Web Design, Development and Security

By Purushottam Panta

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Mathematics

YOUNGSTOWN STATE UNIVERSITY

May 2009

Web Design, Development and Security

Purushottam Panta

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Purushottam Panta

Approvals:

Dr. John Sullins, Thesis Advisor

Dr. Graciela Perera, Committee Member

Dr. Jamal Tartir, Committee Member

Dr. Peter J. Kasvinsky,
Dean of the School of Graduate Studies and Research

Abstract

Websites are the most convenient way to present and disseminate information to the maximum number of people in the world. The web browsers are the means to render the information on web page, the basic building blocks of a website, and web programming is the basic structure (architecture) of each web page.

The thesis on "Web Design, Development and Security" is a complete analysis of website design and development. Web sites should be able to present abundant information to a visitor in well organized manner. In addition, there must be a reliable transfer of secure information between server and client. There exist other major factors such as user friendliness, layout, simplicity, ease of rendering in the browser and so on that are closely related with the quality of website. This thesis will elaborate on a number of issues that are related with web design and development. These principles will be illustrated and demonstrated in the design of some websites that I have designed so far.

Acknowledgement

Special thanks to my parents (Gehendra M. Panta and Bed Kumari Panta), Brother Nagendra M. Panta my friend Sami and other college friends for their continuous encouragement.

I would like to express my special thanks to my thesis advisor, Dr. John Sullins, thesis committee members, Dr. Graciela Perera and Dr. Jamal Tartir for their precious suggestion and encouragement.

Special thanks to my professors Dr. Nathan P. Ritchey, Dr. S. E. Rodabaugh, Dr. Steven Kent, and Dr. Frank Ingram for their encouragement and inspiration to thesis, related study and research work

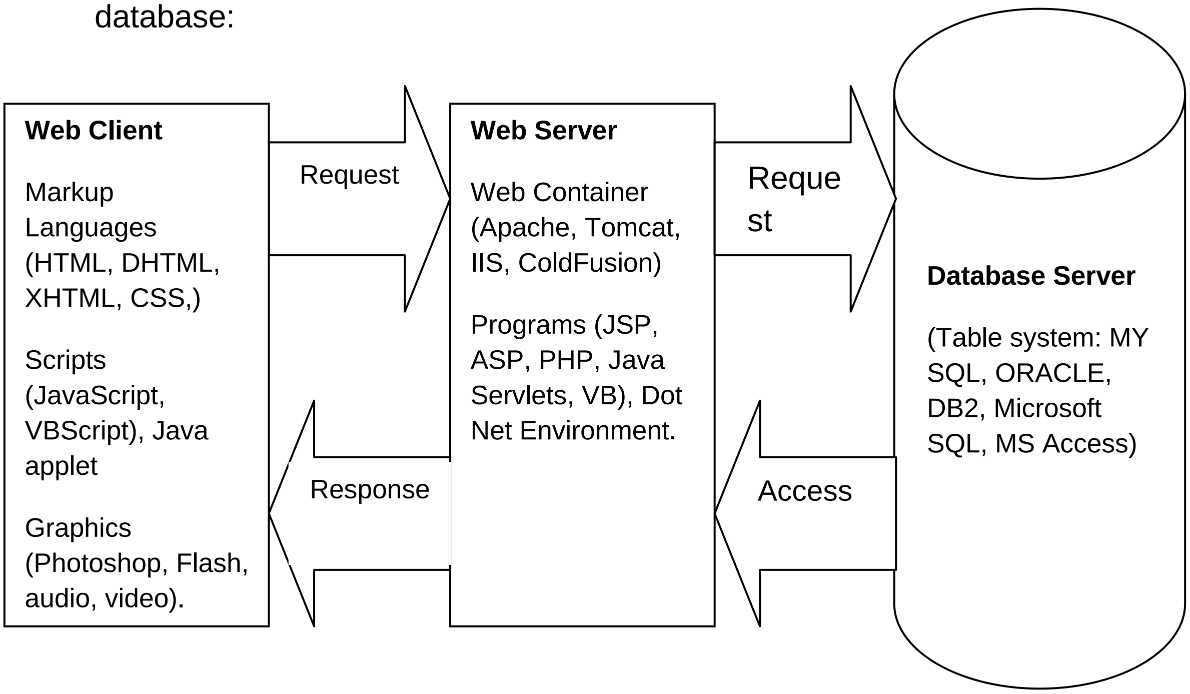
Contents:

(1) Introduction to web design and development and its importance.

(1.1) Overview on web design and development:

Nowadays websites are the most convenient way to present and disseminate information to the maximum number of people in the world. The web browsers are the means to render the information on a webpage, the basic building blocks of a website which has the basic structure (architecture) written in web Program. In today's Information age, almost all organizations have a website with their manifesto and their product and service information. It is probably the most economic and the most convenient way to disseminate information all over the world.

We will define the website as a big container of relevant and related information arranged in some logical way. Web design can be best viewed as a client-server architecture, where a client machine requests for service(s) and the server validates the request to access service, probably from a database:


Web Client
Markup Languages (HTML, DHTML, XHTML, CSS,)
Scripts (JavaScript, VBScript), Java applet
Graphics (Photoshop, Flash, audio, video).
Request
Response
Web Server
Web Container (Apache, Tomcat, IIS, ColdFusion)
Programs (JSP, ASP, PHP, Java Servlets, VB), Dot Net Environment.
Reque st
Access
Database Server
(Table system: MY SQL, ORACLE, DB2, Microsoft SQL, MS Access)

A typical Client Server Architecture

Depending on the functionalities, a complete web design can be further classified into three main components (Ref. Text: Murach's Java Servlets and JSP):

(a) Client Side Design:
Client Side is the actual interface for the user. The application such as web browser (IE, Netscape, Google Chrome) on the client machine sends service-request data to the web server (TOMCAT, APACHE, IIS) running on the Server Machine. The Web Server then either sends an existing page to the Client Machine or generates a new page and sends to the client machine accordingly. The Client Side web page is typically constructed by HTML, CSS and some Script (JavaScript, VBScript).

(b) Server Side Design:
Server side is the logical controlling part of the website. The web container (Such as Apache, Tomcat, IIS) running under the server machine handles the client request, validates with the server side program (written in ASP, JSP, PHP, Java, VB, or C++) and then generates an appropriate page or locates an existing appropriate page and sends that page to the client side. Server pages are typically written in JSP, PHP, or ASP.

(c) Database Design:
Database is always at the back end of the client-server architecture. The data stored in the database is gathered, organized and designed in a sophisticated logical manner (Such as using DFD, RDBMS, OODBMS, or UML) and stored in one or more tables. The web server can pull up data with the help of a database server (Such as MYSQL, Microsoft SQL, and Oracle), fit it into a web page and send it to the client machine.

(1.2) The design and development of website as a process:

The design and development process of a complete website is not a complex process as long as we follow a sequence of steps to make the system be done effectively. This section describes a sequence of steps that generally should be followed in order to accomplish the design effectively and efficiently, which in turn helps to maintain, modify, and update the system professionally:

Step 1: Determine the objective and the structure of the organization:

The design of the website varies depending on the objective and the structure of the organization. For example, a business organization may not have a similar website as a non-profit organization has. Furthermore, the website of a financial institution such as bank shouldn't resemble the website of online auction business organization such as eBay. In the same way, a government organization should have a lot of unique characteristics. The first thing that we must concern ourselves with in the design of the website is to understand the objective of the organization and the organizational structure, which in turn helps us to determine the class, objects and the entities to be included in the website. It helps to sketch a mental model of the website with a number of functions and facilities.

Step 2: Feedback from the possible users (For example the employee of the organization):

Before designing the website, it is a good approach to get feedback from actual users. If there exist already a website, we would get the feedback for the existing website. If we are designing a complete new website, still it's better to know the user requirement before starting design. When I was designing http://www.lightgov.com I had a rapid interaction with my supervisors and co-workers to determine the requirements.

Step 3: Project Planning:

Web designers must be aware of all of the requirements, budget, and time constraints. In this step the main concern is to manage (plan, organize and control) the available resources in such a way that brings about the successful completion of the website project in required time. I was running on predetermined schedule and fixed plan of working in a scheduled manner to complete individual

Step 3: Component wise website analysis and design:

Client Side Design:

- Programs use: XHTML, CSS, Scripts (JavaScript, VBScript), Java Applet.
- Function: Direct Client Interface, Getting user input and do some computation.
- Common interfacing environment: Web browser.
- Objectives: Error Free, Well formatted layout and text, User interactive, Ease of use, Informative, Cross browser supporting.
- Objectives: Error Free, Well formatted layout and text, User interactive, Ease of use, Informative, Cross browser supporting.

Server Side Design:

- Programs use: XHTML, CSS, JavaScript / VBScript, ASP / PHP / JSP, Java Servlet, VB, ColdFusion. And Database Programs (SQL / ORACLE / DB2) to interact with Database server.
- Environment: IIS, Apache Tomcat.
- Function: Grabbing the information from client side send by user, Perform some computation, validate input, request for access to the database, handle the executed query from database (Add, Delete, Modify or Access the database) and generate the appropriate webpage to be render in client side.
- Objectives: Error free program-computation, Error free execution of server program, scripts and markup language.

Database Design:

- Program use to access: SQL / Oracle.
- Design Tools: DFD, DBMS / RDBMS, OODBMS, UML.
- Objectives: Reliability of Information stored, well organized or designed Logical structure and appropriate relation among entities, following a defined structure such as OODBMS.

Step 4: Implement the complete system and testing:

- Establish the connections between Client-Server-Database components and Implement the whole system (Client-Server-Database) for real.
- Inject various inputs from different users to the system, observe the results carefully. Find out if the system is working properly or find out if there are still errors in the page(s). Correct the error(s) discovered.

Step 5: Get the feedback from users:

- Now, launch the website as a Beta-version and don’t forget to include a feedback form for the user so that they can provide different types of comment and valuable feedback about the website.

Step 6: Make any necessary change, modification according as the user feedback:

- As the final step of the design, make any necessary change, modification, update, correction according as the user feedback. However, user feedbacks and appropriate update, modification are never ending entities. We must listen our users for the improvement in our service.
- Don’t forget to encourage user send feedback.

(2) Design issues on Web service components:

Web service design is not only the raw coding and programming but also a careful study of the service type and design of the three different modules integrated together. We describe a number of issues that directly related with the quality of the web service in terms of user (customer) satisfaction. The following design issues plays crucial role to serve better to user:

(1) Error free:

When the webpage renders in the client browser, it is very important to be render without error. It is the frustrating experience for the user to see errors in the webpage. So any types of error Scripting errors or computation errors must not be present in the website.
We can categorize the error as:

Syntax Error	**Logical Error**
- Commonly happens in programming due to the typing mistake. - Easy to find and fix-up - Doesn't cause the change of entire system - For example: The missing semicolon, missing parameter, brackets.	- Insufficient study causes the logical error. - Difficult to find and fix-up. - Sometimes the entire component should be redesign for the logical error. - Example: Buffer overflow for the variable, inappropriate math-formula used in computation.

Some of the common error examples are:

- *Script (JavaScript / VBScript) error:*
  It causes either the page loading with error or not loading at all. If the page loads with error, any event such as OnMouseover, Onclick, Ondblclick or other user defined functions won't work. So, it may disable the submitting a form to the server or loading style of the page (CSS) improperly.
- *Server program error:*
  The erroneous web server programs files (Such as ASP, PHP, JSP or DLL, Servelet and C++) forces the server to behave abnormally, can't handle the request. For example, in a shopping cart system, calculation error may cause the calculation for X+1 quantity for the order of X quantity and the User gets the wrong price.

The logical error in the server side sometimes leads to very serious security problems such as SQL injection.

- *Database design error:*

  Defining the insufficient length of the variable or data type incompatibility are the most common errors in a database

(2) Browser compatible markup:

It is very important that the markup, style sheet, or script that we use in the website should be supported by all of the browsers in use, so that the webpage can be correctly rendered in any web browser. We have to use those properties and codes that are understood by the interpreter of most of the browsers.

(3) Simplicity:

"Keep it simple!"

Simplicity is easily to quote but often ignored in strange ways. Perhaps this is because it is the eye of the beholder.

A language which uses fewer basic elements to achieve the same power is simpler. Sometimes simplicity is confused with “easy to understand". For example, a two-line solution which uses recursion is a pretty simple, even though some people might find it easier to work though a 10-line solution which avoids recursion.

Use very simple primary tags to design a website instead of a complex twisted program or script.

(4) Uniform view:

A website for an entire company should have a similar appearance and layout so that the user won’t be confused about which organization website they are currently visiting

We can assume some common layout to be the same for the entire website of an organization. Generally, a left menu bar, top menu bar, a footer bar and a topmost bar with company logo must appear in each page of the website, which gives a uniform view for the entire website.

(5) Less use of Multimedia data and plug-ins:

Images are costly to download, so the size of images should be minimized if possible. For example, a background with a small image repeated is much less costly than a single large image.

Unless we are designing a commercial website or it is must to use multimedia data, we should try to avoid putting in audio, video or any plug-in files for the following reasons:

- The multimedia data (audio or video) needs a high bandwidth to get a good quality and we cannot assume that all users must have a high bandwidth internet connection.
- The multimedia data (audio or video) can't be played without plug-ins. For example a flash movie needs a flash plug-in installed in web browser to be played, and a lot of users are not even familiar with browser plug-ins.

(6) User Control:

Give as much as possible control to the user. For example the client user should be able to change the font size, background color, font color, page layout appropriately.

(7) Intelligent User Interaction:

The website programming should be intelligent enough for what the client user trying to do and give him/her an informative solution approach. For example, a user unsuccessfully attempting to log in should result in either useful information to reset the password, or to contact to the administrator, or to block the account temporarily.

(8) Printer friendly version, sitemap and site search capability:

When there is much text information in a webpage, there should also be a printer friendly version of that page. "Printer friendly version" of the page refers most of the time to a page designed to print only the black and white text without background color and images.

Moreover, It is very good practice to put a sitemap link in every page of the whole website. It helps the user to find the required page immediately and doesn't let them become lost.

In addition, we can have a word search capability for the entire pages so that user can find out the pages having the exact or similar information they want to see.

(9) Accessible design for the user with various disabilities:

It is very important to design the website to be accessible for the people with different disabilities. Technically, the web-program should be interpreted in such a way that makes ease access for disable users.

- Visual disable users: The background color and the text color should be chosen in such a way that the text can be readable for the colorblind peoples too.
- For blind users: The textual page can be easily fed to the screen reader so it is relatively easy to access for blind peoples.
- Use “ALT” attribute to all images with related text description.
- Use “tabindex” attribute to make the order of all URL used in the page.

(10) Globalization:

Web service can be reached by people around the world, so it has to be generalized to all the peoples in the world.
For example: We can present web site services in different languages targeting the customers from different geographical locations.

(11) Request handling:

A web server may have a very large number of visitors. The server must be capable of handling the maximum flow of information back and forth to the client, or the server should be capable of handling the multiple requests simultaneously.

(12) Legal Issue:

Legal issue is also a major factor to be considered. We have to present the information with our own or give the credit to others as well. The copyright reserved materials, patient, terms and conditions, user agreements are basically written or made with the help of cyber lawyer, so that we are not going to face legal issue in the future.

(13) Solve Atomicity:

The server must be able to handle the problem of Atomicity (a series of database operations in which either all occur, or nothing occurs).
For example: Either both pay and reserve the seat, or neither pay nor reserve the seat.

(14) Following Object Oriented Concept:

- Encapsulation:
  This is very important object oriented concept bundling or collecting related components together. Collecting related WebPages or other correlated website components inside a common bundle in the case of web design. We can bundle related pages (related in term of sharing same or similar data, image and so on.) together for the simplicity and ease of maintenance, update.
  For example, the entire YSU domain has a number of different departments each having separate websites bundling a number WebPages. A chemistry department webpage normally may not be inside mathematics department web-bundle. Each department website directory bundles the correlated page and other website components.

- Modularity:
  When you design a system, or a language, then if the features can be broken into relatively loosely bound groups of relatively closely bound features, then that division is a good thing to be made a part of the design. This is just good engineering. It means that when you want to change the system, you can in the future change only one part, which will only require you to understand (and test) that part. This will allow other people to independently change other parts at the same time.
  For example, a calendar, a login-form, the main menu, submenus can be designed as different modules and then joined together.
  Modular design hinges on the simplicity and abstract nature of the interface definition between the modules. A design in which the insides of each module need to know all about each other is not a modular design but an arbitrary partitioning of the bits. Modular design is the basis of Object Oriented website development.

- Hierarchy (Inheritance):

  The concept of inheritance applies in the website when the components or relevant modules inherit the properties from parents. For example the menu and submenus, page and subpages should be based on concept of inheriting the properties from the parent. It makes the maintenance very easy and flexible to design.

  A clearer example, we can divide a shopping cart in a series of steps (Generally, collecting user information, shipping address, payment information, billing and place order). Commonly we can go to the next step only upon completion of the previous step. So the information or properties from the previous pages are inherited to the next page.

(3) Web server security challenges and defense:

Web server security is a major issue in the current world. There is much exchange of confidential information between hosts, so we can't avoid the security issue in web service.

The following table explores WHO may cause the security problem and WHY:

Adversary	**Goal**
Student	To have fun snooping on people's email, make a prank on friend's email.
Hacker	To test someone's security, experiment with extreme programming, steal information for different purposes.
Sales Representative	To claim to represent all the territories, not just America to force users view their products, advertisements.
Businessman	To discover a competitor's strategic, marketing and product plan.
Ex-employee	To get revenge of being fired. To reveal the secrets of the company to the competitor.
Accountant	To embezzle money from company
Stockbroker	To deny a promise made to a customer by email.
Con Man	To steal credit card information for sale
Spy	To learn an enemy's secret plan, military, political strength and strategies.

Both end-users and Web administrators need to worry about the confidentiality of the data transmitted across the Web. We can categorize the key web service security requirement into following four intertwined areas (Security Requirements) (Ref. text: Cryptography and Network Security, Behrooz Forouzan, McGraw-Hill, ISBN: 978-0-07-287022-0.)

1. Secrecy:

   This can be defined as keeping information out of the hands of unauthorized users. This is what usually comes to mind when people think about network security. It requires that the information in a computer system only be accessible for reading by authorized parties. This type of access includes printing, displaying and other forms of disclosure, including simply revealing the existence of an object.

2. Authentication:

   The determination of the communication parties is crucial issue for secure data transmission. So, authentication basically deals for determining whom you are talking to before revealing sensitive or secret information. It is to determine and ensure the authenticity of the communication party.

3. Non-repudiation:

   Non-repudiation Deals with the signatures of the message (Unique identification of the person. Example: Using signature to each outgoing mail in hotmail)

4. Integrity Control:

   How can you be sure that a message you received was really the one sent and not something that a malicious adversary modified in transit or concocted?

   Integrity Control cares such a requirement that only authorized parties can modify computer system assets. Modification basically includes the operation: Add and/or Delete and/or Update.

   There are four general categories of the attacks:

1. Interruption:

   An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability. Examples include destruction of a piece of hardware, such as a hard disk, the cutting of a communication line, or the disabling of the file management system.

2. Interception:

   An unauthorized party gains access to an asset. This is an attack on confidentiality. The unauthorized party could be a person or a computer program. Examples include wiretapping to capture data in a network and the illicit copying of files or programs.

3. Modification:

   An unauthorized party not only gains the access to but also tempers with an asset. This is an attack on integrity. Example includes changing values on a data file, alerting a program so that it performs different way and modifying the content of the messages being transmitted in a network.

4. Fabrication:

   An unauthorized party inserts counterfeit objects into the file system. This is an attack on authenticity. Example includes the insertion of spurious message in a network or the addition of record to the file.

Malicious inputs can cause the server to behave abnormally and reveal some flaws in the server to the attacker. Therefore, it is crucial to handle every possible input carefully. We can perform input validation in different ways.

The following are the essential methods of protecting the system from various types of attack such as SQL Injection, Bruit-forcing:

(3.1) Client Side Validation:

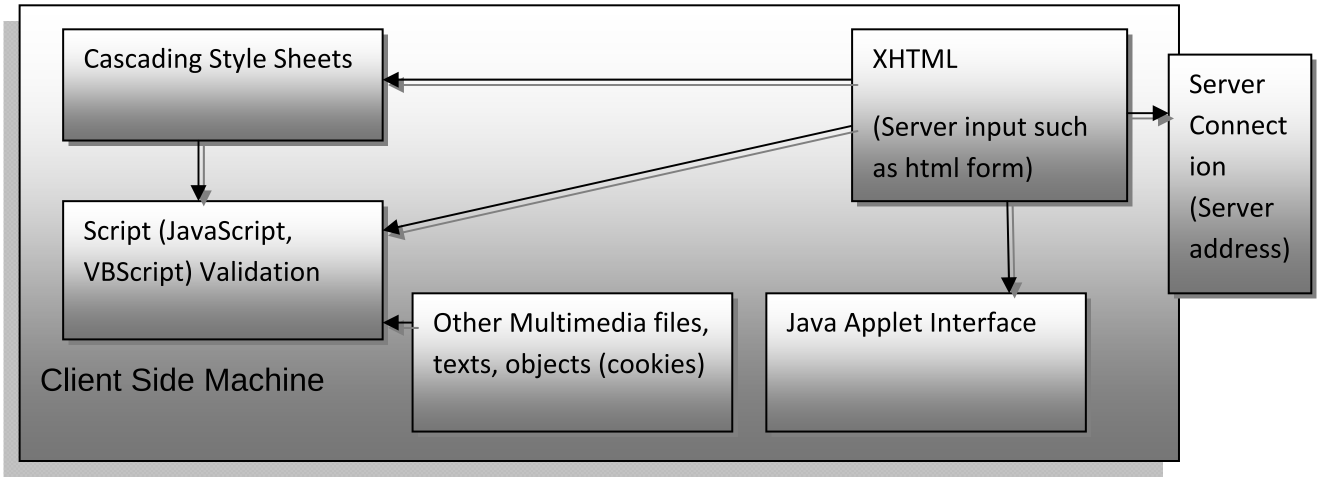
- Generally we validate the user input by script languages such as VBScript, JavaScript running on the client browser.
- We can control the input character with maxlength property in XHTML, as shown below:

  ```
  <input type="text" name="username" size="30" maxlength=10>
  ```

  Where maxlength property limits the length of the text being input.

- We can use dropdown menus as much as possible, which really helps avoiding malicious input itself and simplifies the validation process.
- Client side validation runs under the client browser, so if the script option on client browser is turned off then, the validation may not work. The client side validation scripting and algorithm is generally accessible

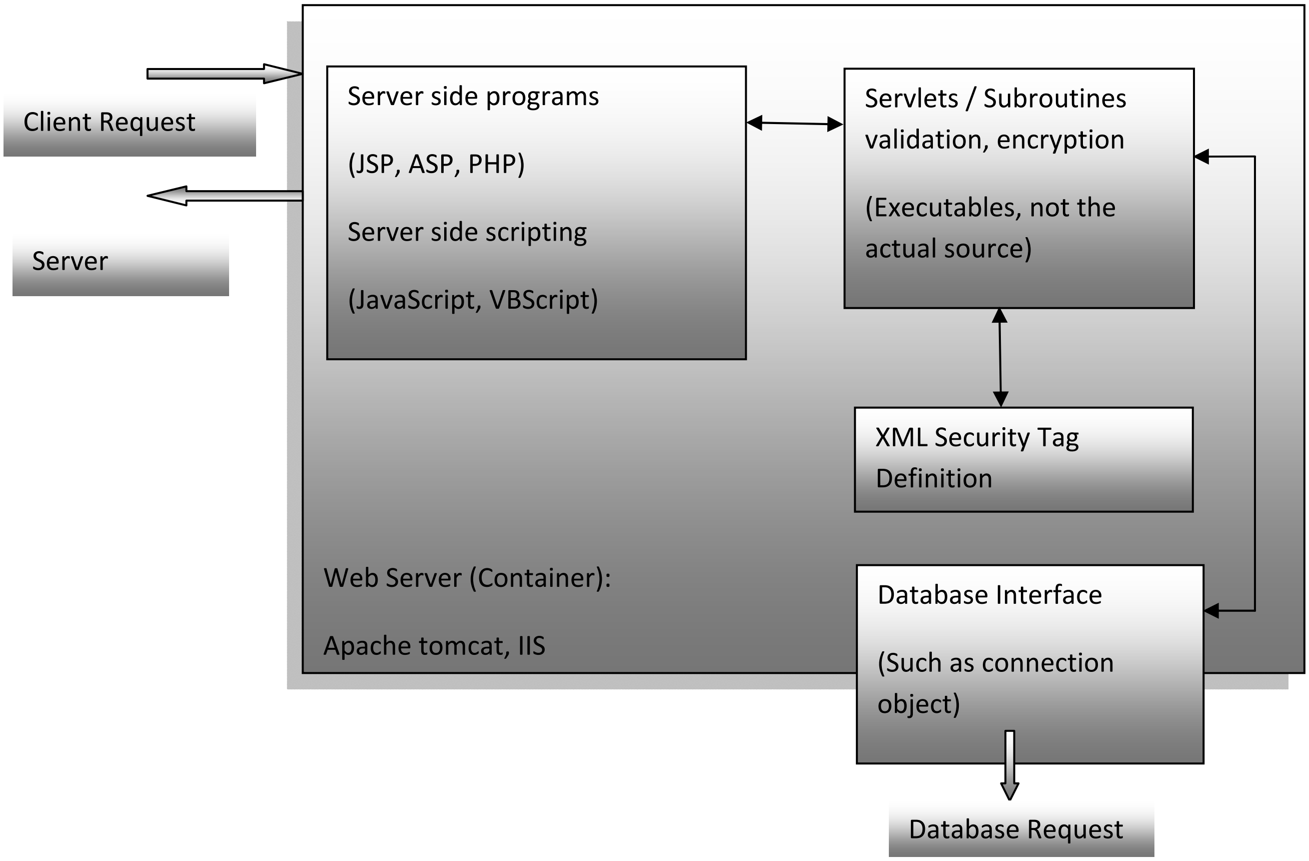
to the client machine so it is not the good practice to rely just on client side validation.


Cascading Style Sheets
XHTML
(Server input such as html form)
Server Connect ion (Server address)
Script (JavaScript, VBScript) Validation
Other Multimedia files, texts, objects (cookies)
Java Applet Interface
Client Side Machine

- I have presented some very crucial fields validation on client side with a brief description of each function on Appendix.

(3.2) Server Side - Database Input Validation:

- Server pages are intelligent enough to generate the client pages depending on the user request. Server page programs and algorithms are not visible to the clients, so it is the better approach to validate input on server side as well.
- We can use the server side programs such as ASP, PHP, JSP to validate the input directly or pass the information to some rigid program components (Servlet, DLL) usually written in Java, Visual Basic, C#,…
  (The server side validation logics as quite similar to the client side validation, just the program syntax and coding difference, so I have assumed the reader have understand the concept from client side validation.)
- A general model of server side validation:



(3.3) Overlapping types of risk:

(1) Bugs or mis-configuration problems in the Web server that allow unauthorized remote users to:

a. Steal confidential documents.
b. Execute commands on the server host machine, allowing them to modify the system.
c. Gain information about the Web server's host machine that will allow them to break into the system.
d. Launch denial-of-service attacks, rendering the machine temporarily unusable.

(2) Browser-side risks, including:

a. Active content that crashes the browser, damages the user's system, breaches the user's privacy, or merely creates an annoyance.
b. The misuse of personal information knowingly or unknowingly provided by the end-user.

(3) Interception of network data sent from browser to server or vice versa via network eavesdropping. Eavesdroppers can operate from any point on the pathway between browser and server including:

a. The network on the browser's side of the connection.
b. The network on the server's side of the connection (including intranets).
c. The end-user's Internet service provider (ISP).
d. The server's ISP.
e. Either ISP’s regional access provider.

It's important to realize that "secure" browsers and servers are only designed to protect confidential information against network eavesdropping. Without system security on the browser, server sides and database server, confidential documents are vulnerable to interception.

(3.4) A common threat: SQL Injection:

SQL injection refers to the act of inserting a SQL statement in such a way that would run on the database without server side program’s permission. Injection usually occurs when you ask a user for input, such as their name, and instead of a name they give inject such a logical SQL statement that will directly run to the database, gain access, retrieve information.

Here is an example:

A normal user and a bad user trying to use SQL Injection. We asked the users for their login, which will be used to run a SELECT statement to get their information.

My SQL & PHP Code:

```
----------------------------------------------------------------------------------
// A good user's name
$name_good = "puru";
$query_good = "SELECT * FROM customers WHERE username = '$name_good'";
echo "Normal: " . $query . "<br />";

// user input that uses SQL Injection
$name_bad = "' OR 1'";
// A bad user name Input
$query_bad = "SELECT * FROM customers WHERE username = '$name_bad'";

// display what the new query will look like, with injection
echo "Injection: " . $query_bad;
----------------------------------------------------------------------------------
Display:
Normal: SELECT * FROM customers WHERE username=’puru’
Injection: SELECT * FROM customers WHERE username= “OR 1”
```

Description:

The normal query is no problem, as our SQL statement will just select everything from customers that has a username equals to the string puru. However, the injection attack has actually made our query behave differently than we intended. By using a single quote (') they have ended the string part of our SQL query and can use other malicious query string after the quote (').

```
username = ' '
```

and then added on to our WHERE statement with an OR clause of 1=1 (always true).

```
username = ' ' OR 1=1--
```

This OR clause of 1 will always be true and so every single entry in the "customers" table would be selected and displayed by this statement!

Sometimes the attack can be worse than expected and may delete the database results a huge loss to the company. Here is the example:

MYSQL & PHP Code:

```
$name_evil = "';
DELETE FROM customers WHERE 1=1 or username = "';

// our MySQL query builder really should check for injection
$query_evil = "SELECT * FROM customers WHERE username =
'$name_evil'";

// the new evil injection query would include a DELETE statement
echo "Injection: " . $query_evil;
```

Display:

```
SELECT * FROM customers WHERE username= ‘ ’; DELETE FROM
customers WHERE 1 OR username=’ ‘;
```

It results to completely empty the “customers” table in the database.

(3.5) Defeating the SQL injection:

(Ref. article: The Unexpected SQL Injection when escaping is not enough by By Alexander Andonov http://www.webappsec.org/projects/articles/091007.shtml)

(1) Write a function in server side (make Servlet or DLL) to filter the bad string.

(2) With MYSQL_REAL_EXCAPE_STRING():

Luckily, this problem has been known for a while and PHP has a specially-made function to prevent these attacks. All you need to do is use the function mysql_real_escape_string.

What mysql_real_escape_string does is take a string that is going to be used in a MySQL query and return the same string with all SQL Injection attempts safely escaped. Basically, it will replace those troublesome quotes(') a user might enter with a MySQL-safe substitute, an escaped quote \'. We will try out this function on our two previous injection attacks and see how it works:

MYSQL & PHP Code:

```
--------------------------------------------------------------------------------
//NOTE: you must be connected to the database to use this function!
// connect to MySQL

$name_bad = "' OR 1'";

$name_bad = mysql_real_escape_string($name_bad);

$query_bad = "SELECT * FROM customers WHERE username = '$name_bad'";
echo "Escaped Bad Injection: <br />" . $query_bad . "<br />";


$name_evil = "'; DELETE FROM customers WHERE 1 or username = '";

$name_evil = mysql_real_escape_string($name_evil);

$query_evil = "SELECT * FROM customers WHERE username = '$name_evil'";
echo "Escaped Evil Injection: <br />" . $query_evil;

--------------------------------------------------------------------------------
```

Display:

```
Escaped Bad Injection:

SELECT * FROM customers WHERE username=’\’ OR 1 ‘\”

Escaped Evil Injection:

SELECT * FROM customers WHERE username =’\’; DELETE FROM customers WHERE 1 OR username=\”
```

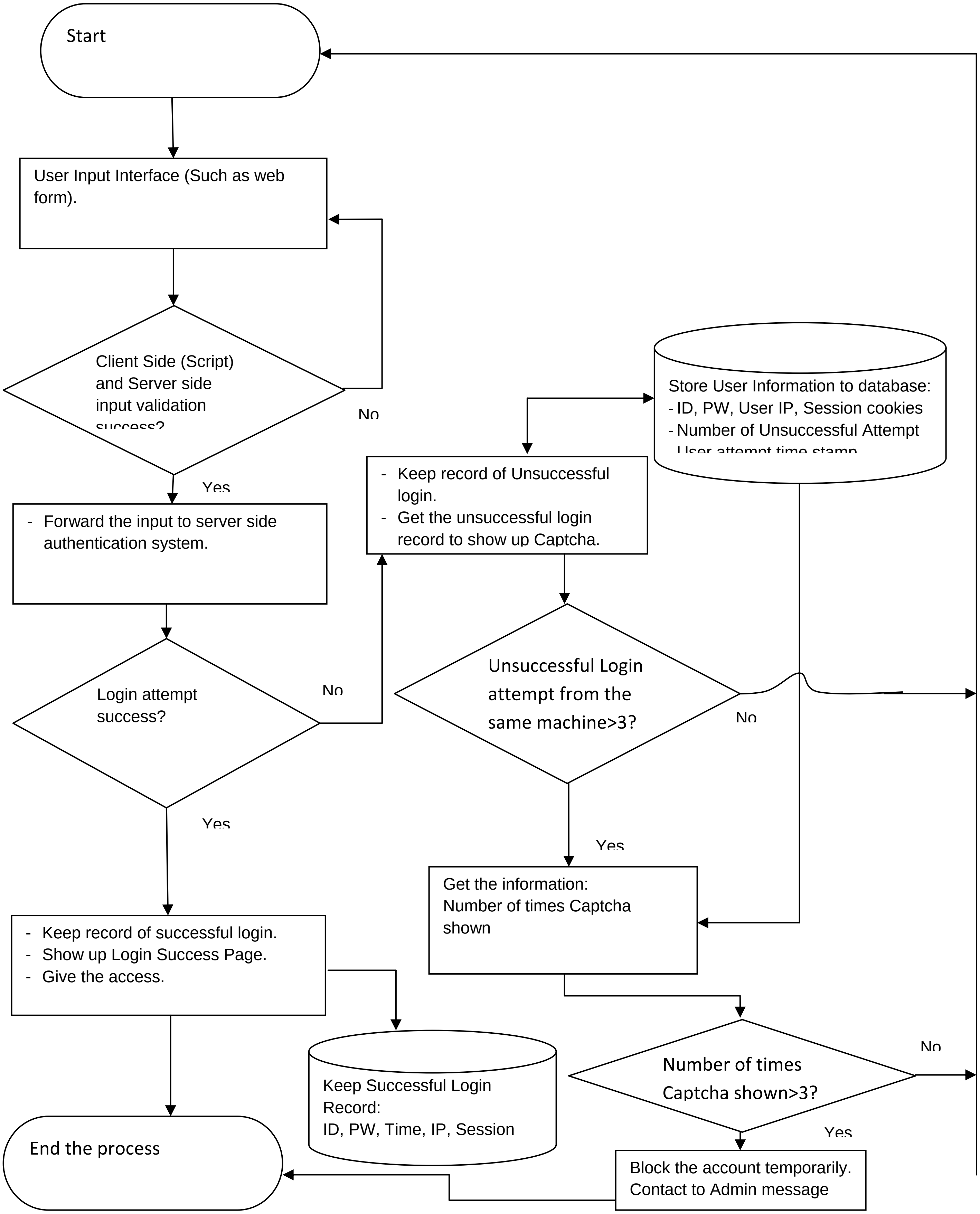
(3.6) Bruit Force defense with Human / Program Recognition (CAPTCHA):

Nowadays we can see a combination of digits and alphabet as the ‘verification code’ appeared when there is enough more unsuccessful login attempt to the server. Those combinations of digits and alphabet are in fact randomly generated images which are not easily recognized by bruit-force software. A bruit- force is a method of automated login attempt using a relevant dictionary for user name and password (Often done with the help of computer software).

Yahoomail, Hotmail, Gmail all are currently using the similar verification code to recognize whether the login request is from a human or from an automated software system. The system of generating these random alphabets is known as Captcha system.

A software generated automated login-request has some predefined characteristics such as the time interval in between any two login attempt, the number of rapid login attempt and so on. But the human login-request has no such predefined characteristics, which makes ease to distinguish the human request or software automated request and the concept of Captcha system was developed. We can use a level of filtering the rapid request in client side also, however the Captcha system would be implementing in server side.

The basic functioning of Captcha in flowchart:


Start
User Input Interface (Such as web form).
Client Side (Script) and Server side input validation success?
No
Yes
- Forward the input to server side authentication system.
Login attempt success?
No
Yes
- Keep record of successful login.
- Show up Login Success Page.
- Give the access.
End the process
Keep Successful Login Record:
ID, PW, Time, IP, Session
Store User Information to database:
- ID, PW, User IP, Session cookies
- Number of Unsuccessful Attempt
User attempt time stamp
- Keep record of Unsuccessful login.
- Get the unsuccessful login record to show up Captcha.
Unsuccessful Login attempt from the same machine>3?
No
Yes
Get the information:
Number of times Captcha shown
Number of times Captcha shown>3?
No
Yes
Block the account temporarily.
Contact to Admin message

Bruit Force Prevention by CAPTCHA

(4) Disaster Recovery Plan:

(Ref. Article: “How to Create a Disaster Recovery Plan “ Learn the basics of creating a plan that will have you prepared to recover your data and keep the business running after an IT-disabling disaster. -by Glen Kunene, Senior Editor)

Web services are the integrated part of the business in this information age. There might be a number of unforeseen disasters that may interrupt the web service, down the business, face a huge loss in term of money, revenue, reputation, and leads to the failure of business.

The key to survive in such types of IT-Disabling disaster for the continuity of the business is a set of policies and procedures called Disaster Recovery Plan (DRP). So, Disaster Recovery Plan is one of the crucial core components in smoothly running the web services and the business.

“Dollars spent in prevention are worth more than dollars spent in recovery”

(1) Risk Analysis:

Let’s generate all the possible risks that may interrupt our system. A through risk analysis would perform as the first step in drafting our DRP. Some common reason that may cause the server out of order, service unavailable is:

- Flood of the request on data center.
- Electric power outage, blacked-out on server.
- Server software crash due to the unexpected malicious request, upload.
- Hacker’s threat on the server.
- Physical breakdown, hardware failure on the server.
- Regular server maintenance.
- Some natural disasters such as: earthquake, fire, storm.

We can have a brainstorming within IT department to find out every possible risk, chance of occurrence and its importance (impact on the service). Now, we rate all the possible risks on the basis of:

- Probability of occurrence and
- Its impact.

We can realize that the Probability of Occurrence and impact both varies depending on the type of organization, services types and so on. For example a hacker’s threat may be higher in a financial institution rather than an educational institution.

For example, a small business organization in Ohio can have:

Disaster Type	Probability of Occurrence	Impact (Importance)
Electric Power Outage	High (7/10)	High (7/10)
Earthquake	Medium (5/10)	High (7/10)
Storm	Low (3/10)	High (7/10)
Hackers Threat	Medium (5/10)	Highest (10/10)

(2) Feasibility Study and Budgeting:

Wisely establish the budget is the second step on DRP. We explore a number of solutions for the same problem and analyze the quality of solution and corresponding cost, to declare a best solution. This process is known as Feasibility Study.

Now, we have a comprehensive list of the entire problem, each with a best solution. Depending on the total cost for the solutions, IT department establishes the budget for DRP. Budget establishment for the DRP varies depending on type of the service, size of the company, importance of the disaster and so on.

For example, a small business organization in Ohio may not establish the budget for earthquake disaster recovery. However they may afford Uninterruptable Power Supply (UPS) as recovery plan for electric power outage.

(3) Develop and implement the plan:

The recovery procedure script should be written in detail by IT department. The IT department will get suggestion and feedback from all other units in the organization. For example if the other department determine and suggest to the IT department that the services in the company must be normal within 48 hours of an incident to stay viable, then we as a DRP team can calculate the amount of time it would take to execute the recovery plan and have the services and business back up at the mentioned timeframe.

(4) Testing.

After setting the DRP in the company, the final stage is to test and test for all the possible consequences and disasters. Observe how our recovery plan gives the solution; make any change if necessary for the best result.

(5) Conclusion:

It is needless to say that website is imperative to promote business, publish article, journal, and discussion through blog, distance learning, online shopping fast and easy financial transaction and so on. Having a website is not just a commodity but seems must for all organization in present information age. Building a website is just not coding and programming but the interpretation of a wise design by web programming. The organized manner of designing website components and integrating, correlating in between, gives a massive flexibility in the in the website which is very important for the future as well. The security is also a vital entity in the system that is to be considering during the design for some sorts of secure transaction.

The website is a source of information. However the design part is not limited only with the information that is going to be present. The design deals with a wide range of factors those are related with the quality of website.

(6) Appendix:

Client Side JavaScript validation coding example:

/*function validate_myform( )

This is a main function that calls a series of sub functions, each of which checks a single form element for compliance. If the element complies than sub function returns an empty string. Otherwise it returns a message describing the error and highlight appropriate element with yellow.
*/

```
function validate_myform(myForm)
{
        var Err_reason = "";

        Err_reason += validate_name(myForm.lastname);
        Err_reason += validate_name(myForm.firstname);
        Err_reason += validate_name(myForm.middlename);
        Err_reason += validate_name(myForm.username);
        Err_reason += validate_pwds(myForm.passwd, myForm.matchpasswd);
        Err_reason += validate_dob(myForm.dateofbirth_month,
myForm.dateofbirth_day, myForm.dateofbirth_year);
        Err_reason += validate_ssn(myForm.my_ssn);
        Err_reason += validate_mailing_add(myForm.mailingaddress1,
myForm.mailingaddress2);
        Err_reason += validate_phone(myForm.my_phone, myForm.ext);
        Err_reason += validate_fax(myForm.my_fax);
        Err_reason += validate_email(myForm.my_email);
        //Err_reason += validate_web(myForm.my_web);
        if (Err_reason != "")
        {
        window.alert("Following fields need correction:\n\n" + Err_reason);
        return false;
        }
        return true;
}
```

/*
function validate_name(fld)
The function checks for valid lastname, firstname and username as:

For the case of mandatory fields: lastname and firstname:

```
    -  Returns "error:You didnt enter the value in field" for any null value in
lastname, firstname and username.
    -  Returns "error: The field is of wrong length" for characters less than 3
and characters more than 15.
    -  Returns "error: The field contains Illegal character" if any character
except letters, number and underscore are entered.
    For the case of optional field: middlename:
    - It doesn’t check if the field middlename is empty or not.
*/

function validate_name(fld)
{
    var error="";
   var ill_char = /\W/; // Allow letters, numbers, and underscores only.

    if(fld.name == "lastname" || fld.name == "firstname" || fld.name ==
"username")
    {
        if (fld.value.length == 0  || fld.value == null || fld.value == "")
        {
            fld.style.background = 'Yellow';
            error = "You didn't enter a "+fld.name+"\n";
        }
        else if ((fld.value.length < 3) || (fld.value.length > 15))
        {
            fld.style.background = 'Yellow';
            error = "The "+fld.name+" is of the wrong length.\n";
        }
        else if (ill_char.test(fld.value))
        {
            fld.style.background = 'Yellow';
            error = "The "+fld.name+" contains illegal
characters.\n";
        }
        else
        {
            fld.style.background = 'White';
        }
    }
    else if(fld.name == "middlename")
    {
        var ill_char1 = /\W/; // allow letters, numbers, and underscores
        if (fld.value.length == 0  || fld.value == null || fld.value.length ==
0)
        {
            fld.style.background = 'Yellow';
```

```
                        error = "You didn't enter a "+fld.name+"\n";
                }
                else if(fld.value == "Last_Name"||fld.value == "First_Name")
                {
                        error = "Invalid "+fld.name+"\n";
                        fld.style.background = 'Yellow';
                }

                else if(fld.value == "Middle_Name")
                {
                        error = "Invalid "+fld.name+"\n";
                        fld.style.background = 'Yellow';
                }
                else
                {
                        fld.style.background = 'White';
                }
        }
        return error;
}

/*
function validate_dob(dob_month,dob_day,dob_year)
The function checks for valid dob_month, dob_day, dob_year as if it is chosen
from the dropdown list or not.
If anyone is not chosen from the dropdown list then it returns "error:Please select
from the dropdown list."
*/

function validate_dob(dob_month,dob_day,dob_year)
{
        var error="";
        if (dob_month.value == "Month")
        {
                dob_month.style.background = "Yellow";
                error += "Please select month from dropdown list. \n";
        }
        else
        {
                dob_month.style.background = "White";
        }
        if (dob_day.value == "Day")
        {
                dob_day.style.background = "Yellow";
                error += "Please select Day from dropdown list. \n";
        }
```

```
    else
    {
        dob_day.style.background = "White";
    }
    if (dob_year.value == "Year")
    {
        dob_year.style.background = "Yellow";
        error += "Please select Year from dropdown list. \n";
    }
    else
    {
        dob_year.style.background = "White";
    }
    return error;
}

/*
function validate_pwds(fld0,fld1)
The function gets two password and
    - Initiate the function validate_pwd(fld) to check for the correct format of
each password field.
    - After validating each password, It compares two password string for
equality. If both password strings are equal then only password would be of
correct format.
*/

function validate_pwds(fld0,fld1)
{
    var error="";
    error = validate_pwd(fld0);
    error = validate_pwd(fld1);
    if (error == 0  || error == null || error == "")
    {
        if((fld0 != fld1) || (fld0.value!=fld1.value) ||
(fld0.value.length!=fld1.value.length))
        {
            error = "Password not matching";
            fld0.style.background = "Yellow";
            fld1.style.background = "Yellow";
        }
    }
    return error;
}

/*
    function validate_pwd(fld0)
```

```
        The function gets the password as input then,
        - Returns "error: You didn’t enter the password" for not entering
password."
        - Returns "error: The password contains the illegal characters." for
entering any character except alphabet and digit.
        - Checks for the illegal character (The letters and numbers are the only
valid characters).
        - Checks for the specific password format:
                        /^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-Z])\w*$/;
        Any combination of the characters: With at least one upper case
character and one lower case character and a digit.
        - Checks for the length of the password: 4<=password length<=15
*/

function validate_pwd(fld0)
{
  var error = "";
  var ill_Chars2 = /[\W_]/;        // Allow only letters and numbers
        //var pw_format = /^[A-Za-z]\w{2,}[A-Za-z]$/;        // The password
must be at least 4 characters long, and it starts and ends with letter.
        var pw_format0 = /^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-Z])\w*$/;        //
The password must contain at least one number, one lower case character and
one upper case character. The length of the password can be any.

        if (fld0.value.length == 0  || fld0.value == null || fld0.value == "")
        {
    error = "You didn't enter the password. \n";
    fld0.style.background = 'Yellow';
  } else if (ill_Chars2.test(fld0.value))
        {
    fld0.style.background = 'Yellow';
    error = "The password contains illegal characters.\n";
  } else if ((fld0.value.length < 4) || (fld0.value.length > 15))
        {
    error = "The password is of wrong length.\n";
    fld0.style.background = 'Yellow';
  } else if (!(pw_format0.test(fld0.value)))
        {
    error = "The password must contain at least one numeral character, one
lower case character and one upper case character.\n";
    fld0.style.background = 'Yellow';
        } else
        {
    fld0.style.background = 'White';
  }
 return error;
```

```
}

/* United States Social Security Number (SSN) Validation:

A valid social security number consist three groups:
            - A group of 3 numbers:
                        These first 3 digits are assigned to individual states, territories
or groups of the people, which can range on 001 to 772.
            - Then a group of 2 numbers:
                        It can range from 01 to 99
            - Then a group of 4 numbers:
                        It can range from 0001 to 9999

So the ssn_format can be defined as:
            /^([0-6]\d{2}|7[0-6]\d|77[0-2])([ \-]?)(\d{2})\2(\d{4})$/;

This means:
            /^([0-6]\d{2}|7[0-6]\d|77[0-2]):
                        - First digit must be in between 0 to 6, followed by 2 digits.
                        - The character "|" stands for "OR".
                        - The digit 7 can be followed by any digit in between 0 and 6,
and
                        - 77 can be followed by any digit in between 0 and 2.
            ([ \-]?):
                        - Checks for separator either space or das to occur 0 or 1 time
                        (The "?" character means "the preceding must occur a
maximum of one time or occur at a 0 time.)
                        - This is the second group, with round parenthesis, so it would
be remembered as the format of second group.
            (\d{2})\2:
                        - Checks for 2 digits followed by space format same as in
second group: \2.
             (\d{4})$/:
                        - End up with 4 digits.
                        - We have validated the zero's case in if-else statement.
*/

function validate_ssn(user_ssn)
{
   var error = "";
            var trimed_ssn = user_ssn.value.replace(/^\s+|\s+$/, '');
            var ssn_format = /^([0-6]\d{2}|7[0-6]\d|77[0-2])([ \-]?)(\d{2})\2(\d{4})$/;

            if((user_ssn.value == "") || (user_ssn.value == null) ||
(user_ssn.value.length == 0))
            {
```

```
				user_ssn.style.background = 'White';
		}
		else
		{
				if (!ssn_format.test(trimed_ssn))
				{
						error = "SSN Format Error, please correct the SSN format.\n";
						user_ssn.style.background = 'Yellow';
				}
				else if(trimed_ssn.value.length<9)
				{
						error = "SSN is of wrong length";
						user_ssn.style.background = 'Yellow';
				}
				else
				{
						user_ssn.style.background = 'White';
				}
				var temp = user_ssn;
				if (user_ssn.indexOf("-") != -1) { temp = (user_ssn.split("-")).join(""); }
				if (user_ssn.indexOf(" ") != -1) { temp = (user_ssn.split(" ")).join(""); }
				if (user_ssn.substring(0, 3) == "000") { return false; }
				if (user_ssn.substring(3, 5) == "00") { return false; }
				if (user_ssn.substring(5, 9) == "0000") { return false; }
		}
		return error;
}

/*
function validate_mailing_add(add1,add2)
		- This function gets two address strings as input and returns error if exists.
		- All the _, -, , . Characters would be replaced from both strings and go for further validation.
		- Returns error if the address string length is not in between 3 and 50
		- Returns error if there exist any character except number, alphabet, and underscore.

*/

function validate_mailing_add(add1,add2)
{
		var error="";
```

```
    var add1_chars = add1.value.replace(/[\ \-\.\_\ ]/g, '');
    var add2_chars = add2.value.replace(/[\ \-\.\_\ ]/g, '');
    var ill_char = /\W/; // Allow letters, numbers, and underscores.

        if (add1.value.length == 0  || add1.value == null || add1.value ==
"")
        {
            add1.style.background = 'Yellow';
            error = "You didn't enter a "+add1.name+"\n";
        }
        else if ((add1.value.length < 3) || (add1.value.length > 50))
        {
            add1.style.background = 'Yellow';
            error = "The "+add1.name+" is of the wrong length.\n";
        }
        else if (ill_char.test(add1_chars))
        {
            add1.style.background = 'Yellow';
            error = "The "+add1.name+" contains illegal
characters.\n";
        }
        else
        {
            add1.style.background = 'White';
        }

        if((add2.value.length == 0)||(add2.value == "")||(add2.value ==
null))
        {
            add2.style.background = 'White';
        }
        else
        {
            if ((add2.value.length < 3) || (add2.value.length > 50))
            {
                error += "The "+add2.name+" is of the wrong
length.\n";
                add2.style.background = 'Yellow';
            }
            else if (ill_char.test(add2_chars))
            {
                error += "The "+add2.name+" contains illegal
characters.\n";
                add2.style.background = 'Yellow';
            }
            else
```

```
                              {
                                        add2.style.background = 'White';
                              }
                    }
                    return error;
}

/*
function validateEmail(fld)
          - This function first checks for if the field is empty. It is empty, returns
error message.
          - If there is space at the beginning or end of the email, the function
trimmed the blank spaces at first and last.
          - Now the if-else statement checks for the correct email format as:
                    /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$/;
                    There are three parts in this format: ppanta@ysu.edu
                    /^([A-Za-z0-9_\-\.]): Must begin with any character A-Z or a-z or
0-9 or - or _ or . of any string length.
                    \@([A-Za-z0-9_\-\.]): Followed by the @ sign and then any
character A-Z or a-z or 0-9 or - or _ or . of any string length.
                    ([A-Za-z]{2,4})$/: Any character A-Z or a-z of length 2 or 3 or 4.
*/

function validate_email(fld)
{
   var error="";
          var trimed_email = fld.value.replace(/^\s+|\s+$/, '');
          var email_format = /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-
z]{2,4})$/;
   if (fld.value == "")
          {
      fld.style.background = 'Yellow';
      error = "You didn't enter an email address.\n";
   }
          else if (!email_format.test(trimed_email))
          {
      fld.style.background = 'Yellow';
      error = "Please enter a valid email address.\n";
   }
          else
          {
      fld.style.background = 'White';
   }
   return error;
}
```

```
/*
function validatePhone(ph,ext)

The function checks if the phone number and / or extension is in valid format.
            - At first we use regular expression and the replace() method to clear out
any spacer characters.
            - Next, we check if the field is empty or not. If the phone number field is
empty it returns error.
            - Now, we use the isNaN() function to check if the phone number contain
only numbers with the length of 10.
            - At last we check the length of the string and permit only phone
numbers with 10 digits.

            (/^[2-9]\d{2}-\d{3}-\d{4}$/
Broken down into sections:
            /^[2-9]: The phone number string must begin (^) with a digit between 2
and 9; because the start of the area code cannot be 0 or 1.
            \d{2}: The first digit of the area code must be followed by any two
additional digits.
            -The area code must be followed by a hyphen.
            d{3}-: The hyphen after the area code must be followed by three digits
which is followed by hyphen.
            d{4}$: The local number prefix d{3} must be followed by four additional
digits ending the phone number string ($)
            The validation for the extension part is done checking if the extension is
numeric value or not with isNAN() method in JavaScript.
*/

function validate_phone(ph,ext)
{
    var error = "";
            var trimed_phone = ph.value.replace(/^\s+|\s+$/, '');
            var phone_digits_only = trimed_phone.replace(/[\(\)\.\-\ ]/g, '');
            var phone_format = /^[2-9]\d{2}-\d{3}-\d{4}$/;
            if (ph.value.length == 0  || ph.value == null || ph.value == "")
            {
        error = "You didn't enter a phone number.\n";
        ph.style.background = 'Yellow';
            }
            else if (isNaN(phone_digits_only) || (phone_digits_only.length!=10))
            {
        ph.style.background = 'Yellow';
        error = "The phone number must be 10 digits.\n";
            }
            else if (!phone_format.test(trimed_phone))
```

```
          {
      error = "Phone number is not in correct format.\n";
      ph.style.background = 'Yellow';
          }
          else
          {
                    ph.style.background = 'White';
          }
          if (isNaN(ext.value))
          {
                    error += "Your phone extension is in incorrect format. \n";
                    ext.style.background = 'Yellow';
          }
          else
          {
                    ext.style.background = 'White';
          }
   return error;
}


/*
function validate_web(web_field)

          - The web field is an optional field in the form, so the function checks if and only if the field is not left null.
          - It checks for any illegal character (Except letter, number and underscore) and returns the error message.
          - The field should be either blank, if not there must be the valid character entered in the field.
*/

function validate_web(web_fld)
{
          var ill_char1 = /\W/; // Allow letters, numbers, and underscores
          if (web_fld.value.length != 0   || web_fld.value != null || web_fld.value.length != 0)
          {
                    if(ill_char1.test(web_fld))
                    {
                              error = "Invalid characters in "+web_fld.name+"\n";
                              web_fld.style.background = 'Yellow';
                    }
                    else
                    {
                              web_fld.style.background = 'White';
```

```
                }
        }
        else
        {
                web_fld.style.background = 'White';
        }
        return error;
}
/*
function validate_fax(fax)
        - The field for fax is optional in the form.
        - If the fields is left blank it does not show error.
        - This validation is more similar to the phone number validation except
extension and it is optional field.
*/
function validate_fax(fax)
{

  var error = "";
        var trimed_fax = fax.value.replace(/^\s+|\s+$/, '');
        var fax_digits_only = trimed_fax.replace(/[\(\)\.\-\ ]/g, '');
        var fax_format = /^[2-9]\d{2}-\d{3}-\d{4}$/;

        if (fax.value.length == 0  || fax.value == null || fax.value == "")
        {
                fax.style.background = 'White';
        }
        else
        {
                if (isNaN(fax_digits_only) || (fax_digits_only.length!=10))
                {
                        fax.style.background = 'Yellow';
                        error = "The fax number must be 10 digits.\n";
                }
                else if (!fax_format.test(trimed_fax))
                {
                        error = "Fax number is not in correct format.\n";
                        fax.style.background = 'Yellow';
                }
                else
                {
                        fax.style.background = 'White';
                }
        }
        return error;
}
```

(7) Acronyms and Abbreviations:

ASP: Active Server Page.
CSS: Cascading Style Sheet.
DBMS: Database Management System.
DFD: Data Flow Diagram.
DRP: Disaster Recovery Plan.
HTML: Hypertext Markup Language.
IIS: Internet Information Server.
IP: Internet Protocol.
ISP: Internet Service Provider.
JSP: Java Server Page.
OODBMS: Object Oriented Database Management System.
OOPS: Object Oriented Program Structure.
RDBMS: Relational Database Management System.
SQL: Structured Query Language.
UML: Unified Modeling Language.
UPS: Uninterruptable Power Supply.
URL: Uniform Resource Location.
XHTML: Extensible HTML.
XML: Extensible Markup Language.

(8) References:

1. Wendy Lehnert, “Web 101 Making the Network for you”, ISBN: 0201704749.
2. Andrea Steelman and Joel Murach, “Murach's Java Servlets and JSP: Murach Books, 2nd Edition”, ISBN 978-1-890774-44-8.
3. Behrooz Forouzan, “Cryprography and Network Security: McGraw-Hill”, ISBN: 978-0-07-287022-0.
4. David Litchfield, “Lateral SQL Injection: A new class of vulnerability in Oracle” 27 Feb, 2008.
5. Andrew S. Tanenbaum, “Computer Networks: 4TH edition” ISBN-13: 978-0130661029
6. Steve Friedl’s Tech Tips, “SQL Injection Attacks by example”
   http://unixwiz.net/techtips/sql-injection.html
7. SQL Server 2008 Books Online (April 2009)
   http://msdn.microsoft.com/en-us/library/ms161953.aspx
8. SecuriTeam “SQL Injection walkthrough: 26 May 2002” (sk at scan-associates.net),
   http://www.securiteam.com/securityreviews/5DP0N1P76E.html
9. W3Schools website, “JavaScript Form Validation”,
   http://www.w3schools.com/jS/js_form_validation.asp
10. Wikipedia, “Web Design” http://en.wikipedia.org/wiki/Website_design
11. Tim Bray, Textuality and Netscape <tbray@textuality.com>, Jean Paoli, Microsoft <jeanpa@microsoft.com> , C. M. Sperberg-McQueen, W3C <cmsmcq@w3.org> , Eve Maler, Sun Microsystems, Inc. <eve.maler@east.sun.com> , François Yergeau , W3C, “Extensible Markup Language (XML) 1.0: Fifth edition”, W3C Recommendation 26 November 2008.
12. Benoit Marchal (bmarchal@pineapplesoft.com), Consultant, Pineapple Software, “Working XML: Processing instructions and parameters”
    http://www.ibm.com/developerworks/xml/library/x-wxxm2/index.html
13. Tim Berners-Lee, 1998 “Principles of Design”
    http://www.w3.org/DesignIssues/Principles.html
14. Apple Inc Website
    http://developer.apple.com/internet/webcontent/validation.html