ABSTRACT

The objective of this Password Management System is to manage passwords for different accounts on the internet. The user shall be able to save all the username and passwords information of the accounts he holds on the internet using this application. These details shall be saved in the database in encrypted format. This will help the user to remember different usernames and passwords for accounts on the internet. The user shall be able to add account, edit and delete account using the system. The user has to login to the system in order to use this tool. It takes user login and password. It allows user to change the password for the tool being used.

This interface is developed using HTML, JavaScript, PHP and an md5 encryption system is managed by md5 algorithm. This Password Management System tool will allow the user to view all the different accounts he holds in a list view on the left screen. All the details related to the account like the user name, password etc shall be displayed on the right screen. The password is visible in encrypted format. The user can un hide the password and view the password details. The user can save any number of account information using this tool.

Features:

- Product and Component based
- Save and Add Password
- Update Password
- Save Password in more comprehensive way
- User Accounts to control the access and maintain security
- Simple Status & Resolutions
- Multi-level Priorities & Severities
- Targets & Milestones for guiding the programmers
- Attachments & Additional Comments for more information

 Robust database back-end

Table of Contents

Chapter 1

INTRODUCTION

1. Introduction

1.1 Purpose

This is simple Password Management System graphical user interface developed using HTML. It allows the user to add account information for different accounts he holds on the internet. Password Management System basically stores the user name and password details in an encrypted format in the database.

This Password Management System tool will allow the user to view all the different accounts he holds in a list view on the left screen. All the details related to the account like the user name, password etc. shall be displayed on the right screen. The password is visible in encrypted format. The user can un hide the password and view the password details. The user can save any number of account information using this tool.

.

1.2 Existing System

The Password Management System was done manually. Users had to remember the list of passwords for different accounts on the internet. The user had to manually maintain a list of all usernames and passwords. This task was very tedious. In cases where privacy for user accounts was required, it was difficult to manually maintain this list of passwords. The user had to save password details list in some word format so that it will help him to remember the username and passwords for different accounts on the internet.

1.3 Proposed System

- This is simple Password Management System graphical user interface developed using HTML. It allows the user to add account information for different accounts he holds on the internet.
- Password Management System basically stores the user name and password details in an encrypted format in the database. The encryption is managed by md5 algorithm.
- It allows the user to add account information for different accounts he holds on the internet. Password Management System basically stores the user name and password details in an encrypted format in the database
- The user has to login to the system. It checks for the validity of the user. The user shall be able to add account.
- The user can add user name and password details and save to the database. The list of accounts held by the user is displayed on the left screen.
- When the user selects the particular account on the left panel, the details of the account gets displayed on the right screen.
- The password shall be displayed in encrypted format. Once the user selects un hide option, he shall be able to view the password on screen.
- The user can add any number of accounts using this application.

Chapter 2

SOFTWARE AND HARDWARE REQUIREMENTS

2.1 Software Requirements

A set of programs associated with the operation of a computer is called software. Software is the part of the computer system which enables the user to interact with several physical hardware devices.

The minimum software requirement specifications for developing this project are as follows:

Designing frontend	:	HTML and CSS
Backend	:	My SQL
Scripting	:	Java Script
UML	:	Rational Rose
IDE	:	Visual Studio Code
Web Server	:	Apache HTTP Server

2.2 Hardware Requirement Specification

The Collection of internal electronic circuits and external physical devices used in building a computer is called Hardware.

The minimum hardware requirement specification for developing this project is as follows:

Processor	:	Pentium IV
RAM	:	512MB RAM
Hard Disk	:	10GB

Chapter 3

LITERATURE SURVEY

Password managers can be one way to mitigate the vulnerabilities that appear when multiple passwords are required. These software programs are able to store multiple passwords with the help of one master password that encrypts the whole password file.

These programs also have the ability to generate safe passwords which decreases the risk of password reuse (Silver, Jana, Boneh, Chen, & Jackson, Theoretical background 14 2014; Reichl, 2016). KeePass is an open source password manager, which is free of charge and works on the most common operating systems (Reichl, 2016). According to the security expert Bruce Schneier, security software should always be open source to be considered secure (Schneier, 1999).

This aspect is important but not relevant for this study. Another popular area of research regarding password managers is the usability aspect of the software. When testing the usability of different password managers it was found that users prefer an older portable solution that demands a software Theoretical background 15 to be run on the computer rather than a modern online-based password manager (Karole, Saxena, & Christin, 2010). This supports the use of KeePass in this study.

A ten-year-old study found that users acted reluctant in using the password management software because they were not comfortable with giving control of their password to a password manager, did not feel the need for it, and felt that the password manager did not provide greater security than before the implementation (Chiasson & Oorschot, 2006). A relevant question would be if this would be the result even ten years later.

Chapter 4

SOFTWARE REQUIREMENTS ANALYSIS

4.1 Overview

The main focus of the analysis phase of Software development is on “What needs to be done”. The objects discovered during the analysis can serve as the framework or Design. The class’s attributes, methods and association identified during analysis must be designed for implementation language. New classes must be introduced to store intermediate results during the program execution.

4.2 Problem Description

There are many challenges in securing passwords in this digital era. When the number of web services used by individuals are increasing year-over-year on one end, the number of cyber crimes is also skyrocketing on the other end. Here are a few common threats to protecting our passwords:

- **Login spoofing** - Passwords are illegally collected through a fake login page by cybercriminals.
- **Sniffing attack** - Passwords are stolen using illegal network access and with tools like key loggers.
- **Shoulder surfing attack** - Stealing passwords when someone types them, at times using a micro-camera and gaining access to user data.
- **Brute force attack** - Stealing passwords with the help of automated tools and gaining access to user data.
- **Data breach** - Stealing login credentials and other confidential data directly from the website database.

All of these threats create an opportunity for attackers to steal user passwords and enjoy unlimited access benefits. Let's take a look at how individuals and businesses typically manage their passwords.

Traditional methods of password management

- Writing down passwords on sticky notes, post-its, etc.
- Sharing them via spreadsheets, email, telephone, etc.
- Using simple and easy to guess passwords
- Reusing them for all web applications
- Often forgetting passwords and seeking the help of 'Forgot Password' option

While hackers are equipped with advanced tools and attacks, individuals and businesses still rely on traditional methods of password management. This clearly raises the need for the best password management practices to curb security threats.

4.3 Solution

- Use strong and unique passwords for all websites and applications
- Reset passwords at regular intervals
- Configure two-factor authentication for all accounts
- Securely share passwords with friends, family, and colleagues
- Store all enterprise passwords in one place and enforce secure password policies within the business environment
- Periodically review the violations and take necessary actions.

Chapter 5

SOFTWARE DESIGN

The main focus of the analysis phase of Software development is on "What needs to be done". The objects discovered during the analysis can serve as the framework or Design. The class's attributes, methods and association identified during analysis must be designed for implementation language. New classes must be introduced to store intermediate results during the program execution.

Emphasis shifts from the application domain of implementation and computer such as user interfaces or view layer and access layer. During analysis, we look at the physical entities or business objects in the system, that is, which players and how they cooperate to do the work of the application. These objects represent tangible elements of the business.

During the Design phase, we elevate the model into logical entities, some of which might relate more to the computer domain as people or employees. Here his goal is to design the classes that we need to implement the system the difference is that, at this level we focus on the view and access classes, such as how to maintain information or the best way o interact with a user or present information.

Design process:

During the design phase the classes identified in object-oriented analysis Must be revisited with a shift focus to their implementation. New classes or attribute and Methods must be an added for implementation purposes and user interfaces.

The following are some of the vies of software design life cycle. They are

- Data Flow Diagrams
- UML Diagrams
- Data Base Design

5.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system. It can also be used for the visualization of data processing (structured design).

There are two types of DFDs. They are:

- Context Level DFD
- Top Level DFD

5.1.1 Context Level DFD

In the Context Level the whole system is shown as a single process.

- No data stores are shown.
- Inputs to the overall system are shown together with data sources (as External entities).

Outputs from the overall system are shown together with their destinations (as External entities).

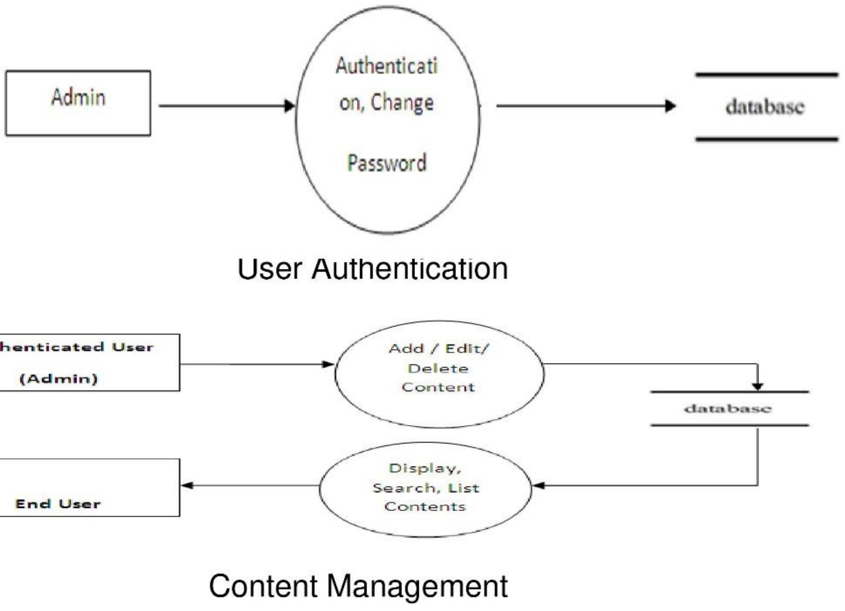

Admin
Authentication, Change Password
database
User Authentication
Authenticated User (Admin)
Add / Edit/ Delete Content
database
End User
Display, Search, List Contents
Content Management

Fig. 5.1 Context Level DFD

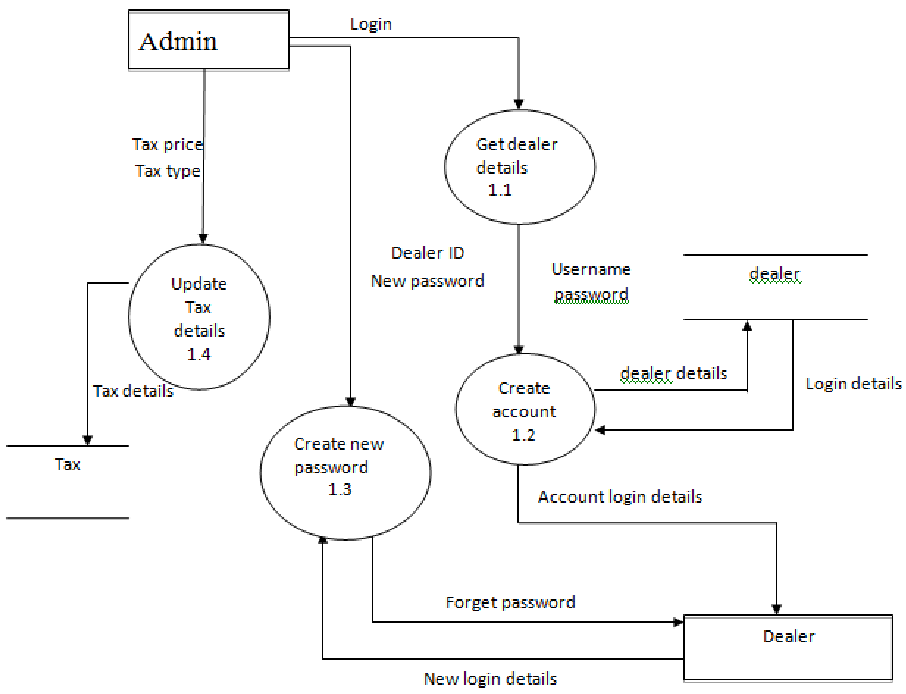


Fig. 5.2 Top Level DFD

5.1.2 Top Level DFD

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by ‘Zooming in’ on its contents and illustrates what it does in more detail.

5.2 UML Diagrams

Unified Modeling Language

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
This UML diagrams must include the following:

- Class diagram
- Interaction Diagram
- Use case Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram

Class Diagrams

The class diagram is the main building block in object oriented modeling. They are being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.

The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class, and
- The bottom part gives the methods or operations the class can take or undertake **An Activity Diagram** shows the flow from activity to activity.

An activity is an ongoing non- atomic execution within a state machine.

5.2.1Class Diagrams

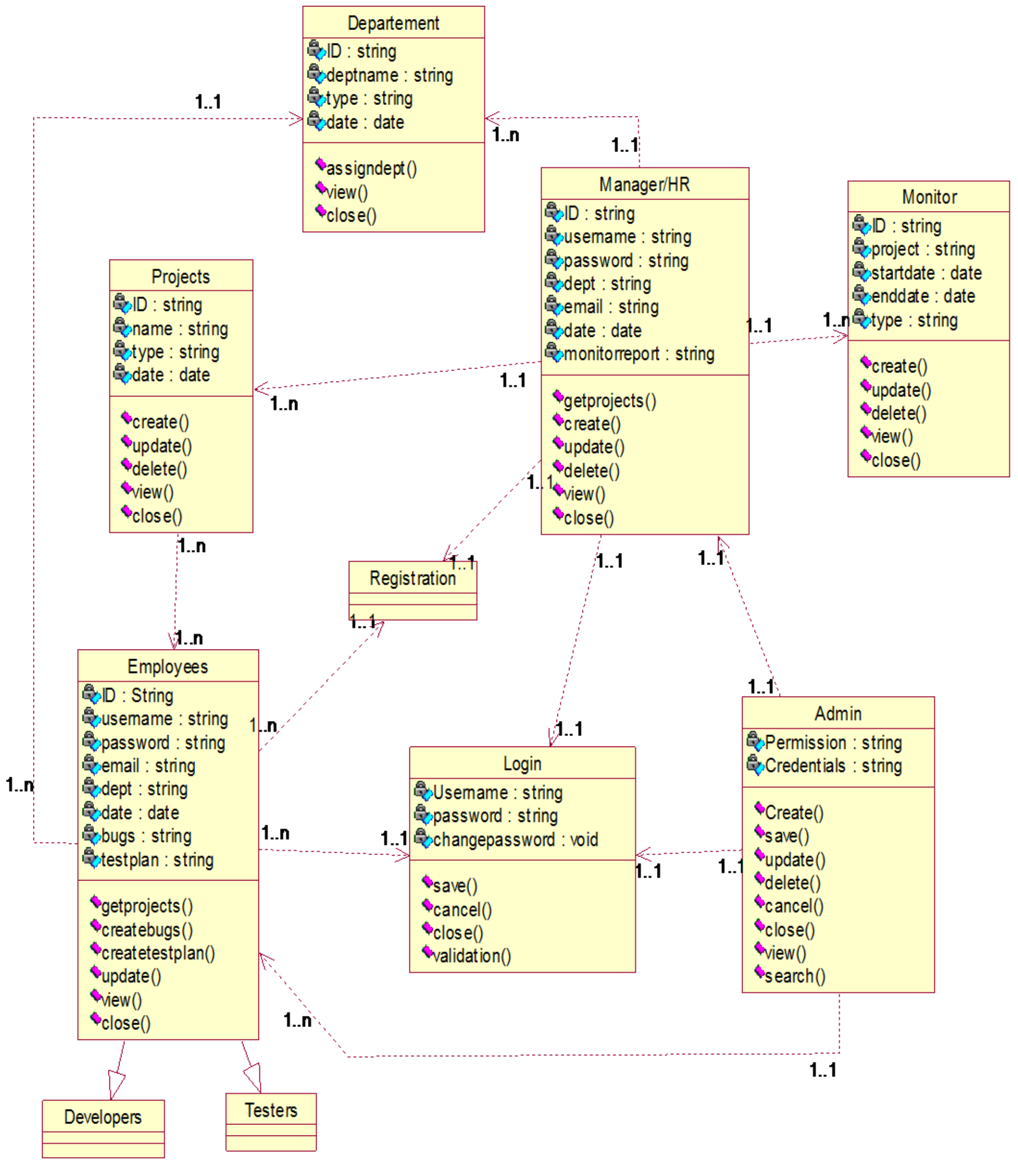


Fig 5.3 Class Diagram

5.2.2 Interaction Diagram

Interaction Diagrams

An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along xaxis and messages, ordered in increasing time, along the y-axis.

A Collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that’s bigger than the sum of all its parts.

5.2.2.1 Sequence Diagram

- An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.
- A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.
- Graphically, a sequence diagram is a table that shows objects arranged along xaxis and messages, ordered in increasing time, along the y-axis.

Contents

- Sequence diagrams commonly contain the following:
  - ➢ Objects
  - ➢ Links
  - ➢ Messages

Like all other diagrams, sequence diagrams may contain notes and constrains.

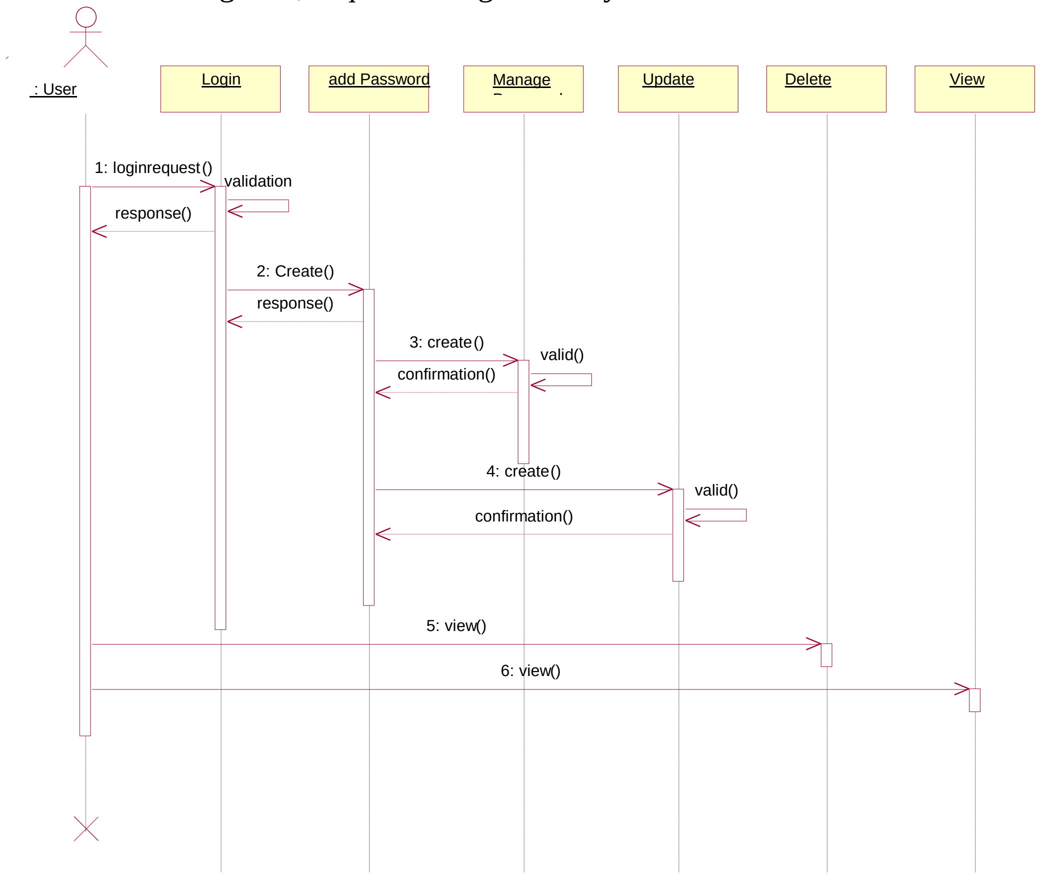


Fig 5.4 User Sequence Diagram

5.2.2.2 Collaborations Diagram

Collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all its parts.

Collaboration is also the specification of how an element, such as a classifier or an operation, is realized by a set of classifiers and associations playing specific roles used in a specific way

Contents

Collaboration diagrams commonly contain the following:

- Objects
- Links
- Messages

Like all other diagrams, sequence diagrams may contain notes and constrains.

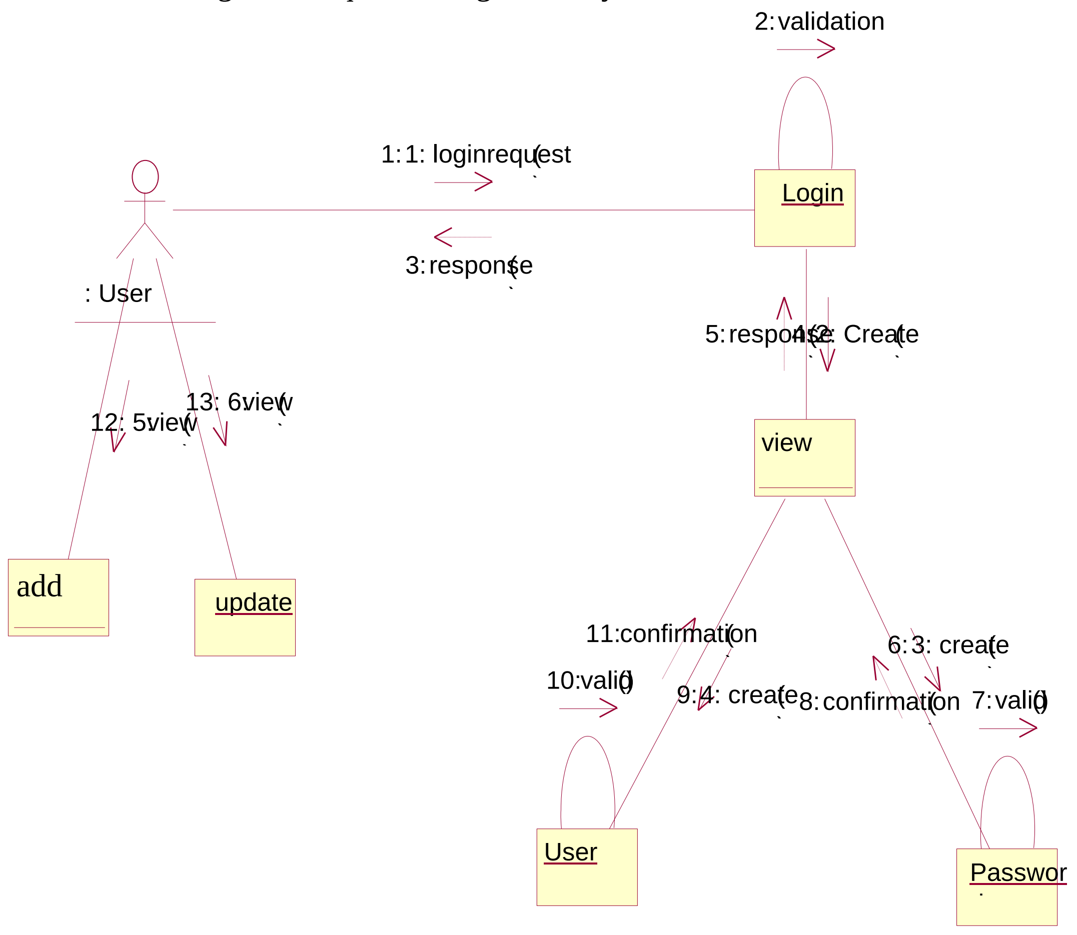


Fig 5.7 User Collaboration

5.2.3 Use case Diagram

A use case diagram is a diagram that shows a set of use cases and actors and relationships.

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system .Use cases focus on the behavior of the system from external point of view.

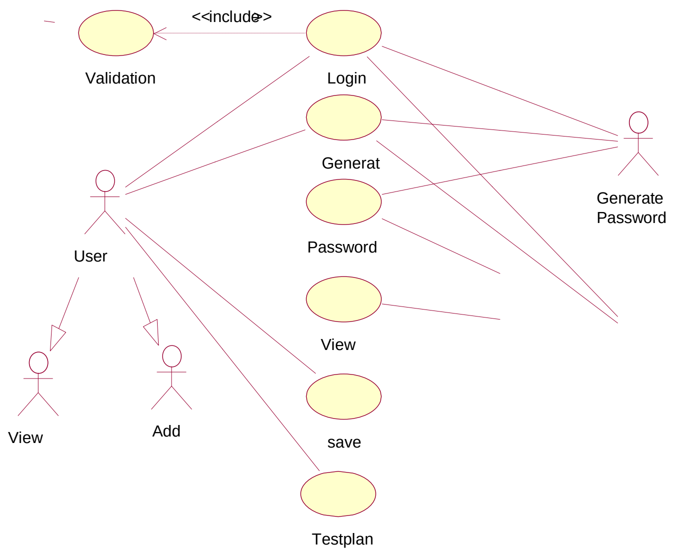


Fig 5.10 Overall Use case Diagram

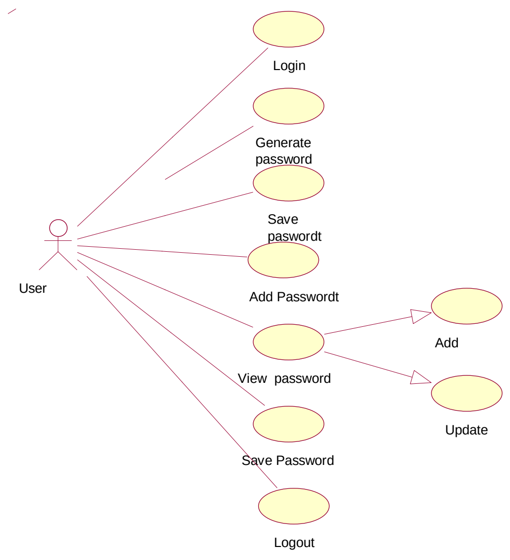


Fig 5.11 User Use case

5.2.4 Component Diagram

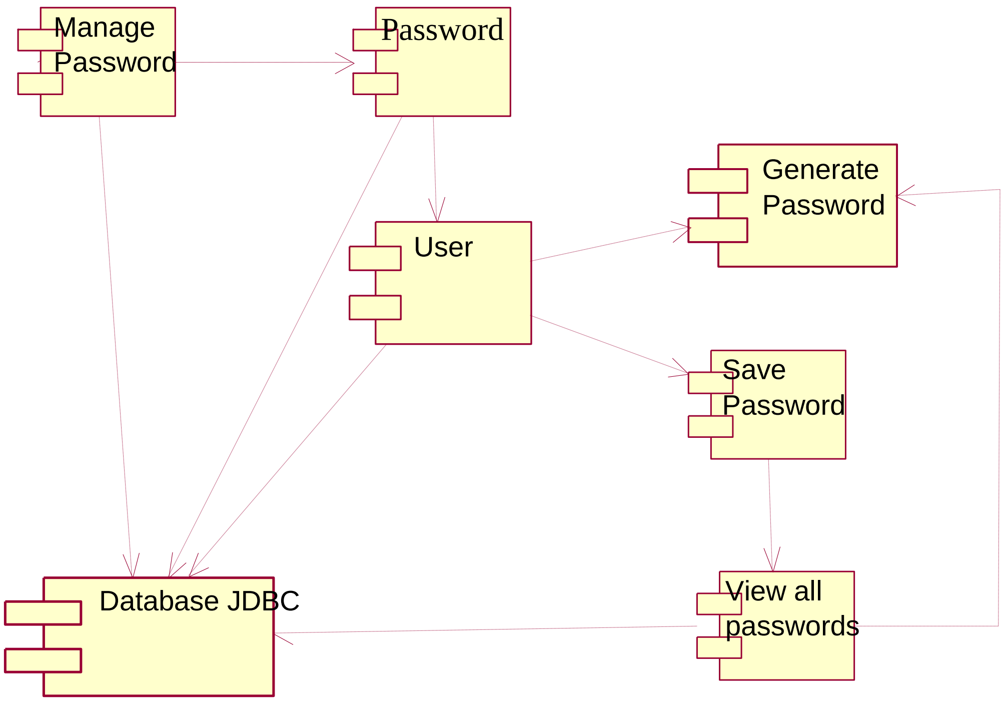


Fig 5.14 Component Diagram

5.2.5 Deployment Diagram

A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.

Graphically, a deployment diagram is collection of vertices and arcs.

Contents

- Deployment diagram commonly contain the following things:
- Nodes
- Dependency and association relationships
- Like all other diagrams, deployment diagrams may contain notes and constraints.
- Deployment diagrams may also contain components, each of which must live on some node.
- Deployment diagrams may also contain packages or subsystems, both of which are used to group elements of your model into larger chunks.

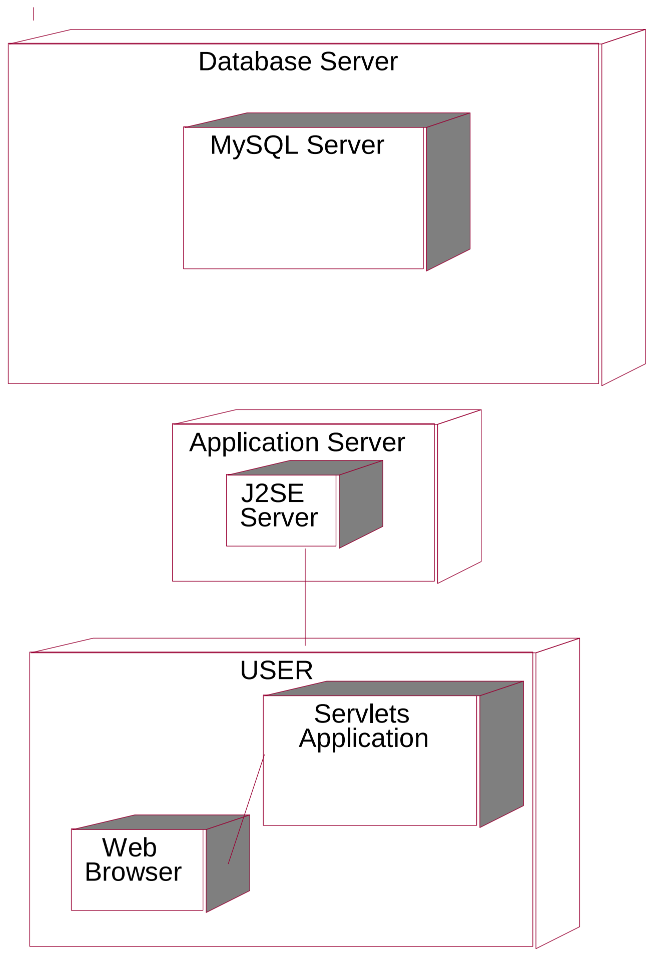


Fig 5.15 Deployment Diagram

5.3 Control Flow diagrams

5.3.1 Activity Diagram

- An activity diagram shows the flow from activity to activity. An activity is an ongoing nonatomic execution within a state machine.
- Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value.

Activity diagrams commonly contain

- Activity states and action states
- Transitions
- Objects

Like all other diagrams, activity diagrams may contain notes and constrains **Login Process**

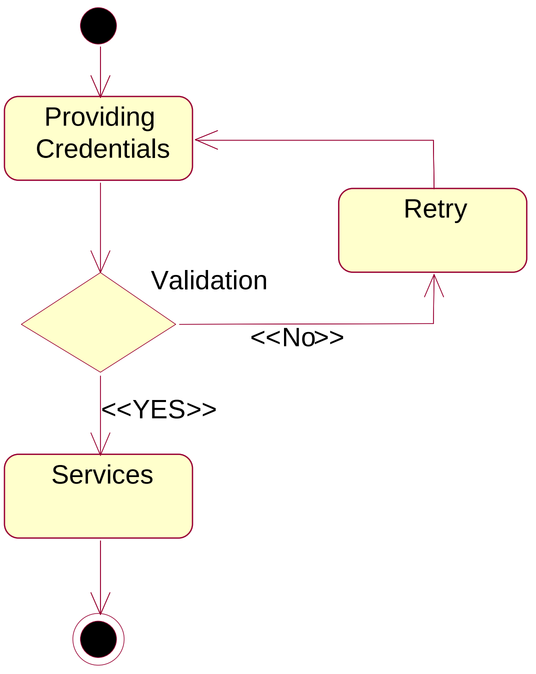


Fig 5.16 Login Activity Diagram

Registration Process

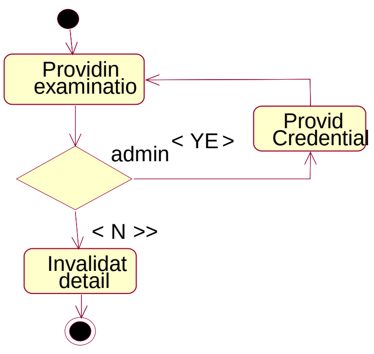


Fig 5.17 Registration Activity Diagram

5.4 Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views.

In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

5.4.1 ER-Diagrams

Entity Relationship Diagrams (ERDs) illustrate the logical structure of databases. An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database.

ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

Entity relationship diagrams are a way to represent the structure and layout of a database. It is used frequently to describe the database schema. ER diagrams are very useful as provide a good conceptual view of any database, regardless of the underlying hardware and software.

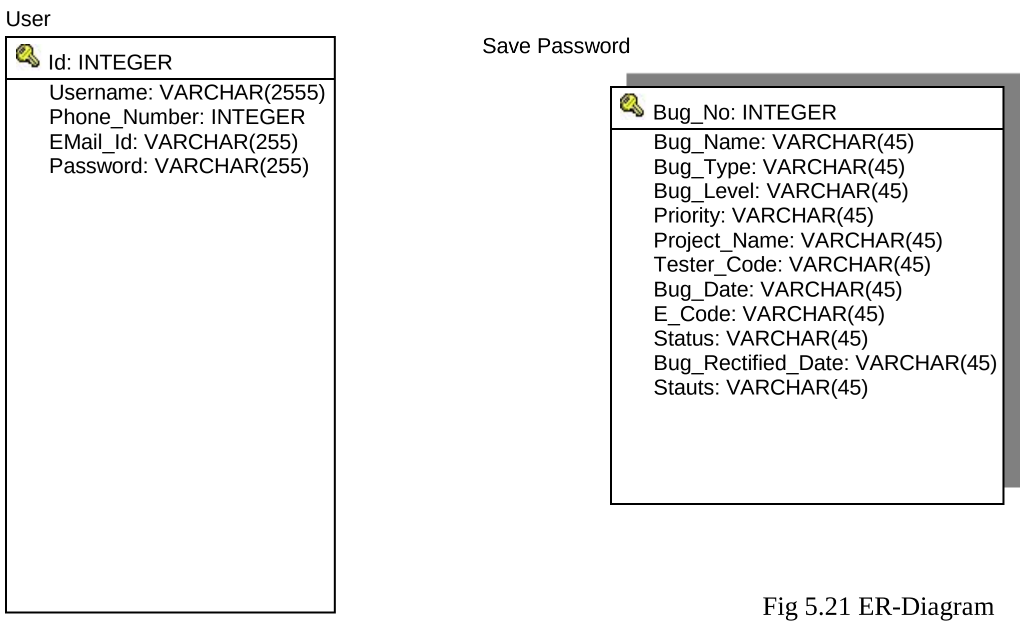


Fig 5.21 ER-Diagram

Chapter 6

IMPLEMANTATION

6.1 General

A programming tool or software tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task. The Chapter describes about the software tool that is used in our project.

6.2. JavaScript

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

JS pages typically comprise of: Static HTML / XML components.

-Special JS tags.

-Optionally, snippets of code written in the java programming language called “script lets.”

- **JS Advantages**

Separation of static from dynamic content: In JS, the logic to generate the dynamic content is kept separate from the static presentation templates by encapsulating it within external Java beans components. When a page designer makes any changes to the presentation template, the JS page is automatically recompiled and reloaded into the web server by the JS engine.

Write Once Run Anywhere: JS technology brings the “Write Once, Run anywhere” paradigm to interactive Web pages.

Dynamic content can be served in a variety of formats: There is nothing that mandates the static template data within a JS page to be of a certain format.

- **JS Architecture:**

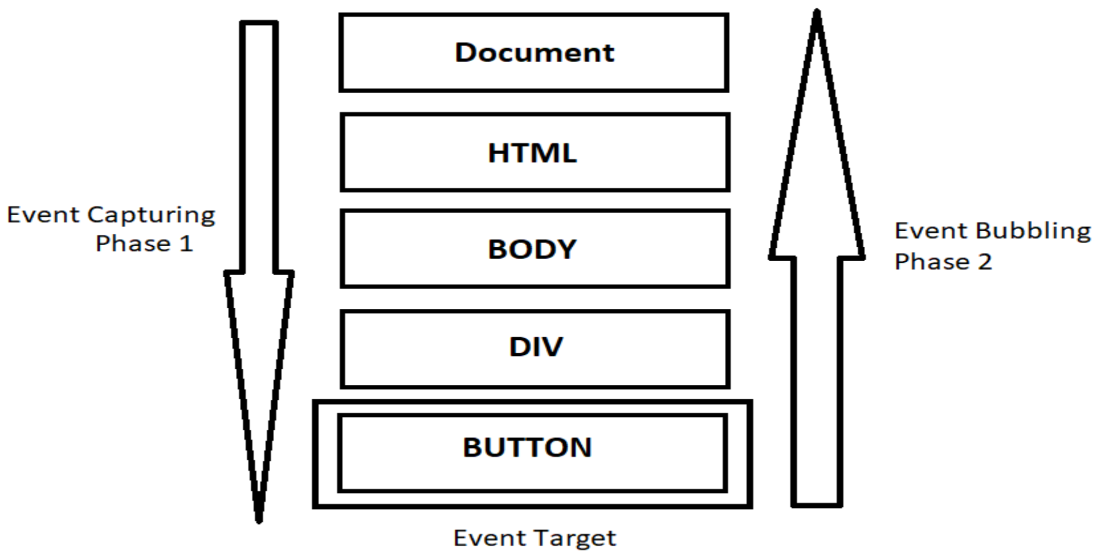
The purpose of JS is to provide a declarative, presentation-centric method of developing servlets. JS pages are subject to a translation phase and a request-processing phase. The translation phase is carried phase is carried out only once, unless the JS page changes, in which case it is repeated. The JS engine itself typically carries out the translation phase, when it receives a request for the JS page for the first time.

Life Cycle of A JS: There are three different phases during lifecycle of an JavaScript event.

- Capturing Phase
- Target Phase
- Bubbling Phase

They follow the same order as listed above.

Capturing Phase is when event goes down to the element. **Target phase** is when event reach the element and **Bubbling phase** is when the event bubbles up from the element.



JS Syntax

- **Directives**

JSs can define information for the container with directives. Here is what directives look like in a general form:

<%@ directive attribute="some Value" attribute="another Value" ... %> There are three directives:

```
(function(){
  "use strict"; // The use strict directive

  // Strict code goes here ...
})();
```

- **Declarations**

Declarations are used to specify supplemental methods and variables. You can think of these are the page's private functions; they can only be called by the JSP where they are defined, or by another JSP that includes it (using the <@ include > directive).

Here is a sample declaration:

<%! // this integer can be used anywhere in this JS page private int myVariable = -1; // this function can be called from anywhere in this JS page public boolean isPositive() { return (myVariable > 0);

- **Scriptlets**

  Scriptlets are bits of Java code. They can do anything but they will most likely concentrate on generating HTML code or setting up variables to be part of later expressions.

- **Expressions**

  Expressions are special-purpose mini-Scriptlets used for evaluating expressions. This could be something as simple as outputting the value of a variable, or a more complicated Java expression, like calling a function and outputting the result. <%= counter %> Note that counter is defined as an int, but we do not need to explicitly convert it to a string.

6.3 Servlets

A servlet is a java programming language class that is used to extend the capabilities of servers that host applications access via a request-response programming mode. Servlets are Java technology's answer to Common Gateway Interface (CGI) Programming. They are programs that run on a Web server, acting as middle layer between request coming from a Web browser or other HTTP client and databases of applications on the HTTP server.

Read any data sent by the user: This data usually entered in a form on a Web page, but could also come from a java applet or a custom HTTP client program.

Look up any other information about the request that is embedded in the HTTP request: This information includes details about browser capabilities, cookies, the host name of the requesting client, and so froth.

Generate the results: This process may require talking to a database, executing an RMI or CORBA call, invoking a legacy application, or computing the response directly.

Format the results inside a document: In most cases, this involves embedding the information inside an HTML page.

Set the appropriate HTTP response parameters: This means telling the browser what type of document is being returned (e.g.HTML), setting cookies and caching parameters, and other such tasks.

Send the document back to the client: This document may be sent in text format (HTML), binary format (GIF images), or even in a compressed format like gzip that is layered on top of some other underlying format.

The Javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines lifecycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the java Servlet API. The HttpServlet classes provide methods, such as doGet and do Post, for handling HTTP-specific services.

To be a servlet, a class should extend HTTPServlet and override doGet or do Post (or both), depending on whether the data is being sent by GET or by POST. These methods take two arguments: An HttpServletRequest and an HttpServletResponse.The HttpServletRequest have methods that let you find out about incoming information such as FORM data, HTTP request headers, and the like. Finally, note that doGet and do Post are called by the service method, and sometimes you may want to override service directly.

Servlet Life Cycle: The life cycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the following steps.

1. If an instance of the servlet does not exist, the Web container:

Loads the servlet class.
Creates an instance of the Servlet class.
Initializes the servlet instance by calling the init method.
2. Invokes the service method, passing request and response objects.
If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method.

- **Cookies**

Cookies are small bits of textual information that a Web server sends to a browser and that the browser returns unchanged when visiting the same Web site or domain later

Browsers generally only accept 20 cookies per site and 300 cookies total, and each cookie is limited to 4KB, cookies cannot be used to fill up someone's disk or launch other denial of service attacks.

- **The Servlet Cookie API**

To send cookies to the client, a servlet would create one or more cookies with the appropriate names and values via new Cookie (name, value), set any desired optional attributes via cookie.setXxx,and add the cookies to the response headers via response.addCookie(cookie).To read incoming cookies, call request.getCookies(), which returns an array of Cookie objects.

Session Management

Many applications require that a series of requests from a client be associated with one another. Sessions are represented by an Http Session object. A session cab be accessed by calling the get Session () method of a request object. This method returns the current session associated with this request, or, if the request does not have a session, it creates one. The timeout period can be accessed by using a session's [get\set] Max Inactive Interval methods.

Session Tracking

A Web container can use several methods to associate a session with a user, all of which involve passing an identifier between the client and the server. The identifier can be maintained on the client as a cookie, or the Web component can include the identifier in every URL that is returned to the client.

In fact, on many servers, they use cookies if the browser supports them, but automatically revert to URL-rewriting when cookies are unsupported or explicitly disabled.

 The Session Tracking API

Using sessions in servlets is quite straightforward, and involves looking up the session object associated with the current request, creating a new session object when necessary, looking up information associated with a session, storing information in a session, and discarding completed or abandoned sessions.

Chapter 7

TESTING

Software Testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. In general, software engineers distinguish software faults from software failures. Our project" Password Management System" is tested with the following testing methodologies.

The test process begins by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework for developing the test methodologies.

The process of executing a system with the intent of finding an error. Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.

Testing

- **Testing Methodologies**
- Black box Testing:
- White box Testing.
- Gray Box Testing.

- **Levels of Testing** Unit Testing.
- Module Testing.
- Integration Testing.
- System Testing.
- User Acceptance Testing.

- **Types Of Testing**
- Smoke Testing.
- Sanitary Testing.
- Regression Testing.
- Re-Testing.
- Static Testing.
- Dynamic Testing.
- Alpha-Testing.
- Beta-Testing.
- Monkey Testing.
- Ext….
- **TCD (Test Case Documentation)**
- **STLC**
- Test Planning.
- Test Development.
- Test Execution.
- Result Analysis.
- Bug-Tracing.
- Reporting.

- **Microsoft Windows – Standards**
- **Manual Testing**
- **Automation Testing (Tools)** Win Runner. Test Director

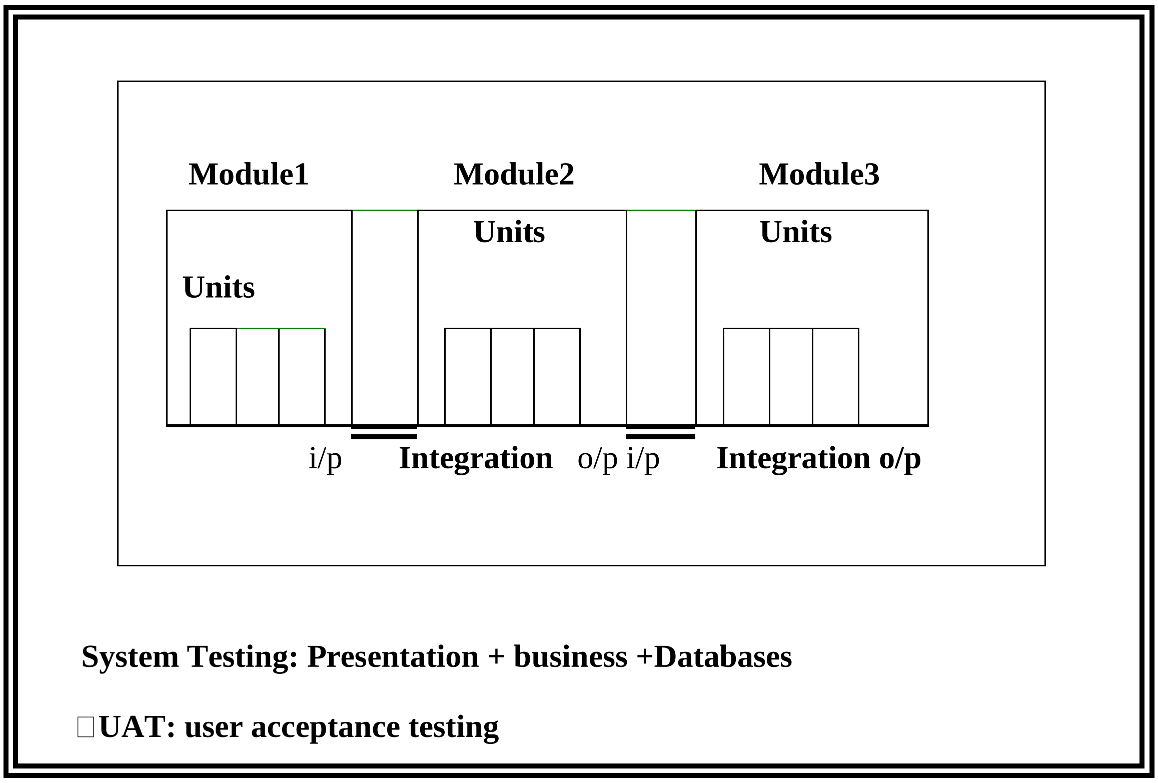
Testing:

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements Defect is nothing but bug.
- Testing --- The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script /program did not execute properly.

Testing Methodologies:

- **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
  Usually Test Engineers are involved in the black box testing.
- **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.
  Usually The Developers are involved in white box testing.
- **Gray Box Testing**: is the process in which the combination of black box and white box techniques are used.

Levels of Testing


Module1
Module2
Module3
Units
Units
Units
i/p
Integration
o/p i/p
Integration o/p
System Testing: Presentation + business +Databases
UAT: user acceptance testing

STLC (SOFTWARE TESTING LIFE CYCLE)

Test Planning: **1.**Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.

2. Objective of testing,

3. Areas that need to be tested,

4. Areas that should not be tested,

5. Scheduling Resource Planning,

6. Areas to be automated, various testing tools used

Test Development: **1.** Test case Development (check list)

2. Test Procedure preparation. (Description of the test cases)

Test Execution: **1.** Implementation of test cases. Observing the result. **Result Analysis**:
1. Expected value: is nothing but expected behavior
Of application.

2. Actual value: is nothing but actual behavior of the
application

Password Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

TCD (Test Case Document)

Test Case Document Contains

- **Test Scope (or) Test objective**
- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Academic details of student project:

Test scope

- Test coverage is provided for the screen " Academic status entry" form of a student module of university management system application
- Areas of the application to be tested

Test Scenario

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively.

Test Cases

- Template for Test Case

T.C.No	Description	Exp	Act	Result

Guidelines for Test Cases

1. GUI Test Cases

- Total no of features that need to be check
- Look & Feel
- Look for Default values if at all any (date & Time, if at all any require)
- Look for spell check

Example for GUI Test cases

T.C. No	Description	Expected value	Actual value	Result
1	Check for all the features in the screen	The screen must contain all the features		
2	Check for the alignment of the objects as per validations	The alignment should be in proper way		

2. Positive Test Cases

- The positive flow of the functionality must be considered
- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

Example for Positive Test cases

T.C. No	Description	Expected value	Actual value	Result
1	Check for the date Time Auto Display	The date and time of the system must be displayed		
2	Enter the valid user id into the user id field	It should accept		

3. Negative Test Cases

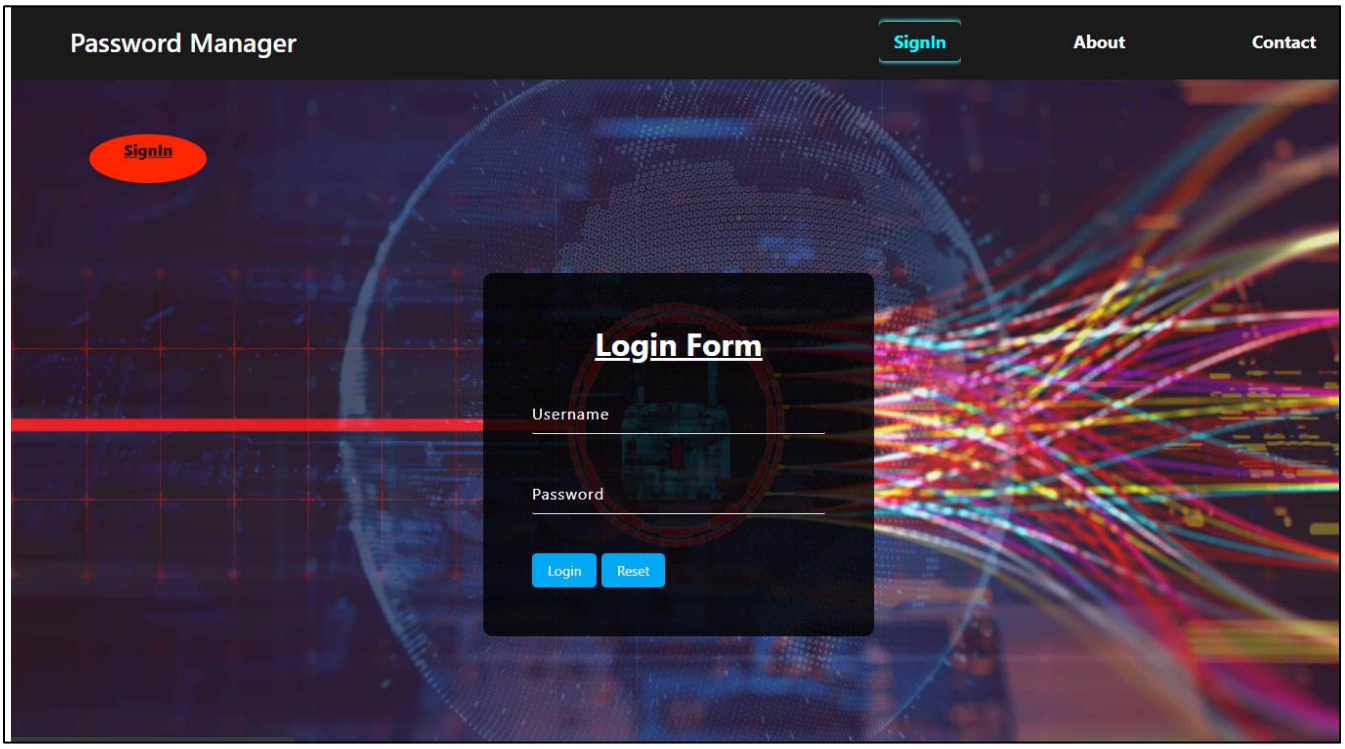
- Must have negative perception.
- Invalid inputs must be used for test. **Example for Negative Test cases**

T.C. No	Description	Expected value	Actual value	Result
1	Try to modify the information in date and time	Modification should not be allow		
2	Enter invalid data in to the login details form, click on Save	It should not accept invalid data, save should not allow		

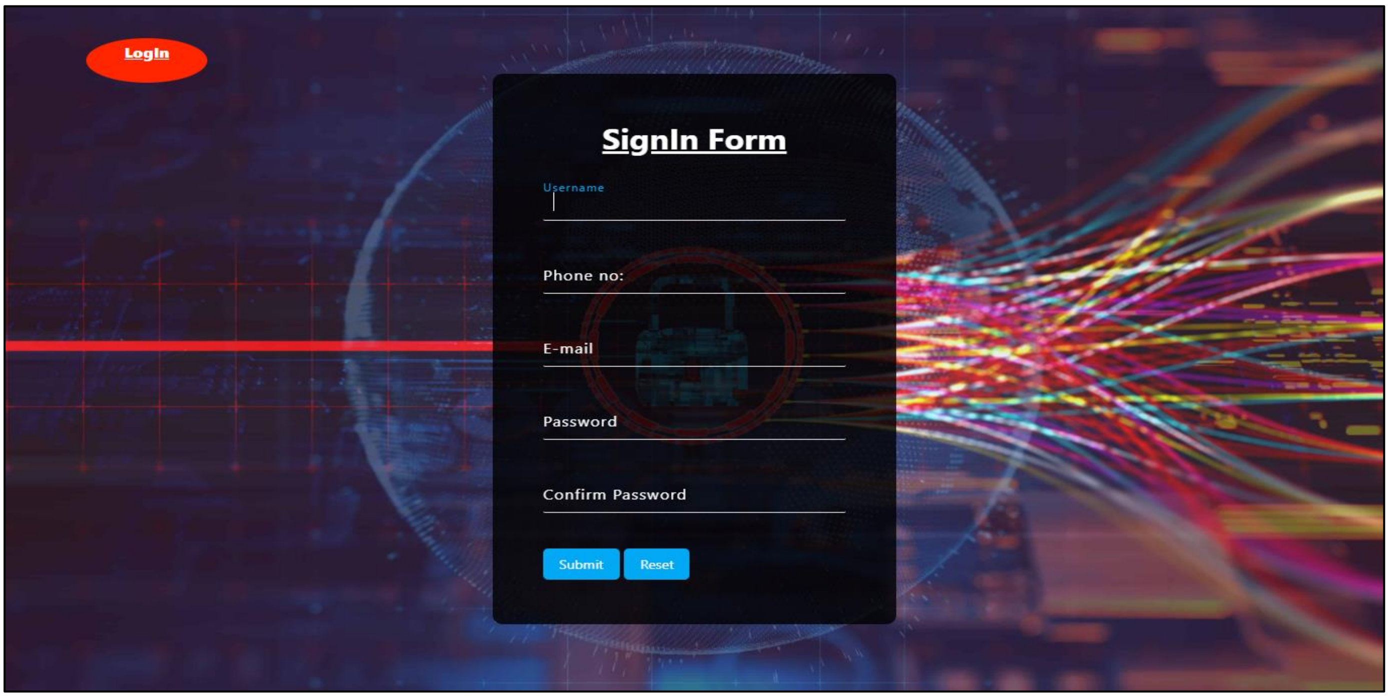
Chapter 8

GRAPHICAL USER INTERFACE

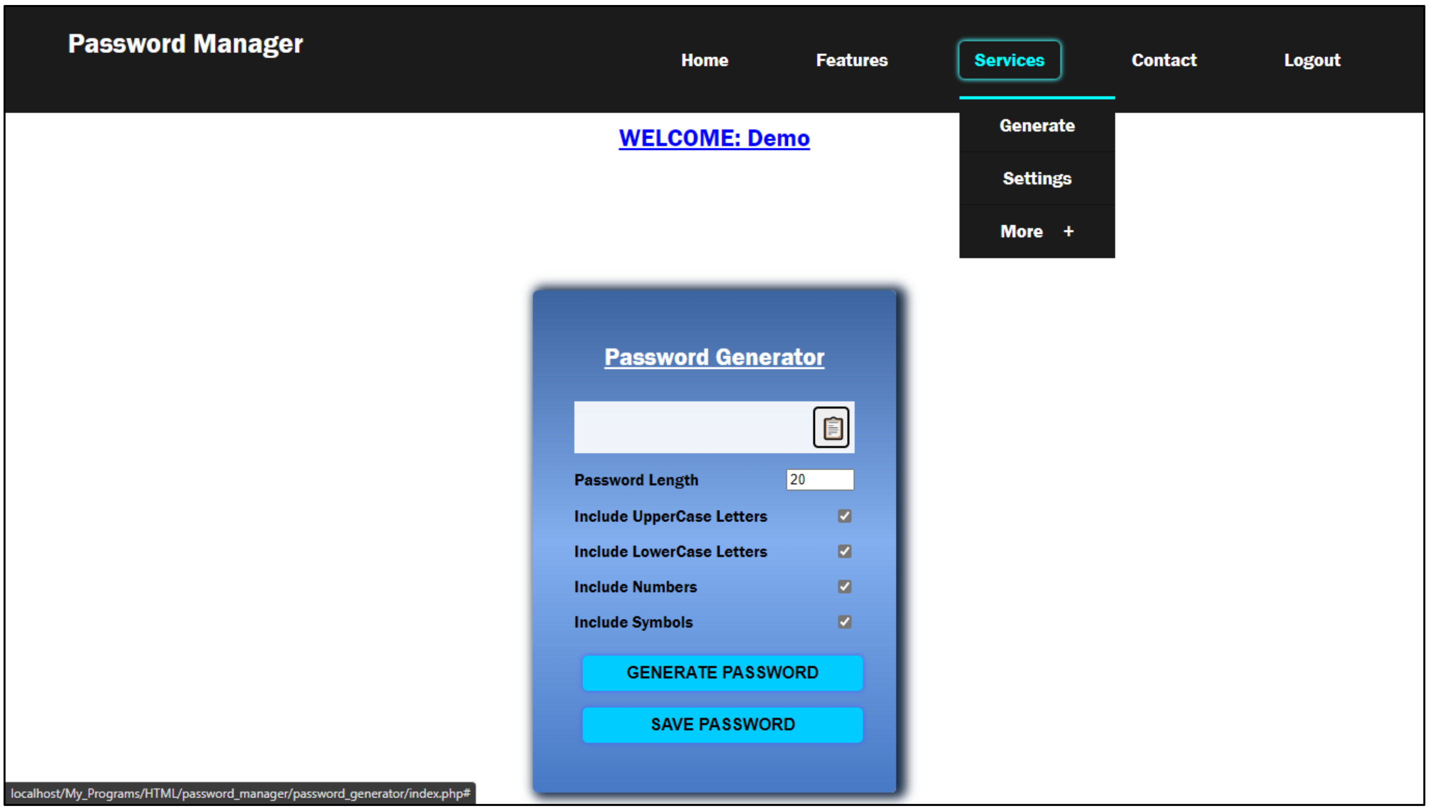
8.1 Home Page



8.2 Registration Form

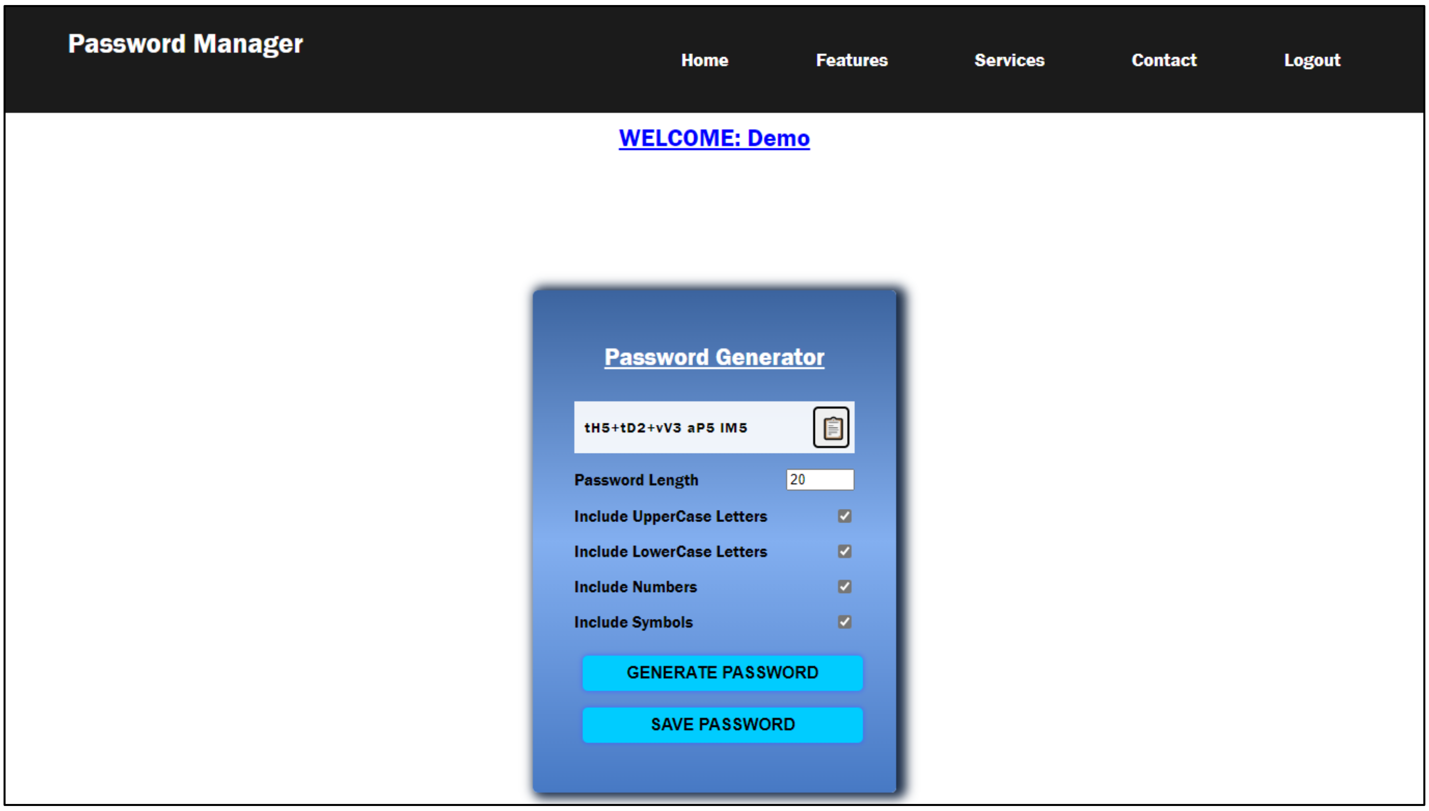


8.3 User home



Password Length 20
Include UpperCase Letters ✓
Include LowerCase Letters ✓
Include Numbers ✓
Include Symbols ✓

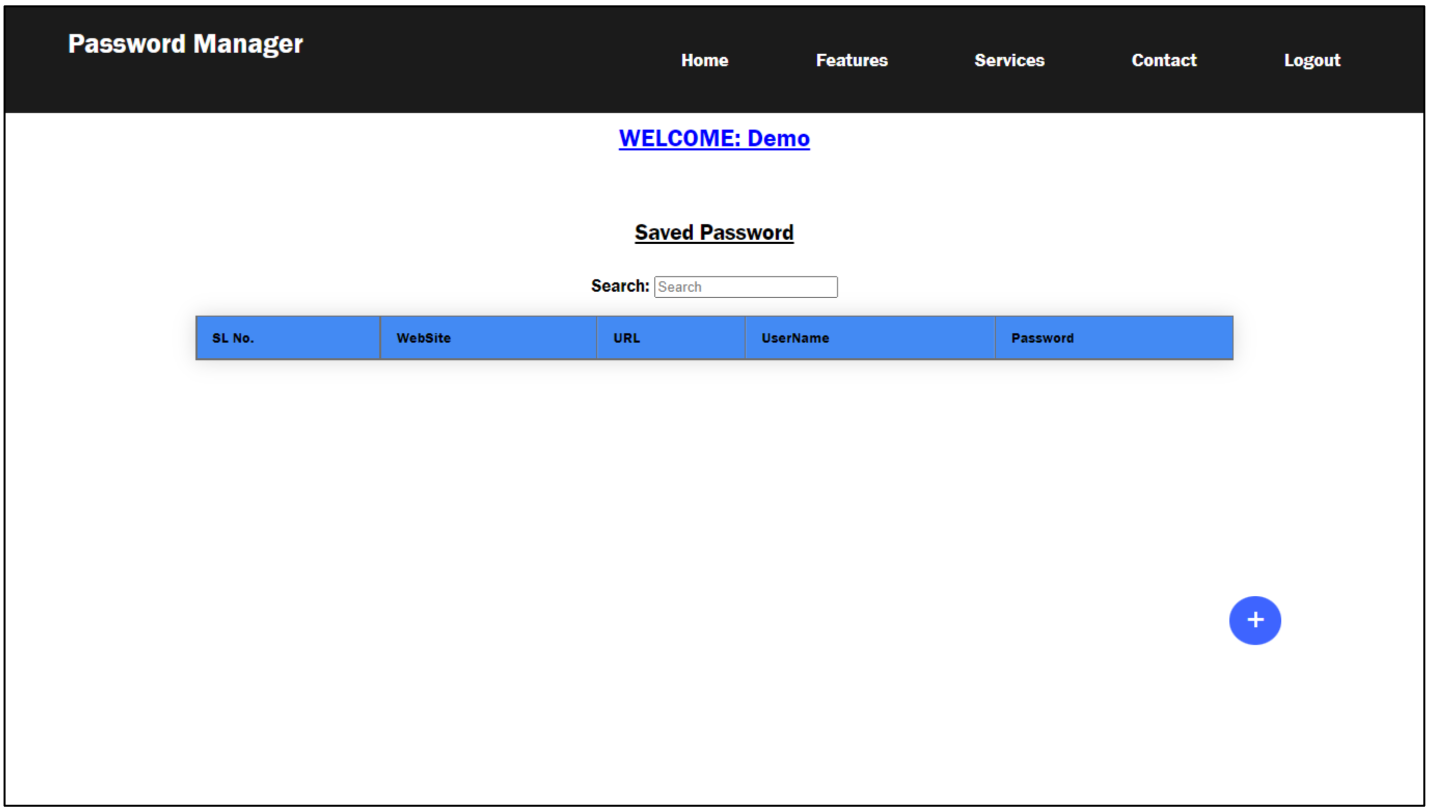
8.4 To Generate Password



tH5+tD2+vV3 aP5 IM5

Password Length 20
Include UpperCase Letters ✓
Include LowerCase Letters ✓
Include Numbers ✓
Include Symbols ✓

8.5 To Save Password



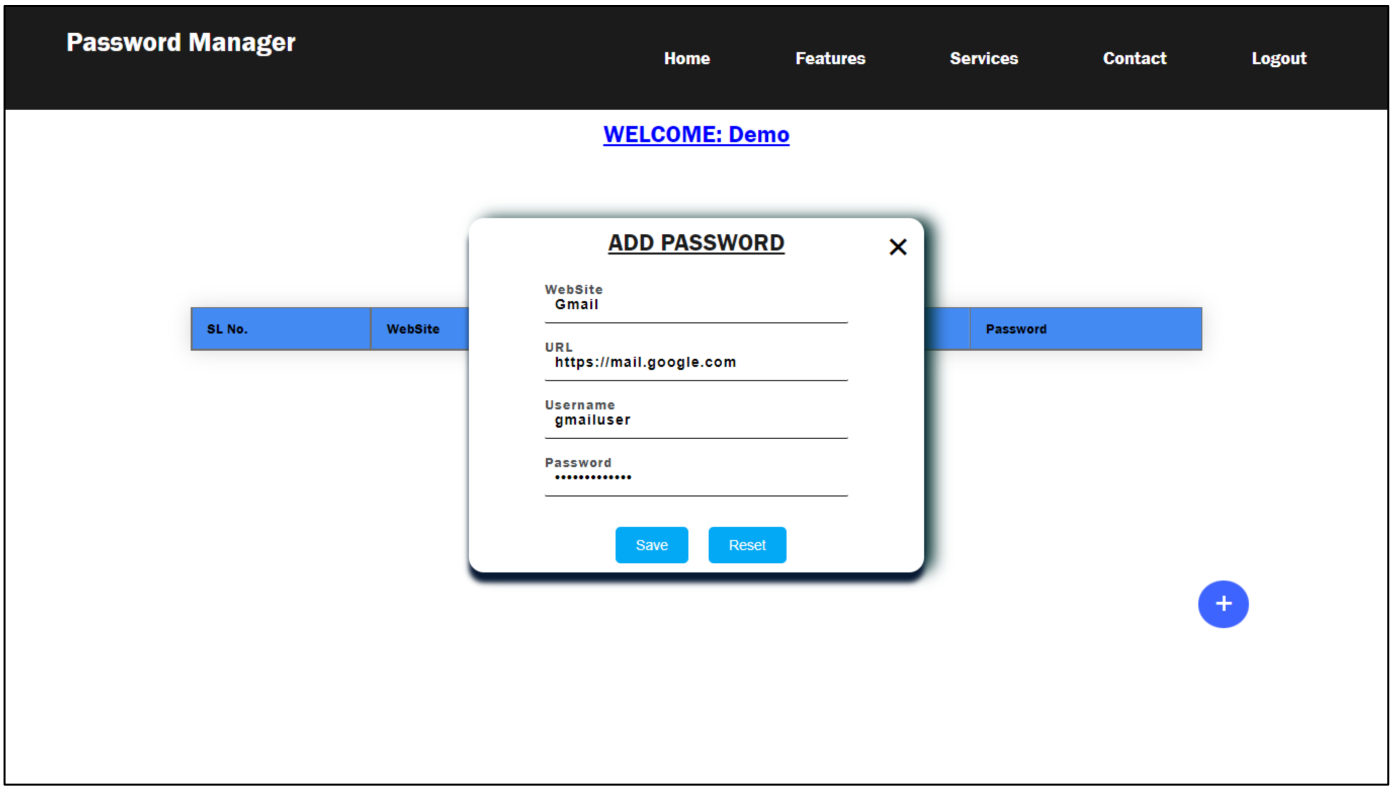
WELCOME: Demo

Saved Password

Search: Search

SL No.	WebSite	URL	UserName	Password

8.6 To Add Password



WELCOME: Demo

ADD PASSWORD

SL No.	WebSite	Password

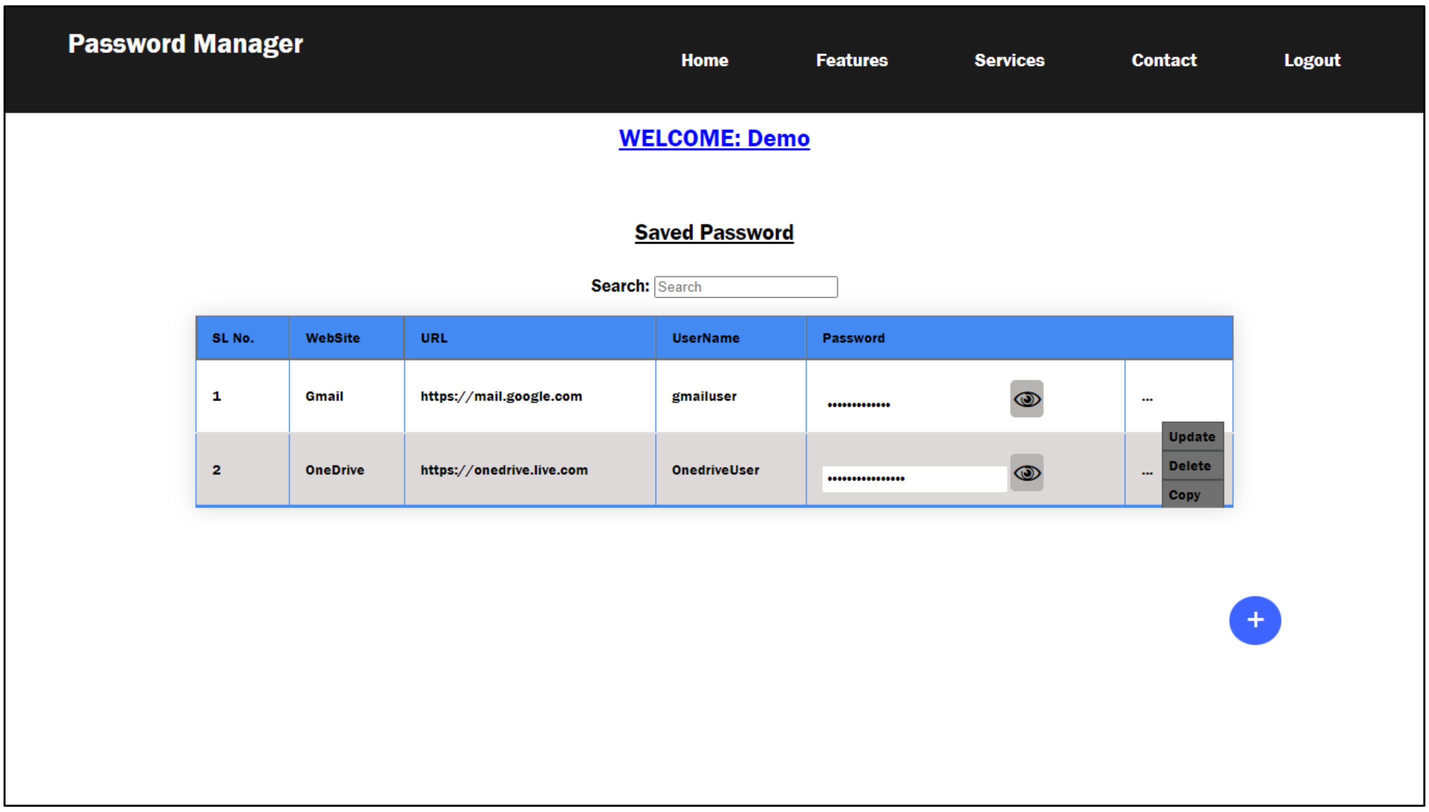
WebSite
Gmail

URL
https://mail.google.com

Username
gmailuser

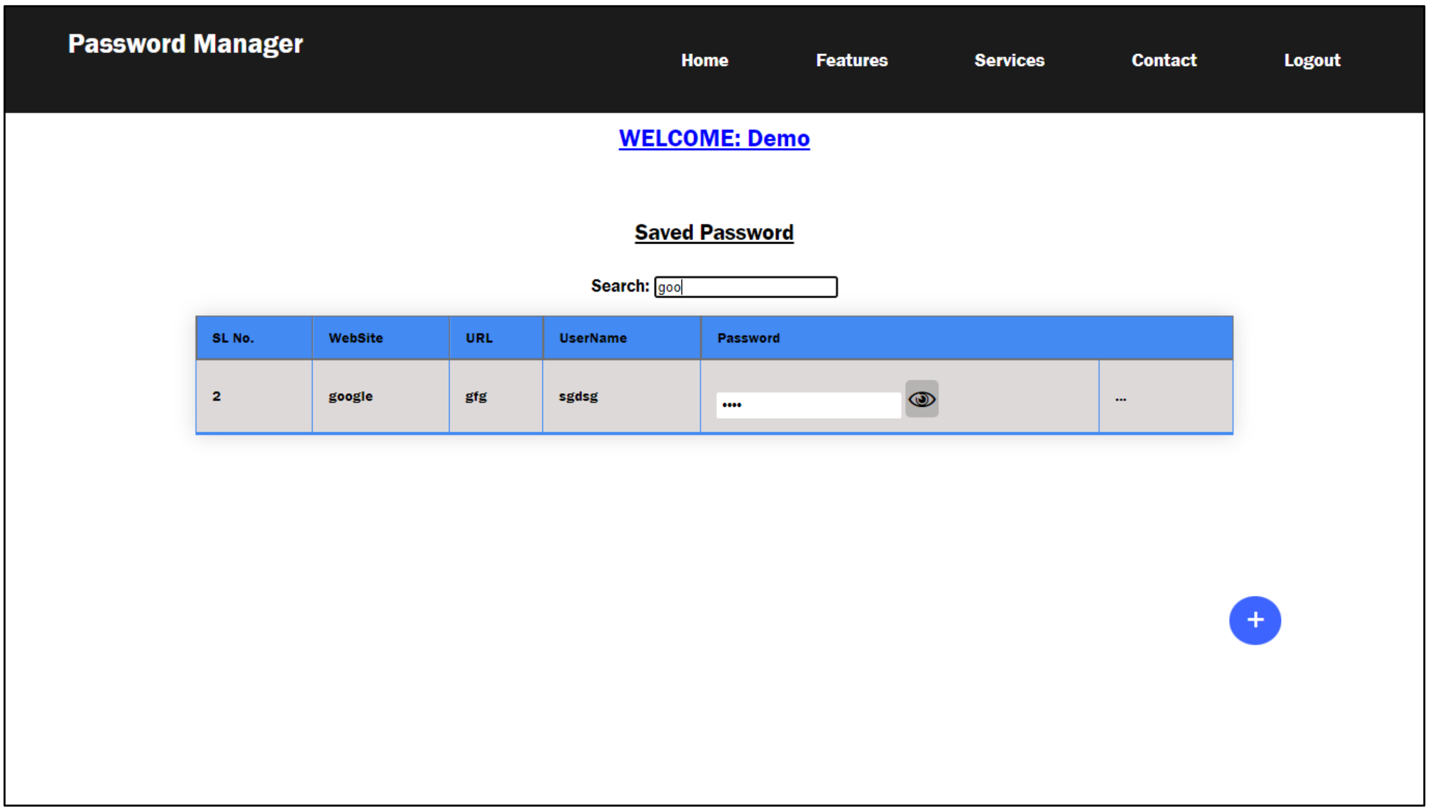
Password
•••••••••••••

8.7 To View All Saved Passwords



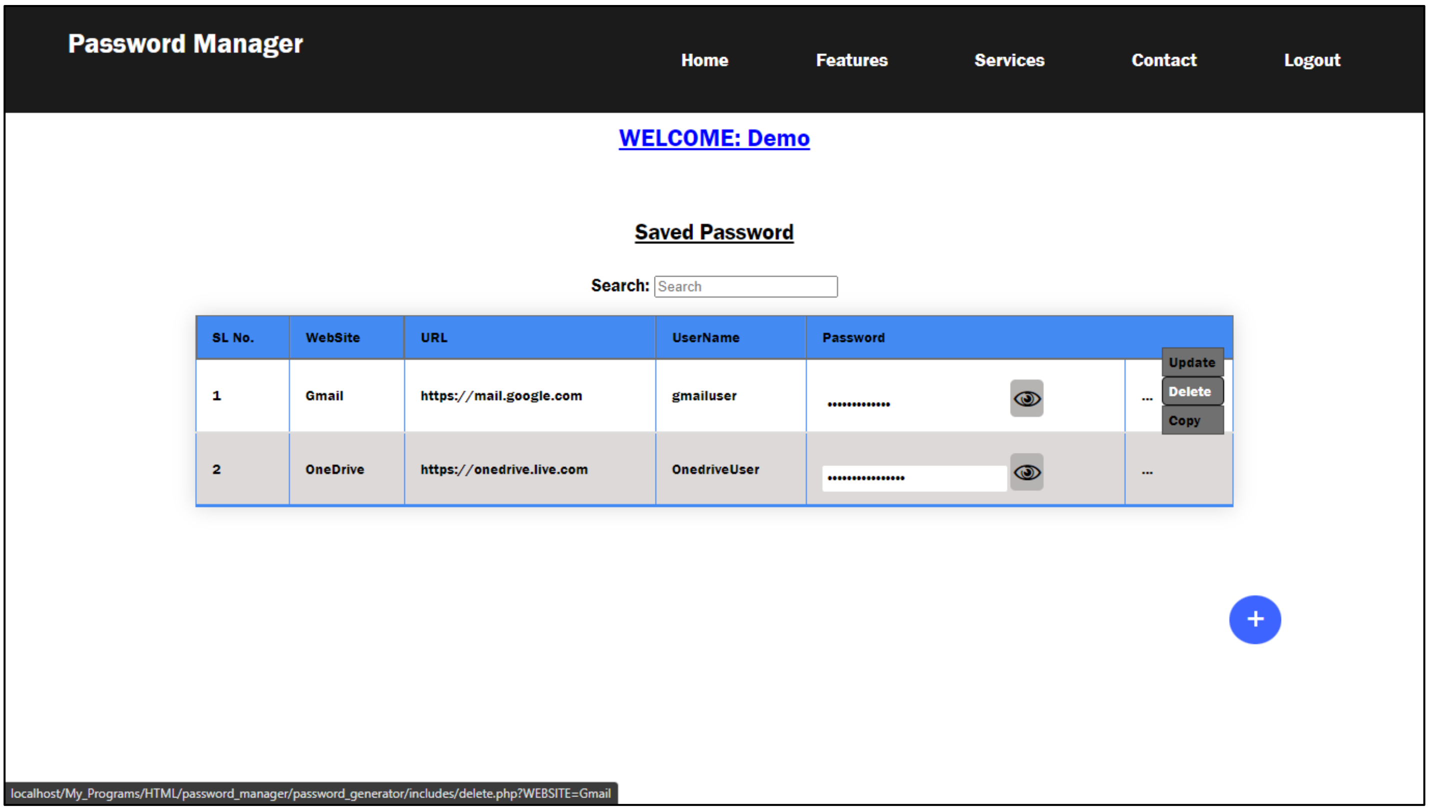
SL No.	WebSite	URL	UserName	Password	
1	Gmail	https://mail.google.com	gmailuser	•••••••••••••	...
2	OneDrive	https://onedrive.live.com	OnedriveUser	•••••••••••••••	...

8.9 To Search Website



SL No.	WebSite	URL	UserName	Password	
2	google	gfg	sgdsg	••••	...

8.8 To Delete Password



Password Manager

WELCOME: Demo

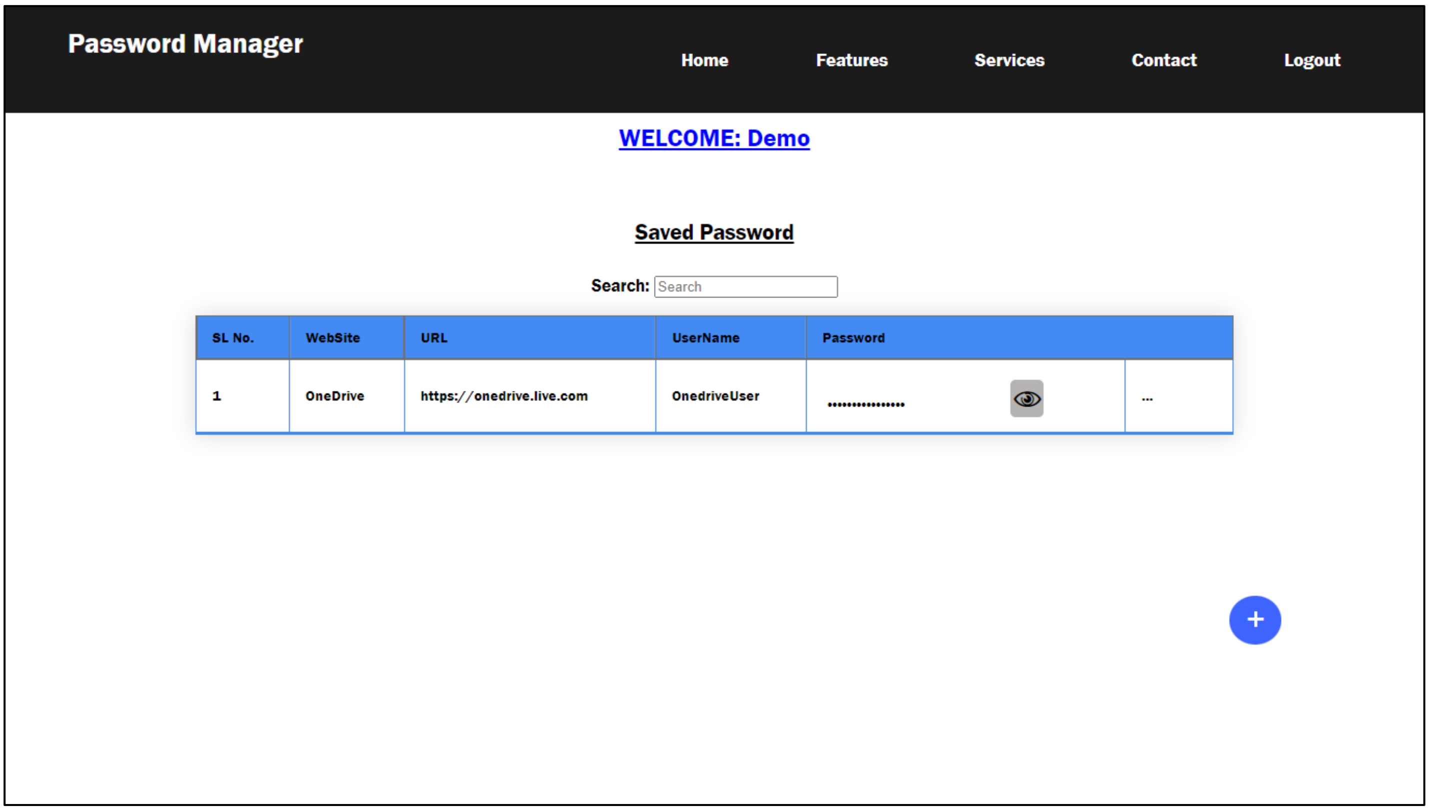
Saved Password

Search: Search

SL No.	WebSite	URL	UserName	Password	
1	Gmail	https://mail.google.com	gmailuser	•••••••••••••	... Update Delete Copy
2	OneDrive	https://onedrive.live.com	OnedriveUser	•••••••••••••••	...

localhost/My_Programs/HTML/password_manager/password_generator/includes/delete.php?WEBSITE=Gmail

8.9 Password Table Deleted



Password Manager

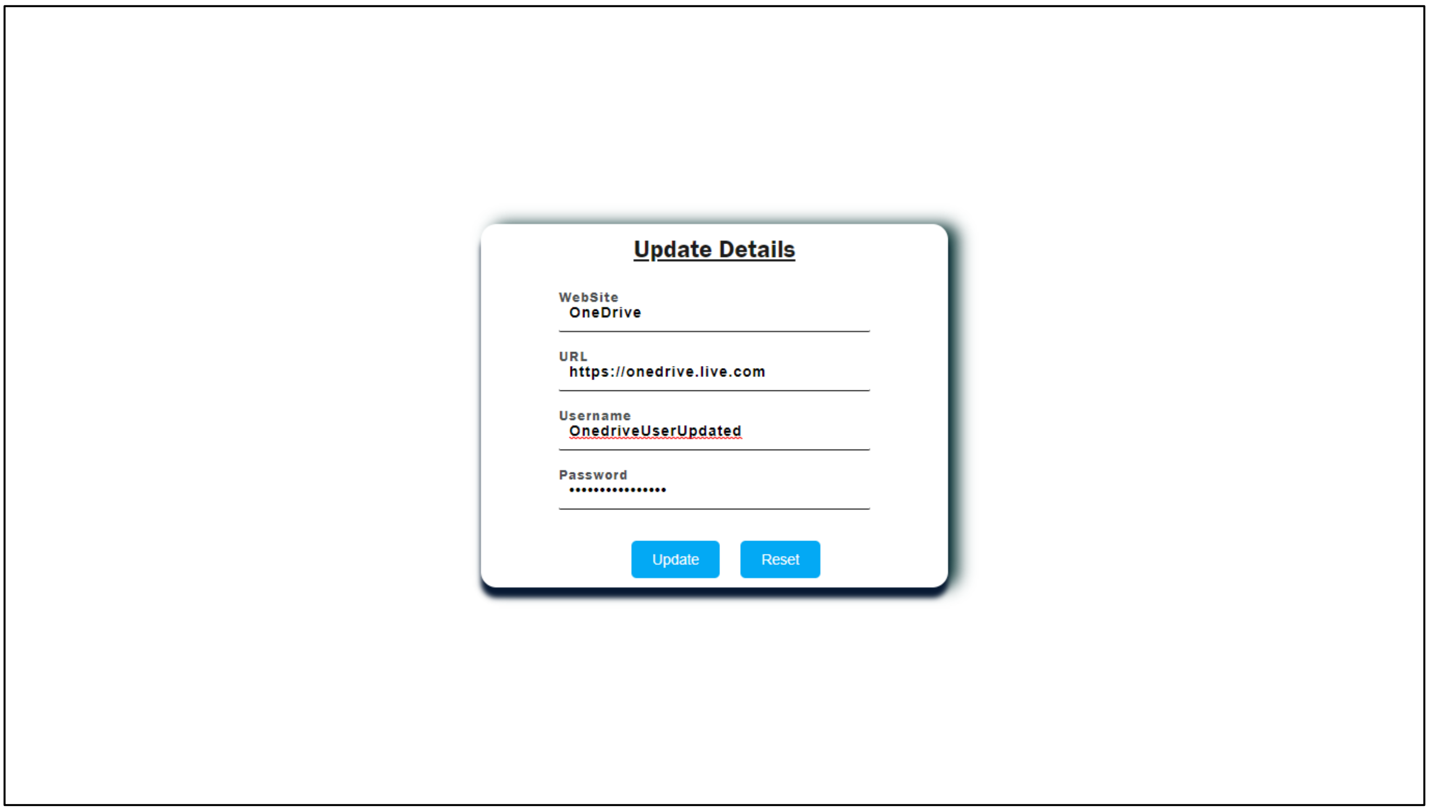
WELCOME: Demo

Saved Password

Search: Search

SL No.	WebSite	URL	UserName	Password	
1	OneDrive	https://onedrive.live.com	OnedriveUser	•••••••••••••••	...

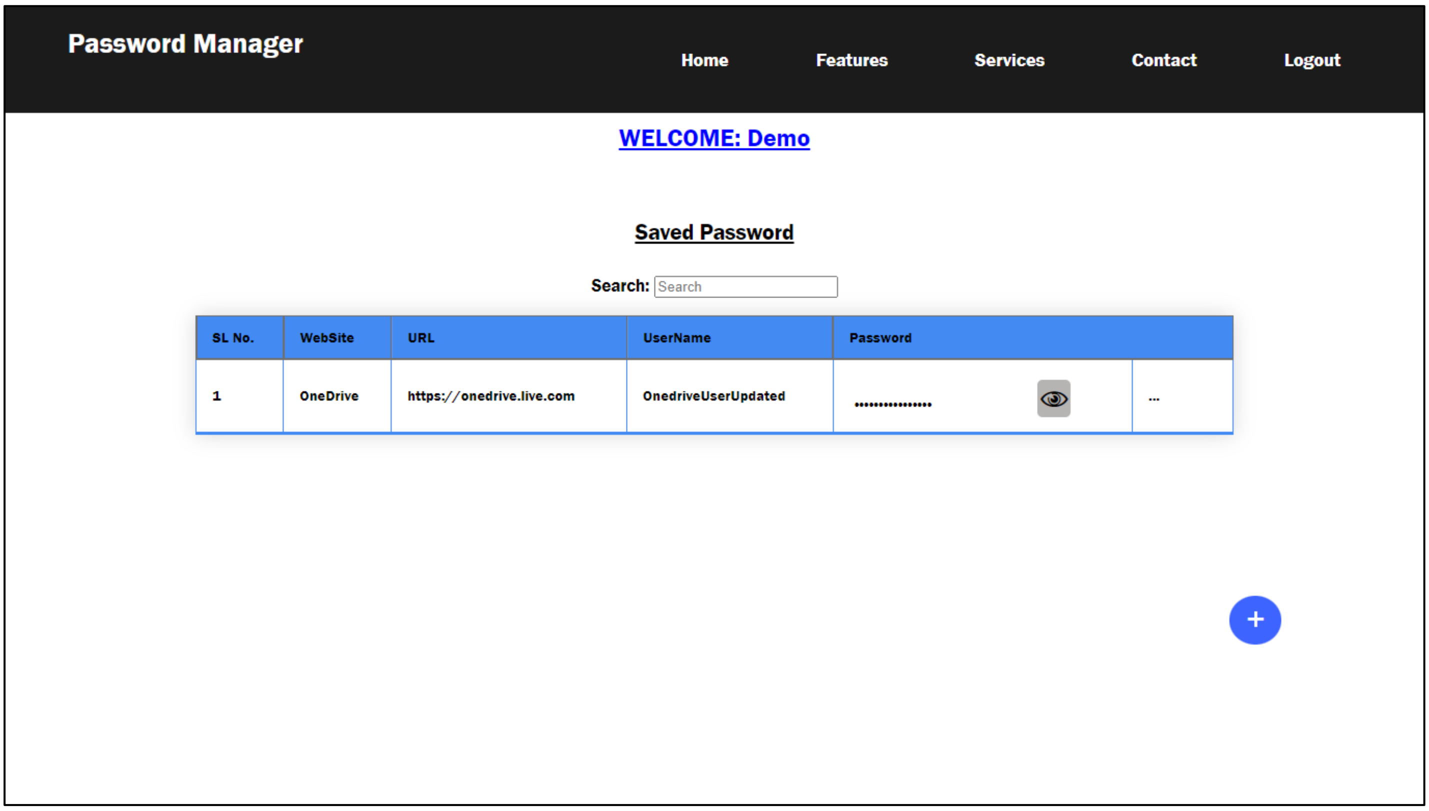
8.10 To Update Password



Update Details

WebSite	OneDrive
URL	https://onedrive.live.com
Username	OnedriveUserUpdated
Password	•••••••••••••••

8.11 Password Updated



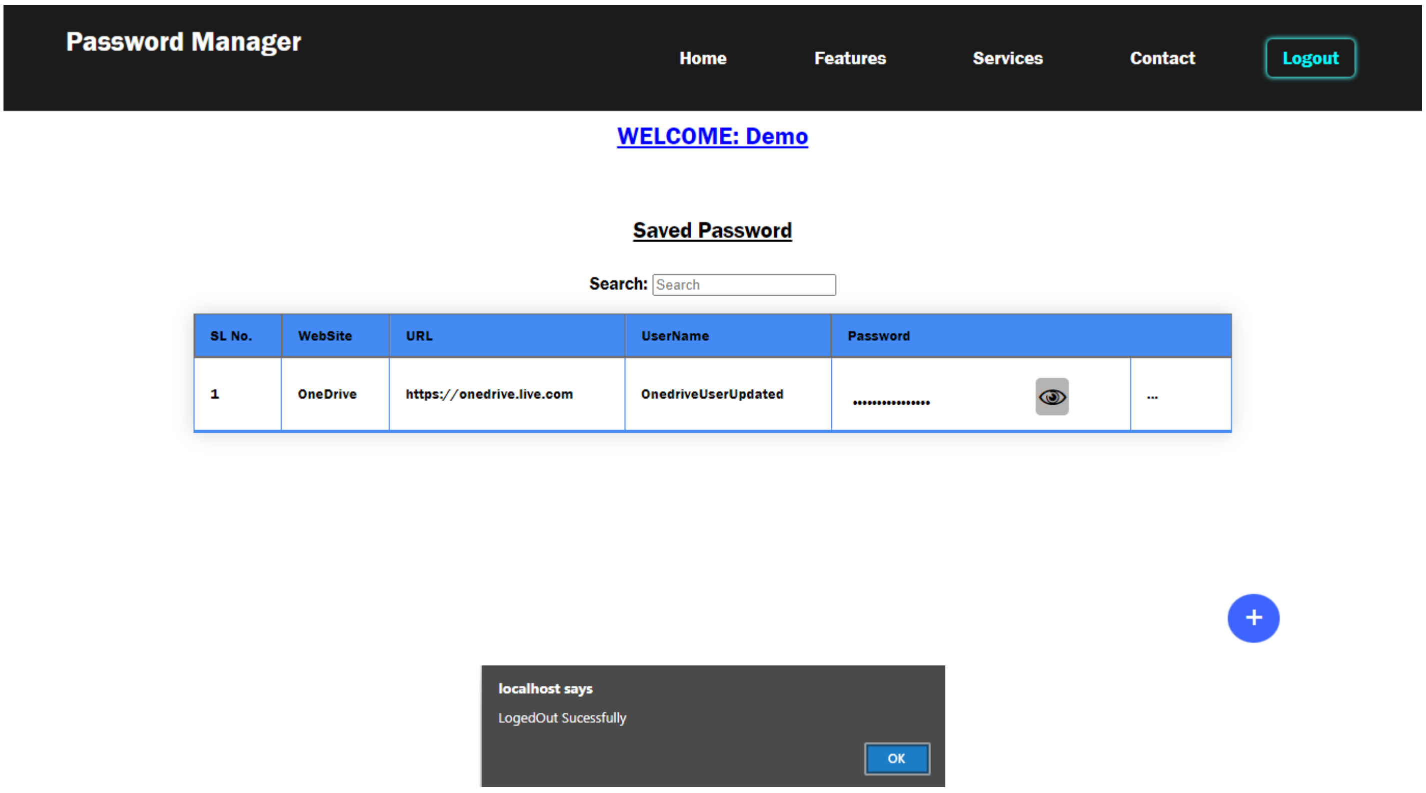
WELCOME: Demo

Saved Password

Search: Search

SL No.	WebSite	URL	UserName	Password	
1	OneDrive	https://onedrive.live.com	OnedriveUserUpdated	•••••••••••••••	...

8.12 LogOut



Password Manager

Home Features Services Contact Logout

WELCOME: Demo

Saved Password

Search: Search

SL No.	WebSite	URL	UserName	Password	
1	OneDrive	https://onedrive.live.com	OnedriveUserUpdated	•••••••••••••••	...

localhost says

LogedOut Sucessfully

OK

Chapter 9

CONCLUSION

User comes to the search engine and makes a query, typically by giving key words, the engine looks up the index and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text.

Most search engines employ methods to rank the results to provide the “best” results first. How a search engine decides which pages are the best matches, and what order the results should be shown in, varies widely from one engine to another.

Search engine is technically the software and algorithms used to perform a search, the term have become synonymous with the website itself.

10. BIBLIOGRAPHY

- **System Analysis And Design**

  By William S. Davis, David C. Yen
- **An Analysis and Design of Information Systems**

  By Simon Bell
- **Software Engineering**

  -By Schach
- **Database Management System**

  -By Jason Price
- **JavaScript: The Definitive Guide**

  -By David Flanagan
- **Java Server Pages Programming: A visual professional guide for Design**

  -By Carsten Thomsen
- **Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5**

  -By Robin Nixon