MODULE 5: USABILITY, ACCESSIBILITY & PERFORMANCE TESTING

SPECIAL SELECTED TOPICS

PERFORMANCE TESTS - BASIC OUTLINE

- It is one type of non-functional testing aimed at determining how a system performs in terms of responsiveness, stability, scalability, and speed under a particular workload.
- It ensures that the application behaves well under expected (and sometimes unexpected) conditions before it goes into production.
- Performance is a component of a user's "good experience" and forms part of an acceptable quality level.

PERFORMANCE TESTS - CRITICAL DIMENSIONS

- There are three Critical dimensions of Performance Testing
  - Time Behavior
  - Resource Utilization
  - Capacity Monitoring

PERFORMANCE TESTS - CRITICAL DIMENSIONS - TIME BEHAVIOR

- **Time behaviour** helps determine how fast a system responds and how long it can sustain certain behaviors.

Metric	Description
Response Time	The total time from the moment a user sends a request until the system responds completely.
Latency	The time it takes for the first byte of the response to return after the request.
Think Time	Simulated pause between user actions in scripts (mimicking real user delays).
Ramp-up Time	The time it takes to gradually increase the number of virtual users in a test.
Test Duration	The total time over which the performance test is run.

PERFORMANCE TESTS - CRITICAL DIMENSIONS - RESOURCE UTILIZATION

- **Resource Utilisation** helps determine system components consumed when processing user requests.

Resource	Description
CPU	Measures how much processing power is used during the test. High CPU usage can be a bottleneck.
Memory (RAM)	Important for tracking memory leaks or inefficient memory handling.
Disk I/O	Read/write operations that affect performance, especially in databases or file-heavy systems.
Network Bandwidth	Relevant in distributed systems or APIs; tests the system's ability to handle network traffic.
Database Connections	Number of concurrent DB sessions; may require tuning of connection pools.

PERFORMANCE TESTS - CRITICAL DIMENSIONS - RESOURCE UTILIZATION

- **Resource Utilisation** helps determine system components consumed when processing user requests.

Resource	Description
CPU	Measures how much processing power is used during the test. High CPU usage can be a bottleneck.
Memory (RAM)	Important for tracking memory leaks or inefficient memory handling.
Disk I/O	Read/write operations that affect performance, especially in databases or file-heavy systems.
Network Bandwidth	Relevant in distributed systems or APIs; tests the system's ability to handle network traffic.
Database Connections	Number of concurrent DB sessions; may require tuning of connection pools.

PERFORMANCE TESTS - CRITICAL DIMENSIONS - CAPACITY MONITORING

- Capacity refers to the maximum load a system can handle while still meeting performance expectations. It helps answer:
  - “How many users can the system support concurrently before degrading?”
  - “What’s the maximum transaction throughput?”

Term	Description
Peak Load Capacity	The highest number of simultaneous users the system can handle without performance degradation.
Throughput Limit	The maximum number of transactions or requests per second the system can process efficiently.
Scaling Threshold	The point at which horizontal or vertical scaling is required to maintain performance.
Utilization Threshold	The resource usage limit (e.g., CPU ~80%) beyond which performance may degrade.

PERFORMANCE TESTS - TYPES

Type	Purpose
Load Testing	Checks system behavior under expected load (e.g., 1000 users simultaneously).
Stress Testing	Determines how the system performs under extreme conditions, often beyond expected limits.
Spike Testing	Tests the system's response to sudden large spikes in load.
Endurance (Soak) Testing	Checks for memory leaks or performance degradation over extended periods.
Scalability Testing	Evaluates how well the system scales with increasing load (horizontal or vertical).

GOALS OF PERFORMANCE TESTING

- Validate speed – Is the application fast enough?
- Validate scalability – Can the application scale when user load increases?
- Validate stability – Is the application stable under stress or prolonged use?
- Identify bottlenecks – Where is the system slowing down?

PERFORMANCE TESTING PROCESS

- Requirements Gathering
  - Identify performance criteria: response time, throughput, CPU/memory usage, etc.
  - Define acceptable thresholds (e.g., login < 2 seconds for 95% of users).
- Test Environment Setup
  - Mirror production environment.
  - Set up monitoring tools (e.g., Grafana, Prometheus, New Relic).
- Test Planning and Scripting
  - Use tools like JMeter, Gatling, Locust, or k6 to script user flows.
  - Parameterise inputs and simulate realistic user behaviour.

PERFORMANCE TESTING PROCESS

- Test Execution
  - Run tests under different loads (baseline, peak, stress).
  - Monitor system health and collect metrics.
- Result Analysis
  - Identify bottlenecks (e.g., slow DB queries, poor caching).
  - Compare results against SLAS or benchmarks.
- Tuning and Optimisation
  - Adjust server configs, database indices, caching strategies, etc.
  - Re-test to validate improvements.
- Reporting
  - Share results with stakeholders.
  - Highlight risks and mitigation recommendations.

BEST PRACTICES

- Start early (Shift Left): Do not wait until late stages to test performance.
- Test different scenarios: Normal, peak, and unexpected load.
- Isolate tests: Run one type at a time to identify specific issues.
- Automate where possible: Integrate with CI/CD pipelines.
- Monitor all layers: App, DB, cache, network, OS.

POPULAR PERFORMANCE TESTING TOOLS

Tool	Language	Notes
Apache JMeter	Java	GUI-based, widely used, supports plugins.
Gatling	Scala	DSL scripting, great for CI/CD.
k6	JavaScript	Developer-friendly, CLI-first, great for automation.
Locust	Python	Python-based, distributed load testing.
Artillery	JavaScript	Simple YAML config, supports HTTP and WebSocket.
LoadRunner	-	Commercial, enterprise-grade.

ACCESSIBILITY TESTING

- is a type of non-functional testing that ensures your application is usable by people with disabilities, such as visual, auditory, motor, or cognitive impairments.
- The goal is to make digital content inclusive and compliant with accessibility standards like WCAG (Web Content Accessibility Guidelines).

WHY ACCESSIBILITY TESTING MATTERS

- Legal Compliance: Many countries (e.g., under ADA, Section 508, EN 301 549) require accessible digital content.
- Inclusive User Experience: Ensures everyone, including users with disabilities, can access your content.
- SEO & Performance Benefits: Many accessibility improvements (like alt text) also help search engines.
- Brand Reputation: Shows your organisation values inclusivity and equal access.

WHAT ACCESSIBILITY TESTING COVERS

Disability Type	Testing Focus
Visual (e.g., blindness, low vision)	Screen reader compatibility, color contrast, keyboard navigation
Auditory (e.g., deafness)	Captioning of videos, transcripts for audio
Motor (e.g., limited hand movement)	Keyboard-only navigation, focus indicators
Cognitive (e.g., dyslexia, ADHD)	Clear layouts, predictable UI, simple language

CORE ACCESSIBILITY STANDARDS

- WCAG 2.1: Most common guideline. Levels:
  - A – Minimal compliance
  - AA – Most widely adopted standard
  - AAA – Highest standard (rare in public-facing apps)
- Key principles: POUR
  - Perceivable – Information must be visible and understandable (e.g., alt text).
  - Operable – UI can be used with keyboard or assistive devices.
  - Understandable – Content is readable, and UI behaves predictably.
  - Robust – Compatible with current and future assistive technologies.
-

HOW TO CONDUCT ACCESSIBILITY TESTING

- Manual Testing
  - Keyboard Navigation: Ensure all functions are accessible via keyboard (Tab, Enter, Esc).
  - Screen Reader Testing: Use tools like NVDA, VoiceOver, or JAWS to test text-to-speech.
  - Colour Contrast Checks: Ensure text/background contrast meets WCAG 2.1 standards.
  - Focus Management: Focus should be visible and move logically.
  - Form Validation: Ensure error messages are accessible and meaningful.

HOW TO CONDUCT ACCESSIBILITY TESTING - AUTOMATED TESTING TOOLS

Tool	Description
axe-core	Open-source engine used in many tools (also has browser extension).
Lighthouse (Chrome DevTools)	Provides an accessibility score and improvement suggestions.
WAVE	Web Accessibility Evaluation Tool – highlights accessibility issues on pages.
Pa11y	CLI for automated accessibility testing (uses axe-core).
Tenon, Deque, Siteimprove	Paid platforms with enterprise features.

ACCESSIBILITY TESTING IN CI/CD

- You can automate accessibility tests as part of your development pipeline:
  - Integrate Axe-Core or Pa11y in unit/E2E tests.
  - Use GitHub Actions to trigger accessibility scans on pull requests.
  - Generate reports and fail builds on critical violations.

BEST PRACTICES OF ACCESSIBILITY TESTING

- Use semantic HTML (e.g., <button>, <nav>) for screen reader support.
- Always include alt text for images.
- Ensure labels are properly associated with input fields.
- Maintain logical tab order and visible focus states.
- Avoid auto-playing media and flashing content.
- Don’t rely on color alone to convey information.