### Spring Framework  
1. What are the features of the Spring Framework?  
2. Explain the concept of Dependency Injection.  
3. What is Inversion of Control (IoC)?  
4. Explain the different types of Spring Bean scopes.  
5. How do you configure Spring Beans?  
6. What are the different types of Spring container?  
7. Explain the difference between constructor injection and setter injection.  
8. What is the role of the ApplicationContext interface?  
9. What are the benefits of using Spring Framework?  
10. How can you integrate Spring with other frameworks?  
11. What is the Spring Bean lifecycle?  
12. Explain the use of the `@Autowired` annotation.  
13. What is Spring Aspect-Oriented Programming (AOP)?  
14. How do you handle transactions in Spring?  
15. Explain the concept of Spring Profiles.  
16. What is Spring Expression Language (SpEL)?  
17. How do you enable annotation-based configuration in Spring?  
18. What is a Spring Boot Starter?  
19. How do you configure a datasource in Spring?  
20. What is Spring Boot Actuator?  
  
### Spring Boot  
1. What is Spring Boot and how is it different from Spring Framework?  
2. Explain the concept of auto-configuration in Spring Boot.  
3. What are Spring Boot starters?  
4. How do you create a Spring Boot application?  
5. What is the Spring Boot CLI?  
6. How do you configure application properties in Spring Boot?  
7. What are the advantages of using Spring Boot?  
8. Explain the use of the `@SpringBootApplication` annotation.  
9. What is Spring Boot DevTools?  
10. How do you monitor a Spring Boot application?  
11. What is Spring Boot Actuator?  
12. How do you deploy a Spring Boot application?  
13. Explain the concept of embedded servers in Spring Boot.  
14. How do you handle exceptions in Spring Boot?  
15. What is a Spring Boot initializer?  
16. How do you create a RESTful web service using Spring Boot?  
17. What is Spring Boot Security?  
18. How do you configure logging in Spring Boot?  
19. Explain the concept of Spring Boot Profiles.  
20. How do you test a Spring Boot application?  
  
### Spring MVC  
1. What is Spring MVC and how does it work?  
2. Explain the concept of DispatcherServlet.  
3. What are the main components of the Spring MVC architecture?  
4. How do you configure a Spring MVC application?  
5. What is a Controller in Spring MVC?  
6. Explain the use of the `@Controller` annotation.  
7. What is a View Resolver in Spring MVC?  
8. How do you handle form submissions in Spring MVC?  
9. What is the role of the Model in Spring MVC?  
10. How do you handle exceptions in Spring MVC?  
11. What is Spring MVC Interceptor?  
12. How do you validate form inputs in Spring MVC?  
13. What is the role of the `@RequestMapping` annotation?  
14. How do you configure static resources in Spring MVC?  
15. Explain the concept of Spring MVC Themes.  
16. How do you upload a file in Spring MVC?  
17. What is Spring MVC Portlet?  
18. How do you integrate Spring MVC with Thymeleaf?  
19. What is the use of the `@RestController` annotation?  
20. How do you handle JSON responses in Spring MVC?  
  
### Spring REST  
1. What is REST and how is it different from SOAP?  
2. How do you create a RESTful web service using Spring?  
3. What is the role of the `@RestController` annotation?  
4. How do you handle HTTP methods in Spring REST?  
5. What is the use of the `@RequestMapping` annotation?  
6. How do you handle path variables in Spring REST?  
7. What is the use of the `@PathVariable` annotation?  
8. How do you handle request parameters in Spring REST?  
9. What is the use of the `@RequestParam` annotation?  
10. How do you handle JSON responses in Spring REST?  
11. How do you handle exceptions in Spring REST?  
12. What is the role of the `@ResponseBody` annotation?  
13. How do you handle HTTP status codes in Spring REST?  
14. How do you configure CORS in Spring REST?  
15. What is the use of the `@RequestBody` annotation?  
16. How do you secure a Spring REST API?  
17. What is the role of the `@CrossOrigin` annotation?  
18. How do you handle file uploads in Spring REST?  
19. How do you test a Spring REST API?  
20. What is HATEOAS and how is it implemented in Spring REST?  
  
### Hibernate  
1. What is Hibernate and what are its advantages?  
2. What is ORM and how does Hibernate implement it?  
3. What is the role of the SessionFactory in Hibernate?  
4. What are the main components of Hibernate architecture?  
5. How do you configure Hibernate in a Java application?  
6. What is the use of the Hibernate configuration file?  
7. How do you map a Java class to a database table in Hibernate?  
8. What is the use of the `@Entity` annotation in Hibernate?  
9. How do you define a primary key in Hibernate?  
10. What is the role of the `@Id` annotation?  
11. How do you perform CRUD operations in Hibernate?  
12. What is HQL and how is it different from SQL?  
13. How do you handle transactions in Hibernate?  
14. What is the use of the `@Transactional` annotation in Spring?  
15. How do you configure caching in Hibernate?  
16. What is the difference between first-level and second-level cache in Hibernate?  
17. How do you handle lazy loading in Hibernate?  
18. What is the use of the `@OneToMany` annotation?  
19. How do you handle relationships in Hibernate?  
20. How do you integrate Hibernate with Spring?  
  
### Spring Data JPA  
1. What is Spring Data JPA and what are its advantages?  
2. How do you configure Spring Data JPA in a Spring Boot application?  
3. What is the role of the `JpaRepository` interface?  
4. How do you define a repository in Spring Data JPA?  
5. What is the use of the `@Repository` annotation?  
6. How do you perform CRUD operations using Spring Data JPA?  
7. How do you define custom queries in Spring Data JPA?  
8. What is the use of the `@Query` annotation?  
9. How do you handle pagination in Spring Data JPA?  
10. What is the use of the `Pageable` interface?  
11. How do you handle sorting in Spring Data JPA?  
12. What is the use of the `Sort` class?  
13. How do you configure auditing in Spring Data JPA?  
14. What is the use of the `@CreatedDate` and `@LastModifiedDate` annotations?  
15. How do you handle transactions in Spring Data JPA?  
16. What is the use of the `@Transactional` annotation?  
17. How do you configure caching in Spring Data JPA?  
18. How do you handle relationships in Spring Data JPA?  
19. What is the use of the `@OneToOne`, `@OneToMany`, `@ManyToOne`, and `@ManyToMany` annotations?  
20. How do you test a Spring Data JPA application?  
  
### Microservices  
1. What are microservices and what are their advantages?  
2. How do you design a microservices architecture?  
3. What is the role of the API Gateway in microservices?  
4. How do you handle communication between microservices?  
5. What are the different types of communication in microservices?  
6. How do you implement service discovery in microservices?  
7. What is the role of Eureka in Spring Cloud?  
8. How do you configure load balancing in microservices?  
9. What is the role of Ribbon in Spring Cloud?  
10. How do you handle configuration in microservices?  
11. What is the role of Spring Cloud Config?  
12. How do you handle security in microservices?  
13. What is the role of OAuth2 and JWT in microservices?  
14. How do you handle transactions in microservices?  
15. What is the role of Sagas in microservices?  
16. How do you implement resilience in microservices?  
17. What is the role of Hystrix in Spring Cloud?  
18. How do you handle logging and monitoring in microservices?  
19. What is the role of ELK Stack in microservices?  
20. How do you test microservices?  
  
### Spring WebFlux  
1. What is Spring WebFlux and how is it different from Spring MVC?  
2. What are the main components of Spring WebFlux?  
3. How do you configure a Spring WebFlux application?  
4. What is the role of the `@Controller` annotation in Spring WebFlux?  
5. How do you handle requests in Spring WebFlux?  
6. What is the use of the `@GetMapping`, `@PostMapping`, `@PutMapping`, and `@DeleteMapping` annotations?  
7. How do you handle responses in Spring WebFlux?  
8. What is the use of the `ServerResponse` class?  
9. How do you handle exceptions in Spring WebFlux?  
10. What is the role of the `@ExceptionHandler` annotation?  
11. How do you handle form submissions in Spring WebFlux?  
12. What is the use of the `@RequestBody` annotation?  
13. How do you handle path variables in Spring WebFlux?  
14. What is the use of the `@PathVariable` annotation?  
15. How do you handle request parameters in Spring WebFlux?  
16. What is the use of the `@RequestParam` annotation?  
17. How do you configure CORS in Spring WebFlux?  
18. How do you secure a Spring WebFlux application?  
  
  
19. How do you test a Spring WebFlux application?  
20. What is the role of Reactive Streams in Spring WebFlux?  
  
### Angular  
1. What is Angular and what are its key features?  
2. How do you create a new Angular project?  
3. What is the role of the Angular CLI?  
4. How do you structure an Angular project?  
5. What is the role of components in Angular?  
6. How do you create a component in Angular?  
7. What is the role of the `@Component` decorator?  
8. How do you handle data binding in Angular?  
9. What are the different types of data binding in Angular?  
10. How do you handle forms in Angular?  
11. What is the role of the `@NgModel` directive?  
12. How do you handle routing in Angular?  
13. What is the role of the `@RouterModule`?  
14. How do you handle HTTP requests in Angular?  
15. What is the role of the `@HttpClientModule`?  
16. How do you create a service in Angular?  
17. What is the role of the `@Injectable` decorator?  
18. How do you handle dependency injection in Angular?  
19. How do you handle directives in Angular?  
20. What are the different types of directives in Angular?  
  
### React JS  
1. What is React JS and what are its key features?  
2. How do you create a new React project?  
3. What is the role of the Create React App?  
4. How do you structure a React project?  
5. What is the role of components in React?  
6. How do you create a component in React?  
7. What is the difference between a class component and a functional component?  
8. How do you handle state in React?  
9. What is the role of the `useState` hook?  
10. How do you handle props in React?  
11. What is the difference between state and props in React?  
12. How do you handle events in React?  
13. How do you handle forms in React?  
14. What is the role of the `useEffect` hook?  
15. How do you handle routing in React?  
16. What is the role of React Router?  
17. How do you handle HTTP requests in React?  
18. How do you handle context in React?  
19. What is the role of the `useContext` hook?  
20. How do you test a React application?  
  
### Unit Testing (JUnit, Mockito)  
1. What is JUnit and what are its key features?  
2. How do you create a test case in JUnit?  
3. What is the role of the `@Test` annotation in JUnit?  
4. How do you handle assertions in JUnit?  
5. What is the role of the `assertEquals` method?  
6. How do you handle exceptions in JUnit?  
7. What is the role of the `@Before` and `@After` annotations in JUnit?  
8. How do you create parameterized tests in JUnit?  
9. What is the role of the `@ParameterizedTest` annotation?  
10. How do you create test suites in JUnit?  
11. What is Mockito and what are its key features?  
12. How do you create a mock object in Mockito?  
13. What is the role of the `@Mock` annotation in Mockito?  
14. How do you handle stubbing in Mockito?  
15. What is the role of the `when` method in Mockito?  
16. How do you verify interactions in Mockito?  
17. What is the role of the `verify` method in Mockito?  
18. How do you handle argument matchers in Mockito?  
19. What is the role of the `any` method in Mockito?  
20. How do you integrate JUnit and Mockito in a Spring Boot application?

Certainly! Here are code-based interview questions for each of the specified topics:  
  
### Spring Framework  
  
1. \*\*Dependency Injection (DI):\*\*  
   - Implement a simple example demonstrating constructor injection in Spring.  
  
2. \*\*Bean Configuration:\*\*  
   - Write a Spring configuration XML to define a bean named `userService` of class `UserService`.  
  
3. \*\*Aspect-Oriented Programming (AOP):\*\*  
   - Implement an Aspect in Spring AOP to log method entry and exit points for a service class.  
  
4. \*\*Spring Bean Lifecycle:\*\*  
   - Write a Spring bean with a custom initialization method and destroy method.  
  
5. \*\*Handling Transactions:\*\*  
   - Implement a service method annotated with `@Transactional` that performs multiple database updates.  
  
6. \*\*Spring Profiles:\*\*  
   - Create two different profiles ("dev" and "prod") in Spring Boot and configure different data sources for each.  
  
7. \*\*Spring Expression Language (SpEL):\*\*  
   - Use SpEL to dynamically set a property value in a Spring Bean configuration.  
  
8. \*\*Spring Boot Starter:\*\*  
   - Create a custom Spring Boot starter that auto-configures a datasource and adds it to the application context.  
  
9. \*\*Spring Boot Actuator:\*\*  
   - Configure and use Spring Boot Actuator to monitor health and metrics of a Spring Boot application.  
  
10. \*\*Exception Handling in Spring MVC:\*\*  
    - Implement a global exception handler using `@ControllerAdvice` to handle exceptions thrown by controllers.  
  
### Spring Boot  
  
1. \*\*Creating a Spring Boot Application:\*\*  
   - Create a basic Spring Boot application that exposes a REST endpoint to retrieve a list of users.  
  
2. \*\*Auto-Configuration in Spring Boot:\*\*  
   - Use `@EnableAutoConfiguration` and `@ComponentScan` to enable auto-configuration in a Spring Boot application.  
  
3. \*\*Spring Boot CLI:\*\*  
   - Write a Groovy script using Spring Boot CLI to bootstrap a new Spring Boot project.  
  
4. \*\*Spring Boot Actuator:\*\*  
   - Configure custom health indicators and expose them using Spring Boot Actuator.  
  
5. \*\*Testing a Spring Boot Application:\*\*  
   - Write unit tests for a Spring Boot REST controller using `@WebMvcTest` and Mockito.  
  
6. \*\*Embedded Servers in Spring Boot:\*\*  
   - Configure an embedded Tomcat server in a Spring Boot application and customize its properties.  
  
7. \*\*Handling Exceptions in Spring Boot:\*\*  
   - Implement a custom error controller to handle specific HTTP error codes in a Spring Boot application.  
  
8. \*\*Spring Boot Security:\*\*  
   - Configure basic authentication for a Spring Boot application using Spring Security.  
  
9. \*\*Spring Boot Profiles:\*\*  
   - Use Spring Boot profiles to load different properties files based on the environment (dev, prod).  
  
10. \*\*Logging Configuration in Spring Boot:\*\*  
    - Configure logback or log4j2 for logging in a Spring Boot application and customize log levels.  
  
### Spring MVC  
  
1. \*\*Creating a Spring MVC Controller:\*\*  
   - Write a Spring MVC controller to handle HTTP GET requests for "/users" and return a list of users.  
  
2. \*\*Form Handling in Spring MVC:\*\*  
   - Implement a form in a Spring MVC application to collect user details and save them to a database.  
  
3. \*\*View Resolution in Spring MVC:\*\*  
   - Configure InternalResourceViewResolver to resolve JSP views in a Spring MVC application.  
  
4. \*\*Handling File Uploads in Spring MVC:\*\*  
   - Implement a controller method to handle file uploads using `@RequestParam` and `MultipartFile`.  
  
5. \*\*Spring MVC Interceptor:\*\*  
   - Create a custom interceptor in Spring MVC to log request processing time for all incoming requests.  
  
6. \*\*Validation in Spring MVC:\*\*  
   - Implement server-side form validation using `@Valid` and `BindingResult` in a Spring MVC controller.  
  
7. \*\*Handling AJAX Requests in Spring MVC:\*\*  
   - Write a controller method to handle AJAX requests and return JSON response using `@ResponseBody`.  
  
8. \*\*Spring MVC REST Integration:\*\*  
   - Expose a RESTful API using Spring MVC to perform CRUD operations on a resource (e.g., `/api/users`).  
  
9. \*\*Exception Handling in Spring MVC:\*\*  
   - Implement a `@ControllerAdvice` class to handle exceptions thrown by controllers and return appropriate error responses.  
  
10. \*\*Spring MVC Testing:\*\*  
    - Write integration tests for a Spring MVC controller using `MockMvc` and validate response status and content.  
  
### Spring REST  
  
1. \*\*Creating a RESTful Web Service with Spring:\*\*  
   - Implement a REST controller in Spring to expose CRUD endpoints for a resource (e.g., `/users`).  
  
2. \*\*Handling JSON Responses in Spring REST:\*\*  
   - Write a controller method to return a JSON response using `@ResponseBody` and a custom DTO.  
  
3. \*\*Exception Handling in Spring REST:\*\*  
   - Implement a global exception handler for a Spring REST API using `@ControllerAdvice` and `@ExceptionHandler`.  
  
4. \*\*RESTful Web Service Security:\*\*  
   - Secure a Spring REST API using Spring Security with JWT authentication.  
  
5. \*\*Versioning RESTful APIs in Spring:\*\*  
   - Implement API versioning in Spring REST using URI versioning or header versioning.  
  
6. \*\*HATEOAS in Spring REST:\*\*  
   - Enhance a Spring REST API to include HATEOAS links for related resources.  
  
7. \*\*Handling Pagination and Sorting in Spring REST:\*\*  
   - Implement pagination and sorting for a Spring REST endpoint using `Pageable` and `Sort` parameters.  
  
8. \*\*Content Negotiation in Spring REST:\*\*  
   - Configure Spring to support content negotiation for JSON and XML responses in a RESTful API.  
  
9. \*\*Spring REST Integration Testing:\*\*  
   - Write integration tests for a Spring REST API using `TestRestTemplate` and verify CRUD operations.  
  
10. \*\*CORS Configuration in Spring REST:\*\*  
    - Configure Cross-Origin Resource Sharing (CORS) for a Spring REST API to allow requests from specific origins.  
  
### Hibernate  
  
1. \*\*Mapping Java Class to Database Table:\*\*  
   - Define a Hibernate entity class and map it to an existing database table using annotations (`@Entity`, `@Table`, etc.).  
  
2. \*\*CRUD Operations with Hibernate:\*\*  
   - Implement DAO layer methods using Hibernate to perform CRUD operations on a database entity (e.g., `findById`, `save`, `delete`).  
  
3. \*\*Hibernate Query Language (HQL):\*\*  
   - Write an HQL query to fetch records based on a specific criteria (e.g., fetching users with a given role).  
  
4. \*\*Hibernate Criteria API:\*\*  
   - Use Hibernate Criteria API to dynamically build queries with filtering and sorting criteria.  
  
5. \*\*Hibernate Transaction Management:\*\*  
   - Implement transaction management using Hibernate `Session` and `Transaction` APIs (`session.beginTransaction()`, `transaction.commit()`).  
  
6. \*\*Hibernate Lazy Loading:\*\*  
   - Configure lazy loading for entity associations in Hibernate and explain its benefits.  
  
7. \*\*Hibernate Caching:\*\*  
   - Configure second-level caching in Hibernate for improved performance and explain its working principles.  
  
8. \*\*Mapping Relationships in Hibernate:\*\*  
   - Implement one-to-many or many-to-many relationships between Hibernate entities using annotations (`@OneToMany`, `@ManyToMany`).  
  
9. \*\*Auditing with Hibernate Envers:\*\*  
   - Configure Hibernate Envers for auditing entity changes (e.g., `@Audited`, `@RevisionEntity`).  
  
10. \*\*Hibernate Native SQL Queries:\*\*  
    - Write a native SQL query in Hibernate to fetch data from multiple tables or perform complex joins.  
  
### Spring Data JPA  
  
1. \*\*Creating a Spring Data JPA Repository:\*\*  
   - Define a repository interface extending `JpaRepository` to perform CRUD operations on an entity.  
  
2. \*\*Custom Queries with Spring Data JPA:\*\*  
   - Write a custom query method in a Spring Data JPA repository interface to fetch users by a specific attribute.  
  
3. \*\*Pagination and Sorting with Spring Data JPA:\*\*  
   - Implement pagination and sorting for a repository query method using `Pageable` and `Sort` parameters.  
  
4. \*\*Auditing with Spring Data JPA:\*\*  
   - Enable auditing in Spring Data JPA to automatically populate `createdBy`, `createdDate`, `lastModifiedBy`, and `lastModifiedDate` fields.  
  
5. \*\*Query DSL with Spring Data JPA:\*\*  
   - Use Querydsl with Spring Data JPA to write type-safe queries for complex search criteria.  
  
6. \*\*Transactional Operations with Spring Data JPA:\*\*  
   - Use `@Transactional` annotation to manage transactions for methods in a Spring Data JPA repository interface.  
  
7. \*\*Spring Data JPA Projections:\*\*  
   - Define a projection interface in Spring Data JPA to fetch selected attributes of an entity.  
  
8. \*\*Batch Processing with Spring Data JPA:\*\*  
   - Implement batch processing using Spring Data JPA to perform bulk updates or deletes.  
  
9. \*\*Using Specifications in Spring Data JPA:\*\*  
   - Implement dynamic queries using Specifications in Spring Data JPA to apply complex predicates.  
  
10. \*\*Testing Spring Data JPA Repositories:\*\*  
    - Write unit tests for Spring Data JPA repositories using in-memory database or mock data.  
  
### Microservices  
  
1. \*\*Creating a Microservice with Spring Boot:\*\*  
   - Implement a microservice using Spring Boot to expose REST endpoints for user management.  
  
2. \*\*Service Discovery with Eureka:\*\*  
   - Configure Eureka server and client in a Spring Boot microservice architecture for service registration and discovery.  
  
3. \*\*Implementing Circuit Breaker Pattern with Hystrix:\*\*  
   - Integrate Hystrix with Spring Cloud to implement circuit breaker pattern for fault tolerance in microservices.  
  
4. \*\*API Gateway with Spring Cloud Gateway:\*\*  
   - Configure Spring Cloud Gateway as an API Gateway to route requests to different microservices based on paths.  
  
5. \*\*Event-Driven Communication with Spring Cloud Stream:\*\*  
   - Implement event-driven microservices communication using Spring Cloud Stream and Kafka or RabbitMQ.  
  
6  
  
. \*\*Distributed Tracing with Spring Cloud Sleuth:\*\*  
   - Configure Spring Cloud Sleuth for distributed tracing across microservices to track request flows.  
  
7. \*\*Centralized Configuration with Spring Cloud Config:\*\*  
   - Implement centralized configuration management using Spring Cloud Config Server for microservices.  
  
8. \*\*Securing Microservices with OAuth2 and JWT:\*\*  
   - Secure microservices communication using OAuth2 with JWT tokens for authentication and authorization.  
  
9. \*\*Deploying Microservices with Docker and Kubernetes:\*\*  
   - Containerize microservices with Docker and deploy to Kubernetes cluster for orchestration and scaling.  
  
10. \*\*Monitoring Microservices with Spring Boot Admin:\*\*  
    - Use Spring Boot Admin to monitor and manage microservices health and metrics in a centralized dashboard.  
  
### Spring Web Flux  
  
1. \*\*Creating a Reactive Web Application with Spring Web Flux:\*\*  
   - Implement a reactive RESTful service using Spring Web Flux to handle asynchronous requests.  
  
2. \*\*Functional Endpoints in Spring Web Flux:\*\*  
   - Define functional endpoints using Spring Web Flux `RouterFunction` and `HandlerFunction`.  
  
3. \*\*Handling Streams in Spring Web Flux:\*\*  
   - Implement a WebSocket endpoint in Spring Web Flux to stream real-time data to clients.  
  
4. \*\*Reactive Data Access with Spring Data R2DBC:\*\*  
   - Integrate Spring Data R2DBC with Spring Web Flux to perform reactive database operations.  
  
5. \*\*Error Handling in Spring Web Flux:\*\*  
   - Implement global error handling for exceptions thrown in Spring Web Flux controllers or handlers.  
  
6. \*\*Testing Reactive Applications in Spring Web Flux:\*\*  
   - Write integration tests for reactive controllers in Spring Web Flux using `WebTestClient`.  
  
7. \*\*Reactive Security with Spring Security Web Flux:\*\*  
   - Secure a reactive web application using Spring Security with reactive authentication and authorization.  
  
8. \*\*Reactive Stream Processing in Spring Web Flux:\*\*  
   - Implement reactive stream processing using operators like `map`, `flatMap`, and `zip` in Spring Web Flux.  
  
9. \*\*Reactive File Upload and Download in Spring Web Flux:\*\*  
   - Implement endpoints for reactive file upload and download using Spring Web Flux and `DataBuffer`.  
  
10. \*\*Integration with Reactive Messaging Systems:\*\*  
    - Integrate Spring Web Flux with reactive messaging systems like Apache Kafka or RabbitMQ for event-driven communication.  
  
### Angular  
  
1. \*\*Creating Components in Angular:\*\*  
   - Create an Angular component for displaying a list of users fetched from a REST API.  
  
2. \*\*Routing in Angular:\*\*  
   - Configure Angular Router to navigate between multiple views (components) based on URL paths.  
  
3. \*\*Forms and Form Validation in Angular:\*\*  
   - Implement a reactive form in Angular with validation rules for input fields.  
  
4. \*\*HTTP Client in Angular:\*\*  
   - Use Angular `HttpClientModule` to make GET, POST, PUT, and DELETE requests to a RESTful API.  
  
5. \*\*Services in Angular:\*\*  
   - Create an Angular service to encapsulate data access logic and share data between components.  
  
6. \*\*Angular Directives:\*\*  
   - Implement a custom directive in Angular for form validation or DOM manipulation.  
  
7. \*\*Component Communication in Angular:\*\*  
   - Implement communication between parent and child components using `@Input` and `@Output` decorators.  
  
8. \*\*Handling Observables in Angular:\*\*  
   - Use RxJS operators (`map`, `filter`, `mergeMap`, etc.) to manipulate data streams in Angular services.  
  
9. \*\*Angular Pipes:\*\*  
   - Create a custom pipe in Angular to format dates or filter data displayed in a component template.  
  
10. \*\*Angular Testing with Jasmine and Karma:\*\*  
    - Write unit tests for Angular components and services using Jasmine test framework and Karma test runner.  
  
### React JS  
  
1. \*\*Creating Components in React:\*\*  
   - Create a functional component in React to display a list of users fetched from a REST API.  
  
2. \*\*State Management in React:\*\*  
   - Implement state management using `useState` hook to handle component-level state changes.  
  
3. \*\*Handling Events in React:\*\*  
   - Write event handlers in React to handle user interactions like button clicks or form submissions.  
  
4. \*\*Forms and Form Validation in React:\*\*  
   - Implement a controlled form in React with validation using state and event handlers.  
  
5. \*\*Component Lifecycle Methods in React:\*\*  
   - Explain the lifecycle methods (`componentDidMount`, `componentDidUpdate`, `componentWillUnmount`) in React class components.  
  
6. \*\*React Router:\*\*  
   - Configure React Router to implement client-side routing and navigate between different views (components).  
  
7. \*\*Handling HTTP Requests in React:\*\*  
   - Use `fetch` API or `axios` library to make HTTP requests to a RESTful API and update component state.  
  
8. \*\*Context API in React:\*\*  
   - Use React Context API to share state across multiple components without prop drilling.  
  
9. \*\*React Hooks:\*\*  
   - Explain the `useState`, `useEffect`, and `useContext` hooks and demonstrate their usage in React functional components.  
  
10. \*\*Testing React Components with Jest and Enzyme:\*\*  
    - Write unit tests for React components using Jest testing framework and Enzyme testing utility.  
  
### Unit Testing (JUnit, Mockito)  
  
1. \*\*Writing a JUnit Test Case:\*\*  
   - Write a JUnit test case to validate a simple method that performs addition or subtraction.  
  
2. \*\*Parameterized Tests in JUnit:\*\*  
   - Implement parameterized tests in JUnit to run the same test method with different inputs.  
  
3. \*\*Mocking with Mockito:\*\*  
   - Use Mockito to mock dependencies (e.g., DAO layer) in a JUnit test for a service layer method.  
  
4. \*\*Verifying Interactions in Mockito:\*\*  
   - Write a Mockito test to verify that a method of a mocked object has been called with specific arguments.  
  
5. \*\*Stubbing with Mockito:\*\*  
   - Stub a method call using Mockito to return a predefined value or throw an exception during testing.  
  
6. \*\*Exception Handling in JUnit:\*\*  
   - Write a JUnit test case to verify that a method correctly throws an expected exception.  
  
7. \*\*Testing Spring Boot Applications with JUnit:\*\*  
   - Write an integration test using JUnit to test a RESTful API endpoint in a Spring Boot application.  
  
8. \*\*Testing Asynchronous Code with JUnit:\*\*  
   - Write a JUnit test to verify the behavior of asynchronous methods using `CompletableFuture` or `@Async`.  
  
9. \*\*Integration Testing with Spring Boot and Mockito:\*\*  
   - Write an integration test using Mockito to test a service layer method in a Spring Boot application context.  
  
10. \*\*Mockito Annotations:\*\*  
    - Explain the usage of `@Mock`, `@InjectMocks`, and `@Spy` annotations in Mockito tests and demonstrate their usage.