QA Fundamentals and Manual Testing

I. Test Management and Issue Tracking

****Roles in Testing**** – most common roles in testing are:

- Test Manager
- Test Lead
- Test Analyst
- Test Engineer
- Manual Tester
- Additional roles can be found depending on the organization

1. Test Manager – responsible for overseeing the testing process to make sure its successful execution.
Some responsibilities are :

- Designs strategies and plans
- Sets priorities
- Mentors the team
- Addresses issues and risks

2. Test Lead – part of the management team who coordinates the activities of the team
Some responsibilities are:

- Assists with planning strategies and tests
- Assigns and tracks tasks given to the team
- Actively assisting the test analysts/engineers
- Organizes meetings for updates and issues

3 .Test Analyst – analyzes product requirements and designs test cases/scenarios
Some responsibilities are:

- Working closely with the TL and managers
- Suggests automation opportunities
- Creates and runs tests and scripts
- Use different testing types

4. Test Engineer – responsible for technical part of the testing including automation tests and execution
Some responsibilities are:

- Develops automation scripts
- Configures test environments and data
- Executes automation tests and reviews of the results
- Troubleshoots technical issues found during testing

5. Manual Tester – responsible to execute tests manually
Some responsibilities are:

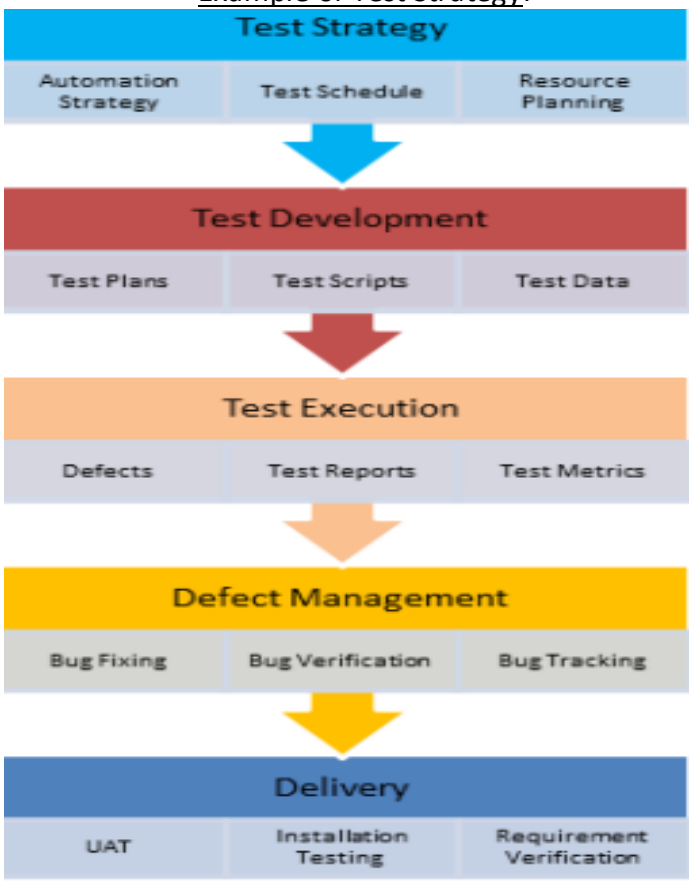
- Manually runs tests and documents the outcomes and defects
- Assists with designing test cases based on specifications
- Logs defects via tracking tools
- Validates bug fixes after resolution

Test management – integral part of the Software Testing Lifecycle (STLC) to ensure testing is performed effectively and efficiently. Includes planning, coordination and control over testing activities within the SDLC (Software Development Lifecycle) and has vital role in delivering high quality software.

Key Components:

- **Test strategy** – sets the overall direction, objectives and scope of the testing. The strategy is developed early in the testing process.

Example of Test Strategy:



- **Test approach** – detailed document that provides specific instructions for executing testing activities for each phase. It defines process of testing, test levels, responsibilities for each team member and types of testing.

- **Entry Criteria** - defines the prerequisites to be achieved before the start of the testing activities. The main task is to check if the testers can perform the needed testing without any major obstacles.

- **Exit Criteria** – defines the conditions to be met before testing can be considered as complete. Indicates that the software is up to the required quality

Recap:

- **Test Strategy sets the overall direction**
- **Test Approach offers specific instructions**
- **Entry Criteria establish predefined conditions that must be met**
- **Exit Criteria outline the conditions that need to be fulfilled**
- **Together these four, form the foundation for effective test management throughout all phases of the testing process**

Test Planning and Estimation

1. Test Planning – process of defining the approach, scope and objectives of testing activities for specific project or release. Involves creation of strategy and plan to guide the testing efforts.

Key Activities:

- Defining testing objectives (goals and expectations)
- Testing Scope and Coverage
- Test Strategy
- Test Schedule and timelines
- Resource planning
- Risk assessment and mitigation
- Test Documentation
- Communication and collaboration

2. Test Design – the process of creating test cases and test scenarios based on the objectives and requirements.

Key Activities:

- Reviewing requirements
- Identifying Test conditions
- Creating test cases and test suits
- Test Data preparation
- Test Case prioritization
- Documentation and maintenance

3. Test Execution – phase where the planned test cases are executed. It involves running the tests, documenting the results and comparing outcomes with the expected results.

Key Activities:

- Test Environment setup
- Test Case execution
- Defects logging
- Defects prioritization
- Defect triage (review meetings)
- Test progress monitoring
- Regression testing
- Documentation and closure

4. Test Monitoring – involves monitoring of the progress, quality and effectiveness of the testing activities throughout the testing lifecycle.

5. Test closure – wrapping up the testing activities and documenting the overall outcomes and lessons learned from the testing effort. Provides detailed summary of the testing outcomes.

II. Testing Techniques Part 1

Testing techniques are systematic approaches for software testing. Their purpose is to address diverse and complex systems, detect different types of defects, optimizing the resources and testing efforts.
There are two main categories:

- Static
- Dynamic

Differences between Static and Dynamic testing techniques

Static testing	Dynamic Testing
Static testing is performed during the early stages of development cycle.	Performed during the later stages of software development life cycle.
Static testing is performed without executing the code.	This type of testing is done by executing the code.
It is performed during the verification stage.	It is executed during the validation stage.
more statement coverage	Less statement coverage.
It requires minimal time for assessment.	It is time consuming as it tests each test case separately.
mainly prevent defects in the software	It find and fix defects.
Involves various assessment methods like walkthrough, inspection, review, and more.	Involves both functional and non-functional testing.
It follows checklist for the process of test execution.	It follows test cases for execution
The percentage of fixing defects is higher in static testing.	The percentage of fixing defects is less in dynamic testing.

Static Testing techniques – improving quality and productivity

1. Reviews – systematic examination of the software or document to identify defects and improve quality. Different type of Reviews include:

- Informal Reviews – casual reviews by collaboration and info share between team members

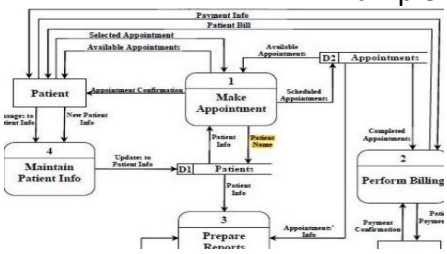
- Walkthroughs – step-by-step exploration of the artifacts and evaluations of test cases, plan, documentation
- Peer Reviews – team members reviewing each other's test cases, scripts, results

- Inspections – formal assessment, ensuring compliance with QA standards, procedures, regulations

2. Static Analysis – looking at a software code, design or documents without executing the programs.
Different type of Analysis include:

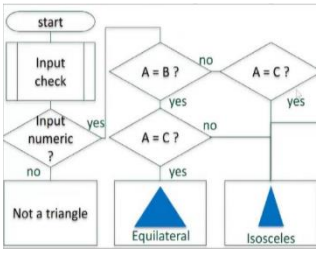
- Data Flow – focuses on the path a data can take throughout a program.

Example:



- Control Flow – examines the order in which the statements or functions go trough a program, execution paths, branches or loops.

Example:



- Peer Reviews – team members reviewing each other's test cases, scripts, results.

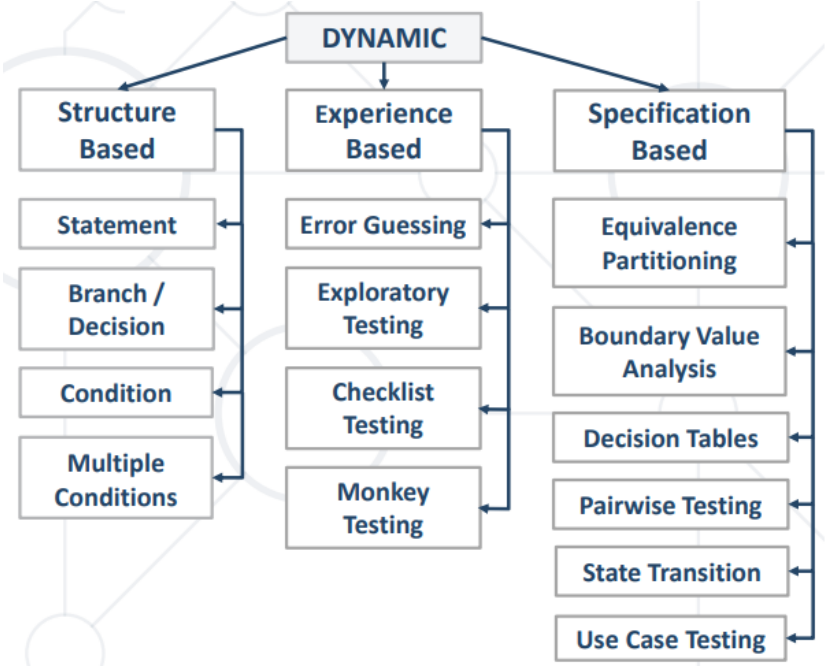
- Inspections – formal assessment, ensuring compliance with QA standards, procedures, regulations.

- Cyclomatic Complexity – measures the program's complexity, determines minimum needed test cases for maximum coverage. More complex code – more testing and potential defects.

Dynamic Testing Techniques – testing throughout execution(testing in action).

These type of techniques are based on three factors:

- Structure
- Experience
- Specification



NOTE:

Structure Based techniques are White-Box

Specification Based techniques are Black-Box

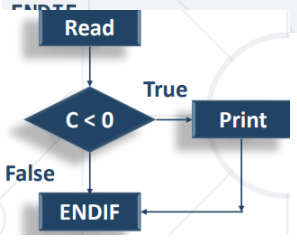
1. **Structure Based techniques**
   - White-Box techniques – test cases are selected based on analysis of the internal structure/components. The aim is to increase test coverage
   - Coverage – defined by the number of items covered in testing divided by total number of items. The aim is to achieve 100% coverage (but 100% does not mean 100% tested).

   - Statement coverage – number of statements covered in testing divided by the total number of statements. The aim is to have 1 test case which can cover all statements.

   - Branch/decision coverage

```
Code sample:

READ A
READ B
IF A > B THEN C = 0
```



2. Experience-based Techniques – relies on the knowledge and experience of the testers. Useful when documentation is incomplete or system is too complex.

- Exploratory testing – software is actively explored with a combination of domain knowledge and creativity. Minimum planning and maximum test execution flexibility.
- Error guessing – based on tester intuition to identify potential software issues. Testers simulate real-world scenarios to uncover unexpected defects.
- Checklist-based testing - predefined checklist of common issues based on experience and similar systems.
- Random/Monkey testing – varied or random inputs to the system randomly to uncover unexpected or hard to find bugs. Balance between randomness and creativity.

Recap:

There are static and dynamic testing techniques:

- **Static techniques include reviews which increase quality and productivity**
- **Dynamic techniques are based on three factors – structure, specification, experience**
- **Structure-based techniques are called white box techniques**
- **Experience-based techniques**
- **Specification-based techniques are called black box techniques**

III. Testing Techniques Part 2

Specification based techniques

Black Box testing – tests software functionalities without the knowledge of the code.

- Does not require understanding of the code implementation
- Suitabe for testing of OS, websites, databases
- Also known as Behavioral Testing

1. Equivalence Partitioning (EP) – divides input data into valid or invalid parts(partitions)

- Each partition is tested at least once
- Selects representative values of each partition
- Reduces number of test cases
- Can be used at any test level

Example: If a percentage for a product can only be between 50-90 as Input then:

Equivalence Partitioning		
Invalid	Valid	Invalid
<=50	50-90	>=90

2. Boundary Value Analysis (BVA) – testing the "edges" of equivalence classes.

- Boundaries are on the edge where the system changes
- Primary focus is on the exact boundaries of the valid ranges
  E.g valid input is from 10 to 99 then lower boundary value is 10 and upper is 99.

- Values close to those range are also important in BVA. E.g 9.99 and 98.99 (from 10 and 99 as highest).
- Bugs are often found during testing of the edges of values

3. Decision Table Testing

- A table that connects combinations of conditions and actions (effects).
- Multiple small decisions table is better than having several big ones (easier to manage)

Example:

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
New customer (15%)	T	T	T	T	F
Loyalty card (10%)	T	T	F	F	T
Coupon (20%)	T	F	T	F	T
Actions					
Discount (%)	invalid	invalid	20	15	30

4. Pairwise Testing – desting method that combines parameters in pairs which reduces the overall number of tests but maintain good test coverage.

Pairwise Testing Tool (ctrl+click to view)

Test Case	A	B	C
Test Case 1	1	1	1
Test Case 2	1	2	2
Test Case 3	1	3	3
Test Case 4	2	1	3
Test Case 5	2	2	1
Test Case 6	2	3	2
Test Case 7	3	1	2
Test Case 8	3	2	3
Test Case 9	3	3	1

For example, the pair (A=1, B=2) is covered in test case 2, and the pair (A=2, C=3) is covered in test case 4, and so on.

5. Use Case testing - a test which is described as a real-life scenario and is used to test if the system can handle a transaction from start to finish.

- Consists of who (actor), what (interaction) and purpose (goal) without looking at how the system handles process internally
- Requires pre-conditions in order for the use case to start
- Benefitial in identifying integration defects that could occur in real world scenarios

6. Choosing a test techniques – each technique is good for one situation but bad for another.

- Structure based are good for finding erros in the code
- Specification based are good for finding missing parts from the code
- Experience based are good for finind missing parts and errors in the code

Recap:

Black box techniques

- **Equivalence Partitioning (EP)**
- **Boundary Value Analysis (BVA)**
- **Decision Table Testing**
- **Pairwise Testing**
- **State Transition Testing**
- **Use-Case Testing**

IV. WebAPI, HTTP Protocol, REST

API - Application Programming Interface

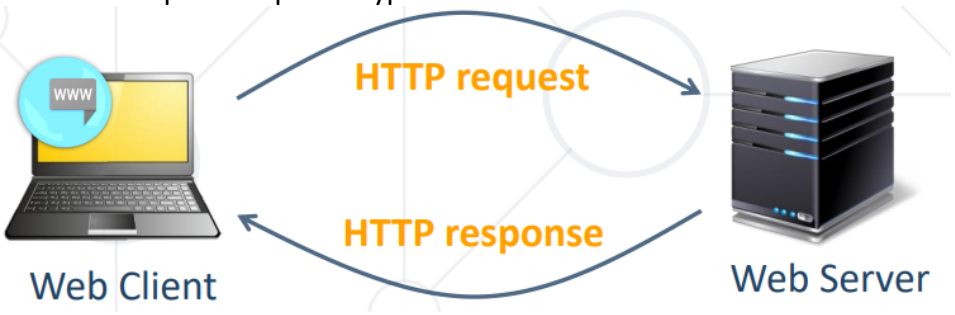
- Programming interface designed for communication between system components
- Set of functions and specifications which programs and components follow in order to talk to each other

1. Web Service

- Services that implement communication between softwares or components over the network
- Standard protocols are HTTPs, JSON and XML
- Web services are hosted on a web Server

2. HTTP (Hyper text transfer protocol)

- Text based client><Server protocol for the Internet
- Used to transfer web resources (html files, images, styles,etc)
- Based on request-response type of communication



3. XML (extensible markup language)

- Markup language that is used to store and transfer data
- Has a Tree-like structure (Root,Tag,Child elements and attributes)

Example:

4. JSON (JavaScript Object Notation)

- Light data format based on JS syntax
- Simple text, key-value based
- Easy for people to read, write and easy for machines to parse
- Used to transmit data between apps and server (alternative to XML)

5. REST/RESTful API (Representational State Transfer)

- Architecture for client-server communication over HTTP
- Provides access to server-side resourses via HTTP
- Resourses have URI address (can be created,retrieved,updated, deleted)

V. Relational Databases

Database - collection of data, organized to be easily accessed, updated and managed. Modern DBs are managed by Database management systems (DBMS)

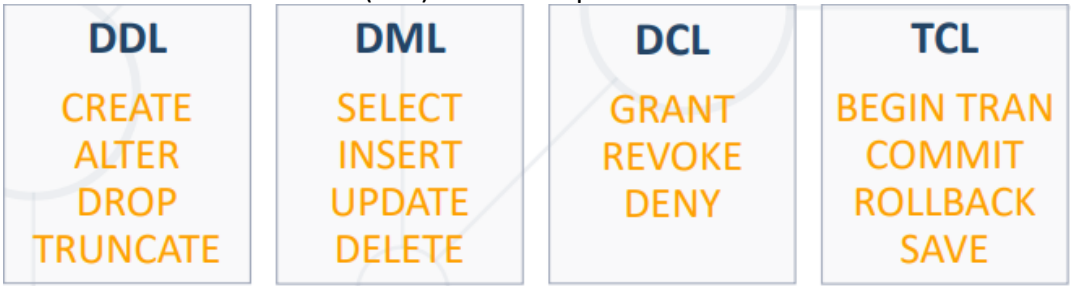
1. SQL Databases

- Organize data in form of tables
- Tables have strict structure (columns, data types, relation to other tables)
- Data is stored in one or more tables

2. SQL – structured query lanaguage designed to manipulate data

Logically is divided by 4 sections:

- Data Definition (DDL) – describe structure of the data
- Data Manipulation (DML) – store and retrieve data
- Data Control (DCL) – define who can access data
- Transaction Control (TCL) – bundle operations and allow rollback



Must have knowledge for QAs

- Recognize **Various Types** of Databases
- Connect Using **Different SQL Clients**
- Comprehend Database **Relationships, Keys, and Indexes**
- Write **Simple and Complex** SQL Queries
- Perform **Data Validation** and Testing Techniques
- Test **Data Modifications** and Transactions
- **Explore** Database Schema
- **Interpret** Complex Queries