SOFTWARE ENGINEERING

UNIT - 4

TOPIC – 1

A STRATEGIC APPROACH TO SOFTWARE TESTING

INTRODUCTION TO TESTING

Software testing is a key step in creating software, ensuring it works correctly and meets user expectations. Think of it as quality control for a product before it reaches customers.

What is Software Testing?

Software testing checks:

1. **Does it work as expected?** Ensures the software produces the correct results.
2. **Does it meet user needs?** Verifies that the software fulfills the requirements it was built for.
3. **Is it free of bugs?** Identifies and fixes errors before users encounter them.

Why is Testing Important?

- Prevents failures in important tasks (e.g., bank transfers or online shopping).
- Ensures the software runs smoothly under all conditions.
- Protects sensitive data from security threats.

Since errors can disrupt the user experience and lead to significant consequences, testing is necessary to catch these issues before the software is deployed. However, finding these errors is only one part of the solution. Once identified, errors must be fixed through debugging.

TESTING VS DEBUGGING

- **Testing**: Finds problems by examining how the software behaves under different conditions.

  - Example: Testing a login page to check if it accepts correct credentials and rejects incorrect ones.
- **Debugging**: Fixes the problems found during testing.
  - Example: Correcting the login page so it stops crashing when wrong credentials are entered.

Aspect	Testing	Debugging
Who Does It?	Testers or Quality Analysts	Developers
Focus	Finds problems in the software	Fixes what caused the problems
Question Asked	Does it work?	Why isn't it working, and how to fix it?

Testing sets the stage for debugging by identifying where the software fails. To structure the testing process efficiently, we use specific software testing strategies.

A STRATEGIC APPROACH TO SOFTWARE TESTING

A **testing strategy** is a systematic plan to ensure all parts of the software are tested effectively. It provides a step-by-step approach to check individual components, their integration, and the entire system.

Goals of Testing Strategies

1. Identify and fix errors early to save time and resources.
2. Ensure the software performs as expected under different conditions.
3. Verify that the software meets both functional and user requirements.

Types of Testing Strategies

1. **Unit Testing**: The first step of testing, focusing on individual components of the software.
   - Example: Testing a login function with valid and invalid inputs to ensure it works correctly.
2. **Integration Testing**: After unit testing, integration testing ensures that different components work together.

   - Example: Checking if the login system successfully connects to the user dashboard.
3. **System Testing**: Tests the entire software as a single system to ensure all parts work together seamlessly.
   - Example: Testing a food delivery app to ensure login, menu selection, and payment features work correctly.
4. **Validation Testing**: Focuses on ensuring the software meets the end user's needs.
   - Example: Allowing real users to test a beta version of the app to provide feedback.

With these strategies, testing progresses logically, starting with small units and expanding to the complete system. Let's now explore these types of testing in more detail.

Black-Box and White-Box Testing

Black-Box Testing

Black-Box Testing focuses on testing the software's functionality without looking at the internal code. It examines inputs and outputs to check if the software behaves as expected.

- **Key Features**:
  - Tester doesn't need programming knowledge.
  - Based on user requirements and specifications.
- **Examples**:
  - Testing a calculator app by entering numbers and verifying results.
  - Checking if an e-commerce website correctly adds items to the cart.

White-Box Testing

White-Box Testing tests the internal structure or logic of the code. It ensures that all possible paths in the code work as intended.

- **Key Features**:
  - Requires programming knowledge.
  - Focuses on code coverage, such as loops, conditions, and paths.
- **Examples**:

- Verifying if a login function correctly checks both username and password.
- Checking the logic of a sorting algorithm to confirm it handles all input cases.

Differences Between Black-Box and White-Box Testing

Aspect	Black-Box Testing	White-Box Testing
Focus	Software functionality	Internal code structure
Knowledge Required	None (only specifications are needed)	Requires programming and code understanding
Approach	Input-output based	Logic and path-based
Example	Testing login functionality from the user's perspective	Checking how the login function processes inputs internally

VERIFICATION AND VALIDATION

Verification: "Are we building the product correctly?" It ensures the software matches its design and specifications.

- **Example**: Checking if a login screen has all the fields and buttons specified in the design.

Validation: "Are we building the right product?" It ensures the software meets user needs.

- **Example**: Testing if users can successfully log in with the "Forgot Password" feature.

Aspect	Verification	Validation
Focus	Checks if software matches the plan	Checks if software meets user needs
Example	Code reviews	Functional and user testing

Testing ensures software is error-free, but when errors are found, the next step is to debug and resolve them.