1. Create a database in postgres or use h2 in memory database. Create 10 entity tables, where should be One-to-one, One-to-many, many-to-many relationships (join table won’t be counted as entity table). Create a DATABASE UML diagram. Upload your diagram with project as PDF file. **FILE SHOULD BE LOCATED INSIDE YOUR PROJECT FOLDER.**

~~2. Upload database backup file with your project, if you use postgres database.~~ **~~Name your database – {$your\_variant\_{$your\_lastname}}.~~** ~~For example~~ **~~variant3\_Urmanov.tar~~**

**~~spring.datasource.url=jdbc:postgresql://localhost:5432/ variant3\_Urmanov~~**

~~3. Create Readme.MD file in project structure. In this file write your project's idea, functionality that you're going to implement etc. (https://github.com/tchapi/markdown-cheatsheet/blob/master/README.md~~

~~4. Use different type of beans annotations.~~

~~5. Use different type of Dependency Injections. (ONLY CONSTRUCTOR and Setter injection. NO FIELD injection)~~

**6. Write good service logic in service classes.** (If your most port of code will consist only calling repo method, -50% from your grade)

~~7. Use @PropertySource, @Lazy, @Scope, @DependsOn.~~

9. ~~Add at least 2 configuration classes.~~

~~10. Add AOP configuration. Use AspectJ annotation style.~~

11. ~~Use next annotations: @Before, @Pointcut, @After, @Aspect, @AfterReturning, @Around, @AfterThrowing.~~

~~12. Add real service/business logic in AOP code.~~

~~13. For your repository classes, use different and more complex methods/code for JdbcTemplate class.~~ **~~DON’T USE JPA Repositories.~~**

~~14. Use batch operations.~~

~~15. Implement a Custom Converter (org.springframework.core.convert.converter.Converter)~~

~~16. Implement a Custom Formatter (FormattingConversionServiceFactoryBean)~~

~~17. Use AssertTrue for Custom Validation (@AssertTrue(message="ERROR! Individual customer should have gender and last name defined")~~

~~18. Write scheduled method. Use @Scheduled annotations with attributes:~~

~~• fixedDelay~~

~~• fixedRate~~

~~• initialDelay~~

~~19. Parameterizing the Schedule. Parameters should be in application.props file.~~

**~~20. Run Tasks in Parallel.~~**

21. Use all next methods:

~~GET~~

~~POST~~

~~PUT~~

HEAD

~~DELETE~~

~~OPTIONS~~

22. Use next annotations:

~~GetMapping~~

~~PostMapping~~

~~PutMapping~~

~~DeleteMapping~~

~~23. Use RequestBody and ResponseBody Annotations.~~ ~~Read HTTP Headers in Spring REST Controllers.~~

**24. Setting Up Spring openapi**

25. Use Spring @ResponseStatus to Set HTTP Status Code. Use Spring ResponseEntity to Manipulate the HTTP Response

26. Add REST Pagination support

27. Add Upload and Download file methods.

**28. Add JUnit test with at least 80% code coverage.**

**29. Write integration test for controller classes. Integration test should be located in other package:**

30. Write JMS service. 3 methods which send data to topic, 3 methods which listen topic.

**31. Add JUnit test with at least 80% code coverage.**

**32. Use OAuth2 and JWT**

33. **DO NOT USE** in memory authentication (auth.inMemoryAuthentication())

34. Prevent Brute Force Authentication Attempts with Spring Security

35. Control the Session with Spring Security

• always

• ifRequired

• never

• stateless

36. Fix 401s with CORS Preflights and Spring Security

37. Prevent Cross-Site Scripting (XSS) in a Spring Application

38. Add 1-2 pages which supports websocket technology. Example - https://www.baeldung.com/websockets-spring

**39. Write CURL in README.md for your ALL endpoints, or upload in project folder POSTMAN collections.**