

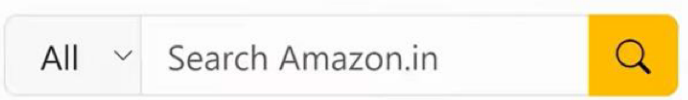
**DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY IV
SEMESTER B.TECH. (FLEXI-CORE) MISAC3: Mid-term Exam****SUBJECT: Full Stack Web Development Tools ICT 3230****Max Marks : 30****05/03/2025****Max. Time: 90minutes**

Q.No	Questions	Marks
1.	<p>Which class in Bootstrap is used to make an image responsive?</p> <ol style="list-style-type: none">1. .responsive-img2. .img-fluid3. .img-responsive4. .img-auto5. <p>Correct option is: 2</p>	0.5
2.	<p>What is the main advantage of using a CSS preprocessor like SASS or LESS?</p> <ol style="list-style-type: none">1. It helps reduce the number of HTML files..2. It allows using variables, mixins, and functions for maintainable styles.3. It makes CSS work in all browsers without prefixes.4. It automatically writes JavaScript code. <p>Correct option is: 2</p>	0.5
3.	<p>What does the following Gulp function do?</p> <pre>gulp.task('default', function() { console.log("Gulp is running..."); });</pre> <ol style="list-style-type: none">1. Runs all tasks in the project2. Logs "Gulp is running..." when executed3. Starts a local development server4. Concatenates JavaScript files <p>Correct option is: 2</p>	0.5
4.	<p>You are designing a dashboard with a navigation menu that should remain hidden by default to maximize screen space but can be revealed when needed. The requirement is to have a side panel that slides in from the edge of the screen when triggered, without overlaying or disrupting the main content. Additionally, the solution should be responsive and allow for smooth transitions while ensuring accessibility. Which Bootstrap component would best fulfill these requirements?</p>	0.5



	<ol style="list-style-type: none">1. Carousel2. Modal3. Accordion4. None of these <p>Correct option is: 4</p>	
5.	<p>Which of the following features is NOT supported by SCSS?</p> <ol style="list-style-type: none">1. Arithmetic operations with variables (e.g., \$width: 100px + 20px)2. Importing external CSS files using @import3. Directly using JavaScript expressions inside SCSS files4. Loops and conditionals (e.g., @for, @if) <p>Correct option is: 3</p>	0.5
6.	<p>Which feature is unique to Node.js compared to browser JavaScript?</p> <ol style="list-style-type: none">1. Event-driven, non-blocking I/O2. CSS manipulation3. DOM access4. Local storage <p>Correct option is: 1</p>	0.5
7.	<p>A dashboard needs to display a user's profile details stored in the Redux store. What function is used to retrieve specific pieces of state from the store?</p> <ol style="list-style-type: none">1. Reducer2. Selector3. Dispatcher4. Middleware <p>Correct option is: 2</p>	0.5
8.	<p>Imagine a robot navigating through a maze. It takes actions like moving forward, turning left, or turning right, and its position in the maze represents its state. What is the function that determines the next state based on the robot's current state and action?</p> <ol style="list-style-type: none">1. Redux.2. Reducer.3. subscribe.4. View <p>Correct option is: 2</p>	0.5
9.	<p>Which of the following is true about MongoDB indexing?</p> <ol style="list-style-type: none">1. Indexes slow down read operations2. Indexes slow down write operations3. Indexes have no effect on performance4. Indexes only work on primary keys <p>Correct option is: 2</p>	0.5



10.	<p>Which MongoDB method is used to remove a specific field from a document?</p> <ol style="list-style-type: none"> 1. \$unset 2. \$remove 3. \$delete 4. \$drop <p>Correct option is: 1</p>	0.5
11.	<p>Create a responsive search bar using appropriate Bootstrap classes that includes a dropdown, a search input, and a button, all within a single UI group as shown below?</p>  <p>[Correct Explanation: 1 Mark , Correct usage of class and attributes in code snippet: 2 Marks]</p> <p>The input-group class in Bootstrap is used to group multiple form elements (like inputs, buttons, or dropdowns) inside a single flexible container, ensuring they are aligned properly.</p> <pre><form> <div class="input-group"> <select class="input-group-text"> <option>All</option> </select> <input type="text" placeholder="Search Amazon.in" class="form-control"> <button class="bi bi-search btn btn-warning"></button> </div> </form></pre>	3
12.	<p>In Gulp, how can you automate the process of merging multiple CSS files into one and minify them for better performance? Demonstrate with an example</p> <p>[Correct Explanation: 0.5 Mark, Correct usage of require/import of modules 0.5 Mark, and writing gulp task snippet: 1 Mark]</p> <p>In Gulp, we can automate the process of merging multiple CSS files into one and minify them using the gulp-concat and gulp-minify-css plugins.</p> <pre>const gulp = require('gulp'); const concat = require('gulp-concat'); const cleanCSS = require('gulp-minify-css'); gulp.task('styles', function () {</pre>	2



	<pre>return gulp.src('src/styles/*.css') // Select all CSS files .pipe(concat('styles.css')) // Merge into one file .pipe(cleanCSS()) // Minify the merged CSS .pipe(gulp.dest('build/styles/')); // Save the output });</pre>	
13	<p>Consider a scenario where you need to create multiple button styles (e.g., primary, danger) with similar base properties but different colors. How would you achieve this using <code>@mixin</code> and <code>@extend</code> in SASS? Compare their approach with examples and analyze their impact on code reusability and maintainability</p> <p>[Correct compare explanation: 1 Mark, Correct usage of <code>@mixin</code>: 1 Mark, and correct usage of <code>@extend</code>: 1 Mark]</p> <p>@mixin</p> <ul style="list-style-type: none">• Allows defining reusable blocks of CSS with parameters.• Can be included multiple times without inheriting styles from other selectors.• More flexible as it allows dynamic values and customization.• <code>@mixin</code> is best for reusable styles with variations <pre>@mixin button-style(\$bg-color, \$text-color) { background-color: \$bg-color; color: \$text-color; padding: 10px 20px; border-radius: 5px; } .btn-primary { @include button-style(blue, white); } .btn-danger { @include button-style(red, white); }</pre>	3



	<pre>} @extend • Inherits styles from an existing selector, reducing redundancy. • Generates shared CSS rules, minimizing file size. • Less flexible as it doesn't allow arguments. • @extend is useful for shared styles .button { padding: 10px 20px; border-radius: 5px; display: inline-block; } .btn-primary { @extend .button; background-color: blue; color: white; } .btn-danger { @extend .button; background-color: red; color: white; }</pre>	
14	<p>Develop a web application as illustrated in the provided figure. The application should have four components: App, Core Courses, Flexicore Courses, and Elective Courses. Each component should display the list of courses offered under its respective category, as illustrated in the given figure. The Core Courses component should include a button labeled "Go To FlexiCore Courses". Clicking this button should navigate the user to the FlexiCore Courses component. When the application is deployed, the Core Courses component should be the default view. Implement the navigation between components using React Router</p>	4



[Core-Courses](#) [Flexicore-Courses](#) [Elective-Courses](#)

List of Core Courses

Engineering Economics and Financial Management

[Go to Flexicore-Courses](#)

SCHEME:

Creating Link or NavLink with correct syntax : 1M

Creation of Route with suitable syntax : 1M

Correct use of useNavigate hook : 1M

Creation of Core, Flexicore and Elective component : 1M

```
import { BrowserRouter as Router, Routes, Route, Link, useNavigate,
} from "react-router-dom";
```

```
function App() {
  return (
    <Router>
      <center>
        <Link to="/">Core Courses1 </Link>
        <Link to="/flexicore">Flexicore Courses1</Link>
        <Link to="/elective">Elective Courses1</Link>
      </center>

      <Routes>
        <Route path="/" element={<Core />} />
        <Route path="/flexicore" element={<Flexicore />} />
        <Route path="/elective" element={<Elective />} />
      </Routes>
    </Router>
  );
}
```

```
const Core = () => {
  const navigate = useNavigate();

  return (
    <div>
      <center>
        <h2>List of core courses</h2>
        <p>Engineering economics and financial management</p>

        <button onClick={() =>
          navigate("/flexicore")}>Go to Flexicore courses</button>
      </center>
    </div>
  );
};
```



```
const Flexicore = () => {  
  return (  
    <div>  
      <center>  
        <h2>List of flexi core courses</h2>  
        <p>Flexicore 1</p>  
        <p>Flexicore 2</p>  
      </center>  
    </div>  
  );  
};
```

```
const Elective = () => {  
  return (  
    <div>  
      <center>  
        <h2>List of elective courses</h2>  
        <p>elective 1</p>  
        <p>elective 2</p>  
      </center>  
    </div>  
  );  
};
```

```
export default App;
```



15	<p>Consider a scenario where you are developing a reusable React component for displaying user profile information. The component receives multiple props, such as name, age, and location. To improve code readability and maintainability, you decide to use props destructuring. Demonstrate its usage with a suitable example in a functional component.</p> <p>SCHEME :</p> <p>destructuring syntax with suitable example : 2M</p> <p>Using parameters after destructuring with example : 1M</p> <p>Props destructuring:</p> <ol style="list-style-type: none">1. Destructure in the function parameter itself2. Destructure in the function body <pre>//App.js function App() { return (<Disp dept="ict" branch="cce" />) } //Disp.js function Disp({dept, branch}) { // const {dept,branch} = props; console.log('hello',dept); return (<> <h1>Department is :{dept}</h1> <h1>Branch is :{branch}</h1> </>) } export default Disp;</pre>	3
16	<p>Consider a scenario where you are asked to develop a weather application that allows users to select a city and view its current weather details. However, you notice that the API call behaves differently depending on how you configure the useEffect hook. Compare the behavior of useEffect when used with different dependency configurations.</p> <p>SCHEME</p> <p>The useEffect hook in React allows us to perform side effects in function components. managing side effects such as data fetching, subscriptions, or manually changing the DOM is a common task.</p>	3



Different dependency configurations are:

1. `useEffect` Without a Dependency Array

Omitting the dependency array causes the effect to run after every render. Ensures the latest weather data is fetched on every render.

Syntax:

```
useEffect(() => {  
  fetchWeatherData();  
});
```

SCHEME: Explanation : 0.5 , Syntax : 0.5

2. `useEffect` With empty array

Providing an empty array `[]` as the dependency list ensures that the effect runs only once after the initial render. Efficient, avoiding unnecessary API requests.

Syntax:

```
useEffect(() => {  
  fetchWeatherData();  
}, []);
```

SCHEME: Explanation : 0.5 , Syntax : 0.5

3. `useEffect` With a Dependency

Specifying dependencies allows to control when the effect runs based on changes to specific values. In given example, the API call is made whenever the city state changes, ensuring users get updated weather data. Efficient because it prevents unnecessary API calls unless city changes.

Syntax:

```
useEffect(() => {  
  fetchWeatherData();  
}, [city]);
```

SCHEME: Explanation : 0.5 , Syntax : 0.5



17	<p>How does the motion.div component in Framer Motion differ from a regular <div> in React?</p> <p>[Write any 3 properties of <motion.div> usage which doesn't exist in <div>, 1 Mark Each]</p> <p>The motion.div component from Framer Motion extends the functionality of a regular <div> in React by adding built-in support for animations, gestures, and transitions. Here are the key functionalities which regular <div> doesn't support:</p> <p>1. Animation Capabilities using props like animate, initial, and exit.</p> <pre><motion.div initial={{ opacity: 0, x: -100 }} animate={{ opacity: 1, x: 0 }} transition={{ duration: 0.5 }} /></pre> <p>2. Gesture & Event Handling: Supports dragging, hover, and tap events out of the box.</p> <pre><motion.div whileHover={{ scale: 1.1 }} whileTap={{ scale: 0.95 }} onHoverStart={() => console.log('hover started!')} /></pre> <p>3. Layout Animations: Makes elements smoothly animate when their size or position changes.</p> <pre><motion.div layout style={{ width: 200, height: 100, background: "green" }} /></pre> <p>4. By wrapping motion components with <AnimatePresence> we gain access to the exit prop</p> <pre><AnimatePresence> {show ? <motion.div key="box" exit={{ opacity: 0 }} /> : null}</pre>	3



	<pre></AnimatePresence></pre> <p>5. Motion supports both types of scroll animations, scroll-triggered and scroll-linked.</p> <pre><motion.div</pre> <pre> initial={{ backgroundColor: "rgb(0, 255, 0)", opacity: 0 }}</pre> <pre> whileInView={{ backgroundColor: "rgb(255, 0, 0)", opacity: 1 }}</pre> <pre>/></pre>	
18	<p>Given a collection named “students” with fields {name, marks, subject}, write an aggregation query to calculate the average marks per subject</p> <p>[Write aggregation query: 1.5 Mark, Explanation: 0.5 Mark]</p> <pre>db.students.aggregate([</pre> <pre> {</pre> <pre> \$group: {</pre> <pre> _id: "\$subject",</pre> <pre> averageMarks: { \$avg: "\$marks" }</pre> <pre> }</pre> <pre> }</pre> <pre>])</pre>	2
19	<p>How would you update multiple documents in MongoDB where the “status” field is “pending” to “completed”?.</p> <p>[Write update query: 1.5 Mark, Explanation: 0.5 Mark]</p> <pre>db.collection.updateMany(</pre> <pre> { status: "pending" }, // Filter: Select documents where status is "pending"</pre> <pre> { \$set: { status: "completed" } } // Update: Set status to "completed"</pre> <pre>)</pre>	2