



---

# ONLINE TEST WEB APPLICATION

---

DXC TASK REPORT



PREPARED FOR:  
DXC  
PREPARED BY:  
B. LOKESH  
N. MANOJ  
P. L. POOJITHA  
V. B. S. PRAHASITH

## Table of Contents

<b>S.No</b>	<b>Name</b>	<b>Page No</b>
1	Introduction	2
2	Project description	3
3	Design and implementation	13

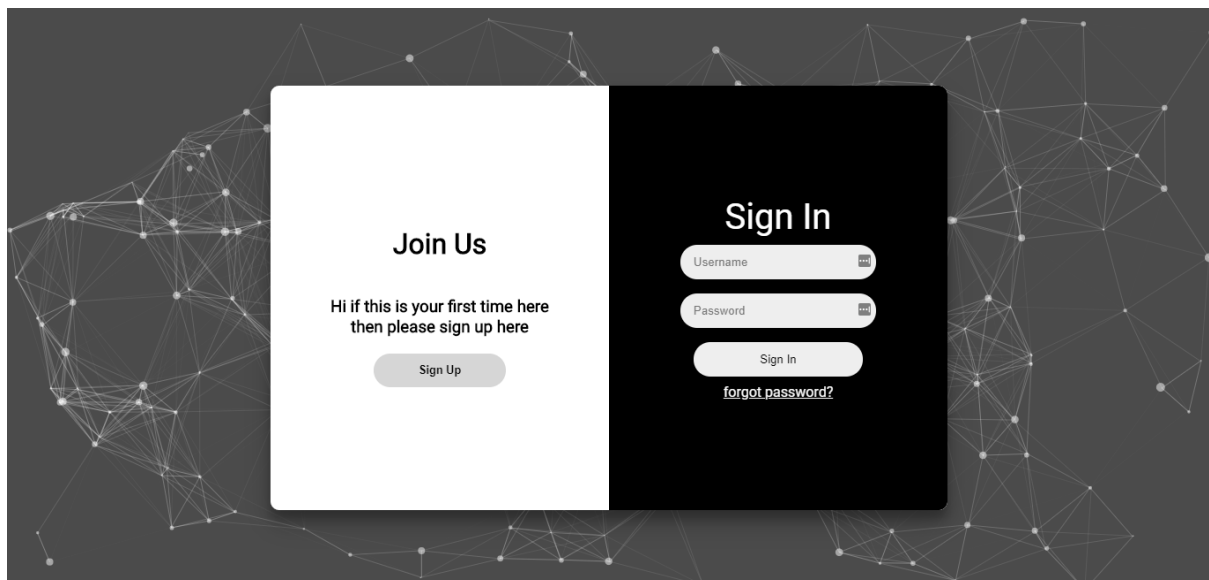
## INTRODUCTION

Our project is an Online Test Web Application. It is a software application which allows a particular company or institute to conduct and manage examinations online. The user has to register and login to use our page. Once the user logs in he is greeted by the home page where he can attempt a new test or see all his previous scores. We have created three versions of test page from which the test giver can choose from. There are presently two tests available. After attempting the test the user gets his results immediately. If he/she satisfies the passing criteria (70%) only then can he/she print their certificate.

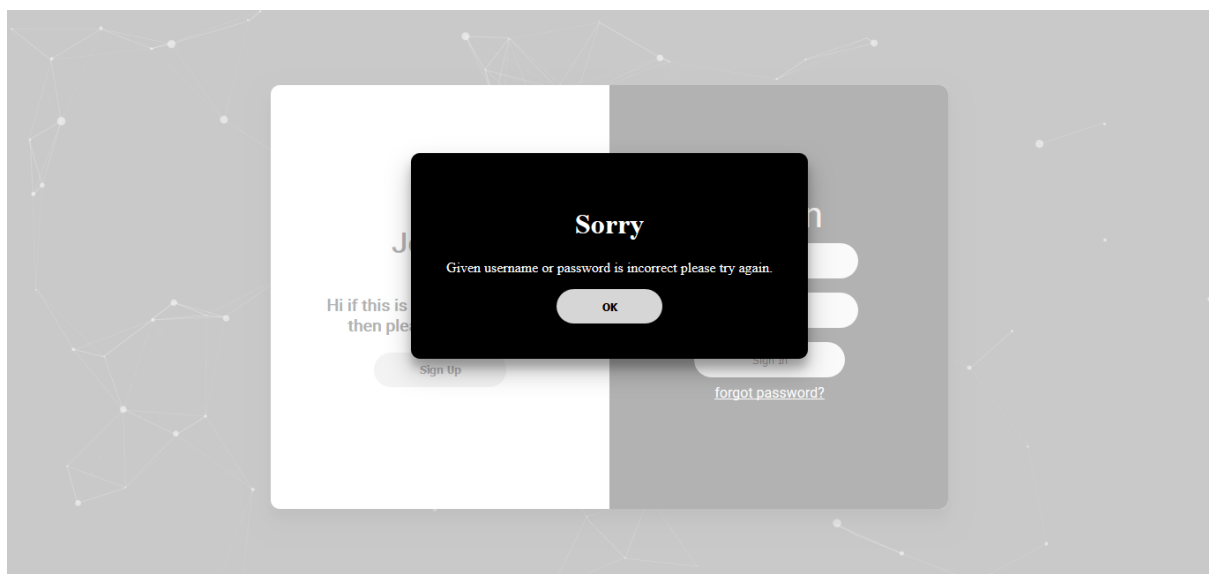
## PROJECT DESCRIPTION

### The Sign-in/Sign-up Page:

This is the page that greets the user. If the user is already registered he can sign-in on the right side of the page. If the username or password is incorrect we get a try again prompt. If the user forgets his password we have a “forgot-password?” option to reset the password which takes us to reset page. If he/she is new user he/she has to sign-up first. We can transition to sign-up form in the same page by clicking the Sign-up button on the left side of the page.

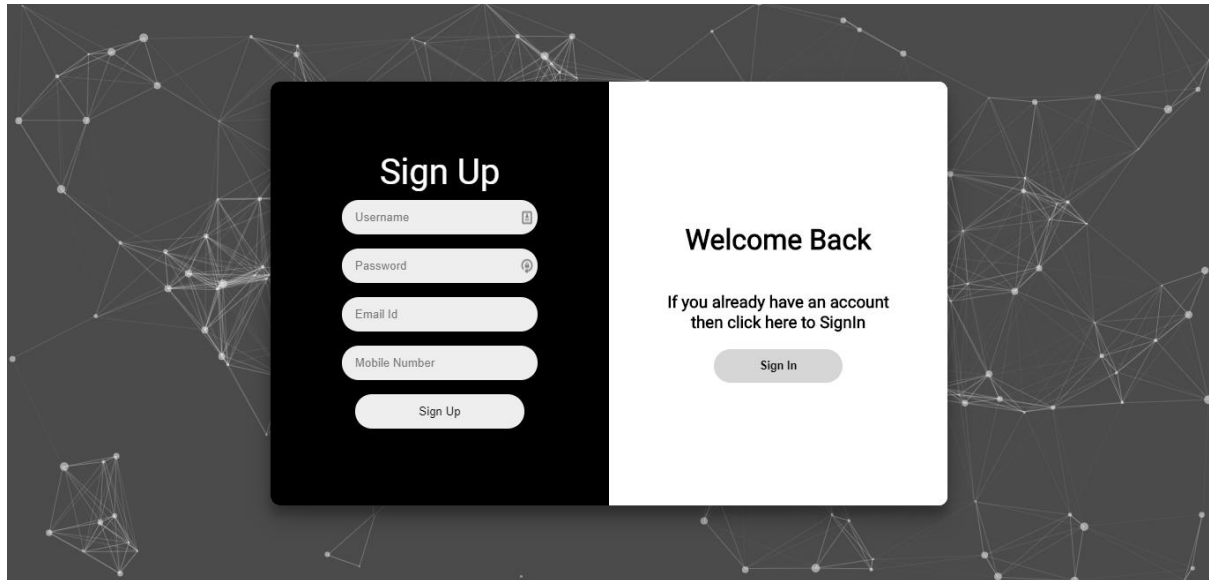


Try again prompt



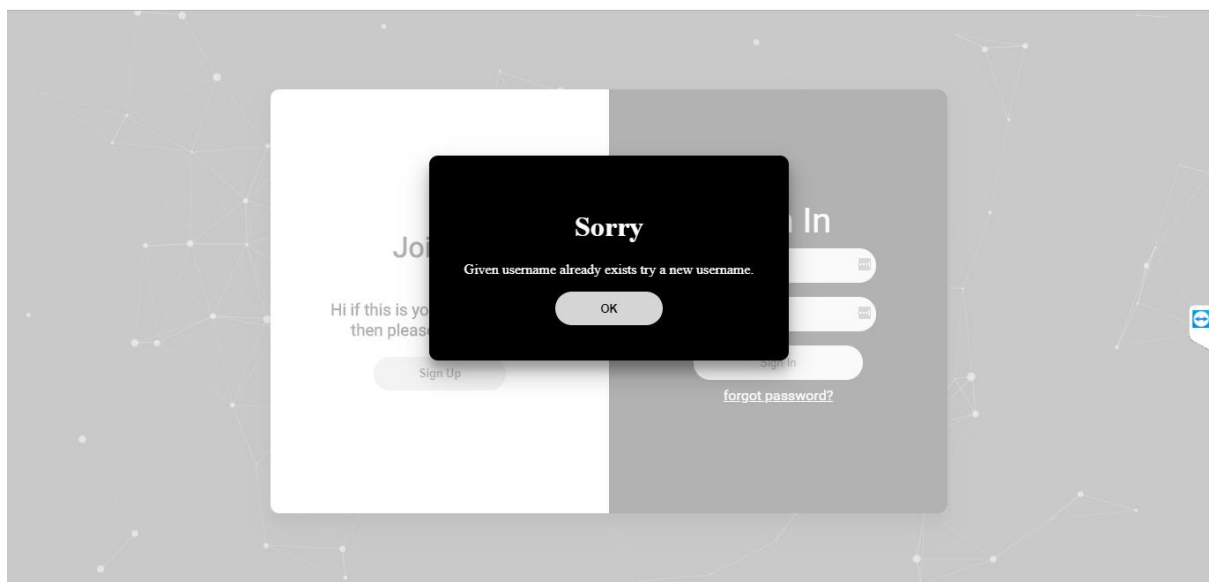
## Online Test Web Application

In this page the user can create an account by filling the form. The user name has to be unique. If username already exists we get a try again prompt. If the user wants to return to Sign-in page he can transition back by clicking Sign-in button on right side.



The screenshot shows a dark-themed web application interface. On the left, a black box contains the 'Sign Up' section with four input fields: 'Username', 'Password', 'Email Id', and 'Mobile Number', each with a corresponding icon. Below these fields is a 'Sign Up' button. On the right, a white box contains the 'Welcome Back' section with the text 'If you already have an account then click here to SignIn' and a 'Sign In' button. The background features a dark gray pattern of interconnected white dots and lines.

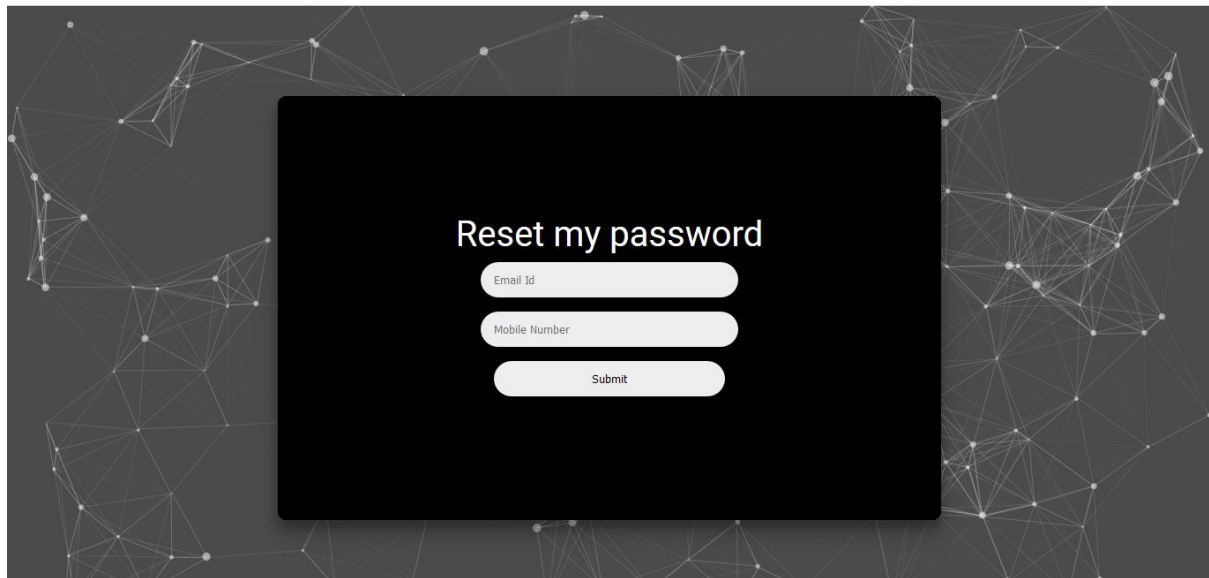
### User name already exist prompt



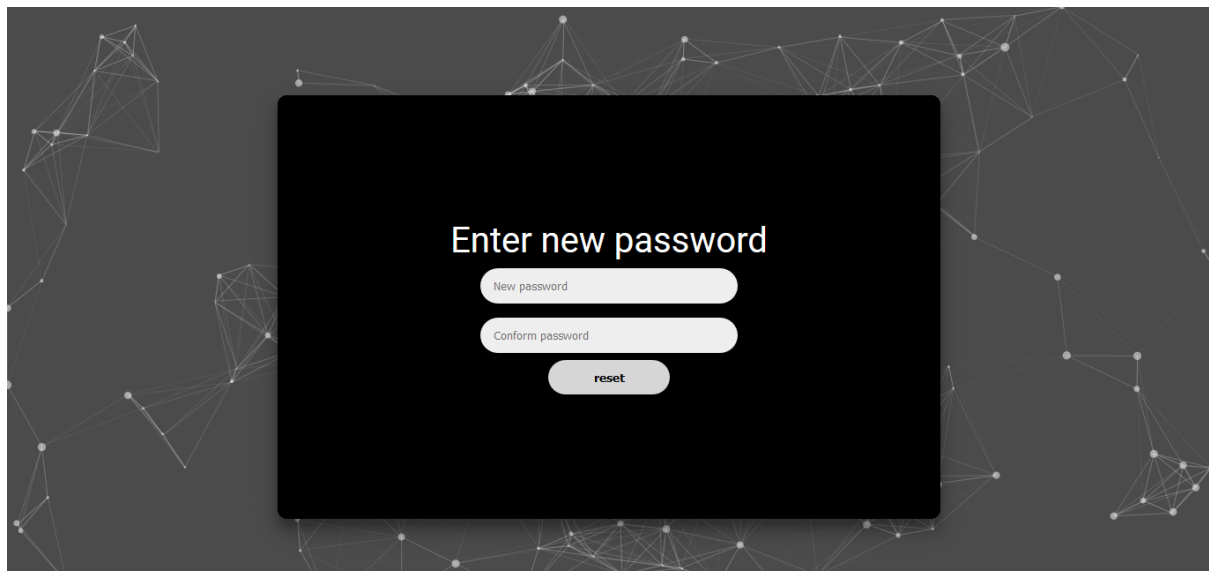
The screenshot shows the same 'Sign Up' and 'Welcome Back' section as the previous image, but with a modal prompt overlaid in the center. The modal is black with white text that reads 'Sorry' and 'Given username already exists try a new username.' Below this text is an 'OK' button. The background is a light gray pattern of interconnected white dots and lines.

[Reset Password page:](#)

If the user forgets their password, they can use their registered email and mobile number to get the new password prompt, which will assist him to reset his password.

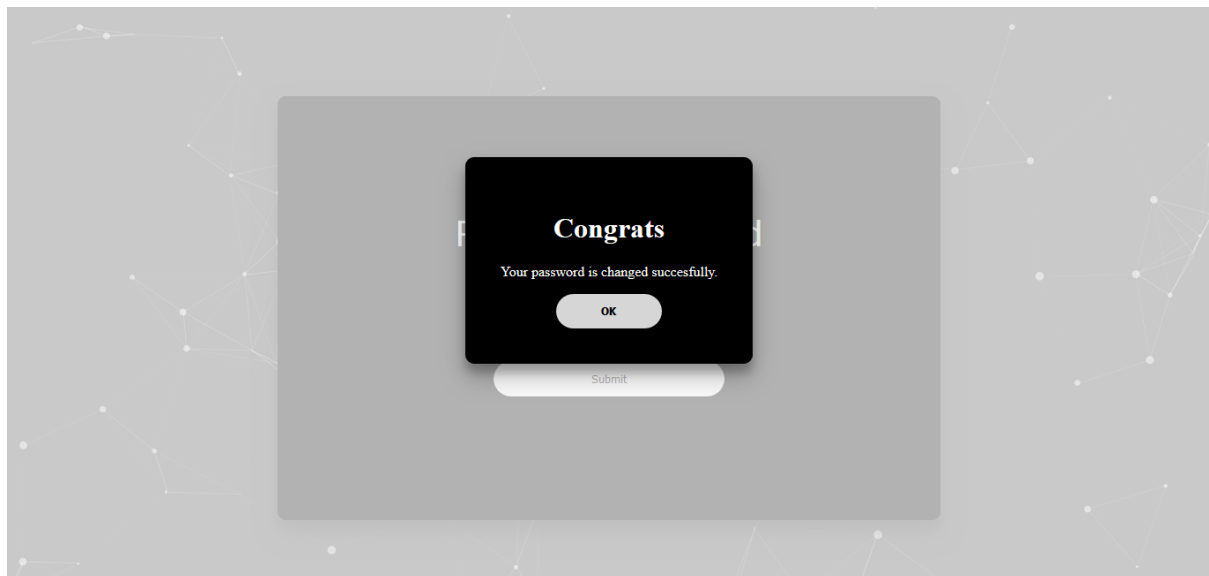
A screenshot of a web application interface for resetting a password. The background is dark gray with a white geometric pattern of dots and lines. In the center is a black rectangular box with white text and input fields. The text 'Reset my password' is at the top. Below it are three white input fields with rounded ends, labeled 'Email Id', 'Mobile Number', and 'Submit'.

New password prompt

A screenshot of a web application interface for entering a new password. The background is dark gray with a white geometric pattern of dots and lines. In the center is a black rectangular box with white text and input fields. The text 'Enter new password' is at the top. Below it are two white input fields with rounded ends, labeled 'New password' and 'Conform password'. At the bottom is a white button with rounded ends labeled 'reset'.

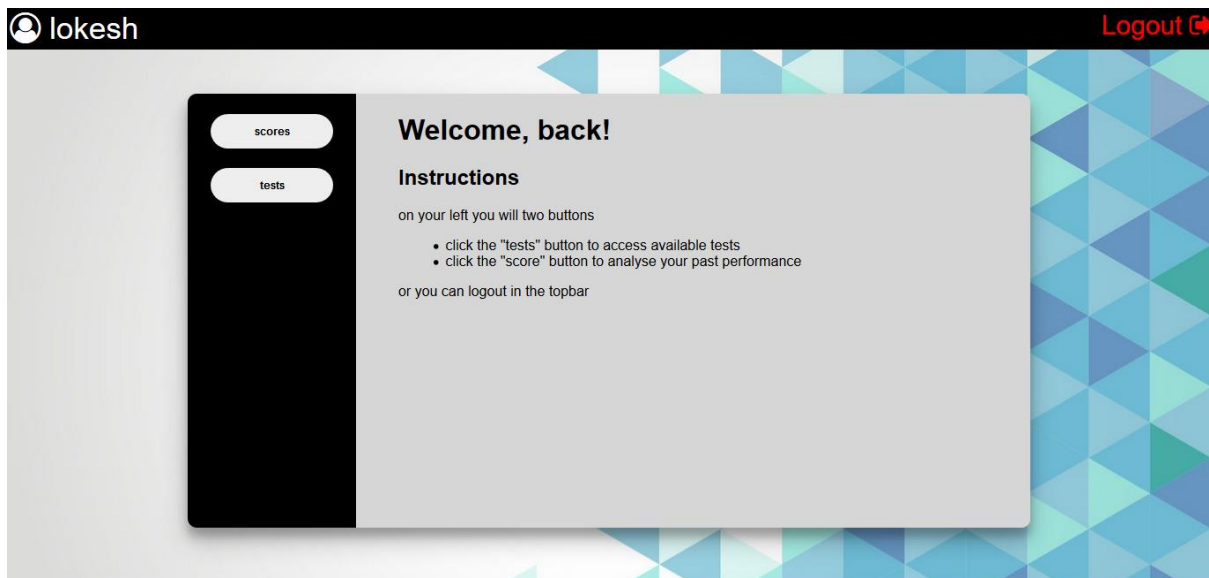
If both the password are same the password will be reset.

### Conformation prompt

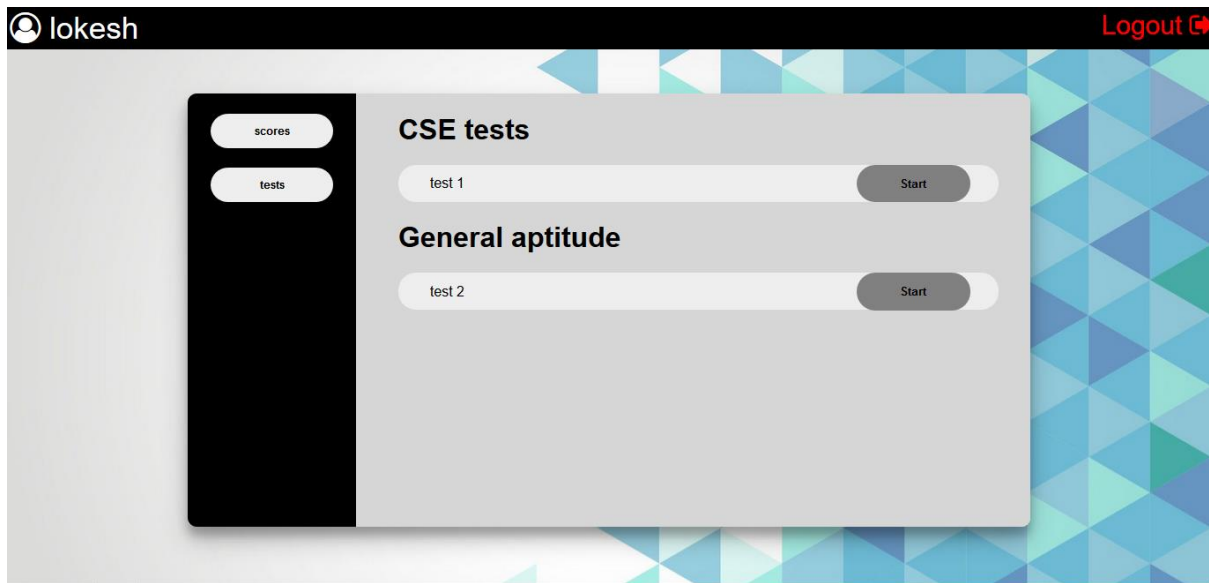


### Home Page:

The user lands in this page after Sign-in. On the top left side we have username of user logged in. On top right we have the logout button. This page offers the user the choice to attempt a test or see their previous scores.

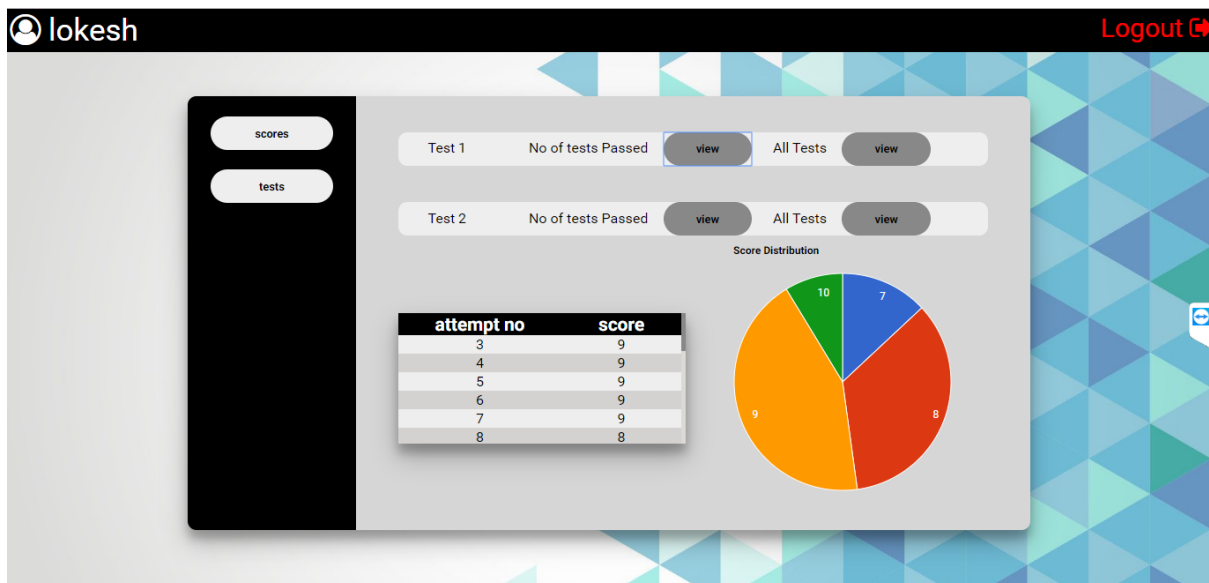


## Test selection



For now we have two tests to choose from.

## Previous scores



The user can choose to see scores of all attempts or scores of only those attempts he/she passed of any test.

The score is represented in two formats a table and a pie-chart.

In the table we have the attempt no and corresponding scores in sequential order.

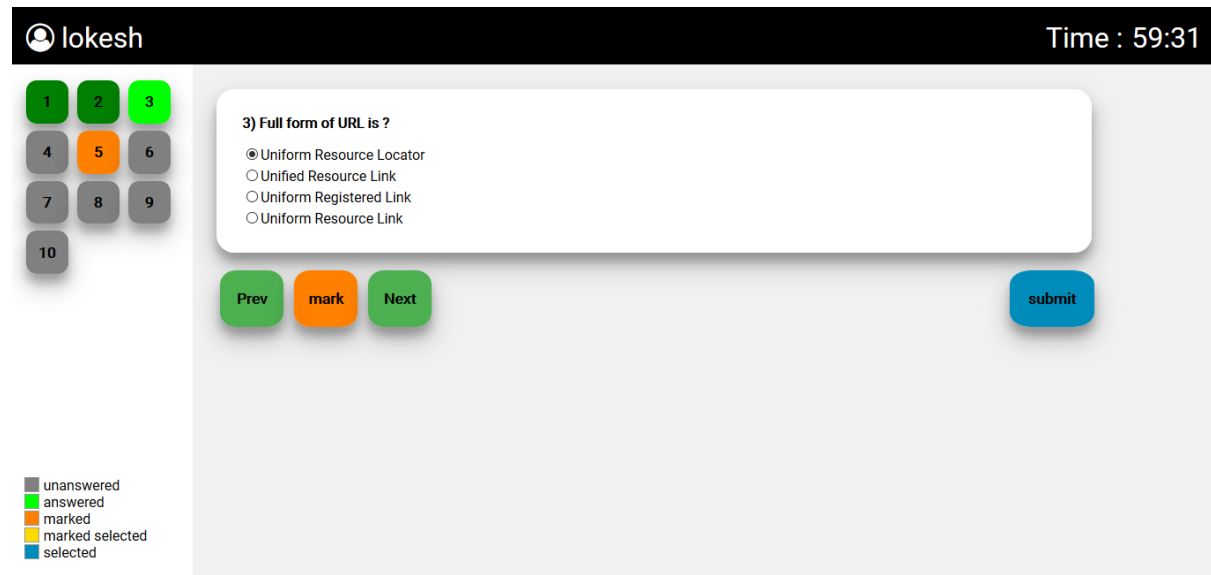
The pie chart shows the score distribution.



### Test Page:

This page is the main focus of this project. The user views the questions and answers them on this page. We have created three versions of this page for the proctor to choose from.

The first version lies in the <<master>> branch of GitHub page.



On the top bar on the left we have username and on the right side we have timer which shows time remaining until the test completes.

On the left panel we have buttons which are mapped to the questions followed by a legend which shows the state of the questions.

These buttons on the left panel can be used to jump to different questions in any order at will.

The presently selected question will be highlighted.

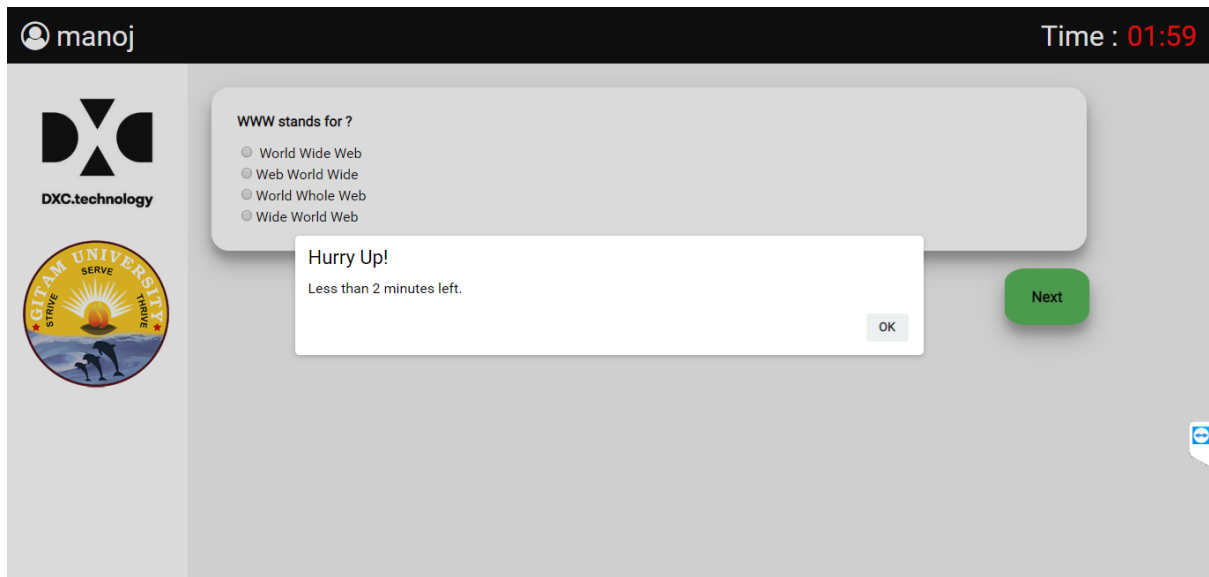
On the right side of the panel the questions are displayed along with the options.

Below the questions we have four buttons:

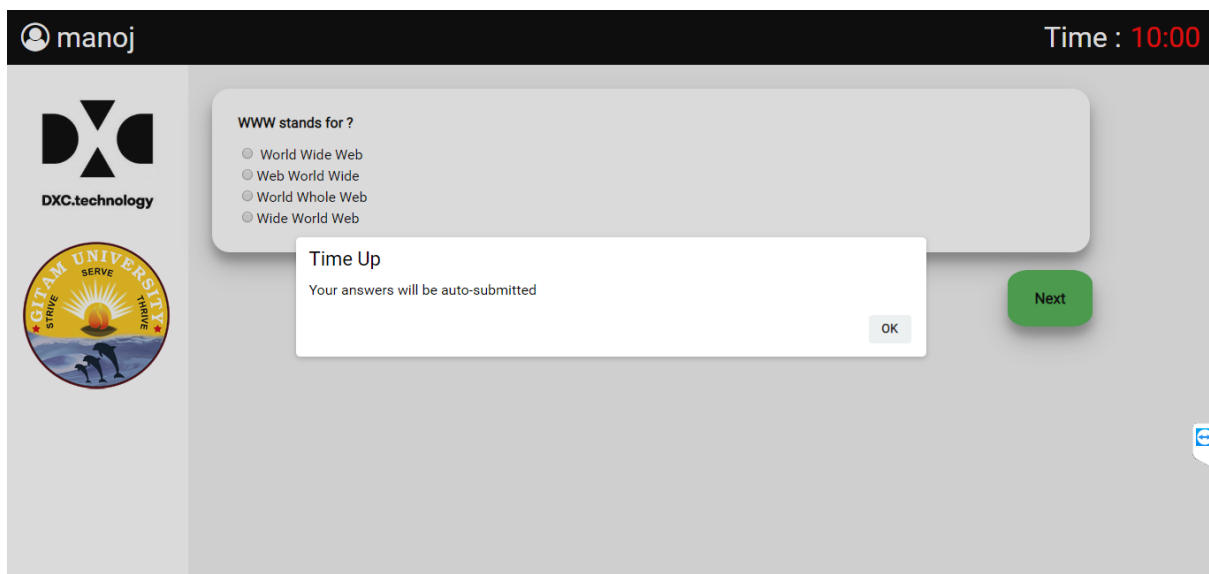
- Previous and Next are used for navigation.
- Mark button is used for marking and unmarking questions for review.
- Submit is used to submit the test immediately.

## Online Test Web Application

In the last 2 minutes of the test the user is alerted with a prompt and timer turns red in colour.

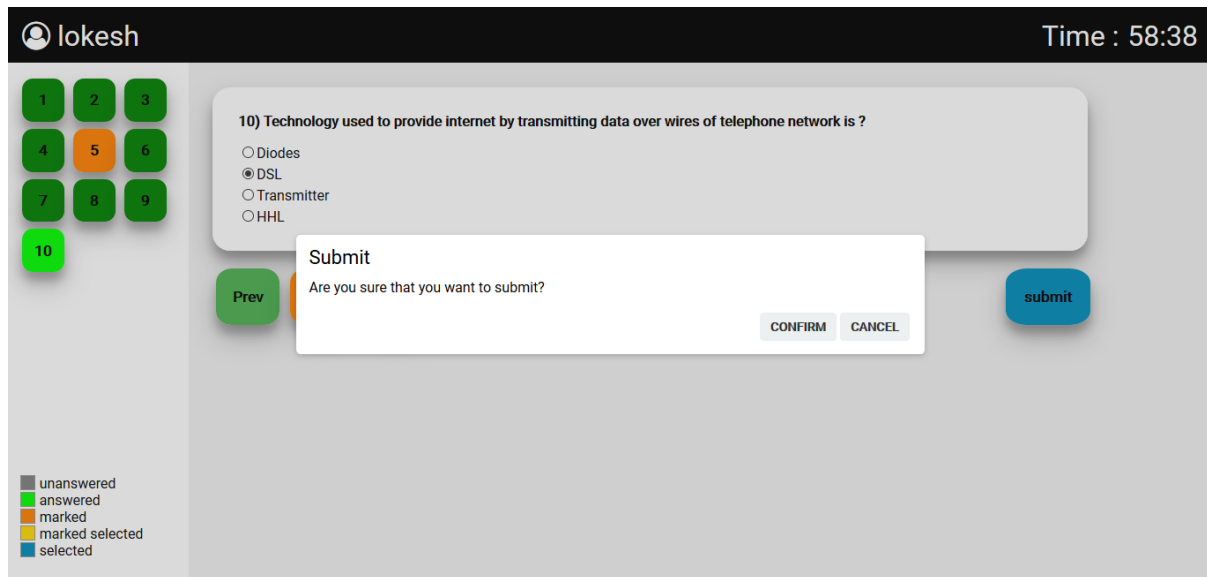


When the timer runs out a prompt is displayed and test is auto submitted.

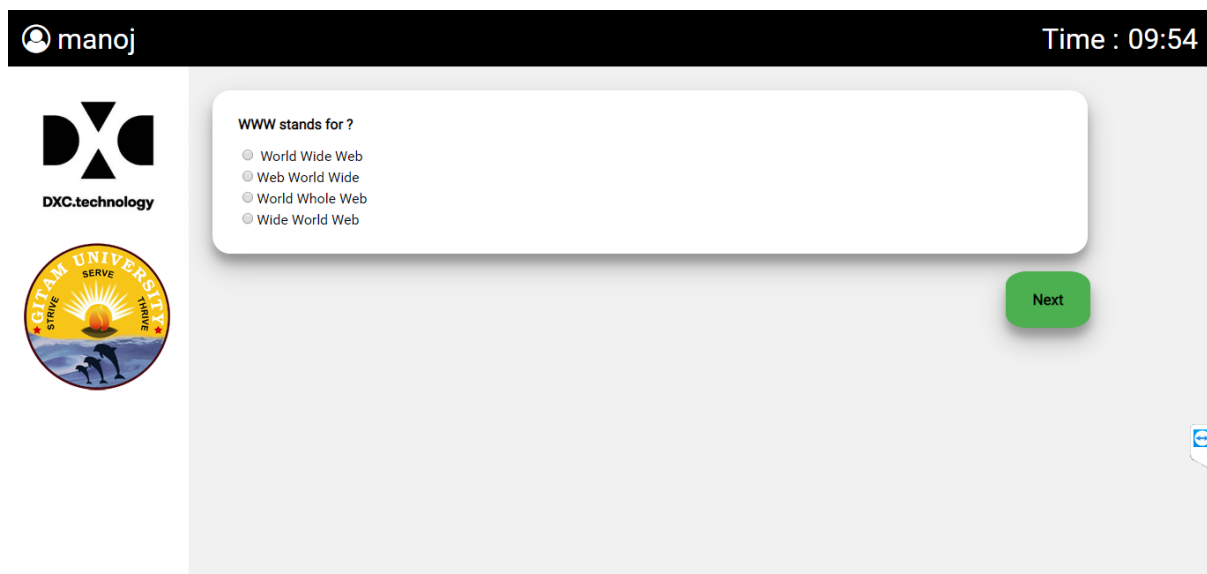


## Online Test Web Application

When the user submits the test before the time runs out a conformation dialog is displayed and test is only submitted if the user selects submit.



The second version lies in the <<q2>> branch of GitHub page.

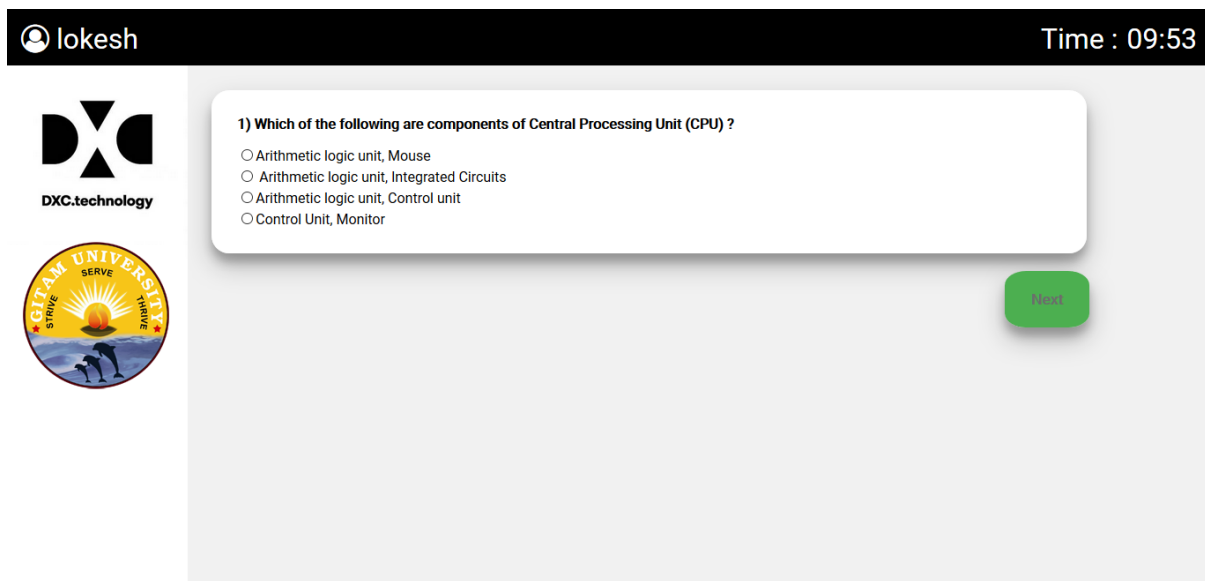


This version is similar to the first version but without most of the buttons.

In this version the left pane is disabled. We can only navigate using next button and the user can't revisit the previous questions.

The user can submit the test only after reaching the last question.

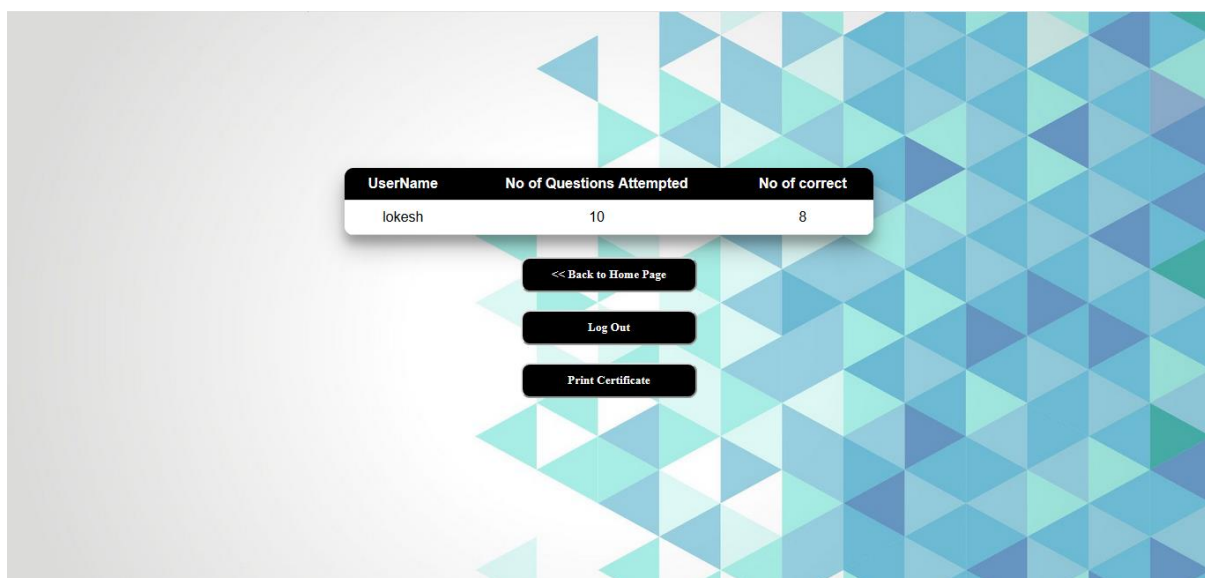
The third version lies in <<q3>> branch of GitHub page



Second and third are almost the same with only one difference which is that the next and submit buttons are disabled until the question is answered.

### Results page:

Once the User submits his/her test, his/her score will be displayed in the form of a table with the number of questions attempted and the number of correct answers with their username.



There are 3 buttons in this page:

- Back to home page

## Online Test Web Application

- Logout
- Print certificate

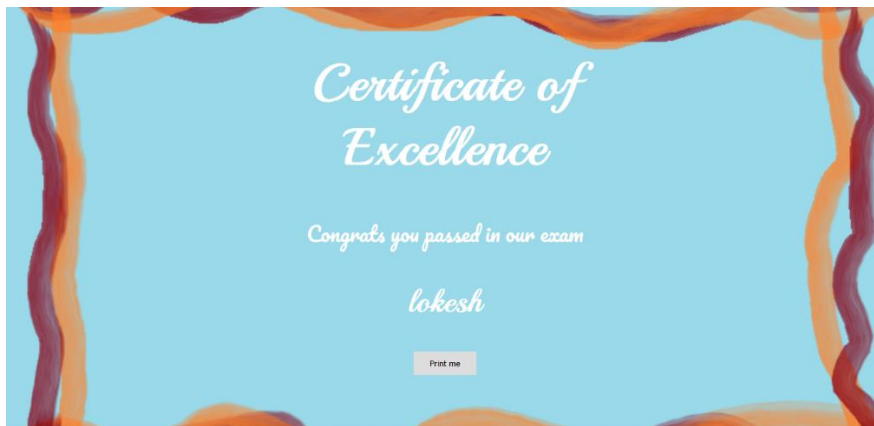
The print certificate button only appears when the user meets the passing criteria.

### Certificate Page:

In the certificate the username of the test taker is included.

We created two version of the certificate:

1. In branch <<master>>



2. In braches <<q2>> and <<q3>>



---

The print button disappears in the image when it's clicked.

## DESIGN AND IMPLEMENTATION

### Technologies used:

- Maria DB
- Java Server Pages
- HTML
- CSS3
- JavaScript
- JQuery
- Google Charts

### Database structure:

We are using three types of tables

#### 1. Users Table

Field	Type	Null	Key	Default	Extra
uname	varchar(30)	NO		NULL	
password	varchar(30)	NO		NULL	
email	varchar(45)	NO		NULL	
mobile	varchar(10)	NO		NULL	
att1	int(2)	NO		NULL	
att2	int(2)	NO		NULL	

6 rows in set (0.049 sec)

```
MariaDB [dxc]> select * from users;
```

uname	password	email	mobile	att1	att2
manoj	iamstoic	manojthestoic@gmail.com	8143220399	11	1
lokeesh	canttraceon	lokeesh.ballal23@gmail.com	9705561827	58	9
sajid	sajid034	sajidhussain98@gmail.com	7569682865	2	1

3 rows in set (0.000 sec)

uname and password fields are used for login.

Email and mobile fields are used for resetting password.

att1 and att2 fields are used to store the number of attempts in respective tests 1 and 2 by the user.

If we want to add new tests then new att fields must be created.

## 2. Score Table

```
MariaDB [dxc]> desc score;
```

Field	Type	Null	Key	Default	Extra
uid	varchar(40)	NO		NULL	
score	int(3)	NO		NULL	
testid	varchar(50)	NO		NULL	
attnum	int(2)	NO		NULL	

4 rows in set (0.025 sec)

```
MariaDB [dxc]> select * from score limit 5;
```

uid	score	testid	attnum
lokesh	0	t2	1
lokesh	6	t1	2
lokesh	9	t1	3
lokesh	9	t1	4
lokesh	9	t1	5

5 rows in set (0.000 sec)

Each entry in the score table has four fields.

The user who took the test, his/her score on the test, testid and his attempt number.

## 3. Questions Table

```
MariaDB [dxc]> desc questions;
```

Field	Type	Null	Key	Default	Extra
qid	varchar(2)	NO	PRI	NULL	
ques	varchar(700)	NO		NULL	
op1	varchar(700)	NO		NULL	
op2	varchar(700)	NO		NULL	
op3	varchar(700)	NO		NULL	
op4	varchar(700)	NO		NULL	
ca	int(1)	NO		NULL	

	ques	op1	op2	op3	op4	ca
1	WWW stands for ?	World Whole Web	Wide World Web	Web World Wide	World Wide Web	4
10	Which level lang...	high-level programmin...	medium-level programmi...	low-level programming ...	machine language	3
11	Documents, Mov...	Application Sever	Web Sever	Print Server	File Server	4
12	Which of followin...	Conductor	Semi Conductor	Vaccum Tubes	Transistor	4
13	The worst case r...	$\Theta(n \log n)$ , $\Theta(n \log n)$ ...	$\Theta(n^2)$ , $\Theta(n^2)$ and $\Theta(...$	$\Theta(n^2)$ , $\Theta(n \log n)$ an...	$\Theta(n^2)$ , $\Theta(n \log n)$ ...	4
14	which of the foll...	muqs	cfg	ondemand	yacs	4
15	Consider an arbi...	Shortest remaining ti...	Round-robin with time q...	Uniform random	Highest priority...	1
16	Consider a carry...	$\Theta(1)$	$\Theta(\log(n))$	$\Theta(\sqrt{n})$	$\Theta(n)$	2

Each entry has seven fields.

Qid, question, four options for the question and the correct option for the question.

For each test we have to create a new question table.

In our implementation we have two tests so we have created only two tables.

### Sign-Up/Sign-In transition:

For sign-in/sign-up page “index.jsp” we have created two overlapping divisions. When the user visits the page he sees the sign-in division.

If he clicks the Sign-Up button the transition to the sign-up page is done.

```
$("#sin").click(function(){
    $("#sinin").fadeToggle("800");
    $("#left-overlay").fadeToggle("800");
});
$("#sup").click(function(){
    $("#sinin").fadeToggle("800");
    $("#left-overlay").fadeToggle("800");
});
```

We used fade toggle from jQuery for smooth transition.

Similarly the user can go back to Sign-In div by clicking the Sign-In button.

### Registration logic:

For registering the new user we take all the input from the sign up form and check to see if username and email are unique or not.

```
ResultSet rs=st.executeQuery("select * from users where uname='"+uid+"' or email='"+mail+"'");
if(rs.next()){
    session.setAttribute("message","false");
    request.getRequestDispatcher("index.jsp").forward(request, response);
}else{
    PreparedStatement ps=con.prepareStatement("insert into users values(?,?,?,?,?,?)");
    ps.setString(1, uid);
    ps.setString(2, pwd);
    ps.setString(3, mail);
    ps.setString(4, phn);
    ps.setInt(5,1);
    ps.setInt(6,1);
    ps.executeUpdate();
    session.setAttribute("message","true");
    request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

If its unique then we insert the data into the user table else we return a session attribute to display a prompt to the user.



### Login logic:

We take the username and password from the sign in form and validate it with the database.

```
ResultSet rs=st.executeQuery("select uname,password from users where uname='"+request.getParameter("uid")+
"+" and password='"+request.getParameter("pwd")+"'");
if(rs.next()){
    session.setAttribute("uid",rs.getString("uname"));
    response.sendRedirect("user.jsp");
}else{
    session.setAttribute("message","lfail");
    request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

If the corresponding username and password exists then a session attribute called uid is set and the user is redirected to the home page.

As long as uid has a value the user is considered logged in.

The username displayed on the top right of the home and questions page is taken from this attribute.

If username and password does not exist we set a session attribute to display a prompt.

### Resetting password:

We first check if the email and phone number the user provided are valid or not.

```
ResultSet rs=st.executeQuery("select uname from users where email='"+request.getParameter("mail")+
"+" and mobile='"+request.getParameter("phn")+"'");
if(rs.next()){
    session.setAttribute("mess","success");
    session.setAttribute("mail",request.getParameter("mail"));
    session.setAttribute("phn",request.getParameter("phn"));
    response.sendRedirect("forgot.jsp");
}else{
    session.setAttribute("mess","fail");
    request.getRequestDispatcher("forgot.jsp").forward(request, response);
}
```

If they are not valid we set a session attribute to display a prompt else the user is greeted with a page where he can set a new password then we find the user with the corresponding email and phone number and reset the password.

```
int i=st.executeUpdate("update users set password='"+request.getParameter("new")+"' where email='"+
+session.getAttribute("mail")+"' and mobile='"+session.getAttribute("phn")+"'");
if(i==1){
    session.removeAttribute("mail");
    session.removeAttribute("phn");
    session.setAttribute("mess","changed");
    request.getRequestDispatcher("forgot.jsp").forward(request, response);
}else{
    session.setAttribute("mess","notchanged");
    request.getRequestDispatcher("forgot.jsp").forward(request, response);
}
```

Then we set session attributes to display prompts to user showing whether the password has changed or not.

### Scores:

On the scores page we can see scores of all the tests attempted by the user. First we retrieve all the scores from score table of that user using uid attribute. Next put the scores of the test in corresponding divs and only scores user wants to see are displayed at any time.

In each div for the corresponding score data a pie chart is displayed showing marks distribution which is generated using the google charts api.

### Questions:

#### 1. Timer

```
function startTimer(duration, display) {
    var start = Date.now(),
        diff,flag=0,minutes,seconds;
    function timer() {
        // get the number of seconds that have elapsed since
        // startTimer() was called
        diff = duration - (((Date.now() - start) / 1000) | 0);
        // does the same job as parseInt truncates the float
        minutes = (diff / 60) | 0;
        seconds = (diff % 60) | 0;
        minutes = minutes < 10 ? "0" + minutes : minutes;
        seconds = seconds < 10 ? "0" + seconds : seconds;
        display.textContent = minutes + ":" + seconds;
        if(diff<2*60 && flag!=1){
            $.alert({
                useBootstrap: false,
                title: 'Hurry Up!',
                content: 'Less than 2 minutes left.',});
            $('#time').css("color","red");
            flag=1;
        }
        if(flag==1 && diff==0){
            submit();
        }
        if (diff <= 0) {
            start = Date.now() + 1000;
        }
    };
    timer();
    setInterval(timer, 1000);
}
```

The timer function uses the `date.now()` function to maintain accuracy. When the time remaining is less than 2 minutes an alert is displayed notifying the user and colour of the time is changed to red. When the time runs out answers are auto submitted after displaying an alert to the user.

```
$(window).on('load', function () {  
    //set timer here  
    var Minutes = 0.5 * 60,  
    display = document.querySelector('#time');  
    $("#1").toggle();  
    $('#prev').prop('disabled', true);  
    startTimer(Minutes, display);  
    $('#b1').css("background-color", "#008CBA");  
});
```

The timer starts when the page is loaded. The duration of the timer is defined in the minutes variable as number of minutes wanted multiplied by 60.

## 2. Shuffling

The questions and its options are shuffled randomly to prevent cheating. The shuffling is achieved using the `collections.shuffle()` and array list from `java.util` package.

```
ArrayList<Integer> mylist = new ArrayList<Integer>();  
for(int i=1;i<=18;i++)  
{  
    mylist.add(i);  
}  
Collections.shuffle(mylist);
```

In this example we have 18 questions in the database so  $i \leq 18$ .

### 3. Retrieval

The questions are retrieved from database based on the qid from the shuffled list.

```
if(request.getParameter("tid").equals("t1")){
    sql="select * from questions where qid=";
    out.print("<input type='hidden' name='tid' value='t1'>");
}else if(request.getParameter("tid").equals("t2")){
    sql="select * from testtwo where qid=";
    out.print("<input type='hidden' name='tid' value='t2'>");
}
while(i<11){
    rs=st.executeQuery(sql+mylist.get(i));
    rs.next();
    ArrayList<Integer> options = new ArrayList<Integer>();
    for(int j=1;j<=4;j++){
        options.add(j);
    }
    Collections.shuffle(options);
    out.print("<div class='qblock' id='"+i+"' style='display:none;'><table><tr><td>"+i+" "+
rs.getString("ques")+"</td></tr><tr class='blank_row'></tr>");
    out.print("<tr class='shuff'><td><input type='radio' name='op'+i+"' value='"+options.get(0)+
"/>"+rs.getString("op"+options.get(0))+"</td></tr>");
    out.print("<tr class='shuff'><td><input type='radio' name='op'+i+"' value='"+options.get(1)+
"/>"+rs.getString("op"+options.get(1))+"</td></tr>");
    out.print("<tr class='shuff'><td><input type='radio' name='op'+i+"' value='"+options.get(2)+
"/>"+rs.getString("op"+options.get(2))+"</td></tr>");
    out.print("<tr class='shuff'><td><input type='radio' name='op'+i+"' value='"+options.get(3)+
"/>"+rs.getString("op"+options.get(3))+"</td></tr>");
    out.print("<input type='hidden' value='"+rs.getString("qid")+"' name='qid'+i+"'></table></div>");
    i++;
}
```

First we select which question table we are going to retrieve from based on the value of session attribute tid.

Then the retrieved questions are put in their respective div which are initially hidden.

Each div consists of question followed by four radio buttons with their corresponding options and one hidden attribute containing qid which used again in evaluation.

### 4. Navigation

At the start all the divs are hidden except the first one. We maintain a variable curr which stores the div number of the question which is currently being displayed. When we move to other questions using the left pane or previous or next button the present div is hidden and the new div is show and curr is updated.

### 5. Colour scheme

We maintain two lists arr and arr\_ans. The arr array stores the question numbers of the questions marked for review and arr\_ans stores the question number of questions which are answered.

First we check the array `arr` if the question is marked the behaviour of the mark button is changed and the colour of left pane button is set to orange.

If the question is not marked then we check the `arr_ans` array. If the question is answer we set the left pane button to green.

If the question is both not marked and not answered is selected then we set the left pane button colour to blue. If it's not selected then the colour is set to grey.

#### 6. Q3

In q3 the next button is disabled by default. When the user clicks on any option the button is enabled. When the user clicks the next button the button is disabled again.

```
$('#input').click(function(){
    var str=$(this).attr('name');
    arr_ans[parseInt(str.slice(2))-1]=1;
    if(arr[parseInt(str.slice(2))-1]!=1)
        $('#b'+str.slice(2)).css("background-color","#00ff00");
    $('#next').prop('disabled',false);
    if(curr==10){
        $('#sub').prop('disabled', false);
    }
});
```

```
$('#next').click( function(){
    $('#next').prop('disabled','true');
```

## Evaluation:

We maintain two variables `no_ques` and `correct`. We first select which table to retrieve the answers from the database based on `tid` attribute. Then for each question using the `qid` hidden attribute we retrieve the correct answer from the table and check the user's choice. If it's correct we increment the `correct` variable. Regardless if the question is correct or wrong the `no_ques` variable is incremented.

```
while(i<=10)
{
    if(request.getParameter("op"+i)!=null)
    {
        no_ques++;
        rs=st.executeQuery(sql+request.getParameter("qid"+i));
        rs.next();
        if(request.getParameter("op"+i).equals(rs.getString("ca")))
        {
            correct++;
        }
    }
    i++;
}
```

We display the score to the user in the form of a table. The `no_ques` variable is used to display the number of questions attempted and `correct` is used to display the score. The score is also inserted into the database along with the attempt number based on the `uid` attribute.

```
<table>
<tr><th>UserName</th><th>No of Questions Attempted</th><th>No of correct</th></tr>
<tr><td><%=session.getAttribute("uid")%></td><td><%out.print(no_ques);%></td><td><%out.print(correct);%></td></tr>
</table>
<%
try{
    Class.forName("org.mariadb.jdbc.Driver");
    java.sql.Connection con = DriverManager.getConnection("jdbc:mariadb://localhost:3306/dxc","root","tracoon");
    Statement st=con.createStatement();
    ResultSet rs=st.executeQuery("select * from users where uname='"+session.getAttribute("uid")+"'");
    rs.next();
    Statement st1=con.createStatement();
    PreparedStatement ps=con.prepareStatement("insert into score values(?,?,?,?)");
    ps.setString(1,rs.getString("uname"));
    ps.setInt(2,correct);
    ps.setString(3,request.getParameter("tid"));
    if(request.getParameter("tid").equals("t1")){
        st1.executeUpdate("update users set att1='"+(rs.getInt("att1")+1)+"' where uname='"+session.getAttribute("uid")+"'");
        ps.setInt(4,rs.getInt("att1"));
    }else if(request.getParameter("tid").equals("t2")){
        st1.executeUpdate("update users set att2='"+(rs.getInt("att2")+1)+"' where uname='"+session.getAttribute("uid")+"'");
        ps.setInt(4,rs.getInt("att2"));
    }
    ps.executeUpdate();
}
```

## Certificate:

Only if the user meets the passing criteria the print button is displayed on the evaluation page.

```
if(correct>=7){
    session.setAttribute("cert","pass");
    out.print("<button id='pcerf' target='_blank'>Print Certificate</button>");
}
```

In this case the passing criteria is 7 questions.

The certificate contains the username which is pulled from the session attribute uid.

```
<div class="hero-image">
  <div class="hero-text">
    <h1 style="font-size:3em">Certificate of Excellence</h1>
    <p>Congrats you passed in our exam</p>
    <h3><%=session.getAttribute("uid")%></h3>
    <button id="cmd">Print me</button>
  </div>
</div>
```

There is a print button in the certificate page which when clicked hides itself and then invokes the browsers print dialog. After printing the button is displayed once again.

```
$('#cmd').click(function(){
    $('#cmd').hide();
    window.print();
    $('#cmd').show();
});
```

## Flow control enforcement:

We have made efforts in various different pages to prevent the user from accessing the pages he is not supposed to using various session attributes. If he/she tries to attempt such action they will be redirected to home page if they are logged in and sign in/sign up page if they are logged out.

```
if(session.getAttribute("uid")==null){  
    response.sendRedirect("index.jsp");  
}  
if(session.getAttribute("test")==null){  
    response.sendRedirect("user.jsp");  
}  
session.removeAttribute("test");  
response.setHeader("Cache-Control","no-cache");  
response.setHeader("Cache-Control","no-store");  
response.setHeader("Pragma","no-cache");  
response.setDateHeader ("Expires", 0);
```

### Logout:

When the user clicks the logout button or closes the browser windows the session is invalidated. When the user revisits the page he has to login again.

```
<%  
session.invalidate();  
response.setHeader("Cache-Control","no-cache");  
response.setDateHeader("Expires", 0);  
response.setHeader("Pragma","no-cache");  
response.sendRedirect("index.jsp");  
%>
```

After the session is invalidated the user is automatically redirected to the sign in/sign up page.