An Auto-Response User Interface for Web Pages

By:

- Muhammad Ismail 094451
- Abdul Wadood 094450

Supervised By:

Hasan Mahmud

Assistant Professor, Department of Computer Science and Engineering

Computer Science and Engineering Department,

Islamic University of Technology.

September, 2013.

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma

Signatures of the Candidates

_____

Abdul Wadood

Student No. 094450, Session 2009-2013

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Board Bazar, Gazipur Bangladesh.

_____

Muhammad Ismail

Student No. 094451, Session 2009-2013

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Board Bazar, Gazipur Bangladesh.

Signature of the Supervisor

_____

Hasan Mahmud

Assistant Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Board Bazar, Gazipur Bangladesh.

Table of Contents

Acknowledgement……5

Dedication……6

Abstract……7

Chapter-1 Introduction

1.1 Background:……8

1.2 Motivation: ……8

1.3 Auto-Response Use0r Interface: What is it?……9

1.4 Properties of an Auto-Response User Interface: ……9

1.4.1 Changing Font Size: ……10

1.4.2 Changing Image Size:……11

1.4.3 Changing Form Size: ……12

1.4.4 Reflect the Viewing Needs: ……12

1.4.5 Bandwidth Sensitive: ……12

1.4.6 Screen Co-ordinates and Viewport Determine the Behavior of the Interface: ……13

1.4.7 Usability Goals .t:……13

1.4.8 User Experience Compliant:……13

Chapter-2 Related Works

2.1 Responsive Web Design:……14

2.2 Segmentation of Web Page:……15

2.3 Relative Importance:……16

2.4 Content Filtering:……17

2.4.1 Data Extraction:……17

2.4.2 Image Extraction:……18

2.4.3 Link Extraction:……18

Chapter-3 Problem Identification

3.1 Screen Size and Resolution and its effect on display: ……20

3.2 Diversity of Screen Sizes and Resolutions:……21

3.3 Problem Identification:……23

3.3.1 Test:……23

3.3.2 Observation:……25

3.3.3 Test:……25

3.3.4 Observation:……26

3.3.5 Conclusion:……26

3.4 Solutions:……26

3.5 Problem Definition: ……27

Chapter-4 Proposed Methodology

4.1 Server Side CONSIDERATIONS:……29

4.1.1 JavaScript:……29

4.1.2 CSS:……30

4.2 Client Side CONSIDERATIONS: ……31

4.2.1 Algorithm: ……32

4.2.1.1 Block Importance: ……32

4.2.1.2 Segmentation:……33

4.2.1.3 Pseudocode:……36

4.3 Performance Analysis:……37

Chapter-5 Implementations

5.1 Extension Implementation:……38

5.2 User Application Implementation: ……40

5.2.1 User Interface (UI): ……40

4.2.2 Code Details:……42

4.2.3 User Application Settings:……42

5.2.4 Scalability: ……44

5.2.5 Challenges: ……45

Future Works and Conclusions ……46

References:……47

Acknowledgement

There is absolutely no doubt that had we not been candidates of sound mind and with good health, we would never have accomplished this demanding task. We, therefore, thank the al-mighty Allah for granting us such success prerequisites that cannot be granted by none other than him.

We are indeed very grateful to our distinguished and intuitive supervisor, Mr. Hasan Mahmud, Assistant Professor Department of Computer Science and engineering IUT. He gave us time from his busy schedule and guided us on the right path. His professional suggestions whetted our appetite enough to venture such a daunting job.

Dedication

I dedicate this to my family whom, encouragement and motivation made me capable accomplish this challenging job.

- Abdul Wadood

To my Mom

- Muhammad Ismail

Abstract

At the time of advent of Internet and its success very limited devices had access to the Internet. Desktop computers were the primary source to browse through web pages. Since then, website developers and designers central efforts were desktop oriented until recently, when market was dominated by small devices. The number of users accessing Internet through these small devices will surpass those using desktop computers to access Internet by 2014. The web pages primarily designed for large screen had problems on small screens like too small font, horizontal scrolling, excess of contents, small images, large page sizes etc. Many firms have made a separate version of their site for mobile devices. Making a separate version of the website spawns new array of problems. Despite that millions of web pages primarily developed for desktop users in mind, still have these problems.

In order to overcome these issues we propose an auto-response user interface for web fronts. Such an interface is context aware and adapts components of interface to the needs of device on which it lands. Components of interface consist of images, fonts, layouts and grids. In case of slow internet speed, interface minimizes bandwidth consuming factors by removing image based Ads, flash based Ads, changing the size of the image, blocking JavaScript, filtering important segments of the webpages etc.

Implementation is carried out and tested against the common issues giving satisfactory initial results. The implementation is based on the algorithm which obtains the DOM tree from HTML stream and divides that tree into segments for filtering and important blocks determination.

Chapter-1 Introduction

In this chapter we give a brief overview of our study so as to have an initial idea about what is coming forth in the next chapters. The goal for this chapter is to introduce the concepts that lie at bottom of our study. Firstly, background will explain the broad picture of the study of this kind. Next discussion is about the drive and the reason for choosing this specific topic as our research interest. The chapter summarizes properties of auto-response user interface at the end.

1.1 Background:

Study any kind of interface, finding out its problems, making improvements to it directly comes under the scope of Human Computer Interaction (HCI). HCI originated within the US military during World War II. During the war military officials realized that a badly designed weapons can kill your own forces instead of the enemy, thus giving birth to usability. The field got widened and its applications were found in computer industry and products development [42].

There are many products we use each day cell phone, computer, personal organizer, remote control, soft drink machine, coffee machine, ATM, ticket machine, library information system, the web, photocopier, watch, printer, stereo, calculator, video game.. The list is endless [43]. The question is, the interfaces these devices offer are usable or not? How many are actually easy, effortless, and enjoyable to use? There are many devices that caused us considerable grief and wastage of time trying to get it to work. There are many products that require users to interact with them to carry out their tasks but when it comes to web it's all about interaction. At the inception of web, design and development target was the users with desktop machines. This worked fine until the eruption of mobile devices in the market and the use of these devices for accessing internet. Design and interface primarily aimed for desktop users did not do well on mobile devices. This led to the birth of WML. Despite ramping processing power and memory up, issues still exist. Failure in finding industry standard solutions will result in user's frustration, wastage of time and dissatisfaction. Our work is a part of the struggle to narrow down the gap between desktop internet users and their mobile counterparts.

1.2 Motivation:

As mobile devices are gaining popularity because of increment in its processing power, sleek designs and price reduction. At same time, with improvement in telecommunication technologies internet bandwidth has increased and internet is more easily accessible than ever. We had been observing for the last few years that people in our social circles always

complained about their mobile Internet browsing experience. We ourselves were confiscated by the same sort of problems when we tried to access internet from mobile devices. Since then, we had been contemplating on any possible solution. This provided us the necessary drive to choose this domain as our primary research interest.

1.3 Auto-Response User Interface: What is it?

The concept of responsiveness comes from responsive architecture which is still a young and evolving field. Tristan [31] in his paper defines responsive architecture as such: a class of architecture or building that demonstrates an ability to alter its form, to continually reflect the environmental conditions that surround it. It stems from the notion of responsive architectural design, whereby a room or space automatically adjusts to the number and flow of people within it. Responsive architecture is made possible by combination of embedded robotics like motion sensors, climate control systems and tensile materials in installations of walls that bend, flex, and expand as crowds approach them.

In designing web interface the same sort of notion was introduced due to the variety of screen sizes and resolutions. Like responsive architecture, web interface should automatically adjust. It shouldn't require countless custom-made solutions for each new category of users. This lead us to the definition of an auto-response user interface:
"*A web based user interface which follows the approach that a design of interface should respond to the user's behavior and environment based on screen size, resolution and orientation of the device without the interruption of the user*".
The definition will become clearer when we read the properties of such a user interface. The effect of screen size and resolution will be discussed in detail later in this chapter.

1.4 Properties of an Auto-Response User Interface:

Before describing the exhaustive list of the properties of an auto-response user interface, it would be helpful to know a few technical terms.

Screen Resolution:

The display resolution of a digital display is the number of distinct pixels in each dimension that can be displayed. A screen resolution of 256×256 has 256 pixels across the screen width and 256 pixels across the screen length and a total of 65536 pixels on the whole screen. A pixel is the smallest addressable point on screen.

Viewport:

The rectangular region of the screen used to display window.

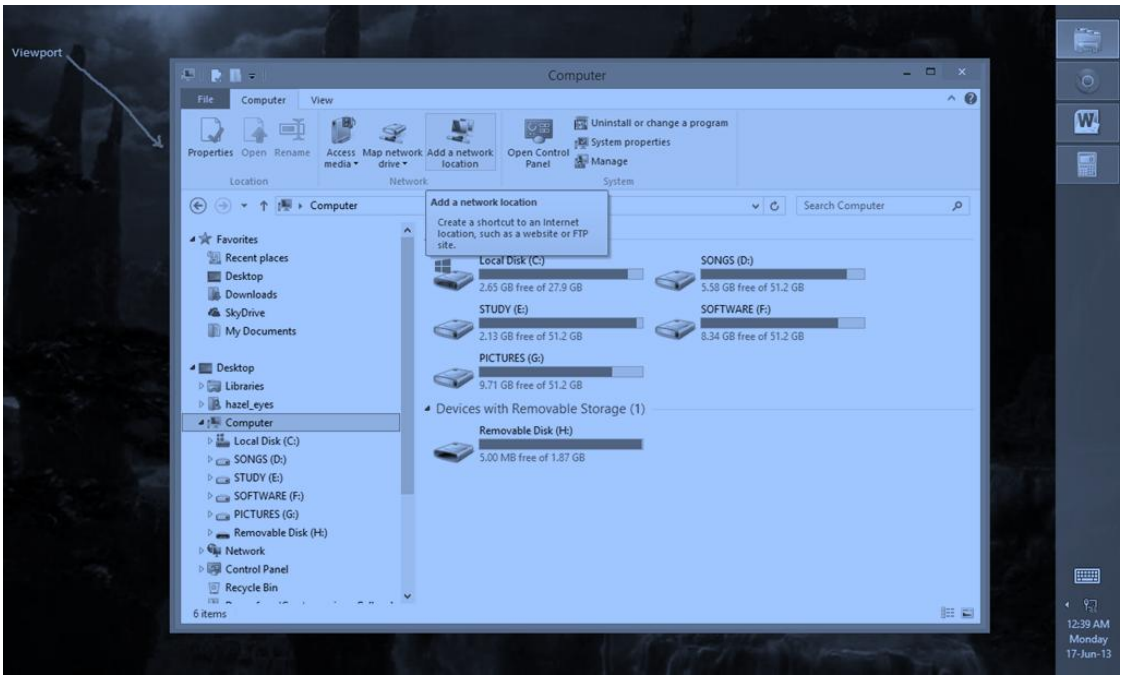

Figure 1.1 the area inside the red lines shows viewport.

Following are the main properties of an auto-response user interface.

1.4.1 Changing Font Size:

Size of the text changes as the size of the screen changes. The proportion of change in font size should be handled so carefully so that it is not outlandishly large or so tiny that its legibility declines from the average. One excellent example is illustrated in Figure-1.2.

Figure 1.2 same web interface on three devices iMac, iPad and iPhone.

1.4.2 Changing Image Size:

Just like font size, images don't keep its size fixed in auto-response user interface. Images must change in such a way to fit the changing dimension. Like the font size, the proportion of change in size of images should be tailored in such a way to fulfill usability requirements. Figure-1.3 and figure-1.4 illustrate the beauty of such an interface. By looking closely to the both the figures we can see that pipe's image and text change in such a way in order to accommodate in the available space with changing dimension. Though this is a simple example but the idea is applicable to complex website with equal elegancy.

Figure 1.3 shows location and size of text and pipe image.

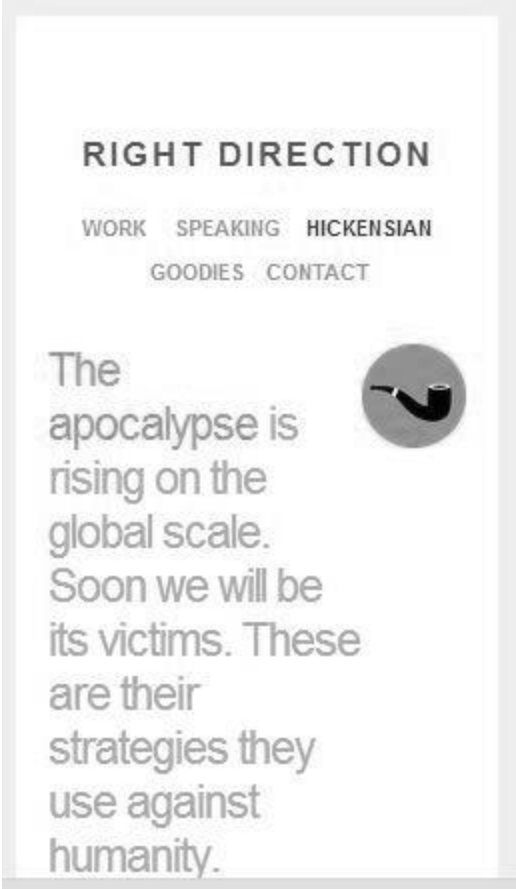

Figure 1.4 shows how relative position and size of text and pipe's picture changes with change of viewport.

1.4.3 Changing Form Size:

Components of form like textboxes, buttons; dropdown lists etc. also change its size with changing dimensions.

1.4.4 Reflect the Viewing Needs:

Auto-response user interface is not passive but responds to user needs. Viewing needs of the user are the user's preferences of viewing a specific content. These might be viewing in full screen, small screen, different screen co-ordinates and resolutions.

1.4.5 Bandwidth Sensitive:

Bandwidth sensitivity is one of the important parameter of such an interface. Depending on the speed of the user end connection, required contents are provided accordingly. When the speed of the connection is slow, only minimum amount and important contents are provided. This approach is also followed by Gmail and Yahoo.

1.4.6 Screen Co-ordinates and Viewport Determine the Behavior of the Interface:

This is the most important of all and is the cynosure of our study. Traditional web based user interface has failed badly when it comes to diverse devices like smartphones and tablets. But this is not the end. The next coming wave belongs to wearable devices and some of them are already in the market like Google Glass. If we are not changing existing websites how do we think it will appear on these new coming devices? To resolve this issue our interface should not remain constant but changes as the environment changes.

1.4.7 Usability Goals Compliant:

If the changing behavior of the interface does not comply with usability goals, then it is no more than a problem per se. We have to ask ourselves that an interface which is efficient and effective on desktop is it so on other devices? Where do we make compromise and how to minimize the effect of that compromise? Many such usability goals should be kept in mind while testing with auto-response user interface.

1.4.8 User Experience Compliant:

Our proposed UI must also fulfill the needs of user experience goals. If an interface changes itself and its look and feel is not up to expectation. Again we will be in trouble and most of the people would avoid to risk its use because when it comes to web, then it's all about aesthetics and satisfaction. Considering user experience goals is also an important property of the auto response user interface.

The properties mentioned above is an exhaustive list based on our observation and study. Core idea of an auto-response user interface sounds right with above list of properties. Our further study and experimentation would be how to satisfy above properties, all in one place.

Chapter-2 Related Works

Extensive research has been done on the web, targeting its different problems and potentials. Since the beginning of World Wide Web it has attracted a lot of attention of researchers and resources. Before we take a deep plunge into the topic, it would be honest to point out the sources of help which enable us step forward. Our work combines the idea of responsiveness in user interface, relevancy indication, segmentation and filtering. We used responsiveness to provide context awareness to the page being viewed so that it adapts interface to the needs of the device. Relevancy helps to remove unnecessary parts, segmentation is the idea behind dividing web page into small chunks so that those chunks can be manipulated according to users' priorities, relevancy and filtering techniques.

2.1 Responsive Web Design:

Many professional are currently experimenting with this very idea of responsive web design. Kayla Knight is one of them. She defines **Responsive Web design** as an approach that design of the website should respond to changes based on screen size, platform and orientation [13]. According to her in the field of Web design and development, it is hard to keep up with the endless new resolutions and devices. For many websites, creating a version specific for each resolution and new device is impractical. On the other hand we lose visitors if we don't fix this problem. She suggests fluid design and flexible grids and layouts as a solution of this dilemma.
Filament Group is trying to finds means of creating responsive images. They are experimenting with HTML5 and have achieved a great degree of success [22].
Ethon Marcotte is another name on this horizon. The term responsive web design was coined by him. In one of his articles he writes that inconsistent window widths, screen resolutions, user preferences, and our users' installed fonts are but a few of the intangibles we negotiate when they publish their work. He says that some client request him to design a website for iPhone. Does that mean there should also be a dedicated site for Samsung Galaxy S series? Thus the nightmare for designers as well as for owner of the websites arises. He thinks that responsive design if properly tailored can overcome this issue. He is experimenting with CSS media queries to achieve responsiveness [23].

Oliver Reichstein applied the idea of responsive design to one of the application for iOS called iA Writer. In his recent article he mentioned three core reasons of why Writer had only one typeface and font size. i) Fumbling with fonts is the main reason of distractions. ii) The text up close helps focus when crafting prose sentence by sentence. iii) They made a compromise choosing the font size for all Mac Screens. They knew the fact that on some older screens the font would seem rather big, while on newer screens it would seem a little too small.

In response to these issues they were inspired by the idea of responsive design .Writer now changes the font size based on window width. They express a high level of satisfaction with this idea [24] [25].

There exists a great debate among community of web designers about responsive web design. Experts have written many great articles. The idea is still in its inception and getting more and more attention as new devices come in the market. For us it is important because we are applying some of the concepts originating from the responsive web design like flexible images, font sizes etc.

2.2 Segmentation of Web Page:

"Page segmentation is the process of dividing a web page into different parts based on some criteria".

To distinguish different information in a web page, we first need to segment a web page into a set of blocks. There are several kinds of methods for web page segmentation. The most popular ones are DOM-based segmentation [10], location-based segmentation [28] and Vision-based Page Segmentation (VIPS) [18]. Here we discuss a few.

A DOM Tree-based segmentation algorithm to extract image segments from webpages using the image characteristics was proposed by Md. Belkhatir et. al [26]. For every image node found in the DOM Tree, the algorithm finds the image segment using heuristic determined by the image characteristics. This is accomplished by detecting the variation in total number of texts in each upward level of the DOM Tree, beginning from the image node. They apply their algorithm to extracting contextual information of the image.

Another segmentation algorithm developed by Microsoft researchers Deng Cai et al. is a web content structure analysis based on visual representation [18]. They represent a web page as a triplet: blocks, separators (vertical and horizontal), weights indicating visibility.

The vision-based content structure of a page is obtained by combining the DOM structure and the visual cues. VIPS is top down and iterative strategy is followed until all appropriate nodes are found to represent the visual blocks. VIPS according to the developers is applicable to information retrieval, information extraction and automatic page adaptation. As Le Liu has applied it to content extraction [27].This very idea is extended by Elgin Akpınar et. al [19]. Their main idea is that if a node is invisible without any child, then this node is partially valid node and need further processing to determine its weight in visibility scale.

Another segmentation algorithm based on semantics of the page has been devised by Hasan Davulcu et. al [16]. This page segmentation algorithm partitions a given web page first into flat segments.

A segment is defined as a contiguous set of leaf nodes within a Web page. The target of algorithm is to find homogeneous segments, where the presentation of the content within each segment is uniform. The flat segments obtained are organized into hierarchical group. Labels are assigned to each group and then they are meta tagged. They applied this algorithm to meta data collection from web pages. K.S. Kuppusamy and G. Aghila [3] applied segmentation to content filtering. They do DOM based segmentation and then pass segments through filters.

2.3 Relative Importance:

In the context of our study we also explored about relative importance of different parts of the web page. The aim was to truncate any part of the web page that has lowest score in relevancy determination. Consequently, noisy contents are reduced and valuable resources on small devices are saved. Its usefulness was also in terms of proper utilization of bandwidth. We discuss similar algorithm in this section.

Throughout our study we found the work of Ruihua Song et al. [4] on relevancy elegant of all. Song et al. has investigated block importance from the user perspective. Their experiments revealed that users do differentiate between highly important and less important blocks. They also determined that there are materials which cause annoyance to the users. They present a block importance determinants model of which are spatial features, and content features.

Flavián Carlos et al. [7], present a concise study of website relevancy using heuristics. Their heuristic evaluation is a way to find problems and good practices in a user interface. This evaluation involves having a small set of evaluators who examine the interface and judge its compliance with recognized usability principles, namely “heuristics”. These heuristics are based on marketing, usability, information systems and new technologies.

Dave Gehrke and Efraim Turban[1] in their study found hundred issues regarding website design. They divided those issues into five major categories. Each category helps determine the relative importance and effectiveness of the components of the site. Among their categories was page loading speed. A component that loads quickly is more important than one which does not. Second category was Business Content whose quality of presentation and usefulness gains higher in terms of relevant importance. Navigational efficiency also increases the weight of importance in a context of web page. Similar are security and marketing which for the most part is not our concern in the context of our study. These five major indicators make our job easier to assign weights to different segments of the web page.

Zhilin Yang et al. [6] identified 14 service quality dimensions. We mention here only those which are important in the context of our work. Among them are responsiveness, ease of use, convenience, personalization and aesthetics. Page content which possess these attributes has higher rank in terms of importance.

Hongxiu Li and Reima Suomi [5] also found quality dimensions mostly similar to [6] but in addition they included empathy which could be one factor to consider in our user interface. R. Baik et al. [14] apply vision based algorithm to indexing relevant blocks in the page but does not show how the relevancy among the blocks is calculated.

2.4 Content Filtering:

A web page is the conglomeration of links, images, texts, media, and flash. Not all of them are required. Web pages are also populated with noisy and redundant data. To obtain the main and actual content of the page, it has been the main concern of data mining for a long time. Extensive research is done targeting this issue. In implementation of our algorithm we have used user defined filtering. Therefore, we shed light on some popular algorithms, techniques and findings regarding filtering.

K.S.Kuppusamy and G.Aghila [3] give a brief account of filtering in their paper on content filtering. They enlist content filtering approaches in rating systems, black listing / white listing and keyword blocking. In rating systems users are asked to rate a web site based on the content. This rating helps as a filtering device [30] as explained by Paul Resnick and Jim Miller. The black listing / white listing maintains a set of URLs manually prepared for filtering. This method suffers from scalability problem. Key word blocking can be used in text classification

Filtering concept can be extended further to:

2.4.1 Data Extraction:

P. YesuRaju and P. KiranSree [8] perform language independent content extraction by using VIPS which is obtained through DOM structure. They detect content data in content structure.

Suhit Gupta et al. [10] use DOM based segmentation for content extraction. In order to extract content, the page is first passed through a parser which creates a Document Object Model tree representation of the web page. The DOM tree is hierarchically arranged and can be analyzed in sections for necessary contents. After the contents have been collected, it can be output in either HTML form or as plain text.

Another work accomplished by Nikolaos Pappas et al. [13] extracts informative textual parts from webpages. Their method processes a web page and partitions it into semantic parts while it ignores noisy contents. Their algorithm exploits visual and non-visual characteristics of a web

page encapsulated in a DOM tree. Their algorithm does region extraction using the optimal values from the SD-tree which is tree built by the algorithm itself.

Srinivas Vadrevu et al. [15] in their work use presentation regularities and domain knowledge for information extraction from web pages. They use semantic partitioning algorithm that works by exploiting the presentation regularities in web pages. They do the whole process in four phases: page segmentation, grouping, promotion and semantic role annotation. The page segmentation algorithm relies on the DOM tree representation of the HTML Web page and traverses it in a top–down fashion in order to segment the contents of the page. In the grouping phase they define a group to be a contiguous collection of instances that are presented together in a web page. In promotion labels are assigned to groups. The last phase annotation with semantic roles takes promoted groups with appropriate label and apply concept from pattern recognition to extract useful contents.

2.4.2 Image Extraction:

Gagan Preet Kaur et al. [9] used partial tree alignment algorithm for image extraction. Their algorithm is five step processes. In first step Weighted Page Rank algorithm takes into account the importance of both the inlinks and the outlinks of the pages. In the second phase, the web page is split into segments, without extracting any data of images. In the third step, the partial tree alignment algorithm is applied to data records identified. Content extraction is the fourth module. It deals with extracting the contents from the Web pages. Display Content is the module deals with the user interface.

2.4.3 Link Extraction:

G. Poonkuzhali et al. [12] uses a mathematical approach for extracting links. In the proposed system, web documents are extracted from the search engines by giving query by the user to the web. Then the obtained web document is divided into web pages based on the links. Then all the pages are preprocessed , by stemming process, stop words removal and except text, images, audio are also removed. This whole process is illustrated in figure.

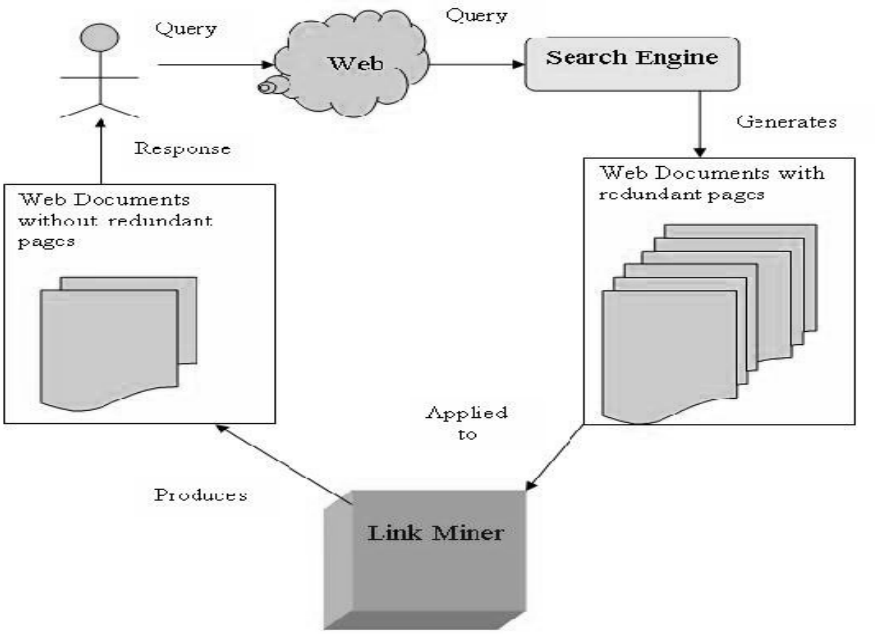

Figure 2.1 shows algorithm base on mathematical approach of [12].

.

Chapter-3 Problem Identification

A decade back desktop computers were the dominant machines for accessing web sites. With advent of cellular technology scenario has changed. Improvements in cellular technology and other wireless technologies like Wi-Fi, Bluetooth, and Wimax have ushered in an era of mobility. Hundreds of mobile devices hit the market with fundamental capability of internet access. By 2014 internet traffic from mobile users accessing the internet will surpass that of desktop users. Then there comes a problem. The websites designed for desktop computer does not behave alike on mobile devices. This created a dilemma among designers and owners of the websites. Almost every new client these days wants a mobile version of their website. It's practically essential after all: one design for the BlackBerry, another for the iPhone, the iPad, netbook, Kindle — and all screen resolutions must be compatible, too. In the next years, designing for a number of additional inventions would be required. When will the madness stop? It won't, of course.

As far as the web based user interface is concerned, developers are quickly getting to a point of being unable to keep up with the endless new resolutions and devices. For many websites, creating a website version for each resolution and new device would be impossible, or at least impractical. Should we just suffer the consequences of losing visitors from one device, for the benefit of gaining visitors from another? Or is there another option? To determine, in this chapter we analyze screen sizes, resolutions, its diversity and impacts. Next we give a brief account of the observations we made, then specify our problem and provide a formal definition.

3.1 Screen Size and Resolution and its effect on display:

New devices with different screen sizes come into the market from different brands every day, and each of these devices there are variations in size, functionality and even color. The amount of information Q that can be displayed is directly dependent on the screen resolution R of the device. Q is directly proportional to R. Mathematically we can write,

$$Q \propto R$$

$$Q = f(R)$$

Therefore, it is necessary to analyze screen resolutions. Besides, we will also see what issues arise when same interface land on screen with different resolutions. In domain of screen of devices we usually encounter two terms commonly:

- Screen Size
- Screen Resolution

Our first responsibility would be to clarify what these terms mean and then differentiate among them on sound grounds.

The size of a screen is simply the diagonal measure of that screen from one corner to the other. A larger size screen does not mean that you can display more information; it only means the displayed data would appear larger. This can be clarified by an example. If we have a television set; program can be viewed on a 32" TV, or watch that same program on a 60" model. This does not mean the later displays more information than former. It only means later displays it bigger. Resolution on the other hand determines how much information is displayed on the screen. *The resolution of display device is the number of distinct pixels in each dimension that can be displayed*. A common resolution found on laptop is 1024 x 768, on 11.6" and 15"6" screens. A 10-point font in a 1024 x 768 resolution will appear smaller than that same point size in a 640 x 480 resolution because the pixels are physically smaller in a 1024 x 768 display. In addition, a higher resolution allows more words per line and more lines per screen than can be displayed in a lower resolution [32].

Concluding with this important point to remember that the size of a screen only determines how large for instance an image will appear on the screen, it's the resolution that determines how much of an image can be displayed.

3.2 Diversity of Screen Sizes and Resolutions:

Morten Hjerde [33] from Rift Labs has collected data that covers about 400 different device models sold in the Norwegian market from 2005 to 2008.

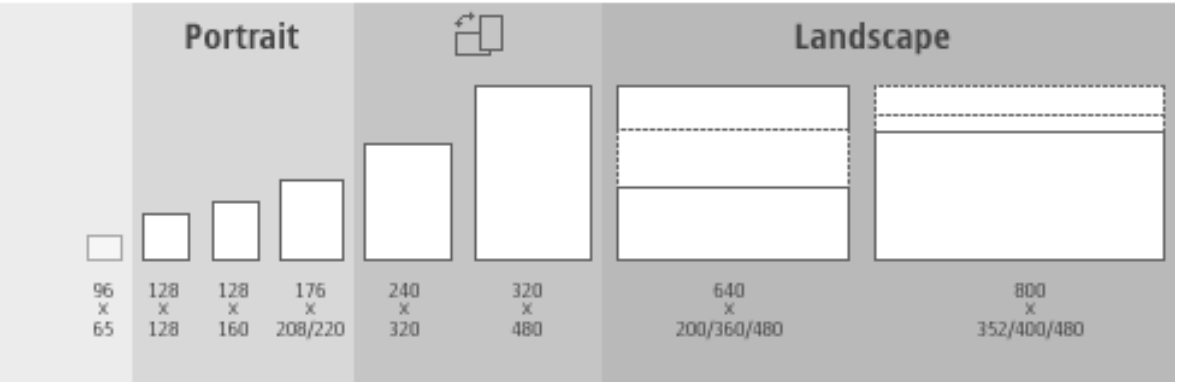

Figure 3.1 shows resolutions of different screens from older to newer.

His data shows that relative screen resolution difference has increased. The difference between the smallest (128 x 128) and the largest (800 x 480) is now a factor of 23. That means the largest screen is 23 times bigger than the smallest one.

Table 3.1 shows variation of screen resolution from different mobile phone making firms.

Manufacturer	small screen	big screen
Sony Ericsson	128 x 160	176 x 220
Nokia	128 x 128	176 x 208
Samsung	128 x 160	176 x 220
Siemens	130 x 130	132 x 176

Table 3.2 shows screen Resolutions of some of the latest mobile devices' displays.

Screen	Size	Model
640x480	5.0"	HTC X7500
800x352	4.0"	Nokia E90 Communicator
800x400	3.0"	Sony Ericsson Xperia X1
1136x640	4.0"	iPhone 5 [34]
1920x1080	5.0"	Sony Xperia Z [35]
1280x768	4.2"	BlackBerry Z10 [36]
1920 x 1080	5.0"	Samsung Galaxy S4 [38]
1280x768	4.5''	Nokia Lumia 1020 [39]
1080x1920	4.7"	HTC One [37]

As the screen resolution in table-3.1 and table-3.2 show that even the same brands are not using constant screen resolutions. Screen resolutions vary with each device. Besides, screen sizes also differ from one device to another. Both size and resolution together have impact on how a user interface appears on the screen specifically web based user interface. Another issue of the screen is whether it is landscape or portrait. Some devices support both sorts of orientation at the same time. Therefore, when a user flips from landscape to portrait then user interface must adjust itself accordingly.

In table-3.2 some of the latest and flagship smartphones from different brands with their resolutions are shown. By observing closely there is one commonality among all: the screen resolution has the tendency of increasing till now. The future might see mobile devices with even higher resolution.

Note: By analyzing sizes and resolutions and their diversity we want to make one point clearer that there are many sizes and resolutions in the market and each one of them has unprecedented effects on user interface.

3.3 Problem Identification:

As we mentioned early in this chapter that influx of devices of different screen sizes and resolution in the market, for all website owners it has become difficult how to organize website contents and user interface for users with tablets, smartphones and desktop. Many Internet giants have opted making different websites for mobile and desktop users. Among them are Facebook, Youtube, Flicker, Digg etc. But still there are millions of websites which has done nothing about it. To know what could be its effects, we ran tests on two websites namely official website of Islamic University of Technology (IUT), Bangladesh [40] and official website of Bangladesh University of Engineering and Technology Dhaka Bangladesh (BUET) [41]. Their appearances on desktop have shown respectively in figure-1 and figure-2.

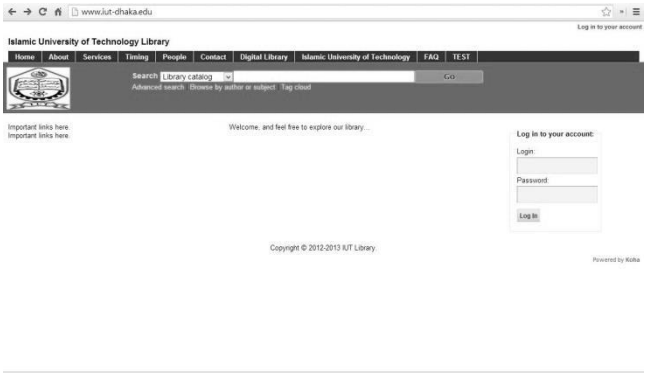

Figure 3.2 shows Official website of IUT library on 17.5" with 1280 x 768 desktop display.

Figure 3.3 shows official website of BUET on 17.5" with 1280 x 768 desktop display.

3.3.1 Test:

Figure-3.2 and figure-3.3 shows websites with desktop users in mind. Both of these institutions don't have separate websites for mobile users. So we would look at them on small screens

individually in order to see its usability and user experience performance. In this test we used Nokia Asha 205 with display size 2.4" and screen resolution 320 x 240 and Nokia N8 with screen size 3.5" and screen resolution 360 x 640 in our experiment. Now we would look at official site of IUT on both mentioned devices.

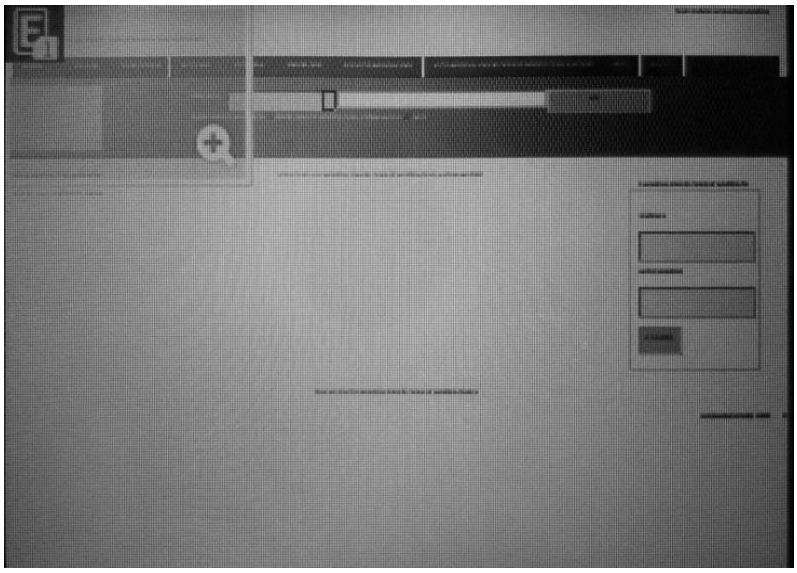

Figure-3.4 shows IUT library official website on Nokia Asha 205

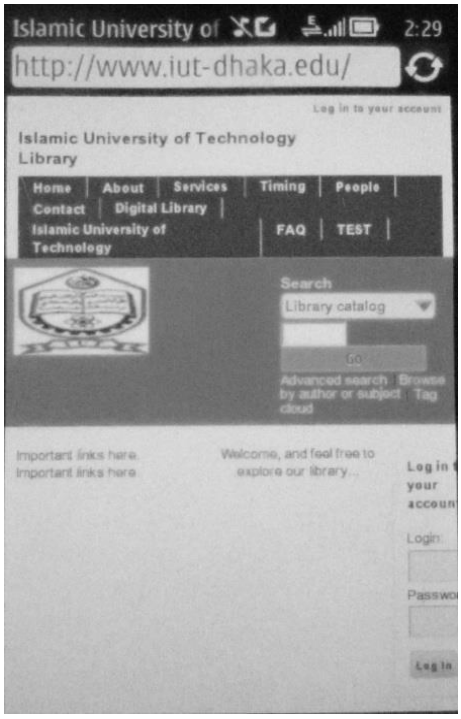

Figure-3.5 shows IUT library official website on Nokia N8

3.3.2 Observation:

From figure-1, figure-3 and figure-4 we can see the difference of user interface and amount of contents displayed on both mobile devices compared to its desktop counterpart. In figure-3 almost nothing is visible. Though there is a built in zooming option but that adversely affects user experience and does not achieve usability goals. In figure-4 not all the contents are available and among the available contents, most of them are badly distorted. Beside that there is horizontal scrolling required which is a problem in its own.

3.3.3 Test:

Again we ran the same test on BUET's official websites. Figure-5 and figure-6 show the results of the test.

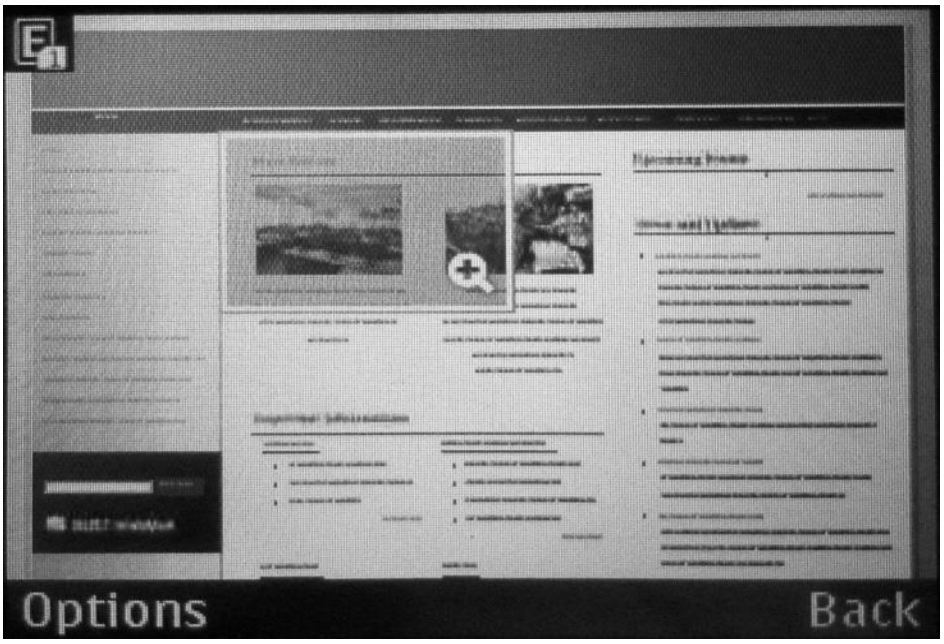

Figure 3.5 Official Website of BUET on Nokia Asha 205.

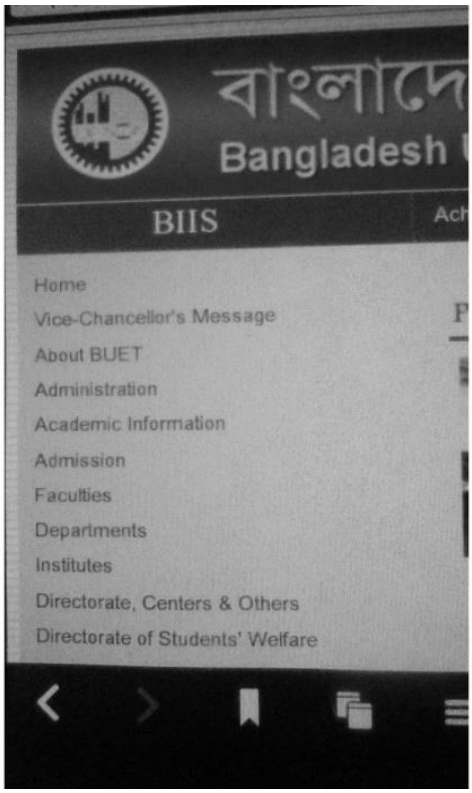

Figure 3.6 Official Website of BUET on Nokia N8.

3.3.4 Observation:

Again if we look carefully at figure 3.5 and figure 3.6 and compare it with figure 3.2, we will see the same kind of problems as we saw in figure 2.3 and figure 3.4. This time in Nokia N8 though text font on the left hand size is big enough that it's visible but the amount of horizontal scrolling is unbearable.

3.3.5 Conclusion:

As we observed that user interface on small screen does not remain same to that on desktop computer. If we don't address this problem, we can lose a huge number of visitors. Our online presence might lose its effectiveness.

3.4 Solutions:

One successful approach that has been followed is that of making a separate website for devices with small screen sizes and mobility but with that comes sacrifices and hard choices to make. As the following list shows:

- Development:

Starting from the system analysis to deployment and testing, development team engages with a whole new project. They try to optimize the website for mobile users by minimizing images, links, contents, flashes. Most important information is put affront.

- Cost:

  The cost of separate website doubles as compared to one website that works for all websites.

- Maintenance:

  There is a need of separate teams for maintenance in both versions of the websites.

- Consistency issues:

  Consistency issues arise when an update must be made. When information on one website is updated it leaves the same piece of information on the other version outdated.

- Hard to Change:

  Like other above problems, change has to be made at two places. This doubles the efforts required.

- Hardware, software and bandwidth cost:

  A separate website obviously requires a separate set of hardware, software and bandwidth. The owners of the website have to bear the additional price of all these equipment.

Now what we are suggesting is that to make a centralize website and tailor the technology in such a way which customizes interfaces according to the device while achieving the usability and user experience goals at same time. Our contribution is finding out how to do it.

3.5 Problem Definition:

From the above discussion it becomes clearer that traditional approach won't succeed as far as online presence is concerned. Making separate website generates additional problems. Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them. Therefore, our response is to make one central website and properly tailor the technology in such a way that performs well

in both domains (mobile devices and desktop). This elegancy can be achieved with what we call auto-response user interface. In the light of above arguments we define our problem as such:

"*Devising a way such that components in the web based user interface like images, forms, text, dialogues etc. change and adapt itself to platform, dimensions, resolutions and orientations of the device from which that particular website is being accessed*."

Chapter-4 Methodology

In the previous work regarding webpage display in screen of diverse sizes we concluded that this issue can be resolved by adopting auto-response user interface, whose properties have been mentioned already. According to our solution a web page will automatically change to adapt to the device from which a user tries to make a request for a particular web page. If a request comes from a desktop PC then full-fledged web page will be served, on the other hand, if a request is from a device with a small screen then serving all the contents will generate more issues as have been mentioned in previous posts. Therefore, some parts of the webpage must be truncated. Now in this chapter we will determine how to determine which part of the webpage is important in order to retain and which is not in order to drop. We can do this by the following two approaches:

4.1 Developer Side Implementations:

The only solution we have come up with on behalf of developer is to use client side scripting languages like JavaScript and CSS3 to identify device nature and serve the contents accordingly with some general guidelines for determining important segments of the whole page. This is what we call auto-response user interface. Its usefulness is in the fact that it is not dependent.

4.1.1 JavaScript:

With proper use of JavaScript responsiveness can be obtained. JavaScript `document` object has properties and methods by which we can traverse the DOM tree of the HTML document and make necessary changes to a specific node. The document property `innerHTML` and method `document.getElementById("para1")` can accomplish wonderful things.

The methodology work like this: we can obtain screen resolution by using `window.screen.availHeight`, `window.screen.availWidth`, `window.screen.width`, `colorDepth`, `availWidth` and `window.screen.height` or jQuery's `$document.height` and `$(document).width`. For different screen resolutions we make necessary modification to the webpage by using CSS methods of the `document` object. A demo list below left (CSS) and right (javascript) is given here [44]:

```
padding-top (paddingTop)
padding-right (paddingRight)
padding-bottom (paddingBottom)
padding-left (paddingLeft)
position (position)
Text-align (textAlign)
Text-decoration (textDecoration)
Text-indent (textIndent)
Text-transform (textTransform)
```

```
Top (top)
vertical-align (verticalAlign)
Visibility (visibility)
white-space (whiteSpace)
Width (width)
word-spacing (wordSpacing)
z-index (zIndex)
```

Besides, using jQuery's methods for animation enable us to show and hide certain features based on the user.

4.1.2 CSS:

CSS media queries are very robust for making user interface context aware. CSS is being supported by almost all modern major browsers. Desktop browsers such as Safari 3+, Chrome, Firefox 3.5+, and Opera 7+ all natively parse media queries. In mobile devices browsers such as Opera Mobile and mobile WebKit media queries are supported.

From CSS 2.1 onward, style sheets possess some measure of device awareness through media types. W3C created media queries as part of the CSS3 specification, improving upon the promise of media types. A media query allows us to target not only certain device classes, but to actually inspect the physical characteristics of the device rendering our work. For example, following the recent rise of mobile WebKit, media queries became a popular client-side technique for delivering a tailored style sheet to the iPhone, Android phones, and their likes. To do so, we could incorporate a query into a linked style sheet's media attribute:

```
<link rel="stylesheet" type="text/css"
  media="screen" and (max-device-width: 600px)" href="change.css" />
```

In above CSS code two important things should be noted: media type (`screen`), and *media feature* (`max-device-width`) to inspect, followed by the target value (`480px`). That means if horizontal resolution (`max-device-width`) is equal to or less than `480px`, then the device will load `change.css`. Otherwise, the `link` is ignored altogether. This provides us with opportunity of separately formatting the contents of our website for small screens. Moreover, we can add other conditions connected by and keyword:

```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px) and (resolution:
163dpi)"
  href="shetland.css" />
```

This works more effectively as compared to one mentioned above. We can also include queries regarding media in CSS other than links by using `@media` rule:

```
@media screen and (max-device-width: 480px) {
  .column {
    float: none;
  }
}
```

Similar is `@import` directive:

```
@import url("change.css") screen and (max-device-width: 480px);
```

But a responsive user interface can not only be achieved with fluid layout but it also allows us to practice some incredibly precise fine-tuning as a web page reshapes itself. To comply with Fitt's Law, we can increase the target area on links for smaller screens, selectively show or hide elements that might enhance a page's navigation. We can even practice responsive typesetting to gradually alter the size of text, optimizing the reading experience for the display providing it.

4.2 User Side Implementations:

Our answer to user side solution is developing a general algorithm which is capable of determining which part of the web page is more important compared to another. The part more important gets displayed while others are being retained. Our final product in the form of our solution is in the form of a proxy server which implements such an algorithm. The proposed architecture of the solution is shown in the figure. The browser sends an http request to the proxy server. Proxy server asks for the same page from the web server. Web server returns the web page to proxy server. Proxy server does the necessary modification in the page based on the proposed algorithm and user's settings then send the modified version of the page to web browser for viewing.

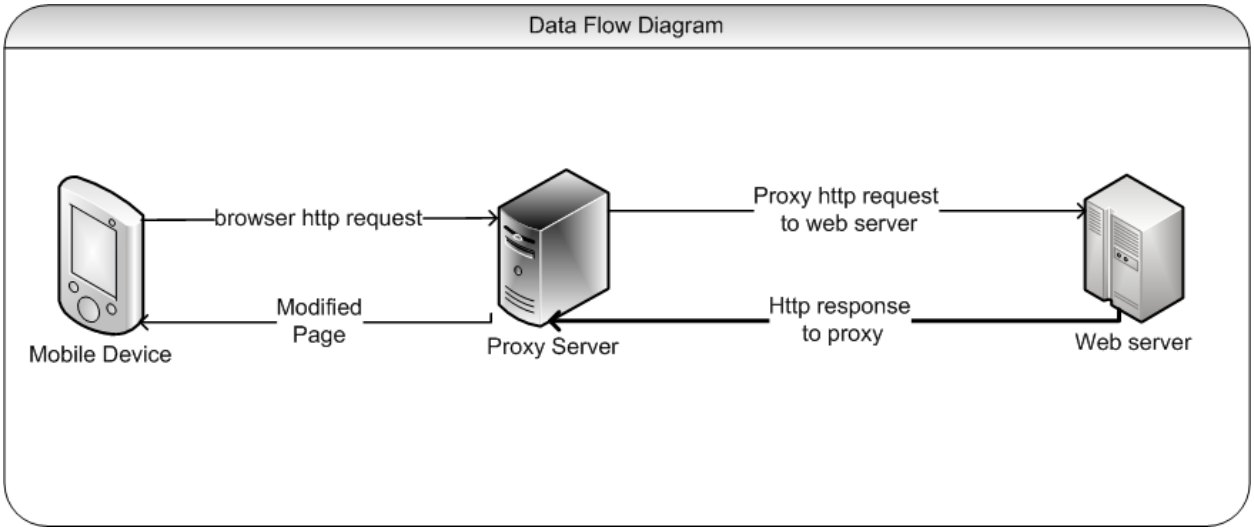

Figure 4.1 the architecture of how the auto response user interface implemented.

There were two effective algorithms for making changes to a web page

We will elaborate on this.

To distinguish different information in a web page and determine their relative importance, we first need to segment a web page into a set of blocks. Here we need to understand what block importance is:

4.2.1 Algorithm:

Before going into details of the algorithm let us shed some light on two basic ideas we use in the supposed algorithm.

4.2.1.1 Block Importance:

At first let us differentiate between importance and attention. Attention is a neurobiological concept. It means turning mental focus to specific one object while ignoring others. At the first sight of a web page, attention may be caught by an image with bright color or animations in advertisement, but generally such an object is not the important part of the page.

Ruihua Song et al. [4] discusses block importance from objective point of view, we do the same. In Figure-1 Yahoo Inc. home page is divided into different segments. Its relative importance is different based on the visitor but by looking at the figure, we can easily understand that login portion, search and services portions are more important as compared to others. This is called block importance.

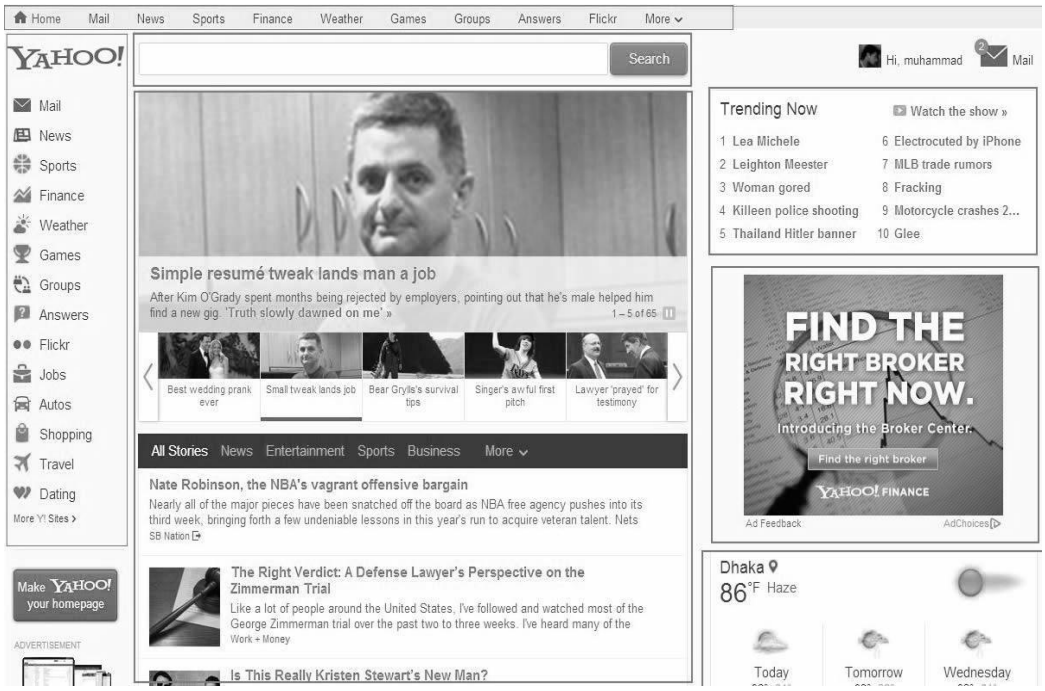

Figure 4.2 Yahoo Inc. home page showing different blocks.

Relative block importance has already been discussed in detail in chapter 2. Block importance can be determined by considering the following factors:

- Location
- content
- Navigational Efficiency
- Empathy
- Aesthetics

Suppose there are n blocks in a web page. Each block has a weight w. Then the set of weights is:

$W = \{w_1, w_2, \ldots\ldots\ldots w_n\}$ while wn denotes the block importance of nth block.

Each w can be calculated by looping through each block and incrementing value for a specific block if it possess a specific property otherwise decrementing it. A demo of it is given here:

```
for(i=0; i<number_of_blocks;i++){
if(location_central) w[i]++;
else w[i]--;
}
```

Now those blocks will be discarded whose score is in negative or zero if client's platform is a mobile device otherwise all of the blocks will be displayed without truncating any.

To determine relative block importance, first we need to divide the page into blocks. Different segmentation algorithms are used to divide page into blocks. We discuss most important of them here.

4.2.1.2 Segmentation:

There are several kinds of methods for web page segmentation. The most popular ones are DOM-based segmentation and Vision-based Page Segmentation (VIPS).

DOM-Base Segmentation:

DOM is document object model. We divide a document based on the html tags. Html tags are of five types:

- The entire document is a document node
- Every HTML element is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

We divide document in segments based on the above nodes as shown in the Figure-2. Each segment can be accessed, changed or deleted.

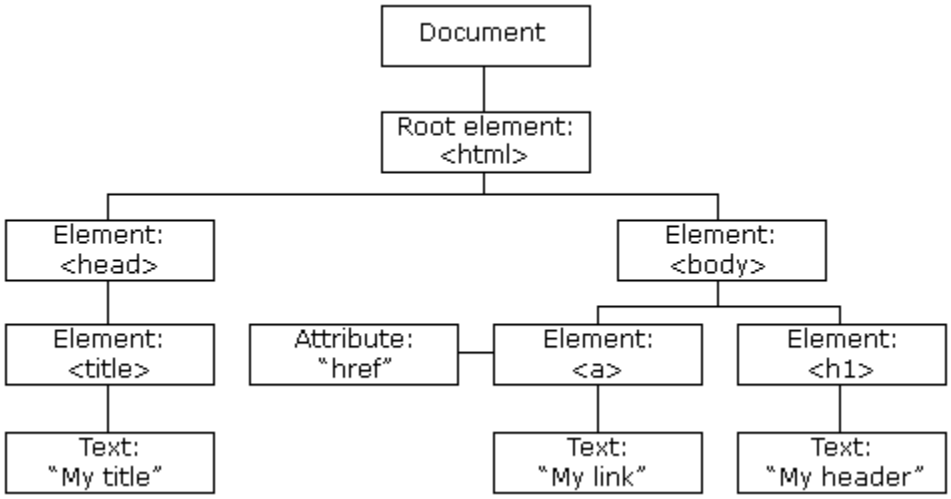

Figure 4.3 DOM based segmentation.

Vision-based Page Segmentation (VIPS):

Compared with the above segmentation, Vision-based page segmentation (VIPS) is better recognized by granularity and coherent semantic aggregation. VIPS makes full use of page layout features such as font, color and size. It first extracts all the suitable nodes from the HTML DOM tree, and then finds the separators between these nodes. Based on these separators, the semantic tree of the web page is constructed. A value called degree of coherence (DOC) is assigned for each node to indicate how coherent it is. Consequently, VIPS can efficiently keep related content together while separating semantically different blocks from each other. Each block in VIPS is represented as a node in a tree. The root is the whole page; inner nodes are the top level coarser blocks, and all leaf nodes consist of a flat segmentation of a web page.

We also worked on VIPS algorithm for customizing a web page for viewing on small screen. VIPS algorithm is shown in the figure.

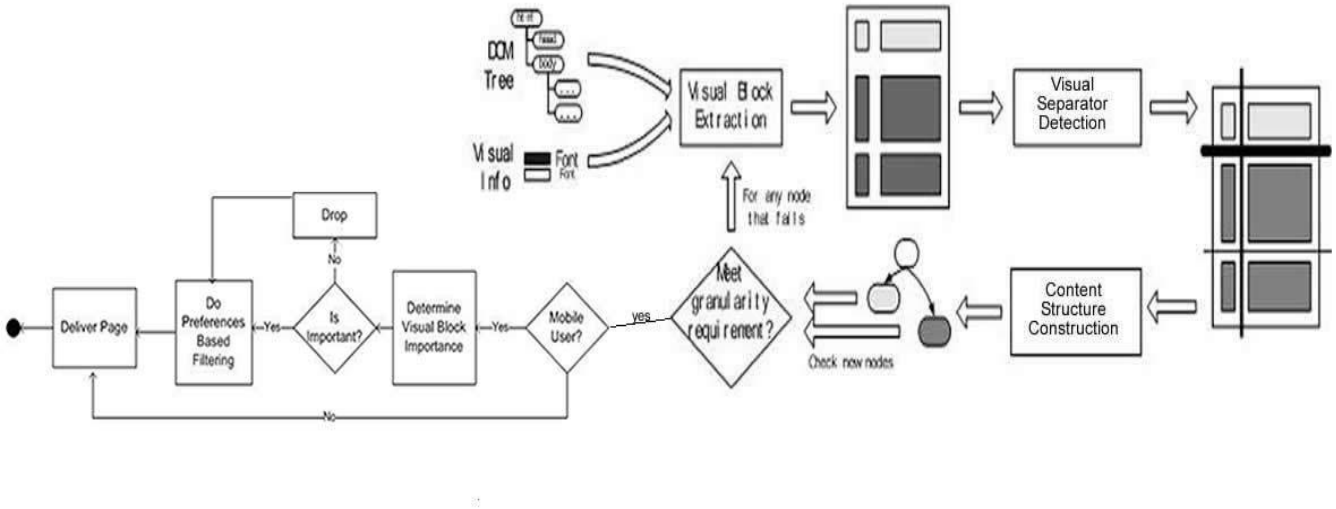

Figure 4.4 Modified VIPS algorithm.

VIPS starts with a DOM tree combined with visual inputs like images, fonts etc. visual blocks from the DOM tree are extracted and then separated. At the end we get another tree formed based on visual blocks. Once we have this tree we can delete any segment we want. That means we can change the web page according to the requirements of the small screens. Unfortunately, we did not work with this algorithm despite its robustness and elegance. In the course of our study we found that its implementation is difficult based on our knowledge of concurrent languages especially javascript.

A DOM based segmentation and customization algorithm which we used is shown in figure. This algorithm we have implemented in our proxy server which we will discuss in detail in next chapter. The proxy server once gets the HTML document from the web server, DOM tree of the HTML document is constructed. Then it is traversed recursively and is partitioned into segments. Segments are assigned their weights based on the criteria mentioned earlier in this chapter. Each segment is then parsed and common patterns are searched. Based on the user settings changes are made accordingly. At last document is formatted and delivered to the browser. User now can view web page optimized for that specific browser.

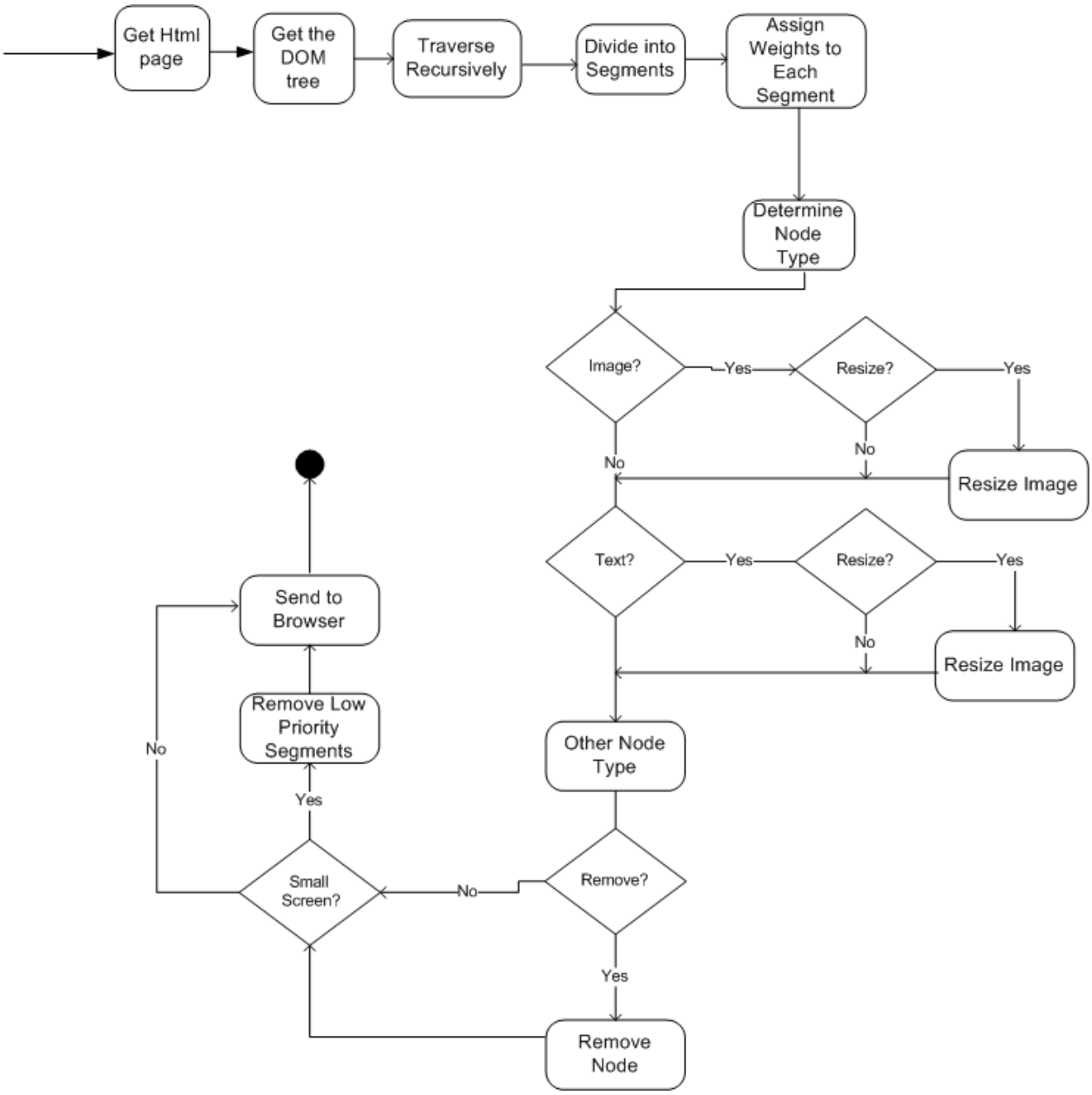

Figure 4.5 Segmentation algorithm for web-page.

4.2.1.3 Pseudocode:

```
1. PROCEDURE makeResponsive()
2. {
3. htmlStream = getHTMLDocument();
4. domTree = getDOMtree(htmlStream);
5. segment=doSegmentation(domTree);
6. assignWeight(segment);
```

```
7. node=getNode(segment)
8. modifyByPreferences(node);
9. if(user_mobile)
10.            filterSegments();
11.      end if
12.      streamToBrowser();
13.      }
```

A little elaboration of the pseudocode is necessary here in order to understand the basic mechanism. It all starts with a user request for the webpage. The request is intercepted by the application and parameters of the request are read. Based on the read parameters application opens a socket connection with web server and start reading the HTML file in the form of HTML stream. HTML stream is then converted to DOM tree by using java's DOM API's. Once we have a complete tree of the HTML document then segmentation module make different segments of the document. Each segment represents a block in the HTML page. Based on the aforementioned parameters each segment is assigned weight. The criterion of assigning weights has already been explained. Weight assigning helps us to determine relative importance of the blocks. A block with highest weight value is the most important of all. Therefore, such a block must be appear to the user in the results. As there are some user's preferences also set. Every node is filtered against those preferences in line 8. And finally application determines whether a given device is mobile. If so then all the filtering is done and resultant contents are streamed to user's application where from the request emerged.

The pseudocode above represents the full set of actions. Each line represents a module to whom a specific task is delegated. In implementation such a module is not necessarily a function or method but a combination of it.

4.3 Performance Analysis:

We can see from the figure that algorithm follows a linear path of execution. When DOM tree is built algorithm makes its way to recursive traversing the DOM tree. For each node the tree algorithm calls itself. At some points there are loops. Therefore, time complexity of the algorithm is no more than O(n2). The greater amount of latency is involved with reading stream from the web server which is dependent on the bandwidth of the connection. Therefore, cumulative execution time is the summation of both.

Execution Time $T = O(n^2) + \text{Latency}$

Memory consumption is primarily dependent on how much is a particular page heavier. Larger the size of the page, higher the memory requirements go. Unfortunately, it is not easier to develop a mathematical model for such a scenario.

Chapter-5 Implementations

Theoretical solution is not important if it's not feasible. Though theoretically we proved that the approach we are following can bridge the gap between screen variations. In this chapter we discuss the implementation details of the hypothesized system. We will also what implementation options we tried and we abandoned one and stuck to the other?

We figured out two client side approaches that can work for the problem at hand.

- Developing extension for the browser to do the required job.
- Develop client side application to accomplish responsiveness.

Each of them we discuss in detail.

5.1 Extension:

The idea was to develop extensions for the major browsers. Extensions allow user to add functionality to browser without diving deeply into native code. Extension can be created for browser with those core technologies that we're already familiar with from web development: HTML, CSS, and JavaScript.

We need four necessary components to create a an extension.

- `manifest.json`: The manifest is nothing more than a JSON-formatted table of contents, containing properties like extension's name and description, its version number, and so on.
- Javascript file: This file is for doing the core job. If we want to add functionality to web browser, we do that in this file.
- Html file: Html file provides the user interface of the extension and can have settings and option for the user.
- Icon: Icon is a 32*32 png image for displaying in browser action. This help create an aesthetically pleasing action button for the browser.

To do any sort of changes to a random website, we need two kinds of content scripts and background script. Content scripts are JavaScript files that run in the context of web pages. By using the standard Document Object Model (DOM), they can read details of the web pages the browser visits, or make changes to them.

Here are some examples of what content scripts can do:

- Find unlinked URLs in web pages and convert them into hyperlinks
- Increase the font size to make text more legible
- Find and process micro format data in the DOM

However, content scripts have some limitations. They cannot:

- Use browser native APIs
- Use variables or functions defined by their extension's pages
- Use variables or functions defined by web pages or by other content scripts

Background scripts are where the main logic is defined while content scripts communicate with background script. We didn't chose extension as product of our implementation because of the following two reasons.

1) First we developed some JavaScript functions and tested them with static pages. Soon we found that making changes to webpages with JavaScript was a tedious job. We wrote some function for DOM tree traversal but its results were very poor.

```
86
87  /**
88   * Extracts the content of the html page
89   */
90
91  function  extractContent(iNode) {
92
93      if (child) {
94          if (mTree == null)
95              mTree = iNode;
96          counter = counter + 1;
97          extract(iNode, mTree);
98          document.write("After extract call");
99          address = new String(null);
00          var site = document.URL;
01          var newTree ;
02          var getNext = new Boolean(true);
03
04          while((linkToAppend !=null) && (linkToAppend != address) && getNext )
05          {
06              address = linkToAppend;
07              linkToAppend = null;
08              site = address;
09                 newTree = document;
10                 linkToAppend = null;
11                  counter++;
12                  extract(newTree,newTree);
13
14                  appendDocument(newTree, mTree);
15          }
16  }
17
18  }
```

Figure 5.1 extractContent function in javascript.

The function shown in figure was not working properly. Because of limitation of the time we decided to migrate to Java because APIs of java are very robust, well-documented, and clean. There are many third party APIs available for manipulating HTML documents.

2) The second reason was the limitations of the extension mentioned earlier. As extension don't have full control over HTML pages.

5.2 Proxy:

The proxy server advocates the implementation of the algorithm. We call it Meddler 1.0. It should be noted that Meddler is not a state of the art piece of software built with software engineering principles in mind. Instead, it shows our developed algorithm in action. We describe some key aspects of the Meddler rest of the chapter.

5.2.1 User Interface (UI):

There is not much complexity in the user interface. The core job is done behind the scene and there is not much to show on user interface except some options for the users as shown in figure. These options are not the exhausted list. We have shown only a few one in order exhibit the implementation of the algorithm. This can be customized even further for Wi-Fi networks, or Ethernets. By checking any option filter for that specific task will be activated. There options for removing adds, which minimizes the burden on host's resources. Resizing of images and fonts provide context aware experience to the user. Interface elements change and adapt to the local environment. Some smartphones with limited memory and processing power have difficulty running flash files in the browsers. Therefore, there is an option for the user to disable any flash files in the web page. Scripts on the other hand can be costly for some devices; therefore, one can add or remove at one's convenience. Most of the personalization options are self-explanatory and does not require any further elaboration. Once a change is made, by clicking save button the change is permanently committed unless changed otherwise. The personalization parameters are saved in settings.ini file. Meddler reads its settings priorities from that file. One can make changes directly to the file as well.

Figure shows an interface for the settings of proxy server. These settings are required for the Meddler to work properly. Figure is elaborated further under the topic proxy settings later in this chapter.

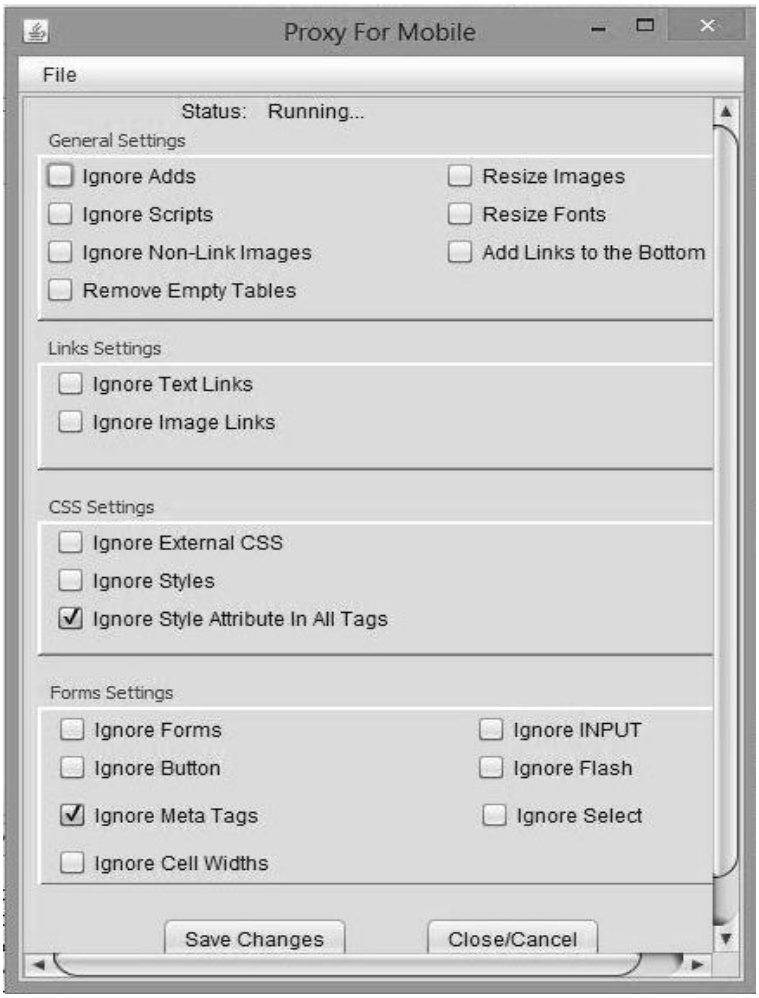

Figure 5.2 shows the user's preferences window.

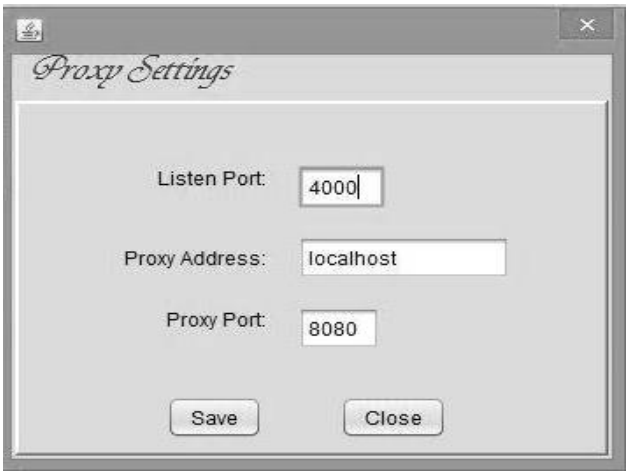

Figure 5.3 shows settings for proxy.

4.2.2 Code Details:

There are four packages in Meddler.

- GUI: It combines classes that deal with user interface of the meddler.
- http: Classes which do http related functions are in this package.
- proxy.content: Here lies the core of the Meddler. All important classes regarding filtering, size modification etc. lies in this package.
- settings: This package contains two classes for user's and proxy setting.

Table 5.1 gives an overview of the classes used in the Meddler.

Name of the Class	Description
Class MainFrame	Renders the user with interface and display the status of the meddler.
Class ProxySettingsPanel	Providing user interface for proxy settings like changing port numbers or IP addresses.
Class HttpMetaData	Acts as a container for storing http headers.
Class HttpResponses	Holds some common http error responses returned by the server.
Class HttpStream	Handling the http stream coming from the server.
Class AdsServers	Responsible for managing the list of ads servers. Ads server list permanently store in a file. This class reads that particular file and searches an address in that manually created database of ads servers.
Class ContentExtractor	The primary class which performs major functions of the Meddler such as locating and removing certain nodes.
Class ContentErrorHandler	A custom written error handler class which implements org.xml.sax.ErrorHandler.
Class ImageSizeModifier	Do context aware image resizing. When an image is found it first checks whether resize image is checked by the user or not.
Class SizeModifier	This class is responsible for necessary Font resizing.
Class SmallStuff	This class contains some important miscellaneous methods. Most of its methods do the parsing job.
Class ProxySettings	This class contains variables and constants necessary for keeping values necessary for Meddler to function properly. It also contains methods for saving and reading settings to and from the file.
Class Settings	This class manages general user based settings needed to help build customized web page. It also has methods for writing and reading to and from settings.ini file.

4.2.3 Proxy Settings:

For the Meddler to work, it should be run. Once the Meddler is running, it is ready to accept connections from the client browser. Before any connection is established there are some minor settings that have to be done. To set up proxy follow the following steps:

Step-1: Any browser can be set up to connect to proxy but we use Mozilla Firefox as an example. Open Firefox and go to Tools->Options.

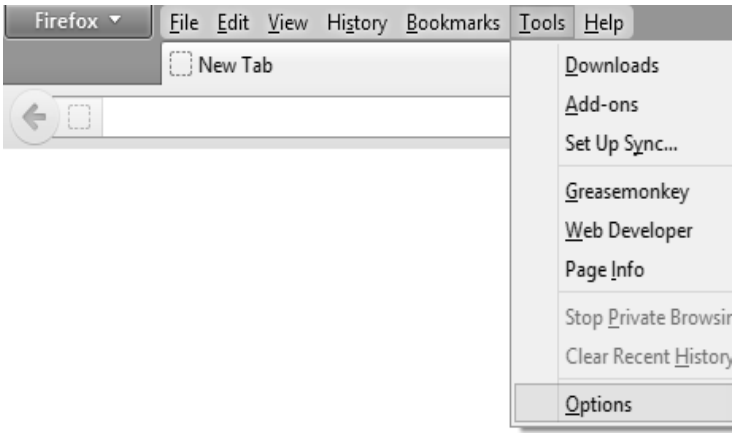

Figure shows Firefox configuration step-1.5.4

Step-2: Once you click the option a pop up window will appear as shown in the figure. Click on Advanced Tab then click on Settings next to Connection as shown in the figure.

Figure 5.5 shows Firefox configuration step-2.

Step-3: Enter the following information when next window appears as shown in figure and press OK.

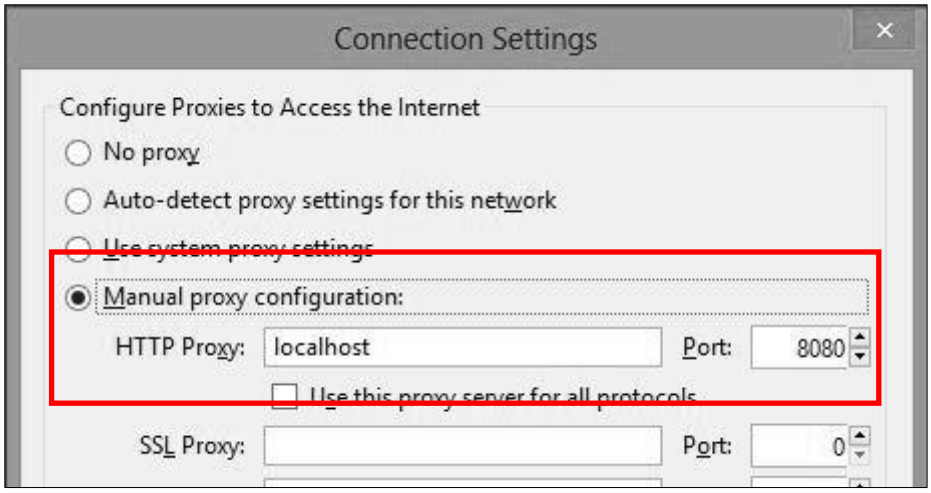

Figure 5.6 shows Firefox configuration step-3.

Step-4: The client side is done. Now run the proxy and go to File->Proxy Settings as shown in the figure and click on it.

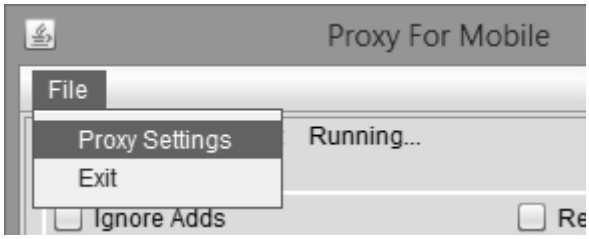

Figure 5.6 shows Firefox configuration step-4.

Step-5: A window will appear showing the very basic settings for the Meddler as shown in the figure. The port number and proxy address should be same to that of client. By clicking Save new settings will be written to settings.ini file. Proxy settings can also be changed by directly editing this file. Now if we want to browse any website then system should work perfectly.

5.2.4 Scalability:

The idea of Meddler can be scaled from one device to multiple network level connections. Appropriate changes and optimization is required to use it for such purpose. For instance one can install it on a Wi-Fi network. A special module determining the nature of device can be used to modify or not modify a specific web page. Besides, functions can be added to Meddler to work like any other proxy servers. Meddler can be customized for any job. Specifically, cellular companies can customize it to provide featured responsive user experience to customers by adding common functions of auto-response user interface and removing personalization.

5.2.5 Challenges:

This approach of providing auto-response user interface experience is not clear from some limitations. As we describe them so:

- The biggest challenge that we faced was inherent latency by involving a middleman between web browser and web server. The latency mainly comes from connection establishment between browser and proxy server and between proxy server and web server. Another factor that adds to the latency is the amount processing required for parsing HTML document and making appropriate changes. Latency adversely affects responsiveness.
- As user application requires user intervention to install. Most users either don't know about it or does not bother to go for it.
- Inexperienced users might get confused with the setting of browsers and application.

Chapter-6 Future Works and Conclusions

In the future we are planning to improve the block importance determination part of the algorithm. It needs more extensive study to know what is important and what is not. As we have already mentioned there is some inherit latency involved while accessing webpages. We will introduce caching to cater for latency. As categories of devices are increasing each coming day, to accommodate any future application we will try to scale our proxy to the gateway level.

The whole study encompasses a two semesters' timeline and went through a process of evolution. After a lot of observations, study and some quantifiable tests, we reached to a point to recognize what exactly we are looking for. There were many possible solutions and we chose the one we could possibly implement. To obtain responsiveness, the concept of proxy is utilized, which if properly tinkered works well for the desired purposed but with the expense of some additional sacrifices like latency, user intervention etc.

References

[1] Dave Gehrke, Efraim Turban "*Determinants of Successful Website Design: Relative Importance and Recommendations for Effectiveness*", Proceedings of the 32nd Hawaii International Conference on System Sciences – 1999.

[2] Ruihua Song, Haifeng Liu, Ji-Rong Wen, Wei-Ying Ma "*Learning Important Models for Web Page Blocks based on Layout and Content Analysis*", SIGKDD Explorations. Volume 6,Issue 2 – Page 22.

[3] K.S.Kuppusamy and G.Aghila, "*A Personalized Web Page Content Filtering Model Based On Segmentation*", International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.1, January 2012.

[4] Ruihua Song, Haifeng Liu, Ji-Rong Wen, Wei-Ying Ma, "*Learning Block Importance Models for Web Pages*".

[5] Hongxiu Li, Reima Suomi, "*A Proposed Scale for Measuring E-service Quality*" International Journal of u- and e-Service, Science and Technology Vol. 2, No. 1, 2009.

[6] Zhilin Yang, Robin T. Peterson, Shaohan Cai "*Services quality dimensions of Internet retailing: an exploratory analysis*", Journal O F Services Marketing , VOL . 17 NO . 72003 , p p . 685 - 700 , # MCBUP Limited, 0887- 6045, DOI10 .1108 /08876040310501241.

[7] Flavián Carlos, Gurrea Raquel and Orús Carlos, "*The Relevance of Web Design for the Website Success: A heuristic analysis*", CollECTeR Iberoamérica 2008.

[8] P YesuRaju, P KiranSree, "*A Language Independent Web Data Extraction Using Vision Based Page Segmentation Algorithm*", ISSN: 2319 - 1163 Volume:2 Issue:4.

[9] Gagan Preet Kaur, Usvir Kaur, Dheerendra Singh"*A Review: Image Extraction with Weighted Page Rank using Partial Tree Alignment Algorithm*", International Journal of Computer Science and Mobile Computing A Monthly Journal of Computer Science and Information Technology ISSN 2320–088X IJCSMC, Vol. 2, Issue. 5, May 2013, pg.118 – 122.

[10] Suhit Gupta, Gail Kaiser, David Neistadt, Peter Grimm "*DOM-based Content Extraction of HTML Documents*".

[11] Nipan Maniar , Emily Bennett, Steve Hand, George Allan, "*The effect of mobile phone screen size on video based learning*"JOURNAL OF SOFTWARE, VOL. 3, NO. 4, APRIL 2008.

[12] G. Poonkuzhali, K.Thiagarajan, K.Sarukesi, "*Elimination of Redundant Links in Web Pages – Mathematical Approach*", World Academy of Science, Engineering and Technology 28 2009

[13] Nikolaos Pappas, Georgios Katsimpras, Efstathios Stamatatos ,"*Extracting Informative Textual Parts from Web Pages Containing User-Generated Content*".

[14] R. Baik , A. Boumaraf , and H. Hocini,"*Indexing relevant blocks in HTML Documents*".

[15] Srinivas Vadrev, Fatih Gelgi, and Hasan Davulcu, "*Information Extraction from Web Pages Using Presentation Regularities and Domain Knowledge*", WorldWide Web (2007) 10:157–179 DOI 10.1007/s11280-007-0021-1

[16] Srinivas Vadrevu, Fatih Gelgi, and Hasan Davulcu, "*Semantic Partitioning of Web pages*"

[17] Steven Dewolfe, "*Small Screen; Big Problems: The Contributions And Promise (Past, Present, & Future) Of Hci In Providing Usable Mobile Interfaces*".

[18] Deng Cai , Shipeng Yu, Ji-Rong Wen and Wei-Ying Ma ,"*VIPS: a Vision-based Page Segmentation Algorithm*", Microsoft Research Microsoft Corporation One Microsoft Way Redmond, WA 98052.

[19] Elgin Akpınar and Yeliz Yesilada, "*Vision Based Page Segmentation: Extended and improved Algorithm*".
[20] Wei Liu, Xiaofeng Meng and Weiyi Meng ,"*Visionbased Web Data Records Extraction*".
[21] Kayla Knight, "*Responsive Web Design: What it is and how to use it*". Published on January 12, 2011 in Smashing Magzine.
[22] http://filamentgroup.com/examples/responsive-images/ last visited in August 2013.
[23] http://alistapart.com/article/responsive-web-design last visited in August 2013.
[24] http://www.iawriter.com/ipad/ last visited in August 2013.
[25] http://ia.net/blog/bringing-responsiveness-the-app-world/ last visited in August 2013.
[26] Fariza Fauzi, Jer-Lang Hong and Mohammed Belkhatir, "*Webpage Segmentation for Extracting Images and TheirSurrounding Contextual Information*", http://www.academia.edu/500853/Webpage_Segmentation_for_Extracting_Images_and_Their_Surrounding_Contextual_Information
[27] Le Liu,"*The Design and Implementation of a visual Segmentation-Based Data Extraction Algorithm on Web Pages*", Graduate project technical report submitted to the Faculty of the Department of Computing and Mathematical Sciences, Texas A&M University-Corpus Christi summer 2006.
[28] Kovacevic, M., Diligenti, M., Gori, M. and Milutinovic, V., "*Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification*", in the proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan, December, 200.
[29] Liu, H., Xie, X., Ma, W.-Y. and Zhang, H.-J., "*Automatic Browsing of Large Pictures on Mobile Devices*", In The Proceedings Of 11th ACM International Conference on Multimedia, Berkeley, CA, USA, Nov. 2003.
[30] Paul Resnick and Jim Miller. "*PICS: Internet Access Controls Without Censorship*" Communications of the AGM, 39(10):87-93, 1996.
[31] Tristan d'Estrée Sterk ,"*Using Actuated Tensegrity Structures to Produce a Responsive Architecture*". http://fishtnk.com/responsivearchitecture/wp-content/uploads/2013/01/sterkACADIA_03.pdf
[32] Brent Bridgeman, Mary Louise Lennon and Altamese Jackenthal, "*Effects of Screen Size, Screen Resolution, and Display Rate on Computer-Based Test Performance*".
[33] http://sender11.typepad.com/sender11/2008/04/mobile-screen-s.html
[34] http://www.apple.com/iphone/specs.html last visited on August 25, 2013.
[35] http://www.sonymobile.com/global-en/products/phones/xperia-z/specifications/ last visited on August 25, 2013.
[36] http://us.blackberry.com/smartphones/blackberry-z10/specifications.html last visited on August 25, 2013.
[37] http://www.htc.com/www/smartphones/htc-one/#specs last visited on August 25, 2013.
[38] http://www.samsung.com/global/microsite/galaxys4/ last visited on August 25, 2013.
[39] http://www.nokia.com/us-en/phones/phone/lumia1020/specifications/ last visited on August 25, 2013.
[40] http://www.iut-dhaka.edu/ last visited on May 17, 2013.
[41] http://www.buet.ac.bd/ last visited on May 17, 2013.
[42] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale, "Human-Computer Interaction", Prentice Hall, 2004.
[43] Yvonne Rogers, Helen Sharp, and Jenny Preece, "*INTERACTION' DESIGN beyond human-computer interaction*", John Wiley & Sons, Inc.