Software Testing

Assignment 1

Q1. Define and differentiate the terms defect, error, bug, and failure. Explain how each term is significant in the context of software testing.

Answer:

- **Error** → A mistake made by a developer while coding or designing. Example: writing wrong logic in code.
- **Defect** → A problem in the software product found during testing, caused by an error in coding or design. Example: the login button not working.
- **Bug** → A defect reported by a tester that developers need to fix. Example: the app crashes when uploading a file.
- **Failure** → When the software doesn't perform as expected in real use. Example: the banking app fails to process transactions correctly.

Significance:

- Errors happen in development.
- Defects are caught in testing.
- Bugs are logged and fixed.
- Failures happen in actual use if not detected earlier.
  Together, these terms help testers understand the quality and reliability of software.

Q2. Analyze why the cost of fixing defects increases as they are discovered later in the software development lifecycle. Use examples to support your explanation.

Answer:
The later a defect is found, the costlier it is to fix.

- **In Requirements stage** → Cheap to fix because it's just a document correction.
- **In Design stage** → Cost increases because design changes affect multiple modules.
- **In Coding stage** → Cost is higher because actual code needs rewriting.
- **In Testing stage** → Even higher, because debugging and retesting take time.
- **In Production stage** → Very expensive, as it affects customers, business, and reputation.

Example:
If a banking system has wrong interest calculation logic:

- Found in requirement → just fix the formula in document.
- Found in production → customers lose money, refunds needed, legal issues may arise.

So, **early defect detection saves time and money**.

Q3. Describe the types of defects (e.g., functional, performance, usability) and explain why maintaining a defect repository is essential for improving test design.

Answer:
Types of defects:

1. **Functional defects** → Wrong or missing functionality. (Example: search bar not showing results).
2. **Performance defects** → Slow response, high memory usage. (Example: website takes 10 seconds to load).
3. **Usability defects** → Poor user experience. (Example: confusing navigation).
4. **Compatibility defects** → Works on one browser but fails on another.
5. **Security defects** → Weakness that allows hacking.

Defect repository importance:

- Stores details of past defects.
- Helps testers avoid repeating the same mistakes.
- Improves test cases by learning from old issues.
- Useful for tracking defect patterns (e.g., most bugs happen in login module).

So, it acts like a **knowledge base** for better testing.

Q4. Explain the essential principles of effective software testing, such as early testing, defect clustering, and the pesticide paradox. Discuss how these principles contribute to improving the software development process.

Answer:
Principles of testing:

1. **Early testing** → Start testing as soon as requirements and design are ready. This reduces cost and catches issues early.
2. **Defect clustering** → Most defects are usually found in a few critical modules. Knowing this helps testers focus more on high-risk areas.
3. **Pesticide paradox** → Running the same test cases repeatedly will stop finding new bugs. Testers need to update and improve test cases regularly.

Contribution:

- Early testing saves time and cost.
- Defect clustering helps focus effort on risky areas.
- Pesticide paradox keeps testing fresh and effective.

Together, they ensure software quality improves with each cycle.

Q5. Elaborate on the roles and responsibilities of testers in the software development lifecycle. Discuss how testers can identify and address defects originating from requirements, design, and coding stages.

Answer:
Roles & responsibilities of testers:

- Understand requirements clearly.
- Design effective test cases.
- Execute tests and report defects.
- Work with developers to fix and retest.
- Ensure software meets quality standards.

How testers handle defects in different stages:

- **Requirements stage** → Testers review requirement documents to check for gaps or ambiguities. (Example: unclear statement "system should be fast").
- **Design stage** → Testers review system design to find issues (e.g., poor database schema).
- **Coding stage** → Testers run functional, performance, and security tests to catch errors in implementation.

This way, testers ensure **defects are caught early** at every stage, improving the final product quality.

Q6. Discuss the importance of collaboration between developers and testers in managing a defect repository. Highlight the tools and practices that facilitate effective defect tracking and reporting, providing examples of their real-world applications.

Answer:

- **Importance of collaboration:** Developers fix defects, testers find and report them. If they don't work together, bugs may be misunderstood, remain unfixed, or get repeated. Collaboration ensures clear communication, faster resolution, and better quality.
- **Tools for defect tracking:**
  - **JIRA** → widely used for reporting, assigning, and tracking defects.
  - **Bugzilla** → open-source tool for defect tracking.
  - **Trello/Asana** → for small teams to manage defects as tasks.

    - **Azure DevOps** → integrates defect tracking with development.
- **Practices:**
    - Use a **defect repository** to log every defect with status, priority, and history.
    - Hold **defect review meetings** where testers and developers discuss fixes.
    - Use **clear defect reports** with steps to reproduce, screenshots, and logs.

Real-world example: In e-commerce apps like **Amazon**, defects in payment or checkout are logged in JIRA, prioritized as "critical," and developers immediately fix them to avoid customer loss.

Q7. Discuss software testing as a systematic and structured engineering discipline. Describe its phased process, including planning, execution, and analysis, and explain its significance in ensuring software quality.

Answer:
Software testing is not random checking — it's a **systematic and structured process** like engineering.

Phased process:

1. **Planning** → Define scope, objectives, resources, test strategy, and timelines.
    - Example: Decide whether to do functional, performance, and security testing.
2. **Execution** → Run the designed test cases, log defects, and validate fixes.
    - Example: Test login with valid/invalid credentials.
3. **Analysis & Reporting** → Analyze results, measure coverage, track defect trends, and report quality.
    - Example: 90% of test cases passed, 5 critical defects remain.

Significance:

- Ensures software meets requirements.
- Reduces risks of failures in production.
- Improves customer trust and satisfaction.

Q8. Analyze the levels of testing maturity and their role in improving the testing process. Discuss the fundamental truths and assumptions in software testing and explain how they align with the essential principles of effective testing.

Answer:
Levels of testing maturity: (like CMMI for testing)

1. **Initial** → Testing is unstructured, ad hoc.
2. **Repeatable** → Basic testing processes are followed.
3. **Defined** → Standardized testing practices across projects.
4. **Managed** → Testing is measured, tracked, and improved.
5. **Optimized** → Continuous process improvement, automation, and innovation.

Fundamental truths in testing:

- Testing shows **presence of defects**, not absence.
- Exhaustive testing is **impossible** (you can't test everything).
- Testing should start **early**.
- Defects are **clustered** in a few modules.
- Same tests won't always find new bugs (**pesticide paradox**).

Alignment with principles:

- Truths confirm why we need early testing, risk-based testing, and updated test cases.
- Maturity levels help organizations move from random testing to structured, high-quality testing.

Q9. Evaluate the importance of the tester's role throughout the software development lifecycle. Discuss how defects from different stages (requirements, design, code, and environmental factors) can impact the quality and cost of the final product, using real-world examples for illustration.

Answer:
Tester's role in SDLC:

- From requirements to production, testers review, design test cases, execute, and ensure software quality.

Defects from different stages:

- **Requirements defects** → Wrong requirements lead to wrong product.
    - Example: If a banking app defines "interest = 1%" instead of "10%," the whole system is wrong.
- **Design defects** → Poor architecture can cause performance issues.
    - Example: Bad database design in a ticket-booking app leads to slow searches.
- **Coding defects** → Syntax, logic, or calculation errors.
    - Example: A bug in payment code causes duplicate charges.
- **Environmental defects** → Issues from OS, hardware, or network.
    - Example: App works on Android but crashes on iOS.

Impact:

- Increases development cost if found late.
- Hurts business reputation (example: Facebook outages due to server issues cost millions).
- Testers help catch these early, reducing risk and saving money.

Q10. Explain the significance of defect classification and maintaining a defect repository. Discuss strategies for fostering collaboration between

developers and testers and evaluate how effective defect tracking and reporting tools contribute to reducing defect recurrence and improving overall software quality.

Answer:
Defect classification:

- Classifying defects helps in prioritization and fixing. Categories include:
  - **Severity** → Critical, Major, Minor.
  - **Type** → Functional, Performance, Security, Usability.
- Helps teams know what to fix first (critical before cosmetic issues).

Defect repository significance:

- Central storage of all defects.
- Helps avoid repeat mistakes.
- Useful for defect trend analysis and improving test cases.

Collaboration strategies:

- Regular developer-tester sync meetings.
- Use common defect-tracking tools (JIRA, Bugzilla).
- Write clear defect reports with steps, screenshots, and logs.

Contribution of tools:

- Tools provide visibility of defect status (Open, Fixed, Retest, Closed).
- Enable faster resolution and accountability.
- Reduce recurrence since developers learn from logged defects.

Result: Better teamwork, fewer repeated bugs, higher software quality.