A Laboratory Manual

for

Web Technologies

(III B.TECH CSE SEMISTER – II)

PRAGNANAM BRAHMA
ANITS

Prepared By
S Bosu
Babu

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision:

Our vision is to emerge as a world class Computer Science and Engineering department through excellent teaching and strong research environment that responds swiftly to the challenges of changing computer science technology and addresses technological needs of the stakeholders.

MISSION:

To enable our students to master the fundamental principles of computing and to develop in them the skills needed to solve practical problems using contemporary computer-based technologies and practices to cultivate a community of professionals who will serve the public as resources on state-of the-art computing science and information technology.

Program Outcomes: (PO's)

Graduate Attribute1:	Engineering Knowledge
PO-A	An ability to apply the knowledge of basic engineering sciences, humanities, core engineering and computing concept in modeling and designing computer based systems.
Graduate Attribute2:	Problem Analysis
PO-B	An ability to identify, analyze the problems in different domains and define the requirements appropriate to the solution.
Graduate Attribute3:	Design/Development of Solution
PO-C	An ability to design, implement & test a computer based system, component or process that meet functional constraints such as public health and safety, cultural, societal and environmental considerations.

Program Specific Outcomes:

1	Programming and software Development skills: Ability to acquire programming efficiency to analyze, design and develop optimal solutions, apply standard practices in software project development to deliver quality software product.
2	Computer Science Specific Skills: Ability to formulate, simulate and use knowledge in various domains like data engineering, image processing and information and network security, artificial intelligence etc., and provide solutions to new ideas and innovations.

Graduate Attribute4:	Conduct Investigations of Complex Problems
PO-D	An ability to apply computing knowledge to conduct experiments and solve complex problems, to analyze and interpret the results obtained within specified timeframe and financial constraints consistently.
Graduate Attribute5:	Modern Tool Usage
PO-E	An ability to apply or create modern techniques and tools to solve engineering problems that demonstrate cognition of limitations involved in design choices.
Graduate Attribute6:	The Engineer and Society
PO-F	An ability to apply contextual reason and assess the local and global impact of professional engineering practices on individuals, organizations and society.
Graduate Attribute7:	Environment and Sustainability
PO-G	An ability to assess the impact of engineering practices on societal and environmental sustainability.

Graduate Attribute8:	Ethics
PO-H	Ability to apply professional ethical practices and transform into good responsible citizens with social concern.
Graduate Attribute9:	Individual and Team Work
PO-I	Acquire capacity to understand and solve problems pertaining to various fields of engineering and be able to function effectively as an individual and as a member or leader in a team.
Graduate Attribute10:	Communication
PO-J	An ability to communicate effectively with range of audiences in both oral and written forms through technical papers, seminars, presentations, assignments, project reports etc.
Graduate Attribute11:	Project Management and Finance
PO-K	An ability to apply the knowledge of engineering, management and financial principles to develop and critically assess projects and their outcomes in multidisciplinary areas.
Graduate Attribute12:	Life-long Learning
PO-L	An ability to recognize the need and prepare oneself for lifelong self learning to be abreast with rapidly changing technology.

Course Outcomes:

CO-1 : Design Static Web pages and Dynamic Web pages using HTML and validate with JavaScript
respectively.
CO-2 : Create website using server side scripting language PHP
CO-3 : Develop interactive Web applications using Node JS and ExpressJS
CO-4 : Demonstrate the CRUD application using Flask and MongoDB

CO PO Mapping

CO	(PO)	(PSO)
CO-1	1,3,5,6,8,10,12	1,2
CO-2	1,2,3,5,6,8,10,11,12	1
CO-3	1,2,3,4,5,6,8,9,10,11,12	1
CO-4	1,2,3,4,5,6,8,9,10,11,12	1

List of Experiments

Lab Programs

1	Design Static Webpage using HTML Components	CO1
2	Design Webpage using CSS	CO1
3	Create Dynamic Webpage using JavaScript	CO1
4	Develop Dynamic Webpage Using PHP Script	CO2
5	Develop PHP application with Database connection	CO2
6	Implement Modules in NodeJS	CO3
7	Develop mini application using Express and NodeJS	CO3
8	Implement HTTP methods using Flask	CO4
9	Implement Sessions concept using Flask	CO4
10	Develop CRUD operations using MongoDB	CO4
11	Develop Main Project using Python/Express and MongoDB/MySql.	CO4

List Of Industry Relevant Skills:

- HTML (Hypertext Markup Language) - the basic building block of web pages.
- CSS (Cascading Style Sheets) - used for styling and formatting web pages.
- JavaScript - a programming language used to add interactivity and dynamic effects to web pages.
- PHP (Hypertext Preprocessor) - a server-side scripting language used for web development.
- MySQL - a popular open-source relational database management system.
- Node.js - a cross-platform, open-source JavaScript runtime environment used for server-side programming.
- Express.js - a popular web application framework for Node.js used for building web applications and APIs.
- Flask - a lightweight web application framework for Python used for building web applications.
- MongoDB - a popular NoSQL document-oriented database used for storing and managing data.

In addition to these specific skills, it is also important for students to have a strong understanding of web development concepts such as responsive design, accessibility, security, and version control. It's also important to develop problem-solving skills, critical thinking skills, and the ability to work collaboratively in a team.

Guidelines To Teachers

- Faculty must verify the observations and records before assign the system.
- Faculty must verify Students Id cards before enter into Lab
- Faculty must take the attendance starting and ending of the lab time period.

Instructions To Students:

- Students should use computer related components smoothly
- Students should not carry other items into lab.
- Students must wear the dress code and ID cards.

Guidelines To Lab Programmers:

- Lab Programmers must verify All the Systems whether they are working properly or not.
- Lab Programmers must verify All the other equipment's(devices like ACs).

Lab Rubrics

Key Performance Criteria(KPC) (25 pts)	**4-Very Good**	**3-Good**	**2-Fair**	**1-Need to improve**
Problem Statement (2)	Detail understanding of the problem (2)	Understanding of the problem (2)	Basic understanding of the problem (1)	Partial understanding of the problem (1)
Experimental Procedure/ algorithm/ flow chart/ analysis (4)	The procedure is explained and well designed the problem with appropriate analysis (4)	The procedure is explained and designed the problem with analysis (3)	Missing some experimental procedure with partial analysis (2)	Missing major experimental details and analysis (1)
Implementation (4)	Implement Optimal solution with appropriate results for all the inputs	Implement solution with correct results for most of the inputs	implement solution with the correct answers for some inputs and results wrong answers for some cases	Implement Solution does not produce the appropriate results for the given inputs
Test Case verification (3)	Produces correct output for all possible test cases(3)	Produces correct output for most of the test cases (2)	Produces correct output for some of the test cases (2)	Produces Wrong output for most of the test cases (1)
Viva voice / oral presentation(5)	In depth knowledge on the concept and answered all the questions(5)	Good knowledge on the concept and answered all the questions(4)	Basic knowledge on the concept and answered some of the questions(3)	With basic knowledge on the concept and answered few questions(2)
Presentation of record / documentation(4)	Presented the content effectively and Submitted on time (4)	Presented the content and Submitted on time (3)	Presented the in-complete content and Submitted . (2)	Presented the wrong content and submitted in delay.(1)
Code of conduct (courtesy, safety, behavioral aspects, ethics etc.)(3)	While conducting the procedure, the student is in proper dress code, always respectful of others and leaves the area clean.(3)	While conducting the procedure, the student is in proper dress code, many times respectful of others and leaves the	While conducting the procedure, the student is in partial dress code, sometimes respectful of others and leaves the area clean only after	While conducting the procedure, the student is not in proper dress code , not respectful of othersand leaves the area messy even after

		area clean only after being reminded.(2)	being reminded.(2)	being reminded.(1)

PRACTICAL EXPERIMENT 1:

● Design Static Webpage using HTML Components	CO1
● Design Webpage using CSS	CO1
● Create Dynamic Webpage using JavaScript	CO1

1. Practical significance :

- This practical experience can help students gain a better understanding of how these technologies work together to create web pages, and can prepare them for more advanced coursework or real-world web development projects.
- Furthermore, these experiments can be practically significant in terms of their potential applications in the workforce. Web development is a rapidly growing field, and the ability to create static and dynamic web pages using HTML, CSS, and JavaScript is a valuable skill in many industries. Therefore, these experiments can provide students with practical skills that can help them succeed in their future careers.

2. Relevant program outcomes:

CO	(PO)	(PSO)
CO-1	1,3,5,6,8,10,12	1,2

3. Compitency and Practical Skills:

- Ability to use responsive design techniques and CSS frameworks like Bootstrap or Foundation to create mobile-friendly layouts.
- Understanding of basic programming concepts like variables, data types, and control structures, and ability to use JavaScript functions to perform specific tasks.

- Understanding of the Document Object Model (DOM) and how it relates to JavaScript, and ability to use event handling to make web pages interactive.

4. Prerequisites:

- At a minimum, students would need a basic understanding of computer programming concepts such as variables, data types, functions, and control structures. They would also need to be familiar with basic web technologies such as HTML tags, attributes, and syntax.

- In addition, for the experiment on designing web pages using CSS, students would need to have a basic understanding of CSS selectors, properties, and values, and how to use them to style HTML elements.

5.Resources Required:

Software Requirements	**Hardware Requirements**
Text Editor	Processor (multi-core recommended)
Web Browser	Memory (RAM): 4GB minimum
Optional: IDE	Storage: Sufficient space for software
Operating System : Windows, macOS, Linux	

6. Precautions:

- **Backup and version control**: It is important to back up all files and code being worked on in case of hardware or software failure. Version control should also be implemented to track changes made to the code and to facilitate collaboration.
- **Security:** Lab computers should be secure, with up-to-date anti-virus software and firewalls in place to protect against malware and unauthorized access.
- **Data privacy**: It is important to ensure that any personal or sensitive data is properly secured and protected. This includes ensuring that login credentials are not shared with others and that any data collected is handled in accordance with relevant privacy laws and regulations.
- **Network security**: Lab computers should be connected to a secure network, and access to the internet and other network resources should be restricted to prevent unauthorized access or downloading of malicious software.

- **Safe coding practices**: Lab students should be encouraged to follow safe coding practices, such as commenting code, testing code thoroughly, and avoiding potential security vulnerabilities such as SQL injection or cross-site scripting.
- **Proper lab environment**: The lab should be a safe and clean environment, with proper ventilation and adequate lighting.

7. Algorithm/circuit/Diagram/Description:

- Define the purpose and requirements of the webpage
- Plan the layout and structure of the webpage using HTML
- Add content to the webpage using HTML
- Apply styling to the webpage using CSS
- Add interactivity to the webpage using JavaScript
- Test and validate the webpage using web development tools and techniques
- Make necessary changes and improvements based on the testing and validation results
- Publish the webpage to a web server or hosting service for public access

8. Test cases:

1. **Functional testing:**
   - Verify that all links and buttons work correctly and navigate to the correct page or location.
   - Test all form fields to ensure that they are functioning properly, and that data entered into the fields is captured and stored as intended.
   - Check that all images and videos are displayed properly, and that they do not take too long to load.
   - Test any interactive elements, such as sliders or dropdown menus, to ensure they are working correctly.
2. **Browser compatibility testing:**
   - Test the webpage on multiple browsers, such as Chrome, Firefox, Safari, and Internet Explorer, to ensure that it is functional and displays correctly on all of them.
   - Test the webpage on multiple devices with different screen sizes and resolutions to ensure that it is responsive and adjusts properly to different screens.
3. **Usability testing:**

- Test the webpage's user interface and user experience to ensure that it is intuitive and easy to use.
- Test the webpage's accessibility and ensure that it conforms to the relevant accessibility standards.

4. **Performance testing:**

- Test the webpage's load time to ensure that it loads quickly and efficiently.
- Use a tool like Page Speed Insights to identify any performance bottlenecks and optimize the webpage accordingly.

9. Sample Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Sample Website</title>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
   <header>
      <nav>
         <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
         </ul>
      </nav>
   </header>
   <section>
      <h1>Welcome to Sample Website</h1>
      <p>This is paragraph</p>
   </section>
   <footer>
      <p>&copy; 2023 Sample Website.</p>
   </footer>
   <script src="script.js"></script>
</body>
</html>
------ -----------------------------------------------------------
header {
   background-color: #333;
   color: #fff;
   padding: 20px;
   height: 80px;
}
nav ul {
```

```
nav ul li {
   display: inline-block;
   margin: 0 10px;
}
nav ul li a {
   color: #fff;
   text-decoration: none;
}

section {
   margin: 40px auto;
   max-width: 800px;
   padding: 20px;
   background-color: #fff;
   border: 1px solid #ccc;
}
section h1 {
   color: #333;
   text-align: center;
}
footer {
   background-color: #333;
   color: #fff;
   text-align: center;
   padding: 20px;
   height: 60px;
}
footer p {
   margin: 0;
   font-size: 14px;
}
@media screen and (max-width: 480px) {
   nav ul li {
      display: block;
      margin: 10px 0;
   }
```

```
  list-style: none;
  margin: 0;
  padding: 0;
}
```

```
}
```

```
// Script for adding a class to the active link in the navigation menu

const currentLocation = location.href;
const navLinks = document.querySelectorAll("nav ul li a");

navLinks.forEach(link => {
 if (link.href === currentLocation) {
  link.classList.add("active");
 }
});
```

Registration Form: Html

```
<form id="registration-form">
 <label for="name">Name:</label>
 <input type="text" id="name" name="name">

 <label for="email">Email:</label>
 <input type="email" id="email" name="email">

 <label for="password">Password:</label>
 <input type="password" id="password" name="password">

 <label for="confirm-password">Confirm Password:</label>
 <input type="password" id="confirm-password" name="confirm-password">

 <button type="submit" id="submit-btn">Submit</button>
</form>

<div id="output"></div>
```

```
label {
 display: block;
 margin-bottom: 0.5rem;
}

input[type="text"],
input[type="email"],
input[type="password"] {
 display: block;
 margin-bottom: 1rem;
 padding: 0.5rem;
 border: 1px solid #ccc;
 border-radius: 4px;
 font-size: 1rem;
}
```

```
button[type="submit"] {
 background-color: #007bff;
 color: #fff;
 border: none;
 border-radius: 4px;
 padding: 0.5rem 1rem;
 font-size: 1rem;
 cursor: pointer;
}

#error {
 color: red;
 font-weight: bold;
}
```

```
const form = document.querySelector('#registration-form');
const output = document.querySelector('#output');

form.addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent the form from submitting

  const nameInput = document.querySelector('#name');
  const emailInput = document.querySelector('#email');
  const passwordInput = document.querySelector('#password');
  const confirmPasswordInput = document.querySelector('#confirm-password');

  const name = nameInput.value;
  const email = emailInput.value;
  const password = passwordInput.value;
  const confirmPassword = confirmPasswordInput.value;

  let errorMessage = '';

  if (name === '') {
    errorMessage += 'Please enter your name.<br>';
  }

  if (email === '') {
    errorMessage += 'Please enter your email address.<br>';
  } else if (!isValidEmail(email)) {
    errorMessage += 'Please enter a valid email address.<br>';
  }

  if (password === '') {
    errorMessage += 'Please enter a password.<br>';
  } else if (password !== confirmPassword) {
    errorMessage += 'Passwords do not match.<br>';
  }

  if (errorMessage === '') {
    output.innerHTML = `<p>Thank you for registering, ${name}!</p>`;
    form.reset();
  } else {
    output.innerHTML = `<p id="error">${errorMessage}</p>`;
  }
});

function isValidEmail(email) {
  // Regular expression for email validation
  const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return regex.test(email);
}
```

Output:

10. Practical Related Question

- How would you summarize the main ideas presented on this webpage, and apply your knowledge of HTML, CSS, and JavaScript to design a webpage that incorporates these ideas?
- Can you explain the key features of a well-designed webpage, and modify an existing webpage to incorporate these features using your knowledge of HTML, CSS, and JavaScript.
- What is the purpose of using HTML, CSS, and JavaScript to create a webpage, and how would you apply this knowledge to design a webpage that meets specific user needs?
- How would you use JavaScript to add interactivity and dynamic features to a webpage, and create a webpage that incorporates these features using HTML and CSS?
- Can you troubleshoot and fix common issues that arise when designing and developing webpages, and apply this knowledge to create a webpage that is accessible, responsive, and optimized for performance?

11. Exercise Questions :

- Create a navigation menu using HTML and style it using CSS. Add JavaScript to make the menu responsive and toggle between mobile and desktop views.
- Create a contact form using HTML and style it using CSS. Use JavaScript to validate the form and prevent submission if required fields are empty.
- Create a photo gallery using HTML, CSS, and JavaScript. Add functionality to allow users to filter and sort images based on different criteria.
- Use JavaScript for develop bill generation report.
- Use JavaScript to develop game application.

PRACTICAL 2:

● Develop Dynamic Webpage Using PHP Script	CO2
● Develop PHP application with Database connection	CO2

1. Practical significance :

The practical significance of the two lab programs, namely the development of a dynamic webpage using PHP script and a PHP application with a database connection, lies in their widespread use in web development. These programs enable website developers to create and manage user data, implement authentication and authorization functionality, and allow users to upload and share files, all of which are essential components of modern websites.

By mastering these programs, developers can build powerful and functional web applications that meet the needs of users in various domains, including e-commerce, social media, and file-sharing platforms.

2. Relevant program outcomes:

CO	(PO)	(PSO)
CO-2	1,2,3,5,6,8,10,11,12	1

3. Competency and Practical Skills:

Competencies: Understanding the basics of web development, working with server-side scripting languages, implementing authentication and session management, developing dynamic web pages, understanding of database management systems, working with SQL queries and database connections, managing user data, implementing like and comment functionality, building scalable web applications.

Practical Skills: Creating user login and registration forms, implementing secure session management, creating and connecting to a database, retrieving and storing user data, uploading and storing files on the server, implementing like and comment functionality, managing user-uploaded files, using SQL queries to interact with a database, developing dynamic web pages using PHP, integrating front-end and back-end components of a web application, building a scalable web application.

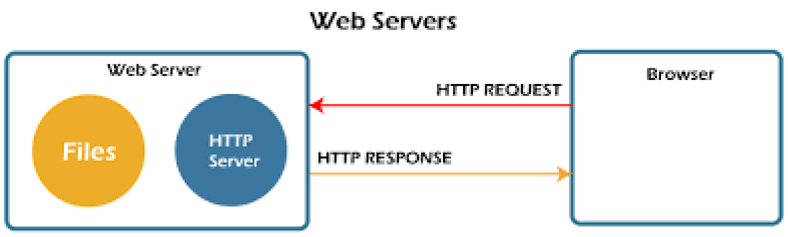
By combining the competencies and practical skills of both lab programs, students can gain a comprehensive understanding of web development and database management. They will be able to build secure and scalable web applications that allow users to interact with the website by uploading and sharing their content while managing and storing the data securely. This combination of skills is highly valued in the web development industry and is essential for anyone who wants to pursue a career in this field.

4. Prerequisites:

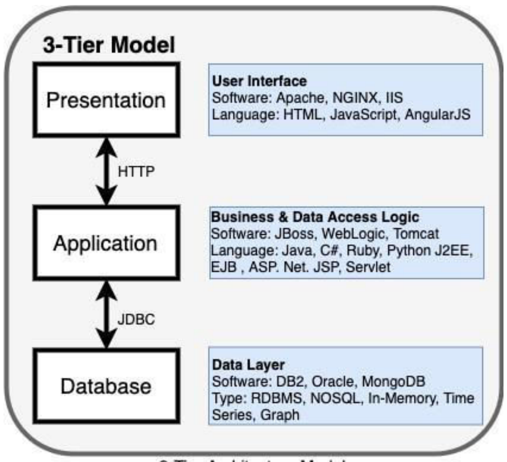
To effectively complete the lab programs of developing a dynamic webpage using PHP script and a PHP application with a database connection, students should have a basic understanding of the following concepts:

Web Development: Students should have an understanding of the basics of web development, including HTML, CSS, and JavaScript. They should be familiar with client-side scripting and the structure of a web page.
Server-Side Scripting: Students should have a basic understanding of server-side scripting languages such as PHP, including variables, control structures, and functions.
Database Management: Students should have a basic understanding of database management systems, including data types, tables, and SQL queries.
Object-Oriented Programming: Although not strictly required, knowledge of object-oriented programming concepts and techniques can be useful for more advanced programming tasks.

Overall, students should have a solid foundation in web development, programming, and database management concepts to successfully complete these lab programs. Having experience with other programming languages or web development frameworks can also be beneficial but is not strictly necessary.
Client Server Communication:



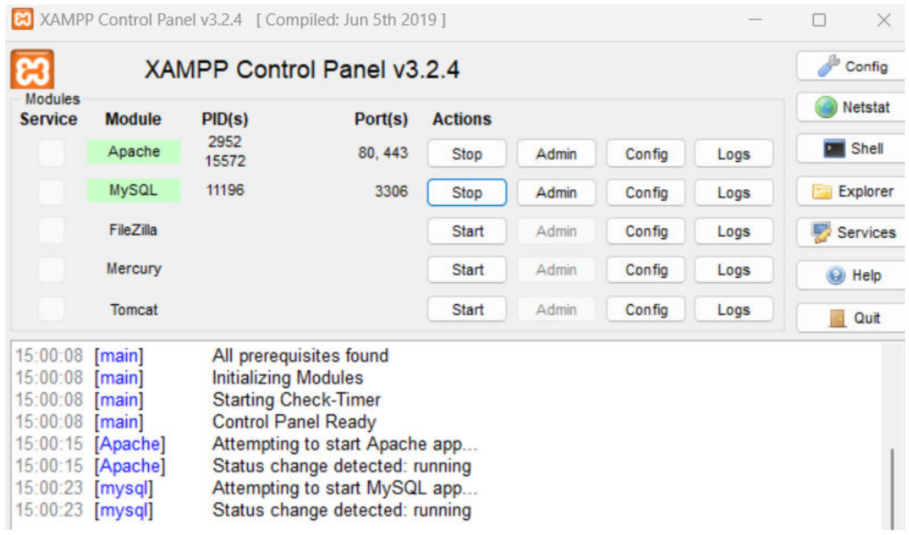
3 tier Architecture:


3-Tier Model
Presentation
HTTP
Application
JDBC
Database
User Interface
Software: Apache, NGINX, IIS
Language: HTML, JavaScript, AngularJS
Business & Data Access Logic
Software: JBoss, WebLogic, Tomcat
Language: Java, C#, Ruby, Python J2EE, EJB , ASP. Net. JSP, Servlet
Data Layer
Software: DB2, Oracle, MongoDB
Type: RDBMS, NOSQL, In-Memory, Time Series, Graph
3-Tier Architecture Model

XAMPP is a free, open-source web server solution that includes the Apache HTTP Server, MySQL database, and PHP scripting language. It allows developers to create a local web server environment on their computer for testing and developing web applications.

Here are some of the key features and benefits of using XAMPP:

1. Cross-Platform: XAMPP is available for Windows, macOS, and Linux, making it a convenient choice for developers working on different operating systems.
2. Easy Installation: XAMPP is easy to install and configure, and it includes all the components needed to set up a web server environment.
3. Local Web Server: XAMPP allows developers to create a local web server environment on their computer, which can be used for testing and developing web applications without the need for an internet connection.
4. Database Integration: XAMPP includes MySQL, which is a popular open-source database management system, making it easy to develop and test applications that use a database.
5. PHP Support: XAMPP includes PHP, a popular scripting language for web development, which allows developers to test and develop PHP applications on their local web server environment.


XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]
XAMPP Control Panel v3.2.4
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	2952 15572	80, 443	Stop	Admin	Config	Logs
	MySQL	11196	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
15:00:08  [main]     All prerequisites found
15:00:08  [main]     Initializing Modules
15:00:08  [main]     Starting Check-Timer
15:00:08  [main]     Control Panel Ready
15:00:15  [Apache]   Attempting to start Apache app...
15:00:15  [Apache]   Status change detected: running
15:00:23  [mysql]    Attempting to start MySQL app...
15:00:23  [mysql]    Status change detected: running
```

5.Resources Required:

Software Requirements	Description
Web Server	A web server software such as Apache or Nginx
PHP	PHP 5.6 or above installed on the server
MySQL	MySQL database management system installed on the server
Text Editor	A text editor for writing and editing PHP and HTML files

Hardware Requirements	Description
Computer	A computer with a modern operating system such as Windowsor Linux
RAM	At least 2 GB of RAM
Storage	Sufficient storage space for the web server, PHP, MySQL
Internet Connection	An internet connection with sufficient speed for uploading and downloading files

6. Precautions:

- **Backup and version control**: It is important to back up all files and code being worked on in case of hardware or software failure. Version control should also be implemented to track changes made to the code and to facilitate collaboration.
- **Security:** Lab computers should be secure, with up-to-date anti-virus software and firewalls in place to protect against malware and unauthorized access.
- **Data privacy**: It is important to ensure that any personal or sensitive data is properly secured and protected. This includes ensuring that login credentials are not shared with others and that any data collected is handled in accordance with relevant privacy laws and regulations.
- **Network security**: Lab computers should be connected to a secure network, and access to the internet and other network resources should be restricted to prevent unauthorized access or downloading of malicious software.
- **Safe coding practices**: Lab students should be encouraged to follow safe coding practices, such as commenting code, testing code thoroughly, and avoiding potential security vulnerabilities such as SQL injection or cross-site scripting.
- **Proper lab environment**: The lab should be a safe and clean environment, with proper ventilation and adequate lighting.

7. Algorithm/circuit/Diagram/Description:

- Start the web server and open the code editor.
- Create a new PHP file and save it with a .php extension.
- Add the PHP opening tag <?php at the beginning of the file.
- Establish a connection to the MySQL database using PHP code.
- Write PHP code to retrieve and display user-uploaded files on the web page.
- Implement a form to allow users to like or comment on a file.
- Write PHP code to store the like and comment information in the database.
- Implement a user authentication system to ensure only authenticated users can like or comment on files.

8. Test cases:

- Test if the login and logout functionality is working properly.

- Test if the file upload functionality is working properly and validating the input correctly.
- Test if the file upload information is properly stored in the database.
- Test if the user-specific content is properly displayed on the main page.
- Test if the session variables are properly set and destroyed.

9. Sample Code :

- Set up a web server with PHP and MySQL installed.
- Create a database in MySQL to store user information and uploaded files information.
- Create a PHP script to handle user registration and login. Use session variables to keep track of the user's login status.
- Create a PHP script to handle file uploads. This script should check if the user is logged in and has the appropriate permissions to upload files.
- Store uploaded files in a directory on the server.
- Create a PHP script to display uploaded files. This script should retrieve file information from the database and display it on the web page.
- Allow users to like and comment on uploaded files. Create a form for users to submit comments and likes. Store this information in the database along with the uploaded file information.
- Display comments and likes on the web page with the uploaded files.

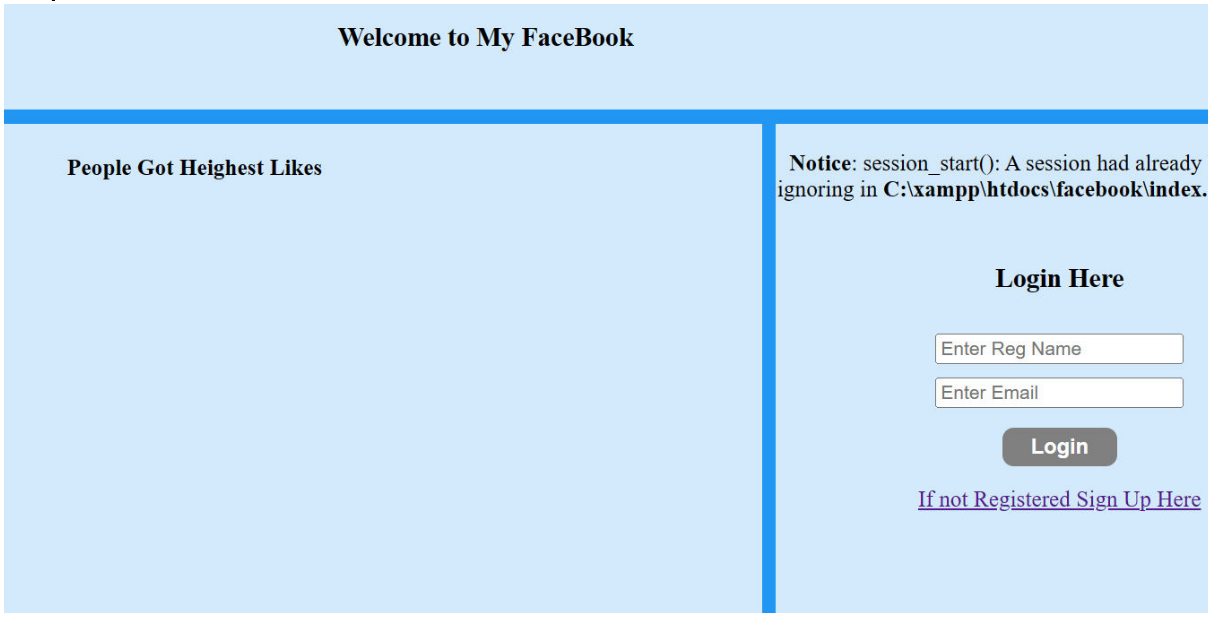
PHP Code:

```
<?php
session_start();
if (!isset($_SESSION['username'])) {
  header('Location: login.php');
  exit;
}
if (isset($_POST['submit'])) {
  $filename = $_FILES['file']['name'];
  $filesize = $_FILES['file']['size'];
  $filetype = $_FILES['file']['type'];
  $filetemp = $_FILES['file']['tmp_name'];
  $filedesc = $_POST['desc'];

  // Check if file is uploaded successfully
  if (move_uploaded_file($filetemp, 'uploads/' . $filename)) {
    // Store file information in database
    $conn = mysqli_connect('localhost', 'username', 'password', 'database');
    $query = "INSERT INTO files (filename, filesize, filetype, filedesc) VALUES ('$filename',
'$filesize', '$filetype', '$filedesc')";
    mysqli_query($conn, $query);
  }
}
$conn = mysqli_connect('localhost', 'username', 'password', 'database');
$query = "SELECT * FROM files";
$result = mysqli_query($conn, $query);

while ($row = mysqli_fetch_assoc($result)) {
  echo '<h2>' . $row['filename'] . '</h2>';
  echo '<p>' . $row['filedesc'] . '</p>';
  $fileid = $row['fileid'];
  $query2 = "SELECT COUNT(*) as likes FROM likes WHERE fileid = $fileid";
  $result2 = mysqli_query($conn, $query2);
  $row2 = mysqli_fetch_assoc($result2);
  $likes = $row2['likes'];
  echo '<p>' . $likes . ' likes</p>';
  $query3 = "SELECT * FROM comments WHERE fileid = $fileid";
  $result3 = mysqli_query($conn, $query3);
  while ($row3 = mysqli_fetch_assoc($result3)) {
    echo '<p>' . $row3['comment'] . '</p>';
  }
  echo '<form method="post" action="comment.php">';
  echo '<input type="hidden" name="fileid" value="' . $fileid . '">';
  echo '<textarea name="comment"></textarea>';
  echo '<input type="submit" name="submit" value="Comment">';
  echo '</form>';
  echo '<form method="post" action="like.php">';
  echo '<input type="hidden" name="fileid" value="' . $file
```

CREATE TABLE users (userid INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(50) NOT NULL, email VARCHAR(100) NOT NULL, password VARCHAR(255) NOT NULL);	CREATE TABLE files (fileid INT AUTO_INCREMENT PRIMARY KEY, filename VARCHAR(255) NOT NULL, filesize INT NOT NULL, filetype VARCHAR(50) NOT NULL, filedesc VARCHAR(255) NOT NULL,

	userid INT NOT NULL, FOREIGN KEY (userid) REFERENCES users(userid));
CREATE TABLE likes (likeid INT AUTO_INCREMENT PRIMARY KEY, userid INT NOT NULL, fileid INT NOT NULL, FOREIGN KEY (userid) REFERENCES users(userid), FOREIGN KEY (fileid) REFERENCES files(fileid));	CREATE TABLE comments (commentid INT AUTO_INCREMENT PRIMARY KEY, userid INT NOT NULL, fileid INT NOT NULL, comment VARCHAR(255) NOT NULL, FOREIGN KEY (userid) REFERENCES users(userid), FOREIGN KEY (fileid) REFERENCES files(fileid));

Output:

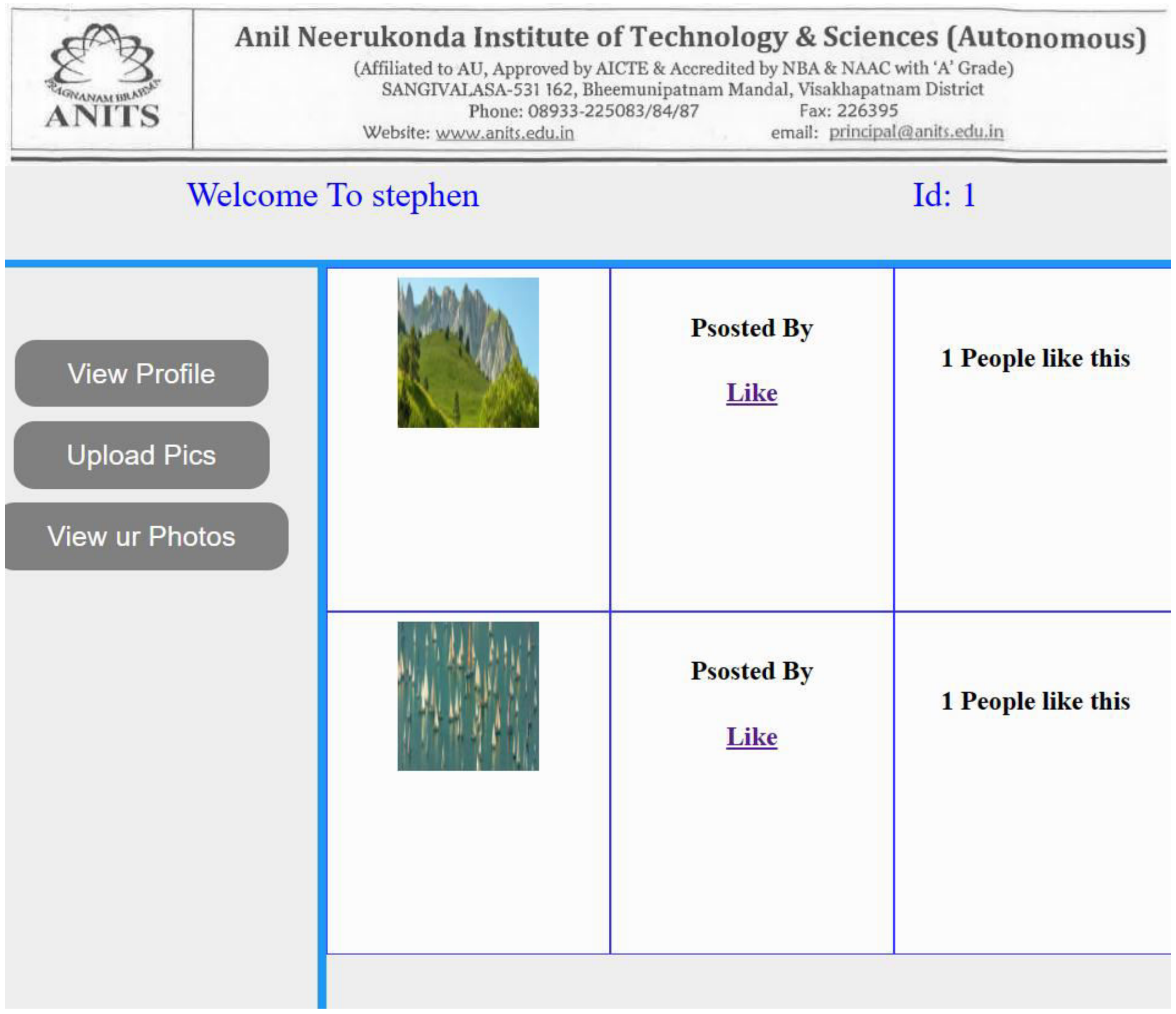


Welcome to My FaceBook

People Got Heighest Likes

Notice: session_start(): A session had already
ignoring in **C:\xampp\htdocs\facebook\index.**

Login Here


Anil Neerukonda Institute of Technology & Sciences (Autonomous)

(Affiliated to AU, Approved by AICTE & Accredited by NBA & NAAC with 'A' Grade)
SANGIVALASA-531 162, Bheemunipatnam Mandal, Visakhapatnam District
Phone: 08933-225083/84/87 Fax: 226395
Website: www.anits.edu.in email: principal@anits.edu.in

Welcome To stephen Id: 1

	Psosted By Like	1 People like this
	Psosted By Like	1 People like this

10. Practical Related Question

- How do I store the user's like for a particular file in the database?
- How do I prevent a user from liking a file more than once?
- How do I retrieve the number of likes for a particular file from the database?
- How do I display the number of likes for a file on the main page?
- How do I store a user's comment for a particular file in the database?
- How do I retrieve all comments for a particular file from the database?
- How do I display all comments for a file on the main page?
- How do I prevent a user from leaving an empty comment?
- How do I limit the length of a comment to a certain number of characters?
- How do I handle errors when adding likes or comments to the database?

11. Exercise Questions :

- Create a login page that allows users to log in with their username and password. Use sessions to keep track of the logged-in user.
- Create a file upload form that allows logged-in users to upload files to the server. Make sure to validate the file type and size before allowing the upload.
- Create a page that displays all uploaded files along with their name, description, likes, and comments. Allow users to like and comment on each file.
- Implement the functionality to store likes and comments for each file in the database using MySQL.
- Add a search bar to the main page that allows users to search for files by name or description.

PRACTICAL EXPERIMENT 3:

● Implement Modules in NodeJS	CO3
● Develop mini application using Express and NodeJS	CO3

1. Practical significance :

1. OS Module:

a) System Information: With the OS module, you can easily retrieve information about the system you are running on, such as the CPU, memory, and disk usage. This information can be useful for monitoring system resources, identifying performance bottlenecks, and optimizing system performance.

b) File System: The OS module also provides functionality for interacting with the file system, such as creating, reading, and deleting files and directories. This can be useful for managing files and directories, such as backing up data, synchronizing files across multiple systems, and performing batch processing tasks.

c) Processes: The OS module also provides functionality for working with processes, such as creating, killing, and managing processes. This can be useful for managing system resources, such as limiting the number of processes running simultaneously, or for controlling the behavior of long-running processes.

2. **Network Module:**

a) Socket Programming: With the Network module, you can create and manage network connections, send and receive data over the network, and handle network events. This can be useful for building networked applications, such as web servers, chat applications, and real-time data streaming applications.

b) Remote Procedure Calls: The Network module also provides functionality for making remote procedure calls (RPCs) over the network, which can be useful for building distributed applications that run across multiple systems.

c) Security: The Network module provides functionality for securing network connections, such as encrypting data transmissions, authenticating users, and validating certificates. This can be useful for building secure applications that require data protection, such as online banking, e-commerce, and data storage applications.

Overall, the OS and Network modules in Node.js provide a wide range of functionality for managing system resources, working with network resources, and building secure and scalable applications. By leveraging these modules, developers can build more efficient and reliable applications, and reduce the complexity of managing these resources themselves.

2.Relevant program outcomes:

CO	(PO)	(PSO)
CO-3	1,2,3,5,6,8,10,11,12	1,2

3. Competency and Practical Skills:

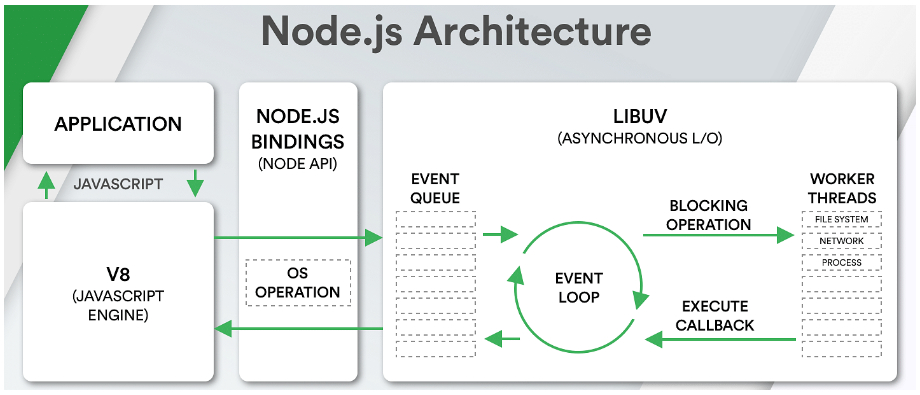
1. JavaScript Fundamentals: Node.js and Express.js are both built using JavaScript, so it is essential to have a strong understanding of JavaScript fundamentals, such as variables, data types, functions, and control structures.

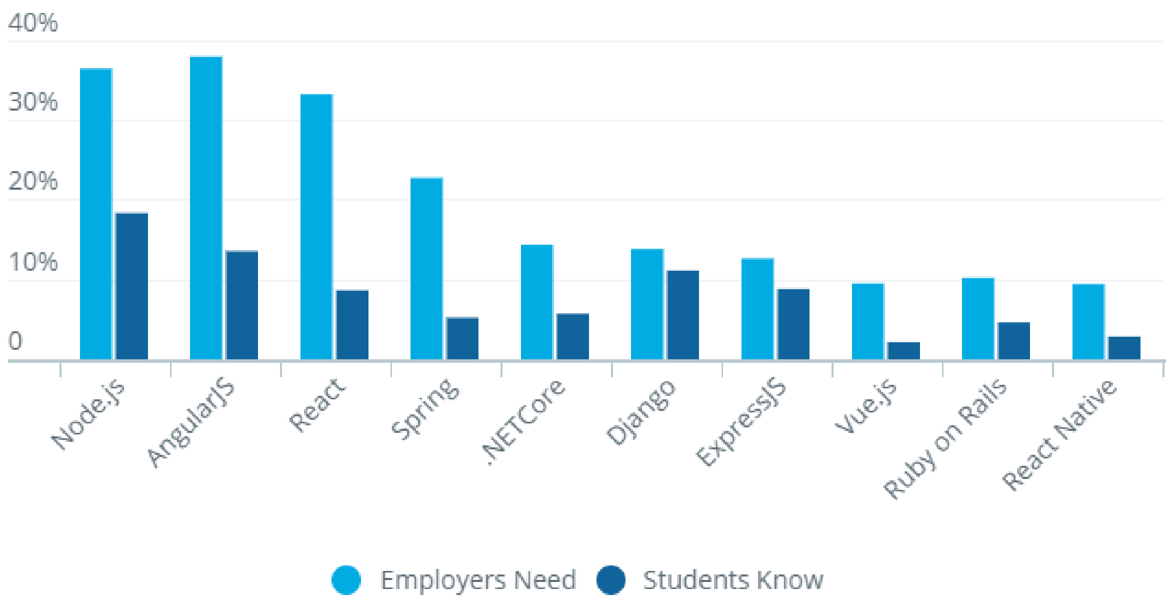
2. Asynchronous Programming: Node.js is known for its asynchronous programming model, which enables developers to write non-blocking code that can handle multiple concurrent requests. Practical skills in asynchronous programming are critical to building scalable and efficient applications using Node.js.
3. Web Development: Express.js is a popular framework for building web applications in Node.js, so practical skills in web development are essential, including knowledge of HTTP protocols, RESTful API design, and handling request and response objects.
4. Server-Side Development: Node.js is primarily used for server-side development, so practical skills in server-side programming are essential, including knowledge of databases, server-side caching, and authentication and authorization mechanisms.
5. Package Management: Node.js uses npm (Node Package Manager) for package management, so practical skills in package management, including installing, updating, and configuring packages, are essential.

4. Prerequisites:

Node.js and Express.js are two popular open-source frameworks used for building server-side web applications in JavaScript.

Express.js is a web application framework built on top of Node.js, which provides a set of features and tools for building web applications and APIs.





5.Resources Required:

Software Requirements:

Software	Version
Node.js	v14.x or higher
NPM	v6.x or higher
Express.js	v4.x or higher
MongoDB	v4.x or higher

Hardware Requirements:

Hardware	Specification
Processor	Dual-core or higher
RAM	8 GB or higher
Storage	256 GB SSD or higher
Operating System	Windows 10, macOS, or Linux
Internet Connection	Broadband or higher

6. Precautions:

- **Backup and version control**: It is important to back up all files and code being worked on in case of hardware or software failure. Version control should also be implemented to track changes made to the code and to facilitate collaboration.
- **Security:** Lab computers should be secure, with up-to-date anti-virus software and firewalls in place to protect against malware and unauthorized access.
- **Data privacy**: It is important to ensure that any personal or sensitive data is properly secured and protected. This includes ensuring that login credentials are not shared with others and that any data collected is handled in accordance with relevant privacy laws and regulations.
- **Network security**: Lab computers should be connected to a secure network, and access to the internet and other network resources should be restricted to prevent unauthorized access or downloading of malicious software.
- **Safe coding practices**: Lab students should be encouraged to follow safe coding practices, such as commenting code, testing code thoroughly, and avoiding potential security vulnerabilities such as SQL injection or cross-site scripting.
- **Proper lab environment**: The lab should be a safe and clean environment, with proper ventilation and adequate lighting.

7. Algorithm:

Developing Mini Application using Express and NodeJS:

1. Install Node.js on your system.
2. Open your command prompt or terminal and navigate to your project directory.
3. Create a new directory for your application.
4. Navigate to the new directory and initialize a new Node.js project using the **npm init** command.
5. Install the Express framework using the **npm install express** command.
6. Create a new file with the .js extension for your application.
7. Import the Express module using the **require()** function and store it in a variable.
8. Create a new Express application using the **express()** function and store it in a variable.
9. Define your application routes and their corresponding handlers.
10. Start the server by calling the **listen()** function on the Express application object with a specified port number.

8. Test Cases:

1. **Implement Http methods using Flask:**

- Test that the GET method returns the correct data from the server.
- Test that the POST method correctly adds new data to the server.
- Test that the PUT method updates existing data on the server.
- Test that the DELETE method removes the specified data from the server.
- Test that the server returns the appropriate status codes for each method.

2. **Implement Sessions using Flask:**

- Test that session variables are correctly set and retrieved.
- Test that session data is preserved across multiple requests.
- Test that session data is deleted when the user logs out or the session expires.
- Test that session data cannot be accessed by unauthorized users.

3. **Develop CRUD Application using Flask and MongoDB:**

- Test that the Create operation adds new data to the database.
- Test that the Read operation correctly retrieves data from the database.
- Test that the Update operation modifies existing data in the database.
- Test that the Delete operation removes the specified data from the database.
- Test that the server returns the appropriate status codes for each operation.
- Test that the application handles errors or exceptions that may occur during database operations.

9. Sample Code:

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const session = require('express-session');

app.set('views', './views');
app.set('view engine', 'ejs');
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session(
{
  secret:"2011",
  resave:false,
  saveUninitialized:false
}
));
app.get('/', function(req, res) {
    res.sendFile(__dirname +'/public'+ '/index.html');
```

```
app.post('/login', (req, res) => {
    const { username, password } = req.body;
    if (username === 'admin' && password ===
'admin') {
      req.session.username = username;
      res.redirect('/dashboard');
    } else {
      res.redirect('/login?error=Invalid username
or password');
    }
  });
  app.get('/logout', (req, res) => {
    // Clear session data
    req.session.destroy();
    res.send('Logout successful!');
  });
  app.get('/dashboard', function(req, res){
    res.render('dashboard', { session:
});
app.get('/about', (req, res) => {
  res.render('about');
});

app.get('/register', (req, res) => {
  res.render('register');
});
app.get('/login',(req,res)=>{
  res.render('login')
});
```

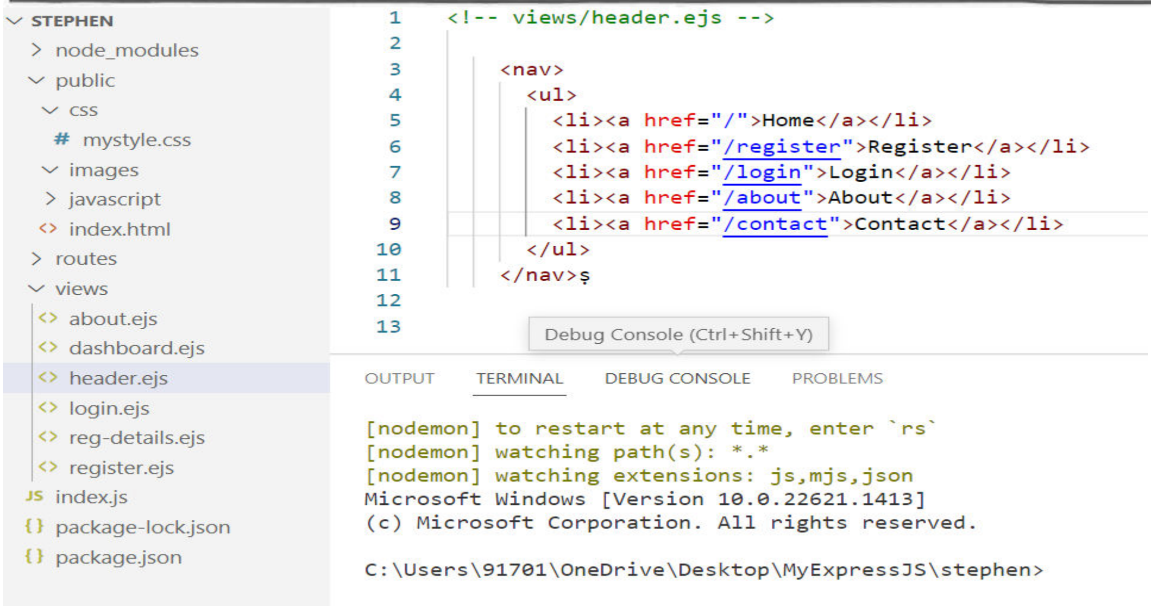
```
req.session });
  });
  app.post('/register', (req, res) => {
    const { name, email, password } = req.body;

    res.render('reg-details', { name, email,
password });
  });
app.listen(5000,()=>console.log("Server is
Running"));
```

Package.json

```
{
  "name": "expressjs",
  "version": "1.0.0",
  "description": "first application",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "server": "nodemon index.js"
  },
  "author": "Stephen",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-session": "^1.17.3",
    "mongodb": "^5.1.0",
    "mongoose": "^7.0.3"
  },
  "devDependencies": {
    "nodemon": "^2.0.21"
  }
}
```

Index.html

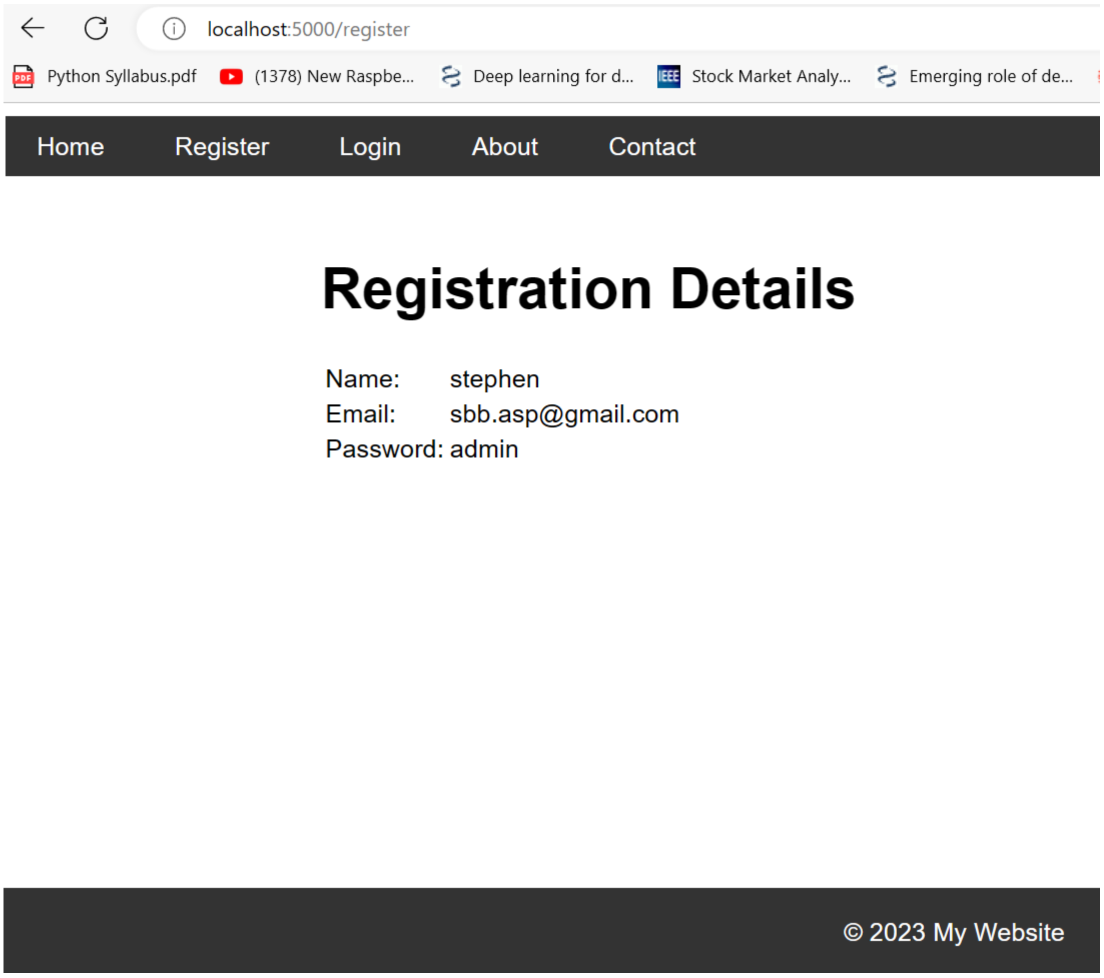
```
<html>
<head>
  <title>My Website</title>
  <link rel="stylesheet" type="text/css"
href="/css/mystyle.css">
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/register">Register</a></li>
        <li><a href="/login">Login</a></li>
        <li><a href="/about">About</a></li>
        <li><a href="/contact">Contact</a></li>
      </ul>
    </nav>
  </header>
    <main>
      <h3>Welcome to My Site..</h3>
    </main>
  <footer>
    <p>Copyright © 2023 My Website</p>
  </footer>
</body>
</html>
```



```
<!-- views/header.ejs -->

  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/register">Register</a></li>
      <li><a href="/login">Login</a></li>
      <li><a href="/about">About</a></li>
      <li><a href="/contact">Contact</a></li>
    </ul>
  </nav>ş
```

```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91701\OneDrive\Desktop\MyExpressJS\stephen>
```

9. Output:



Registration Details

Name: stephen
Email: sbb.asp@gmail.com
Password: admin

© 2023 My Website

10.Practical related Questions:

1. How can you use Express and NodeJS to develop a mini application for managing a database of customer information?
2. What are some key features of Express and NodeJS that you would use to develop a real-time chat application?
3. How would you design and implement a RESTful API using Express and NodeJS for a web-based project management tool?
4. What are the key components of an Express application and how would you leverage them to create a custom e-commerce website?
5. How would you use NodeJS and Express to create a mini application for collecting and analysing data from IoT devices?

PRACTICAL EXPERIMENT 4:

● Implement Http methods using Flask	CO4
● Implement Sessions using Flask	CO4
● Develop CRUP Application using Flask and MongoDB	CO4

1. Practical significance :

The three topics you mentioned - implementing HTTP methods, implementing sessions, and developing a CRUD application using Flask and MongoDB - have practical significance in web development.

1. Implementing HTTP methods using Flask: HTTP methods are essential for web development because they enable communication between the client and server. Flask is a micro web framework that allows you to implement HTTP methods such as GET, POST, PUT, DELETE, etc. By implementing these methods, you can perform various operations such as retrieving data, submitting data, updating data, and deleting data. This is essential for building web applications that can interact with databases, APIs, and other web services.
2. Implementing sessions using Flask: Sessions are used to store user-specific data on the server side. This is essential for building web applications that require authentication, such as user login and user registration. Flask provides a built-in session management system that allows you to create, read, update, and delete session variables. By implementing sessions, you can securely store user data such as login credentials, user preferences, and shopping cart items.
3. Developing CRUD application using Flask and MongoDB: CRUD stands for Create, Read, Update, and Delete, which are the four basic operations that can be performed on a database. Flask is a web framework that provides a lightweight and flexible approach to building web applications, and MongoDB is a popular NoSQL database that is widely used for building scalable and flexible applications. By combining Flask and MongoDB, you can easily develop a CRUD application that can perform these basic database operations. This is essential for building web applications that require data persistence, such as e-commerce sites, social media platforms, and content management systems.

2.Relevant program outcomes:

CO	(PO)	(PSO)
CO-4	1,2,3,5,6,8,10,11,12	1,2

3. Competency and Practical Skills:

1. Implementing HTTP methods using Flask: To implement HTTP methods using Flask, you need to have a good understanding of HTTP protocols and the Flask web framework. You should be familiar with the various HTTP methods such as GET, POST, PUT, and DELETE, and how they are used in web development. Additionally, you should be proficient in using Flask's routing system to map URLs to Python functions that handle HTTP requests. Practical skills related to this topic include creating routes for handling different HTTP methods, parsing HTTP request data, and sending HTTP responses.
2. Implementing sessions using Flask: To implement sessions using Flask, you need to have a good understanding of session management and security principles. You should be familiar with how sessions work in web applications, including how session data is stored and retrieved, and how to manage session lifetimes. Additionally, you should be proficient in using Flask's session management system, which provides methods for creating, reading, updating, and deleting session variables. Practical skills related to this topic include creating and managing session variables, implementing session-based authentication, and securing session data.
3. Developing CRUD application using Flask and MongoDB: To develop a CRUD application using Flask and MongoDB, you need to have a good understanding of database management and web development principles. You should be familiar with the basic CRUD operations and how they are performed in MongoDB. Additionally, you should be proficient in using Flask's web development tools, such as Flask-SQLAlchemy, Flask-WTF, and Flask-RESTful, which provide a convenient way to interact with databases and build web APIs. Practical skills related to this topic include creating database models and tables, handling database queries, and implementing RESTful API endpoints for performing CRUD operations.

4. Prerequisites:

To effectively learn and understand the topics you mentioned, there are some prerequisites you should have:

1. Knowledge of Python programming language: Flask is a Python web framework, so you should have a good understanding of the Python programming language. You should be familiar with basic programming concepts such as variables, data types, loops, and functions. Additionally, you should be familiar with object-oriented programming principles and how to work with Python modules and packages.
2. Understanding of web development concepts: To work with Flask, you should have a good understanding of web development concepts such as HTTP protocols, URLs,

and web APIs. You should be familiar with the client-server model and how web applications communicate with servers using HTTP requests and responses.

3. Basic understanding of databases: To work with MongoDB, you should have a basic understanding of databases and how they work. You should be familiar with concepts such as tables, fields, and records, and how to perform basic CRUD operations.
4. Familiarity with HTML, CSS, and JavaScript: Although Flask is a server-side framework, you will still need to have a basic understanding of HTML, CSS, and JavaScript to build web pages and user interfaces. You should be familiar with basic HTML tags and how to use CSS to style web pages. Additionally, you should be familiar with JavaScript and how it is used to add interactivity to web pages.

By having these prerequisites, you will be better equipped to learn and understand the topics you mentioned and to build robust and scalable web applications using Flask and MongoDB.

Here are the steps to create an account in MongoDB Atlas:

1. Go to the MongoDB Atlas website (https://www.mongodb.com/cloud/atlas) and click on the "Sign up" button in the top-right corner.
2. Enter your email address and create a password. You can also sign up with your Google or Github account.
3. Select the type of organization you're signing up for: "Personal" or "Team". If you're signing up for a team, you'll need to provide additional information about your organization.
4. Fill in your personal information, including your name and country.
5. Review the Terms of Service and Privacy Policy, and click on the "Create Account" button.
6. You'll be taken to the MongoDB Atlas dashboard, where you can create your first project. Click on the "Create a New Project" button to get started.
7. Give your project a name, select your preferred cloud provider and region, and click on the "Create Project" button.
8. Once your project is created, you can start creating clusters and adding data to your database.

Top of Form

5.Resources Required:

Software Requirements	**Hardware Requirements**
Python 3.6 or higher	1.6 GHz or faster processor
Flask 2.0 or higher	4 GB RAM
MongoDB 4.4 or higher	1 GB available disk space
Pymongo 3.11 or higher	
Text editor or IDE : VS Code	
Web browser : Chrome	

6. Precautions:

- **Backup and version control**: It is important to back up all files and code being worked on in case of hardware or software failure. Version control should also be implemented to track changes made to the code and to facilitate collaboration.
- **Security:** Lab computers should be secure, with up-to-date anti-virus software and firewalls in place to protect against malware and unauthorized access.
- **Data privacy**: It is important to ensure that any personal or sensitive data is properly secured and protected. This includes ensuring that login credentials are not shared with others and that any data collected is handled in accordance with relevant privacy laws and regulations.
- **Network security**: Lab computers should be connected to a secure network, and access to the internet and other network resources should be restricted to prevent unauthorized access or downloading of malicious software.
- **Safe coding practices**: Lab students should be encouraged to follow safe coding practices, such as commenting code, testing code thoroughly, and avoiding potential security vulnerabilities such as SQL injection or cross-site scripting.
- **Proper lab environment**: The lab should be a safe and clean environment, with proper ventilation and adequate lighting.

7. Algorithm:

1. Implement Http methods using Flask:

- Step 1: Import Flask and create an instance of the Flask class.
- Step 2: Define a function for each HTTP method (GET, POST, PUT, DELETE).
- Step 3: Map the function to the appropriate URL using the app.route() decorator.

- Step 4: Inside the function, handle the request data as needed (e.g., query parameters, form data, JSON).
- Step 5: Return a response object with the appropriate status code and data.

2. Implement Sessions using Flask:

- Step 1: Import Flask and create an instance of the Flask class.
- Step 2: Use the Flask session object to store and retrieve session data.
- Step 3: Use the session data in your application to personalize the user experience (e.g., display user-specific content).
- Step 4: Set session variables based on user input or other application data.
- Step 5: Clear session variables when they are no longer needed.

3. Develop CRUP Application using Flask and MongoDB:

- Step 1: Import Flask and PyMongo libraries and create an instance of the Flask class.
- Step 2: Set up a connection to the MongoDB database using PyMongo.
- Step 3: Define a function for each CRUD operation (Create, Read, Update, Delete).
- Step 4: Map the function to the appropriate URL using the app.route() decorator.
- Step 5: Inside the function, perform the corresponding MongoDB operation (e.g., insert_one(), find_one(), update_one(), delete_one()).
- Step 6: Handle any errors or exceptions that may occur during the database operation.
- Step 7: Return a response object with the appropriate status code and data.

These are just high-level steps, and the actual implementation may vary depending on the specific requirements of the application.

9.Sample Code

```
from flask import Flask, render_template, request, redirect, url_for
from pymongo import MongoClient
app = Flask(__name__)

# Connect to the MongoDB database
```

```
client = MongoClient('mongodb+srv://<username>:<password>@<cluster-
address>/test?retryWrites=true&w=majority')

db = client['shopping_cart']
collection = db['items']
@app.route('/')

def index():
   return render_template('index.html')
@app.route('/add_item', methods=['GET', 'POST'])

def add_item():
   if request.method == 'POST':
      name = request.form['name']
      price = request.form['price']
      quantity = request.form['quantity']
      # Validate user input
      if name and price and quantity:
         item = {
            'name': name,
            'price': price,
            'quantity': quantity
         }
         # Insert the new item into the MongoDB database
         collection.insert_one(item)
         return redirect(url_for('shopping_cart'))
   return render_template('add_item.html')


@app.route('/shopping_cart')

def shopping_cart():
   # Retrieve all items from the MongoDB database
   items = collection.find()
   return render_template('shopping_cart.html', items=items)


@app.route('/update_item', methods=['GET', 'POST'])

def update_item():
   if request.method == 'POST':
      item_id = request.form['item_id']
      quantity = request.form['quantity']
      # Validate user input
      if item_id and quantity:
         # Update the quantity of the item in the MongoDB database
         collection.update_one({'_id': ObjectId(item_id)}, {'$set': {'quantity': quantity}})
         return redirect(url_for('shopping_cart'))
   return render_template('update_item.html')


@app.route('/delete_item', methods=['GET', 'POST'])
```

```
def delete_item():
    if request.method == 'POST':
        item_id = request.form['item_id']
        # Validate user input
        if item_id:
            # Remove the item from the MongoDB database
            collection.delete_one({'_id': ObjectId(item_id)})
            return redirect(url_for('shopping_cart'))
    return render_template('delete_item.html')


if __name__ == '__main__':

    app.run(debug=True)
```

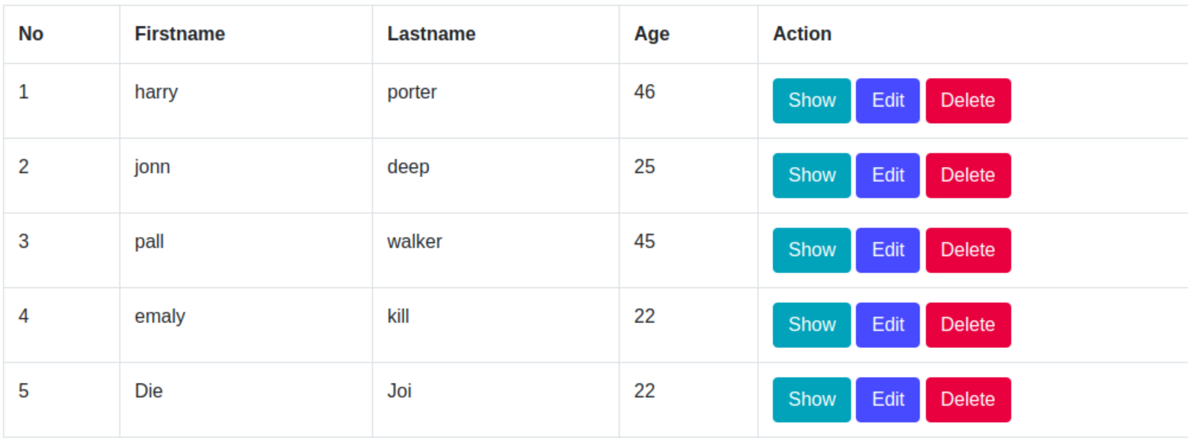
Add Item:

```
<form method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required><br>
  <label for="price">Price:</label>
  <input type="number" id="price" name="price" min="0" step="0.01" required><br>
  <label for="quantity">Quantity:</label>
  <input type="number" id="quantity" name="quantity" min="1" required><br>
  <button type="submit">Add Item</button>
</form>
```

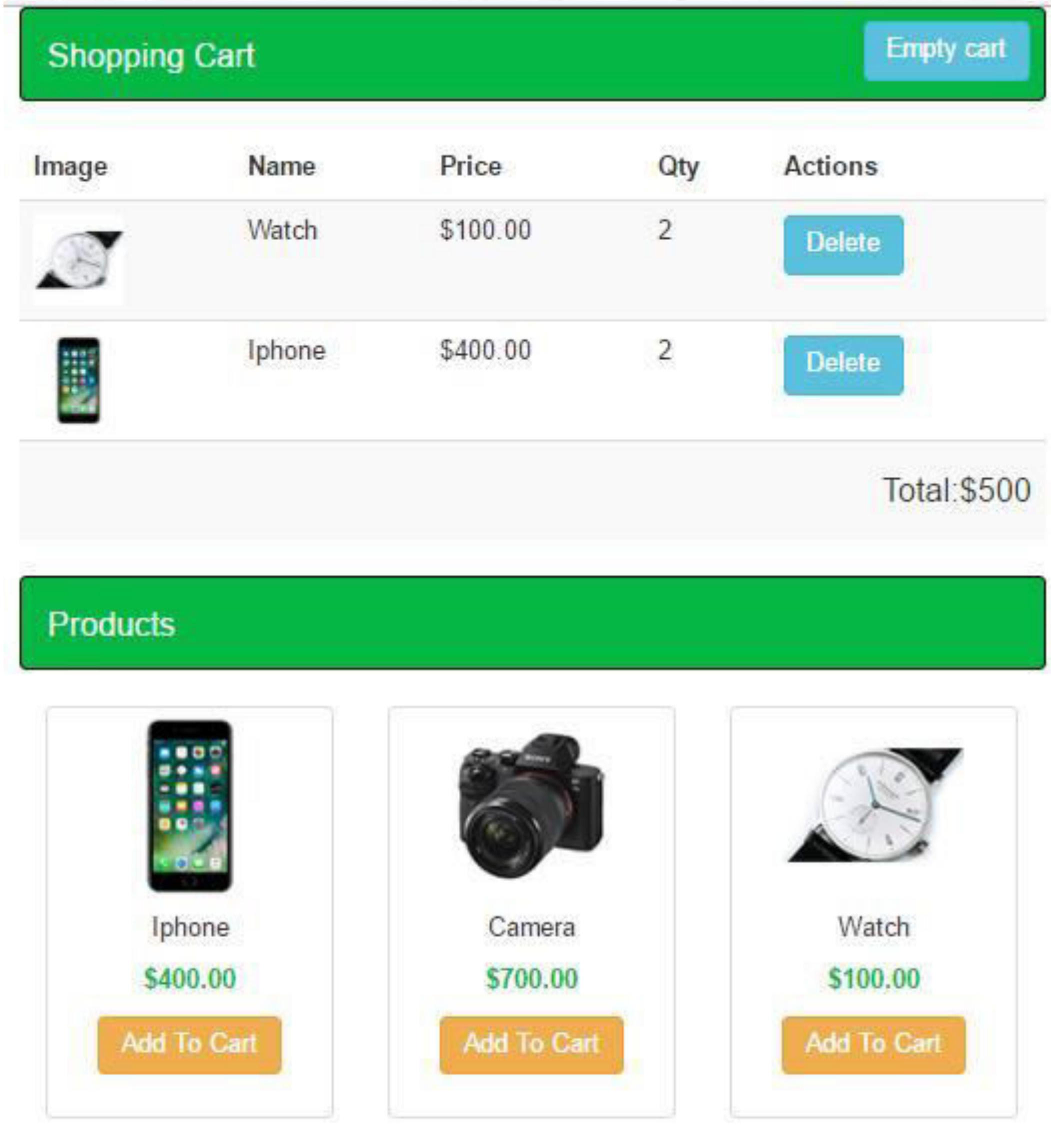
Shopping Cart Page:

```
{% if items %}
  <table>
    <tr>
      <th>Name</th>
      <th>Price</th>
      <th>Quantity</th>
      <th>Action</th>
    </tr>
    {% for item in items %}
      <tr>
        <td>{{ item['name'] }}</td>
        <td>{{ item['price'] }}</td>
        <td>{{ item['quantity'] }}</td>
        <td>
          <form method="post" action="/update_item">
            <input type="hidden" name="item_id" value="{{ item['_id'] }}">
            <input type="number" name="quantity" value="{{ item['quantity'] }}" min="1"
required>
            <button type="submit">Update</button>
          </form>
          <form method="post" action="/delete_item">
            <input type="hidden" name="item_id" value="{{ item['_id'] }}">
            <button type="submit">Delete</button>
          </form>
        </td>
      </tr>
    {% endfor %}
  </table>
{% else %}
  <p>No items in shopping cart.</p>
{% endif %}

<a href="/add_item">Add Item</a>
```

Output



No	Firstname	Lastname	Age	Action
1	harry	porter	46	Show Edit Delete
2	jonn	deep	25	Show Edit Delete
3	pall	walker	45	Show Edit Delete
4	emaly	kill	22	Show Edit Delete
5	Die	Joi	22	Show Edit Delete



Shopping Cart

Empty cart

Image	Name	Price	Qty	Actions
	Watch	$100.00	2	Delete
	Iphone	$400.00	2	Delete

Total:$500

Products

Iphone	Camera	Watch
$400.00	$700.00	$100.00
Add To Cart	Add To Cart	Add To Cart

Practical related Questions:

1. What are the security implications of using sessions in a Flask application and how can they be mitigated?
2. How can you use Flask sessions to implement user authentication and authorization?
3. How do you configure session settings such as the session timeout and session encryption in a Flask application?
4. How do you test Flask sessions in a unit testing framework such as pytest?
5. How can you use Flask sessions with AJAX requests in a single-page application?

Develop CRUP Application using Flask and MongoDB:

1. How can you implement database transactions in a Flask application that uses MongoDB?
2. How do you handle errors and exceptions when working with MongoDB in a Flask application?
3. What are some strategies for optimizing database performance in a Flask application that uses MongoDB?
4. How can you use Flask blueprints to organize and modularize your CRUD application code?
5. How do you implement data validation and input sanitization in a Flask application that uses MongoDB

Sign of Coordinator Sign of HoD,CSE

Tools and Technologies

Types of Web Browsers

1. Google Chrome: Developed by Google, Chrome is a popular browser known for its speed, simplicity, and ease of use.
2. Mozilla Firefox: Developed by the Mozilla Foundation, Firefox is a popular open-source browser known for its customizable interface and strong privacy features.
3. Microsoft Edge: Developed by Microsoft, Edge is a modern browser that comes pre-installed on Windows 10 and offers a range of features such as support for extensions and improved battery life.

4. Apple Safari: Developed by Apple, Safari is the default browser on all Apple devices and offers a range of features such as integration with Apple services and enhanced privacy features.
5. Opera: Developed by Opera Software, Opera is a popular browser known for its speed and innovative features such as built-in VPN and ad-blocker.
6. Brave: Brave is a privacy-focused browser that blocks ads and trackers by default, and offers a built-in cryptocurrency wallet.
7. Vivaldi: Vivaldi is a highly customizable browser that allows users to personalize their browsing experience by modifying the interface, keyboard shortcuts, and other settings.

There are also many other browsers available, both popular and niche, such as Tor, DuckDuckGo Privacy Browser, and Epic Privacy Browser.

Types of Web Servers

1. **Apache HTTP Server**: Apache is one of the most widely used web servers in the world, and is open-source software that can run on a variety of operating systems. It supports a wide range of features, including dynamic content generation, SSL/TLS encryption, and proxying.
2. **Nginx: Nginx** is a high-performance web server and reverse proxy server that is known for its scalability and efficiency. It is often used for serving static content, as well as for load balancing and proxying.
3. **Microsoft IIS:** Microsoft Internet Information Services (IIS) is a web server that is included with Windows Server. It supports a range of features, including SSL/TLS encryption, dynamic content generation, and load balancing.
4. **Apache Tomcat**: Apache Tomcat is a Java-based web server that is designed to run Java Servlets and JavaServer Pages (JSP). It is often used for deploying Java web applications and is known for its scalability and reliability.
5. **Node.js**: Node.js is a JavaScript runtime that can be used as a web server. It is often used for building scalable, real-time applications and is known for its event-driven architecture.
6. **Caddy**: Caddy is a modern, easy-to-use web server that supports HTTPS by default, and automatically configures SSL/TLS certificates using Let's Encrypt. It also includes a range of other features, such as HTTP/2 support and automatic HTTP/2 server push.

These are just a few examples of the many types of web servers available today. Each server has its own strengths and weaknesses, and the choice of which one to use will depend on factors such as the specific needs of your application, your level of technical expertise, and your budget.

Types of Database Servers

1. **Relational database servers**: Relational database servers store data in tables with rows and columns. They use SQL (Structured Query Language) to interact with the data. Some popular examples include MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and SQLite.
2. **NoSQL database servers**: NoSQL database servers are designed to store and retrieve unstructured data. They can handle large amounts of data and provide flexible data models. Some popular examples include **MongoDB, Cassandra, Couchbase, and Redis**.
3. **Object-oriented database servers**: Object-oriented database servers store data as objects, which are instances of a class or data structure. They are often used for object-oriented programming and can provide better performance for certain types of applications. Some popular examples include ObjectDB and Versant.
4. **Cloud database servers**: Cloud database servers are hosted on cloud platforms like Amazon **Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP**). They provide scalability, flexibility, and high availability, and are often used for large-scale applications.
5. **Graph database servers**: Graph database servers are designed to store and manage graph data, which consists of nodes and edges. They are often used for social networks, recommendation engines, and fraud detection. Some popular examples include Neo4j, ArangoDB, and Amazon Neptune.

Types of IDE

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. Some of the most popular types of IDEs include:

1. Eclipse: Eclipse is a popular IDE that supports multiple programming languages such as Java, C++, and Python. It includes various features such as syntax highlighting, debugging, and code completion.
2. Visual Studio: Visual Studio is a widely used IDE developed by Microsoft. It supports several programming languages such as C#, VB.NET, and F#. It also includes various features such as code profiling, unit testing, and code refactoring.
3. IntelliJ IDEA: IntelliJ IDEA is a Java-focused IDE developed by JetBrains. It includes features such as intelligent code completion, code inspections, and refactoring tools.
4. NetBeans: NetBeans is an open-source IDE that supports various programming languages such as Java, C++, and PHP. It includes features such as code debugging, code profiling, and code coverage tools.
5. PyCharm: PyCharm is a Python-focused IDE developed by JetBrains. It includes various features such as code completion, debugging, and unit testing.