Tell me about your project.

1.Explain how a User will navigate through your website.

2.Explain what responsibilities an Admin has in your website.

3.Start with the technical explanation of backend:

* We have developed a REST API using Spring Boot framework.
* Authentication and authorization process.
* JWT token generation.
* Spring Security filter chain, which allows access to the endpoints with specific roles only.
* Layered architecture of our REST API: Controller layer, Service Layer and Repository Layer.
* Our repository layer makes use of Spring Data JPA.
* Centralized exception handling.
* Dedicated error page.

4.Explian the technologies used in frontend:

* ReactJS
* Axios

Why ReactJS?

* ReactJs uses SPA model.
* Faster rendering because of Virtual DOM and component-based architecture.
* Its unopinionated behavior.

What is Axios? Why Axios?

Axios is an HTTP client library that allows you to make requests to a given endpoint. It is a better alternative to the Fetch API because it works well with JSON objects. For example, if we use Fetch API, we have to convert our request body to a JSON string.

What is a REST API/Why REST API?

An API (application programming interface), is a set of rules that define how applications/devices can connect and communicate with each other. Whereas a REST API is an API that follows the design principles of the REST (representational state transfer) architectural style.

Design principals of REST:

**Uniform interface**: All API requests for the same resource should look the same, no matter where the request comes from. The REST API should ensure that the same piece of data, such as the name or email address of a user, belongs to only one uniform resource identifier (URI).

**Client-server decoupling**: In REST API design, client and server applications must be completely independent of each other. The only information the client application should know is the URI of the requested resource.

**Statelessness**: REST APIs are stateless, meaning that each request needs to include all the information necessary for processing it. In other words, REST APIs do not require any server-side sessions.

**Layered system architecture**: In REST APIs, the calls and responses go through different layers.

Explain what is a JWT token. How did you manage session in your website?

JWT token is generated at the backend by our REST API once the user has been authenticated. A JWT token consists of 3 sections, header, payload and its signature.

The payload consists of all the necessary information required to identify the user (to maintain the session). This JWT token can be stored in session storage or in cookies at the frontend.

The JWT token is appended with the request header for every request made by client for that website, post authentication.

The authenticity of a JWT token is checked using the Secure Key stored at our REST API.

Session Storage vs Local Storage vs Cookies

Both session storage and local storage are APIs of JavaScript and are used to store data in the form of key-value pairs. Whereas cookies are used to store JSON objects.

**Session storage** objects are stored only for one session, the data is deleted once the browser is closed. Session storage is unique to each browser tab. If you open a new tab and navigate to the same site, you'll start a new session with its own storage.

**Local storage** maintains data across all instances of a site, whether they're in different tabs or windows. It is also permanent storage, so the data won't disappear when you close your browser. Local storage has no expiration date, the only way to clear local storage is to explicitly delete it.

Annotations used in out project

1. @SpringBootApplication = @Configuration+@EnableAutoConfiguration+@ComponentScan
2. @Component
3. @Bean
4. @Autowired
5. @Controller = Used in Spring MVC applications
6. @RestController
7. @PathVariable
8. @RequestBody
9. @RequestMapping
10. @GetMapping, @PostMapping, @PostMapping, etc
11. @Repository
12. @Service
13. @Transactional

Features of Spring Boot/Why Spring Boot?

* It is easy to integrate a Spring Boot application with its Spring Ecosystem, which includes Spring Security, Spring Data JPA, etc.
* Reduces the efforts of writing all the boilerplate codes and xml configurations.
* Provides a lot of plugins for the development and testing of Spring Boot applications. Example Swagger API, Password Encoder API, etc.

Inversion of Control and Dependency Injection

**Dependency injection** is a design pattern in which an object or function receives other objects or functions that it depends on. A form of **Inversion of Control**, dependency injection aims to separate the concerns of constructing objects and using them, leading to loosely coupled programs.

Types of DI: Constructor based, Setter based and Field/Property based.

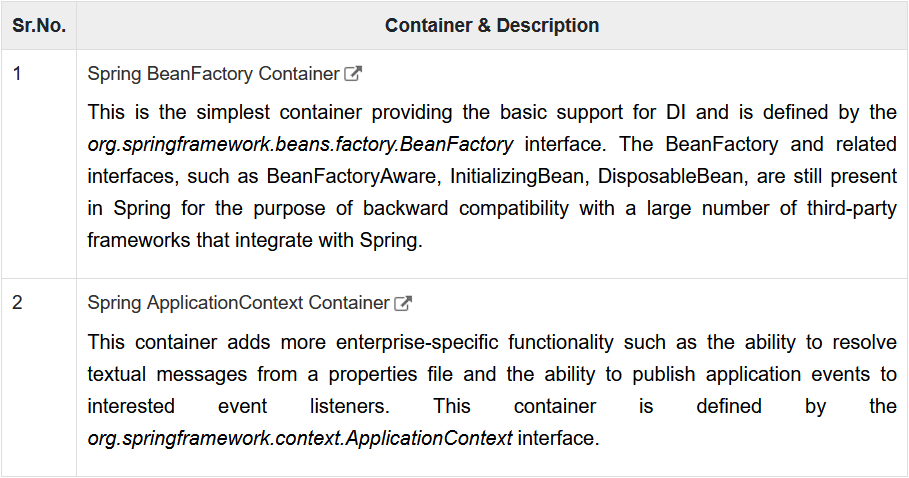
**IoC Container:**

The Spring container is at the core of the Spring Framework. It provides us with ready to use objects. These objects are called beans.

This container will create the objects, hold them in memory, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components.

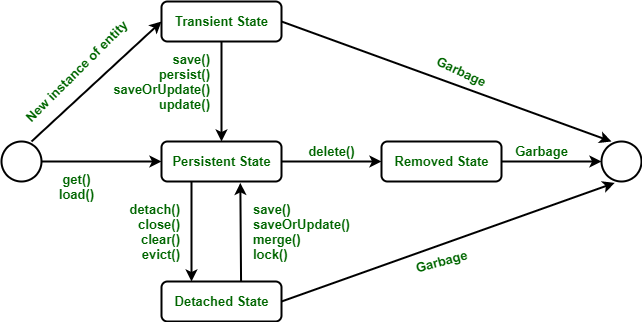
These containers get their configuration instructions from metadata provided by XML file or Java annotations. It contains information like which bean depends on which another bean.

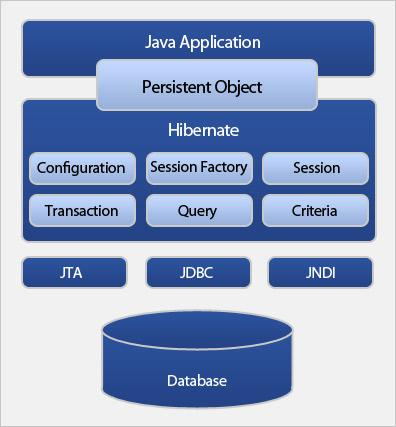
There are two types of containers: BeanFactory and ApplicationContext Container.



Regarding Hibernate

Hibernate is an implementor of JPA. It is a java based ORM tool.





Spring Data JPA

Spring Data JPA is not a JPA provider. It is a library/framework that adds an extra layer of abstraction on the top of our JPA provider (like Hibernate).

One advantage of using Spring Data JPA is, we can control transaction boundaries declaratively using the @Transactional annotation. Hence it reduces the boilerplate code.