C4 evaluation (Spring boot): SB-201

Note
All the questions are compulsory.
*The duration of the test is **3 hours**.*
Don't seek help from any person/resource during the test.

Marks Distribution is as follows:

Question	*Marks*
1	*6*
2	*4*

Q2. Consider the following Entity class:

Employee:

empId: Integer
empName: String
salary: Integer
address: String
email: String
mobile: String
password: String

- Generate the Rest API for the following Services of the EmployeeService interface using **Spring Data JPA**:

```
public interface EmployeeService{

   public Employee registerEmployee(Employee emp)throws EmployeeException;

    public Employee getEmployeeById(Integer empId)throws EmployeeException;

    public List<Employee> getAllEmployeeDetails()throws EmployeeException;

    public Employee deleteEmployeeById(Integer empId)throws EmployeeException;

    public Employee loginEmployee(String email, String password)throws
    Employee Exception;
```

public List<Employee> getEmployeeDetailsByAddress(String address)throws EmployeeException;

public Employee updateEmployee(Employee emp)throws EmployeeException;

public String getNameAndAddressOfEmplyeeById(Integer empId)throws EmployeeException

public List<EmployeeDTO> getNameAddressSalaryOfAllEmployee()throws EmployeeException;

}

class EmployeeDTO:
name:String
address:String
salary: Integer

- **Apply the Validation of the User Request Data.**
- **Handle all the Exceptions in proper ExcpetionHandler.**
- **EmployeeException should be the checked exception**

Q2/- Consider the following Entity class:

Account:
accountNo: Integer
accountHolderName: String
address: Sring
email: String
balance: Integer
mobile: String

- Generate the Rest API for the following **AccountService** interface using **Spring Data JPA.**

  public interface AccountService{

  public Account openAccount(Account acc)throws AccountException;

```
public Account closeAccount(Integer accno)throws AccountException;

public Account depositAmount(Integer accno, Integer amount)throws
AccountException;
//after deposit amount return the updated Account object.


public Account withdrawAmount(int accno, Integer amount)throws
AccountException, InsufficientFundException;

//after withdraw amount returns the updated Account object.
//if the insufficient amount is there throw the InsufficientFundException with proper message
//if Invalid accno is there then throw the AccountException with proper message

public String transferAmount(AccountDTO dto)throws AccountException,
InsufficientFundException.

//after successful transfer, returns the Sucess message.
//if Invalid srcAccno and desAccno is there then throw the AccountException with proper
//message
//if the insufficient amount is there in destination account throw the InsufficientFundException
//with proper message

}

class  AccountDTO {

        Integer srcAccno;
        Integer desAccno;
        Integer amount;

}
```

- **AccountException and InsufficientFundException should be the checked exception.**
- **Handle all the exceptions in proper ExceptionHandler.**

Sample of application.properties file: modify it accordingly.

```
#changing the server port
server.port=8888

#db specific properties
spring.datasource.url=jdbc:mysql://localhost:3306/sb201db
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root

#ORM s/w specific properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```