Question - 1
Mocking Beans

Given the following implementation of *DataService* which uses *ApiClient* component to call external APIs:

```
@Component
public class ApiClient {
  public void call() {
   //not relevant code
  }
}

@Service
public class DataService {
  @Autowired
  private ApiClient apiClient;
  public void collectData() {
    apiClient.call();
  }
}
```

The requirement is to implement the test and don't call external API during the test execution. The following code has been implemented:

```
@SpringBootTest
class DataServiceTest {
  <CODE HERE>
  @Autowired
  DataService dataService;
  @Test
  void collectData() {
    dataService.collectData();
    //test code
  }
}
```

What should be inserted instead of *<CODE HERE>* to achieve the goal?

- @Mock ApiClient apiClient; @InjectMocks
- @InjectMocks
- @MockBean ApiClient apiClient;
- @Captor ApiClient apiClient;
- @SpyBean ApiClient apiClient;

Question - 2
Secure Method

Given the following controller method, assume that *usersService* is correctly implemented and autowired.

```
<CODE_HERE>
@GetMapping("/users/{id}")
public ResponseEntity<UserResponse> get(@PathVariable @NotNull UUID id) {
  return ResponseEntity.ok(usersService.get(id));
}
```

The requirement is to make sure that only users with any of *ROLE_ADMIN* or *ROLE_USER_MANAGER* roles assigned will be able to execute this method. What can be put in place of '<CODE_HERE>' to implement this requirement using Spring Security?

- @PostAuthorize("hasRole('ROLE_ADMIN') or hasRole('ROLE_USER_MANAGER')")
- @PreAuthorize("hasAnyRole('ROLE_ADMIN','ROLE_USER_MANAGER')")
- @Secured("hasRole('ROLE_ADMIN') or hasRole('ROLE_USER_MANAGER')")
- @PreAuthorize("hasRole('ROLE_ADMIN') and hasRole('ROLE_USER_MANAGER')")

Question - 3
Transactional Tests

Given the following test code, which of the statements is true?

```
@ExtendWith(SpringExtension.class)
@Transactional
@ContextConfiguration
class UserRepositoryTest {
  @Test
  @Commit
  void test1() {
    /* non relevant code */
  }
  @Test
  @Rollback(false)
  void test2() {
    /* non relevant code */
  }
}
```

- Only the transaction for method test1() will be committed.
- Only the transaction for method test2() will be committed.
- Transactions for both methods will be committed.
- @Transactional will lead to a runtime error when running the tests.

Question - 4
Postgres Specific Service

The requirement is to implement a Spring Boot *@Service* that should be loaded to the spring context only if the *org.postgresql.Driver* class is present on the classpath, and the *application.properties* file contains the property *database.vendor=postgres*.

```
<CODE HERE>
@Service
public class PostgresSpecificService {
```

```
  /* not relevant code*/
}
```

Which of the following annotation options can replace *<CODE HERE>* to achieve this?

- @ConditionalOnProperty(name = "database.vendor", value = "postgres") @ConditionalOnClass(name = "org.postgresql.Driver")
- @ConditionalOnProperty(prefix = "database", name = "vendor", havingValue = "postgres") @ConditionalOnClass(name = "org.postgresql.Driver")
- @ConditionalOnProperty(name = "database.vendor", havingValue = "postgres") @ConditionalOnMissingBean(org.postgresql.Driver.class)
- @ConditionalOnProperty(prefix = "database", name = "vendor", havingValue = "postgres") @ConditionalOnBean(name = "org.postgresql.Driver")
- None of the above

Question - 5
Bean Scopes

Consider the following code.

```
<CODE HERE>
public class Processor {
  public void process() {
    /* not relevant code */
  }
}

@RestController
public class ProcessController {
  @Autowired
  private Processor processor;
  @GetMapping("/process")
  public void process() {
    processor.process();
  }
}
```

What should be inserted in place of *<CODE HERE>* to have a new instance of processor created every time the */process* endpoint is called?

- @Scope("prototype") @Component
- @Prototype
- @Component @Scope(scopeName = "prototype", proxyMode= ScopedProxyMode.TARGET_CLASS)
- @Component(`autowireCandidate=true`)
- @Service(alwaysNew=true)

Question - 6
Spring Circular Bean Trap

Consider the following code.

Circular.java

```
package spring.circular;

public interface Circular {
    void doCircularThings();
}
```

CircularBeanA.java

```
package spring.circular;
import org.springframework.stereotype.Component;
import javax.annotation.PostConstruct;

//X
@Component
public class CircularBeanA implements Circular {
    private Circular circularBeanB;

    public CircularBeanA(
        //Y
        Circular circularBeanB) {
        this.circularBeanB = circularBeanB;
    }

    @Override
    public void doCircularThings() {
        System.out.println("CircularBeanA: did bad things");
    }

    @PostConstruct
    private void init() {
        System.out.println("CircularBeanA: initialized");
    }
}
```

CircularBeanB.java

```
package spring.circular;

import org.springframework.context.annotation.Lazy;
import org.springframework.stereotype.Component;
import javax.annotation.PostConstruct;

@Component
public class CircularBeanB implements Circular {
    private Circular circularBeanA;

    public CircularBeanB(
        @Lazy
        //Z
        Circular circularBeanA) {
        this.circularBeanA = circularBeanA;
    }

    @Override
    public void doCircularThings() {

    }

    @PostConstruct
    public void init() {
        System.out.println("CircularBeanB: initialized");
        circularBeanA.doCircularThings();
```

```
    }
}
```

Which of the following options are true regarding this code?

- The application runs successfully and it'll output
  CircularBeanA: initialized
  CircularBeanB: initialized
  CircularBeanA: did bad things
- The application won't run.
  The code doesn't compile.
- The application runs but it will throw `NoUniqueBeanDefinitionException` and it will exit.
- If @Qualifier("circularBeanA") annotation is put on Z and @Qualifier("circularBeanB") put on Y, the application will not throw BeanCurrentlyInCreationException and runs successfully.
- The application runs and it will write
  CircularBeanB: initialized into console then
  it will throw org.springframework.beans.factory.BeanCurrentlyInCreationException.
- If @PostConstruct annotation is removed in CircularBeanB, it will run successfully and will print
  CircularBeanA: initialized into console
- if @Primary annotation is put in X place, it will run successfully.

Question - 7
Spring AOP Usage

Consider the following code.

NotifierMetricLogger.java

```
package spring.listener;

import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.AdviceName;
import org.springframework.stereotype.Component;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.ProceedingJoinPoint;
import java.util.logging.Logger;

//X
@Component
public class NotifierMetricLogger {
    private static final Logger log = Logger.getLogger(NotifierAspect.class.getName());
    //Y
    public Object beforeNotifyLogging(ProceedingJoinPoint joinPoint) throws Throwable {
        long startDate = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        long executionTime = System.currentTimeMillis() - startDate;
        log.info("Notify process time :" + executionTime);
        return proceed;
    }
}
```

TwitterNotifier.java

```
package spring.service.impl;
```

```
import spring.service.Notifier;
import org.springframework.stereotype.Component;
import java.util.logging.Logger;

@Component
public class TwitterNotifier implements Notifier {
    private static final Logger log = Logger.getLogger(TwitterNotifier.class.getName());
    @Override
    public void notify(String message) {
        log.info("TwitterNotifier: " + message);
        //send notification to home page
    }
}
```

Notifier.java

```
package spring.service;

public interface Notifier {
    void notify(String message);
}
```

Assuming the Spring Boot application is configured to use AOP with @EnableAspectJAutoProxy(proxyTargetClass = true) annotation, to capture *TwitterNotifier.notify(String message)* method's process time, which of the following options should be placed in the *X* and *Y* positions in NotifierMetricLogger.java?

- X = @Aspect
  Y = @Before("execution(* spring.service.impl.*.notify(..))")
- X = @AdviceName("NotifierMetricLogger")
  Y = @Around("execution(* spring.service.impl.*.notify(..))")
- X = @AdviceName("NotifierMetricLogger")
  Y = @Before("execution(* spring.service.impl.*.notify(..))")
- X = @Aspect
  Y = @Around("execution(* spring.service.Notifier.notify(..))")

Question - 8
Bean definition enhancement

During the startup of a Spring Boot application, it needs to read bean configuration metadata and change it before the container instantiates any beans.

How can this be achieved in an efficient and scalable way?

- Implement BeanPostProcessor.
- Implement BeanFactoryPostProcessor.
- It is not possible to change beans metadata on runtime. All beans metadata is defined at compile time.
- Implement Aspect.

Question - 9
Behavior Inheritance

A Spring Boot application has the following hierarchy of classes.

```
public class Animal {
  @PostConstruct
  private void init() {
    System.out.println("Animal init");
  }
}

@Component
public class Cat extends Animal{
  @PostConstruct
  public void init() {
    System.out.println("Cat init");
  }
}

@Lazy
@Component
public class Dog extends Animal{
  @PostConstruct
  public void init() {
    System.out.println("Dog init");
  }
}
```

What is the output?

- IllegalBeanDefinitionException: @PostConstruct should be applied to a public method
- Cat init Dog init
- Animal init Cat init
- Animal init Cat init Animal init Dog init
- Animal init Cat init Dog init or Animal init Dog init Cat init

Question - 10
Multiple Beans Definition

A Spring application has an interface called *Server,* and two implementations: *ServerA* and *ServerB*. There is a class, *ServerManager,* that uses the *Server* bean as a dependency.

```
public interface Server {
}

@Service
public class ServerA implements Server {
}

@Service
public class ServerB extends ServerA {
}


@Service
public class ServerManager {
  @Autowired
```

```
  Server server;
}
```

Which of the following statements is true about this code?

- The code throws `InterfaceNotInstantiatableException`.
- The code runs fine. A random Server implementation is injected into the `server` field.
- The code throws `NoUniqueBeanDefinitionException`.
- The code does not compile.

Question - 11
Testing a Spring Application

There is a Spring boot web application that uses a relational database for data storage. The *org.springframework.boot:spring-boot-starter-data-jdbc* starter is used in the implementation of the data access layer. Now tests are needed that cover the functionality of the data layer in isolation.

What Spring test annotation is recommended when constructing the test context?

- @DataJpaTest
- @DataJdbcTest
- @SpringBootTest
- @WebMvcTest

Question - 12
Path Variable

In the Spring MVC controller, which of these are valid uses of the @PathVariable annotation?

- @RequestMapping(value="/users/{userId}/addresses/{addressId}") public String viewUserAddress(@PathVariable String userId, @PathVariable String addressId, Model m)
- @RequestMapping(value="/users/{userId}") public String viewUser(@PathVariable("users") String user, Model m)
- @RequestMapping(value="/users/{userId}") public String viewUser(@PathVariable String userId, Model m)
- @RequestMapping(value="/users/{userId}") public String viewUser(@PathVariable("userId") String personnelId, Model m)

Question - 13
Component Dependency

The following @Configuration contains definitions for 2 beans: ServiceA and ServiceB. Imports are omitted.

```
@Configuration
public class ServiceConfiguration {
  @Bean
  public ServiceA serviceA(ServiceB serviceB) {
```

```
    return new ServiceA(serviceB);
  }
  @Bean
  public ServiceB serviceB(ServiceC serviceC) {
    return new ServiceB(serviceC);
  }
  @Bean
  public ServiceC serviceC(ServiceA serviceA) {
    return new ServiceC(serviceA);
  }
}
```

Which of the following statements is true?

- This code does not compile.
- This code runs fine and creates 3 Beans: `ServiceA`, `ServiceB` and `ServiceC`.
- The code throws `BeanCurrentlyInCreationException` when run.
- The code throws `StackoverflowError` when run.

Question - 14
Application Properties Override

In the classpath of a Spring Boot application, there are 2 files with properties *application.properties* and *application-prod.properties*. It is required to always load properties from *application.properties* and override with values in the *application-prod.properties* file only when the application is deployed on the production server.

What should the value of the environment be to achieve this on the production server?

- spring.properties=application-prod.properties
- spring.profiles.active=application-prod
- spring.profiles.active=prod
- environment=prod

Question - 15
Spring Dependency Injection

```
@Service
public ProductService {
    private ProductRepository productRepository;
    private ProductMapper productMapper;

    @Autowired
    public ProductService(ProductRepository productRepository,
                          ProductMapper productMapper){
        this.productRepository = productRepository;
        this.productMapper = productMapper;
    }

}
```

Which type of injection is implemented in the ProductService?

- Setter
- Getter
- Property
- Construction

Question - 16
Create a User Controller

An *HTTP POST* endpoint accepts a JSON representation of a *User* object. It uses the following *User* and *UserController* classes.

```
public class User {
  @NotEmpty
  private String name;
  @NotEmpty
  private String surname;
}

@RestController
public class UserController {

  @PostMapping("/users")
  public void create(@RequestBody User user) {
   System.out.println("name: "+user.getName() + " surname: "+user.getSurname());
  }
}
```

What is the result of a call with the payload below?

```
{
 "name": "Duke",
 "surname": null
}
```

- BAD REQUEST 400 and a message that `user.surname` is validated by @NotEmpty annotation.
- OK 200 and prints 'name: Duke surname: null'
- SERVER ERROR 500 because of a NullPointerException during validation of `null` string for @NotEmpty
- OK 200 and prints 'name: Duke surname:'

Question - 17
Query Annotation

Which of the following annotations can be used for a specific custom query in a Spring data JPA repository method?

- @JpaQuery
- @SqlQuery

- @DataQuery
- @Query

Question - 18
Property Value Injection

Which of the following annotations can be used to inject property values into Spring Boot beans and configuration classes?

- @Values
- @Property
- @Inject
- @Value

Question - 19
Dependency Injection

Which of the following options can be used for dependency injection in Spring Boot?

- Setter injection
- jsonConfiguration
- Constructor injection
- @Autowired

Question - 20
Endpoints

Which Spring-based code snippet must fill the blank such that Actuator security rules are present, and all endpoints fetch for actuator?

```
@Bean
public PrivacyChain demoChain(ServerHttpSecurity http) {
    return http.authorizeExchange()
    .pathMatchers(_____________).permitAll()
    .and().build();
}
```

- /actuators/**
- /actuators.**
- /actuator/**
- /actuator/info/**

Question - 21
Actuator

A dependency has been added to pom.xml in a Spring application as follows.

```
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>
```

What is the starter used for?

- application analysis
- creation of an application
- automation of an application
- production of application instances

Question - 22
Validator

In a RESTful application using Spring, which of the following is not used to create an object validator?

- @Min(value = 1)
  @Max(999999)
  private int id;
- @Size(limit = 100)
  private String name;
- @NotNull
  private Boolean isActive;
- @ValidCategory(categoryType="sample")
  private String category;

Question - 23
Rest Handle

When building a Spring-based RESTFul application, a new resource must be created with the request URI of "/handle". Which of the following code segments is most appropriate?

```
@RestController
public class demoHandler {
 private static Map<String, Product> dataRepo = new HashMap<>();
 @RequestMapping(value = "/handle", method = RequestMethod.POST)
 public ResponseEntity<Object> createProduct(@RequestBody Product product) {
    .......
 }
}
```

```
@RestController
public class demoHandler {
  private static Map<String, Product> dataRepo = new HashMap<>();
  @RequestMapping(value = "/handle", method = RequestMethod.DELETE)
   public ResponseEntity<Object> createProduct(@RequestBody Product product) {
      .......
  }
}
```

```
@RestController
public class demoHandler {
   private static Map<String, Product> dataRepo = new HashMap<>();
   @RequestMapping(value = "/handle", method = RequestMethod.CREATE)
    public ResponseEntity<Object> createProduct(@RequestBody Product product) {
       .......
    }
}
```

```
@RestController
public class demoHandler {
   private static Map<String, Product> dataRepo = new HashMap<>();
   @RequestMapping(value = "/handle", method = RequestMethod.GET)
    public ResponseEntity<Object> createProduct(@RequestBody Product product) {
       .......
    }
}
```

Question - 24

Classes Segment

A Spring application has four classes. Select the option which would contain the business logic.

- @Service public class A{ .. }
- @Repository public class A{ .. }
- @Primary public class A{ .. }
- @Session public class A{ .. }

Question - 25
Mechanism

Which annotation should be used instead of <<blank>> that will allow one to use the class for data storage, update, and retrieval?

```
<<blank>>
public class DemoSession implements DemoInterface{

  @Override
  public void save(Student student) {
   .....
  }
}
```

- @Service
- @Repository
- @Session
- @Autowired

Question - 26
Spring Annotation

What does the annotation in this Spring-based code segment do?

```
@Component
public class ComponentExample {
  int x;
  public void show(){
    System.out.println("Hello");
  }
}
```

- It allows Spring to create containers.
- It allows Spring to create components.
- It allows Spring to collect bean instances.
- It allows Spring to detect custom beans automatically.

Question - 27
Employee

In the following Spring Boot code, what kind of dependency is injected by the annotation?

```
public class Employee {

  @Autowired
  private Roll roll;

  Employee() {
  }
}
```

- constructor based
- setter bean
- field based
- Spring IOC

Question - 28
Book

```
@Entity
public class Book {

    @Id
    Long id;
    String author;
    String year;
    ....
}
```

For this model, which option is a valid derived query method for its corresponding JpaRepository?

- findAuthorById
- findByAuthor
- findByAuthorAndIdAndYear
- findBook

Question - 29
FooBar

Which link will hit the following method, where the application context is '/home'?

```
@RequestMapping(value = {"/ex/basic/bar", "/ex/basic/foo"}, method = RequestMethod.GET)
public String getPath() {
    return "FooBar";
}
```

- "GET request http://localhost:8080/home/ex/basic/bar"
- "GET request http://localhost:8080/home/ex/basic"
- "POST Request http://localhost:8080/home/ex/basic/bar"
- Two routes cannot be mapped to one resource.

Question - 30
RESTful

In Spring's approach to building RESTful web services, how are the HTTP requests handled?

- They are handled by a controller and identified by the @RestController annotation.
- They are handled by an object.
- They are handled by a class.
- They are handled by a controller and identified by the @GreetingController annotation.

Question - 31
Microservices Security

A company deals has implemented the GPT-3 AI Text Generation using microservices infrastructure for handling thousands of users. The operations team receives a notification that three of their services are down. There is evidence of image pullback failure. Upon investigating, they deduced that one of their services was internet facing and was exploited. The exploited service started messaging other services with corrupted data. The team applied security patches to fix the problem.

What is the fix?

- The team added mTLS and added rules for inter-service communications. After that, they pushed the internet-facing server behind a load balancer and applied JWT tokens for authentication.
- The team cut off the internet-facing service from the infrastructure and applied a JWT token for each call to other services.
- A and B

Question - 32
Microservices Reliability

A chat-based app like Whatsapp is using microservices to connect thousands of real-time users together. They have a flaw in the design of the microservices that all the reads and writes go through one service, "Service A". Service A provides them a great deal of volatility and unpredictability. The company would like to refactor the codebase to address the unpredictability of the system.

Which of the following decisions should they go with?

- Separate the read and write APIs into separate services.
- Add a queue to the write API and a cache on the read API so the system can efficiently handle messages.

Add redundancy to Service A.

Question - 33
Microservices Usage

A services-based company is building an AI tooling system that can help their client streamline their training and inference data set using AWS Fargate. However Fargate does not provide GPU access to users, and it is expensive for the client. The team decided to implement a custom solution that can scale and train custom models of users. Given the short time span, they have two options:

1. Build the custom Solution using Microservices, LinkerD, MicroK8s.
2. Use an existing system like KubeFlow and build custom features on top of it.

Which of the following is the best way forward?

- Try 2 then 1.
- Try and test 1.
- Try 1 and then 2.

Question - 34
Microservices Communication

A microservices-based architecture for an e-commerce platform is in the design phase. A core requirement is to ensure that the system can handle eventual consistency while ensuring that the platform's various services remain loosely coupled. Two services, Order Service and Inventory Service, need to communicate regarding the availability of products.

When a customer places an order, the Order Service needs to check if the product is in stock using the Inventory Service. However, the design must ensure that even if the Inventory Service is temporarily unavailable, the Order Service can still accept orders, albeit with some potential delay in processing.

Which of the following communication patterns is most suitable to ensure that the services remain decoupled and the system handles eventual consistency?

- Synchronous HTTP REST API calls from Order Service to Inventory Service
- RPC calls from Order Service to Inventory Service using a protocol like gRPC.
- Order Service writes to a shared database which Inventory Service polls periodically.
- Order Service publishes an event (like "OrderPlaced") to a message queue, and Inventory Service subscribes to these events.

Question - 35
Client Side Load Balancing

A streaming service giant uses microservices architecture in its infrastructure. They want to provide users with a seamless experience so they started load balancing their servers on the client side (frontend). Whenever a server is busy, the client automatically connects to other servers. There is one security flaw: all the server locations and tokens are stored on the client side. How can this problem be mitigated?

- Each service will need to authenticate the requests individually and cannot trust other services.
- Add more layers of authentication and authorization in between the servers/clients
- Add mTLS certificates that expire after 10 minutes.

B and C

Question - 36
Transactions

A payment gateway company that integrates PayPal, Stripe, and many other vendors uses microservices for their infrastructure. Whenever a user clicks on the payment button, a request to the backend is made and money is instantly deducted. Whenever a user presses the button twice, the charges are deducted twice. How can you fix this problem?

- Disable the button after it is pressed one time.
- Disable the transaction from the backend once it is approved one time.
- Add an idempotence key to the request and add the transaction to the queue.
- A and C
- A and B

Question - 37
Distributed System Characteristics

Which of the following are advantages of a distributed system?

- If one component fails in a distributed system, the remaining components may be able to continue operating.
- It is less difficult to implement a distributed database system because of its low cost of installation.
- The amount of processing overhead is less than with a monolithic architecture.
- It overcomes bottlenecks of the processing pipeline easier than with a centralized system.