T.E. AIML

Web Development

Practical Programs

1. Write a Node.js program using the **fs module** to perform the following operations:
2. Create a new file named **student.txt** and write the text "Welcome to Node.js File System Module" into it.
3. Read the contents of the file and display them on the console.
4. Delete the file and display a confirmation message after deletion.

2. Create one module file named **student.ts** that exports:

- A class Student with the following:
    - Properties: name, rollNo, marks (array of numbers).
    - Methods:
        - getAverage() → returns the average marks.
        - displayInfo() → logs student name, roll number, and average marks.

➢ Create another file **main.ts** that imports the Student class, creates at least two student objects, and displays their details.

3. Write a Node.js program to create an **HTTP web server** that listens on **port 3000** and displays the message
"Welcome to My First Node.js Web Server!"

4. Create a module file named **mathOperations.ts** that exports the following functions:

- add
- subtract
- multiply
- divide

➢ Create another file named **main.ts** that imports these functions from the module and performs the following:

- Accept two numbers (you can hardcode them).
- Display results for all four operations on the console.

5. Create a new Angular project named **component-demo** using Angular CLI.

➢ Generate two components using Angular CLI:

- **Parent component:** app.component (created by default)
- **Child component:** student
- In the **child component (student.component.ts)**, define a property studentName and display it in the template.
- In the **parent component (app.component.ts)**, send data (like a message or title) to the child component using **property binding**.
- Display both the parent message and the child component data in the browser.

6. Create an Angular application which will do following actions: Register User, Login User, Show User Data on Profile Component.

7. Create a new Angular project named **pipes-demo**.

   ➢ In the app.component.ts file:

- Create variables:
  - studentName → string
  - course → string
  - todayDate → Date
  - fees → number

   ➢ In the app.component.html file, use the following **Angular built-in pipes**:

- uppercase → to display the student name in uppercase
- lowercase → to display the course name in lowercase
- date → to display the current date in full format
- currency → to display the fees in Indian currency format (₹)

8. Create a new Angular project named **form-demo**.

   ➢ Enable **FormsModule** in your project by importing it in the app.module.ts file.

➢ Create a **student registration form** in the app.component.html that includes the following fields:

- Name (text input)
- Email (email input)
- Course (select dropdown)
- Gender (radio buttons)
- Submit button

➢ When the form is submitted, display the entered details below the form.

9. Create a file named **data.txt** containing some text (for example, student details).

- Create an **HTML file (index.html)** with a button.
- When the user clicks the button, use **AJAX** to read the content of the file and display it inside a <div> on the same page.
- Use **XMLHttpRequest** for making the AJAX request.

10. Create a Node.js application that connects to a MongoDB database and performs basic database operations.

➢ **Requirements:**

1. **Database Connection:**
   - Connect your Node.js application to a MongoDB database named studentDB.
   - Use the official **MongoDB Node.js Driver** or **Mongoose** for database communication.
2. **Collection:**
   - Create a collection named students with the following fields:
   - Name, Age, Course, Marks
3. **Operations:**
   Implement the following CRUD operations using Node.js:

- **Insert a new student record** into the database.
- **Retrieve and display** all student records.
- **Update** a student’s marks using their name.
- **Delete** a student record based on their name.

11. Create a **MEAN Stack application** to perform basic **CRUD (Create, Read, Update, Delete)** operations on a **Student** collection stored in **MongoDB**.

12. Create a **simple, responsive mobile webpage** for a fictional **Coffee Shop** using **HTML5** and **CSS3**.
The page should look good on **both desktop and mobile screens**.

Requirements:

1. The webpage should display:
   - A header with the shop name
   - A navigation bar with links (Home, Menu, Contact)
   - A section showing shop information and an image
   - A "Contact Us" form with Name, Email, and Message fields
   - A responsive layout that adapts to small (mobile) screens
2. Use **media queries** to make the page mobile-friendly.
3. Use **semantic HTML5 elements** (<header>, <nav>, <section>, <footer>, etc.).

13. Create a simple Mobile Website using jQuery Mobile.

14. Create a React JS application that demonstrates the use of **React Hooks** (useState, useEffect, and useRef) to build a small interactive app.

➢ **Program Requirements**

1. **Component Name:** UserTimerApp
2. **Functionality:**
   - Display a **text input** to enter a user's name.

  - Display a **timer** that counts how many seconds the user has been on the page.
  - Use a **button** to start and stop the timer.
  - Display a **greeting message** like "Hello, [UserName]! You have been here for [x] seconds."

➢ **Hooks to Use:**

- **useState** → to manage user name, timer count, and start/stop state.
- **useEffect** → to start/stop the timer automatically when the user clicks the button.
- **useRef** → to store the timer ID returned by setInterval().

15. Create a **responsive personal portfolio webpage** using **Bootstrap 5** components and utilities.

➢ **Program Requirements :**

1. **Page Structure:**
   - Use Bootstrap's **container**, **row**, and **col** classes to design a responsive layout.
   - Include the following sections:
     - **Header/Navbar**
     - **About Me**
     - **Projects**
     - **Contact Form**
     - **Footer**
2. **Detailed Specifications:**

 Header / Navbar

   - Use the Bootstrap **navbar** component.
   - Include links: *Home, About, Projects, Contact*.

- Make it **sticky** at the top and **collapse** into a hamburger menu on smaller screens.

About Section

- Use a **Bootstrap card** or **Jumbotron (container-fluid)** to display your photo, name, and a short bio.
- Use **grid system** (2 columns) — one for the image, one for the text.

Projects Section

- Display at least **three project cards** using **Bootstrap Cards** inside a responsive grid.
- Each card should have:
  - A project image
  - Project title
  - Short description
  - A “View More” or “Visit” button

Contact Section

- Create a **Bootstrap form** with fields:
  - Name
  - Email
  - Message
  - Submit Button
- Use **Bootstrap form controls** and **button classes**.

Footer

- Add a simple footer with social media icons (use **Bootstrap Icons** or Font Awesome).
- Center-align the text and icons.

3. **Styling & Responsiveness:**

- Use Bootstrap’s **text utilities**, **spacing classes** (mt, mb, p, etc.), and **colors**.
- Ensure the layout adapts perfectly on mobile, tablet, and desktop screens.