[00:00:03] **LayMui Toh** Hi, everyone. We'll come to today's Automation Guild 2022. My name is LayMui. Today, I'm going to share with you about bringing BDD to the API integration testing. What I'll be covering with you today is how we can use BDD to enhance our Agile methodology. I'll be sharing with you using the BDD features mapping for our microservices API and why I choose Serenity JS as my test automation framework and how I apply the screenplay pattern and the layered approach in my test automation. I'll also be sharing with you some best practices in Cucumber principles. I'll be using a simple project based on RESTful API and do a code walkthrough review. And how I integrate my test into the CI. So the CI server that I'll be using will be the circle CI. So I'll just also run the quick demo and generate that report and how this test reporting can serve as living documentation. With that, let me first share my screen.

[00:01:31] First thing first is how we can use BDD to enhance our Agile methodology? Many agile team, so what I observe is that without the BDD process they usually struggle when it comes to story estimation during the spring planning. And what I normally observe is that usually this meeting usually would take an hour or even longer than that. I can understand why they are struggling because really they estimate the task based on the Jira tickets is very difficult because there's not much information for them to estimate the user story. By bringing the BDD up process, we have this 3 Amigos Session where we can engage all the relevant stakeholders in the brainstorming process in discussing the user story detail. By doing that, they can be aware of the level of complexity for each task in a way, it helps them in better estimate the user stories or the tasks. BDD is also able to bring the QA to the front. When you involve them in the Amigo sessions, you are discussing the user story, you are discussing the acceptant criteria. Through that upfront, they can really think through all the possible test scenarios. This is also how the shift left strategy.

[00:03:24] By doing that, you actually help to eliminate and prevent that. It's a very collaborative effort, you encourage teamwork here. By having this discussion of user stories, we have shared an understanding of what we are trying to develop. This BDD process greatly helps augment and clarify what was in the Jira. What I also noticed that in Jira, either they have now they are too extreme, is either that not much information or no information, or they are too much information that the reader or the developer are lost.

Using the BDD and the technique that I'm going to share the BDD features maybe it helps to track the other details and the outcome is it helps to deliver new features faster.

[00:04:27] The very first thing that we want to do here is the bringing this BDD features mapping for our user stories. This can be applied even to the microservice API. It is more thorough than the sample mapping because it helps you to map out your user journey, so it also makes a transition to the Gherkins smoother. This transition will only be smooth if we applied some best practices here. For me, the tools that I'm going to use is the Miro board. Of course, you can use other tools like Figma as long as you have these color stickie notes.

[00:05:12] What I normally do is I will create a frame and this frame will be used to encapsulate my new features and I just name this frame according to the feature. For my case, in my demo version, I'm actually trying to perform this CRUD on this API, RESTful API endpoint.

[00:05:38] First thing first is actually I would just pull out a yellow card to describe my user story, so this will give the team an overall view of what these creatures is about? It's very helpful if you are able to describe these user stories on the yellow card, and then once you have this, then you are able to map out all your business through using the blue card.

[00:06:06] For my demo purpose, I would just result to about because they are a lot more. The first of one is able to create a new message. I'll be using this to what you through that my code later on. With each business rule, I will have a corresponding example.

[00:06:27] The example usually, you'll see some real data there, it's basically to illustrate your business through. This is really helpful for the developer to understand better what this business rule is trying to illustrate. And this is also directly called the thing into your test scenario for the test automation QA. And once you have this example, you are going to break it into steps.

[00:07:02] This is what BDD features mapping is about is to break down your example into different steps. These steps are actually your given step which mapped very nicely into your Gherkins. If you apply best practice here, moving into the Gherkins that will be very smooth. Let me jump to this slice on the best practice. When we come to API people tend to be technical here. You need not be so. Usually, we do not use technical terms either Gherkins even for API. Terms like endpoint, response code because all these might not mean anything to the TO or the business stakeholders or we try to avoid all these technical terms at this level because we're not going to engage them more. So we also don't want to talk about the implementation details at this level. By applying all these best practices.

[00:08:14] The next step is writing out the Gherkins. This will be just a method of cut and paste. Before I jump in to show you the code, I want to spend a bit of time talking about the framework of my choice. For me, I chose Serenity JS and why? Serenity JS is free and open-source. It will provide a set of libraries that helps you to write more maintainable, acceptance regression tests. You meet the criteria of readability and maintainability and allow me to apply the screenplay pattern, and this follows the solid principle.

[00:08:56] This is very important when it comes to programming. I list that why I choose why screenplay patterns and the layered approach. I notice some of those frameworks in a market. You tend to use a lot of helper functions, helper methods, and classic page objects. I tend not to avoid them because when you are testing scenario grows ids, my this

little maintenance issue. That's why I go for a screenplay. Screenplay pattern helped me to model my business domain requirements, and the metaphor is little scenarios, each that is little stitched actors performing actions. So the whole idea is actually to move away from thinking about interaction details about focus more on the behavior and the intent so it fits into that BDD concept very well.

[00:09:59] Because Screenplay pattern doesn't have anything to do with whether you are testing through UI or not. Screenplay pattern, in fact, is free and gives you more choices of where to connect your app. So that's why I can apply this screenplay pattern even to my API integration and testing. It keeps my code better and organized. And why the layered approach is? I want to do de-coupling as much as possible so that in a way, I make my code automation more modular. Serenity JS can integrate with the different test runners. One of them is a Cucumber, Cucumber, as you all know, is basically reading your Gherkins and mapping into your tests. Your code, all your automation code.

[00:10:51] One of the principles of Cucumber is that your scenarios are entirely independent of each other. How are you going to remember those sticks or right? Serenity JS gives you an API call? Takenote of and Note of. Takenote of is to record down state that you want to remember and then Note of is to retrieve that state, that value. With that, I will go into my code walkthrough, so this is the GitHub repository for my demo for this talk. It's not really accessible so you can take a look later on. So basically I would just start a web server using these scripts, then it was that better running on the localhost. And then once the webserver starts that I would run my test using these scripts that I defined in my package.Json.

[00:11:52] Then at the end of my test, I would just run the report to see the report generator, and then I will show you how I trigger the pipeline, the CI pipeline. And then once the circle CI pipeline tests are completed this test will be uploaded to the S3 bucket, which I create a static website to host my report so that the other team can access this report. Report is actually very important when it comes to test automation. This report is able to serve as are leaving documentation. Before I jump into my, show you the code, I want to show you how I organized my project. Usually, I had a features folder where I put all my features, and then if I have any objects Java domain object I can find them under the dto. Then I have the support folder where I put my support configuration command utility, then I have a task folder where all my task class.

[00:13:15] This is how the screenplay pattern that you have to actor one file to describe your act. In my case, I had only one actor. Basically, that's the definition you can put into one big format then this is not a good practice. Normally I would just have one feature for each step definition. So let me jump to my code here. So if you noticed under feature folders, I put my features, here I only had one feature. This features I will map to the corresponding steps. I named accordingly, so CRUD. And CRUD.steps. You also noticed there are 3 layers, first is the first, the first layer is acting like a single source is true. It's very important that you follow best practice to keep it concise and readable because you want to also present this later to the team, the PO to let them review.

[00:14:35] This file is. Keep in mind that you are going to. Your reader is going to be the team that they might not be technical people. I'ts good that you be mindful of some of the Gherkins best practices for example when I said trying not to be more than five steps and. If you remember the Miro Board, I have this user story, I just transferred it over. It's good practice that you had this session here so that you can people are aware of what this

feature file is trying to do. Then you have a given, which is a common step. And so you put it under the background.

[00:15:27] These are common steps, all your tests scenario. You also notice I used some pack here. If I go to the package.Json this is my test script, if I will quickly run my tests, what I do is npm test and I run the test scripts. This test, the colon stage will also run your two things, one plus is clean up first and you run that test executes, then the test report. The test is this line 11 where you call the Cucumber js utility and then you're passing tag option. So this case is a test.

[00:16:13] I passed the test pack. You will only pick up those tests scenario with the test pack. So now is running the tests. So there's this compromise, I think I haven't started the server yet. Okay you see that the execution is successful. You can just sit to run the report. Once the test is complete, it will generate a report to the target site. This is the report that you see here. The report will be generated to this folder to the target site Serenity. It's a collection of other HTML key. Let me go on to this. This is the first layer, Cucumber with read your Gherkins and then map you to the steps. The first one, line seven, given James is at that base url. So first time I would just mention the person that James subsequently that when, I can just use a pronoun to refer to this actor. This given James will be mapped to the steps.

[00:18:03] Here, line seven. You'll notice this, James, is not how code that is replaced by this expression actor. This actor well, is actually defined in your support configuration? You will notice the parameter is define, as long as you mapped those regular expressions A to Z correct. It will return you, this actor, you will map to this parameter name actor. Same thing for the pronoun. You mapped even any of this expression, he or she then is mapped to this pronoun. I mentioned that the layup was right. This is the features five the first and then the second layer is the steps. You noticed that even at this, second layer, I can still make my code very business domain using business domain language because of what the Serenity JS library has provided.

[00:19:17] The actor attempt to. These are API that comes from this library. It makes even the steps that definition very readable in terms of using business domain-specific language. In this case, you are trying to set up the base URL, then the when's that right? Again, you noticed this string, all these are Cucumber expressions. These are the parameters they pass in the author end the message. Because this author, I want to remember so that I can do my verification later, I call this TakeNote of to record down the value of my author so that I can pull out this value later at my 10th step, and then I do a session verification.

[00:20:12] Which is this line here, line 43. Actually, I'm using this. I need to make sure that you stick to this that was being recorded earlier. You'll notice that lines 31 to 32 I can actually call the post request directly, but I want to have this layered approach, I want to keep this level not so technical. I will instead of calling this directly the post request that put it high under that task. This is my task, I called to perform.

[00:20:58] When you read you are what you are trying to do. You're trying to create a message. Under the task class, I have defined my task class, which is actually basically a collection of all my lower-level activities. You notice that post requests are being defined here. This is how I design my test automation using this layered approach. Basically, that's it. Because I just make some changes, I want to show you how this committing this change? We'll trigger the CI. I pushed my change once I push my change. If I go to my

circle CI dashboard. This is my project that I set up in the circle CI. You can actually create an account it's a free account, you see the test is run, is triggered, and is running the test now.

[00:22:20] The circle CI configuration, these are the jobs are actually defined in my config.yml. I think for the interest of that I won't be able to go through the details of the completion, but mainly this is all the steps raised as actually what I was run locally in my local run. I started a webserver I think that I would run my test. And at the end of it, it will generate a report to my target site serenity folders then I will just call the s3 AWS CLI command to upload this to the S3 bucket so that I can able to post my report.

[00:23:12] This test is completed and it's passed, the test result is uploaded to S3 bucket. The S3 bucket is, this is my local. This is the website, the URL for the S3 bucket. Again, is also accessible by any people in the team. If you look at this is exactly the same report that you will see locally. And it's a very nice detailed report that serves as living documentation, it shows you the high level user stories. This, of course, is picked up from the features file. Now you can drill down to the detail as well. Even the steps, you can see the rest query, the response body as well, it's also very good when it comes to debugging. In case the test, you can also use this report for debugging purposes. I guess I have finished my sharing.

[00:24:33] I stop my sharing. Just to wrap out, I think I have show you the QA process using the BDD, bringing in the BDD, and applying the BDD features mapping as the first step to describe your user stories and then mapping all your test scenarios and then applying best practice into your BDD features mapping so that you can transit into your Gherkins very smooth and end with these Gherkins written in a very readable and concise manner.

[00:25:12] You can also choose a framework that fits the BDD, and this framework actually allows you to apply solid principles using the screenplay pattern so that overall, your code is very easy to maintain, easy to scale, and easy to read. A map into your step definitions and I hope my talk is very useful for those you guys out there to consider the framework and the QA process for your API integration testing. Thank you.