TestCopilot Codebase Analysis

Codebase reliability rating: E - High Risk (score: 37.0)

Analyzed 2 test files.
 2 file(s) contain patterns that may cause flakiness or unreliable results.

Improving flagged files will make your test suite more stable and trustworthy for the whole team.

File: C:\Users\ishie\Documents\Projects\testcopilot\test-files\sample.cy.js

Checker: raceConditionAnalysis
Score: 26.7 (E - High Risk)

Issue	Severity	Line	Context	Explanation	Suggested Fix
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	14	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	23	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	32	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	41	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.

Issue	Severity	Line	Context	Explanation	Suggested Fix
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	50	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.

This test file was rated E - High Risk for async reliability (score: 26.7).

It contains patterns that may cause flakiness, such as hardcoded waits, missing intercepts, or UI actions without follow-up checks.

These issues mean the tests might pass even when the app is broken, or fail when the app is actually working — leading to wasted time debugging false results.

Improving these tests will make them more stable, trustworthy, and maintainable for both developers and QA teams.

File: C:\Users\ishie\Documents\Projects\testcopilot\test-files\sample.cy copy.js

Checker: raceConditionAnalysis

Score: 47.3 (D - Moderate Risk)

Issue	Severity	Line	Context	Explanation	Suggested Fix
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	14	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.
User action "cy.click()" is not followed by a check to confirm the app responded.	medium	23	cy.get('#notifications')	The test simulates a user interaction using "cy.click()", but it doesn't verify whether the app responded correctly. Without a follow-up check, the test might pass even if the application fails to react — leading to false confidence in test results.	After calling "cy.click()", add a UI assertion such as "cy.get(...).should(...)" to confirm the expected change happened. This helps ensure the app responded as intended.

This test file was rated D - Moderate Risk for async reliability (score: 47.3).

It contains patterns that may cause flakiness, such as hardcoded waits, missing intercepts, or UI actions without follow-up checks.

These issues mean the tests might pass even when the app is broken, or fail when the app is actually working — leading to wasted time debugging false results.

Improving these tests will make them more stable, trustworthy, and maintainable for both developers and QA teams.