| Lesson Plan No. 1 | Course Name: Software Testing & automation<br>Topic: Overview of software testing principles and concepts | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the fundamental principles of software testing.<br>b. Differentiate between verification and validation.<br>c. Identify and describe different levels and types of software testing. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- Why do we test software?"<br>- Briefly explain the importance of software testing in the software development lifecycle.<br>- **Development** (30 minutes)<br>- Present the seven fundamental testing principles:<br>- Testing shows presence of defects, not absence.<br>- Exhaustive testing is impossible.<br>- Early testing saves time and money.<br>- Defects cluster together (Pesticide Paradox).<br>- Testing is context dependent.<br>- Absence of errors is a fallacy.<br>- Explain each principle with examples and real-life analogies.<br><br>Test Levels and Types<br>- Unit testing<br>- Integration testing<br>- System testing<br>- Acceptance testing<br>- Explain each level with examples.<br>-<br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. Suggested video lecture<br>https://www.youtube.com/watch?v=NC1aqG4tWl4<br><br>Reference website:<br>https://www.geeksforgeeks.org/software-engineering-seven- |

| | |
|---|---|
| | principles-of-software-testing/<br><br>Suggested Reading books:<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>3. Homework<br>- Describe the "pesticide paradox" in software testing. How can testers mitigate its effects?<br>- What does it mean that "testing is context dependent"? Provide examples of how testing approaches might vary based on context.<br>- Explain the concept of "exhaustive testing is impossible." How does this influence test planning?<br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on software testing principles<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 2 | Course Name: Software Testing & automation<br>Topic: Testing process and its phases | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the overall software testing process.<br>b. Identify and describe the different phases of the testing process.<br>c. Explain the activities performed in each phase.<br>d. Understand the importance of each phase in ensuring software quality.<br>e. Relate the testing phases to the software development lifecycle (SDLC). |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- What are some steps involved in testing a software application?"<br>- Briefly review the importance of software testing.<br>- Introduce the concept of a structured testing process.<br><br>- **Development** (30 minutes)<br>- **Overview of the Testing Process:**<br>- Present a high-level overview of the testing process.<br>- Explain that the testing process is iterative and integrated with the SDLC.<br>- Highlight the importance of planning, execution, and reporting.<br>- Show a simple diagram of the testing process flow.<br>- **Phases of the Testing Process :**<br>A. **Test Planning**<br><br>Define test planning and its importance.<br>**Explain the activities performed in this phase:**<br>• Defining the scope and objectives.<br>• Identifying risks and mitigation strategies.<br>• Developing the test strategy and plan.<br>• Estimating resources and timelines.<br>• Defining entry and exit criteria.<br><br>Provide examples of test plan components.<br>B. **Test Design**<br><br>Define test design and its importance.<br>**Explain the activities performed in this phase:**<br>• Creating test scenarios and test cases. |

| | |
|---|---|
| | • Defining test data requirements.<br>• Designing the test environment.<br>• Selecting test techniques.<br><br>Discuss the importance of traceability between requirements and test cases.<br>C. **Test Execution**<br><br>Define test execution and its importance.<br>**Explain the activities performed in this phase:**<br>• Setting up the test environment.<br>• Executing test cases.<br>• Logging defects and tracking their status.<br>• Re-testing fixed defects.<br><br>Discuss the importance of test execution logs and reports.<br>D. **Test Closure**<br><br>Define test closure and its importance.<br>**Explain the activities performed in this phase:**<br>• Evaluating test results against exit criteria.<br>• Creating a test summary report.<br>• Documenting lessons learned.<br>• Obtaining sign-off from stakeholders.<br><br>Discuss the importance of knowledge transfer and process improvement.<br><br>IV. **Relationship to the SDLC**<br><br>• Explain how the testing phases align with different stages of the SDLC.<br>• Show a diagram illustrating the integration of testing with the SDLC (e.g., V-model, Agile).<br>• Discuss the benefits of early testing and continuous testing.<br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. **Suggested video lecture** |

| | |
|---|---|
| | https://www.youtube.com/watch?v=HylDB3bN6hQ&t=24s<br><br>**Reference website:**<br>https://testsigma.com/blog/phases-of-testing/<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>3. Homework<br>- What are the primary phases of the software testing process, and what is the general purpose of each phase?<br>- In the "test planning" phase, what key activities are performed, and why is this phase crucial?<br>- What is the significance of "test design," and what are the main deliverables of this phase?<br>- How can test metrics be used to evaluate the effectiveness of the testing process? Provide examples.<br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Testing process and its phases<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 3 | Course Name: Software Testing & automation<br>Topic: Software Development Life Cycle | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Define the Software Development Life Cycle (SDLC).<br>b. Identify and describe the key phases of the SDLC.<br>c. Understand the importance of each phase in the SDLC |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- Begin with a discussion about software that students use daily.<br>- "How do you think software is created?"<br>- Introduce the concept of the Software Development Life Cycle (SDLC) as a structured process for developing software.<br>- Explain the importance of having a systematic approach to software development.<br>- **Development** (30 minutes)<br>- Discuss the benefits of using SDLC:<br>- Improved project management<br>- Reduced development costs<br>- Increased software quality<br>- Better communication and collaboration<br>- Improved risk management.<br>**Briefly describe the phases common to many models:**<br>- Planning/Requirement Gathering<br>- Design<br>- Implementation/Coding<br>- Testing<br>- Deployment<br>- Maintenance<br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. **Suggested video lecture**<br>https://www.youtube.com/watch?v=OT2O7uNldQk<br><br>**Reference website:** |

| | |
|---|---|
| | https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>3. Homework<br>- What is the Software Development Life Cycle (SDLC)?<br>- What are the typical phases of the SDLC?<br>- Why is the SDLC important in software development?<br>- What is the purpose of the requirements gathering phase?<br>- What are the different types of software testing performed during the SDLC?<br>- Explain the importance of the maintenance phase of the SDLC.<br><br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Software Development Life Cycle<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 4 | Course Name: Software Testing & automation<br>Topic: Types of software defects | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Identify and describe different types of software defects.<br>b. Classify software defects based on various criteria. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- Engage students with a short discussion: "Why do software defects occur? “How do you think software is created?"<br>- Briefly introduce the concept of software defects and their impact on software quality.<br>- Explain the importance of defect classification for effective defect management, analysis, and prevention.<br><br>- **Development** (30 minutes)<br>- **Types of Software Defects**<br><br>- **Functional Defects:**<br>- Definition: Defects related to incorrect or missing functionality.<br>- Examples: Incorrect calculations, missing features, unexpected behavior.<br>- Discussion: How do functional defects impact user experience?<br><br>- **Logical Defects:**<br>- Definition: Errors in the program's logic or algorithm.<br>- Examples: Incorrect conditional statements, infinite loops, incorrect data flow.<br>- Discussion: How can logical defects be difficult to detect?<br><br>- **Performance Defects:**<br>- Definition: Issues related to the software's speed, responsiveness, and resource usage.<br>- Examples: Slow loading times, memory leaks, high CPU usage.<br>- Discussion: Why is performance testing crucial?<br>- **Usability Defects:**<br>- Definition: Problems with the user interface and user experience.<br>- Examples: Confusing navigation, unclear error messages, |

| | |
|---|---|
| | inconsistent design.<br>- Discussion: How does usability affect user satisfaction?<br><br>- **Security Defects:**<br>- Definition: Vulnerabilities that can be exploited by malicious actors.<br>- Examples: SQL injection, cross-site scripting, buffer overflows.<br>- Discussion: The importance of security testing in today's world.<br><br>- **Interface Defects:**<br>- Definition: Problems related to the communication between different software components or systems.<br>- Examples: Incorrect data exchange, incompatible APIs, communication failures.<br>- Discussion: The increasing importance of API testing.<br><br>- **Documentation Defects:**<br>- Definition: Errors or omissions in the software's documentation.<br>- Examples: Incorrect user manuals, outdated API documentation, missing comments.<br>- Discussion: The influence of good documentation.<br><br>- **Syntax Defects:**<br>- Definition: Errors in the programming language's syntax.<br>- Examples: Misspelled keywords, missing semicolons, incorrect data types.<br>- Discussion: How are syntax defects usually found?<br><br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. **Suggested video lecture**<br>https://www.youtube.com/watch?v=f6OYpQO3BWY<br><br>**Reference website:**<br>https://www.geeksforgeeks.org/what-is-defect-in-software-testing/ |

| | |
|---|---|
| | **Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- Define a "functional defect" and provide two examples.<br>- Explain the difference between a "logical defect" and a "syntax defect."<br>- What are "performance defects," and why are they important to address in software development?<br>- Describe how "usability defects" can impact the end-user experience.<br>- What types of vulnerabilities are considered "security defects," and why are they a critical concern?<br>- Explain what "interface defects" are and give an example of where they might occur.<br>- How do "compatibility defects" differ from other defect types, and what factors contribute to them?<br><br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Types of software defects<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 5 | Course Name: Software Testing & automation<br>Topic: Classification of software defects | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. To identify and explain various classifications of software defects.<br>b. To understand the benefits of classifying software defects.<br>c. To apply defect classification to real-world scenarios. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- Engage students with a short discussion: "Why do software defects occur? “How do you think software is created?"<br>- Briefly introduce the concept of software defects and their impact on software quality.<br>- Explain the importance of defect classification for effective defect management, analysis, and prevention.<br><br>- **Development** (30 minutes)<br>- **Discuss the importance of identifying and classifying defects for:**<br>- Improving software quality.<br>- Reducing development costs.<br>- Enhancing user satisfaction.<br>- Preventing future defects.<br>- Enabling better metrics and root cause analysis.<br>- **Introduction to Classification:**<br>Briefly introduce the concept of classifying defects to organize and analyze them.<br><br>- **Types of Defect Classifications**<br><br>- **Classification by Severity:**<br>Explain the levels of severity (e.g., Critical, Major, Minor, Trivial).<br>Provide examples of each severity level.<br>Discuss how severity impacts prioritization.<br>- **Classification by Priority:**<br>Explain the levels of priority (e.g., Urgent, High, Medium, Low).<br>Explain the difference between severity and priority.<br>Discuss how business needs and release schedules influence priority.<br>- **Classification by Functional Area:** |

| | |
|---|---|
| | Explain how defects can be categorized based on the specific module or functionality affected (e.g., UI, database, security, business logic).<br>Provide examples related to a hypothetical or real software application.<br>Discuss the importance of functional area classification for assigning defects to the correct developers.<br>- **Classification by Root Cause:**<br>Explain how defects can be categorized based on their origin (e.g., requirements, design, coding, testing, documentation).<br>Discuss the importance of root cause analysis for preventing recurring defects.<br>**Give examples such as:**<br>• **Requirements: Ambiguous or incomplete requirements.**<br>• **Design: Flawed architectural decisions.**<br>• **Coding: Syntax errors, logic errors.**<br>• **Testing: Inadequate test coverage.**<br><br>- **Classification by Phase Detected:**<br>Explain how defects can be classified by the software development life cycle (SDLC) phase in which they were discovered (e.g., requirements phase, design phase, coding phase, testing phase, post-release).<br>Discuss the cost implications of finding defects later in the SDLC.<br>- **Classification by Type:**<br>Explain how defects can be classified by their type, for example:<br>• Logic errors<br>• Syntax errors<br>• Performance defects<br>• Security defects<br>• Usability defects<br>• Interface defects<br>- **Benefits of Defect Classification**<br>• Improved Communication<br>• Enhanced Tracking and Reporting<br>• Effective Resource Allocation<br>• Better Root Cause Analysis<br>• Process Improvement<br>• Risk Mitigation<br><br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session. |

| | |
|---|---|
| | Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>**2. Suggested video lecture**<br>https://www.youtube.com/watch?v=_A_qQa4YIvc<br><br>**Reference website:**<br>https://www.softwaretestingstuff.com/2008/05/classification-of-defects-bugs.html<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- What are the primary purposes of classifying software defects?<br>- Explain the concept of "defect origin" in software defect classification<br>- Describe the difference between classifying defects by severity and classifying them by priority. Provide examples of each.<br>- What is the "Orthogonal Defect Classification" (ODC) method, and what key attributes does it use to classify defects?<br>- How can automated defect tracking tools assist in the classification and analysis of software defects?<br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Classification of software defects<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 6 | Course Name: Software Testing & automation<br>Topic: Test case design and test coverage | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the importance of test case design and test coverage.<br>b. Learn various test case design techniques (e.g., boundary value analysis, equivalence partitioning, decision tables).<br>c. Understand different types of test coverage (e.g., statement coverage, branch coverage, condition coverage).<br>d. Be able to apply test case design techniques to create effective test cases.<br>e. Understand how to measure and improve test coverage. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Briefly introduce the importance of software testing in the software development lifecycle.<br>- Explain the role of test cases in verifying software functionality.<br>- Introduce the concept of test coverage and its significance in ensuring quality.<br>- Briefly explain that good test case design increases test coverage.<br><br>- **Development** (30 minutes)<br>- **Test Case Design Techniques**<br>-<br>- **A. Equivalence Partitioning**<br>- Explain the concept of equivalence classes.<br>- Demonstrate how to identify valid and invalid equivalence classes.<br>- Provide examples and exercises.<br>- **B. Boundary Value Analysis**<br>- Explain the concept of boundary values and their importance.<br>- Demonstrate how to identify boundary values for different input ranges.<br>- Provide examples and exercises.<br>- **C. Decision Table Testing**<br>- Explain what decision tables are, and when to use them.<br>- Walk through creating a decision table from requirements.<br>- Show how to create test cases from the decision table.<br>- Provide examples and exercises.<br>- **D. (Optional) Other Techniques:**<br>- State Transition Testing |

| | |
|---|---|
| | - Use Case Testing<br>-<br>- **Test Coverage**<br><br>- **A. Introduction to Test Coverage**<br>- Explain the concept of test coverage and its purpose.<br>- Emphasize the importance of measuring test coverage.<br>- **B. Types of Test Coverage**<br>- Statement Coverage:<br>- Explain how statement coverage measures the percentage of executed statements.<br>- Provide examples and demonstrate how to calculate statement coverage.<br>- **Branch Coverage:**<br>- Explain how branch coverage measures the percentage of executed branches (decision points).<br>- Provide examples and demonstrate how to calculate branch coverage.<br>- **Condition Coverage:**<br>- Explain how condition coverage measures the percentage of boolean sub-expressions that are tested.<br>- Explain how it is different from branch coverage.<br>- **Path Coverage:**<br>- Briefly explain path coverage as an ideal, but often impractical, form of coverage.<br>- **C. Measuring and Improving Test Coverage**<br>- Discuss tools and techniques for measuring test coverage.<br>- Explain how to analyze coverage results and identify gaps.<br>- Discuss how to write more test cases to increase coverage.<br><br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. **Suggested video lecture**<br>https://www.youtube.com/watch?v=OtE7tjzaQPQ<br>https://www.youtube.com/watch?v=T-TIY9qaiXc<br><br>**Reference website:**<br>**https://www.geeksforgeeks.org/test-design-coverage-in-software-testing/** |

| | |
|---|---|
| | **Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- What is the primary purpose of test case design in software testing?<br>- What is test coverage, and why is it important?<br>- Explain the difference between statement coverage and branch coverage. Which provides more comprehensive testing?<br>- Discuss the challenges of achieving 100% test coverage and whether it is always necessary.<br>- Explain the relationship between test case design techniques and achieving high test coverage.<br><br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Test case design and test coverage<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 7 | Course Name: Software Testing & automation<br>Topic: Black-box testing techniques | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the concept of black-box testing.<br>b. Learn and apply various black-box testing techniques.<br>c. Understand the strengths and weaknesses of different black-box testing techniques. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Briefly introduce the importance of software testing in the software development lifecycle.<br>- Explain the role of test cases in verifying software functionality.<br>- Introduce the concept of test coverage and its significance in ensuring quality.<br>- Briefly explain that good test case design increases test coverage.<br><br>- **Development** (30 minutes)<br>- What is Black-Box Testing?<br>- Explain the concept of black-box testing and its focus on functionality without internal knowledge.<br>- Contrast it with white-box testing.<br>- Discuss the benefits of black-box testing (e.g., user perspective, independence from implementation).<br>- When to use Black-Box Testing.<br>- Explain the stages of testing when black box testing is most useful.<br>- **Black-Box Testing Techniques**<br><br>- **A. Equivalence Partitioning**<br>• Explain the concept of equivalence classes (valid and invalid).<br>• Demonstrate how to identify equivalence classes from requirements.<br>• Provide examples and exercises.<br><br>- **B. Boundary Value Analysis**<br>• Explain the concept of boundary values and their importance.<br>• Demonstrate how to identify boundary values for different input ranges.<br>• Provide examples and exercises. |

| | |
|---|---|
| | - **C. Decision Table Testing**<br>• Explain what decision tables are and when to use them.<br>• Demonstrate how to create decision tables from requirements.<br>• Show how to derive test cases from decision tables.<br>• Provide examples and exercises.<br><br>- **D. State Transition Testing**<br>• Explain the concept of state transitions and state diagrams.<br>• Explain how to create test cases based on state transitions.<br>• Provide simple examples.<br><br>- **E. Use Case Testing**<br>- Briefly explain how use cases can be used to derive test cases.<br><br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>**2. Suggested video lecture**<br>https://www.youtube.com/watch?v=qCs66O2NnH0<br><br>**Reference website:**<br>**https://www.geeksforgeeks.org/software-engineering-black-box-testing/**<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- What is the core principle of black-box testing, and how does it differ from white-box testing?<br>- Why is it important to test software from a "black-box" perspective?<br>- Explain how equivalence partitioning is used to design effective black-box test cases<br>- What are the limitations of black-box testing?<br>- How can use case diagrams be used to derive black-box test cases?<br>Spend 5 minutes to wrap up and consolidate the learnings |

| | |
|---|---|
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Black-box testing techniques<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 8 | Course Name: Software Testing & automation<br>Topic: Equivalence partitioning | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the concept of equivalence partitioning.<br>b. Learn how to identify valid and invalid equivalence classes.<br>c. Be able to apply equivalence partitioning to create effective test cases.<br>d. Understand the benefits of using equivalence partitioning. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Ask questions.<br>- Begin by asking students about the challenges of testing all possible inputs.<br><br>- **Development** (30 minutes)<br>- Introduce the concept of black-box testing.<br>- Explain the need for test design techniques to reduce the number of test cases while maintaining good coverage.<br>- Introduce Equivalence Partitioning as a black-box testing technique.<br><br>**Concept of Equivalence Partitioning**<br>• Define Equivalence Partitioning<br>• Explain the core idea<br><br>**Introduce the terms:**<br>• Valid Equivalence Class<br>• Invalid Equivalence Class<br><br>**Steps for Equivalence Partitioning:**<br>• Identify Input Domains<br>• Divide into Equivalence Classes<br>• Create Test Cases<br><br>**Explain the importance of selecting representative values:**<br>• boundary values,<br>• typical values, and<br>• error values.<br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |

| | |
|---|---|
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>2. **Suggested video lecture**<br>https://www.youtube.com/watch?v=jFNnKZTkWp8&list=PLGXz0cVCten6X9n6sthvZN6Zbg3iOCFbR&index=6<br><br>**Reference website:**<br>**https://www.geeksforgeeks.org/equivalence-partitioning-method/**<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>3. **Homework**<br>- What is Equivalence Partitioning, and what is its primary purpose in software testing?<br>- Why is it more efficient to use Equivalence Partitioning than to test every possible input value?<br>- Describe the steps involved in applying the Equivalence Partitioning technique to a given input domain.<br>- What are the limitations of Equivalence Partitioning? In what situations might it not be sufficient?<br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Equivalence partitioning<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 9 | Course Name: Software Testing & automation<br>Topic: Boundary value clustering | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the concept of Boundary Value Analysis.<br>b. Identify boundary values for input domains.<br>c. Understand the importance of boundary values in testing. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Briefly review the concept of input domains and the challenges of comprehensive testing.<br>- Introduce Boundary Value Analysis as a black-box testing technique.<br>- Explain that errors tend to occur at the boundaries of input domains.<br>- **Development** (30 minutes)<br>- Concept of Define Boundary Value Analysis:<br>- Boundary Value Analysis<br>- Explain the core idea<br>- **Boundary Values and Clustering Considerations**<br>- Here, we can discuss how boundary values can, in a way, "cluster" test cases around critical points.<br>- Explain that while not traditional "clustering," BVA groups test cases where errors are most likely.<br>- Also touch on how in data analysis, clustering algorithms are used to group similar data points, and that boundary values can be very important data points to consider when running those algorithms.<br>- For example, when looking at data ranges, the maximum and minimum values are often very important.<br>- Clarify that BVA is a focused test design technique, not a data clustering algorithm.<br>-<br>- **Exercise** (10 minutes) –<br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these. |

| | |
|---|---|
| | **2. Suggested video lecture**<br>https://www.youtube.com/watch?v=jFNnKZTkWp8&list=PLGXz0cVCten6X9n6sthvZN6Zbg3iOCFbR&index=6<br><br>**Reference website:**<br>**https://www.geeksforgeeks.org/software-testing-boundary-value-analysis/**<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- Emphasize the importance of clear requirements specifications for accurate boundary identification.<br>- Explain that BVA is often used in conjunction with other testing techniques, such as Equivalence Partitioning.<br><br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on Boundary value clustering<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |

| Lesson Plan No. 10 | Course Name: Software Testing & automation<br>Topic: White Box Testing | Course No: MCA-403 B |
|---|---|---|

| | |
|---|---|
| **Objectives** | At the end of the lesson the student shall be able to:<br>a. Understand the concept of White Box Testing.<br>b. Identify different White Box Testing techniques (Statement Coverage, Branch Coverage, Path Coverage, etc.).<br>c. Understand the advantages and disadvantages of White Box Testing. |
| **Teaching Aids (if any)** | a. Power point presentation |
| **Teaching Development** | - **Introduction** (5 minutes)<br>- Briefly introduce the importance of software testing in the software development lifecycle.<br>- Explain the role of test cases in verifying software functionality.<br>- Introduce the concept of test coverage and its significance in ensuring quality.<br>- Briefly explain that good test case design increases test coverage.<br><br>- **Development** (30 minutes)<br>- **Path Coverage**<br>- Define Path Coverage: Ensuring that every possible path through the code is executed at least once.<br>- Explain the challenges of achieving 100% path coverage (especially in complex code).<br>- Provide a code example with nested if statements or loops.<br>- Demonstrate how to identify and test different paths.<br><br>- **Condition Coverage**<br>- Define Condition Coverage: Each boolean sub-expression is evaluated to both true and false.<br>- Distinguish between branch and condition coverage.<br>- Provide examples.<br><br>- **Code Coverage Tools**<br>- Introduce popular code coverage tools (e.g., JUnit with JaCoCo, Coverage.py).<br>- Demonstrate how to use a tool to measure code coverage on a sample project.<br>- Explain how to interpret code coverage reports.<br>- **Advantages and Disadvantages of White Box Testing**<br>- **Advantages:**<br>- Comprehensive testing of internal code logic.<br>- Early detection of hidden errors. |

| | |
|---|---|
| | - Improved code quality and maintainability.<br>- **Disadvantages:**<br>- Requires in-depth knowledge of the code.<br>- Can be time-consuming and complex.<br>- May not detect missing requirements or usability issues.<br><br>- **Exercise** (10 minutes) –<br><br>Group Discussion at the end of the session.<br>Question and Answer session in between the lesson to check the presence of mind of students. |
| **Closure** | 1. Summarize the Lesson Learning Outcomes and get affirmation from students on these.<br><br>**2. Suggested video lecture**<br>https://www.youtube.com/watch?v=KMYtW4wVdEM&list=PLGXz0cVCten6X9n6sthvZN6Zbg3iOCFbR&index=10<br><br>**Reference website:**<br>**https://www.geeksforgeeks.org/software-engineering-white-box-testing/**<br><br>**Suggested Reading books:**<br>- Software Testing: Principles and Practices by Srinivasan Desikan Pearson Education , edition 1st (2005).<br>- Software Testing: Concepts and Operations by Ali Mili, Fairouz Tchier Wiley, edition 3rd (2015)<br><br>**3. Homework**<br>- What is the core principle of black-box testing, and how does it differ from white-box testing?<br>- Why is access to the source code essential for White Box Testing?<br>- What are the primary goals of White Box Testing?<br>- How can use case diagrams be used to derive black-box test cases?<br>- Describe the difference between "Branch Coverage" and "Path Coverage." Which provides more thorough testing?<br>Spend 5 minutes to wrap up and consolidate the learnings |
| **Evaluation** | 1. Reflective Questions (What, Why, Who?). Allow students to answer and discuss.<br>2. Quiz on White-box testing techniques<br><br>Spend 5 minutes to evaluate student assimilation of the lesson contents |