**Springboot:**

**1)How would we handle inter communication in a microservice architecture using springboot?**

**For synchronous or direct communication, I would use feign client, which allows services to send requests and receive responses like a two way conversation. Feign client simplifies declaring and making webservice clients, making the code cleaner and the process more efficient.**

**For asynchronous communication, where immediate response are not necessary, I would use message brokers like RabbitMq or kafka. These act like community boards, where services post messages that other services can read and act upon later. This approach ensures a robust, flexible communication system between microservices.**

**2)can you explain the caching mechanism available in springboot and how to implement caching?**

**There is a spring cache abstraction in springboot and it is like a smart memory layer for our application. It is designed to save time and resources by remembering the results of expensive operations.**

**To implement caching in a springboot application, first add caching dependency , like spring-boot- starter-cache.**

**Then, enable caching in application in the applications by adding @EnableCaching annotation in main class.**

**Define cacheble operations using @Cacheble on methods whose results we want to cache, optionally customize cache behaviour with annotations like @CacheEvict and @Cacheput.**

**Choose a cache provider (like EH cache or redis etc.,) or use default concurrent map-based cache provided by spring.**

**3)If our springboot app is experiencing performance issues under high load. What are the steps we need need take to identify and address the performance issues?**

**First we need to identify the performance issues using monitoring tools like springboot actuator or splunk.**

**We can also analyze the app logs and metrics to spot any patterns or errors, especially under high load.**

**Then we need to start performance tests to replicate the issue and use a profiler for code level analysis.**

**After getting findings, we may optimize the database, implement caching, or use scaling options.**

**4)what are the best practices for versioning REST APIs in springboot application?**

**5) How does springboot simplify the data access layer?**

**Autoconfigures essential settings like data source and JPA/Hibernate based on the libraries present in the classpath, reducing manual setup. It also support built-in repository support, such as Jpa repository and enabling easy crud operations.**

**6)what are conditional annotations and purpose of conditional annotations?**

**Conditional annotations in springboot helps us to create beans or configurations only if certain conditions are met. It is like setting rules “ if condition is true then do this” A common example is @ConditionalOnClass, which creates a bean only if a specific class is present.**

**This makes our app flexible and adaptable to different environments without changing the code, enhancing its modularity and efficiency.**

**6)Explain role of @EnableAutoconfiguration?**

**The primary role of @EnableAutoConfiguration is to enable Spring boot’s auto configuration mechanism. This means that spring boot will automatically configure the application based on the jars/dependencies present in the classpath.**

**Explain role of @Configuration?**

**In spring , The @Configuration annotation is used to mark a class as a configuration class. Configuration classes are used to define bean configurations and other spring related configurations. When spring detects the @Configuration on a class, it treats that class as a source of bean definitions and other configuration metadata.**

**7)springboot actuator endpoints?**

**Springboot actuator helps for monitoring and managing our application. It gives us endpoints where we can check health, view configurations, gather metrics, and more. It helps us to keep an eye on how our application doing.**

**Note:**

**In a production environment (which is like real world where our app is being used by people). These endpoints can reveal sensitive information about your application. So now we don’t want that someone seeing into the internals of our application.**

**So how can we secure the actuator endpoints?**

**Limit Exposure : By default, not all actuator endpoints are exposed. We can control which ones are available over the web.**

**Use spring security: we can configure spring security to require authentication for accessing actuator endpoints.**

**Actuator Role : create a specific role, like ACTUATOR\_ADMIN, and assign it to users who should have access.**

**Common actuator end points:**

**/actuator: Provides a list of all available actuator endpoints.**

**/actuator/health:**

**/actuator/info:**

**/actuator/metrics: Exposes various metrics collected by the application, such as memory usage, garbage collection statistics, and more**

**/actuator/loggers: shows and modifies the logging level of the application at runtime.**

**/actuator/env: Displays the current environment properties including system properties, environment variables, and configuration properties.**

**/actuator/mappings: Display a list of all @RequestMapping paths and their corresponding handlers.**

**/actuator/scheduledtasks: Displays information about scheduled task in the application.**

**/actuator/httptrace: Shows HTTP request and response traces. This is useful for understand HTTP traffic to and from our application.**

**/actuator/cache:provides details about available cache.**

**Enabling and securing end points:**

**By default, most actuator endpoints are disabled or secured. We can enable or configure security for these endpoints in application.properties or application.yml file.**

**To enable all endpoints, we can set this property in properties file.**

**management.endpoints.web.exposure.include=\***

**To enable specific endpoints, we can set properties in application.properties.**

**management.endpoints.web.exposure.include=health,info,metrics**

**8)what strategies would you use to optimize the performance of a springboot application?**

**If our application is taking too long to respond to user requests, I could**

**->implement caching for frequently accessed data.**

**->optimize database queries to reduce the load on the database.**

**-> use asynchronous methods for operations like sending emails**

**-> load balancer if traffic is high.**

**9)How to handle multiple beans of same type?**

**To handle multiple beans of same type in spring, we use @Qualifier annotation. This lets us to specify which bean to inject when there are multiple.**

**For example, if there are two beans of type datasource, we can give each a name and use @Qualifier(“bean name”) to tell spring which one to use.**

**Another way is to use @Primary annotation on one of the beans, marking it as the default choice when injecting that type.**

**10)best practices for managing transactions in springboot application?**

**1) Use @Transactional:**

**@Transactional annotation in spring boot is an annotation that we put on methods or classes. It tells springboot, “Hey , please handle this as a single transaction”.**

**Put @Transactional on service methods where we perform database operations. If anything goes wrong with this method, springboot automatically roll back the changes to avoid partial updates.**

**2)Keep transactions on the service layer:**

**It’s usually best to handle transactions in this service layer of our application. The service layer is where we perform business logic.**

**Why here?**

**This is the place where we can access different parts our application(likedata access and business logic).**

**11)How do we approach testing in springboot application?**

**In springboot we have some great tools for this , including @SpringBootTest and @MockBean.**

**->Unit testing: this is like testing each part of application(or methods) individually.**

**->Integration testing: Integration testing means how different components interacts with each other and with spring context.**

**12)Use of @SpringBootTest and @MockBean anootations?**

**@SpringBootTest annotation used for integration testing in springBoot.” It says startup the spring context when this test runs”.**

**Use @SpringBootTest annotation when we need to test how different parts of our application work together. Use this when we need full behaviour of our application.**

**@MockBean is used to create mock(a fake) version of a component or service. This is useful when we want to test a part of our application without involving its dependencies.**

**Use @MockBean in tests where we need to isolate the component being tested. For example, if we are testing a service that depends on a repository, we can mock the repository to control how it behaves and tests the service.**

**13) what advantages does YAML offers over properties files?**

**YAML supports hierarchial configuration, which are more readable and easier to manage, especially for complex structures. And YAML is less familiar to developers compared to the straight forward key-value format of properties files.**

**14)How springboot profile works?**

**Profiles in springBoot allows us to separate different parts of our application configuaration and makes it available only in certain environments. For example we might have one set of settings for development, another for testing, and yet another for production.**

**By using profiles helps to keep our application flexible and maintainable. We can easily switch environments without changing our code.**

**15)what is aspect-oriented-programming in springboot framework?**

**AOP is a programming approach that helps in separating concerns in your application, especially those cut across multiple parts of an application.**

**Our main program code focuses on the core functionality while the “aspects” take care of other common tasks that need to happen in various places, like logging, security checks or managing transactions.**

**For example, In an application, we might have methods where we want to log information every time they’re called or checks that a user has the right permission. Instead of putting this logic in every method, we can define it once in an “aspect” and then specify where and when this code should be applied across our application. This keeps our main code cleaner and more focused on its primary tasks.**

**16)what is spring cloud and how it is useful for building microservices?**

**17)Describe a springboot project where you significantly improved performance. What techniques did you use?**

**18)How does the spring boot make DI easier compared to traditional spring?**

**By providing auto-configuring beans and reducing the need for explit configuration, In traditional spring, we had to define the beans and their dependencies in XML files or with annotations, which is complex for large applications.**

**But in springboot we use auto-configuration and component scanning to automatically discover and register beans based on the application context and class path. This means now we don’t have to manually wire Up the beans.**

**Springboot figures out whats needed and configures it for us. This auto-configuration feature simplifies application setup and development.**

**19)How does springboot simplify the management of application secrets and sensitive configurations, especially when deployed in different environments?**

**SpringBoot helps manage application secrets by allowing configurations to be externalized and kept separate from the code.**

**This means we can use properties files , YAML files, environment variables and command-line arguments to adjust settings for different environments like development, testing and production.**

**For sensitive data, spring boot can integrate with systems like spring cloud config server, which securely stores and provides access to secrets. This setup simplifies managing sensitive configurations without hard coding them, enhancing security and flexibility across various deployment environments.**

**20)Explain Springboot’s approach to handling asynchronous operations?**

**Springboot uses @Async annotation to handle asynchronous operations. This lets us run tasks in the background without waiting for them to be complete before moving to the next line of code.**

**To make a method asynchronous, we just add @Async above its definition, and spring takes care of running it in a separate thread. This is handy for operations that are independent and can be run parallel, like sending emails or processing files, so the main flow of the application doesn’t get blocked.**

**To work with async operations, we also need to enable it in the configuration by adding @EnableAsync to one of the configuration classes.**

**21)How can we enable and use asynchronous methods in a springboot application?**

**To enable and use asynchronous methods in springboot applications.**

**First, I would add the @EnableAsync annotation to one of my configuration classes. This enables spring’s asynchronous method execution capability.**

**Next, we need to mark methods we want to run asynchronously with @Async annotation. These methods can return void or a future type if we want to track the result.**

**Finally, We can call these methods like any other methods. Spring takes care of running them in separate threads, allowing the calling thread to proceed without waiting for the task to finish.**

**Note:**

**For the @Async annotation to be effective, the method calls must be made from outside the class. If we call an asynchronous method from within the same class, it won’t execute asynchronously due to the way spring proxying work.**

**22)Describe how you would secure sensitive data in a springboot application that is accessed by multiple users with different roles?**

**`20:30**

**How to create endpoint in a springboot app that allows users to upload files. And how can you handle the file upload ?**

**To handle file uploads in springboot applications. We can use @PostMapping annotation to create an endpoint that listens for post requests.**

**Then we need to add a method that accepts MultiPartFile as a parameter in the controller. This method would handle the incoming file**

**23)difference between authentication and authourization in spring security?**

**Authentication is verifying who I am, it checks our identity using methods like passwords or tokens.**

**Authourization decides what I am allowed to do after I am identified.**

**Authentication is about confirming my identity, and authourization is about my access rights based on that identity.**

**24)describe how would you send a welcome email to the registered users?**

**First we need to add the springboot-starter mail dependency is in our project’s pom.xml**

**In application.properties we need to setup our mail server details like host, port, username and password.**

**Then we need to write a service class that uses JavaMailSender to send emails. In this service, I craft the welcome email content and use the send method to dispatch emils.**

**Finally, after user successful registration, we need to call our mail service from within the registration logic to send the welcome email.**

**25)How is spring security implemented in a spring boot application?**

**First we need to add spring security dependency in pom.xml file.**

**Thenc, we need to create configuration class extending webSecurityConfigurer adapter to customize security settings, such as specifying secured endpoints and configure the login and logout process, we also implement the UserDetails service interface to load user information, usually from a database, and use a password encoder like BcryptpasswordEncoder for secure password storage.**

**We can secure specific endpoints using annotations like @PreAuthourize, based on the roles or permissions. This setup ensures that our springboot application is secure, managing both authentication and authourization.**

**26)difference between cache eviction and cache expiration?**

**Cache eviction is when data removed from the cache to free up space, based on a policy like least recently used”.**

**Cache expiration is when data is removed because it’s too old based on a predetermined time to live.**

**Note: Eviction manages cache size, while expiration ensures data freshness.**

**27) If we had to scale a springboot application to handle high traffic, what strategies do we need to use?**

**To scale a springboot app for high traffic, we can**

**Add more app instances (horizontal scaling) and use a load balancer to spread out the traffic.**

**Break our app into microservices so each part can be scaled independently.**

**Use cloud services that can automatically adjust resources based on our app’s needs.**

**Use caching to store frequently accessed data, reducing the need to fetch it from the database every time.**

**Implement an Api gateway to handle requests and take care of things like authentication.**

**28)How to implement security in a microservice architecture using springboot and spring security?**

**To secure micro services with springboot and spring security:**

**Add spring security to each microservice for authentication and authourization.**

**Create a central authentication service that gives out tokens(JWT token) when a user logs in.**

**Ensure each microservice checks these tokens to let only allowed users in.**

**Implement an API gateway to manage security checks and route requests.**

**29)How the session management configured and handled in distributed systems?**

**In spring boot for distributed systems, session management is done by storing session information in a shared location using spring session.**

**This way, any server can access the session data, allowing users to stay logged in across different servers.**

**We set it up by adding spring session to our project and choosing where to store the sessions, like in a database or cache.**

**This makes our app more scalable and keeps user sessions consistent.**

**30)Imagine you are designing a spring boot app that interfaces with multiple external APIs. How would you handle API rate limits and failures?**

**To handle API rate limits and failures in a springboot application:**

**Use a circuit breaker to manage failures**

**Implement rate limiting to avoid exceeding API limits.**

**Add a retry mechanism with exponential behalf for temporary issues.**

**Use caching to reduce number of requests.**

**31)How would you manage externalized configuration and secure sensitive configuration properties in a microservices architecture?**

**32)can we create a non- web app in springboot?**

**Yes, we can make a non-web application with springboot. Springboot is not just for web projects. We can use it for other types like running scripts or processing data.**

**If we don’t addd web parts to our projects, it won’t start a web server. Instead.**

**33)what does the @SpringBootApplication do internally?**

**@SpringBootApplication is a combination of 3 other anootations.**

**@Configuration: telling spring that this class has configurations and beans that spring should manage.**

**@EnableAutoConfiguration: which allows spring boot to automatically setup the application based on the libraries on the class path.**

**@ComponentScan: Which tells spring to look for other components, configurations and services in the current package, allowing it to find and register them.**

**34)How odes spring boot support internationalization?**

**Spring boot supports internationalization by showing our application’s text in different languages by using property files.**

**We put these in a folder named src/main/resources. Each file has a name like messages\_xx.properties , where xx stands for the language code. Spring boot uses these files to pick the right language based on the users settings. We can set rules on how to choose the user’s language with something called localeResolver.**

**This way app speak to users in their language, making it more user-friendly for people from different parts of the world.**

**35)What is springboot devtools?**

**Spring boot devtools makes developing application faster and easier. It automatically restarts our application when we change code, so we can see updates immediately without restarting manually.**

**36)How do you mock microservices during testing?**

**To mock microservices during tests, I use tools like wiremock or Mockito to pretend I am talking to real microservices.**

**With these tools I setup fake responses to our requests, so if my app asks for something from another service, the tool steps in and gives what should I told it to, just like if the real service had answered.**

**37)Explain the process of creating a docker image for a spring boot application?**

**To make a Docker image for a spring boot app, we start by writing a docker file. This files tells docker how to build our app’s image.**

**We maintain which java version to use, add our app’s jar file, and specify how to run our app.**

**After writing the dockerfile, we run a command docker build-t myapp:latest. In the terminal.**

**This command tells docker to create the image with everything our app needs to run by doing this, we can easily run our spring boot app anywhere docker is available, making our app portable and easy to deploy.**

**38)Discuss the configuration of spring security to address common security concerns?**

**39)How would you secure a springboot application using JSON web token(JWT)?**

**40)How can springboot applications be made more-resilient to failures, especially in microservices architectures?**

**41)How can spring cloud gateway be configured for routing, security and monitoring?**

**For routing, we define routes in the application.properties or through java config, specifying paths and deatinations for incoming requests.**

**For security we integrate spring security to add authentication, authourization and protection against common threats.**

**To enable monitoring, we use spring actuator, which provides buili-in endpoints for monitoring and managing the gateway.**

**This setup allows us to control how requests are handled, secure the gateway, and keep an eye on its performance and health, all within the spring ecosystem.**

**42)our application needs to process notifications asynchronously using a message queue, Explain how would you setup the integration and send messages from springboot app?**

**To manage and monitor asynchronous tasks in a springboot app. I would use the @Async annotation to run tasks in the background and completable Future to track their progress and handling results or failures for thread management. I would configure a threadTaskExecutor to customize thread settings.**

**43) we need to secure a springboot app to ensure that only authenticated users can access certain endpoints?**

**First I will add spring security dependency , Then I would configure wensecurityConfigureAdapter to customize security settings.**

**In this configuration we need use http.authourizeRequest() method to specify which endpoint require authentication.**

**44)How to tell an autoconfiguration to back away when a bean exists?**

**In springboot, to make an auto configuration step back when a bean already exists, we use the @ConditionalOnMissingBean annotation. This tells springboot to only create a bean if it doesn’t already exist in the context.**

**45)How to deploy springboot web app as jar and war files?**

**To deploy springboot web applications, we can package them as either JAR orWAR files. For aJAR file we use springboot’s embedded server like tomcat, by running the command mvn package and then java – jar target/myApplication.jar.**

**If we need a war file for deployment on an external server, we change the packaging in pom.xml to <packaging>war</packaging>, ensure the application extends springboot servlet initializer and then build with mvn package, The WAR file can be deployed to any java servlet container, like tomcat or jetty.**

**Note:**

**When deploying a springboot app on a tomcat server within the context of a cloud services like AWS,The tomcat server acts as a application server handling the execution of your springboot app.**

* **Our springboot application is a java based web application that we have developed. It contains logic, endpoints, and other functionalities.**
* **Tomcat is web server and servlet container. In the context of deploying a springboot application, Tomcat serves as the container for our application. It receives the requests and handles the communication between your app and outside world.**
* **When we deploy our SB app to tomcat, we are essentially packaging it as a WAR(Web application archive) file. This file contains our application code,resources and configuration.**
* **Aws provides the infrastructure where our Tomcat server is hosted.**
* **Tomcat processes the incoming request and forwards it to the appropriate servlet, which in this case is our springboot application.**
* **Then springboot application executes the logic on the received request.**
* **Once our SB has processed the request, It generates a response, and Tomcat handles sending that response back to user.**

**Tomcat server is a intermediatory between our SB and the outside world, managing the deployment and execution of our application.**

**46)What does it mean springboot provides relaxed binding?**

**Springboot’s relaxed binding it’s flexible in how properties are defined in configuration files. This flexibility allows us to use various formats for property names.**

**For example, if we have a property named server. Port, we can write it in different ways server.port, server-port or SERVER\_PORT.**

**47)Discuss the integration of Spring boot app with CI/CD pipelines?**

**Integration spring boot app with CI/CD pipelines means making the process building, testing and deploying automated.**

**When we make changes to our code and push them, the pipeline automatically builds the app, run tests and if everything looks good, deploys it. This uses tools like Jenkins or GitHub Actions to automate tasks, such as compiling the code and checking for errors.**

**If all tests pass, the app can be automatically sent to a test environment or directly to users. This setup helps us quickly find and fix errors, improve the quality of our app and make updates faster without manual steps.**

**48)How to resolve whitelabel error page in springboot application?**

**To fix white label error pages in , we need to check our URLs are correctly mapped in the controllers. If a URL doesn’t match any controller, spring boot shows this error page.**

**We should add or update our mappings to cover the URLs we are using. Also we can create custom error pages or use @ControllerAdvice to handle errors globally. This way instead of default page, visitors can see custom message when something goes wrong.**

**49)How can we implement pagination in springboot application?**

**To implement pagination in a spring boot application, I use springDataJpa’s pageable interface. In the repository layer, I modify my query methods to accept a pageable object as a parameter, when calling these methods from my service layer,I create an instance of page request, specifying the page number and page size I want.**

**This page request is then passed to the repository method. springData Jpa handles the pagination logic automatically, returning a page object that contains the requested page of data along with useful information like total pages and total elements. This approach allows us to efficiently manage large datasets by retrieving only a subset of data at a time.**

**50)How to handle 404 error in springboot?**

**To handle 404 error in springboot, we make a custom controller. We implement the ErrorController interface and mark it with@Controller.**

**Then we create a method that returns error page or message for 404 errors, and we map this method to the /error URL using @RequestMapping.**

**In this way we can check the error type and customize what users see when they hit a page that doesn’t exist.**

**51)How to springboot to be used to implement event-driven architecture?**

**52)Basic annotations that springboot offers?**

**@SpringbootApplication is a combination of @Configuration @EnableAutoConfiguration and @ComponentScan**

**@RestController and @RequestMapping are essential for creating RESTful webservices, allowing us to define controller classes and map URL paths to methods.**

**@Service and @Repository annotations mark service and data access layer, respectively , promoting separation of concerns @Autowired enables dependency injection, automatically wiring beans. These annotations are crucial in reducing boiler plate code, speeding up development and maintaining clear architecture.**

**53)Integration and use of distributed tracing in springboot applications for monitoring and troubleshooting?  
we can achieve distributed tracing in springboot by using sleuth and zipkin. When a request travels through microservices, these tools assign and propagate unique IDs for request, creating detailed traces of its journey. This makes easier to understand the flow, pin point delays and identify errors in complex, distributed environment.**

**54)our application needs to store and retrieve files from a cloud storage service. Describe how you would integrate this functionality into a springboot app?**

**To integrate cloud storage in a springboot application, we need to use a cloud SDk, like AWS SDK for s3 or google cloud storage libraries, depending on the cloud provider.**

**First we need to add dependency in pom.xml. Then we need to configure the necessary credentials and settings in application properties, for accessing cloud storage.**

**Then we need to create a service class to encapsulate the storage operations-uploading and deleting files. By autowiring this service where needed. We can interact with cloud storage seamlessly.**

**55)To protect your application from abuse and ensure fair usage , you decide to implement rate limiting on your API endpoints. Describe a simple approach to achieve this in springboot.**

**A simple approach is like Bucket4j or spring cloud gateway with built-in-rate-limiting capabilities. By integrating one of these libraries , we can define policies directly on API end points to limit the number requests a user can make in a given time frame.**

**This involves configuring few annotations or settings in application.properties to specify rate limits.**

**56)why we use global exception handling with @ControllerAdvice?**

**@Controller advice can be used handle exceptions globally across all controllers. This means we can define common exception handling logic in one place instead of repeating it in each controller.**

**We can define a method to handle exceptions using @ExceptionHandler annotation inside a class defined with @ControllerAdvice**

@ControllerAdvice public class GlobalExceptionHandler {

@ExceptionHandler(Exception.class)

public ModelAndView handleException(Exception ex) {

ModelAndView modelAndView = new ModelAndView("error"); modelAndView.addObject("message", ex.getMessage()); return modelAndView; }

57) How to control the order of execution of beans in springboot?

Suppose, to control the order of execution of beans in spring configuration class , we can use the **@order** annotation or the **ordered** interface These annotations and interfaces help us specify the order in which beans should be initialized or the priority of execution .

**@Configuration**

**public class AppConfig {**

**@Bean**

**@Order(2)**

**public FirstBean firstBean() {**

**return new FirstBean();**

**}**

**@Bean**

**@Order(1)**

**public SecondBean secondBean() {**

**return new SecondBean();**

**}**

**}**

**20-21 -22-29-30-31-34-36-37-39-49-51-54-55**