



Estd.2001

Sri Indu

College of Engineering & Technology

UGC Autonomous Institution

Recognized under 2(f) & 12(B) of UGC Act 1956,

NAAC, Approved by AICTE &

Permanently Affiliated to JNTUH



NAAC

NATIONAL ASSESSMENT AND
ACCREDITATION COUNCIL



SKILL DEVELOPMENT COURSE LAB MANUAL
(Node.js/React/Django)

LAB MANUAL

II Year II Semester

DEPARTMENT OF INFORMATION TECHNOLOGY



SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

B. TECH –INFORMATION TECHNOLOGY

INSTITUTION VISION

To be a premier Institution in Engineering & Technology and Management with competency, values and social consciousness.

INSTITUTION MISSION

- IM₁** Provide high quality academic programs, training activities and research facilities.
- IM₂** Promote Continuous Industry-Institute Interaction for Employability, Entrepreneurship, Leadership and Research aptitude among stakeholders.
- IM₃** Contribute to the Economical and technological development of the region, state and nation.

DEPARTMENT VISION

To be a recognized knowledge center in the field of Information Technology with self - motivated, employable engineers to society.

DEPARTMENT MISSION

The Department has following Missions:

- DM₁** To offer high quality student centric education in Information Technology.
- DM₂** To provide a conducive environment towards innovation and skills.
- DM₃** To involve in activities that provide social and professional solutions.
- DM₄** To impart training on emerging technologies namely cloud computing and IOT with involvement of stake holders.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

- PEO1: Higher Studies:** Graduates with an ability to apply knowledge of Basic sciences and programming skills in their career and higher education.
- PEO2: Lifelong Learning:** Graduates with an ability to adopt new technologies for ever changing IT industry needs through Self-Study, Critical thinking and Problem solving skills.
- PEO3: Professional skills:** Graduates will be ready to work in projects related to complex problems involving multi-disciplinary projects with effective analytical skills.
- PEO4: Engineering Citizenship:** Graduates with an ability to communicate well and exhibit social, technical and ethical responsibility in process or product.

PROGRAM OUTCOMES (POs) & PROGRAM SPECIFIC OUTCOMES (PSOs)

PO	Description
PO 1	Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO 2	Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO 3	Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO 4	Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO 5	Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
PO 6	The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO 7	Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
PO 8	Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO 10	Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO 11	Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)
Program Specific Outcomes	
PSO 1	Software Development: To apply the knowledge of Software Engineering, Data Communication, Web Technology and Operating Systems for building IOT and Cloud Computing applications.
PSO 2	Industrial Skills Ability: Design, develop and test software systems for world-wide network of computers to provide solutions to real world problems.
PSO 3	Project implementation: Analyze and recommend the appropriate IT Infrastructure required for the implementation of a project.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
 - a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
 - b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
 - c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department

Principal

R22 – B.TECH. – INFORMATION TECHNOLOGY

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution under UGC, New Delhi)

B.Tech. - II Year – II Semester

L T P C

0 0 2 1

(R22) SKILL DEVELOPMENT COURSE (NODE JS/ REACT JS/ DJANGO)

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 stand alone 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 sessiontracking 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.

List of Experiments

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 stand alone 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 sessiontracking 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 CRUDoperations 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 usingchart.js.
14. Create a TODO application in react with necessary components and deploy it into github.

SKILL DEVELOPMENT PROGRAMS

EXERCISE 1:

Aim:

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

Solution :

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%"></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 <fontsize=5>fun</font>
and <font size=5>savings</font>.</br>Shop withus in this special
<font
size=5>discount</font> season and save upto<font size=5>90%</font> onall 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 FBSebooks</font></footer>
</div>
</body>
</html>

```

login.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet"href="./style.css">
<title>Login - FBS</title>
</head>
<body>
<div class="wrapper">
<div class="container">
<header>
<table width="100%" align="center" cellpadding="0" cellspacing="2">
<tr>
<thwidth="20%"><imgsrc="fbs.png"alt="FBS LOGO"width="130"height="100"/></th>
<thcolspan=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">
<tbodyalign="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></tr>
<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 FBSebooks</font></footer>
</div>
</body>
</html>

```

registration.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet"href="/style.css">
<title>Registration - FBS</title>
</head>
<body>
<div class="wrapper">
<div class="container">
<header>
<table width="100%"align="center"cellpadding="0"cellspacing="2">
<tr>
<thwidth="20%"><imgsrc="fbs.png"alt="FBS LOGO"width="130"height="100"/></th>
<thcolspan=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">
<tbodyalign="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">
```


cart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet"href="/style.css">
<title>Cart - FBS</title>
</head>
<body>
<div class="wrapper">
<div class="container">
<header>
<table width="100%"align="center"cellpadding="0"cellspacing="2">
<tr>
<thwidth="20%"><imgsrc="fbs.png"alt="FBS
LOGO"width="130"height="100"/></th>
<thcolspan=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">
<tbodyalign="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="registraton.html">Registraton</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>
<thwidth="40%"><hr>BookName<hr></th>
<thwidth="20%"><hr>Price<hr></th>
```

```

<thwidth="20%"><hr>Quantity<hr></th>
<thwidth="20%"><hr>Amount<hr></th></tr>
</tbody>
<tbodyalign=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 FBSebooks</font></footer>
</div>
</body>
</html>

```

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;
}
```

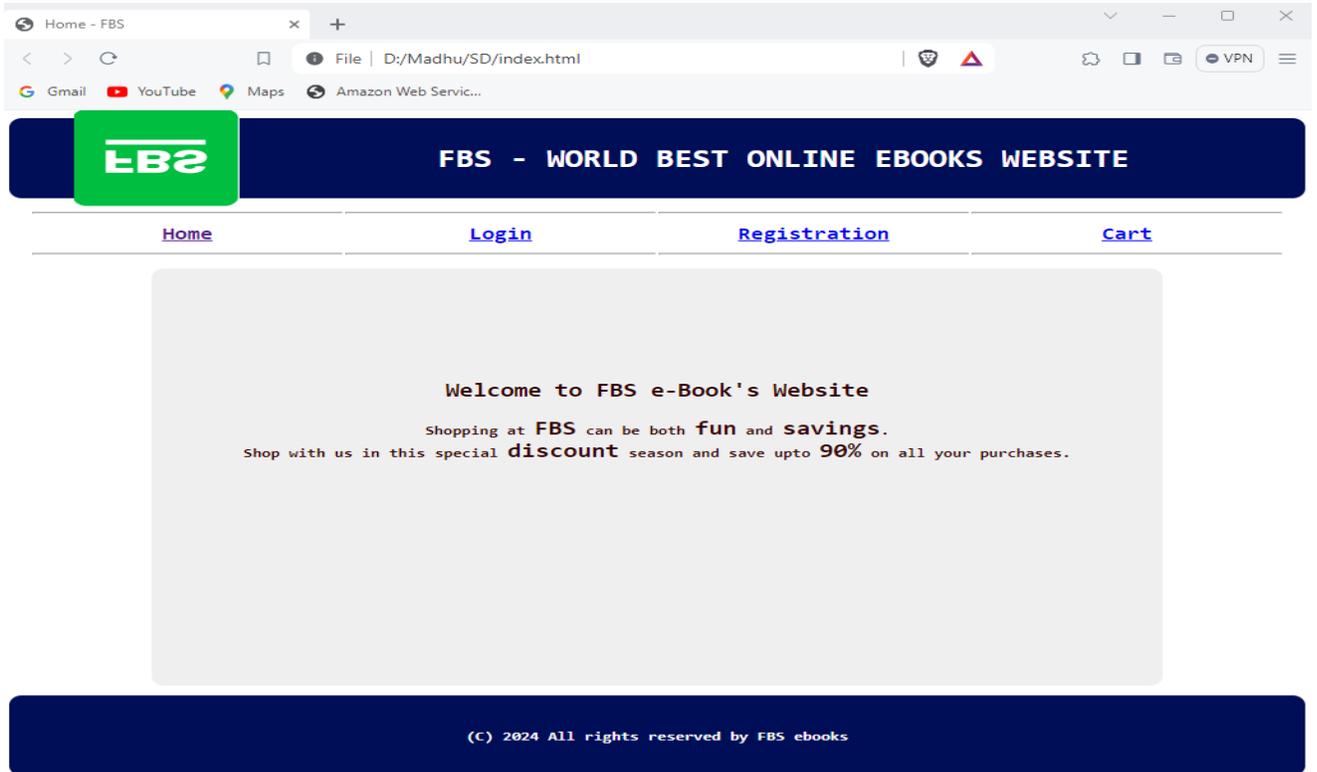
```
.container1 { display:
flex;
}
```

```
.container2{ display:
flex;
flex-direction: column;flex: 1;
}
```

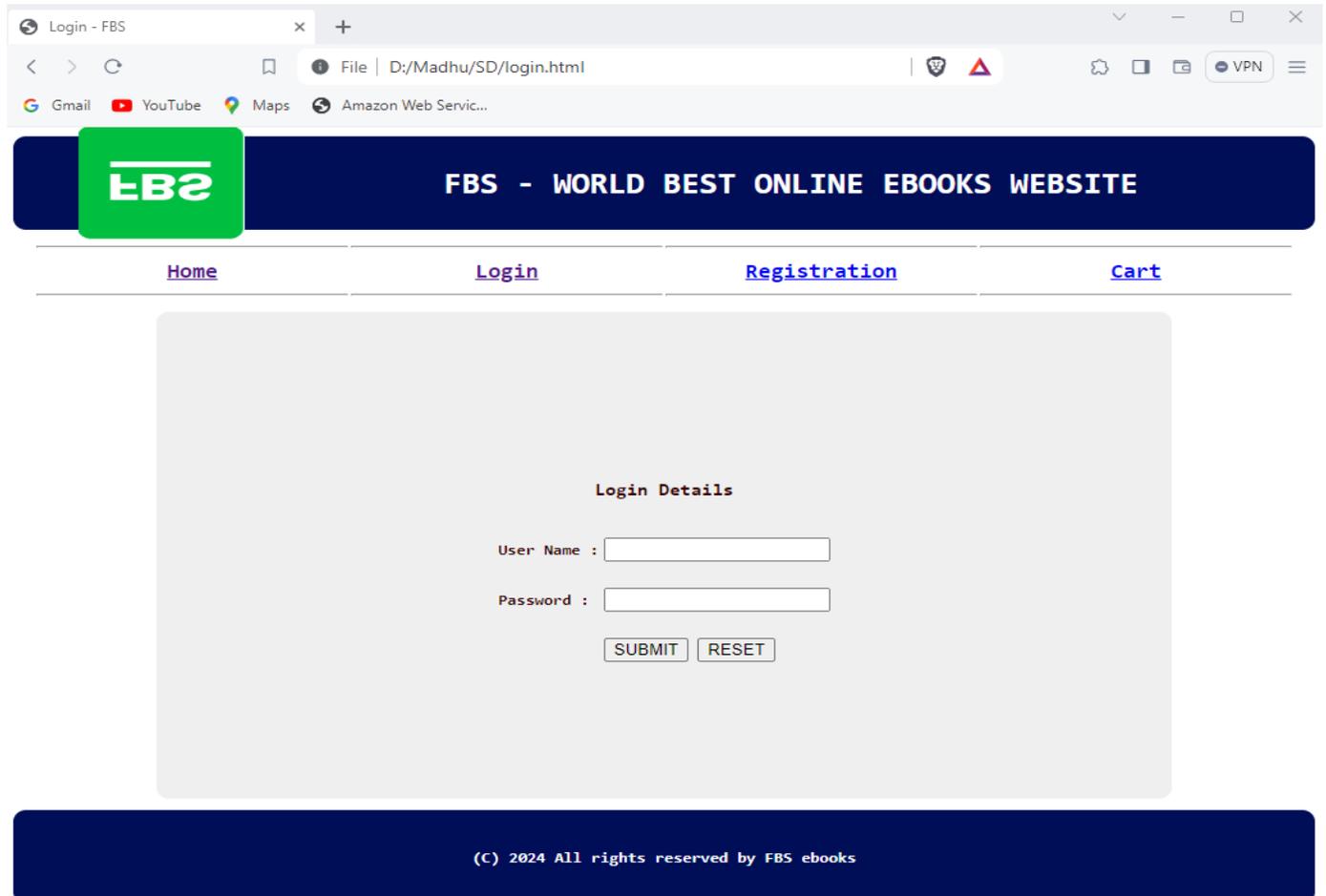
```
header,nav, main, .sidebar1, .sidebar2, footer{display: flex;
align-items: center; justify-
content: center;border-
radius:10px;
}
```

```
.wrapper{ display:
flex;
flex-direction: column;
font-weight:600;
}
```

Output :



The screenshot shows a web browser window with the address bar displaying "File | D:/Madhu/SD/index.html". The page features a dark blue header with a green "EBS" logo on the left and the text "FBS - WORLD BEST ONLINE EBOOKS WEBSITE" on the right. Below the header is a navigation menu with links for "Home", "Login", "Registration", and "Cart". The main content area is a light gray box containing the text: "Welcome to FBS e-Book's Website", "Shopping at FBS can be both fun and savings.", and "Shop with us in this special discount season and save upto 90% on all your purchases." At the bottom of the page is a dark blue footer with the text "(C) 2024 All rights reserved by FBS ebooks".



The screenshot shows a web browser window with the address bar displaying "File | D:/Madhu/SD/login.html". The page features a dark blue header with a green "EBS" logo on the left and the text "FBS - WORLD BEST ONLINE EBOOKS WEBSITE" on the right. Below the header is a navigation menu with links for "Home", "Login", "Registration", and "Cart". The main content area is a light gray box containing the text "Login Details" followed by two input fields: "User Name : " and "Password : ". Below the input fields are two buttons: "SUBMIT" and "RESET". At the bottom of the page is a dark blue footer with the text "(C) 2024 All rights reserved by FBS ebooks".



[Home](#)

[Login](#)

[Registration](#)

[Cart](#)

Registration Form

Name:*
Password:*
Email ID:*
Phone Number:*
Gender:* Male Female
Language Known:* English
 Telugu
 Hindi
 Tamil
Address:*

*fields are mandatory

(C) 2024 All rights reserved by FBS ebooks



[Home](#)

[Login](#)

[Registration](#)

[Cart](#)

Cart

BookName	Price	Quantity	Amount
Java Programming	Rs. 2300/-	2	Rs. 4600/-
Web Technologies	Rs. 3000/-	1	Rs. 3000/-
Total Amount:		3	Rs. 7600/-

(C) 2024 All rights reserved by FBS ebooks

Result :

Hence the program was executed successfully.

Viva Questions :

1. Expand HTML & CSS.
2. Explain in brief about below tags
 - `<h1>`
 - `<p>`
 - `<a>`
 - ``
 - `<div>`
3. What is CSS?
4. Write a short note on following form elements:
 - `<input>`
 - `<label>`
 - `<select>`
 - `<textarea>`
 - `<button>`
5. Write a short note on following form elements:
 - `<fieldset>`
 - `<legend>`
 - `<datalist>`
 - `<output>`
 - `<option>`

EXERCISE 2:

Aim:

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

Solution :

This is views.py

```
from django.shortcuts import render

def home(request):
    return render(request, "home.html")
```

this is project level urls.py

```
#from django.contrib import admin
from django.urls import path

urlpatterns = [
    # path('admin/', admin.site.urls),
]
```

This is app level urls.py

```
from django.contrib import admin
from django.urls import path, include

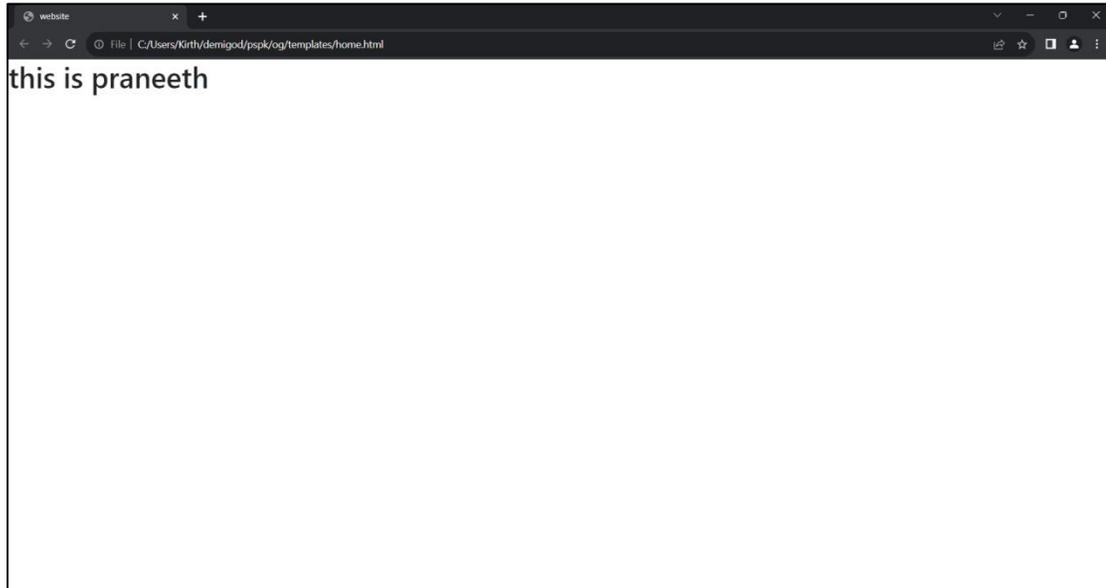
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('app.urls'))
]
```

This is in home.html

```
<!doctypehtml>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>website</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
  </head>
  <body>
    <h1>this is praneeth</h1>
```

```
<scriptsrc="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
C6RzsynM9kWDrMNeT87bh950GNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"crossorigin="anonymous">
</script>
</body>
</html>
```

Output:



Result :

Hence the program was executed successfully.

Viva Questions :

1. What is Bootstrap?
2. What is Responsive Web design?
3. Write advantages of bootstrap.
4. How to create web page with Bootstrap.
5. Write a short note on `.table-responsive` class .

EXERCISE 3:

Aim:

- Use JavaScript for doing client – side validation of the pages.

Solution :

Client - Side validation of Registration Page registration.html

```
<html>
<head>
  <title> Welcome to NNRG 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 validationvar
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 validationlet
flag=false;

```



```

<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>
<input type="checkbox" name="lang" value="Tamil">Tamil
</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 :

The screenshot shows a web browser window with the address bar displaying "D:\Madhu\SD\registrationJS.html". A modal dialog box is open, titled "This page says", with the message "Please Enter UserName" and an "OK" button. Below the dialog, the registration form is visible. It includes the following fields and controls:

- User Name:* (text input)
- Password:* (password input)
- Email ID:* (text input)
- Phone Number:* (text input)
- Gender:* (radio buttons for Male and Female)
- Language Known:* (checkboxes for English, Telugu, Hindi, and Tamil)
- Address:* (text input)

At the bottom of the form, there are "SUBMIT" and "RESET" buttons. A red asterisk message at the bottom of the page reads "*fields are mandatory".

Welcome to NNRG e-Book's we x +

File D:/Madhu/SD/registrationJS.html

Gmail YouTube Maps Ama

All Bookmarks

This page says

Please enter Password not less than 6

OK

User Name:* studyglance123

Password:* ****

Email ID:*

Phone Number:*

Gender:* Male Female

Language Known:* English
 Telugu
 Hindi
 Tamil

Address:*

SUBMIT RESET

**fields are mandatory*

Welcome to NNRG e-Book's we x +

File D:/Madhu/SD/registrationJS.html

Gmail YouTube Maps Ama

All Bookmarks

This page says

Please enter a valid Email ID

OK

User Name:* studyglance123

Password:*

Email ID:* abc

Phone Number:*

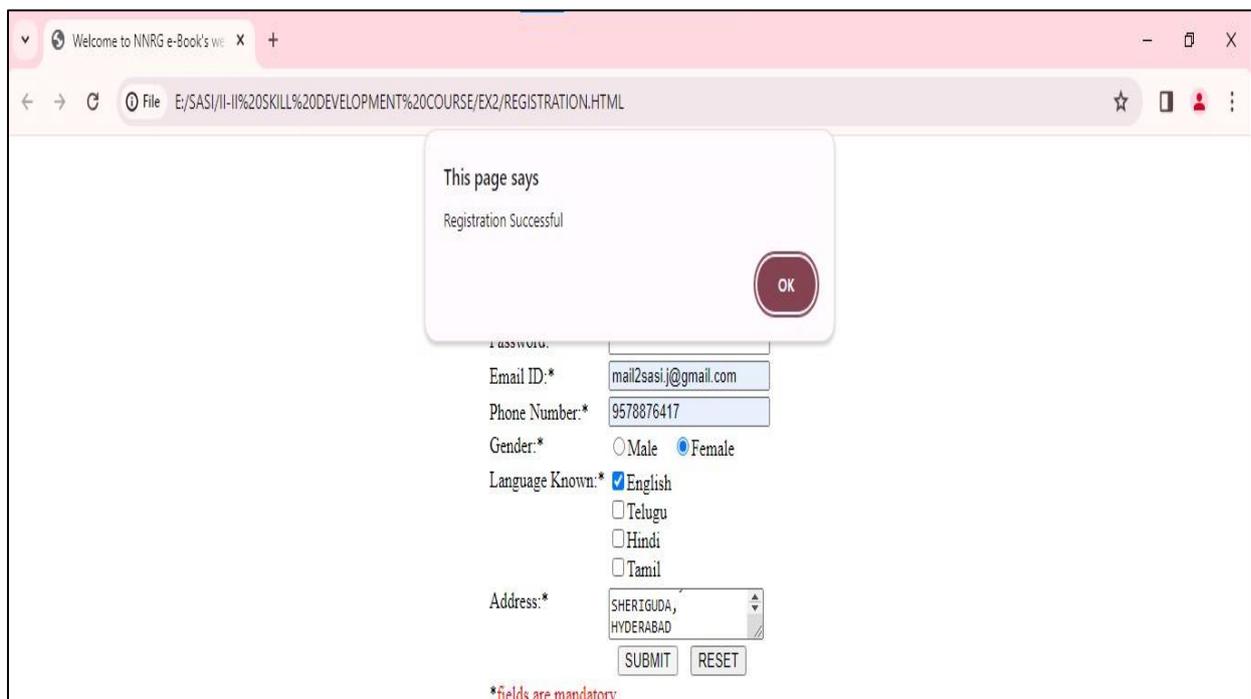
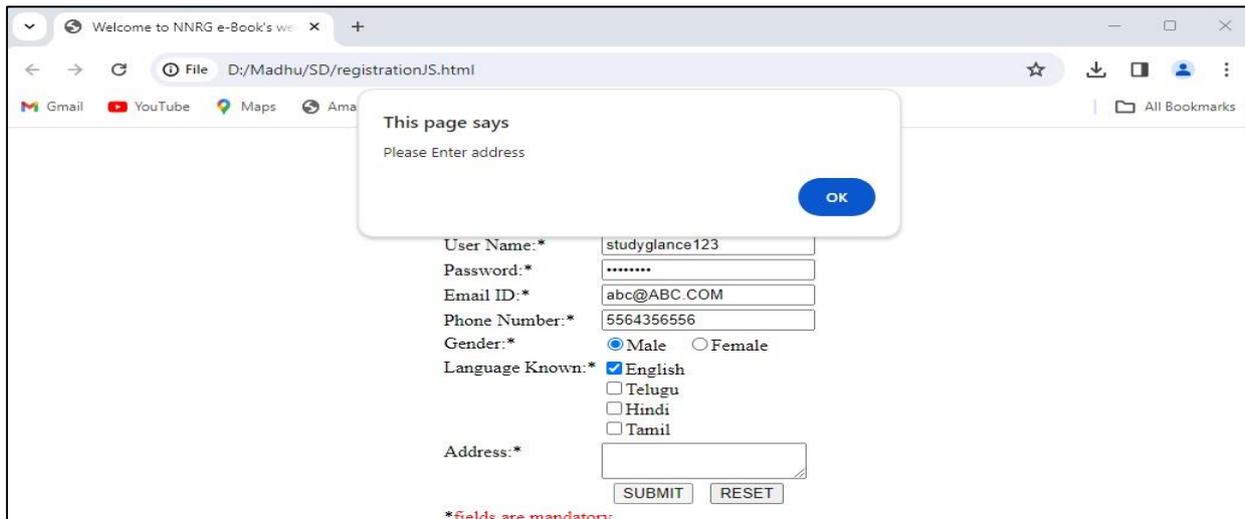
Gender:* Male Female

Language Known:* English
 Telugu
 Hindi
 Tamil

Address:*

SUBMIT RESET

**fields are mandatory*



Result :

Hence the program was executed successfully.

Viva Questions :

- o Define Validation.
- o Define Verification.
- o What is the use of alert message?
- o Define table tag.
- o Expand isNaN.

EXERCISE 4:

Aim:

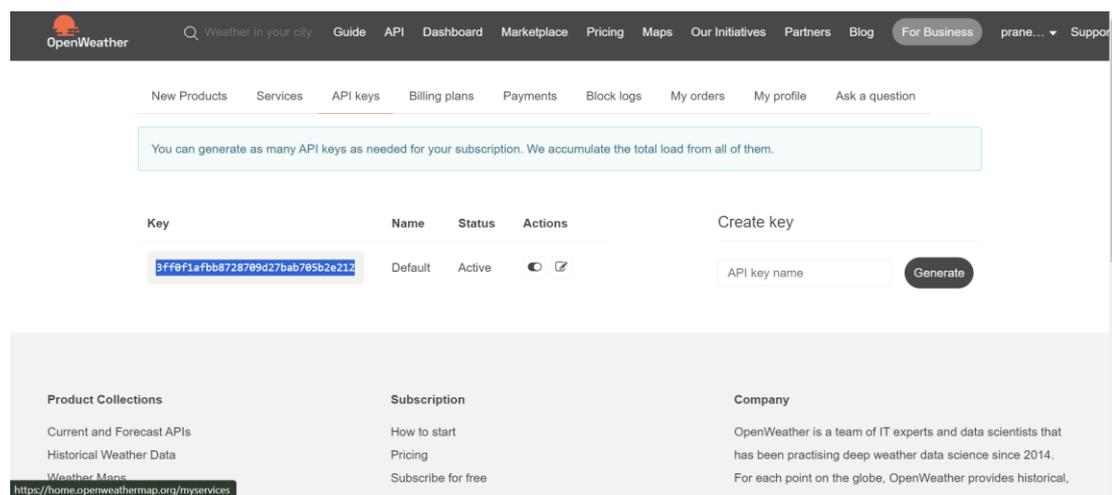
Explore the features of ES6 like arrow functions, call backs, promises, async/wait. Implement an application for reading the weather information from openweathermap.org and display the information in the form of a graph on the web page.

Solution :

STEPS TO CREATE WEATHER PROJECT .

1. CREATE THE PROJECT AND APPLICATION USING CMD(DJANGO INSTALLATION PROCESS) .
2. PROJECT NAME – WEATHERPROJECT.
3. APPLICATION NAME – WEATHERAPP.
4. CREATE THE APPLICATION LEVEL URLS.PY

OPEN WEATHER WEBSITE AND SIGN IN.



CLICK ON API, THEN CLICK ON THE API DOC UNDER THE CONTINUE WEATHER REPORT .

COPY THE URL IN THE NEW WEB PAGE .

```
https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

CITY NAME – REQUIRED
API KEY – PROFILE KEY .

The screenshot shows the OpenWeather API dashboard. At the top, there's a navigation bar with links like 'Weather in your city', 'Guide', 'API', 'Dashboard', 'Marketplace', 'Pricing', 'Maps', 'Our Initiatives', 'Partners', 'Blog', and 'For Business'. Below this, there's a sub-navigation bar with 'New Products', 'Services', 'API keys', 'Billing plans', 'Payments', 'Block logs', 'My orders', 'My profile', and 'Ask a question'. The main content area has a message: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.' Below this is a table with columns: 'Key', 'Name', 'Status', 'Actions', and 'Create key'. The 'Key' column contains a highlighted value: '3ff0f1afbb8728709d27bab705b2e212'. The 'Name' column has 'Default', 'Status' has 'Active', and 'Actions' has icons for a lightbulb and a checkmark. To the right of the table is a 'Create key' section with an input field for 'API key name' and a 'Generate' button. Below the table, there are three sections: 'Product Collections' (with links for 'Current and Forecast APIs', 'Historical Weather Data', and 'Weather Maps'), 'Subscription' (with links for 'How to start', 'Pricing', and 'Subscribe for free'), and 'Company' (with text: 'OpenWeather is a team of IT experts and data scientists that has been practising deep weather data science since 2014. For each point on the globe, OpenWeather provides historical,').

AS PER HIGHLIGHTED FORMAT

The screenshot shows a web browser displaying the JSON response from the OpenWeather API. The URL in the address bar is 'api.openweathermap.org/data/2.5/weather?q=hyderabad&appid=3ff0f1afbb8728709d27bab705b2e212'. The JSON response is as follows:

```
{
  "coord": {
    "lon": 78.4744,
    "lat": 17.3753
  },
  "weather": [
    {
      "id": 300,
      "main": "Drizzle",
      "description": "light intensity drizzle",
      "icon": "09d"
    },
    {
      "id": 500,
      "main": "Rain",
      "description": "light rain",
      "icon": "10d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 294.38,
    "feels_like": 295.01,
    "temp_min": 294.38,
    "temp_max": 294.38,
    "pressure": 1013,
    "humidity": 94,
    "visibility": 4000,
    "wind": {
      "speed": 3.6,
      "deg": 18
    },
    "rain": {
      "1h": 0.21
    },
    "clouds": {
      "all": 180
    },
    "dt": 1701756224,
    "sys": {
      "type": 1,
      "id": 9214,
      "country": "IN",
      "sunrise": 1701738140,
      "sunset": 1701778256,
      "timezone": 19800,
      "id": 1269843,
      "name": "Hyderabad",
      "cod": 200
    }
  }
}
```

NOW IN VS CODE IN VIEWS.PY:

```
from django.shortcuts import render
# import json to load json data to python dictionary
import json
# urllib.request to make a request to api
import urllib.request

def index(request):
    if request.method == 'POST':
        city = request.POST['city']
        ''' api key might be expired use your own api_key
            place api_key in place of appid = "your_api_key_here " '''

        # source contain JSON data from API

        source = urllib.request.urlopen('https://api.openweathermap.org/d
ata/2.5/weather?q=' + city + '&appid=f39864907892c57a3ac10b3b4c73db7d').rea
d()

        # converting JSON data to a dictionary
        list_of_data = json.loads(source)
```



```
rap.min.js"></script>

</head>

<body>
  <navclass="row" style="background: red; color: white;">
    <h1class="col-md-3 text-center">weather</h1>
  </nav>
  <br/>
  <br/>
  <centerclass="row">
    <formmethod="post" class="col-md-6 col-md-offset-3">
      {% csrf_token %}
      <divclass="input-group">
        <inputtype="text" class="form-
control" name="city" placeholder="Search">
        <divclass="input-group-btn">
          <buttonclass="btn btn-default" type="submit">
            <iclass="glyphicon glyphicon-search"></i>
          </button>
        </div>
      </div>
    </form>
  </center>
  <divclass="row">
    {% if country_code and coordinate and temp and pressure and
humidity %}
    <divclass="col-md-6 col-md-offset-3">
      <h3>country code : {{country_code}}</h3>
      <h5>coordinate : {{coordinate}}</h5>
      <h5>temp : {{temp}}</h5>
      <h5>pressure : {{pressure}} </h5>
      <h5>humidity : {{humidity}}</h5>
    </div>
    {% endif %}
  </div>
</body>

</html>
```

Output :



Result :

Hence the program was executed successfully.

Viva Questions :

1. Write short notes on Django Views.
2. What is the function of path()?
3. Give use of `urlpatterns[]` list.
4. Define render() function.
5. How to import python libraries in Django.

EXERCISE 5:

Aim:

- ▣ Develop a java stand-alone application that connects with the database (Oracle / MySQL) and perform the CRUD operation on the database tables.

Solution :

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_name VARCHAR(255),  
s_address VARCHAR(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 ta  
ble:");  
  
            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 t
able");

} 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 +
e +
        "', 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=" + s
id;
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:mys
ql://
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.ge
tString
("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

Result :

Hence the program was executed successfully.

Viva Questions :

1. What are CRUD operations.
2. Write syntax for create & insert command
3. Write short note on select and delete command.
4. Define Primary Key.
5. Define Database.

EXERCISE 6:

Aim:

- 👉 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 :

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

Output :

The screenshot displays an XML validation interface. It is divided into two main sections: 'XML data to validate' and 'XML schema (XSD) data'. The 'XML data to validate' section contains the following XML code:

```
10 <bookstore>
11 <book>
12 <title>Introduction to XML</title>
13 <author>John Doe</author>
14 <price>29.99</price>
15 </book>
16 <book>
17 <title>Programming with XML</title>
18 <author>Jane Smith</author>
19 <price>39.99</price>
20 </book>
21 </bookstore>
22
```

The 'XML schema (XSD) data' section contains the following XSD code:

```
18 </xs:complexType>
19
20 <xs:complexType name="bookType">
21 <xs:sequence>
22 <xs:element name="title" type="xs:string"/>
23 <xs:element name="author" type="xs:string"/>
24 <xs:element name="price" type="xs:decimal"/>
25 </xs:sequence>
26 </xs:complexType>
27
28 </xs:schema>
29
```

At the bottom right of the interface, there is a 'Validate' button. Below the main interface, a green bar displays the text 'Document Valid'.

Result :

Hence the program was executed successfully.

Viva Questions :

1. What is Xml?
2. Difference between Xml and Html.
3. Difference between DTD and XSD.
4. Define Schema.
5. What is a well formed XML document?

EXERCISE 8:

Aim:

▣ Maintaining the transactional history of any user is very important .Explore the various session tracking mechanism (cookies ,HTTP session).

Solution :

Steps:

1. Create a Django project named as sessionapp.
2. Create a Django app named as seapp

This is views.py:

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 # set sessions
5 def setsession(request):
6     request.session['name'] = 'sriniketh'
7     return render(request, 'setsessions.html')
8
9 # get sessions
10 def getsession(request):
11     name = request.session['name'] = 'sriniketh'
12     return render(request, 'getsessions.html', {'name':name})
13
14
15 # del sessions
16
17 def delsession(request):
18     if 'name' in request.session:
19         del request.session['name']
20     return render(request, 'delsessions.html')
```

This is project level urls.py:

```
from django.contrib import admin
from django.urls import path
from seapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('set/', views.setsession),
    path('get/', views.getsession),
```

```
    path('del/', views.delsession),  
]
```

No need to create app level urls.py..

3]create templates folder in app level.

delsessions.html:

```
<!DOCTYPE html>  
<html lang = "en">  
<head>  
    <meta charset = "UTF-8">  
    <meta name="viewport" content = "width=device-width,initial-scale=1.0">  
    <title>del sessions</title>  
</head>  
<body>  
    <h4>sessions deleted</h4>  
</body>  
</html>
```

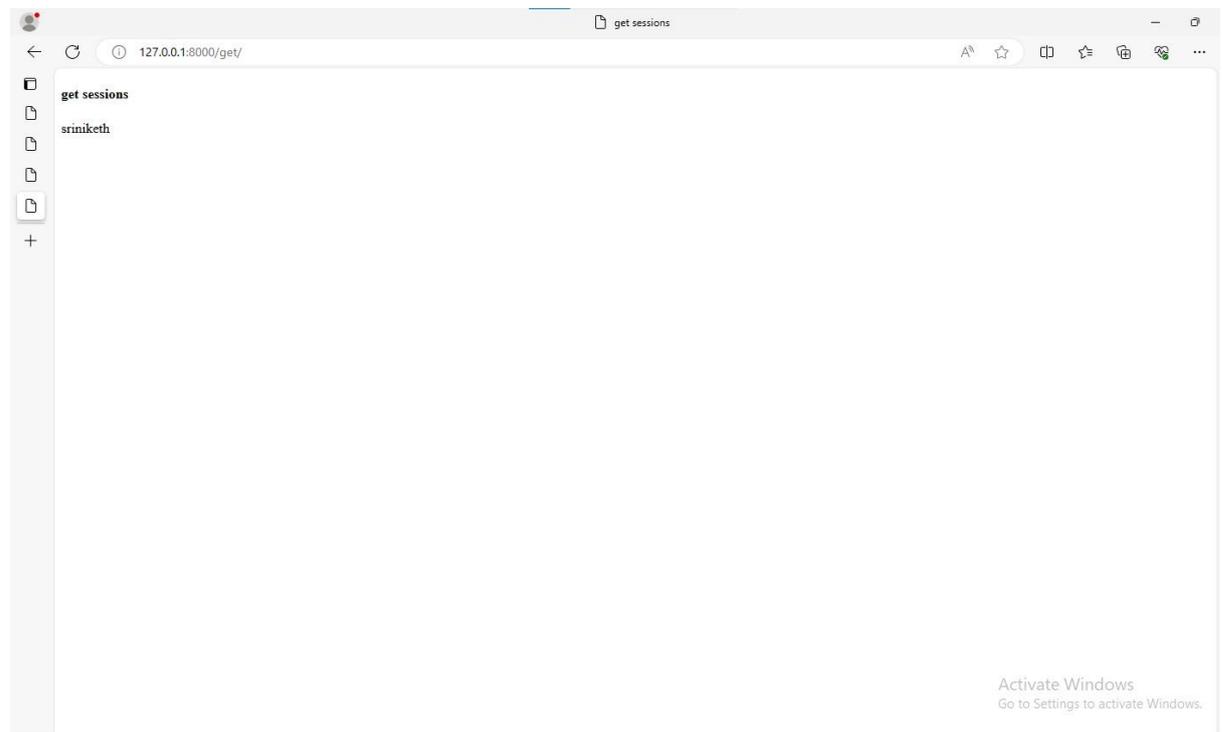
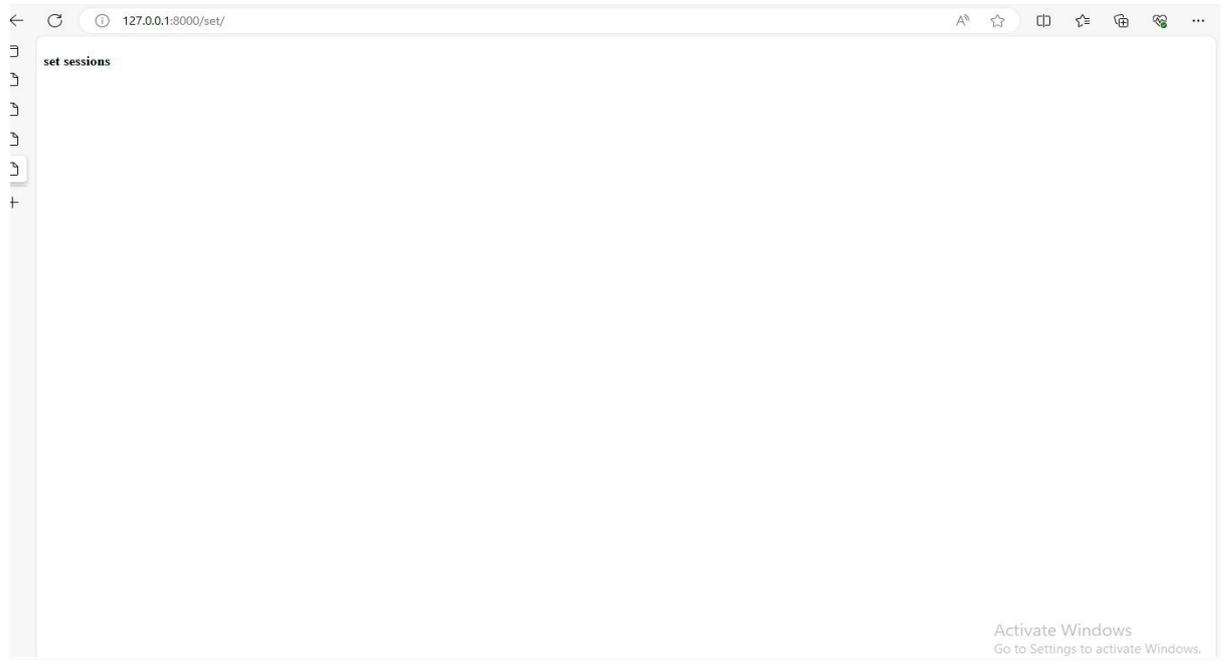
setsessions.html:

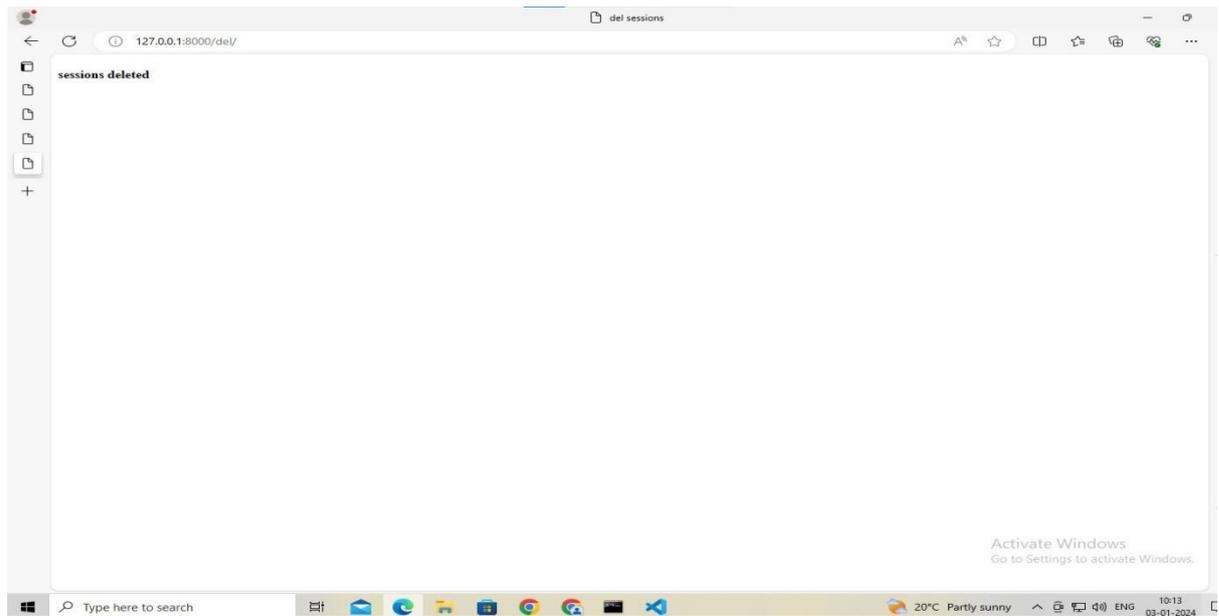
```
<!DOCTYPE html>  
<html lang = "en">  
<head>  
    <meta charset = "UTF-8">  
    <meta name="viewport" content = "width=device-width,initial-scale=1.0">  
    <title>set sessions</title>  
</head>  
<body>  
    <h4>set sessions</h4>  
    {{ name }}  
</body>  
</html>
```

getsessions.html:

```
<!DOCTYPE html>  
<html lang = "en">  
<head>  
    <meta charset = "UTF-8">  
    <meta name="viewport" content = "width=device-width,initial-scale=1.0">  
    <title>get sessions</title>  
</head>  
<body>  
    <h4>get sessions</h4>  
    {{ name }}  
</body> </html>
```

Output :





Cookies:

- 1) Create a project
- 2) Create a app in the command prompt

This is views.py:

```
from django.shortcuts import render
from datetime import datetime, timedelta
# Create your views here.

# set sessions
def setcookie(request):
    response = render(request, 'cookieapp/setcookies.html')
    response.set_cookie('name', 'praneeth', expires =
datetime.utcnow()+timedelta(days=2))
    return response
```

```
# get sessions
def getcookie(request):
    name = request.COOKIES.get('name')
    return render(request, 'cookieapp/getcookies.html', {'name':name})
```

```
# del sessions
```

```
def delcookie(request):
    response = render(request, 'cookieapp/delcookies.html')
    response.delete_cookie('name')
    return response
```

This is project level urls.py:

```
from django.contrib import admin
from django.urls import path
from cookieapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('set/', views.setcookie),
    path('get/', views.getcookie),
    path('del/', views.delcookie),
]
```

THIS IS getcookie.html :

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <meta charset = "UTF-8">
  <meta name="viewport" content = "width=device-width,initial-scale=1.0">
  <title>get cookie</title>
</head>
<body>
  <h4>get cookie</h4>
  {{name}}
</body>
</html>
```

This is setcookie.html:

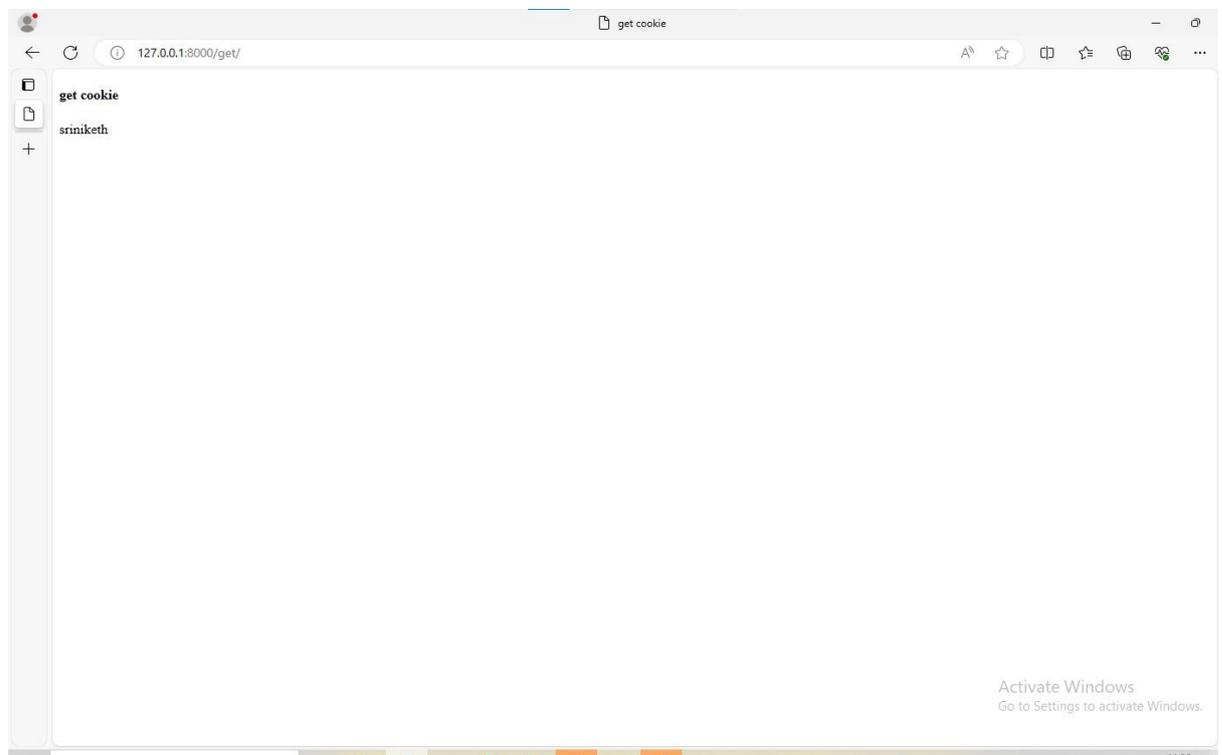
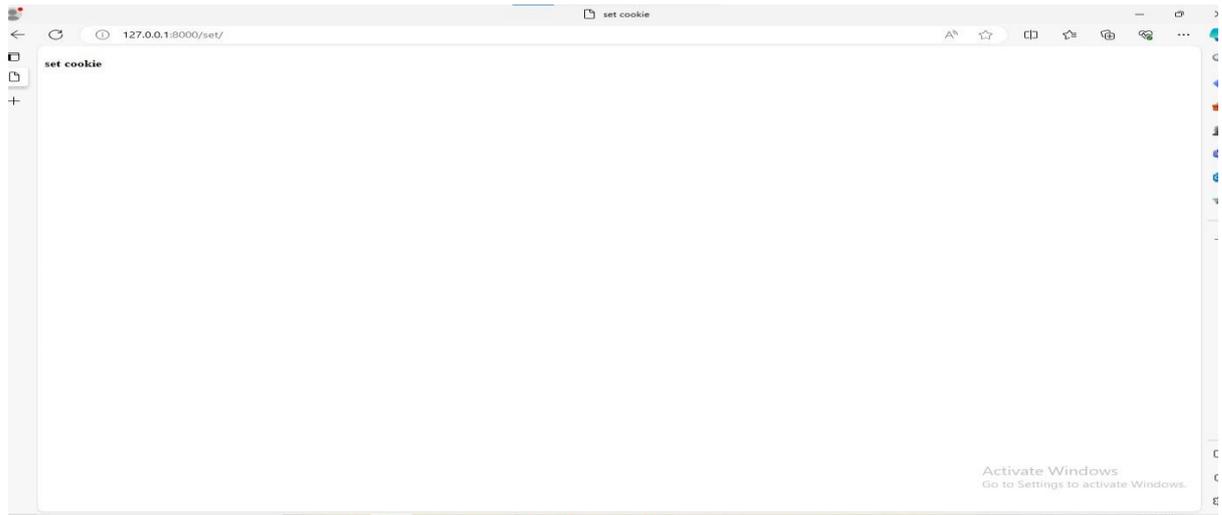
```
<!DOCTYPE html>
<html lang = "en">
<head>
  <meta charset = "UTF-8">
  <meta name="viewport" content = "width=device-width,initial-scale=1.0">
  <title>set cookie</title>
</head>
<body>
  <h4>set cookie</h4>
  {{name}}
</body>
</html>
```

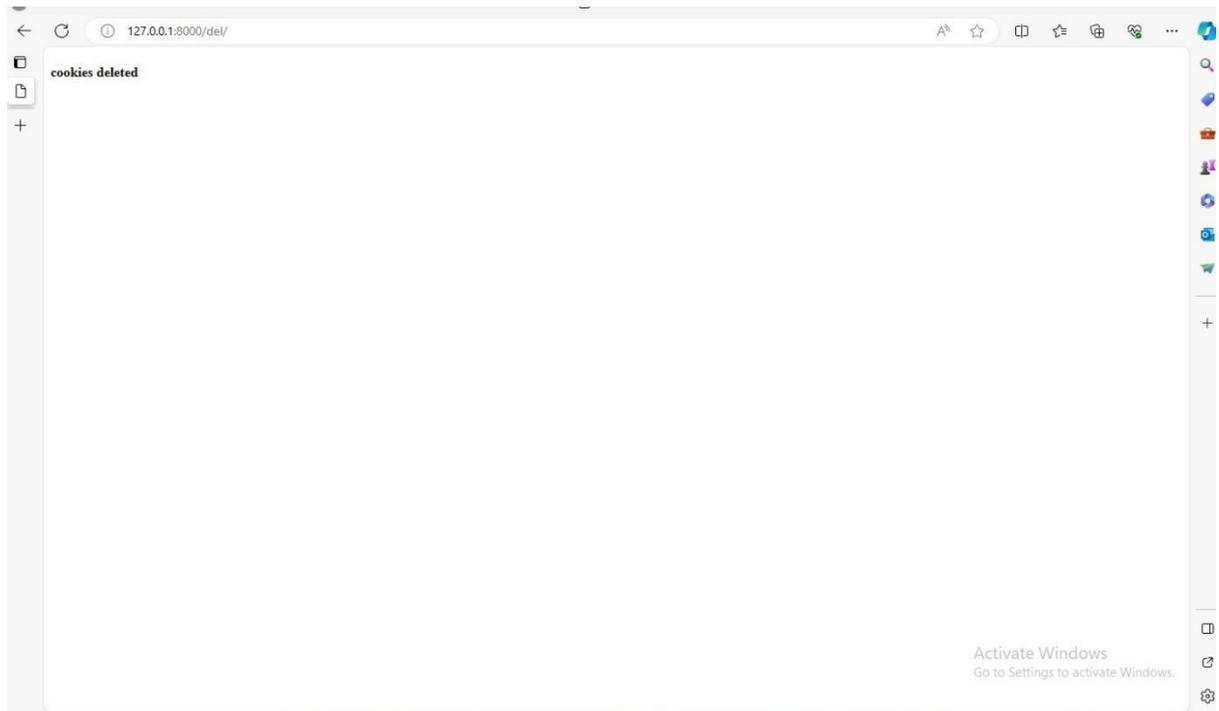
This is delcookie.html :

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

```
<body>
  <h4>cookies deleted</h4>
</body>
</html>
```

Output :





Result :

Hence the program was executed successfully.

Viva Questions :

- 1. Define cookies.**
- 2. What is session mechanism?**
- 3. How to delete cookies.**
- 4. Define Http session.**
- 5. How to set cookies.**

EXERCISE 9:

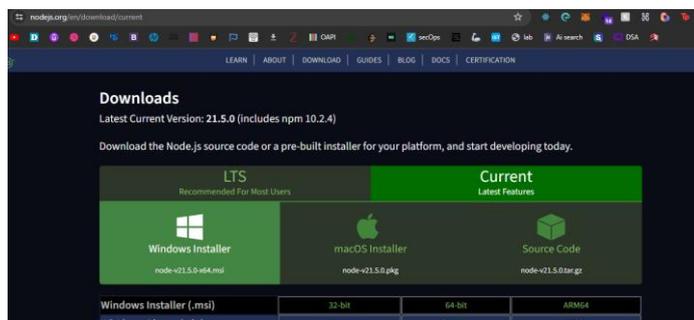
Aim:

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

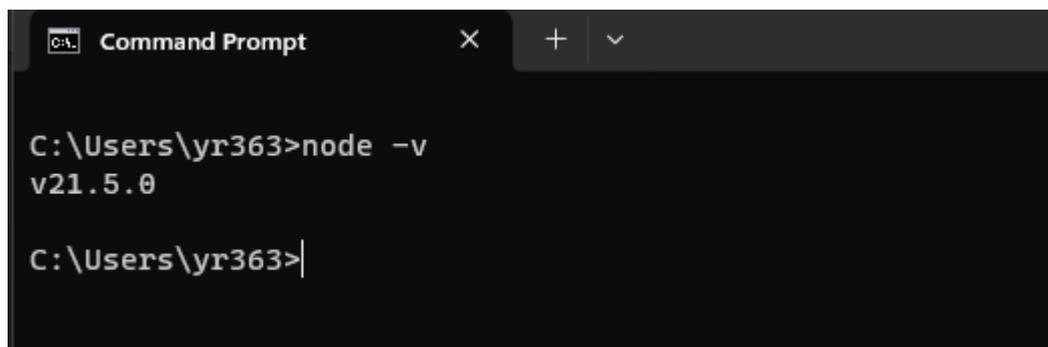
Solution :

NodeJs Installation Steps

1. Download the Node.js installer from the official website.



2. Run the downloaded .msi file.
3. Follow the setup wizard instructions, accept the End-User License Agreement (EULA), set the destination folder, and choose the components to install.
4. Complete the installation.
5. Verify the installation by opening your command prompt or Windows Powershell and running the command `node -v`



Now Installation of node is done.

To create custom server we need to write a javascript code, mostly we use file name server.js or index.js

Server.js

```
const http = require('http');
const os = require('os');
const path = require('path');
const EventEmitter = require('events');
```

```

const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello, this is your custom server!\n');
});

const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});

// os:
console.log('OS Platform:', os.platform());
console.log('CPU Architecture:', os.arch());
console.log('Free Memory:', os.freemem());
console.log('Total Memory:', os.totalmem());

// path
const filePath = 'C:/Users/yr363/Downloads/exp9/file.txt';
//replace the path with your system path
console.log('File Name:', path.basename(filePath));
console.log('Directory Name:', path.dirname(filePath));
console.log('File Extension:', path.extname(filePath));

// events:
class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();

myEmitter.on('customEvent', (arg) => {
  console.log('Custom Event Occurred:', arg);
});

myEmitter.emit('customEvent', 'This is an argument for the event');

```

Now We have created our server.js file to run the server using node we need to follow:

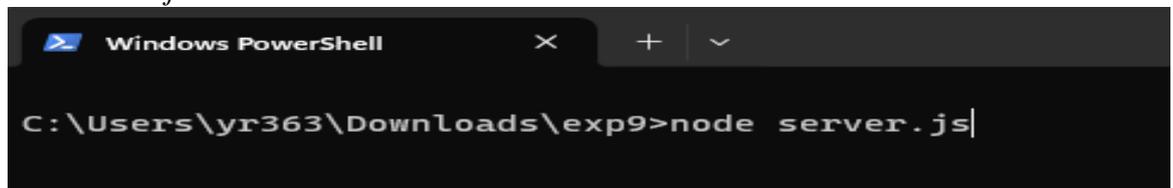
1. Open a terminal or command prompt in the directory where the server.js file is located. (In My Case it is "C:\Users\yr363\Downloads\exp9").

```
C:\Users\yr363\Downloads\exp9>dir
Volume in drive C is Acer
Volume Serial Number is AA2C-9B17

Directory of C:\Users\yr363\Downloads\exp9

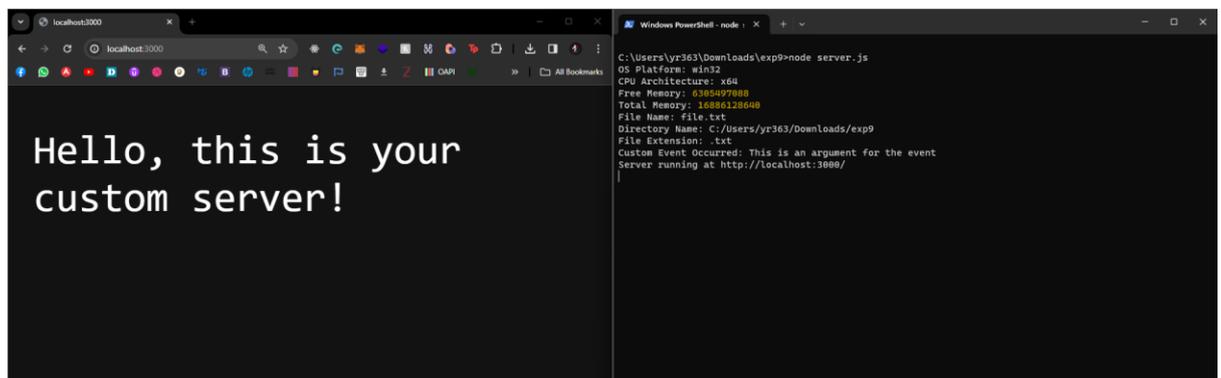
08-01-2024  23:31    <DIR>          .
08-01-2024  23:12    <DIR>          ..
08-01-2024  23:26                0 file.txt
08-01-2024  23:28                1,199 server.js
                2 File(s)      1,199 bytes
                2 Dir(s)    264,603,234,304 bytes free
```

2. Run the following command to start the server:
`node server.js`



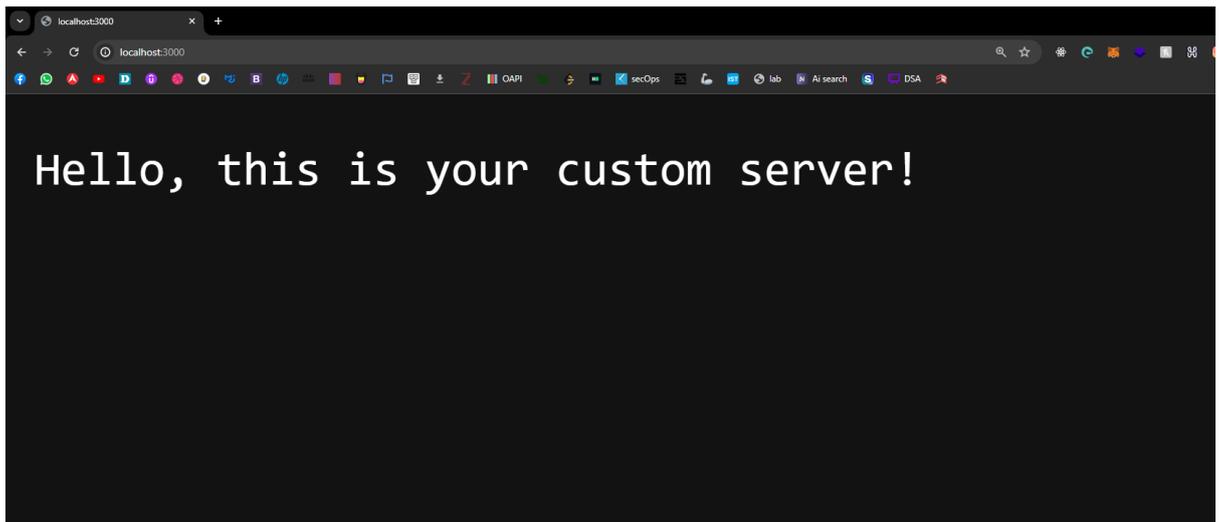
```
Windows PowerShell
C:\Users\yr363\Downloads\exp9>node server.js|
```

Output :



Now our server is running

Below is the response came from the server after it is running without any error



Below it shows that the server is running in localhost:3000 as we used os, path, events as we run the server we will get the stats of our system, path of a file we want to know and event occurred

```
Windows PowerShell - node : X + v
C:\Users\yr363\Downloads\exp9>node server.js
OS Platform: win32
CPU Architecture: x64
Free Memory: 6305497088
Total Memory: 16886128640
File Name: file.txt
Directory Name: C:/Users/yr363/Downloads/exp9
File Extension: .txt
Custom Event Occurred: This is an argument for the event
Server running at http://localhost:3000/
```

The last line “Server running at <http://localhost:3000/>” refers that our server is running on the port number 3000

Result :

Hence the program was executed successfully.

Viva Questions :

1. What is Node.js? Where can we use it.
2. How does Node.js work?
3. Write the steps to install Node.js.
4. Difference between angular and node.js
5. Define server.

EXERCISE 10:

Aim:

👉 **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:

```
• npm init -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.');
```

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 = ? WH
ERE 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 });
  });
});

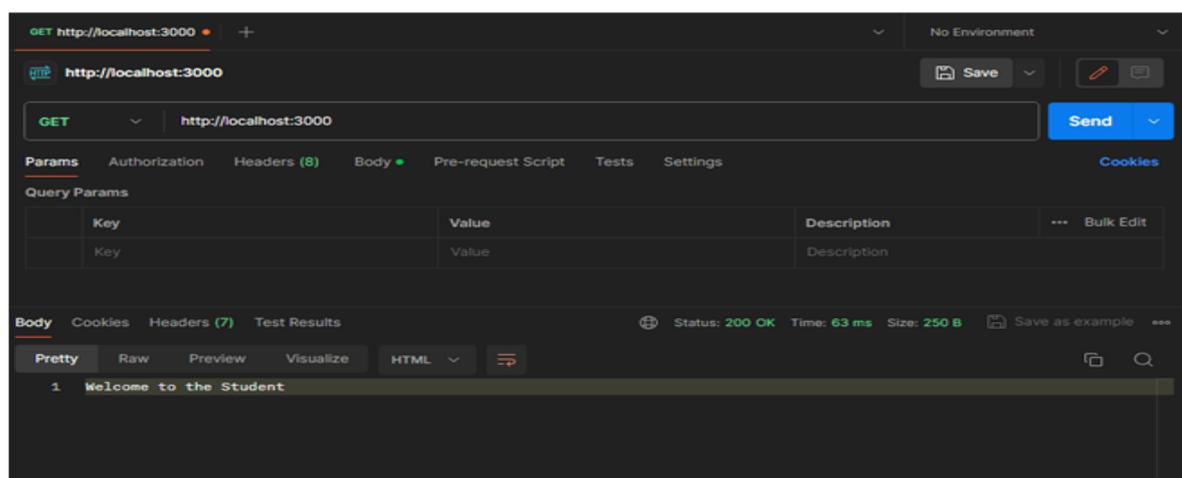
app.listen(port, () => {
  console.log('Server running at http://localhost:${port}');
});

```

Output :

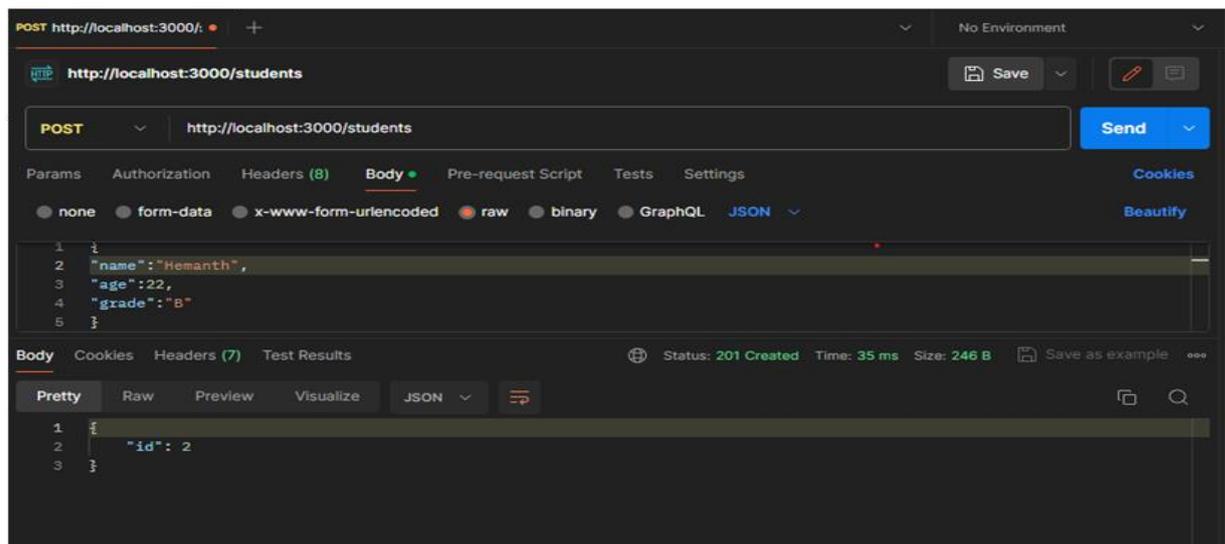
GET:

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



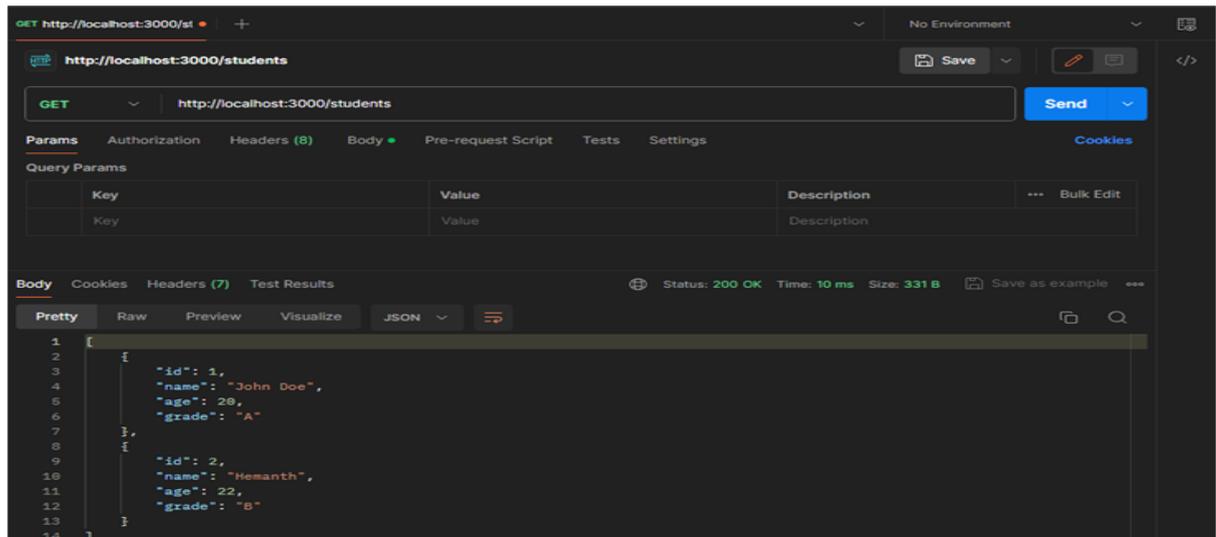
POST : Create a New Student

- 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.



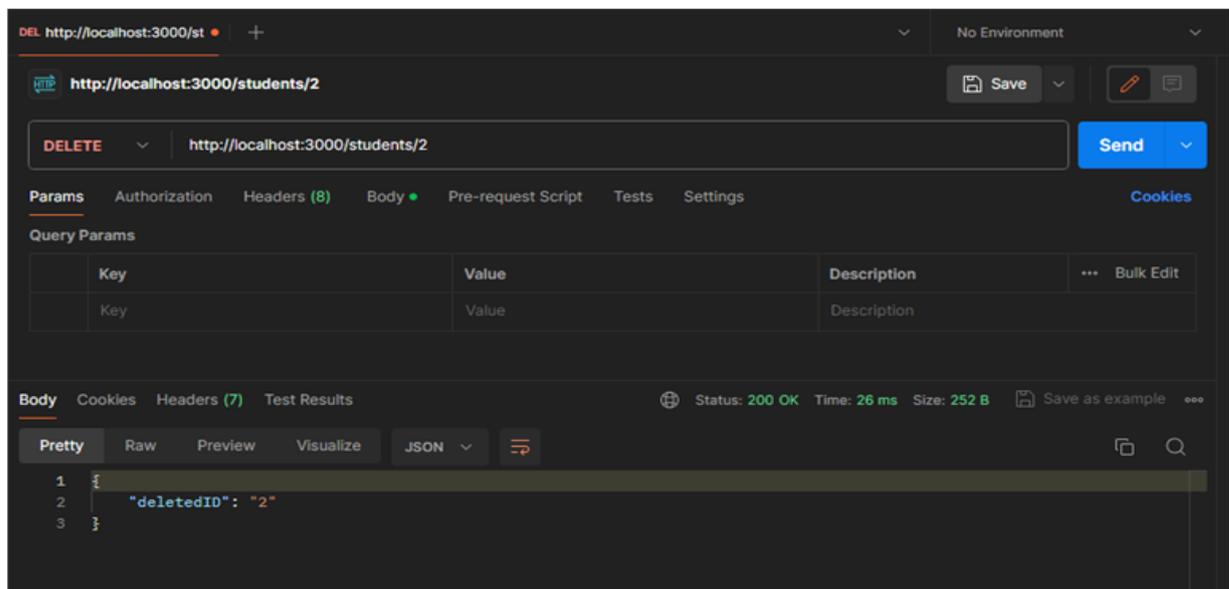
GET: #all Students

- 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:

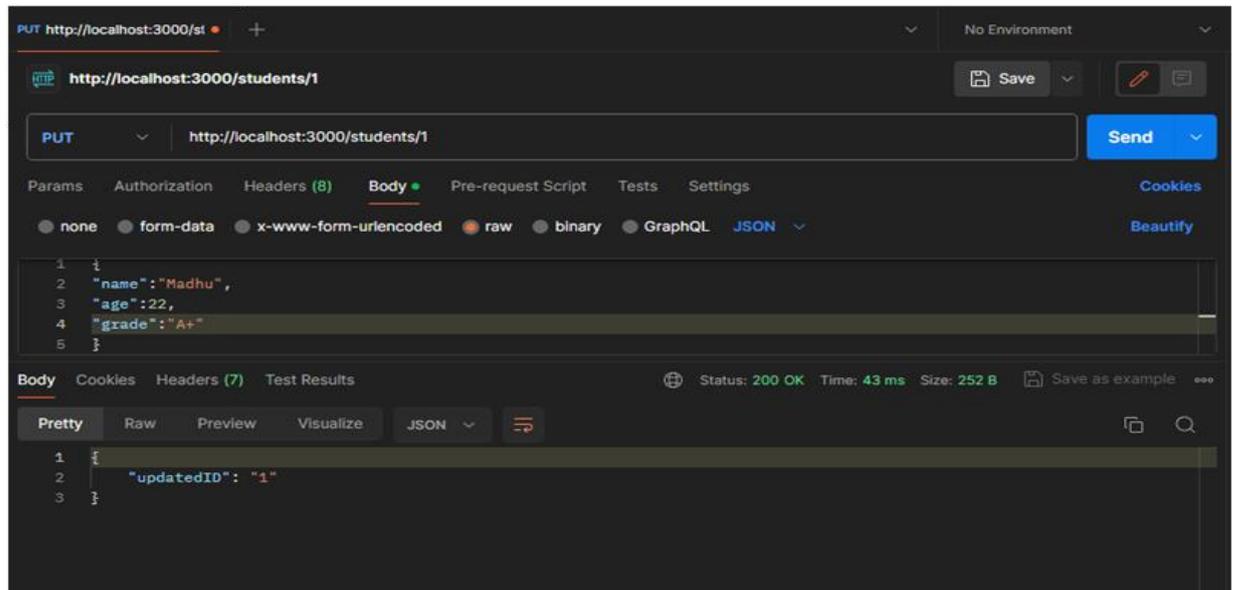
- Set the request type to DELETE.
- Enter the URL for the student you want to delete (replace: id with an actual student ID): *http://localhost:3000/students/:id*
- Place instead of ID which replace with number that is ID to be deleted.
- Then click **send**



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



Result :

Hence the program was executed successfully.

Viva Questions :

1. What is web application?
2. What Rest Api.
3. Write short note on postman api.
4. Explain about GET & PUT.
5. How to create a serializers.py to convert complex data types.

EXERCISE 11:

Aim:

- For the above application create authorized end points using JWT(JSON Web Token).

Solution :

- 1) create a project
- 2) create an application
- 3) in app/models.py create email and password fields in place of admin login .
add auth_user_model in settings.py
- 4)include that in installed apps.
- 5)python manage.py makemigrations,
python manage.py migrate
python manage.py makemigrations appname
- 6)make changes in admin.py at application level
- 7)create superuser
- 8) once check with the server .and login into the admin and go through users .
- 9)Next Step is to install Django Rest Framework in your Project Directory. Django Rest Framework (DRF) is a toolkit build in web application which is used for creating web API's. It returns web API's in the form of raw JSON. Install DRF using following command.
-->pip install djangorestframework
-->add rest_framework in settings.py
- 10) create serializers.py in app
- 11) define the views in application
- 12)define urls for the views in application
- 13) add urls of application at project level
- 14)now we will use simplejwt to login user
We will use simple JWT to login user an generate access and refresh Token for authentication of user. Install simple JWT in your project directory with the pip command.

-->pip install djangorestframework-simplejwt

15) add rest_framework_simplejwt in installed apps of settings

16)configure jwt at settings

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
}
```

17) lastly, In our app level urls.py we'll add routes for simple jwt as TokenObtainPairView and TokenRefreshView views.

18) use postman

first register using post request- because to store into database

-body

-form-data

19) after registering login we will get two tokens access and refresh token

20)Access Token : Access token is the encoded string which contains information about user, permissions etc. Token are used as a bearer token, Bearer means which hold data in it. An access token is put in the Authorization header of your request for the user's API.

Refresh Token : An access tokens have very short life span because of the security purpose. When it expires a user need to generated new access token for authentication. So refresh token is used to request new access tokens without user interaction.

21)We can also customize the behavior of simple JWT by changing some of the settings variables in the settings.py. Copy and paste the following code in settings.py file.

THIS IS VIEWS.PY:

```
from django.shortcuts import render
from rest_framework.views import APIView
from .serializers import UserSerializer
from rest_framework.response import Response

# view for registering users
class RegisterView(APIView):
    def post(self, request):
        serializer = UserSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        serializer.save()
        return Response(serializer.data)
```

THIS IS URLS.PY IN APP LEVEL:

```
from django.urls import path
from .views import RegisterView
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
)

urlpatterns = [
    path('api/login/', TokenObtainPairView.as_view(),
name='token_obtain_pair'),
    path('api/login/refresh/', TokenRefreshView.as_view(),
name='token_refresh'),
    path('api/register/', RegisterView.as_view(), name="sign_up"),
]
```

THIS IS URLS.PY IN PROJECT LEVEL:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('account/', include('myapp.urls')),
]
```

THIS IS MODEL.PY:

```
from django.db import models

# Create your models here.
#from django.db import models
from django.contrib.auth.models import AbstractUser, BaseUserManager

class UserManager(BaseUserManager):

    use_in_migration = True

    def create_user(self, email, password=None, **extra_fields):
        if not email:
            raise ValueError('Email is Required')
        user = self.model(email=self.normalize_email(email),
**extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user
```

```

def create_superuser(self, email, password, **extra_fields):
    extra_fields.setdefault('is_staff', True)
    extra_fields.setdefault('is_superuser', True)
    extra_fields.setdefault('is_active', True)

    if extra_fields.get('is_staff') is not True:
        raise ValueError('Superuser must have is_staff = True')
    if extra_fields.get('is_superuser') is not True:
        raise ValueError('Superuser must have is_superuser = True')

    return self.create_user(email, password, **extra_fields)

class UserData(AbstractUser):

    username = None
    name = models.CharField(max_length=100, unique=True)
    email = models.EmailField(max_length=100, unique=True)
    date_joined = models.DateTimeField(auto_now_add=True)
    is_admin = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['name']

    def _str_(self):
        return self.name

```

THIS IS ADMIN.PY:

```

from django.contrib import admin
from .models import UserData

admin.site.register(UserData)

```

THIS IS SERIALIZERS.PY:

```

from rest_framework import serializers
from .models import UserData

class UserSerializer(serializers.ModelSerializer):

    class Meta:
        model = UserData

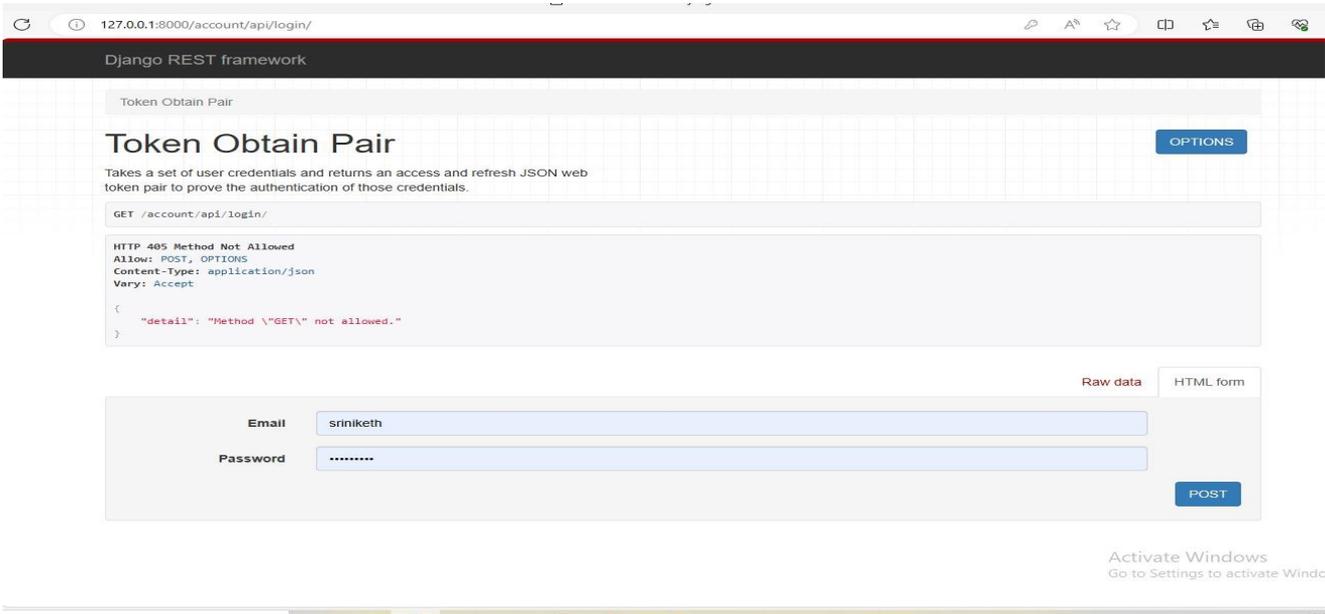
```

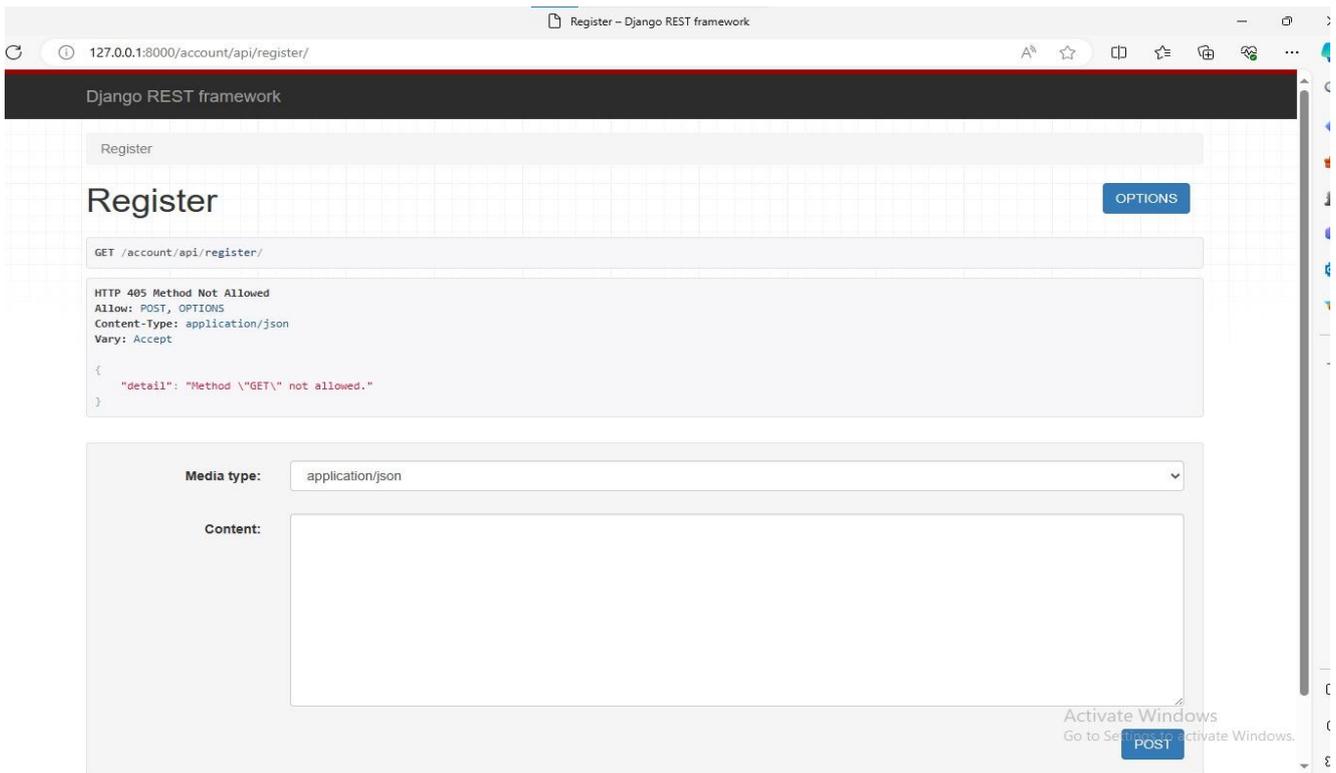
```
fields = ["id", "email", "name", "password"]

def create(self, validated_data):
    user = UserData.objects.create(email=validated_data['email'],
                                   name=validated_data['name']
                                   )
    user.set_password(validated_data['password'])
    user.save()
    return user
```

In settings.py include the app name in installed apps and also add the rest_framework in it

Output :





Result :

Hence the program was executed successfully.

Viva Questions :

1. What is JWT.
2. What is the need for JWT.
3. What are advantages of JWT.
4. What is structure of JWT.
5. How to install Django Rest Framework in your Project Directory

EXERCISE 12:

Aim:

👉 Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.

Solution :

Creating a React application for a student management system involves several steps. Below is a basic example to get you started with registration, login, contact, and about pages using React Router for navigation.

Firstly, make sure you have Node.js installed on your machine. Then, follow these steps:

1. ****Create React App:****

Open your terminal and run the following command to create a new React app:
npx create-react-app student-management-system

2. ****Install React Router:****

Navigate to your project folder:
cd student-management-system
Install React Router:
npm install react-router-dom

3. ****Create Components:****

Inside the `src` folder, create components for `Registration`, `Login`, `Contact`, and `About`. For instance, in the `src` folder, create files like `Registration.js`, `Login.js`, `Contact.js`, and `About.js`. These files will contain your component logic and JSX.

4. ****Set Up Routing:****

Create a file called `AppRouter.js` to handle routing:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Registration from './Registration';
import Login from './Login';
import Contact from './Contact';
import About from './About';

const AppRouter = () => {
  return (
```

```

    <Router>
      <Switch>
        <Route exact path="/" component={Login} />
        <Route path="/registration" component={Registration} />
        <Route path="/contact" component={Contact} />
        <Route path="/about" component={About} />
      </Switch>
    </Router>
  );
};

export default AppRouter;

```

5. ****Implement Navigation:****

Update your `App.js` to use the `AppRouter` component:

```

import React from 'react';
import AppRouter from './AppRouter';

function App() {
  return (
    <div className="App">
      <AppRouter />
    </div>
  );
}

```

export default App;

6. ****Add Navigation Links:****

In each component (`Login.js`, `Registration.js`, etc.), use `Link` from `react-router-dom` to navigate between pages:

```

import React from 'react';
import { Link } from 'react-router-dom';

const Login = () => {
  return (
    <div>
      <h1>Login Page</h1>
      { /* Add Link to other pages */ }
      <Link to="/registration">Register</Link>
      <Link to="/contact">Contact</Link>
      <Link to="/about">About</Link>
    </div>
  );
};

```

export default Login;

7. ****Run the App:****

Start your React app:

npm start

This will launch your application in the browser. You should be able to navigate between the pages using the links provided.

Output :

Login

Email

Password

SIGN IN

Don't have an account? [Sign up](#)

Sign up

Email Address

Username

Phone Number

Password

SIGN UP

Already have an account? [Login](#)

Result :

Hence the program was executed successfully.

Viva Questions :

1. What is React and JSX?
2. What are the major features of React?
3. How to create components in React.
4. What are the advantages of using React?
5. What are the limitations of React?

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>
  );
};

export default App;
```

Your output should look like this:

Output :

Initial Screen



Enter city name Get Weather

Loading weather data...

After Supply the City name

london	Get Weather
London	
Temperature: 21.83°C	
Description: overcast clouds	
Feels like : 21.35°C	
Humidity : 49%	
Pressure : 1015	
Wind Speed : 2.57m/s	

Result :

Hence the program was executed successfully.

Viva Questions :

1. What is use state Hook?
2. What is use effect Hook?
3. How to create weather component.
4. What is axios?
5. What are the rules that must be followed while using React Hooks?

EXERCISE 14:

Aim:

- 👉 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, and you will 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>
  );
}
export default 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>
);
}
export default 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's an example of how you can style the todo items. In the `App.css` file, add the following styles:

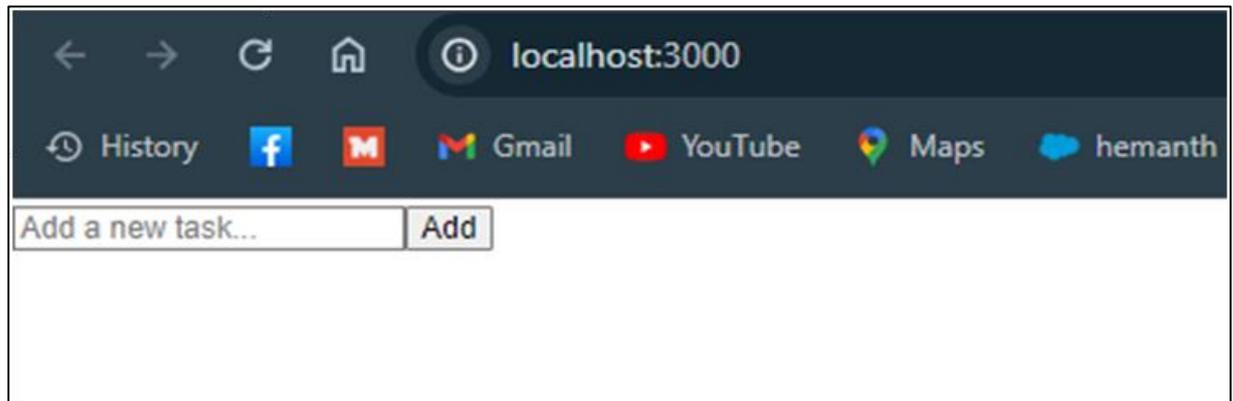
```

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

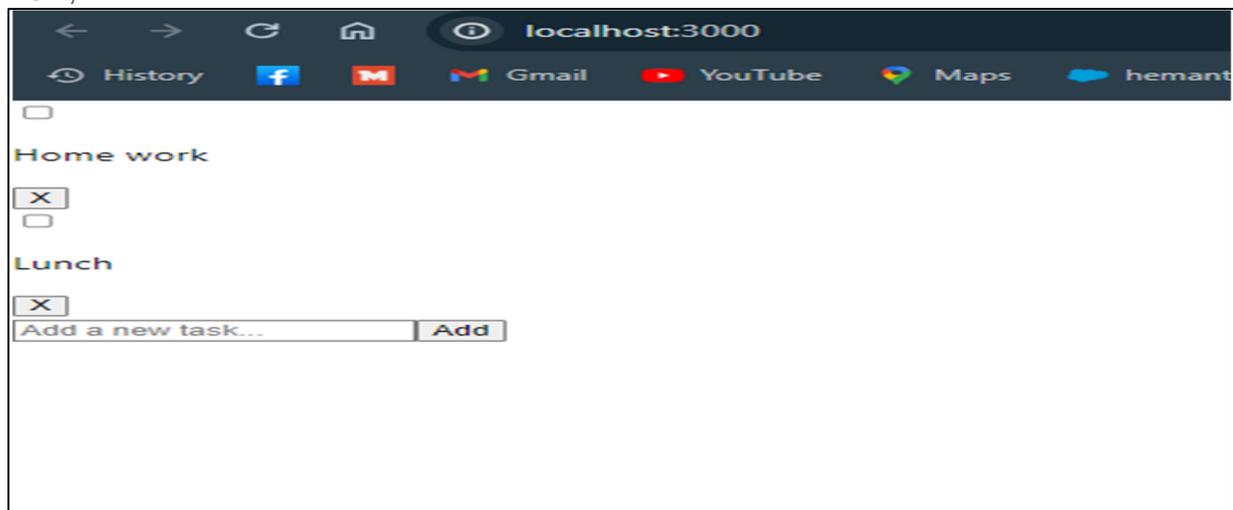
```

Output :

Initially it looks like:



Next,



Result :

Hence the program was executed successfully.

Viva Questions :

- 1.How to create TODO list.
- 2.Explain about the advantages of TODO list.
- 3.what is `<form method="POST">`?
4. Explain the following:
`<form action="/del/{i.id}" method="POST" style=" padding-right: 4%; padding-bottom: 3%;">`
- 5.What is react?