TEST SOLUTION
Granular Data

Submit By - RAJEEV ROY

Ques1. You encounter a bug that occurs inconsistently across different environments. What's the best approach to diagnose the issue?
Answer- Use logging and debugging tools to gather more information when the bug occurs.

Ques 2. What is the purpose of useEffect in React?
Answer- To perform side effects in a functional component.

Ques 3. What would be the best way to determine if a point lies inside a polygon in geospatial data?
Answer- Applying the ray-casting algorithm.

Ques 4. What is the difference between useEffect and useLayoutEffect in React?
Answer- useEffect runs after paint, while useLayoutEffect runs before paint.

Ques 5. Which of the following is NOT true about React keys?
Answer- Keys should be random values for better performance.

Ques 6. What will happen when you try to modify the state directly in React?

```
const [count, setCount] = useState(0);
count = 5;
```

Answer- It will cause a syntax error.

Ques 7. You have to integrate with a drone imaging tool called Loveland Imaging. They don't provide any API documentation. You have access to the tool and you want to integrate with it. How will you do it?
Answer- See network calls, look for tokens, api endpoints, check calling endpoints with tokens, and go from there.

Ques 8. Which of these techniques can be used to improve performance when rendering thousands of geospatial features on a map?
Answer- Clustering nearby points to reduce visible features.

Ques 9. You're building a web app that displays properties on a map and allows the user to visualize a buffer zone (e.g., a 100-meter radius) around each property. The buffer zones should be calculated in real-time when a property is clicked. Which approach would be most efficient for calculating and displaying the buffer zone ?
Answer- Use the turf.buffer() function to calculate buffer zones around the property polygons in the frontend.

Ques 10. You need to deliver a critical feature within a tight deadline, but it involves a technology you're unfamiliar with. What's your first step?
Answer- Research tutorials, documentation, and examples to quickly gain an understanding.

Ques 11. How can you prevent a component from re-rendering in React?
Answer- Return false from shouldComponentUpdate in a class component or use React.memo in a functional component.

Ques 12. You are building a property inspection tool that uses tiled map data (e.g., satellite images). The user frequently pans and zooms around the map. Which strategy would ensure the best performance while reducing unnecessary network requests?
Answer- Cache tiles client-side using Service Workers to reuse previously loaded tiles.

Ques 13. You encounter a critical bug that breaks production, and the error message is vague or non-existent. What's the most effective next step?
Answer- Roll back the breaking change, reproduce bug locally, push a fix.

Ques 14. Given a Polygon with vertices defined by (1,1), (5,1), (5,5), and (1,5), what would be the centroid of the polygon?
Answer- (3,3)

Ques 15. You receive geospatial data in the EPSG:4326 (WGS 84) coordinate system, but your frontend app uses a map that displays coordinates in EPSG:3857 (Web Mercator). Which of the following methods would correctly convert the data for display?
Answer- Use the proj4 library in the frontend to transform the EPSG:4326 coordinates to EPSG:3857 before rendering them.

Ques 16. Which of the following is NOT a common HTTP status code used in REST APIs?
Answer- 305

Ques 17. Given a GeoJSON object containing multiple polygons, which of the following code snippets correctly counts the number of connected components (i.e., distinct polygons or sets of connected polygons) in the object?

Answer- (C)

```
function countConnectedComponents(geojson) {
  const visited = new Set();
  let count = 0;

  const polygons = geojson.features.filter(feature => feature.geometry.type ===
'Polygon');

  function intersects(poly1, poly2) {
ct
    return turf.intersect(poly1, poly2);
  }

  function dfs(polygon, polygons) {
    visited.add(polygon);
    polygons.forEach(otherPolygon => {
      if (!visited.has(otherPolygon) && intersects(polygon, otherPolygon)) {
        dfs(otherPolygon, polygons);
      }
    });
  }

  polygons.forEach(polygon => {
    if (!visited.has(polygon)) {
      count += 1;
      dfs(polygon, polygons);
    }
  });

  return count;
}
```