ITS 4243 – Microservices and Cloud Computing

Assignment - 01

Index – ICT/21/814

Name – M.S. Sidhdhika Banu

Part 1

1. What is Spring Boot and why is it used?

   Spring Boot is a framework built on top of the Spring Framework that makes it easy to create stand-alone, production ready Spring applications with minimal setup. It removes the need for complex configuration and lets developers start coding quickly.
   It is mainly used because,

   - it supports auto-configuration**,** which automatically sets up the application based on the dependencies you add.
   - It also provides embedded servers like Tomcat and Jetty, so you don't have to install them separately.
   - Spring Boot includes production-ready features such as health checks, metrics, and logging.
   - It allows faster development**,** especially when building REST APIs and microservices, with minimal code.

2. Explain the difference between Spring Framework and Spring Boot.

Spring Framework	**Spring Boot**
The Spring Framework requires a lot of manual configuration using XML or Java-based setup.	Spring Boot provides auto-configuration based on the dependencies added.
In the Spring Framework, an external server like Tomcat or JBoss is needed for	Spring Boot includes an embedded server.

deployment.	
The Spring Framework has more boilerplate code.	Spring Boot reduces it to a minimum.
Spring Framework is a general-purpose framework.	Spring Boot mainly focuses on rapid development and building Microservices.
Spring Framework has a steeper learning curve.	Spring Boot is easier to learn and use.

3. What is Inversion of Control (IoC) and Dependency Injection (DI)?

Inversion of Control (IoC):
Inversion of Control is a design principle where the control of object creation and management is given to the framework instead of the programmer. In Spring, this means the framework creates and manages the objects for you, so you don't have to create them manually.

Dependency Injection (DI):
Dependency Injection is the main way Spring implements IoC. Instead of a class creating its own dependent objects, the framework injects them automatically. This helps in making the code easier to maintain and test.

4. What is the purpose of application.properties / application.yml?

The 'application.properties' or 'application.yml' file is used to store the configuration settings of a Spring Boot application. These files allow developers to define important values such as the database connection details, server port number, logging levels, and other custom application settings.

They help make the application flexible because you can easily change configurations without modifying the source code. Both application.properties and application.yml serve the same purpose, the only difference is in their format**:**

- application.properties uses key-value pairs
- application.yml uses a YAML structure with indentation

5. Explain what a REST API is and list HTTP methods used.

REST (Representational State Transfer) is an architectural style used to build web services that communicate over HTTP. A REST API allows clients, such as Postman or a frontend application, to interact with a server using URLs and HTTP methods. REST APIs usually return data in JSON format**.**

Common HTTP Methods used in REST APIs:

- GET- Retrieves data from the server
- POST- Creates new data on the server
- PUT- Updates existing data completely
- DELETE- Deletes data from the server
- PATCH- Partially updates existing data

6. What is Spring Data JPA? What is an Entity and a Repository?

Spring Data JPA:
Spring Data JPA is a part of Spring that simplifies database access in Spring Boot applications. It allows developers to perform CRUD operations (Create, Read, Update, Delete) on the database without writing SQL queries manually.

Entity:

An Entity is a Java class that is mapped to a database table. Each instance of the class represents a row in the table.

Repository:

A Repository is an interface that provides database operations for an entity. It allows you to save, update, delete, and fetch data easily.

7. What is the difference between @Component, @Service, @Repository, @Controller, @RestController?

- **@Component:** This is a generic Spring annotation used to mark any class as a Spring managed bean. It is for general purpose components that don't fit into a specific layer.

- **@Service:** This annotation is used to mark a class as part of the service/business logic layer. It indicates that the class contains business logic.

- **@Repository:** This marks a class as part of the data access layer. It also helps Spring convert database exceptions into Spring's DataAccessException, making error handling easier.

- **@Controller:** Used in MVC web applications to define controllers that handle web requests and usually return views like HTML pages.

- **@RestController:** This is a combination of @Controller and @ResponseBody. It is used in REST APIs where the controller returns JSON responses instead of views.

8. What is @Autowired? When should we avoid it?

@Autowired is used in Spring to automatically inject dependencies into a class. Spring's IoC container takes care of creating the required object and injecting it wherever needed.

When should we avoid it:

- In constructor injection**,** you can omit @Autowired because Spring Boot 2.6+ automatically injects dependencies through the constructor.
- Avoid using @Autowired excessively on fields because it tightly couples classes and makes testing harder.

9. Explain how Exception Handling works in Spring Boot (@ControllerAdvice).
Spring Boot provides centralized exception handling using the @ControllerAdvice annotation. This allows you to handle exceptions globally for all controllers instead of writing try/catch blocks in every controller method.

Some purpose of using @controllerAdvice:

- Catches exceptions thrown by controllers across the application.
- Returns meaningful error messages and appropriate HTTP status codes.
- Keeps the code clean and maintainable by avoiding repetitive try/catch blocks.

10. What is the role of Maven/Gradle in a Spring Boot project?

Maven and Gradle are build automation tools used to manage:

- Project dependencies
- Compilation
- Packaging
- Testing
- Deployment