Group Coaching Session

The broad breakdown of the concepts taught in this module can be found in the Curriculum attached here.

Objective

This is the group coaching session that learners are attending after completing their course on **Microservice Debugging and Troubleshooting.** You can find the list of topics covered in the module here.

The agenda for this session is to help them learn some additional concepts around unit testing and integration testing, and help them understand why it is important for developers.

[2 minutes per learner for below activity]

- The coach will focus on getting to know the learners and check if they completed the module. He will try to understand if learners faced any road blockers in completing the module.
- The coach will take the doubts of learners from the module.

Group Persona Category	Work Ex.	Background
ET (Experienced-Tech)	High (> 7-8)	Technical
NET (Non-Experienced-Tech)	Low (< 7-8)	Technical
ENT (Experienced-Non-Tech)	High (> 7-8)	Non-Technical
NENT (Non-Experienced-Non-Tech)	Low (< 7-8)	Non-Technical

Agenda

Part-I: Introduction (5 mins)
Part-II: Focused Teaching (30 mins)
Part-III: Doubt Resolution (5 mins)
Part-IV: Focused Teaching (30 mins)
Part-V: Doubt Resolution (5 mins)
Part-VI: MCQs Quiz (5 mins)
Part-VII: Common Interview questions (10 mins)

Note for the coach:

- Use IntelliJ, JAVA[15] and the latest and stable version of Spring Boot for demos.
- **Follow live coding in the session and learners are expected to code along with you. Please keep them following you.**
- Please make sure to test run your codes/queries prior to the session to avoid any hick-up during the session.
- During the hand-on as the learners are supposed to code parallelly, make sure to explain the steps clearly, and keep track of the student's progress.
- Wherever PPT based presentations are there, please try to use slides with less text and bulleted points. As you take the learners through the concepts, respective points would appear on slides.
- Please provide the learners with some good reading links around concepts covered in the session.
- Please keep the session interactive.

Coding Demos

- You can create any application of your choice to show unit testing concepts. Please give them a quick walkthrough of the applications . Since the focus is on teaching the testing concepts, it is recommended to keep application simple.

Detailed Lesson Plan

Component	Instruction Task/Learner Task	Element of Engagement
Part-I: Introduction (5 mins)	Greet the learners and give a brief introduction of yourself and your work/background. Get to know about each of the learners one by one. *(Keep the introductions a bit brief as the focused teaching component might consume a fair amount of time.)* Ask the learners to summarise and explain their learnings briefly from the past week module on Microservices Debugging and Troubleshooting and understand the areas where they faced difficulty. Try to get the students to arrive at a broad consensus so that the focused teaching component can be driven through that.	Social Support

Part-II: Focused Teaching (30 mins)	Please note that in each group, a particular time frame must be dedicated to explaining the respective topics with appropriate examples depending on the group of learners. The main focus areas of teaching according to different personas are given below: **ET:** ● Help learners understand concepts around unit testing. You can keep this brief and focus more on the demo part. ● Give them a walkthrough of some of the unit testing code.[JUnit library can be used] ● Explain depth what each test case is doing, do talk about annotations **ENT:** ● Help learners understand concepts around unit testing. You can keep this brief and focus more on the demo part. ● Give them a walkthrough of some of the unit testing code.[JUnit library can be used] ● Explain depth what each test case is doing, do talk about annotations **NET:** ● Help learners understand concepts around unit testing in detail. ● Do live coding and show them how to create some of the unit test cases.[JUnit library can be used] ● Explain depth what each test case is doing, do talk about annotations **NENT:** ● Help learners understand concepts around unit testing in detail. ● Do live coding and show them how to create some of the unit test cases.[JUnit library can be used]	Content + Task-Support

	• Explain depth what each test case is doing, do talk about annotations	
Part-III: Doubt Resolution (5 mins)	Clear the doubts that the learners might have. *Don't spend too much time on unnecessary or vaguely-constructed doubts. Also, urge the other students on the call to answer their peers' doubts to induce peer interactions.*	Doubt Resolution
Part-IV: Focused Teaching (30 mins)	**Integration Testing-** **ET,NET, ENT,NENT:** • What integration testing is? • Why do integration testing? • Approaches, Strategies, Methodologies of Integration Testing **Note:** For ENT and NENT group, please try to explain things from very basic level	Content + Task-Support
Part-V: Doubt Resolution (5 mins)	Clear the doubts that the learners might have. *Don't spend too much time on unnecessary or vaguely-constructed doubts. Also, urge the other students on the call to answer their peers' doubts to induce peer interactions.*	Doubt Resolution

Questions

Please find a broad pool of questions below. This pool might contain two kinds of questions:

- Most frequent doubts that our learners face in these topics
- Common interview questions from these topics

Please try and address at least 4-5 of these during the session, irrespective of whether the students ask their doubts or not. Some of these might be covered as a part of the doubts coming from the students, so please make sure there aren't any repetitions from your side.

Some questions are more suited for one type of TG than the others. Such questions have been tagged with the appropriate personas and should be addressed during the session in accordance with your group's persona.

Questions:

ET

1. Can you explain what @Test annotation does?
2. What is the unit testing method naming convention that you should follow?
3. How will you test a private method?
4. Explain the functioning of the test runner.
5. How will you do exception handling of unit tests using @Test annotation?
6. What is mocking & stubbing? What are the key differences between them? Did you use any mocking framework?
7. What is code coverage? How is it measured? Name some JUnit code coverage tools?
8. Instead of worrying so much about writing test cases, why not just print outputs using System.out.println()?

ENT

1. Is there any difference between manual testing and automated testing?
2. Which is a better approach: manual testing or automated testing. Justify.
3. Do you think one should write code for every logic present in the code base?
4. Can you explain what @Test annotation does?
5. What is the unit testing method naming convention that you should follow?
6. Instead of worrying so much about writing test cases, why not just print outputs using System.out.println()?

NET

1. Can you explain what @Test annotation does?
2. What is the unit testing method naming convention that you should follow?
3. How will you test a private method?
4. Explain the functioning of the test runner.
5. How will you do exception handling of unit tests using @Test annotation?
6. What is mocking & stubbing? What are the key differences between them?
7. What is code coverage? How is it measured? Name some JUnit code coverage tools?
8. Instead of worrying so much about writing test cases, why not just print outputs using System.out.println()?

NENT

1. What is testing?
2. What is unit testing?
3. What is Junit? Can you name some libraries offered by Spring Boot which help in testing your application?
4. What do you understand about TDD?

5. Can you explain what @Test annotation does?
6. What is the unit testing method naming convention that you should follow?

NENT

1.	What is the main benefit that unit testing offers? a. It simplifies integration b. It provides easy documentation c. It facilitates changes **d. All of the above**
2.	Which development approach will you take to ensure overdesigning of an interface is not taking place? a. Agile Development b. Base Design Development **c. Test Driven Development** d. Waterfall Development
3.	Which of the following is the correct file extension for JUnit test files? a. .test **b. .java** c. .unit d. .junit
4.	Which of the following are the levels of testing? a. Integration testing b. Unit testing c. System testing **d. All of the above**
5.	Which of the following,according to you, is the key objective of Integration testing? a. Design Errors **b. Interface Errors** c. Procedure Errors d. None of the mentioned

ET

1.	What do you understand by the term fixtures in JUnit? a. Fixtures are objects that specify when to run a test **b. Fixtures are fixed state of a set of objects which are used as a baseline for running tests** c. Fixtures are bundle of few test cases which are run together d. Time objects
2.	What does the given syntax do? assertArrayEquals(“message”, A, B) a. It will check that “message” is in both A and B b. It will check that “message” is in A but not B c. It will check that “message” is in B but not A **d. It will assert the equality of the arrays A and B**
3.	Suppose you want to run the file TestClass.class using the command line.Which of the following commands will you type on the command line? a. java TestClass b. javac Testclass **c. java org.junit.runner.JUnitCore TestClass** d. org.junit.runner.JUnitCore TestClass
4.	What will the fail() method do in JUnit? **a. It will throw an assertion error unconditionally** b. It will call the default constructor c. It will print the message “Fail” on the console d. All of the above
5.	Do you think that automation testing should be performed before starting the manual testing? a. Yes **a. No**

NET

1.	What does the given syntax do? assertArrayEquals(“message”, A, B) e. It will check that “message” is in both A and B f. It will check that “message” is in A but not B g. It will check that “message” is in B but not A **h. It will assert the equality of the arrays A and B**
2.	What will the fail() method do in JUnit? **e. It will throw an assertion error unconditionally** f. It will call the default constructor g. It will print the message “Fail” on the console h. All of the above
3.	If you annotate a public void method with @Before,it will cause the method to be run before each Test method. Is the above mentioned statement true? **a. True** b. False
4.	Suppose you want to verify the actual and expected results in the Junits library. Which of the following methods will you use? **a) assert()** b) equals() c) == d) isEqual()

5.	Which of the following is the correct syntax to include Junit dependency in your Spring Boot application?
	
	a)
	
	`<dependency>`
	`<groupId>junit</groupId>`
	`<artifactId>junit</artifactId>`
	`<version>4.13.2</version>`

	</dependency> b) <dependency> <groupId>org.junit</groupId> <artifactId>junit</artifactId> <version>4.13.2</version> </dependency> c) <dependency> <groupId>mock.junit</groupId> <artifactId>junit</artifactId> <version>4.13.2</version> </dependency> d) <dependency> <groupId>junits</groupId> <artifactId>junit</artifactId> <version>4.13.2</version> </dependency> **Correct answer: a**

ENT

1.	Which of the following is the correct file extension for JUnit test files? e. .test **f. .java**

	g. .unit h. .junit
2.	What is the main benefit that unit testing offers? e. It simplifies integration f. It provides easy documentation g. It facilitates changes **h. All of the above**
3.	Which of the following,according to you, is the key objective of Integration testing? e. Design Errors **f. Interface Errors** g. Procedure Errors h. None of the mentioned
4.	Which of the following approaches are part of Integration testing? a. Top-down approach b. Bottom-up approach c. Big-bang approach **d. All of the above**
5.	Do you think that automation testing should be performed before starting the manual testing? b. Yes **b. No**