



JK Lakshmipat University Jaipur
INSTITUTE OF ENGINEERING AND TECHNOLOGY

End Term Examination, May 2026
B. Tech., Semester VI, 2025-26

Roll No.....

CS1237: Full Stack Application Development

Time: 2:30 hours

Max. Marks: 30

Instructions to students:

1. Do not write anything, other than your roll number, on the question paper.
2. Mention all the assumptions for your answers clearly.
3. Calculators are not allowed

All questions are compulsory. Each question carries 3 marks.

[Total: 10 × 3 = 30 Marks]

Using the data provided in orders.json, answer questions 1 to 4.

orders.json

```
[{
  "_id": {"$oid": "64a1f1000000000000000001"},
  "orderId": "ORD-1001",
  "customerId": "CUST-201",
  "customerName": "Rahul Sharma",
  "email": "rahul.sharma@example.com",
  "city": "Jaipur",
  "status": "delivered",
  "orderDate": {"$date": "2024-01-05T10:30:00Z"},
  "deliveryDate": {"$date": "2024-01-08T14:00:00Z"},
  "items": [
    {"productId": "P101", "name": "Wireless Mouse", "qty": 2, "price": 850},
    {"productId": "P105", "name": "USB Hub", "qty": 1, "price": 650}
  ],
  "payment": { "method": "UPI", "status": "paid" },
  "totalAmount": 2350,
  "discount": 100,
  "finalAmount": 2250
},
{.....}, {.....}, {.....}, .....]
```

1. Which products have been ordered the most across all orders? Show top 3. (CO 6, 7)
2. Generate a report showing revenue **per city per month**, sorted by month first, then revenue descending. (CO 6, 7)
Expected Output:

```
{_id: { month: "2024-01", city: "Jaipur" }, totalOrders: 1, totalRevenue: 2250 },
{_id: { month: "2024-01", city: "Delhi" }, totalOrders: 1, totalRevenue: 4150 },
{_id: { month: "2024-02", city: "Pune" }, totalOrders: 1, totalRevenue: 7698 },
...
```

3. From all orders (any status), find each month's total revenue, total orders, and **average order value** — then show only months where average order value is **greater than ₹4000**. (CO 6, 7)
Expected Output:

```
{ month: "2024-03", totalOrders: 3, totalRevenue: 26900, avgOrderValue: 8966.67 },
{ month: "2024-02", totalOrders: 4, totalRevenue: 27648, avgOrderValue: 6912.00 },
...
```

4. Find total revenue per payment method and calculate what **percentage** each method contributes to overall revenue. **Expected Output:** (CO 6, 7)

```
{ method: "EMI", revenue: 30200, percentage: 37.42 }
{ method: "Credit Card", revenue: 11497, percentage: 14.24 }
{ method: "UPI", revenue: 9200, percentage: 11.40 }
...
```

5. Consider two MongoDB collections: orders and customers.

orders.json

(CO 6, 7)

```
[
  { "_id": 1, "product": "Laptop", "customer_id": 101, "amount": 80000 },
  { "_id": 2, "product": "Phone", "customer_id": 102, "amount": 30000 },
  { "_id": 3, "product": "Tablet", "customer_id": 101, "amount": 20000 }
]
```

customers.json

```
[
  { "_id": 101, "name": "Amit", "city": "Delhi" },
  { "_id": 102, "name": "Neha", "city": "Mumbai" }
]
```

- I. Write a MongoDB aggregation query using \$lookup to join orders with customers.
- II. Display the following fields in the output: product, amount, customer name, city.
- III. Write the expected output for the above query.

6. The following code makes three API calls using Promise.all.

Write the exact output and explain what happens if fetchPosts rejects.

(CO 4, 5)

```
function fetchUser() { return Promise.resolve('User: JKLU'); }
function fetchPosts() { return Promise.reject('Error: Posts unavailable'); }
function fetchProfile() { return Promise.resolve('Profile: loaded'); }

Promise.all([fetchUser(), fetchPosts(), fetchProfile()])
  .then(results => console.log('Success:', results))
  .catch(error => console.log('Failed:', error));
```

7. Write the exact output of the code below.

Explain the structure of each object inside the results array.

(CO 4, 5)

```
const apis = [
  Promise.resolve({ id: 1, name: 'Rohan' }),
  Promise.reject('Network timeout'),
  Promise.resolve({ id: 3, name: 'Geeta' }),
  Promise.reject('Auth failed'),
];

Promise.allSettled(apis).then(results => {
  results.forEach((result, i) => {
    if (result.status === 'fulfilled') {
      console.log(`API ${i+1} OK:`, result.value);
    } else {
      console.log(`API ${i+1} FAILED:`, result.reason);
    }
  });
});
```

8. Write the exact output of the code below.

Explain how the variable 'count' is accessible inside the returned function and using callstack show the sequence of execution.

(CO 4, 5)

```
function makeCounter(start) {  
  let count = start;  
  
  return function increment() {  
    count++;  
    console.log('Count:', count);  
  };  
}  
  
const counterA = makeCounter(0);  
const counterB = makeCounter(10);  
  
counterA();  
counterA();  
counterB();  
counterA();
```

9. Two JavaScript code snippets are given below: Code 1 uses a regular function inside an object method, and Code 2 uses an arrow function inside the same method.

Code 1	Code 2
<pre>this.name = "Arijit Singh"; const A = { name: "Shreya", callback1: function () { function abc() { console.log(this.name); } abc(); } }; A.callback1();</pre>	<pre>var name = "Sonu Nigam"; const A = { name: "Kavita", callback1: function () { abc = () => { console.log(this.name); }; abc(); } }; A.callback1();</pre>

Predict the console output of both codes in Browser and Node.js environments. Justify each output in one sentence.

(CO 4, 5)

10. Write a JavaScript function `memoize(fn)` that takes any function as input and returns a new function. The returned function should compute the result on the first call with given arguments, store it in a cache, and return the cached result on subsequent calls with the same arguments without recomputing. Demonstrate your solution by memoizing a function that computes the factorial of a number. Show at least 3 function calls in your output, where at least one call reuses a cached result. (CO 2)

END

