**Manual Testing Cheat Sheet (Complete)**

**1. Types of Testing**

* **Unit Testing:** Testing individual units/components in isolation.
* **Integration Testing:** Verifying how different modules interact.
* **System Testing:** Testing the complete integrated system.
* **Acceptance Testing:** Ensuring the system meets business requirements (UAT).
* **Regression Testing:** Ensuring no new bugs have been introduced after changes.
* **Exploratory Testing:** Unscripted testing based on the tester’s experience.
* **Sanity Testing:** Quick checks after minor code changes to verify critical functionality.
* **Smoke Testing:** Testing the most critical features to ensure basic functionality before deeper testing.
* **Usability Testing:** Validating the ease of use and user-friendliness of the system.
* **Performance Testing:** Checking the system's responsiveness, stability, and scalability.
* **Load Testing:** Assessing system performance under expected user loads.
* **Stress Testing:** Evaluating how the system behaves under extreme conditions.
* **Security Testing:** Ensuring the application is secure from vulnerabilities like SQL injection, XSS, etc.
* **Localization Testing:** Checking if the system adapts correctly to different languages and regional settings.
* **Internationalization Testing:** Verifying the system's readiness for multiple languages, currencies, time zones, etc.
* **Ad-Hoc Testing:** Testing without any formal test plans or cases, often conducted spontaneously.
* **End-to-End Testing:** Testing the complete flow of the application from start to finish.
* **Alpha Testing:** Performed internally by the development team before releasing to customers.
* **Beta Testing:** Conducted by real users in the production environment.
* **Installation Testing:** Verifying that the software installs and uninstalls correctly on different platforms.
* **Compatibility Testing:** Ensuring the software works across different browsers, devices, and platforms.
* **A/B Testing:** Testing two versions of the application to determine which performs better.
* **Monkey Testing:** Random testing of the application with no specific test cases, trying to break it.

**2. Advanced Test Case Design Techniques**

* **Equivalence Partitioning:** Dividing input data into valid and invalid partitions.
* **Boundary Value Analysis:** Focusing on the edge cases of input ranges.
* **Error Guessing:** Anticipating where errors might occur based on past experience.
* **State Transition Testing:** Testing different states of the system and transitions between them.
* **Decision Table Testing:** Creating a decision table to map conditions and actions.
* **Use Case Testing:** Testing based on real-world user workflows and scenarios.
* **Pairwise Testing:** Testing combinations of input values in pairs to minimize total test cases.
* **Orthogonal Array Testing:** Systematically selecting test cases that cover all parameter combinations with minimum effort.
* **Graph-Based Testing:** Using nodes and edges to model and test possible interactions in the system.
* **Domain Testing:** Testing the domain (or range) of input values.
* **Error Seeding:** Intentionally adding errors into the application to test the detection of defects.
* **Back-to-Back Testing:** Comparing outputs from two versions of a system (or two systems) for the same input.

**3. Test Case Writing Best Practices**

* **Unique ID:** Use a unique ID for each test case.
* **Test Case Title:** Clearly describe the objective of the test.
* **Preconditions:** Specify the conditions or data setup required before executing the test.
* **Test Steps:** Provide clear, detailed, and numbered steps for execution.
* **Test Data:** Include specific data (e.g., user inputs) required for the test.
* **Expected Result:** Define the expected behavior/output.
* **Actual Result:** Capture the actual outcome during test execution.
* **Pass/Fail Status:** Mark if the test passed or failed.
* **Postconditions:** Define the state of the system after the test is executed.
* **Priority:** Assign a priority to the test case (High, Medium, Low).
* **Execution Time:** Estimate how long the test will take to execute.

**4. Test Execution Strategies**

* **Environment Setup:** Ensure test environments are identical to production.
* **Test Data Preparation:** Use realistic data that reflects production scenarios.
* **Test Prioritization:** Test high-risk and critical functionalities first.
* **Parallel Testing:** Run tests concurrently when applicable (especially for independent modules).
* **Cross-Browser Testing:** Test across all supported browsers and devices.
* **Continuous Testing:** Perform testing as part of Continuous Integration (CI) pipelines to validate code on each commit.
* **Retesting:** Retest failed test cases after bug fixes to ensure resolution.
* **Regression Testing:** Conduct regression testing to check for new bugs after changes.
* **Session-Based Testing:** Time-box exploratory testing to focus on specific areas.
* **Defect Tracking:** Log detailed defects with reproduction steps, severity, priority, and screenshots/logs.

**5. Defect Lifecycle**

* **New:** When a defect is logged and awaiting review.
* **Assigned:** The defect is assigned to a developer.
* **Open:** The defect is being actively worked on.
* **Fixed:** The defect has been fixed by the developer.
* **Pending Retest:** Waiting for the QA team to verify the fix.
* **Retest:** QA tests the fix.
* **Closed:** The defect is verified as fixed and closed.
* **Reopen:** The defect is re-opened if the issue persists.
* **Rejected:** The defect is not valid or reproducible.
* **Deferred:** The defect is postponed to a future release or version.

**6. Exploratory Testing Techniques**

* **Session-Based Testing:** Plan and execute testing sessions with clear objectives and time limits.
* **Charter-Based Testing:** Define a testing charter with specific goals for exploratory sessions.
* **Bug Hunts:** Conduct bug hunts where multiple testers search for as many bugs as possible within a limited timeframe.
* **Mind Mapping:** Visualize the areas to explore by using mind maps to structure the testing process.
* **Heuristics-Based Testing:** Apply heuristics like SFDPO (Structure, Function, Data, Platform, Operations) to guide exploratory sessions.

**7. Team Collaboration & Communication**

* **Daily Stand-Ups:** Participate in daily stand-up meetings to discuss testing progress, roadblocks, and next steps.
* **Cross-Functional Collaboration:** Work closely with developers, product owners, and other stakeholders to clarify requirements and share feedback.
* **Pair Testing:** Test in pairs (tester-developer, tester-tester) to enhance collaboration and ensure no area is overlooked.
* **Defect Triage Meetings:** Attend triage meetings to prioritize defects and decide the severity, priority, and timelines for fixes.
* **Retrospectives:** After each release cycle, participate in retrospectives to discuss lessons learned and areas of improvement for future tests.

**8. Key Metrics for Manual Testing**

* **Test Coverage:** Percentage of requirements covered by test cases.
* **Defect Density:** Number of defects found per unit of code.
* **Test Execution Rate:** The percentage of test cases executed over time.
* **Defect Severity Distribution:** Categorization of defects based on severity (Critical, Major, Minor).
* **Defect Leakage Rate:** Number of defects found after release compared to the number found before release.
* **Test Case Effectiveness:** The ratio of passed test cases to total executed test cases.
* **Mean Time to Defect Detection:** Average time taken to discover defects from code release.

**9. Testing Tools & Utilities**

* **Defect Tracking Tools:** JIRA, Bugzilla, Trello, Asana.
* **Test Case Management:** TestRail, Zephyr, qTest, TestLink.
* **Cross-Browser Testing:** BrowserStack, SauceLabs.
* **API Testing Tools:** Postman, Insomnia, Swagger.
* **Screen Recording:** Loom, Snagit.
* **Mocking and Stubbing Tools:** Mockoon, WireMock.
* **Test Reporting Tools:** Allure, ReportPortal.
* **Performance Testing Tools:** JMeter, LoadRunner.

**10. Testing Checklists**

**Test Preparation Checklist**

* All requirements are documented and approved.
* Test cases are written, reviewed, and approved.
* Test data is ready and verified.
* Test environment is set up (including all dependencies).
* All necessary tools (bug trackers, collaboration tools) are available and functional.
* Smoke tests are prepared for initial testing.

**Post-Test Checklist**

* All defects are logged and tracked in the defect management tool.
* All critical test cases are executed.
* Test cases are updated based on feedback.
* All regression tests are run, and no new issues are found.
* A test closure report is prepared and shared with stakeholders.

**11. Manual Testing Principles**

* **Early Testing:** Begin testing as soon as possible in the development lifecycle.
* **Defect Clustering:** Most defects are found in a small number of modules (80/20 rule).
* **Pesticide Paradox:** Regularly update test cases to avoid redundancy.
* **Exhaustive Testing is Impossible:** Focus on critical functionality—testing everything is not feasible.
* **Testing Shows Presence of Defects:** Testing proves that defects exist, but cannot prove their absence.
* **Risk-Based Testing:** Prioritize testing based on areas of highest risk.

s