Test Assessment:

I have been given the problem statement i.e.

Singtel wants to launch prepaid customer portal. For that one of our product owners created below user stories

User Story 1: Customer should be able to search for available plans.

Example data: Plan name: “Singtel-Basic”, cost: “10 SGD”, details: “100 SMS, 500 calls, 5 GB”.

User Story 2: Customer should be able to top up with selected plan.

User Story 3: Customer should be able to search for add on features.

feature 1: data:10 GB, cost: 5 SGD

feature 2: 100 SMS, cost: 1 SGD

feature 3: data 100 GB, cost:20 SGD

User Story 4: Customer should be able to top-up their selected add-ons with their existing plan.

User Story 5: Customer should be able to remove their addon feature.

Description:

Please note that above is the basic requirement and you can have your own imagination on creating the sample data.

There is no right or wrong approach only thing you need to provide justification on your solution if anyone asks.

Acceptance Criteria:

Each user story contains each API endpoints and make sure you should follow below steps,

1. Please provide API contract details. It can be swagger or postman or any other document.
2. API must use minimum of 3 to 4 HttpMethods (GET/POST/PUT/DELETE/PATCH/HEAD/etc.,)
3. API must be protected you can use any security mechanism.
4. Project must keep readme.md file to explain the project.
5. You can use any Database.

Good to have:

1. Need to have integration test using mockmvc.
2. Make sure 80% code coverage you can use jacacco plugin in your pom file.

Expected to use, Spring Boot 3, Maven, Java 17

For this Problem Statement, My Solution is :

* I created 3 microservices, namely planMS, addOnMS and customerMS.
* I created planMS so that the company can add plans, delete, update, search plans by name, cost, and details.
* I created addOnMS so that company can add addOns according to them. They can also delete and update the addOns. Using this microservice we can implement the searching of addOns based on featureName and cost.
* I created the customerMS so that the new customer can be added and deleted and also the plan selected by the customer is also registered. The addOns can also be added for a respective customer and removed.

For this problem statement my flow is :

* First the customer lands on the website (frontend is not implemented due to time constraint) and if he is already a registered user then he will login and if he is not a registered user he will register himself and then login. (Register MicroService and Login Microservice is not implemented as of now due to time constraint).
* After logging in the customer will see a interface where he can see all the plans available(this is implemented in the planMs i.e. getAllPlansHandler in the controller). Here the customer can also be able to search for his/her required plans by its name, or cost, or plan details( this is implemented in planMs i.e. getPlansByNameHandler, getPlansByCost and getPlansByDetils).
* The customer will now select a plan and he will be redirected to a different page (in the frontend when he selects a plan the planId associated with it is temporarily stored in the local storage) now in this redirected page he will be able to see add the different addOns he can select if he requires. Here he can also search for a particular AddOn that he likes, he van search based on addOn FeatureName or addOn Cost (this is implemented in addOnMS i.e. getAddOnsByCostHandler and getAddOnsByFeatureNameHandler) . If the customer doesnot select any addOns and proceed with is recharge then the customer details and the plan id which he selected will be registered in the database( this is implemented in the customerMs i.e. selectedPlanHandler).
* If the customer add few addOns in the previous steps (selected addOns ids will be temporarily stored in the local storage and then used later for registering) and then proceed with the recharge then the customer details and the plan id and the addOnsIds which he selected will be registered in the database(this is implemented in the customerMs i.e. addAddOnHandler).
* The customer can also remove the addOns he selected (this is implemented in the customerMs i.e. removeAddOnsHandler.
* By this way the customer can search in the available plans, select his/her required plan, search for the addOns the customer requires and add or remove a respective addOn.

If given more time, I would have been able to implement the registerMS and loginMS and implement the JWT authentication along with it and also proper exception handling for the solution . And also would have implemented the respective frontend for this.

How to run the solution:

Import the 3 microservices in STS or any ide and after importing build the microservices once or update the microservices and then run the microservices individually. After running the microservices, test the Microservices by using the applications such as postman, insomnia, etc.

There is a file along with the solution named “API List” in this it is clearly explained what each API can be used for.