Week 1 Theory SOA REST MICROSERVICE

1. **What is SOA? What are the benefits that SOA bring to software development?**

   SOA is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.

   For properties of a service according to SOA design:

   - It logically represents a business activity with a specified outcome.
   - It is self-contained.
   - It is a black box for its consumers.
   - It may consist of other underlying services

   Benefits :

   In general we have many benefits that SOA bring to a product such as:

   - Service case: Improved information flow, Ability to expose internal functionality, Organizational flexibility
   - Service Re-use: Lower software development and management costs
   - Service Discovery: Ability to optimize performance, functionality, and cost, Easier introduction of system upgrades
   - Vituralization:Improved reliability, Ability to scale operations to meet different demand levels

   And many other benefit in many other case of Software Development

2. What is REST architecture? Why does it become more and more popular?

   Representational State Transfer (REST) is a style of architecture based on a set of principles that describe how networked resources are defined and addressed. It has proved to be a popular choice for implementing Web Services.

   Popular Reason:

   In more recent years the REST design pattern emerged which provided new benefits for both the client and server. On the server-side REST scales and performs better than SOAP. With SOAP there is significant overhead with parsing and routing each request to local web service function implementations. While REST leverages standard HTTP/S requests and does not require complex SOAP Envelope repackaging of data. The SOAP overhead is small when looking at a single request, but at scale those numbers add up, creating lower throughput per server.

3. **How software architecture can contribute to software quality?**

   Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams. The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that a design approach will yield an acceptable system. By building

effective architecture, you can identify design risks and mitigate them early in the development process.

4. **What is microservices? How is it different compared to SOA?**
   Microservices is a distinctive method of developing software systems that has grown in popularity in recent years. In fact, even though there isn't a whole lot out there on what it is and how to do it, for many developers it has become a preferred way of creating enterprise applications. Thanks to its scalability, this architectural method is considered particularly ideal when you have to enable support for a range of platforms and devices—spanning web, mobile, Internet of Things, and wearables—or simply when you're not sure what kind of devices you'll need to support in an increasingly cloudy future.

   Different with SOA:
   The core difference between SOA and microservices lies in the size and scope. As the word "micro" suggests, it has to be significantly smaller than what SOA tends to be. Microservice is a smaller independently deployable unit. Beware of very small microservice antipattern - nanoservice. A SOA can be either a monolith or it can be comprised of multiple microservices.

WEEK 2 DESIGN PATTERN

1. **What are benefits of using design patterns?**
   Design pattern help improve in the coding period can help the coder keep the original template of product and also improve or fix the product by their way.

2. **What do you think when from 23 design patterns originally, now there are more than 100 patterns? What is problem with that?**
   In fact, there are many way to design an application, the creator can use their creative to create a product according to 23 design patterns. It's nothing wrong if we explore more patterns. It is great to let every design work with their own opinion and create something interesting for this world

   Problem:
   23 design pattern is covered nearly every that a coder need to use for design something. With me, it is very flexible for now and the pattern which designed from them is from the creative and the need of human condition. So, nothing wrong if every pattern is necessary for human.

3. **Discuss some use cases where use of observer pattern is relevant**
   When we need to backup an information before fixing and restored after fix.

4. **Give some examples where singleton pattern is appropriate**

Use in the simple system that can be cover 1 function of this system. Example In game lottery machine, the singleton have responsibility to cover the function generate number of winer people

WEEK 3 more design pattern

3. What is the design pattern that is related to clone method of Java Object?

Observer

4. In programing language, how compiler evaluate an expression such as 1+3? What design pattern is used?

Singleton

WEEK 4 Spring

1. What is the core design pattern Spring uses?
   Dependency injection and IoC

2. What are benefits of using Spring?

   1) Predefined Templates :Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.
   2) Loose Coupling: The Spring applications are loosely coupled because of dependency injection.

   3) Easy to test: The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

   4) Lightweight: Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

   5) Fast Development: The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

   6) Powerful abstraction: It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

   7) Declarative support: It provides declarative support for caching, validation, transactions and formatting.

3. Can Spring do what Java Enterprise Edition (JEE) can do and more?
   Yes because JEE is a base structure to create Spring. Spring is more advantage stuff in java to let the coder easier interact with hard thing in java as the simple way.

4. How do you understand about enterprise systems? Give some example of enterprise systems that you have used/worked with.

WEEK 5 Hibernate

1. **What are benefits of using ORM?**

- Hibernate supports mapping of java classes to database tables and vice versa. It provides features to perform CRUD operations across all the major relational databases.
- Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business use cases rather than making sure that database operations are not causing resource leaks.
- Hibernate supports transaction management and make sure there is no inconsistent data present in the system.
- Since we use XML, property files or annotations for mapping java classes to database tables, it provides an abstraction layer between application and database.
- Hibernate helps us in mapping joins, collections, inheritance objects and we can easily visualize how our model classes are representing database tables.
- Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association.
- Hibernate also offers integration with some external modules. For example Hibernate Validator is the reference implementation of Bean Validation (JSR 303).
- Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentations and help is easily available in forums.
- Hibernate is easy to integrate with other Java EE frameworks, it's so popular that Spring Frameworkprovides built-in support for integrating hibernate with Spring application
-

2. **What is a transaction? Why does Hibernate need to deal with transaction issue?**

   Allows the application to define units of work, while maintaining abstraction from the q underlying transaction implementation (eg. JTA, JDBC).
   A transaction is associated with a Session and is usually instantiated by a call to Session.beginTransaction(). A single session might span multiple transactions since the notion of a session (a conversation between the application and the datastore) is of coarser granularity than the notion of a transaction. However, it is intended that there be at most one uncommitted Transaction associated with a particular Session at any time.

   Hibernate need to deal with transaction : to link every component of main service with all of function in a project

3. **What are three common relationships in RDBMS? Give one example each.**

4. **What are differences between database relationship and class relationship?**

WEEK 6: Spring MVC

1. **What are benefits of using MVC pattern?**
   Lightweight: Spring is a lightweight framework, there is no performance issues in Spring based web application.
   High productive: Spring MVC is a productive framework which can boost your development.
   Secure: Fully secure, that's why most of the online baking web applications are developed using Spring MVC. Spring Security is a great API for enterprise grade security implementation.
   MVC Supported: As its based on MVC. Its a great way to develop modular web applications.
   TDD Supported: Fully support for Test Driven Development(TDD) technique.
   Fit for Agile Development: Best fit for Agile based web application development.
   Role Separation: Separate class for specific roles like Model, Command, Validator etc..
   RESTful Service Support: In-build RESTful web services supported.
   Mature and Great Community Support : Spring Web MVC is a mature framework and millions of web application have been already developed including eCommerce, CMS, eGovernance, Digital Banking and Finance applications. Its innovative and prompt community can handle your all development problems.
   Even More: Internationalization(i18n), theme support, multiple views supported and seamless integration with database framework like Hibernate, JPA, TopLink and more…

2. **What is rest controller? How to convert a class into a rest controller?**

   Rest controller is Spring's annotation based MVC framework simplifies theprocess of creating RESTful web services.

3. **How many ways to send data with a request?**
   Put rest controller annotation on the top to class name and autowise for varible in side and also request mapping for all function in side it

4. **What are differences between RequestParam, PathVariable, and RequestBody?**
   RequestParam: to add new variable in column
   PathVariable : to access into one column to view
   Request body: Simply put to update and save the column in one service

5. **What is ResponseBody? What format SpringMVC uses for RequestBody?**