SYSTEM REQUIREMENTS SPECIFICATION

FOR

EXPENSIVE TRACKING SYSTEM

DONE BY:

CB.EN.U4CSE15236 N Sai Rahul

CB.EN.U4CSE15233 M RAJA

CB.EN.U4CSE15231 M PRANAY

CB.EN.U4CSE15249 M SATHVIK

CB.EN.U4CSE15243 R SAKETH

**1.INTRODUCTION**

**1.1. PURPOSE**

The main purpose of this application is to track all the expenses made by a user on a day to day basis.The user can generate report of his expenses based on the transaction date of the expenses or category of items

**1.2 SCOPE**

The name of the software is "EXPENSIVE TRACKING SYSTEM" which provides the ability to group our HomeBudget supports tracking of both your expenses and income, and includes support for budgets, accounts, payees and bill tracking. HomeBudget provides the ability to group your expenses into categories and sub-categories, and lets you set a budget, and track expenses at the category/sub-category level.

**1.3 OVERVIEW**

The rest of the document deals about all the main features of this software and its main fuctions.It also gives the details abput the interface with other products and ralated functionality of each product.

**2. OVERALL DESCRIPTION**

**2.1 PRODUCT PERSPECTIVE**

The “EXPENSIVE TRACKING SYSTEM” software is an independent application. It is a self-contained product. The system interfaces and user interfaces related with this software are defined as follows.

2**.1.1 System Interfaces**

The client systems should be able to share the data available in the data base through the network connection.

**2.2.2 User Interfaces**

The screen formats and menu structure should be in such a way that even users will find it easy to use. The product must be use-friendly and very inter-active. The functionality provided by the system like displaying error messages should adapt i- tself to the different users of the software.

**2.1.3 Software Interfaces Name of the language:**

Net Beans, Eclipse

**2.1.4 Operation**

The users can give the details of the expenses made by them for a particular date by categeory wise.The system provides the customer with a notification which states that transation details has received and it gives acesss to the user to make any changes in the transation details.

**2.2 PRODUCT FUNCTIONS**

The major functions include, **User must have an account that is maintained by system.To keep track of money of the user and give the views on what the user spend his/her money. Any activities in the account must be kept and recorded the purchases made based on categories and must be reported to users.User can have his own categories** and he **can have reports on his previous purchases.**

**2.3 USER CHARACTERISTICS**

No technical experience is required basic knowledge of handling system is sufficient

**2.4 CONSTRAINTS**

 **Control functions:**

The software must be very user-friendly and display appropriate error messages.

 **Interfaces to other applications:**

Applicable..

 **Reliability requirements:**

Data redundancy and use of special/blank characters must be avoided.

**2.5 ASSUMPTIONS AND DEPENDENCIES**

 One user can have only one account.

 One category can have many items.

 Each user can define his own category list.

 In current implementation of this project one transaction contains only one item but the structure that we provide,is enabling us to do further extension if we want to apply one transaction contains many items. For the sake of simplicity and the time constraint we decided to implement the current condition.

 User can generate the expense report based on the transaction date or category type.

**3. Specific Requirements**

3.1 **Functional Requirements**

**Functional requirements describe the behaviors (functions or services) of the system that support the user’s goals, tasks or activities.**

**3.1.1 Register a new a Account for a new User:**

**Users need to create a new account before they can add balance and keep record of their purchase transactions.**

**3.1.2 Login User:**

**System has to make sure that the user is the user that have authorization to access the system so the user must be authenticated by login process.**

**3.1.3 Create Item Categeory:**

**Users can have their own item categories for items that they entered. The category is personalized for users for their own using.**

**3.1.4 Create Transactions:**

**User can Add item name, price, description and the purchase date in the transaction.**

**3.1.5 View Reports:**

**3.1.5.1       View all transaction by transaction date**

**3.1.5.2      View Transaction by category**

**3.2 Non Functional requirements**

**3.2.1 Security**

**3.2.1.1 User Authentication:**

To ensure the security of the user, the system needs to authenticate on every user before the user can access the system. The way to do this is to make sure that the user needs to login first. The user information will be saved in session.

**3.2.1.2 Encrypting password:**

For completely protecting the user’s information from outside or from the system’s administrator itself, the system encrypt the password when storing into the database. So even if the administrator of the system open and retrieve the data on the database, password information of the user is in encrypted format and cannot be stolen.

**DESIGN DOCUMENT**

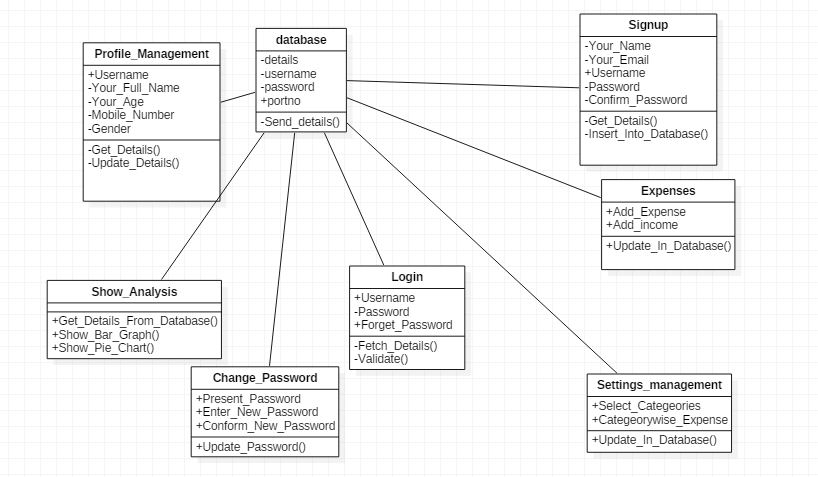
DATABASE DESIGN:

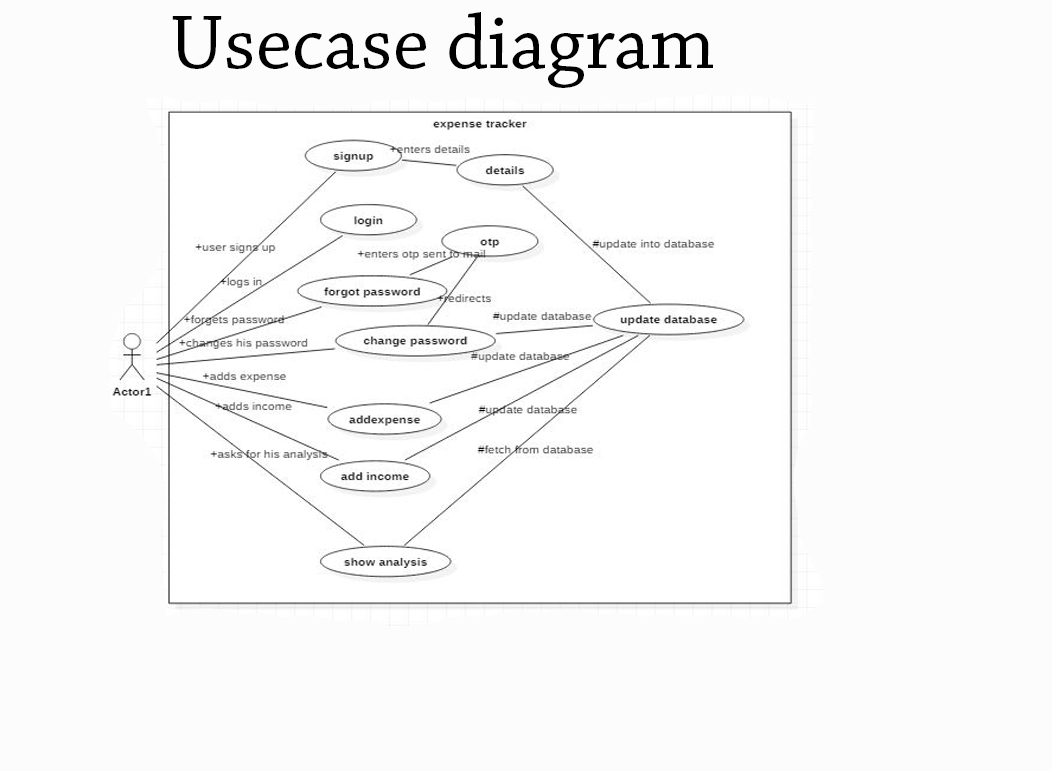
Weused oracle sql database (a real time data base) in our project this database uses sql language and for connecting between front end and back end we used ojdbc jar (oracle java data base connection) , this is a structured database and the information is stored in form of tables

FRONT END DESIGN :

For front end we used java & html and for styling we used css , for sending mail we used java mailer (a functionality provided by java ) ,for pie charts we used google api in java script , for validation of forms we used java script

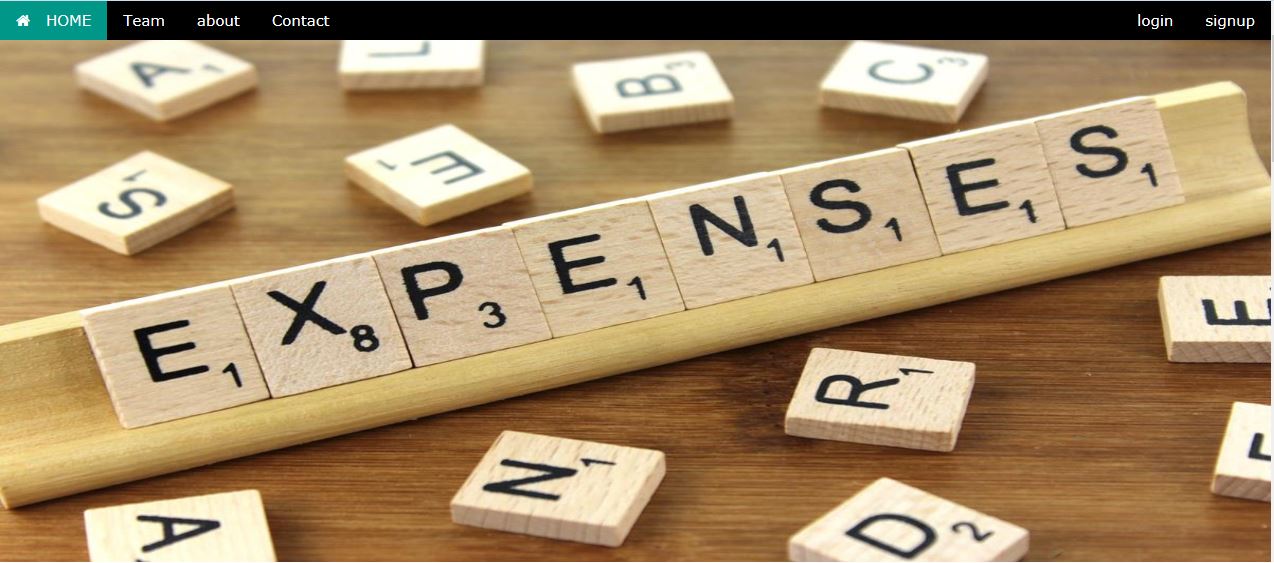
CLASS DIAGRAM



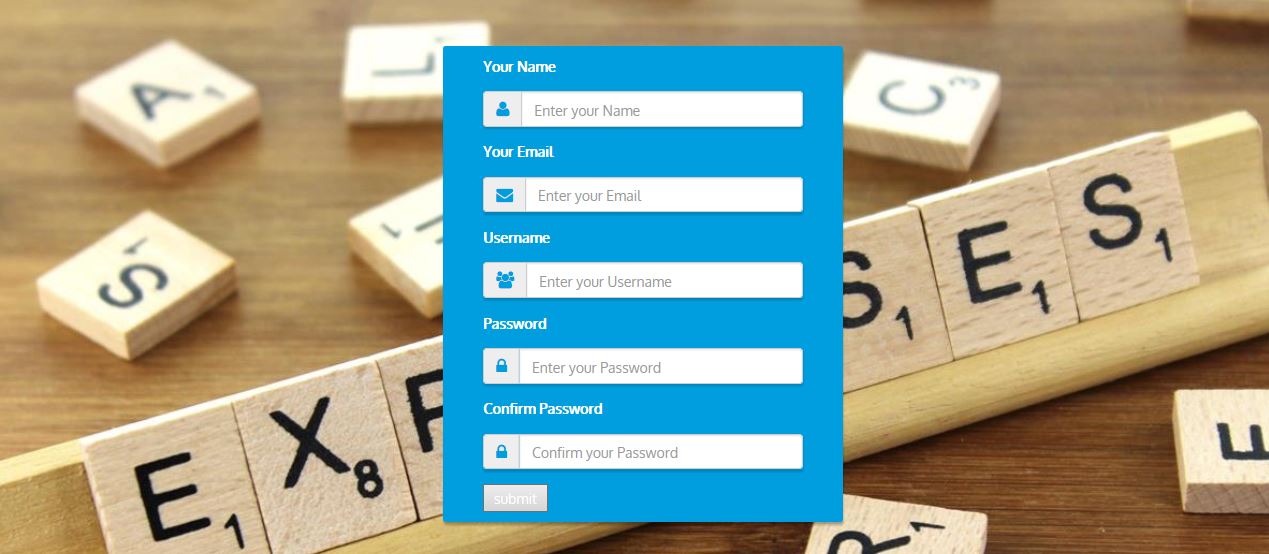


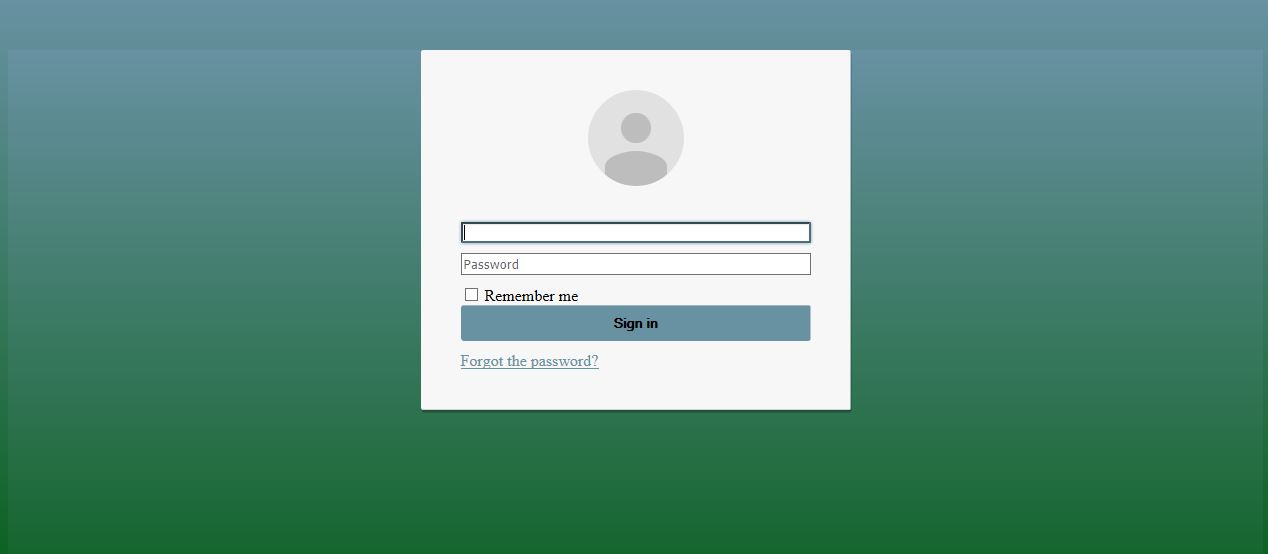
FRONT END:

HOME PAGE:



SIGNUP :

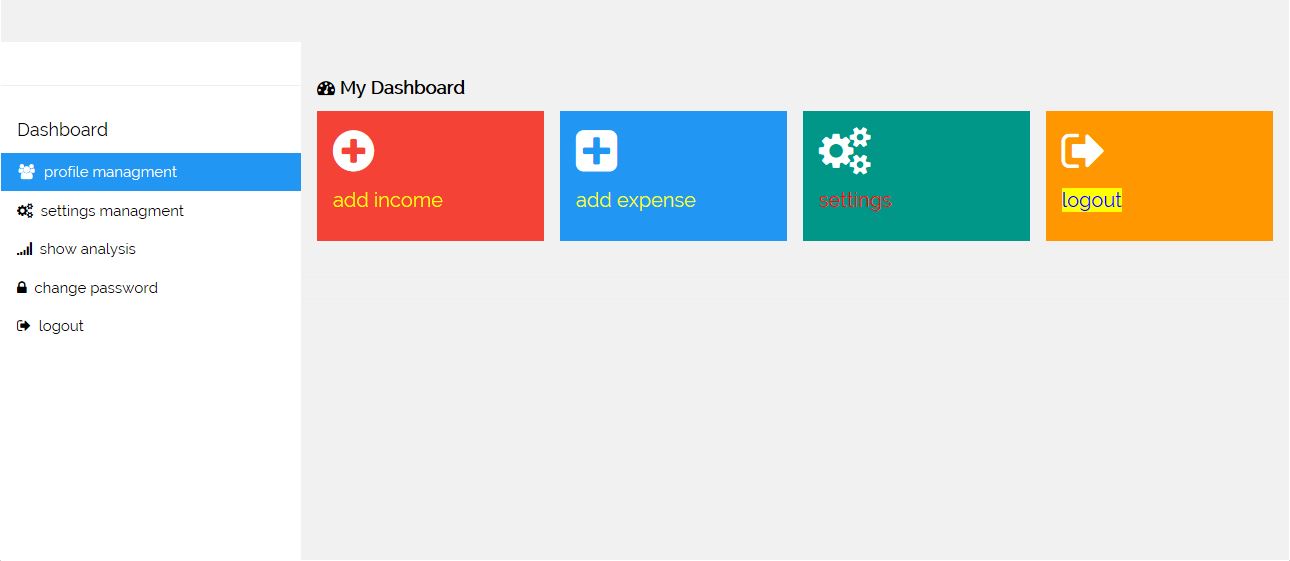


LOGIN:

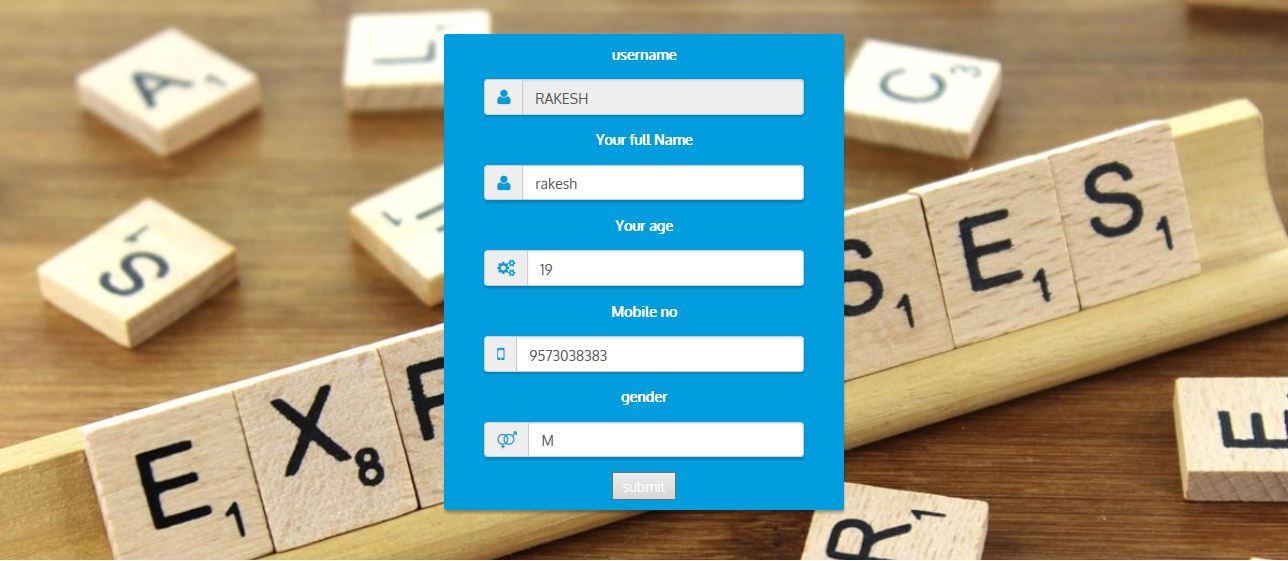
SHOW ANALYSIS:

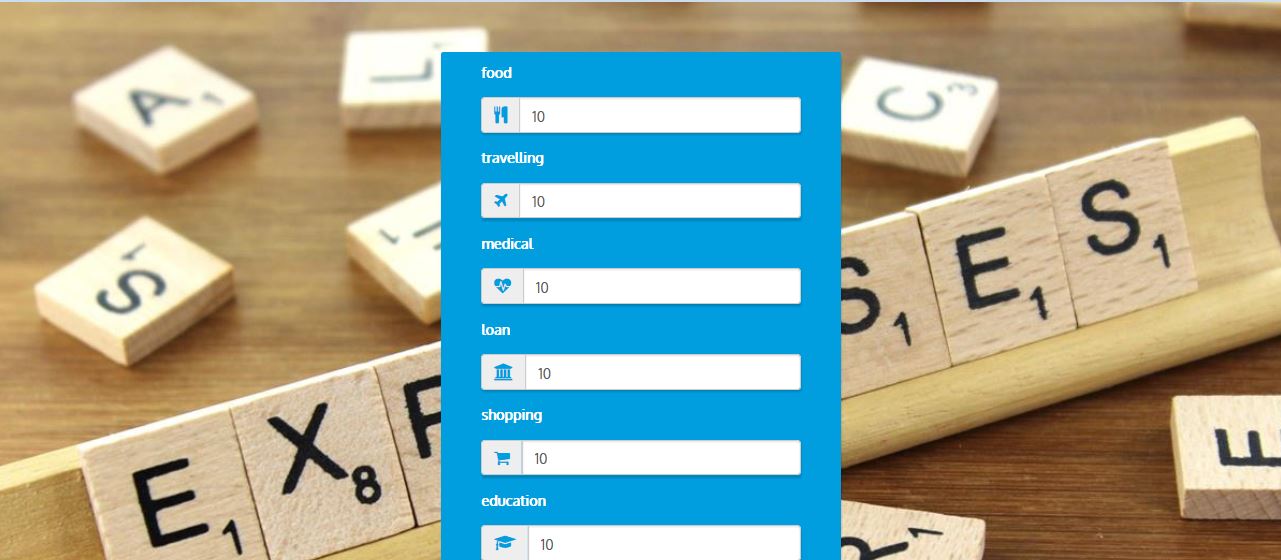


DASHBORD:



PROFILE MANAGEMENT:





DEVELOPMENT DOCUMENT

Login page:

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

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

<html>

<head>

<script >

$( document ).ready(**function**() {

// DOM ready

// Test data

/\*

\* To test the script you should discomment the function

\* testLocalStorageData and refresh the page. The function

\* will load some test data and the loadProfile

\* will do the changes in the UI

\*/

// testLocalStorageData();

// Load profile if it exits

loadProfile();

});

/\*\*

\* Function that gets the data of the profile in case

\* thar it has already saved in localstorage. Only the

\* UI will be update in case that all data is available

\*

\* A not existing key in localstorage return null

\*

\*/

**function** getLocalProfile(callback){

**var** profileImgSrc = localStorage.getItem("PROFILE\_IMG\_SRC");

**var** profileName = localStorage.getItem("PROFILE\_NAME");

**var** profileReAuthEmail = localStorage.getItem("PROFILE\_REAUTH\_EMAIL");

**if**(profileName !== **null**

&& profileReAuthEmail !== **null**

&& profileImgSrc !== **null**) {

callback(profileImgSrc, profileName, profileReAuthEmail);

}

}

/\*\*

\* Main function that load the profile if exists

\* in localstorage

\*/

**function** loadProfile() {

**if**(!supportsHTML5Storage()) { **return** **false**; }

// we have to provide to the callback the basic

// information to set the profile

getLocalProfile(**function**(profileImgSrc, profileName, profileReAuthEmail) {

//changes in the UI

$("#profile-img").attr("src",profileImgSrc);

$("#profile-name").html(profileName);

$("#reauth-email").html(profileReAuthEmail);

$("#inputEmail").hide();

$("#remember").hide();

});

window.history.forward();

window.history.forward();

}

/\*\*

\* function that checks if the browser supports HTML5

\* local storage

\*

\* @returns {boolean}

\*/

**function** supportsHTML5Storage() {

**try** {

**return** 'localStorage' **in** window && window['localStorage'] !== **null**;

} **catch** (e) {

**return** **false**;

}

}

/\*\*

\* Test data. This data will be safe by the web app

\* in the first successful login of a auth user.

\* To Test the scripts, delete the localstorage data

\* and comment this call.

\*

\* @returns {boolean}

\*/

**function** testLocalStorageData() {

**if**(!supportsHTML5Storage()) { **return** **false**; }

localStorage.setItem("PROFILE\_IMG\_SRC", "//lh3.googleusercontent.com/-6V8xOA6M7BA/AAAAAAAAAAI/AAAAAAAAAAA/rzlHcD0KYwo/photo.jpg?sz=120" );

localStorage.setItem("PROFILE\_NAME", "César Izquierdo Tello");

localStorage.setItem("PROFILE\_REAUTH\_EMAIL", "oneaccount@gmail.com");

}

</script>

<link rel=*"stylesheet"* type=*"text\css"* href=*"F:/expenseteacker/WebContent/login.css"*>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<link href=*"login.css"* rel=*"stylesheet"* id=*"bootstrap-css"*>

<script src=*"//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js/bootstrap.min.js"*></script>

<script src=*"//code.jquery.com/jquery-1.11.1.min.js"*></script>

<div class=*"container"*>

<div class=*"card card-container"*>

<img id=*"profile-img"* class=*"profile-img-card"* src=*"//ssl.gstatic.com/accounts/ui/avatar\_2x.png"* />

<p id=*"profile-name"* class=*"profile-name-card"*></p>

<form class=*"form-signin"* action =*"loginvalidation.jsp"*>

<span id=*"reauth-email"* class=*"reauth-email"*></span>

<input type=*"text"* id=*"username"* name=*"username"* class=*"form-control"* placeholder=*"username"* required autofocus>

<input type=*"Password"* id=*"Password"* name=*"Password"* class=*"form-control"* placeholder=*"Password"* required>

<div id=*"remember"* class=*"checkbox"*>

<label>

<input type=*"checkbox"* value=*"remember-me"*> Remember me

</label>

</div>

<button class=*"btn btn-lg btn-primary btn-block btn-signin"* type=*"submit"*>Sign in</button>

</form>

<a href=*"12212.jsp"* class=*"forgot-password"*>

Forgot the password?

</a>

</div>

</div>

</body>

</html>

Database connectivity :

<%@page import=*"java.io.IOException"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

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

<html>

<head>

</head>

<body bgcolor=*"green"*>

<center>

<%@page import=*"java.sql.\*,java.util.\*"*%>

<%

String l\_id = request.getParameter("username");

String l\_phone\_no = request.getParameter("Password");

String s=l\_phone\_no;

String k1 = **null**;

**char** c;

**for**(**int** i=0; i<s.length();i++){

c=s.charAt(i);

c=(**char**) (c+5);

k1=k1+c;

}

l\_phone\_no=k1;

System.err.println(l\_phone\_no);

String query="select \* from signup where username = '";

query=query+l\_id;

String k="'";

query=query+k;

String query2=" and pwd = '";

query=query+query2;

query=query+l\_phone\_no+k;

out.println(l\_id);

out.println(l\_phone\_no);

System.err.println(query);

**try**{

System.err.println(query);

session.setAttribute("username",l\_id);

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "rahul", "rahul");

PreparedStatement st = con.prepareStatement(query);

//st.setString(1, l\_id);

//st.setString(2, l\_phone\_no);

**int** l=0;

ResultSet i = st.executeQuery();

**while**(i.next()){

l=l+1;

}

**if** (l>0)

{

response.sendRedirect("cvc.jsp");

}

**else**

{

response.sendRedirect("error.jsp");

}

} **catch** (Exception e2) {

System.out.println(e2);

}

%>

</body>

</html>

Mailer code :

import java.util.Properties;

import javax.mail.\*;

import javax.mail.internet.\*;

class Mailer1{

public static void send(String from,String password,String to,String sub,String msg){

//Get properties object

Properties props = new Properties();

props.put("mail.smtp.host", "smtp.gmail.com");

props.put("mail.smtp.socketFactory.port", "465");

props.put("mail.smtp.socketFactory.class",

"javax.net.ssl.SSLSocketFactory");

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.port", "465");

//get Session

Session session = Session.getDefaultInstance(props,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(from,password);

}

});

//compose message

try {

MimeMessage message = new MimeMessage(session);

message.addRecipient(Message.RecipientType.TO,new InternetAddress(to));

message.setSubject(sub);

message.setText(msg);

//send message

Transport.send(message);

System.out.println("message sent successfully");

} catch (MessagingException e) {throw new RuntimeException(e);}

}

}

public class SendMailSSL{

public static void main(String[] args) {

//from,password,to,subject,message

Mailer1.send("rahul.nagubandi@gmail.com","raahul999","rqhul38@gmail.com","software project","How r u?");

//change from, password and to

}

}

SERVELET CODE :

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* version=*"3.1"*>

<display-name>expenseteacker</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>SendMail1</servlet-name>

<servlet-class>SendMail1</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>SendMail1</servlet-name>

<url-pattern>/SendMail1</url-pattern>

</servlet-mapping>

</web-app>

Testing Document