PALLAVI ENGINEERING COLLEGE

Approved by AICTE, New Delhi & Affiliated to JNTU-Hyderabad
Certified by ISO 9001 : 2015 | ISO 14001 : 2015 | ISO 50001 : 2018
Nagole, Hyderabad, R.R Dist 501505, Telangana State, India
www.pallaviengineeringcollege.ac.in

Department of CSE-CYBER SECURITY

SKILL DEVELOPMENT COURSE (NODE JS/ REACT JS/ DJANGO)

II B. Tech – II Semester

Branch: CSE-Cyber Security

Mr. SABAVATH RAJU
Assistant Professor

Pallavi Engineering College

Kuntloor(v), Abdullapurmet(M), Hyderabad, R.R. District

SKILL DEVELOPMENT COURSE (NODE JS/ REACT JS/ DJANGO)

B.Tech. II Year II Sem. **L T P C**
0 0 2 1

Prerequisites: Object Oriented Programming through Java, HTML Basics.

Course Objectives:

- To implement the static web pages using HTML and do client side validation using JavaScript.
- To design and work with databases using Java.
- To develop an end to end application using java full stack.
- To introduce Node JS implementation for server side programming.
- To experiment with single page application development using React.

Course Outcomes:

At the end of the course, the student will be able to,

- Build a custom website with HTML, CSS, and Bootstrap and little JavaScript.
- Demonstrate Advanced features of JavaScript and learn about JDBC
- Develop Server – side implementation using Java technologies like
- Develop the server – side implementation using Node JS.
- Design a Single Page Application using React.

Exercises:

1. Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.
2. Make the above web application responsive web application using Bootstrap framework.
3. Use JavaScript for doing client – side validation of the pages implemented in experiment 1 and experiment 2.
4. Explore the features of ES6 like arrow functions, callbacks, promises, async/await. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page.
5. Develop a java standalone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables.
6. Create an xml for the bookstore. Validate the same using both DTD and XSD.
7. Design a controller with servlet that provides the interaction with application developed in experiment 1 and the database created in experiment 5.
8. Maintaining the transactional history of any user is very important. Explore the various session tracking mechanism (Cookies, HTTP Session)
9. Create a custom server using http module and explore the other modules of Node JS like OS, path, event.
10. Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)
11. For the above application create authorized end points using JWT (JSON Web Token).

12. Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.
13. Create a service in react that fetches the weather information from openweathermap.org and the display the current and historical weather information using graphical representation using chart.js
14. Create a TODO application in react with necessary components and deploy it into github.

REFERENCE BOOKS:

1. Jon Duckett, Beginning HTML, XHTML, CSS, and JavaScript, Wrox Publications, 2010.
2. Bryan Basham, Kathy Sierra and Bert Bates, Head First Servlets and JSP, O'Reilly Media, 2nd Edition, 2008.
3. Vasan Subramanian, Pro MERN Stack, Full Stack Web App Development with Mongo, Express, React, and Node, 2nd Edition, A Press.

1. Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.

HTML:

- HTML stands for HyperText Markup Language.
- It is the standard language used to create and design web pages on the internet.
- It was introduced by Tim Berners-Lee in 1991 as a simple markup language. Since then, it has evolved through versions from HTML 2.0 to HTML5 (the latest 2024 version).
- HTML is a combination of Hypertext and Markup language.
- Hypertext defines the link between the web pages and Markup language defines the text document within the tag.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" href="./style.css">
   <title>Home - PEC</title>
</head>
<body>
   <div class="wrapper">
      <div class="container">
         <header>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tr>
            <th width="20%"><img src="PEC.png" alt="PEC LOGO"  width="130"
height="100"/> </th>
            <th colspan=4>
            <h1 style="color:white;">PEC - WORLD BEST ONLINE EBOOKS WEBSITE</h1>
            </th>
            </tr>
            </table>
         </header>
         <nav>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tbody align="center"  style="font-weight:bold;font-size:18px;"">
            <tr>
            <td width="20%"><hr><a href="index.html">Home</a><hr></td>
            <td width="20%"><hr><a href="login.html">Login</a><hr></td>
            <td width="20%"><hr><a href="registration.html">Registration</a><hr></td>
            <td width="20%"><hr><a href="cart.html" >Cart</a><hr></td>
            </tr>
            </tbody>
            </table>
```

```
        </nav>
      </div>
      <div class="container1">
        <div class="sidebar1"></div>
        <div class="container2">
          <main>
            <center>
            <h2>Welcome to PEC e-Book's Website</h2>
              <p>Shopping at <font size=5>PEC</font> can be both <font size=5>fun</font>
              and <font size=5>savings</font>.</br>Shop with us in this special
<font size=5>discount</font> season and save upto <font size=5>90%</font>
on all your purchases.</br></p>
              <br/><br/><br/><br/><br/><br/><br/><br/>
          </main>
        </div>
        <div class="sidebar2"></div>
      </div>
      <footer><font color="white">(C) 2024 All rights reserved by PEC ebooks</font></footer>
    </div>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="./style.css">
    <title>Login - PEC</title>
</head>
<body>
    <div class="wrapper">
        <div class="container">
            <header>
                <table width="100%" align="center" cellpadding="0" cellspacing="2">
                <tr>
                <th width="20%"><img src="PEC.png" alt="PEC LOGO"  width="130"
height="100"/></th>
                <th colspan=4>
                <h1 style="color:white;">PEC - WORLD BEST ONLINE EBOOKS WEBSITE</h1>
                </th>
                </tr>
                </table>
            </header>
            <nav>
```

```
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tbody align="center"  style="font-weight:bold;font-size:18px;"">
            <tr>
            <td width="20%"><hr><a href="index.html">Home</a><hr></td>
            <td width="20%"><hr><a href="login.html">Login</a><hr></td>
            <td width="20%"><hr><a href="registration.html">Registration</a><hr></td>
            <td width="20%"><hr><a href="cart.html" >Cart</a><hr></td>
            </tr>
            </tbody>
            </table>
          </nav>
        </div>
        <div class="container1">
          <div class="sidebar1"></div>
          <div class="container2">
            <main>
              <center><br>
              <h3> Login Details</h3> <br/>
                <form name="f1">
                  <table width="100%" align="center" >
                  <tr>
                  <td> User Name : </td>
                  <td> <input type="text" name="username"></td>
                  </tr>
                  <tr><td><br></td></tr>
                  <tr>
                  <td> Password : </td>
                  <td> <input type="password" name="password"></td>
                  </tr>
                  <tr><td><br></td></tr>
                  <tr><td></td>
                  <td><input type="submit" value="SUBMIT">
                  <input type="reset" value="RESET"></td>
                  </tr>
                  </table>
                </form>
              </center>
            </main>
          </div>
          <div class="sidebar2"></div>
        </div>
        <footer><font color="white">(C) 2024 All rights reserved by PEC ebooks</font></footer>
      </div>
    </body>
    </html>
```

registration.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" href="./style.css">
   <title>Registration - PEC</title>
</head>
<body>
   <div class="wrapper">
      <div class="container">
         <header>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tr>
            <th width="20%"><img src="PEC.png" alt="PEC LOGO"  width="130"
height="100"/></th>
            <th colspan=4>
            <h1 style="color:white;"><marquee>PEC - WORLD BEST ONLINE EBOOKS
WEBSITE;"</marquee></h1>
            </th>
            </tr>
            </table>
         </header>
         <nav>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tbody align="center"  style="font-weight:bold;font-size:18px;"">
            <tr>
            <td width="20%"><hr><a href="index.html">Home</a><hr></td>
            <td width="20%"><hr><a href="login.html">Login</a><hr></td>
            <td width="20%"><hr><a href="registration.html">Registration</a><hr></td>
            <td width="20%"><hr><a href="cart.html" >Cart</a><hr></td>
            </tr>
            </tbody>
            </table>
         </nav>
      </div>
      <div class="container1">
         <div class="sidebar1"></div>
         <div class="container2">
            <main>
               <center><br>
                  <h3>Registration Form </h3>
                  <br/>
                  <form name="f1">
```

```
                <table cellpadding="1" align="center" >
                <tr><td> Name:*</td>
                <td><input type="text" name="username"></td></tr>
                <tr><td>Password:*</td>
                <td><input type="password" name="password"></td></tr>
                <tr><td>Email ID:*</td>
                <td><input type="text" name="email"></td></tr>
                <tr><td>Phone Number:*</td>
                <td><input type="text" name="phno"></td></tr>
                <tr><td valign="top">Gender:*</td>
                <td><input type="radio" name="radio" value="1">Male   
                <input type="radio" name="radio" value="2">Female</td></tr>
                <tr> <td valign="top">Language Known:*</td>
                <td> <input type="checkbox" name="checkbox" value="English">English<br/>
                <input type="checkbox" name="checkbox" value="Telugu">Telugu<br>
                <input type="checkbox" name="checkbox" value="Hindi">Hindi<br>
                <input type="checkbox" name="checkbox" value="Tamil">Tamil
                </td></tr>
                <tr> <td valign="top">Address:*</td>
                <td><textarea name="address"></textarea></td>
                <tr><td></td><td><input type="submit" value="submit" hspace="10">
                <input type="reset" value="reset"></td></tr>
                <tr> <td colspan=2 >*<font color="#FF0000">fields are mandatory</font>
                </td>
                </tr>
                </table>
                </form>
              </center>
            </main>
          </div>
          <div class="sidebar2"></div>
      </div>
      <footer><font color="white">(C) 2024 All rights reserved by PEC ebooks</font></footer>
   </div>
</body>
</html>
```

cart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" href="./style.css">
   <title>Cart - PEC</title>
</head>
```

```
<body>
   <div class="wrapper">
      <div class="container">
         <header>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tr>
            <th width="20%"><img src="PEC.png" alt="PEC LOGO"  width="130"
height="100"/></th>
            <th colspan=4>
            <h1 style="color:white;">PEC - WORLD BEST ONLINE EBOOKS WEBSITE</h1>
            </th>
            </tr>
            </table>
         </header>
         <nav>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tbody align="center"  style="font-weight:bold;font-size:18px;"">
            <tr>
            <td width="20%"><hr><a href="index.html">Home</a><hr></td>
            <td width="20%"><hr><a href="login.html">Login</a><hr></td>
            <td width="20%"><hr><a href="registration.html">Registration</a><hr></td>
            <td width="20%"><hr><a href="cart.html" >Cart</a><hr></td>
            </tr>
            </tbody>
            </table>
         </nav>
      </div>
      <div class="container1">
         <div class="sidebar1"></div>
         <div class="container2">
            <main>
               <center>
               <h3>Cart</h3>
                  <table width="100%" align="center" >
                  <tbody>
                  <tr>
                  <th width="40%"><hr>BookName<hr></th>
                  <th width="20%"><hr>Price<hr></th>
                  <th width="20%"><hr>Quantity<hr></th>
                  <th width="20%"><hr>Amount<hr></th> </tr>
                  </tbody>
                  <tbody align=center>
                  <tr> <td>Java Programming </td>
                  <td>Rs. 2300/-</td>
                  <td>2</td>
                  <td>Rs. 4600/-</td></tr>
```

```
                <tr><td>Web Technologies</td>
                <td>Rs. 3000/-</td>
                <td>1</td>
                <td>Rs. 3000/-</td></tr>
                <tr><td></td>
                <td><hr><font color="#996600">Total Amount:</font><hr></td>
                <td><hr>3<hr></td>
                <td><hr>Rs. 7600/-<hr></td> </tr>
                </tbody>
                </table>
              </center>
            </main>
          </div>
          <div class="sidebar2"></div>
        </div>
        <footer><font color="white">(C) 2024 All rights reserved by PEC ebooks</font></footer>
    </div>
</body>
</html>
```

CSS:

- CSS is the acronym for "Cascading Style Sheet".
- It's a style sheet language used for describing the presentation of a document written in a markup language like HTML.
- CSS helps the web developers to control the layout and other visual aspects of the web pages.
- CSS plays a crucial role in modern web development by providing the tools necessary to create visually appealing, accessible, and responsive websites.

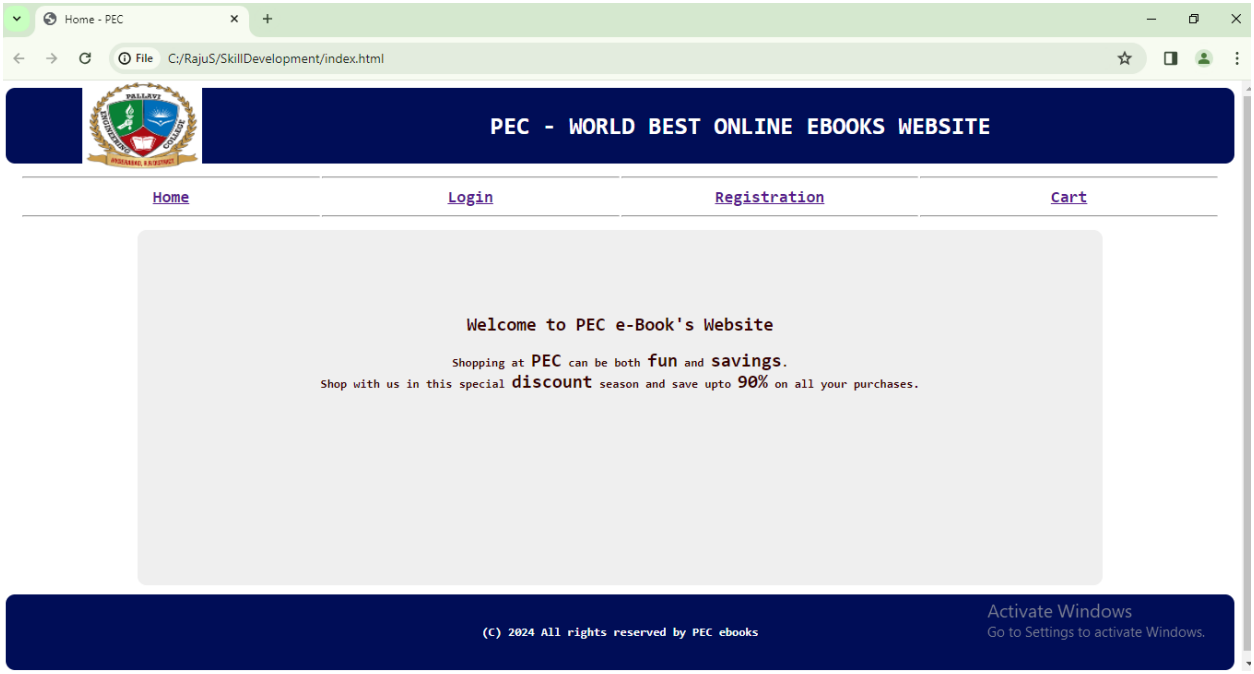
style.css

```
body{
  font-family: monospace;
}
main {
  background-color: #efefef;
  color: #330000;
  margin-left: 10px;
  height: 60vh;
}
 header, footer {
  background-color: #000d57;
  color: #fff;
  padding: 1rem;
  height: 50px;
}
header, nav{
```

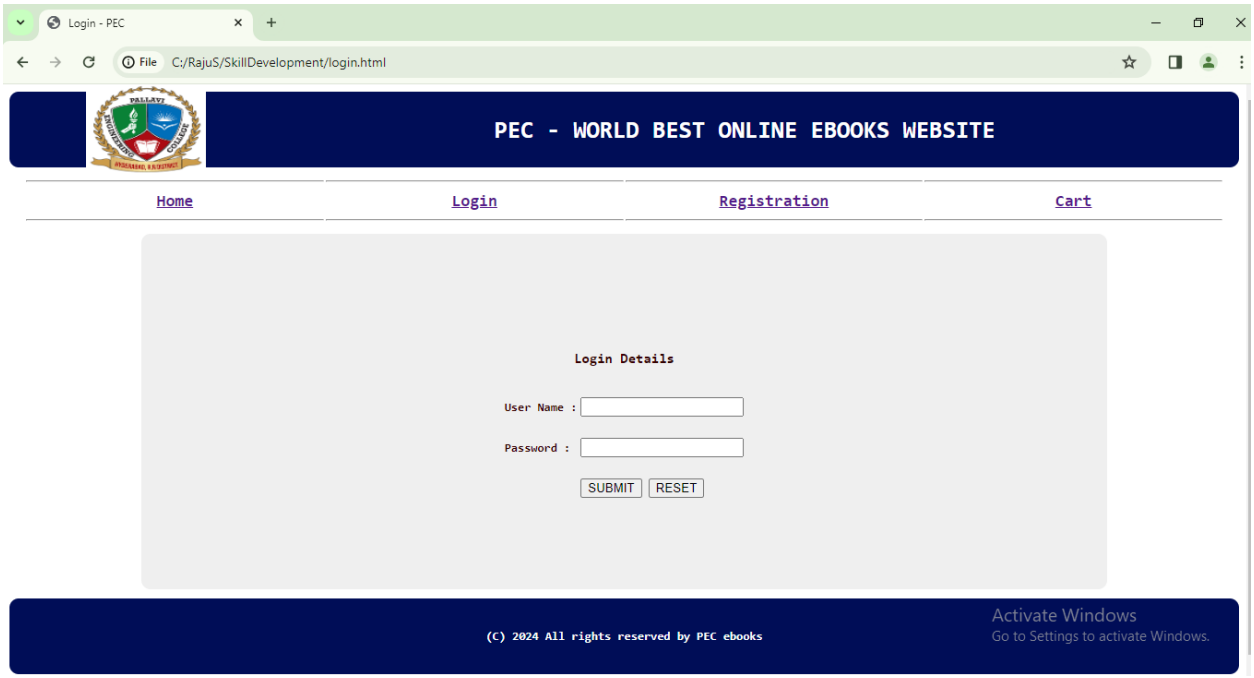
```
  margin-bottom: 10px;
  flex-basis: 50%;
}
footer{
  margin-top: 10px;
}
nav {
  background-color: #fff;
  color: #000;
  padding: 1rem;
  height: 20px;
}
.sidebar1, .sidebar2 {
  flex-basis: 10%;
  background-color: #fff;
  color: #000;
}
.sidebar2{
  margin-left: 10px;
}
.container1{
  display: flex;
}
.container2 {
  display: flex;
  flex-direction: column;
  flex: 1;
}
header, nav, main, .sidebar1, .sidebar2, footer{
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 10px;
}
.wrapper {
  display: flex;
  flex-direction: column;
  font-weight: 600;
}
```

Output:

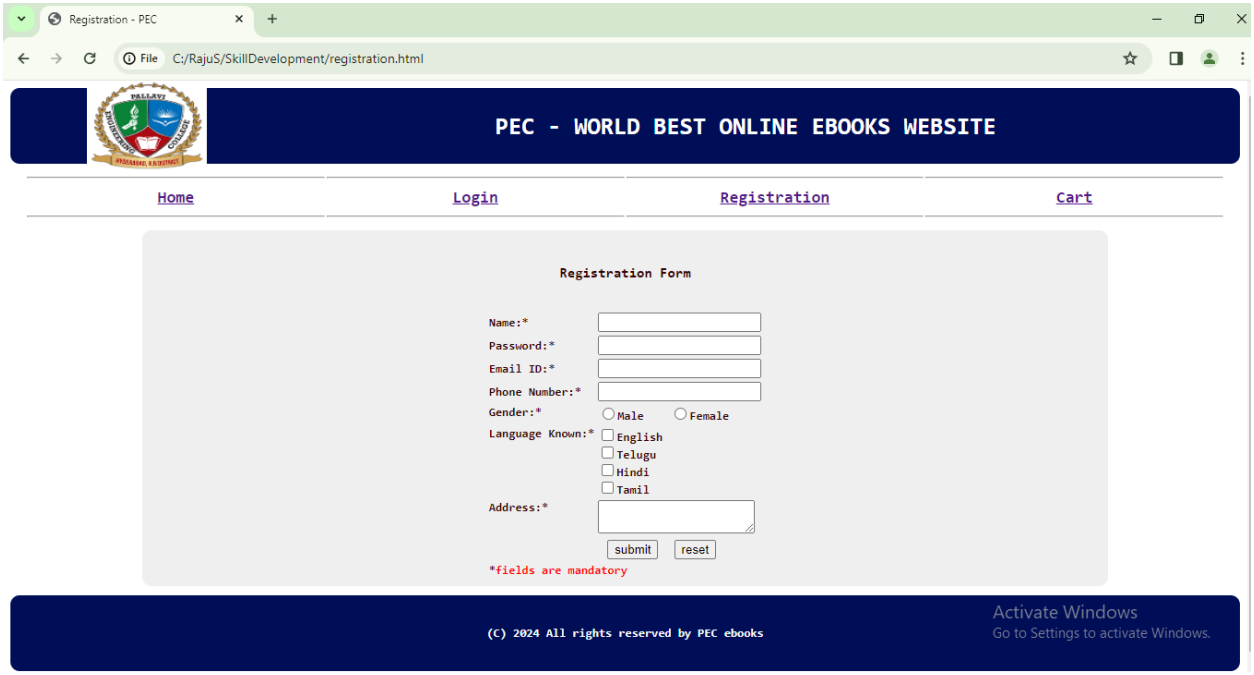
index.html



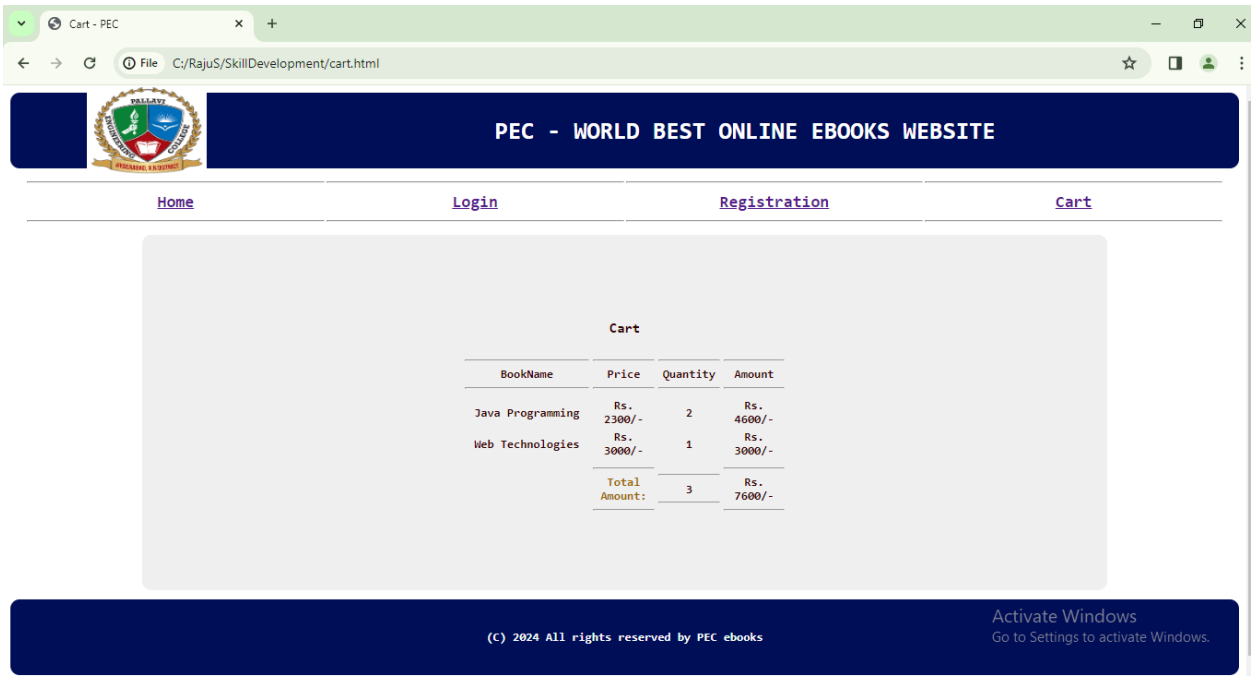
login.html



registration.html



cart.html



2. Make the above web application responsive web application using Bootstrap framework.

Bootstrap:

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

- Bootstrap is completely free to download and use.
- Front end languages like HTML, CSS, and JavaScript for this framework, as it holds the libraries for these languages. You must import the bootstrap CSS and JS files to use the framework in your project.

Framework:

A framework, or software framework, is a platform that provides a foundation for developing software applications. Think of it as a template of a working program that can be selectively modified by adding code. It uses shared resources – such as libraries, image files, and reference documents – and puts them together in one package. That package can be modified to suit the specific needs of the project. With a framework, the developer can add or replace features to give new functionality to the application.

Back-End Web Frameworks:

Back-end web frameworks, or just web frameworks, are the most commonly used programming frameworks. Web frameworks assist developers in making web applications and dynamic websites.

Common back-end web frameworks include:

- **Django** – an open-source Python web development framework.
- **Rails** – another open-source framework written in the Ruby Language, a programming language designed specifically to be easy to use. Rails is used to run websites like Airbnb, Github, and Shopify.

Front-End Frameworks:

Where back-end web frameworks are loaded on a server, front-end frameworks are executed in a user's browser. They allow web developers to design what the users see on the website, things like the management of AJAX requests, defining file structures, and styling the website's components

The most common front-end frameworks are:

- **Angular JS** – a front-end JavaScript framework. It was developed and is supported by Google.
- **React** – another front-end JavaScript framework developed by the team at Facebook to help them make changes to the site's code easily.
- **Bootstrap** – a front-end CSS framework that is a collection of reusable HTML, CSS, and JavaScript code. Having all this code pre-defined in a downloadable file allows for developers and designers to save time creating fully-responsive websites.

Mobile Development Frameworks:

With the rise of mobile rapidly gaining traction, mobile development frameworks have seen an equally impressive uptick. Like their desktop counterparts, mobile development frameworks give developers a structure that supports the mobile app-building process.

Common mobile development frameworks are:

- **Flutter** – A cross-platform app framework that forms native code, meaning you can use one programming language and code base to create an app for both Apple and Android. It was developed by Google and is free and open-source.
- **React Native** – A cross-platform app framework that also forms a native code developed by the team at Facebook. It was created with both JavaScript and ReactJS programming languages and comes with ready-made components that can save developers time as they don't have to create elements from scratch.

AIM: Make the above web application responsive web application using Bootstrap framework.
DESCRIPTION: Bootstrap is a popular CSS framework that makes it easy to create responsive web applications. The previous example can be modified using Bootstrap by following these steps:

Project Structure:
1. index.html - Main HTML file containing the structure of the web application with Bootstrap.
2. script.js - JavaScript file for handling interactions and logic (no changes from the previous example).
3. styles.css - You can include additional custom styles if needed.
4. images/ - Folder for storing images.

Index.html:

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">

 <!-- Custom CSS -->
 <link rel="stylesheet" href="styles.css">
 <title>Shopping Cart</title>
 </head>
 <body>
 <header class="bg-dark text-white text-center py-3">
 <h1>Shopping Cart</h1>
 <nav>
```

```
 <ul class="nav justify-content-center">
 <li class="nav-item"><a class="nav-link" href="#catalog">Catalog</a></li>
 <li class="nav-item"><a class="nav-link" href="#cart">Cart</a></li>
 <li class="nav-item"><a class="nav-link" href="#login">Login</a></li>
 <li class="nav-item"><a class="nav-link" href="#register">Register</a></li>
 </ul>
 </nav>
 </header>
 <main class="container mt-3" id="content">
 <!-- Content will be loaded dynamically using JavaScript -->
 </main>
 <!-- Bootstrap JS (optional, for certain features) -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
 <script src="script.js">
</script>
 </body>
</html>
```

Styles.css:

```
/* You can include additional custom styles here if needed */
```

Explanation:

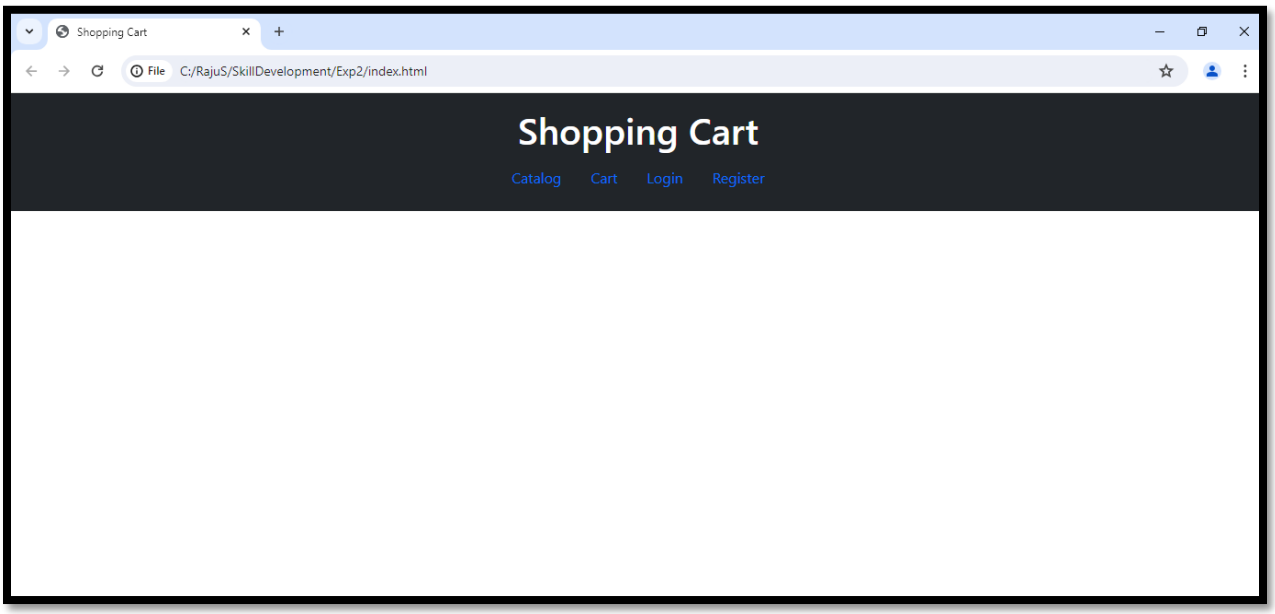
1. Bootstrap Integration: In the <head> section, we added links to the Bootstrap CSS and JS files from a CDN (Content Delivery Network). This allows us to use Bootstrap's styling and functionality.
2. Bootstrap Classes: We applied Bootstrap classes to the HTML elements. For example, we used container to create a responsive fixed-width container and various utility classes for styling the header and navigation.
3. Responsive Navigation: Bootstrap's grid system and utility classes help in creating a Responsive navigation bar. The justify-content-center class is used to center the navigation links.
4. Responsive Main Content: The container class ensures that the main content area is responsive. Bootstrap will automatically adjust the width based on the screen size.

Output:

- When you open index.html in a web browser, you'll see that the web application is now responsive.
- The Bootstrap framework takes care of making the layout adapt to different screen sizes, providing a more user-friendly experience on various devices.
- Remember to test the responsiveness by resizing your browser or using different devices to see how the layout adjusts.

Output before applying Bootstrap:

Output after applying Bootstrap:



3. Use JavaScript for doing client – side validation of the pages implemented in experiment 1 and experiment 2.

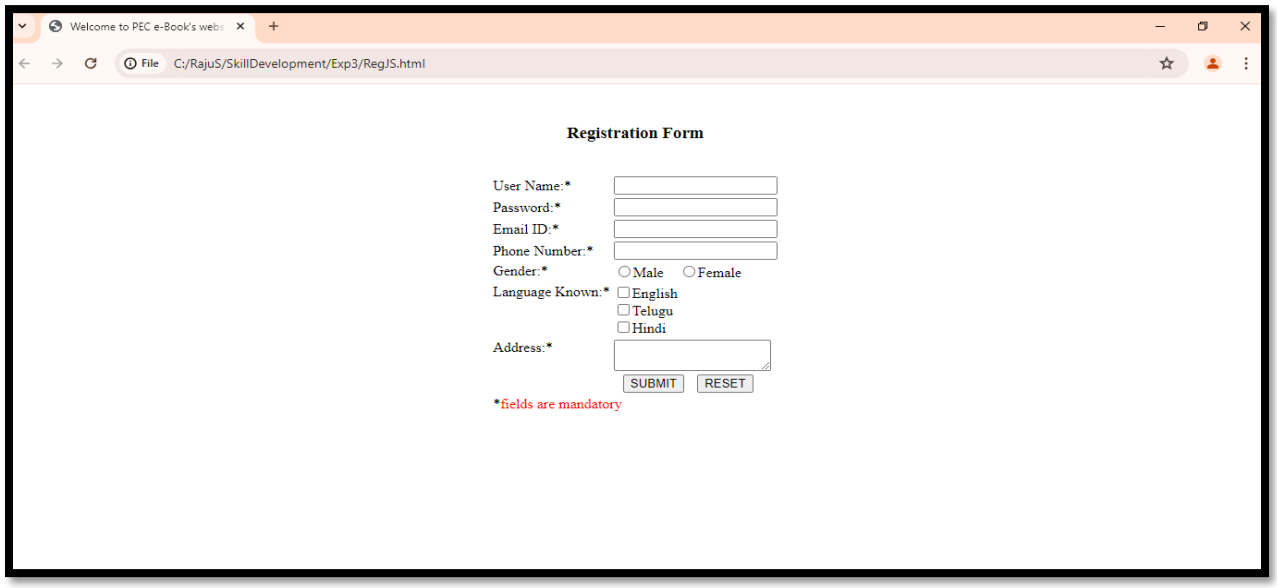
```
<html>
<head>
   <title> Welcome to PEC e-Book's website</title>
   <script language="javascript">
   function validate()
         {
      // username validation
      var uname = f1.username.value;
      if (uname.length<=0)
      {
         alert("Please Enter UserName");
         f1.username.focus();
         return false;
      }
      if (uname.length < 8)
      {
         alert("Please enter UserName not less than 8");
         f1.username.focus();
         return false;
      }
      //password validation
      var pwd = f1.password.value;
      if (pwd.length<=0)
      {
         alert("Please Enter password");
         f1.password.focus();
         return false;
      }
      if (pwd.length < 6)
      {
         alert("Please enter Password not less than 6");
         f1.password.focus();
         return false;
      }
      // email validation
      var email = f1.email.value;
      if (email.length<=0)
      {
         alert("Please Enter email");
         f1.email.focus();
         return false;
      }
      else {
```

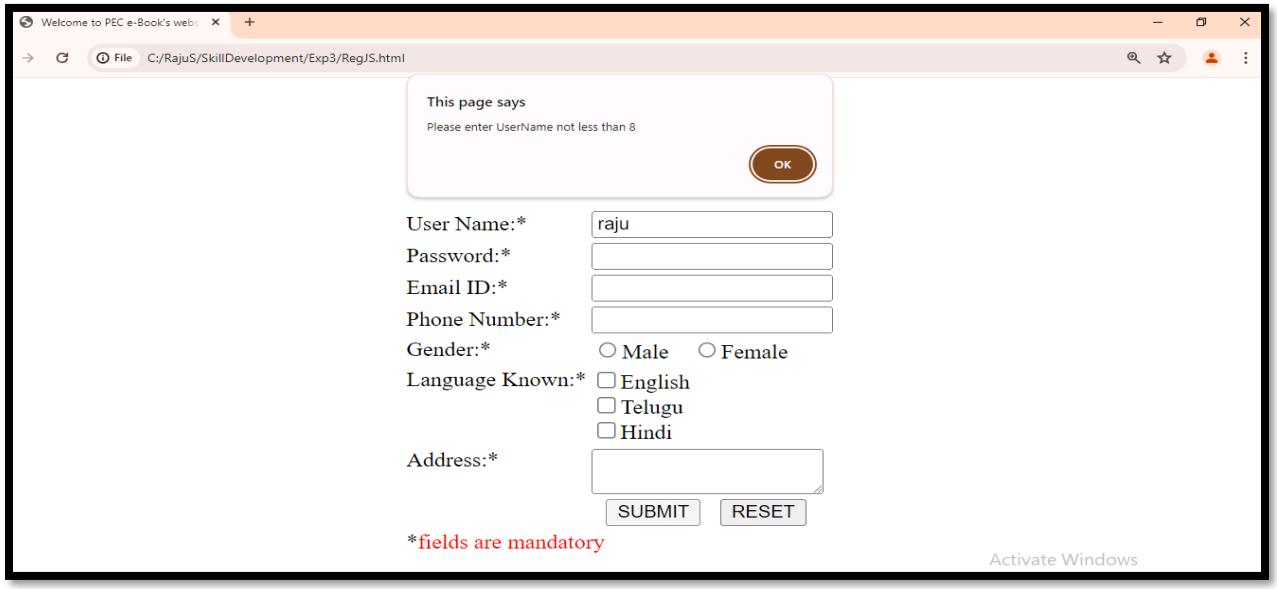
```
    let eflag=false;
    for(i=0;i<email.length;i++) {
        if(email.charAt(i)=="@")
            {
                eflag=true;
            }
    }
        if(!(eflag))
        {
            alert("Please enter a valid Email ID");
            f1.email.focus();
            return false;
        }
}
// phone number validation
var phno = f1.phno.value;
if (phno.length<=0)
{
    alert("Please Enter Phone Number");
    f1.phno.focus();
    return false;
}
if (isNaN(phno))
{
    alert("Please Enter Valid Phone Number");
    f1.phno.focus();
    return false;
}
if (phno.length != 10)
{
    alert("Please Enter Valid Phone Number");
    f1.phno.focus();
    return false;
}
// gender validation
let flag=false;
for(i=0;i<f1.gen.length;i++)
    if(f1.gen[i].checked)
        flag=true;
if(!(flag))
{
alert("Please choose a Gender");
return false;
}
// Language validation
flag=false;
```

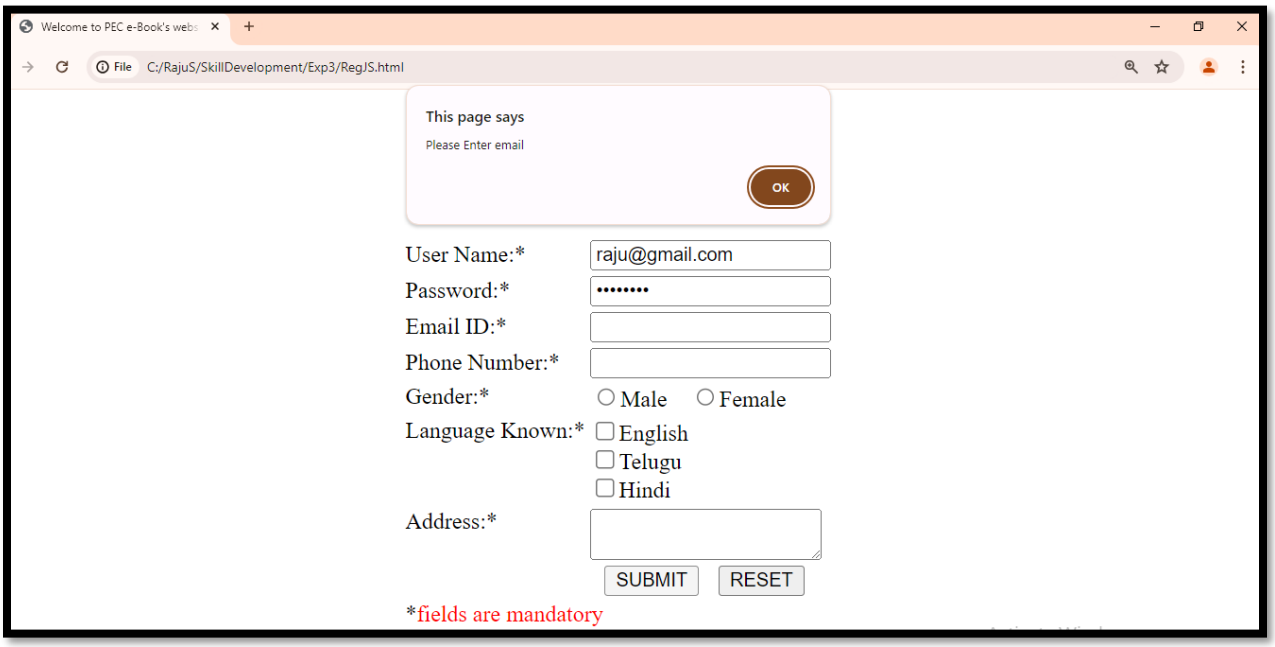
```
        for(i=0;i<f1.lang.length;i++)
           if(f1.lang[i].checked)
              flag=true;
        if(!(flag))
        {
        alert("Please select at least one of the Language options.");
        return false;
        }
        // address validation
        var addr = f1.address.value;
        if (addr.length<=0)
        {
           alert("Please Enter address");
           f1.address.focus();
           return false;
        }
        // to display Success message
        alert("Registration Successful");
     }
     </script>
</head>
<body>
<center><br>
     <h3>Registration Form </h3>
     <br/>
     <form name="f1">
     <table cellpadding="1" align="center" >
     <tr><td> User Name:*</td>
     <td><input type="text" name="username"></td></tr>
     <tr><td>Password:*</td>
     <td><input type="password" name="password"></td></tr>
     <tr><td>Email ID:*</td>
     <td><input type="text" name="email"></td></tr>
     <tr><td>Phone Number:*</td>
     <td><input type="text" name="phno"></td></tr>
     <tr><td valign="top">Gender:*</td>
     <td><input type="radio" name="gen" value="Male">Male   
     <input type="radio" name="gen" value="Female">Female</td></tr>
     <tr> <td valign="top">Language Known:*</td>
     <td> <input type="checkbox" name="lang" value="English">English<br/>
     <input type="checkbox" name="lang" value="Telugu">Telugu<br>
     <input type="checkbox" name="lang" value="Hindi">Hindi<br>
     </td></tr>
     <tr> <td valign="top">Address:*</td>
     <td><textarea name="address"></textarea></td>
     <tr><td></td><td><input type="button" value="SUBMIT" hspace="10" onclick="validate()">
```

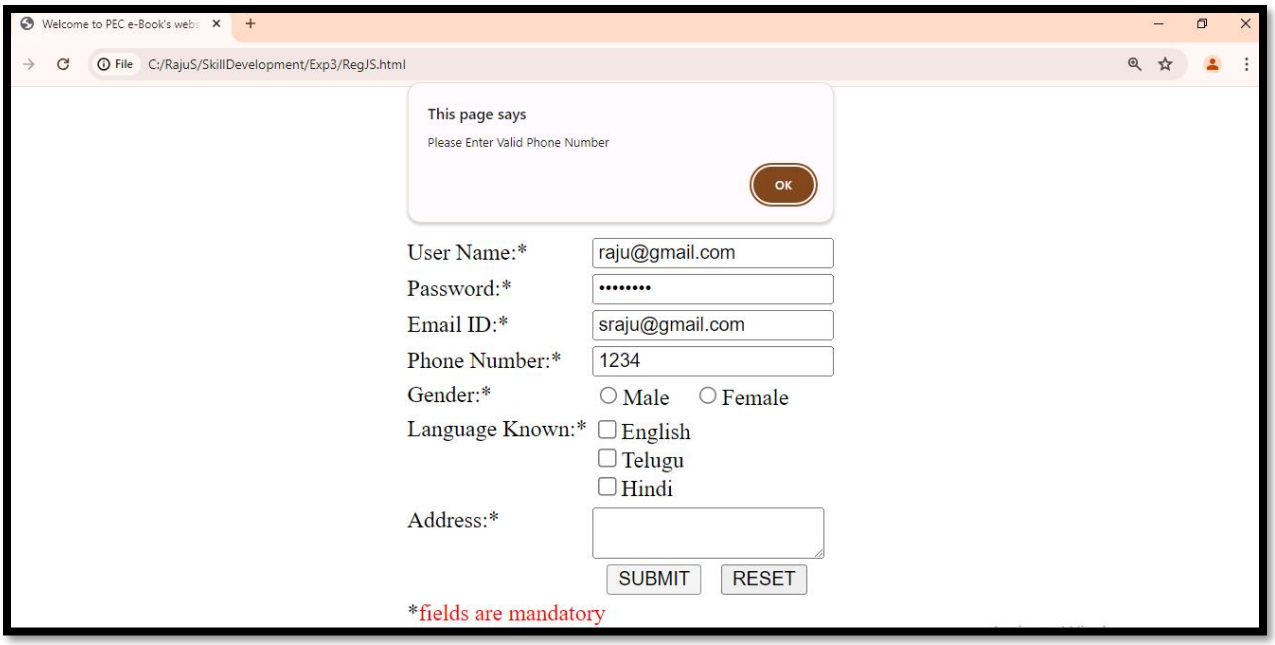
```
   <input type="reset" value="RESET"></td></tr>
   <tr> <td colspan=2 >*<font color="#FF0000">fields are mandatory</font>
   </td>
   </tr>
   </table>
   </form>
</center>
</body>
</html>
```

Output:






Welcome to PEC e-Book's webs
File C:/RajuS/SkillDevelopment/Exp3/RegJS.html
This page says
Please Enter email
OK
User Name:*
raju@gmail.com
Password:*
Email ID:*
Phone Number:*
Gender:*
Male
Female
Language Known:*
English
Telugu
Hindi
Address:*
SUBMIT
RESET
*fields are mandatory


Welcome to PEC e-Book's webs
File C:/RajuS/SkillDevelopment/Exp3/RegJS.html
This page says
Please Enter Valid Phone Number
OK
User Name:*
raju@gmail.com
Password:*
Email ID:*
sraju@gmail.com
Phone Number:*
1234
Gender:*
Male
Female
Language Known:*
English
Telugu
Hindi
Address:*
SUBMIT
RESET
*fields are mandatory

4. Explore the features of ES6 like arrow functions, callbacks, promises, async/await. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page.

Solution:
First, install the required npm packages:

- npm install express axios
- npm install express –save

Create a file named server.js for the backend:

server.js

```
// server.js
constexpress = require('express');
constaxios = require('axios');
constapp = express();
constport = 3000;
app.use(express.static('public'));
app.get('/weather/:city', async (req, res) => {
const{ city } = req.params;
  try {
    constapiKey = 'c97c0c1086d42990e89d64d76f50bb61';
    constresponse =
awaitaxios.get(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}`
);
    consttemperature = response.data.main.temp;
    res.json({ temperature });
  } catch (error) {
    console.error('Error fetching weather data:', error);
    res.status(500).json({ error:'Internal Server Error' });
  }
});
app.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

Create a folder named public and create an index.html file for the frontend:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <link rel="stylesheet" href="./style.css">
   <title>Home - FBS</title>
</head>
<body>
   <div class="wrapper">
      <div class="container">
         <header>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tr>
            <th width="20%"><img src="fbs.png" alt="FBS LOGO"  width="130" height="100"/></th>
            <th colspan=4>
            <h1 style="color:white;">FBS - WORLD BEST ONLINE EBOOKS WEBSITE</h1>
            </th>
            </tr>
            </table>
         </header>
         <nav>
            <table width="100%" align="center" cellpadding="0" cellspacing="2">
            <tbody align="center"  style="font-weight:bold;font-size:18px;"">
            <tr>
            <td width="20%"><hr><a href="index.html">Home</a><hr></td>
            <td width="20%"><hr><a href="login.html">Login</a><hr></td>
            <td width="20%"><hr><a href="registration.html">Registration</a><hr></td>
            <td width="20%"><hr><a href="cart.html" >Cart</a><hr></td>
            </tr>
            </tbody>
            </table>
         </nav>
      </div>
      <div class="container1">
         <div class="sidebar1"></div>
         <div class="container2">
            <main>
               <center>
               <h2>Welcome to FBS e-Book's Website</h2>
                  <p>Shopping at <font size=5>FBS</font> can be both <font size=5>fun</font>
                  and <font size=5>savings</font>.</br>Shop with us in this special <font
                  size=5>discount</font> season and save upto <font size=5>90%</font> on all your
                  purchases.</br></p>
               <br/><br/><br/><br/><br/><br/><br/><br/>
            </main>
         </div>
         <div class="sidebar2"></div>
```

```
    </div>
    <footer><font color="white">(C) 2024 All rights reserved by FBS ebooks</font></footer>
  </div>
</body>
</html>
```

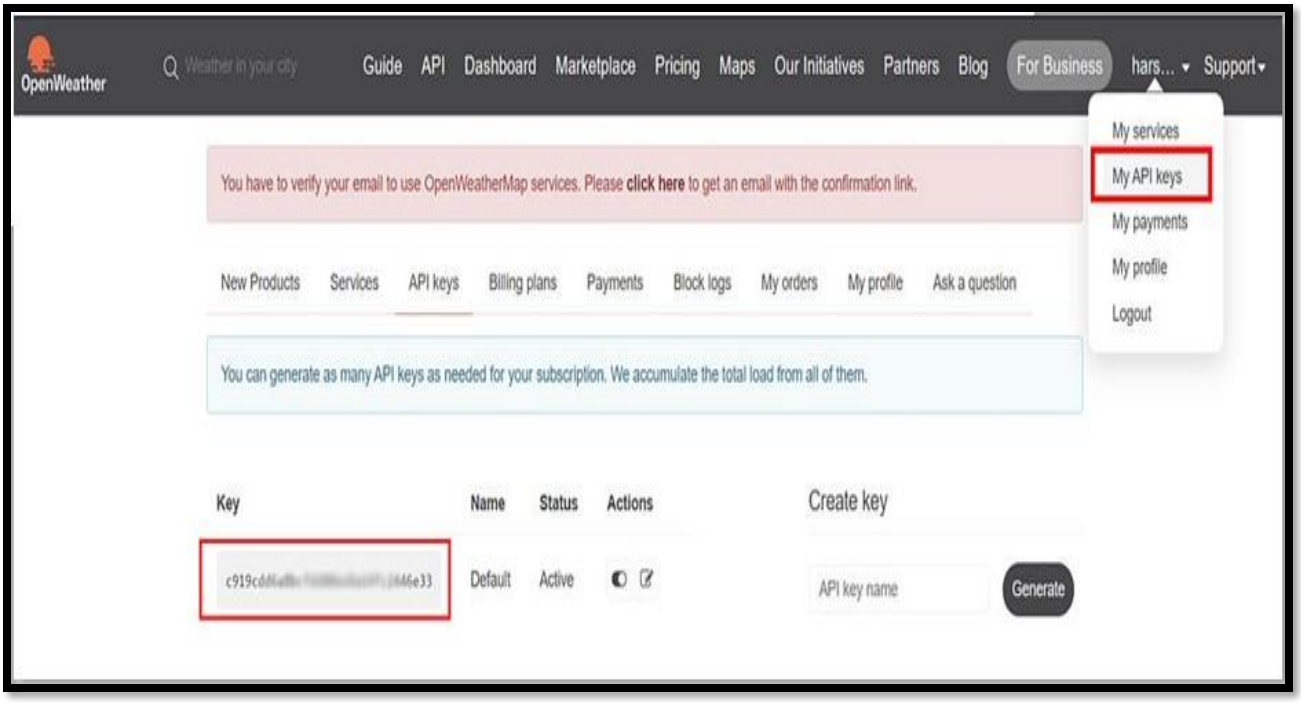
Now, you can run your Node.js server:

node server.js

Visit http://localhost:3000 in your web browser.
Create an OpenWeatherMap Account and Generate API Key
Visit the OpenWeatherMap website **(https://openweathermap.org/)** and click on **"Sign Up"** or **"Log In"** to create an account or log into your existing account.

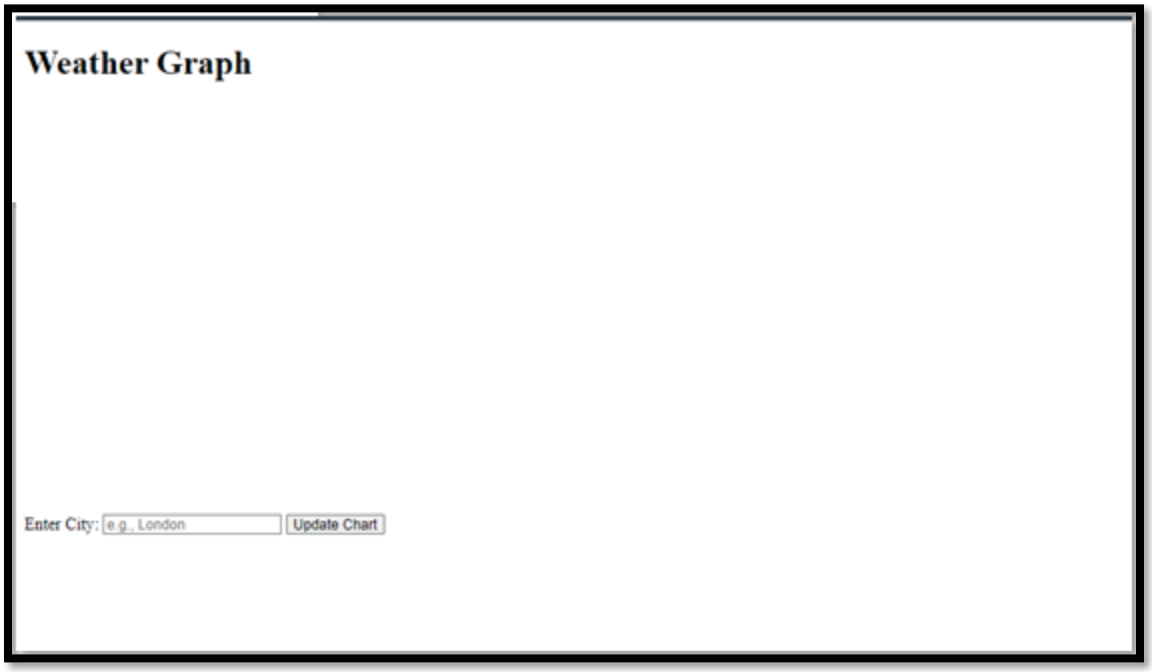
Once logged in, navigate to your account dashboard.
From the dashboard, locate my API Keys section and click on **"Create Key"** or **"API Keys"** to generate a new API key.
Provide a name for your API key (e.g., "**WeatherApp**") and click on the **"Generate"** or **"Create"** button.
Your API key will be generated and displayed on the screen. Make sure to copy it as we will need it later.

Output:

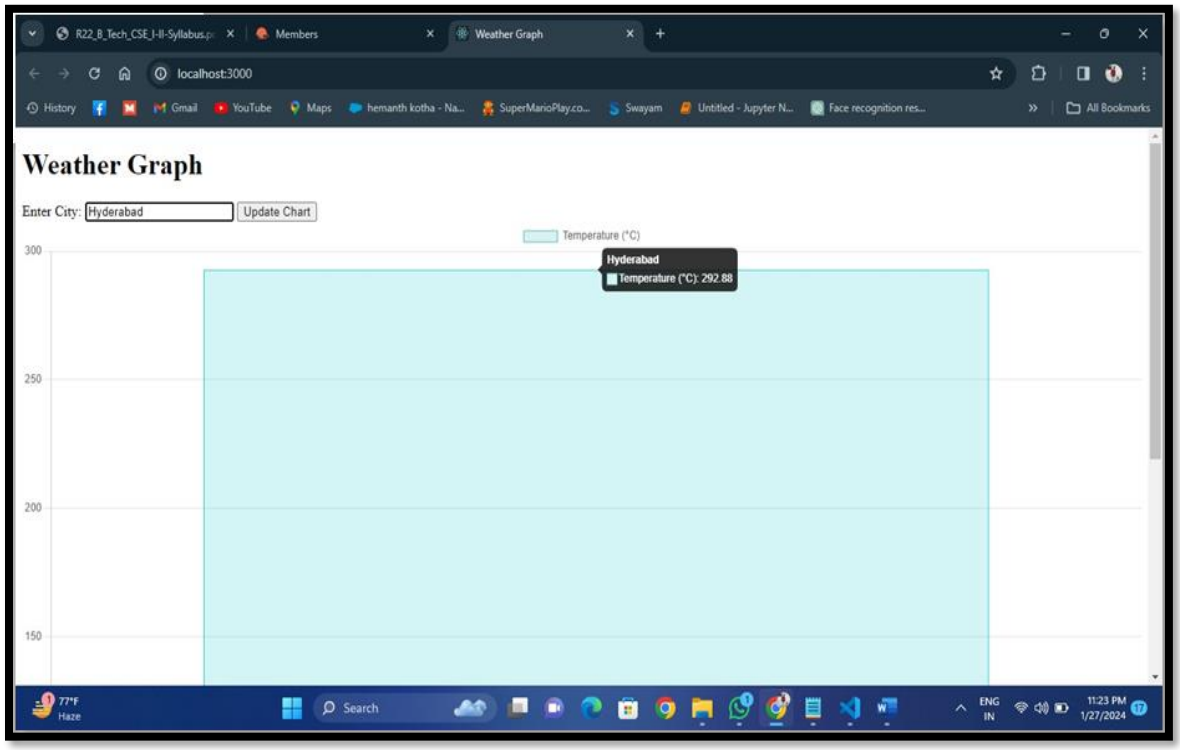


Locate API key

Initially it look like:



Then, by entering the city then click update chart then



5. Develop a java standalone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables.

I will provide you with the MySQL code for creating the student table and make some changes to your Java code to improve it:

MySQL Code:

sql> CREATE TABLE IF NOT EXISTS student (s_id INT PRIMARY KEY, s_nameVARCHAR(255), s_addressVARCHAR(255));

This SQL code creates a table with three columns: s_id for student ID (primary key), s_name for student name, and s_address for student address.

Java Code:

InsertData.java

```
import java.sql.*;
import java.util.Scanner;
public class InsertData
{
   public static void main(String[] args)
          {
          try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");
          Statements = con.createStatement())
          {
                    Scanner sc = new Scanner(System.in);
                    System.out.println("Inserting Data into student table:");
                    System.out.println("_____");
                    System.out.print("Enter student id: ");
                    int sid = sc.nextInt();
                    System.out.print("Enter student name: ");
                    String sname = sc.next();
                    System.out.print("Enter student address: ");
                    String saddr = sc.next();
                    String insertQuery = "INSERT INTO student VALUES(" + sid + ",'" + sname + "','" +
saddr + "')";
                    s.executeUpdate(insertQuery);
                    System.out.println("Data inserted successfully into student table");

      }
          catch (SQLException err)
          {
         System.out.println("ERROR: " + err);
      }
   }
}
```

Output:

Inserting Data into student table:

_____

Enter student id: 101
Enter student name: John Doe
Enter student address: 123 Main Street

Data inserted successfully into student table

UpdateData.java

```
import java.sql.*;
import java.util.Scanner;

public class UpdateData {
   public static void main(String[] args) {
      try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");
          Statements = con.createStatement()) {

         Scanner sc = new Scanner(System.in);
         System.out.println("Update Data in student table:");
         System.out.println("_____");
         System.out.print("Enter student id: ");
         int sid = sc.nextInt();
         System.out.print("Enter student name: ");
         String sname = sc.next();
         System.out.print("Enter student address: ");
         String saddr = sc.next();

         String updateQuery = "UPDATE student SET s_name='" + sname + "', s_address='" + saddr + "'
WHERE s_id=" + sid;
         s.executeUpdate(updateQuery);

         System.out.println("Data updated successfully");

      } catch (SQLException err) {
         System.out.println("ERROR: " + err);
      }
   }
}
```

Output :

Update Data in student table:

_____

Enter student id: 101
Enter student name: Jane Doe
Enter student address: 456 Broad Street
Data updated successfully

DeleteData.java

```
import java.sql.*;
import java.util.Scanner;
public class DeleteData
{
   public static void main(String[] args)
{
      try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");
          Statements = con.createStatement())
          {
          Scanner sc = new Scanner(System.in);
          System.out.println("Delete Data from student table:");
          System.out.println("_____");
          System.out.print("Enter student id: ");
          int sid = sc.nextInt();
          String deleteQuery = "DELETE FROM student WHERE s_id=" + sid;
          s.executeUpdate(deleteQuery);
          System.out.println("Data deleted successfully");
      }
          catch (SQLException err)
          {
                    System.out.println("ERROR: " + err);
          }
   }
}
```

Output:

```
Delete Data from student table:
_____
Enter student id: 101
Data deleted successfully
```

DisplayData.java

```
import java.sql.*;
public class DisplayData
{
   public static void main(String[] args)
{
      try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root", "");
          Statement s = con.createStatement())
          {
           ResultSet rs = s.executeQuery("SELECT * FROM student");
            if (rs != null) {
             System.out.println("SID \t STU_NAME \t ADDRESS");
             System.out.println("_____");
             while (rs.next())
```

```
            {
                System.out.println(rs.getString("s_id") + " \t " + rs.getString("s_name") + " \t " +
rs.getString("s_address"));
                System.out.println("_____");
              }
            }

        }
            catch (SQLException err)
            {
                    System.out.println("ERROR: " + err);
             }
    }
}
```

Output:

SID STU_NAME ADDRESS

_____

102 Alice Smith 789 Oak Avenue

_____

103 Bob Johnson 567 Pine Road

_____

6. Create an xml for the bookstore. Validate the same using both DTD and XSD.

Solution:

XML data to validate:

bookstore.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore[
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, author, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]>
<bookstore>
<book>
<title>Introduction to XML</title>
<author>John Doe</author>
<price>29.99</price>
</book>
<book>
<title>Programming with XML</title>
<author>Jane Smith</author>
<price>39.99</price>
</book>
</bookstore>
```

XML schema (XSD) data:

bookstore.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://example.com"
        xmlns="http://example.com">
<xs:element name="root">
<xs:complexType>
<xs:sequence>
<xs:element name="bookstore" type="bookstoreType"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="bookstoreType">
<xs:sequence>
<xs:element name="book" type="bookType" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
```

```
</xs:complexType>
<xs:complexType name="bookType">
<xs:sequence>
<xs:element name="title" type="xs:string"/>
<xs:element name="author" type="xs:string"/>
<xs:element name="price" type="xs:decimal"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

To Check the Validity:

Go to the below link,
https://www.liquid-technologies.com/online-xsd-validator
Place the **XML code** in the XML Validate.
Place the **XSD code** in the XML Schema Data.
Then click the **validate** Button.
Then it will show the Document as Valid.

Output:

7. Create a custom server using http module and explore the other modules of Node JS like OS, path, event.

Solution:

☛ Open Terminal or Command Prompt:

Open a terminal or command prompt in the directory where you saved your **server.js** file.

☛ Run the Server Script:

Execute the server script using the Node.js runtime. In the terminal, run:
node server.js

This will start the HTTP server, and you should see the message "*Server running at http://127.0.0.1:3000/*".

☛ Access the Server:
Open your web browser and navigate to *http://127.0.0.1:3000/* or *http://localhost:3000/*. You should see the response "**Hello, World!**".

☛ Check OS Information:
In the same terminal where your server is running, you'll see information about your operating system (OS) type, platform, architecture, CPU cores, etc.

☛ Check Current Working Directory:
The current working directory of the script is printed in the terminal.

☛ Check Joined Path:
The joined path using the path module is printed in the terminal.

☛ Check Custom Event:
The script emits a custom event and listens for it. In the terminal, you should see the message "Custom Event Triggered: { message: 'Hello from custom event!' }".

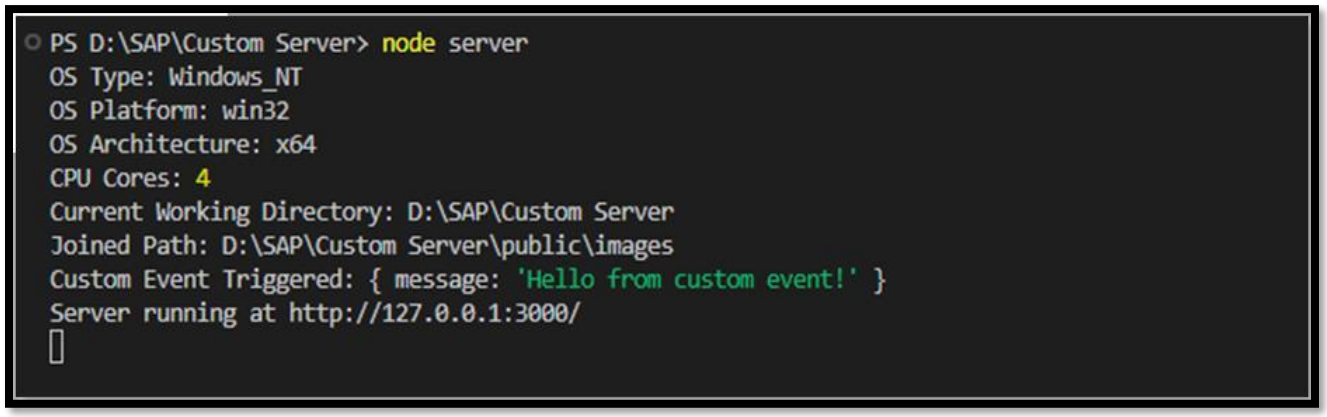
☛ Stop the Server:
To stop the server, press Ctrl+C in the terminal where the server is running.
server.js

```
// Step 1: Import required modules
const http = require('http');
const os = require('os');
const path = require('path');
const { EventEmitter } = require('events');

// Step 2: Create an instance of EventEmitter
const eventEmitter = new EventEmitter();

// Step 3: Create a simple HTTP server
const server = http.createServer((req, res) => {
```

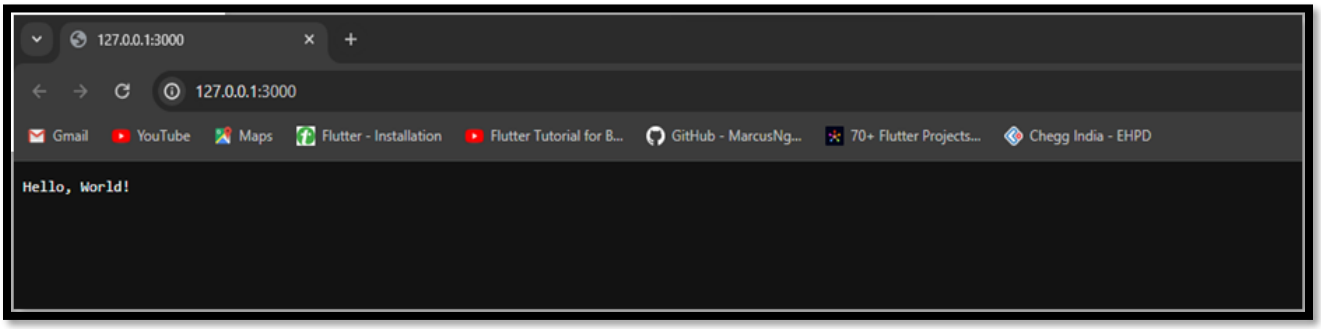
```
  res.writeHead(200, { 'Content-Type':'text/plain' });
  res.end('Hello, World!\n');
});

// Step 4: Define server port and hostname
constport = 3000;
consthostname = '127.0.0.1';

// Step 5: Listen for requests on the specified port and hostname
server.listen(port, hostname, () => {
  console.log('Server running at http://${hostname}:${port}/');
});

// Step 6: Print OS information
console.log('OS Type:', os.type());
console.log('OS Platform:', os.platform());
console.log('OS Architecture:', os.arch());
console.log('CPU Cores:', os.cpus().length);

// Step 7: Print current working directory
console.log('Current Working Directory:', process.cwd());

// Step 8: Join paths using the path module
const joinedPath = path.join(__dirname, 'public', 'images');
console.log('Joined Path:', joinedPath);

// Step 9: Handle a custom event
eventEmitter.on('customEvent', (data) => {
  console.log('Custom Event Triggered:', data);
});

// Step 10: Emit a custom event
eventEmitter.emit('customEvent', { message:'Hello from custom event!' });
```

Output:

In the Terminal:



In the Browser:
Link: http://127.0.0.1:3000/



8. Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)

Solution:

Firstly we need to create a new folder and open the folder in the command prompt and enter a command as below:

npminit -y

Open that folder in the vscode by entering *code*.

Next in the terminal we need to install all the packages we need, so we mainly use express and sqlite3. The Command to install express and sqlite3 is

npm install express sqlite3

Then create file named as the **app.js** and **db.js**

db.js

```
const sqlite3 = require('sqlite3').verbose();

// Function to initialize the database schema
function initializeDatabase() {
    const db = new sqlite3.Database('./mydatabase.db', (err) => {
        if (err) {
            console.error(err.message);
        } else {
            console.log('Connected to the SQLite database.');
            createStudentsTable(db);
        }
    });

    // Close the database connection when the Node process exits
    process.on('exit', () => {
        db.close((err) => {
            if (err) {
                console.error(err.message);
            } else {
                console.log('Disconnected from the SQLite database.');
            }
        });
    });
}
// Function to create the 'students' table if it doesn't exist
function createStudentsTable(db) {
    const createTableQuery = `
        CREATE TABLE IF NOT EXISTS students (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
            name TEXT,
            age INTEGER,
            grade TEXT
        );
    `;
    db.run(createTableQuery, (err) => {
        if (err) {
            console.error(err.message);
        } else {
            console.log('The students table has been created or already exists.');
        }
    });
}

module.exports = { initializeDatabase };
```

when we execute both the **db.js** then the database will be created that is **mydatabase.db**

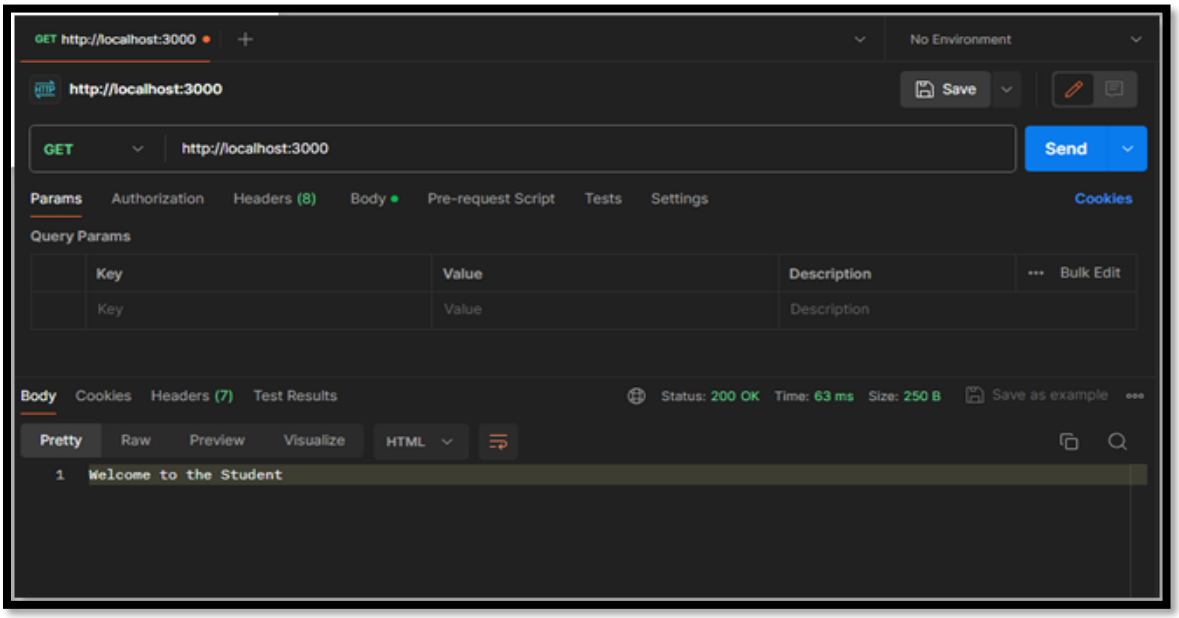
app.js

```
const express = require('express');
const sqlite3 = require('sqlite3');
const{ initializeDatabase } = require('./db');
const app = express();
const port = 3000;

// Connect to SQLite database
const db = new sqlite3.Database('./mydatabase.db', (err) => {
    if (err) {
        console.log(err.message);
    } else {
        console.log('Connected to the SQLite database.');
    }
});

// Middleware to parse request body as JSON
app.use(express.json());

app.get('/', (req, res) => {
    res.send('Welcome to the Student');
});

// Get all Students
app.get('/students', (req, res) => {
    db.all('SELECT * FROM students', [], (err, rows) => {
        if (err) {
            return console.error(err.message);
        }
        res.json(rows);
    });
});
```

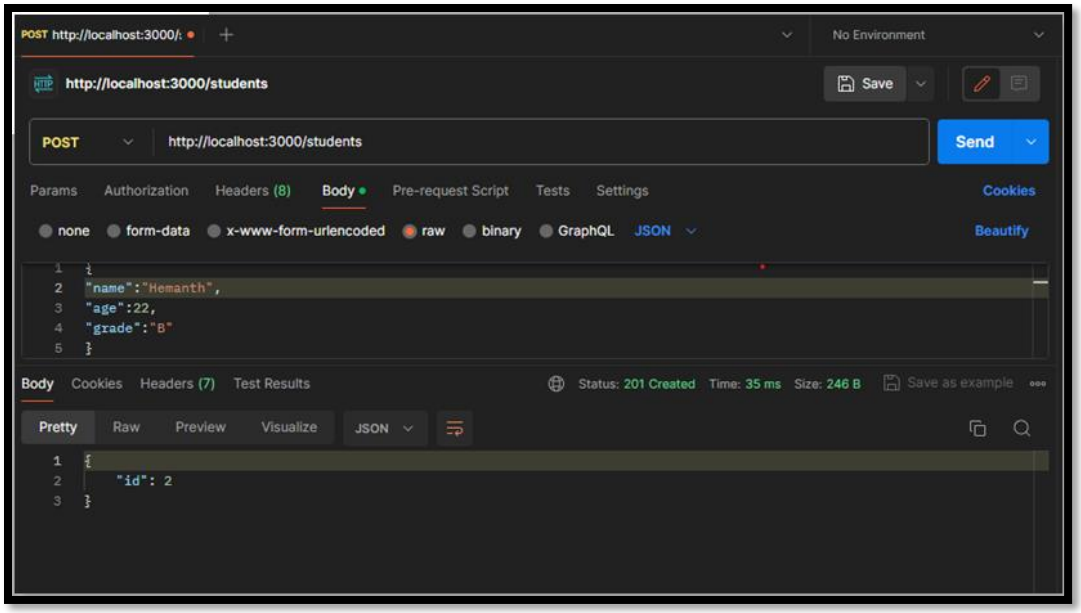
```
// Get a single student by id
app.get('/students/:id', (req, res) => {
    const id = req.params.id;
    db.all('SELECT * FROM students WHERE id = ?', [id], (err, row) => {
        if (err) {
            return console.error(err.message);
        }
        res.json(row);
    });
});

// Create a new student
app.post('/students', (req, res) => {
    const{ name, age, grade } = req.body;
    db.run('INSERT INTO students (name, age, grade) VALUES (?, ?, ?)', [name, age, grade], function
(err) {
        if (err) {
            return console.error(err.message);
        }
        res.status(201).json({ id:this.lastID });
    });
});
// Update a student
app.put('/students/:id', (req, res) => {
    const id = req.params.id;
    const{ name, age, grade } = req.body;
    db.run('UPDATE students SET name = ?, age = ?, grade = ? WHERE id = ?', [name, age, grade, id],
function (err) {
        if (err) {
            return console.error(err.message);
        }
        res.json({ updatedID:id });
    });
});
// Delete a student
app.delete('/students/:id', (req, res) => {
    const id = req.params.id;
    db.run('DELETE FROM students WHERE id = ?', id, function (err) {
        if (err) {
            return console.error(err.message);
        }
        res.json({ deletedID:id });
    });
});

app.listen(port, () => {
    console.log('Server running at http://localhost:${port}');
});
```

Output:

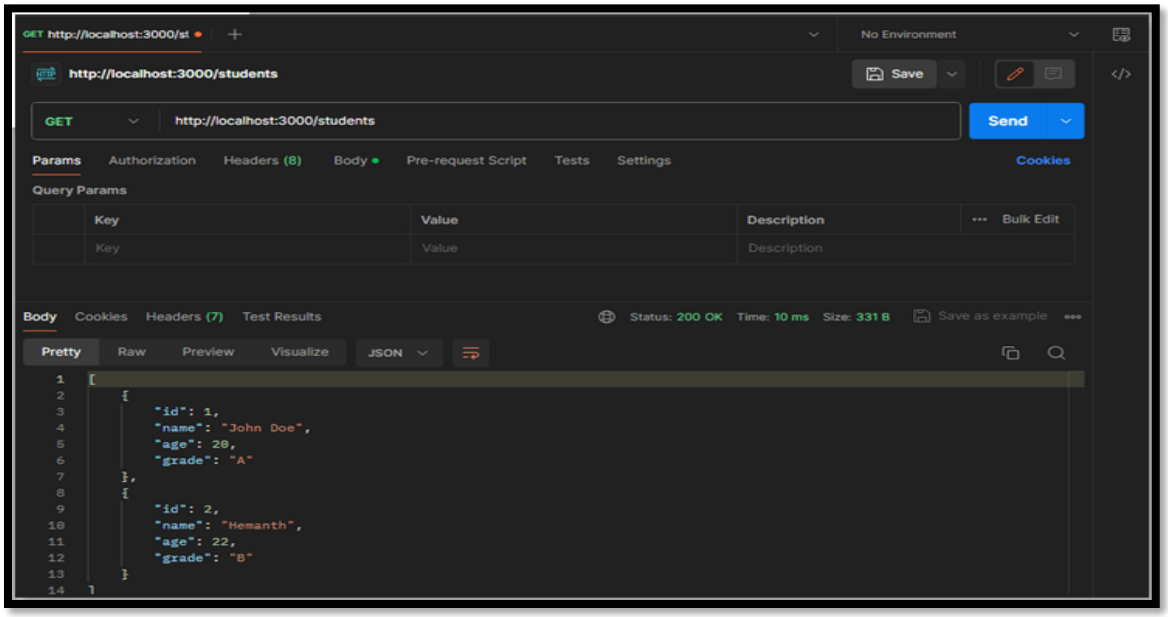
GET:
Open Postman.
Set the request type to GET.
Enter the URL: *http://localhost:3000/students*.



POST:
Create a New Student
Open Postman.
Set the request type to POST.
Enter the URL: *http://localhost:3000/students.*
Go to the "**Body**" tab.
Select raw and set the body to JSON format.



GET: #all Students
Set the request type to GET.
Enter the URL: *http://localhost:3000/students*.
Click on the "**Send**" button
You should receive a response with details of all students in your SQLite database.

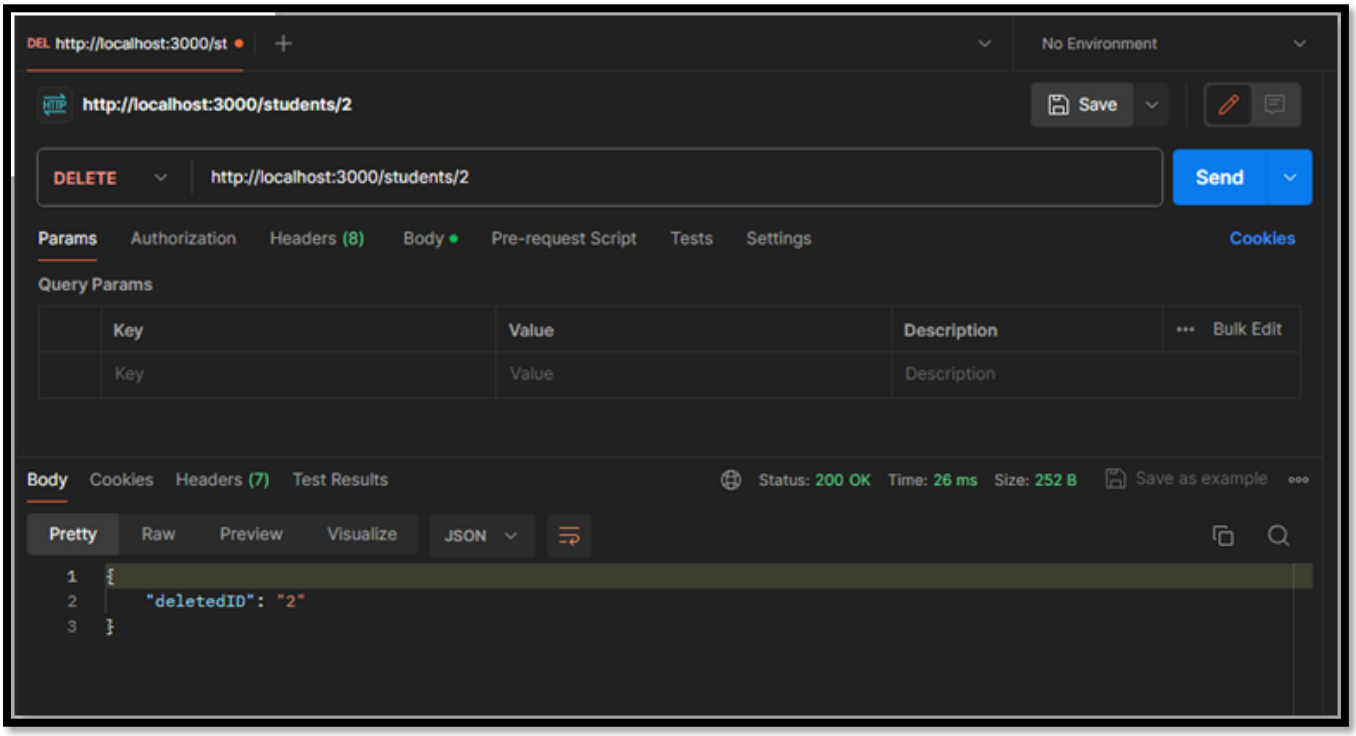


DELETE:
Set the request type to DELETE.

Enter the URL for the student you want to delete (replace: id with an actual student ID): *http://localhost:3000/students/:id*

Place instead of ID which replace with number that is ID to be deleted.

Then click **send**

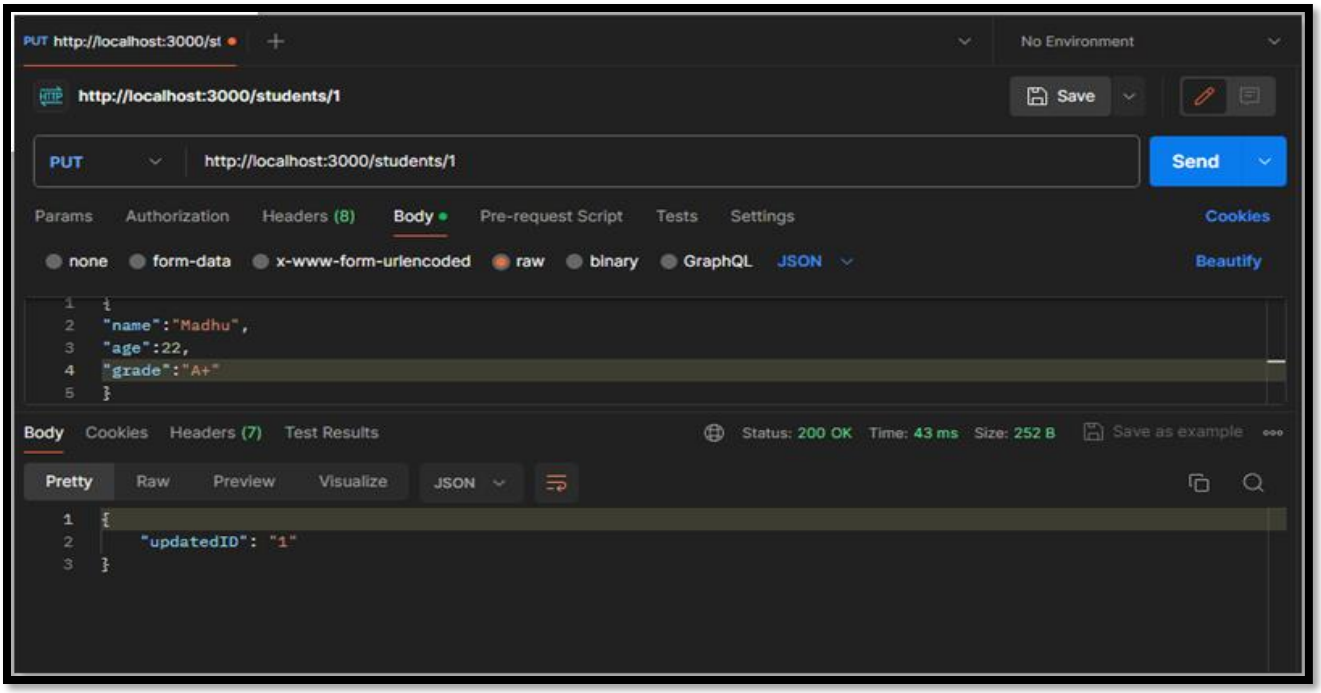


PUT:
Set the request type to PUT.

Enter the URL for the student you want to delete (replace: id with an actual student ID): *http://localhost:3000/students/:id*

Go to the **"Body"** tab.

Select raw and set the body to JSON format


PUT http://localhost:3000/st
No Environment
http://localhost:3000/students/1
Save
PUT
http://localhost:3000/students/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Madhu",
"age":22,
"grade":"A+"
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 43 ms
Size: 252 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"updatedID": "1"
}

9. Create a service in react that fetches the weather information from openweathermap.org and the display the current and historical weather information using graphical representation using chart.js

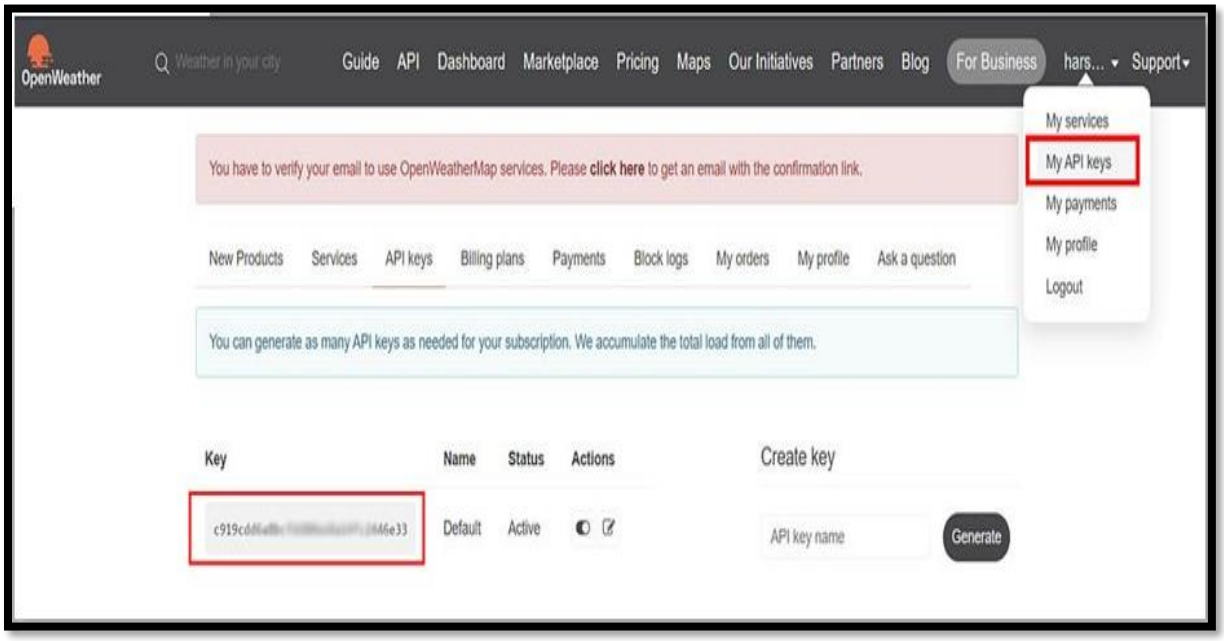
Solution:

Step 1: Create an OpenWeatherMap Account and Generate API Key.

Visit the OpenWeatherMap website *(https://openweathermap.org/)* and click on "**Sign Up**" or "**Log In**" to create an account or log into your existing account.

Once logged in, navigate to your account dashboard.

From the dashboard, locate my API Keys section and click on "**Create Key**" or "**API Keys**" to generate a new API key.
Provide a name for your API key (e.g., "**WeatherApp**") and click on the "**Generate**" or "**Create**" button.
Your API key will be generated and displayed on the screen. Make sure to copy it as we will need it later.
Locate API key



Step 2: Set up a new React project

Open your terminal or command prompt.
Run the following command to create a new React project:

npxcreate-react-app weather-app

Once the project is created, navigate into the project directory:

cd weather-app

Step 3: Install required packages

In the project directory, install the necessary packages by executing the following command

npm install axios

We will use the Axios library to make HTTP requests to the OpenWeatherMap API.

Step 4: Create a Weather component
Inside the "src" directory, create a new file called "Weather.js" and open it in your code editor.
Add the following code to define a functional component named Weather:

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const Weather = () => {
  const [city, setCity] = useState('');
  const [weatherData, setWeatherData] = useState(null);

  const fetchData = async () => {
    try {
      const apiKey = 'c97c0c1086d42990e89d64d76f50bb61';
// Replace with your OpenWeatherMap API key

      const response = await axios.get(
        'https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}'
      );
      setWeatherData(response.data);
      console.log(response.data); //You can see all the weather data in console log
    } catch (error) {
console.error(error);
    }
  };


useEffect(() => {
fetchData();
  }, []);

  const handleInputChange = (e) => {
    setCity(e.target.value);
  };

  const handleSubmit = (e) => {
e.preventDefault();
fetchData();
  };

  return (
<div>
<form onSubmit={handleSubmit}>
<input
          type="text"
```

```
        placeholder="Enter city name"
        value={city}
        onChange={handleInputChange}
      />
<button type="submit">Get Weather</button>
</form>
    {weatherData ? (
<>
<h2>{weatherData.name}</h2>
<p>Temperature: {weatherData.main.temp} C</p>
<p>Description: {weatherData.weather[0].description}</p>
<p>Feels like : {weatherData.main.feels_like} C</p>
<p>Humidity : {weatherData.main.humidity}%</p>
<p>Pressure : {weatherData.main.pressure}</p>
<p>Wind Speed : {weatherData.wind.speed}m/s</p>
</>
    ) : (
<p>Loading weather data...</p>
    )}
</div>
  );
};

export default Weather;
```

Replace {YOUR_API_KEY} in the API URL with the API key you generated from OpenWeatherMap.

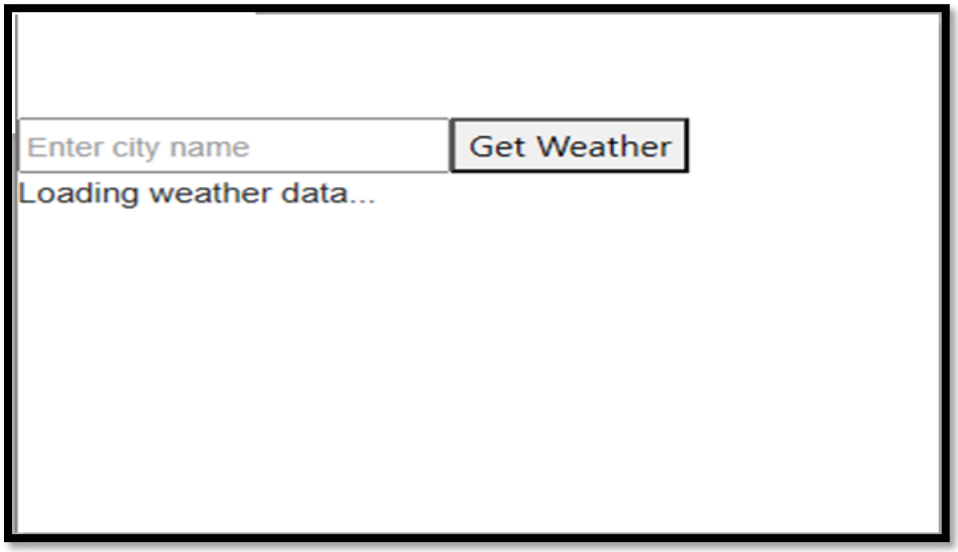
Step 5: Connect the Weather component to your app.

Open the " App.js" file in the "src" directory.
Replace the existing code with the following code:

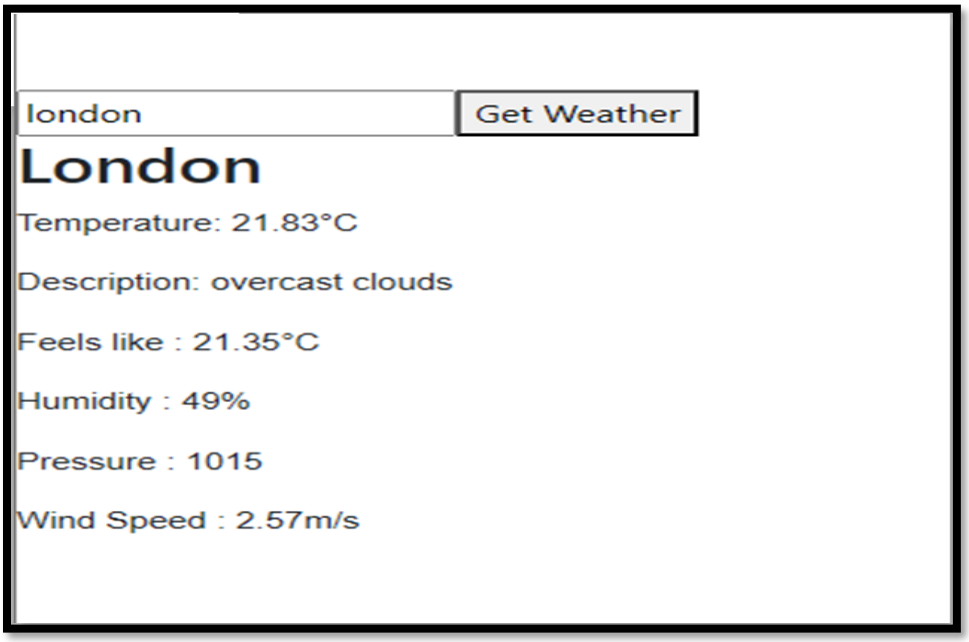
```
import React from'react';
import Weather from'./Weather';

const App = () => {
return (
<div>
<h1>Weather Forecast App</h1>
<Weather />
</div>
  );
};

exportdefault App;
```

Output:

Initial Screen



After Supply the City name



10. Create a TODO application in react with necessary components and deploy it into github.

Solution:

Step 1: Set Up the Project

Our first task is to set up the React project. This step involves creating the necessary project structure. Here's how you can do it:

1. Create a React App:
Open your terminal and navigate to your preferred directory. Run the following command to generate a new React app. Replace **"todo-app"** with your desired project name:

```
npx create-react-app todo-app
```

This command will create a directory named "todo-app" with all the initial code required for a React app.
2. Navigate to the Project Directory:
Change your working directory to the "todo-app" folder:

```
cd todo-app
```

3. Start the Development Server:
Launch the development server with the following command:

```
npm start
```

This will open your React app app, and you'll see the default React starter page in your web browser at *'http://localhost:3000'*.
Step 2: Create the App Component
In this step, we create the App component, which serves as the entry point to our Todo List application.

```
import React from'react';
import TodoList from'./components/TodoList';
function App() {
return (
<div className="App">
<TodoList />
</div>
 );
}
exportdefault App;
```

Step 3: Create the TodoList

src->Component
Now, let's create the 'TodoList' component, which is responsible for managing the list of tasks and handling task-related functionality.

```
import React, { useState } from 'react';
```

```
import TodoItem from './TodoItem';
function TodoList() {
const [tasks, setTasks] = useState([
 {
id: 1,
text: 'Doctor Appointment',
completed: true
 },
 {
id: 2,
text: 'Meeting at School',
completed: false
 }
 ]);

const [text, setText] = useState('');
function addTask(text) {
const newTask = {
id: Date.now(),
 text,
completed: false
 };
 setTasks([tasks, newTask]);
 setText('');
 }
function deleteTask(id) {
 setTasks(tasks.filter(task => task.id !== id));
 }
function toggleCompleted(id) {
 setTasks(tasks.map(task => {
if (task.id === id) {
return {task, completed: !task.completed};
 } else {
return task;
 }
 }));
 }
return (
<div className="todo-list">
 {tasks.map(task => (
<TodoItem
 key={task.id}
 task={task}
 deleteTask={deleteTask}
 toggleCompleted={toggleCompleted}
 />
 ))}
<input
 value={text}
 onChange={e => setText(e.target.value)}
 />
```

```
<button onClick={() => addTask(text)}>Add</button>
</div>
 );
}
exportdefault TodoList;
```

Step 4: Create the place the TodoItem in

src->Component
In this step, we create the 'TodoItem' component, which represents an individual task in our Todo List.

```
import React from'react';
function TodoItem({ task, deleteTask, toggleCompleted }) {
function handleChange() {
 toggleCompleted(task.id);
 }
return (
<div className="todo-item">
<input
 type="checkbox"
 checked={task.completed}
 onChange={handleChange}
 />
<p>{task.text}</p>
<button onClick={() => deleteTask(task.id)}>
 X
</button>
</div>
 );
}
exportdefault TodoItem;
```

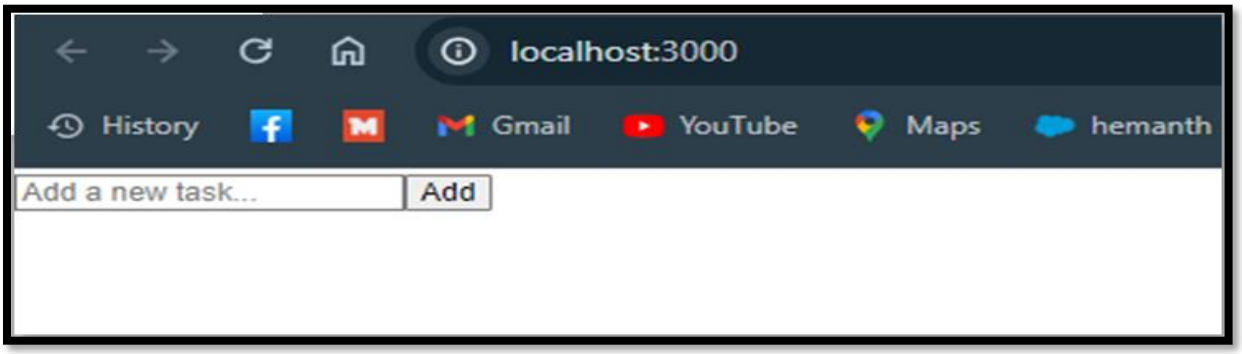
These three components, ' App', 'TodoList', and 'TodoItem', work together to create a functional Todo List application in React. The 'TodoList' component manages the state of the tasks, and the 'TodoItem' component represents and handles individual tasks within the list.

Step 5: Styling
To enhance the visual appeal of your Todo List, you can apply some basic styling. Here is an example of how you can style the todo items. In the `App.css` file, add the following styles:

```
.todo-item
{
        display: flex;
        justify-content: space-between;
        margin-bottom: 8px;
}
.todo-itemp
{
        color: #888;
        text-decoration: line-through;
}
```

Output:

Initially it looks like:



Next,