**Purpose of the Document**

Creating a comprehensive test strategy for the JDoodle website involves defining the scope, objectives, and testing approach. Here's a general outline for a test strategy:

1. Objective:

* This document focuses on strategies around functionality, usability, security, and performance of the JDoodle website.

1. Scope:

* Test across major web browsers (Chrome, Edge, Firefox, Safari).
* Validate the website's responsiveness on different devices (desktops, tablets, and mobile phones).
* Verify the core features such as code execution, language support, and API integration.
* Test security aspects related to user data and code execution.

1. Testing Types:

* Functional Testing:
* Validate code execution across various programming languages.
* Verify the correctness of compiler functionality.
* Test features like syntax highlighting, auto-complete, and code.
* Usability Testing:
* Evaluate the user interface for consistency and ease of use.
* Check the user experience for writing, executing, and saving code.
* Test the website's navigation and overall user flow.
* Security Testing:
* Assess the website for vulnerabilities such as injection attacks.
* Ensure secure data transmission and storage.
* Validate user authentication and authorization mechanisms.
* Performance/Stress Testing:
* Evaluate website performance under different loads.
* Assess response times for code execution.
* Check the website's scalability and resource usage.
* Compatibility Testing:
* Verify cross-browser compatibility.
* Test on various devices to ensure responsiveness.
* Regression Testing:
* Run regression tests to ensure that new features or updates do not introduce issues.
* API Testing:
* Test the API endpoints for validating functionality, Response time and security related testing.

1. Test Environment:

* Ensure the test environment reflects the production environment closely.
* Set up GitHub configurations for covering cross browser and parallel runs.
* Use emulators for Jdoodle Mobile APP testing.

1. Test Data:

* Use a variety of test data, including different programming languages, code snippets, and user scenarios.
* Include invalid data to test error handling.

1. Test Automation:

* Implement test automation for repetitive and critical test scenarios.
* Use automation tools compatible with web technologies.

1. Defect Reporting & Monitoring:

* Clearly document and prioritize defects.
* Provide detailed steps to reproduce issues and include screenshots or videos when necessary.
* Implement monitoring tools to track system performance and user behavior

1. Documentation:

* Maintain test documentation, including test cases, test scripts, and test data.
* Keep API documentation up-to-date.

1. Collaboration:

* Foster collaboration between development, testing, and other stakeholders.
* Conduct regular review meetings to address issues and improvements.

1. Continuous Improvement:

* Gather feedback from testing cycles and implement improvements.
* Stay informed about the latest web technologies and testing tools.