2009

Integration of mapping web services and scalable vector graphics

Suneetha Chinamuthevi

INTEGRATION OF MAPPING WEB SERVICES AND SCALABLE VECTOR GRAPHICS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Suneetha Chinamuthevi

March 2009

INTEGRATION OF MAPPING WEB SERVICES AND SCALABLE VECTOR GRAPHICS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Suneetha Chinamuthevi

March 2009

Approved by:

[redacted]

Dr. Richard Botting, Chair, Computer Science and Engineering

Feb 12/2009
Date

[redacted]

Dr. George Georgiou

[redacted]

Dr. Ernesto Gomez

ABSTRACT

The purpose of this Master's Project was to develop a web-based application integrating mapping web services and SVG (Scalable Vector Graphics). This project helps a user locate a set of addresses anywhere in the United States on a map.

The underlying mapping and address locator software systems are very complex, requires large data sets and updates on regular basis. The web services technology encapsulates these complexities and exposes the required functionality in a secure manner over the internet. The main stream mapping websites such as Google and Yahoo provides the ability to enter one address at a time interactively, but can't upload a file with multiple addresses. Also, their cartography is predefined; there is no ability to change color and sizes. There are several commercial desktop software packages are available to address this complex cartographic functionality. However, SVG (Scalable Vector Graphics) royalty-free vendor-neutral software provides very rich cartographic functions in a web browser environment. An attempt was made to integrate mapping Web Services and SVG to enable the user to upload the address data using the Web browser, locate them on map with a specified color and size.

ACKNOWLEDGMENTS

Thanks to all my professors for being patient with me and for the encouragement. Due to my family commitments and my health I had to take long break between the course work and final milestone of Project for the graduation. Dr. Richard Botting and Dr. George Georgiou acted as my project advisors and helped a lot during the whole master's process. I thank Dr. Ernesto Gomez and Dr. Josephine Mendoza for reviewing my project work. Also, to ESRI (Environmental Systems Research Institute, Redlands, CA) for promoting Web Services technology, and giving me the required access to their web services.

Of course, without the help and support of my family, I would have never completed this project. My sincere appreciation to my husband Kumar and to my wonderful kids Surya and Saketh from the bottom of my heart.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

1.1.1 Purpose

The project focuses on integrating customized data on top of a mapping interface using public domain services. The end product was a web application. The user accomplishes this functionality without installing any desktop software. This task was accomplished by integrating Java Technology, Web services and SVG. The goal of the project was to integrate mapping Web Services and SVG. In the popular web applications such as maps.yahoo.com, maps.google.com, mapquest.com, or maps.msn.com, the user can interactively type in a single address and find its corresponding location on the map. If the user has multiple addresses, he can't upload that data and find corresponding locations simultaneously on the map. More over the user is constrained by the symbols and rendering features provided by them. The user may not have the ability to change the color and size of the dots on the map.

The project provides an interface in a browser environment to upload the data of multiple customers,

geo-code and map them using web services. Several commercial vendors provide the geo-coding and mapping web services. An attempt was made to use public domain services wherever possible. SVG provides a powerful framework to render the graphics in a browser-based environment. Custom tools were provided using SVG to change the color of graphics (geo-coded locations) or the size of the graphics or to render the locations using the different columns in the uploaded file.

1.1.2 Scope

The project tried to use open source and public domain resources as far possible. The possible public domain web services are identified. The SVG rendering is limited to color, size and possibly shapes such as rectangle, circle or triangles. The locations on the map are symbolized using list of colors with different sizes, perhaps rendering can be done using different shapes.

1.1.3 Definitions, Acronyms, and Abbreviations

The table of definitions, acronyms, and abbreviations used in the document as follows:

- GEOCODING [13] is the process of adding the latitude and longitude (spatial position) to an address. This process is also known as ADDRESS

MATCHING, the ability to locate an address on a map.

- WEB SERVICES [13] are defined by a set of open standard (XML, SOAP, etc.) based Web applications that interact with other web applications for the purpose of exchanging data. Initially used for the exchange of data on large private enterprise networks, web services are evolving to include transactions over the public Internet.
- SOAP (Simple Object Access Protocol) [13] is a way for a program running in one kind of operating system (such as Windows 2000) to communicate with a program in the same or another kind of an operating system (such as Linux) by using the World Wide Web's Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML) as the mechanisms for information exchange. Since Web protocols are installed and available for use by all major operating system platforms, HTTP and XML provide an already at-hand solution to the problem of how programs running under different operating systems in a network can communicate with each

other. SOAP specifies exactly how to encode an HTTP header and an XML file so that a program in one computer can call a program in another computer and pass it information. It also specifies how the called program can return a response.

- XML (Extensible Markup Language) [13] is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. For example, computer makers might agree on a standard or common way to describe the information about a computer product (processor speed, memory size, and so forth) and then describe the product information format with XML. Such a standard way of describing data would enable a user to send an intelligent agent (a program) to each computer maker's Web site, gather data, and then make a valid comparison. XML can be used by any individual or group of individuals or companies that wants to share information in a consistent way.
- SVG (Scalable Vector Graphics) [11] is a language for describing two-dimensional graphics

and graphical applications in XML. SVG files are compact and provide high-quality graphics on the Web, in print, and on resource-limited handled devices. In addition, SVG supports scripting and animation, so is ideal for interactive, data-driven, personalized graphics. SVG is a royalty-free vendor-neutral open standard developed under the W3C (World Wide Web Consortium) Process.

- The Java Servlets [2] allow a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML.
- JSP (JavaServer Pages) [4] is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.
- CSV (Comma separated values) [13] indicates a computer data file used for storage of data structured in a table form. Each line in the CSV file corresponds to a row in the table. Within a

line, fields are separated by commas, each field belonging to one table column.

- WAR File [13] a WAR file (which stands for "web application archive") is a JAR file used to distribute a collection of JavaServer Pages, servlets, Java classes, XML files, tag libraries and static Web pages (HTML and related files) that together constitute a Web application.
- JAR File [13] a JAR file (or Java Archive) is used for aggregating many files into one. It is generally used to distribute Java classes and associated metadata.
- APACHE TOMCAT [13] a web container that functions as a web server supporting servlets and JSPs
- J2EE [6] Java 2 Platform, Enterprise Edition includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc, and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, servlets, portlets (following the Java Portlet specification), JavaServer Pages

and several web service technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies.

1.2 Overall Description

1.2.1 Project Perspective

1.2.1.1 System Interfaces. The web application can be deployed in any j2EE compliant application servers. Users are able to access the Web pages and run the application with Internet Explorer with SVG plug-in installed.

1.2.1.2 User Interfaces. The Web application can be invoked in by entering a URL http://hostname:port/wssvg. The interface provides an input control to enter the text file and takes the user through the rest of the workflow. The responses are served using Java Server Pages from server.

1.2.1.3 Hardware Interfaces. This project was written and tested on a Windows Computer, under Java environment. This program may run on any machine that has j2EE compliant application server installed. The system specifications are governed by application server used. The minimum system requirements to run Tomcat are shown below.

Table 1. Minimum System Requirements for Tomcat

Operating Platform	Processor Speed	Memory (Minimum Requirement)	Hard Disk Space Required (For Installation)	Extra Disk Space Required (For Execution)
Windows	733 MHz	256 MB RAM	320 MB	200 MB
Linux	733 MHz	256 MB RAM	320 MB	200 MB

1.2.1.4 Software Interface. This project was designed to run over the Internet. By writing it in Java [1], the code will run on any computer independent of the operating system. As long as the user's computer has an Internet connection, and appropriate browser environment, there will be no other limits on the users system. Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture. The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance

with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler. The requests are processed by Servlets on the server side and the dynamic content is served by Java Server pages. The application requires any J2EE complain application server to serve the Servlets and JSPs (Java Server Pages). On server side JDK 1.3.1 or higher is needed for the application server.

1.2.1.5 Communications Interfaces. Figure 1, the Deployment Diagram, shows the relationship between the Web Application, Web Services and the Internet user. To access this site, the user needs an Internet connection to the Web Application. The web address is: URL http://hostname:port/wssvg. If the web application is running at port 80, port number is not required in the URL. The user will also need a web browser with SVG plug-in installed, such as, Internet Explorer versions 6.0 or higher. The web server will communicate with Web services over the internet.

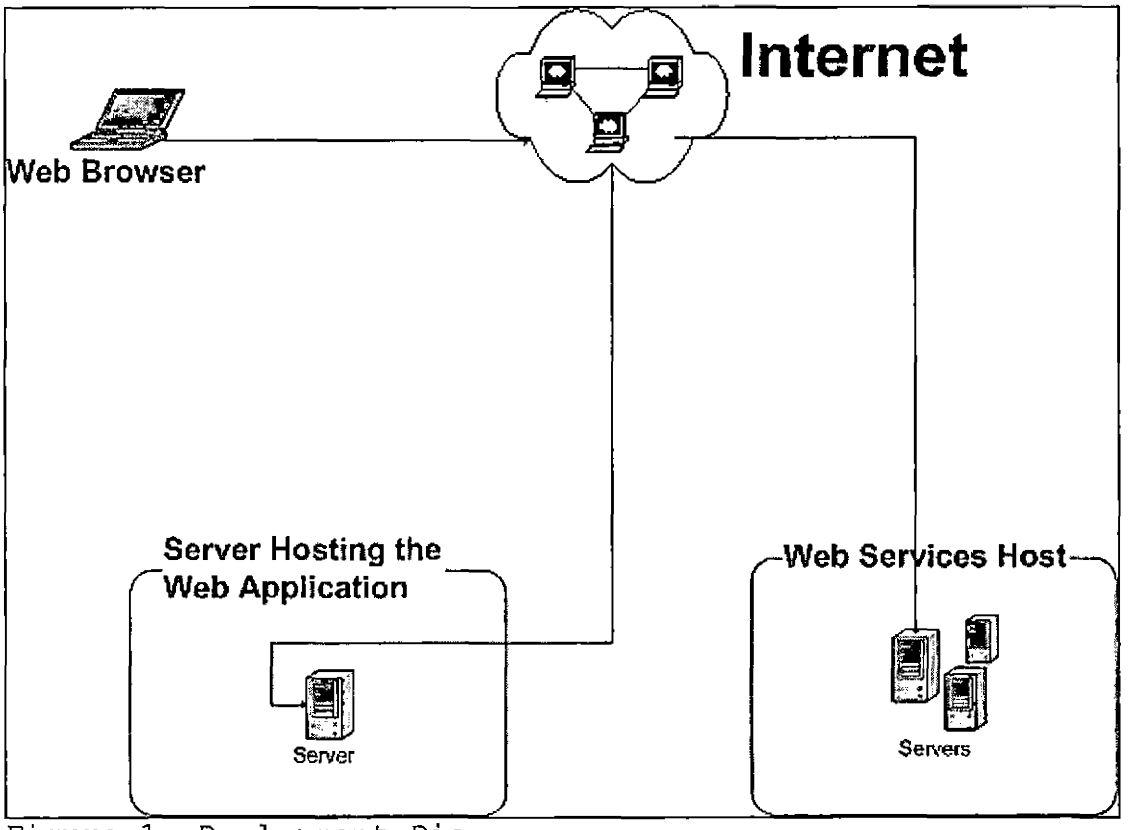

Figure 1. Deployment Diagram

1.2.1.6 Memory Constraints. The minimum RAM, Random Access Memory, tested, for the client side, was 512 Megabytes. However, any computer with enough memory to effectively use the Internet should be able to use this program.

1.2.1.7 Operations. The Web application will be accessed on a server on the World Wide Web. It will remain active as long as the browser containing the URL is running and the user has an internet connection. The program is interactive and needs the user to interface

with it. Currently, the program does not use a database. Therefore, there are no backup or recovery operations required. Once running, user needs to upload the data containing address information. The subsequent pages are generated by Java Server Pages. However, if a problem should occur, simply using the browser refresh button will reset the web page.

1.2.1.8 Site Adaptation Requirements. This project is being written and tested with windows machine with Windows XP Operating System and Internet Explorer 7.0. Earlier versions of Internet Explorer have not been tested.

1.2.2 Project Functions

The application will display the address location on a map with a selected color and size. The address information is uploaded by the user. This application will also provide an interactive environment to display maps. The maps are obtained over the internet using Web Services technologies. An overview of the system and user interaction [7] is shown in figure 2.

1.2.3 User Characteristics

The audience for this project can be anyone who wants to display the address information on a map. The user needs understand how to prepare the input with the address information. The input file will be a comma delimited

format. The user needs to correlate the information in various columns with header information.

1.2.4 Constraints

The Web application developed and tested on Internet Explorer with SVG plug-in. The application will not work, if this plug-in not installed.

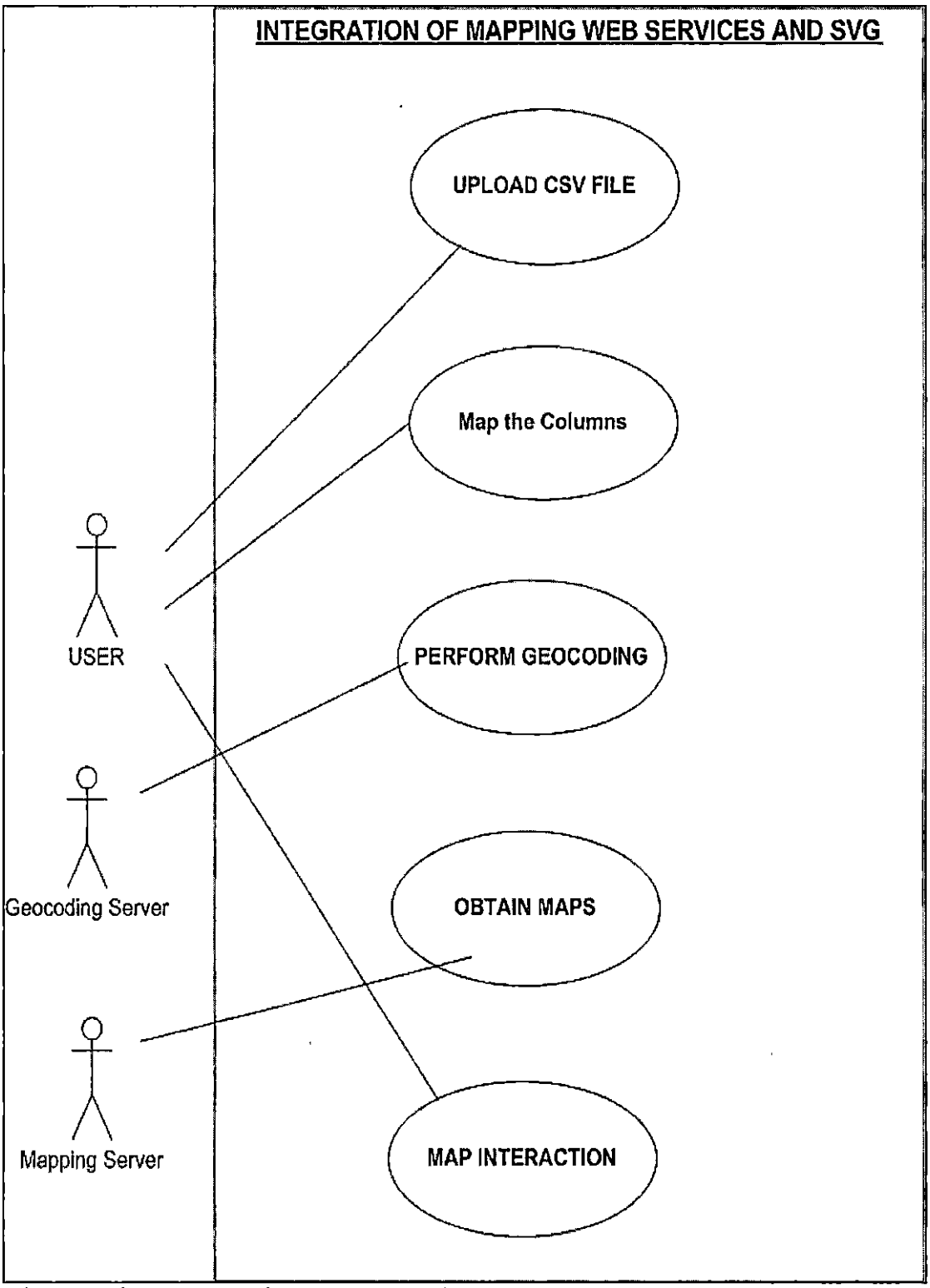

Figure 2. Case Diagram Showing the Overview

1.2.5 Assumptions and Dependencies

The application requires internet to communicate with the Web Services for obtaining maps and to perform geocoding using SOAP protocol. If those external services are not accessible, the application will not work.

1.2.6 Apportioning of Requirements

In the context of this project, there are no elements that are being delayed until future versions are developed. The program code has been finalized at this time. However, there are several items that could be improved by another computer science student in the future.

1.3 Specific Requirements

1.3.1 External Interfaces

The entry page for the application is shown below. The user needs to upload the data with the address information.

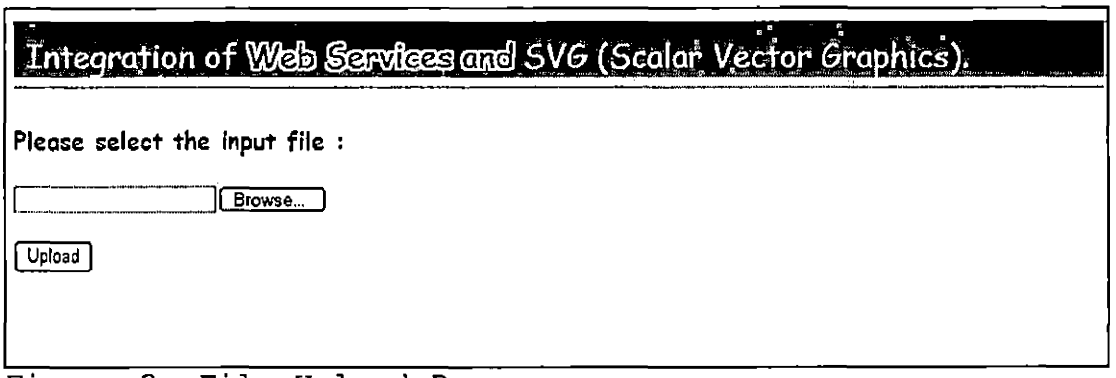

Figure 3. File Upload Page

The next page the user needs to map columns, the text above those drop downs describes the requirement. Once the columns are mapped, geocoding is performed.

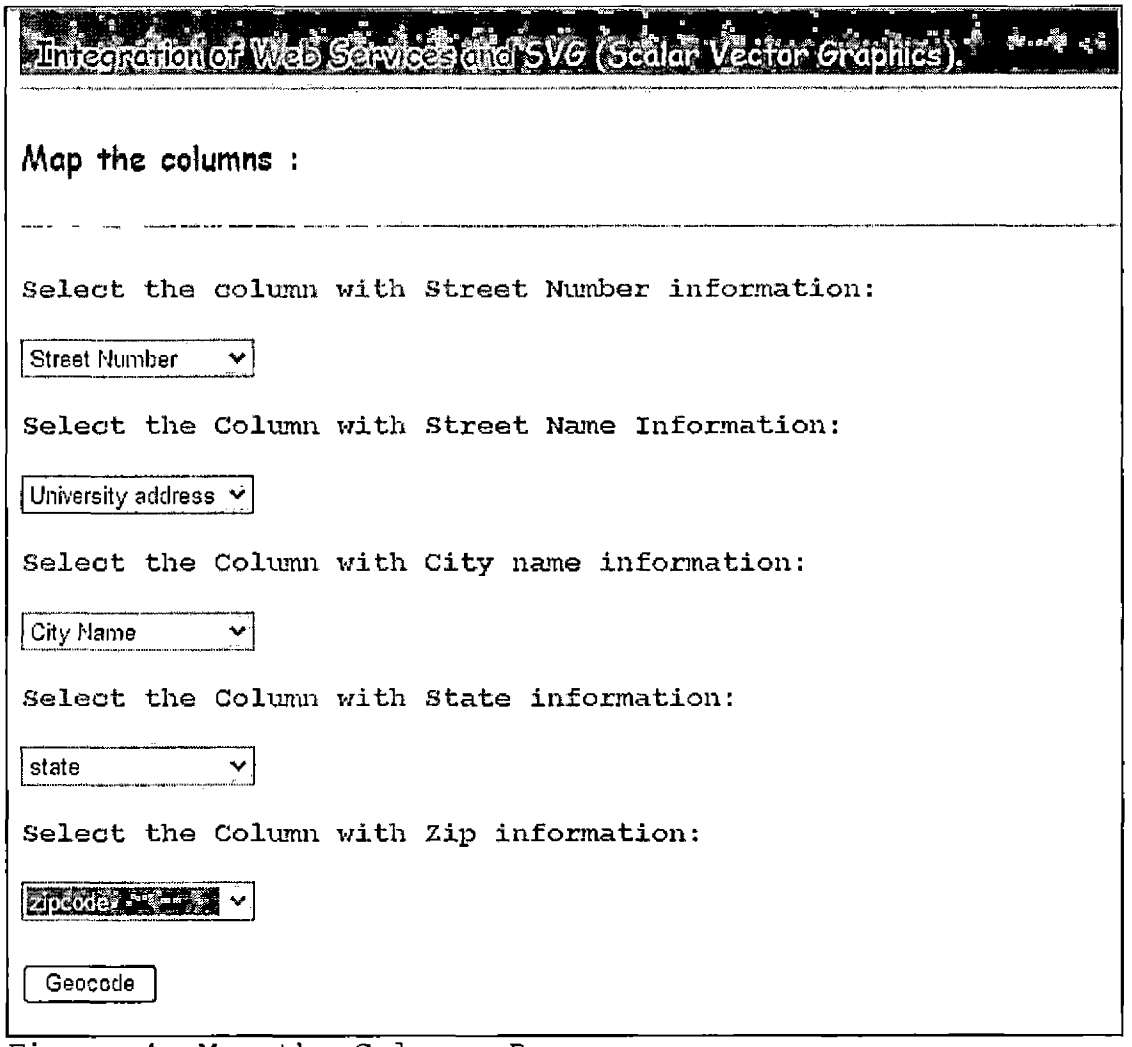

Figure 4. Map the Columns Page

Final step before Mapping is to select the color and size of dots.

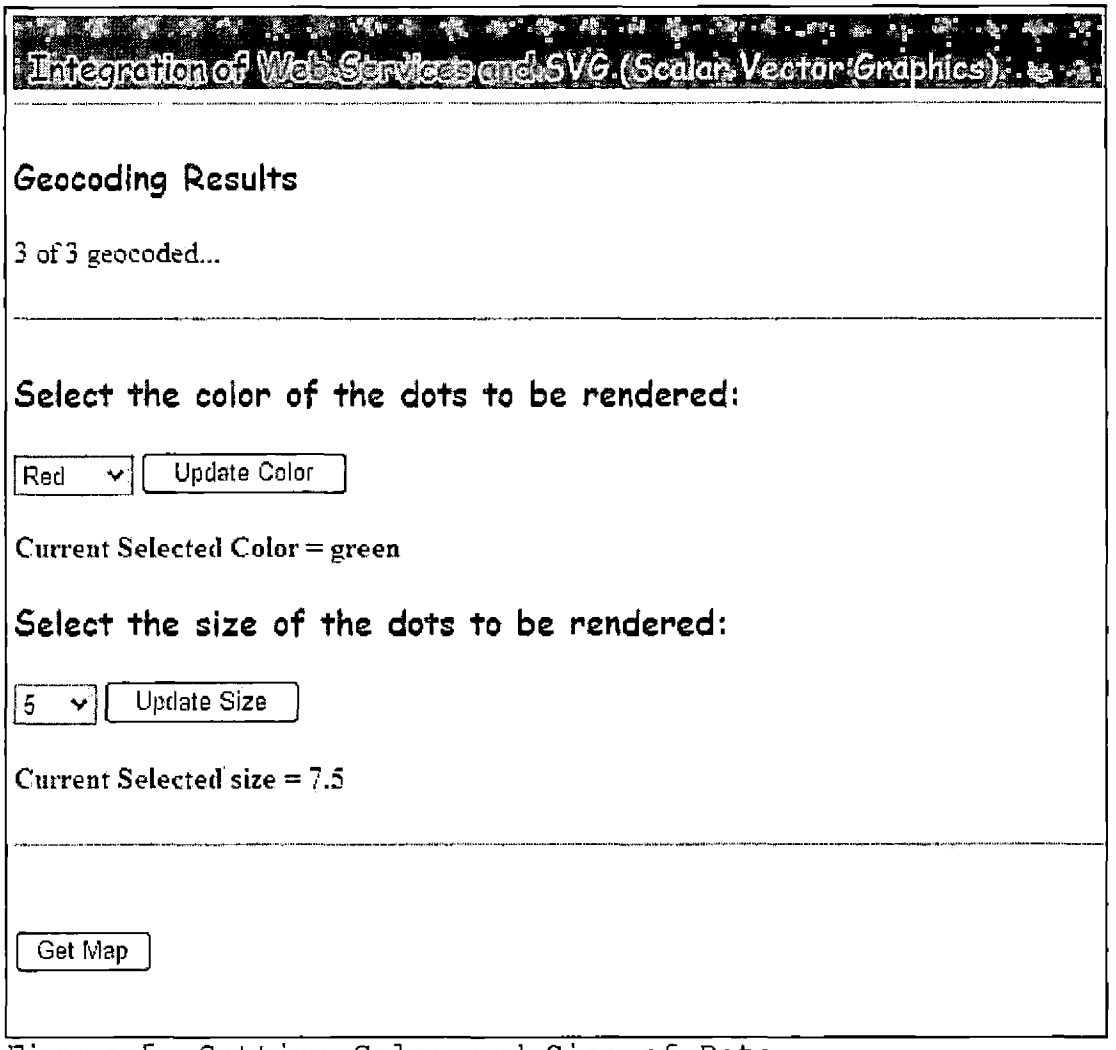

Figure 5. Setting Color and Size of Dots

The final page in the application is SVG mapping page, maps are obtained for the area where address information is required. The address locations are drawn on top the map using SVG specifications. In fact the layout shown below was created using SVG specifications.

To interact with the map, several functions are available at the top of the top.

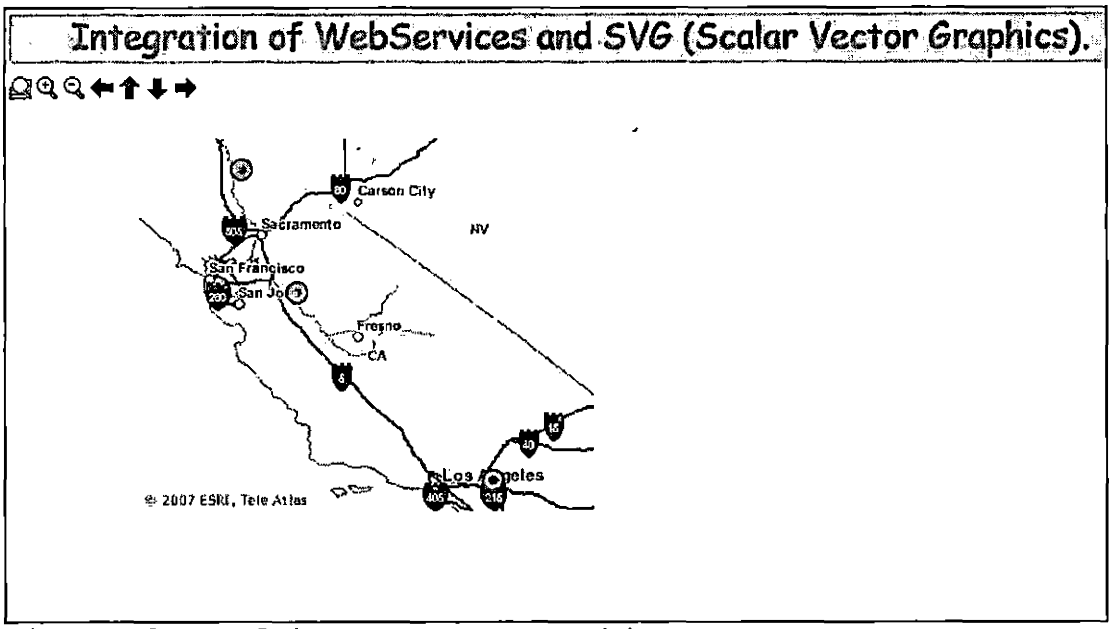

Figure 6. Scalable Vectors Graphics Map Page

1.3.2 Functions

The input for the system a file containing the address information and user has the ability to map the columns. Several checks are performed on server side to prevent failures. Abnormal situations like communication failures may occur, those errors are communicated to the user using error.jsp.

1.3.3 Performance Requirements

The number of simultaneous users supported is limited by the bandwidth of the Web server and is beyond the scope of this project.

1.3.4 Logical Database Requirements

The Web application has no database requirement.

1.3.5 Design Constraints

In the design phase, use case diagrams were used. A use case diagram is a type of behavioral diagram defined by the Unified Modeling Language (UML) and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actors. Roles of the actors in the system can be depicted. Then appropriate technologies and frameworks are selected to achieve this functionality. The processes involved in software development are identifying the classes and objects needed, identifying the relationships among these classes, and specifying the interfaces required.

1.3.6 Software System Attributes

1.3.6.1 Reliability. The reliability of the system was verified through extensive testing of all features. The accuracy the geo-coding translation of address into latitude and longitude depends on input data. The incomplete data may lead to inaccurate results. If the input address information is accurate, the dots are displayed in correct location.

1.3.6.2 Availability. This application is available anytime the Web server is running.

1.3.6.3 Security. There is no security, such as passwords, required to access the web site. However, the Web Services require authentication and each request to the Web Service will be authenticated with a user name and password. This transaction is transparent to the user; web application performs that authentication on the behalf of user.

1.3.6.4 Maintainability. The web application runs in the Java (JDK 1.3 or higher) environment. The application was developed under j2EE specifications; application needs to be tested for any deprecated java classes. There are no other maintenance requirements for this application.

1.3.6.5 Portability. Java as a programming language was designed to run on most platform types. For the

Internet user, the code will port to any computer with the proper web enabled browser. For the user, SVG plug-in for the browser is required.

CHAPTER TWO

DESIGN

2.1 Architecture (Components of the System)

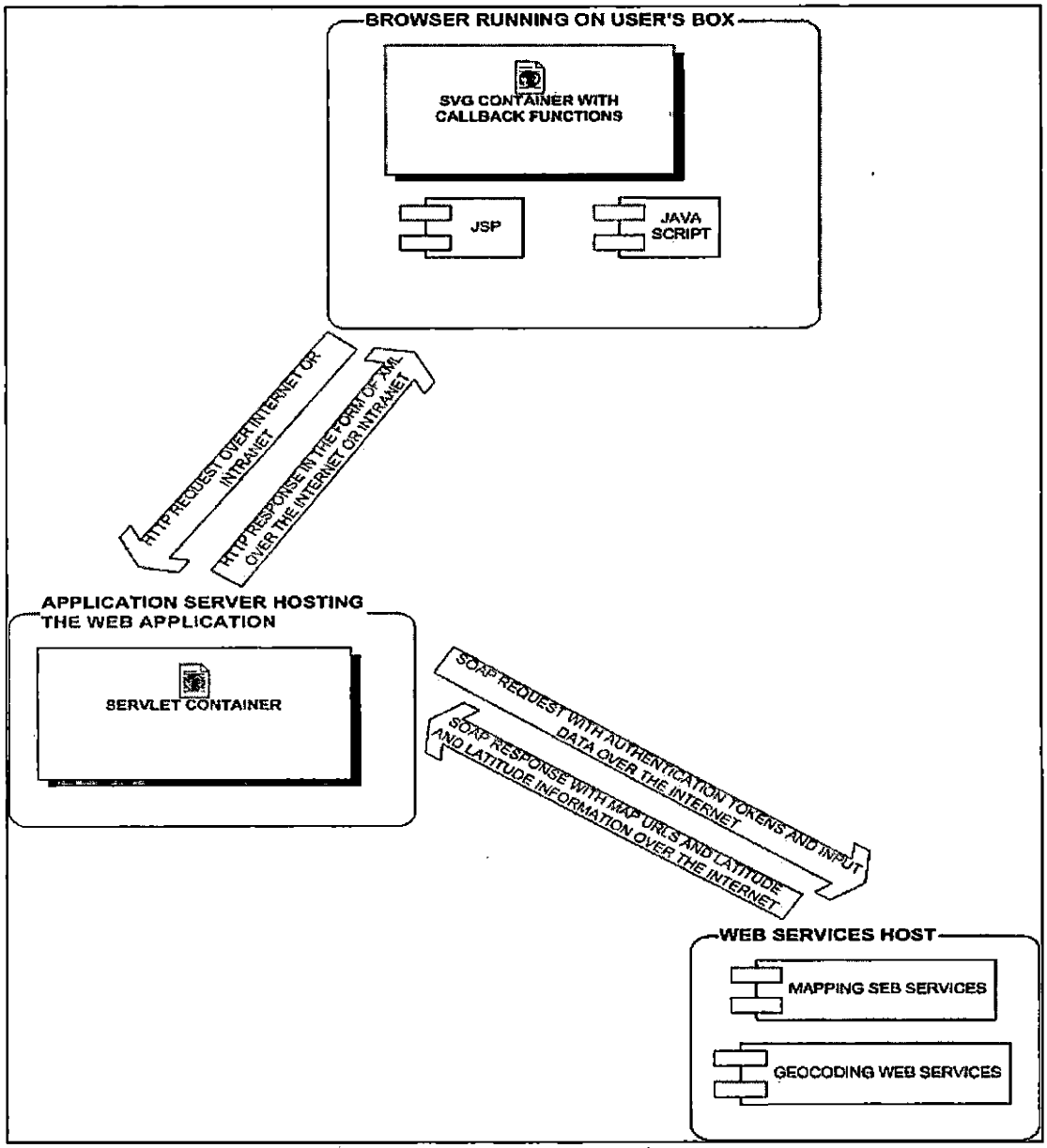

Figure 7. Component Diagram - Overview

In Figure 7, the component diagram, an overview of various components in the system is shown. The SVG plug-in for the browser provides SVG container with the browser, the MIME Type for SVG documents is image/svg+xml (contentType = "image/svg+xml"). The SVG specification uses XML to render the web page. The dynamic content in this SVG pages is provided by Java Server Pages, and mouse events are captured by java script functions. The SVG provide a function called getURL to communicate with the server. The function getURL() is a JavaScript function commonly used in SVG scripts to fetch another file. It allows scripts to download data without having to reload the SVG document. As simple as this concept may sound, it is very powerful. Using getURL it is possible for SVG graphics to be dynamically updated with remote data as a result of user interactions or timed data feeds. The syntax is as follows:

```
getURL(url, callback);
```

URL is a string containing the URL of the file that is to be fetched. Once getURL has finished downloading the file it will invoke the callback function and pass it an object. The appropriate URL is constructed based on user actions, the getURL sends the request to the Server, the Server returns SVG file in XML format (an example XML is

shown below in Figure 20), the callback function onMAP refreshes the various SVG elements, from the end user perspective map and address locations will be refreshed. As shown in the following XML has href location for the map image and dots locations with their sizes. The Geocoding server returns the address locations in latitude and longitude format and those coordinates are transformed into image units.

```
<g><image xlink:href="http://mesaarcweb.esri.com/out/maps/Tele_ArcWeb_US_f10063aps0010.phx1.aens.net2636732408.png"
x="100" y="100" width="330" height="270" />
<circle cx=" 400.8007276135316" cy=" 256.2730528527585" r=" 10 " stroke="black" stroke-width="2" fill=" green"/>
<circle cx=" 402.49999999999324" cy=" 262.0561570012643" r=" 10 " stroke="black" stroke-width="2" fill=" green"/>
<circle cx=" 127.49999999998795" cy=" 207.9438429987345" r=" 10 " stroke="black" stroke-width="2" fill=" green"/>
<circle cx=" 402.49999999999324" cy=" 262.0561570012643" r=" 10 " stroke="black" stroke-width="2" fill=" green"/>
</g>
```

Figure 8. Scalable Vector Graphics XML Returned by the Server

The web application was developed using Java servlet technology, advantages of Servlets over CGI technology are well documented. Several design patterns [3] and implementations are available for the servlet implementation. For the project Command pattern was employed. In this pattern only one Servlet called controller servlet was used. This controller servlet receives an parameter called "CMD". Based on command appropriate action is invoked. The business logic is included in the action class. In the figure 11 shown

below, the different URL string has as values such as getFile, MapColumns, Geocode and GetMap. The servlet forwards appropriate JSP based on command as shown in the figure 9.

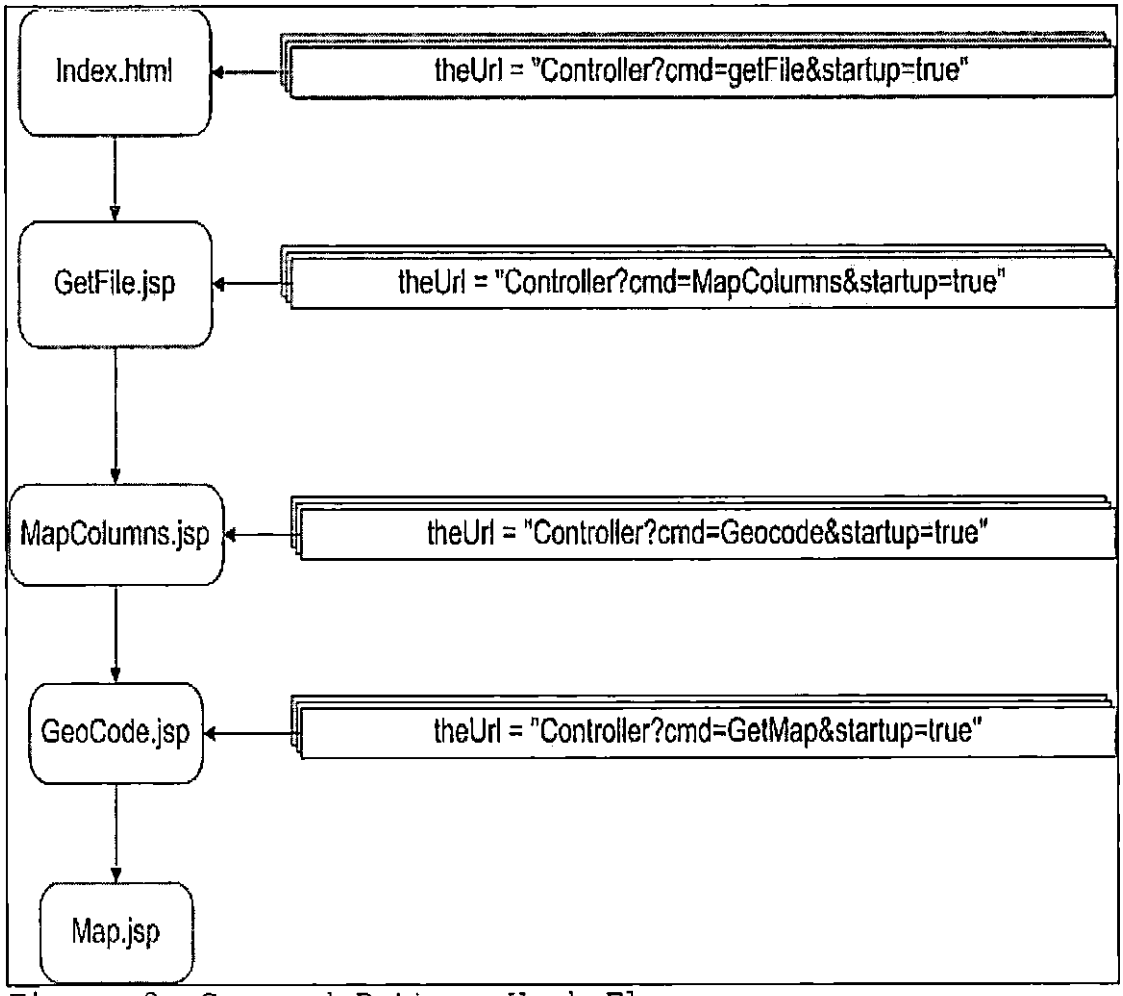

Figure 9. Command Pattern Work Flow

The Web application communicates with ESRI ArcWeb Services over the internet using SOAP protocol. A secure account was obtained with ESRI ArcWeb Servies and that

username and password will be used for authenticating the web services. The mapping services return the URL to the image location stored on ESRI web server such as http://mesaarcweb.esri.com/out/maps/Tele_ArcWeb_US_f10068aps0018.phx1.aens.net2636732408.png. This image URL will be passes to the browser using XML as shown in the Figure 8.

2.2 Use Case Diagrams

The tables 2 to 6 below describes detailed use cases for each steps performed in the project. In each of the use cases actors, steps and issues are discussed. Darek Coleman [8] suggested use case templates based on experience of the practical usage of use cases. The following diagrams were developed using those templates.

Table 2. Upload the Comma Separated Values FILE Use Case

Use Case	Upload CSV File
Description	User upload the CSV file using the Web Application
Assumptions	User will have CSV file created prior to using the web application. The CSV file will contain necessary address information required to Geo-code the locations. Also the CSV file may contain any additional columns that are useful in rendering.
Actors	User
Steps	1 User opens the browser 2 User logs into the web application 3 User clicks on Upload button 4 User navigates to the location of CSV file 5 User Clicks the OK button.
Issues	The exceptions are caught in the file formatting issues and the user will be notified.

Table 3. USE CASE - Map the Columns Use Case

Use Case	Map the Columns
Description	User matches the user defined columns with system defined columns. This data will be used to geo-coding by the web services and SVG rendering.
Assumptions	The user successfully uploaded the CSV file.
Actors	User
Steps	1 The user will select the following columns Address City State Zip Column to be rendered with SVG.
Issues	If the required columns are not mapped for geo-coding the next form will not be presented to the user.

Table 4. Perform Geo-coding Use Case

Use Case	Perform Geo-coding.
Description	The uploaded address information will be sent to Web services and resulting coordinate information is obtained.
Assumptions	The user successfully mapped the required columns required for address geocoding.
Actors	User, Geo-coding Server
Steps	1 The system will send the address information using appropriate web services protocols in batch mode. 2 The response from web services are captured by the system
Issues	If the geo-coding web services are not accessible, the User will be informed.

Table 5. Obtain Maps Use Case

Use Case	Obtain Maps
Description	Generate maps from the web services.
Assumptions	The geo-coding is performed successfully and coordinate information is obtained.
Actors	User, Mapping Server
Steps	1 The system will send the coordinate information using appropriate web services protocols to obtain the map. 2 The map response from web services are captured by the system and presented to the user.
Issues	If the mapping web services are not accessible, the User will be informed.

Table 6. Map Interaction User Case

Use Case	Map Interaction
Description	User can change symbology of locations using SVG
Assumptions	The maps are obtained successfully by the system and presented to the user.
Actors	User
Steps	1 The maps interface will presented to the user with custom button around it. 2 SVG plug-in for the browser needed to perform the customer rendering. 3 User can change the symbology based on predefined template of symbols on the interface.
Issues	SVG plug in needs to be installed.

2.3 Technologies Used in the Project

2.3.1 Apache File Upload Control

This control is part of Apache's Commons utilities set of reusable java components (http://commons.apache.org). The file upload component provides interface file upload capability to the servlets and the web applications. The complete user guide for this component can be found at http://commons.apache.org/fileupload/using.html. This control is used to parse the input CSV file from the user.

2.3.2 Apache AXIS for Web Services

Apache Axis is an implementation of the SOAP ("Simple Object Access Protocol"). SOAP is a lightweight protocol for exchanging structured information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. The detailed documentation for AXIS can be found at http://ws.apache.org/axis/java/index.html. ESRI developed SOAP API for mapping and geo-coding. The web application as a consumer uses Apache's Axis to get maps and to perform geo-coding.

2.3.3 Java Command Pattern

This design pattern was used as framework to deliver the web functionality. This topic is discussed above with reference to the figure 11.

2.3.4 Scalable Vector Graphics

SVG was used to render the map which is an end product of the workflow.

2.3.5 The NetBeans Integrated Development Environment

This IDE was used to develop the application.

2.3.6 Tomcat

The tomcat application server was used to deploy the application.

CHAPTER THREE

DEPLOYMENT AND OPERATIONS

3.1 Deployment Instructions

3.1.1 Introduction

A WAR file assists the Java server administrator in deploying a server-based application. It has the same format as a JAR file but has an additional manifest which instructs the application server how to map a particular URL to an application and which servlet class to run. When a WAR file is placed in the server's root folder, it is automatically extracted and prepared for use.

3.1.2 Prerequisites

A Java application server such as Apache Tomcat, IBM WebSphere or JBOSS must be used.

3.1.3 War File Structure

In addition to the HTML, JAR, class, servlet and/or JSP files which make up the application, a XML file deployment descriptor file must be created and included in the WAR file. The descriptor provides information to the application server about how to map a particular URL to the application, which servlet class to run plus additional per-application configuration parameters. It is named web.xml and must be placed in the WEB-INF

sub-directory of the WAR file. The web.xml for this project is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchem
   <servlet>
      <servlet-name>ViewerControllerServlet</servlet-name>
      <servlet-class>edu.csush.cs.scproject.servlet.ControllerServlet</servlet-class>
      <init-param>
         <description>ESRI Web Services login User Name</description>
         <param-name>username</param-name>
          <!--<param-value>Suneetha</param-value> -->
          <param-value>xxxx</param-value>
      </init-param>
      <init-param>
         <description>ESRI Web Services Pass word</description>
         <param-name>passwd</param-name>
          <param-value>xxxx</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>ViewerControllerServlet</servlet-name>
      <url-pattern>/Controller</url-pattern>
   </servlet-mapping>
   <session-config>
      <session-timeout>
         30
      </session-timeout>
   </session-config>
   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
   </welcome-file-list>
</web-app>
```

Figure 10. Web Descriptor File

The <servlet-mapping> section instructs the server that any URL containing the pattern "/controller" should

use the ViewerControllerServlet. The <servlet> section indentifies the class containing the ViewerControllerServlet as "edu.csusb.cs.scproject.servlet.ControllerServlet". The summary of war file has the following layout shown in Figure 11.

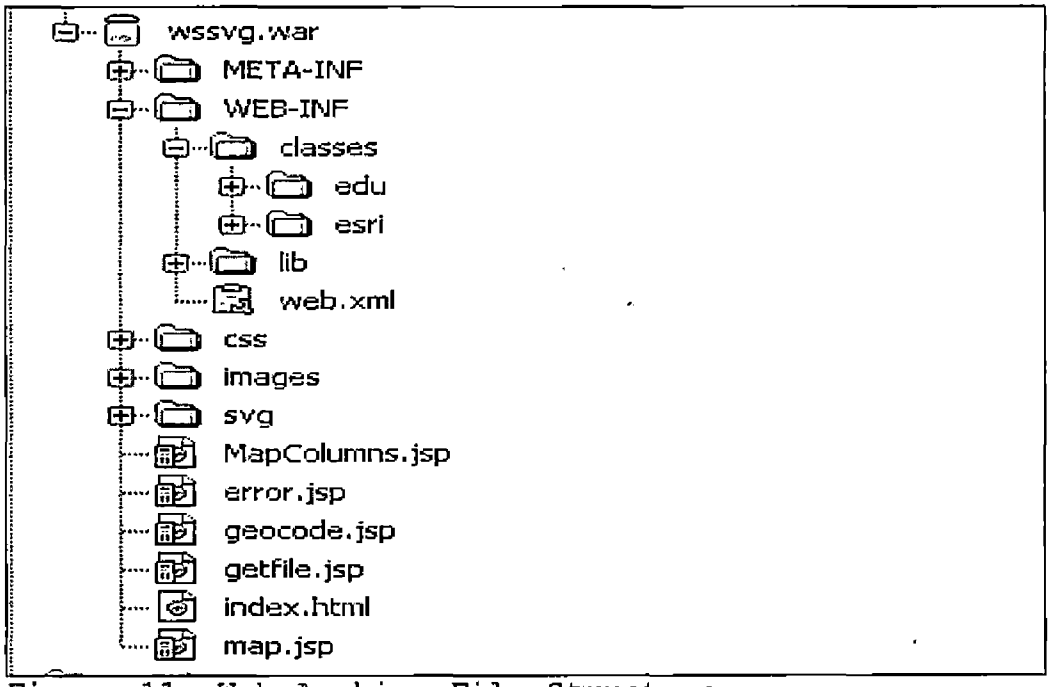

Figure 11. Web Archive File Structure

All compiled server classes are located under WEB-INF/classes directory. The web descriptor file (web.xml) is located under WEB-INF directory. All JSP and html files are located under root folder.

3.2 Operating Instructions

3.2.1 Data Preparation

The user needs to prepare a (comma-separated values) CSV file with the address information. A CSV is one implementation of a delimited text file, which uses a comma to separate values. The first line in the file should have header information. The header acts a column names in a table, this type of format is called a "flat file" because only one table can be stored in a CSV file. The filename can have either ".csv" or ".txt" extension. A sample input file shown in below, first line highlighted in red is the header. The name of columns can be anything as long the user understands what they are, during the map the columns step user will have to map them appropriately. The order of columns can vary, however information in the subsequent rows should correspond to the header columns. The list of columns specified in the header will be presented to the user in a drop down during the map the columns step. If the information for all columns are not available leave a blank, for example for the last row in the following Figure 12 street number is not available.

```
Street Number,University address, City Name, state ,zipcode
5500, University Parkway, San Bernardino, CA, 92407
400, West First Street, Chico, CA, 95929
,one University Circle, Turlock, California, 95382
```

Figure 12. Sample Input Comma Separated Values File

3.2.2 User Guide

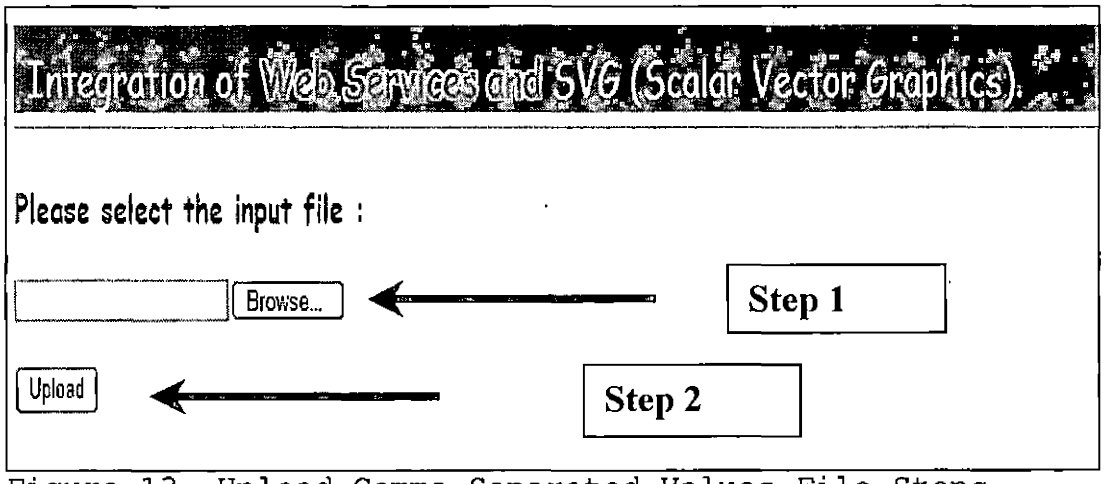

Figure 13. Upload Comma Separated Values File Steps

3.2.2.1 Work Flow. Steps 1 (Figure 13): Click on "Browse" button and navigate to the input CSV file located on local file system which is prepared by the user.

Step 2 (Figure 13): Click on "Upload" button.

Step 3 - 7(Figure 14): Map the columns from the drop down menus. The text above each dropdown indicates the requirement.

Step 8 (Figure 14): After mapping columns click on "Geo-code" button.

Step 9 (Figure 15): Select the color of the dots from the dropdown.

Step 10 (Figure 15): Click on "Update Color" button, the current selected color is displayed in the caption below.

Step 11 (Figure 15): Select the size of the dots from the dropdown.

Step 12 (Figure 15): Click on "Update Size" button, the current selected size is displayed in the caption below.

Step 13 (Figure 15): Click on "Get Map" button.

Step 14 (Figure 16): The map tool bar has several button to interact with the map, Table 7 describes each of those buttons.

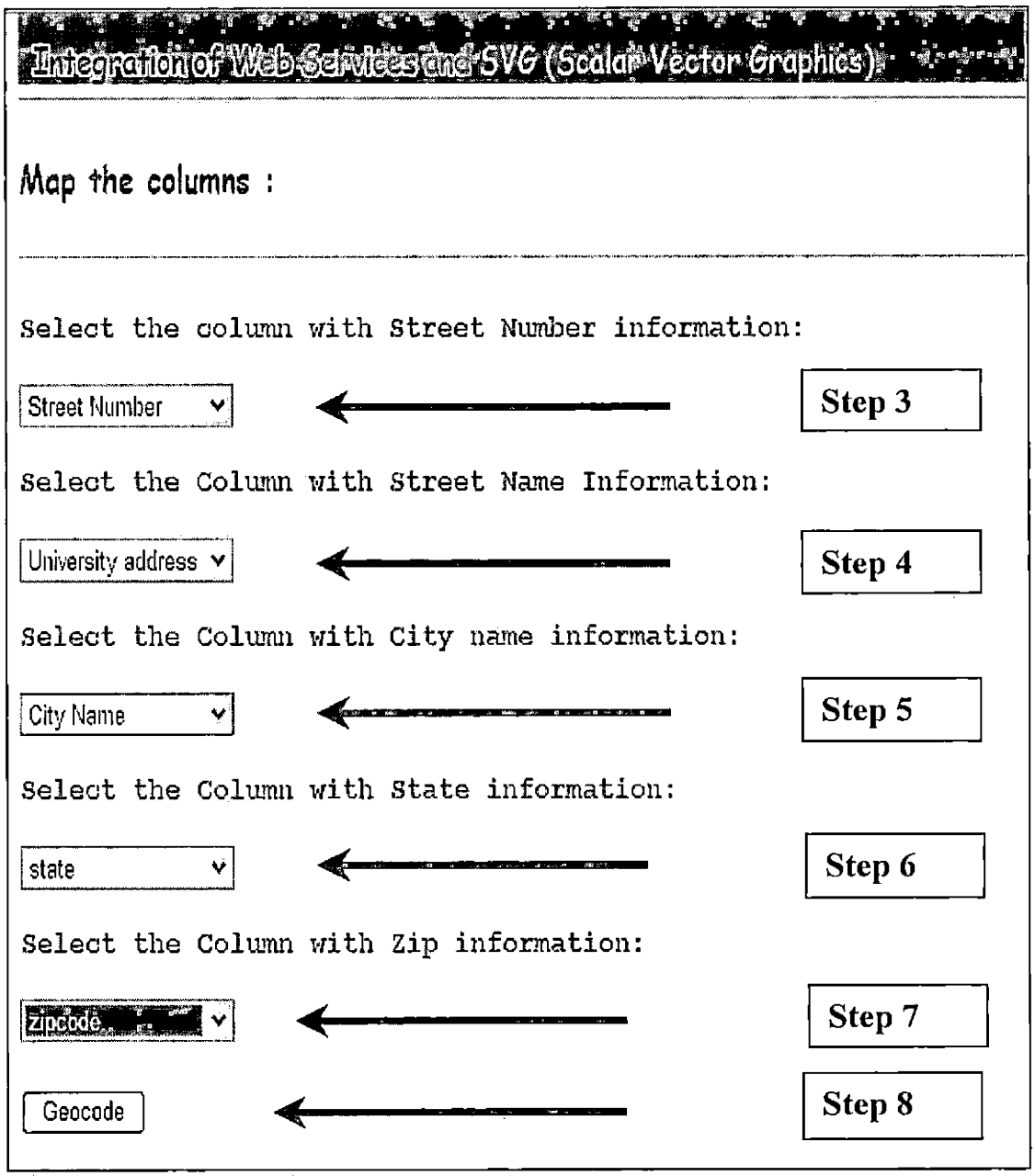

Figure 14. Map the Columns Steps

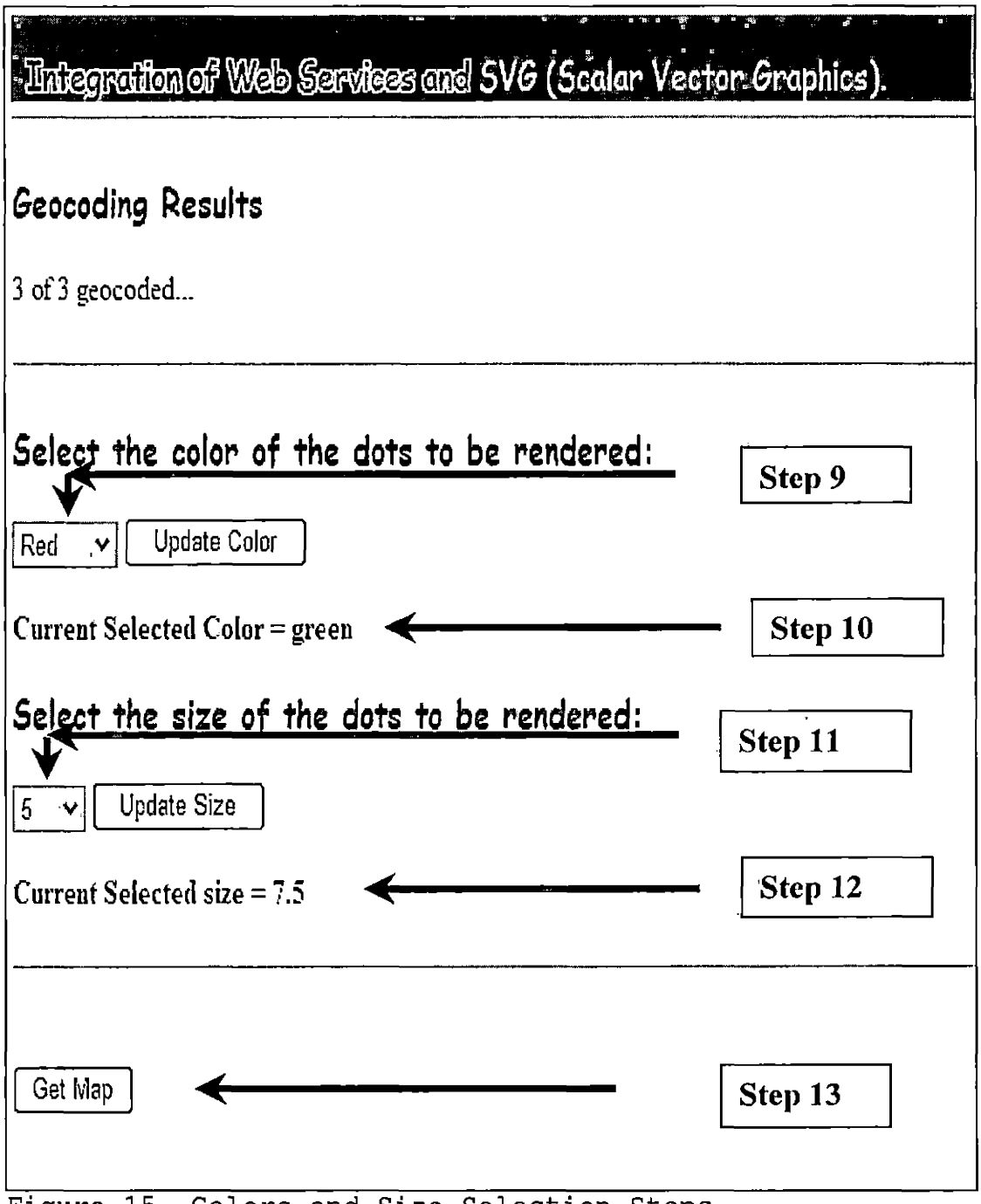

Figure 15. Colors and Size Selection Steps

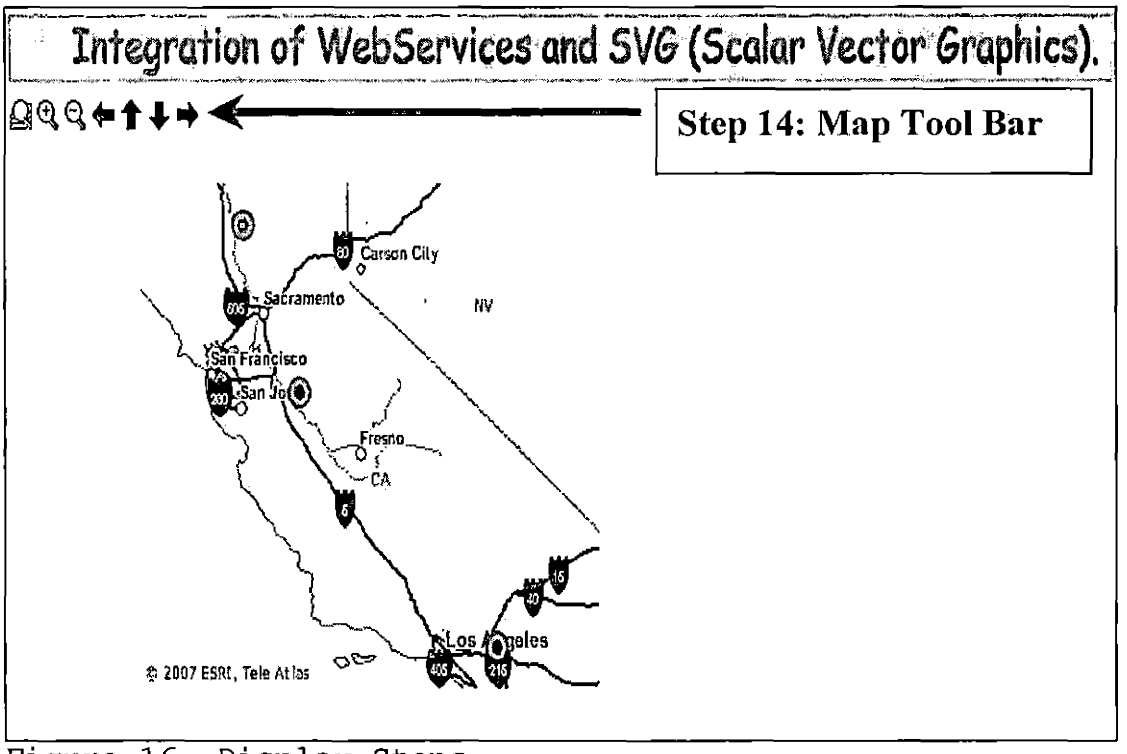

Figure 16. Display Steps

Table 7. Map Toolbar Description

Button	Name	Function
	Full Extent	Allows you to zoom to the full extent of your map
	Zoom In	Allows you to zoom in to a geographic window by clicking a point or dragging a box
	Zoom Out	Allows you to zoom out from a geographic window by clicking a point or dragging a box
	Pan Left	Pan to the Left
	Pan Right	Pan to the Right
	Pan Up	Pan Up
	Pan Down	Pan Down

3.3 Testing

The user can verify the results visually, whether the dots are rendered with the specified color and sizes. Also, user can verify the geographical location of dots on the maps.

CHAPTER FOUR

FUTURE DEVELOPMENTS AND CONCLUSIONS

4.1 Ideas for Future Developments

Adobe FLEX [12] is an emerging technology similar to SVG in terms of mapping functionality. FLEX can be used in place of SVG to perform the interactive mapping. FLEX has advanced vector graphics capable of handling the most demanding, data-intensive applications while performing at desktop application speeds.

4.2 Conclusions

The web services can deliver complex functionality over the internet in a secure manner. These web services can seamlessly be integrated with other software components over the internet. This project demonstrated this capability by integrating the mapping web services with SVG software for interactive mapping.

APPENDIX

WEB APPLICATION SOURCE CODE

ViewerControllerServlet.java

```
/*
 * ViewerControllerServlet.java
 * Purpose:   This class is the gateway to the web application
 *            This class implements Servlet using CMD pattern.
 *            Based on the CMD parameter appropriate action is invoked.
 */
package edu.csusb.cs.scproject.servlet;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.csusb.cs.scproject.common.IAction;
import edu.csusb.cs.scproject.action.MapAction;
import edu.csusb.cs.scproject.action.GetFile;
import edu.csusb.cs.scproject.action.Geocode;
import edu.csusb.cs.scproject.action.MapColumns;
import edu.csusb.cs.scproject.common.scprojectException;
import java.io.*;

public class ControllerServlet extends HttpServlet {

    /**
     * Handles all requests.
     */
    private String _documentRoot = null;
    private String _UserName = null;
    private String _passwd = null;

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String jspPage = null;

        try {
            // process the request
            jspPage = processRequest(request, response);
        } catch (Exception ex) {
            // put exception in the standard JSP location
            request.setAttribute("javax.servlet.jsp.jspException", ex);
            jspPage = "error.jsp";
        }

        String startup = null;
        if (request.getParameter("startup") != null) {
            startup = request.getParameter("startup");
        }

        String cmd = request.getParameter("cmd");
```

```
        if (cmd.equalsIgnoreCase("getMap") && (startup == null)) {
            response.sendRedirect(jspPage);
        } else if (jspPage != null) {

            System.out.println("Forwarding JSP Page.. " + jspPage);
            // turn browser and proxy caching off
            response.setHeader("Expires", "Thu, 01 Jan 1970 00:00:00 GMT");
            response.setHeader("Cache-Control", "no-cache"); // HTTP 1.1

            response.setHeader("Pragma", "no-cache"); // HTTP 1.0

            // forward the request to the JSP page
            RequestDispatcher dispatcher = request.getRequestDispatcher(jspPage);
            dispatcher.forward(request, response);
        }


    } // doGet


    /**
     * Calls doGet.
     */
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    @Override
    public void init(
            javax.servlet.ServletConfig config) throws javax.servlet.ServletException {
        super.init(config);

        _documentRoot = getServletConfig().getServletContext().getRealPath("/");
        _UserName = config.getInitParameter("username");
        _passwd = config.getInitParameter("passwd");

    }

    /**
     * Processes incoming request and forwards it to its Action class.
     */
    private String processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws scprojectException {


        IAction action = null;

        // get command
        String cmd = request.getParameter("cmd");
        if (cmd == null) {
```

```
            cmd = "";
        }

        System.out.println("CMD : " + cmd);
        // create the action

        if (cmd.equalsIgnoreCase("Geocode")) {
            action = new Geocode();
        } else if (cmd.equalsIgnoreCase("getPlace")) {
            //action = new GetPlaceAction();
        } else if (cmd.equalsIgnoreCase("getFile")) {
            //action = new GetPlaceAction();
            action = new GetFile();
        } else if (cmd.equalsIgnoreCase("MapColumns")) {
            action = new MapColumns();
        } else if (cmd.equalsIgnoreCase("getMap")) {
            // default action
            action = new MapAction();

        }

        request.getSession().setAttribute("documentRoot", _documentRoot);
        request.getSession().setAttribute("username", _UserName);
        request.getSession().setAttribute("passwd", _passwd);


        // execute the action
        String jspPage = null;
        jspPage = action.execute(request, response);

        return jspPage;
    } // processRequest

}
```

IAction.java

```
/*
 * IAction.java
 * Purpose: This class provides interface definition for IAction.
 *
 *
 */

package edu.csusb.cs.scproject.common;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Action class interface.
 */
```

```
public interface IAction {
	public String execute(HttpServletRequest request, HttpServletResponse response)
		throws scprojectException;
}
```

scprojectException.java

```
/*
 * scprojectException.java
 * Purpose: This class handles all of the exceptions in the edu.csusb.cs.scproject package.
 *
 */

package edu.csusb.cs.scproject.common;

public class scprojectException extends Exception {

	public scprojectException(String msg) {
		super(msg);
	}

	public scprojectException(Throwable ex) {
		super(ex.getMessage());
	}
}
```

Geocode.java

```
/*
 * Geocode.java
 *
 * Purpose:  Geocodes the address locations.
 *           This class reads input CSV files and parses each line.
 *           The parsed tokens are sent to ESRI web services.
 *           Returned results are persisted in
 *           Session level variables.
 */
package edu.csusb.cs.scproject.action;

import edu.csusb.cs.scproject.common.scprojectException;
import edu.csusb.cs.scproject.common.IAction;
import esri.arcwebservices.v2.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javax.servlet.http.HttpSession;

public class Geocode implements IAction {

	// Name of the JSP page to be forwarded.
	private String jsp_page = "geocode.jsp";
```

```
    private String _username = null;
    private String _password = null;

    @Override
    public String execute(HttpServletRequest request, HttpServletResponse response)
            throws scprojectException {

        if (request.getParameter("startup") != null) {
            String startup = request.getParameter("startup");
            String color = request.getParameter("color");
            String size = request.getParameter("size");
            if (startup.equalsIgnoreCase("false")) {
                if (startup.equalsIgnoreCase("false") && (color != null)) {
                    System.out.println("Color = " + color);
                    request.getSession().setAttribute("color", color);
                }
                if (startup.equalsIgnoreCase("false") && (size != null)) {
                    System.out.println("Size = " + size);
                    request.getSession().setAttribute("size", size);
                }
                return jsp_page;
            }
        }

        HttpSession session = request.getSession(false);
        org.apache.commons.fileupload.FileItem file =
(org.apache.commons.fileupload.FileItem) session.getAttribute("file");
        String csvfilenamePath = file.getName();
        int index = csvfilenamePath.lastIndexOf('\\');
        if (index == -1) {
            index = csvfilenamePath.lastIndexOf('/');
        }
        int len = csvfilenamePath.lastIndexOf('.');
        if (len == -1) {
            len = csvfilenamePath.length();
        }
        String csvfilename = csvfilenamePath.substring(index + 1, len);

        _username = (java.lang.String) request.getSession().getAttribute("username");
        _password = (java.lang.String) request.getSession().getAttribute("passwd");

        try {
            Map map = new HashMap();
            java.util.Enumeration enumparams = (java.util.Enumeration)
request.getParameterNames();
            System.out.println("PARAMETER NAMES and VALUES .....");
            for (; enumparams.hasMoreElements();) {
                // Get the name of the request header
                String value = null;
                String name = (String) enumparams.nextElement();
                value = (String) request.getParameter(name);
                //System.out.println(name + ":" + value);
                if (name.equalsIgnoreCase("Street_Num")) {
```

```
            if (!value.equalsIgnoreCase("n/a")) {
                map.put(name, value);
            } else {
                map.put(name, -1);
            }
        }
        if (name.equalsIgnoreCase("address")) {
            if (!value.equalsIgnoreCase("n/a")) {
                map.put(name, value);
            } else {
                map.put(name, -1);
            }
        }
        if (name.equalsIgnoreCase("city")) {
            if (!value.equalsIgnoreCase("n/a")) {
                map.put(name, value);
            } else {
                map.put(name, -1);
            }
        }
        if (name.equalsIgnoreCase("state")) {
            if (!value.equalsIgnoreCase("n/a")) {
                map.put(name, value);
            } else {
                map.put(name, -1);
            }
        }
        if (name.equalsIgnoreCase("zip")) {
            if (!value.equalsIgnoreCase("n/a")) {
                map.put(name, value);
            } else {
                map.put(name, -1);
            }
        }
}


java.util.ArrayList results;
results = readfile(file.getInputStream(), map);
session.setAttribute("results", results);

InputStream InputS = file.getInputStream();
BufferedReader in = new BufferedReader(new InputStreamReader(InputS));

String str = in.readLine();
int total = 0;
while ((str = in.readLine()) != null) {
    total += 1;
}

esri.arcwebservices.v2.Envelope env = getFullExtent(results);
session.setAttribute("fullextent", env);
```

```
            // System.out.println(env.getMinx() + "," + env.getMiny() + "---" + env.getMaxx()
+ "," + env.getMaxy() + "," + csvfilename);

            request.getSession().setAttribute("resultlist", results);
            request.getSession().setAttribute("total", total);
            request.getSession().setAttribute("csvfilename", csvfilename);

        } catch (java.io.IOException ex) {
            ex.printStackTrace();
        }

        return jsp_page;
    }

    protected Point getPoint(String housenumber, String street, String city, String state_prov,
String zone) {


        Point resultPoint = null;
        try {

            int expiration = 60; // in seconds

             //System.out.println("HOUSE_NUM = " + housenumber + ", STREET = " +
street + ", CITY = " + city + " , STATE = " + state_prov + ", ZIP = " + zone);


            String country_code = "US";
            String dataSource = "GDT.Address.US";

            // ----------Locate Authentication Web services ----------
            AuthenticationLocator locator = new AuthenticationLocator();
            IAuthentication_PortType authentication = locator.getIAuthentication();

            // ----------Call to Authentication Web Services and get the token--------
            String token = authentication.getToken(_username, _password, expiration);

            // System.out.println("Token is: " + token);

            // ----------Locate to AddressFinder Web services ----------
            AddressFinderLocator afLocator = new AddressFinderLocator();
            IAddressFinder_PortType addressFinder = afLocator.getIAddressFinder();

            // Populate the address object with user input.
            Address address = new Address();
            address.setHouseNumber(housenumber);
            address.setStreet(street);
            address.setCity(city);
            address.setState_prov(state_prov);
            address.setZone(zone);
            address.setCountry(country_code);

            // set AddressFinderOptions
```

```
            AddressFinderOptions addressFinderOptions = new AddressFinderOptions();
            addressFinderOptions.setDataSource(dataSource);

            // ----------Call to AddressFinder Web Service ----------
            LocationInfo locInfo = addressFinder.findAddress(address,
addressFinderOptions, token);

            //Get the location array of candidates
            Location[] loc = locInfo.getCandidates();


            //Location includes x,y coordiates and the complete address description
            // Print first candidate

            System.out.println(loc[0].getDescription1() + " (" + loc[0].getPoint().getX() + "," +
loc[0].getPoint().getY() + ")");

            resultPoint = loc[0].getPoint();

        } catch (Exception e) {
            System.out.println("Error : " + e.getMessage());
        } finally {
            return resultPoint;
        }


    }

    protected java.util.ArrayList readfile(InputStream InputS, Map map) throws IOException {

        String str = null;
        String[] fields = null;
        ArrayList resultslist = new ArrayList();
        BufferedReader in = new BufferedReader(new InputStreamReader(InputS));
        str = in.readLine(); // header

        fields = str.split(",");

        int Street_NumPos = -1;
        int addressPos = -1;
        int cityPos = -1;
        int statePos = -1;
        int zipPos = -1;
        java.util.ArrayList Results = new java.util.ArrayList();

        for (int i = 0; i < fields.length; i++) {
            String name = fields[i].trim();
            //System.out.println("Name = " + name );
            if (name.equalsIgnoreCase(map.get("Street_Num").toString())) {
                Street_NumPos = i;
            }
            if (name.equalsIgnoreCase(map.get("address").toString())) {
                addressPos = i;
```

```
                }
                if (name.equalsIgnoreCase(map.get("city").toString())) {
                    cityPos = i;
                }
                if (name.equalsIgnoreCase(map.get("state").toString())) {
                    statePos = i;
                }
                if (name.equalsIgnoreCase(map.get("zip").toString())) {
                    zipPos = i;
                }

            }

            // System.out.println(fields.length + "::" + Street_NumPos + "," + addressPos + "," +
cityPos + "," + statePos + "," + zipPos);

            while ((str = in.readLine()) != null) {

                if (str.trim().length() > 0) {
                    fields = str.split(",");
                    String st_num = null, addr = null, city = null, state = null, zip = null;
                    System.out.println(str);
                    try {
                        if ((Street_NumPos != -1) && (fields[Street_NumPos] != null)) {
                            st_num = fields[Street_NumPos];
                        }
                        if ((addressPos != -1) && (fields[addressPos] != null)) {
                            addr = fields[addressPos];
                        }
                        if ((cityPos != -1) && (fields[cityPos] != null)) {
                            city = fields[cityPos];
                        }
                        if ((statePos != -1) && (fields[statePos] != null)) {
                            state = fields[statePos];
                        }
                        if ((zipPos != -1) && (fields[zipPos] != null)) {
                            zip = fields[zipPos];
                        }
                    } catch (Exception e) {
                        System.out.println("Error : " + e.getMessage());
                    } finally {
                        System.out.println("");
                    }

                    // Call Get Point
                    Point p;
                    p = getPoint(st_num, addr, city, state, zip);
                    if (p != null) {
                        resultslist.add(p);
                    }
                }
            }
            in.close();
```

```
            return resultslist;

        }

        private esri.arcwebservices.v2.Envelope getFullExtent(final java.util.List list) {
            double minx = Double.POSITIVE_INFINITY;
            double miny = Double.POSITIVE_INFINITY;
            double maxx = Double.NEGATIVE_INFINITY;
            double maxy = Double.NEGATIVE_INFINITY;

            final int length = list.size();
            if (length > 0) {
                for (int i = 0; i < length; i++) {
                    final esri.arcwebservices.v2.Point mapStop =
(esri.arcwebservices.v2.Point) list.get(i);

                    final double x = mapStop.getX();
                    final double y = mapStop.getY();
                    minx = Math.min(x, minx);
                    miny = Math.min(y, miny);
                    maxx = Math.max(x, maxx);
                    maxy = Math.max(y, maxy);
                }

                final double dx = (maxx - minx) / 10.0;
                minx -= dx;
                maxx += dx;
                final double dy = (maxy - miny) / 10.0;
                miny -= dy;
                maxy += dy;
            } else {
                minx = miny = maxx = maxy = 0.0;
            }

            esri.arcwebservices.v2.Envelope env = new esri.arcwebservices.v2.Envelope();
            env.setMinx(minx);
            env.setMiny(miny);
            env.setMaxx(maxx);
            env.setMaxy(maxy);
            return env;
        }
}
```

GetFile.java

```
/*
 * GetFile.java
 * Purpose: Helps to forward getfile.jsp to the browser.
 *
 *
 */
```

```
package edu.csusb.cs.scproject.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.csusb.cs.scproject.common.scprojectException;
import edu.csusb.cs.scproject.common.IAction;

/*
 *
*/
public class GetFile implements IAction {

	private String jsp_page = "getfile.jsp";

	@Override
	public String execute(HttpServletRequest request, HttpServletResponse response)
			throws scprojectException {
		return jsp_page;
	}
}
```

MapAction.java

```
/*
 * MapAction.java
 * Purpose:  This class handles Map Related functions and communicates with
 *           map.jsp via servlet.
 *           Given the bounding coordinates (extent), converts the image coordinates into
 *           Map coordinates. Those converted real world coordinates are used to fetch the
 *           map from the Mapping web service. Image URL is populated in a session level
 *           variable. Finally Map.jsp is forwarded to the browser.
 *
 */
package edu.csusb.cs.scproject.action;

// import com.sun.org.apache.bcel.internal.verifier.statics.DOUBLE_Upper;
import java.rmi.RemoteException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.csusb.cs.scproject.common.scprojectException;
import edu.csusb.cs.scproject.common.IAction;

import esri.arcwebservices.v2.*;
import javax.servlet.http.HttpSession;
import javax.xml.rpc.ServiceException;
import java.util.Random;
import java.io.*;

/**
 *
 *
 */
```

```
public class MapAction implements IAction {

    // Name of the JSP page to be forwarded.
    private String jsp_page = "map.jsp";
    final int _mapWidth = 330;
    final int _mapHeight = 270;
    final double _x_offset = 100;
    final double _y_offset = 100;
    double _xmin;
    double _ymin;
    double _xmax;
    double _ymax;

    public double toImageX(final double x, final double xmin, final double xmax) {

        double unitsPerPixelX = (xmax - xmin) / _mapWidth;
        return (x - xmin) / unitsPerPixelX;

    }

    public double toImageY(final double y, final double ymin, final double ymax) {

        double unitsPerPixelY = (ymax - ymin) / _mapHeight;
        return _mapHeight - (y - ymin) / unitsPerPixelY;

    }

    public double toWorldX(final double x) {

        double unitsPerPixelX = (_xmax - _xmin) / _mapWidth;
        return unitsPerPixelX * x + _xmin;
    }

    public double toWorldY(final double y) {

        double unitsPerPixelY = (_ymax - _ymin) / _mapHeight;
        return unitsPerPixelY * (_mapHeight - y) + _ymin;

    }

    public void SetDefaultExtent(esri.arcwebservices.v2.Envelope env) {

        _xmin = env.getMinx();
        _ymin = env.getMiny();
        _xmax = env.getMaxx();
        _ymax = env.getMaxy();
    }

    public String GetCSVFileName(HttpServletRequest request) {

        HttpSession session = request.getSession(false);
        org.apache.commons.fileupload.FileItem file =
(org.apache.commons.fileupload.FileItem) session.getAttribute("file");
```

```
        String csvfilenamePath = file.getName();
        int index = csvfilenamePath.lastIndexOf('\\');
        if (index == -1) {
            index = csvfilenamePath.lastIndexOf('/');
        }
        int len = csvfilenamePath.lastIndexOf('.');
        if (len == -1) {
            len = csvfilenamePath.length();
        }
        String csvfilename = csvfilenamePath.substring(index + 1, len);

        System.out.println("csvfilename = " + csvfilename);
        return csvfilename;

    }

    @Override
    public String execute(HttpServletRequest request, HttpServletResponse response)
            throws scprojectException {

        if (request.getParameter("startup") != null) {
            String startup = request.getParameter("startup");
            if (startup.equalsIgnoreCase("TRUE") && (jsp_page != null)) {
                // System.out.println("FORWARDING JSP PAGE " + jsp_page + " because
STRARTUP prameters IS TRUE..." );
                return jsp_page;
            }
        }

        HttpSession session = request.getSession(false);
        esri.arcwebservices.v2.Envelope env = (esri.arcwebservices.v2.Envelope)
session.getAttribute("fullextent");
        esri.arcwebservices.v2.MapImageInfo pre_ext =
(esri.arcwebservices.v2.MapImageInfo) session.getAttribute("MapImageInfo");
        java.util.ArrayList results = (java.util.ArrayList) session.getAttribute("results");

        if (pre_ext != null) {

            _xmin = pre_ext.getMapExtent().getMinx();
            _ymin = pre_ext.getMapExtent().getMiny();
            _xmax = pre_ext.getMapExtent().getMaxx();
            _ymax = pre_ext.getMapExtent().getMaxy();

        } else {

            // Intiate Default EXT
            SetDefaultExtent(env);

        }

        Random generator = new Random();
        int counter = generator.nextInt();
```

```
String contextPath = request.getContextPath();

//jsp_page = "http://localhost:8080/wssvg/svgimage" + counter + ".xml";
jsp_page = contextPath + "/svg/svgimage" + counter + ".xml";

String username = (java.lang.String) request.getSession().getAttribute("username");
String password = (java.lang.String) request.getSession().getAttribute("passwd");
int expiration = 60; // in seconds

// ----------Locate Authentication Web services ----------
AuthenticationLocator locator = new AuthenticationLocator();
try {
    IAuthentication_PortType authentication = locator.getIAuthentication();
    try {

        // ----------Call to Authentication Web Services and get the token--------
        String token = authentication.getToken(username, password, expiration);
        MapImageLocator mpLocator = new MapImageLocator();
        IMapImage_PortType MapImageFinder = mpLocator.getIMapImage();

        Envelope extent = new Envelope();

        double minx = 0;
        double miny = 0;
        double maxx = 0;
        double maxy = 0;

        if (request.getParameter("minx") != null) {
            minx = Double.parseDouble(request.getParameter("minx"));
        }
        if (request.getParameter("miny") != null) {
            miny = Double.parseDouble(request.getParameter("miny"));
        }

        if (request.getParameter("maxx") != null) {
            maxx = Double.parseDouble(request.getParameter("maxx"));
        }

        if (request.getParameter("maxy") != null) {
            maxy = Double.parseDouble(request.getParameter("maxy"));
        }

        if (minx != 0 && miny != 0 && maxx != 0 && maxy != 0) {

            double xmin_new = toWorldX(minx - _x_offset);
            double ymin_new = toWorldY(maxy - _y_offset);
            double xmax_new = toWorldX(maxx - _x_offset);
            double ymax_new = toWorldY(miny - _y_offset);

            extent.setMinx(xmin_new);
            extent.setMiny(ymin_new);
            extent.setMaxx(xmax_new);
            extent.setMaxy(ymax_new);
```

```
} else if (minx == 0 && miny != 0 && maxx != 0 && maxy != 0) {

    double xmin_new = _xmin;
    double ymin_new = toWorldY(maxy - _y_offset);
    double xmax_new = toWorldX(maxx - _x_offset);
    double ymax_new = toWorldY(miny - _y_offset);

    extent.setMinx(xmin_new);
    extent.setMiny(ymin_new);
    extent.setMaxx(xmax_new);
    extent.setMaxy(ymax_new);

} else if (minx != 0 && miny == 0 && maxx != 0 && maxy != 0) {

    double xmin_new = toWorldX(minx - _x_offset);
    double ymin_new = toWorldY(maxy - _y_offset);
    double xmax_new = toWorldX(maxx - _x_offset);
    double ymax_new = _ymax;

    extent.setMinx(xmin_new);
    extent.setMiny(ymin_new);
    extent.setMaxx(xmax_new);
    extent.setMaxy(ymax_new);

} else if (minx != 0 && miny != 0 && maxx == 0 && maxy != 0) {

    double xmin_new = toWorldX(minx - _x_offset);
    double ymin_new = toWorldY(maxy - _y_offset);
    double xmax_new = _xmax;
    double ymax_new = toWorldY(miny - _y_offset);

    extent.setMinx(xmin_new);
    extent.setMiny(ymin_new);
    extent.setMaxx(xmax_new);
    extent.setMaxy(ymax_new);

} else if (minx != 0 && miny != 0 && maxx != 0 && maxy == 0) {

    double xmin_new = toWorldX(minx - _x_offset);
    double ymin_new = _ymin;
    double xmax_new = toWorldX(maxx - _x_offset);
    double ymax_new = toWorldY(miny - _y_offset);

    extent.setMinx(xmin_new);
    extent.setMiny(ymin_new);
    extent.setMaxx(xmax_new);
    extent.setMaxy(ymax_new);

} else {

    SetDefaultExtent(env);
    extent.setMinx(_xmin);
```

```
                    extent.setMiny(_ymin);
                    extent.setMaxx(_xmax);
                    extent.setMaxy(_ymax);

                }
                System.out.println("MINX: " + extent.getMinx() + " MINY " +
extent.getMiny() + " MAXX :" + extent.getMaxx() + " MAXY " + extent.getMaxy());


                MapImageSize mapImageSize = new MapImageSize();
                mapImageSize.setWidth(_mapWidth);
                mapImageSize.setHeight(_mapHeight);


                // set map image options
                MapImageOptions mapImageOptions = new MapImageOptions();
                mapImageOptions.setBackgroundColor("237,237,238");
                String mapDataSource = "TA.Streets.US";
                mapImageOptions.setDataSource(mapDataSource);
                mapImageOptions.setMapImageSize(mapImageSize);


                MapImageInfo mpInfo = MapImageFinder.getMap(extent,
mapImageOptions, token);

                // System.out.println(" After MINX: " + mpInfo.getMapExtent().getMinx() + "
After MINY " + mpInfo.getMapExtent().getMiny() + " After MAXX :" +
mpInfo.getMapExtent().getMaxx() + "After MAXY " + mpInfo.getMapExtent().getMaxy());

                request.getSession().setAttribute("MapImageInfo", mpInfo);


                try {
                    String documentRoot = (java.lang.String)
request.getSession().getAttribute("documentRoot");

                    String color = (java.lang.String) session.getAttribute("color");
                    String size = (java.lang.String) session.getAttribute("size");

                    if (color == null) {
                        color = "red";
                    }
                    if (size == null) {
                        size = "5";
                    }
                    FileWriter fileWriter = new FileWriter(documentRoot +
"\\svg\\svgimage" + counter + ".xml");
                    BufferedWriter out = new BufferedWriter(fileWriter);
                    out.write("<g>");
                    out.write("<image xlink:href=\"" + mpInfo.getMapUrl() + "\" x=\"100\"
y=\"100\" width=\"330\" height=\"270\" />");

                    final int length = results.size();
```

```
                        if (length > 0) {
                            for (int i = 0; i < length; i++) {
                                final esri.arcwebservices.v2.Point mapStop =
(esri.arcwebservices.v2.Point) results.get(i);
                                final double wx = mapStop.getX();
                                final double wy = mapStop.getY();
                                System.out.println(" World X = " + wx + ", World Y = " + wy);
                                final double ix = _x_offset + toImageX(wx,
mpInfo.getMapExtent().getMinx(), mpInfo.getMapExtent().getMaxx());
                                final double iy = _y_offset + toImageY(wy,
mpInfo.getMapExtent().getMiny(), mpInfo.getMapExtent().getMaxy());
                                //System.out.println(" Image X = " + toImageX(wx,
mpInfo.getMapExtent().getMinx(), mpInfo.getMapExtent().getMaxx()) +
                                //          ", Image Y = " + toImageY(wy,
mpInfo.getMapExtent().getMiny(), mpInfo.getMapExtent().getMaxy()));
                                //System.out.println(" X off set + Image X = " + ix + ", Y
offset + Image Y = " + iy);
                                out.write("<circle cx=\" " + ix + "\" cy=\" " + iy + "\" r=\" " + size
+ " \" stroke=\"black\" stroke-width=\"2\" fill=\" " + color + "\"/>");
                            }
                        }
                        out.write("</g>");
                        out.close();

                    //System.out.println("Wrote XML FILE");

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

                    //Get the location array of candidates
                    // System.out.println("Map URL : " + mpInfo.getMapUrl());
                    System.out.println(jsp_page);


                } catch (RemoteException ex) {
                    ex.printStackTrace();
                }
            } catch (ServiceException ex) {
                ex.printStackTrace();
            }

            return jsp_page;

        }
}
```

MapColumns.java

```
/*
 * MapColumns.java
 * Purpose:  This class generates map the columns jsp.
 *           The input file header row is read and tokenized with comma delimiter.
 *           The list of columns are presented in the dropdown controls in a JSP.
 *
 */
package edu.csusb.cs.scproject.action;

import edu.csusb.cs.scproject.common.scprojectException;
import edu.csusb.cs.scproject.common.IAction;

import java.util.*;
import java.io.*;
import javax.servlet.http.*;

import org.apache.commons.fileupload.disk.*;
import org.apache.commons.fileupload.servlet.*;
import org.apache.commons.fileupload.*;

/**
 *
 */
public class MapColumns implements IAction {

    private String jsp_page = "MapColumns.jsp";

    @Override
    public String execute(HttpServletRequest request, HttpServletResponse response)
            throws scprojectException {

        DiskFileItemFactory diskFileFactory = new DiskFileItemFactory();
        diskFileFactory.setSizeThreshold(40960); // the unit is bytes.

        // Parse the form data contanined inthe HTTP request
        ServletFileUpload servletFileUpload = new ServletFileUpload(diskFileFactory);
        servletFileUpload.setSizeMax(81920); // the unit is butes


        // Checking if an HTTP request in Encoded in Multipart Format
        if (ServletFileUpload.isMultipartContent(request)) {
            // Parse request
            try {
                List fileItemsList = servletFileUpload.parseRequest(request);
                Iterator it = fileItemsList.iterator();

                while (it.hasNext()) {
                    FileItem fileItem = (FileItem) it.next();
                    if (fileItem.isFormField()) {
                        // The File item contains a simple name-value pair of form fields
                    } else {

                        ArrayList headerlist = null;
```

```
                                try {
                                    headerlist = ParseHeader(fileItem.getInputStream());
                                    headerlist.add("N/A");
                                    HttpSession session = request.getSession(true);
                                    session.setAttribute("file", fileItem);

                                    request.setAttribute("headerlist", headerlist);

                                } finally {
                                    // place holder
                                }


                            }
                        }
                    } catch (FileUploadException ex) {
                        ex.printStackTrace();
                    } catch (java.io.IOException ex) {
                        ex.printStackTrace();
                    }

                }


                return jsp_page;
            }

            protected ArrayList ParseHeader(InputStream InputS) throws IOException {

                String header = null;
                ArrayList list = new ArrayList();
                BufferedReader in = new BufferedReader(new InputStreamReader(InputS));
                header = in.readLine();
                String[] temp = null;
                temp = header.split(",");
                for (int i = 0; i < temp.length; i++) {
                    list.add(temp[i]);
                }
                in.close();
                return list;

            }
        }
```

MapColumns.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/base.css">
        <script language="JavaScript" type="text/javascript">
    function init() {
     parent.mainFrame.src = "MapColumns.jsp";
    }
    function formSubmit()
    {
        document.getElementById("mcForm").submit();
    }
        </script>

    </head>
    <body onLoad="JavaScript:init();" >
        <table width="100%" border="0" cellspacing="0" cellpadding="0" bgcolor="#808080">
            <td><img width="5" height="35" border="0">
                <span class="yellow"> Integration of </span>
                <span class="saddlebrown"> Web Services and </span>
                <span class="white"> SVG (Scalar Vector Graphics). </span>
            </td>
        </table>
        <hr>
        <h3>Map the columns : <h3>
        <hr>
        <script language="JavaScript" type="text/javascript">
            var ix =0;
            var list = new Array();
            <% java.util.ArrayList hlist =
(java.util.ArrayList)request.getAttribute("headerlist");%>
            var count = <%= hlist.size() %>;
            <% for (int i =0; i < hlist.size(); i++) { %>
                list[ix] = "<%= hlist.get(i)%>";
                ix = ix +1;
            <% } %>
            var theUrl = "Controller";
            theUrl += "?cmd=Geocode&startup=true";
            document.write(" <FORM id= " + "mcForm" + " METHOD= 'POST' ACTION= "
+ theUrl + ">");

            document.write("<pre <b>Select the column with Street Number
information:</b></pre>");
            document.write("<select name=Street_Num>");
            for(x in list) {
                document.write("<option> " + list[x] + "</option>");
                }
                document.write ("</select></br>");

            document.write("<pre <b>Select the Column with Street Name
Information:</b></pre>");
            document.write("<select name=address>");
            for(x in list) {
```

```
                    document.write("<option> " + list[x] + "</option>");
                    }
                    document.write ("</select></br>");

                document.write("<pre <b>Select the Column with City name
information:</b></pre>");
                document.write("<select name=city>");
                for(x in list) {
                    //alert (list[x]);
                    document.write("<option> " + list[x] + "</option>");
                    }
                    document.write ("</select></br>");

                document.write("<pre <b>Select the Column with State
information:</b></pre>");
                document.write("<select name=state>");
                for(x in list) {
                    //alert (list[x]);
                    document.write("<option > " + list[x] + "</option>");
                    }
                    document.write ("</select></br>");

                document.write("<pre <b>Select the Column with Zip information:</b></pre>");
                document.write("<select name=zip>");
                for(x in list) {
                    //alert (list[x]);
                    document.write("<option> " + list[x] + "</option>");
                    }
                    document.write ("</select></br></br>");
                    document.write ( "<input type= " + "button" + " value= " + "Geocode" + "
onclick=" + "formSubmit()" + ">");
                    document.write ("</form>");
        </script>
    </body>
</html>
```

Map.jsp

```
<%@page import = "java.util.*" contentType="image/svg+xml"%>
<%@page pageEncoding="UTF-8"%>

<svg width="1200" height="1200" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" zoomAndPan="disable"
onload="onSVGLoad(evt)">

    <script>

    var svgdoc;
    var mapgrp;
    var loading;
    var eventrect;
```

```
var zoomrect;
var downx;
var downy;
var isdown=false;
var zoomdx=120;
var zoomdy=100;
var swidth=330;
var sheight=270;
var ampersand = String.fromCharCode(38);
var csvFileName;

// response.sendRedirect to send the xml???



function onSVGLoad(evt) {

 // alert ("onSVGLoad Test");

      svgdoc= evt.target.ownerDocument;
      loading=svgdoc.getElementById('loading');
      fullextent=svgdoc.getElementById('fullextent');
      zoomin=svgdoc.getElementById('zoomin');
      zoomout=svgdoc.getElementById('zoomout');
      west=svgdoc.getElementById('west');
      north=svgdoc.getElementById('north');
      south=svgdoc.getElementById('south');
      east=svgdoc.getElementById('east');
      mapgrp=svgdoc.getElementById('mapgrp');
      eventrect=svgdoc.getElementById('eventrect');
      zoomrect=svgdoc.getElementById('zoomrect');
      onFullExtent(evt);

      // alert("Done OnSVGLOAD");

}

function setImageHref(evt,idx){

      var target=evt.target;
target.setAttribute('xlink:href','./images/'+target.getAttribute('id')+'_'+idx+'.gif');
}

function onToolOut(evt){
      setImageHref(evt,1);
}

function onToolOver(evt){
      setImageHref(evt,2);
}

function loadingShow(){
```

```
        loading.style.setProperty('display','inline');
        fullextent.style.setProperty('pointer-events','none');
        zoomin.style.setProperty('pointer-events','none');
        zoomout.style.setProperty('pointer-events','none');
        west.style.setProperty('pointer-events','none');
        north.style.setProperty('pointer-events','none');
        south.style.setProperty('pointer-events','none');
        east.style.setProperty('pointer-events','none');

 // alert ("Done LoadingShow");

}

 function loadingHide(){

        loading.style.setProperty('display','none');
        fullextent.style.setProperty('pointer-events','fill');
        zoomin.style.setProperty('pointer-events','fill');
        zoomout.style.setProperty('pointer-events','fill');
        west.style.setProperty('pointer-events','fill');
        north.style.setProperty('pointer-events','fill');
        south.style.setProperty('pointer-events','fill');
        east.style.setProperty('pointer-events','fill');

        // alert ("Done LoadingHide");
}

function removeAllChildren(grp){
        var child = grp.getFirstChild();
        while (child != null){
                if (child.getNodeType() == 1){
                 grp.removeChild(child);
                }
        child = child.getNextSibling();
        }
}

function GetCSVFileName(xmldoc){
        var child = xmldoc.getFirstChild();
        var nodeList = child.childNodes();
// alert( nodeList.length());
return ( nodeList.item(nodeList.length() - 1).getNodeName());
}

function onMap(data){

        if(data.success){
                // alert(data.content);
                var xmldoc=parseXML(data.content,svgdoc);
                removeAllChildren(mapgrp);
                // mapgrp.appendChild(xmldoc.getFirstChild());
                //alert(xmldoc.getFirstChild().getNodeName());
        csvFileName = GetCSVFileName(xmldoc);
```

```
        // alert (csvFileName);
            mapgrp.appendChild(xmldoc);
            loadingHide();
        } else {
            loadingHide();
            alert( "Error:Cannot read image data from the map server !");
        }

}

function getMap(url){
    //getURL(maurl, onMap);
    getURL(url, onMap);
}

function onFullExtent(evt){

    var theurl = "Controller";
    theurl += "?cmd=getMap";
    loadingShow();
    getMap(theurl);

}


function zoomDown(evt) {
if(evt.button==0){
    downx=evt.clientX;
    downy=evt.clientY;
    zoomrect.setAttribute("x",downx);
    zoomrect.setAttribute("y",downy);
    zoomrect.setAttribute("width","0");
    zoomrect.setAttribute("height","0");
    zoomrect.style.setProperty('display','in');
    isdown=true;
}
}

function zoomMove(evt) {
if(isdown) {
    var x=evt.clientX;
    var y=evt.clientY;
    zoomrect.setAttribute("x",Math.min(x,downx));
    zoomrect.setAttribute("y",Math.min(y,downy));
    zoomrect.setAttribute("width",Math.abs(x-downx));
    zoomrect.setAttribute("height",Math.abs(y-downy));
}
}

function doZoomIn(x,y,w,h){

        var url= 'Controller?cmd=getMap';
        url+= ampersand + 'width='+swidth;
```

```
            url+= ampersand + 'height='+sheight;

            url+= ampersand + 'minx='+x;
            url+= ampersand + 'miny='+y;
            url+= ampersand + 'maxx='+(x+w);
            url+= ampersand + 'maxy='+(y+h);
            loadingShow();
            getMap(url);
}

function zoomUp(evt) {
if(evt.button==0){
      isdown=false;
            zoomrect.style.setProperty('display','none');
            eventrect.style.setProperty('pointer-events','none');
            eventrect.style.setProperty('display','none');
            var x=evt.clientX;
            var y=evt.clientY;
            var sx=Math.min(x,downx);
            var sy=Math.min(y,downy);
            var sw=Math.abs(x-downx);
            var sh=Math.abs(y-downy);
            doZoomIn(sx,sy,sw,sh);
                  }
      }


function onZoomIn(evt){

      eventrect.setAttribute("onmousedown","zoomDown(evt)");
eventrect.setAttribute("onmousemove","zoomMove(evt)");
eventrect.setAttribute("onmouseup","zoomUp(evt)");
eventrect.style.setProperty('pointer-events','fill');
eventrect.style.setProperty('display','svg');
      }

function onZoomOut(evt){

      var url= 'Controller?cmd=getMap';
      url+= ampersand + 'width='+swidth;
      url+= ampersand + 'height='+sheight;
      url+= ampersand + 'minx='+(-zoomdx);
      url+= ampersand + 'miny='+(-zoomdy);
      url+= ampersand + 'maxx='+(swidth+zoomdx);
      url+= ampersand + 'maxy='+(sheight+zoomdy);
      loadingShow();
      getMap(url);
}

function onWest(evt){

 var url= 'Controller?cmd=getMap';
      url+=ampersand + 'width='+swidth;
```

```
        url+= ampersand + 'height='+sheight;
        url+= ampersand + 'minx='+(-zoomdx);
        url+= ampersand + 'miny=0';
        url+= ampersand + 'maxx='+(swidth-zoomdx);
        url+= ampersand + 'maxy='+sheight;
        loadingShow();
        getMap(url);
}

function onEast(evt){
        var url= 'Controller?cmd=getMap';
        url+= ampersand + 'width='+swidth;
        url+= ampersand + 'height='+sheight;
        url+= ampersand + 'minx='+zoomdx;
        url+= ampersand + 'miny=0';
        url+= ampersand + 'maxx='+(swidth+zoomdx);
        url+= ampersand + 'maxy='+sheight;
        loadingShow();
        getMap(url);
}

function onNorth(evt){
        var url= 'Controller?cmd=getMap';
        url+= ampersand + 'width='+swidth;
        url+= ampersand + 'height='+sheight;
        url+= ampersand + 'minx=0';
        url+= ampersand + 'miny='+(-zoomdy);
        url+= ampersand + 'maxx='+swidth;
        url+= ampersand + 'maxy='+(sheight-zoomdy);
        loadingShow();
        getMap(url);
}

function onSouth(evt){

        var url= 'Controller?cmd=getMap';
        url+= ampersand + 'width='+swidth;
        url+= ampersand + 'height='+sheight;
        url+= ampersand + 'minx=0';
        url+= ampersand + 'miny='+zoomdy;
        url+= ampersand + 'maxx='+swidth;
        url+= ampersand + 'maxy='+(sheight+zoomdy);
        loadingShow();
        getMap(url);
}

function goBack(evt){
        var url= 'Controller?cmd=geocode';
        url+= ampersand + 'startup=false';
        getMap(url);
}
```

```
function shade(evt)
{
    var rect = evt.target;
    var currentFill = rect.getAttribute("fill");
     if (currentFill == "gray")
        rect.setAttribute("fill", "#E0E0E0");
        else
        rect.setAttribute("fill", "gray");
}

function noop(evt){

}

</script>

<style type="text/css"><![CDATA[
    .eventsfill{pointer-events:fill;}
    .eventsnone{pointer-events:none;}
    .zoomrect{opacity:0.5;fill:yellow;stroke:black;stroke-width:1;display:none;pointer-
events:none;}
    .eventrect{display:none;opacity:0;pointer-events:none;}
    .loadingrect{opacity:0.5;fill:yellow;stroke:black;stroke-width:1;display:inline;pointer-
events:none;}
.headerrect{opacity:0.5;fill:#808080;stroke:black;stroke-width:1;display:inline;pointer-
events:none;}
    .loadingtext{opacity:0.8;fill:black;stroke:black;stroke-width:2;display:inline;pointer-
events:none;text-anchor:middle;font-family:Arial;font-size:30;}
.headertext{opacity:0.8;fill:saddlebrown;stroke:saddlebrown;stroke-
width:1;display:inline;pointer-events:none;text-anchor:left;font-family:"comic sans ms", arial,
'sans serif';font-size:25;}
]]></style>

<g id="heading" style="display:inline">
    <rect x="5" y="10" width="100%" height="35" class="headerrect"/>
    <text x="50" y="35" width="100%" class="headertext">Integration of WebServices
and SVG (Scalar Vector Graphics).</text>
</g>
<image id="fullextent" xlink:href="./images/fullextent_1.gif" x="5" y="55" width="16"
height="16" onclick="onFullExtent(evt)" onmouseover="onToolOver(evt)"
onmouseout="onToolOut(evt)" />
<image id="zoomin" xlink:href="./images/zoomin_1.gif" x="25" y="55" width="16"
height="16" onclick="onZoomIn(evt)" onmouseover="onToolOver(evt)"
onmouseout="onToolOut(evt)"/>
<image id="zoomout" xlink:href="./images/zoomout_1.gif" x="45" y="55" width="16"
height="16" onclick="onZoomOut(evt)" onmouseover="onToolOver(evt)"
onmouseout="onToolOut(evt)"/>
<image id="west" xlink:href="./images/west_1.gif" x="65" y="55" width="16" height="16"
onclick="onWest(evt)" onmouseover="onToolOver(evt)" onmouseout="onToolOut(evt)"/>
<image id="north" xlink:href="./images/north_1.gif" x="85" y="55" width="16" height="16"
onclick="onNorth(evt)" onmouseover="onToolOver(evt)" onmouseout="onToolOut(evt)"/>
<image id="south" xlink:href="./images/south_1.gif" x="105" y="55" width="16" height="16"
onclick="onSouth(evt)" onmouseover="onToolOver(evt)" onmouseout="onToolOut(evt)"/>
```

```
    <image id="east" xlink:href="./images/east_1.gif" x="125" y="55" width="16" height="16"
onclick="onEast(evt)" onmouseover="onToolOver(evt)" onmouseout="onToolOut(evt)"/>

    <g id="mapgrp" style="display:inline"/>
    <rect id="ttrect" class="ttrect" width="0" height="15"/>
    <text id="tttext" class="tttext" x="0" y="0">ToolTip</text>

    <rect id="zoomrect" class="zoomrect" width="0" height="0"/>
    <rect id="eventrect" class="eventrect" width="100%" height="100%"
onmousedown="noop(evt)" onmousemove="noop(evt)" onmouseup="noop(evt)"/>

    <g id="loading" style="display:inline">
        <rect x="100" y="100" width="330" height="270" class="loadingrect"/>
        <text x="250" y="250" class="loadingtext">GETTING MAP FROM WEB
SERVICE</text>
    </g>


</svg>
```

Geocodde.jsp

```
<%@page import = "esri.arcwebservices.v2.*" contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/base.css">
        <script language="JavaScript" type="text/javascript">
    function init() {
     parent.mainFrame.src = "geocode.jsp";
    }
    function ValidateThis() {

        <% java.util.ArrayList resultslist =
(java.util.ArrayList)request.getSession().getAttribute("resultlist");%>
        var resultcount = <%= resultslist.size() %>;
        if (resultcount == 0) {
            alert('None of the records are geocoded, Map can not be obtained.')
            return false ;
        } else { return true; }
    }
    </script>

    </head>
    <body onLoad="JavaScript:init();" >
        <table width="100%" border="0" cellspacing="0" cellpadding="0" bgcolor="#808080">
            <td><img width="5" height="35" border="0">
```

```
                    <span class="yellow"> Integration of </span>
                    <span class="saddlebrown"> Web Services and </span>
                    <span class="white"> SVG (Scalar Vector Graphics). </span>
                </td>
            </table>
            <hr>
            <h3> Geocoding Results</h3>
            <script language="JavaScript" type="text/javascript">

                var x = 0 ;
                var y = 0;

                <% int total = (java.lang.Integer)request.getSession().getAttribute("total");%>

                var count = <%= resultslist.size() %>;
                var total = <%= total %>;


                <% for (int i =0; i < resultslist.size(); i++) { %>
                    <% esri.arcwebservices.v2.Point p = (esri.arcwebservices.v2.Point)
resultslist.get(i);%>
                    x = <%= p.getX()%>;
                    y = <%= p.getY()%>;
                    i = <%= (i + 1) %>;
                    // document.write("\n Result :" + i + " X = " + x + " , Y = " + y );
                <% } %>

                document.write ( count + " of " + total + " geocoded...");

            </script>
            <p>
             <hr>
             <h3>Select the color of the dots to be rendered:</h3>
                <form action="Controller">
                    <select name="color">
                        <option value="red">Red</option>
                        <option value="green">Green</option>
                        <option value="blue">Blue</option>
                        <option value="yellow">Yellow</option>
                    </select>
                    <input type="submit" value="Update Color" >
                    <script language="JavaScript" type="text/javascript">
                     <% String dotcolor =
(java.lang.String)request.getSession().getAttribute("color");
                        if (dotcolor == null) dotcolor = "red"; %>
                        var color = "<%= dotcolor %>";
                     document.write ( " <h4> Current Selected Color = " + color + "<h4>");
                     </script>
                    <input type="hidden" name="cmd" value="geocode" >
                    <input type="hidden" name="startup" value="false" >
                </form>
            </p>
            <p>
```

```
        <h3>Select the size of the dots to be rendered:</h3>
            <form action="Controller">
                <select name="size">
                    <option value="5">5</option>
                    <option value="7.5">7.5</option>
                    <option value="10">10</option>
                </select>
                <input type="submit" value="Update Size" >
                <script language="JavaScript" type="text/javascript">
                 <% String dotsize =
(java.lang.String)request.getSession().getAttribute("size");
                    if (dotsize == null) dotsize = "5"; %>
                    var size = "<%= dotsize %>";
                 document.write ( "<h4> Current Selected size = " + size + "<h4>");
                 </script>
                <input type="hidden" name="cmd" value="geocode" >
                <input type="hidden" name="startup" value="false" >
            </form>
        </p>
        <hr>
        <p>
            <!-- <a href="map.jsp"> Get Map </a> -->
            <form action="Controller" onsubmit="javascript:return ValidateThis()">
                <br>
                <input type="submit" value="Get Map" >
                <input type="hidden" name="cmd" value="getMap" >
                <input type="hidden" name="startup" value="true" >
            </form>
        </p>

    </body>
</html>
```

Getfile.jsp

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <link rel="stylesheet" type="text/css" href="css/base.css">
        <script language="JavaScript" type="text/javascript">
    function init() {
     parent.mainFrame.src = "getfile.jsp";
    }
    function IsEmpty(aTextField) {
     if ((aTextField.value.length==0) ||
     (aTextField.value==null)) {
          return true;
     }
     else { return false; }
    }
```

```
    function ValidateForm(form)
    {
     if(IsEmpty(form.aCSVFile))
     {
          alert('You have not entered the input file')
          form.aCSVFile.focus();
          return false;
     }
     return true;
    }
        </script>
    </head>
    <body onLoad="JavaScript:init();" >
        <table width="100%" border="0" cellspacing="0" cellpadding="0" bgcolor="#808080">
            <td><img width="5" height="35" border="0">
                <span class="yellow"> Integration of </span>
                <span class="saddlebrown"> Web Services and </span>
                <span class="white"> SVG (Scalar Vector Graphics). </span>
            </td>
        </table>
        <hr>
        <FORM ENCTYPE = 'multipart/form-data' METHOD= 'POST' ACTION =
"Controller?cmd=MapColumns&startup=true" onsubmit="javascript:return ValidateForm(this)">
            <h3>Please select the input file : <h3>
            <input type="file" name="aCSVFile" value="" width="50" />
            <p>
                <input type="submit" value="Upload" name="upload">
            </p>
        </FORM>
    </body>
</html>
```

REFERENCES

[1]. Core Java 2, Volume One: Fundamentals (7^{th} edition), Cay S. Horstmann and Gary Cornell, Sun Microsystems Press 2001.

[2]. Java Servlet Programming, Hunter & Crawford, O Reilly 1998

[3]. Applied Java Patterns, Stephen Stelting & Olava Maassen, Sun Microsystems Press 2002.

[4]. JavaServer Pages Developers Handbook, Nick Todd & Mark Szolkowski, Sams Publishing 2003.

[5]. JavaScript by Example, Ellie Quigley, Prentice Hall 2003

[6]. Beginning J2EE 1.4, James L. Weaver et al, Apress 2004.

[7]. Unified Modeling Language User Guide, Grady Booch et al, Addison Wesley 1999.

[8]. A Use Case Template, Derek Coleman, Hewlett-Packard Corporation, available at http://www.bredemeyer.com/pdf_files/use_case.pdf

[9]. Scalable Vector Graphics (SVG) 1.1 Specification: http://www.w3.org/TR/SVG/

[10]. ESRI Arc Web Services web site. http://www1.arcwebservices.com/v2006/index.jsp

[11]. SVG Developer Knowledgebase: http://support.adobe.com/devsup/devsup.nsf/svgkb.htm

[12]. FLEX overview http://www.adobe.com/products/flex/overview/

[13]. Wikipedia http://www.wikipedia.org/