Internal Assessment Test 2 – March 2025
Scheme and Solutions

Sub:	**Full Stack Development**					Sub Code:	BIS601	Branch:	ISE
Date:	27/05/2025	Duration:	90 min's	Max Marks:	50	Sem/Sec:	V I/ A, B, C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT

1 a)	Apply conditional rendering in building an issue tracker app in React? Can you give an example where you conditionally display different components based on the issue's status (e.g., open vs closed)?
	
	Scheme:
	● Explanation of conditional rendering in React: 2M
	● Example use case (Issue Tracker): 1M
	● Code snippet showing conditional rendering based on issue status: 2M
	
	Solution: Conditional rendering allows React to render different UI based on conditions. In an issue tracker:
	
	`function IssueStatus({ isOpen }) {`
	`  return (`
	`  <div>`
	`  {isOpen ? <OpenIssueComponent /> : <ClosedIssueComponent />}`
	`  </div>`
	`  );`
	`}`
	
	This renders different components based on the `isOpen` status
		5M	CO2	L3

1 b)	What are React class components, and how do they differ from function components? Describe the lifecycle methods associated with class components?
	
	Scheme:
	● Definition of class and function components: 2M
	● Key differences: 1M
	● Lifecycle methods (componentDidMount, shouldComponentUpdate, etc.): 2M
	
	Solution: Class components use ES6 classes and lifecycle methods, while function components use hooks.
	
	`class MyComponent extends React.Component {`
	`  componentDidMount() {`
	`  console.log("Mounted");`
	`  }`
	`  render() {`
	`  return <div>Hello</div>;`
	`  }`
	`}`
		5M	CO2	L2

2 a)	Justify on passing functions as props to child components. How would you handle events like button clicks from a child component in the parent?	5M	CO2	L3

	Scheme:
	
	● Purpose of passing functions as props: 2M
	● Handling child-to-parent communication: 1M
	● Example: 2M
	
	Solution: Functions are passed as props to allow child components to communicate with parent:
	
	function Parent() {
	const handleClick = () => alert("Clicked!");
	return <Child onClick={handleClick} />;
	}
	function Child({ onClick }) {
	return <button onClick={onClick}>Click Me</button>;
	}			

2 b)	What is dynamic component composition, and how can it improve the flexibility of a React app? How would you conditionally render different components based on user input or application state?
	
	Scheme:
	
	● Define dynamic component composition: 2M
	● Example with condition-based component rendering: 2M
	● Benefits: 1M
	
	Solution: Dynamic composition allows rendering different components based on logic:
	
	function Dashboard({ role }) {
	return role === 'admin' ? <AdminPanel /> : <UserPanel />;
	}
		5M	CO2	L1

3 a)	Explain the process of updating state both in class components (`this.setState`) and functional components (`useState`). What are the risks of direct state mutation, and how can you use hooks like `useEffect` to handle side effects efficiently?
	
	Scheme:
	
	● State update in class (`this.setState`) and functional (`useState`): 2M
	● Risks of direct mutation: 1M
	● Role of `useEffect` in handling side effects: 2M
	
	Solution: Class:
	
	this.setState({ count: this.state.count + 1 });
	
	Function:
	
	const [count, setCount] = useState(0);
	useEffect(() => { console.log(count); }, [count]);
	
	Avoid direct mutation to prevent unpredictable behavior.
		5M	CO2	L2

3 b)	Describe how Express handles HTTP requests and responses. What are some best practices for organizing routes and middleware, and how does Express enable efficient request handling for RESTful APIs? **Scheme:** ● Request/Response in Express: 2M ● Routing and middleware usage: 2M	5M	CO3	L2

	• Best practices: 1M
	
	Solution:
	
	app.get('/api', (req, res) => res.send('Hello'));
	app.use(express.json());
	
	Use routers for modules, middleware for auth/logging.			

4 a)	Compare and contrast REST and GraphQL from the perspective of querying data, performance (e.g., over fetching/under fetching), and the flexibility they offer for clients. Discuss when you would prefer one over the other. **Scheme:** • REST vs GraphQL data fetching: 2M • Performance comparison: 2M • When to prefer each: 1M **Solution:** REST sends fixed data. GraphQL lets client specify fields. REST may overfetch; GraphQL solves this. Use REST for simple APIs, GraphQL for complex querying needs.	5M	CO3	L2

4 b)	Describe the process of lifting state up in React. Why is it necessary, and what problems does it solve in managing state between multiple components? Provide an example where lifting state up improves the flow of data between components.
	
	Scheme:
	
	• Definition and purpose of lifting state: 2M
	• Problem it solves: 1M
	• Example: 2M
	
	Solution: Lifting state means moving shared state to the closest common parent.
	
	function Parent() {
	const [value, setValue] = useState('');
	return <><Child1 value={value} /><Child2 onChange={setValue} /></>;
	}
		5M	CO2	L2

5 a)	Apply MongoDB CRUD operations to explain methods like `insertOne()`, `insertMany()`, `find()`, `updateOne()`, `deleteOne()`, and `deleteMany()` in MongoDB with suitable examples.
	Scheme:
	
	• Explanation of each MongoDB method: 3M
	• Example for each: 2M
	
	Solution:
	
	// Insert
	collection.insertOne({ name: "A" });
	// Find
	collection.find({ name: "A" });
	// Update
	collection.updateOne({ name: "A" }, { $set: { name: "B" } });
	// Delete
	collection.deleteOne({ name: "B" });
		5M	CO4	L3

5 b)	Discuss how Webpack modularizes the application code, using entry points to define the starting points and output configurations to specify how and where bundles are generated. How does this impact the structure of a large application? **Scheme:** ● Webpack entry/output concept: 2M ● How it structures large apps: 2M ● Example config: 1M **Solution:** Webpack uses `entry` to start and `output` to build bundle: entry: './src/index.js', output: { filename: 'bundle.js', path: __dirname + '/dist' }	5M	CO5	L2
6 a)	Explain projection in MongoDB and how to use it to limit or modify the fields returned in query results. For example, how do you return only certain fields from documents using the projection feature in the `find()` method? **Scheme:** ● Define projection: 2M ● Syntax and usage: 2M ● Example: 1M **Solution:** To limit fields: collection.find({}, { projection: { name: 1, _id: 0 } }); Returns only `name` field.	5M	CO4	L2
6 b)	What is a typical Webpack production build configuration, and how do you optimize it for performance? Describe how to configure Webpack for production builds. **Scheme:** ● Production config steps: 3M ● Performance optimizations: 2M **Solution:** Set mode: mode: 'production' Use minification, code splitting, caching. optimization: { minimize: true, splitChunks: { chunks: 'all' } }	5M	CO5	L1