MERN Stack Developer Interview Test - Answer

Name: Anish Kumar | Mob: 8539873997 | Email: anishkr2842003@gmail.com

---

Q. 1. Explain how you would scale a MERN application (5 Marks)

(e.g., horizontal scaling, caching, optimizing queries, deployment with PM2/NGINX, etc.)

Answer:

I don't have much idea about scaling a MERN application, but I know some points.

- **Caching**: We can use caching layers like Redis or in-memory caching for frequently accessed data to reduce database load and improve response time.
- **Database Query**: Use MongoDB indexes, aggregation pipelines, and efficient query patterns to minimize execution time and reduce unnecessary data retrieval.
- **Process Management**: Deploy the Node.js backend with PM2 for auto-restart and zero-downtime reloads

---

Q. 2. What are the pros and cons of using MongoDB for relational data? (5 Marks)

Answer:

Pros of using MongoDB for relational data

1. Flexible schema design allows easy changes.
2. High write performance for large datasets.
3. Good for handling nested data.

Cons of using MongoDB for relational data

1. No true native joins like SQL databases.
2. Data duplication can increase storage needs.
3. Complex queries may require aggregation pipelines.

Q. 3. Fix the following code and explain the issue: *(5 Marks)*

```
app.get('/user/:id', async (req, res) => {
  const user = await User.findById(req.params.id);
  res.send(user.name);
});
```

Answer:

Correct code

```
app.get('/user/:id', async (req, res) => {
  try {
    const user = await User.findById(req.params.id);
    if (!user) {
      return res.status(404).send('User not found');
    }
    res.send(user.name);
  } catch (error) {
    res.status(500).send('Server error');
  }
});
```

Issue Explain (these are best practices):

1. No try-catch use
2. No check for user is exists or not

Q. 4. How does React rendering work when state is updated? Explain with example. *(5 Marks)*

Answer:

When we update any state then react does not update the full DOM because react has virtual dom. When we update any ui or data via state then react create a virtual DOM with updated data and then compare with previous DOM and new DOM and only render changed ui or data.

Example:

```
import { useState } from "react";
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p> {/* re-renders when count changes */}
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  )}
```

Explanation:

When we click on the button for increase number then react render only p tag because other parts are also available in previous DOM so react render only that part who not matched in virtual DOM

Q. 5. Given this JSON response, write React code to display a table of users. *(5 Marks)*

Json

```
[
  { "id": 1, "name": "Amit", "email": "amit@example.com" },
  { "id": 2, "name": "Sara", "email": "sara@example.com" }
]
```

Answer:

```
import React from "react";
const users = [
  { id: 1, name: "Amit", email: "amit@example.com" },
  { id: 2, name: "Sara", email: "sara@example.com" }
];

export default function UserTable() {
  return (
    <table border="1" cellPadding="5">
      <thead>
        <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Email</th>
        </tr>
      </thead>
      <tbody>
        {users.map(user => (
          <tr key={user.id}>
            <td>{user.id}</td>
            <td>{user.name}</td>
            <td>{user.email}</td>
          </tr>
        ))}
      </tbody>
    </table>
  );
}
```

Q. 6. What is the difference between PUT and PATCH? *(5 Marks)*

Answer:

Put:

1. Replaces the entire resource with new data.
2. Requires full resource data, even if only one field changes.
3. Update a full record

Patch:

4. Updates part of an existing resource.
5. Only requires the fields that need to be updated.
6. Update only given record