What is Spring Boot - 1 multiple choice

autoconfiguration

REST API's = short answer (How REST API's work) = 5 points (It's okay Flow) client to server

-Provide interface for systems to talk to each other

-We use it to communicate with internal and external parties in a consistent and predictable way

-Client sends a request to the server

-Server processes the request

-Server sends a response to the client

-Client processes the response

Endpoints = GET, POST, PUT and DELETE

How Endpoints Works = 5 points

-Function that are executed when a specific URL is called

-HTTP methods can be used on any endpoint which map to CRUD (GET, POST, PUT and DELETE)

-HTTP headers an information such as authentication tokens or cookies can be contained in the HTTP request header

-Body Data is normally transmitted in the HTTP body in an identical way to HTML <form> submissions or by sending a single JSON-encoded data string

200 OK is used for successful request, 201 Created could also be return when record is created

400 Bad Request, 401 Unauthorized, 404 Not Found

Explain how controller and how it works (Explain)= 5 points - Rest Services and talks about Controller it's fine

-Send data to model for further processing

-Get processed data from Model

-create to handle the various requests we would like to allow our RESTful API to respond to

About Spring Annotations (PostMapping, PutMapping and GetMapping). RequestBody and ResponseMapping, ResponseBody

Multiple Choice about ResponseEntity (HTTP Response (Status Code, headers and body)

OAuth 2 = 5 points = potential short answer and multiple choice

- Allow limited access to user accounts on an HTTP service
- User authentication is delegated to the host of the user's account
- Third-party applications redirect users to the delegated authentication
- Allows third-party applications to access that user's account

Roles = multiple choice (Which is not a role)

JSON Web Tokens

Java Web Token

JWT Token Structure (Header, Payload, Signature)

POM - used by Maven to build a project

Dependencies = what dependencies we use

- spring-boot-starter-data-jpa, mysql-connector-java, spring-boot-starter-validation
- spring-boot-starter-security, spring-security-oauth2-autoconfigure

JJWT

```
   -<groupId>io.jsonwebtoken</groupId>
           <artifactId>jjwt</artifactId>
```

XML Bind

```
    -<groupId>javax.xml.bind</groupId>
           <artifactId>jaxb-api</artifactId>
```

Spring Security Filters = 5 points = Short answer (Explain), how it works,

- create your own filters
- The request is sequential filter
- Each filter is checked
- Once all the filters pass validation, the request is then passed to the servlet
- AccountEntity (getPassword,getUsername, isAccountNonExpired)

Beans - set of rules

JWT Filter - Used to retrieve the user's authentication and authorization information

What is the core interface for spring security- userdetails, class

JWT Utility Class - Generate Token (ID, Email),setSubject, setIssuer(Who created it, From)

setIssuedAt, SetExpiration, signWith

Object Relation Mapping (ORM) = Deleting data (CRUD)

JPA - multiple choice = data persistence

application.properties = spring.datasource(rul, username and password)

DB Annotations = 5 points = short answer (5pts) explain each

-Entity = Defines that a class can be mapped to a table

-Table = Defines how to name the table the entity is mapped to

-ID = Used to have a primary key

-GeneratedValue = generated AUTO, TABLE, SEQUENCE and IDENTITY

The Repositoy Pattern = possible question (What repository we use? CrudReposito, PagingOrSortingRepos, JPAReposi)

Extending the JPA Repository = 5 points = if short question about it

- Extending to jparepository to gain all function
- provide type of the key and provide data class)

-The code that must be implemented is simplified

- Mainly used for managing the data

-We get a bunch of generic CRUD methods into our type that allows saving, deleting and so on

-Extending the interface

-Allows the Spring Data JPA repository infrastructure to scan for this interface and create a Spring bean for it

Finding Records = multiple choice = derived queries (findBy)

Many Options = By and find