**1 [10 minutes]**

In this course we looked at different ways how applications can communicate over HTTP:

1. REST

2. Webflux

3. Websockets

4. GraphQL

In 1 to 3 sentences describe for each of these communication styles for what kind of applications this communication style is the best choice. In other words, for what type of applications is it better to use REST, for what type of applications is it better to use Webflux, etc.

**2 [15 minutes]**

a. Explain the disadvantage of stateful security and why we need stateless security.

b. What are the 3 parts of a JWT token?

c. JWT makes use of a secret (= a secret phrase). If we use JWT to secure our REST endpoints, where is this secret stored? On the client or on the server?

d. Give 3 different techniques that you can use to minimize the problem of stolen JWT tokens.

**3 [25 minutes]**

Suppose you need to write a simple product application using **Spring Boot** that can be called through a **REST**interface.   
A product has the attributes productNumber (String), name (String), category (String)

The application should support the following API calls:  
1.    Add a product  
2.    Get all products from a certain category

The product application stores products in a **MongoDB**database.

The application should perform the following **validation**:  
•    All attributes should have a value and cannot be empty  
•    The productNumber should be a String between 4 and 6 characters

a.    Write the **names** of **ALL classes necessary** to implement this application according the best practices we learned in this course. For every class, in one short sentence describe the responsibility of this class. Important for this question is that you write the **names of ALL necessary classes**.  
b.    Write the Java code of the **ProductController**including required annotations. Do not write the imports.  
c.    Write the Java code of the **ProductRepository**. Do not write the imports.

**4 [10 minutes]**

Suppose you need to write a simple product application using **Spring Boot** that can be called through a **GraphQL interface**.   
A product has the attributes productNumber (String), name (String), category (String)

The application should support the following API calls:  
1.    Add a product  
2.    Get all products  
3.    Get all products from a certain category

Write the **graphQL schema** for this application

**5 [10 minutes]**

a.    Write an example **URL** where a **request parameter** is used  
b.    Write the **Java code (including annotation)** of a Spring  Boot controller method that is called by the URL of part a. Show how we get the value of the **request parameter** .  
c.    Write an example **URL** where a **path variable** is used  
d.    Write the**Java code (including annotation)** of a Spring  Boot controller method that is called by the URL of part c. Show how we get the value of the **path variable** .

**6 [20 minutes]**

Suppose you need to develop a REST interface for the functionality of the shopping cart in a webshop. The REST API needs to implement the following functionality:

* **getShoppingCart(cartNumber)**  *//returns the content of the shopping cart*
* **addProductToShoppingCart(cartNumber, productNumber, quantity)**  *//adds a product to the shopping cart*
* **removeProductFromShoppingCart(cartNumber, productNumber, quantity )**  *//removes a product from the shopping cart*
* **clearShoppingCart(cartNumber)**  *//removes all products from the shopping cart*
* **checkoutShoppingCart(cartNumber)** *//transforms the shopping cart into an order*

Write the **signature of all methods** in the shoppingCartController, including the **necessary annotations.**  
A signature of a method contains the following content:   
***return-type method-name (parameter-type parameter-name, …)***

An example method signature is with corresponging annotation is:

*@Getmapping(...)*

*Customer getCustomer(int customerNumber)*

Do **NOT**write the body of these methods.  
Follow the API best practices we have learned in this course.

For this question it is important that you show for each method the **mapping annotation** and the **signature** of every method.

**7 [10 minutes]**

When we studied websockets we saw that it is fairly easy to write a chat application with websockets. Below is the skeleton code of a TextWebSocketHandler that we need to write if we use websockets.

**Write the missing code** of the SocketTextHandler so that this class implements a simple chat application. Only write the body of the 3 methods. Do not rewrite all given code, this takes too much time. Only write the missing code pieces, and with a comment indicate where this piece of code should be written.

**@Component  
public class SocketTextHandler extends TextWebSocketHandler {**

**@Override  
    public void handleTextMessage(WebSocketSession session, TextMessage message)  
            throws Exception {  
        // body of the method**

**}**

**@Override  
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {  
        // body of the method**

**}**

**@Override  
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {  
        // body of the method**

**}**

**}**

**8 [15 minutes]**

Suppose we have a SpringMVC web application (that shows webpages in the browser) where the customersPage.html shows a list of customers. The application allows the user to add new customers to the list. The controller method to handle the functionality to add a new customer to the list looks like this:      
**@PostMapping("customers")  
    public ModelAndView addCustomer(@ModelAttribute("customer") Customer customer) {  
          
          customerService.addCustomer(customer);  
          
          Map<String, Object> params = new HashMap<>();  
          params.put("customerlist", customerService.getCustomers());  
          return new ModelAndView("customersPage", params);      
    }**

Assume that we have a customerService class and the application works correctly.

a.    The given controller method has a problem related to POST requests, and should not be implement like shown above. What is the problem with the given controller method?

b.    Rewrite the given code so that this problem does not exist anymore. Write all necessary controller methods that replace the given method so that the application works correctly again.

**9 [10 minutes]**

Describe how we can relate the concept of **Model-View-Controller** to the principles of Science & Technology of Consciousness (SCT). Your answer should be about **2 to 3 paragraphs**, but should not exceed one page (handwritten). The number of points you get for this question depend on how well you explain the relationship between **Model-View-Controller** and the principles of SCT.