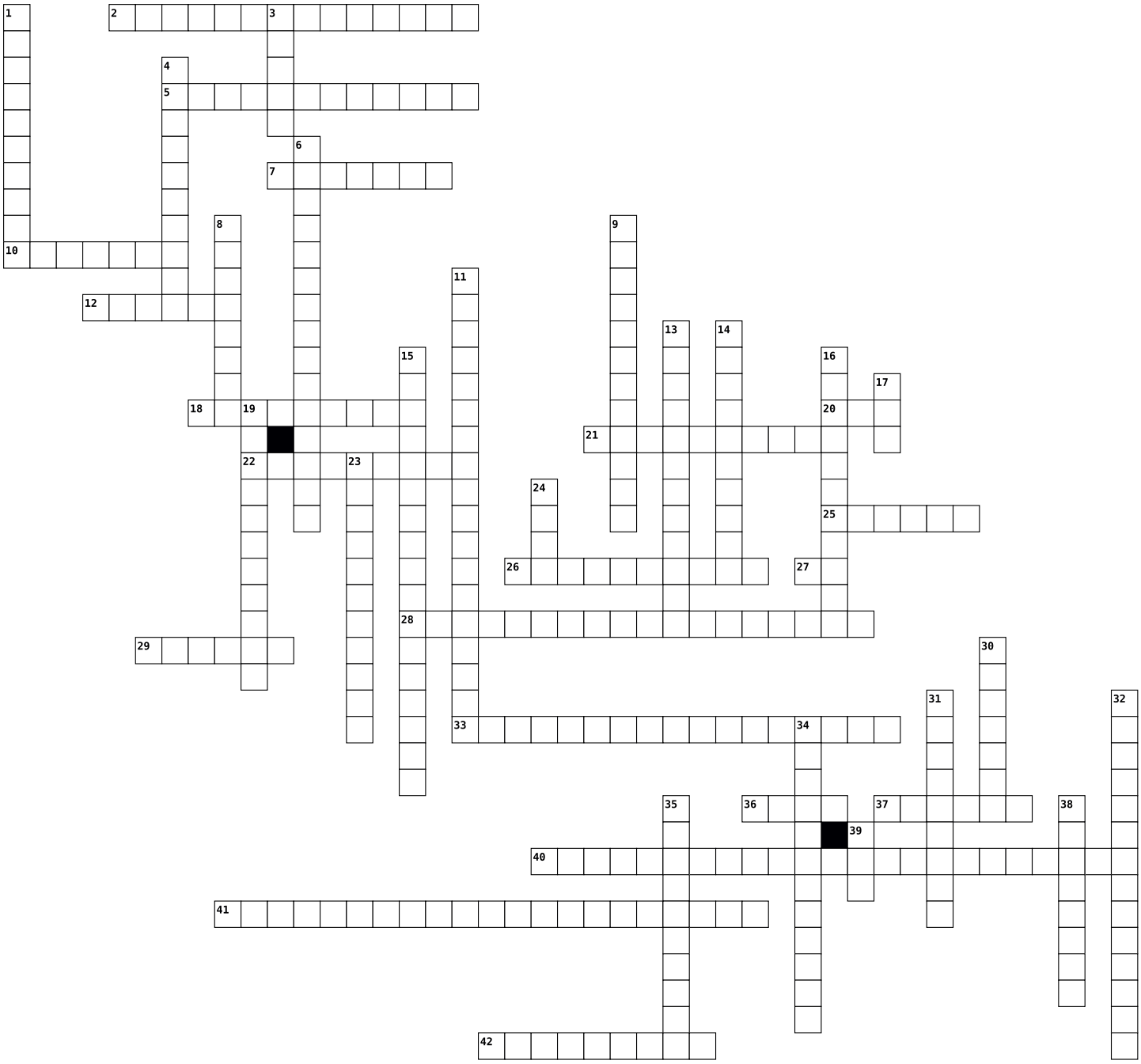
Software Testing



Across

2. Real-world sensitive information.

5. A metric indicating how much of the source code is exercised by automated tests.

7. An anti-pattern where teams test at different levels in isolation, creating redundancy and limited integration.

10. A domain-specific language used to write structured, human-readable test scenarios.

12. A testing framework for Python that supports simple unit testing and plugins like Hypothesis.

18. Testing focused on evaluating user experience, accessibility, and usability.

Down

1. A test smell caused by embedding fixed values directly in tests, making them harder to maintain.

3. A commercial tool for model-based test automation.

4. A level of testing that validates the system meets business requirements and user needs.

6. A technique where the code is slightly modified to ensure tests detect the changes, measuring their effectiveness.

8. A testing method that requires internal knowledge of the codebase.

9. Tests tightly coupled to implementation details, often breaking after minor code refactoring.

20. A methodology where tests are written before the code they are meant to validate.

21. A generic term for various test objects such as mocks, stubs, and fakes used to isolate code.

22. A strategy that promotes early testing in the development lifecycle to catch bugs sooner.

25. An open-source tool used for automating mobile applications across platforms.

26. Outdated or deprecated tests that no longer reflect system behavior or requirements.

27. A development practice that integrates code changes frequently and runs automated tests to catch issues early.

28. The practice of creating, maintaining, and managing data needed for automated tests.

29. A level of testing that validates the complete and integrated software system.

33. A Python-based tool used for validating, documenting, and profiling data.

36. A test double that verifies interactions by checking if methods were called with expected parameters.

37. A type of performance testing that evaluates system behavior under extreme conditions.

40. A technique that divides input data into valid and invalid partitions to reduce the number of test cases.

41. A test design technique focused on the values at the edges of input ranges.

42. A subset of tests run to verify basic functionality before deeper testing.

11. A simultaneous learning and testing approach where test design and execution happen in real time.

13. A pattern in which an autonomous component monitors and acts on system state for testing or resilience purposes.

14. A pattern that encapsulates UI element logic in classes to make UI tests more maintainable.

15. Re-running tests to ensure recent code changes haven’t broken existing functionality.

16. A level of testing where multiple components or systems are tested together to verify interactions.

17. A collaborative approach where business, development, and testing use natural language examples to define expected behavior.

19. Mechanisms for determining whether a test has passed or failed.

23. A visual model that emphasizes unit tests at the base and fewer UI tests at the top, optimized for cost, speed, and feedback.

24. A level of testing that targets individual functions or methods in isolation.

30. A JavaScript end-to-end testing tool commonly used for web applications.

31. A statement in a test that checks if a specific condition holds true.

32. A dependency that allows creation of mocks and stubs for testing.

34. An anti-pattern where the majority of tests are GUI or manual with very few unit tests.

35. Tests that fail intermittently without code changes due to poor design or external dependencies.

38. A testing technique that minimizes the number of test cases by combining input parameters in every possible pair.

39. A test structuring pattern based on Arrange, Act, and Assert phases.