5. Software testing

Q1) Software testing process

=> (1) Purpose of testing.
(i) Software is tested to find errors made during design and development.
(ii) Testing takes up a large portion of project resources, so it is essential to have a plan or strategy developed by project manager

(2) Cost of testing
(i) Testing can make up about 40% of the total project cost
(ii) A testing strategy outlines the steps for planning tests, creating tests cases, running tests and collecting tests.

(3) Validation
(i) Validation makes sure the software meets the customer requirements.

(4) Testing strategies for software.
(i) Unit testing
(a) It focuses on individual parts of code.
(b) Tests each part thoroughly to catch as many errors as possible
(ii) Integration testing
(a) Focuses on combining software components and checking how they work together.
(b) Tests whether different parts of the software communicate properly.
(i) its types: (1) Top down approach
(2) Bottom up approach
(3) Sandwich approach

(iii) Validation Testing

(a) It checks if software meets all functional behavorial and performance expectations of the customer

(b) its types : (1) Alpha testing
(2) Beta testing

(iv) System testing

(a) In this testing, the entire system is tested as a whole.

(b) its types : (1) Performance testing
(2) Recovering testing
(3) Security testing
(4) Stress testing.

PAGE No.
DATE / /

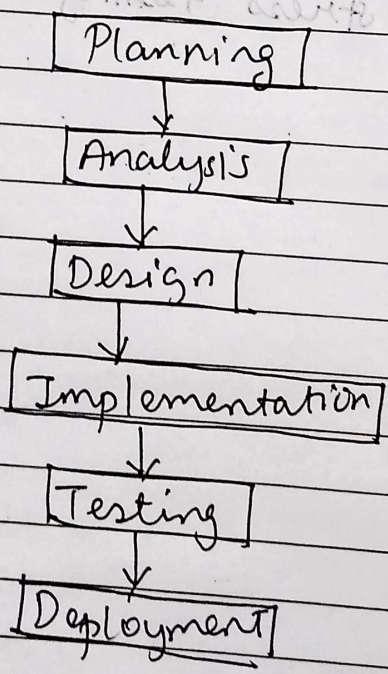
Q3) Explain software Re-engineering process

⇒ (1) Re-engineering is the process of updating and improving the existing software

(2) It involves studying current software to identify areas that can be made better in terms of performance, quality and maintenance.

(3) It means making the software work on newer platforms, adding new features or improving the overall design.

(4) Process of software re-engineering



(i) Planning

(a) The first step is to plan the re-engineering process which involves identifying why it is needed and what the goals are.

(ii) Analysis

(a) The next step is to analyze the existing system, including the coding, documentation.

(b) It helps to examine the existing software to find the strengths and areas of development.

(iii) Design

(a) Based on the analysis, the next step is to design the new or updated software system.

(b) It is used to plan the changes needed to meet the goal.

(iv) Implementation

(a) In this stage, the code is updated, the features are added and the documentation is adjusted.

(v) Testing :

(a) This stage is used to check that the changes meet the required standards.

(vi) Deployment

(a) In deployment stage, the improved software is released to the users.

PAGE No.
DATE / /

Q4) Explain software reverse engineering

=> (1) Reverse engineering uncovers the design and functionality of a product or system by working backwards from its final form.

(2) Purpose of reverse engineering:

(i) Helps in understanding how the program works.

(ii) Used to check security vulnerabilities.

(iii) Enables adding features in existing programs

(iv) Helps in creating similar products at low cost.

(3) Levels of reverse engineering:

(i) Abstraction level

(a) The highest level where the information is extracted from the source code.

(b) It helps developers understand the program's overall structure and functionality.

(ii) Completeness

(a) It checks if the information gathered at information level is detailed and complete.

(b) As the abstraction level goes up, completeness usually decreases.

(iii) Directionalities

(a) It describes how information flows in reverse engineering process.

(b) There are two directionalities type:

(1) One-way: Information is simply extracted from the code.

(2) Two-way: The extracted information is used in a tool that recognizes and improves old programs.

PAGE No.
DATE / /

(4) Process overview

(i) Start with raw source code and refine it to get more cleaner, more understandable code.

(ii) Breakdown the code to understand:

(a) How the system functions

(b) connection between different components.

(c) Types of databases used.

(iii) With this knowledge, you get the initial specifications of the system.

(iv) Add new features to simplify and refine the system further.

PAGE No.
DATE / /

Q5) List out different software testing strategies

=> 1. White Box Testing

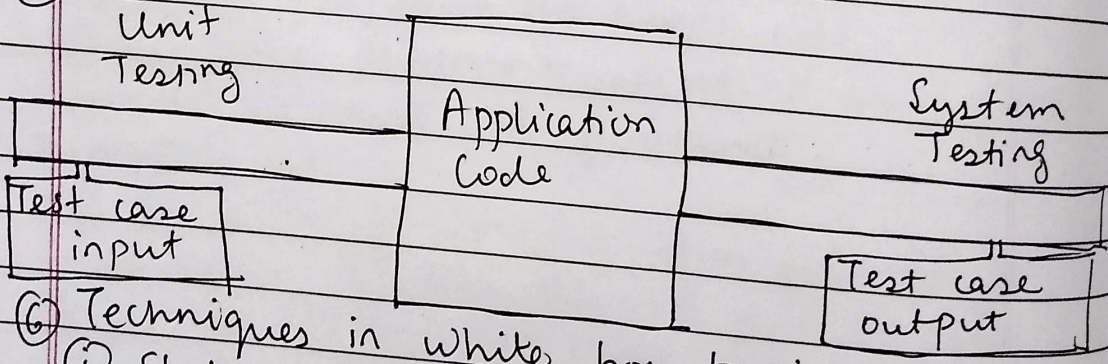
(1) White Box testing is used to test the internal structure of the software by examining the code.

(2) It is also called as Glass Testing

(3) The testers in the whitebox testing should know the internal structure or the programming of software. It is done by the software developer

(4) Whitebox tools
   (i) Py Unit
   (ii) CS Unit
   (iii) Cpp Unit
   (iv) Nmap

(5)

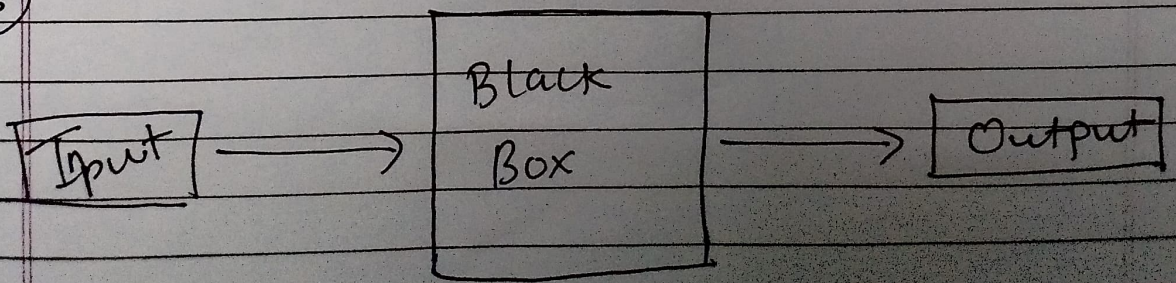


(6) Techniques in white box testing:
   (i) Statement coverage
   (a) Tests every line of code to catch any faulty line of code.
   (ii) Branch coverage
   (a) Ensures all decision points are tested at least once.
   (iii) Conditional coverage
   (a) Tests individual conditions within the statements.

(iv) Basis path testing :
(a) Uses control flow graphs to find and test independent paths.
(v) Loop testing
(a) Test loops for proper functioning, covering simple, nested loops.
(7) Types of whitebox testing
(i) Unit testing
(ii) Integration testing
(iii) Regression testing

Black Box testing

(1) Black-box testing is a software testing approach where the tester focuses on the functionality of the software without knowing its internal code.
(2) The aim is to ensure the software behaves as expected according to its requirements.
(3) The testers in the blackbox testing should not necessarily have the knowledge about the internal structure of the software.
(4) Blackbox testing tools
(i) Postman
(ii) Selenium
(iii) Appium
(iv) TestComplete.
(5)



(6) Techniques in blackbox testing

(i) Equivalence partitioning

(a) The technique divides inputs into groups partitions where each group should behave similarly

(ii) Boundary value analysis

(a) Tests the boundaries of input ranges, as errors often occurs at the extreme ends.

(b) Eg: if the range is 1-100, testing values like 0, 1, 100, 101 helps catch the issues.

(iii) Cause-effect graphing

(a) Creates a graph, linking inputs to expected outputs to design test cases.

(iv) Requirement based testing

(a) Ensures each software function meets its specific requirements.

(v) Compatibility testing

(a) Checks if the software is compatible or works across different environments.

(7) Types of blackbox testing

(i) Functional testing

(ii) Non-functional testing

(iii) Regression testing

Q. Design the test cases for medical management application

=>

1. Patient Registration

- **Test Case 1**: Verify that a new patient can register with valid details (e.g., name, age, contact number).
    - **Expected Result**: Patient is successfully registered, and a unique patient ID is generated.
- **Test Case 2**: Verify the system does not allow registration with missing mandatory fields (e.g., missing name or contact number).
    - **Expected Result**: An error message is displayed, prompting the user to fill in the required fields.

2. Appointment Scheduling

- **Test Case 1**: Verify that a patient can schedule an appointment with an available doctor.
    - **Expected Result**: The appointment is successfully added to both the patient's and doctor's schedules, with confirmation sent to the patient.
- **Test Case 2**: Verify that the system prevents scheduling an appointment if the doctor's schedule is already full for the selected date and time.
    - **Expected Result**: An error message is displayed, indicating the doctor's unavailability.

3. Doctor Management

- **Test Case 1**: Verify that an admin can add a new doctor with all required details (name, specialization, contact).
    - **Expected Result**: Doctor is added to the system and appears on the doctors' list.
- **Test Case 2**: Verify that the system prevents adding a doctor if the record already exists (duplicate name, contact, and specialization).
    - **Expected Result**: An error message is displayed, stating that the doctor record is already in the system.

4. Prescription and Medication Management

- **Test Case 1**: Verify that doctors can issue a prescription to a patient with medication details and instructions.
    - **Expected Result**: Prescription details are saved in the patient's medical records and are accessible to both the doctor and patient.
- **Test Case 2**: Verify that patients can view their past prescriptions.
    - **Expected Result**: The patient can see all previous prescriptions, listed chronologically, with all details intact.

5. Billing and Payments

- **Test Case 1**: Verify that billing is accurately calculated based on services provided, such as consultation fees and medication.
    - **Expected Result**: A detailed bill is generated, reflecting all services and charges accurately.
- **Test Case 2**: Verify that patients can make payments using different payment methods (e.g., credit card, debit card, insurance).
    - **Expected Result**: Payment is processed, and a receipt is generated, confirming the transaction.

6. Data Security and Compliance

- **Test Case 1**: Verify that patient data is encrypted when stored in the database and during data transmission.
    - **Expected Result**: Data inspection shows encryption is applied both at rest and in transit, protecting patient information.
- **Test Case 2**: Verify that only authorized users can access sensitive patient information.
    - **Expected Result**: Unauthorized access attempts are blocked, and the system logs these attempts for security audits.

Q. Difference between Alpha and Beta Testing

=>

Parameters	Alpha Testing	Beta Testing
Technique Used	Alpha testing uses both white box and black box testing.	Beta testing commonly uses black-box testing.
Performed by	Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Performed at	Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user of the product.
Reliability and Security	Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
Ensures	Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
Requirement	Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Execution	Alpha testing may require a long execution cycle.	Beta testing requires only a few weeks of execution.
Issues	Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product.
Test Cycles	Multiple test cycles are organized in alpha testing.	Only one or two test cycles are there in beta testing.

Q. Difference between Blackbox and Whitebox testing

=>

Aspect	Black Box Testing	White Box Testing
Knowledge of Internal Code	Not required	Required
Other Names	Functional testing, data-driven testing, closed box testing	Structural testing, clear box testing, code-based testing, transparent testing
Approach	Trial and error, based on external functionality	Verification of internal coding, system boundaries, and data domains
Test Case Input Size	Largest	Smaller compared to Black Box
Finding Hidden Errors	Difficult	Easier due to internal code access
Algorithm Testing	Not suitable	Well-suited and recommended
Time Consumption	Depends on functional specifications	High due to complex code analysis