Masterclass Certificate in AI-Driven Release Management

Testing

Agile Testing: A type of testing that follows Agile methodologies, which emphasize flexibility, collaboration, and customer satisfaction. Agile testing is iterative, with testing activities happening concurrently with development activities. It prioritizes automation, continuous feedback, and the ability to adapt to changing requirements.

Artificial Intelligence (AI): A branch of computer science that focuses on creating intelligent machines that can learn from data, make decisions, and perform tasks that typically require human intelligence. AI-driven testing uses AI technologies like machine learning, natural language processing, and computer vision to automate and optimize testing processes.

Behavior-driven Development (BDD): A software development methodology that encourages collaboration between developers, QA, and non-technical stakeholders to define the expected behavior of a system. BDD uses a simple, domain-specific language (Gherkin) to describe scenarios, making it easier for non-technical stakeholders to understand the tests and provide feedback.

Continuous Integration (CI): A software development practice where developers frequently merge their code changes into a shared repository, triggering automated builds and tests to ensure the software remains in a releasable state. CI helps catch integration issues early, reducing the time and effort required to resolve them.

Continuous Deployment (CD): A software development practice that automatically deploys passing builds to production or staging environments, eliminating the need for manual intervention. CD enables rapid release cycles, allowing organizations to deliver value to customers quickly and frequently.

Continuous Testing: A testing approach that is integrated into the continuous integration and continuous delivery (CI/CD) pipeline. Continuous testing involves running automated tests at every stage of the software development lifecycle, providing rapid feedback on the quality and functionality of the software.

Defect Management: The process of tracking, prioritizing, and resolving defects or bugs in software. Defect management includes identifying the root cause of defects, determining their severity, and tracking them through resolution. Effective defect management helps ensure the timely delivery of high-quality software.

DevOps: A set of practices that combines software development (Dev) and IT operations (Ops) to improve collaboration, automation, and communication between teams. DevOps aims to shorten the software development lifecycle and provide rapid, frequent delivery of high-quality software.

Exploratory Testing: A testing approach that emphasizes discovery, learning, and ad-hoc testing. Exploratory testing is often used when formal test cases are not available or when testers want to investigate potential issues or risks in the software. It involves simultaneous test design, execution, and learning, often uncovering issues that scripted testing may miss.

Gherkin: A business-readable domain-specific language used in behavior-driven development (BDD) to describe scenarios. Gherkin uses a simple syntax to define preconditions, actions, and expected outcomes, making it easy for stakeholders to understand and provide feedback on the tests.

Machine Learning (ML): A subset of artificial intelligence (AI) that focuses on developing algorithms that can learn from and make predictions or decisions based on data. Machine learning can be used in testing to identify patterns, detect anomalies, and make intelligent decisions about test execution and analysis.

Regression Testing: A type of testing that verifies that modifications to a system have not introduced new defects or caused existing features to fail. Regression testing is essential in maintaining the quality of software throughout its lifecycle, especially when changes are made to the codebase.

Shift-left Testing: A testing strategy that moves testing activities earlier in the software development lifecycle, closer to the requirements and design phases. Shift-left testing aims to catch defects and issues earlier, reducing the time and cost required to resolve them.

Test Automation: The use of software tools and frameworks to automate repetitive or time-consuming testing tasks. Test automation can improve testing efficiency, consistency, and coverage, enabling teams to test more frequently and thoroughly.

Test Case Management: The process of creating, organizing, and maintaining test cases to ensure comprehensive testing of software. Test case management includes designing test cases, linking them to requirements, and tracking their execution and results.

Test Data Management: The process of creating, maintaining, and protecting test data required for testing activities. Test data management ensures that test data is accurate, secure, and representative of production data, enabling teams to perform effective tests and maintain data privacy and compliance.

Test Design: The process of defining test cases, scenarios, and test plans to ensure comprehensive testing of software. Test design includes understanding requirements, identifying test objectives, and selecting appropriate test techniques and strategies.

Test Execution: The process of running test cases, scenarios, or test scripts and recording the results. Test execution can be manual or automated and includes tracking pass/fail status, defect identification, and test environment management.

Test Harness: A suite of software tools and frameworks that enable the execution of tests and the evaluation of their results. A test harness typically includes test runners, assertion libraries, and reporting tools.

Test Plan: A document that outlines the testing approach, resources, schedule, and objectives for a specific project or release. A test plan typically includes details about test environments, test cases, test data, and risk assessments.

Test Pyramid: A visual representation of the recommended distribution of different types of tests in a testing strategy. The test pyramid typically includes a broad base of unit tests, a smaller layer of integration tests, and a narrow peak of end-to-end or UI tests.

Test Reporting: The process of communicating testing results, status, and metrics to stakeholders. Test reporting includes creating test reports, dashboards, and other visualizations that help stakeholders understand the quality and progress of testing activities.

Test Strategy: A high-level plan that outlines the testing approach, methods, tools, and resources for a specific project or release. A test strategy typically includes details about test levels, test types, test environments, and risk assessments.

Test Suite: A collection of test cases, scenarios, or test scripts that are executed together as a single unit. A test suite typically focuses on a specific functionality or feature of the software.

Unit Testing: A type of testing that focuses on testing individual components or units of software in isolation. Unit testing is typically performed by developers and is an essential part of test-driven development (TDD) and continuous integration (CI) practices.