**ABSTRACT**

When you sign up for a new website account, do you create a complex, non-guessable password? Or do you just use "One Password" for everything? Each simple-minded password you use creates an opportunity for some malefactor to take over your account by a lucky guess or a brute-force attack. And if you use the same password everywhere, one hack or lucky guess means that all of your accounts are compromised. SafeHouse is a typical password manager that is used as a web-application.

When you create a new secure account or update a weak password, you don't want to strain your brain trying to come up with something strong and unique. Why bother? You don't have to remember it. SafeHouse includes a built-in password generator that generates an automatic password of up to 12 characters.

SafeHouse provides you with the functionalities of adding a new account for which you want to save the id and password, retrieving a saved account’s id and password, deleting a saved id and password and updating an account’s password.

All these functionalities are guarded by the Master Password that the user creates and is saved only in the user’s mind. The master password never flows on the network and thus if the user happens to forget the master password he needs to contact the SafeHouse team to destroy his master account by sending an email. The encryption technique that we have used is Advanced Encryption Standard (AES) and the message digest algorithm used is Secure Hash Algorithm 3 (SHA-3) which shows that the application is highly secure.

SafeHouse performs many complex security functions in the background, but by design the user need not be aware of these processes. For most users, SafeHouse is a simple, easy, and effective way to securely store and share their passwords. More technically sophisticated users can more fully appreciate the lengths to which SafeHouse goes to protect their confidential data; but both groups benefit fully from the high levels of security and convenience that SafeHouse delivers.

**Chapter – 1**

**Introduction**

A password manager is a software application that helps a user store and organize passwords. Password managers usually store passwords encrypted, requiring the user to create a master password; a single, ideally very strong password which grants the user access to their entire password database. Some password managers store passwords on the user's computer, whereas others store data in the cloud. While the core functionality of a password manager is to securely store large collections of passwords, many provide additional features such as form filling and password generation.

**Types of Password Managers**

Password managers come in six often-combined flavours:

* Desktop - desktop/laptop software storing passwords on a computer hard drive.
* Portable - portable software storing passwords and program on a mobile device, such as a [PDA](https://en.wikipedia.org/wiki/PDA), [smart phone](https://en.wikipedia.org/wiki/Smart_phone), or as a [portable application](https://en.wikipedia.org/wiki/Portable_application) on a USB memory stick.
* Token - credentials are protected using a [security token](https://en.wikipedia.org/wiki/Security_token), thus typically offering [multi-factor authentication](https://en.wikipedia.org/wiki/Multi-factor_authentication) by combining "something the user has" ([smart card](https://en.wikipedia.org/wiki/Smart_card) or[USB stick](https://en.wikipedia.org/wiki/USB_stick)) , "something the user knows" (PIN or password) and/or "something the user is" ([biometrics](https://en.wikipedia.org/wiki/Biometrics) - such as a fingerprint, hand, retina, or face scanner).
* Web-based - Online password manager where passwords are viewed and copied to/from a provider's website.
* Cloud-based - Online password manager where credentials are stored on a service provider's servers on the Internet, but handled by password management software running on the client's machine.
* Stateless - Passwords are generated on the fly from a master passphrase and a tag using a [key derivation function](https://en.wikipedia.org/wiki/Key_derivation_function).

Password managers can protect against [keyloggers](https://en.wikipedia.org/wiki/Keyloggers) or [keystroke logging](https://en.wikipedia.org/wiki/Keystroke_logging) malware. When using a multi-factor authentication password manager that automatically fills in logon fields, the user does not have to type any user names or passwords for the keylogger to pick up. While a keylogger may pick up the PIN to authenticate into the smart card token, for example, without the smart card itself (something the user has) the PIN does the user no good. An online password manager is a website that securely stores login details. They are a web-based version of more conventional desktop-based password manager.

The advantages of online password managers over desktop-based versions are portability (they can generally be used on any computer with a [web browser](https://en.wikipedia.org/wiki/Web_browser) and a network connection, without having to install software), and a reduced risk of losing passwords through theft from or damage to a single PC - also the same risk is present for the server that is used to store the users passwords on. In both cases this risk can be prevented by ensuring secure [backups](https://en.wikipedia.org/wiki/Backup) are taken.

**Chapter - 2**

**Study Of Contents**

**2.1 Database**

A database is an organized collection of data. The data is typically organized to model relevant aspects of reality (for example, the availability of rooms in hotels), in a way that supports processes requiring this information (for example, finding a hotel with vacancies).

Database management systems (DBMSs) are specially designed applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose database management system (DBMS) is a [software](http://en.wikipedia.org/wiki/Computer_software) system designed to allow the definition, creation, querying, update, and administration of databases.

Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information. Databases are set up, so that one set of software programs provides all users with access to all the data. Databases use a table format, that is made up of rows and columns. Each piece of information is entered into a row, which then creates a record. Once the records are created in the database, they can be organized and operated in a variety of ways that are limited mainly by the software being used. Databases are somewhat similar to spreadsheets, but databases are more demanding than spreadsheets because of their ability to manipulate the data that is stored. It is possible to do a number of functions with a database that would be more difficult to do with a spreadsheet. The word data is normally defined as facts from which information can be derived. A database may contain millions of such facts. From these facts the database management system (DBMS) can develop information.

A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional [information technology](http://en.wikipedia.org/wiki/Information_technology), the term database is sometimes used casually to refer to any collection of data (perhaps a [spreadsheet](http://en.wikipedia.org/wiki/Spreadsheet), maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.

The interactions catered for by most existing DBMS fall into three main groups:

* Data definition. Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
* Update. Inserting, modifying, and deleting data.
* Retrieval. Obtaining information either for end-user queries and reports or for processing by applications.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

There are many models for database management system

**2.1.1 Relational database model**

Relational model is most widely used data model for commercial data-processing. The reason it’s used so much is, because it’s simple and easy to maintain. The model is based on a collection of tables. Users of the database can create tables, insert new tables or modify existing tables. There are several languages for database programming. SQL, Oracle, etc. The relational model gives us a single way to represent data: as a two-dimensional table called a relation.

The data in RDBMS is stored in database objects called tables. The table is a collection of related data entries and it consists of columns and rows. A field is a column in a table that is designed to maintain specific information about every record in the table. A record, also called a row of data, is each individual entry that exists in a table. A column is a vertical entity in a table that contains all information associated with a specific field in a table.

**2.1.2 MySQL**

A fast, reliable, easy-to-use, multi-user multi-threaded relational database system

* It is freely available and released under GPL (GNU General Public License).
* MySQL server can handle very large databases.
* Offers rich and very useful set of functions.
* Connectivity, speed and security make MySQL very suited for accessing database on a network.
* A lot of contributed software availableMySQL Server works in client/server or embedded systems.

**MYSQL ARCHITECTURE**

Three layer model:

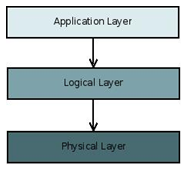


Figure 2.1 MYSQL Architecture

* The application layer contains common network services for connection handling, authentication and security. This layer is where different clients interact with MySQL these clients can written in different API's:.NET, Java, C, C++, PHP, Python, Ruby, Tcl, Eiffel, etc...
* The Logical Layer is where the MySQL intelligence resides, it includes functionality for query parsing, analysis, caching and all built-in functions (math, date...). This layer also provides functionality common across the storage engines.
* The Physical Layer is responsible for storing and retrieving all data stored in “MySQL”. Associated with this layer are the storage engines, which MySQL interacts with very basic standard API's. Each storage engine has it strengths and weakness, some of this engines are MyISAM, InnoDB, CSV, NDB Cluster, Falcon, etc...

SQL is structured Query Language which is a computer language for storing, manipulating and retrieving data stored in relational database.

SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server uses SQL as standard database language.

**SQL**

SQL is followed by unique set of rules and guidelines called Syntax:

* SQL SELECT STATEMENT :

SELECT column1, column2....columnN

FROM table\_name

WHERE CONDITION;

* SQL CREATE TABLE Statement:

CREATE TABLE table\_name( TUTORIALS POINT Simply Easy Learning

column1datatype,

column2datatype,

column3datatype,

.....

columnNdatatype,

PRIMARY KEY( one or more columns )

);

* SQL DROP TABLE Statement:

DROP TABLE table\_name;

* SQL ALTER TABLE Statement:

ALTER TABLE table\_name {ADD|DROP|MODIFY} column\_name {data\_ype};

* SQLALTER TABLE Statement (Rename):

ALTER TABLE table\_name RENAME TO new\_table\_name;

* SQL INSERT INTO Statement:

INSERT INTO table\_name( column1, column2....columnN)

VALUES ( value1, value2....valueN);

* SQL UPDATE Statement:

UPDATE table\_name TUTORIALS POINT Simply Easy Learning

SET column1 = value1, column2 = value2....columnN=valueN

[ WHERE CONDITION ];

* SQL DELETE Statement:

DELETE FROM table\_name

WHERE {CONDITION};

* SQL CREATE DATABASE Statement:

CREATE DATABASE database\_name;

* SQL DROP DATABASE Statement:

DROP DATABASE database\_name;

**2.2 PHP**

PHP : Hypertext Pre-processor is a widely used, general-purpose [scripting language](http://en.wikipedia.org/wiki/Scripting_language) that was originally designed for [web development](http://en.wikipedia.org/wiki/Web_development) to produce [dynamic web pages](http://en.wikipedia.org/wiki/Dynamic_web_page). For this purpose, PHP code is embedded into the [HTML](http://en.wikipedia.org/wiki/HTML) source document and interpreted by a [web server](http://en.wikipedia.org/wiki/Web_server) with a PHP processor module, which generates the [web page](http://en.wikipedia.org/wiki/Web_page) document. It may also function as a graphical application. PHP is available as a processor for most modern web servers and as standalone interpreter on most [operating systems](http://en.wikipedia.org/wiki/Operating_systems) and [computing platforms](http://en.wikipedia.org/wiki/Platform_(computing))

**2.2.1 Characteristics of PHP**

Five important characteristics make PHP's practical nature possible:

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

**2.2.2 Common uses of PHP:**

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
* You add, delete, modify elements within your database thru PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

**2.2.3 PHP Environment Setup**

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

* Web Server - PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here: http://httpd.apache.org/download.cgi
* Database - PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
* PHP Parser - In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

**2.2.4 PHP Variables**

All variables in PHP are denoted with a leading dollar sign ($).

* The value of a variable is the value of its most recent assignment.
* Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
* Variables can, but do not need, to be declared before assignment.
* Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
* Variables used before they are assigned have default values.
* PHP does a good job of automatically converting types from one to another when necessary.
* PHP variables are Perl-like.

**2.2.5 PHP Global Variables - Superglobals**

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file.

Some of the PHP superglobal variables are:

* $\_POST: PHP $\_POST is widely used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.
* $\_GET: PHP $\_GET can also be used to collect form data after submitting an HTML form with method="get". $\_GET can also collect data sent in the URL.
* $\_SESSION: $\_SESSION gives session variable values.

**2.2.6 PHP Constants**

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.

**2.2.7 PHP Decision Making**

The if, elseif ...else and switch statements are used to take decision based on the different condition. You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements:

* if...else statement - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
* elseif statement - is used with the if...else statement to execute a set of code if one of several condition are true
* switch statement - is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

**2.2.8 PHP Loop Types**

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

* for - loops through a block of code a specified number of times.
* while - loops through a block of code if and as long as a specified condition is true.
* do...while - loops through a block of code once, and then repeats the loop as long as a special condition is truer.
* foreach - loops through a block of code for each element in an array.

**2.2.9 PHP Arrays**

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables it’s easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

* Numeric array - An array with a numeric index. Values are stored and accessed in linear fashion
* Associative array - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
* Multidimensional array - An array containing one or more arrays and values are accessed using multiple indices.

**2.2.10 PHP Strings**

They are sequences of characters, like "PHP supports string operations".

For ex.

|  |  |
| --- | --- |
| $string\_1 = "This is a string in double quotes"; |  |
| To concatenate two string variables together, use the dot (.) operator: |
| **2.2.11 PHP File Inclusion**  You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.   * The include() Function * The require() Function   This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.  **The include() Function:**  The include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.  **The require() Function:**  The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the require() function generates a fatal error and halt the execution of the script.  So there is no difference in require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed. |

**2.2.12 Database Connectivity:**

To connect PHP to MySQL database you need to know following important things:

* Host name (localhost)
* MySQL user name.
* MySQL password.

The default MySQL user is "root" while there is no default MySQL password.

**PHP MySQLi functions:** PHP MySQLi = PHP MySQL Improved!

* **Opening connection with database:** The mysqli\_connect() function opens a new connection to the MySQL server and returns an object representing the connection.

**mysqli\_connect(host,username,password,dbname,port,socket);**

* + host: Optional. Specifies a host name or an IP address
  + username: Optional. Specifies the MySQL username
  + password: Optional. Specifies the MySQL password
  + dbname: Optional. Specifies the default database to be used
  + port: Optional. Specifies the port number to connect to the MySQL server
  + socket: Optional. Specifies the socket or named pipe to be used.
* **Executing queries:** The mysqli\_query() function performs a query against the database. For successful SELECT, SHOW, DESCRIBE, or EXPLAIN queries it will return a mysqli\_result object. For other successful queries it will return TRUE, FALSE on failure.

**mysqli\_query(connection,query,resultmode);**

* + connection: Required. Specifies the MySQL connection to use
  + query: Required. Specifies the query string
  + resultmode: Optional. A constant. Either:
    - MYSQLI\_USE\_RESULT (Use when retrieving large amount of data).
    - MYSQLI\_STORE\_RESULT (This is default).
* **Retrieving values from result set:** The mysqli\_fetch\_assoc() function fetches a result row as an associative array with database column names as key values.

**mysqli\_fetch\_assoc(result);**

* + result: Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result().

**2.2.13 PHP Sessions**

A normal HTML website will not pass data from one page to another. In other words, all information is forgotten when a new page is loaded. This makes it quite a problem for tasks like a shopping cart, which requires data(the user's selected product) to be remembered from one page to the next.

**PHP sessions - overview**

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping cart items, etc). However, this session information is temporary and is usually deleted very quickly after the user has left the website that uses sessions.

It is important to ponder if the sessions' temporary storage is applicable to your website. If you require a more permanent storage you will need to find another solution, like a MySQL database. Sessions work by creating a unique identification (UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same webpage.

**Starting a PHP session**

Before you can begin storing user information in your PHP session, you must first start the session. When you start a session, it must be at the very beginning of your code, before any HTML or text is sent.

Below is a simple script that you should place at the beginning of your PHP code to start up a PHP session.

**PHP Code:**

<?php

session\_start(); // start up your PHP session!

?>

This tiny piece of code will register the user's session with the server, allow you to start saving user information and assign a UID (unique identification number) for that user's session.

**Storing a session variable**

When you want to store user data in a session use the $\_SESSION [associative array](http://www.tizag.com/phpT/arrays.php). This is where you both store and retrieve session data. In previous versions of PHP there were other ways to perform this store operation, but it has been updated and this is the correct way to do it.

**PHP Code:**

<?php

session\_start();

$\_SESSION['views'] = 1; // store session data

echo "Pageviews = ". $\_SESSION['views']; //retrieve data

?>

**PHP sessions: using PHP's isset function**

Now that you are able to store and retrieve data from the $\_SESSION array, we can explore some of the real functionality of sessions. When you create a variable and store it in a session, you probably want to use it in the future. However, before you use a session variable it is necessary that you check to see if it exists already!

This is where PHP's isset function comes in handy. isset is a function that takes any variable you want to use and checks to see if it has been set. That is, it has already been assigned a value.

With our previous example, we can create a very simple page view counter by using isset to check if the page view variable has already been created. If it has we can increment our counter. If it doesn't exist we can create a page view counter and set it to one. Here is the code to get this job done:

**PHP Code:**

<?php

session\_start();

if(isset($\_SESSION['views']))

$\_SESSION['views'] = $\_SESSION['views']+ 1;

else

$\_SESSION['views'] = 1;

echo "views = ". $\_SESSION['views'];

?>

The first time you run this script on a freshly opened browser the if statement will fail because no session variable views would have been stored yet. However, if you were to refresh the page the if statement would be true and the counter would increment by one. Each time you reran this script you would see an increase in view by one.

**Cleaning and destroying your session:**

Although a session's data is temporary and does not require that you explicitly clean after yourself, you may wish to delete some data for your various tasks.

Imagine that you were running an online business and a user used your website to buy your goods. The user has just completed a transaction on your website and you now want to remove everything from their shopping cart.

**PHP Code:**

<?php

session\_start();

if(isset($\_SESSION['cart']))

unset($\_SESSION['cart']);

?>

You can also completely destroy the session entirely by calling the session\_destroy function.

**PHP Code:**

<?php

session\_start();

session\_destroy();

?>

**2.2.14 PHP Form Handling**

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

**Form Validation**

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

# PHP $\_GET Variable

The predefined $\_GET variable is used to collect values in a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

<form action="welcome.php" method="get">

A file can use the $\_GET variable to collect form data (the names of the form fields will automatically be the keys in the $\_GET array):

Welcome <?php echo $\_GET["fname"]; ?>

**PHP $\_POST Function**

The predefined $\_POST variable is used to collect values from a form sent with method="post".

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

<form action="welcome.php" method="post">

A file can use the $\_POST variable to collect form data (the names of the form fields will automatically be the keys in the $\_POST array):

Welcome <?php echo $\_POST["fname"]; ?>

**2.4 HTML**

**2.4.1 Hyper Text Markup Language**

Hypertext Markup Language (HTML) is a language for describing how pages of text, graphics, and other Information are organized, formatted, and linked together. It is not really a programming language in the sense of COBOL or Visual Basic, but it does provide powerful capabilities for text formatting and output display. The original purpose of HTML was primarily as a tool for making text documents readily available on the Internet. Basic graphic support was added, and the things started to take off. Now we have sound, live video, retail catalogs, and much, much more available to us.[

HTML pages are the standard interface to the Internet. This basic language provides the necessary nuts and bolts for building Web pages.

Using a markup language means that tags can be added to the words in the document and web enable them. A tag is a set of descriptive formatting codes used in HTML document that instructs a web browser how to display text and graphics on a web page.

<html>

<head>

<title>My First Webpage</title></head>

<body>

This is my first homepage.<b>This text is bold</b>

</body>

</html>

What you just made is a skeleton html document. This is the minimum required information for a web document and all web documents should contain these basic components. The first tag in your html document is <html>. This tag tells your browser that this is the start of an html document. The last tag in your document is </html>. This tag tells your browser that this is the end of the html document.

The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window.

The text between the <title> tags is the title of your document. The <title> tag is used to uniquely identify each document and is also displayed in the title bar of the browser window.

The text between the <body> tags is the text that will be displayed in your browser.

The text between the <b> and </b> tags will be displayed in a bold font.

**2.4.2 Logical vs. Physical Tags**

In HTML there are both logical tags and physical tags. Logical tags are designed to describe (to the browser) the enclosed text's meaning. An example of a logical tag is the <strong></strong> tag. By placing text in between these tags you are telling the browser that the text has some greater importance. By default all browsers make the text appear bold when in between the <strong> and</strong> tags.

Physical tags on the other hand provide specific instructions on how to display the text they enclose. Examples of physical tags include:

<b>: Makes the text bold.

<big>: Makes the text usually one size bigger than what's around it. <i>: Makes text italic.

Physical tags were invented to add style to HTML pages because style sheets were not around, though the original intention of HTML was to not have physical tags. Rather than use physical tags to style your HTML pages, you should use style sheets.

**2.5 CSS**

**C**ascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML

**2.5.1 CSS Syntax – Selectors**

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

**Selector:** A selector is an HTML tag at which style will be applied. This could be any tag like<h1> or <table> etc.

**Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color or border etc.

**Value:** Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows:

selector { property: value }

**The Type Selectors:**

This is the same selector we have seen above. Again one more example to give a color to all level 1 headings:

h1 {

color: #36CFFF;

}

**The Universal Selectors:**

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type:

\* {

color: #36CFFF;

}

**The Descendant Selectors:**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <em> element only when it lies inside <ul> tag.

ul em {

color: #36CFFF;

}

**The Class Selectors:**

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

.black {

color: #000000;

}

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

h1.black {

color: #000000; }

**The ID Selectors:**

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

#black {

color: #000000;

}

This rule renders the content in black for every element with id attribute set to *black* in our document. You can make it a bit more particular. For example:

h1#black {

color: #000000;

}

This rule renders the content in black for only <h1> elements with id attribute set to black.

**Embedded CSS - The <style> Element:**

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside <head>...</head> tags. Rules defined using this syntax will be applied to all the elements available in the document. Here is the generic syntax:

<head>

<style type="text/css" media="..."> Style Rules

............

</style>

</head>

**Inline CSS - The style Attribute:**

You can use *style* attribute of any HTML element to define style rules. These rules will be applied to that element only. Here is the generic syntax:

<element style="...style rules....">

**External CSS - The <link> Element:**

The <link> element can be used to include an external style sheet file in your HTML document.

An external style sheet is a separate text file with **.css** extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

Here is the generic syntax of including external CSS file:

<head>

<link type="text/css" href="..." media="..." />

</head>

**Attributes:**

Attributes associated with <style> elements are:

* type - text/css: Specifies the stylesheet language as a content-type (MIME type). This attribute is required.
* href – URL: Specifies the stylesheet file having style rules type). This attribute is required

**Imported CSS - @import Rule:**

@import is used to import an external style sheet in a manner similar to the <link> element. Here is the generic syntax of @import rule.

<head>

<@import "URL";

</head>

**2.6 INTRODUCTION TO XAMPP**

**2.6.1 XAMPP**

XAMPP is a [free and open source](http://en.wikipedia.org/wiki/Free_software) [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) [web server](http://en.wikipedia.org/wiki/Web_server) [solution stack](http://en.wikipedia.org/wiki/Solution_stack) package, consisting mainly of the [Apache HTTP Server](http://en.wikipedia.org/wiki/Apache_HTTP_Server), [MySQL](http://en.wikipedia.org/wiki/MySQL) [database](http://en.wikipedia.org/wiki/Database), and [interpreters](http://en.wikipedia.org/wiki/Interpreter_(computing)) for scripts written in the [PHP](http://en.wikipedia.org/wiki/PHP) and [Perl](http://en.wikipedia.org/wiki/Perl) [programming languages](http://en.wikipedia.org/wiki/Programming_language).

XAMPP's name is an [acronym](http://en.wikipedia.org/wiki/Acronym_and_initialism) for:

* X (to be read as "cross", meaning [cross-platform](http://en.wikipedia.org/wiki/Cross-platform))
* [Apache HTTP Server](http://en.wikipedia.org/wiki/Apache_HTTP_Server)
* [MySQL](http://en.wikipedia.org/wiki/MySQL)
* [PHP](http://en.wikipedia.org/wiki/PHP)
* [Perl](http://en.wikipedia.org/wiki/Perl)

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. In practice, however, XAMPP is sometimes used to actually serve web pages on the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web). XAMPP also provides support for creating and manipulating databases in MySQL and SQLite among others.

**2.6.2 MySQL**

**2.6.2.1 Introduction**

MySQL is an open source database management system. It is the world's second most widely used [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) and most widely used open-source RDBMS.

**2.6.2.2 MySQL Architecture**

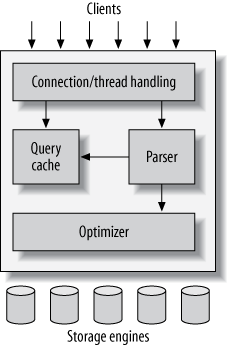


Figure 2.2: MySQL server architecture

* The topmost layer contains the services that aren’t unique to MySQL. They’re services most network-based client/server tools or servers need: connection handling, authentication, security, and so forth.
* The second layer has much of MySQL’s brains, including the code for query parsing, analysis, optimization, caching, and all the built-in functions (e.g., dates, times, math, and encryption). Any functionality provided across storage engines lives at this level: stored procedures, triggers, and views, for example.
* The third layer contains the storage engines. They are responsible for storing and retrieving all data stored “in” MySQL. Like the various file systems available for GNU/Linux, each storage engine has its own benefits and drawbacks. The server communicates with them through the *storage engine API*. This interface hides differences between storage engines and makes them largely transparent at the query layer. The API contains a couple of dozen low-level functions that perform operations such as “begin a transaction” or “fetch the row that has this primary key.” The storage engines don’t parse SQL or communicate with each other; they simply respond to requests from the server.

**2.7 jQuery**

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto − Write less, do more.

jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery −

* **DOM manipulation −** The jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
* **Event handling −** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
* **AJAX Support −** The jQuery helps we a lot to develop a responsive and feature-rich site using AJAX technology.
* **Animations −** The jQuery comes with plenty of built-in animation effects which we can use in our websites.
* **Lightweight −** The jQuery is very lightweight library - about 19KB in size ( Minified and gzipped ).
* **Cross Browser Support −** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
* **Latest Technology −** The jQuery supports CSS3 selectors and basic XPath syntax.

**2.7.1 Advantages of jQuery:**

* **Cross Browser Compatibility:** Cross browser compatibility is a major problem in AJAX/JavaScript programming. For example, a website may work perfectly in opera but may not run in Mozilla 7. jQuery developers were aware of all this browser issues and they also knew the reason behind these inconsistencies. Hence, they have documented these solutions in the library by fixing all these problems to make sure whatever code we write, web pages look same across all browsers.
* **Lightweight and Fast:** To keep the jQuery library lightweight, a lot of functions have been omitted and some are transferred to the plug-in section. If we want any particular feature on a page, we can add these plug-in in the website. This keeps the coding at a limited level and saves bandwidth for faster loading. The jQuery core application.
* **Open source library:**  it’s an open source library, free and well supported across all types of applications. Therefore, anyone can use jQuery in their application without any compatibility and licensing issues. Also, Microsoft has integrated jQuery officially into its IDE Visual Studio 2010 and jQuery intellisense is well supported in Visual Studio 2010 now.
* **Easy to Learn and Flexible:** jQuery is very compact and easy to understand. Fact is, even if we have basic knowledge of coding, we can start writing jQuery very easily after going through a short tutorial.
* **Lots of extendable and reusable Plug-ins:** The jQuery team has kept the plug-in creation process simple. They have given a framework for extending the library. We can develop our own customized plug-in, use it in jQuery projects, and share it with fellow developers. In the case we don’t want to create plug-ins, there are hundreds of high quality plug-ins available for download to do our work.
* **CSS3 Selectors Compliant:**jQuery completely supports the CSS3 selector requirements. We can learn and start using CSS3 selectors in our production code right away.
* **DOM traversal manipulation:**  Dom Traversal Manipulation is made very simple and easy with jQuery. DOM manipulation like adding new elements, removing elements, traversing library is only 24 kb in size. That makes it very easy to add in the applications and hence, improves the performance of the web across elements etc with querying and chaining is extremely wonderful & robust.
* **Cool Effects and animations:**  very cool animations like fade-in/fade-out expand/contract etc can be achieved very easily on any element with couple of lines of code.
* **Utility Features:**jQuery offers utility functions that help coding string trimming, iteration, array manipulation and much more. These functions offer seamless integration between jQuery and JavaScript. It makes the code writing process easier and hassle-free.
* **jQuery User Interface:** jQuery UI offers accordions, sliders, dialog boxes, date pickers, and many more features. All of them are ready to use. The UI controls are configurable and we don’t need to spend any extra time in building these features. However, for .Net developers, all these utilities are available in Ajax too.
* **The Whole World is Embracing jQuery:** Various IT behemoths on web have already accepted jQuery. Companies such as IBM, Microsoft and Google are using jQuery. Microsoft even included jQuery with its MVC framework and is working with the open-source jQuery project to add new features to the jQuery library.

**2.8 JavaScript**

JavaScript was released by Netscape and Sun Microsystems in 1995.

JavaScript:

• Is a programming language.

• Is an interpreted language.

• Is object-based programming.

• Is widely used and supported

• Is accessible to the beginner.

**2.8.1 Uses of JavaScript**

• **Use it to add multimedia elements:** With JavaScript we can show, hide, change, resize images, and create image rollovers. We can create scrolling text across the status bar.

• **Create pages dynamically:** Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.

• **Interact with the user:** It can do some processing of forms and can validate user input when the user submits the form.

**2.8.2 About JavaScript**

JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser. Here are some tips to remember when writing JavaScript commands.

• JavaScript code is case sensitive.

• White space between words and tabs are ignored.

• Line breaks are ignored except within a statement.

• JavaScript statements end with a semicolon(;).

**The SCRIPT Tag**

The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML.

<SCRIPT language = "JavaScript">

statements

</SCRIPT>

**Chapter – 3**

**Software Requirement Specification**

**3.1 Software Development Life Cycle:**

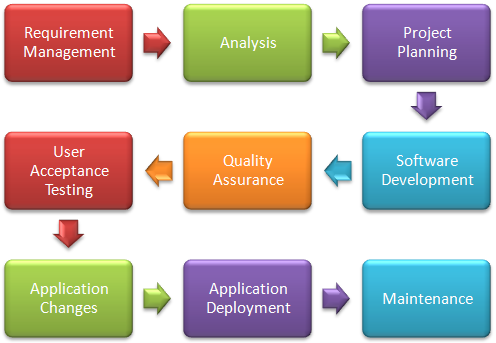
****

Figure 3.1: Software Development Life Cycle

The **Software Development Life Cycle or Systems Development Life Cycle (SDLC)** in software engineering is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system: the software development process.

A SDLC adheres to important phases that are essential for developers, such as planning, analysis, design, and implementation, and are explained in the section below. A number of SDLC models have been created: waterfall, fountain, spiral, build and fix, rapid prototyping, incremental, and synchronize and stabilize. The oldest of these, and the best known, is the waterfall model: a sequence of stages in which the output of each stage becomes the input for the next. These stages can be characterized and divided up in different ways, including the following:

* **Preliminary analysis**: The objective of phase1 is to conduct a preliminary analysis, propose alternative solutions, describe costs and benefits and submit a preliminary plan with recommendations. Conduct the preliminary analysis: in this step, you need to find out the organization's objectives and the nature and scope of the problem under study. Even if a problem refers only to a small segment of the organization itself then you need find out what the objectives of the organization itself are. Then you need to see how the problem being studied fits in with them. Propose alternative solutions: In digging into the organization's objectives and specific problems, you may have already covered some solutions. Alternate proposals may come from interviewing employees, clients, suppliers, and/or consultants. You can also study what competitors are doing. With this data, you will have three choices: leave the system as is, improve it, or develop a new system. Describe the costs and benefits.
* **Systems analysis, requirements definition**: The goal of system analysis is to determine where the problem is in an attempt to fix the system. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined. Defines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.
* **Systems design**: In systems design the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems. The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input design. Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.
* **Development**: The real code is written here. The developer team works on the software project to implement the ideas logically and practically.
* **Integration and testing**: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability. The code is tested at various levels in software testing. Unit, system and user acceptance testing’s are often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much, if any iteration occurs. Iteration is not generally part of the waterfall model, but usually some occur at this stage. In the testing the whole system is test one by one. Following are the types of testing:
* Defect testing
* Path testing
* Data set testing
* Unit testing
* System testing
* Integration testing
* Black-box testing
* White-box testing
* Regression testing
* Automation testing
* User acceptance testing
* Performance testing
* **Acceptance, installation, deployment**: The final stage of initial development, where the software is put into production and runs actual business.
* **Maintenance**: What happens during the rest of the software's life: changes, correction, additions, moves to a different computing platform and more. This is often the longest of the stages.
* **Disposition:** The tenth phase occurs when the system is disposed of and the task performed is either eliminated or transferred to other systems.

Not every project will require that the phases be sequentially executed. However, the phases are interdependent. Depending upon the size and complexity of the project, phases may be combined or may overlap.

**3.2 Development Environment**

**3.2.1 Hardware and Software Requirements**

The program doesn’t need any special hardware or software to run it can run on minimum hardware requirements.

**Hardware requirement**

Main Processor Pentium Core i2

Hard-disk Capacity 8 GB

RAM 1 GB

Clock Speed 2.40GHz

Keyboard Standard Keyboard

Monitor Resolution greater than 1024X768

**Software Requirement**

Operating System Window XP or above

Version PHP 5.0

**Technologies Used**:

Front End : HTML, CSS, jQuery, Java Script

Server Side Scripting Language : PHP

RDBMS(Back end) : MySQL

**3.2.2 Programming Interface**

The interface is designed in HTML and has used CSS for designing the interfaces. The interfaces are cross browser compatible which means the interfaces do not change with the change in browser. The backend coding is implemented in PHP and jQuery. The database model that we have used is MySQL as it provides the simple way of creating, maintaining and handling database. The forms which our project is generating on the behalf of users also takes html as front end and PHP and MySQL as back end. The database is maintained in phpmyadmin.

**Chapter – 4**

**System Model**

**4.1 Functional requirements**

Functional requirements are those that refer to the functionality of the system, i.e. what services it will provide to the users. A **functional requirement** defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>". The plan for implementing functional requirements is detailed in the system design.

Functional requirements specify particular results of a system. The plan for implementing non-functional requirements is detailed in the system architecture. Functional requirements pertain to other information needed to produce the correct system and are detailed separately. The client has to simply requests to the server. The server will authenticate the client and accordingly will process the request. The functional requirements of our project are that user should be able to successfully create, maintain and handle the database. The interface should be very easy to use. The user may not have the sound knowledge about the database functionality and form handling.

**4.2 Non functional requirements**

A **non-functional requirement** is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

Non-functional requirements are often called **qualities** of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioural requirements". Informally these are sometimes called the "ileitis", from attributes like stability and portability. Qualities, that is non-functional requirements, can be divided into two main categories:

* Execution qualities, such as security and usability, which are observable at run time.
* Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

**4.3 Use Cases**

This section lists use cases or scenarios from the use-case model which depict significant, central functionality of the final system, or if they have a large architectural coverage.

**Use case for signup:-**

* First of all user will sign-up by supplying his details: user id and password.
* These details will be sent to server for authentication.
* After that, account will be successfully created.

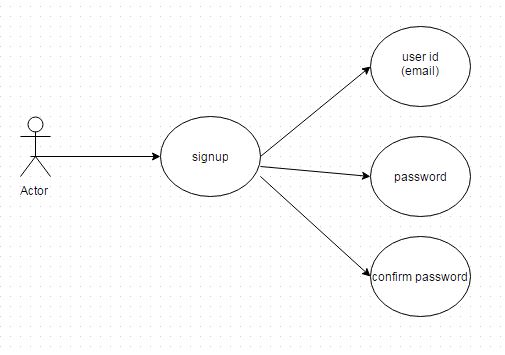


Figure 4.1: Use Case for Signup

**Use case for Login:**

* User will login by supplying details like unique email id and master password.
* Server will execute the request and will send the appropriate message.

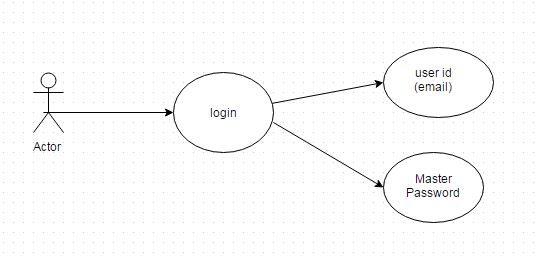


Figure 4.2: Use Case For Login

**Use case for User Page:**

* After the user logs in by supplying the correct user id and password, he is directed to the user page.
* He is given options like home, password generator, security, change password, logout from SafeHouse, contact us.

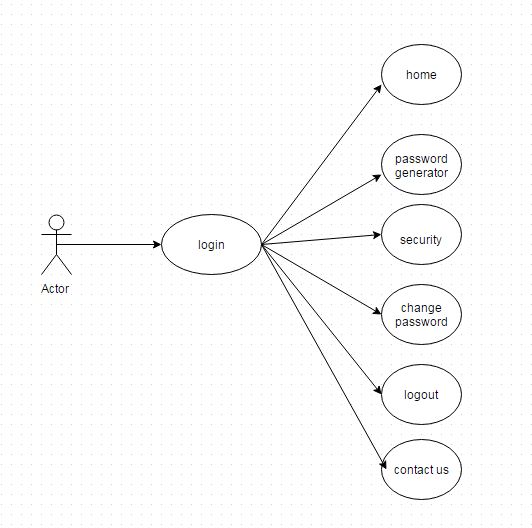


Figure 4.3: Use Case For User Page

**Use case for Home Page:**

* Home page has the options like retrieve password, add password, update password, delete password and login from safehouse.

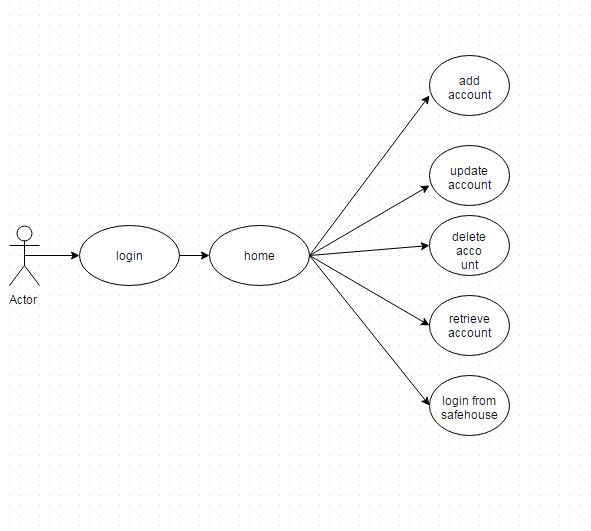


Figure 4.4: Use Case For Home

**4.4 Data Flow Diagram (DFD)**

The DFD was first developed by Larry Constantine as a way of expressing system requirements system requirements and depicting the information flow.A DFD also known as a ‘BUBBLE CHART’ has the purpose of clarifying system requirements and identifying major transformation that will become programs in system design.

**DFD symbols:** In the DFD there are four symbols as described below:

1. A square defines a source or destination of the system data:

I/O

2. An arrow identifies data flow-data in motion:

It is a pipeline through which information flows.

3. A square with round edges:

It represents a process.

**Level 0 DFD**

* This data flow diagram represents the flow of data at the time of sign up

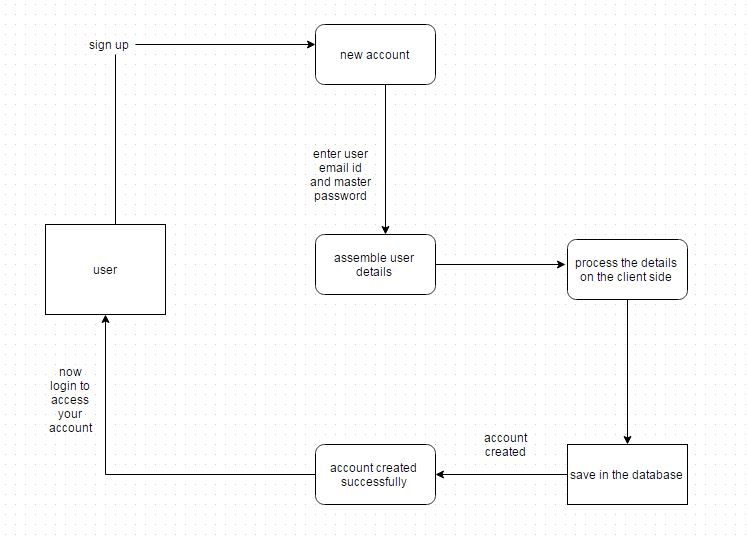


Figure 4.5 Level 0 DFD for sign up

* This data flow diagram represents the flow of data at the time of login

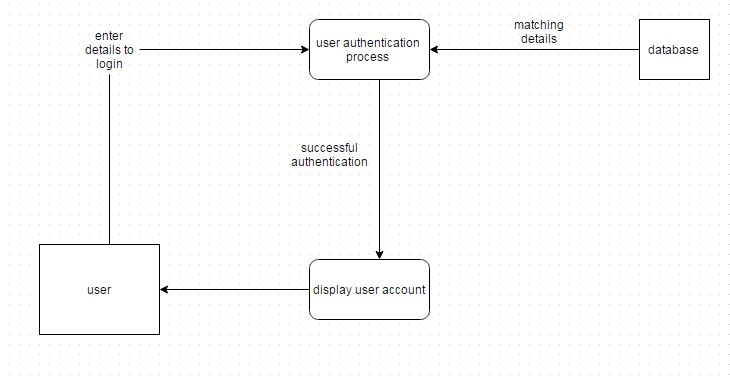


Figure 4.6 Level 0 DFD for login

**Level 1 DFD**

* Adding an account

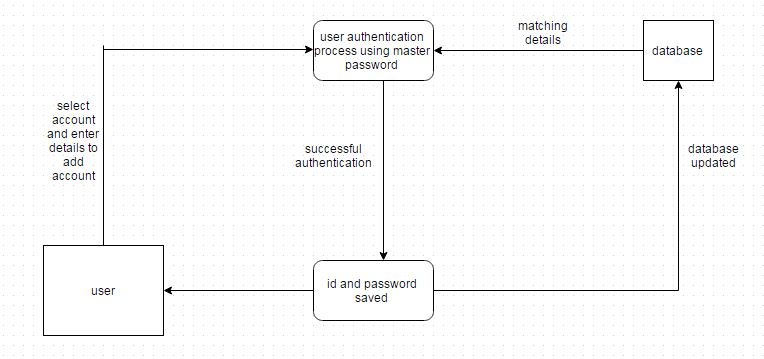


Figure 4.7 Level 1 DFD for adding an account

* Retrieving an account

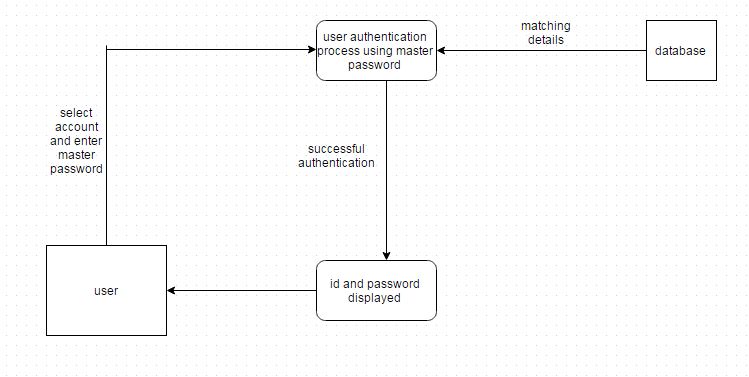


Figure 4.8 Level 1 DFD for retrieving an account

* Updating an account

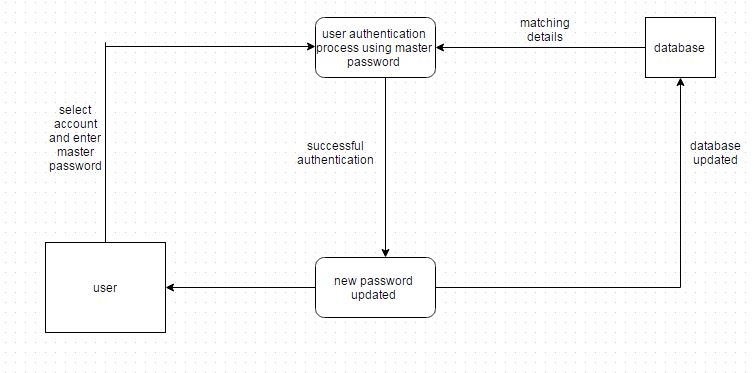


Figure 4.9 Level 1 DFD for updating an account

* Deleting an account

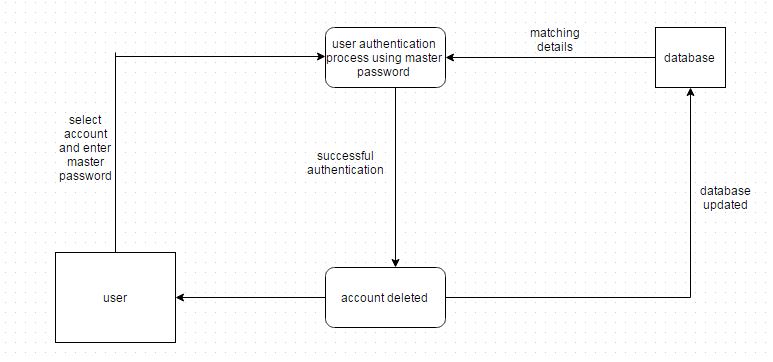


Figure 4.101 Level 1 DFD for deleting an account

**Chapter -5**

**Project Description**

**5.1 Screenshots and Descriptions**

**5.1.1 SafeHouse Home page-**

**5.1.1.1 Signup**

This is the main home page where the user first interacts with the SafeHouse application. The user is supposed to first sign up to access the application. At the time of sign up the use is asked to provide the Email id (which acts as the username), master password and confirm master password. This Master Password is the only password that the user needs to remember after becoming an authenticated user of SafeHouse.

At the backend, first the user id is sent to the server where using it a customer id (cid) is generated which is the primary key for the customer table. This customer id is sent to the client side where it gets encrypted using the message digest of the master password provided by the user. This encrypted customer id is known as the authentication id (aid). Now, this authentication id is saved at the server in the customer database.

All these functionalities are guarded by the Master Password that the user creates and is saved only in the user’s mind. The master password never flows on the network and thus if the user happens to forget the master password he needs to contact the SafeHouse team to destroy his master account by sending an email. The encryption technique that we have used is Advanced Encryption Standard (AES) and the message digest algorithm used is Secure Hash Algorithm 3 (SHA 3) which shows that the application is highly secure and encrypted.



Figure 5.1: Sign Up page

**5.1.1.2 Login Page**

After sign up the user needs to Login to access his SafeHouse account. The user needs to provide the username and master password here. Message digest (MD) of the master password is calculated. The client requests the server for the cid and aid of the given username. The requested aid is then decrypted using the MD. This decrypted text is matched with the cid and if both the values match, the user is authentic and is directed to his SafeHouse account.



Figure 5.2: LoginPage

**5.1.2 User Page**

The user is provided with the options such as Home, Password Generator, Security, Change Password, Logout form SafeHouse and Contact us.



Figure 5.3: User Page

On the home page you have options like Retrieve, Add, Update and Delete password and also an option for Login from SafeHouse.

**5.1.2.1 Add Password**

Select the account which you want to add. Enter the details such as account id, account password and the master password. Here again the authentication process will take place using the master password as was done at the time of login. The MD of the master password will encrypt the account password and save it on the server database.

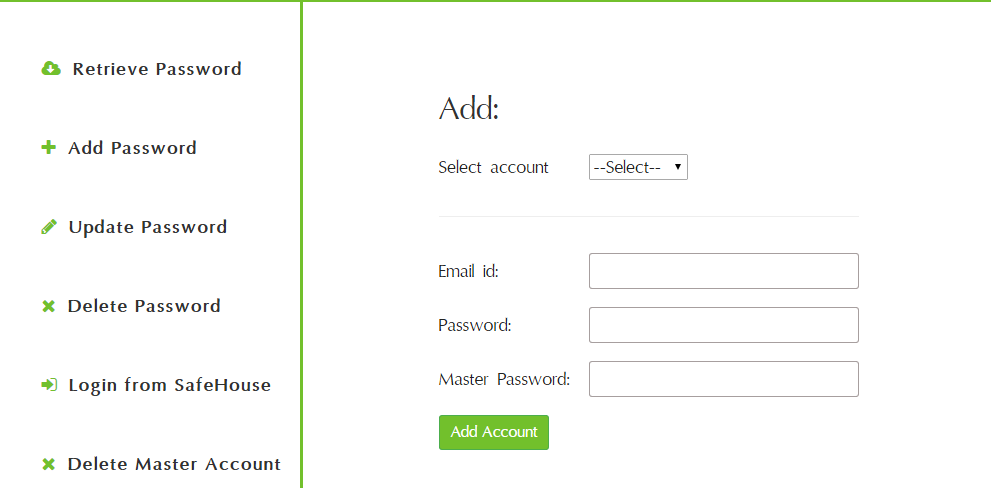


Figure 5.4: Add Password Page

**5.1.2.2 Retrieve Password**

Select the account for which you want to retrieve the id and password. Enter the master password. Here again the authentication process will take place using the master password as was done at the time of login. If the authentication is successful, the server sends the required id and password in encrypted form. The MD of the master password will decrypt these details and display them to the server.

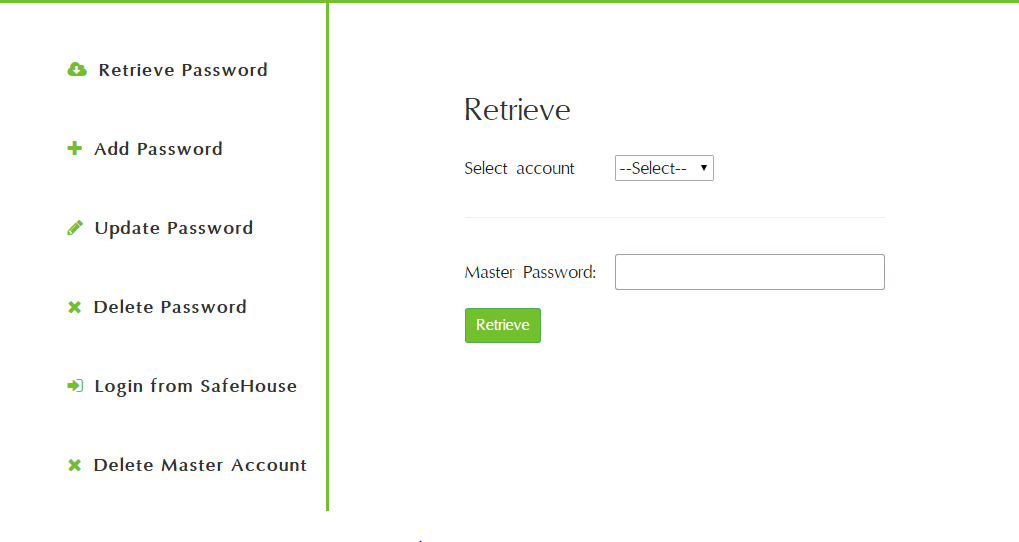


Figure 5.5: Retrieve Password Page

**5.1.2.3 Update Password**

Select the account for which you want to update the password. Enter the master password. Here again the authentication process will take place using the master password as was done at the time of login. If the authentication is successful, the server will encrypt the new password with the MD of the master password and replace the old password with the new password.

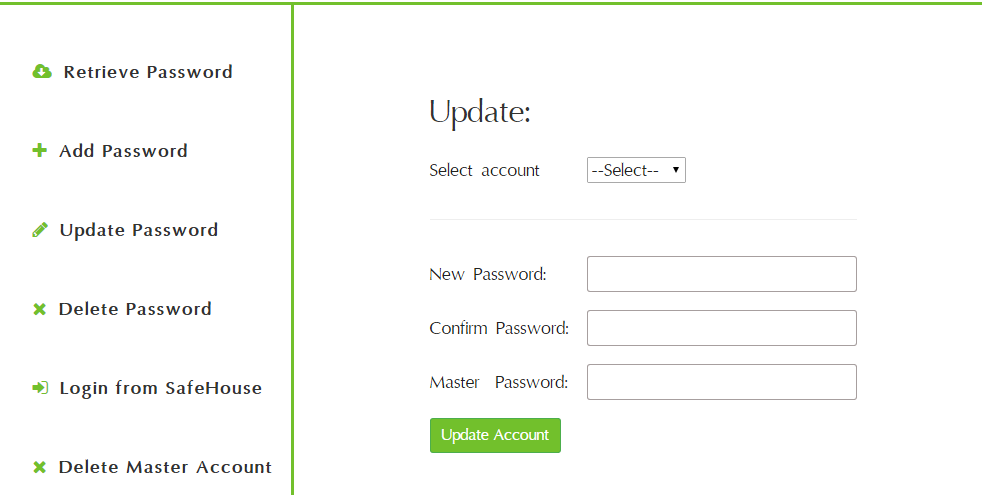


Figure 5.6: Update Password Page

**5.1.2.4 Delete Password**

Select the account which you want to delete. Enter the master password. Here again the authentication process will take place using the master password as was done at the time of login. If the authentication is successful, the server delete the record for that account.

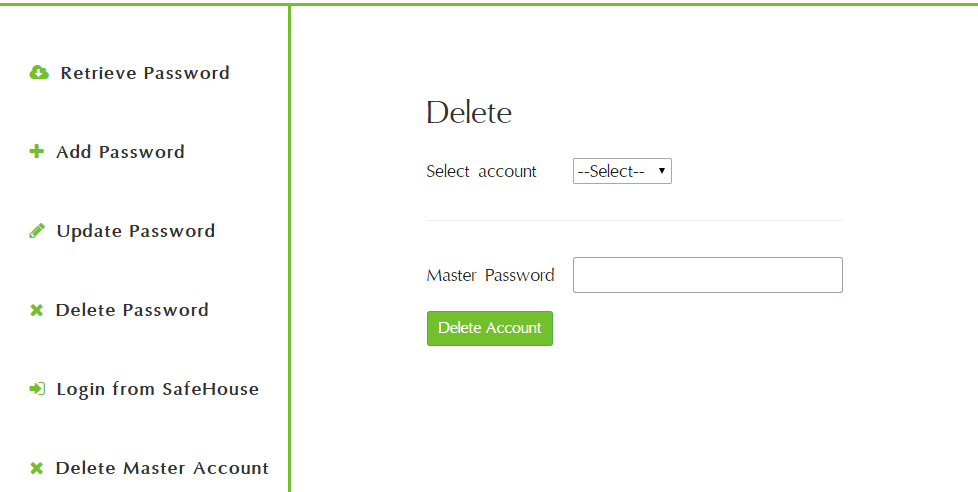


Figure 5.7: Delete Password Page

**5.1.2.5 Login from SafeHouse**

SafeHouse provides you with the option of logging in to your various accounts without manually opening the respective website.

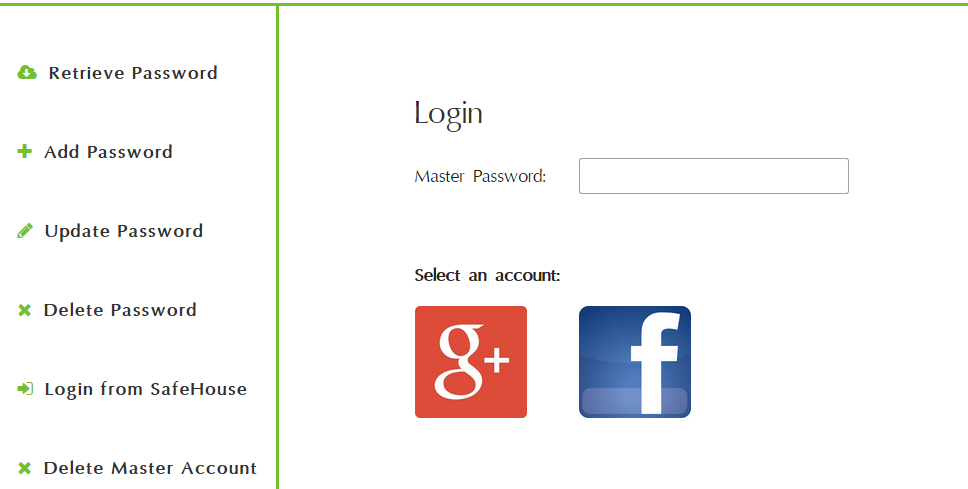


Figure 5.8: Login From SafeHouse Page

**5.1.3 Password Generator**

Password generator generates passwords for the user so the he does not have to bother about a strong, confidential and non-guessable password for the various accounts that he adds. It provides the user with the options of customized length passwords. Passwords are generated using an intelligent combination of numbers, alphabets and special characters.



Figure 5.9: Password Generator Pop-up Page

**5.1.4 Security**

**5.1.4.1 Crypto JS**

CryptoJS is a growing collection of standard and secure cryptographic algorithms implemented in JavaScript using best practices and patterns. They are fast, and they have a consistent and simple interface.

Hsashers are functions that take an input (no matter how large) and maps it to a fixed size, smaller one (the hash, or checksum). You can’t convert a hash back to the original input, yet you can check if the original data has been corrupted comparing the hashes. The hashing algorithms used by CryptoJS are:

* MD5
* SHA-1
* SHA-2
* SHA-3
* RIPEMD-160
* HMAC
* Progressive HMAC Hashing
* PBKDF2
* Ciphers
* The Cipher Algorithms
* AES
* DES, Triple DES
* Rabbit
* RC4, RC4Drop

CryptoJS contains two folders:

* components – with both minified and commented JS files.
* rollups – minified files (one for each algorithm) bundled with core code.

Components files have dependencies: you have to link at least core.js, while rollups are quite self contained.

**5.1.4.2 Message Digest Algorithm- SHA-3**

We have used five rounds of Secure Hash Algorithm SHA-3 for deriving the message digest of the Master password.

We have used SHA-3 for deriving the message digest because of the various features mentioned below

* It has a high security margin.
* It received good amount of high-quality analysis.
* It has excellent hardware performance.
* It has good overall performance.
* It is very different from SHA-2.
* It provides a lot of flexibility.
* Support for variable-length hashes
* Considering options: One capacity: c = 512, with output size encoding
* Two capacities: c = 256 and c = 512, with output size encoding, or
* Four capacities: c = 224, c = 256, c=384, and c = 512 without output size encoding (preferred by the Keccak team).
* Input format for SHA-3 hash function(s) will contain a padding scheme to support tree hashing in the future.
* NIST will standardize 224, 256, 384 and 512 alternative hashes to the 4 hash sizes of SHA-2.

**Keccak Features**

* Keccak supports the same hash-output sizes as SHA-2 (i.e., SHA-224, -256, -384, -512).
* Keccak works fine with existing applications, such as DRBGs, KDFs, HMAC and digital signatures.
* Keccak offers flexibility in performance/security tradeoffs.
* Keccak supports tree hashing.
* Keccak supports variable-length output.

**5.1.4.3 Advanced Encryption Standard (AES 256) – For encryption**

* Symmetric-key algorithm
* It supersedes DES
* Symmetric and parallel architecture
* Adapted to modern processor
* Suited to smart card
* AES has a fixed block size of 128 bits.
* Key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits.
* 128 bit plain text block, 256 bit key
* AES operates on a 4×4 column-major order matrix of bytes, termed the *state*

**High-level description of the algorithm**

* Key Expansion—round keys are derived from the cipher key using Rijndael's key schedule
* Initial Round
  + Add Round Key—each byte of the state is combined with the round key using bitwise XOR
* Rounds
  + Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
  + Shift Rows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
  + Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
* Add Round Key
* Final Round (no Mix Columns)
  + Sub Bytes
  + Shift Rows
  + Add Round Key

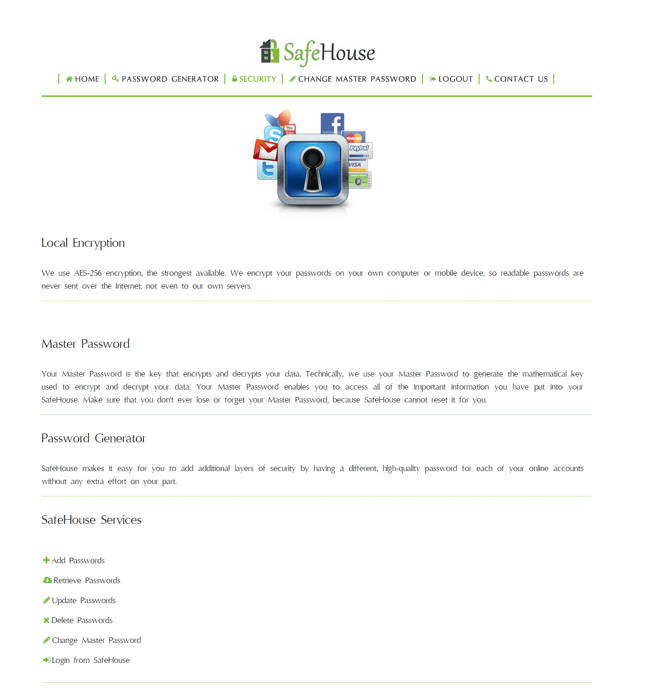


Figure 5.9: Security Page

**5.1.5 Change Master Password**

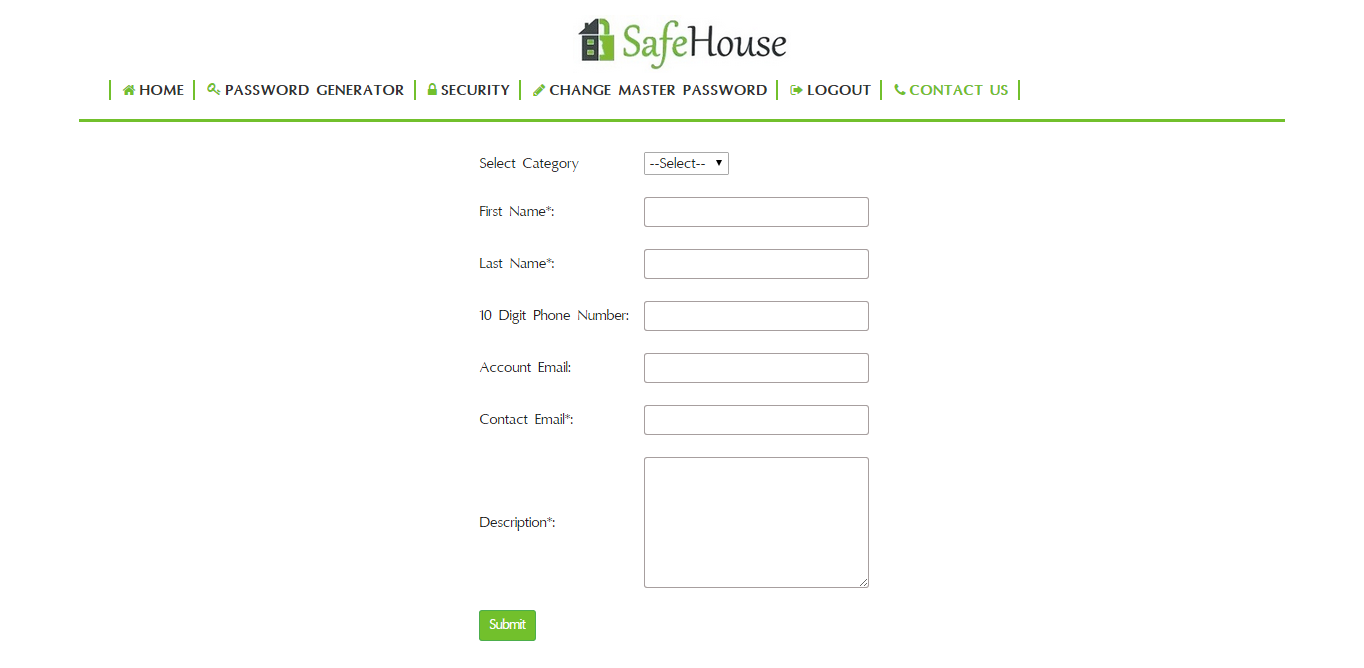
The Change Master Password option helps the user to change his master password as and when needed. This option allows the user to enter the old password, the new password and confirm the new password. If it is successful, the master password is changed and the user is directed to the login page again. He needs to login to access his account.



Figure 5.9: Change Password Page

**5.1.6 Contact Us**

This is the feedback and query where the user can either give his feedback on the SafeHouse application or can ask any query. The user is supposed to fill in the form and submit his request in order to send the feedback or query mail.

Figure 5.10: Contact Us Page

**5.2 Database**

**5.2.1 Database SafeHouse**

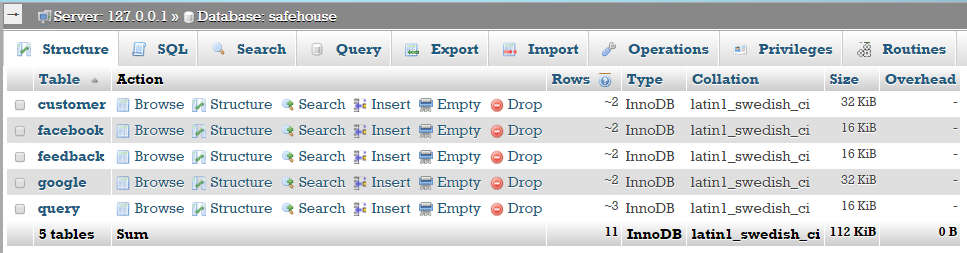


Table 5.1 Database safehouse tables

**5.2.2 Table: customer**

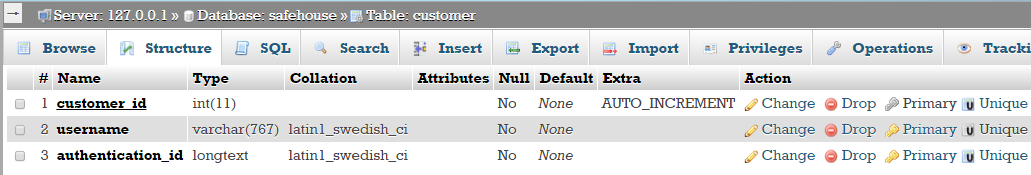


Table 5.2 customer

**5.2.3 Table: facebook**

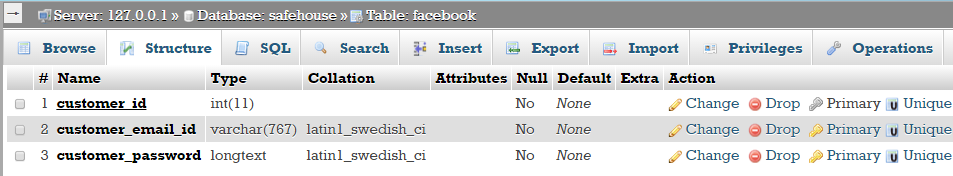


Table 5.3 Table facebook

Query for adding Foreign Key-

ALTER TABLE facebook ADD CONSTRAINT fk\_cust\_id\_facebook FOREIGN KEY (customer\_id) references customer(customer\_id) ON DELETE CASCADE

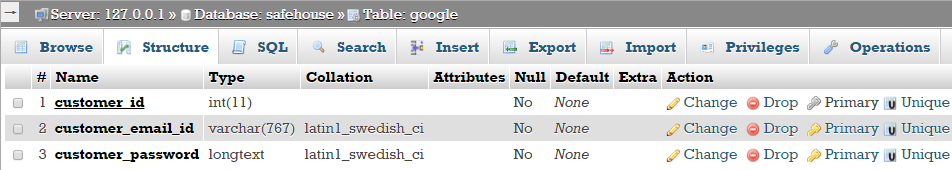
**5.2.3 Table: google**

Table 5.4 google

Query for adding Foreign Key-

ALTER TABLE google ADD CONSTRAINT fk\_cust\_id\_google FOREIGN KEY (customer\_id) references customer(customer\_id) ON DELETE CASCADE

**5.2.5 Table: query**

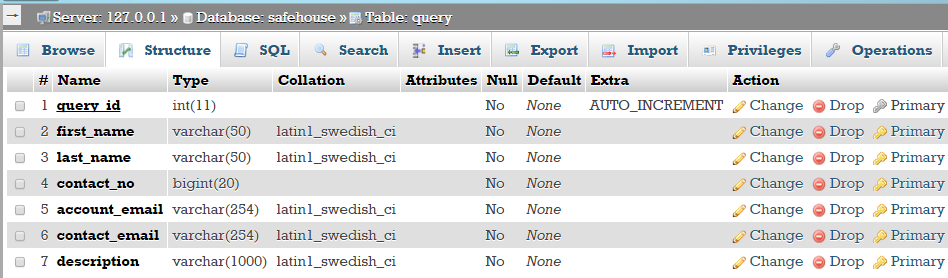


Table 5.5 query

**5.2.6 Table: feedback**

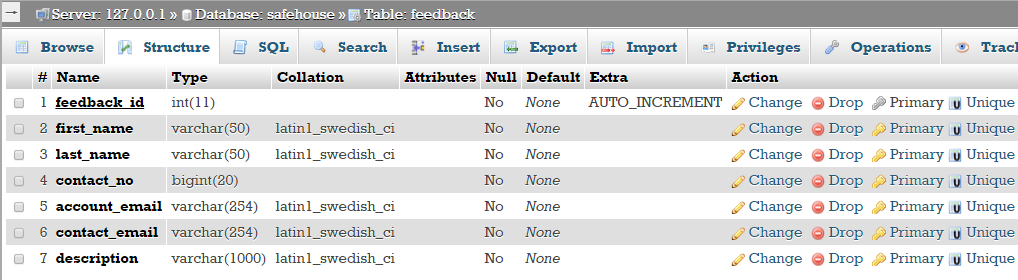


Table 5.6 feedback

**5.3 PHPMailer**

PHPMailer is a code library to send(transport) emails safely and easily via PHP codes from a webserver (MUA to the MSA server). Sending emails directly by PHP codes requires a high-level familiarity to SMTP standard protocol (rfc821, rfc2821 and rfc5321) and related issues(such as Carriage return) and vulnerabilities about Email injection for spamming. From 2001 up to 2013 PHPMailer is one of the popular solutions for these matters on PHP.

**Features:**

* Partial list of features:
* plain text, HTML and multipart batched files
* SSL, TLS
* SMTP, Qmail, POP3
* Debugging system
* php send\_mail and mail methods

**5.4 Captcha**

A CAPTCHA (an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used in computing to determine whether or not the user is human. A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot.

**Applications of CAPTCHAs**

CAPTCHAs have several applications for practical security, including (but not limited to):

**Preventing Comment Spam in Blogs.** Most bloggers are familiar with programs that submit bogus comments, usually for the purpose of raising search engine ranks of some website (e.g., "buy penny stocks here"). This is called comment spam. By using a CAPTCHA, only humans can enter comments on a blog. There is no need to make users sign up before they enter a comment, and no legitimate comments are ever lost!

**Protecting Website Registration**. Several companies (Yahoo!, Microsoft, etc.) offer free email services. Up until a few years ago, most of these services suffered from a specific type of attack: "bots" that would sign up for thousands of email accounts every minute. The solution to this problem was to use CAPTCHAs to ensure that only humans obtain free accounts. In general, free services should be protected with a CAPTCHA in order to prevent abuse by automated scripts.

**Protecting Email Addresses From Scrapers**. Spammers crawl the Web in search of email addresses posted in clear text. CAPTCHAs provide an effective mechanism to hide your email address from Web scrapers. The idea is to require users to solve a CAPTCHA before showing your email address. A free and secure implementation that uses CAPTCHAs to obfuscate an email address can be found at reCAPTCHA MailHide.

**Online Polls**. In November 1999, http://www.slashdot.org released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own program and the poll became a contest between voting "bots." MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll ensures that only humans can vote.

**Preventing Dictionary Attacks**. CAPTCHAs can also be used to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring it to solve a CAPTCHA after a certain number of unsuccessful logins. This is better than the classic approach of locking an account after a sequence of unsuccessful logins, since doing so allows an attacker to lock accounts at will.

**Search Engine Bots**. It is sometimes desirable to keep webpages unindexed to prevent others from finding them easily. There is an html tag to prevent search engine bots from reading web pages. The tag, however, doesn't guarantee that bots won't read a web page; it only serves to say "no bots, please." Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly guarantee that bots won't enter a web site, CAPTCHAs are needed.

**Worms and Spam**. CAPTCHAs also offer a plausible solution against email worms and spam: "I will only accept an email if I know there is a human behind the other computer." A few companies are already marketing this idea.

****

Figure 5.11: Captcha

**Chapter - 6**

**Conclusion**

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in PHP and MySQL, but also about all handling procedure related with “Web Based Password Manager”. It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

It also helped me study and implement various cryptographic methods and enhanced my knowledge about the same.

**Benefits of SafeHouse:**

* Data is encrypted and decrypted locally on your device
* Your key is never saved anywhere, neither on the client device, nor on the server
* Your key never flows over the network
* Highly secured algorithms such as SHA 3 has been used for deriving the message digest of the Master Password. Furthermore, five rounds of SHA 3 have been used to ensure maximum security.
* We use AES 256, the strongest encryption available.

**Chapter - 7**

**Future Scope**

This project is extensible in many ways. We can add many more features to it in future. We can enhance this system by adding functionalities like:

* The user can add multiple accounts to his SafeHouse for the same as well as various sites. Presently he is allowed to add only two accounts.
* One tap login can b added for various accounts.
* Biometrics can be used for user authentication
* It will also be able to create encrypted notes while keeping track of credit cards, passports, memberships and other sensitive personal data in your digital wallet. This information will always be available at a click.

**Chapter - 8**

**References**

* + - 1. Steven Suehring, Janet Valade, “PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies”, *ISBN 978-1-118-21370-4*, John Wiley & Sons, Inc., 2013
      2. [Kevin Tatroe](http://www.google.co.in/search?sa=G&tbm=bks&tbm=bks&q=inauthor:%22Kevin+Tatroe%22&ei=VMzCUYabL87yrQfu4YHgCQ&ved=0CDYQ9AgwAQ), [Peter MacIntyre](http://www.google.co.in/search?sa=G&tbm=bks&tbm=bks&q=inauthor:%22Peter+MacIntyre%22&ei=VMzCUYabL87yrQfu4YHgCQ&ved=0CDcQ9AgwAQ), [RasmusLerdorf](http://www.google.co.in/search?sa=G&tbm=bks&tbm=bks&q=inauthor:%22Rasmus+Lerdorf%22&ei=VMzCUYabL87yrQfu4YHgCQ&ved=0CDgQ9AgwAQ) , [Programming PHP](http://books.google.co.in/books?id=2S_Y6Zm02BQC&printsec=frontcover&dq=php&hl=en&sa=X&ei=VMzCUYabL87yrQfu4YHgCQ&ved=0CDQQ6AEwAQ), 2013
      3. Steve Suehring, “JavaScript Step by Step, Third Edition”, *ISBN 978-0-7356-6593-4*, O’Reilly Media Inc., 2013
      4. Atul Kahate, “Cryptography and Network Security”, *ISBN 0-07-049483-5*, Tata McGraw Hill.
      5. https://github.com/PHPMailer/PHPMailer as cited on 5th June, 2015
      6. http://www.captcha.net/ as cited on 1st June, 2015
      7. [https://msdn.microsoft.com/en-us/magazine/hh653584.aspx as cited on 3rd July 2014](https://msdn.microsoft.com/en-us/magazine/hh653584.aspx%20as%20cited%20on%203rd%20July%202014)
      8. <http://www.webdesign.org/the-relationship-between-html5-and-responsive-web-design.22342.html#ixzz3Y10cAhMX> as cited on 1st Nov 2014
      9. <https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity> as cited on 1st Oct 2014
      10. <http://www.w3schools.org> as cited on 1st April 2015
      11. <http://github.com/rmurphey/jqfundamentals> as cited on 15st Feb 2015
      12. <http://arstechnica.com/civis/viewtopic.php?t=1158314> as cited on 20th April 2015
      13. <https://www.boxcryptor.com/en/features> as cited on 25th May 2015
      14. <http://csrc.nist.gov/groups/ST/hash/sha-3/documents/dang_ietf86_march2013_presentation.pdf> as cited on 1st June 2015
      15. <http://www.drdobbs.com/security/keccak-the-new-sha-3-encryption-standard/240154037> as cited on 25th April 2015
      16. http://www.getbootstrap.com/ as cited on 25th Nov 2014
      17. https://jquery.com/ as cited on 2nd Oct 2014
      18. https://www.safaribooksonline.com/library/view/high-performance-mysql/ as cited on 5th Nov 2014