

**A JAVA APPLET FOR GENERATING GRAPHS  
FROM A UNIX FILE**

By

**SAI PRASANNA BATTINA**

Bachelor of Engineering

Osmania University


Hyderabad, India

1992

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
**MASTER OF SCIENCE**  
December, 1997

A JAVA APPLLET FOR GENERATING GRAPHS  
FROM A UNIX FILE

Thesis Approved:



Thesis Adviser



H. Lu



Dean of the Graduate College

## **PREFACE**

The scope of this study is to develop a Web-based application called the Remote File Access and Graphing Tool (RFAGT). It consists of a server-side program for building a data file on the Unix Web server, a CGI program interfacing this server-side application with a Java applet on the same server, the Java applet that accepts data from the data file passed by the CGI program and draws a 2D graph on the Web client's browser, and an HTML file that forms the user interface on the client's browser. The method of implementation includes several WWW technologies.

RFAGT converts a data file to tabular and graphical forms. It is implemented on a Linux system using C++ and Java. The data file is a list of (keyword, attribute) pairs. An user can use the keyword to access the associated attribute values. This data is accessible from any Web browser

## ACKNOWLEDGMENTS

I wish to express my sincere gratitude and appreciation to my major advisor, Dr. K. M. George for his intelligent supervision, guidance, support, and inspiration. This work would not have been complete but for his expert advice through its different stages. My gratitude to the Dr. J. P. Chandler for his encouragement and support both as the graduate adviser and as a committee member. My sincere appreciation extends to Dr. H. Lu, for serving on my committee. I also wish to express my gratitude to Dr. Nick Street for his help in the final phase of this endeavor. Their guidance, assistance, and encouragement are invaluable.

Thanks also go to my family and friends for their invaluable support and encouragement throughout this process.

## TABLE OF CONTENTS

Chapter	Page
I INTRODUCTION.....	1
II LITERATURE REVIEW .....	3
2.1 Internet.....	3
2.2 Client/Server Model.....	5
2.3 TCP/IP.....	5
2.4 World Wide Web.....	7
2.5 SGML.....	8
2.6 HTML.....	10
2.7 HTTP.....	10
III COMMON GATEWAY INTERFACE (CGI).....	13
3.1 Introduction .....	13
3.2 Communication between the Server and the Gateway Program .....	15
3.3 Communication from the Gateway Program to the Server .....	16
IV JAVA .....	17
4.1 Security.....	21
4.2 Advantages.....	23
4.3 Disadvantages.....	24
V REMOTE FILE ACCESS AND GRAPHING TOOL .....	25
5.1 Implementation Model.....	25
5.2 Implementation .....	28
5.3 Platform and Environment.....	33
VI CONCLUSIONS .....	35
REFERENCES.....	36
APPENDICES.....	38

APPENDIX - A GLOSSARY AND TRADEMARK INFORMATION.....	39
APPENDIX - B SOME POPULAR INTERNET RESOURCES.....	40
APPENDIX - C PROGRAM LISTINGS .....	41

## LIST OF FIGURES

Figure	Page
1. Flow of information between the client and the server-side gateway program ...	14
2. Execution of a Java applet .....	19
3. Security Model of an applet showing four phases related to the applet's life cycle .....	22
4. System Architecture of RFAGT.....	27
5. Web page for the user input.....	29
6. Web page after the user input .....	30
7. Graph drawn by RFAGT based on the user input.....	32

## I. INTRODUCTION

In this age of information explosion, the Internet is playing an important role in the areas of telecommunications, business, commerce, medicine, education, politics, transportation, and entertainment. It is making itself virtually omnipresent and indispensable to some fields of scientific research and to the academia. It is touching the lives of the common people in remote lands, giving a new hue to the meaning of technology.

One of the areas in which the Internet is proving extremely useful is the accessing of data and information remotely. Remote databases are becoming more and more popular with more and more companies and institutions realizing their effectiveness and potential. Apart from easy access and efficiency, the remote access offers an opportunity to solve a problem collectively [1].

An Internet information retrieval tool, the World Wide Web, revolutionized remote access through hypermedia computer interfaces [2]. It also makes less cumbersome the dynamic query and retrieval interfaces between the Web clients and data sources. Databases, geographic information systems, and modeling systems form a few categories that are actively exploiting the advantage of such an interface [2]. Also, more recently, the medical diagnosis and analysis, robotics, and astronomy are some of the areas using the Web to extend their reach, productivity, and efficiency. Work in these areas related to the Web has actually led to manipulating the real world objects through the Web [1].

Applications available on the Web can be shared by several users. Because of this, several applications are being developed. The purpose of this thesis is to develop a



WWW application that displays the data available in a specific format on a remote server, in tabular and graphical form on a Web browser.

The data is contained in a file on a Web server. This data file is created, maintained, and modified by a program on the same server. All the data in the file is arranged in a specific format. And each set of data is referred to by a keyword. This data must be accessible to any user who has access to the Internet and who has knowledge of the keyword(s) in the data file. In addition to displaying the data, the application must draw a graph on the user's Web browser.

The organization of this thesis is as follows: Chapter II provides a review of the relevant literature pertaining to the Internet, Internet protocols and the Internet resources. Chapter III discusses the Common Gateway Interface and its methods. Chapter IV takes a look at the Java programming language, its advantages and disadvantages and the security issue. Chapter V describes the implementation of the tool developed for this thesis and its platform and environment. Conclusions, and suggestions for future work form Chapter VI.

## II. LITERATURE REVIEW

### 2.1 INTERNET

The Internet may be described in simple terms as a group of networks of computers connected together [3]. The trend towards the networking started owing to the advantages of the distributed systems over the centralized systems [4]. The distributed systems have proven to be more economical, faster, have more distribution capabilities, are more reliable, and have an incremental growth rate [4]. The Internet provides collections of information resources placed on the computers worldwide [3]. The individual computers, which may be mainframes, personal computers, or workstations, are connected through some media to form a local area network; a local network of computers is again connected to other such networks and so on, to form a worldwide network of computers [3]. And this worldwide computer network is what is commonly referred to as the Internet. The history of the Internet dates back to Arpanet; the Arpanet was a project that was sponsored by the United States Department of Defense in 1969.

Different types of networks include local area networks (LANs), metropolitan area networks (MANs), and the wide area networks (WANs). A local area network is a group of computers, two or more, connected together physically, usually by some kind of cable [3]. One of the advantages of building a LAN is the sharing of the information and resources [5]. Some of the various types of LANs are dedicated server LANs, peer-to-peer LANs, and zero-slot LANs [4]. The MANs connect together backbone networks (BNs) and LANs, and usually span anywhere between 3 to 30 miles [5]. WANs are used to connect MANs and BNs that are located up to thousands of miles apart [5]. Routers, hubs, and switches are some of the devices used for linking the LANs, MANs, BNs, and

WANs [5]. The wide area networks are usually connected through telephone lines, leased specially for the purpose of transferring digitized data [3]. Other agents for carrying information are satellite links, though the telephone systems remain the most popularly used carriers [3].

Once a connection has been established between any two computers the next important issue is for these computers to interact with each other so as to serve the purpose of a network. Anyone using the Internet will come across the terms such as electronic mail, ftp, WAIS, gopher, etc. These are but some of the services supported by the software that supports the Internet [3]. Some of the important Internet services are the *mail* service that enables users to send and receive information with other users across the network; the *telnet* service that allows a user to login into a remote server/computer; the *ftp* which allows transfer of files and directories from one computer to another; the *client-server* facility that allows a client program to establish connection with another computer and seek information or any other task from a server program on that computer; and the *gopher* which enables a user to search for any information on the Internet through a hierarchical menu system [3].

Each computer on the Internet is called a 'host' [3]. Each of these computers is also referred to as a 'node' [3]. This term is borrowed from the network terminology wherein each of the connecting points in a network is called a node [3].

## **2.2 CLIENT-SERVER MODEL**

One of the principal uses of computer networks is information and resource sharing [3]. And the fact that this sharing is done amongst the networks world-wide adds to the beauty and awe that surrounds the Internet. With the number of individual computers on the Internet increasing at an exponential rate every day, the credit for making the sharing of information and resources possible and easy goes to the client-server technology. When the client-server technology is used, the sharing is accomplished by two separate programs on different computers [3]. The server program provides a certain resource. The client program requests and uses that resource. The improvements in computer architecture have endowed the client-server relationship with new opportunities. The clients are now becoming more mobile and remote login sessions becoming increasingly popular. Now, the applications on the network see the underlying architecture of a machine as a programming interface which can be used for distributing procedures and functions [6]. Among the proven benefits of the client-server model are systems interoperability, data sharing, increase in the speed of implementation of latest systems, better systems quality and flexibility, improved maintenance and control, and the ability to control the harmful complexity that is incurred by the growth in scope, size, and heterogeneity of the systems [7].

## **2.3 TCP/IP**

The establishment of the connection and the ensuing communication between the client and the server are supported by certain protocols which are in the transmitters and receivers connected to the communications network [6]. As distributed applications became increasingly popular, the need for better networking capabilities arose [8]. Since

networking itself could not change as fast, the end-to-end protocols are used to suit and incorporate the ever changing needs of the applications [8]. A protocol is a set of rules describing how something should be done. There are well-established protocols that are used by different types of computers on the networks. Programs that need to be used over the wide networks are written using standard protocols.

TCP/IP is a group of protocols that is used to connect different computer and network systems. It allows organizing the computers and communication devices into a network. There are a number of reasons for the TCP/IP to be considered a viable approach for Internet networking. Apart from already being prevalent in the arena, it is well-founded [9]. Many products already use the TCP/IP protocol suites [9]. Also, TCP/IP is a hierarchy that is designed conceptually as a physical network interface [9]. This physical interface provides the communication interface to the hardware or the network [9]. The layer above the network services is the Internet protocol(IP) layer. The data transmission within the Internet is in the form of *packets* [3]. *Packets* are small packages of data. This breaking up of data into *packets* to enable data transmission across the network is done by the Transmission Control Protocol(TCP). Each of the packets is given a sequence number and the destination address [3]. The TCP also inserts additional information for error control into these data *packets* [3]. When the *packets* are ready for transmission, the Internet Protocol transports them to the recipient which is just another computer on the network [3]. The information received is checked for errors by the host computer at the other end [3]. This is again accomplished by the TCP. In case of an error, the sender is asked by the TCP to resend the message [3]. The received data is then reconstructed by the TCP using the sequence numbers [3]. TCP/IP, thus facilitating

the communication among the multitude components of the Internet, is what the Internet thrives on.

## **2.4 WORLD WIDE WEB**

Of all the multifaceted applications of the Internet, the World Wide Web (also known as the Web or the WWW), is perhaps the most popular and fastest growing [10]. Spanning all major platforms, the Web provides a multimedia interface viz., text, images, audio, and video [1]. Due to the interoperable multimedia graphical user interface and the existence and availability of client and server software and network protocols, Web-based systems may be developed in short spans of time [11]. The Web documents are written in hypertext format which allows the linking of various documents [10]. The recent advances in the areas of hypertext and multimedia have enabled these links to be connected to various application programs such as electronic mail, an audio file, a gopher menu, etc. In addition, the network protocols that underlie the Web enable interactive access to data from a Web browser [11]. The program that lets a user read the Web documents is called a browser [10]. There are a number of browsers existing such as the Netscape Navigator, the Microsoft Internet Explorer, the mosaic, lynx, and cello. While the browsers help view the information on a Web page from virtually any type of computer system, the search engines or tools aid the users in locating the desired information [10].

Any document on the Web is called a Web page. A Web page created for displaying one's personal information is called a homepage. A collection of Web documents related to an individual or a company is referred to as a Web site [10]. These Web pages are accessible by any computer on the network. The Web pages themselves

are stored in a computer, commonly referred to as a server, that is connected to the Internet [10]. Hence the location of the server is immaterial to the person viewing the page [10].

Two of the important protocols that make the Web so useful, interactive, versatile, and powerful are the HyperText Markup Language(HTML), and the HyperText Transfer Protocol(HTTP) [12].

## **2.5 SGML**

Today, the variety of the computers available to us is large. This variety includes the PCs, the Macintoshes, the workstations, etc. [13]. These user-friendly computers are being used both at home and in offices, entering the domain of administration [13]. Most of the administrative applications involve word- and text-processing [13]. The number of products available for processing text documents is large [13]. One can choose from MS Word, WordPerfect, WordStar, etc. [13]. But when the aspect of transmitting the text or word-processing documents across the Internet is considered, the problems inherent in such a task become apparent. An example is graphics not being constrained by the document's structure [13].

The Standard Generalized Markup Language(SGML) obviates these problems by imposing the images to be isolated from the other information in a document [13]. SGML is a standard used for exchanging information in textual form [13].

SGML may be described as a set of rules for identifying and defining different types of documents according to their structures and for allowing the machines to recognize and process the documents by their marked up structures [14]. SGML has been chosen for the Web for several reasons [14]. It has the capability to provide a reliable and

standard file type that is compatible with the already existing Internet application programs and protocols [14]. Thus it had the potential for building Web applications [14].

The SGML, HTML, and other markup languages have changed the way information is propagated and have revolutionized its transportation across the various computer platforms and systems. The markup languages allow a document to retain its original structure and format, whatever platform it is transported to on the network [14]. The SGML also renders modularity, interchangeability, and flexibility to the documents [14]. This feature of SGML of retaining the original document structure is used by hypertext to its advantage [14]. This is evident by the fact that a highlighted text points to another file or directory of information [14]. This is made possible by the encoding or marking up of the related information. Since SGML offers flexibility and characteristically allows interchangeability, it is an ideal choice for coding the Web pages [14].

Any markup language describes what the document text means, and what it should look like. It does so by means of instructions embedded within a document [14]. It may be described as a language which consists of codes embedded into a document, the codes reflecting the desired formatting for the text [15]. For example, if a word is to be displayed in italics, then that word should be preceded or followed or surrounded by the code that explicitly states in the textual form that the word is italicized. This feature of using a descriptive markup as opposed to the procedural markup style makes the document more robust or compatible with any platform [1]. When this kind of approach is used, a given document is displayed and interpreted as it is intended on any system



with little scope for misunderstanding. And hence, electronic text-processing programs use markup languages [15].

## **2.6 HTML**

HTML is one of the several markup languages, and is very closely related to SGML [15]. It is specifically used for marking up documents for electronically transmitting the documents over the Internet [15]. It is a rich language ideally suited for today's multimedia, allowing for a large variety of displays of a Web page [15]. HTML contains absolutely no physical formatting commands [15]. It consists of only printable characters [15]. HTML is a structured and extensible language [15].

The extensions of HTML allow for *hypertext* links which enable the linking of a document to another; it is this particular facility of the HTML that made the Web so popular [15]. The result of using HTML made the Web a loosely woven collection of documents placed all over the network, that are easily accessible by following certain search tools or procedures.

## **2.7 HTTP**

Once the HTML document is ready, complete with all its markup codes, it should be made accessible to other users across the Internet network. This brings forth the need for a protocol that allows for the communication or interaction between the different computers on the Web [15]. The HTTP protocol presents itself as the solution, with its numerous communication specific methods [15]. These methods include GET, POST, or HEAD and make possible the client-server computing model. Apart from conveying the clients' requests to the relevant servers and the servers' replies to the clients, the HTTP allows the communication of other information that is critical for any transaction, like the

status of a transaction, data types, character sets, or languages accepted by the clients, the data encryption information, etc. [15]. Requests are made by the clients during the client-server transactions. The clients may request the server for a file or it may expect some processing by the server on the information that the client sends [15]. If the client request is for a file, the server sends the appropriate file to the client [15]. But in the second case, the servers forward the data sent by the client to the programs that are specifically developed to handle such client requests [15]. Such programs are called the gateway programs [15]. A gateway program receives the client data from its server, processes that data and hands back the results or conclusions to the server which then responds to the client with these processed results [15]. The mechanism used by the server to correspond with the gateway program is called the Common Gateway Interface(CGI) [15].

The knowledge of the CGI specifications is thus vital for coding and implementing the server-side gateway programs and the HTML Web pages [15]. The gateway programs are accessed using the URLs. The Uniform Resource Locator(URL) gives the browser the address of the file or the program on the Web, tells the browser the type of service that the address contains, etc. [16]. The URL is a form of URI(Uniform Resource Identifier) that is used to access an object on the Internet [16]. Whenever the Web site or the URL pointing to the gateway program is reached by the client, the program is made active by the server and the data from the client is sent to the program for processing [15].

HTTP is a stateless Internet client-server protocol that efficiently transmits hypertext materials over the Internet [15]. It is stateless due to the fact that once a server has delivered the processed data to the client, it loses the connection with the client,

retaining no memory of the address of the client or the transaction [15]. The speed of an HTTP server may partly be attributed to the fact that the client-server transactions are stateless [15]. Data is generally transmitted in the form of 8-bit characters.

An HTTP connection may be summarized in the following four steps [15] :

1. *Contacting the server*: The client contacts a Uniform Resource Locator describing that server.
2. *Request from the Client*: Once the client establishes contact with the server, it passes its request for a certain service from the server, to the server, using the HTTP request methods.
3. *Response from the Server*: The data or information requested by the client is now sent to it by the server.
4. *Breaking the connection*: Once the server fulfills the client's request, it breaks the connection with the client and does not retain any memory of the transaction.

### III. COMMON GATEWAY INTERFACE

#### 3.1 INTRODUCTION

HTML documents are plain text documents with embedded formatting and displaying instructions [16]. If the client has requested a URL that points to an executable program and expects some processing from that URL, then the server activates that program, passing on data from the client to it and returning the processed data to the client [16].

The standard used to specify communication methods between the HTTP servers and such executable programs on the server-side is called the Common Gateway Interface(CGI). Thus the server-side gateway programs are extending the capability of HTTP connections by providing the Web clients access to the independent programs that are not run by the server [17]. The CGI specifications describe the passing of information from the server to the gateway program and the program's returning the processed data to the server [15].

Gateway programs are just ordinary programs, commonly referred to as CGI scripts/programs. They are written in common programming languages such as C, C++, or Visual Basic, or in Unix shell scripts like awk, sed, or perl [15]. By far, perl seems to be the widely used language for writing CGI scripts for the Web.

Sometimes the client may be more ambitious and try to access a well-established and defined program that is not particularly set up for the use on the Web. It may be accessing an Oracle or a Visual Basic or a C database or a multimedia graphics package, with attached audio and video files. Access to such programs are made possible with the CGI. CGI scripts also make possible the facility for playing games on the Internet. The

location of the players is irrelevant as long as each of them is able to access the same URL or Web site. Figure 1 shows the flow of information between the client and the server with CGI mechanisms included:

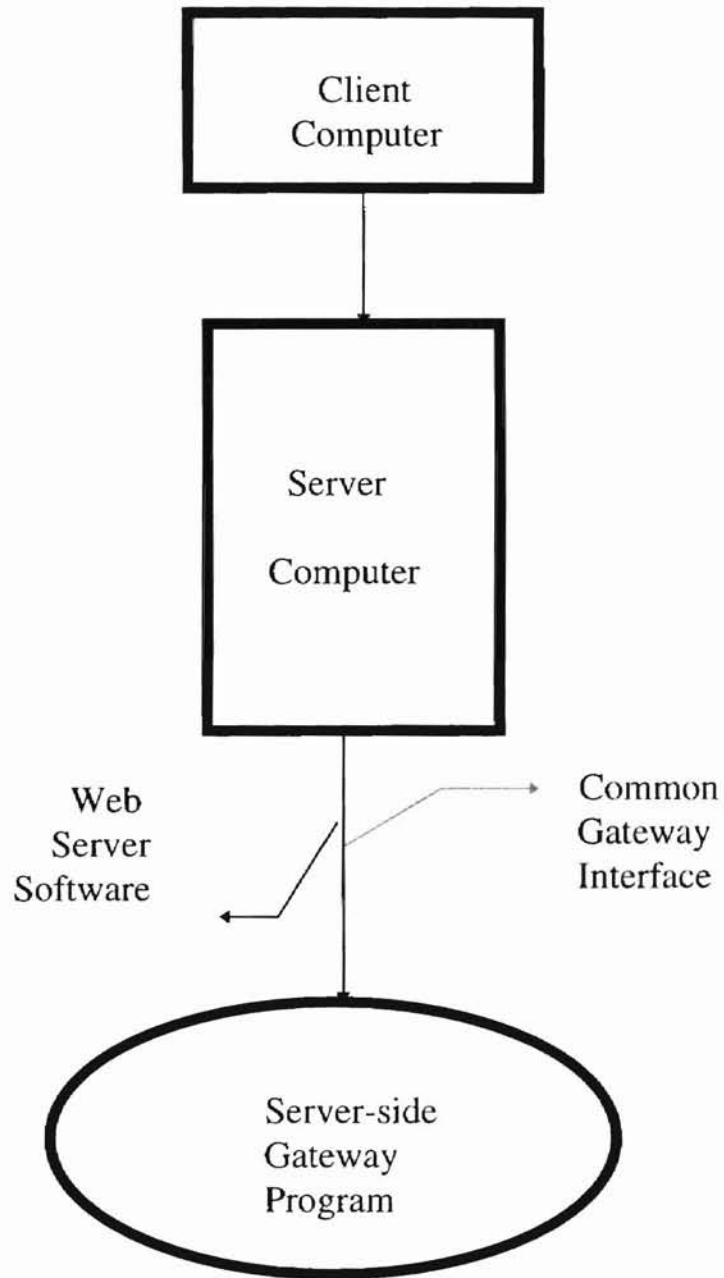


Figure 1. Flow of information between the client and the server-side gateway program

Some of the ways in which CGI applications are used today are receiving and processing user input from the HTML pages, and displaying the result; and creating new HTML pages dynamically. Also they have the ability to efficiently process large amounts of user input from the HTML forms.

The data sent by a client is passed to the gateway program in three different ways [15]. The data is processed by the gateway program and returned to the server in two different ways [15]. These are described in the next two sections.

### **3.2 COMMUNICATION BETWEEN THE CLIENT AND THE GATEWAY PROGRAM (CGI):**

The three ways in which the data is communicated to a gateway program from the server are the command-line arguments, standard input, and the environment variables [15].

1. *Command-line arguments*: These are launched by the server for the gateway program. This is done by using the HTML tag ISINDEX in the Web document.
2. *Standard Input*: The data is read from the stdin by the gateway program. For this, the HTML POST method should be used in the FORM element. This method has the advantage of being able to accept and interpret large blocks of input successfully. The user data is passed on to the CGI program in attribute-value pairs; where attributes are names given to the input values and values are the actual input.
3. *Environment Variables*: These variables contain the state of the local environment or system. The gateway program can access these variables. The particulars provided by the client such as the extra path information, request header field content etc. are

some of the examples of environment variables. The user input is obtained by using the HTML GET method inside the FORM element. This data is stored in the *query\_string* environment variable. This thesis uses this form of communication between the server and the CGI program.

### **3.3 COMMUNICATION BETWEEN THE GATEWAY PROGRAM AND THE SERVER :**

The gateway program communicates with the server in two different ways [15]:

1. *Standard Output*: The processed data is conveyed back to the server on its standard output. The server then sends it to the client. A server directive such as

*Content-type: type/subtype*

should be the first line of the output from the gateway program.

2. *Name of the Gateway Program*: Any gateway program whose name begins with *nph-* returns the HTTP response header file to the server. And the server directs the output of the program to the client without parsing it. The first line of the CGI output in this case must be the server protocol and version number. Other information that should be output are the date the document is made, server software, and the MIME version. An example statement is: `cout << "MIME-version: 1.0"<<endl;`

## IV. JAVA

Integrating multivariate platforms and developing applications with increased portability are some of the important tasks the computer world faces today.

The narrowing of gap between the different countries of the world through the information highway has brought forth the need for software that will overcome the language barriers and the differences in standards and character set encodings [18]. The Java environment, consisting of the Java programming language, the Java Virtual Machine, and the Java Development Kit(JDK), offers itself for the creation of globally portable software. To develop “write once and run anywhere” software, unicode, object-oriented design, multilingual support, platform independence, backward compatibility, and locale sensitive classes are some of the features supported by Java [18].

The Java programming language has an architecturally neutral code which is ideal for transferring to and running on the Internet [20]. The Java code instructions are distributed (transferred to clients) and interpreted at run time, unlike in CGI scripts where the code is run at the server end and the executable is distributed [16].

The use of the CGI programs makes the static HTML pages dynamic, productive, and interactive. This is achieved by the client communicating with the gateway program that can be activated by the server. Java introduced the concept that at least part of the server-side gateway program code will be transferred to the client side when a HTTP connection is made, and this can be activated and executed by the client.

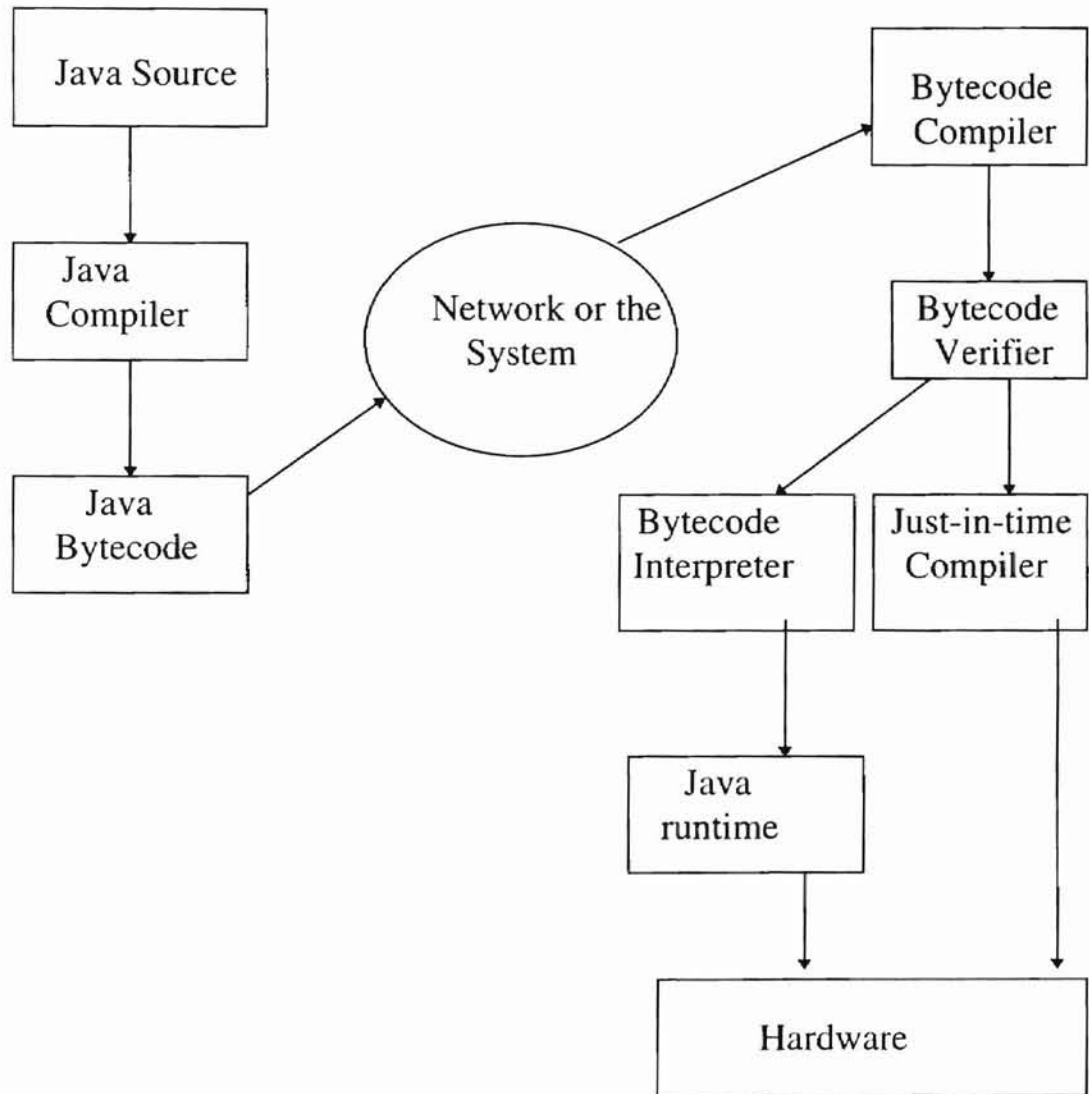
Java provides the server with the ability to send both the data and the program necessary to manipulate the data to the client. The client executes the program. That is,



each time a user loads a Web page, the user is loading both the content and the code necessary for loading that content [20].

All these features have been made possible due to the Java Virtual Machine (JVM), which when loaded on a machine acts as a virtual operating system [19]. The Virtual Machine is ported to a wide variety of vendor platforms [21]. It defines an interface for executing the bytecode programs. JVM runs on top of a host operating system and interprets Java bytecode [22]. Apart from the Java interpreter, JVM also includes the runtime environment, and forms the base for executing Java programs on remote machines [22]. The Java Remote Method Invocation (RMI) enables the method invocation on remote objects, adding significant functionality to the communication in distributed systems. More importantly, it changes the way distributed computing systems work [19]. The RMI system consists of: (a) client-side proxies, and server-side dispatchers, (b) remote reference, (c) connection setup and management, and (d) remote garbage collection [19].

Figure 2 below shows the execution model of a Java applet:



**Figure 2. Execution of a Java applet**

Courtesy: Marc A. Hamilton, "Java and the Net-Centric Computing", Computer, August 1996, pp 31-39.

Java's foothold in Web computing is being consolidated by the support and confidence it enjoys from several companies. Its adoption rate into the technology mainstream is one of the steepest [25]. Furthermore, some of these corporations are abetting the usefulness of Java by customizing their products to the Java environment. For instance, Netscape Communications Corporation's Internet Foundation Classes(IFC) is a library of user interface tools that is written in Java and runs on top of AWT [27]. IFC enhances Java's user interface.

Many academic and industrial research groups today are interested in Java's potential as the network programming language. Various projects have been undertaken to extend the functionality of Java. One such is the design and successful implementation of a custom embedded operating system to support the JVM [ 22]. Other projects include optimizing NET compilers for better performance [21].

In the absence of a standards body, big companies are now voting to use Java as their binary platform standard [28]. It is now foreseen by the industry observers that the corporate intranet sites of the big companies will be run using Java [28].

IBM is devoting its research to Java-based products and is looking to products like JavaBeans for developing software applications in short spans of time [28]. JavaBeans is the effort of Sun, IBM, and others for use with the Java language [28]. JavaBeans are the reusable components that can be assembled to form more complex Java programs [28]. IBM, along with Sun voted for JavaBeans to be the container technology of its choice [28]. One of IBM's projects, BeanMachine, dynamically analyzes various JavaBeans components and combines them to form applications without the need for programming [28].

## 4.1 SECURITY

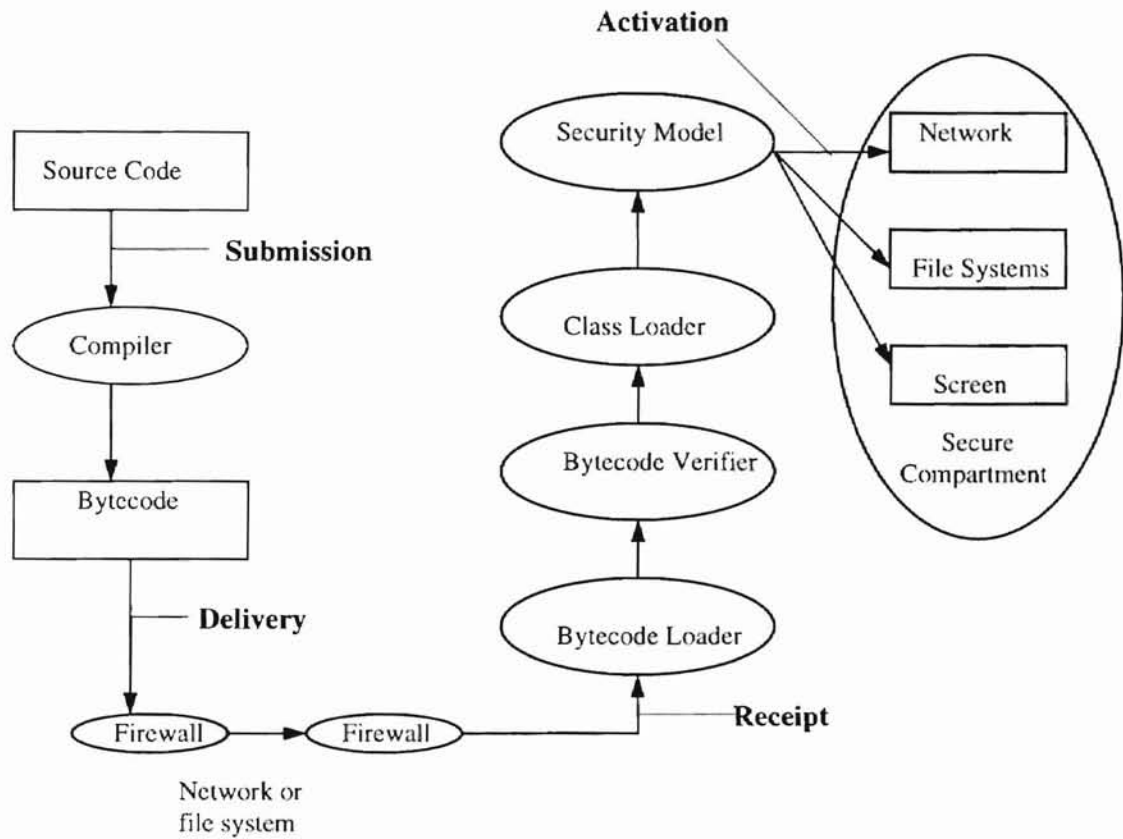
The proliferation of code on the Web and distributed systems these days necessitates authentication and secure code distribution [23]. The Java applet is the most popular code distribution scheme and much emphasis is currently laid on the security measures used in Java applet transfer [23].

The Java Virtual Machine controls potentially harmful access to system resources during activation time, thus ensuring safe execution of the code locally [23].

Java does not allow for code corruption before execution [21]. The bytecode specification ensures safer software distribution as compared to the traditional languages such as C/C++ [21].

Java uses a four-pass security process; three passes are conducted when the class is first loaded and the fourth pass is conducted when the code is executed [21]. In this process, the bytecode is examined for any malicious content during the code distribution on a network [21]. Scrutiny of language semantics and structure, dataflow analysis of each method in each of the classes, access permission and type checking are all part of the verification process that Java uses [21].

Figure 3 below shows the security model of an applet [23]:



**Figure 3. Security Model of an applet showing four phases related to the applet's life cycle.**  
 Courtesy: 'Secure Code Distribution' by X. Nick Zhang.

Notwithstanding all the security measures that are in-built in Java language and platform, code distribution does have the danger of transferring harmful objects or code over the network, intentionally or otherwise [23].

## 4.2 ADVANTAGES

One of the features of Java that makes it very efficient is that it can run on any platform that has a Java enabled browser, thus making it virtually platform independent. It runs on Windows 95, Windows NT, MacOS, Sun Solaris etc.

The stack model used in Java Virtual Machine allows for the architectural independence of the interpreter [24].

Java has the ability to dynamically link a new class or interface at runtime [21]. Even though Java closely resembles other traditional programming languages in its structure and rules, its popularity as a language for the Web is unsurpassed, at least for now. It is described as simple, object-oriented, platform independent, robust, secure and multithreaded [20]. Java allows for interactive graphics and 2D and 3D animation [25].

A single development platform that enables easy distribution of application programs over a heterogeneous network of computers lends itself to the development of application programs with increased flexibility and ease of use [20].

The Java applets are loaded just as image files or audio files are loaded. Java has the ability to support any protocol and it removes the need to know the helper application of the specific file that is being loaded [20].

Though CGI provides an interface between the HTTP server and the user applications on the server, the advent of Java brought to the arena an open distributed environment, along with some advantages over a CGI-based approach for remote data access [25]. Java obviates the need for the communication between a client and a back-end database to go through a HTTP server [25]. The other advantages that Java has over CGI-based communication are presentation graphics, ability to handle session-oriented

database transactions, and greater security [25]. Also, the Java language has superior built-in networking capabilities such as using sockets to start a network session; and has the ability to access a file on the remote server directly, though this feature currently does not work on all the browsers. The CGI-based approach, relying on the stateless HTTP protocol, requires the opening and closing of the connection even for subsequent queries by the same user [25]. When a Java applet is used, the database connection need not be revived, as long as the user is in session and the applet is active [25]. Java has a rich set of drawing tools in its Abstract Window Toolkit or AWT [25]. And the mobility of the Java code does not have a substitute in CGI-based interfaces [25].

#### **4.3 DISADVANTAGES**

Though Java appears to be becoming a universal language, it is not without its shortcomings. *I/O*, APIs to operating system, etc. are some of the features lacking in Java [26]. It retains some of the undesirable features of C/C++ [26].

In execution speed, Java loses out to the traditional languages. Despite the promise of being a universal programming language, the Java does not run on all computers, platforms, and operating systems, yet [26].

## **V. REMOTE FILE ACCESS AND GRAPHING TOOL (RFACT)**

The demand and the need to develop applications at an unprecedented pace has led to the intelligent merging of the existing applications to perform a specific task [29]. With the Web being ubiquitous in the computing and the networking world today, it is imperative that at least in most of such cases, it be one of the components used to build an application. And this brings into the spectrum a wide range of Web servers, component models, platforms, and environments to chose from [29].

This is where the classic client-server computing model gives in to a new, three-tiered model, with the independent databases and other application programs forming a third tier [29].

This thesis, a Web-based tool, uses a three-tier model for its implementation.

### **5.1 IMPLEMENTATION MODEL**

The data file and the program to modify it, the client, and the server, form the three tiers of this tool. The data file and the program to modify it are on the server. The client is any computer connected to the Internet network, that has a Web browser. The server is also a computer that is connected to the Internet network.

Every set of data in the data file has a keyword associated with it. The user can see this data remotely from a client computer by entering the keyword on the Web browser. This input from the user is read by a CGI program that resides on the server computer. This program looks up the data file, finds the data associated with the keyword, and passes it on to a Java applet that also resides on the server. It also displays the data in textual form on the client browser. The Java applet receives the data from the CGI program and draws a 2D graph on the Web browser on the client computer.



Figure 4 shows the schematic architecture of RFAGT. The user input (or the keyword) entered by the user on the client browser on file ViewGraph.html is transmitted to GetData, an executable of the CGI program GetData.cc, which is located on the Web server. GetData reads in the keyword, opens the server-side data file YearVal.dat (which is maintained by YearVal.cc) and fetches the data pertaining to the keyword from the file. It then displays this data on the server; and also passes it on to DP9.class, which is obtained from compiling a Java applet DP9.java. DP9.class reads in this data as parameters and draws a 2D graph with it on the client browser.

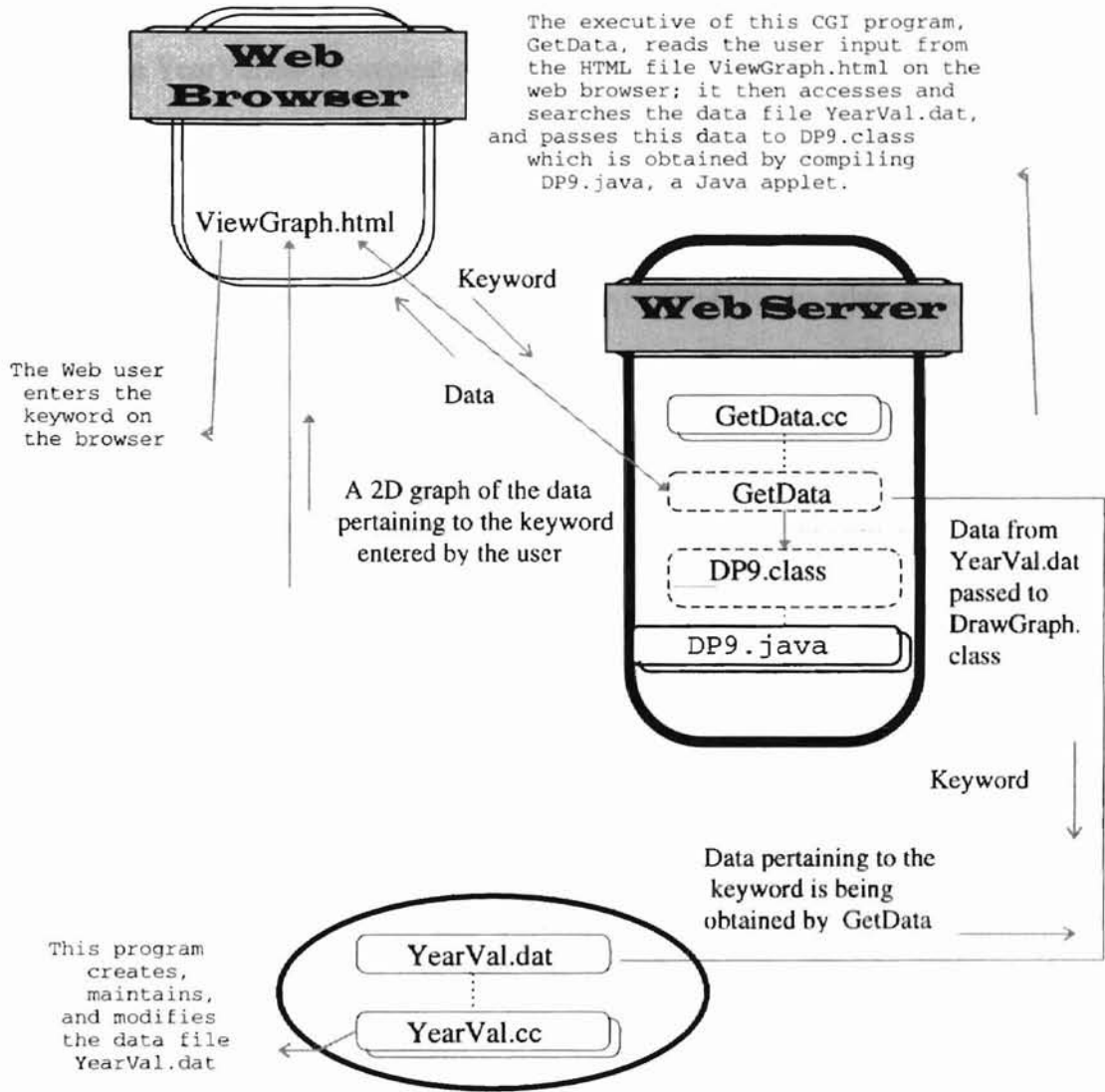


Figure 4. System Architecture of RFAQT

## 5.2 IMPLEMENTATION

The data file YearVal.dat is created using the program YearVal.cc. The executable of this C++ program may also be used to modify the data file; data objects or keywords and their values may be added, deleted, or listed. YearVal.dat is the Unix data file on the Web server. It contains the data that is retrieved remotely by RFAGT. In other words, it is the server-end information base, that together with the program YearVal.cc forms a server-end application program. The RFAGT retrieves specific data from YearVal.dat and displays it in textual and graphical forms.

The keyword should consist of 1 to 13 characters. The data pertaining to each keyword consists of year-value pairs with each pair beginning and ending with '<' and '>' marks respectively. Each pair is separated from the other by a space or a tab. The 'year' should be 4 digits in length. The 'year' and its corresponding 'value' are separated by a comma. Each line in the data file YearVal.dat has the following format:

```
Keyword    [space]    Number_of_elements_related_to_this_keyword  
[space]    <Year,Value> [space/tab] <Year,Value> [space/tab] .....<Year,Value>  
C_End-of-line_character.
```

To view the data pertaining to each keyword remotely through a Web browser, the user has to know the name of the object or the keyword. The HTML file that forms the user interface on the web is named ViewGraph.html. This file will be located in thehtdocs directory on the server. Its address on the Web is <http://chester.cs.okstate.edu/ViewGraph.html>.

Figure 5 below shows the Web page where the user can input the keyword :

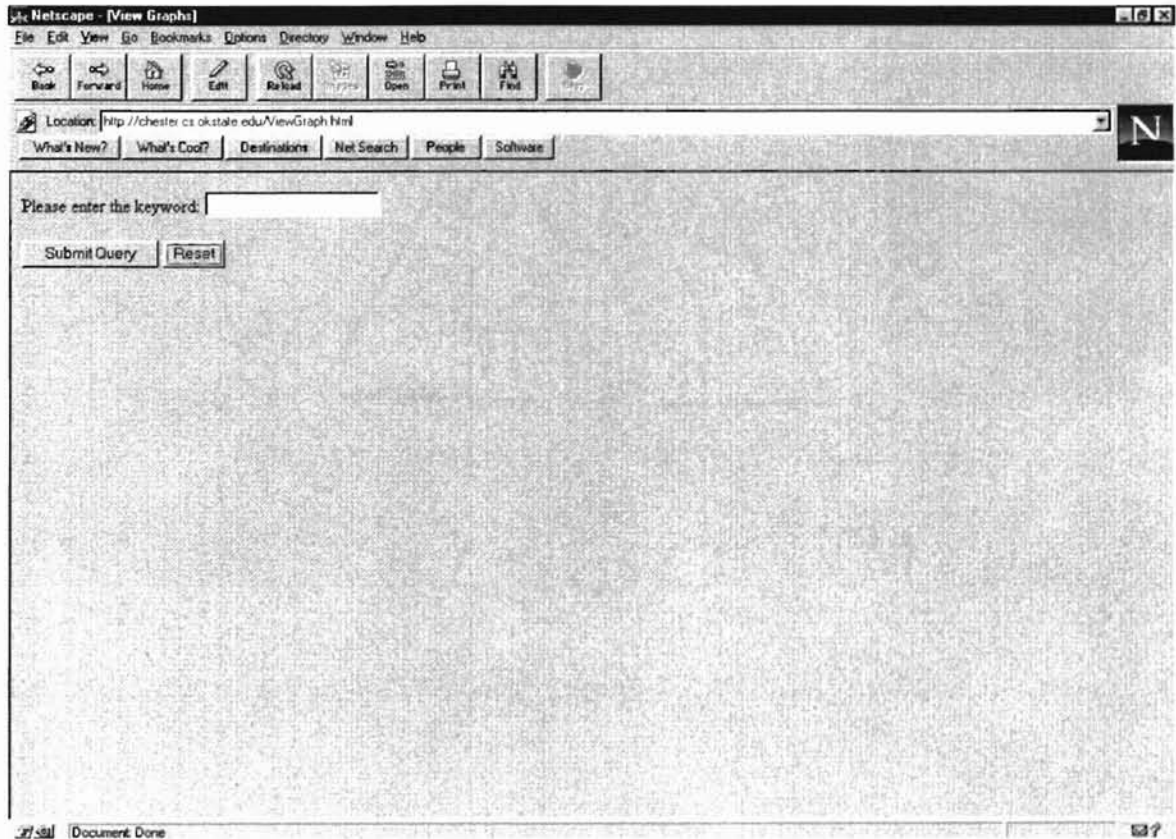


Figure 5. Web page for the user input.

Figure 6 below shows the Web page after the user enters the keyword :

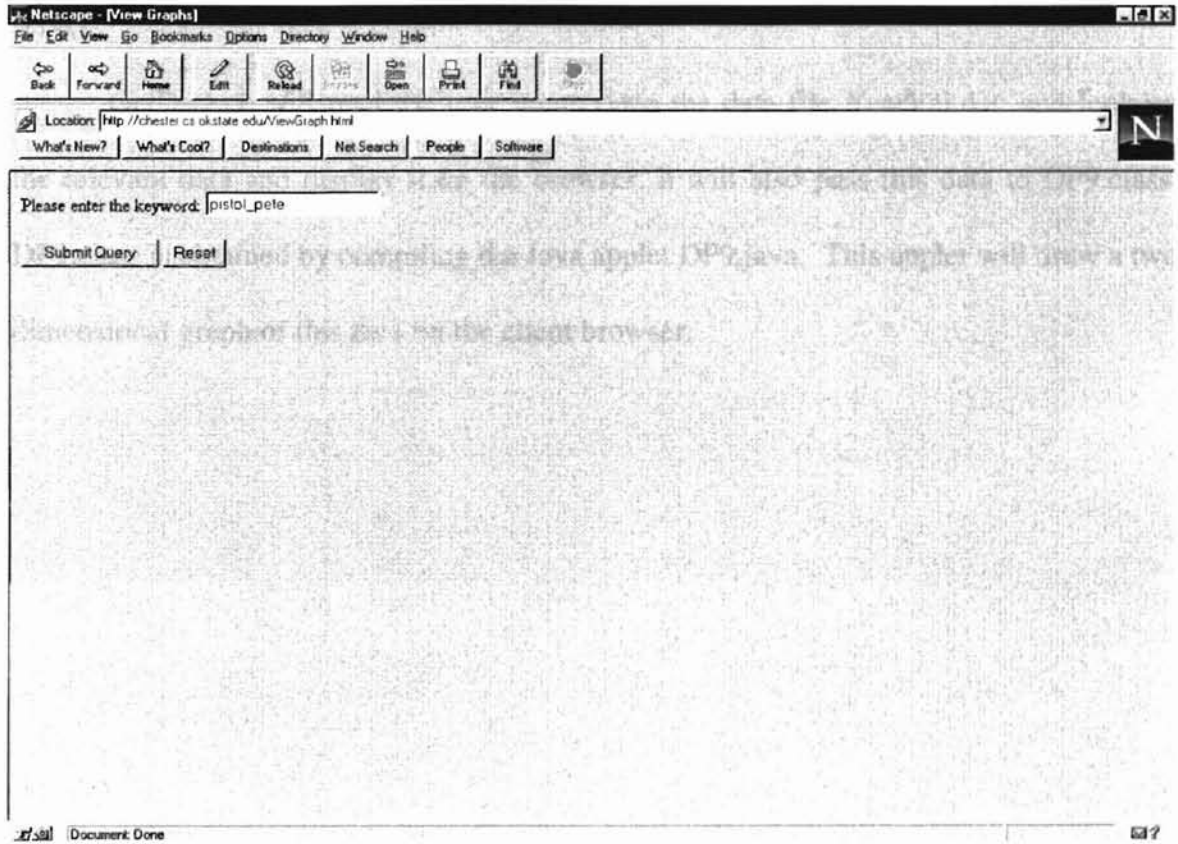


Figure 6. Web page after the user input.

Once the user enters the keyword on the browser, the input is processed by the CGI program `GetData.cc`. This C++ program is compiled; and its executable is located in the `cgi-bin` directory on the server, for the data to be accessed and processed. In this case, the address of the file is `chester.cs.okstate.edu/cgi-bin/GetData` (`GetData` being the name of the executable). The program `GetData.cc` itself will be in the `cgi-src` directory on the server.

`GetData.cc` will read the user input, open the data file `YearVal.dat`, and look-up the relevant data and display it on the browser. It will also pass this data to `DP9.class`. `DP9.class` is obtained by compiling the Java applet `DP9.java`. This applet will draw a two dimensional graph of this data on the client browser.

Figure 7 shows the Web page that is displayed to the user after the user enters a keyword present in the data file and clicks on the 'Submit Query' button.

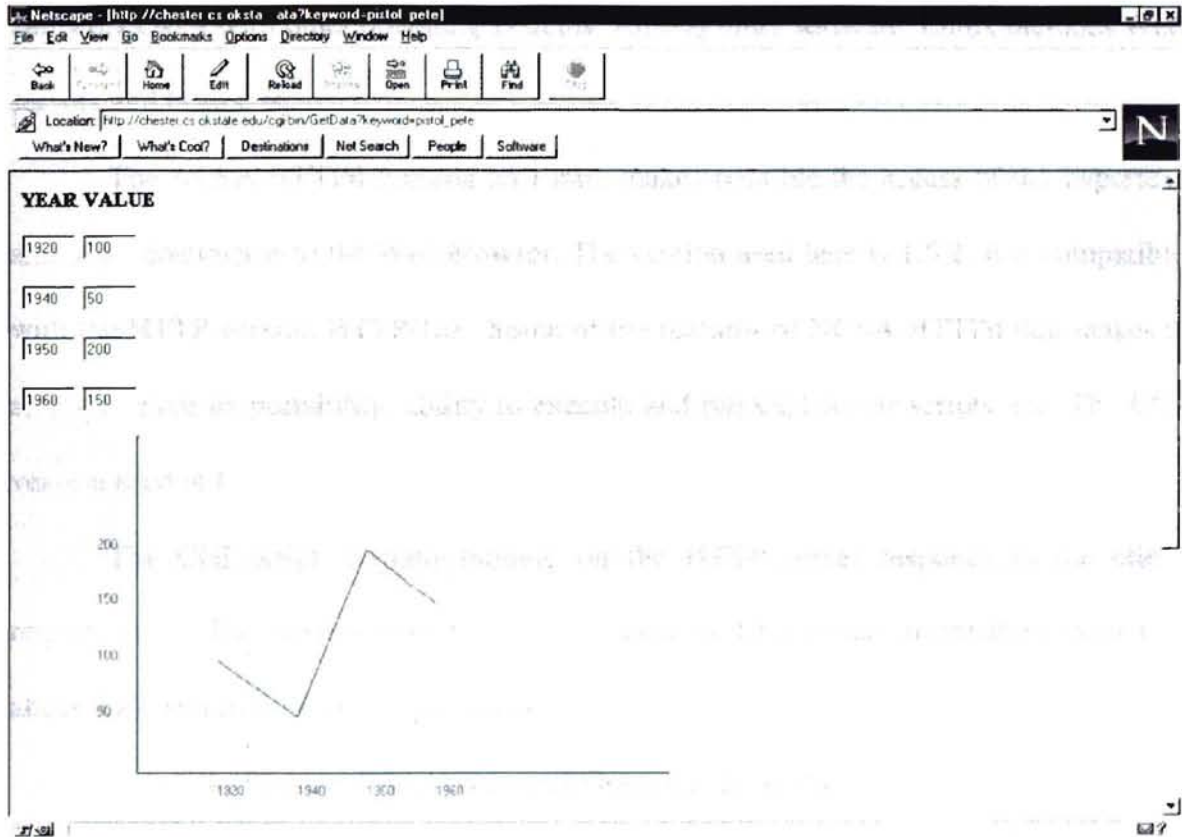


Figure 7. Graph drawn by RFACT based on the user input.

### 5.3 PLATFORM AND ENVIRONMENT

The operating system on the server computer is Linux 2.0.28. Linux is an implementation of Unix. It is freely available and it supports software such as X Windows, TCP/IP, Emacs, etc. [30]. It runs on i386+, PowerPC, SPARC and DEC ALPHA hardware [30]. It is a multi-user, multi-tasking operating system that interacts and works well with other operating systems. Among other software, Linux includes Web servers and browsers.

The NCSA HTTPd running on Linux makes possible the access of the hypertext and other documents to the Web browser. The version used here is 1.5.2. It is compatible with the HTTP version HTTP/1.0. Some of the features of NCSA HTTPd that makes it so popular are its portability, ability to execute and run CGI server scripts, etc. The CGI version used is 1.1.

The CGI script/program running on the HTTP server responds to the client request [31]. The environment variables defined by CGI reveal information from and about the client to the CGI script/program.

In Remote File Access and Graphing Tool, the CGI program executes in real-time, opens the data file on the server, searches for pertinent information in the file, and passes it on to the Java applet. The Java Development Kit version 1.1.3 is a product of the Sun Microsystems, Inc. It is used for developing this applet. It is developed by a company called JavaSoft [32]. The JDK environment facilitates the development of the Java applets and applications. The Java applets need browsers supporting the Java Platform 1.1 to run. Currently, Netscape Navigator 2.0 and above, Microsoft Internet Explorer 3.0 and above, HotJava version 1.0 and above can run Java applets on Microsoft



Windows 95/NT 4.0, Sun Solaris 2.4, 2.5 SPARC, and Sun Solaris 2.5 x86. The Java applications can run without a browser. The JDK consists of Java Core Classes, Java Source Files for Public Classes, Java Compiler, Java Interpreter, Java Appletviewer, AWT 1.1, etc. [32].

## VI. CONCLUSIONS

Online access of data and its graphical representation is a fast, convenient, and efficient mode of sharing information. It is also an economic and efficient means of data access. With the widespread use of the WWW and the PCs, new applications are being developed to fully exploit the client-server model on which the Internet is based; and the Web's multimedia features. New protocols and programming languages are making possible real-time access of data and increased speed of data transfer.

This thesis has dealt with developing a Web application tool RFAGT. RFAGT performs three tasks. Firstly, it develops a server-side application program that creates and maintains a data file; this data file contains the data that must be made accessible to remote computers. Secondly, it develops the interface program that accepts the user input from remote terminals via WWW, retrieves the information from the data file on the server, and processes it. It also develops a graphing tool using the programming language Java that draws a line graph representation of the data retrieved by the interface program. Lastly, it develops a user-interface for accepting the user inputs from the remote computers. RFAGT is useful for displaying data on a remote computer in graphical and tabular forms.

RFAGT can be used as a basis for building tools. For example, it can be used to develop an application to display daily stock movements. Such applications are considered future work.

## REFERENCES

- [1] Goldberg, Mascha, Gentner, Rossman, Rothenberg, Sutter, and Weigley, "Beyond the Web: manipulating the real world", Computer Networks and ISDN Systems, 1996.
- [2] A. N .Boston and D. R .B. Stockwell, "Interactive species distribution reporting, mapping and modeling using the World Wide Web", Computer Networks and ISDN Systems, 1996.
- [3] Harley Hahn & Rick Stout, "The Internet Complete Reference", Osborne McGraw-Hill, Berkeley, California 9410.
- [4] Andrew S. Tanenbaum, "Distributed Operating Systems", Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1995.
- [5] Jerry Fitzgerald and Alan Dennis, "Business Data Communications and Networking", Fifth Edition, John Wiley & Sons, Inc., USA, 1995.
- [6] Colin Smythe, "Internetworking: Designing the Right Architectures", Addison-Wesley Publishing Company, UK, 1995.
- [7] Alex Berson, "Client/Server Architecture", The McGraw Hill Companies, Inc., USA, 1996.
- [8] Daniel C. Lynch and Marshall T. Rose, "Internet System Handbook", Addison Wesley Publishing Company, Inc., 1993.
- [9] Ulysses Black, "TCP/IP and Related Protocols", Second Edition, McGraw-Hill, Inc., USA, 1995.
- [10] "Selected Tax Policy Implications of Global Electronic Commerce",  
<ftp://ftp.fedworld.gov/pub/tel/internet.txt>
- [11] Gary Jason Mathews and Syed S. Towheed, "WWW-based data systems for interactive manipulation of science data", Computer Networks and ISDN Systems, 1996.
- [12] Hal Berghel, <http://www1.acm.org:82/~hb/index2.html>
- [13] Eric Van Herwijnen. "Practical SGML", Second Edition, Kluwer Academic Publishers, Norwell, Massachusetts, 02061, 1994.
- [14] Martin Colby and David S. Jackson, "Special Edition Using SGML", Que Corporation, USA, 1996.

- [15] Ian S. Graham, "The HTML Sourecbook : A complete guide to HTML 3.0", Second Edition, John Wiley & Sons Inc., USA 1996.
- [16] "Special Edition Using CGI - Online Version", [http://www.mcp.que/et/se\\_cgi](http://www.mcp.que/et/se_cgi)
- [17] Mark Swank and Drew Kittel, "World Wide Web Database Developer's Guide", Sams.net Publishing, Indianapolis, IN 46290.
- [18] Vartan Piroumian, "Internationalization Support in Java", IEEE Micro, 1997.
- [19] Ann Wollrath, Jim Waldo, Roger Riggs, "Java-Centric Distributed Computing", IEEE Micro, 1997.
- [20] Tim Ritchey, "Java!", New Riders Publishing , Indianapolis, IN 46290. USA, 1995.
- [21] Cheng-Hsueh A. Hseih, Marie T. Conte, Teresa L. Johnson, John C. Gyllenhaal, Wen-mei W. Hwu, "Optimizing NET Compilers for Improved Java Performance", IEEE Micro, June 1997.
- [22] Bruce R. Mantague, "JN: OS for an embedded Java network computer", IEEE Micro, 1997.
- [23] X. Nick Zhang, "Secure Code Distribution", Computer , 1997
- [24] K. Arnold and J. Gosling. The Java Programming Language, Addison-Wesley, Reading, Mass., 1996.
- [25] Nick N. Duan, "Distributed database access in a corporate environment using Java", Computer Networks and ISDN Systems, 1996.
- [26] Ted Lewis, "If Java is the answer, what was the question?", Computer, March 1997.
- [27] Andy Turk, "Building a better interface with Java", Byte, August 1997.
- [28] M. Weatherford, "IBM bets on beans", IEEE Micro, 1997.
- [29] Dick Poutin and John Montgomery, "Web Components", Byte, August 1997.
- [30] <http://sunsite.unc.edu/pub/Linux>
- [31] <http://www.ast.cam.ac.uk/%7Edrtr/draft-robinson-www-interface-01.txt>
- [32] <http://java.sun.com/products/jdk/1.1/README>

## APPENDICES

## APPENDIX - A

### GLOSSARY

**Internet** A world-wide network of computers spanning various platforms, operating systems, and architectures.

**HTTP** The hypertext transfer protocol allows for communication between computers connected to the Internet.

**HTML** The hypertext markup language allows for hypertext linking between documents.

**Client-Server model** The client computer on the network 'requests' a service from the server computer. The server computer provides the service to the client. The communication is stateless.

**Browser** Program from which a document is viewed.

**TCP/IP** The Transmission Control Protocol and Internet Protocol is a collection of protocols upon which the communication between the computers connected to the Internet is based.

**CGI** The Common Gateway Interface protocol allows for the Web server to interact with the application programs on the server-end, thereby extending the services provided by the server.

**Java** An object-oriented programming language that allows the code to be transferred and executed on the client machine.

### TRADEMARK INFORMATION

UNIX is a registered trademark of AT&T.

Java is a registered trademark of Sun Microsystems, Inc.

JDK is a registered trademark of Sun Microsystems, Inc.

NCSA HTTPd is a registered trademark of the Board of Trustees of the University of Illinois (UI).

Netscape Navigator is a registered trademark of Netscape Communications Corporation.

Microsoft Internet Explorer is a registered trademark of Microsoft Corporation.

## APPENDIX - B

### SOME POPULAR INTERNET RESOURCES

**Electronic Mail** Multimedia mail can be sent and received from anyone on the Internet.

**Remote login** Using the telnet facility, the user may login into his/her account on a computer remotely.

**Newsgroups** Internet supports a system of discussion groups, each group formed by people of similar interests.

**FTP** File Transfer Protocol is a service that allows files to be transferred from one computer to another on the Internet.

**Internet Relay Chat** This feature the Internet users to 'chat' with more than one other user in real-time.

**Gopher** An information resource that contains textual information in a hierarchical system.

**World Wide Web** A multimedia, hypertext tool to access and view information.

**Microsoft NetMeeting** A teleconferencing tool that supports the multimedia.

**CUSeeMe** CU-SeeMe is a free videoconferencing program. By using a reflector, multiple parties at different locations can participate in a CU-SeeMe conference, each from his or her own desktop computer.

**Virtual Reality Modeling Language (VRML)** is a Web language for the animation and 3D modeling of geometric shapes.

**Internet Phone** Allows making calls from the PC to a telephone.

## APPENDIX - C

### ViewGraph.html

This HTML file forms the user interface on the client browser.

```
<html>
<head>
<title> View Graphs </title>
</head>
<body>
<form method = "get" action="http://chester.cs.okstate.edu/cgi-bin/GetData">
Please enter the keyword:
<input type="text" name="keyword" maxlength="13"><p>
<input type="submit">
<input type="reset">
</form>
</body>
</html>
```



## GetData.cc

This is the CGI program that accepts the user input from the html file, and checks the data file YearVal.dat on the server for the relevant data; if the data is found, the program displays it on the user's Web browser; it also passes on this data to the Java applet DrawGraph.java on the server which draws a 2D graph on the user's browser using this data.

```
#include <iostream.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <fstream.h>

int main(int argc, char **argv) {

    //sending MIME header
    cout<<"Content-type:text/html\n\n"<<endl;
    cout<<"<html>"<<endl;

    char *cap;
    char *empty="<empty>";

    //macro for displaying environment variables
    #define safenv(a) ((cap=getenv(a))?cap:empty)

    // declaring and allotting dynamic memory for the keyword
    char *kw = new char[25];
    strcpy(kw, safenv("QUERY_STRING"));

    fstream input;
    char *datFile = "YearVal.dat";

    input.open(datFile, ios::in); //making datFile a pointer to input file

    //checking while opening file
    if(input.fail()) {
        cerr<<"Error in opening file."<<endl;
        exit(-1);
    }
}
```

```

char *key = new char[13]; //the keyword entered by the user
char *buffer = new char[13]; //holds keywords while checking for a match
int flag = 0; //used for indicating if object exists
int n; //stores number of data elements
char *info = new char[12];
char *op = new char[12];
char *cp = new char[12];
int *year = new int[1000];
int *value = new int [1000];
int MaxX=0,
    MaxY=0;

strcpy(key, kw+8);

while (!input.eof()) {
    input>>buffer;
    if (strcmp (key, buffer) == 0) {
        flag = 1;
        input >> n;
        for(int i = 0; i<n; i++) {
            input>>info;

            strcpy(op, info, 5);
            year[i] = atoi(op+1);
            if (MaxX < year[i])
                MaxX = year[i];
            strcpy(cp, info, 12);
            value[i] = atoi(cp+6);
            if (MaxY < value[i])
                MaxY = value[i];
            info ++;
        } //end of for loop for reading in the data elements
        cout<<"<h3><b>YEAR " <<"VALUE</b></h3>\n";

        cout<<"<FORM NAME = \" a_form\">\n" <<endl;
        for(int k=0; k<n; k++){
            //cout<<year[i]<<"\t" <<value[i]<<endl;
            cout<<"<INPUT SIZE=5 NAME=\"arrYr\" <<i<<"\"
                VALUE=\"\" <<year[i]<<"\">\n" <<"<tab>" <<endl;
            cout<<"<INPUT SIZE=5 NAME=\"arrVl\" <<i<<"\"
                VALUE=\"\" <<value[i]<<"\">\n" <<"<p>" <<endl;
        }
        cout<<"</FORM><p>" <<endl;

        cout<<"<applet codebase=\"http://chester.cs.okstate.edu\" code=\"DP9.class\"
width=800 height=700>" <<endl;

```

```

cout<<"<param name=\"numElem\"><< n <<\"><<endl;
cout<<"<param name=\"MaxX\"><< MaxX<<\"><<endl;
cout<<"<param name=\"MaxY\"><< MaxY<<\"><<endl;
for(int z=0; z<n; z++) {
    cout<<"<param name=\"a\"><< z <<\"><< year[z]<<\"><<endl;
    cout<<"<param name=\"b\"><< z <<\"><< value[z]<<\"><<endl;
}
cout<<"</applet><<endl;
cout<<"<p><<endl;
cout<<"</html>\n\n"<<endl;
} //end of if for keyword check
memset(buffer, '\0', 13);
} //end of while
// cout<<"flag: " <<flag<<endl;
if (flag != 1){
    cout<<"<br><br><h3><b><i><center><blink><< \"\nKeyword non-existent in data
file!\"<<\"</center></h3></b></i></blink><<endl;
    cout<<"</html>\n\n"<<endl;
}
return 0;
}

```

## YearVal.cc

This file is the server-side program that creates, maintains, and modifies the data file YearVal.dat. The data file it creates and maintains stores keywords and their data in the following format:

```
keyword      num_of_data_elements      <Year,Val>  <Year,Val>  <Year,Val>
```

```
#include <iostream.h>
#include <unistd.h>
#include <stdio.h>
#include <iomanip.h>
#include <fstream.h>
#include <string.h>
#include <stdlib.h>
```

```
fstream input, append, app_tmp, out_tmp, buf_tmp;
char *datFile="YearVal.dat", //this is the file containing data
    *tmpFile="YearVal.tmp",
    *bufFile="YearVal.buf";
```

```
char *incrCount(char *);
char *decrCount(char *);
```

```
void menu(void) {
    cout<<"\nPlease choose from the following options:\n"
        <<"\t"<<"1.Add a new object\n"
        <<"\t"<<"2.Add to an existing object list\n"
        <<"\t"<<"3.Delete an object from the data file\n"
        <<"\t"<<"4.Delete a data element from an object list\n"
        <<"\t"<<"5.List all the objects in the file\n"
        <<"\t"<<"7.Quit\n\n";
}
```

```
void AddObject(char *newObj) {
    //establishing connection with files ...
    input.open(datFile, ios::in);
    append.open(datFile, ios::app);
    out_tmp.open(tmpFile, ios::out);
    app_tmp.open(tmpFile, ios::app);
```

```
//checking the file modes ...
if ((input.fail())|| (append.fail())) {
```

```

cerr<<"Cannot open "<<datFile<<endl;
exit(-1);
}
if ((out_tmp.fail())||app_tmp.fail()){
cerr<<"Cannot open "<<tmpFile<<endl;
exit(-1);
}
char *num = "\t0";
//adding the object ...
char *temp;
temp = new char[13];
while(!input.eof()) {
input>>temp;
if (strcmp(temp, newObj) == 0){
cerr<<"Object exists in file - please choose another option"<<endl;
exit(-1);
}
}
//end of 'while' for EOF
strcat(newObj, num);
append <<newObj<<"\n";
append.close();
cout<<newObj<<" added to the file."<<endl;
input.close();
}
//end of AddObject()

char *incrCount(char *oriLine) {
char cnt[10];
char *restLine = new char[1000];
char *bufLine = new char[1000];
char *tmpLine = new char[1000];
char *key = new char[1000];
key = strtok(oriLine, "\t");
char *num = strtok(NULL, "\t");
int i = atoi(num);
i++;
sprintf(cnt, "%d", i);
strcat(key, "\t");
strcat(key, cnt);
restLine =strtok(NULL, "\n");
if(restLine != NULL) {
strcpy(bufLine, restLine);
strcat(key, "\t");
strcat(key, bufLine);
}
return key;
}

```

```

AddData(char *existObj) {

//declaring & initializing variables et al.
int flag = 0,           //to check if the keyword exists
    line_len = 0,       //line length
    key_len = 0,        //key length
    i = 0,
    count = 0;
char c;
char str[10];
char *chg = new char[1000];
char *data = new char[25],
    *data_tmp = new char[25],
    *keyword = new char[13],
    *line = new char[1000], //a line from the data file in which the
                            //keyword exists-dynamically allocated memory.
    *dup_line = new char[1000], //used for copying 'line'
    *buf_line = new char[1000], //used for copying 'line'
    *mod_line = new char[1000]; //used for copying 'line'

//establishing connection with files ...
input.open(datFile, ios::in);
append.open(datFile, ios::app);
app_tmp.open(tmpFile, ios::app);
out_tmp.open(tmpFile, ios::out);

//getting defensive ...
if((input.fail())||!(append.fail())){
    cerr<<"Cannot open "<<datFile<<endl;
    exit(-1);
}
if((out_tmp.fail())||!(app_tmp.fail())){
    cerr<<"Cannot open "<<tmpFile<<endl;
    exit(-1);
}
memset(line, '\0', 1000);
memset(mod_line, '\0', 1000);
memset(dup_line, '\0', 1000);
memset(buf_line, '\0', 1000);

//..adding data ...
input.get(c);           //reading first char from input(data) file
while(!input.eof()) {
    if(c == '\n') {

```

```

i = 0;
memset(dup_line, '\0', 1000);
memset(keyword, '\0', 13);
strcpy(dup_line, line);
keyword = strtok(dup_line, "\t");
if (strcmp(keyword, existObj) == 0)
{
    flag = 1;
    cout<<"Please enter the data in the following format: \n"
        <<"\t <year,value>\nwhere 'year' is a four-digit integer"
        <<"and 'value' is an integer.\n"<<endl;
    cin>>data;
    memset(mod_line, '\0', 1000);
    strcpy(mod_line, line);
    strcpy(buf_line, line);
    out_tmp<<mod_line<<endl;
    chg = incrCount(buf_line);

    strcat(chg, "\t");
    strcat(chg, data);
    app_tmp<<chg<<endl;
    system("sort YearVal.tmp YearVal.dat > YearVal.buf");
    system("uniq -u YearVal.buf > YearVal.dat");
    unlink(bufFile);
} // end of 'if' for keyword match
memset(line, '\0', 1000);
} //end of 'if' for end-of-line
else
    line[i++] = c;
    input.get(c);
} //end of while for EOF
if (flag != 1)
    cout<<"Sorry, the object doesn't exist in file!\n"<<endl;
input.close();
append.close();
out_tmp.close();
app_tmp.close();
unlink(bufFile);
} //end of AddData()

void rmObject(char *rmObj) {

    //variable declarations and initializations(sigh!)
    int i = 0,
        flag = 0, //used for checking the presence of keyword

```

```

    key_len = 0;           //length of a line
char c,
    *key,
    *keyword,
    *line,
    *dup_line,
    *new_line,
    *mod_line;
key = new char[13];
keyword = new char[13];
line = new char[1000];
dup_line = new char[1000];
mod_line = new char[1000];
new_line = new char[1000];

//establishing connection with the files...
input.open(datFile, ios::in);
append.open(datFile, ios::app);
out_tmp.open(tmpFile, ios::out);
app_tmp.open(tmpFile, ios::app);

//checking the file modes ...
if ((input.fail())||(append.fail())) {
    cerr<<"Cannot open "<<datFile<<endl;
    exit(-1);
}
if ((out_tmp.fail())||(app_tmp.fail())){
    cerr<<"Cannot open "<<tmpFile<<endl;
    exit(-1);
}

memset(keyword, '\0', 13);
memset(line, '\0', 1000);
memset(dup_line, '\0', 1000);
memset(mod_line, '\0', 1000);
memset(new_line, '\0', 1000);

//removing the object ...
input.get(c);           //getting first character from file
while(!input.eof())
{
    line[i++] = c;
    if(c=='\n')
    {
        //if the end of the line is reached
        i = 0;
        strcpy(dup_line, line);
    }
}

```



```

key = strtok(dup_line, "\t");
strncpy(keyword, key, strlen(rmObj));
if (strcmp(keyword, rmObj) == 0) {
    flag = 1;
    strcpy(mod_line, line);
    out_tmp<<mod_line<<endl;
    system("sort YearVal.dat YearVal.tmp > YearVal.buf");
    system("uniq -u YearVal.buf > YearVal.dat");
    unlink(bufFile);
    cout<< rmObj <<" - removed from the file"<<endl;
    memset(mod_line, '\0', 1000);
} //end of if for keyword check
memset(dup_line, '\0', 1000);
memset(line, '\0', 1000);
memset(keyword, '\0', 13);
memset(key, '\0', 13);
} //end of if for a line
input.get(c);
} //end of while for eof

if(flag == 0)
    cout<<"Object doesnt exist in file - cannot be deleted!"<<endl;
input.close();
exit(-1);
} //end of rmObject() function.

char *decrCount(char *oriLine) {
    char cnt[10];
    char *restLine = new char[1000];
    char *bufLine = new char[1000];
    char *tmpLine = new char[1000];
    char *key = new char[1000];
    key = strtok(oriLine, "\t");
    char *num = strtok(NULL, "\t");
    int i = atoi(num);
    i--;
    sprintf(cnt, "%d", i);
    strcat(key, "\t");
    strcat(key, cnt);
    restLine =strtok(NULL, "\n");
    if(restLine != NULL) {
        strcpy(bufLine, restLine);
        strcat(key, "\t");
        strcat(key, bufLine);
    }
    return key;
}

```

```

}

void rmData(char *Obj, char *data) {

//variable declarations and initializations
int i = 0,
    flag = 0,                //for checking the existence of an object
    p_len = 0,
    diff_len = 0,
    line_len = 0;
char c,
    *chg,
    *keyword,
    *line,
    *dup_line,
    *mod_line,
    *new_line,
    *buf_line,
    *rest_line;
keyword = new char[13];
line = new char[1000];      //stores a line from the data file
dup_line = new char[1000]; //used to get the keyword
new_line = new char[1000];
mod_line = new char[1000]; //output to .tmp file
buf_line = new char[1000];
rest_line = new char[1000];
chg = new char[1000];

//establishing connection with the files...
input.open(datFile, ios::in);
append.open(datFile, ios::app);
out_tmp.open(tmpFile, ios::out);
app_tmp.open(tmpFile, ios::app);

//checking the file modes ...
if ((input.fail())||!(append.fail())) {
    cerr<<"Cannot open "<<datFile<<endl;
    exit(-1);
}
if ((out_tmp.fail())||!(app_tmp.fail())){
    cerr<<"Cannot open "<<tmpFile<<endl;
    exit(-1);
}

memset(line, '\0', 1000);

```

```

memset(keyword, '\0', 13);
memset(dup_line, '\0', 1000);
memset(buf_line, '\0', 1000);
memset(mod_line, '\0', 1000);
memset(new_line, '\0', 1000);
memset(rest_line, '\0', 1000);

input.get(c);
while(!input.eof())
{
    line[i++] = c;
    if(c == '\n')
    {
        //if the end of the line is reached
        i = 0;
        line_len = strlen(line);
        strcpy(buf_line, line);
        strcpy(dup_line, line);
        keyword = strtok(dup_line, "\t");        //dup_line destroyed
        if(strcmp(keyword, Obj) == 0)
        {
            flag = 1;
            strcpy(mod_line, line);
            //cout<<"mod_line inside kw search: "<<mod_line<<endl;
            line_len = strlen(mod_line);
            out_tmp<<mod_line<<endl;
            char *p = strstr(mod_line, data);
            if (p == NULL)
            {
                cerr<<"Data not found in this object list\n"<<endl;
                exit(-1);
            }
            char *d = strstr(p, "\t");
            if (d != NULL)
            {
                strcpy(rest_line, d);
                p_len = strlen(p-1);
                diff_len = line_len - p_len;
                strncpy(new_line, buf_line, diff_len);
                strcat(new_line, rest_line);
            }

            else if (d==NULL)
            {
                p_len = strlen(p-1);
                diff_len = line_len - p_len;
                strncpy(new_line, buf_line, diff_len);
            }
        }
    }
}

```

```

    }

    chg = decrCount(new_line);
    app_tmp<<chg<<endl;
    memset(keyword, '\0', 13);
    system("sort YearVal.tmp YearVal.dat > YearVal.buf");
    system("uniq -u YearVal.buf > YearVal.dat");
    unlink(bufFile);
    exit(-1);

} //end of if for keyword match
memset(dup_line, '\0', 1000);
memset(mod_line, '\0', 1000);
memset(buf_line, '\0', 1000);
memset(new_line, '\0', 1000);
memset(rest_line, '\0', 1000);
memset(line, '\0', 1000);
} //end of 'if for one line
input.get(c);
} //end of while for eof

if(flag != 1)
    cout<<"Object doesnt exist in file - cannot be deleted!"<<endl;
input.close();
out_tmp.close();
menu();
exit(-1);
} //end of rmData()

void ListObject(void) {

//variable initializations and declarations ...
char c;
int i=0,
    j=0,
    key_len = 0;
char *line,
    *keyword,
    *object;
line = new char[1000]; //dynamically allocating memory
keyword = new char[13];
object = new char[13];

input.open(datFile, ios::in); //establishing the connection
if (input.fail()) {

```

```

cerr<<"Cannot open "<<datFile<<endl;
exit(-1);
}
memset(object, '\0', 13);
memset(keyword, '\0', 13);
memset(line, '\0', 1000);

input.get(c);                //get first character

while(!input.eof()){
    if(c != '\n')
        line[i++] = c;
    else {
        keyword = strtok(line, "\t");
        i = 0;
        if (keyword != NULL){
            strcpy (object, keyword);
            memset(line, '\0', 1000);
            memset(keyword, '\0', 13);
            cout<<object<<endl;
            memset(object, '\0', 13);
        }
        // end of if for keyword
    }
    //end of else for eol
    input.get(c);
}
//end of while for eof
input.close();
//exit(0);
}
//end of ListObject() function

```

```

int main(void) {

    menu();

    int choice;
    cin>>choice;

    switch(choice) {

    case 1:
        cout<<"\nPlease enter the object name: "<<endl;
        char *newObject;
        newObject = new char[13];
        cin >> newObject;
        AddObject(newObject);
        break;

```

case 2:

```
cout<<"\nPlease enter the object to which data is to be added: "<<endl;
char *existObj;
existObj = new char[13];
cin >> existObj;
AddData(existObj);
break;
```

case 3:

```
cout<<"\nPlease enter the object to be deleted: "<<endl;
char *delObject;
delObject = new char[13];
cin >> delObject;
rmObject(delObject);
break;
```

case 4:

```
cout<<"\nPlease enter the object from which data is to be deleted: "
<<endl;
char *rmObj;
rmObj = new char[13];
cin >> rmObj;
cout<<"\nPlease enter the data to be deleted: "<<endl;
char *data;
data = new char[25];
cin >> data;
rmData(rmObj, data);
break;
```

case 5:

```
ListObject();
break;
```

case 7:

```
cout<< "Have a nice day!"<<endl;
exit(0);
```

default:

```
cout<<"Sorry, invalid option!"<<endl;
menu();
}
//unlink(tmpFile);
return 0;
}
```

## DP9.java

```
import java.applet.Applet;
import java.awt.*;
public class DP9 extends Applet
{
    int i, j,
        x, y,
        nElem,           // the number of x or y coordinates read
        x1 = 0;
    int a[] = new int[1000];           // used for storing the data as integers
    int b[] = new int[1000];
    private String sx[] = new String[1000]; // used for reading in the parameters
    private String sy[] = new String[1000];

    ////////////////////////////////////////////////////////////////////
    // This method receives the data passed to the applet by the CGI program
    // GetData in the form of parameters, parses the data into integer
    // values, and stores them in separate arrays.
    ////////////////////////////////////////////////////////////////////

    public void getParam()
    {
        String MX,
            MY,
            num;
        String tempA,
            tempB;
        int t1, t2;
        num = getParameter("numElem");
        MX = getParameter("MaxX");
        MY = getParameter("MaxY");
        nElem = Integer.parseInt(num);
        x = Integer.parseInt(MX);
        y = Integer.parseInt(MY);
        for (int k=0; k<nElem; k++)
        {
            tempA = getParameter("a" + k); // obtaining the integer from
```

```

// the parameter.
    sx[k] = tempA;
    t1 = Integer.parseInt(tempA);
    a[k] = t1;
    tempB = getParameter("b" + k);
    sy[k] = tempB;
    t2 = Integer.parseInt(tempB);
    b[k] = t2;
}
}

/////////////////////////////////////////////////////////////////
// This method initializes the program. It also scales the data for the
// graph.
/////////////////////////////////////////////////////////////////

public void init()
{
    getParam();
    for(i = 0, j = 50; i < nElem; i++,j+=50)
    {
        a[i] = (a[i]%100 + j);
        if (x1 < a[i]) x1 = a[i];
    }
    for (i=0; i<nElem; i++)
        a[i]+=100;
    for (i=0; i<nElem; i++)
        b[i]= y-b[i]+100;
}

/////////////////////////////////////////////////////////////////
// This method uses the data in the arrays to draw the graph on the
// client browser.
/////////////////////////////////////////////////////////////////

public void paint(Graphics g)
{
    g.setColor(Color.red);           //setting the color of graph to red
    g.drawPolygon(a, b, nElem);      //drawing the graph
    g.setColor(Color.black);        //setting the color of axes to black
    g.drawLine(100, y+150, x+90, y+150); //drawing the x-axis
}

```



```
g.drawLine(100, y+150, 100, 20); //drawing the y-axis
for (i = 0; i<nElem; i++) //printing out values of x & y coordinates
{
  g.drawString(sx[i], a[i], y+115);
  g.drawString(sy[i],50 , b[i]);
}
}
}
```

## VITA

Sai P. Battina

Candidate for the Degree of

Master of Science

Thesis: A JAVA APPLLET FOR GENERATING GRAPHS FROM A UNIX FILE

Major Field: Computer Science

Biographical:

Education: Received Bachelor of Engineering degree in Mechanical Engineering from Osmania University, Hyderabad, India in June 1992. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 1997.

Experience: Employed by Oklahoma State University, Computing and Information Services as a Computer Lab Consultant from August 1995 to May 1997; employed as a Engineer(Materials) by M/s. Prudential Mouli Sugars Ltd., Hyderabad, India from January 1993 to July 1994.