MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Department of Computer Science and Engineering (AIML)

Skill Development Lab(NodeJS/ReactJS/Django)

II B.TECH-II SEMESTER

MLRS-BT25

A.Y.2026-2027

INDEX

Department of Computer Science & Engineering (AI&ML)

CERTIFICATE

This is to certify that this manual is a Bonafide record of practical work carried out **in NODE JS/ REACT JS/ DJANGO LAB** for the **II B.Tech Computer Science Engineering (AI&ML) II Semester** Programme during the academic year **2025–2026.**

This manual has been prepared by **Mr. B NAGENDRA REDDY(Assistant Professor),** Department of Computer Science Engineering (AI&ML), with my own efforts and to the best of our knowledge.

Signature of Lab Faculty **Signature of HOD**

PREFACE

This course on Node JS / React JS / Django is designed to equip students with the essential skills required for modern web application development. It covers both front-end and back-end technologies, starting from the basics of HTML, CSS, and JavaScript to advanced concepts like ES6 features, responsive design, and client-side validation. The course emphasizes hands-on learning through practical exercises, enabling students to build dynamic and interactive web interfaces.

Further, the course introduces server-side programming using Java, Node.js, and Express, along with database connectivity and CRUD operations. It also focuses on important concepts such as session management, authentication, and RESTful services. Additionally, students gain experience in developing single-page applications using React, integrating APIs, and deploying real-time applications, thereby preparing them for industry-level full-stack development.

.

By,

Mr. B.NAGENDRA REDDY

ACKNOWLEDGEMENT

It was really a good experience, working at Node JS / React JS / Django Lab. First, I would like to thank Mr. B. NAGENDRA REDDY , Assistant Professor, Department of Computer Science & Engineering (AI&ML), Marri Laxman Reddy Institute of technology & Management for giving the technical support in preparing the document.

I express my sincere thanks to Dr. B Ravi Prasad, Head of the Department of Computer Science & Engineering (AI&ML), Marri Laxman Reddy Institute of technology & Management, for his concern towards me and gave me opportunity to prepare SOFT COMPUTING laboratory manual.

I am deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr. B Ravi Prasad, Dean Academics, Marri Laxman Reddy Institute of technology & Management. I am unboundedly grateful to him for timely corrections and scholarly guidance.

I express my heartful thanks to Dr. P. Sridhar, Director, and Dr. R. Murali Prasad, Principal, Marri Laxman Reddy Institute of technology & Management, for giving me this wonderful opportunity for preparing the Node JS / React JS / Django laboratory manual.

At last, but not the least I would like to thank the entire Computer Science & Engineering Department faculties those who had inspired and helped me to achieve my goal.

By,

Mr. B NAGENDRA REDDY

Department of Computer Science & Engineering (AI&ML)

GENERAL INSTRUCTIONS

1. Students are instructed to come to Node JS / React JS / Django laboratory on time.
2. Late comers are not entertained in the lab.
3. Students should be punctual to the lab. If not, the conducted experiments will not be repeated.
4. Students are expected to come prepared at home with the experiments which are going to be performed.
5. Students are instructed to display their identity cards before entering into the lab.
6. Students are instructed not to bring mobile phones to the lab.
7. Any damage/loss of system parts like keyboard, mouse during the lab session, it is student's responsibility and penalty or fine will be collected from the student.
8. Students should update the records and lab observation books session wise. Before leaving the lab the student should get his/her lab observation book signed by the faculty.
9. Students should submit the lab records by the next lab to the concerned faculty members in the staff room for their correction and return.
10. Students should not move around the lab during the lab session.
11. If any emergency arises, the student should take the permission from faculty member concerned in written format.
12. The faculty members may suspend any student from the lab session on disciplinary grounds.
13. Never copy the output from other students. Write down your own outputs.

Department of Computer Science & Engineering (AI&ML)

SAFETY MEASURES

To ensure the safe and efficient use of the Computer Science and Engineering(AI&ML) laboratory, all students must strictly adhere to the following safety guidelines:

1. General Conduct

- Maintain silence and discipline during lab sessions.
- Do not bring food, drinks, or chewing gum into the lab.
- Use lab resources responsibly and follow all instructions provided by the instructor or lab assistant.

2. Electrical Safety

- Do not touch electrical switches, sockets, or plugs with wet hands.
- Avoid overloading power sockets with unauthorized devices.
- Immediately report any loose connections, sparks, or unusual noises from equipment.

3. Computer and Equipment Handling

- Handle all computer systems, keyboards, mice, and peripherals with care.
- Do not attempt to open or tamper with any hardware components.
- Use only the assigned computer system; do not switch systems without permission.

4. Software and Data Safety

- Use only authorized software installed by the lab administrator.
- Do not attempt to install, uninstall, or modify any software without approval.
- Save your work frequently and ensure backups of important files.

5. Cybersecurity and Network Usage

- Keep your login credentials confidential.
- Do not attempt to access restricted websites or server
- Avoid activities such as hacking, gaming, or the use of pirated content.

6. Emergency Preparedness

- Be familiar with the location of emergency exits, fire extinguishers, and first aid kits.
- In the event of a fire, electrical hazard, or anyemergency, remain calm and inform the lab instructor immediately.
- Follow the evacuation procedure as instructed.

7. Post-Lab Procedures

- Log out of your session and shut down the system properly after use.
- Leave your workstation clean and organized.
- Return any borrowed materials or equipment to their proper place.

8. Hygiene and Cleanliness

- Wash or sanitize your hands before and after using shared devices.
- Do not write or place unnecessary items on the workstation.
- Report any spills or cleanliness issues to the lab staff.

Department of Computer Science & Engineering (AI&ML)

VISION & MISSION OF THE INSTITUTE

Vision of the Institute:

To be a globally recognized institution that fosters innovation, excellence, and leadership in education, research, and technology development, empowering students to create sustainable solutions for the advancement of society.

Mission of the Institute:

- To foster a transformative learning environment that empowers students to excel in engineering, innovation, and leadership.
- To produce skilled, ethical, and socially responsible engineers who contribute to sustainable technological advancements and address global challenges.
- To shape future leaders through cutting-edge research, industry collaboration, and community engagement.

VISION & MISSION OF THE DEPARTMENT

Department Vision:

To nurture globally competent professionals in Artificial Intelligence and Machine Learning through excellence in education, research, and innovation, committed to developing sustainable and impactful solutions for the betterment of society.

Department Mission:

- To provide a transformative learning environment that equips students with in-depth knowledge and practical skills in Artificial Intelligence and Machine Learning, fostering innovation, leadership, and lifelong learning.
- To advance AI and ML through cutting-edge research, strong industry collaboration, and community engagement, preparing students to address real-world challenges on a global scale.
- To produce competent and ethical AI professionals who contribute to technological progress while addressing societal and environmental challenges with sustainable solutions.
- To foster a research-driven culture by partnering with industry and academia, encouraging entrepreneurship, and engaging in community-centered technology development.

Program Educational Objectives (PEOs)

PEO:

Professional Competence:

Graduates will possess strong theoretical and practical knowledge in Artificial Intelligence and Machine Learning, enabling them to solve complex real-world problems, pursue higher education, or excel in professional careers.

Innovation and Research Orientation:

Graduates will engage in innovative practices, cutting-edge research, and contribute to the advancement of AI and ML technologies through collaboration with industry and academia.

Leadership and Lifelong Learning:

Graduates will exhibit leadership qualities, effective communication, and teamwork skills, and will continuously upgrade their knowledge to adapt to evolving technological landscapes.

Entrepreneurial and Community Engagement:

Graduates will leverage entrepreneurial skills and a sense of civic responsibility to create AI-driven solutions that benefit local and global communities.

Course Outcomes (COs)

1. Design responsive web applications using HTML5, CSS3, Flexbox, Grid, Bootstrap, and client-side scripting techniques.
2. Apply JavaScript and ES6 features including validation logic, asynchronous programming, API consumption, and graphical data representation.
3. Develop server-side applications using Java, Servlets, Node.js, Express framework, and database connectivity with CRUD functionality.
4. Implement session management, authentication mechanisms, secure RESTful services, and authorization using JWT for web applications.
5. Build single-page applications using React components, routing, services, state management, external APIs, and deployment practices.

Program Outcomes (POs)

PO:

PO1.Engineering Knowledge:
Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. Problem Analysis:
Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3.Design/Development of Solutions:
Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct Investigations of Complex Problems:
Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern Tool Usage:
Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The Engineer and Society:
Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and Sustainability:

Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8.Ethics:

Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and Team Work:

Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication:

Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project Management and Finance:

Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long Learning:

Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Able to identify, analyze and solve the problems related to Artificial Intelligence and Machine Learning by applying the fundamental knowledge of Computer Science and Engineering.

1. Problem Identification and Formulation
2. Application of Computer Science Fundamentals
3. AI/ML Techniques and Tools Proficiency
4. Analytical and Critical Thinking

PSO2: Build innovative tools and techniques to develop project models in the areas related to Deep Learning, Machine learning, Artificial Intelligence.

1. Innovation and Tool Development
2. Implement end-to-end project models using real-world datasets in domains like image processing, NLP, or predictive analytics.
3. Advanced Technical Proficiency
4. Evaluation and Optimization.

PSO 3: Make use of the Artificial Intelligence and Machine Learning knowledge to assess societal, environmental, health, safety issues, Sustainable development goals with professional ethics and can also pursue higher studies, involve in research activities, be employable or entrepreneur.

1. Application of AI/ML for Societal and Environmental Impact
2. Ethical and Responsible AI Practice
3. Lifelong Learning and Research Orientation
4. Employability and Entrepreneurship

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AIML)

B.TECH VI SEMESTER
CSM TIME TABLE

ACADEMIC YEAR: 2025-2026

Period	1	2	3		4	5	6
Time	09:40 to 10:35	10:35 to 11:30	11:30 to 12:25	12:25 to 1:15	1:15 to 02:10	02:10 to 03:05	03:05 to 04:00
MON				LUNCH BREAK			
TUE							
WED							
THU							
FRI							
SAT							

25X0581: 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:

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

Course Outcomes: After Completion of the Course, Students Should be able to:

1. Design responsive web applications using HTML5, CSS3, Flexbox, Grid, Bootstrap, and client-side scripting techniques.
2. Apply JavaScript and ES6 features including validation logic, asynchronous programming, API consumption, and graphical data representation.
3. Develop server-side applications using Java, Servlets, Node.js, Express framework, and database connectivity with CRUD functionality.
4. Implement session management, authentication mechanisms, secure RESTful services, and authorization using JWT for web applications.
5. Build single-page applications using React components, routing, services, state management, external APIs, and deployment practices.

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 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, APress.

Virtual lab details:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NODE JS/ REACT JS/ DJANGO LABORATORY

LAB PLANNER CSM-A BATCH-1

S.No	Experiment	CO	Virtual Lab Availabilty	Date planned	Date Conducted
1	Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.	CO1	NA		
2	Make the above web application responsive web application using Bootstrap framework.	CO1	NA		
3	Use JavaScript for doing client – side validation of the pages implemented in experiment 1 and experiment 2.	CO2	NA		
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.	CO2	NA		
5	Develop a java stand alone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables	CO3	NA		
6	Create an xml for the bookstore. Validate the same using both DTD and XSD.	CO3	NA		
7	Design a controller with servlet that provides the interaction with application developed in experiment 1 and the database created in experiment 5.	CO3	NA		
	MID-I				
8	Maintaining the transactional history of any user is very	CO4	NA		

	important. Explore the various session tracking mechanism (Cookies, HTTP Sessions)				
9	Create a custom server using http module and explore the other modules of Node JS like OS, path, event.	CO4	NA		
10	Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)	CO5	NA		
11	For the above application create authorized end points using JWT (JSON Web Token).	CO5	NA		
12	Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.	CO5	NA		
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.				
MID-II					

CSM-A BATCH-2

LAB PLANNER CSM-A BATCH-1

S.No	Experiment	CO	Virtual Lab Availabilty	Date planned	Date Conducted
1	Build a responsive web application for shopping cart with registration, login, catalog and cart pages using CSS3 features, flex and grid.	CO1	NA		
2	Make the above web application responsive web application using Bootstrap framework.	CO1	NA		
3	Use JavaScript for doing client – side validation of the pages implemented in experiment 1 and experiment 2.	CO2	NA		
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.	CO2	NA		
5	Develop a java stand alone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables	CO3	NA		
6	Create an xml for the bookstore. Validate the same using both DTD and XSD.	CO3	NA		
7	Design a controller with servlet that provides the interaction with application developed in experiment 1 and the database created in experiment 5.	CO3	NA		
	MID-I				
8	Maintaining the transactional history of any user is very important. Explore the various session tracking mechanism (Cookies, HTTP Session) s.	CO4	NA		

9	Create a custom server using http module and explore the other modules of Node JS like OS, path, event.	CO4	NA		
10	Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)	CO5	NA		
11	For the above application create authorized end points using JWT (JSON Web Token).	CO5	NA		
12	Create a react application for the student management system having registration, login, contact, about pages and implement routing to navigate through these pages.	CO5	NA		
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.				
MID-II					

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SOFT COMPUTING LABORATORY

LAB PLANNER

Date planed																																
Date conducted																																
Roll Number	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*		Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*	Exp No	CO	VL*

Note:VL*-Virtual Lab Availabilty

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING SOFT

COMPUTING LABORATORY

RUBRICS USED TO ASSESS LEARNINGS IN LABORATORIES

1. RUBRICS FOR DAY TO DAY EVALUATION

Parameter	Max Marks	Level-1 (Very Poor)	Level-2 (Poor)	Level-3 (Average)	Level-4 (Good)	Level-5 (Excellent)
Observation Book	05	No observations or irrelevant data. (0–1)	Incomplete or incorrect data. (2)	Basic values with some errors. (3)	Mostly correct with good format. (4)	Fully correct, clear, and well-formatted. (5)
Record Writing	05	Not submitted. (0-1)	Submitted but mostly incomplete. (2)	Submitted with some missing/wrong parts. (3)	Submitted with minor issues. (4)	Fully complete, correct algorithm & flowchart. (5)
Result	05	No result or major errors. (0–1)	Result partially obtained. (2)	Acceptable result with limited error. (3)	Near-correct result and reasonable error. (4)	Accurate result. (5)
Viva-Voce	05	Did not answer any questions. (1)	Answered very few questions. (2)	Answered some questions with help. (3)	Answered most questions correctly. (4)	Answered all questions accurately. (5)

2. RUBRICS FOR INTERNAL EVALUATION

Criterion	Max Marks	Level-1 *(Very Poor)*	Level-2 *(Poor)*	Level-3 *(Average)*	Level-4 *(Good)*	Level-5 *(Excellent)*
Design/Tool/Apparatus Selection	2 Marks	Incorrect tool/design and no reasoning. **(0)**	Tool/design selection attempted with unclear logic. **(0.5)**	Satisfactory selection with partial justification. **(1)**	Correct selection and proper analysis with few errors. **(1.5)**	Smart selection with accurate, relevant analysis. **(2)**
Execution (Code/Debug/Run) /Analysis/Method Used	4 Marks	Did not attempt or completely failed to execute. **(0)**	Attempted but unable to proceed or with major errors. **(1)**	Partial execution with some logic/syntax errors. **(2)**	Mostly correct execution with minimal help. **(3)**	Fully correct and independently executed program. **(4)**
Results& Documentation	2 Marks	Incomplete or poorly presented. **(0)**	Basic structure but lacks clarity or formatting. **(0.5)**	Complete but genericor with formatting issues. **(1)**	Well-structured and mostly clear. **(1.5)**	Well-organized, professional, and engaging documentation. **(2)**
Viva-Voce (Understanding of Concepts)	2 Marks	No understandin g; could not answer questions. **(0)**	Answered a few with difficulty. **(0.5)**	Answered half the questions with basic clarity. **(1)**	Good understandin g with confident answers. **(1.5)**	Answered all questions with clarity and depth. **(2)**

3. RUBRICS FOR SEMESTER END EXAMINATIONS

Criterion	Max Marks	Level-1 (Very Poor (*0–2 marks)*	Level-2 (Poor)*(3–4 marks)*	Level-3 (Average)*(5–6 marks)*	Level-4 (Good)*(7–9 marks)*	Level-5 (Excellent)*(10–12 marks)*
Preparedness for the Experiment	12 marks	No clarity or objective or procedure. Unable to explain basics.	Limited idea of the objective/procedure. Needed prompting.	Has basic understanding; minor gaps in concept or preparation.	Well-prepared, with clear understanding of steps and background.	Fully prepared with strong conceptual clarity and confident explanation.
Performance in the Laboratory	12 marks	Unable to perform experiment. Relied entirely on examiner's help.	Performed with multiple errors and constant support.	Performed with some errors; required occasional help.	Performed mostly independently with minimal support.	Performed independently, efficiently, and with precision.
Calculations & Graphs	12 marks	No or incorrect calculations. Graphs missing or irrelevant.	Multiple calculation errors. Graphs/plots inaccurate or poorly labeled.	Calculations partially correct. Graphs present but with some flaws.	Correct calculations and graphs with minor errors.	Accurate calculations and well-labeled graphs with proper interpretation.
Results & Error Analysis	12 marks	No result or invalid result. No error analysis attempted.	Incorrect result with vague or no error discussion.	Acceptable result. Error analysis attempted but limited.	Correct result with sound error discussion.	Accurate result with detailed and relevant error analysis.
Viva-Voce (Subject Knowledge)	12 marks	Unable to answer any questions. No conceptual understanding.	Answered few questions with poor logic.	Answered half of the questions with average understanding.	Answered most questions with clarity and confidence.	Answered all questions with depth, clarity,and reasoning.

EXPERIMENT-1

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

DESCRIPTION:

HTML, CSS, and JavaScript are essential components for creating a functional responsive web application. A condensed example of a shopping cart with registration, login, catalogue, and cart pages is provided.

Project Structure:

1. index.html - Main HTML file containing the structure of the web application.
2. styles.css - CSS file for styling the web pages.
3. script.js - JavaScript file for handling interactions and logic.
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">
    <title>Online Store</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <nav>
        <ul>
            <!-- Navigation links can be added here if needed -->
        </ul>
    </nav>
    <main id="content">
        <!-- Catalog and cart content will be loaded here -->
    </main>
    <script src="script.js"></script>
</body>
</html>
```

Styles.css:

```
body {
    background-color: #333;
    color: #fff;
    padding: 10px;
    text-align: center;
}
nav ul {
```

```
    list-style: none;
    padding: 0;
    display: flex;
    justify-content: center;
}
nav li {
    margin: 0 10px;
}
.product-card {
    /* Add styling for product cards */
}
```

Script.js:

```
// Dummy data for the catalog
const catalog = [
    { id: 1, name: 'Product 1', price: 20 },
    { id: 2, name: 'Product 2', price: 30 },
    { id: 3, name: 'Product 3', price: 25 },
];
// Function to load the catalog
function loadCatalog() {
    const catalogContainer = document.getElementById('content');
    catalogContainer.innerHTML = '<h2>Catalog</h2>';
    catalog.forEach(product => {
        const productCard = document.createElement('div');
        productCard.classList.add('product-card');
        productCard.innerHTML = `
            <h3>${product.name}</h3>
            <p>$${product.price}</p>
            <button onclick="addToCart(${product.id})">Add to Cart</button>
        `;
        catalogContainer.appendChild(productCard);
    });
}
// Function to add a product to the cart
function addToCart(productId) {
    // Implement cart functionality here
    console.log(`Product ${productId} added to cart`);
}
// Initial load
loadCatalog();
```

HTML Output (index.html):

A webpage will be rendered with a header containing a navigation bar and a main content area. Inside the main content area, a section titled "Catalog" will be displayed.

Under the "Catalog" section, product cards will be displayed for each product in the catalog array.
Each product card will contain the product name, price, and an "Add to Cart" button.
CSS Output (styles.css):
The background color of the webpage will be set to dark gray (#333), with white text color.
Navigation links (if added) will be displayed in a horizontal list at the center of the page.
Product cards will have specific styling applied (not defined in the provided code snippet).
JavaScript Output (script.js):
The loadCatalog function will populate the main content area with product cards based on the data provided in the catalog array.
Each product card will contain the product name, price, and an "Add to Cart" button.
Clicking on the "Add to Cart" button for any product will trigger the addToCart function, which will log a message to the console indicating the ID of the product that was added to the cart.

Viva Questions:

S.No	Viva Questions
1	What is a responsive web application and why is it important?
2	Explain the role of HTML, CSS, and JavaScript in this shopping cart project.
3	What is the purpose of the <meta name="viewport"> tag?
4	How do you link external CSS and JavaScript files to an HTML page?
5	What is DOM manipulation? Give an example from your project.
6	What is an array in JavaScript? How is it used in the catalog?
7	What is the function of loadCatalog() in your application?
8	What happens when the "Add to Cart" button is clicked?
9	What is the difference between innerHTML and createElement()?
10	What is an event in JavaScript? Give an example from this project.
11	What is Flexbox? Where did you use it in your code?
12	What is CSS Grid? How is it different from Flexbox?
13	How do you center elements using CSS?
14	What is the difference between class and id selectors in CSS?
15	What is the purpose of display: flex in navigation styling?
16	How can you make your webpage mobile-friendly?
17	What is the use of functions in JavaScript?
18	What are template literals (` `) in JavaScript?
19	How can you improve the current project to make it a complete shopping cart system?
20	What are the limitations of the current implementation of this project?

EXPERIMENT: 2

Aim: Make the above web application responsive web application using Bootstrap framework

- Procedure:
  Integrating the Bootstrap framework into your web application is an excellent way to ensure responsiveness and enhance the UI with minimal effort.
- Bootstrap provides a wide range of CSS classes designed for responsive layouts, components, and interactive elements, which can significantly streamline the development process.
- Below, we will go through adapting the previous project to utilize Bootstrap, focusing on the registration page as an example.
- we can apply similar principles to the login, catalog, and cart pages.

Step 1: Include Bootstrap

First, include Bootstrap in your HTML files. You can do this by adding the Bootstrap CDN links in the **<head>** section of your HTML documents. This example uses Bootstrap 5, but you should check for the latest version available.

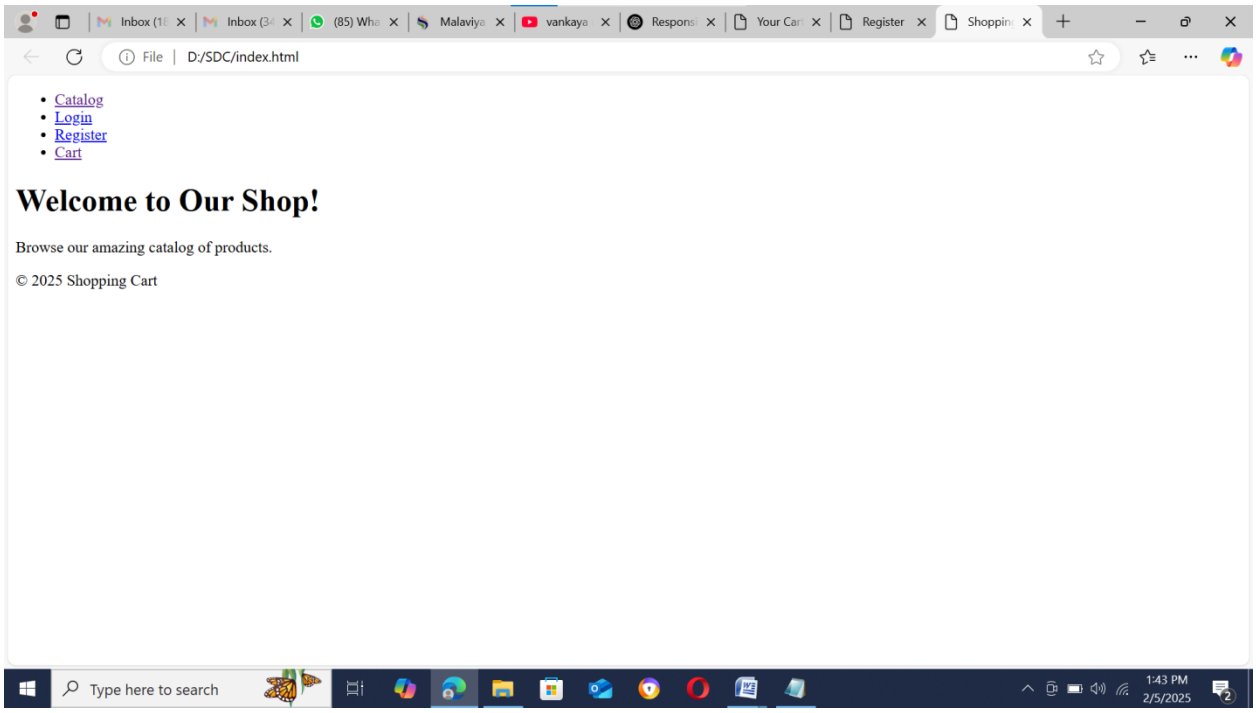
```
<!-- Bootstrap CSS -->
<link href="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/css/bootstrap.min.css" rel="stylesheet">
<!-- Bootstrap Bundle JS (includes Popper) -->
<script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.bundle.min.js"></script>
```

Step2: Html code for register page:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Shopping Cart</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="catalog.html">Catalog</a></li>
        <li><a href="login.html">Login</a></li>
        <li><a href="register.html">Register</a></li>
        <li><a href="cart.html">Cart</a></li>
      </ul>
    </nav>
  </header>

  <main>
```

```
    <h1>Welcome to Our Shop!</h1>
    <p>Browse our amazing catalog of products.</p>
  </main>

  <footer>
    <p>© 2025 Shopping Cart</p>
  </footer>
</body>
</html>
```

Output:

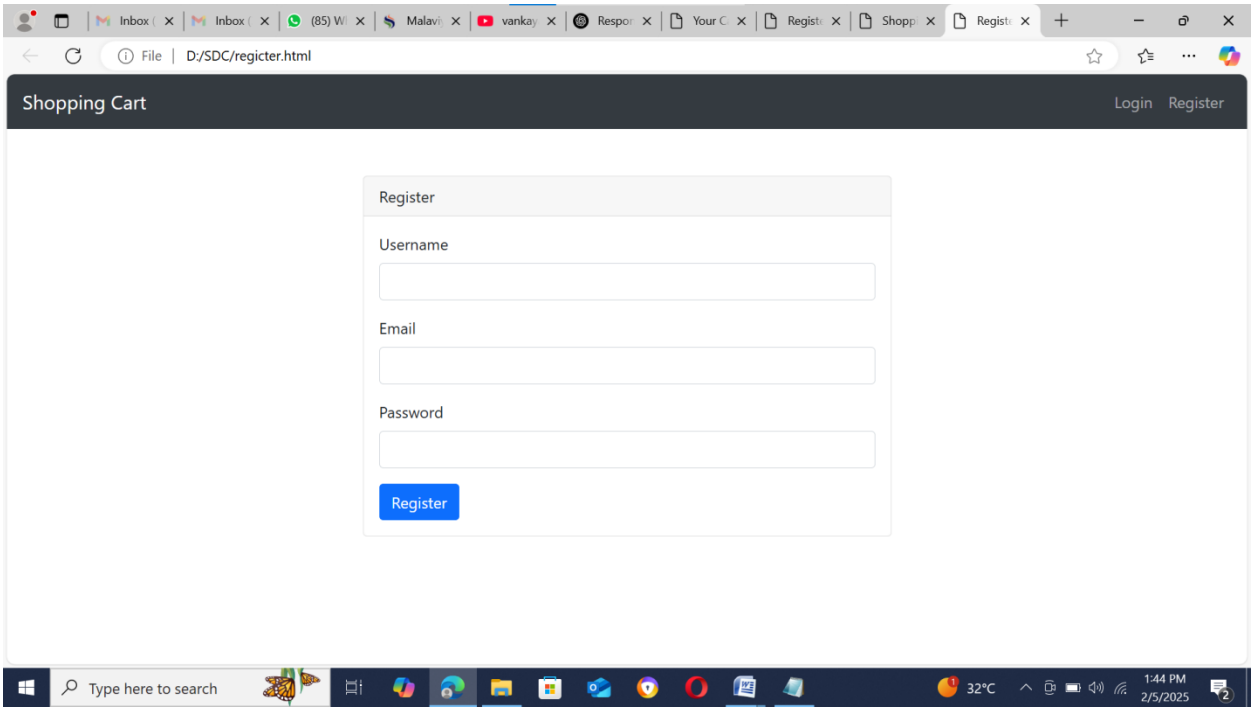


Step 3: Adjust above HTML Structure for Bootstrap

Here's how you can adjust the **register.html** structure to use Bootstrap classes for layout and form elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Shopping Cart</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ml-auto">
```

```
            <li class="nav-item">
              <a class="nav-link" href="login.html">Login</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="register.html">Register</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>

    <div class="container mt-5">
      <div class="row justify-content-center">
        <div class="col-md-6">
          <div class="card">
            <div class="card-header">
              Register
            </div>
            <div class="card-body">
              <form id="registrationForm">
                <div class="mb-3">
                  <label for="username" class="form-label">Username</label>
                  <input type="text" class="form-control" id="username" name="username" required>
                </div>
                <div class="mb-3">
                  <label for="email" class="form-label">Email</label>
                  <input type="email" class="form-control" id="email" name="email" required>
                </div>
                <div class="mb-3">
                  <label for="password" class="form-label">Password</label>
                  <input type="password" class="form-control" id="password" name="password" required>
                </div>
                <button type="submit" class="btn btn-primary">Register</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>

    <!-- Bootstrap Bundle JS -->
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```



Key Points for Bootstrap Integration

- **Navigation Bar:** Use the **navbar** component for navigation. Bootstrap's navbar automatically adjusts for different screen sizes and can be toggled in smaller screens.
- **Forms:** Utilize Bootstrap's form classes (**form-control**, **mb-3** for margin) to style and space form elements.
- **Grid System:** Bootstrap's grid system (using classes like **container**, **row**, and **col-md-6**) helps create responsive layouts. The grid system is based on Flexbox and allows you to easily adjust the layout for different screen sizes.
- **Buttons:** Use Bootstrap's button classes (**btn**, **btn-primary**) to style buttons.

Step 3: Responsiveness

Bootstrap's grid system and components are designed to be responsive. By using the grid classes and responsive utilities, you can ensure that your web application looks good on devices of all sizes without writing custom media queries.

Final Notes

- We can explore more components and classes provided by Bootstrap to enhance other pages of your application (e.g., cards for product listings, modals for product details, alerts for notifications).
- Test your application on various devices and screen sizes to ensure the layout adjusts as expected.
- Bootstrap also supports customizing themes and components if you need a more tailored look for your application.

By following these steps and utilizing Bootstrap's extensive documentation and examples, you can quickly develop a responsive and aesthetically pleasing web application.

S.No	Viva Question
1	What is Bootstrap and why is it used in web development?
2	What are the advantages of using Bootstrap?
3	How do you include Bootstrap in an HTML file?
4	What is a CDN in Bootstrap?
5	What is the Bootstrap grid system?
6	What are container, row, and col classes in Bootstrap?
7	How does Bootstrap help in making a website responsive?

S.No	Viva Question
8	What is the difference between container and container-fluid?
9	What is a navbar in Bootstrap?
10	What is the use of navbar-expand-lg class?
11	What is the purpose of form-control class?
12	What is the use of spacing classes like mb-3 in Bootstrap?
13	What are Bootstrap cards and where are they used?
14	What is the use of btn and btn-primary classes?
15	What is the role of collapse class in navigation bar?
16	What is the difference between Bootstrap and normal CSS?
17	How does Bootstrap handle mobile responsiveness?
18	What are Bootstrap components? Give examples.
19	How can you customize Bootstrap styles?
20	What improvements can be made to this project using Bootstrap components?

EXPERIMENT - 03

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

To add client-side validation using JavaScript to the pages from the previous experiments (both custom CSS and Bootstrap-based), we'll focus on validating user input in forms such as **Login**, **Registration**, and the **Cart** (optional input fields).

Here are the steps for integrating **JavaScript validation** for each form:

1. Login Page Validation

We'll validate that the **email** and **password** fields are filled out correctly and that the email follows the correct format.

HTML (login.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="js/app.js" defer></script> <!-- Link to external JavaScript -->
</head>
<body>

  <header class="bg-dark text-white p-3">
    <div class="container">
      <nav class="nav">
        <a class="nav-link text-white" href="index.html">Home</a>
        <a class="nav-link text-white" href="login.html">Login</a>
        <a class="nav-link text-white" href="register.html">Register</a>
        <a class="nav-link text-white" href="catalog.html">Catalog</a>
        <a class="nav-link text-white" href="cart.html">Cart</a>
      </nav>
    </div>
  </header>

  <div class="container my-5">
    <h1 class="text-center">Login</h1>
    <form id="loginForm" onsubmit="return validateLogin()">
```

```
    <div class="mb-3">
      <label for="email" class="form-label">Email address</label>
      <input type="email" class="form-control" id="email" placeholder="Enter email" required>
      <div class="invalid-feedback" id="emailError">Please enter a valid email.</div>
    </div>

    <div class="mb-3">
      <label for="password" class="form-label">Password</label>
      <input type="password" class="form-control" id="password" placeholder="Enter password" required>
      <div class="invalid-feedback" id="passwordError">Password is required.</div>
    </div>

    <button type="submit" class="btn btn-primary">Login</button>
  </form>
 </div>

 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

JavaScript (app.js)

```
function validateLogin() {
 // Clear any previous error messages
 document.getElementById('emailError').style.display = 'none';
 document.getElementById('passwordError').style.display = 'none';

 const email = document.getElementById('email').value;
 const password = document.getElementById('password').value;

 let valid = true;

 // Validate email
 const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
 if (!email || !email.match(emailPattern)) {
   document.getElementById('emailError').style.display = 'block';
   valid = false;
 }

 // Validate password
 if (!password) {
   document.getElementById('passwordError').style.display = 'block';
   valid = false;
 }
 return valid; // Prevent form submission if validation fails
}
```

2. Registration Page Validation

In the **Registration page**, we'll validate the **name**, **email**, **password**, and **confirm password** fields. We'll ensure the email is valid, and the password and confirm password match.

HTML (register.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="js/app.js" defer></script>
</head>
<body>

  <header class="bg-dark text-white p-3">
    <div class="container">
      <nav class="nav">
        <a class="nav-link text-white" href="index.html">Home</a>
        <a class="nav-link text-white" href="login.html">Login</a>
        <a class="nav-link text-white" href="register.html">Register</a>
        <a class="nav-link text-white" href="catalog.html">Catalog</a>
        <a class="nav-link text-white" href="cart.html">Cart</a>
      </nav>
    </div>
  </header>
  <div class="container my-5">
    <h1 class="text-center">Register</h1>
    <form id="registerForm" onsubmit="return validateRegister()">
      <div class="mb-3">
        <label for="name" class="form-label">Full Name</label>
        <input type="text" class="form-control" id="name" placeholder="Enter your name" required>
        <div class="invalid-feedback" id="nameError">Name is required.</div>
      </div>
      <div class="mb-3">
        <label for="email" class="form-label">Email address</label>
        <input type="email" class="form-control" id="email" placeholder="Enter email" required>
        <div class="invalid-feedback" id="emailError">Please enter a valid email.</div>
      </div>

      <div class="mb-3">
        <label for="password" class="form-label">Password</label>
        <input type="password" class="form-control" id="password" placeholder="Enter password" required>
        <div class="invalid-feedback" id="passwordError">Password is required.</div>
      </div>

      <div class="mb-3">
        <label for="confirmPassword" class="form-label">Confirm Password</label>
        <input type="password" class="form-control" id="confirmPassword" placeholder="Confirm password"
required>
        <div class="invalid-feedback" id="confirmPasswordError">Passwords must match.</div>
      </div>

      <button type="submit" class="btn btn-primary">Register</button>
    </form>
  </div>
```

```
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

JavaScript (app.js):

```
function validateRegister() {
  // Clear any previous error messages
  document.getElementById('nameError').style.display = 'none';
  document.getElementById('emailError').style.display = 'none';
  document.getElementById('passwordError').style.display = 'none';
  document.getElementById('confirmPasswordError').style.display = 'none';

  const name = document.getElementById('name').value;
  const email = document.getElementById('email').value;
  const password = document.getElementById('password').value;
  const confirmPassword = document.getElementById('confirmPassword').value;

  let valid = true;

  // Validate name
  if (!name) {
    document.getElementById('nameError').style.display = 'block';
    valid = false;
  }

  // Validate email
  const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
  if (!email || !email.match(emailPattern)) {
    document.getElementById('emailError').style.display = 'block';
    valid = false;
  }

  // Validate password
  if (!password) {
    document.getElementById('passwordError').style.display = 'block';
    valid = false;
  }

  // Validate confirm password
  if (password !== confirmPassword) {
    document.getElementById('confirmPasswordError').style.display = 'block';
    valid = false;
  }

  return valid; // Prevent form submission if validation fails
}
```

3. Cart Page Validation (Optional)

If the user can change the quantity of items in the cart, you may want to ensure the quantity is a positive integer. This example validates the quantity field.

HTML (cart.html)

```
<!-- Add quantity validation to the cart -->
<div class="cart-item d-flex justify-content-between align-items-center mb-3 p-3 border">
  <img src="assets/images/product1.jpg" alt="Product 1" class="img-fluid" style="width: 50px; height: 50px;">
  <span>Product 1</span>
  <span>$25.99</span>
  <input type="number" value="1" min="1" max="99" class="form-control w-25" id="quantity1"
oninput="validateQuantity('quantity1')">
  <button class="btn btn-danger">Remove</button>
</div>
```

JavaScript (app.js)

```
// Validate cart quantity input
function validateQuantity(id) {
  const quantity = document.getElementById(id).value;
  if (quantity <= 0) {
    document.getElementById(id).setCustomValidity("Quantity must be a positive integer.");
  } else {
    document.getElementById(id).setCustomValidity(""); // Reset the custom validity
  }
}
```

- **Login Form**: Validates email format and checks if the password is entered.
- **Registration Form**: Validates name, email, password, and confirm password.
- **Cart Page**: Ensures quantities are valid positive integers.

S.No	Viva Question
1	What is client-side validation in web applications?
2	What are the advantages of client-side validation?
3	What is the difference between client-side and server-side validation?
4	What is the purpose of the onsubmit event in forms?
5	What does the validateLogin() function do?
6	What is a regular expression? How is it used in email validation?
7	Explain the email validation pattern used in the code.
8	What is the purpose of required attribute in input fields?
9	What does document.getElementById() do?
10	What is the role of innerHTML and value properties?
11	What is the use of style.display = 'none' and 'block'?
12	What is the purpose of returning true or false in validation functions?
13	How does JavaScript prevent form submission when validation fails?
14	What is the purpose of confirm password validation?
15	What is setCustomValidity() in JavaScript?
16	How is quantity validation implemented in the cart page?
17	What is the use of oninput event in input fields?
18	What are error messages and how are they displayed in this project?
19	What are common validation checks in forms?
20	How can you improve this validation system further?

EXPERIMENT - 04

Aim:Explore the features of ES6 like arrow functions, callbacks, promises, a sync /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:

Step 1: Features of ES6

1. Arrow Functions:

Arrow functions provide a concise way to write functions and maintain the context of this within methods. They are often used for callbacks.

Example:

const greet = (name) => `Hello, ${name}`;

2. Callbacks:

A callback function is passed into another function as an argument and executed later. ES6 encourages using arrow functions as callbacks due to their concise syntax.

Example:

```
setTimeout(() => {  console.log("This is a callback!");

}, 2000);
```

3. Promises:

Promises allow you to handle asynchronous operations more effectively and provide a cleaner approach to handle asynchronous data.

Example:

```
const fetchData = new Promise((resolve, reject) => {

  const success = true;

  if (success) {

    resolve("Data fetched successfully");
```

```
  } else {
    reject("Error fetching data");
  }
});
```

4. Async/Await:

async and await are introduced in ES6 to make working with asynchronous code cleaner and more readable. The async keyword is used before a function to indicate that it returns a promise, and await is used to wait for the promise to resolve.

Example:

```
async function fetchData() {
  let response = await fetch("https://api.example.com/data");
  let data = await response.json();
  return data;
}

fetchData.then(result => console.log(result)).catch(error => console.error(error));
```

Step 2: Weather Application

Objective:

- Use OpenWeatherMap API to fetch weather data.
- Display the temperature and humidity on a graph.
- Use ES6 features like Promises and Async/Await for fetching data.
- Utilize charting library like Chart.js for displaying the graph.

Prerequisites:

- Sign up at OpenWeatherMap to get an API key.
- Basic knowledge of HTML, CSS, and JavaScript.
- Use a charting library like Chart.js for displaying data graphically.

Steps:

1. **HTML Setup:** Create a simple HTML structure to display the weather and graph.
2.

1.Index.html:

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather Application</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h1>Weather Data</h1>
    <div>
        <label for="city">Enter City: </label>
        <input type="text" id="city" placeholder="City name" />
        <button onclick="getWeather()">Get Weather</button>
    </div>
    <div>
        <canvas id="weatherChart"></canvas>
    </div>

    <!-- Place this at the bottom to ensure the DOM is fully loaded before the script runs -->
    <script src="app.js"></script>
</body>
</html>
```

2. JavaScript (app.js):

In this script, we'll create an async function to fetch weather data from the OpenWeatherMap API and display it on the chart.

```
const apiKey = ' YOUR_OPENWEATHERMAP_API_KEY';  // Replace with your API key
const weatherChartCtx = document.getElementById('weatherChart').getContext('2d');
```

```
async function getWeather() {
    const city = document.getElementById('city').value;
    if (!city) {
        alert('Please enter a city');
        return;
    }
    try {
        const weatherData = await fetchWeatherData(city);
        if (weatherData) {
            const { temp, humidity } = weatherData.main;
            const { name, country } = weatherData;
            console.log(`Temperature in ${name}, ${country}: ${temp}°C`);
            console.log(`Humidity: ${humidity}%`);
            // Call function to update graph
            updateGraph([temp, humidity], [name]);
        } else {
            alert('City not found or error fetching data');
        }
    } catch (error) {
        console.error('Error fetching weather data:', error);
    }
}
async function fetchWeatherData(city) {
    const url = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}`;
    const response = await fetch(url);
    const data = await response.json();
    if (data.cod === 200) {
        return data;
```

```
    } else {
        return null;
    }
}
function updateGraph(dataValues, labels) {
    // Check if the weatherChart already exists, if so destroy it
    if (window.weatherChart && window.weatherChart.destroy) {
        window.weatherChart.destroy();
    }
    // Create a new chart
    window.weatherChart = new Chart(weatherChartCtx, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: 'Weather Data (Temperature, Humidity)',
                data: dataValues,
                backgroundColor: ['#FF5733', '#33CFFF'],
                borderColor: ['#FF5733', '#33CFFF'],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true
                }            }        }    });}
```

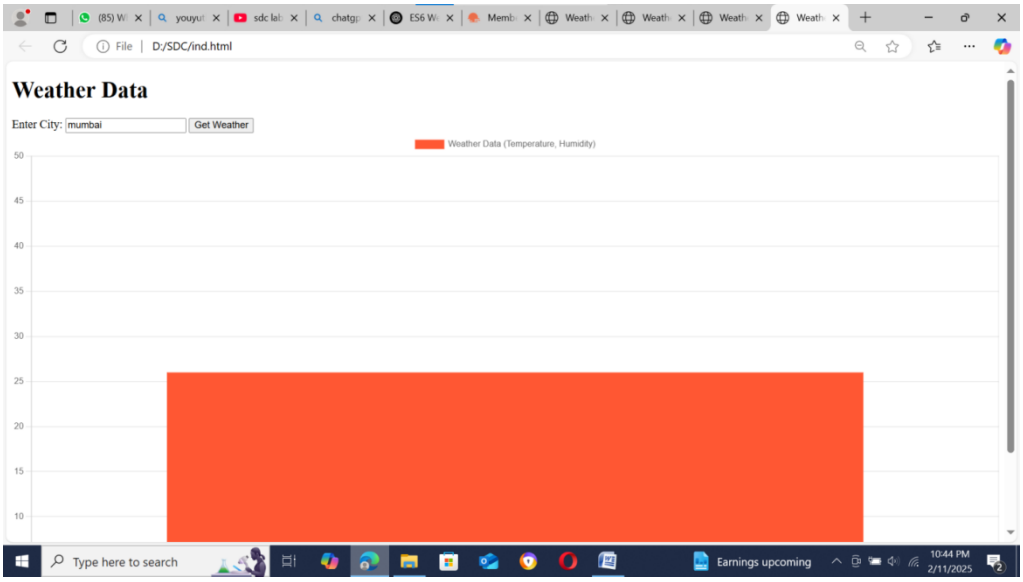
Steps to Create an OpenWeatherMap Account and Generate API Key:

1. 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.
2. Once logged in, navigate to your account dashboard.
3. From the dashboard, locate my API Keys section and click on **"Create Key"** or **"API Keys"** to generate a new API key.
4. Provide a name for your API key (e.g., "**WeatherApp**") and click on the **"Generate"** or **"Create"** button.
5. Your API key will be generated and displayed on the screen. Make sure to copy it as we will need it later.

Step 3: Running the Application

1. Make sure you have replaced 'YOUR_OPENWEATHERMAP_API_KEY' with your actual API key from OpenWeatherMap.
2. Open the HTML file in your browser.
3. Enter a city name, click "Get Weather", and the weather data (temperature and humidity) will be displayed on the graph.

Output:



S.No	Viva Question
1	What are the main features of ES6?
2	What is an arrow function? How is it different from a normal function?
3	What are callbacks in JavaScript?
4	What is a Promise? Why is it used?
5	What are the states of a Promise?
6	What is async/await in JavaScript?
7	What is the difference between async/await and Promises?
8	What is the purpose of the fetch() function?
9	What is an API? Give an example used in this project.
10	What is OpenWeatherMap API?
11	What is an API key and why is it required?
12	What does the getWeather() function do?
13	What is the purpose of await keyword in API calls?
14	What is JSON and how is it used in this project?
15	What is destructuring in JavaScript? Give example from code.
16	What is Chart.js and why is it used?
17	What type of chart is used in this application?
18	What is the purpose of updateGraph() function?
19	How do you handle errors in async functions?
20	What improvements can be made to this weather application?

Experiment 5

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

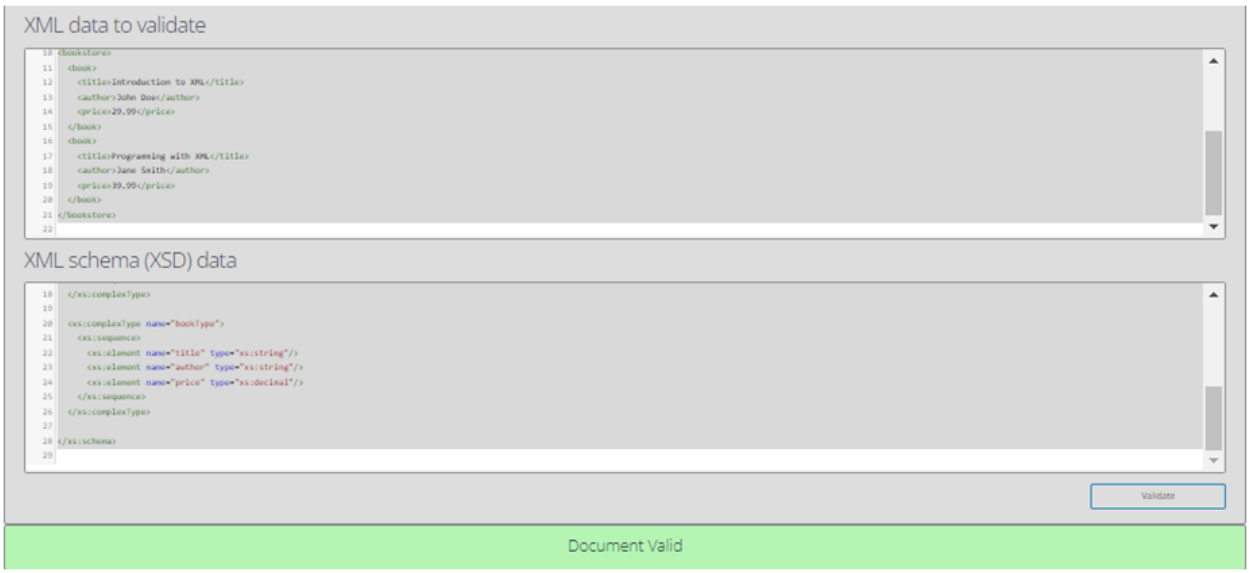
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://example.com"
        xmlns="http://example.com">
<xs:element name="root">
<xs:complexType>
<xs:sequence>
<xs:element name="bookstore" type="bookstoreType"/>
</xs:sequence>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

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 :



2)To validate an XML document with a **DTD (Document Type Definition)**, the XML file must either **include a reference** to an external DTD or define the DTD within the XML file itself. The process involves ensuring that the XML file follows the rules and structure defined by the DTD.

Steps to Validate XML with DTD:

1. **Reference the DTD in the XML document** (either inline or external).
2. **Use an XML parser or validator** that supports DTD validation to check the XML document against the DTD.

1. Inline DTD (DTD defined within the XML file)

You can define the DTD directly within the XML document, usually inside the <!DOCTYPE ... > declaration at the top of the file.

Example of an XML document with an inline DTD:

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore [
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, author, price, year, genre)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>

```
  <!ELEMENT year (#PCDATA)>
  <!ELEMENT genre (#PCDATA)>
]>
<bookstore>
   <book>
      <title>Introduction to XML</title>
      <author>John Doe</author>
      <price>29.99</price>
      <year>2021</year>
      <genre>Technology</genre>
   </book>
   <book>
      <title>Learning Python</title>
      <author>Jane Smith</author>
      <price>39.99</price>
      <year>2022</year>
      <genre>Programming</genre>
   </book>

</bookstore>
```

In this example:

- The **DTD** is included directly within the <!DOCTYPE> declaration.
- The bookstore element is defined to contain one or more book elements, and each book element contains the elements title, author, price, year, and genre.

2. External DTD (DTD in a separate file)

You can also reference an external DTD file, which is useful for larger documents or when reusing the same DTD across multiple XML documents.

Example of an XML document with an external DTD reference:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore SYSTEM "bookstore.dtd">
<bookstore>
   <book>
      <title>Introduction to XML</title>
      <author>John Doe</author>
      <price>29.99</price>
      <year>2021</year>
      <genre>Technology</genre>
   </book>
   <book>
      <title>Learning Python</title>
      <author>Jane Smith</author>
      <price>39.99</price>
      <year>2022</year>
      <genre>Programming</genre>
   </book>
```

</bookstore>

In this case:

- The bookstore.dtd file contains the DTD definitions, and the XML file refers to it using <!DOCTYPE bookstore SYSTEM "bookstore.dtd">.

Bookstore.dtd:

```
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, author, price, year, genre)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT genre (#PCDATA)>
```

3. Validate the XML Document
Once the XML document includes a reference to a DTD, you can validate it using different tools or methods:
a. Using an XML Validator Online:
There are many online XML validators where you can upload both your XML and DTD files, and it will validate whether the XML conforms to the DTD.

S.No	Question
1	What is XML?
2	What is the purpose of XML in data representation?
3	What is a well-formed XML document?
4	What is a valid XML document?
5	What is a DTD (Document Type Definition)?
6	What is an XML Schema (XSD)?
7	What is the difference between well-formed and valid XML?
8	What is the role of <!DOCTYPE> in XML?
9	What are the different types of DTD?
10	What is the difference between internal and external DTD?
11	What does #PCDATA mean in DTD?
12	What is the significance of + in (book+)?
13	What is a complex type in XSD?
14	What is the difference between simple type and complex type in XSD?
15	What are the advantages of XSD over DTD?
16	Why is XSD considered more powerful than DTD?
17	What is the role of namespaces in XSD?
18	What is the purpose of minOccurs and maxOccurs in XSD?
19	How does XML validation work using DTD and XSD?
20	What will happen if XML does not follow the structure defined in DTD/XSD?

Experiment 6

Aim: Develop a java stand alone application that connects with the database (Oracle / mySql) and perform the CRUD operation on the database tables.

Solution :

☛ **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_addre
ss"));
                    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

_____
```

Viva Questions

S.No	Question
1	What is JDBC?
2	What is the purpose of JDBC?
3	What are the steps to connect Java with a database?
4	What is a Connection in JDBC?
5	What is a Statement in JDBC?
6	What is the difference between Statement and PreparedStatement?
7	What is ResultSet?
8	What is the use of executeUpdate() method?
9	What is the use of executeQuery() method?
10	What are CRUD operations?
11	Which SQL command is used for inserting data?
12	Which SQL command is used for updating data?
13	Which SQL command is used for deleting data?
14	What is a primary key?
15	Why is s_id declared as PRIMARY KEY?
16	What happens if duplicate primary key is inserted?
17	What is SQL Injection?
18	How can SQL Injection be prevented?
19	What is the role of MySQL driver in JDBC?
20	What is exception handling in JDBC?

EXPERIMENT - 07

Aim:Design a controller using Servlet that provides interaction between the application (HTML form) and the database (MySQL) to perform CRUD operations.

Requirements:

Java (JDK)

Apache Tomcat Server

MySQL Database

Database Table:

```
CREATE TABLE student (
   s_id INT PRIMARY KEY,
   s_name VARCHAR(255),
   s_address VARCHAR(255)
);
```

HTML Form (index.html):

```
<!DOCTYPE html>
<html>
<head>
<title>Student CRUD</title>
</head>
<body>
<h2>Student CRUD Operations</h2>
<form action="StudentServlet" method="post">
ID: <input type="text" name="sid"><br><br>
Name: <input type="text" name="sname"><br><br>
Address: <input type="text" name="saddr"><br><br>
<input type="submit" name="action" value="Insert">
<input type="submit" name="action" value="Update">
<input type="submit" name="action" value="Delete">
<input type="submit" name="action" value="Display">
</form>
</body>
</html>
```

Servlet Controller (StudentServlet.java):

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class StudentServlet extends HttpServlet {

Connection con;

public void init() {
try {
Class.forName("com.mysql.cj.jdbc.Driver");
```

```
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "");
} catch (Exception e) {
System.out.println(e);
}
}

protected void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {

res.setContentType("text/html");
PrintWriter out = res.getWriter();

int sid = Integer.parseInt(req.getParameter("sid"));
String name = req.getParameter("sname");
String addr = req.getParameter("saddr");
String action = req.getParameter("action");

try {
Statement st = con.createStatement();

if (action.equals("Insert")) {
st.executeUpdate("INSERT INTO student VALUES(" + sid + ",'" + name + "','" + addr + "')");
out.println("Record Inserted Successfully");
}
else if (action.equals("Update")) {
st.executeUpdate("UPDATE student SET s_name='" + name + "', s_address='" + addr + "' WHERE s_id=" + sid);
out.println("Record Updated Successfully");
}
else if (action.equals("Delete")) {
st.executeUpdate("DELETE FROM student WHERE s_id=" + sid);
out.println("Record Deleted Successfully");
}
else if (action.equals("Display")) {
ResultSet rs = st.executeQuery("SELECT * FROM student");
while (rs.next()) {
out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
}
}
} catch (Exception e) {
out.println("Error: " + e);
}
}
}
```

web.xml:

```
<web-app>
<servlet>
<servlet-name>StudentServlet</servlet-name>
<servlet-class>StudentServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>StudentServlet</servlet-name>
<url-pattern>/StudentServlet</url-pattern>
</servlet-mapping>
```

</web-app>
Output:
Insert: Record Inserted Successfully
Update: Record Updated Successfully
Delete: Record Deleted Successfully
Display: Records shown from database.
Result:
Successfully developed a Java Servlet controller to perform CRUD operations.

S.No	Question
1	What is a Servlet?
2	What is the role of a Servlet in web applications?
3	What is the difference between Servlet and JSP?
4	What is a web container?
5	What is Apache Tomcat?
6	What is the life cycle of a Servlet?
7	What is the purpose of init() method?
8	What is the purpose of doPost() method?
9	What is HttpServletRequest?
10	What is HttpServletResponse?
11	How do you read form data in a Servlet?
12	What is the use of PrintWriter in Servlet?
13	How does Servlet connect to database?
14	What is JDBC?
15	What is the role of DriverManager?
16	What is the difference between GET and POST methods?
17	What is web.xml?
18	What is servlet mapping?
19	How are CRUD operations implemented in Servlet?
20	What happens if database connection fails in Servlet?

EXPERIMENT-8

Aim:Maintaining the transactional history of any user is very important. Explore the various session tracking mechanism (Cookies, HTTP Session)

1. Cookies:

Cookies are small pieces of data sent by the server to the client's browser and are stored on the client's machine. They are then sent back to the server with subsequent requests. Cookies can be used for session tracking by storing a unique session identifier on the client's machine.

Example:

Suppose we have a simple Node.js web server that serves a basic HTML page and sets a cookie to track the user's session.

cookie-example.js:

```
const http = require('http');
// Create a simple HTTP server
const server = http.createServer((req, res) => {
    // Set a cookie with a unique session ID
    res.setHeader('Set-Cookie', 'sessionID=123456789; HttpOnly; Path=/');
    // Serve a basic HTML page
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<html><head><title>Session Tracking with Cookies</title></head>');
    res.write('<body><h1>Hello, Cookies!</h1></body></html>');
    res.end();
});
// Start the server on port 3000
server.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

In this example:

When the server receives a request, it sets a cookie named sessionID with a value of 123456789.

The HttpOnly attribute ensures that the cookie is only accessible through HTTP requests and not by client-side scripts.

The Path=/ attribute specifies that the cookie should be sent with all requests to the root path of the server.

1. Cookies Example:

Type the first code provided for setting up a simple Node.js HTTP server and serving a basic HTML page while setting a cookie.

Save the code in a file named cookie-example.js.
Install Node.js if you haven't already done so. You can download and install it from the official website: Node.js Downloads.
Open your terminal or command prompt.
Navigate to the directory where you saved cookie-example.js.
Run the following command to execute the script:
D:>node cookie-example.js
Once the server starts running, open your web browser and go to http://localhost:3000/. You should see the message "Hello, Cookies!" displayed in the browser, and a cookie named sessionID should be set.

2. HTTP Session:
HTTP sessions are managed entirely on the server-side. The server generates a unique session identifier for each client and stores session data associated with that identifier.

Example:
Let's use Express.js, a popular Node.js framework, to demonstrate HTTP session tracking.
session-example.js:

```
const express = require('express');
const session = require('express-session');

// Create an Express application
const app = express();

// Configure session middleware
app.use(session({
    secret: 'my-secret-key', // Secret key for session encryption
    resave: false,
    saveUninitialized: true
}));
// Route to set and retrieve session data
app.get('/', (req, res) => {
    // Set session data
    req.session.username = 'John';
    // Retrieve session data
    const username = req.session.username;
    res.send(`Hello, ${username}!`);
});
// Start the server on port 3000
app.listen(3000, () => {
```

```
    console.log('Server running at http://localhost:3000/');
});
```

In this example:

We use the Express(a node.js Frame work) along with express-session (middleware to manage sessions) in our Express application.

The middleware generates a unique session identifier and stores it in a cookie named connect.sid by default.

Session data can be accessed and modified through the req.session object.

Here, we set the username property in the session and retrieve it to personalize the response.

2. HTTP Session Example:

Type the second example provided for setting up an Express.js application with session middleware.

Save the code in a file named session-example.js.

Before running the HTTP session example, you need to install Express.js and express-session. You can do this by running the following command in the terminal:

Cmd: npm install express express-session

After installing the dependencies, navigate to the directory where you saved session-example.js.

Run the following command to execute the script:

Cmd : node session-example.js

Once the server starts running, open your web browser and go to http://localhost:3000/. You should see a personalized message displayed in the browser, indicating that the session data is being stored and retrieved successfully.

Both cookies and HTTP sessions are widely used for session tracking in web applications, each with its own advantages and considerations. The choice between them depends on factors such as security requirements, ease of implementation, and the nature of the application.

S.No	Question
1	What is session tracking?
2	Why is session tracking required in web applications?
3	What are cookies?
4	Where are cookies stored?
5	What is the purpose of cookies in session tracking?
6	What is meant by HttpOnly cookie?
7	What is the use of Path attribute in cookies?
8	What is the difference between session cookies and persistent cookies?
9	What are HTTP sessions?
10	How are HTTP sessions maintained?

S.No	Question
11	What is a session ID?
12	Where is session data stored in HTTP sessions?
13	What is Express.js?
14	What is express-session middleware?
15	What is the role of the secret key in session management?
16	What is req.session object?
17	What is the default cookie name used in express-session?
18	Difference between cookies and HTTP sessions?
19	Which is more secure: cookies or sessions? Why?
20	What happens if session examples?

EXPERIMENT-9

Aim: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:

```
// Importing required modules
const http = require('http');
const os = require('os');
const path = require('path');
const EventEmitter = require('events');
// Creating an instance of EventEmitter
const eventEmitter = new EventEmitter();
// Define a custom event listener
eventEmitter.on('customEvent', (data) => {
  console.log('Custom event occurred:', data);
});
// Define a custom event emitter function
function emitCustomEvent() {
  eventEmitter.emit('customEvent', { message: 'Hello, custom event!' });
}
// Creating a simple HTTP server
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello, World!\n');
});
// Fetching system information using the os module
const systemInfo = {
  platform: os.platform(),
  arch: os.arch(),
  totalMemory: os.totalmem(),
  freeMemory: os.freemem(),
};
// Starting the HTTP server
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
  console.log('System Information:', systemInfo);
  console.log('Path Module Example:', path.join(__dirname, 'example.txt'));
  // Emitting custom event after server starts
  emitCustomEvent();
});
```

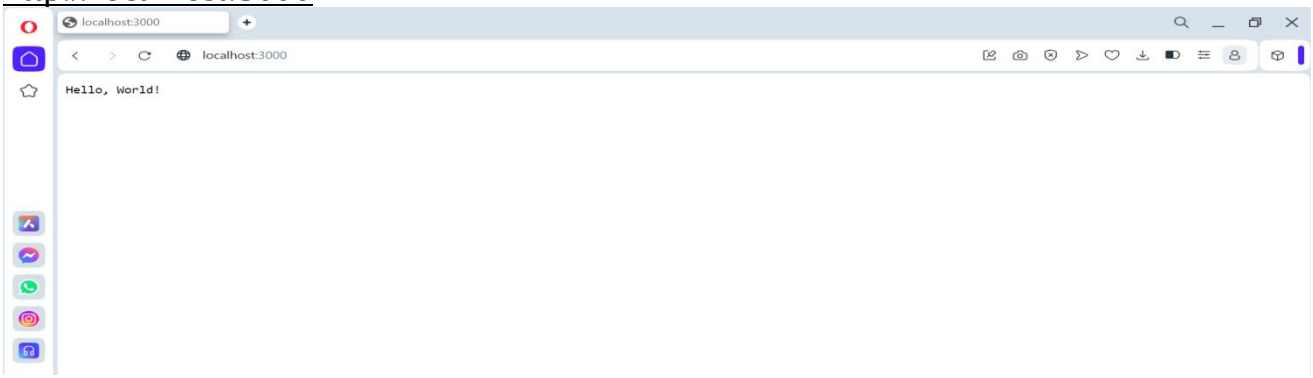
Output :

In the Terminal:

```
C:\>d:

D:\>cd bnr

D:\BNR>node server.js
Server running at http://localhost:3000/
System Information: {
  platform: 'win32',
  arch: 'x64',
  totalMemory: 8466083840,
  freeMemory: 3779747840
}
Path Module Example: D:\BNR\example.txt
Custom event occurred: { message: 'Hello, custom event!' }
^C
D:\BNR>
D:\BNR>
```

In the Browser:

http://localhost:3000



Explantion:

- We import the **http** module which provides functionality to create HTTP servers and handle HTTP requests and responses.
- We create a server using the **createServer()** method. This method takes a callback function which is called every time a request is made to the server. Inside this callback function, we set the response header and send a simple text response (**Hello, World!** in this case).
- We use the **listen()** method to make the server listen on port **3000**. Once the server starts listening, it logs a message to the console.

Now let's explore the other modules:

1. OS Module:

The **os** module provides a way of interacting with the operating system's underlying architecture. It can be used to gather system information such as CPU architecture, memory, and network interfaces.

2. Path Module:

The **path** module provides utilities for working with file and directory paths. It can be used to manipulate file paths in a platform-independent way

3. Event Module:

The **events** module provides an EventEmitter class that can be used to handle and respond to events in Node.js. It's a fundamental part of Node.js's asynchronous event-driven architecture.

Experiment 10

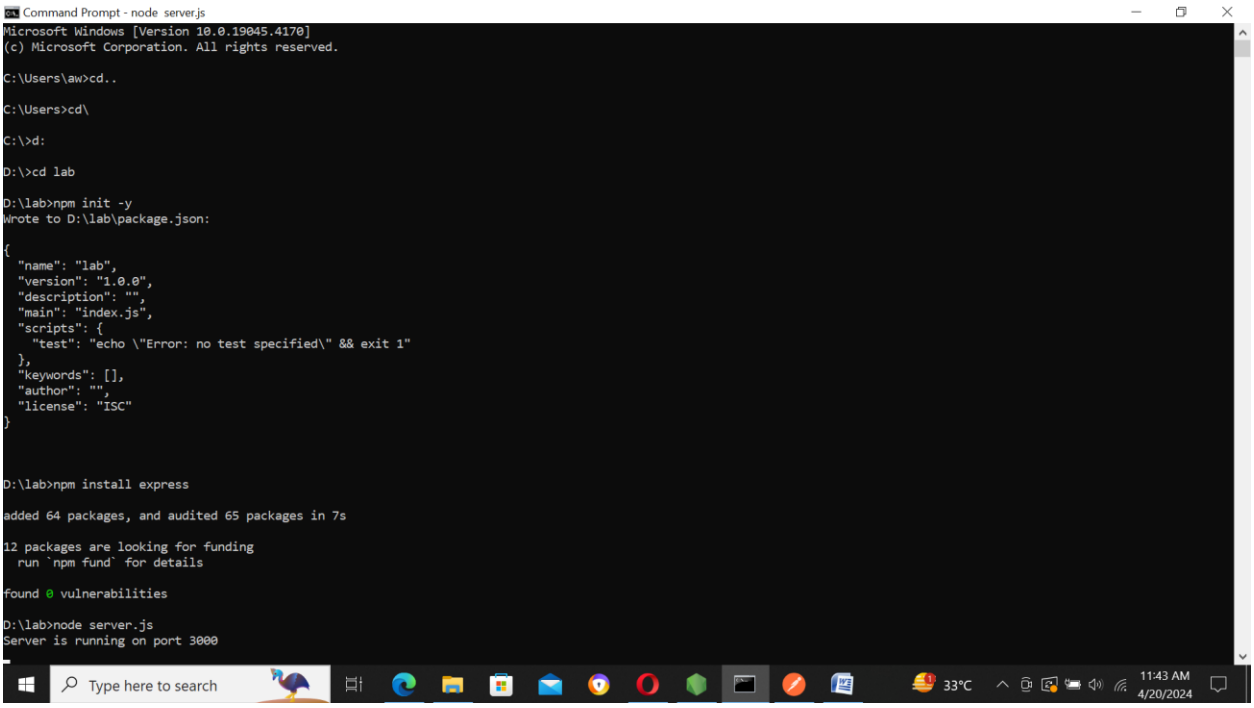
Aim:Web Develop an express web application that can interact with REST API to perform CRUD operations on student data. (Use Postman)
Procedure:

1. **Install Node.js**: If you haven't already, download and install Node.js from the official website: Node.js Downloads. This will also install npm, the Node Package Manager, which you'll use to install dependencies and run your application.
2. **Create a Project Directory**: Create a new directory where you'll store your Express application files. You can create this directory anywhere on your system.
3. **Open Command Prompt**: Open the Command Prompt (cmd) on your Windows machine. You can do this by searching for "cmd" in the Start menu and clicking on it.
4. **Navigate to Project Directory**: Use the **cd** command to navigate to the directory you created for your Express application. For example:
5. **Initialize Project**: If you haven't already initialized your project with npm, you can do so by running:

This will create a package.json file in your project directory.

6. **Install Express**: Install the Express.js framework by running the following command:
   npm install express
7. **Create server.js File**: Create a file named **server.js** in your project directory and paste the code provided earlier in that file.
8. **Run the Server**: In the Command Prompt, run the following command to start the server:
   node server.js

You should see the message "Server is running on port 3000" indicating that your server is up and running.

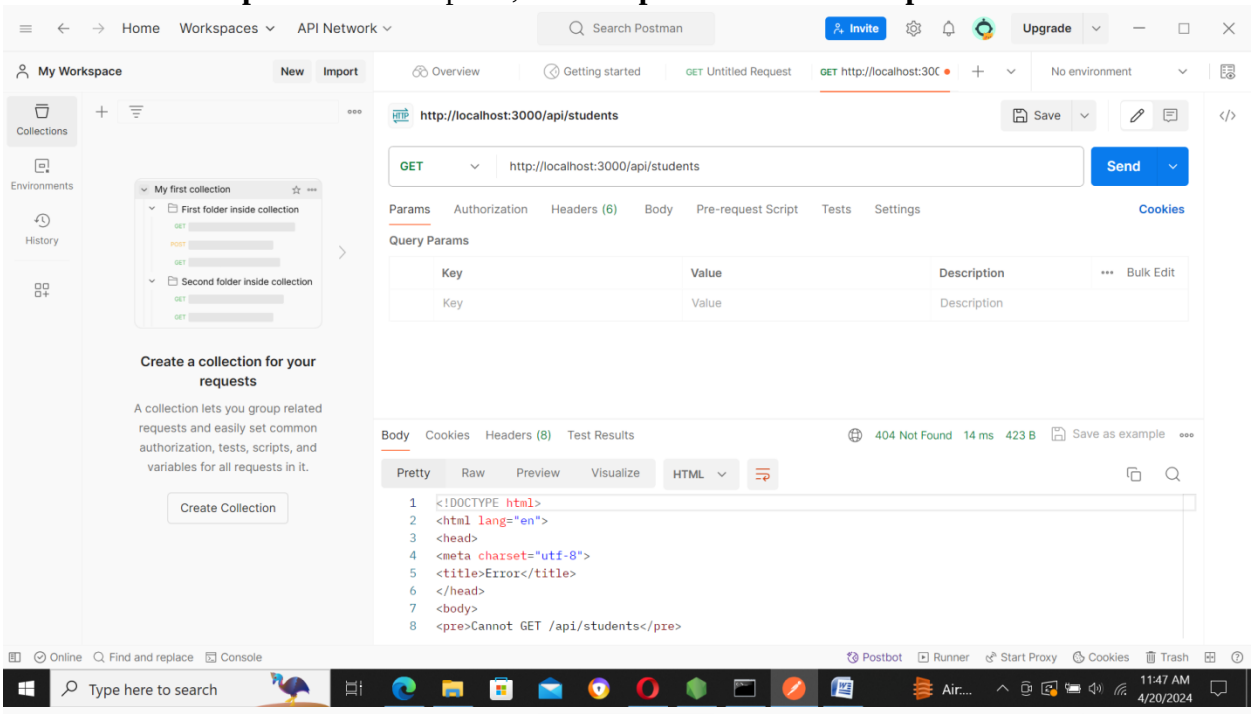


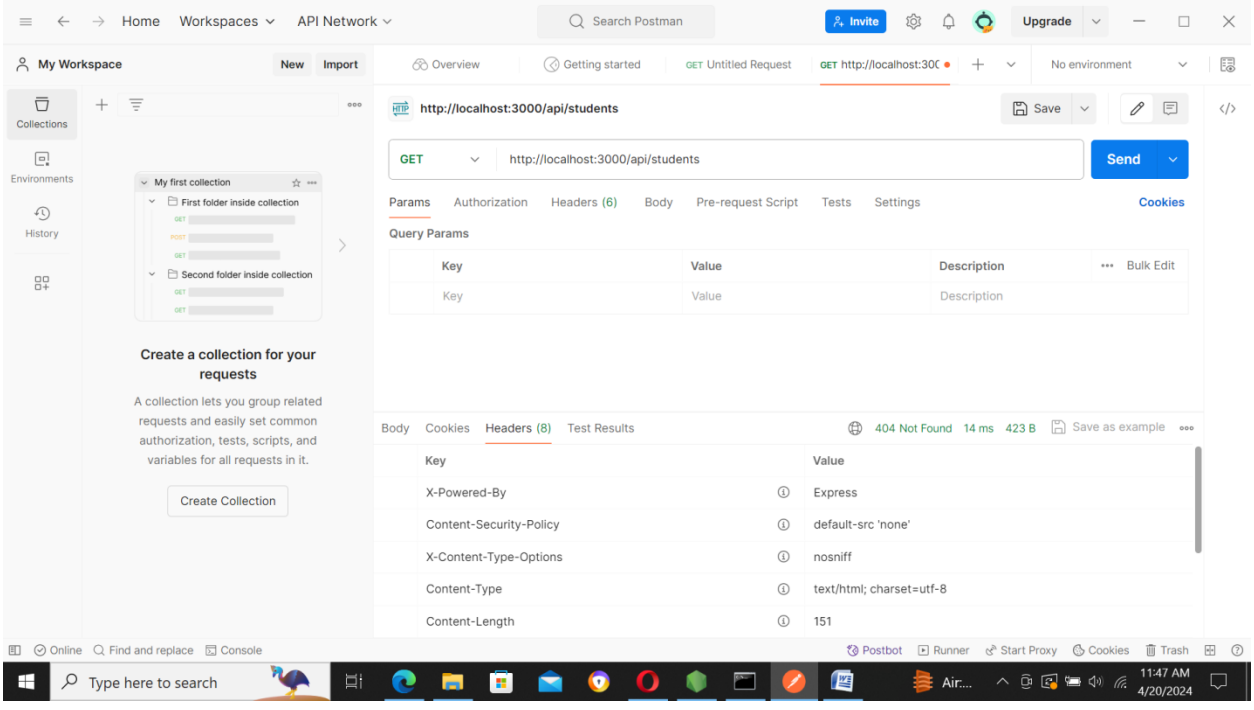
9. **Test with Postman**: Open Postman and send requests to the endpoints (**GET /api/students**, **GET /api/students/:id**, **POST /api/students**, **PUT /api/students/:id**, **DELETE /api/students/:id**) to perform

CRUD operations on student data.

To test your Express web application using Postman, follow these steps:

1. **Open Postman**: Open the Postman application on your computer. If you haven't installed it yet, you can download it from postman.com.
2. **Create a New Request**: Click on the "New" button in the top-left corner of the Postman window and select "Request."
3. **Enter Request Details**: In the request window, enter the URL for your Express server along with the endpoint you want to test. For example, if your server is running locally on port 3000 and you want to test the **GET /api/students** endpoint, enter **http://localhost:3000/api/students** in the address bar.





S.No	Question
1	**What is Express.js?**
2	**What is Node.js?**
3	**What is a REST API?**
4	**What is CRUD operation?**
5	**Which HTTP methods are used in CRUD?**
6	**What is middleware in Express?**
7	**What is the purpose of express.json()?**
8	**How do you start a Node.js server?**
9	**What is Postman used for?**
10	**What are route parameters in Express?**
11	**What is the difference between PUT and PATCH?**
12	**What is the use of app.get() method?**
13	**What is the use of app.post() method?**
14	**What is the use of app.put() method?**
15	**What is the use of app.delete() method?**
16	**What is a request object in Express?**
17	**What is a response object in Express?**
18	**What is HTTP status code?**
19	**What is the default port used in Express apps?**
20	**What is package.json file?**

EXERMENT 11

Aim: For the above application create authorized end points using JWT (JSON Web Token)

Step 1:First, install the jsonwebtoken package:

npm install jsonwebtoken

Step 2: update the server.js file:

Source code:

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const app = express();
const PORT = 3000;
const JWT_SECRET = 'your_secret_key'; // Replace with your own secret key
// Dummy user data
const users = [
  { id: 1, username: 'admin', password: 'password' }
];
// Dummy student data
let students = [
  { id: 1, name: 'John Doe', age: 20 },
  { id: 2, name: 'Jane Smith', age: 22 }
];
app.use(bodyParser.json());
// Middleware to authenticate JWT token
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (token == null) return res.sendStatus(401);
  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}
// Login endpoint to generate JWT token
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username && u.password === password);
  if (!user) return res.sendStatus(401);
  const accessToken = jwt.sign({ username: user.username, id: user.id }, JWT_SECRET);
  res.json({ accessToken });
});
// Authorized endpoints
app.get('/students', authenticateToken, (req, res) => {
```

```
  res.json(students);
});
app.post('/students', authenticateToken, (req, res) => {
  // Same as before
});
app.put('/students/:id', authenticateToken, (req, res) => {
  // Same as before
});
app.delete('/students/:id', authenticateToken, (req, res) => {
  // Same as before
});
// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

With this setup, the /login endpoint accepts a username and password and returns a JWT token. You can then use this token in the Authorization header (Bearer <token>) to access the authorized endpoints (/students). The authenticateToken middleware verifies the JWT token before allowing access to these endpoints.
Make sure to replace 'your_secret_key' with a strong, unique secret key in a real-world application.

THE STEPS TO EXECUTE THE PROVIDED PROGRAM:

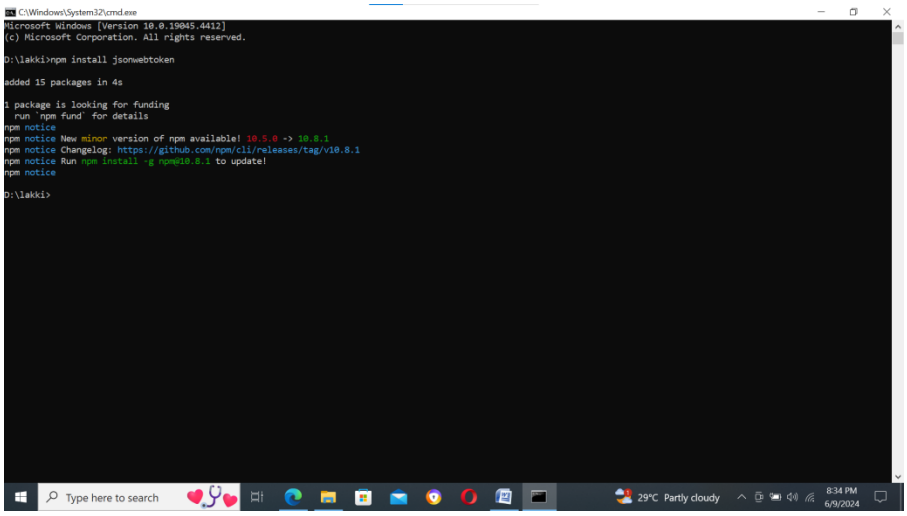
1. **Setup the Environment**:
   - Make sure you have Node.js installed on your system. If not, download and install it from the official Node.js website.
   - Create a new directory for your project.
   - Open a terminal or command prompt and navigate to the project directory.

2. **Initialize the Project**:
   - Run npm init -y in the terminal to initialize a new Node.js project with default settings.

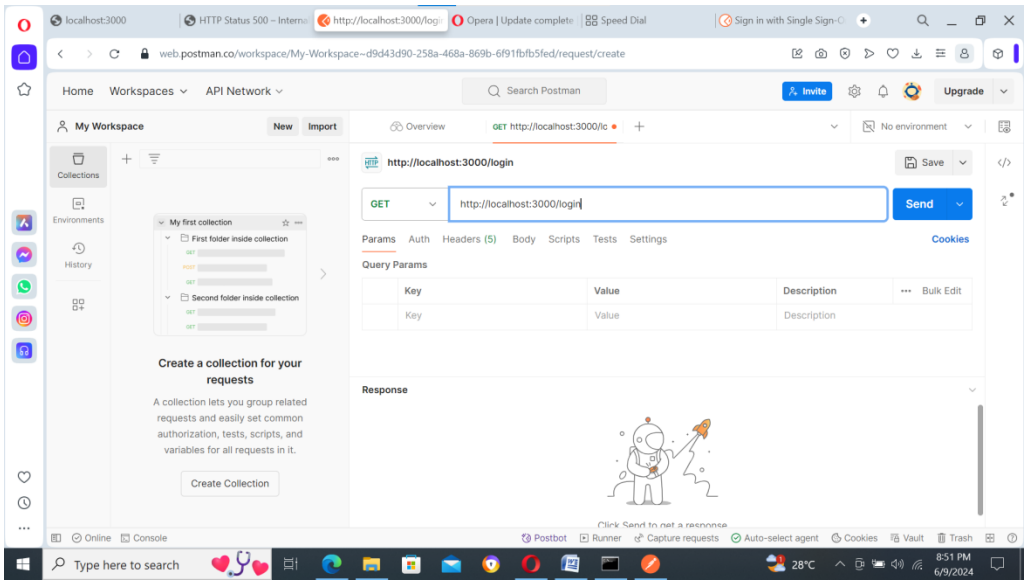
```
D:\lakki>npm init -y
Wrote to D:\lakki\package.json:

{
  "dependencies": {
    "jsonwebtoken": "^9.0.2"
  },
  "scripts": {
    "start": "node server.js"
  },
  "name": "lakki",
  "version": "1.0.0",
  "main": "server.js",
  "devDependencies": {},
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}


D:\lakki>
```

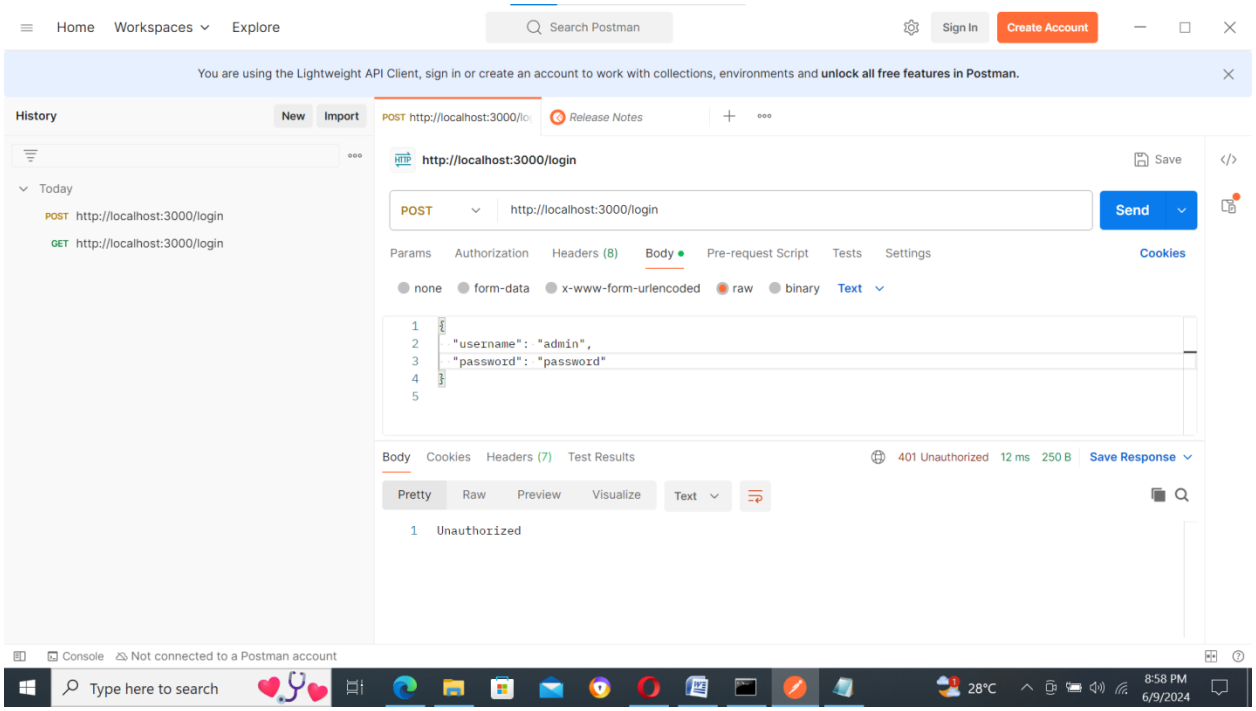
3. **Install Dependencies**:
   - Run npm install express body-parser jsonwebtoken to install Express, Body-parser, and Jsonwebtoken packages.

Step 1:First, install the jsonwebtoken package:



4. **Create the Server File**:
   - Create a file named server.js in your project directory.
5. **Copy and Paste Code**:
   - Copy the provided code snippet and paste it into the server.js file.
6. **Replace the Secret Key**:
   - Replace 'your_secret_key' with a strong and unique secret key of your choice. It's important to keep this key secure.
7. **Save the File**:
   - Save the server.js file.
8. **Run the Server**:
   - In the terminal, run node server.js to start the Express server.
9. **Test the Endpoints**:
   - Use a tool like Postman to test the endpoints.
   - Send a POST request to http://localhost:3000/login with a JSON body containing username and password to obtain a JWT token.
   - Use the obtained token to make requests to the authorized endpoints (/students). For example, you can send a GET request to http://localhost:3000/students with the token in the Authorization header (Bearer <token>).





10. **Verify the Output**:
    - Verify that you receive the expected responses from the endpoints.

S.No	Question
1	**What is JWT (JSON Web Token)?**
2	**What are the three parts of a JWT?**
3	**What is the purpose of JWT in web applications?**
4	**What is authentication vs authorization?**
5	**What is the use of jsonwebtoken package?**
6	**What is a secret key in JWT?**
7	**What does jwt.sign() do?**
8	**What does jwt.verify() do?**
9	**What is the purpose of the Authorization header?**
10	**What is Bearer token?**
11	**What is middleware in JWT authentication?**
12	**Why do we use authenticateToken function?**
13	**What happens if token is missing?**
14	**What happens if token is invalid?**
15	**What is the role of /login endpoint?**
16	**How is a token generated after login?**
17	**Why should the secret key be kept secure?**
18	**What is the use of req.user in middleware?**
19	**How does JWT improve security?**
20	**What are the limitations of JWT?**

EXERMENT 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: The basic outline of how you can create a React application for a student management system with registration, login, contact, and about pages, and implement routing to navigate through these pages.

First, 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 application.

npx create-react-app student-management-system

2.Navigate to the Project Directory: Go to the project directory.

cd student-management-system

3.Install React Router: Install React Router, which will be used for navigation.

npm install react-router-dom

4. 4. **Create Components**: Inside the `src` folder, create components for each page:

- `Registration.js`
- `Login.js`
- `Contact.js`
- `About.js`

5. **Implement Routing**: Create a file named `App.js` in the `src` folder and implement routing there.

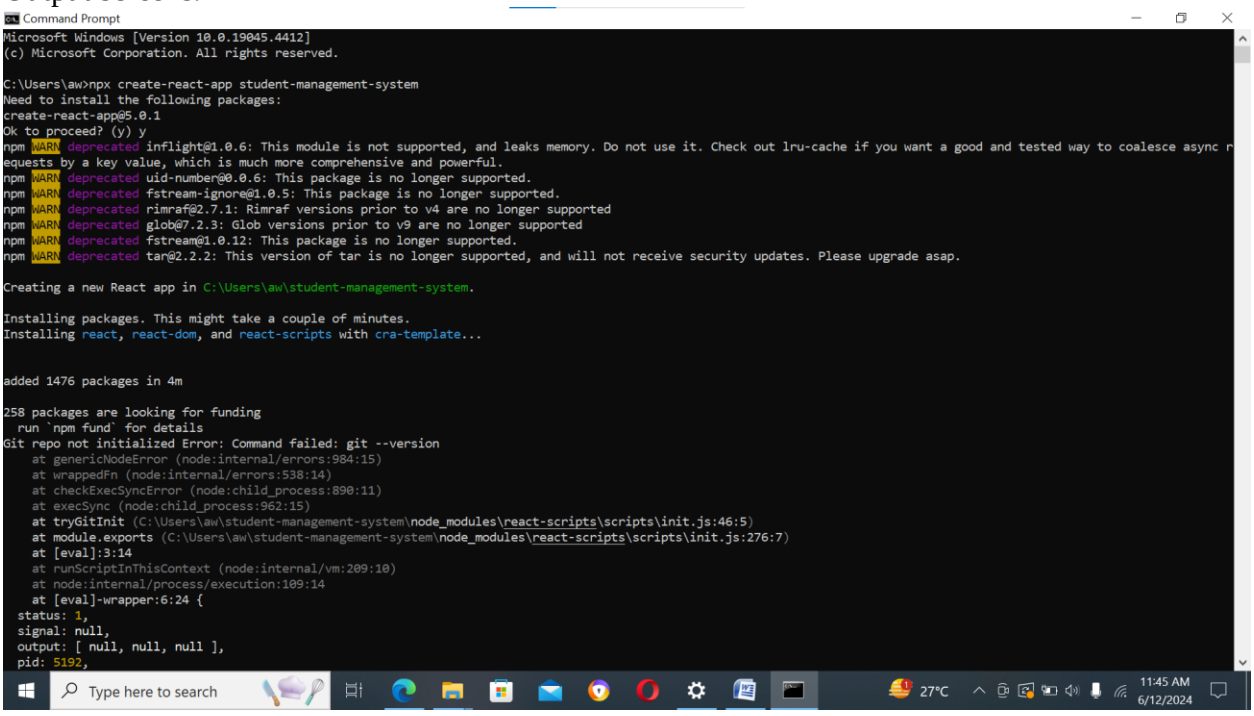
```
import React from 'react';

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';



function App() {

 return (

  <Router>
```

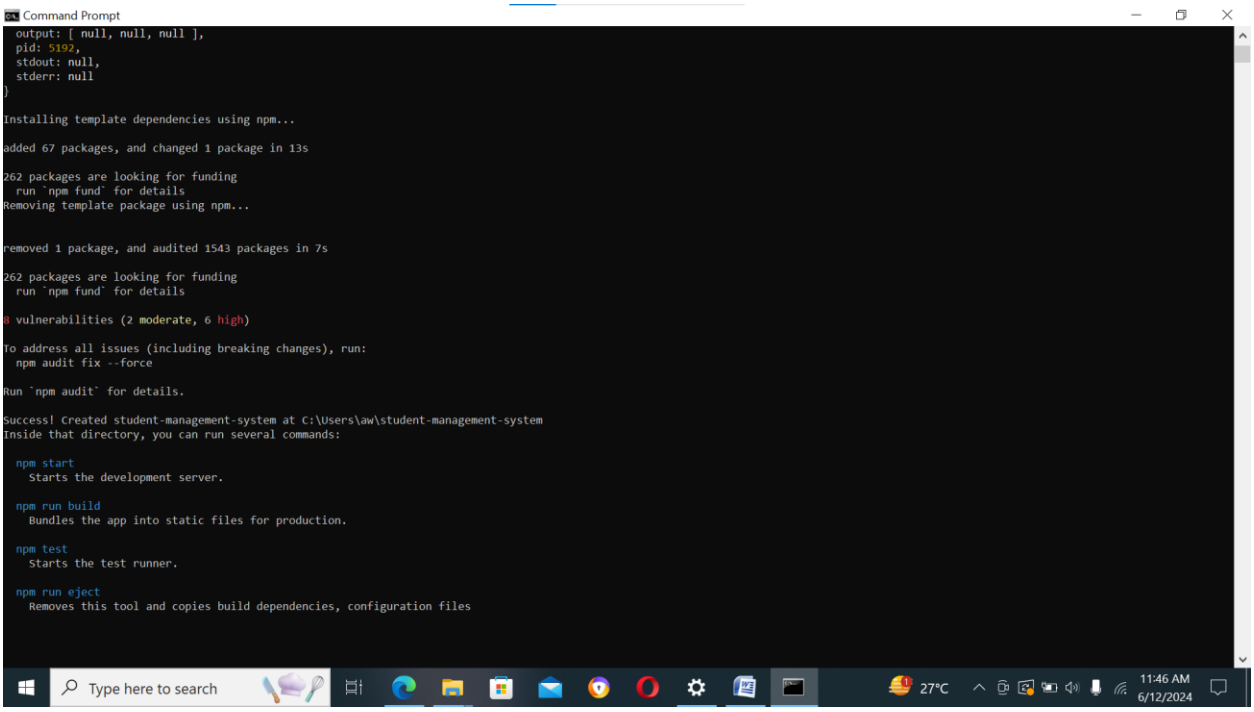
```
      <div>
        <Switch>
          <Route exact path="/" component={Login} />
          <Route path="/registration" component={Registration} />
          <Route path="/contact" component={Contact} />
          <Route path="/about" component={About} />
        </Switch>
      </div>
    </Router>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

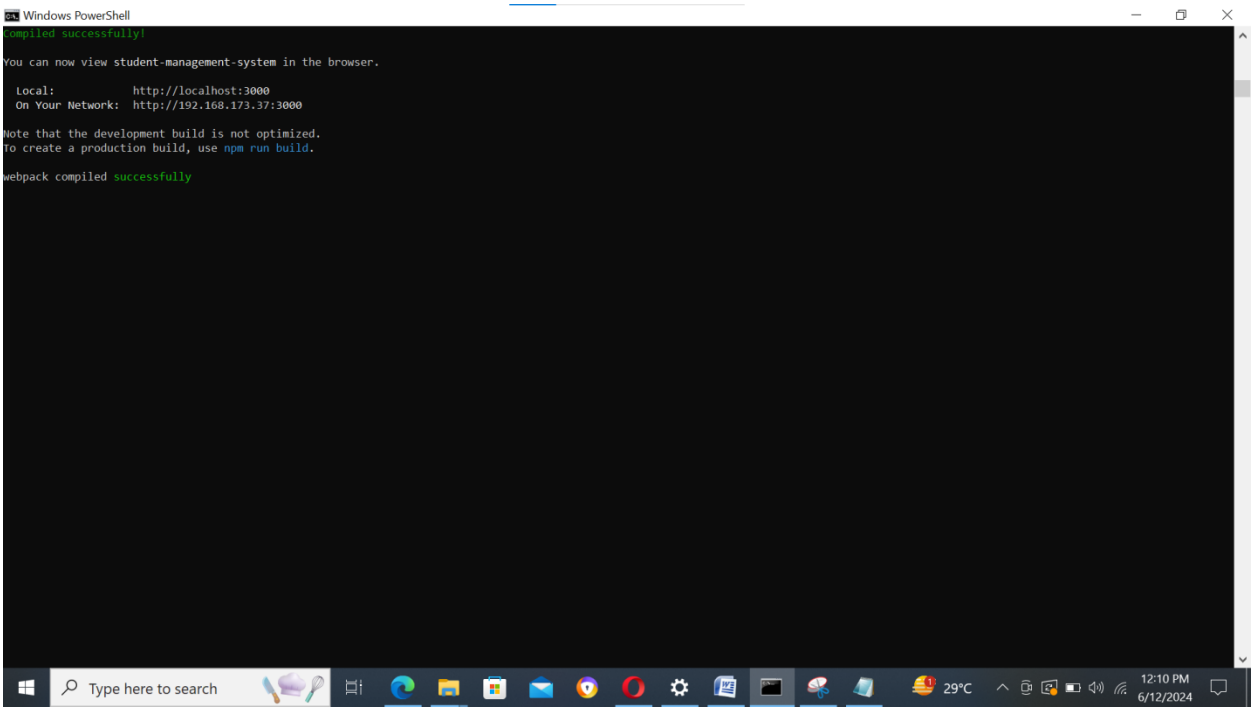
- **Create Navigation Links**: You can create navigation links in each component to navigate between pages.
- **Styling**: Style your components and pages as needed using CSS or any CSS-in-JS solution.
- **Start the Development Server**: Run the following command to start the development server:

  npm start

  Output screens:




Command Prompt
output: [null, null, null],
pid: 5192,
stdout: null,
stderr: null
}
Installing template dependencies using npm...
added 67 packages, and changed 1 package in 13s
262 packages are looking for funding
run `npm fund` for details
Removing template package using npm...
removed 1 package, and audited 1543 packages in 7s
262 packages are looking for funding
run `npm fund` for details
8 vulnerabilities (2 moderate, 6 high)
To address all issues (including breaking changes), run:
npm audit fix --force
Run `npm audit` for details.
Success! Created student-management-system at C:\Users\aw\student-management-system
Inside that directory, you can run several commands:
npm start
Starts the development server.
npm run build
Bundles the app into static files for production.
npm test
Starts the test runner.
npm run eject
Removes this tool and copies build dependencies, configuration files
Type here to search
27°C
11:46 AM
6/12/2024


Windows PowerShell
Compiled successfully!
You can now view student-management-system in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.173.37:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
Type here to search
29°C
12:10 PM
6/12/2024

```
Happy hacking!

C:\Users\aw>cd student-management-system

C:\Users\aw\student-management-system>npm install react-router-dom

added 3 packages, and audited 1546 packages in 9s

262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\aw\student-management-system>
```

Result: Nowwe created react application and it is ready to deploy on the server.

If you've completed building your React application and you're ready to deploy it, here are the general steps you can follow:

1. **Build Your Application**: First, you need to build your React application for production. Run the following command in your terminal:

npm run build

This will create a production-ready build of your React app in the `build` folder.

```
C:\Users\aw\student-management-system>npm run build

> student-management-system@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled with warnings.

[eslint]
src\App.js
  Line 2:42:  'Switch' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

File sizes after gzip:

  48.66 kB  build\static\js\main.4f0a116b.js
  1.78 kB   build\static\js\453.4c5687d4.chunk.js
  263 B     build\static\css\main.e6c13ad2.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  serve -s build

Find out more about deployment here:

  https://cra.link/deployment


C:\Users\aw\student-management-system>
```

S.No	Question
1	**What is React?**
2	**What is a component in React?**
3	**What is JSX?**
4	**What is the purpose of React Router?**
5	**What is routing in React?**
6	**What is the difference between BrowserRouter and HashRouter?**
7	**What is a Route in React Router?**
8	**What is the use of Link component?**
9	**What is the difference between Link and NavLink?**
10	**What is state in React?**
11	**What is props in React?**
12	**What is the use of useState hook?**
13	**What is the use of useEffect hook?**
14	**What is form handling in React?**
15	**How do you implement login functionality in React?**
16	**What is conditional rendering?**
17	**What is a single-page application (SPA)?**
18	**What is the purpose of registration page?**
19	**How does navigation work between pages in React?**
20	**What is the folder structure of a React application?**

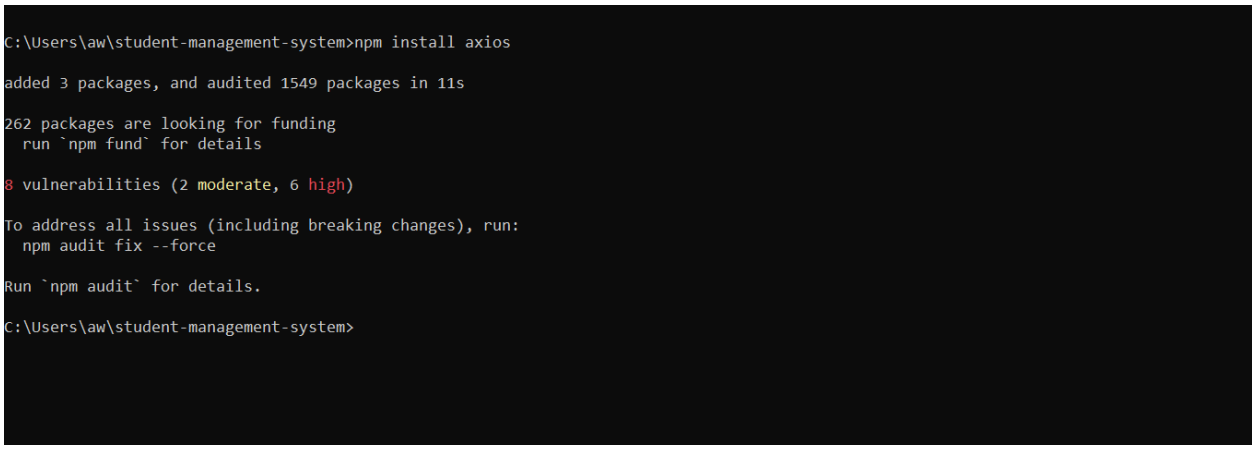
EXPERIMENT-13

AIM: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

Solution:I can outline how you could set this up in React.

1. **Set up React App**: If you haven't already, set up a React application using Create React App or any other method you prefer.
2. **Install Axios**: Axios is a popular HTTP client for making requests in JavaScript. You can use it to fetch data from the OpenWeatherMap API. Install Axios using npm or yarn:

npm install axios



Install Chart.js: Chart.js is a JavaScript library for creating charts and graphs. You can use it to display weather information graphically. Install Chart.js using npm or yarn:

npm install chart.js

```
C:\Users\aw\student-management-system>npm install chart.js

added 2 packages, and audited 1551 packages in 8s

262 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\aw\student-management-system>
```

Create a WeatherService Component: This component will handle fetching weather data from the OpenWeatherMap API.

```
// WeatherService.js


import axios from 'axios';


const API_KEY = 'YOUR_API_KEY';

const BASE_URL = 'https://api.openweathermap.org/data/2.5';


const WeatherService = {

  getCurrentWeather: async (city) => {

    try {

      const response = await axios.get(`${BASE_URL}/weather?q=${city}&appid=${API_KEY}`);

      return response.data;

    } catch (error) {

      console.error('Error fetching current weather:', error);

      throw error;

    }

  },
```

```
  getHistoricalWeather: async (city, startDate, endDate) => {
    try {
      const response = await
axios.get(`${BASE_URL}/onecall/timemachine?lat=${lat}&lon=${lon}&dt=${date}&appid=${API_KEY}`);
      return response.data;
    } catch (error) {
      console.error('Error fetching historical weather:', error);
      throw error;
    }
  }
};
export default WeatherService;
```

Create a WeatherChart Component: This component will use Chart.js to display the weather information graphically.

```
// WeatherChart.js

import React, { useEffect, useState } from 'react';
import Chart from 'chart.js';

const WeatherChart = ({ data }) => {
  const [chart, setChart] = useState(null);

  useEffect(() => {
    if (data) {
      const ctx = document.getElementById('weatherChart').getContext('2d');
      const newChart = new Chart(ctx, {
```

```
      type: 'line',
      data: {
        labels: data.map(entry => entry.date),
        datasets: [
          {
            label: 'Temperature (°C)',
            data: data.map(entry => entry.temperature),
            borderColor: 'rgb(255, 99, 132)',
            fill: false
          }
        ]
      }
    });
    setChart(newChart);
  }
 }, [data]);


 return (
  <div>
    <canvas id="weatherChart" />
  </div>
 );
};export default WeatherChart;
```

Make sure to replace `'YOUR_API_KEY'` with your actual OpenWeatherMap API key, and `'cityName'`, `startDate`, and `endDate` with appropriate values for the city and date ranges you want to fetch historical weather data for.

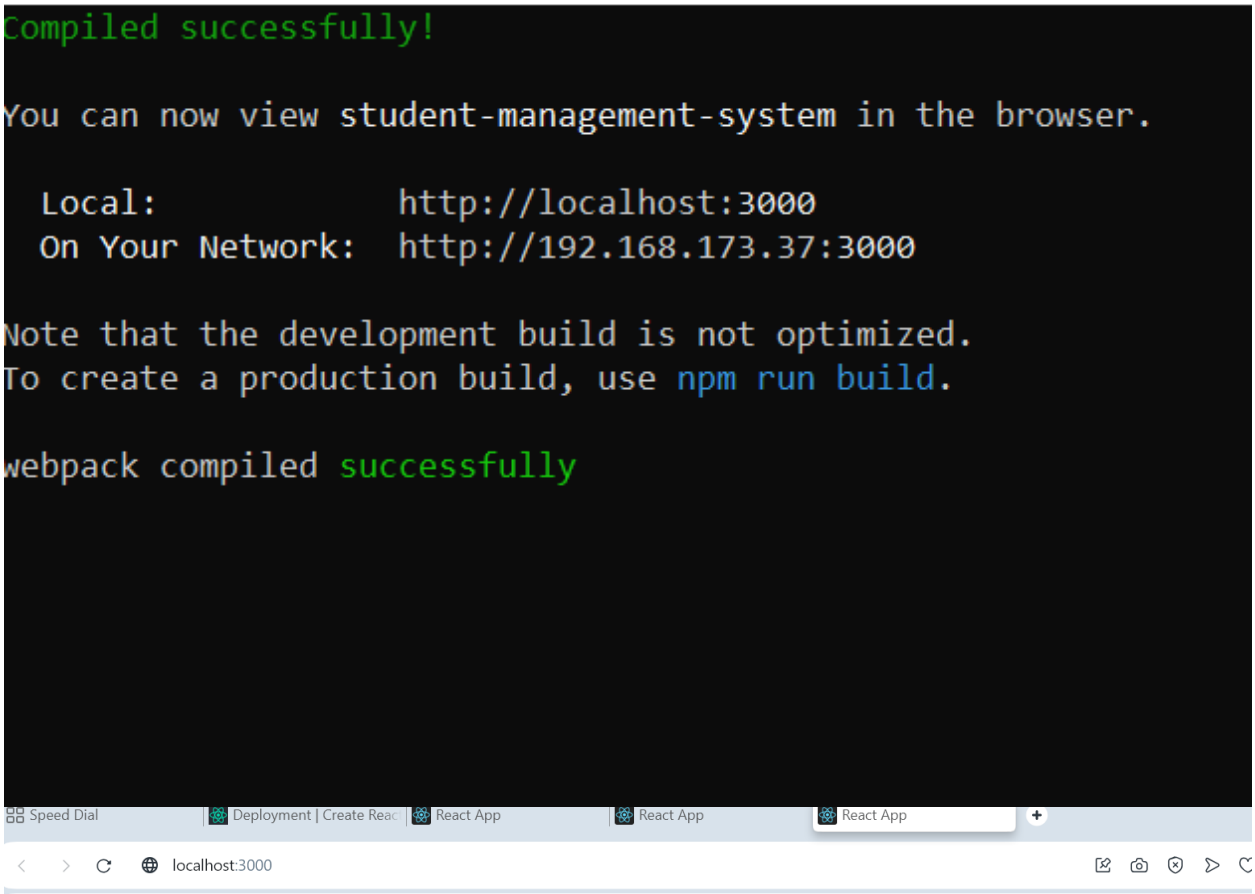
With this setup, you should be able to fetch current and historical weather data from OpenWeatherMap and display it graphically using Chart.js in your React application

After thet you need to update app.js with following code.

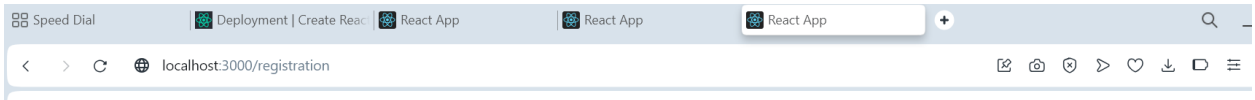
```
// App.js


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


function App() {
  return (
    <Router>
      <Routes>
        <Route exact path="/" element={<Login />} />
        <Route path="/registration" element={<Registration />} />
        <Route path="/contact" element={<Contact />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}
export default App;
```

Then run the server.

```
Npm start
```



Login Page



Registration Page

S.No	Question
1	What is Axios in React?
2	Why is Axios used instead of fetch?
3	What is an API?
4	What is OpenWeatherMap API?
5	What is an API key?
6	Why is the API key required in weather applications?
7	What is asynchronous programming in JavaScript?
8	What is the use of async and await?
9	What is Chart.js?
10	How do you integrate Chart.js in React?
11	What is a line chart in Chart.js?
12	What is useEffect hook used for?
13	What is useState hook used for?
14	How do you fetch data in React?
15	What is the purpose of WeatherService component?
16	What is the purpose of WeatherChart component?
17	What is props in React?
18	How is data passed from one component to another?
19	What is the difference between current and historical weather data?
20	How do you handle errors while fetching API data?

EXPERIMENT-14

Aim:To create a TODO application using React with necessary components such as adding, deleting, and displaying tasks, and to deploy the application on GitHub.

Procedure:

1. Install Node.js and npm.
2. Create a new React application using:
   `npx create-react-app todo-app`
3. Navigate to the project folder:
   `cd todo-app`
4. Create required components such as TodoList, TodoItem, and AddTodo.
5. Implement functionality to add, delete, and display tasks using React hooks.
6. Style the application using CSS.
7. Run the application using:
   `npm start`
8. Initialize Git repository:
   `git init`
9. Add files and commit:
   `git add .`
   `git commit -m "Initial commit"`
10. Create a repository on GitHub and push the code:
    `git remote add origin <repository-url>`
    `git push -u origin main`
11. Deploy using GitHub Pages (optional):
    `npm install gh-pages`
    Configure homepage in package.json and deploy.

Source Code:

App.js

```
import React, { useState } from 'react';
import TodoList from './TodoList';
import AddTodo from './AddTodo';

function App() {
  const [todos, setTodos] = useState([]);

  const addTodo = (text) => {
    setTodos([...todos, { id: Date.now(), text }]);
  };

  const deleteTodo = (id) => {
```

```
    setTodos(todos.filter(todo => todo.id !== id));
  };

  return (
    <div>
      <h1>TODO App</h1>
      <AddTodo addTodo={addTodo} />
      <TodoList todos={todos} deleteTodo={deleteTodo} />
    </div>
  );
}

export default App;
```

AddTodo.js:

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


function AddTodo({ addTodo }) {
  const [input, setInput] = useState('');


  const handleSubmit = (e) => {
    e.preventDefault();
    if (!input) return;
    addTodo(input);
    setInput('');
  };


  return (
    <form onSubmit={handleSubmit}>
      <input
        value={input}
        onChange={(e) => setInput(e.target.value)}
```

```
        placeholder="Enter task"
      />
      <button type="submit">Add</button>
    </form>
  );
}
export default AddTodo;
```

TodoList.js:

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

function TodoList({ todos, deleteTodo }) {
  return (
    <ul>
      {todos.map(todo => (
        <TodoItem key={todo.id} todo={todo} deleteTodo={deleteTodo} />
      ))}
    </ul>
  );
}
export default TodoList;
```

TodoItem.js

```
import React from 'react';

function TodoItem({ todo, deleteTodo }) {
```

```
  return (
    <li>
      {todo.text}
      <button onClick={() => deleteTodo(todo.id)}>Delete</button>
    </li>
  );}
export default TodoItem;
```

Output:

User can add tasks

User can view tasks

User can delete tasks

Application runs successfully in browser

S.No	Question
1	What is React?
2	What is a TODO application?
3	What is a component in React?
4	What is useState hook?
5	How do you add a new task in a TODO app?
6	How do you delete a task in a TODO app?
7	What is props in React?
8	How is data passed between components?
9	What is event handling in React?
10	What is conditional rendering?
11	What is the purpose of keys in React lists?
12	What is the role of App component?
13	What is the difference between state and props?
14	What is GitHub?
15	What is Git?
16	What is the purpose of git init?
17	What is git add command?
18	What is git commit?
19	What is git push?

S.No	Question
20	What is GitHub Pages?