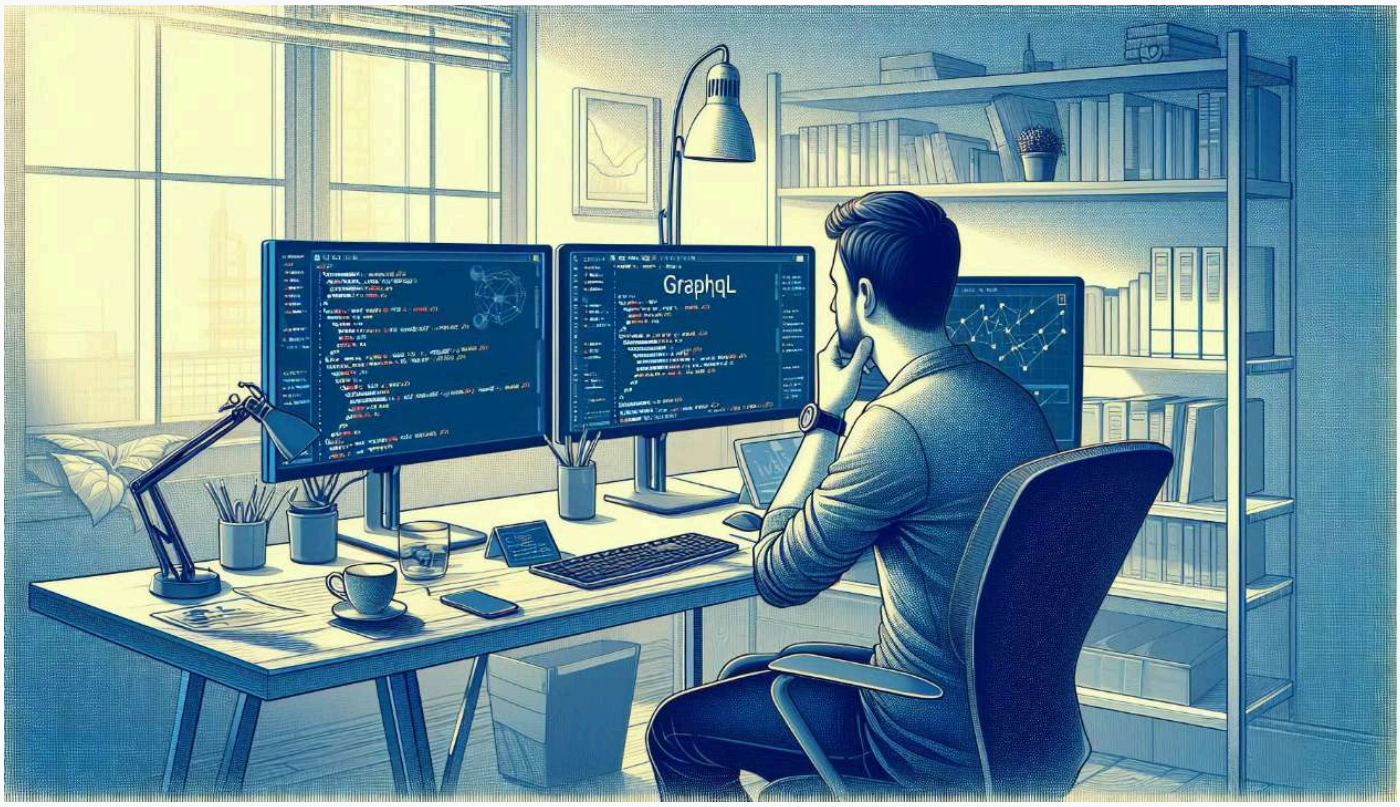
INTERVIEW QUESTIONS

GraphQL Testing FAQs: Interview Questions You Must Know

Md Eamin Hossain 10 Feb 2025 0 336 0 Share



1. What is GraphQL, and how does it differ from REST?

GraphQL is a query language for APIs that allows clients to request only the data they need, reducing data over- and under-fetching. Unlike REST, which relies on multiple endpoints, GraphQL provides a single endpoint for flexible queries and mutations. This results in more efficient data retrieval and better performance for complex applications.

2. Why is testing important in GraphQL applications?

Testing ensures that GraphQL APIs function correctly, handle queries efficiently, return the expected data, and maintain security. It also helps prevent breaking changes, verify data integrity, and ensure API stability as the application scales.

3. What are the main types of testing used in GraphQL?

- **Unit Testing**: Tests individual resolvers and schema components.

- **Security Testing**: Identifies vulnerabilities like unauthorized data access, query injection attacks, and excessive data exposure.

4. How do you perform unit testing for GraphQL resolvers?

Use a testing framework like **Jest** or **Mocha** along with **mock data**. Example:

```
import { getUser } from '../resolvers';
test('should return user data', async () => {
  const result = await getUser(null, { id: 1 });
  expect(result).toHaveProperty('name');
});
```

5. What tools are commonly used for GraphQL testing?

- **Jest** – Unit testing for resolvers and schema.
- **Apollo Test Client** – Integration testing for GraphQL APIs.
- **Postman/Insomnia** – Manual API testing.
- **GraphQL Playground** – Interactive query testing.
- **K6 & Artillery** – Performance testing.

6. How do you test a GraphQL query using Postman?

1. Open **Postman** and select **POST** request.
2. Enter the GraphQL API endpoint.
3. In the **body**, select **GraphQL** and write a query like:

```
{
  user(id: 1) {
    name
    email
  }
}
```

7. What is mocking in GraphQL testing, and why is it useful?

Mocking simulates API responses without querying a real database, making tests faster and independent of backend changes. Example using **Apollo Server Mocking**:

```
const { ApolloServer } = require('@apollo/server');
const { makeExecutableSchema } = require('@graphql-tools/schema');
const { addMocksToSchema } = require('@graphql-tools/mock');

const schema = makeExecutableSchema({ typeDefs });
const mockedSchema = addMocksToSchema({ schema });
```

8. How can you validate GraphQL schema during testing?

Use graphql from graphql-js to validate schema correctness:

```
);
console.log(schema.getQueryType().getFields());
```

9. What are the best practices for writing GraphQL tests?

- Use **mocking** for unit tests.
- Perform **schema validation** before testing.
- Implement **authentication testing** for secured APIs.
- Cover **edge cases** in query and mutation testing.
- Use **performance testing** for complex queries.

10. How do you handle authentication in GraphQL tests?

Pass authentication headers during tests:

```
const response = await request(app)
  .post('/graphql')
  .set('Authorization', `Bearer ${token}`)
  .send({ query: "{ user { name email } }" });
```

11. How do you test GraphQL mutations?

Use Jest with Apollo's request package:

```
const mutation = `
  mutation {
    createUser(name: "John", email: "john@example.com") {
      id
      name
    }
  }
`;
const response = await request(app).post('/graphql').send({ query: mutation });
expect(response.body.data.createUser.name).toBe("John");
```

12. How do you test query complexity in GraphQL?

Use tools like **graphql-depth-limit** to prevent excessively deep queries that can overload the server. This ensures that users cannot send recursive or deeply nested queries that might cause performance issues or Denial-of-Service (DoS) attacks.

13. How do you test GraphQL caching mechanisms?

Check response headers and measure query response times before and after enabling caching. Tools like Apollo Cache, Redis, or CDN caching can be used to ensure caching is working effectively. Cache-hit and cache-miss metrics should be monitored to evaluate performance improvements.

Ensure that users only access permitted data by sending requests with varying permissions and asserting correct authorization responses.

15. How do you test GraphQL error handling?

Send invalid queries and check for proper error messages. Example:

```
const response = await request(app).post('/graphql').send({ query: "{ unknownField }"
});
expect(response.body.errors[0].message).toContain("Cannot query field");
```

16. How do you test for GraphQL injection attacks?

Attempt to inject malicious queries and ensure the API properly validates and sanitizes input. Example:

```
{
  user(id: "1 OR 1=1") {
    name
  }
}
```

If the API returns unexpected results, it indicates vulnerability to injection attacks.

17. How do you ensure backward compatibility in GraphQL?

Use versioned schemas and deprecate old fields gradually instead of removing them suddenly. Implement feature flags and maintain compatibility layers to allow smooth transitions.

18. How do you test for excessive data fetching in GraphQL?

Implement query cost analysis and test responses to prevent over-fetching of unnecessary data. GraphQL query complexity analysis tools help monitor and limit the resources consumed by complex queries.

19. How do you implement rate limiting tests in GraphQL?

Simulate high query volumes and check for HTTP 429 status responses. Tools like Redis-based rate limiting or API Gateway rate limiting can be used to monitor and control request limits.

20. How do you test GraphQL APIs in a CI/CD pipeline?

Integrate automated test scripts into CI/CD tools like GitHub Actions, Jenkins, or GitLab CI/CD. Use containerized environments with tools like Docker and Kubernetes to run automated GraphQL tests efficiently.

21. How do you test GraphQL subscriptions?

- Example:

```
const subscription = gql`
  subscription {
    newMessage {
      content
    }
  }
`;
expect(subscriptionResult.data.newMessage.content).toBe("Hello World");
```

22. How do you handle pagination testing in GraphQL?

Pagination testing ensures proper data chunking and performance optimization:

- Verify limit and offset or cursor mechanisms.
- Test with small and large datasets to check boundary conditions.
- Example:

```
{
  users(first: 5, after: "cursor123") {
    edges {
      node {
        name
      }
    }
  }
}
```

23. How do you measure GraphQL performance?

- Use tools like **Apollo Tracing**, **GraphQL Metrics**, or **New Relic**.
- Measure query response time and resolver execution time.
- Identify bottlenecks using profiling tools and optimize queries.

24. What are GraphQL query batching tests?

Query batching reduces multiple requests into one:

- Test batch processing by sending multiple queries in a single request.
- Ensure the response includes all requested queries.
- Verify no data corruption or performance issues occur.

25. How do you test GraphQL rate limiting?

- Simulate multiple requests from the same client.
- Use tools like **K6** or **JMeter** to exceed API limits.
- Expect HTTP 429 Too Many Requests responses when limits are reached.
- Example:

26. How do you verify GraphQL field deprecations?

- Ensure deprecated fields return warnings but remain functional.
- Example:

```
{
  oldField @deprecated(reason: "Use newField instead")
}
```

- Check logs for deprecation messages and ensure backward compatibility.

27. How do you test GraphQL file uploads?

GraphQL supports file uploads via **Apollo Upload Client**:

- Test single and multiple file uploads.
- Verify correct MIME types and file size restrictions.
- Example:

```
const mutation = `
  mutation($file: Upload!) {
    uploadFile(file: $file) {
      url
    }
  }
`;
```

28. How do you test GraphQL multi-tenancy?

Multi-tenancy ensures data segregation between tenants:

- Use different authorization tokens per tenant.
- Verify that tenant A cannot access tenant B's data.
- Example:

```
const response = await request(app)
  .post('/graphql')
  .set('Authorization', `Bearer tenantAToken`)
  .send({ query: "{ getTenantData }" });
expect(response.body.data).not.toContain("Tenant B's Data");
```

29. How do you test GraphQL custom directives?

Custom directives enhance GraphQL functionality:

- Test directive behavior on queries and mutations.
- Example directive for authentication:

```
type Query {
  secretData: String @auth
}
```

- Use **structured logging** to capture API errors.
- Test error messages for proper formatting and security compliance.
- Example:

```
expect(response.body.errors[0].message).toBe("Unauthorized access");
```

testing qa sqa testingtool graphql rest endtoende2etesting

Share your thoughts Or Start discussion

Related Blogs

INTERVIEW QUESTIONS 0 0 217

Smoke Testing Interview Questions: Most Asked QA Topics

1. What is Smoke Testing?Answer: Smoke Testing is a type of software testing per

ridwan
13 Mar 2025

INTERVIEW QUESTIONS 0 0 366

Mastering Katalon Studio: Common Interview Questions Explained

1. What is Katalon Studio?Answer:Katalon Studio is an automated testing tool f

Sebastian Leon
17 Feb 2025

Popular Tags

View All

Popular Post

Can a Software Tester Become a Game Tester? Here's What You Need t...

As the gaming industry continues to grow, fueled by innovations in virtual reali

Understanding Java Object-Oriented Programming (OOP) Concepts

Java is a powerful and widely used programming language known for its versatilit

Essential Bugs to Check for in Game Testing: A Guide for Beginners

Game testing is crucial to ensure a smooth, engaging, and bug-free experience fo

JMeter: Short technique for Generating an HTML load test report using...

Pre-requisites:Install Java:Java Version: "1.8.0_291" or higher (minimum require

View All

Popular Discussion

01 Top Software Testing Interview Questions and Expert Tips from QA Leaders

02 AI tools for QA engineer

03 What is SQL?

04 Appium, WebDriver

05 What are the most effective strategies you've found for balancing speed and...

View All

QA Brains is the ultimate QA community to exchange knowledge, seek advice, and engage in discussions that enhance Quality Assurance testers' skills and expertise in software testing.

QA Topics

Web Testing

Interview Questions

Game Testing

See more →

Quick Links

Discussion

About Us

Terms & Conditions

Privacy Policy

Follow Us

For Support

support@qabrains.com