Project 2 - All Possible Interview Questions and Answers

Which Java version you are using in this Project?

Currently we are using Java 11

Why did you choose Java 11?

We chose Java 11 for our project because it offers long-term support, improved performance, and enhanced security features over earlier versions. Java 11 also provides new APIs and language enhancements that simplify development and maintenance.

Can you tell me some new features that were introduced in Java 11?

HTTP Client, Epsilon Garbage Collector, Z Garbage Collector, Local-Variable Syntax for Lambda Parameters are some of the new features and along with these new features, isBlank(), strip(), stripLeading(), stripTrailing(), and repeat() were also introduced for strings

Which springboot version you are using in this project?

In this project, we are using Spring Boot version 2.5.4. We chose this version because it offers the latest features, improvements, and bug fixes provided by the Spring Boot framework. Additionally, it ensures compatibility with other libraries and tools used in the project ecosystem.

What's the improvements of 2.5.4 version?

Testing enhancements, and dependency management improvements are the major improvements in this version

What was the challenged you faced in this project?

[Choose any one challenge]:

1: One of the challenges we faced in the Claims Processing System project was managing real-time updates through Kafka. Kafka is a tool that helps send updates instantly, which is very important for keeping insurance agents and clients informed about the status of their claims as soon as anything changes. However, setting up Kafka to deliver these updates quickly and reliably, especially when there are a lot of users, was tricky. We had to carefully adjust Kafka's settings and improve our system to make sure that updates are sent out immediately and without any delays, ensuring everyone involved gets the latest information right away.

2: One Challenge we encountered was ensuring regulatory compliance with data protection laws. Our project, being based in USA and handling sensitive data, required strict adherence to these regulations. In order to overcome this, we updated our services to ensure that

personal data was anonymized or masked when not necessary for processing. We used some encryption techniques like AES-256, and used @JsonSerialze (using = PartialMaskingSerializer.class) on DTO to prevent sensitive data in logging.

3: Migrating this project from Java 8 to Java 11 was also one of the challenges for us. We updated our code to replace old, unsupported features For example, Java 11 removed the javax.xml.bind package and adjusted our project settings to include new dependencies needed for Java 11. We then ran extensive tests to make sure everything still worked correctly.

What was the latest feature you have implemented in your project? OR give any feature, which you have worked on?

Recently, I worked on implementing real-time claim status updates within the Claims Processing System for Statewide Insurance Group. This feature was crucial as it aimed to enhance the transparency and responsiveness of the claims handling process. Previously, clients and agents had to manually check for updates, which was not only time-consuming but also didn't meet the expectation for instant service.

To implement this, I made several changes [Choose any one or two changes below]:

- **Backend Update:** I enhanced the Claims Processing Module to push status updates immediately when any change occurs in the claim's lifecycle. This involved integrating Kafka to publish claim status events, which the system now listens for and processes in real-time.
- **API Development:** Developed RESTful APIs to facilitate real-time data retrieval at the frontend. This ensured that whenever there was a change in the claim status, the system could immediately fetch and reflect this in the user interface.
- **Notification System:** Created a service within our backend application that handles the notification logic. This service is responsible for sending real-time alerts to the client or custer by sending emails (Interviewer might ask, how did you send the mails, check question below) ensuring that both agents and clients receive immediate updates about claim statuses.
- **Kafka Integration:** Configured Kafka topics specifically for claim status updates. This setup allows for efficient message processing and ensures that updates are not only immediate but also accurately distributed without overloading the system.

How do you send the mails by using java mail service?

First, I ensure the Spring Boot Starter Mail dependency is in my project's pom.xml. Next in application.properties, I set up my mail server details, like host, port, username, and password. Then I write a service class that uses JavaMailSender to send emails.

In this service, I craft the email content and use the send method to dispatch emails. And finally, I call my mail service from within the registration logic to send the email.

Can you please explain the schema in your project? OR Can you please name few major tables in your project?

This is already covered in Point 9 of the project explanation.

Can you please provide the name of few REST APIs in your project?

This is already covered in Point 10 of the project explanation.

How are you handle exceptions in your project?

In our project, we handle exceptions by using a centralized exception handling mechanism through Spring Boot's @ControllerAdvice class. This allows us to catch and handle exceptions across all controllers uniformly. We define custom exception classes for specific errors, and our exception handlers return appropriate HTTP status codes and error messages. This approach ensures that errors are managed consistently and that users receive clear, useful feedback on what went wrong.

Can you please give some name of custom exceptions

ExternalApiIntegrationException, *UnauthorizedAccessException*, *ClaimIdIsInvalid*, *UserNotFoundException*, *NotificationSentFailed* are some of the exceptions that we have create so far.

Can you explain the monolithic architecture you used in the Claims Processing System? Why did you choose this architecture?

We chose a monolithic architecture for this project because it simplifies deployment, debugging, and testing processes. This architecture is particularly beneficial for our application as it simplifies the interaction between different components involved in claims processing, such as claim validation, user management, and report generation.

What databases did you choose for this project and why? How did you handle data modeling?

We chose MySQL for its robustness and its ability to handle complex queries, which are essential for claims data processing. Data modelling was managed by clearly defining database schemas that support efficient querying and data integrity.

Can you tell me How did you integrate a MySQL in your project?

First we have added the MySQL dependency in our project's POM file. Then, in our application properties, we set up the connection details for MySQL, like the database URL and credentials.
Then we created repository interfaces in our code using Spring Data, which helps in interacting with MySQL.
Finally, we use these repositories in our services to save, retrieve, and manage data in our MySQL database.

How did you ensure data consistency across different services in the Credit Score Analysis Tool project?

In this project, we ensured data consistency by using the MySQL database for whole project, which helped keep our data uniform and accessible. We also used Kafka to update data across the project and modules in real time and implemented Redis to store and quickly access frequently used data like claim status updates, and hence making sure the information is consistent everywhere it's used.

Can you describe how you implemented security measures in the Claims Processing System? Detail the authentication and authorization strategies.

In this project, we made sure security was tight by using OAuth for user authentication. This means when an agent officers log in, they are verified securely and given a token that they need to access other parts of the system. This token helps ensure that only authorized users can make requests.

What role does Kafka play in this project, and what are its primary functions?

In this project, Kafka is used primarily for handling asynchronous messaging and notifications related to claim status updates. This allows for real-time communication within the system without impacting the performance of the main application.

What are some of the best practices you follow while using Kafka in your projects?

Best practices include defining clear topic strategies, ensuring proper partitioning and replication of topics for fault tolerance, and monitoring Kafka performance continuously to handle scalability and throughput effectively.

Can you explain the OAuth flow used in your project?

In our project, the OAuth flow works like this: bank officers log in and get a token from the User Management module after being authenticated. They use this token for all their requests to access different services. There is a filer in this project so before going into the actual service logic, filter file validates the token and its correct token then the further flow will be executed otherwise not.

What measures did you take to prevent unauthorized access to user data?

To prevent unauthorized access to user data, we implemented validation methods in the filter file to validate access tokens, and applied role-based access control (RBAC) to restrict access based on user roles. Also, we encrypted sensitive data both in transit and at rest.

What was your approach to logging in this project?

We are using Logging here that is implemented by log4j and it is configured to provide detailed logs for debugging and monitoring. We also made the logs centralized via Splunk for easy access and analysis.

How did you integrate log4j in your project ?

1. Excluded the Default Logging in POM.
2. Added Log4j2 Dependencies.
3. Set logging properties in the application.yml file.
4. Created a configuration class for log4j2
5. Finally, injecting looging wherever we are required

How did you integrate Splunk in your code?

First, I added Log4j2 and Splunk HTTP Event Collector dependencies in the POM file

Then I created a class to configure Log4j2. This class sets up the Splunk appender with the required URL and token, and adds it to the root logger.

Note*: The Splunk Appender is a component used in Log4j2 configuration files to send log data directly to Splunk. It defines how log messages are formatted and where they are sent, specifically pointing to the Splunk server.

How would you determine which data should be cached in this project?

In the "Credit Score Analysis Tool" project, we determine which data to cache by identifying frequently accessed and computationally expensive data, such as frequently access claimed, external API results, and frequent users.

Describe the process of integrating Redis caching into the codebase of your project?

- Step 1: We added the Dependencies in the POM file
- Step 2: we configured the redis in our application.yml file such as host, port and password.
- Step 3: Then we defined a configuration class to set up RedisTemplate.
  Then we annotated our service methods with @Cacheable (Fetch credit score from database and cache), @CachePut (Update the credit score in the database and cache), and @CacheEvict (Remove the credit score from database and cache) to apply caching behaviour.

What strategies are used to handle large data loads and maintain performance in your system?

To handle large data loads and maintain performance, we use techniques such as database indexing, query optimization, and load balancing across multiple server instances. Additionally, heavy computations are offloaded to background processes where possible to keep the user interface responsive.

Describe the lifecycle of a claim in the Claims Processing System.

The lifecycle begins when an insurance claim is submitted through the front end. It then goes through initial validation checks for completeness and accuracy. If validated, the claim is processed where calculations and assessments are performed based on the policy details. Following approval or denial, the result is communicated back to the submitting party, and relevant financial transactions are initiated if the claim is approved.

How do you handle testing in your project?

In our project, we use automated tests to check our code. We run small unit tests, larger integration tests, and full system tests using tools like JUnit and Mockito.

Which framework do you use for unit testing and why?

We use JUnit and Mockito for unit testing in our project. JUnit lets us run tests to check different parts of our code and ensuring everything works as expected. Mockito helps us by simulating the parts of the system our code interacts with, so we can test each part in isolation without needing the whole system to be up and running. This makes our testing process quicker and more thorough.

Can you explain the CI/CD pipeline set up for this project? What tools were involved?

In our CI/CD pipeline for the Credit Score Analysis Tool project, we used Jenkins to automate the build and deployment processes. Jenkins pulls code changes from GitLab, builds the project using Maven, runs tests with JUnit and Mockito, and then deploys the application using Docker and Kubernetes for containerization and orchestration.

Imagine you need to update the new feature without downtime. How would you proceed?

We would use blue-green deployment techniques to deploy the new version alongside the old and gradually shifting traffic to the new version once it's proven stable and ensuring no downtime.

How would you handle a sudden increase in load,?

We would use auto-scaling capabilities in Kubernetes to dynamically allocate more resources based on the load and ensuring the system can handle spikes without degradation of performance.

Which methodology do you use for your project management?

We use the Agile methodology, which allows for flexible planning, progressive development, early deployment, and continuous improvement.

Describe how you structure sprints in your project. What is the typical duration of a sprint, and how are tasks prioritized?

Sprints are typically two weeks long. Tasks are prioritized based on their business impact and urgency, guided by the product owner in collaboration with the team.

Which Agile framework (Scrum, Kanban, etc.) do you use in your project, and why was it chosen?

We use Scrum because of its structured approach to managing large projects, its emphasis on regular updates, and its ability to handle complexity through roles like Scrum Master and Product Owner.

How are user stories created and maintained for your project? Who is responsible for writing these stories?

User stories are created by the product owner with input from stakeholders/clients and the development team. They are maintained in a product backlog, which is regularly groomed and prioritized.

How is testing integrated into your Agile process? Are there dedicated sprints for testing, or is it continuous?

Testing is continuous throughout the development process. Each sprint includes development and testing tasks and ensuring that new features are both developed and tested within the same sprint.

How do you determine the coverage of unit tests in our project?

In our project, we integrate SonarQube directly to measure test coverage. I configure our continuous integration pipeline to run unit tests and generate a coverage report which SonarQube analyzes. This setup allows me to keep track of how well our tests cover the code base and ensures that any new code submissions do not decrease our overall coverage percentage.

What's the minimum sonar coverage rule in your project?

In our project, the minimum coverage threshold set on SonarQube is 80%. This rule applies to both line and branch coverage. I monitor this closely, and if any code commit causes the coverage to drop below this threshold, SonarQube flags it, and the code cannot be merged until the coverage is improved.

How do you write a basic JUnit test case?

To write a basic JUnit test case, I start by creating a new class annotated with @Test. Inside this class, I write methods that also carry the @Test annotation. Each method typically involves setting up a scenario, executing a function of the application, and then using assertions like assertEquals() to verify that the function behaves as expected.

Can you explain a challenging unit test you wrote for your Claims Processing System?

A challenging unit test I developed for our Claims Processing System involved testing the asynchronous processing of claims when multiple claims arrive simultaneously. The test aimed to ensure that our system could handle bursts of high traffic without loss of data or processing errors. Using Mockito, I mocked the behavior of our Kafka message queues to simulate the arrival of multiple claims at once. The test then verified that each claim was processed correctly and in the right order, ensuring that our system's concurrency mechanisms were robust and reliable. This was crucial for maintaining high performance and accuracy during peak operational times.

How do you manage branching and merging in your project’s Git repository?

In our project, we use the Git Flow branching model. I create feature branches for every new feature or bug fix. Once a feature is developed and tested locally, I push the branch to the remote repository and create a pull request. The code is then reviewed by peers, and after successful review and passing all checks in our CI pipeline, it is merged into the develop branch. For releases, we merge develop into master and tag it appropriately.