1. All annotations used in Spring Boot applications and purpose of each:

Annotations is a form of metadata that provides data about a program.

**@EnableAutoConfiguration:** It auto-configures the bean that is present in the classpath and configures it to run the methods

**@ComponentScan:** The process of discovering Spring Components within classpath of the application is called Component Scanning in Spring. The classes annotated with @Controller, @Configuration, @RestController etc will be treated as Spring Components.

**@Configuration:** It is used as a source of bean definitions. Indicates that a class declares one or more @Bean methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

**@SpringBootApplication:** It is a combination of three annotations @EnableAutoConfiguration, @ComponentScan, and @Configuration.

1. **Core Java 8 features**

**Functional interface:** interface having only one abstract method, can have other default and static methods. Lambda can be used instead of anonymous inner class, return type no need to specify, as type inference is done by compiler, just pass arguments if required [return type of arguments also not required], name of method no need, access modifier no need as not a method or class). Since there is only one abstract function in the functional interfaces, there is no confusion.

**Lambda expressions** are means to create anonymous classes of functional interfaces. Lambda Expression are the way through which we can visualize functional programming in the java object oriented world, it provide support for using lambda expressions only with functional interfaces

**Streams**: Collection interface has been extended with *stream()* and *parallelStream()* default methods to get the Stream for sequential and parallel execution to perform filter/map/reduce like operations on different collection types, by which high performance can be achieved on large collections.

**Method reference**: Java 8 method references are the shortened versions of lambda expressions calling a specific method.

**Date/time**: There was no standard approach or API in java for date and time in Java, to streamline the process of working with time in java.

1. **@RestController vs @Controller**:

The @RestController annotation in Spring MVC/Spring BOOT is nothing but a combination of @Controller and @ResponseBody annotation.

1. **CRUD operations**: Create, Read, Update, Delete
2. **Http Methods & Status Code**:

a. POST Create 201 (Created)

b. GET Read 200 (OK)

c. PUT Update/Replace 405 (Method Not Allowed)

d. PATCH Update/Modify 405 (Method Not Allowed)

e. DELETE Delete 405 (Method Not Allowed)

1. **@CrossOrigin:**

Cross-Origin Resource Sharing (CORS) is a security concept that allows restricting the resources implemented in web browsers. It prevents the JavaScript code producing or consuming the requests against different origin.

For example, In use case development web application(Angular) is running on 4200 port and Spring Boot microservices are running on port 8080, 8081, 8082. To access Spring Boot’s backend endpoints from Angular application @CrossOrigin should be used in Spring Boot Controller or Main class in Spring Boot Application.

1. **@ControllerAdvice:** It is similar to interceptor/filter**.** It pre-process requests to controller. Post-process response to handle exceptions. Used for global exception handling.
2. **Spring Core Annotations:**

**@Component**: It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with @Component is found during the classpath. The Spring Framework pick it up and configure it in the application context as a Spring Bean.

**@Bean:** Spring @Bean Annotation is applied on a method to specify that it returns a bean to be managed by Spring context. Spring Bean annotation is usually declared in Configuration classes methods. The @Bean annotations are used at the method level and indicate that a method produces a bean that is to be managed by Spring container. It is an alternative to the XML<bean> tag.

**@Service:** It is used at the class level. It shows that the annotated class is a service class, such as business basic logic, and call external APIs.

**@Repository:** It is a Data Access Object (DAO) that accesses the database directly. It indicates that the annotated class is a repository. The repository does all the operations related to the database.

1. **JPARepository VS CrudRepository:** JpaRepository extends PagingAndSortingRepository which in turn extends CrudRepository. Crud repository is base interface and extends repository interface.
2. @Query:

1. **MongoTemplate:**
2. MongoRepository:
3. @Valid for validations

Mention it is used in use case dev

1. Advantages of Microservice architecture
2. Purpose of Zuul/Spring Cloud Gateway,
3. Eureka Service Registry/Discovery related annotations, configurations
4. Does Microservices interact with each other, RestTemplate