Project 1: Tic-Tac-Toe

Level 1 : Build a rest API to play 3X3 . The UI should call the services to move to the next board state. 1 player=you, 2 player=computer. Must have the ability to reset the game at anytime and start over.

UI :
- React / Angular
- TDD with jasmine/karma/enzyme
- State management – Redux

Build:

Services:
- Java,.Net, Scala, Spring,
- TDD with junit, Mockito
- Integration tests
- Contract Tests (spring contracts)

Cloud : Azure, AWS, K8 OpenShift

Pipeline : AWS Cloudformation, AWS CodePipeline, Jenkins, Azure pipelines

Level 2 : Have a Player 2 play the game (instead of the computer) . Call the services asynchronously. 1 player=you, 2 player=Player2.

Service : Spring Webflux

Messaging: SQS, Azure Service Bus, RabbitMQ, Kafka

Level 3 : Give the user a choice to pick best of 'x' games to call out the winner. Keep track of the 'x' scores and tally the winner. Users will be authenticated and only registered users will be allowed to play the game. Use config server to store application config properties.

Database : MySql, DynamoDb, Mongo

Security : OAuth, JWT, Cognito

Level 4: Add Monitoring and alerting. Monitor if the service takes more than 'x' ms to generate the next board state, if it does generate an alert, which will send an email notification to you. (You can add additional delay, to simulate an alert).

Profiling: Micrometer, cloudwatch, Prometheus

Project 2: Rock Paper Scissors

Level 1 : Build a rest API to play Rock-Paper-Scissors . The UI should call the services to move to the next board state. 1 player=you, 2 player=computer.

UI :
- React / Angular
- TDD with jasmine/karma/enzyme
- State management – Redux

Build : gradle

Services:
- Java, .Net, Scala, Spring ,
- TDD with junit, Mockito
- Integration tests
- Contract Tests (spring contracts)

Cloud : Azure, AWS, K8 OpenShift

Pipeline : AWS Cloudformation, AWS CodePipeline, Jenkins, Azure pipelines

Level 2 : Have a Player 2 play the game (instead of the computer) . Call the services asynchronously. 1 player=you, 2 player=Player2.

Service : Spring Webflux

Messaging: SQS, Azure Service Bus, RabbitMQ, Kafka

Level 3 : Give the user a choice to pick best of 'x' games to call out the winner. Keep track of the 'x' scores and tally the winner. Users will be authenticated and only registered users will be allowed to play the game. Use config server to store application config properties.

Database : MySql, DynamoDb, Mongo

Security : OAuth, JWT, Cognito

Level 4: Add Monitoring and alerting. Monitor if the service takes more than 'x' ms to generate the next board state, if it does generate an alert, which will send an email notification to you.

Profiling: Micrometer, cloudwatch, Prometheus

Project 3: Product Availability

Level 1 :
Build a product microservice which maintains information of productid, name, dept Id, dept name eg
11234, Long Sleeves,100, Shirts

Build a location microservice which maintains information of a location- locationId, locname, zipcode eg. 500, Irving, 75063

Build a balance service, which has the available items in a given location. productid, location Id, balance e.g 11234, 500,10 implying there are 10 long sleeve shirts in zip 75063

Usecase – Find the availability of a given item and/or department in all locations where the item/department is available.

UI :
- React / Angular
- TDD with jasmine/karma/enzyme
- State management – Redux

Build : gradle
Database : MySql, DynamoDb, Mongo
Services :
- Java, .Net, Scala, Spring ,
- TDD with junit, Mockito
- Integration tests
- Contract Tests (spring contracts)

Cloud : Azure, AWS, K8 OpenShift
Pipeline : AWS Cloudformation, AWS CodePipeline, Jenkins, Azure pipelines

Level 2 : Usecase - Given an item and a zipcode, find the nearest x(given) locations within x(given) miles radius. (Use an external api - http://www.geonames.org/export/web-services.html#findNearbyPostalCodes)

Cache : Redis , Memcached
Fault Tolerence – hystrix , spring retry

Level 3: Use Case – Be able to view availability in real time for the items in a given location so the items that are running low on availability can be ordered sooner.

Build a batch job, which will call the balance service and update the availability for an item at a location. Stream the update events and display the data in real time graph.

Batch : AWS Batch
Streaming: Kinesis, apache beam, Kafka streams

Level 4 : Add security – Only the batch job has to be authorized to update the balance table. Users will be authenticated and only authorized users will be allowed to see the real time graph. Use config server to store application config properties.

Security : OAuth, JWT, Cognito

Level 5: Add Monitoring and alerting. Monitor if the service takes more than 'x' ms to generate the next board state, if it does generate an alert, which will send an email notification to you.

Profiling: Micrometer, cloudwatch, Prometheus

Bench Advisors:

- Each Bench Advisor to assist 5-6 associates
- Level 1 tasks and Level 2 tasks should take ~3 weeks, each additional level to take up to 2 weeks.
- Have a daily standup to review the tasks people are working on
- Have a weekly demo of what has been completed.
- Submit a weekly assessment for the associates. (I think we should only provide and document feedback for associate who are under performing or doing exceptional). The assessments should be handled by delivery managers – Asha/Vinoth

- Schedule a bi-weekly demo with other teams, will choose an associate to demo their project.
- Attend the bi-weekly review meetings hosted by Jyothi/Cikia to assess the program/feedback/ improvements.

Assessment Criteria:

1 = Not participating 2 = barely managing, 3 = satisfactory , 4 = Exceptional

- Is the associate learning and improving his/her technical skills
- Is the associate engaged – completing tasks for the day and the progress is satisfactory
- Attendance - Associate is attending all scheduled meetings, participating actively, absences are informed and recorded
- Associate is completing the tasks for the days and progress is satisfactory
- Did the associate commit for certification and is on track