PASSWORD MANAGEMENT TOOL

A project report submitted to

Rajiv Gandhi University of Knowledge Technologies

SRIKAKULAM

In partial fulfilment of the requirements for the
Award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

3rd year B. Tech 2nd semester

Bhargavi Gollapalli – S170033

CH.G N D Prasad – S170172

Dhaalakshmi Budda – S170389

Under the Esteemed Guidance of

Asst.Prof.Mrs. V.Pavani Mam

Rajiv Gandhi University of Knowledge Technologies - SKLM

CERTIFICATE

This is to certify that the thesis work titled "**PASSWORD MANAGEMENT TOOL**" was successfully completed by **G. BHARGAVI (S170033)**, **CH. G. N. D PRASAD (S170172)**, **B. DHANALAKSHMI (S170389)**. In fulfilment of the requirements for the Mini Project in Computer Science and Engineering of Rajiv Gandhi University of Knowledge Technologies under my guidance and output of the work carried out is satisfactory.

Asst Prof. V. Pavani Mam	**Asst Prof.Lakshmi Sri Mam**
Project Guide	**Head of the Department**

DECLARATION

I declared that this thesis work titled "**PASSWORD MANAGEMENT TOOL**" is carried out by me during the year 2021-22 in fulfillment of the requirements for the Mini Project in **Computer Science and Engineering.** I further declare that this dissertation has not been submitted elsewhere for any Degree. The matter embodied in this dissertation report has not been submitted elsewhere for any other degree. Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

G BHARGAVI (S170033)

CH G N D PRASAD (S170172)

B DHANALAKSHMI (S170389)

ACKNOWLEDGEMENT

I would like to articulate my profound gratitude and indebtedness to my project guide **Asst.Prof.V Pavani**, Assistant Professor who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for me to get an opportunity to work under his guidance and complete the thesis work successfully. I wish to extend my sincere thanks to **Asst.Prof.LAKSHMI SRI, M.Tech,** Head of the Computer Science and Engineering Department, for her constant encouragement throughout the project.

I am also grateful to other members of the department without their support my work would have been carried out so successfully.I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

Project Associate

G BHARGAVI - S170033

CH G N D PRASAD - S170172

B DHANALAKSHMI - S170389

ABSTRACT

The main aim of our project is to build secured Password Management Tool which supports various interfaces like web, smart devices, browser plugins, Application plugins. As technology is growing day by day we have to remember many passwords. It is common to forgot any one of the password. In order to solve this problem , we came up with the idea of an **"PASSWORD MANAGEMENT TOOL"** which maintains the passwords for each application with strong encryption before storing the user's personal data to hard drive or any where. It is easier to manage passwords for different websites/ applications/accounts with high security and user can remember only one master password. The application is scalable and maintainable.

Technologies : Jdbc, Jsp,Maven, MySQL, Spring boot, Spring Security,junit

INDEX

Chapter-1

INTRODUCTION

1.1 Introduction

In this age of IT revolution, Computer and Information security is the important issue, when whole world is connected to the internet. Authentication is the most ubiquitous form of identification method used as user access control to the system. Passwords are the key to security, it secure our account and personal data from others. Passwords are the cheapest way to secure our account.The main aim of our project, **"Pasword Management tool (PMT)"** is to solve the issues that were arised in our daily life. We have to remember many passwords for our day to day activities. But it is very tedious job as the secured application list is growing day by day.

So one can use " Password Management Tool " to maintain the passwords for each application and user can remember only one master password. As PMT stores data in PC's Memory, it should have strong encryption before storing the user's personal data to the hard drive or any where.This application include basic CRUD operations, Encryption operation. Here one can manage passwords for different applications /websites / accounts with high security, view saved passwords securely, group the password accounts, Delete the passwords, update the group name, delte account groups. The user is mainly responsible for adding and modifying passwords for account.

In this **PMT** application password Length and Characters should be defined by user and while creating account password we need to validate. A Password account should belong to one group only.The password account name should be unique with in the group.User should be able to create a unique group and group name should have only alphabet chars.URL for account should have proper format. It supports various interfaces like Web, Smart Devices, browser plugins. The application should be scalable, maintainable

1.2 Statement of the problem

As technology is growing day by day we have to remember many passwords. Passwords are fundamental for information security. The idea to implement this software came up from our regular experience. They are used as a first -line defense in securing almost all our electronic information, networks, servers, devices, accounts, databases, files, and more. It is common to forgot any one of the password. In order to solve this problem , we proposed our idea i.e., **"PASSWORD MANAGEMENT TOOL"** which can solve these issues up to some extent.

1.3 Objective

- A person can register or login into the web application to store the passwords with high security and strong encryption.
- One can perform basic CRUD (Create, read, Update, Delete) operations on passwords which are stored in the application.
- View the saved passwords securely and group the accounts.
- Save any number of passwords using this tool.
- This application is scalable and maintainable.

1.4 Goals

- User Friendly
- Simple and Fast
- Effective
- Scalable
- Maintainable

1.5 Scope

Our software application can be used by all the people who wants to store their passwords in most secured way. It is easier to manage passwords for different applications/ websites /accounts with high security. The user can

- Remember one master password
- View saved passwords securely and group the password accounts.
- view all the different accounts he/she hold in a list view and also all the details related to the account like the user name, password etc .
- save any number of account information using this tool.
- Perform basic CRUD (create, Read, Update, Delete) Operations.

1.6 Applications

What users can do in Password Management Tool:

- **Adding Account Password**
  One can add account information like username , password, Url, group assigned for a particular account.
- **Group the accounts**

  One can add groups to the application and can add accounts to the groups. Any number of groups can be created. We can directly add the account to group by assigning them in beginning of the add account or else we can update it later.

- **View the account Id and Password**

A person can view the saved passwords in most secured way . The whole application is user friendly and one can easily understand where the add account information rely.

- **Security and Encryption**

  As we are using a new technology spring boot framework to built this application. So we are using a spring security for encrypting password. Spring recommends to use spring security to provide authentication and authorization.

- **Remember one Master Password**
  One can remember only one password i.e., master password. So by using that one can login into the application and can use it.

1.7 Limitations

- Master password should be remember , there is no option like forgot password.
- Master password should not be changed once created.
- A Password account should belong to one group only.

Chapter – 2

LITERATURE SURVEY

2.1 Collect Information

We have taken the information from the other sources like password manager websites to check how they are categorized and organized and we proposed this ourselves.

We gone through various new technologies and found out the more secured way of encrypting the password. We choosen a path entirely on java and collected the information on frameworks like spring boot and spring security. Spring security is a framework which provides security features like authentication and authorization to create secure java Enterprose applications.

2.2 Study

Password management tool key features:

- Storing different account usernames and passwords
- View saved passwords securely
- Group the accounts
- Performing CRUD operations on passwords

2.3 Benefits

- The main benefit of PMT is to store and modify the passwords with strong encryption and high security.
- A user friendly web application
- Add different account/ website/ application passwords in most easiest way and perform the basic CRUD operations on the passwords and usernames.
- Encrypted password can't be decrypted.
- Web application is scalable and maintainable.

2.4 Summary

We build an web application with all these features of storing passwords in most secured way and supports various interfaces like web, Smart Devices, browser plugins. The application should be scalable, maintainable.

Chapter – 3

ANALYSIS

3.1 Existing System

A design and implementation of Passwords Management System (PMS), by which you can manage your usernames and passwords details in an encrypted format in the database. It was made based on Blowfish Algorithm, which is a symmetric block cipher. 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 any number of passwords.There are so many web applications and apps to store the passwords and performing operations on the passwords like update, read, delete etc.

3.2 Disadvantages

One of the application that we discussed above is Password Management system and that was based on blowfish algorithm. The disadvantages of blowfish algorithm are:

- Each pair of users needs a unique, so as number of users increase, key management becomes complicated.
- It cannot provide authentication and non-repudiation as two people have same key.
- It also has weakness in decryption process over other algorithms in terms of time consumption and serially in throughput.
- There is high chance of decrypt the encrypted password.

There are some other password managers that use encryption algorithms like blowfish, MD5,DES etc. Those are encryption algorithms which are decryptable and high chance getting hack.

3.3 Proposed System

In order to solve the problems of storing passwords in most secured way , we came up with the idea “Password Management Tool” .

Building a secured password management tool which supports various interfaces like web, Smart Devices, application Plugins. This application is scalable and maintainable. We are using technologies like maven, spring boot, spring security, junit, MySQL, jsp. Providing high security to the system by spring security framework. Spring Security provides an module for securing passwords.The passwords are encoded in database in most secured way.

We are using AES and NoOppassword Encoder which is recommended by spring... Spring security also provides a secured session using JWT.The combination spring boot and spring security makes java enterprise applications more secure. A spring Security framework provides following features to spring boot application:

- Protection against attacks like session fixation, and clickjacking.
- JAAS(Java Authentication and Authorization Service) Module
- Password Storage
- Support Java Configuration
- Protect against brute force attacks
- HTTP authentication
- Web form authentication
- Basic Access Authentication

3.4 Advantages

- A user friendly Web application
- Encrypting all the passwords in database function
- No chance of hacking the password
- Stroing the passwords with high security and strong encryption.
- View the saved accounts and group the account
- Saving any number of passwords and creating any number of groups
- Application is scalable and maintainable
- Performing CRUD operations on passwords.
- Spring security provides secured session using JWT.

3.5 System Requirements

Software Requirements

- Intellij Community Edition
- Xampp Control panel
- Windows 10
- Spring Boot, Spring Security, MySQL, JSP, Maven

Hardware Requirements

- RAM : 4GB or above
- Hard Disk : 500 GB or above

CHAPTER – 4

SYSTEM DESIGN

4.1 Design of the system

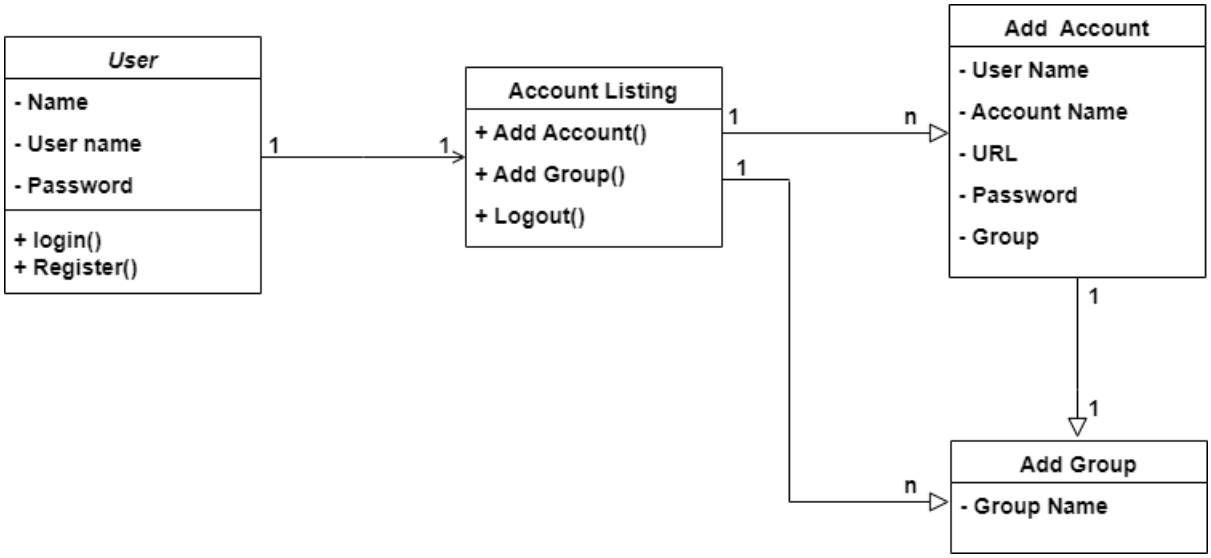
Unified Modelling Language (UML) was created in 1995 by using merging diagramming conventions used by three application development methodologies: OMT by James Rumbaugh, Objector y by Invar Jacobson and the Brooch procedure by using Grady Brooch. Previous to this time, these three amigos, together with a few dozen other practitioners had promoted competing methodologies for systematic program development, each and every with its possess system of diagramming conventions. The methodologies adopted a sort of cookbook sort of pushing a application task via a succession of life cycle stages, culminating with a delivered and documented software.

One purpose of UML was once to slash the proliferation of diagramming techniques by way of standardizing on a original modelling language, as a result facilitating verbal exchange between builders. It performed that goal in 1997 when the (international) Object administration team (OMG) adopted it as a commonplace. Some critics don't forget that UML is a bloated diagramming language written by means of a committee. That said, I do not forget it to be the nice manner to be had today for documenting object-oriented program progress.

It has been and is fitting more and more utilized in industry and academia. Rational Rose is a pc Aided program Engineering (CASE) software developed by way of the Rational organization underneath the course of Brooch, Jacobson and Rumbaugh to support application progress using UML. Rational Rose is always complex due to its mission of wholly supporting UML. Furthermore, Rational Rose has countless language extensions to Ada, C++, VB, Java, J2EE, and many others. Considering that Rational Rose has so many capabilities it is a daunting task to master it. Happily, loads can be executed making use of only a small subset of these capabilities.

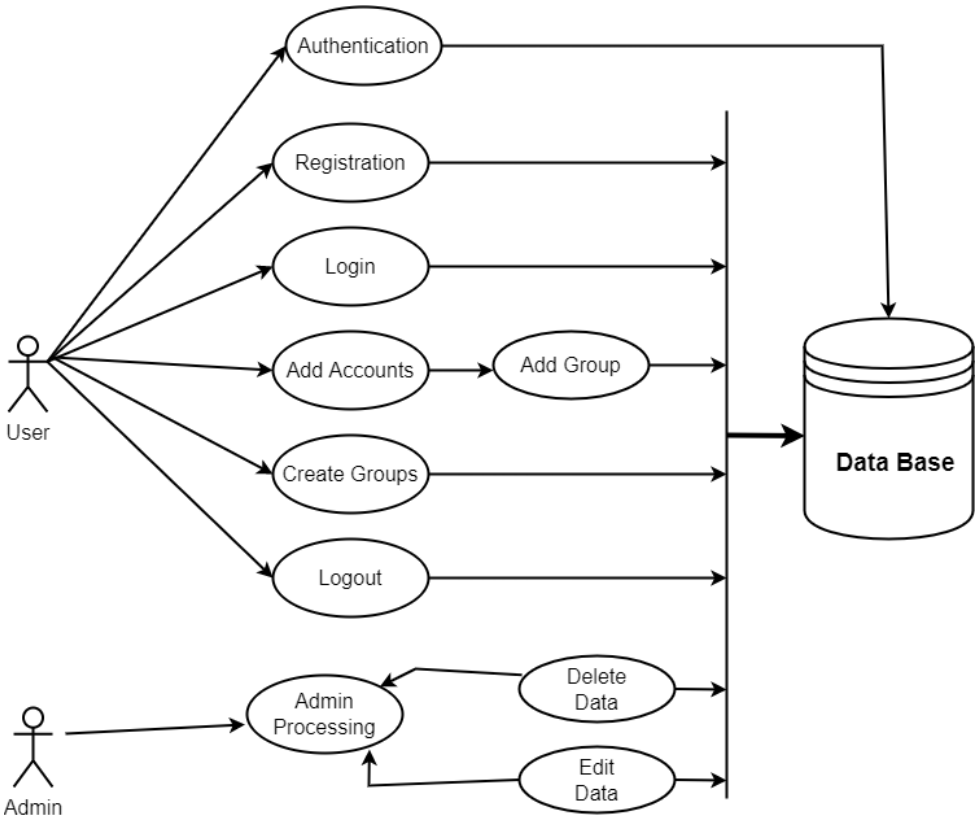
4.1.1 Class Diagram

Class diagram in the Unified Modelling Language (UML), is a kind of static structure diagram hat describes the constitution of a process through showing the system's classes, their attributes, and the relationships between the class. The motive of a class diagram is to depict the classes within a model. In an object-oriented software, classes have attributes (member variables), operations (member capabilities) and relation.



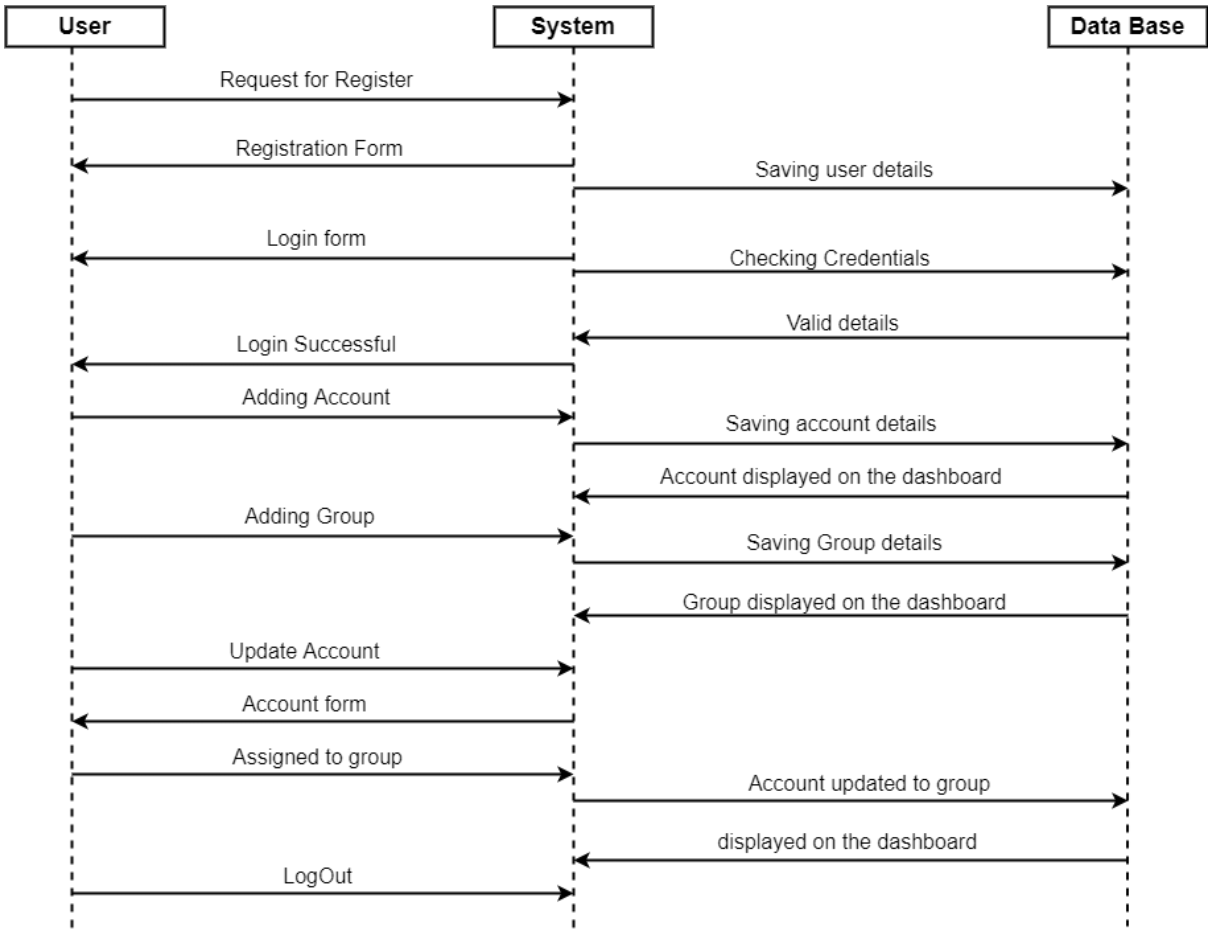
4.1.2 UseCase Diagram:

It is a visually representation what happens when actor interacts with system. A use case diagram captures the functional aspects of a system. The system is shown as a rectangle with name of the system inside ,the actor are shown as stick figures, the use case are shown as solid bordered ovals labeled with name of the use case and relationships are lines or arrows between actor and use cases. Symbols used in Use case are as follows-



4.1.3 Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is one variety of interaction diagram that suggests how methods operate with one other and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are quite often referred to as event-hint diagrams, event situations, and timing diagrams. A sequence diagram suggests, as parallel vertical traces (lifelines), special systems or objects that are residing at the same time, and, as horizontal arrows, the messages exchanged between them, within the order the place they occur.



4.1.4 DFD Diagram:

A data flow diagram or bubble chart (DFD) is a graphical representation of the "flow" of data .through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of

processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

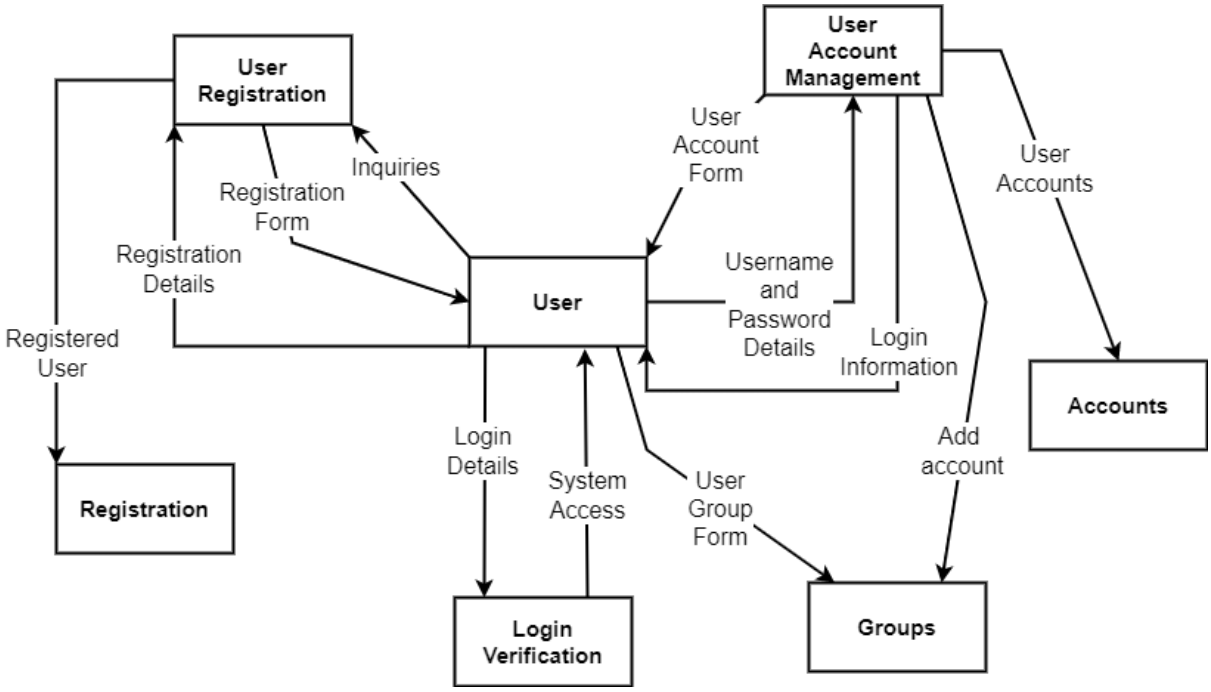
The primitive symbols used for constructing DFD's are:

Symbols used in DFD

A Circle represents a process.

A rectangle represents external entity.

An arrow identifies dataflow.



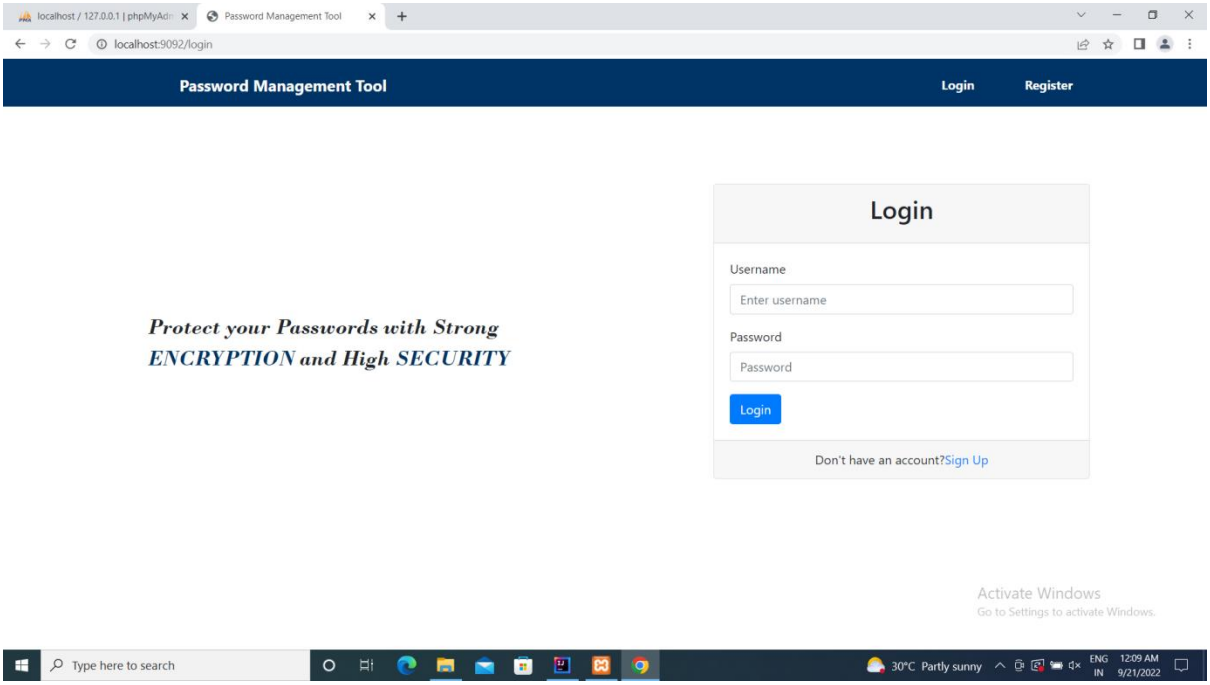
CHAPTER – 5

SYSTEM IMPLEMENTATION

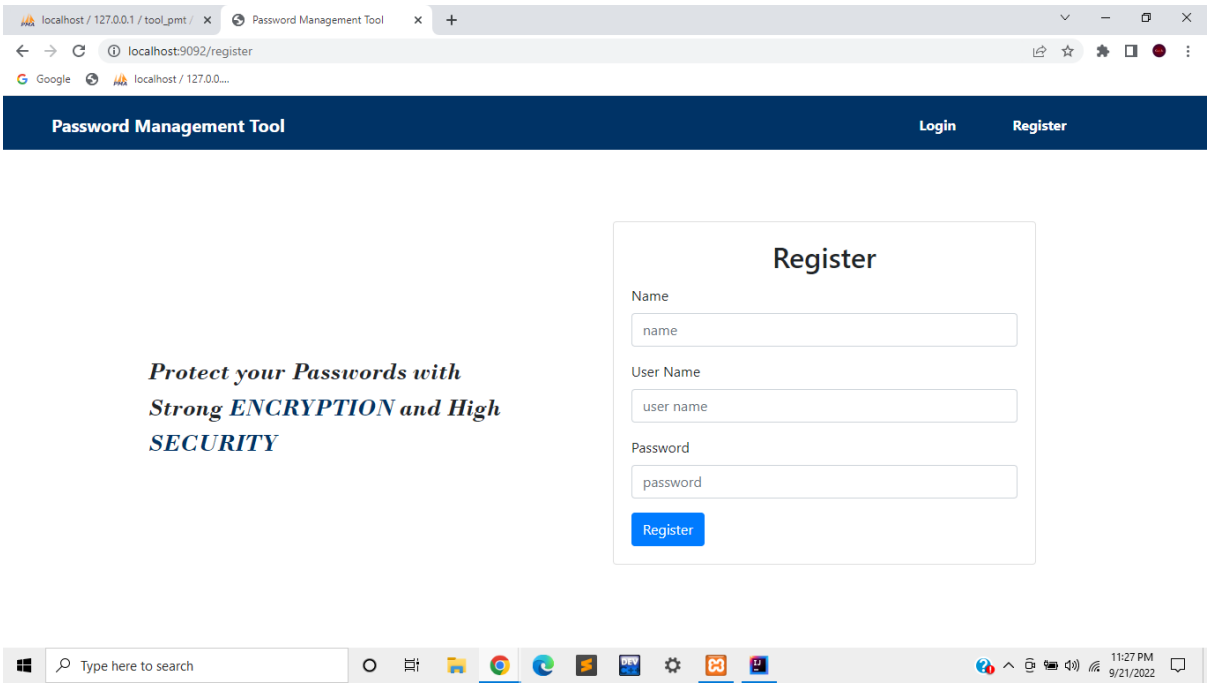
5.1 Password Management Tool Web Application

It is implemented by using spring boot, spring security, MySQL, jsp, Junit, JDBC, Hybernate, Maven. We use the necessary components to implement this. With jsp we implemented front end for our application. By using spring security we added more security features to the web application because spring recommends to use spring security.

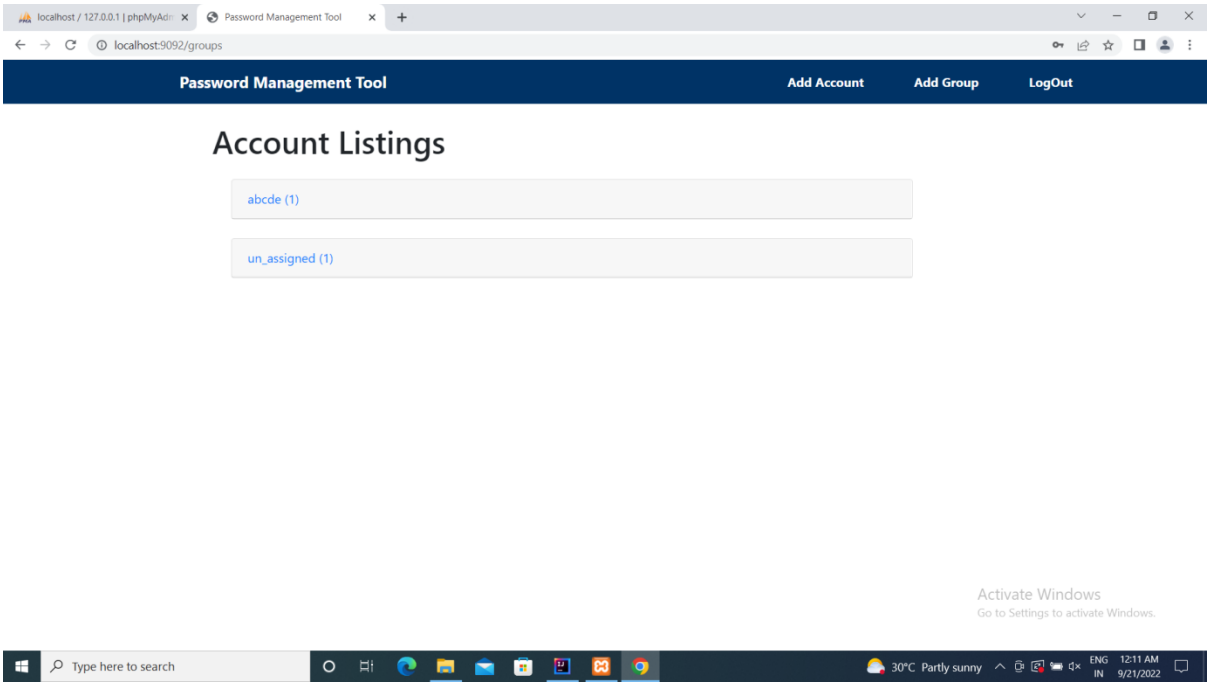
5.1.1 Output of Login page



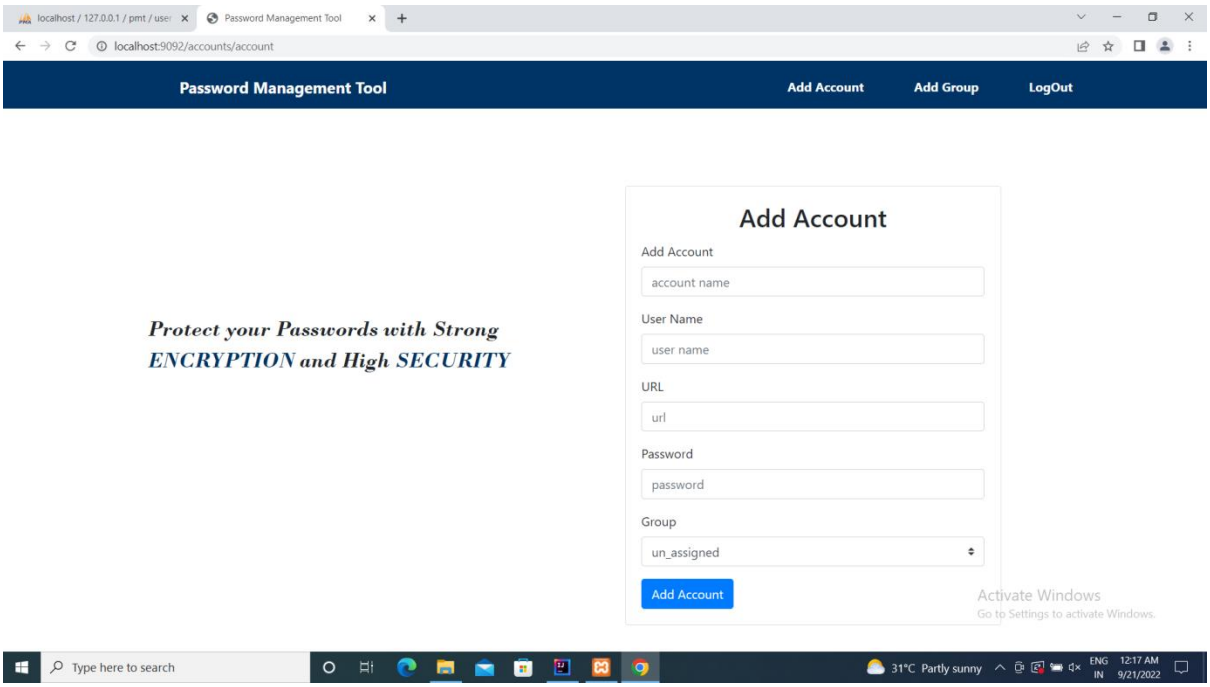
5.1.2 Output of register page



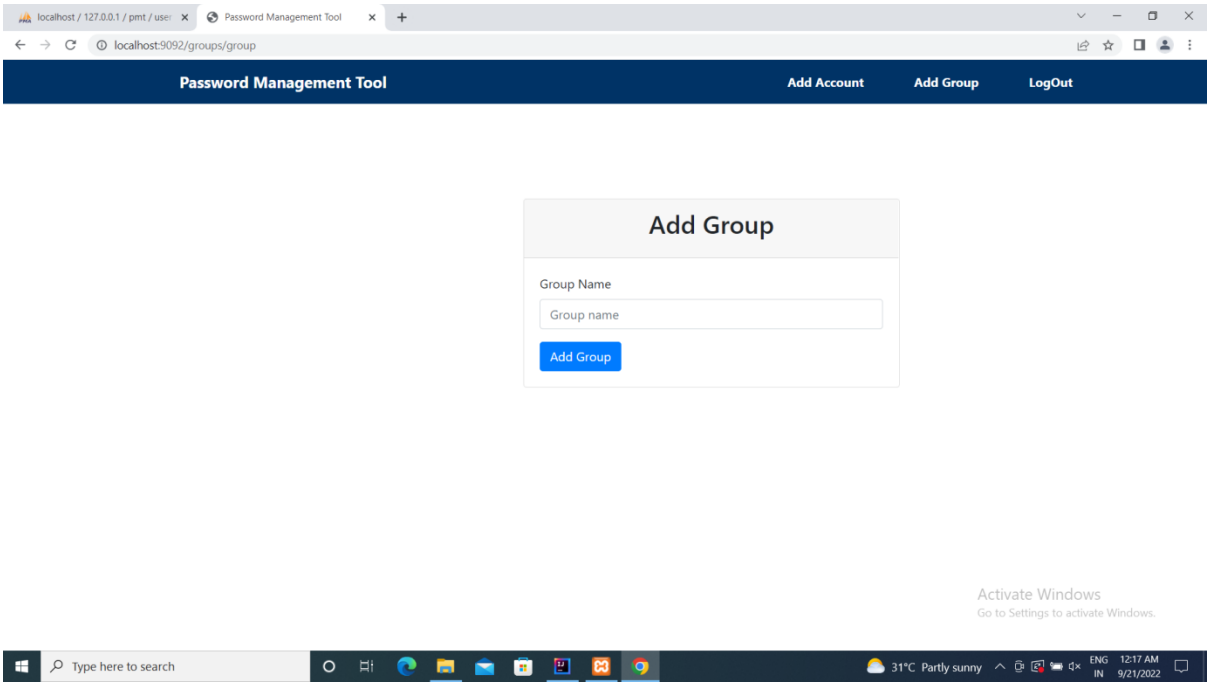
5.1.3 Home Page



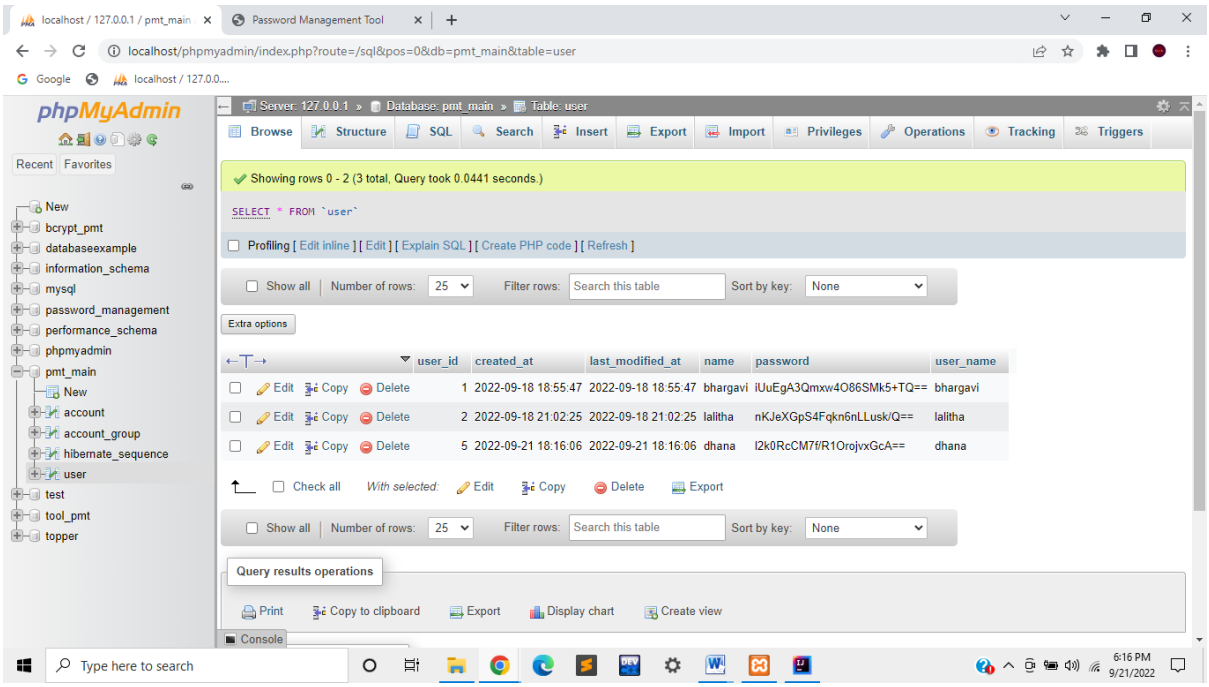
5.1.4 Add account Page



5.1.5 Add Group page



5.1.6 Database Maintainance



CHAPTER – 6

SOURCE CODE

6.1 Spring Boot Main Application Class

```
package com.password.tool;

import org.modelmapper.ModelMapper;
import org.modelmapper.convention.MatchingStrategies;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
@SpringBootApplication
public class PasswordManagementToolApplication {

   public static void main(String[] args) {
      SpringApplication.run(PasswordManagementToolApplication.class, args);
   }

   @Bean
   public ModelMapper getModelMapper(){
      ModelMapper modelMapper = new ModelMapper();

modelMapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRIC
T);
      return modelMapper;
   }

}
```

6.2 Main_Header.jsp File

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"
%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
      <!DOCTYPE html>
      <html>

      <head>
        <meta charset="ISO-8859-1">
        <link href="<c:url value=" /resources/bootstrap.css" />" rel="stylesheet">
        <link href="<c:url value=" /resources/style.css" />" rel="stylesheet">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.2/jquery.min.js" />
```

```
        <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"><
/script>
        <title>Password Management Tool</title>
        <style>

                .navbar-custom {
                 background-color:#003366;
                }
                .head1
                {
                   font-weight:bold;

                }
                .sidehead{
                   font-weight:700;
                   font-size:27px;
                   padding-left:12%;
                   padding-top:17%;
                   font-family:Bodoni MT;
                   font-style:italic;
                   }
                   .body1{
                     display:flex;
                   }
                   .changecolor{
                      color:#003366;
                   }
                   .container2{
                   margin-right:42px;
                   padding-top:5px;
                   }
             </style>
      </head>

      <body>
        <nav class="navbar navbar-expand-lg navbar-dark navbar-custom">
          <div class="container container2">
            <a class="navbar-brand head1" href="#">Password Management
Tool</a>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
              <ul class="navbar-nav ml-auto">
                <c:if test="${pageContext.request.userPrincipal.name == null}">
                  <li class="nav-item active"><a class="nav-link"
                      href="${pageContext.request.contextPath}/login">Login
                    </a></li>
                  <li class="nav-item active"><a class="nav-link"
```

```
href="${pageContext.request.contextPath}/register">Register</a>
                    </li>
                  </c:if>
                  <c:if test="${pageContext.request.userPrincipal.name != null}">
                    <%--<li class="nav-item active"><a class="nav-link"
                        href="${pageContext.request.contextPath}/accounts">View
Accounts
                      </a></li>--%>
                    <li class="nav-item active"><a class="nav-link"

href="${pageContext.request.contextPath}/accounts/account">Add Account
                      </a></li>

                    <li class="nav-item active"><a class="nav-link"
                        href="${pageContext.request.contextPath}/groups/group">Add
Group</a>
                    </li>

                    <li class="nav-item active"><a class="nav-link"

href="${pageContext.request.contextPath}/perform_logout">LogOut</a>
                    </li>
                  </c:if>
                </ul>
              </div>
            </div>
          </nav>
          <div class="body1">
            <div class="sidehead">Protect your Passwords with Strong <span
class="changecolor">ENCRYPTION</span> and High <span
class="changecolor">SECURITY<span> </div>
```

6.3 Login .jsp File

```
<%@include file="header.jsp"%>
<div class="container">
    <div class="card mx-auto vcenter ">
      <div class="card-header">
        <h2 class="card-title justify-content-
center"><center>Login</center></h2>
      </div>
      <div class="card-body">
        <c:if test="${param.error != null}">
          <div class="alert alert-danger" role="alert">Invalid Username /
Password</div>
        </c:if>

        <form:form action="perform_login" method="post">
          <div class="form-group">
```

```
            <label for="exampleInputEmail1">Username</label>
            <input
              name="username" required type="text" class="form-control"
              id="exampleInputEmail1" aria-describedby="emailHelp"
              placeholder="Enter username">
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Password</label>
            <input
              required name="password" type="password" class="form-control"
              id="exampleInputPassword1" placeholder="Password">
          </div>
          <button type="submit" class="btn btn-primary justify-content-
center">Login</button>
        </form:form>
      </div>

      <div class="card-footer">
        <div class="d-flex justify-content-center links">
          Don't have an account?<a
            href="${pageContext.request.contextPath}/register">Sign Up</a>
        </div>
      </div>

    </div>
  </div>

</body>
</html>
```

6.4 Register.jsp File

```
<%@include file="header.jsp"%>
  <div class="container">
    <div class="card mx-auto vcenter ">
      <div class="card-body">
        <h2 class="card-title"><center>Register</center></h2>
        <c:if test="${errorMessage != null}">
          <div class="alert alert-danger" role="alert">${errorMessage}</div>
        </c:if>
        <form:form modelAttribute="userBean" method="post">
          <div class="form-group">
            <label for="exampleInputEmail1">Name</label>
            <input type="text" name="name" class="form-control" required
placeholder="name" value="${userBean.name}">
            <form:errors cssClass="has-error" path="name" />
          </div>
          <div class="form-group">
            <label for="exampleInputEmail1">User Name</label>
```

```
            <input type="text" name="userName" class="form-control" required
placeholder="user name" value="${userBean.userName}">
            <form:errors cssClass="has-error" path="userName" />
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Password</label>
            <input type="password" name="password" class="form-control"
required placeholder="password" value="${userBean.password}">
            <form:errors cssClass="has-error" path="password" />
          </div>
          <button type="submit" class="btn btn-primary">Register</button>
        </form:form>
      </div>

    </div>
  </div>

</body>
</html>
```

6.5 Add acoount jsp File

```
<%@include file="header.jsp"%>
  <div class="container">
    <div class="card mx-auto vcenter ">
      <div class="card-body">
         <h2 class="card-title"><center>${head}</center></h2>
        <c:if test="${errorMessage != null}">
          <div class="alert alert-danger" role="alert">${errorMessage}</div>
        </c:if>
        <form:form modelAttribute="accountBean" method="post">
          <div class="form-group">
            <label for="exampleInputEmail1">${head}</label>
            <input type="text" class="form-control" name="accountName"
required placeholder="account name" value="${accountBean.accountName}">
            <form:errors cssClass="has-error" path="accountName" />
          </div>
          <div class="form-group">
            <label for="exampleInputEmail1">User Name</label>
            <input type="text" class="form-control" name="userName" required
placeholder="user name" value="${accountBean.userName}">
            <form:errors cssClass="has-error" path="userName" />
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">URL</label>
            <input type="text" class="form-control" name="url" placeholder="url"
value="${accountBean.url}">
            <form:errors cssClass="has-error" path="url" />
```

```
          </div>
             <div class="form-group">
            <label for="exampleInputPassword1">Password</label>
            <input type="password" class="form-control" name="password"
required placeholder="password" value="${accountBean.password}">
            <form:errors cssClass="has-error" path="password" />
          </div>
          <div class="form-group">
            <label for="exampleInputPassword1">Group</label>
            <form:select cssClass="custom-select" path="group.groupId" >
              <form:option   value = "0" label="None"/>
              <form:options  items="${groupDropDown}" itemValue="groupId" />
            </form:select>

          </div>
          <button type="submit" class="btn btn-primary">${head}</button>
        </form:form>
      </div>
    </div>
  </div>

</body>
</html>
```

6.6 Add Group jsp file

```
<%@include file="header.jsp"%>
  <div class="container">
    <div class="card mx-auto vcenter ">
      <div class="card-header">
        <h2 class="card-title"><center>${head}</center></h2>
      </div>
      <div class="card-body">
        <c:if test="${errorMessage != null}">
          <div class="alert alert-danger" role="alert">${errorMessage}</div>
        </c:if>
        <form:form modelAttribute="groupBean" method="post">
          <div class="form-group">
            <label for="exampleInputEmail1">Group Name</label>
            <input type="text" name="groupName" class="form-control" required
placeholder="Group name" value="${groupBean.groupName}">
            <form:errors cssClass="has-error" path="groupName" />
          </div>

          <button type="submit" class="btn btn-primary">${head}</button>
        </form:form>
      </div>
    </div>
  </div>
```

```
</body>
</html>
```

6.7 Spring Security Configurations Class

```
package com.password.tool.config;

import com.password.tool.service.interfaces.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authenti
cationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSe
curity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurity
ConfigurerAdapter;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {
   @Autowired
   UserService userService;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws
Exception {


auth.userDetailsService(userService).passwordEncoder(passwordEncoder());

   }

   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http.csrf().disable().authorizeRequests()
            .antMatchers("/anonymous*").anonymous()
            .antMatchers("/login*").permitAll()
            .antMatchers("/resources/**").permitAll()
            .antMatchers("/register*").permitAll()
```

```
            .anyRequest()
            .authenticated().and().httpBasic()
            .and().formLogin().loginPage("/login")
            .loginProcessingUrl("/perform_login")
            .defaultSuccessUrl("/groups", true)
            .and()
            .logout()
            .logoutUrl("/perform_logout")
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID")
            .logoutSuccessUrl("/login");
    }

    @Bean
    public PasswordEncoder passwordEncoder() {

       return NoOpPasswordEncoder.getInstance();
    }
}
```

6.8 Constants java class to check Validations for password and username

```
package com.password.tool.config;

public final class Constants {

    private Constants(){
    }

    public static final String PASSWORD_REGEX = "^(?=.*[0-9])"
            + "(?=.*[a-z])(?=.*[A-Z])"
            + "(?=.*[@#$%^&+=])"
            + "(?=\\S+$).{8,}$";

    public static  final String URL_REGEX = "(((http|https)://)(www.)?"
            + "[a-zA-Z0-9@:%._\\+~#?&//=]"
            + "{2,256}\\.[a-z]"
            + "{2,6}\\b([-a-zA-Z0-9@:%"
            + "._\\+~#?&//=]*)|(^$))";
    public static final String ACCOUNT_GROUP_NAME_REGEX = "^[a-z_]*$";

}
```

6.9 User.java Class to store user details in database

```
package com.password.tool.model;

import com.password.tool.config.AttributeEncryptor;
```

```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "user")
@Getter
@Setter
@NoArgsConstructor
public class User {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private int userId;

   @Column(name = "name")
   private String name;

   @Column(name = "user_name")
   private String userName;

   @Convert(converter = AttributeEncryptor.class)
   @Column(name = "password")
   private String password;

   @Column(name = "created_at")
   private Date createdAt;

   @Column(name = "last_modified_at")
   private Date lastModifiedAt;

   @OneToMany(mappedBy = "user")
   List<Account> accounts;

   @OneToMany(mappedBy = "user")
   List<Group> groups;

   public User(String name, String userName, String password) {
      this.name = name;
      this.userName = userName;
      this.password = password;
   }

}
```

6.10 UserDto.java class

```
package com.password.tool.dto;


import com.password.tool.config.Constants;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.stereotype.Component;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;
import java.util.Date;
import java.util.List;
import java.util.Objects;

@Getter
@Setter
@NoArgsConstructor
@Component("userBean")
public class UserDto {
   private int userId;

   @NotNull
   @Size(min = 5 , max = 20 , message = "Name should be of min 5 chars and max
20 chars")
   private String name;

   @NotNull
   @Size(min = 5 , max = 30 , message = "User Name should be of min 5 chars and
max 30 chars")
   private String userName;

   @NotNull
   @Pattern(regexp = Constants.PASSWORD_REGEX, message = "Password
Should Contain Lower case , Upper case , Numbers and Special chars with min of
8 chars")
   private String password;

   private Date createdAt;

   private Date lastModifiedAt;

   private List<AccountDto> accountDtos;
   private List<GroupDto> groupDtos;

   public UserDto(String name, String userName, String password) {
      this.name = name;
      this.userName = userName;
```

```
        this.password = password;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        UserDto userDto = (UserDto) o;
        return userId == userDto.userId;
    }

    @Override
    public int hashCode() {
        return Objects.hash(userId);
    }
}
```

6.11 Appplication.properties

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost/pmt_main
spring.datasource.username=root
spring.datasource.driver-class-name =com.mysql.cj.jdbc.Driver
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
server.port=9092
```

CHAPTER – 7

SYSTEM TESTING

7.1 INTRODUCTION:

The cause of testing is to detect mistakes. Making an attempt out is the technique of looking for to realize each viable fault or weakness in a piece product. It presents a method to determine the performance of add-ons, sub-assemblies, assemblies and/or a completed product. It is the method of ex excising g program with the intent of constructing certain that the application procedure meets its necessities and client expectations and does no longer fail in an unacceptable process. There are rather plenty of forms of scan. Each experiment sort addresses a special trying out requirement.

7.2 TYPES OF TESTS:

Unit Testing:

Unit checking out involves the design of scan circumstances that validate that the Internal application good judgment is functioning safely, and that program inputs produce legitimate outputs. All decision branches and interior code float must be validated. It's the checking out of character application items of the application . It is achieved after the completion of an person unit earlier than integration. It is a structural checking out, that relies on competencies of its construction and is invasive. Unit exams participate in common exams at component level and scan a distinct business approach, utility, and/or process configuration. Unit assessments be certain that every specified course of a industry method performs appropriately to the documented requisites and involves clearly outlined inputs and anticipated results.

Integration Testing:

Integration Testing are designed to scan built-in program accessories to determine within the occasion that they evidently run as one software. Trying out is occasion driven and is more concerned with the fundamental final result of screens or fields. Integration assessments reveal that despite the fact that the accessories had been for my part pleasure, as proven through effectively unit checking out, the combo of accessories is correct and regular. Integration checking out is chiefly aimed at exposing the issues that come up from the performance of different components.

Functional Testing:

Functional Testing checks provide systematic demonstrations that capabilities established are to be had as particular by means of the business and technical

specifications, method documentation, and consumer manuals. Functional testing is working on below mentioned data:

Legitimate input : identified lessons of legitimate input ought to be accredited.

Invalid ente : recognized lessons of unacceptable effort must be rejected.

Capabilities: recognized features ought to be exercised.

Output : recognized courses of software outputs have got to be exercised.

Systems/Procedures : performance of the system here was invoked

Individual and team work of useful checks is fascinated by specifications, key capabilities, or special scan instances. Moreover, systematic insurance plan concerning establish business method flows; data fields, predefined processes, and successive strategies have to be regarded for trying out. Before useful trying out is whole, extra checks are recognized and the strong price of present checks be strongminded.

System Testing:

An illustration of procedure testing is the configuration oriented approach integration scan. System testing is based on approach descriptions and flows, emphasizing pre-driven system links and integration aspects.

White Box Testing:

This testing is a trying out wherein where the application tester has competencies of the interior workings, constitution and software language, or at least its cause.

It's rationale. It's used to test areas that can't be reached from a black box stage.

Black Box Testing:

Black box testing is a type of software testing in which the functionality

of the software is not known. The testing is done without the internal knowledge of the products.

7.3 Levels Of Testing

Unit test strategy

Unit checking out is most commonly performed as a part of a mixed code and unit experiment part of the software lifecycle, though it be not exceptional for coding and unit checking out be performed as two targeted phases Test strategy and approach:

Field testing out can be carried out manually and sensible assessments

shall be written in element.

Test objectives

Each field must be work correctly.

Each page must be activated through the specified link.

Features to be tested Verify that the entries are of the correct format No duplicate entries should be allowed

Integration testing strategy

Software integration testing is the incremental integration checking out of two otherwise further included software gears on top of a solo stage to fabricate failure induced with the aid of interface defects. The project of the mixing scan is to check that components or program applications,
e.g. Components in a program approach or œ one step up œ software purposes at the company degree œ interact without error.

Test Results:

All of the scan circumstances recounted above passed efficiently. No defects encountered.

Acceptance Testing

User Acceptance testing trying out is a crucial section of any mission and requires enormous participation by the tip user. It additionally ensures that the procedure meets the functional specifications.

Test Results:

The entire test cases recounted above passed effectually. No defects Encountered

CONCLUSION

This project Password Management Tool was successfully executed by our team to check the end results using a database and it is found that our proposed system provides efficient and effective result when compared to existing system. It is a web application to store our passwords securely and it is implemented through jsp, spring boot and spring security. It also provides best user experience and user interface. This software is purely for solving the problems related to the security.

FUTURE ENHANCEMENT

With the goal to provide a web application to store passwords securely. It is implemented using technologies spring boot , spring security and jsp.Spring boot of capable of creating applications with its auto configuration and several other features . Being such an amazing tool, its attracts a lot of developers for different application development purposes.

Based on our project concept , it can be further extended to make an android application by adding Ionic framework. Ionic is an open source framework designed to help you build mobile applications with web technologies. It can be further developed by changing encryption algorithm and providing high security.

APPENDIX

REFERENCES

https://spring.io/projects/spring-boot

https://www.javatpoint.com/jsp-tutorial

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Design and Implementation of password Management System (IEEE)

https://ieeexplore.ieee.org/document/5385014