Testing Techniques

1. <u>Boundary Value Analysis (BVA):</u>

- Focuses on testing software at the edges or boundaries of input ranges, where errors are more likely to occur.
- Identifies valid and invalid equivalence classes based on input specifications.

Example:
Valid Boundary Values:

- Online form that accepts age between 18 and 65.

  Test cases:

  Age = 18 (minimum valid)
  Age = 65 (maximum valid)
- Temperature sensor with range -40°C to 150°C.

  Test cases:

  Temperature = -40°C (minimum valid)
  Temperature = 150°C (maximum valid)

Invalid Boundary Values:

- Same form as above.

  Test cases:

  Age = 17 (just below minimum valid)
  Age = 66 (just above maximum valid)
- Same temperature sensor.

  Test cases:

  Temperature = -41°C (invalid, below minimum)
  Temperature = 151°C (invalid, above maximum)

Advantages:

- Simple and effective technique for identifying potential errors.
- Easy to understand and implement.

Disadvantages:

- May not catch all errors, especially for complex logic.
- Does not consider all possible combinations of input values.

2. <u>Decision Table Testing</u>

Decision Table Testing is a systematic approach to identify and document the different conditions and actions (or decisions) for a system. It is particularly useful for testing complex business logic where different combinations of inputs lead to different outputs.
Example: Consider a simple loan approval system that depends on two conditions: income level (high/low) and credit score (good/bad).

Income	Credit Score	Action
High	Good	Approve Loan
High	Bad	Review Manually
Low	Good	Review Manually
Low	Bad	Reject Loan

Test cases derived from this decision table ensure that the system behaves correctly for each combination of income level and credit score.

Advantages:

- Encourages systematic test design by considering all input combinations.
- Improves test case traceability.

Disadvantages:

- Can become complex for applications with many input conditions.
- Maintaining tables for large datasets can be challenging.

3. Use Case Testing (UCT):

- Focuses on testing from the user's perspective, exercising the system's functionality as described in use cases.
- Ensures the system meets user requirements and can perform intended tasks successfully.

Example:

Use case: Online banking login

Scenario: A valid user logs in with correct credentials.

Test cases:

Valid username and password entered.
Case-sensitive username test (if applicable).
Password history check (if enforced).

Scenario: An invalid username or password is entered.

Test cases:

Invalid username entered.
Incorrect password entered.
Locked account scenario (if applicable).

Advantages:

- Improves test case relevance by focusing on user needs.
- Identifies missing or incorrect requirements.

Disadvantages:

- Can be time-consuming to create and maintain use cases, especially for large systems.
- May not cover all edge cases or error conditions.

4. Equivalence Partitioning (LCSAJ) Testing (Not commonly referred to as LCSAJ):

- Divides the input domain into equivalence partitions (valid and invalid), where all inputs within a partition are expected to behave similarly.
- Tests representative values from each partition to ensure the system behaves as expected.

Example:

Form field that accepts a phone number in the format +91 XXXXX XXXXX.

Valid Equivalence Partition: Phone number with correct format (EX.+91 12345 12345).

Test cases:

- Valid phone number with various digits.
- Phone number with extensions (if allowed).

Invalid Equivalence Partition: Phone number with incorrect format (EX.123456)