Modified

CBCS SCHEME

USN | | | | | | | | | | |

BIS601

Sixth Semester B.E./B.Tech. Degree Examination, June/July 2025
Full Stack Development

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Explain the difference between var, let and const with suitable examples.	5	L2	CO1
	b.	Describe the various data types in JavaScript. Give examples for each.	5	L2	CO1
	c.	Write a program that creates an array of 5 cities and performs the following: i) Adds a city at the end ii) Removes the first city iii) Logs the total numbers of cities iv) Finds the index of a special city v) Search for a specific city vi) Replace specific city with another	10	L3	CO1
		OR			
Q.2	a.	Create a JavaScript object named student with properties: name, grade and subjects. Add a method displayInfo( ) to log student details.	5	L3	CO1
	b.	Explain the structure of a JavaScript function. How are parameters and return values used?	5	L2	CO1
	c.	Explain the usage of at least five different string methods in JavaScript with the help of suitable code snippets.	10	L2	CO1
		Module – 2			
Q.3	a.	What is the Document Object Model (DOM)? Explain its significance in web development.	5	L2	CO2
	b.	Explain event delegation and how it helps improve performance in DOM manipulation.	5	L2	CO2
	c.	Explain any six different DOM method used to access or manipulate HTML elements in JavaScript, including their syntax, use cases and when each in preferred.	10	L2	CO2
		OR			
Q.4	a.	How can you select HTML elements using JavaScript? List and explain at least three methods.	5	L2	CO2
	b.	What are event listeners in JavaScript? How do they differ from traditional event attributes (like onclick) for binding events?	5	L2	CO2

	c.	Create a button in your HTML with the text "Click Mc". Add an event listener to log "Button clicked!", to the console when the button is clicked. Select on image and add a mouseover event listener to change its border color. Add on event listener to the document that logs the key pressed by the user.	10	L3	CO2
		Module – 3			
Q.5	a.	Explain the components of the MERN stack and discuss how they interact in a full stack application. Highlight the role of each component with examples.	10	L2	CO3
	b.	Describe how to implement a simple REST API using express to return a list of issues. Include an explanation of routing, request handling and how JSON data is sent as a response.	10	L3	CO3
		OR			
Q.6	a.	Discuss how react uses JSX for rendering UI components. What are the benefits of using JSX over plain JavaScript in react applications?	10	L3	CO3
	b.	Describe the steps involved in creating a react components using ES6 class syntax. What are the essential life cycle methods used in such a component?	10	L3	CO3
		Module – 4			
Q.7	a.	Explain how state is initialized and update in a react class component. Illustrate with an example from the issue tracker application.	10	L2	CO4
	b.	Discuss how event handling is implemented in react. How does it differ from traditional DOM event handling in vanilla JavaScript?	10	L3	CO4
		OR			
Q.8	a.	Differentiate between stateless and stateful components in react. When should each be used in a component – based architecture?	10	L2	CO4
	b.	Write a react class component that displays a button and a counter. Each time the button is clicked, increase the count and display it. Use constructor to initialize state and setstate( ) to update it.	10	L3	CO4
		Module – 5			
Q.9	a.	Discuss the key differences between insert, update and find operations in MongoDB. How does MongoDB handle flexible schema and embedded documents?	10	L2	CO5
	b.	Create a simple webpack. config.js file that: i) Bundles on entry point file App.jsx ii) Uses babel – loader to transpile JSX iii) Outputs app. bundle.js in a static directory. iv) Uses ES6 presets for react.	10	L3	CO5

		OR			
Q.10	**a.**	Write Mongo shell commands to perform the following operations: • Insert three employee documents with different fields. • Update one document to add a middle name. • Delete one document by-id • Create an index on the age field • Query employees whose age is greater than 30.	**10**	**L3**	**CO5**
	b.	Explain the purpose of using webpack in a full stack project. Describe how webpack helps in modularization and bundling.	**10**	**L2**	**CO5**

* * * * *

VTUEXAMINATION June-2025/July-2025
SCHEME OF EVALUATION

Sub:	**Full Stack Development**					Sub Code:	**BCS601**	Branch:	**ISE**
Exam Date:	16-06-2025	Duration:	**3 Hrs**	MaxMarks:	**100**	Sem/Sec:	**VI/ A, B &C**		**OBE**

Answer any **FIVE FULL** Questions		MARKS	CO	RBT
Sl. No.	Solutions			
1.a)	Var: Function-scoped, can be rededared and updated, Hoisted to the top of its scope. var x=10, var x=20, let: Block scoped, cannot be redeclared in the same scope, but can be updated. let y=15; y=25; const z=30; //Error: Assignment to constant variable explanation and examples.	5	CO1	L2
1.b)	Javascript data types: String: Represents textual data. E.g: let name=”Alice”; Number: Represents numeric values. E.g: let age=25; Boolean: True or false, E.g: let isActive=true; Undefined: Varidable declared but not assigned. E.g: let city; Null: Explicily no value. E.g: let score=null; Object: Collection of key-value pairs. E.g: let student={name:”Tom”} Array: A type of object for ordered collections. E.g: let colors=[“red”,”blue”]	5	CO1	L2
1.c)	let cities = ["New York", "Los Angeles", "Chicago", "Houston", "Phoenix"]; cities.push("San Francisco"); cities.shift(); console.log(cities.length, cities.indexOf("Chicago"), cities.includes("Houston")); if (cities.includes("Los Angeles")) cities[cities.indexOf("Los Angeles")] = "San Diego"; console.log(cities);	10	CO1	L3
2.a)	const student = { name: "John Doe", grade: 10, subjects: ["Math", "Science", "English"], displayInfo() { console.log(`Name: ${this.name}`); console.log(`Grade: ${this.grade}`); console.log(`Subjects: ${this.subjects.join(", ")}`); } }; student.displayInfo();	5	CO1	L3
2.b)	function add(a,b){ return a+b; } let result=add(5,3); add is the function name. (a,b) are parameters. return a+b; is the return statement.	5	CO1	L2
2.c)	Let text=”javascript programming”; console.log(text.length); console.log(text.slice(0,10)); console.log(text.replace(“javascript”, “JS”); console.log(text.toUppercase(1);	10	CO1	L2

	console.log(text.indexof("Pro"));			
3.a)	The DOM is a programming interface for HTML/XML documents. It represents the structure as a tree of nodes. Each HTML element is represented as an object (node). JavaScript can manipulate content, structure and style dynamically using the DOM.	5	CO2	L2
3.b)	Event delegation is a technique of handling event at a parent element instead of assigning to individual child elements. It works using event bubbling. Advantages: Better performance (Less memory used). Works for dynamically added elements	5	CO2	L2
3. c)	i) getElementByID("id"): Selects on element by ID. ii) getElementByClassName("class"):Selects elements by class. iii) getElementByTagName("tag"): Selects elements by tag. iv)QuerySelector: Selects the first match v) iii) innerHTML: Gets or sets HTML content. vi) setAttribute(attr, value): sets an attribute value.	10	CO2	L2
4. a)	getElementByID() getElementByClassName() querySelector() Each method returns references(s) to matching DOM elements for manipulation.	5	CO2	L2
4. b)	addEventListener() is used to assign multiple events and keep JS separate from HTML. Inline onclick is written in HTML and limited to one handler. Document.getElementById("btn").addEventListener("click", function(){ Alert("clicked"); });	5	CO2	L2
4. c)	<!DOCTYPE html> <html> <head> <title>Event Listeners Example</title> <style> /* Add some basic styling for the image */ #myImage { border: 2px solid black; margin: 20px; cursor: pointer; } </style> </head> <body> <button id="myButton">Click Me</button> <img id="myImage" src="your-image-source.jpg" alt="A sample image" width="200"> Press any key on your keyboard! <script src="script.js"></script> </body> </html> JavaScript Code (script.js): javascript *// 1. Button Click Event Listener* const myButton = document.getElementById('myButton'); myButton.addEventListener('click', () => { console.log('Button clicked!'); }); *// 2. Image Mouseover Event Listener* const myImage = document.getElementById('myImage'); myImage.addEventListener('mouseover', () => { myImage.style.borderColor = 'red'; *// Change border color to red* }); *// Optional: Add mouseout to change border color back (if desired)*	10	CO2	L3

	myImage.addEventListener('mouseout', () => { myImage.style.borderColor = 'black'; *// Change border color back to black* }); *// 3. Document Keydown Event Listener* document.addEventListener('keydown', (event) => { console.log(`Key pressed: ${event.key}`); });			
5. a)	The MERN stack includes: MongoDB: NoSQL databases that stores data as JSON-like documents. Express.js: Node.js framework used to build RESTful APIs. React.js: Front-end library for building UI components. Node.js: JavaScript runtime for running server-side code. Interaction: React sends HTTP requests to Express. Express handles routes and business logic. Express interacts with MongoDB for data storage/retrieval. Node.js powers the server environment.	10	CO3	L2
5. b)	const express=require (‘express’); Const app=express(); Const issues=[{ id:1, title: “Bug in login”}, {id:2, title: UI not responsive”}]; app.get (‘/api/issues’, (reg, res)=>{ res.json (issues); }); app.listen (3000, ()=> console.log (‘server running on part 3000’));	10	CO3	L3
6 (a)	**Question**: How React uses JSX. Benefits over JS. **Scheme**: • JSX definition (2M) • Example JSX (2M) • Advantages: readability, abstraction, etc. (6M) **Solution**: JSX = JavaScript + XML. Syntax extension used in React. const element = Hello, world! ; Benefits: • Easier to read and write UI code • Prevents injection attacks • Supports inline expressions • Strongly integrated with React DOM	10	L3	CO3
6 (b)	**Question**: Steps to create React component using ES6 class syntax. **Scheme**: • ES6 class syntax (3M) • `constructor`, `state`, `setState` (3M)	10	L3	CO3

	• Lifecycle methods: `componentDidMount`, `render`, etc. (4M) **Solution**: class Welcome extends React.Component { constructor(props) { super(props); this.state = { count: 0 }; } componentDidMount() { // API call } render() { return Hello, {this.props.name} ; } } Lifecycle methods: `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`			
7 (a)	**Question**: Explain state initialization and update with example. **Scheme**: • Definition of state (2M) • How to initialize (2M) • Example from issue tracker (3M) • Updating state using `setState` (3M) **Solution**: class IssueTracker extends React.Component { constructor(props) { super(props); this.state = { issues: [] }; } componentDidMount() { this.setState({ issues: [{ id: 1, title: 'Bug' }] }); } render() { return {this.state.issues[0].title} ; } }	10	L2	CO4
7(b)	Question**: Event handling in React vs DOM.** **Scheme**: • React event system (3M) • `onClick`, `onChange` usage (3M) • Difference from vanilla JS: synthetic events, camelCase (4M) **Solution**: React: <button onClick={this.handleClick}>Click</button> Vanilla JS:	10	L3	CO4

	document.getElementById("btn").addEventListener("click", handler); React uses synthetic event system.			
8 (a)	Question**: Stateless vs Stateful components.** **Scheme**: • Definition and difference (4M) • When to use each (3M) • Examples (3M) **Solution**: Stateless: Functional components without internal state. Stateful: Class or function components with state (via `useState` or `this.state`). Use stateless for simple UI, stateful for dynamic data.	10	L2	CO4
8 (b)	Question**: React class component with button and counter.** **Solution**: class Counter extends React.Component { constructor(props) { super(props); this.state = { count: 0 }; } handleClick = () => { this.setState({ count: this.state.count + 1 }); }; render() { return (<button onClick={this.handleClick}>Click me</button> Count: {this.state.count}); } }	10	L3	CO4

9 (a)	Question**: Insert, update, find in MongoDB. Flexible schema.** **Scheme**: • Insert example (2M) • Update example (2M) • Find example (2M) • Flexible schema explanation (2M) • Embedded docs (2M) **Solution**: db.students.insert({ name: "John", age: 21 }); db.students.update({ name: "John" }, { $set: { age: 22 } }); db.students.find({ age: 22 }); MongoDB supports flexible schema — different documents can have different fields	10	L2	CO5
9(b)	Question**: Simple webpack config.js file.** **Solution**: module.exports = { entry: './App.jsx', output: { path: __dirname + '/static', filename: 'bundle.js' }, module: { rules: [{ test: /\.jsx$/, exclude: /node_modules/, use: { loader: 'babel-loader', options: { presets: ['@babel/preset-react', '@babel/preset-env'] }	10	L3	CO5

	} }] } };			
10 (a)	Question**: Mongo shell commands for insert, update, delete, index, query.** **Solution**: // Insert db.employees.insertMany([{ name: "Alice", age: 28 }, { name: "Bob", age: 35 }, { name: "Charlie", age: 40 }]); // Update db.employees.updateOne({ name: "Alice" }, { $set: { middle_name: "Marie" } }); // Delete db.employees.deleteOne({ _id: ObjectId("id_here") }); // Create index db.employees.createIndex({ age: 1 });	10	L3	CO5

	// Query db.employees.find({ age: { $gt: 30 } }); 			
10(b)	**Question**: Purpose of using Webpack. **Scheme**: • Webpack definition (2M) • Modularization (3M) • Bundling (3M) • Benefits (2M) **Solution**: Webpack is a module bundler that converts assets into a small number of files. • **Modularization**: Breaks code into manageable pieces. • **Bundling**: Combines modules into one output file. • Supports loaders for JSX, CSS, etc. • Improves performance and maintainability.	10	L2	CO5

FacultySignature CCISignature HODSignature