AMRITA SCHOOL OF COMPUTING

STUDENT PROFILE

Group 5

Name	Roll number
CB.EN.U4CSE20342	Nithin KM
CB.EN.U4CSE20368	Uvan Shankar SJ
CB.EN.U4CSE20462	Sivabalamurugan M
CB.EN.U4CSE20468	Tharan RC

Abstract:

Student Profile System is a web based application that helps students to showcase their profile, view their academic performance, easily view and apply for Internships and placements available ,and pay fees online. This website will allow to showcase student profile to other accounts.

As we know recruiters opt linkedin rather than a well formed resume and not all the offers coming to campus is made known to all. Most of the time important news and announcements are not reaching everyone. All the above problems will be solved using our website.

1. Login/Signup: Easy logging and signing up structure provided.

2. Home page: Provides a platform for college administration to post news, announcements, posts which will be shown to each students portals.

3. Jobs & Internships: All the job, internship and project opportunities will be posted here to apply.

4. Profile: Here students can view and edit their profile and moreover
    a. Personal details: Here a page filled with personal details and can be edited.
    b. Academic details: Here particular Academic details will be visible.
    c. Payment: payment portal and history of payments will be available.

XML-XSD:

XML Schema Definition or XSD is a recommendation by the World Wide Web Consortium (W3C) to describe and validate the structure and content of an XML document. It is primarily used to define the elements, attributes and data types the document can contain. The information in the XSD is used to verify if each element, attribute or data type in the document matches its description.

An XSD is similar to earlier XML schema languages, such as Document Type Definition (DTD), but it is a more powerful alternative as it provides greater control over the XML structure.

In general, a schema is an abstract representation of an object's characteristics and relationship to other objects in a document. An XML schema represents the relationships between the attributes and elements of an XML object.

The process of creating a schema involves analyzing the document's structure and defining each structural element encountered. For example, a schema for a document describing a website would define a website element, a webpage element and other elements that describe possible content divisions within any page on that site. These elements are defined within a set of tags in HTML and also in XML.

As the language of the XML schema, XSD is similar to a database schema that describes the data within a database. It defines the building blocks of an XML document, including:

- its attributes and elements;
- the number and order of child elements;
- the corresponding data types of multiple elements and attributes; and
- the default and fixed values for elements and attributes.

The need for XML schemas and XSD

There are hundreds of standardized XML formats that are used globally. Most of these XML standards can be defined and understood with XML schemas.

The key benefits of XML schemas, and therefore of XSD, are as follows:

- They support data types, making it easier to validate data correctness, define data facets and restrictions and convert data between different data types.
- They are written in XML, making them easier to extend and edit.
- Schemas allow the use of XML Parser to parse files, document modification with XML document object model (DOM) and transformation with Extensible Stylesheet Language Transformations (XSLT).
- Schemas help create a common language for XML content, allowing senders to describe information (e.g., date) in ways that all receivers will understand so that the data is maintained consistently.

In addition, XSD makes it easy to reuse schema in other schema, reference multiple schemas in the same document and create new data types derived from standard data types.

XSLT:

XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.

Here are the advantages of using XSLT −

  - Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
  - Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

- The XSLT processor takes one or more XML source documents, plus one or more XSLT stylesheets, and processes them to produce an output document. In contrast to widely implemented imperative programming languages like C, XSLT is declarative. The basic processing paradigm is pattern matching. Rather than listing an imperative sequence of actions to perform in a stateful environment, template rules only define how to handle a node matching a particular XPath-like pattern, if the processor should happen to encounter one, and the contents of the templates effectively comprise functional expressions that directly represent their evaluated form: the result tree, which is the basis of the processor's output.
- A typical processor behaves as follows. First, assuming a stylesheet has already been read and prepared, the processor builds a source tree from the input XML document. It then processes the source tree's root node, finds the best-matching template for that node in the stylesheet, and evaluates the template's contents. Instructions in each template generally direct the processor to either create nodes in the result tree, or to process more nodes in the source tree in the same way as the root node. Finally the result tree is serialized as XML or HTML text.

JSON:

JSON (**JavaScript Object Notation**, pronounced /ˈdʒeɪsən/; also /ˈdʒeɪˌsɒn/) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). It is a common data format with diverse uses in electronic data interchange, including that of web applications with servers.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. JSON filenames use the extension `.json`. Any valid JSON file is a valid JavaScript (`.js`) file, even though it makes no changes to a web page on its own.

JSON's basic data types are:

- Number: a signed decimal number that may contain a fractional part and may use exponential E notation, but cannot include non-numbers such as NaN. The format makes no distinction between integer and floating-point. JavaScript uses IEEE-754 double-precision floating-point format for all its numeric values (later also supporting BigInt), but other languages implementing JSON may encode numbers differently.
- String: a sequence of zero or more Unicode characters. Strings are delimited with double quotation marks and support a backslash escaping syntax.
- Boolean: either of the values `true` or `false`
- Array: an ordered list of zero or more elements, each of which may be of any type. Arrays use square bracket notation with comma-separated elements.
- Object: a collection of name–value pairs where the names (also called keys) are strings. The current ECMA standard states: "The JSON syntax does not impose any restrictions on the strings used as names, does not require that name strings be unique, and does not assign any significance to the ordering of name/value pairs." Objects are delimited with curly brackets and use commas to separate each pair, while within each pair the colon ':' character separates the key or name from its value.
- `null`: an empty value, using the word `null`

XML has been used to describe structured data and to serialize objects. Various XML-based protocols exist to represent the same kind of data structures as JSON for the same kind of data interchange purposes. Data can be encoded in XML in several ways. The most expansive form using tag pairs results in a much larger (in character count) representation than JSON, but if data is stored in attributes and 'short tag' form where the closing tag is replaced with `/>`, the representation is often about the same size as JSON or just a little larger. However, an XML attribute can only have a single value and each attribute can appear at most once on each element.

Angular JS:

AngularJS is a discontinued free and open-source JavaScript-based web framework for developing single-page applications. It was maintained mainly by Google and a community of individuals and corporations. It aimed to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures, along with components commonly used in web applications and progressive web applications.

AngularJS was used as the frontend of the MEAN stack, that consisted of MongoDB database, Express.js web application server framework, AngularJS itself (or Angular), and Node.js server runtime environment.

The AngularJS framework worked by first reading the Hypertext Markup Language (HTML) page, which had additional custom HTML attributes embedded into it. Angular interpreted those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables could be manually set within the code or retrieved from static or dynamic JSON resources.

AngularJS was built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming was better suited to defining an application's business logic. The framework adapted and extended traditional HTML to present dynamic content through two-way data-binding that allowed for the automatic synchronization of models and views. As a result, AngularJS de-emphasized explicit Document Object Model (DOM) manipulation with the goal of improving testability and performance.

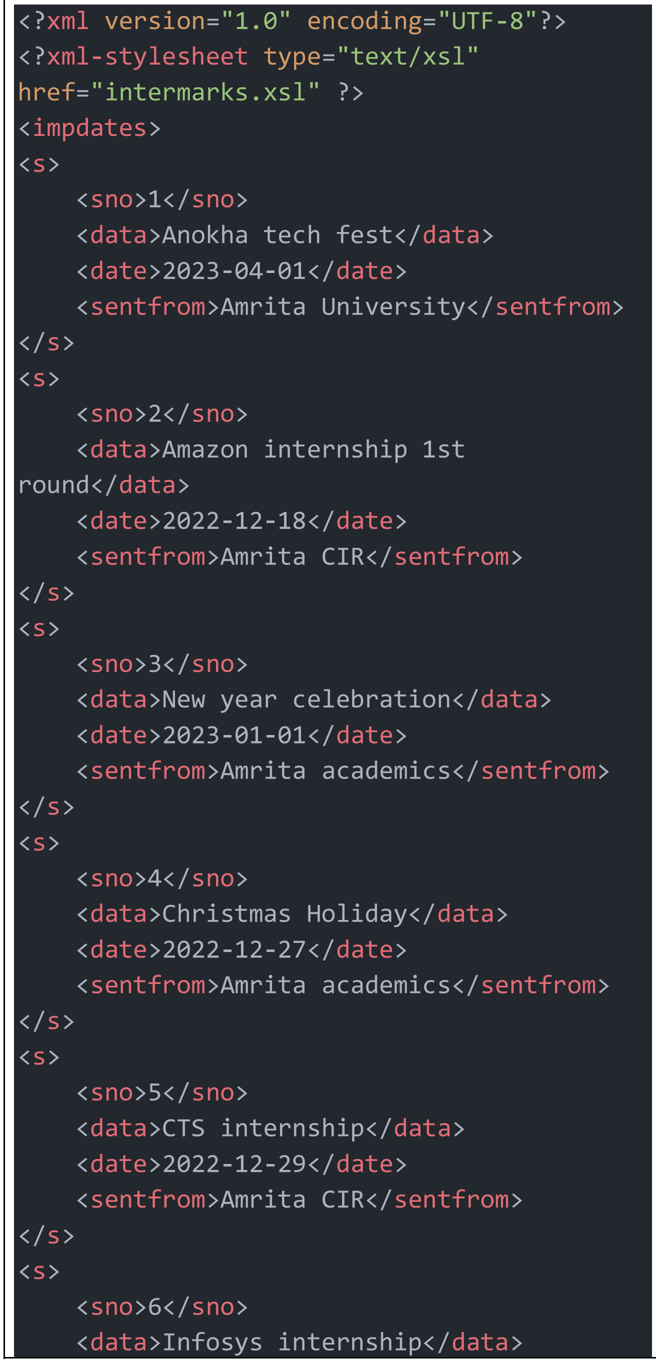
AngularJS's design goals included:

- to decouple DOM manipulation from application logic. The difficulty of this is dramatically affected by the way the code is structured.
- to decouple the client side of an application from the server-side. This allows development work to progress in parallel and allows for reuse of both sides.
- to provide structure for the journey of building an application: from designing the UI, through writing the business logic, to testing.

AngularJS implemented the MVC pattern to separate presentation, data, and logic components. Using dependency injection, Angular brought traditionally server-side services, such as view-dependent controllers, to client-side web applications. Consequently, much of the burden on the server could be reduced.

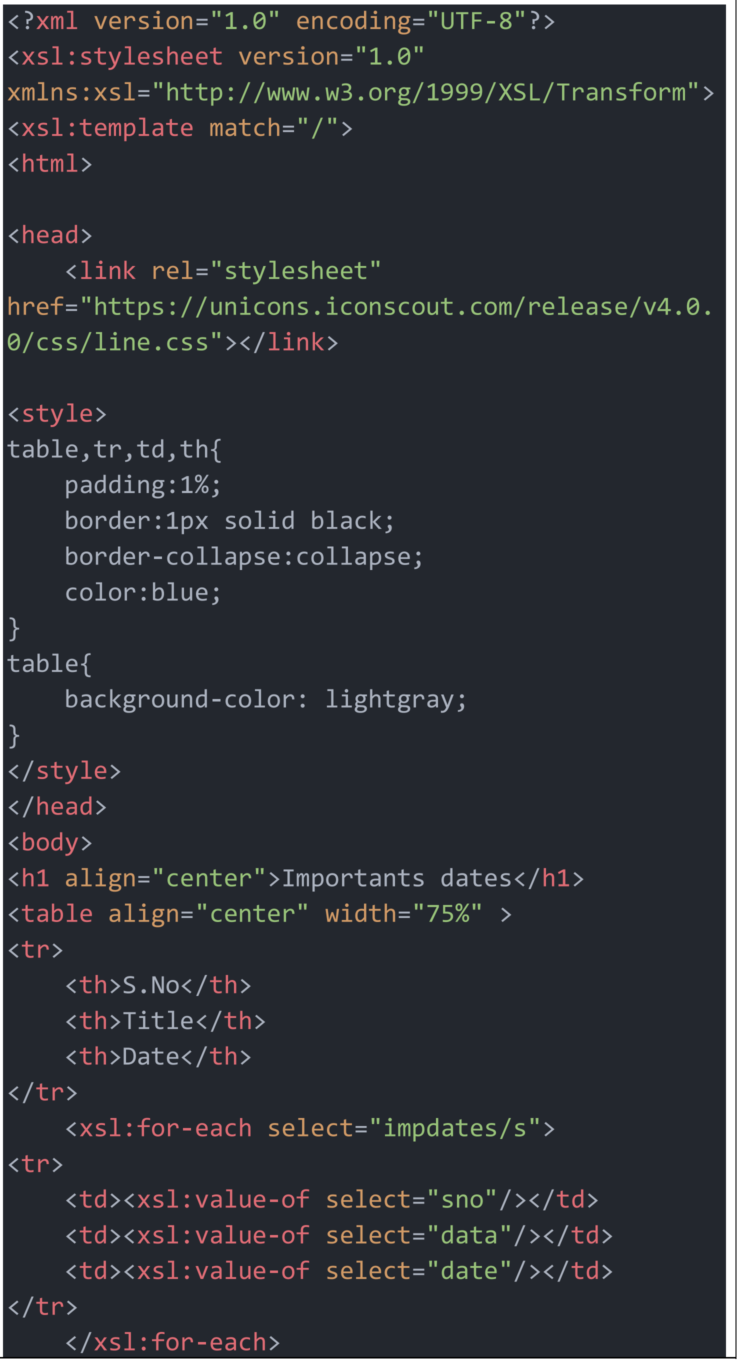
Nithin:

XML

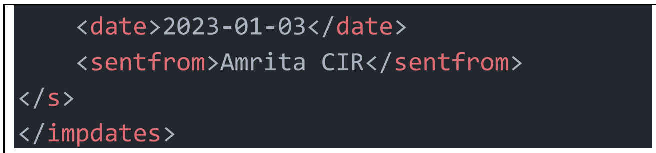


```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
href="intermarks.xsl" ?>
<impdates>
<s>
    <sno>1</sno>
    <data>Anokha tech fest</data>
    <date>2023-04-01</date>
    <sentfrom>Amrita University</sentfrom>
</s>
<s>
    <sno>2</sno>
    <data>Amazon internship 1st
round</data>
    <date>2022-12-18</date>
    <sentfrom>Amrita CIR</sentfrom>
</s>
<s>
    <sno>3</sno>
    <data>New year celebration</data>
    <date>2023-01-01</date>
    <sentfrom>Amrita academics</sentfrom>
</s>
<s>
    <sno>4</sno>
    <data>Christmas Holiday</data>
    <date>2022-12-27</date>
    <sentfrom>Amrita academics</sentfrom>
</s>
<s>
    <sno>5</sno>
    <data>CTS internship</data>
    <date>2022-12-29</date>
    <sentfrom>Amrita CIR</sentfrom>
</s>
<s>
    <sno>6</sno>
    <data>Infosys internship</data>
```

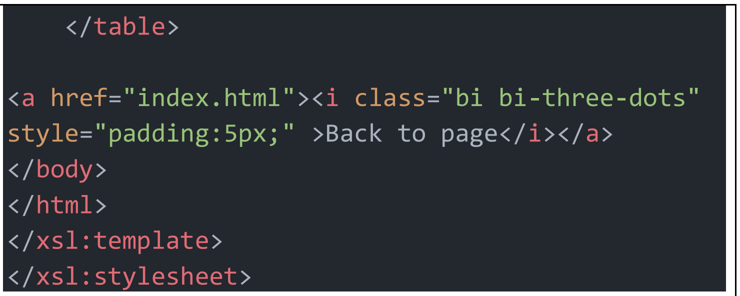
XSLT



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>

<head>
    <link rel="stylesheet"
href="https://unicons.iconscout.com/release/v4.0.
0/css/line.css"></link>

<style>
table,tr,td,th{
    padding:1%;
    border:1px solid black;
    border-collapse:collapse;
    color:blue;
}
table{
    background-color: lightgray;
}
</style>
</head>
<body>
<h1 align="center">Importants dates</h1>
<table align="center" width="75%" >
<tr>
    <th>S.No</th>
    <th>Title</th>
    <th>Date</th>
</tr>
    <xsl:for-each select="impdates/s">
<tr>
    <td><xsl:value-of select="sno"/></td>
    <td><xsl:value-of select="data"/></td>
    <td><xsl:value-of select="date"/></td>
</tr>
    </xsl:for-each>
```

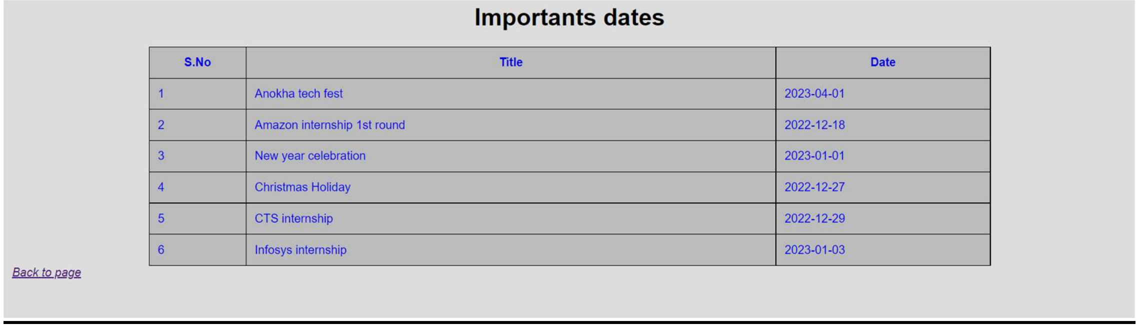


```
    <date>2023-01-03</date>
    <sentfrom>Amrita CIR</sentfrom>
</s>
</impdates>
```



```
    </table>

<a href="index.html"><i class="bi bi-three-dots"
style="padding:5px;" >Back to page</i></a>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

XSLT table used:

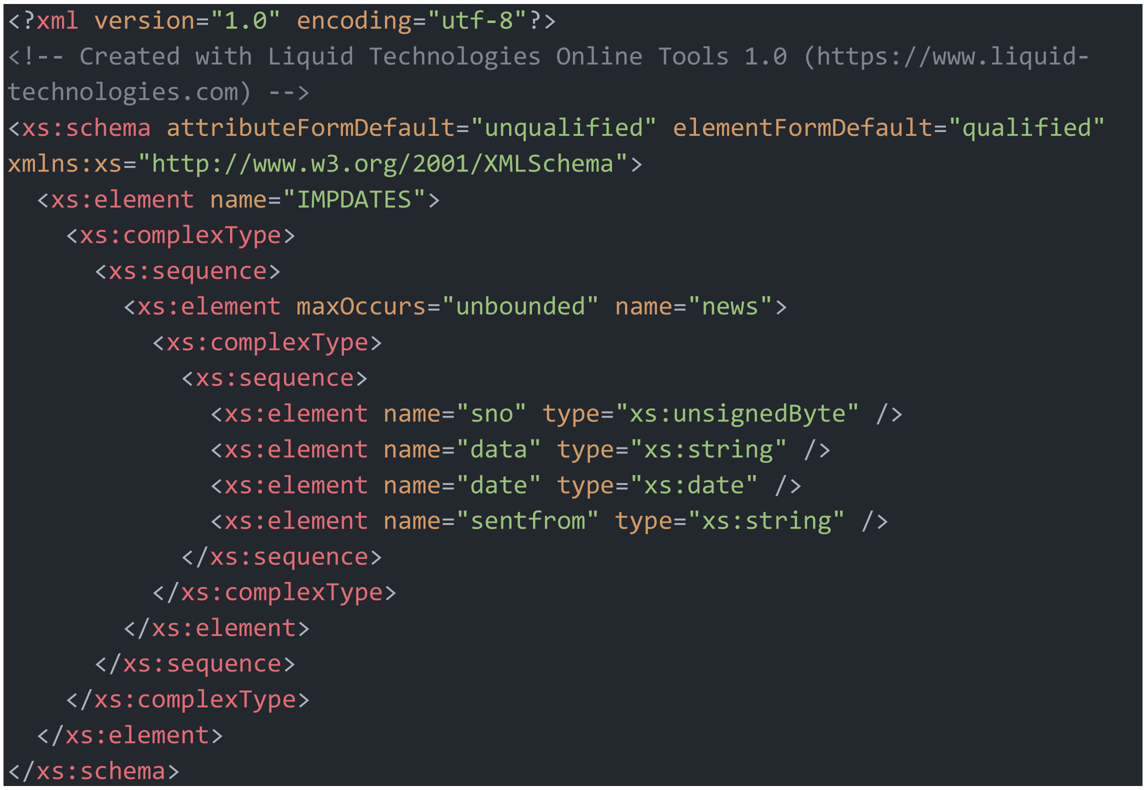


Importants dates

S.No	Title	Date
1	Anokha tech fest	2023-04-01
2	Amazon internship 1st round	2022-12-18
3	New year celebration	2023-01-01
4	Christmas Holiday	2022-12-27
5	CTS internship	2022-12-29
6	Infosys internship	2023-01-03

Back to page

XML SCHEMA:



```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid Technologies Online Tools 1.0 (https://www.liquid-
technologies.com) -->
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="IMPDATES">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="news">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="sno" type="xs:unsignedByte" />
              <xs:element name="data" type="xs:string" />
              <xs:element name="date" type="xs:date" />
              <xs:element name="sentfrom" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

JSON:

```
let jsonarr='[{"name":"Nithin",
"email":"nithin@gmail.com","password":"nithin","codesign":"CSE20342","phone":7
540001402},{"name":"Kupesh",
"email":"kupesh@gmail.com","password":"kupesh","codesign":"CSE20328","phone":7
540001402}]';
```

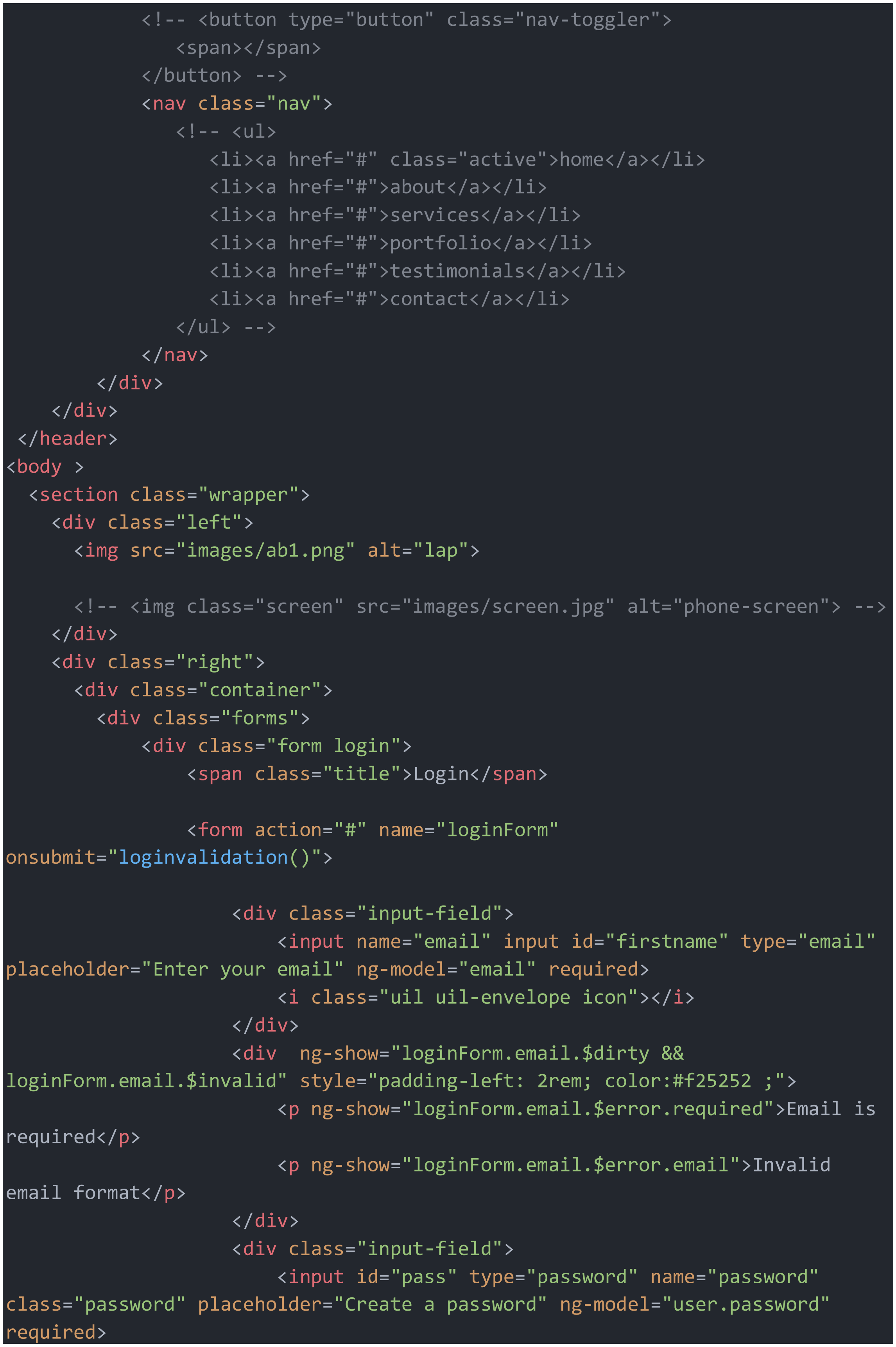
```
function loginvalidation(){
    let emaillogin=document.getElementById("firstname").value;
    let passwordlogin=document.getElementById("pass").value;
    const abc=JSON.parse(jsonarr);

    for(var i=0;i<abc.length;i++){
        var object=abc[i];
        if(object.email == emaillogin && object.password==passwordlogin){
            window.location.href="../postsection/public/index.html";
            break;
            module.exports = {curruser};
        }
        else if(emaillogin=="admin@gmail.com"&&passwordlogin=="admin"){
            window.location.href="../postsection - Copy/public/index.html";
            break;
            module.exports = {curruser};
        }
        else{
                alert("Entered details are wrong please try again");
                break;
            }
    }

}

function signUpValidation(){
    let signname=document.getElementById("namesign").value;
    let signemail=document.getElementById("emailsign").value;
    let signpassword =document.getElementById("passsign").value;
    let signconfirmpsw=document.getElementById("cpasssign").value;
    let signcodesign =document.getElementById("codesign").value;
    let signphone=document.getElementById("phonesign").value;

    alert("hello")
    let
def={name:signname,email:signemail,password:signpassword,codesign:signcodesign
,phone:signphone};
```

```
    jsonarr.push(def);

}
```

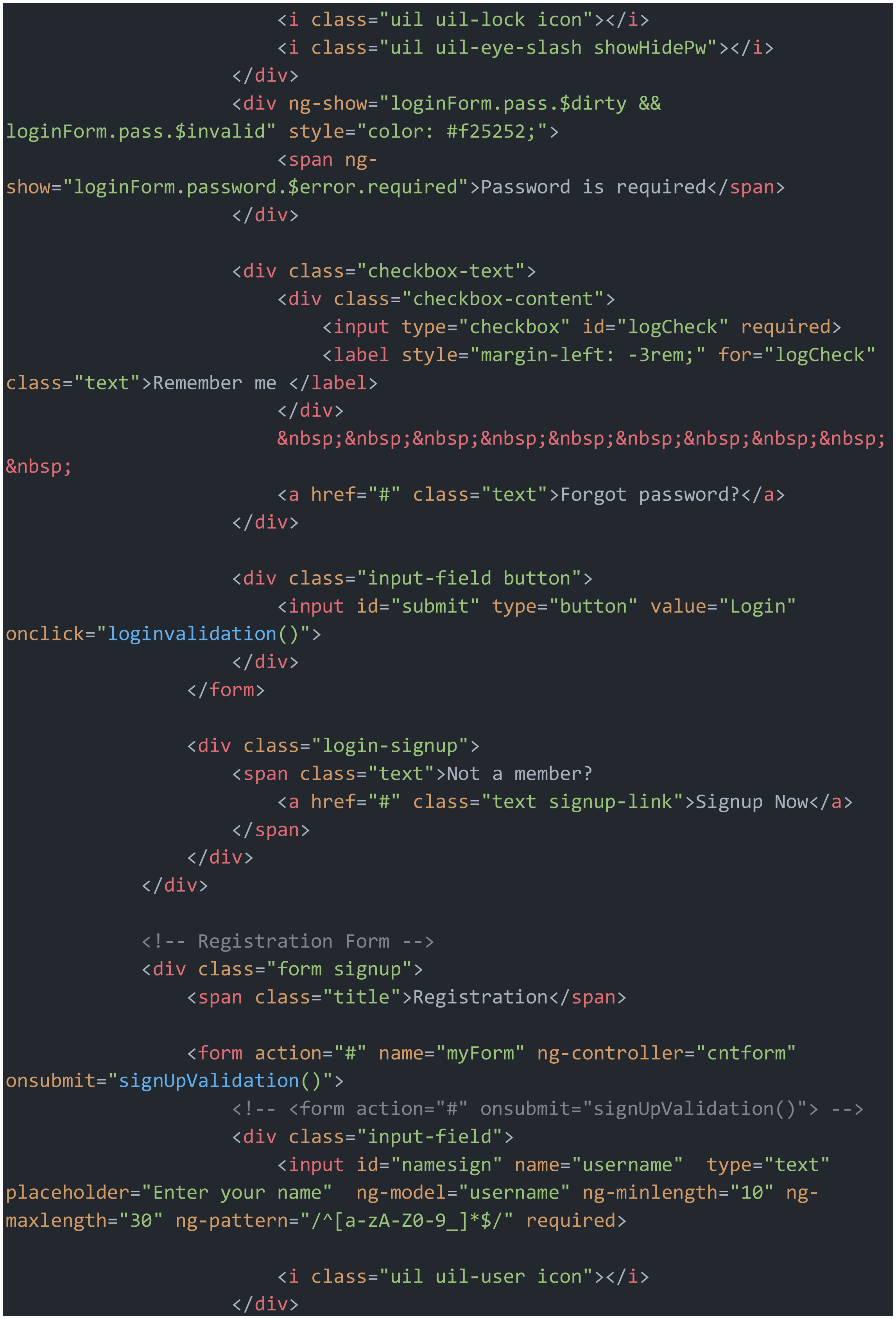
JSON is used to store all the login details of the users and verify that with the input given in the login page inorder to login to the page for that particular user.

ANGULARJS:

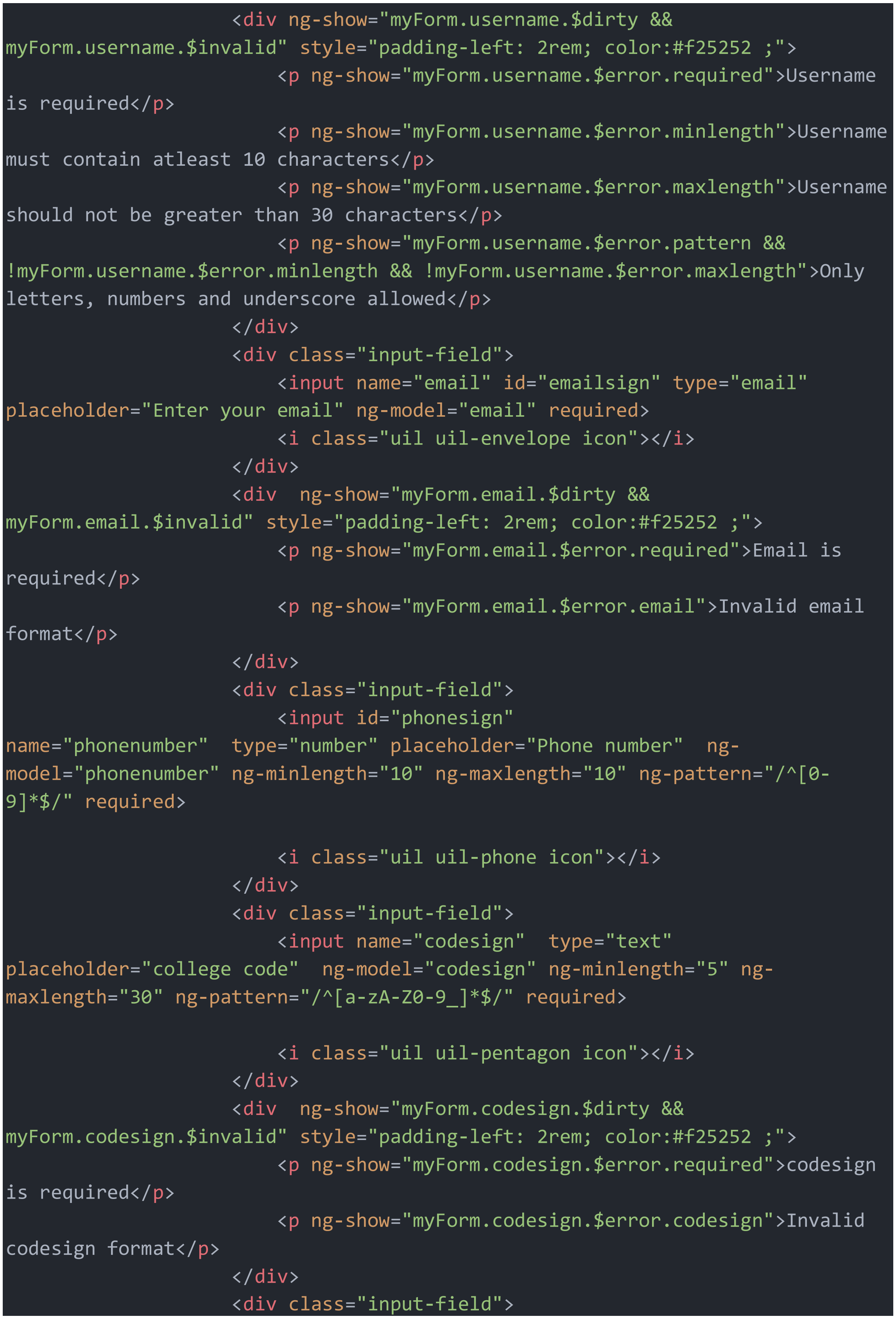
```
<!DOCTYPE html>
<html  ng-app="formvalidation">
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></s
cript>
<head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></s
cript>
  <meta charset="utf-8">
  <title>SPS</title>
  <!-- Stylesheet -->
  <link rel="stylesheet"
href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="style.css">
  <!-- Font awesome -->
  <script src="https://kit.fontawesome.com/4e6b50c3ce.js"
crossorigin="anonymous"></script>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;700&display=sw
ap" rel="stylesheet">
</head>

<header class="header">
    <div class="cont">
        <div class="row align-items-center justify-content-between">
            <div class="logo">
               <a href="#" class="abc"><b style="color:#019ED3">S</b>tudent <b
style="color:#019ED3">P</b>rofile <b style="color:#019ED3">S</b>ystem</a>
            </div>
```



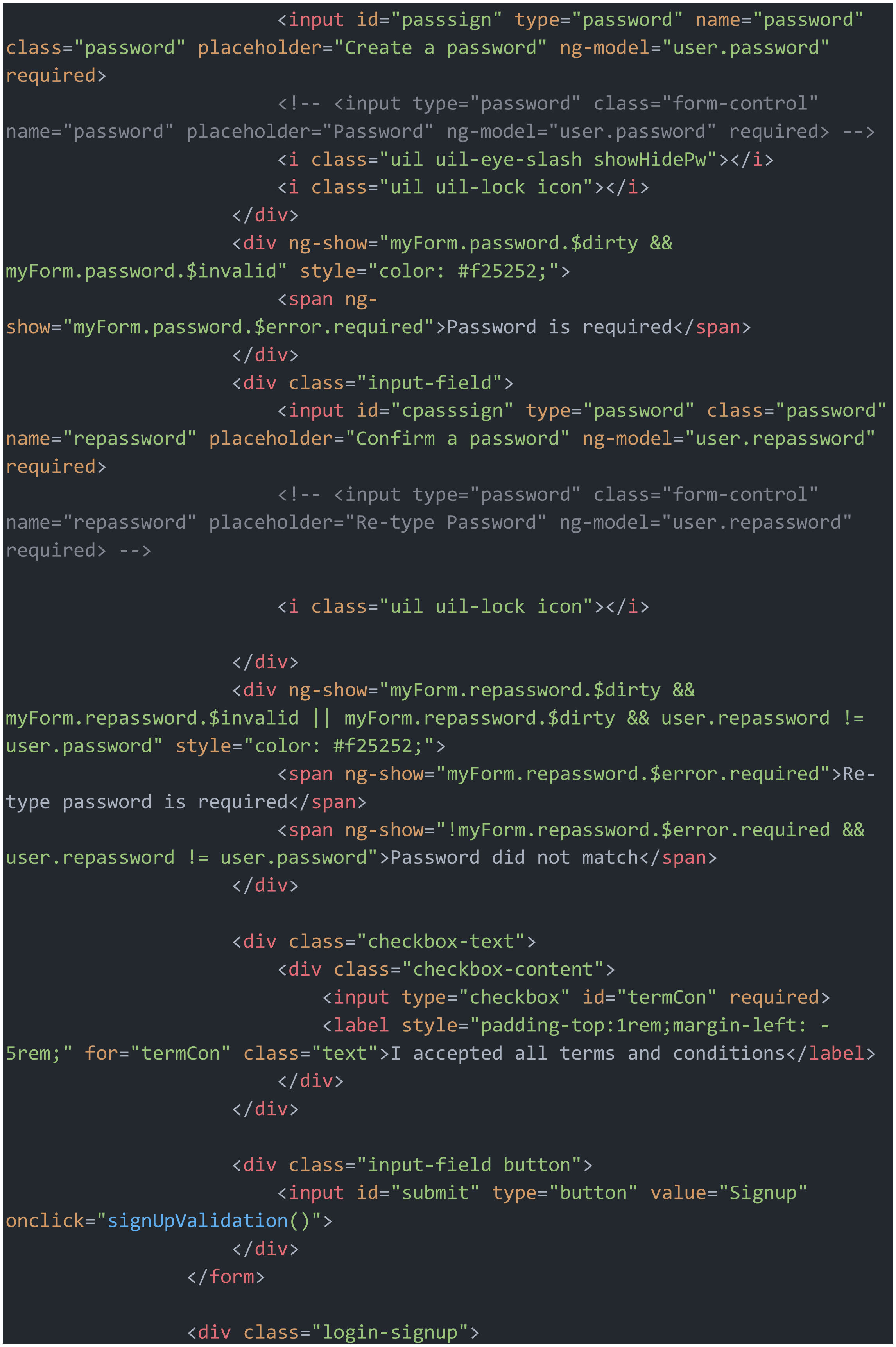
```
            <!-- <button type="button" class="nav-toggler">
               <span></span>
            </button> -->
            <nav class="nav">
               <!-- <ul>
                  <li><a href="#" class="active">home</a></li>
                  <li><a href="#">about</a></li>
                  <li><a href="#">services</a></li>
                  <li><a href="#">portfolio</a></li>
                  <li><a href="#">testimonials</a></li>
                  <li><a href="#">contact</a></li>
               </ul> -->
            </nav>
        </div>
    </div>
 </header>
<body >
  <section class="wrapper">
    <div class="left">
      <img src="images/ab1.png" alt="lap">

      <!-- <img class="screen" src="images/screen.jpg" alt="phone-screen"> -->
    </div>
    <div class="right">
      <div class="container">
        <div class="forms">
            <div class="form login">
                <span class="title">Login</span>

                <form action="#" name="loginForm"
onsubmit="loginvalidation()">

                    <div class="input-field">
                        <input name="email" input id="firstname" type="email"
placeholder="Enter your email" ng-model="email" required>
                        <i class="uil uil-envelope icon"></i>
                    </div>
                    <div  ng-show="loginForm.email.$dirty &&
loginForm.email.$invalid" style="padding-left: 2rem; color:#f25252 ;">
                        <p ng-show="loginForm.email.$error.required">Email is
required</p>
                        <p ng-show="loginForm.email.$error.email">Invalid
email format</p>
                    </div>
                    <div class="input-field">
                        <input id="pass" type="password" name="password"
class="password" placeholder="Create a password" ng-model="user.password"
required>
```



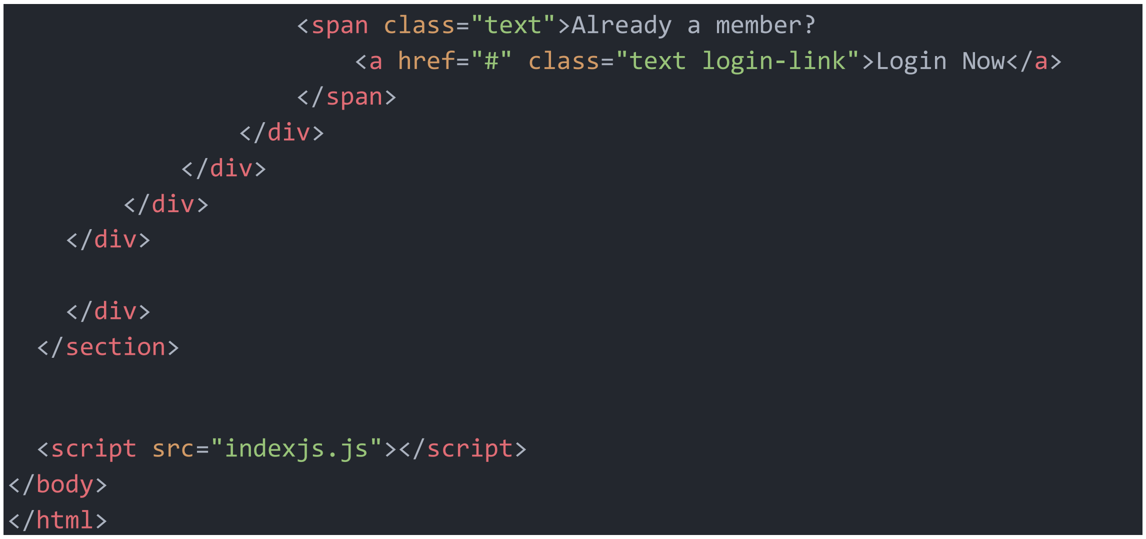
```
                        <i class="uil uil-lock icon"></i>
                        <i class="uil uil-eye-slash showHidePw"></i>
                    </div>
                    <div ng-show="loginForm.pass.$dirty &&
loginForm.pass.$invalid" style="color: #f25252;">
                        <span ng-
show="loginForm.password.$error.required">Password is required</span>
                    </div>

                    <div class="checkbox-text">
                        <div class="checkbox-content">
                            <input type="checkbox" id="logCheck" required>
                            <label style="margin-left: -3rem;" for="logCheck"
class="text">Remember me </label>
                        </div>
                        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;
                        <a href="#" class="text">Forgot password?</a>
                    </div>

                    <div class="input-field button">
                        <input id="submit" type="button" value="Login"
onclick="loginvalidation()">
                    </div>
                </form>

                <div class="login-signup">
                    <span class="text">Not a member?
                        <a href="#" class="text signup-link">Signup Now</a>
                    </span>
                </div>
            </div>

            <!-- Registration Form -->
            <div class="form signup">
                <span class="title">Registration</span>

                <form action="#" name="myForm" ng-controller="cntform"
onsubmit="signUpValidation()">
                    <!-- <form action="#" onsubmit="signUpValidation()"> -->
                    <div class="input-field">
                        <input id="namesign" name="username"  type="text"
placeholder="Enter your name"  ng-model="username" ng-minlength="10" ng-
maxlength="30" ng-pattern="/^[a-zA-Z0-9_]*$/" required>

                        <i class="uil uil-user icon"></i>
                    </div>
```



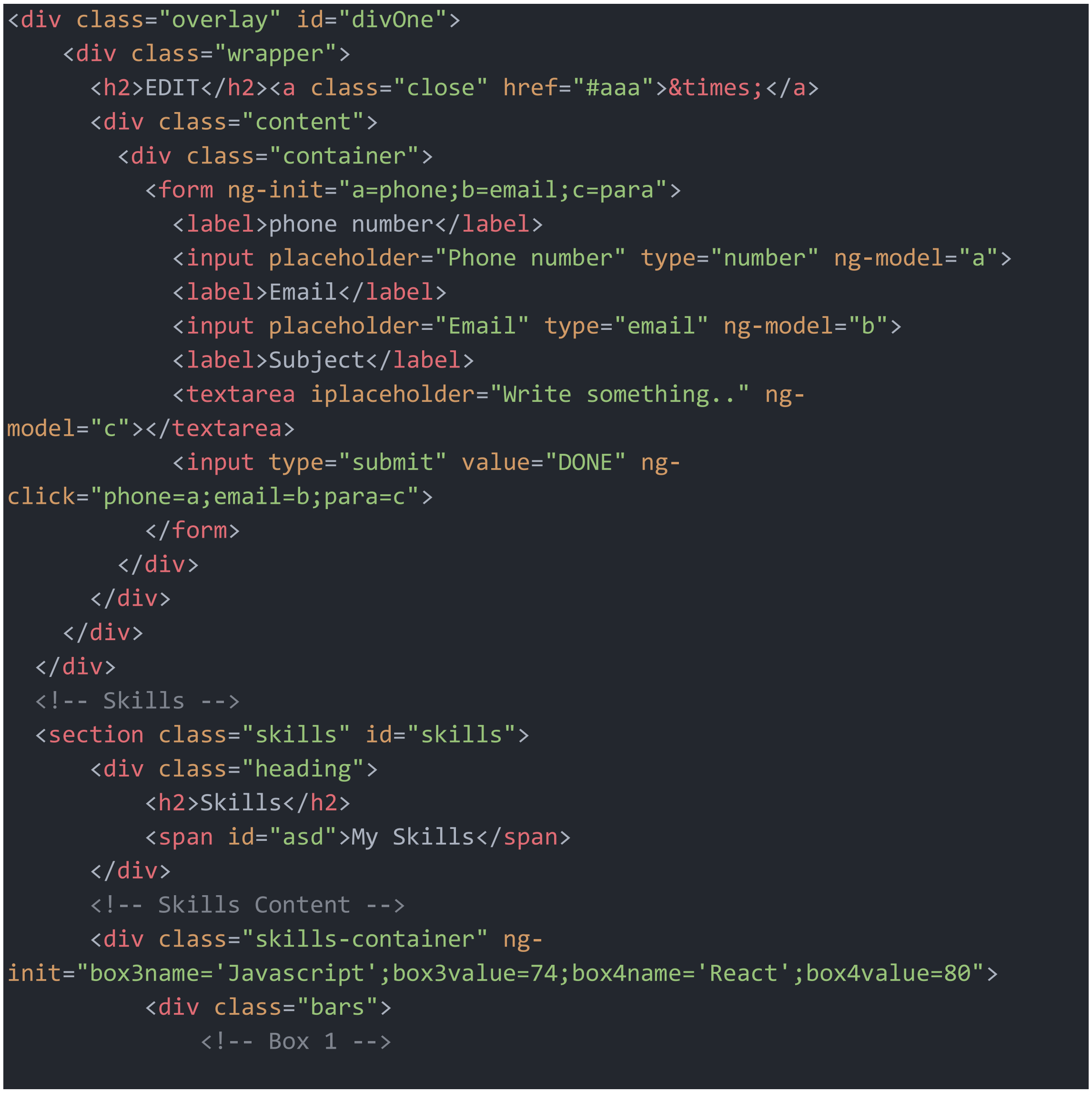
```
                    <div ng-show="myForm.username.$dirty &&
myForm.username.$invalid" style="padding-left: 2rem; color:#f25252 ;">
                        <p ng-show="myForm.username.$error.required">Username
is required</p>
                        <p ng-show="myForm.username.$error.minlength">Username
must contain atleast 10 characters</p>
                        <p ng-show="myForm.username.$error.maxlength">Username
should not be greater than 30 characters</p>
                        <p ng-show="myForm.username.$error.pattern &&
!myForm.username.$error.minlength && !myForm.username.$error.maxlength">Only
letters, numbers and underscore allowed</p>
                    </div>
                    <div class="input-field">
                        <input name="email" id="emailsign" type="email"
placeholder="Enter your email" ng-model="email" required>
                        <i class="uil uil-envelope icon"></i>
                    </div>
                    <div  ng-show="myForm.email.$dirty &&
myForm.email.$invalid" style="padding-left: 2rem; color:#f25252 ;">
                        <p ng-show="myForm.email.$error.required">Email is
required</p>
                        <p ng-show="myForm.email.$error.email">Invalid email
format</p>
                    </div>
                    <div class="input-field">
                        <input id="phonesign"
name="phonenumber"  type="number" placeholder="Phone number"  ng-
model="phonenumber" ng-minlength="10" ng-maxlength="10" ng-pattern="/^[0-
9]*$/" required>

                        <i class="uil uil-phone icon"></i>
                    </div>
                    <div class="input-field">
                        <input name="codesign"  type="text"
placeholder="college code"  ng-model="codesign" ng-minlength="5" ng-
maxlength="30" ng-pattern="/^[a-zA-Z0-9_]*$/" required>

                        <i class="uil uil-pentagon icon"></i>
                    </div>
                    <div  ng-show="myForm.codesign.$dirty &&
myForm.codesign.$invalid" style="padding-left: 2rem; color:#f25252 ;">
                        <p ng-show="myForm.codesign.$error.required">codesign
is required</p>
                        <p ng-show="myForm.codesign.$error.codesign">Invalid
codesign format</p>
                    </div>
                    <div class="input-field">
```



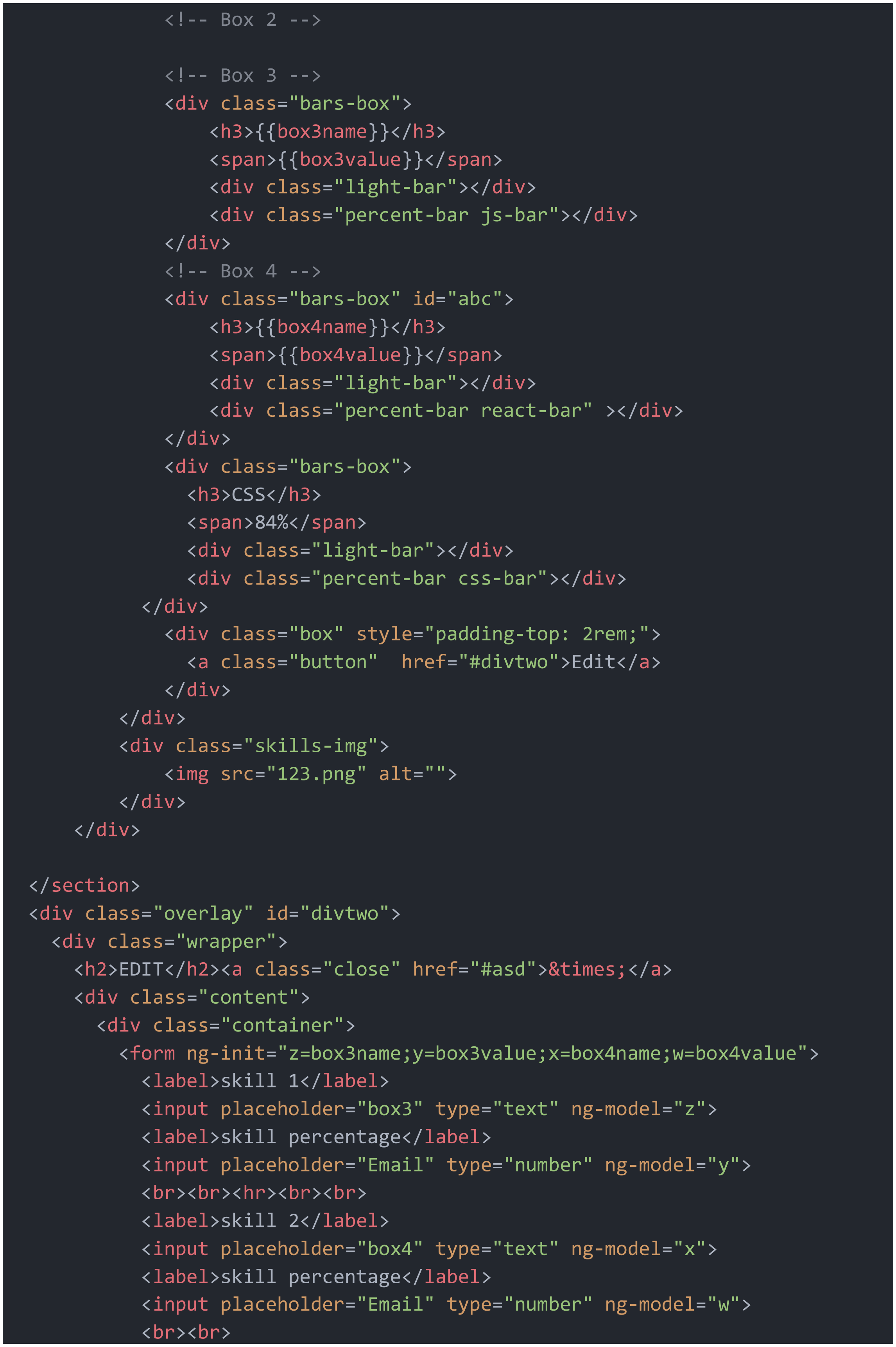
```
                        <input id="passsign" type="password" name="password"
class="password" placeholder="Create a password" ng-model="user.password"
required>
                        <!-- <input type="password" class="form-control"
name="password" placeholder="Password" ng-model="user.password" required> -->
                        <i class="uil uil-eye-slash showHidePw"></i>
                        <i class="uil uil-lock icon"></i>
                    </div>
                    <div ng-show="myForm.password.$dirty &&
myForm.password.$invalid" style="color: #f25252;">
                        <span ng-
show="myForm.password.$error.required">Password is required</span>
                    </div>
                    <div class="input-field">
                        <input id="cpasssign" type="password" class="password"
name="repassword" placeholder="Confirm a password" ng-model="user.repassword"
required>
                        <!-- <input type="password" class="form-control"
name="repassword" placeholder="Re-type Password" ng-model="user.repassword"
required> -->

                        <i class="uil uil-lock icon"></i>

                    </div>
                    <div ng-show="myForm.repassword.$dirty &&
myForm.repassword.$invalid || myForm.repassword.$dirty && user.repassword !=
user.password" style="color: #f25252;">
                        <span ng-show="myForm.repassword.$error.required">Re-
type password is required</span>
                        <span ng-show="!myForm.repassword.$error.required &&
user.repassword != user.password">Password did not match</span>
                    </div>

                    <div class="checkbox-text">
                        <div class="checkbox-content">
                            <input type="checkbox" id="termCon" required>
                            <label style="padding-top:1rem;margin-left: -
5rem;" for="termCon" class="text">I accepted all terms and conditions</label>
                        </div>
                    </div>

                    <div class="input-field button">
                        <input id="submit" type="button" value="Signup"
onclick="signUpValidation()">
                    </div>
                </form>

                <div class="login-signup">
```



```
                    <span class="text">Already a member?
                        <a href="#" class="text login-link">Login Now</a>
                    </span>
                </div>
            </div>
        </div>
    </div>

    </div>
  </section>


  <script src="indexjs.js"></script>
</body>
</html>
```



```
<div class="overlay" id="divOne">
    <div class="wrapper">
      <h2>EDIT</h2><a class="close" href="#aaa">&times;</a>
      <div class="content">
        <div class="container">
          <form ng-init="a=phone;b=email;c=para">
            <label>phone number</label>
            <input placeholder="Phone number" type="number" ng-model="a">
            <label>Email</label>
            <input placeholder="Email" type="email" ng-model="b">
            <label>Subject</label>
            <textarea iplaceholder="Write something.." ng-
model="c"></textarea>
            <input type="submit" value="DONE" ng-
click="phone=a;email=b;para=c">
          </form>
        </div>
      </div>
    </div>
  </div>
  <!-- Skills -->
  <section class="skills" id="skills">
      <div class="heading">
          <h2>Skills</h2>
          <span id="asd">My Skills</span>
      </div>
      <!-- Skills Content -->
      <div class="skills-container" ng-
init="box3name='Javascript';box3value=74;box4name='React';box4value=80">
          <div class="bars">
              <!-- Box 1 -->
```



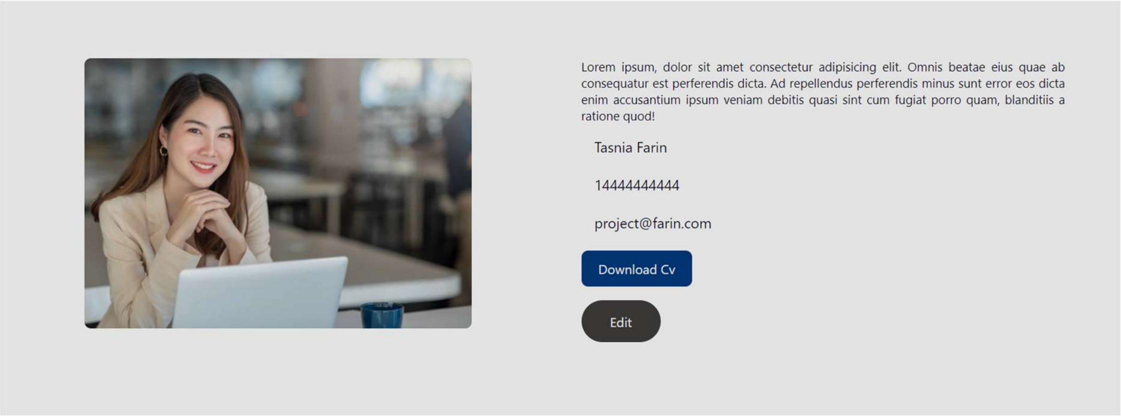
```
            <!-- Box 2 -->

            <!-- Box 3 -->
            <div class="bars-box">
                <h3>{{box3name}}</h3>
                <span>{{box3value}}</span>
                <div class="light-bar"></div>
                <div class="percent-bar js-bar"></div>
            </div>
            <!-- Box 4 -->
            <div class="bars-box" id="abc">
                <h3>{{box4name}}</h3>
                <span>{{box4value}}</span>
                <div class="light-bar"></div>
                <div class="percent-bar react-bar" ></div>
            </div>
            <div class="bars-box">
              <h3>CSS</h3>
              <span>84%</span>
              <div class="light-bar"></div>
              <div class="percent-bar css-bar"></div>
          </div>
            <div class="box" style="padding-top: 2rem;">
              <a class="button"  href="#divtwo">Edit</a>
            </div>
        </div>
        <div class="skills-img">
            <img src="123.png" alt="">
        </div>
    </div>

</section>
<div class="overlay" id="divtwo">
  <div class="wrapper">
    <h2>EDIT</h2><a class="close" href="#asd">&times;</a>
    <div class="content">
      <div class="container">
        <form ng-init="z=box3name;y=box3value;x=box4name;w=box4value">
          <label>skill 1</label>
          <input placeholder="box3" type="text" ng-model="z">
          <label>skill percentage</label>
          <input placeholder="Email" type="number" ng-model="y">
          <br><br><hr><br><br>
          <label>skill 2</label>
          <input placeholder="box4" type="text" ng-model="x">
          <label>skill percentage</label>
          <input placeholder="Email" type="number" ng-model="w">
          <br><br>
```

```
            <input type="submit" value="DONE" ng-
click="box3name=z;box3value=y;box4name=x;box4value=w"  onclick="changecss(f,g)
">
          </form>
        </div>
      </div>
    </div>
  </div>
```

Angular JS is used for login and signup form validation and I have used as edit the content in the profile section also.

Validation:

Before:

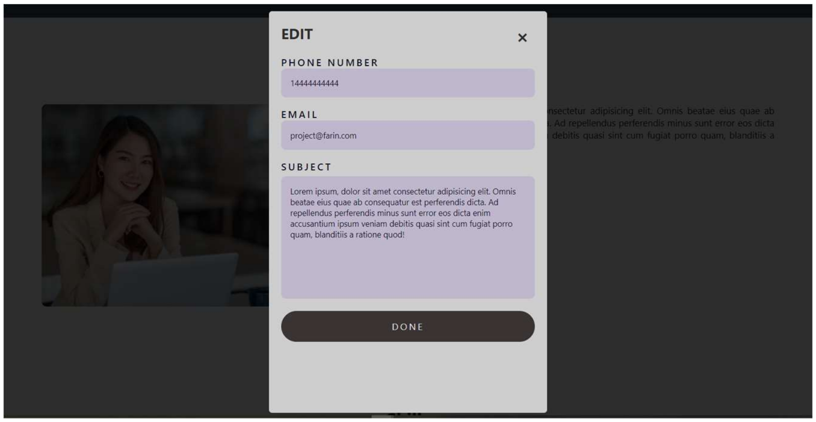


Lorem ipsum, dolor sit amet consectetur adipisicing elit. Omnis beatae eius quae ab consequatur est perferendis dicta. Ad repellendus perferendis minus sunt error eos dicta enim accusantium ipsum veniam debitis quasi sint cum fugiat porro quam, blanditiis a ratione quod!

Tasnia Farin

14444444444

project@farin.com



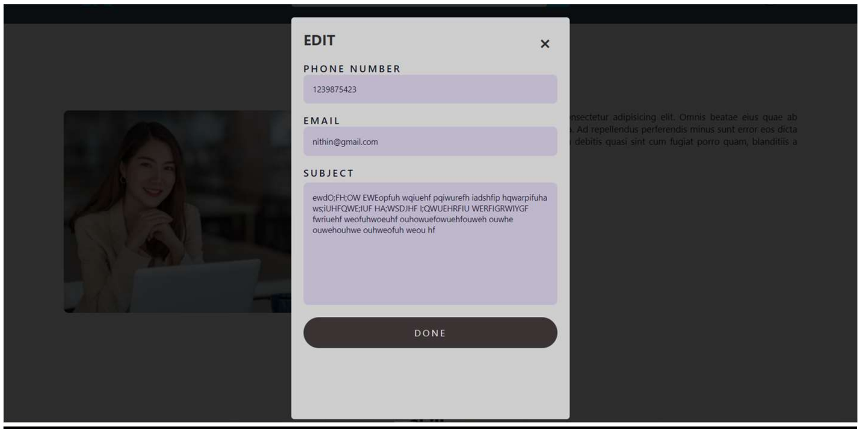
EDIT

PHONE NUMBER: 14444444444

EMAIL: project@farin.com

SUBJECT: Lorem ipsum, dolor sit amet consectetur adipisicing elit. Omnis beatae eius quae ab consequatur est perferendis dicta. Ad repellendus perferendis minus sunt error eos dicta enim accusantium ipsum veniam debitis quasi sint cum fugiat porro quam, blanditiis a ratione quod!

After:



PHONE NUMBER

1239875423

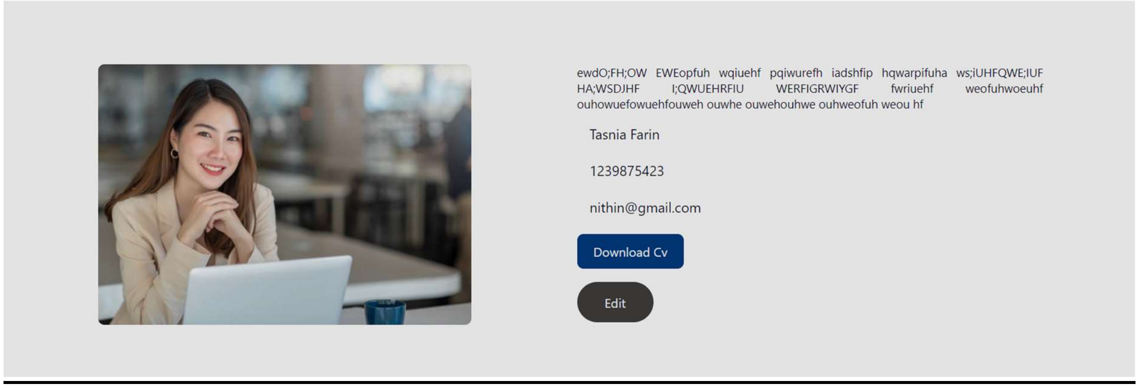
EMAIL

nithin@gmail.com

SUBJECT

ewdO;FH;OW EWEopfuh wqiuehf pqiwurefh iadshfip hqwarpifuha ws;iUHFQWE;IUF HA;WSDJHF I;QWUEHRFIU WERFIGRWIYGF fwriuehf weofuhwoeuhf ouhowuefowuehfouweh ouwhe ouwehouhwe ouhweofuh weou hf

nsectetur adipisicing elit. Omnis beatae eius quae ab
. Ad repellendus perferendis minus sunt error eos dicta
debitis quasi sint cum fugiat porro quam, blanditiis a

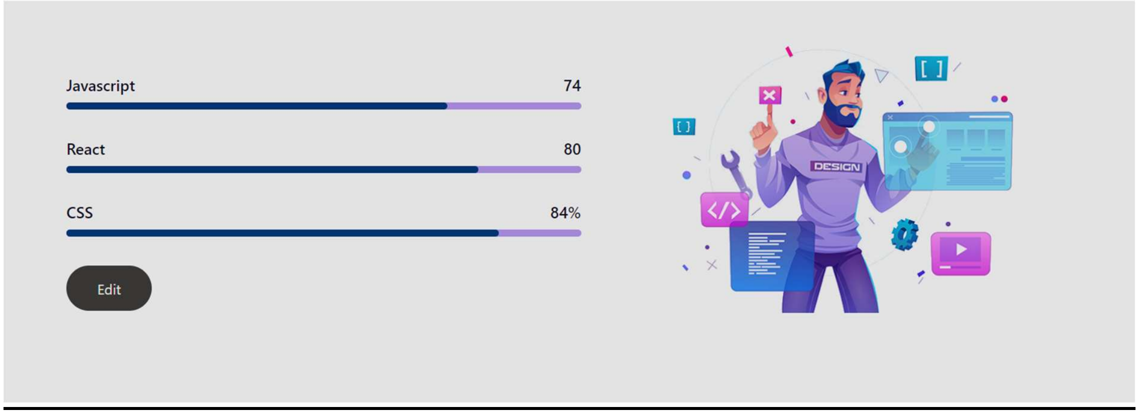


ewdO;FH;OW EWEopfuh wqiuehf pqiwurefh iadshfip hqwarpifuha ws;iUHFQWE;IUF HA;WSDJHF I;QWUEHRFIU WERFIGRWIYGF fwriuehf weofuhwoeuhf ouhowuefowuehfouweh ouwhe ouwehouhwe ouhweofuh weou hf

Tasnia Farin

1239875423

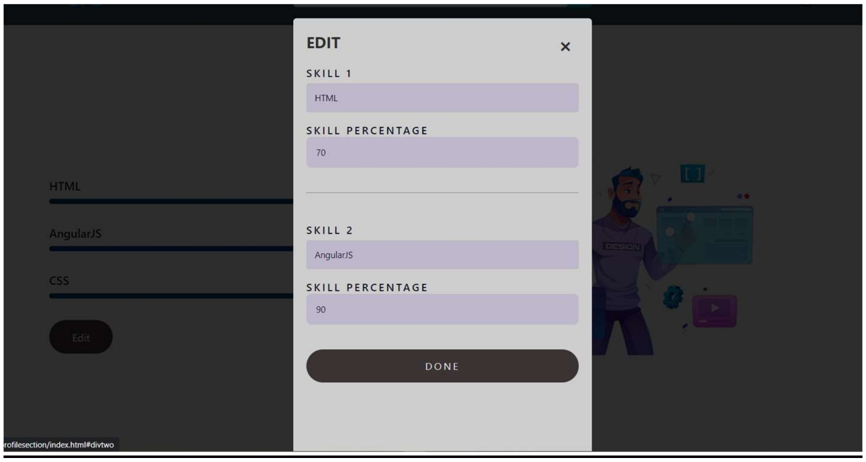
nithin@gmail.com



Javascript 74

React 80

CSS 84%



EDIT

SKILL 1

HTML

SKILL PERCENTAGE

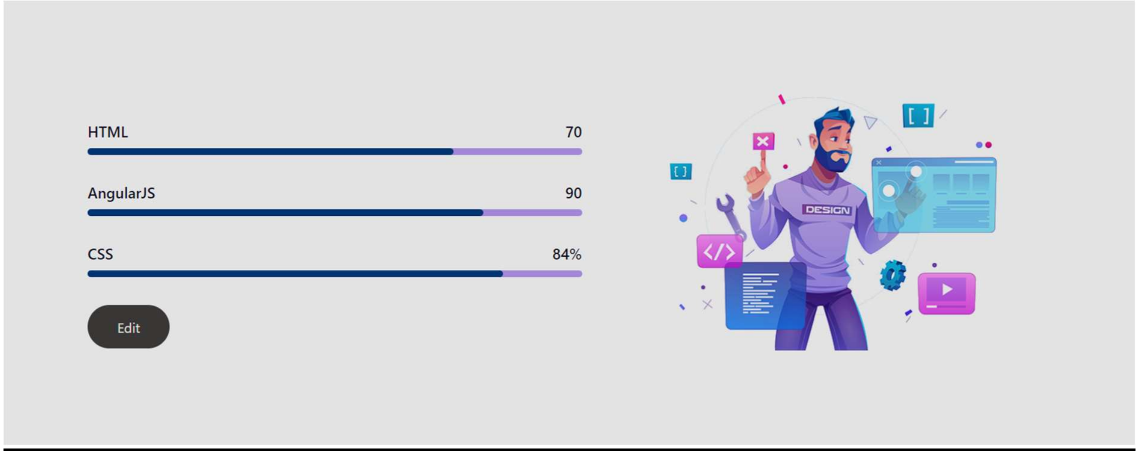
70

SKILL 2

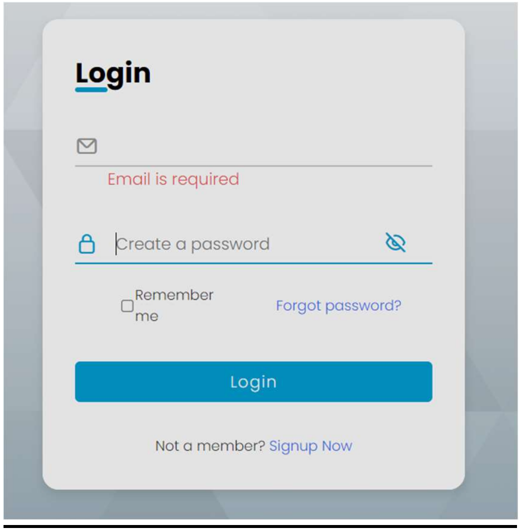
AngularJS

SKILL PERCENTAGE

90



HTML	70
AngularJS	90
CSS	84%

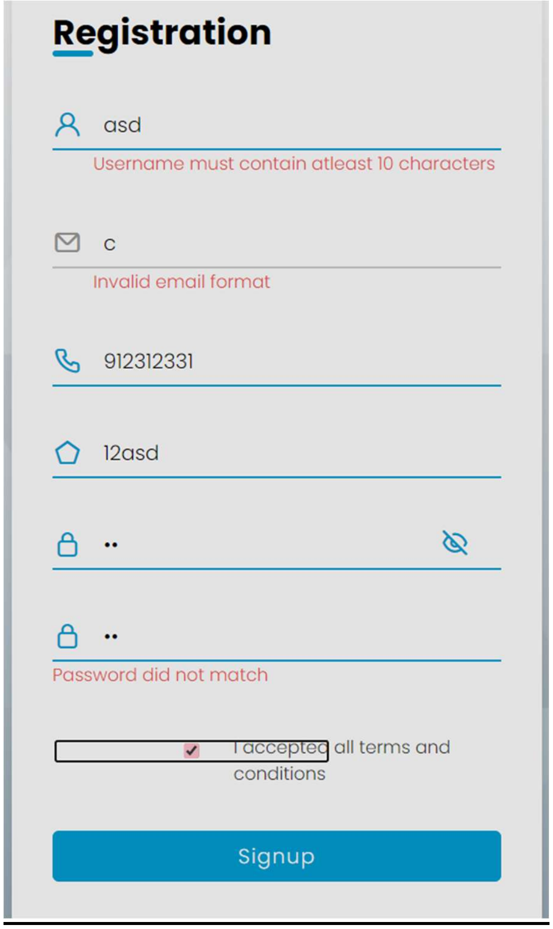


Login

Email is required

Create a password

□ Remember me



Registration

asd

Username must contain atleast 10 characters

c

Invalid email format

912312331

12asd

••

••

Password did not match

☑ I accepted all terms and conditions