.79 seconds, whereas the clock prototype had a mean time μ = 89.32 seconds, and the table prototype had a mean time μ=78.99 seconds. The one-way ANOVA results suggest that there was a significant difference in the meantime on time completion, F(2,79)= 8.734,sig =0.00. Table 4.1 shows the statistics for completion time using the three paper prototypes.

ANNOVA

Tools

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	11585.675	2	5792.837	8.734	.000
Within Groups	52396.819	79	663.251		
Total	63982.494	81			

Table 4.1 One-Way ANNOVA statistics table for time completion

Accuracy

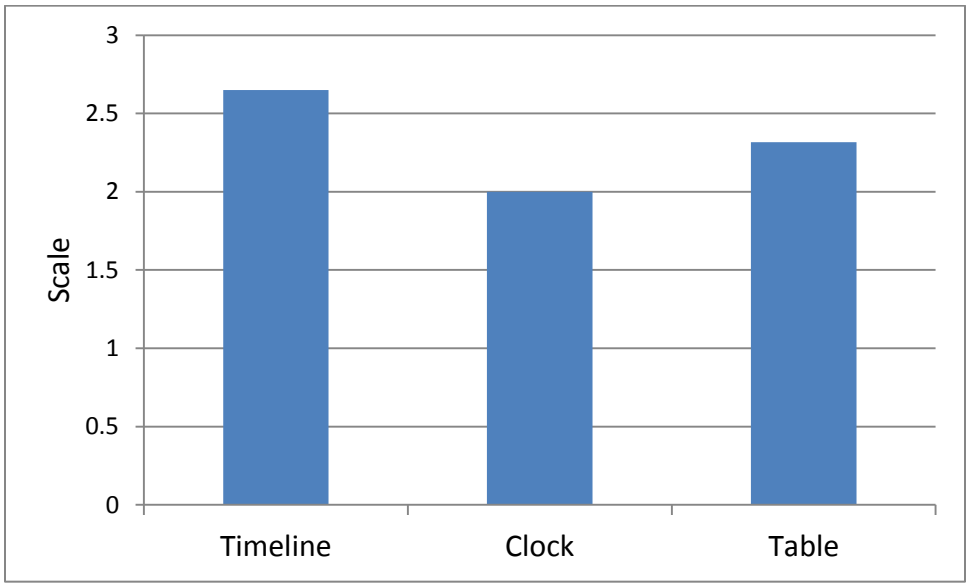


Figure 4.3 Mean values for the accuracy results chart

The overall mean values indicate that usage of the timeline prototype resulted in the highest accuracy, with a mean value μ=2.65 on a 3 point scale. The use of the table and clock prototypes resulted in lower accuracies. The clock prototype had a mean value μ=2.31, whereas table prototype had a mean value μ= 2.18. The one-way ANOVA results suggest that there was a significant difference in the mean value on accuracy, F (2, 87) = 4.49, sig =0.014.

Tools

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	3.467	2	1.733	4.494	.014
Within Groups	33.558	87	.386		
Total	37.025	89			

Table 4.2 One-Way ANNOVA statistics table for accuracy

Likert Scale

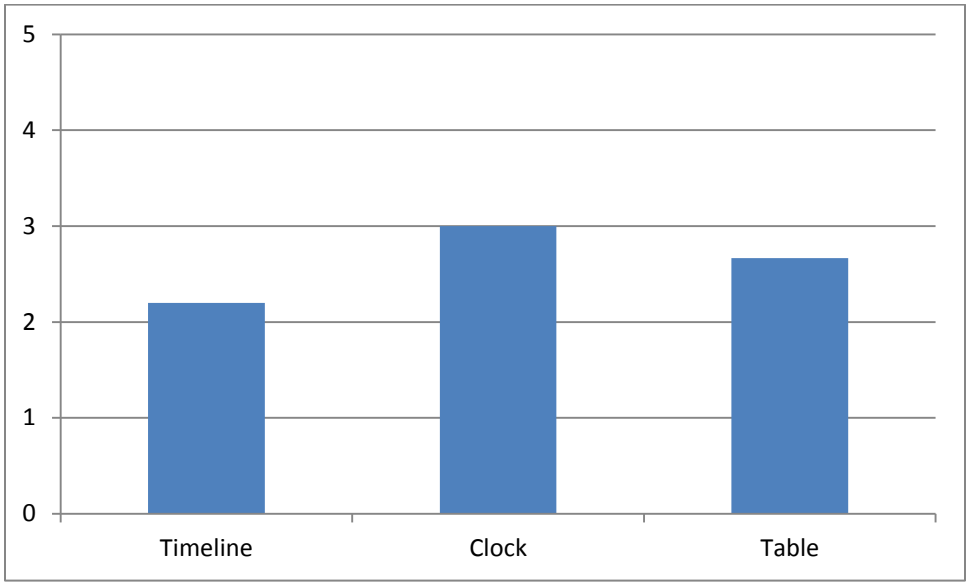


Figure 4.4 Mean values for the Likert scale results chart

The overall mean values indicate that the users rated timeline as the easiest prototype, with a mean value μ=2.18, whereas table was rated as the second easiest prototype with a mean μ= 2.66. The clock design was rated as a medium interface to understand with a mean value μ= 3. The results suggest that there was a significant difference in the mean value on Likert scale, F (2, 87) = 3.92, sig =0.023.

ANNOVA

Tools

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	9.689	2	4.844	3.922	.023
Within Groups	107.467	87	1.235		
Total	117.156	89			

Table 4.3 One-Way ANNOVA statistics table for Likert Scale.

The mean values indicate that the users perceived the timeline prototype as being the most convenient and intuitive to use for displaying the Web pages. Based on the results that are provided here, the timeline performed well among the three prototypes.

4.6 Discussion

The experimental results suggest that the timeline design provided the best performance and is the preferred method for displaying the history of Web pages among the three prototypes. While conducting the experiment we observed that the users were taking more time to answer the question using with the clock prototype. We asked the users after completing the experiment and most of the users replied that the design was not easy to understand. Some users replied that they are not good at remembering the numbers so that is the reason it took them time to answer the task. Some participants were not answering task#1 correctly using timeline. We came to know by the users that the small chunks in the box are not easy to interpret. The users were not able to understand the concept of line under the day in the table prototype and due to this they answered the task#1 wrong.

The unstructured interview revealed many suggestions for future improvements to the designs. We asked few questions to determine which features the participants thought were beneficial for developing a working prototype. Most of the participants have suggested that we include a thumbnail along with the title of the Web page. The participants enjoyed the experience of answering the questions with three different prototypes.

Example of one of the user's answers to the unstructured interview questions after interacting with each tool.

Questions	Answers/Suggestions
1. Which feature you like the most in all the three prototypes?	Tabs and windows information according to the time.
2. Do you use different tabs while browsing?	Yes. I like the way that is representing which Web page belongs to which tab.
3. Which prototype you like the most and why?	Timeline and the table prototype because the design is very easy to understand. Out of those two prototypes, I would select timeline.
4. What suggestions you would like to give us?	Add the zooming features and keep the days in a rectangle shape. Keep the thumbnail with the title.

Table 4.4 Example of Unstructured interview questions

In the next chapter, the timeline tool will be compared with the text-based history to analyze the results.

Chapter 5

Prototype Implementation

This chapter describes in detail the design and implementation decisions of the timeline prototype. Here we discuss the functionality of the prototype and design decisions for its appearance. We also detail an experiment that was conducted to compare the timeline prototype against the existing text-based history in the Mozilla browser.

5.1 Designing process

The following technologies were involved in the development of the timeline tool:

- Java with collection framework
- Servlets and JSP
- JDBC
- JQuery (JavaScript library)
- HTML
- CSS
- JSON

We implemented the horizontal timeline structure with JavaScript code using JQuery library (Appendix F). We used Java for dealing with the JSON format in the sessionstore.js file using libraries such as Org.JSON. This library was mainly used for parsing the JSON data. All the necessary information data was retrieved and placed in the MySQL database. The database was connected to Java code using JDBC driver (see Appendix D). The server that we used in this project was MYSQL server. Using Java code, the data from the table was sent to the JavaScript code to get the timeline output (see Appendix E). The entire design architecture can be seen in Figure 5.1.

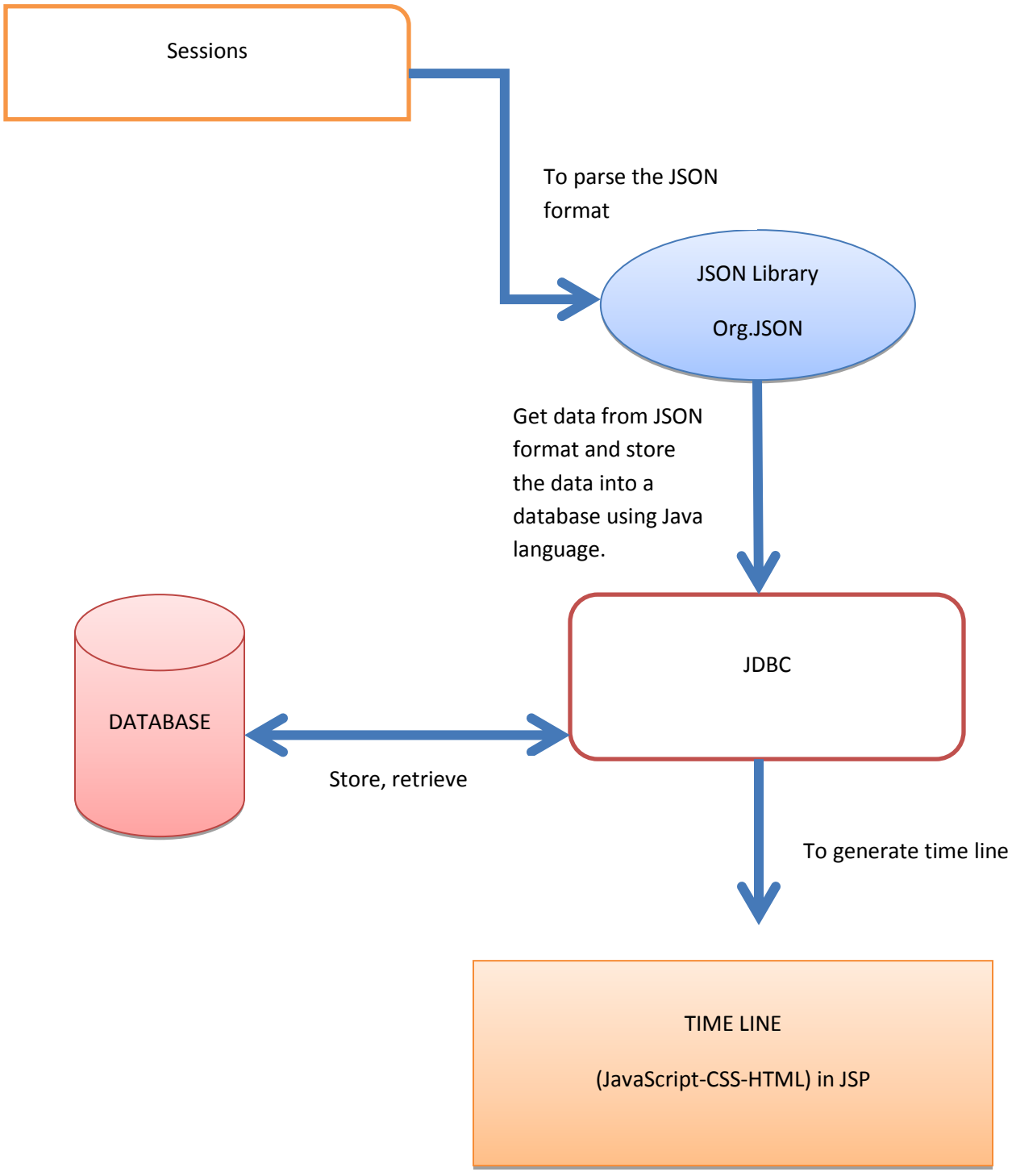


Figure 5.1 Back-end architecture

5.2 Timeline features

The timeline was implemented based on the suggestions from the users in the experiment 1, as well as our own ideas (see discussion in chapter 3). The entire timeline tool consists of three sections: days of a week, hourly-format and the list of visited Web pages which include tabs and windows information. The general screen can be seen in Figure 5.1. The entire history of the past seven days can be seen in one place. The number of events indicates the number of Web pages that were browsed in a day. The user can click a particular day to see the details of a Web page.

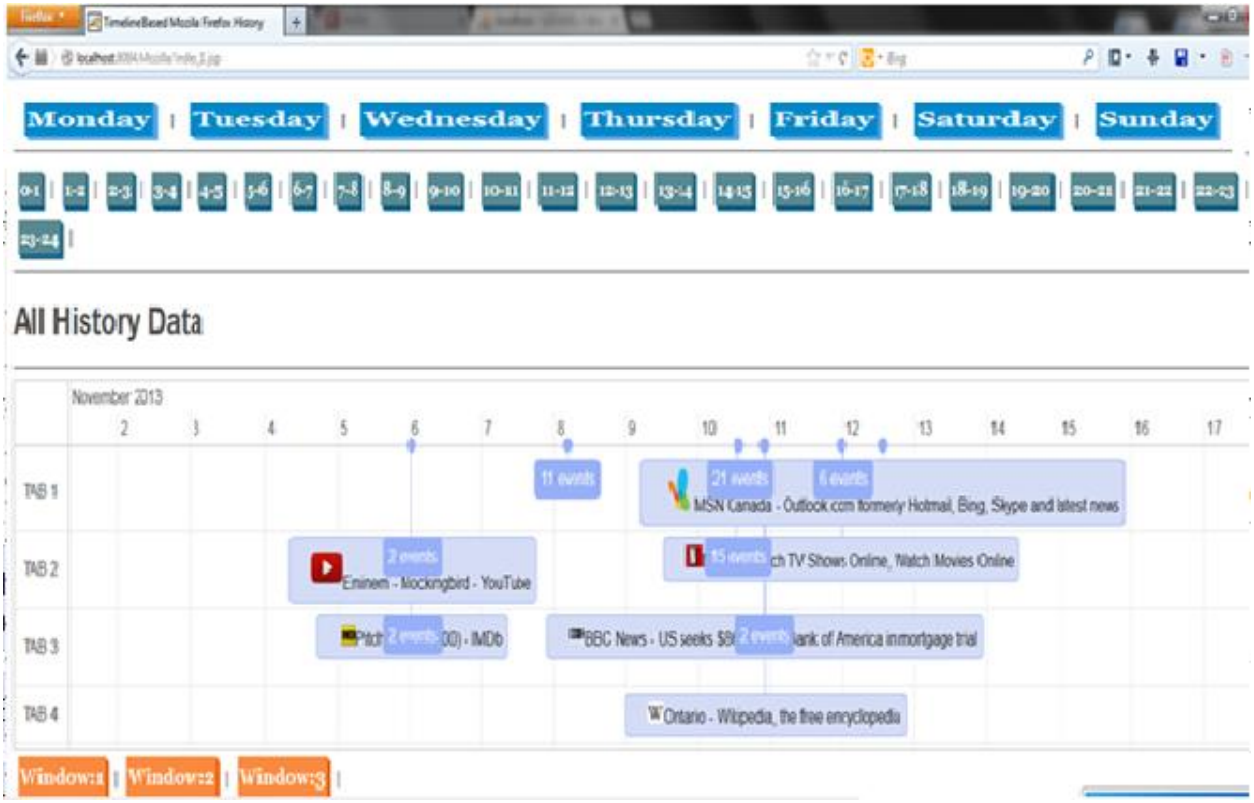


Figure 5.2 General Screen of the timeline prototype

5.2.1 Days of a week

The history logs of previous browsing sessions which contains at least seven days were imported into the timeline tool. Each day is represented using a rectangle box. Each box is represented using a sky blue color. When the user clicks on a particular day, the rectangle box is highlighted after displaying the Web pages (see Figure 5.3). No result found appears on the screen if there was no browsing done on any day (see Figure 5.7).

Figure 5.3 Days of a week in the timeline prototype

5.2.2 Hourly- format

When the user clicks on a particular day, then the browsing details of that day is shown in an hourly- format. Suppose if the user clicks on Sunday, then the browsing session timings of that particular day will be shown (see Figure 5.4). The user can click on any of the timings to navigate the Web page.

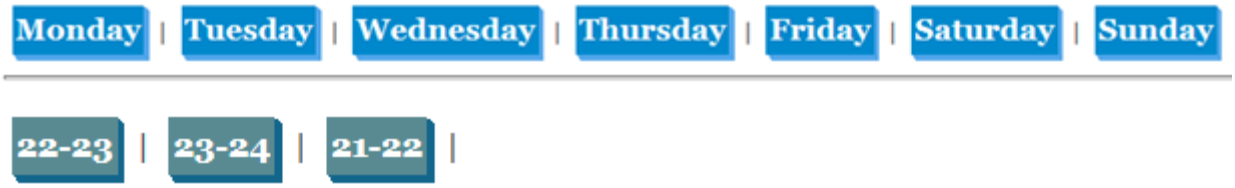


Figure 5.4 Hourly-wise browsing details

5.2.3 List of visited Web pages

The user can see the visited Web pages during a selected time interval. Every Web page was represented by a rectangle box, and these rectangles can be accessed from a left to right direction. The user can see the Web pages by dragging them with either a mouse or a touch pad. Every Web page contains a favicon and a page title (see Figure 5.5).

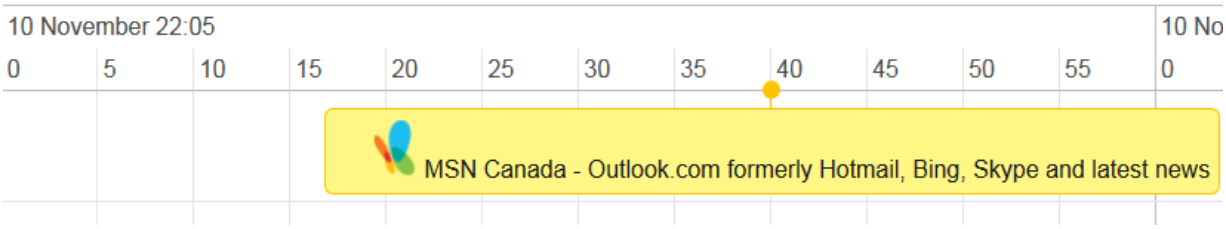


Figure 5.5 Favicon along with title of a Webpage

Each page rectangle was placed according to the tab number. Window rectangles are placed at the bottom of the screen, which consists of multiple tabs information. When the user clicks on

any Web page then the title, URL and the duration of the page can be seen at the bottom of the screen. The features include Zooming, where the user can zoom up to milliseconds (Figure 5.6).

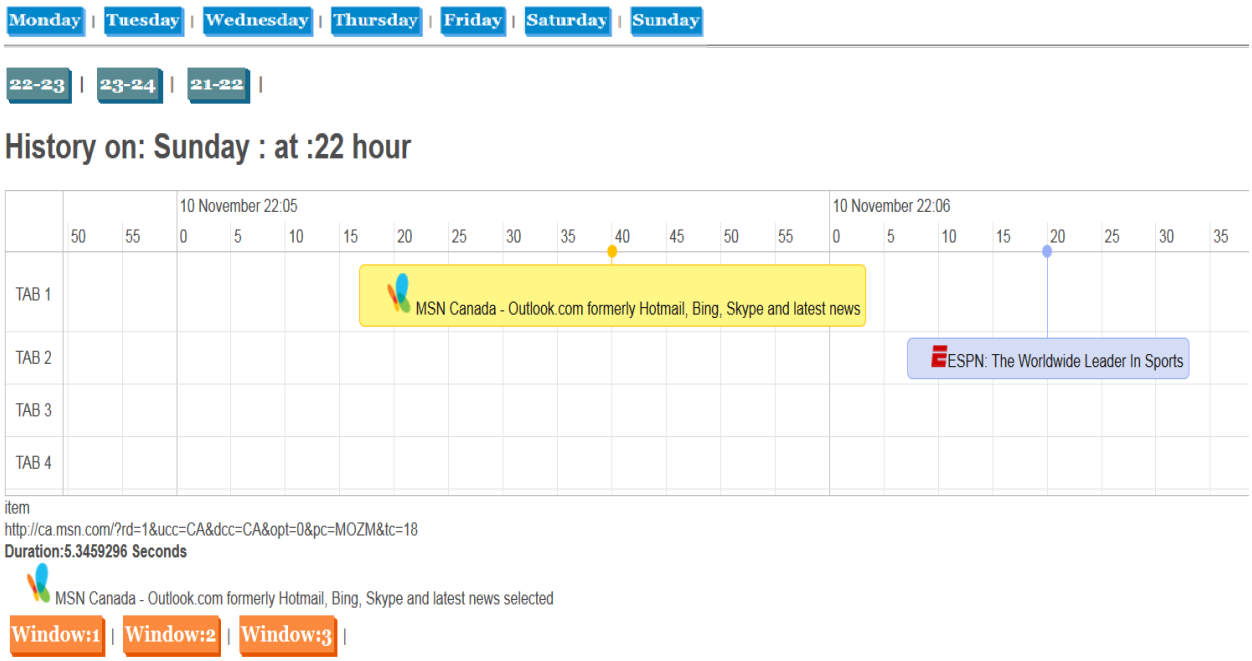


Figure 5.6 Duration of a Web page and zooming

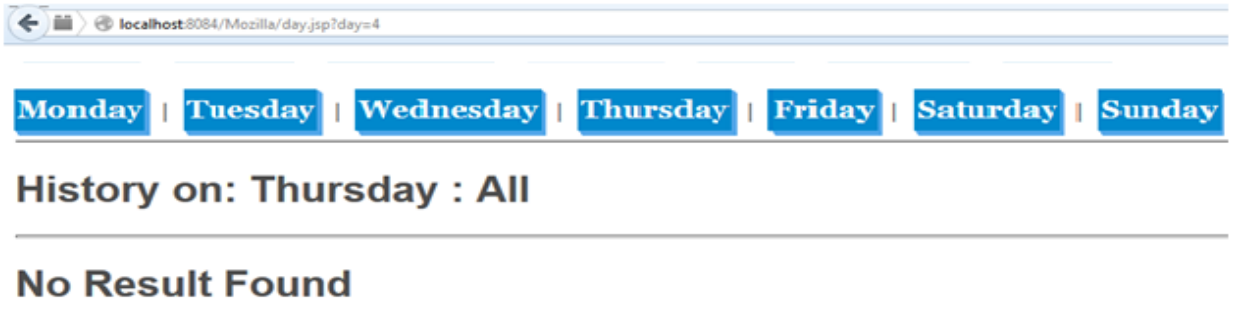


Figure 5.7 History information not available

The timeline tool was tested for its usability by performing some evaluation tests. The timeline tool is compared with the existing text-based history tool as detailed below.

5.3 Implementation of the final design

Aim: The aim of the study was to compare the timeline tool with the text-based history tool in the Mozilla browser with regards to the completion time, accuracy, and user satisfaction.

5.3.1 Research Methods

Our approach was to conduct an experiment with two different tools, in which participants completed a questionnaire with each tool. For this experiment, the user interacted with the existing history log files using two methods: the timeline visualization tool and the text-based history in the Mozilla browser. After finishing the questionnaire, the participants are asked to rate the difficulty level of each tool. The results were analyzed based on the three variables, which are the completion time, the accuracy and the Likert scale. The tasks can be seen in Appendix B. Since the tasks were the same for each tool, we counterbalanced the tools so to eliminate the effect caused by order. This was accomplished by changing the orders of the prototypes in two different ways. The two tools are referred as Timeline (T) and the text-based history (B).

The two orders are given below:

Order 1: T, B

Order 2: B, T

5.3.2 Participants

We conducted the experiment with 20 participants, consisting of 11 males and 9 females and they were recruited from the Laurentian University's Computer Science department. The participants were divided into two different groups, each group containing 10 participants. The participants from each group had finished the experiment with at least one order from the given orders. The participants were familiar with the text-based history in the Mozilla browser. A consent form was given to each participant who had joined in the study for letting them know the procedure for the experiment (see Appendix A). All the participants signed the consent form before starting the experiment.

5.3.3 Pilot Study

A pilot testing was conducted with 4 participants to test the prototypes as well as to practice and perfect the experimental procedure. By conducting the pilot testing, we were able to refine the design as well as the questionnaire. The changes include adding the duration of a Web page

to the timeline tool at the bottom of the screen (see figure 5.6). The questionnaire was refined based on the grammatical errors.

5.3.4 Procedure

The experiment started with a brief introduction, in which the experimenter explained the study to the participant. Before starting the experiment, participants were trained in the use of timeline tool. Training included informing the participants about the features of timeline tool, and how it could be used for navigating the Web pages. Training was not given to text-based history tool as it was assumed users were already familiar with the Mozilla browser. Participants from each group were asked to answer four questions by interacting with each tool, and the time was noted for each question using a stop watch. After completing the tasks, the participants were asked to indicate the level of difficulty for each tool based on a five point Likert scale (See chapter 4, figure 4.1). Each participant had finished the experiment within 15 minutes.

5.3.5 Results

The results were analyzed using a t-test to identify the statistical differences between the two tools. The independent sample t-test is widely known statistical test and it is widely used [40]. The mean differences were tested using an independent sample t-test with an alpha level of 0.05. The completion time, the accuracy and the Likert scale were the three dependent variables tested. The outliers from the completion time variable were removed for analyzing the results.

Number of outliers removed from the time variable in both tools.

1. Timeline – 2 outliers.
2. Mozilla browser – 6 outliers.

After removing the outliers, the overall mean values in the four questions were calculated individually. The summary of results chart is shown in Figure 5.8.

In the results, we can see two types of tables;

1. Descriptive statistics table: In this table, we can see the number of participants, mean values and the standard deviation for each group.
2. Independent samples test: By default, the table shows Levenes's test for equality of variances. In this type, there are two assumptions that are made: one is when equal variances are assumed and the other, when variances are not assumed to be equal. If the p-value in this table is less than the 0.05 then there is a significant difference

between the groups. The measure of significance in this table is denoted using sig(2-tailed) column.

Time completion

Task #1

The task was to find the number of days browsing occurred during the week in the history logs. The user had completed the task#1 by interacting with each tool separately. The mean values show that the users took less time to answer the task#1 using the timeline tool, with mean time μ= 25.98, whereas text-based history tool had a mean time μ = 72.38. The results of t-test suggests that there was a significant difference in the meantime in finishing task #1 using both tools, $F(20,19)=13.075$, $p=0.001$. The statistical difference table can be seen in the Appendix C.

Task#2

The task was to know what web page was visited between given time intervals. The participants completed the task#2 using timeline tool with a mean time μ= 21.71 seconds, whereas with the text-based history it took more time to answer the question which had a mean time μ = 30.67 seconds. The results of t-test suggests that users finished the task #2 quickly when compared to text-based history, $F(18, 17) =6.288$, $p=0.017$. The statistical difference table can be seen in the Appendix C.

Task#3

The task was to find a particular Web page start time and the tab number that was opened in the browser. The text-based history had a mean time μ= 85.54 seconds, whereas timeline tool had a mean time μ = 81.26 seconds. The results suggest that there was no significant difference for the completion time using both tools. The results table can be seen in the Appendix C.

Task#4

The task was to find the amount of time spent on a Web page. The users completed the task by interacting with both the tools. The mean values show that the users finished the task#4 faster with a mean time μ= 26.77 seconds as compared to text-based history tool. The results from the t-test suggest that there was a significant difference in the mean time using both the tools with a level of confidence $p = 0.000$. The statistics table can be seen in Appendix C.

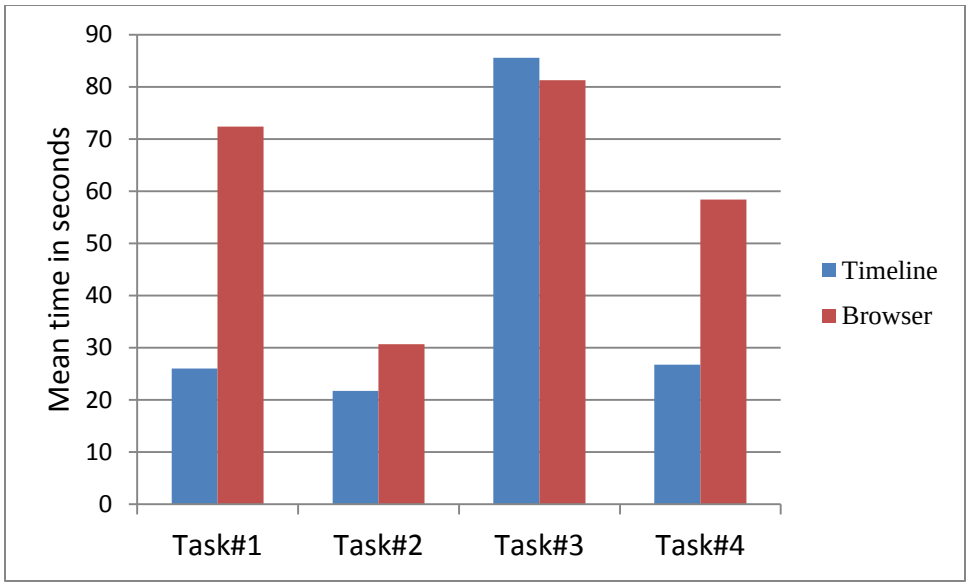


Figure 5.8 Summary of mean time in the completion time chart

Accuracy

The overall mean values indicate that the users answered well with a mean value μ = 3.70 while interacting with the timeline tool. The text-based history had a mean value μ = 2.25. The t-test results suggest that user's accuracy rate was high using with the timeline tool when compared to text-based history tool. The statistical summary can be seen in Appendix section.

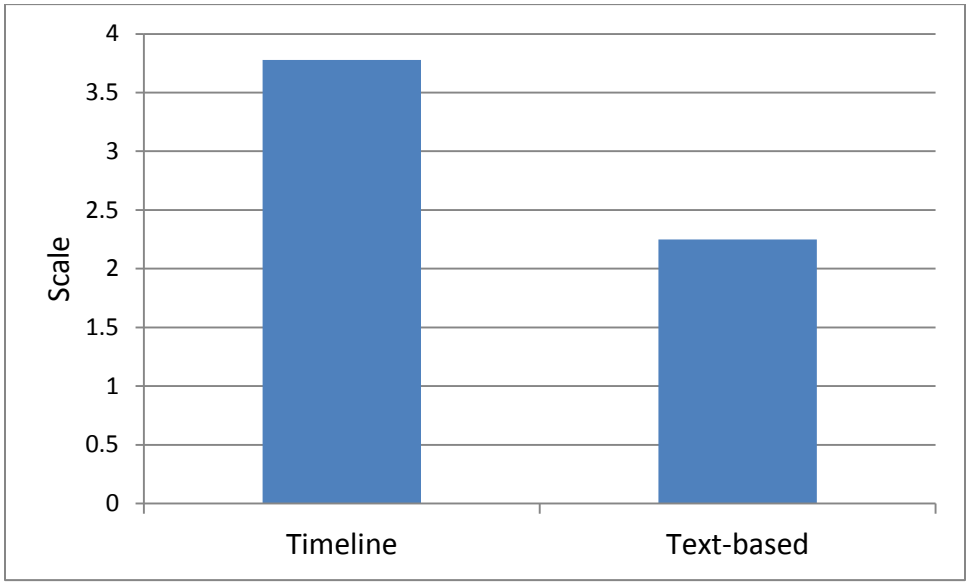


Figure 5.7 Mean values chart for the accuracy

Likert Scale

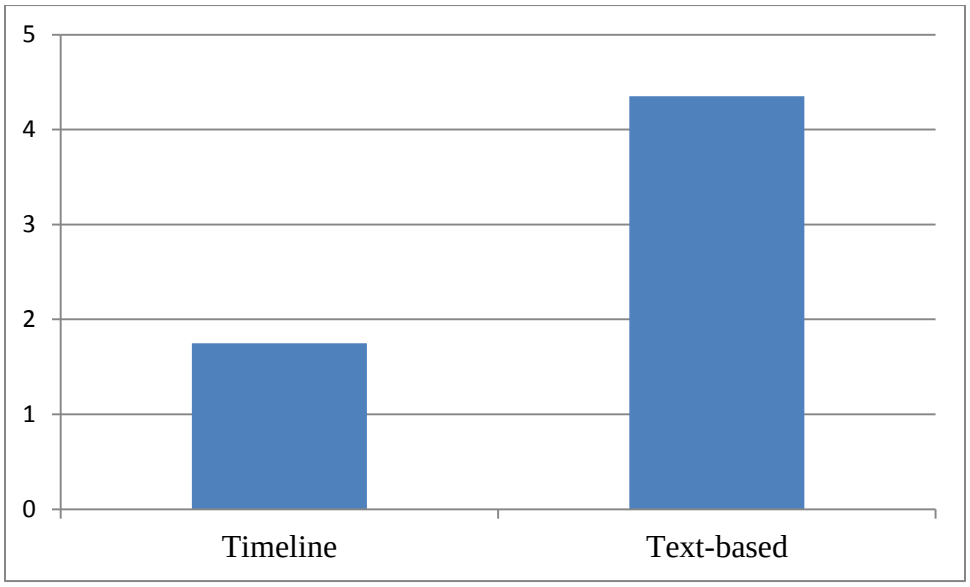


Figure 5.8 Mean values chart for the Likert scale

Most of the participants had ranked the timeline as the most convenient and intuitive to use.

5.4 Discussion

The results suggest that majority of the participants found timeline tool as the most intuitive and easy to use when compared to text based history in the Mozilla browser. The users expressed high satisfaction performing the required tasks using timeline tool. In 3 out of 4 tasks the users took less time to answer the questions using timeline tool. The user completed task#3 faster with the text-based history tool. This happened as the users could not find the answer in the text-based history tool, so they skipped to the next task without continuing this task. The accuracy rate was high using with the timeline tool when compared to the text-based history. The mean values indicate that the users performed well while interacting with the timeline tool. The users ranked the timeline tool as the easiest to use. We asked the users to determine which feature was the most useful resource for interacting with the tool. Most of the participants liked the zooming feature and the way the Web pages are placed based on the time along with the tabs information.

While conducting the experiment, we observed that participants were looking for a Web page in the entire history list in a day rather than in an hourly-format while interacting with timeline tool which leads to a future research. Most of the participants replied that they are used to search the Web page in the entire history information rather than filtering to an hour.

Example of one of the participant's opinion after interacting with each tool:

Text-based history:

It was very difficult to find the Web page in the history especially when I have to activate the options such as visit date, visit count which are hidden in the options folder. I had to use lot of memory to find the Web page.

Timeline tool

It was very easy to navigate the Web page due to options that are shown everything in one place. I like the zooming option where the user can zoom up to milliseconds. I like the way the Web pages are placed according to the tabs information. This is really useful for today's browsing habits. I appreciate the work.

Chapter 6

Conclusions and Future Work

This chapter contains the summary of the thesis work, including suggestions for improvements to the timeline tool which can lead to a future research.

6.1 Conclusions

In this thesis work, we investigated the problem of visualizing Web history. We developed three prototypes and selected one based on user performance and preference. The timeline tool is an interactive visualization tool for the Web browsing history to aid navigating the Web pages easily and in a faster way. Our results suggest that the timeline tool helped users locate the Web pages faster, and the users also expressed a higher degree of satisfaction using the tool. The research started after conducting a literature review about the page visitation method in the Web browser history. Previous research findings suggest that the history list is barely used for finding the Web pages. We summarized the problems that are present in current browsers.

1. The browser history does not provide information about the amount of time spent on a particular Web page.
2. Modern browsers do not show the relationship existing between the Web pages.
3. They do not show the parallel browsing paradigm.
4. Current browsers lack graphical visualization concepts such as a zooming feature.

After several studies, we came up with a concept of breadcrumb navigation, which can show the path taken between Web pages in the history of a browser. By using this concept, we decided to develop a visualization tool which can address the existing problems in current browsers. After examining the benefits of visualization, three different prototypes were designed. There was an experiment conducted with three different paper prototypes, in which participants completed tasks on each prototype. The reason for testing these three designs was to know whether the user will be able to interact with the designs without prior knowledge. The best design was finalized based on the three dependent variables, which are the completion time, the accuracy and the Likert scale. One-way ANOVA (Analysis of Variance) was used for analyzing the results. The ANOVA results suggest that users preferred the timeline prototype as the most convenient and intuitive to use for displaying the Web pages. The timeline tool was implemented as a working prototype based on the suggestions from the users in the experiment 1, as well as our own ideas. After developing this tool, we decided to conduct

an experiment to compare the timeline tool with the existing text-based history in the Mozilla browser. In this experiment, the users completed tasks by interacting with each tool separately. Twenty participants were recruited from the Laurentian university, and they were divided into two groups. The t-test was used for analyzing the results in the experiment 2. The results were analyzed based on the three variables, which are the completion time, the accuracy and the Likert scale. The t-test results suggest that users performed well with the timeline tool when compared to text based history.

6.2 Future research

The conducted studies raised some improvements in the timeline tool to make it more intuitive to use, which can be implemented in the future research. One further improvement which can be added to the tool is the display option. The display option should include a linear view, which can display the Web pages from top to bottom. By adding this feature, the user can switch between the horizontal and the vertical view, using whichever is most convenient to use. Another such improvement would be to highlight the timings of the day that are browsed. While conducting the testing, we observed that most of the users are not viewing the browsing times for a particular day, which made the user to navigate the Web page slowly. This feature will help the user to navigate the Web page more efficiently.

A further challenge includes visual improvements, such as the representation of pages. For our tool, we used a thumbnail and the title to represent the Webpage in a rectangle box. What if the title was removed from the thumbnail, leaving only the thumbnail to represent the Webpage? All of these aspects provide avenues for future improvements to the timeline tool.

.

References

1. Cockburn, A., & McKenzie, B. (2001). What do web users do? An empirical analysis of web use. *International Journal of Human-Computer Studies, 54*(6), 903-922. doi: http://dx.doi.org/10.1006/ijhc.2001.0459
2. Achyut S Godbole and Atul Kahate. Web technologies: TCP/IP, architecture, and Java programming. Tata McGraw-Hill, New Delhi, 2008. ISBN 9780070142954 0070142955.
3. Group, M. M. (2011). "Internet Usage Statistics." World Internet Users and Population Stats Retrieved December 31, 2011, from: http://www.internetworldstats.com/stats.htm.
4. Anderson, J. 2009. Consumer Behavior Online: A 2009 Deep Dive. Forrester Market Report. Retrieved from www.forrester.com/go?docid=54327, September 2009.
5. Dan Tobias "Dan's Web tips" http://webtips.dantobias.com/brand-x/index.html.
6. Huang, J., & White, R. W. (2010, June). Parallel browsing behavior on the web. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia* (pp. 13-18). ACM.
7. Labaj, M., & Bieliková, M. (2012, November). Modeling parallel web browsing behavior for web-based educational systems. In *Emerging eLearning Technologies & Applications (ICETA), 2012 IEEE 10th International Conference on* (pp. 229-234). IEEE.
8. Browser statistics. http://www.w3schools.com/browsers/browsers stats.asp, December 2011. URL http://www.w3schools.com/browsers/browsers_stats.asp.
9. Apple Safari 5; http://www.apple.com/safari/

10. Milic-Frayling, N. et al. (2004). Smart Back: Supporting Users in Back Navigation. WWW2004, New York, USA. ACM.
11. Obendorf, H., Weinreich, H., Herder, E., & Mayer, M. (2007). *Web page revisitation revisited: implications of a long-term click-stream study of browser usage*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, USA.
12. WWW Surveying Team "GVU's 6th WWW user survey," Graphic Visualization and Usability Center (GVU), Georgia Institute of Technology, available at http://www.cc.gatech.edu/gvu/user_surveys/survey-10-1996/.
13. Aula, A., Jhaveri, N., K, M., #228, & ki. (2005). *Information search and re-access strategies of experienced web users*. Paper presented at the Proceedings of the 14th international conference on World Wide Web, Chiba, Japan.
14. Killam, B. (2001). *A Study of Three Browser History Mechanisms for Web Navigation*. Paper presented at the Proceedings of the Fifth International Conference on Information Visualisation.
15. Miura, M., Kunifuji, S., Sato, S., & Tanaka, J. (2005). Capturing Window Attributes for Extending Web Browsing History Records. In R. Khosla, R. Howlett & L. Jain (Eds.), *Knowledge-Based Intelligent Information and Engineering Systems* (Vol. 3681, pp. 418-424): Springer Berlin Heidelberg.
16. Ayers,E and Stasko,J. "*Using Graphic History in Browsing the World Wide Web*", The 4th InternationalWorldWideWeb Conference,www.w3.org/Conferences/WWW4/Program_Full.html, December 11-14, 1996.

17. Hightower, R. R., Ring, L. T., Helfman, J. I., Bederson, B. B., & Hollan, J. D. (1998). *Graphical multiscale Web histories: a study of padprints*. Paper presented at the Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems, Pittsburgh, Pennsylvania, USA.
18. Gandhi, R., Kumar, G., Bederson, B., & Shneiderman, B. (2000, 2000). *Domain name based visualization of Web histories in a zoomable user interface.* Paper presented at the Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop on.
19. Cockburn A., Greenberg S., McKenzie B., Jasonsmith M., Kaasten S., (1999). *WebView: A Graphical Aid for Revisiting Web Pages*. In Proceedings of the OZCHI'99 . Australian Conference on Human Computer Interaction, November 28-30, Wagga Australia.
20. Mayer, M. and Bederson, B.B. *Browsing Icons: A Task-Based Approach for a Visual Web History*. In HCIL-200119, CS-TR-4308, UMIACS-TR-2001, vo-lume 85, 2001.
21. Nagel, T., Ren & Sander. (2005). *HyperHistory*. Paper presented at the Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, Salzburg, Austria.
22. Kawase, R., Papadakis, G., & Herder, E. (2011). *Supporting revisitation with contextual suggestions*. Paper presented at the Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries, Ottawa, Ontario, Canada.
23. History Tree 1.2(2009).WWW.Firefox/mozilla/add-ons.com.
24. Yamaguchi, T., Hattori, H., Ito, T., & Shintani, T. (2004). *On a web browsing support system with 3d visualization*. Paper presented at the Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA.

25. Tauscher, L., & Greenberg, S. (1997, March). Revisitation patterns in world wide web navigation. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (pp. 399-406). ACM.
26. Kleek, M. V., Moore, B., Xu, C., & Karger, D. R. (2010). *Eyebrowse: real-time web activity sharing and visualization*. Paper presented at the CHI '10 Extended Abstracts on Human Factors in Computing Systems, Atlanta, Georgia, USA.
27. Khaksari, G. H. (2011). Direct Manipulation of Web Browsing History.*International Journal of Computer Science & Information Technology*, *3*(5).
28. Norman, D. A. (2002). *The design of everyday things*. Basic books.
29. Ware, C. (2000). Information Visualization, perception for design. Morgan Kaufmann Publishers,San Francisco, USA.
30. Laurel, B. (1997). Interface agents: Metaphors with character. *Human Values and the design of Computer Technology*, 207-219.
31. http://www.forensicswiki.org/wiki/Mozilla_Firefox_3_History_File_Format#Gathering_browser_history
32. Walker, M., Takayama, L., & Landay, J. A. (2002, September). High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 46, No. 5, pp. 661-665). SAGE Publications.
33. Snyder, C. (2003). *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann.

34. Bertin, J. (1999). Graphics and graphic information processing. In K. C. Stuart, D. M. Jock & S. Ben (Eds.), *Readings in information visualization* (pp. 62-65): Morgan Kaufmann Publishers Inc.

35. Tufte, E. R., & Graves-Morris, P. R. (1983). *The visual display of quantitative information* (Vol. 2). Cheshire, CT: Graphics press.

36. Röber, N., Möller, T., Celler, A. M., & Strothotte, T. (2000, May). Multidimensional analysis and visualization software for dynamic SPECT. In*JOURNAL OF NUCLEAR MEDICINE* (Vol. 41, No. 5, pp. 191P-191P). 1850 SAMUEL MORSE DR, RESTON, VA 20190-5316 USA: SOC NUCLEAR MEDICINE INC.

37. Hamilton, A. (2000). Metaphor in theory and practice: the influence of metaphors on expectations. *ACM J. Comput. Doc., 24*(4), 237-253. doi: 10.1145/353927.353935

38. Sachinopoulou, A. (2001). Multidimensional visualization. Technical report, Espoo 2001.Technical Research Centre of Finland, VTT Tiedotteita, Meddelanden, Research Notes 2114.http://www.inf.vtt.fi/pdf/tiedotteet/2001/T2114.pdf.

39. Shneiderman, B. (1996). *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. Paper presented at the Proceedings of the 1996 IEEE Symposium on Visual Languages.

40. Robb, R. A., & Hanson, D. P. (1996). 10 The ANALYZE Software System for Visualization and Analysis in Surgery Simulation. *Computer Integrated Surgery: Technology and Clinical Applications*, 175

Appendix A

Approved Ethics Certificate

APPROVAL FOR CONDUCTING RESEARCH INVOLVING HUMAN SUBJECTS

Research Ethics Board – Laurentian University

This letter confirms that the research project identified below has successfully passed the ethics review by the Laurentian University Research Ethics Board (REB). Your ethics approval date, other milestone dates, and any special conditions for your project are indicated below.

TYPE OF APPROVAL / New X / Modifications to project / Time extension

Name of Principal Investigator and school/department	Makkena Anudeep
Title of Project	**The design and evaluation of novel prototypes to visualize Web browsing history.**
REB file number	2012-10-06
Date of original approval of project	February 21, 2013
Date of approval of project modifications or extension *(if applicable)*	
Final/Interim report due on	February 21, 2014
Conditions placed on project	Final report due on February 21, 2014 Do not use a personal phone number on your recruitment poster.

During the course of your research, no deviations from, or changes to, the protocol, recruitment or consent forms may be initiated without prior written approval from the REB. If you wish to modify your research project, please refer to the Research Ethics website to complete the appropriate REB form.

All projects must submit a report to REB at least once per year. If involvement with human participants continues for longer than one year (e.g. you have not completed the objectives of

the study and have not yet terminated contact with the participants, except for feedback of final results to participants), you must request an extension using the appropriate REB form.

In all cases, please ensure that your research complies with Tri-Council Policy Statement (TCPS). Also please quote your REB file number on all future correspondence with the REB office.

Congratulations and best of luck in conducting your research.

Susan James, Acting chair

Laurentian University Research Ethics Board

Consent form used for selecting the designs

CONSENT TO PARTICIPATE IN RESEARCH

You are asked to participate in a research study conducted by Anudeep Makkena under the supervision of Dr. Ratvinder Singh Grewal from the department of computer science at Laurentian university.

If you have any questions or concerns about the research, please feel free to contact

Anudeep Makkena : Ax_makkena@laurentian.ca

Dr. Ratvinder Singh Grewal: rgrewal@cs.laurentian.ca

Phone .no: 705-675-1151 ext : 2351

PURPOSE OF THE STUDY

To know the user's opinion regarding the three different designs for building a visualization tool.

PROCEDURES

If you volunteer to participate in this study, we would ask you to do the following things:

The participants have to look at three different designs individually and answer three questions for each design. The time will be noted for answering the questions using a stopwatch. The participants can indicate the ease of use for each design using a Likert scale.

POTENTIAL RISKS AND DISCOMFORTS

There won't be any risks or discomforts if you are regular computer user. The user can withdraw from the study if there is a feeling of discomfort or stressed. You can rejoin in the study if you are interested.

POTENTIAL BENEFITS TO PARTICIPANTS AND/OR TO SOCIETY

Participants will get to know the insights of the current research, and there are no benefits.

PAYMENT FOR PARTICIPATION

There will not be any payment for the participation.

CONFIDENTIALITY

Every effort will be made to ensure confidentiality of any identifying information that is obtained in connection with this study. Any individual information obtained during the study will not be discussed with anyone other than the supervisor if required. The data thus obtained will be stored in a safe locker and will be deleted if you are not interested to participate in the study.

DATA STORAGE

The data will be kept until the project finishes. The data will be stored in USB or a Hard Disk which has a password to access. The USB will be kept in a safe locker. This password and the locker can be accessed only by Anudeep Makkena. Once the project is completed, the data will be deleted.

PARTICIPATION AND WITHDRAWAL

You can choose whether to be in this study or not. If you volunteer to be in this study, you may withdraw at any time without consequences of any kind. You may exercise the option of removing your data from the study.

RESULTS

Please fill your contact information if you are interested in knowing the results of your participation. I will contact you after the results have been finalized.

ISSUES / ETHICAL CONCERNS

The below is the address to contact for any issues or ethical concerns on the research.

(**Research Ethics Officer**, **telephone # 705-675-1151 ext 2436)**

SIGNATURE OF RESEARCH PARTICIPANT

I have read the information provided for the study “Reducing the searching time of revisited web pages in the history of a browser using visualization techniques” as described herein. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

_____

Name of Participant (please print)

_____ _____

Signature of Participant Date

Consent form for implementing the final design

CONSENT TO PARTICIPATE IN RESEARCH

You are asked to participate in a research study conducted by Anudeep Makkena under the supervision of Dr. Ratvinder Singh Grewal from the department of computer science at Laurentian university.

If you have any questions or concerns about the research, please feel free to contact

AnudeepMakkena : Ax_makkena@laurentian.ca , phone .no: 705-507-9545

Dr. Ratvinder Singh Grewal: rgrewal@cs.laurentian.ca

Phone .no: 705-675-1151 ext : 2351

PURPOSE OF THE STUDY

To compare the existing browser's history tool with the visualization tool which will show the history logs.

PROCEDURES

If you volunteer to participate in this study, we would ask you to do the following things:

The user will be asked to see the history logs using the new visualization tool and also with the existing browser history and answer four questions for each tool. The time will be noted for answering the questions using a stopwatch. The participants can indicate the ease of use for each design using a Likert scale. It may not exceed more than 15 minutes for a session. The location of the procedure will be done in a well-lit office room setting with quiet atmosphere.

POTENTIAL RISKS AND DISCOMFORTS

There won't be any risks or discomforts if you are regular computer user. The user can withdraw from the study if there is a feeling of discomfort or stressed. You can rejoin in the study if you are interested.

POTENTIAL BENEFITS TO PARTICIPANTS AND/OR TO SOCIETY

Participants will get to know the insights of the current research, and there are no benefits.

PAYMENT FOR PARTICIPATION
There will not be any payment for the participation.

CONFIDENTIALITY

Every effort will be made to ensure confidentiality of any identifying information that is obtained in connection with this study. Any individual information obtained during the study will not be discussed with anyone other than the supervisor if required. The data thus obtained will be stored in a safe locker and will be deleted if you are not interested to participate in the study.

DATA STORAGE

The data will be kept until the project finishes. The data will be stored in USB or a Hard Disk which has a password to access. The USB will be kept in a safe locker. This password and the locker can be accessed only by Anudeep Makkena. Once the project is completed, the data will be deleted.

PARTICIPATION AND WITHDRAWAL

You can choose whether to be in this study or not. If you volunteer to be in this study, you may withdraw at any time without consequences of any kind. You may exercise the option of removing your data from the study. You may also refuse to answer any questions you don't want to answer and still remain in the study.

RESULTS

Please fill your contact information if you are interested in knowing the results of your participation. I will contact you after the results have been finalized.

ISSUES / ETHICAL CONCERNS

The below is the address to contact for any issues or ethical concerns on the research.

(**Research Ethics Officer**, **telephone # 705-675-1151 ext 2436)**

SIGNATURE OF RESEARCH PARTICIPANT
I have read the information provided for the study "Reducing the searching time of revisited web pages in the history of a browser using visualization techniques" as described herein. My

questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

_____

Name of Participant (please print)

_____ _____

Signature of Participant Date

Appendix B

Tasks for the Design Selection

Task# 1

How many days are browsed during the week?

Task# 2

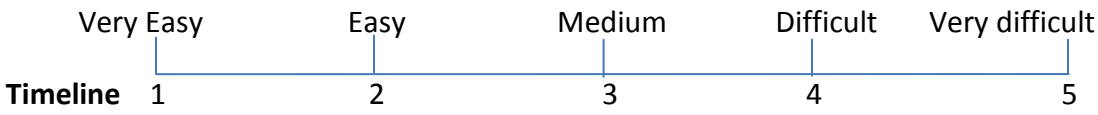
How many times did the user browse on Thursday?

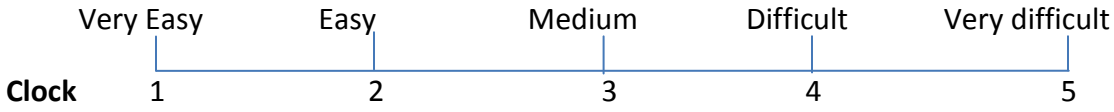
Task# 3

On Thursday, in which tab did the user visit the vacations web page and also find the start time they visited vacations web page?

Tab number	Start time

Indicate how easy/difficult you found each design?





Tasks for the final design

Task# 1

How many days browsing occurred during the week in the history logs?

Task# 2

On Tuesday, which web page was visited between 13:00-14:00?

Task# 3

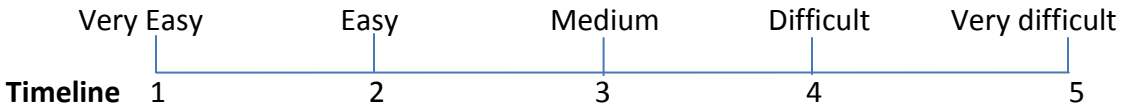
On Sunday, in which tab did the user visit the BBC News-US and Canada web page and also find the start time they visited BBC News-US and Canada web page?

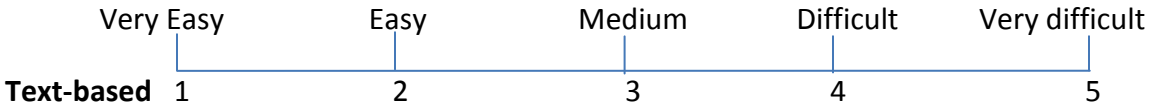
Tab number	Start time

Task# 4

On Friday, find the amount of time spent on the Canada-Wikipedia web page?

Indicate how easy/difficult you found each task using these tools?





Appendix C

T-test results for the Timeline design

Task#1

Group Statistics

	Tools	N	Mean	Std. Deviation	Std. Error Mean
Time	Timeline	20	25.9860	13.31310	2.97690
	Browser history	19	72.3868	37.93050	8.70185

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	13.075	.001	-5.150	37	.000	-46.40084	9.00970	-64.65624	-28.14545
	Equal variances not assumed			-5.045	22.172	.000	-46.40084	9.19697	-65.46561	-27.33608

Task#2

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	6.288	.017	-3.218	33	.003	-9.51569	2.95722	-15.53219	-3.49919
	Equal variances not assumed			-3.169	24.852	.004	-9.51569	3.00299	-15.70233	-3.32904

Group Statistics

Tools		N	Mean	Std. Deviation	Std. Error Mean
Time	Timeline	18	21.1567	6.09676	1.43702
	Browser History	17	30.6724	10.87197	2.63684

Task# 3

Group Statistics

Tools		N	Mean	Std. Deviation	Std. Error Mean
Time	Timeline	20	85.5420	32.31007	7.22475
	Browser history	20	81.2600	32.72904	7.31843

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	.001	.979	.416	38	.679	4.28200	10.28380	-16.53646	25.10046
	Equal variances not assumed			.416	37.994	.679	4.28200	10.28380	-16.53658	25.10058

Task#4

Group Statistics

Tools		N	Mean	Std. Deviation	Std. Error Mean
Time	Timeline	20	26.7705	6.72281	1.50327
	Browser history	18	58.4222	27.55591	6.49499

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Time	Equal variances assumed	18.789	.000	-4.982	36	.000	-31.65172	6.35351	-44.53724	-18.76620
	Equal variances not assumed			-4.748	18.822	.000	-31.65172	6.66669	-45.61421	-17.68924

Accuracy results

Group Statistics

Tools		N	Mean	Std. Deviation	Std. Error Mean
Accuracy	Timeline	18	3.7778	.39191	.09237
	Browser history	16	2.2500	.36515	.09129

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Accuracy	Equal variances assumed	.016	.899	11.714	32	.000	1.52778	.13043	1.26210	1.79345
	Equal variances not assumed			11.764	31.918	.000	1.52778	.12987	1.26321	1.79234

Appendix D

// connecting to database using JDBC driver

```
package DB;

/**
 *
 * @author User
 */
public class MyDBBean {
   public static java.sql.Connection getDBConnection() throws Exception
   {
      Class.forName("com.mysql.jdbc.Driver");
      java.sql.Connection mycon=java.sql.DriverManager.getConnection("jdbc:mysql://localhost:
3306/mozdb","root","");
      return mycon;

   }
}
```

Appendix E

// This code was used for getting all the sessions from session manager

```
package DAO;


import java.io.File;


public class GetAllSessionsFromSessionManager

{
    public static java.util.ArrayList getFiles(String folderofsessions)
   {

    File folder = new File(folderofsessions);
   File[] listOfFiles = folder.listFiles();
   java.util.ArrayList al=new java.util.ArrayList();


  for (int i = 0; i < listOfFiles.length; i++)

{
    if (listOfFiles[i].isFile())

{
      System.out.println("File " + listOfFiles[i].getName());
      al.add(listOfFiles[i].getName());
    }
   }
   return al;
   }
```

// **the below is the Java code**

sqliteDAO.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLiteDAO

{
   public static void main(String[] args)

 {
  Connection connection = null;
   ResultSet resultSet = null;
   Statement statement = null;

      try

 {
          Class.forName("org.sqlite.JDBC");
          connection = DriverManager
                .getConnection("jdbc:sqlite:C:\\t1\\places.sqlite");
          statement = connection.createStatement();

         // String q="SELECT time(v.visit_date/1000000,'unixepoch','localtime') as vist_date , p1.ur
l \"From_Url\", time(p2.last_visit_date/1000000,'unixepoch','localtime')as last_visit,p2.url \"To
_Url\", time((p2.last_visit_date-
v.visit_date)/1000000,'unixepoch')as duration     FROM moz_historyvisits v     INNER JOIN moz_
places p1 ON v.from_visit=p1.id     INNER JOIN moz_places p2 ON v.place_id=p2.id ";
```

```
String qdelete="drop view if exists Moz_view ";
int j= statement.executeUpdate(qdelete) ;
 if(j>0)

{
   System.out.println("drop ok");
}
java.sql.Statement   statement1 = connection.createStatement();

 String q="create view Moz_view as SELECT v.visit_date/1000000 as vist_date , p1.url as From_
Url, p1.last_visit_date/1000000 as last_visit,p1.url as To_Url, (p1.last_visit_datev.visit_date) as
duration , f.url as favicon , p1.id ,v.visit_date as visit_anudeep_date FROM moz_historyvisits v
INNER JOIN moz_places p1 ON v.place_id=p1.id     INNER JOIN moz_favicons f ON p1.favicon_id
= f.id;";
        statement1.executeUpdate(q) ;

         String qselect="select * from Moz_view";
        resultSet = statement
              .executeQuery(qselect);
         while (resultSet.next())

{
            System.out.println("Visit_date:"
                  + resultSet.getString(1));
            System.out.println("From_Url"+resultSet.getString(2));
            System.out.println("last_visit"+resultSet.getString(3));
            System.out.println("To_URL"+resultSet.getString(4));
            System.out.println("Duration"+resultSet.getInt(5));
            System.out.println("*****************************************");
         }
```

```
    } catch (Exception e) {
        System.out.println(e);
    } finally {
      /*  try {
            resultSet.close();
            statement.close();
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }  */
    }
}

 static public String bootstrarpPlaces()
{

Connection connection = null;
    ResultSet resultSet = null;
    Statement statement = null;

    try {
        Class.forName("org.sqlite.JDBC");
        connection = DriverManager
                .getConnection("jdbc:sqlite:C:\\t1\\places.sqlite");
        statement = connection.createStatement();


      String qdelete="drop view if exists Moz_view ";
      int j= statement.executeUpdate(qdelete) ;
```

```
        if(j>0)

        {
           System.out.println("drop ok");
        }
        java.sql.Statement   statement1 = connection.createStatement();

           String q="create view Moz_view as SELECT v.visit_date/1000000 as vist_date , p1.url as
From_Url, p1.last_visit_date/1000000 as last_visit,p1.url as To_Url, (p1.last_visit_date-
v.visit_date) as duration , f.url as favicon , p1.id ,v.visit_date as visit_anudeep_date FROM moz_
historyvisits v    INNER JOIN moz_places p1 ON v.place_id=p1.id     INNER JOIN moz_favicons f O
N p1.favicon_id = f.id;";
        statement1.executeUpdate(q) ;

         String qselect="select * from Moz_view";
        resultSet = statement
              .executeQuery(qselect);
         while (resultSet.next())

{
           System.out.println("Visit_date:"
                 + resultSet.getString(1));
           System.out.println("From_Url"+resultSet.getString(2));
           System.out.println("last_visit"+resultSet.getString(3));
           System.out.println("To_URL"+resultSet.getString(4));
           System.out.println("Duration"+resultSet.getInt(5));
           System.out.println("****************************************");
         }
      } catch (Exception e)
```

```
 {
         System.out.println(e);
      }
finally {

      }

      return "Generated";
   }
}
```

//parsing the JSON data

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Iterator;
import org.apache.commons.io.FileUtils;
import org.apache.commons.io.IOUtils;
import org.apache.commons.io.LineIterator;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class Test {
    public static void main(String[] args) {

   JSONParser parser = new JSONParser();
```

```
  try {

 String url=null;
 Object obj = parser.parse(new FileReader("d:\\sessionstore.js"));
JSONObject jsonObject = (JSONObject) obj;

  // loop array
     JSONArray msg = (JSONArray) jsonObject.get("windows");
     Iterator iterator = msg.iterator();
          int windowcount=0;
     while (iterator.hasNext())

{
               windowcount++;
               JSONObject jsonObject1 = (JSONObject)(Object)iterator.next();
               JSONArray t = (JSONArray) jsonObject1.get("tabs");
          Iterator titerator = t.iterator();
       int tabcount=0;
            while(titerator.hasNext())
            {
             JSONObject tjsonObject1 = (JSONObject)(Object)titerator.next();

              tabcount++;

           JSONArray e = (JSONArray) tjsonObject1.get("entries");
          Iterator eiterator = e.iterator();
                int sitescountintab=0;
            while(eiterator.hasNext())
            {
              sitescountintab++;
               JSONObject ejsonObject1 = (JSONObject)(Object)eiterator.next();
              url =(String)ejsonObject1.get("url");
              String refererfrom=(String)ejsonObject1.get("referrer");
              if(refererfrom==null)
              {
                 System.out.println("From"+url);
              }
              else
              {
```

```
                System.out.println("From"+refererfrom);
                }

                 System.out.println(url);

              }
System.out.println("Window :----"+windowcount+"-
Total Sites Opened In "+ tabcount +"Tab are "+      sitescountintab);

              }

       }

          System.out.println("Closed Tabs in  Windows");

          Windows.ClosedTabsInWondows.main123("d:\\sessionstore.js");

          System.out.println("Closed Tabs in Closed Windows123123132");

              NewClass.main123("d:\\sessionstore.js");

              System.out.println("Tabs in Closed Windows AAAAAAAAAAAAAAAA from Sesion Man
ger");
              ClosedWindows.TabsInClosedWindows.main("d:\\sessionstore.js");
              SessionManager.TabsInPreviousSessions.main("d:\\t\\backup-2.session");
                }

                 catch (FileNotFoundException e)

     {
       e.printStackTrace();
   }

catch (IOException e)

 {
       e.printStackTrace();
   }

catch (ParseException e)
```

```
 {
      e.printStackTrace();
   }

    }

}
```

// Getdaylinks.java

```
package DAO;

/**
 *
 *
 */
public class GetDayLinks {
    static public String getDayLinks(String d)
   {
   if(d.equals("no"))
   {
   return "<a style='margin: 5px; padding-left: 2px;padding-right: 2px;' class='metro three-
d earnmoney-link' href='day.jsp?day=1'>Monday | <a style='margin: 5px; padding-
left: 2px;padding-right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=2'>Tuesday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=3'>Wednesday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=4'>Thursday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=5'>Friday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=6'>Saturtday | <a style='margin: 5px; padding-left: 2px;padding-
```

```
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=0'>Sunday"</a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=0'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=6'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=5'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=4'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=3'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-
link' href='day.jsp?day=2'></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;' class='metro three-d earnmoney-link' href='day.jsp?day=1'>;
   }
   else
   {
      String mon="";
       String tue="";
        String wed="";
         String thu="";
          String fri="";
           String sat="";
            String sun="";

      if(d.equals("1"))
      {mon="background-color:#53A7EA;";}
       if(d.equals("2"))
      {tue="background-color:#53A7EA;";}
        if(d.equals("3"))
      {wed="background-color:#53A7EA;";}
         if(d.equals("4"))
      {thu="background-color:#53A7EA;";}
           if(d.equals("5"))
      {fri="background-color:#53A7EA;";}
            if(d.equals("6"))
      {sat="background-color:#53A7EA;";}
```

```
        if(d.equals("0"))
      {sun="background-color:#53A7EA;";}

      return "<a style='margin: 5px; padding-left: 2px;padding-right: 2px;"<
font="">+mon+"' class='metro three-d earnmoney-
link' href='day.jsp?day=1'>Monday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                    font="">+tue+"' class='metro three-d earnmoney-
link' href='day.jsp?day=2'>Tuesday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                  font="">+wed+"' class='metro three-d earnmoney-
link' href='day.jsp?day=3'>Wednesday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                    font="">+thu+"' class='metro three-d earnmoney-
link' href='day.jsp?day=4'>Thursday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                     font="">+fri+"' class='metro three-d earnmoney-
link' href='day.jsp?day=5'>Friday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                    font="">+sat+"' class='metro three-d earnmoney-
link' href='day.jsp?day=6'>Saturtday | <a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<                                  font="">+sun+"' class='metro three-d earnmoney-
link' href='day.jsp?day=0'>Sunday";


   }}
}

\

</a style='margin: 5px; padding-left: 2px;padding-right: 2px;"<></a style='margin: 5px; padding-
left: 2px;padding-right: 2px;"<></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<></a style='margin: 5px; padding-left: 2px;padding-
right: 2px;"<></a style='margin: 5px; padding-left: 2px;padding-right: 2px;"<>
```

// getduration.java

```
package DAO;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Calendar;

/**
 *
 *
 */
public class GetDuration

{
   public static int getDuration(String fromurl,String fromurl1,int tabcount,int windowcount,int
sessionnumber) throws Exception
   {
    Connection connection = null;
      ResultSet resultSet = null;
      Statement statement = null;
       Class.forName("org.sqlite.JDBC");
         connection = DriverManager  .getConnection("jdbc:sqlite:C:\\t1\\places.sqlite");
          statement = connection.createStatement();
          String qselect="select * from Moz_view where From_Url='"+fromurl+"'";
         resultSet = statement  .executeQuery(qselect);
         Integer duration=0;
         java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
         java.sql.PreparedStatement pstmt= mycon.prepareStatement("insert into durationtab val
ues(?,?,?,?,?,?,?,?,?,?,?)");
        // java.util.ArrayList days=new java.util.ArrayList();
        // days.add("Monday");

          while (resultSet.next()) {
            duration =resultSet.getInt(5);
```

```
            String visitdate=resultSet.getString(1);
            // The follwoing is the code for getting day from the date

       long timestamp= Long.parseLong(visitdate);//  today.getTime()/1000;
   System.out.println(timestamp);
   java.util.Date time=new java.util.Date((long)timestamp*1000);
   System.out.print(time);
   Calendar cal = Calendar.getInstance();
 cal.setTime(time);
  int d= cal.get(Calendar.DAY_OF_WEEK);
        pstmt.setString(1, visitdate);
        pstmt.setString(2, resultSet.getString(2));
        pstmt.setString(3, resultSet.getString(3));
        pstmt.setString(4, resultSet.getString(4));
        pstmt.setInt(5, tabcount);
        pstmt.setInt(6, windowcount);
        pstmt.setString(7, duration.toString());
        pstmt.setInt(8, sessionnumber);
        pstmt.setInt(9, (d-1));
        pstmt.setString(10, resultSet.getString(6));
        pstmt.setString(11, resultSet.getString(9));
        pstmt.executeUpdate();


          //  System.out.println("Visit_date:"+ resultSet.getString(1));
          //  System.out.println("From_Url"+resultSet.getString(2));
           // System.out.println("last_visit"+resultSet.getString(3));
           // System.out.println("To_URL"+resultSet.getString(4));
           // System.out.println("Duration"+duration);
           // System.out.println("****************************************");
        }

        return duration;

         }
   }
```

// Gethour.java

```
package DAO;

import java.util.Calendar;

/**
 *
 *
 */
public class GetHour

{

    static public int gethour(long timestamp)
    {

    java.util.Date time=new java.util.Date(timestamp);
    System.out.print(time);
   Calendar cal = Calendar.getInstance();
  cal.setTime(time);
   int d= cal.get(Calendar.DAY_OF_WEEK);
   System.out.println(d);
   int h=cal.get(Calendar.HOUR_OF_DAY);
   return h;

    }

}
```

// Getwindowlinks.java

```
package DAO;

public class GetWindowLinks {
  static public String getWindowLinks() throws Exception
  {

       java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
  String q="select DISTINCT windownumber from durationtab ";
       java.sql.PreparedStatement pstmt=mycon.prepareStatement(q);
       java.sql.ResultSet rs= pstmt.executeQuery();
       String r="";
       while(rs.next())
       {
          int wn=rs.getInt("windownumber");
       r=r+"<a style='margin: 5px; padding-left: 2px;padding-right: 2px;' class='metroora three-
d_orange earnmoney-
link' href='windows.jsp?windownumber="<font="">+wn+"'>Window:"+wn+" | ";

       }
       return r;
  }

}</a style='margin: 5px; padding-left: 2px;padding-right: 2px;' class='metroora three-
d_orange earnmoney-link' href='windows.jsp?windownumber="<>
```

// ClosedTabsInClosedWindows.java

```
package ClosedWindows;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Iterator;
```

```
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class ClosedTabsInClosedWindows

{
   public static void main123(String sessionfile)

{

  JSONParser parser = new JSONParser();

  try {
        System.out.println("List of closed tabs in closed windows");

 String url=null;
 Object obj = parser.parse(new FileReader("C:\\t1\\sessionstore.js"));
JSONObject jsonObject = (JSONObject) obj;
// loop array
     JSONArray msg = (JSONArray) jsonObject.get("_closedWindows");
     Iterator iterator = msg.iterator();
          int windowcount=0;
     while (iterator.hasNext())

{
              windowcount++;
              JSONObject jsonObject1 = (JSONObject)(Object)iterator.next();
             JSONArray t = (JSONArray) jsonObject1.get("_closedTabs");
       Iterator titerator = t.iterator();
       int tabcount=0;
            while(titerator.hasNext())
            {
             JSONObject tjsonObject1 = (JSONObject)(Object)titerator.next();

              tabcount++;

           // JSONArray e = (JSONArray) tjsonObject1.get("state");
```

```
  // Iterator eiterator = e.iterator();
        int sitescountintab=0;
//    while(eiterator.hasNext())
   {
     sitescountintab++;
     JSONObject ejsonObject1 = (JSONObject)(Object)tjsonObject1.get("state");
     JSONArray ee = (JSONArray) ejsonObject1.get("entries");
     Iterator eiterator123 = ee.iterator();
     int sitescountintab123=0;
   while(eiterator123.hasNext())


   {

     JSONObject ejsonObject1123 = (JSONObject)(Object)eiterator123.next();
     url =(String)ejsonObject1123.get("url");
     String refererfrom=(String)ejsonObject1123.get("referrer");
     if(refererfrom==null)
     {
        System.out.println("From"+url);
     }
     else
     {
     System.out.println("From"+refererfrom);
     }
      System.out.println(url);
      System.out.println("Favicon:" + favicon.Favicons.getFavicon(url));

   }



   }

      System.out.println("Window"+windowcount+"Total Sites Opened In "+ tabcount
  +"Tab are "+     sitescountintab);

   }
```

```
      }

   } catch (Exception e) {
      e.printStackTrace();
   }

    }

}
```

// Tabs in ClosedWindows.java

```
package ClosedWindows;import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Iterator;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class TabsInClosedWindows

 {
    public static void main(String file)

 {
 JSONParser parser = new JSONParser();
  try

{

   System.out.println("Tabs In Closed Windows");
   String url=null;
  Object obj = parser.parse(new FileReader(file));
 JSONObject jsonObject = (JSONObject) obj;
 // loop array
```

```
JSONArray msg = (JSONArray) jsonObject.get("_closedWindows");
Iterator iterator = msg.iterator();
 int windowcount=0;
 while (iterator.hasNext())

{
  windowcount++;
  JSONObject jsonObject1 = (JSONObject)(Object)iterator.next();
  JSONArray t = (JSONArray) jsonObject1.get("tabs");
  Iterator titerator = t.iterator();
  int tabcount=0;
  while(titerator.hasNext()

   {
     JSONObject tjsonObject1 = (JSONObject)(Object)titerator.next();
     tabcount++;

    JSONArray e = (JSONArray) tjsonObject1.get("entries");
    Iterator eiterator = e.iterator();
     int sitescountintab=0;
     while(eiterator.hasNext())
          {
            JSONObject ejsonObject1 = (JSONObject)(Object)eiterator.next();
            url =(String)ejsonObject1.get("url");
            String refererfrom=(String)ejsonObject1.get("referrer");
            if(refererfrom==null)
            {
               System.out.println("From"+url);
            }
            else
            {
            System.out.println("From"+refererfrom);
            }

             System.out.println(url);
               System.out.println("Favicon:" + favicon.Favicons.getFavicon(url));

          }
          System.out.println("Window "+windowcount+"Total Sites Opened In "+ tabcount +
```

```
"Tab are "+     sitescountintab);

            }

      }

   }
catch (Exception e)

 {
      e.printStackTrace();
   }

    }

}
```

// t.java

```
public class T {
   public static void main(String[] args) throws Exception

 {
      System.out.println("*****************************************");
      System.out.println("Tabs in last Window");
       System.out.println("*****************************************");
      Windows.Test.main("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Closed Tabs in last Window");
      System.out.println("*****************************************");
      Windows.ClosedTabsInWondows.main123("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Tabs in Closed Window");
      System.out.println("*****************************************");
      ClosedWindows.TabsInClosedWindows.main("C:\\t1\\sessionstore.js");
```

```
    System.out.println("*******************************************");
    System.out.println("Closed Tabs in Closed Window");
    System.out.println("*******************************************");
    ClosedWindows.ClosedTabsInClosedWindows.main123("C:\\t1\\sessionstore.js");
    System.out.println("*******************************************");
    System.out.println("Tabs in Previous Sessions from Session Mangaer");
    System.out.println("*******************************************");
    SessionManager.TabsInPreviousSessions.main("C:\\t\\backup-2.session");
     java.util.ArrayList listofsessionfiles= DAO.GetAllSessionsFromSessionManager.getFiles("C:\\
t");
     java.util.Iterator itlistofsessionfiels=listofsessionfiles.iterator();
    int sessionfileidcount=0;
    while(itlistofsessionfiels.hasNext())

    {
       sessionfileidcount++;
       String file= itlistofsessionfiels.next().toString();
       System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
       System.out.println("For Session File NUmber "+sessionfileidcount+"From File :"+file);
       System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
       String path= "C:\\t\\"+file;
      System.out.println("*******************************************");
          System.out.println("Tabs in Previous Sessions from Session Mangaer &&&&&&&& ses
sion file number"+sessionfileidcount);
      System.out.println("*******************************************");
      SessionManager.TabsInPreviousSessions.main(path,sessionfileidcount);
      System.out.println("*******************************************");
    System.out.println("Closed Tabs in Previous Sessions from Session Mangaer &&&&&&&& s
ession file number"+sessionfileidcount);
    System.out.println("*******************************************");
    SessionManager.ClosedTabsInWondows.main123(path,sessionfileidcount);
    System.out.println("*******************************************");
    System.out.println(" Tabs in Closed Windows for Previous Sessions from Session Mangaer
&&&&&&&& session file number"+sessionfileidcount);
   System.out.println("*******************************************");
   SessionManager.TabsInClosedWindows.main(path,sessionfileidcount);
   System.out.println("*******************************************");
    System.out.println(" Tabs in Closed Windows for Previous Sessions from Session Mangaer
```

```
&&&&&&&& session file number"+sessionfileidcount);
        System.out.println("*****************************************");
       SessionManager.ClosedTabsInClosedWindows.main123(path,sessionfileidcount);

      }

   }

  static public String bootstrap1() throws Exception
  {

   System.out.println("*****************************************");
    System.out.println("Tabs in last Window");
    System.out.println("*****************************************");
     Windows.Test.main("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Closed Tabs in last Window");
      System.out.println("*****************************************");
      Windows.ClosedTabsInWondows.main123("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Tabs in Closed Window");
      System.out.println("*****************************************");
      ClosedWindows.TabsInClosedWindows.main("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Closed Tabs in Closed Window");
      System.out.println("*****************************************");

      ClosedWindows.ClosedTabsInClosedWindows.main123("C:\\t1\\sessionstore.js");
      System.out.println("*****************************************");
      System.out.println("Tabs in Previous Sessions from Session Mangaer");
      System.out.println("*****************************************");
      SessionManager.TabsInPreviousSessions.main("C:\\t\\backup-2.session");
      java.util.ArrayList listofsessionfiles= DAO.GetAllSessionsFromSessionManager.getFiles("C:\\
t");
      java.util.Iterator itlistofsessionfiels=listofsessionfiles.iterator();
       int sessionfileidcount=0;
      while(itlistofsessionfiels.hasNext())
```

```
        {
          sessionfileidcount++;
           String file= itlistofsessionfiels.next().toString();
           System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
           System.out.println("For Session File NUmber "+sessionfileidcount+"From File :"+file);
           System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$");
           String path= "C:\\t\\"+file;
           System.out.println("*****************************************");
            System.out.println("Tabs in Previous Sessions from Session Mangaer &&&&&&&& sessio
n file number"+sessionfileidcount);
           System.out.println("*****************************************");
           SessionManager.TabsInPreviousSessions.main(path,sessionfileidcount);
            System.out.println("*****************************************");
          System.out.println("Closed Tabs in Previous Sessions from Session Mangaer &&&&&&&&
session file number"+sessionfileidcount);
         System.out.println("*****************************************");
        SessionManager.ClosedTabsInWondows.main123(path,sessionfileidcount);
        System.out.println("*****************************************");
        System.out.println(" Tabs in Closed Windows for Previous Sessions from Session Mangaer
&&&&&&&& session file number"+sessionfileidcount);
        System.out.println("*****************************************");
         SessionManager.TabsInClosedWindows.main(path,sessionfileidcount);
         System.out.println("*****************************************");
        System.out.println(" Tabs in Closed Windows for Previous Sessions from Session Mangaer
&&&&&&&& session file number"+sessionfileidcount);
        System.out.println("*****************************************");
        SessionManager.ClosedTabsInClosedWindows.main123(path,sessionfileidcount);


        }

        return "Loaded";

    }
}
```

// datetime.java

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;


/**
 *
 * Anudeep
 */
public class NewClass1 {
   public static void main(String[] args) throws ParseException

 {
   long timestamp= (long)(1382961595);//  today.getTime()/1000;
   System.out.println(timestamp);
   java.util.Date time=new java.util.Date((long)timestamp*1000);
   System.out.print(time);
   Calendar cal = Calendar.getInstance();
  cal.setTime(time);
   int d= cal.get(Calendar.DAY_OF_WEEK);
   System.out.println(d);
   System.out.println(cal.get(Calendar.HOUR_OF_DAY));
   }
```

Appendix F

// for generating the Timeline

Day.jsp

```
<%@page import="java.net.URL"%>
 <script type="text/javascript" src="http://www.google.com/jsapi">
  <script src="js/jquery-1.8.3.min.js"</script src=>
  <script type="text/javascript" src="timeline.js">
   <link rel="stylesheet" media="screen" href="buttons.css">
    <link rel="stylesheet" type="text/css" href="timeline.css">

<style type="text/css">
        body {
           color: #4D4D4D;
           font: 10pt arial;
        }
     <script type="text/javascript">
        var timeline = null;
        var data = null;
        var order = 1;
        var truck = 1;

google.load("visualization", "1");
 function getSelectedRow() {
  var row = undefined;
  var sel = timeline.getSelection();
  if (sel.length)

{
   if (sel[0].row != undefined)

{
   row = sel[0].row;
            }
        }
        return row;
     }
```

```
        // Set callback to run when API is loaded
        google.setOnLoadCallback(drawVisualization);

        // Called when the Visualization API is loaded.
  function drawVisualization()
{

          // Instantiate our timeline object.
   timeline = new links.Timeline(document.getElementById('mytimeline'));
   google.visualization.events.addListener(timeline, 'select', onselect);
          // Create and populate a data table.
   data = new google.visualization.DataTable();
   data.addColumn('datetime', 'start');
   data.addColumn('datetime', 'end');
   data.addColumn('string', 'content');
   data.addColumn('string', 'group');

    data.addRows([
          <%
             java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
              String day= request.getParameter("day");
              int daytoget=Integer.parseInt(day);
              session.setAttribute("day", day);
              String q="select * from durationtab where dayvisited=?";
              java.sql.PreparedStatement pstmt= mycon.prepareStatement(q);
              pstmt.setInt(1, daytoget);
              java.sql.ResultSet rs=pstmt.executeQuery();
              int i=0;
              while(rs.next())
            {
              i++;
              String fromtime=rs.getString(1);
              String gapduration=rs.getString(7);
             Float g= Integer.parseInt(gapduration)+(float)(Math.random()*10);
              String favicon= rs.getString(10);
               String url1=rs.getString(2);
                String title=rs.getString(11);
```

```
                URL url = new URL(url1);
                url.getProtocol();
                url.getUserInfo();
                url.getAuthority();
                url.getHost();
                url.getPort();
                url.getPath(); // document part is contained within the path field
                url.getQuery();
                 url.getRef(); // gets #part
                int tabnumber=rs.getInt(5);
                int windownumber=rs.getInt(6);
                long time=(Long.parseLong(fromtime)*1000);
                long endtime=time+123450;
                int h= DAO.GetHour.gethour(time);
                %>


 [new Date(<%=time%>), '<div class="hi"><%=url1%>Duration:<%=g%>  Seconds</div </script t
ype=</style type=</link rel=</link rel=</script type=</script type= <img src="<%=favicon%>"</i
mg src=/><%=title%>',"TAB <%=tabnumber%>"],
      <%

   }

%>]);

            // specify options
            var options = {

            animate: false,
            eventMargin: 10,  // minimal margin between events
            eventMarginAxis: 5, // minimal margin between events and the axis
            showMajorLabels: true,
            cluster: true,
            axisOnTop: true,
            snapEvents: true,
            dragAreaWidth: 20
              //groupsWidth : "100px",
              //groupsOnRight: true
```

```
        };

        // Draw our timeline with the created data and options
        timeline.draw(data, options);

       google.visualization.events.addListener(timeline, 'edit',
        function() {
          //console.log('edit')
        }
      );

      google.visualization.events.addListener(timeline, 'change',
        function() {
          //console.log('change')
          //timeline.cancelChange();
        }
      );

        google.visualization.events.addListener(timeline, 'add',
        function() {
          //console.log('add')
          //timeline.cancelAdd();
        }
      );


      }
        // Make a callback function for the select item
    var onselect = function (event) {
      var row = getSelectedRow();

      if (row != undefined) {
        document.getElementById("info").innerHTML = "item " + data.getValue(row, 2) + " sel
ected
";
        // Note: you can retrieve the contents of the selected row with
        //      data.getValue(row, 2);
      }
      else {
```

```
            document.getElementById("info").innerHTML += "no item selected
";
        }
    };

        /**
         * Get URL parameter
         * http://www.netlobo.com/url_query_string_javascript.html
         */
        function gup( name ) {
            name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
            var regexS = "[\\?&]"+name+"=([^&#]*)";
            var regex = new RegExp( regexS );
            var results = regex.exec( window.location.href );
            if( results == null )
                return "";
            else
                return results[1];
        }

        var count = (Number(gup('count')) || 100);

        var d1=new Date();
        <body onresize="/*timeline.checkResize();*/">
    <%=DAO.GetDayLinks.getDayLinks(day)%>
    <%=DAO.GetHours.getHours_by_visit_status(daytoget)%>
    <%

        String dd="";
        if(daytoget==0)
                    {

        dd="Sunday";
        }
         if(daytoget==1)
                    {

        dd="Monday";
        }
```

```
 if(daytoget==2)
            {

dd="Tuesday";
}

 if(daytoget==3)
            {

dd="Wednesday";
}

 if(daytoget==4)
            {

dd="Thursday";
}
 if(daytoget==5)
            {

dd="Friday";
}
 if(daytoget==6)
            {

dd="Saturday";
}

%>
History on: <%=dd%> : All

<%
if(i>0)
    {
%>
  <div id="mytimeline">

  <div id="info">
```

```
      <%=DAO.GetWindowLinks.getWindowLinks()%>

      <%

}
   else
        {

         %>
No Result Found
<%

         }
%>
```

// hours.jsp

```
<%@page import="java.net.URL"%>


    <script type="text/javascript" src="http://www.google.com/jsapi">
      <script src="js/jquery-1.8.3.min.js"</script src=>
      <script type="text/javascript" src="timeline.js">
      <link rel="stylesheet" media="screen" href="buttons.css">

      <link rel="stylesheet" type="text/css" href="timeline.css">

      <style type="text/css">
         body {
            color: #4D4D4D;
            font: 10pt arial;
         }
```

```
    <script type="text/javascript">
      var timeline = null;
      var data = null;
      var order = 1;
      var truck = 1;

      google.load("visualization", "1");
function getSelectedRow() {
      var row = undefined;
      var sel = timeline.getSelection();
      if (sel.length) {
        if (sel[0].row != undefined) {
          row = sel[0].row;
        }
      }
      return row;
    }

      // Set callback to run when API is loaded
      google.setOnLoadCallback(drawVisualization);

      // Called when the Visualization API is loaded.
      function drawVisualization() {

        // Instantiate our timeline object.
        timeline = new links.Timeline(document.getElementById('mytimeline'));
  google.visualization.events.addListener(timeline, 'select', onselect);
        // Create and populate a data table.
        data = new google.visualization.DataTable();
        data.addColumn('datetime', 'start');
       data.addColumn('datetime', 'end');
        data.addColumn('string', 'content');
        data.addColumn('string', 'group');

        data.addRows([

     <%
          java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
```

```
String q="select * from durationtab";
java.sql.PreparedStatement pstmt= mycon.prepareStatement(q);
String h= request.getParameter("j");

int hh=Integer.parseInt(h);

session.setAttribute("hh", hh);
java.sql.ResultSet rs=pstmt.executeQuery();
int i=0;
while(rs.next())
                        {

String fromtime=rs.getString(1);
String gapduration=rs.getString(7);
Float g= Integer.parseInt(gapduration)+(float)(Math.random()*10);
String lastvisit=rs.getString(3);
String favicon= rs.getString(10);
 String url1=rs.getString(2);
  String title=rs.getString(11);
URL url = new URL(url1);
 url.getProtocol();
 url.getUserInfo();
url.getAuthority();
 url.getHost();
 url.getPort();
  url.getPath(); // document part is contained within the path field
url.getQuery();
 url.getRef(); // gets #part

int tabnumber=rs.getInt(5);
int windownumber=rs.getInt(6);
long time=(Long.parseLong(fromtime)*1000);
long endtime=time+123450;
                int h1= DAO.GetHour.gethour(time);
                    if(hh==h1){
                       i++;
%>
[new Date(<%=time%>), , '<div class="hi"><%=url1%>
```

```
Duration:<%=g%>  Seconds</div </script type=</style type=</link rel=</link rel=</script type=<
/script type <img src="<%=favicon%>"</img src=/><%=title%>','TAB <%=tabnumber%>"],
        <%
                        }
                                }

%>]);

            // specify options
            var options = {
               //width:  "100%",
               //height: "auto",
               //minHeight: 50, // pixels
               //height: "300px",
               //layout: "box",
             //   start: new Date(),
              //  end: new Date(1000*60*60*24 + (new Date()).valueOf()),
            //    editable: true,
               animate: false,
               eventMargin: 10,  // minimal margin between events
               eventMarginAxis: 5, // minimal margin between events and the axis
               showMajorLabels: true,
               //showCustomTime: true,
               //showNavigation: true,
               cluster: true,
               axisOnTop: true,
               snapEvents: true,
               dragAreaWidth: 20
               //groupsWidth : "100px",
               //groupsOnRight: true
            };

            // Draw our timeline with the created data and options
            timeline.draw(data, options);



            google.visualization.events.addListener(timeline, 'edit',
```

```
            function() {
              //console.log('edit')
            }
        );

            google.visualization.events.addListener(timeline, 'change',
            function() {
              //console.log('change')
              //timeline.cancelChange();
            }
        );

            google.visualization.events.addListener(timeline, 'add',
            function() {
              //console.log('add')
              //timeline.cancelAdd();
            }
        );

            /*
             console.profile();
             var count = 10;
             for (var i = 0; i < count; i++) {
             timeline.redraw();
             }
             console.profileEnd();
             //*/
        }
            // Make a callback function for the select item
    var onselect = function (event) {
        var row = getSelectedRow();

        if (row != undefined) {
            document.getElementById("info").innerHTML = "item " + data.getValue(row, 2) + " sel
ected
";
            // Note: you can retrieve the contents of the selected row with
            //      data.getValue(row, 2);
```

```
        }
        else {
           document.getElementById("info").innerHTML += "no item selected
";
        }
     };


        function gup( name ) {
           name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
           var regexS = "[\\?&]"+name+"=([^&#]*)";
           var regex = new RegExp( regexS );
           var results = regex.exec( window.location.href );
           if( results == null )
              return "";
           else
              return results[1];
        }

        var count = (Number(gup('count')) || 100);


              var d1=new Date();

     <body onresize="/*timeline.checkResize();*/">
     <%  String day=session.getAttribute("day").toString();
              %>
     <%=DAO.GetDayLinks.getDayLinks(day)%>


  <%
        int daytoget=Integer.parseInt(session.getAttribute("day").toString());

     %>
   <%=DAO.GetHours.getHours_by_visit_status(daytoget)%>

     <%

        String dd="";
```

```
if(daytoget==0)
            {

dd="Sunday";
}
 if(daytoget==1)
            {

dd="Monday";
}
 if(daytoget==2)
            {

dd="Tuesday";
}

 if(daytoget==3)
            {

dd="Wednesday";
}

 if(daytoget==4)
            {

dd="Thursday";
}
 if(daytoget==5)
            {

dd="Friday";
}
 if(daytoget==6)
 {

dd="Saturday";
}
```

```
        %>
        History on: <%=dd%> : at :<%=h%> hour
         <%
  if(i>0)
          {
     %>

     <div id="mytimeline">

     <div id="info">
  <%=DAO.GetWindowLinks.getWindowLinks()%>
  <%

   }
     else
        {
              %>
   No History Found


     <%
              }
  %>
```

// Index_1.jsp

```
<%@page import="java.net.URL"%>


 <script type="text/javascript" src="http://www.google.com/jsapi">
  <script src="js/jquery-1.8.3.min.js"</script src=>
  <script type="text/javascript" src="timeline.js">
   <link rel="stylesheet" media="screen" href="buttons.css">
    <link rel="stylesheet" type="text/css" href="timeline.css">

     <style type="text/css">
          body {
             color: #4D4D4D;
             font: 10pt arial;
```

```
        }


    <script type="text/javascript">
        var timeline = null;
        var data = null;
        var order = 1;
        var truck = 1;

google.load("visualization", "1");
function getSelectedRow() {
var row = undefined;
var sel = timeline.getSelection();
if (sel.length)

{
          if (sel[0].row != undefined)

{
             row = sel[0].row;
          }
        }
        return row;
    }


        google.setOnLoadCallback(drawVisualization);

        // Called when the Visualization API is loaded.
        function drawVisualization() {

          // Instantiate our timeline object.
  timeline = new links.Timeline(document.getElementById('mytimeline'));
  google.visualization.events.addListener(timeline, 'select', onselect);
          // Create and populate a data table.
          data = new google.visualization.DataTable();
          data.addColumn('datetime', 'start');
         data.addColumn('datetime', 'end');
          data.addColumn('string', 'content');
```

```
        data.addColumn('string', 'group');

data.addRows([

      <%

            java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
            String q="select * from durationtab";
            java.sql.PreparedStatement pstmt= mycon.prepareStatement(q);

            java.sql.ResultSet rs=pstmt.executeQuery();

            while(rs.next())
          {

            String fromtime=rs.getString(1);
            String gapduration=rs.getString(7);
             Float g= Integer.parseInt(gapduration)+(float)(Math.random()*10);
            String lastvisit=rs.getString(3);
            String favicon= rs.getString(10);
            String url1=rs.getString(2);

            //title
            String title=rs.getString(11);

            URL url = new URL(url1);
              url.getProtocol();
              url.getUserInfo();
              url.getAuthority();
               url.getHost();
               url.getPort();
               url.getPath(); // document part is contained within the path field
            url.getQuery();
            url.getRef(); // gets #part
            int tabnumber=rs.getInt(5);
            int windownumber=rs.getInt(6);
            long time=(Long.parseLong(fromtime)*1000);
            long endtime=time+123450;
```

```
                int h= DAO.GetHour.gethour(time);
                %>
 [new Date(<%=time%>),, '<div class="hi"><%=url1%>
Duration:<%=g%>  Seconds</div </script type=</style type=</link rel=</link rel=</script type=<
/script typ<img src="<%=favicon%>"</img src=/><%=title%>','TAB <%=tabnumber%>"],
 <%

            }
 %>]);

          // specify options
          var options = {
             //width:  "100%",
             //height: "auto",
             //minHeight: 50, // pixels
             //height: "300px",
             //layout: "box",
           //   start: new Date(),
            //  end: new Date(1000*60*60*24 + (new Date()).valueOf()),
          //    editable: true,
             animate: false,
             eventMargin: 10,  // minimal margin between events
             eventMarginAxis: 5, // minimal margin between events and the axis
             showMajorLabels: true,
             //showCustomTime: true,
             //showNavigation: true,
             cluster: true,
             axisOnTop: true,
             snapEvents: true,
             dragAreaWidth: 20
             //groupsWidth : "100px",
             //groupsOnRight: true
          };

          // Draw our timeline with the created data and options
    timeline.draw(data, options);
```

```
      google.visualization.events.addListener(timeline, 'edit',
      function()

{
              //console.log('edit')
            }
        );

       google.visualization.events.addListener(timeline, 'change',
       function()

{
              //console.log('change')
              //timeline.cancelChange();
            }
        );

            google.visualization.events.addListener(timeline, 'add',
            function() {
              //console.log('add')
              //timeline.cancelAdd();
            }
        );


             console.profileEnd();
             //*/
        }
            // Make a callback function for the select item
      var onselect = function (event)

{
        var row = getSelectedRow();

        if (row != undefined) {
            document.getElementById("info").innerHTML = "item " + data.getValue(row, 2) + " sel
ected
";
            // Note: you can retrieve the contents of the selected row with
```

```
            //      data.getValue(row, 2);
          }
          else {
            document.getElementById("info").innerHTML += "no item selected
";
          }
        };

          /**
           * Get URL parameter
           * http://www.netlobo.com/url_query_string_javascript.html
           */
          function gup( name ) {
            name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
            var regexS = "[\\?&]"+name+"=([^&#]*)";
            var regex = new RegExp( regexS );
            var results = regex.exec( window.location.href );
            if( results == null )
              return "";
            else
              return results[1];
          }

          var count = (Number(gup('count')) || 100);


              var d1=new Date();

         <body onresize="/*timeline.checkResize();*/">
        <%=DAO.GetDayLinks.getDayLinks("no")

%>
     <%=DAO.GetHours.getHours()

%>
       <div id="mytimeline"

  <div id="info">
    <%=DAO.GetWindowLinks.getWindowLinks()
```

```
%>
```

// Windows.jsp

```
<%@page import="java.net.URL"%>

<script type="text/javascript" src="http://www.google.com/jsapi">
<script src=jsjquery-1.8.3.min.js="">
<script type="text/javascript"</script type=</script src=js> src="timeline.js">
<link rel="stylesheet" media="screen" href="buttons.css">

<link rel="stylesheet" type="text/css" href="timeline.css"<style type="text/css">
 body
{
            color: #4D4D4D;
            font: 10pt arial;
         }

      <script type="text/javascript">
         var timeline = null;
         var data = null;
         var order = 1;
         var truck = 1;

         google.load("visualization", "1");
     function getSelectedRow()
{
 var row = undefined;
         var sel = timeline.getSelection();
         if (sel.length) {
            if (sel[0].row != undefined) {
```

```
            row = sel[0].row;
          }
        }
        return row;
    }

        // Set callback to run when API is loaded
        google.setOnLoadCallback(drawVisualization);

        // Called when the Visualization API is loaded.
        function drawVisualization() {

          // Instantiate our timeline object.
  timeline = new links.Timeline(document.getElementById('mytimeline'));
  google.visualization.events.addListener(timeline, 'select', onselect);
          // Create and populate a data table.
  data = new google.visualization.DataTable();
  data.addColumn('datetime', 'start');
  data.addColumn('datetime', 'end');
  data.addColumn('string', 'content');
 data.addColumn('string', 'group');

data.addRows([

        <%
          String window=request.getParameter("windownumber");
           int w= Integer.parseInt(window);
           String day=session.getAttribute("day").toString();
           int daytoget=Integer.parseInt(day) ;
           java.sql.Connection mycon=DB.MyDBBean.getDBConnection();
          java.sql.PreparedStatement pstmt=null;
          String q="select * from durationtab where dayvisited=? and windownumber=?";
           pstmt= mycon.prepareStatement(q);
         pstmt.setInt(1, daytoget);
         pstmt.setInt(2, w);
        java.sql.ResultSet rs=pstmt.executeQuery();
 int i=0;
 while(rs.next())
```

```
    {

     i++;
     String fromtime=rs.getString(1);
     String gapduration=rs.getString(7);
     Float g= Integer.parseInt(gapduration)+(float)(Math.random()*10);
     String favicon= rs.getString(10);
     String title=rs.getString(11);
     String url1=rs.getString(2);
     URL url = new URL(url1);
     url.getProtocol();
     url.getUserInfo();
     url.getAuthority();
     url.getHost();
     url.getPort();
     url.getPath(); // document part is contained within the path field
     url.getQuery();
     url.getRef(); // gets #part
     int tabnumber=rs.getInt(5);
     int windownumber=rs.getInt(6);
     long time=(Long.parseLong(fromtime)*1000);
     long endtime=time+123450;
     if(session.getAttribute("hh")!=null)

{

      int h=DAO.GetHour.gethour(time);
              %>
 [new Date(<%=time%>), , '<div class="hi"><%=url1%>
Duration:<%=g%> Seconds</div </script type=</style type=</link rel=</link rel=</script type=
<img src="<%=favicon%>"</img src=/><%=title%>','TAB <%=tabnumber%>"],
        <%

           }
             else
                 {


                 }
```

```
                    }


%>]);

        // specify options
        var options = {

          animate: false,
          eventMargin: 10,  // minimal margin between events
          eventMarginAxis: 5, // minimal margin between events and the axis
          showMajorLabels: true,
          //showCustomTime: true,
          //showNavigation: true,
          cluster: true,
          axisOnTop: true,
          snapEvents: true,
          dragAreaWidth: 20
          //groupsWidth : "100px",
          //groupsOnRight: true
        };

        // Draw our timeline with the created data and options
        timeline.draw(data, options);
       google.visualization.events.addListener(timeline, 'edit',
        function()

{
          //console.log('edit')
        }
      );

        google.visualization.events.addListener(timeline, 'change',
        function()

{
          //console.log('change')
          //timeline.cancelChange();
```

```
            }
        );

            google.visualization.events.addListener(timeline, 'add',
            function() {
                //console.log('add')
                //timeline.cancelAdd();
            }
        );

            /*
             console.profile();
             var count = 10;
             for (var i = 0; i < count; i++) {
             timeline.redraw();
             }
             console.profileEnd();
             //*/
        }
            // Make a callback function for the select item
    var onselect = function (event) {
      var row = getSelectedRow();

        if (row != undefined) {
            document.getElementById("info").innerHTML = "item " + data.getValue(row, 2) + " sel
ected
";
            // Note: you can retrieve the contents of the selected row with
            //      data.getValue(row, 2);
        }
        else {
            document.getElementById("info").innerHTML += "no item selected
";
        }
    };


        function gup( name ) {
```

```
            name = name.replace(/[\[]/,"\\\[").replace(/[\]]/,"\\\]");
            var regexS = "[\\?&]"+name+"=([^&#]*)";
            var regex = new RegExp( regexS );
            var results = regex.exec( window.location.href );
            if( results == null )
               return "";
            else
               return results[1];
         }

         var count = (Number(gup('count')) || 100);

          var d1=new Date();
         <body onresize="/*timeline.checkResize();*/">
<%=DAO.GetDayLinks.getDayLinks(day)%>


<%=DAO.GetHours.getHours_by_visit_status(daytoget)%>
            <%

         String dd="";
         if(daytoget==0)
                     {

         dd="Sunday";
         }
           if(daytoget==1)
         {

         dd="Monday";
         }
           if(daytoget==2)
        {

         dd="Tuesday";
         }

           if(daytoget==3)
                     {
```

```
        dd="Wednesday";
        }

         if(daytoget==4)
                    {

        dd="Thursday";
        }
         if(daytoget==5)
                    {

        dd="Friday";
        }
         if(daytoget==6)
                    {

        dd="Saturday";
        }

        %>
        History on: <%=dd%> : in Window :<%=w%>
        <%
  if(i>0)
          {
  %>
     <div id="mytimeline">

     <div id="info">
     <%=DAO.GetWindowLinks.getWindowLinks()%>

     <%
     }
  else
     {
          %
```

No Result Found

```
<%
    }
 %>
```