NAME: S. PREETHI

COLLEGE NAME: VIT

REG.NO:19MCS0052

COMPANY: NAVIGUS

*VIT Campus Hiring 21*

Assignment-1

Build a Presence service, similar to that of google docs, which shows people that are

currently viewing a particular doc.

HTML FILE:

**CUSTOMER REGISTARTION:**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<META http-equiv="Content-Type" content="text/html; charset=utf-8">

<link rel=stylesheet href="registration1.css">

</head>

<script>

function validate()

{

var invalid=" ";

var minLength=6;

var pw1=document.myForm.psw.value;

var pw2=document.myForm.cpsw.value;

if(pw1==''||pw2=='')

{

alert("please enter your password twice.");

return false;

}

else {

if (document.myForm.psw.value.length<minLength)

{

alert("your password must be at least"+minLength+"characters long. Try again.");

return false;

}

}

if(document.myForm.psw.value.indexOf(invalid)>-1)

{

alert("Sorry, spaces are not allowed.");

return false;

}

else

{

if(pw1!=pw2)

{

alert("You did not enter the same password twice. Please re-enter your password.");

return false;

}

else

{ alert("Password Accepted.");

return true;

}

}

}

</script>

<body>

<h1>Customer Registration:</h1><br>

<form name="myForm" method="post" action="customerdetails.jsp" onSubmit="return validate()">

(Fields marked with \* are mandatory)<br><br>

<b>\*First Name:</b><br>

<input type="text" name="fname" required><br><br>

<b>\*Last Name:</b><br>

<input type="text" name="lname" required><br><br>

<b>\*Age:</b><br>

<input type="text" name="age" required><br><br>

<b>\*Gender:</b><br>

<input type="radio" name="gender" value="male" required> Male<br>

<input type="radio" name="gender" value="female" required> Female<br><br>

<b>\*Email:</b><br>

<input type="email" name="email" required><br><br>

<b>\*Country:</b><br>

<input type="text" name="country" required><br><br>

<b>\*Address:</b><br>

<input type="text" name="address" required><br><br>

<b>\*Mobile Number:</b><br>

<input type="text" name="phone" required><br><br>

<b>Alternate Mobile Number:</b><br>

<input type="text" name="alt\_phone"><br><br>

Enter your password twice.

<br>

(At least 6 characters, 12 characters max, and spaces are not allowed.)

<br><br>

<b>\*Password</b><br>

<input type="password" name="psw" maxlength=12><br><br>

<b>\*Confirm Password</b><br>

<input type="password" name="cpsw" maxlength=12>

<input type="checkbox" checked> Remember\_me<br><br>

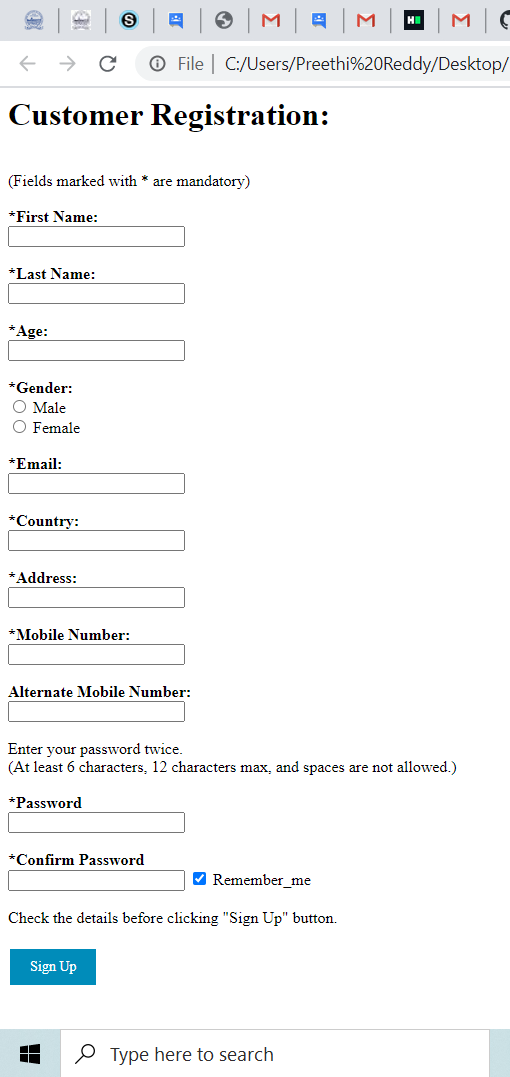
Check the details before clicking "Sign Up" button.<br><br>

<input type="submit" value="Sign Up">

</form>

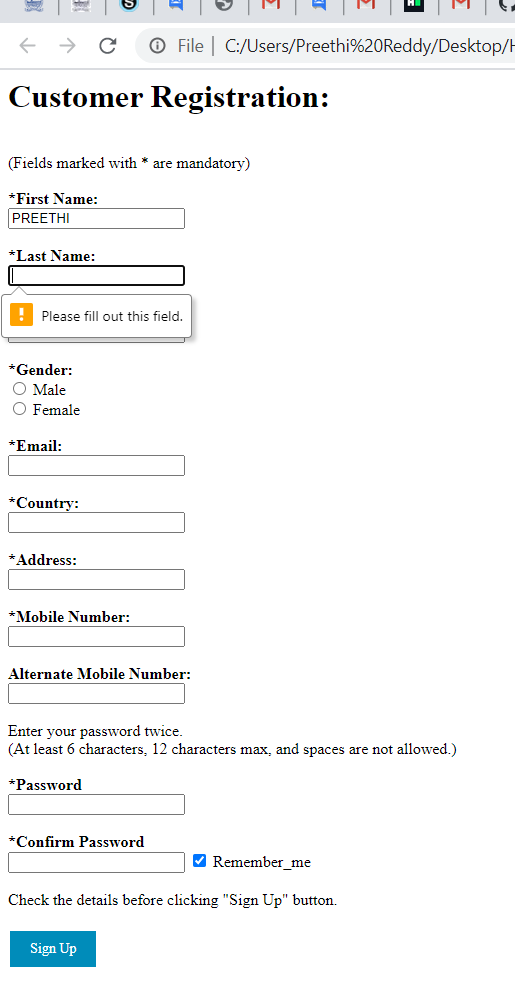
</body>

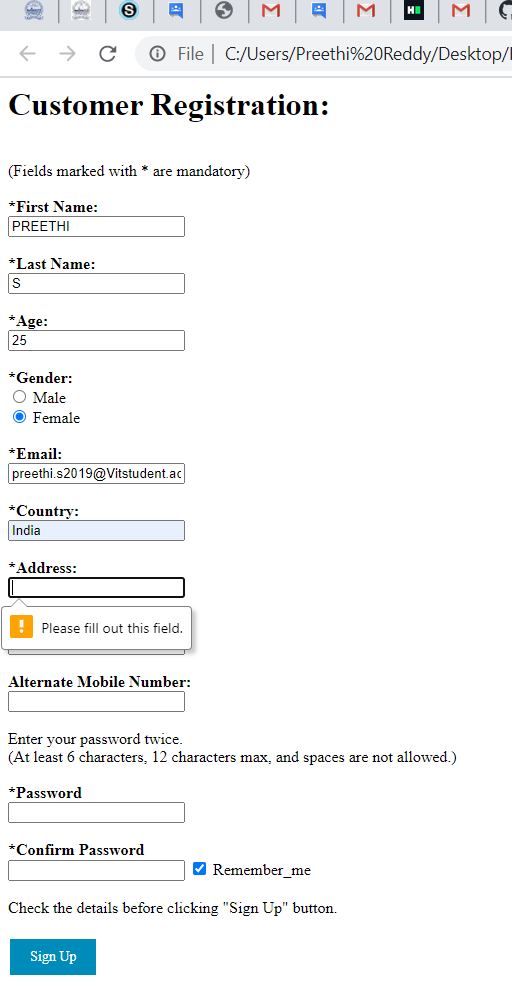
</html>



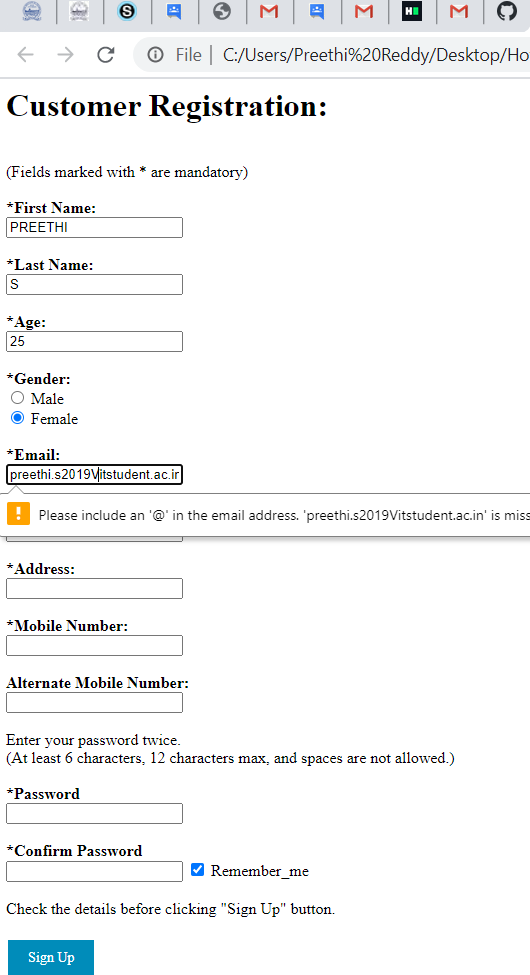
TESTCASES:

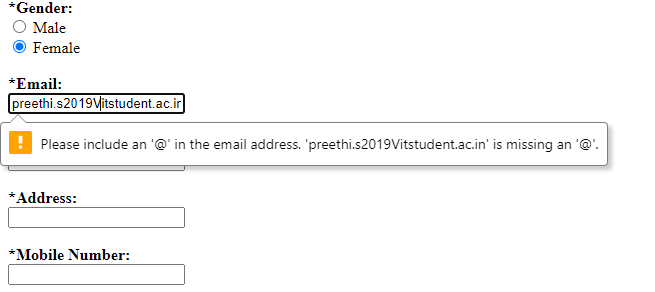
IF ANY FIELD IS NOT FILLED-LASTNAME AND ADDRESS



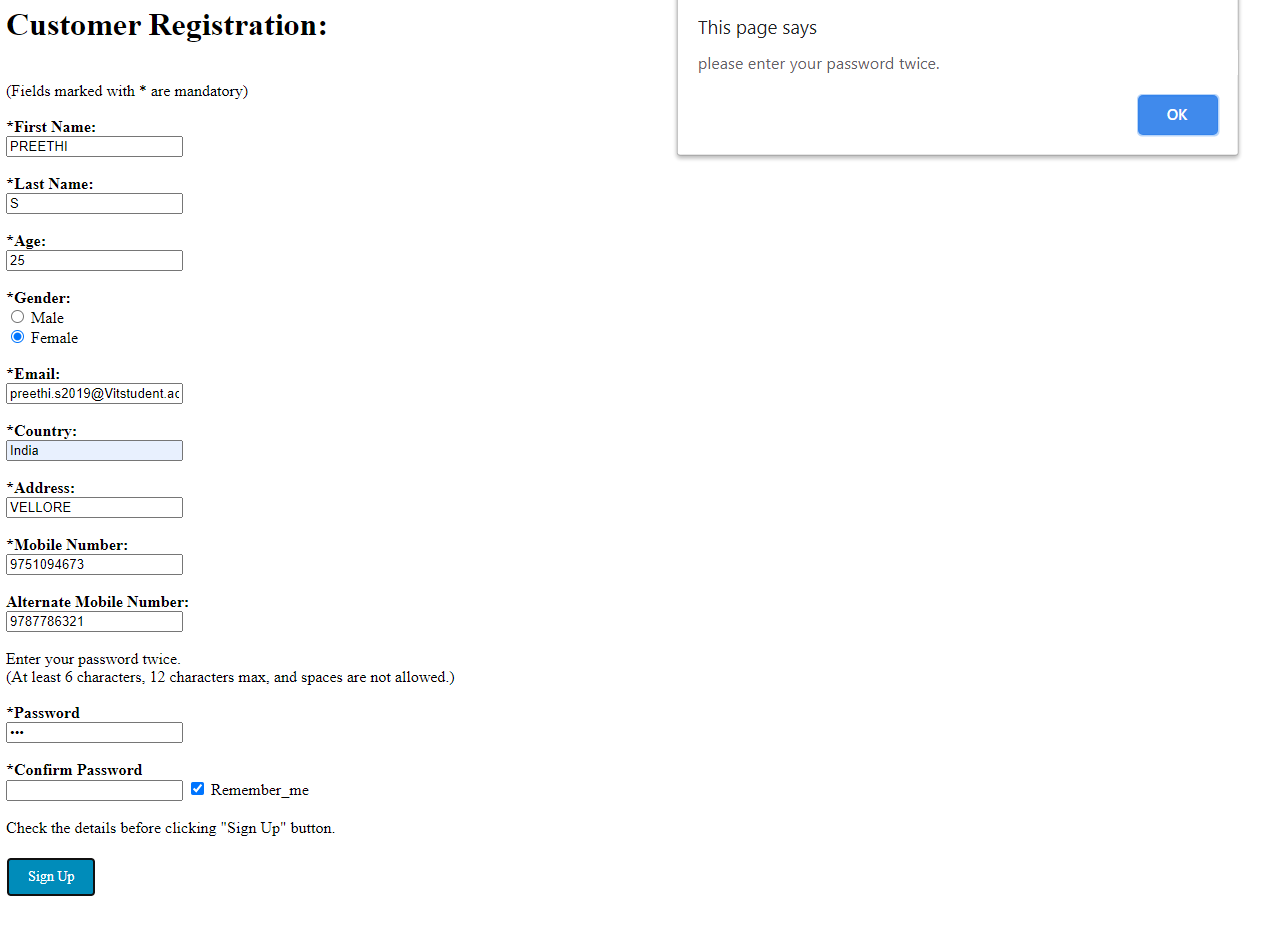


E-MAILID:@SYMBOL IS MISSING

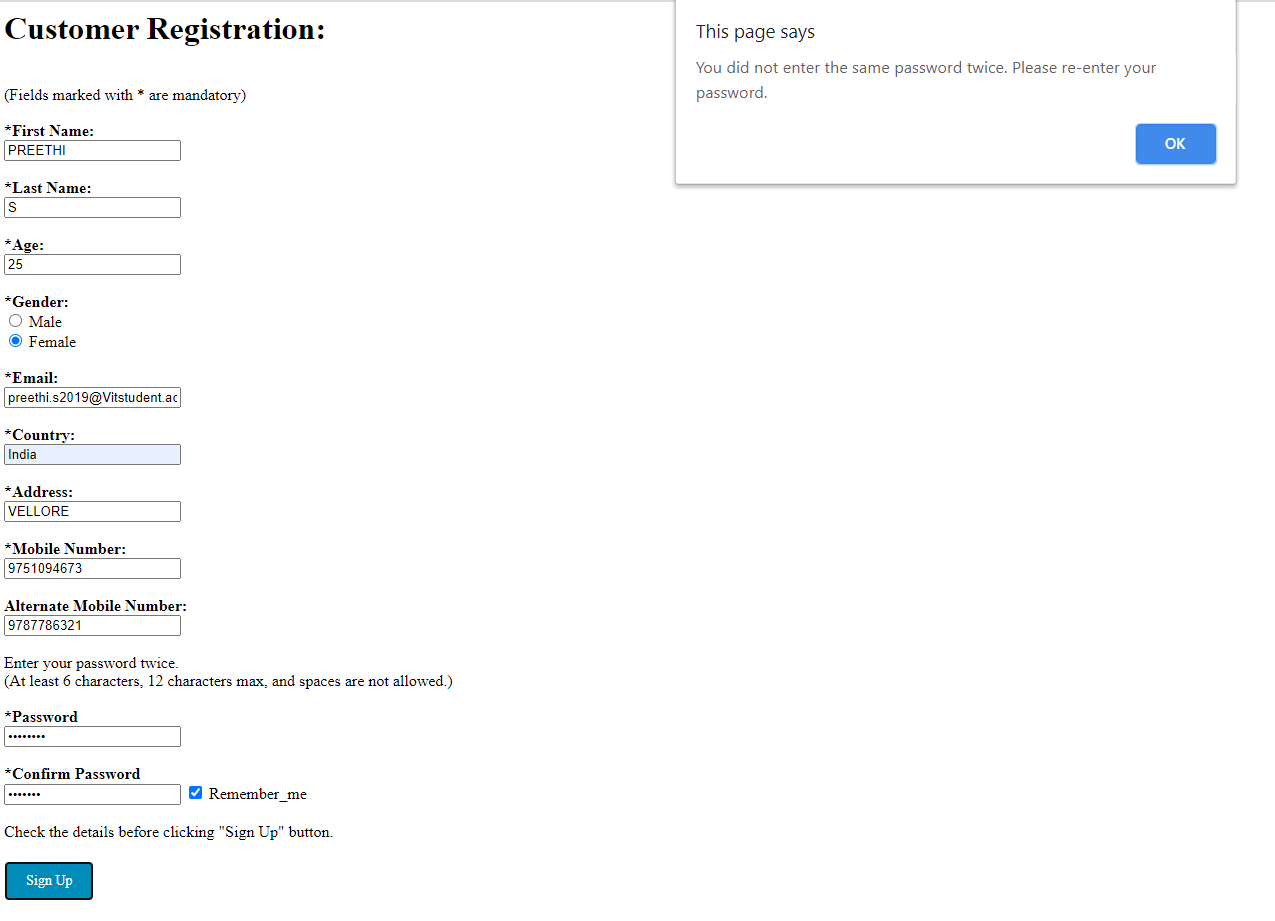




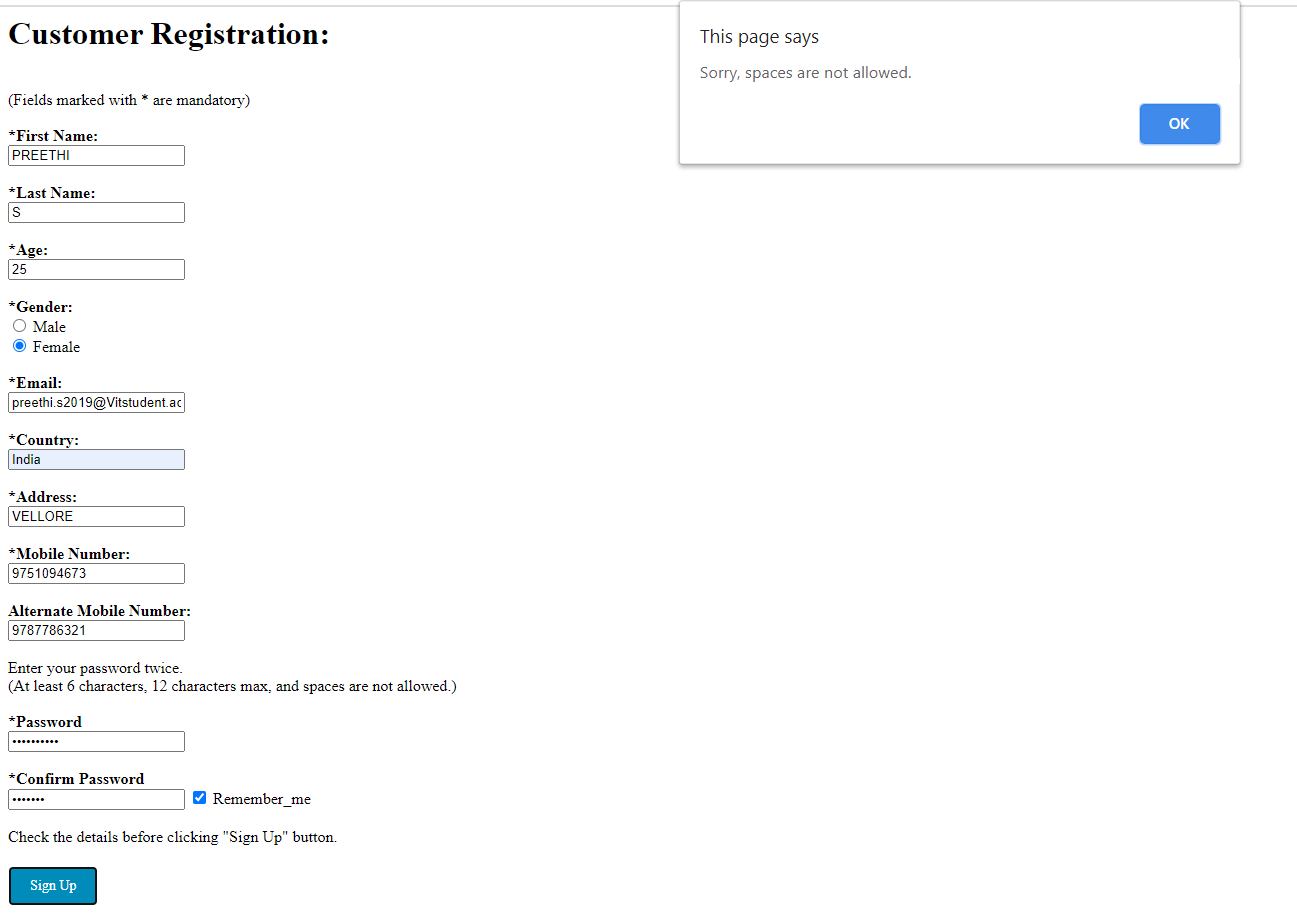
PASSWORD VALIDATION:



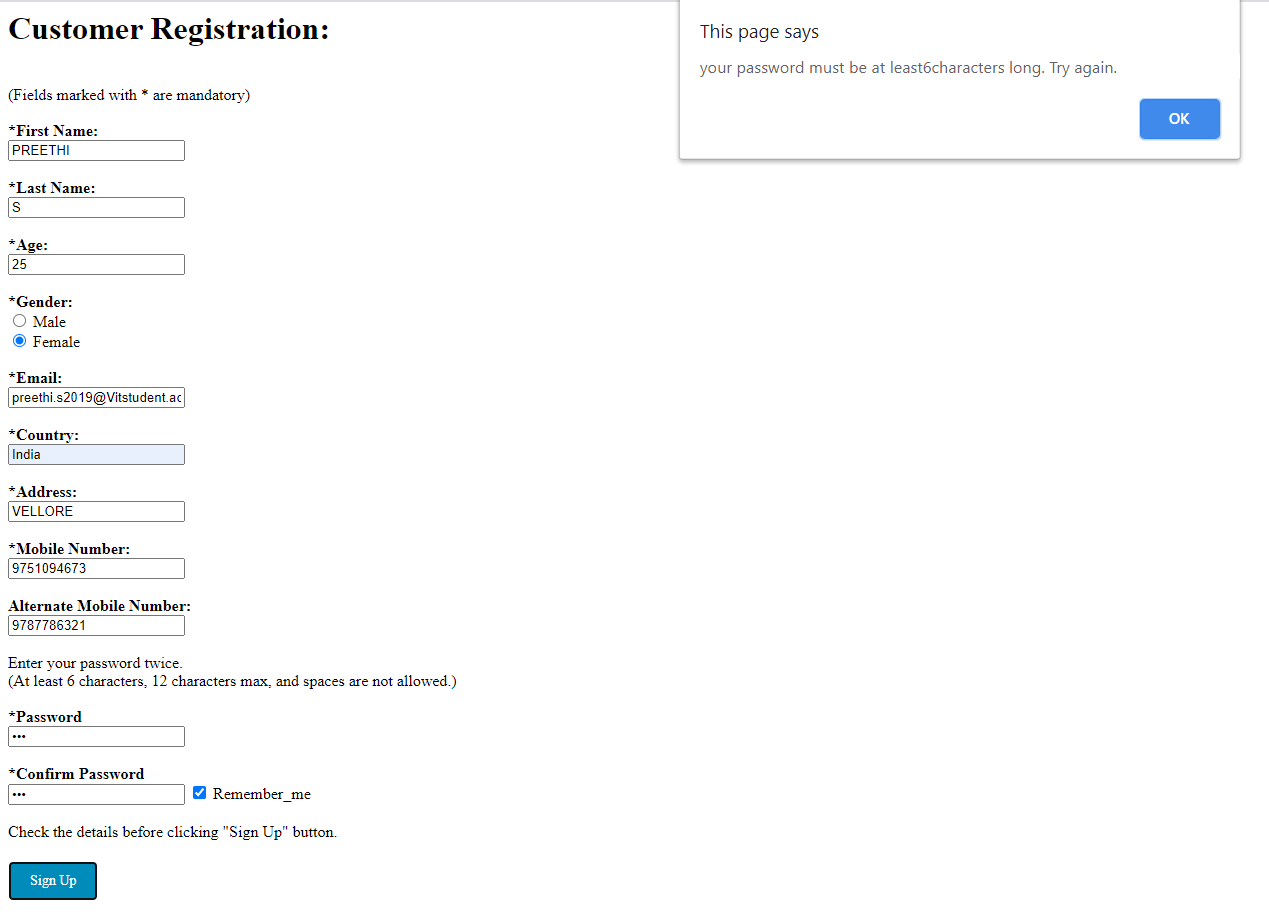
PASSWORD MISMATCHES THEN OUTPUT WILL BE



IF SPACES ARE GIVEN:



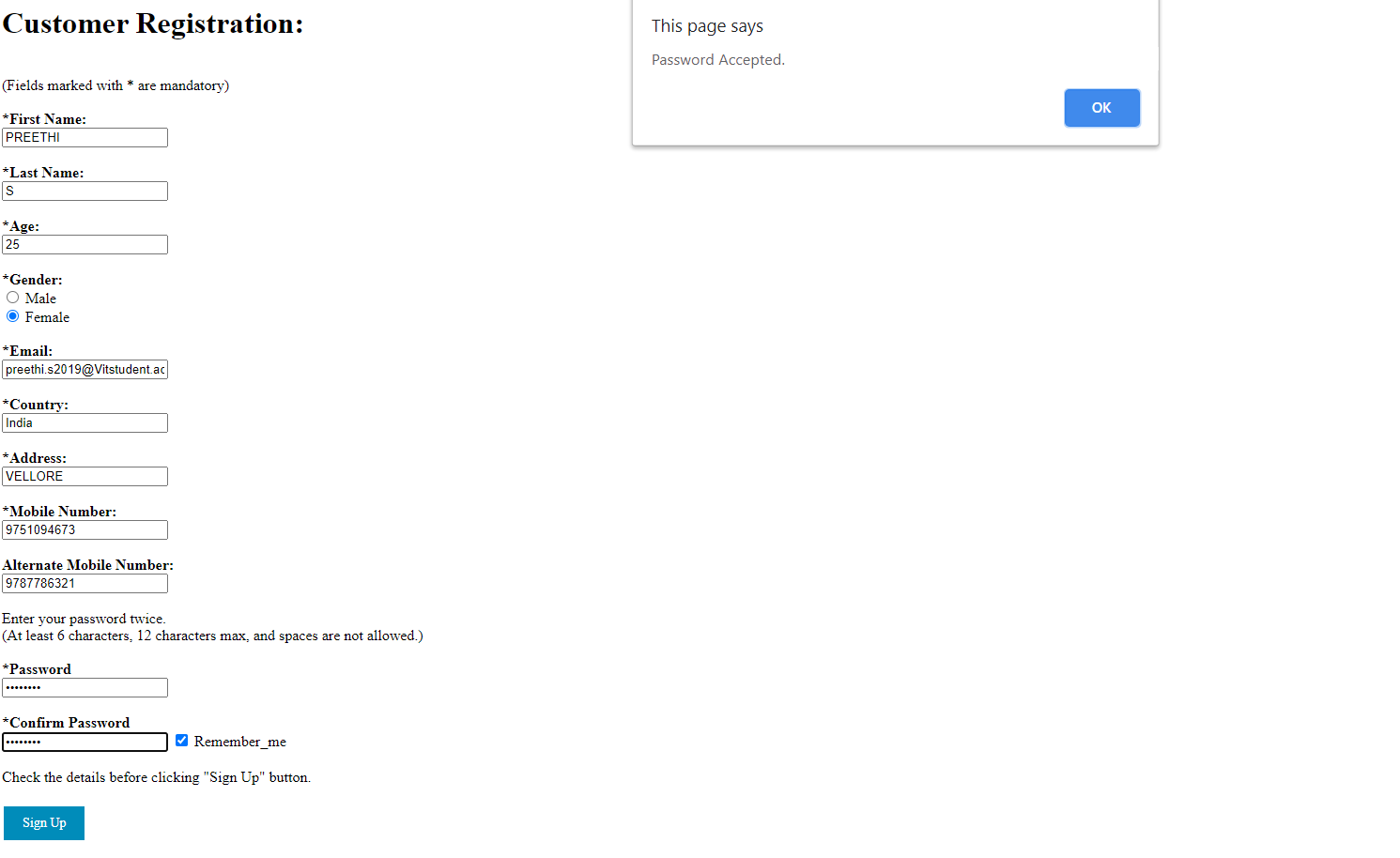
IF THE PASSWORD GIVEN IS LESS THAN 6 LETTES



ALL SPECIFICATIONS ARE CORRECT THEN PASSWORD IS ACCEPTED:

THIS IS THE LINK FOR USER REGISTRATION:

<file:///C:/Users/Preethi%20Reddy/Desktop/HotelManagementSystem/WebContent/userregistration.html>



JSP CODE:

LOGIN PAGE:

LOGIN.JS

package p1;

import java.util.Map;

import org.apache.struts2.dispatcher.SessionMap;

import org.apache.struts2.interceptor.SessionAware;

import com.opensymphony.xwork2.ActionSupport;

public class Login extends ActionSupport implements SessionAware

{

    String email;

    String password;

    SessionMap<String,String> s;

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String execute()

    {

        String result=ERROR;

        if(CustomerLogin.validate(email, password))

        {

            result=SUCCESS;

        }

        else

        {

            result=ERROR;

        }

        return result;

    }

    public void setSession(Map map)

    {

        s=(SessionMap)map;

        s.put("login", "true");

    }

    public String logout()

    {

        s.invalidate();

        return ERROR;

    }

}

CUSTOMER LOGIN:

package p1;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class CustomerLogin

{

    public static boolean validate(String email,String password)

    {

        boolean r=false;

        Connection con=null;

        String query="select \* from customer where email=? and password=?";

        try

         {

            Class.forName("com.mysql.jdbc.Driver");

            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test4","root","Krishna9!");

            PreparedStatement pstmt = con.prepareStatement(query);

            pstmt.setString(1, email);

            pstmt.setString(2, password);

            ResultSet rs = pstmt.executeQuery();

            r=rs.next();

         }

         catch (Exception e)

         {

             System.out.println(e);

         }

         finally

         {

            if (con != null)

            {

            try

            {

                con.close();

            }

            catch (Exception e)

            {

                System.out.println(e);

            }

            }

         }

        return r;

    }

}

GOOGLE SHEET WITH ANGULARJS JSON AND SERVER AS MONGODB.

CHECK VALIDATION:

Our validation flow for our register.js file will go as follows:

Pull in validator and is-empty dependencies

Export the function validate Register Input, which takes in data as a parameter (sent from our frontend registration form, which we’ll build in Part 2)

Instantiate our errors object

Convert all empty fields to an empty string before running validation checks (validator only works with strings)

Check for empty fields, valid email formats, password requirements and confirm password equality using validator functions

Return our errors object with any and all errors contained as well as an is Valid boolean that checks to see if we have any errors

CODE:

const Validator = require("validator");

const isEmpty = require("is-empty");

module.exports = function validateRegisterInput(data) {

let errors = {};

// Convert empty fields to an empty string so we can use validator functions

data.name = !isEmpty(data.name) ? data.name : "";

data.email = !isEmpty(data.email) ? data.email : "";

data.password = !isEmpty(data.password) ? data.password : "";

data.password2 = !isEmpty(data.password2) ? data.password2 : "";

// Name checks

if (Validator.isEmpty(data.name)) {

errors.name = "Name field is required";

}

// Email checks

if (Validator.isEmpty(data.email)) {

errors.email = "Email field is required";

} else if (!Validator.isEmail(data.email)) {

errors.email = "Email is invalid";

}

// Password checks

if (Validator.isEmpty(data.password)) {

errors.password = "Password field is required";

}

if (Validator.isEmpty(data.password2)) {

errors.password2 = "Confirm password field is required";

}

if (!Validator.isLength(data.password, { min: 6, max: 30 })) {

errors.password = "Password must be at least 6 characters";

}

if (!Validator.equals(data.password, data.password2)) {

errors.password2 = "Passwords must match";

}

return {

errors,

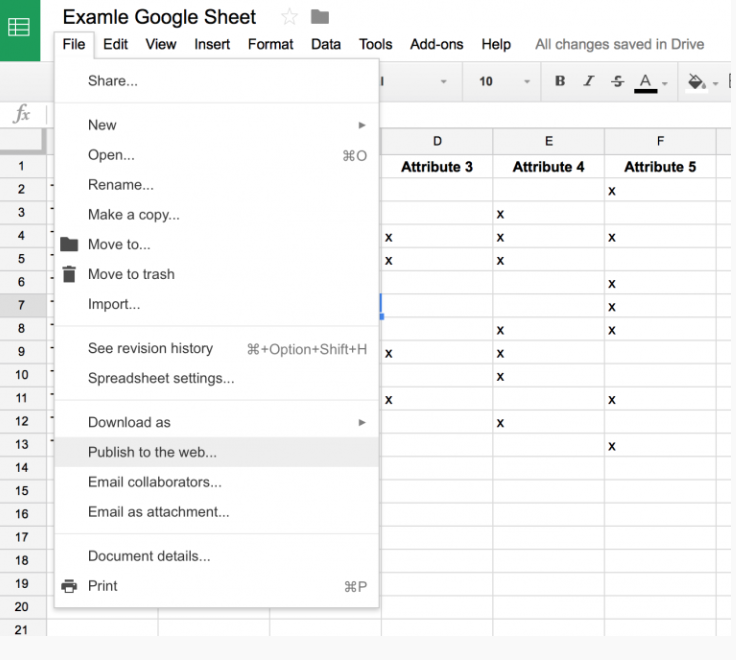
isValid: isEmpty(errors)

};

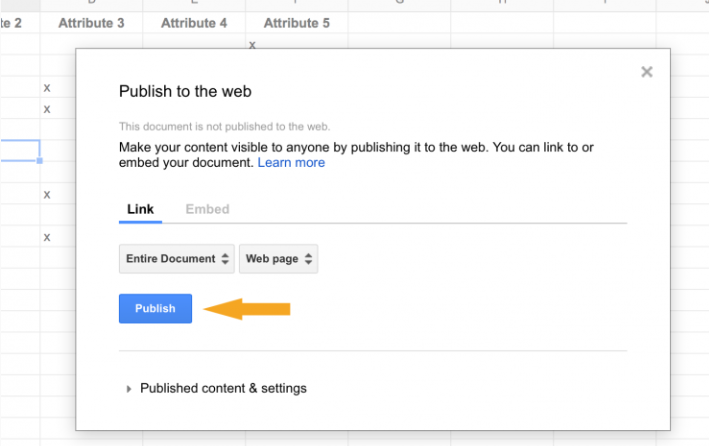
};

Get a JSON Feed of the Google Sheet Data

1. File > Publish to the web

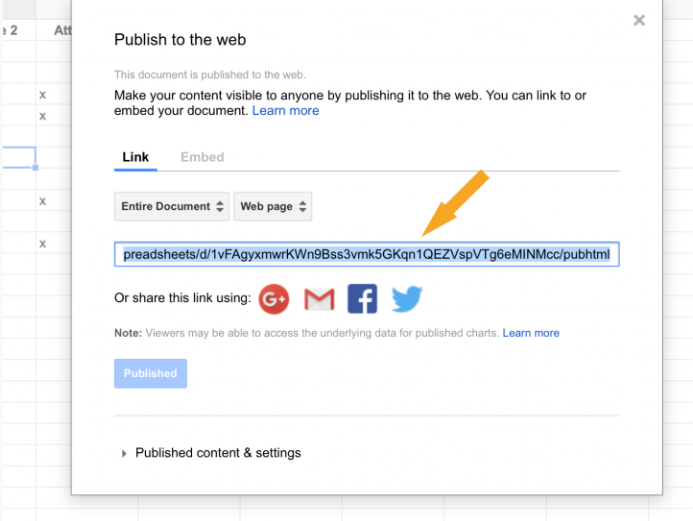


2. Click Publish for the entire document or sheets you want available on the web.

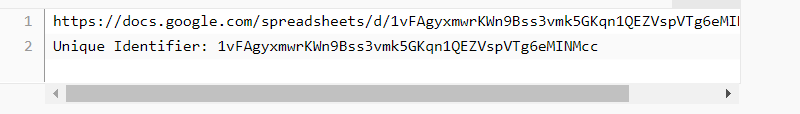


3. Confirm choice.

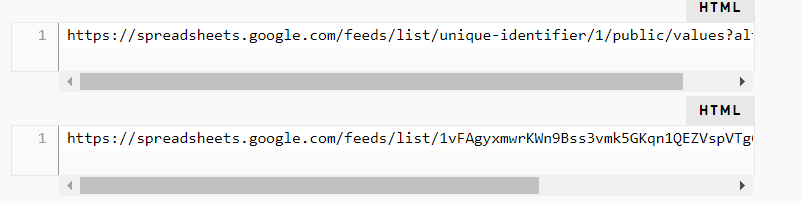
4. Copy the URL link and paste the URL into a text editor.



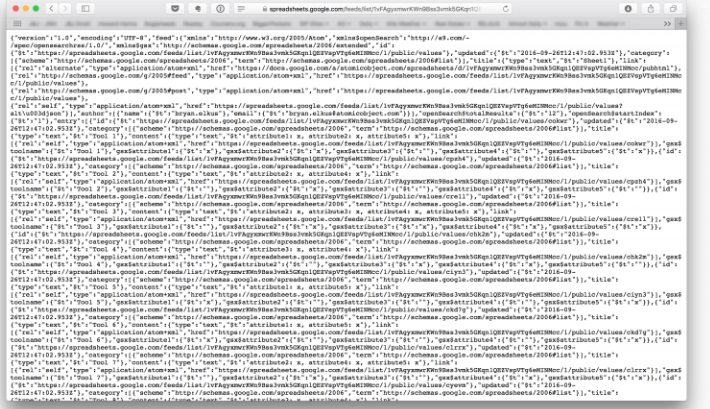
5. Find the Sheet’s unique identifier in the URL:



6. Copy/paste the unique identifier into unique-identifier in the code below:



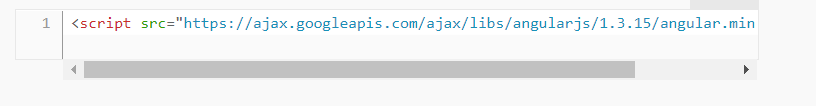
7. Copy this new URL and paste it into a browser to confirm that you have the data. It should look something like this:



access to the Google Sheet in JSON format

Add the AngularJS code in the header of your HTML file:

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>



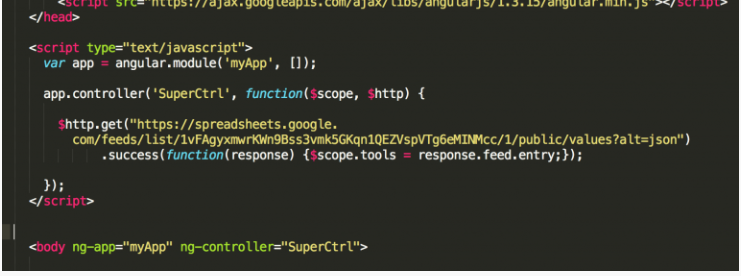
Add the code for AngularJS to pull in the JSON feed and set up the angular application. Place this code in the script tags you just created.

var app = angular.module('myApp', []);

app.controller('SuperCtrl', function($scope, $http) { $http.get("https://spreadsheets.google.com/feeds/list/1vFAgyxmwrKWn9Bss3vmk5GKqn1QEZVspVTg6eMINMcc/1/public/values?alt=json").success(function(response)

{$scope.tools = response.feed.entry;});

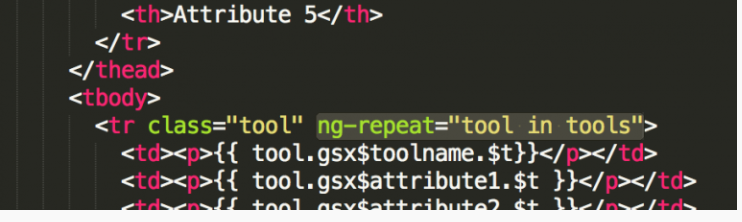
});



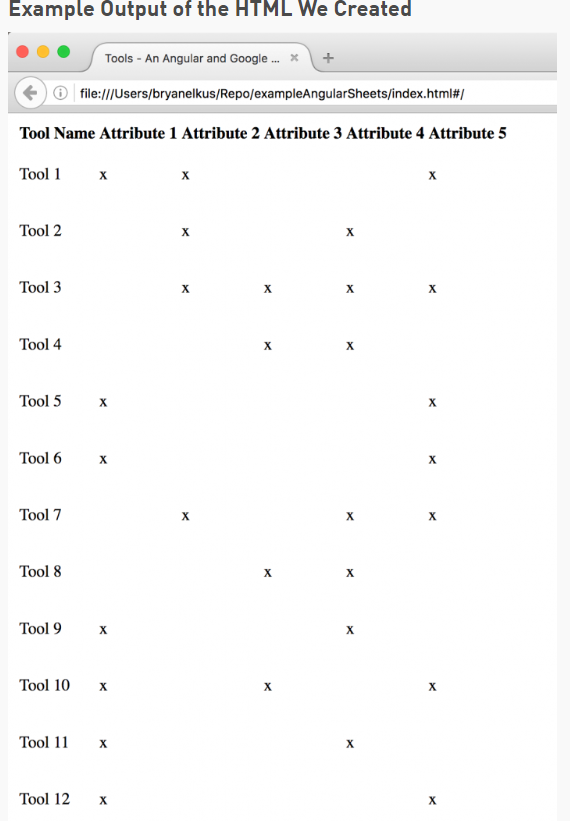


Create the HTML structure for displaying the data. In this case, I added an HTML table element.





HTML table with the data from the Google Sheets example.



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>Tools - An Angular and Google Sheet Example</title>

<meta name="generator" content="BBEdit 11.1" />

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular.min.js"></script>

</head>

<script type="text/javascript">

var app = angular.module('myApp', []);

app.controller('SuperCtrl', function($scope, $http) {

$http.get("https://spreadsheets.google.com/feeds/list/1vFAgyxmwrKWn9Bss3vmk5GKqn1QEZVspVTg6eMINMcc/1/public/values?alt=json")

.success(function(response) {$scope.tools = response.feed.entry;});

});

</script>

<body ng-app="myApp" ng-controller="SuperCtrl">

<table>

<thead>

<tr>

<th>Tool Name</th>

<th>Attribute 1</th>

<th>Attribute 2</th>

<th>Attribute 3</th>

<th>Attribute 4</th>

<th>Attribute 5</th>

</tr>

</thead>

<tbody>

<tr class="tool" ng-repeat="tool in tools">

<td><p>{{ tool.gsx$toolname.$t}}</p></td>

<td><p>{{ tool.gsx$attribute1.$t }}</p></td>

<td><p>{{ tool.gsx$attribute2.$t }}</p></td>

<td><p>{{ tool.gsx$attribute3.$t }}</p></td>

<td><p>{{ tool.gsx$attribute4.$t }}</p></td>

<td><p>{{ tool.gsx$attribute5.$t }}</p></td>

</tr>

</tbody>

</table>

<hr>

First Tool Name Value: {{ tools[0].gsx$toolname.$t }}

<br>

First Attribute Value: {{ tools[0].gsx$attribute1.$t }}

<br>

{{ tools }}

</body>

</html>