Project 1 - 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: Challenge we faced in the project was with the security setup, specifically managing user login tokens across different parts of the system. At first, our services weren't always recognizing the login tokens correctly, which led to errors and blocked users from accessing the system. To fix this, we adjusted our settings in the Spring Cloud Gateway, which is the main entry point for requests, to make sure it checked the tokens properly.

Follow up: What was the specific setting you did in Spring Cloud Gateway?

We did route configurations to ensure all requests had valid OAuth tokens by using TokenRelayGatewayFilterFactory for automatic token passing and validation.

2: One Challenge We faced was related to handling high load and data throughput in real-time credit score updates using Kafka. Initially, our system struggled with latency and message loss during peak traffic times, which led to delays in credit score calculations. To overcome this, we optimized Kafka's configuration by adjusting partitioning strategies to distribute the load more evenly across the Kafka brokers

3: One Challenge we encountered was ensuring regulatory compliance with data protection laws, particularly the European Union's General Data Protection Regulation (GDPR). Our project, being based in Sweden and handling sensitive financial 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.

4: 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.

5: One of the most challenging tasks I worked on was setting up real-time credit scoring using Kafka. This meant that Swedbank could quickly get the latest credit scores for customers. By connecting different parts of the system through Kafka, we made sure that whenever new financial data came in, it was processed right away. This helped the bank make faster decisions and provide better service to its customers.

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

Recently I worked on feature enhancement that involved the integration of automated alerts for credit score changes into the Credit Score Analysis Tool. Initially, the system lacked this communication features, meaning users had to manually check for updates to credit scores.

To implement this, I did some code changes like:

- I updated the Credit Scoring Service to trigger alerts whenever a credit score changed beyond a pre-defined threshold.
- I created new service file in credit calculation microservice, to handle the delivery of these alerts via email and in-app notifications. (Interviewer might ask, how did you send the mails, check question below)

Some UI changes were also done by but different UI dedicated team

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*, *ReportGenerationException*, *UserNotFoundException*, *CreditScoreNotFoundException* are some of the exceptions that we have create so far.

Can you explain the microservices architecture you used in the Credit Score Analysis Tool? Why did you choose this architecture?

We have divided our microservice architecture into the different services, the major services are: Data Collection Service for gathering and preprocessing financial data, Credit Scoring Service for calculating credit scores, User Management Service for authentication and user management, and Report Service for generating detailed credit reports. We also use Spring Cloud Gateway to direct requests to the right service.

We used a microservices architecture just to convert our project into smaller, independent services that perform specific tasks. Also, this architecture was chosen because it allows for easier scaling, independent deployment of features, and better fault isolation.

How do microservices communicate in your architecture?

In our project, microservices communicate through Spring Cloud Gateway, which routes requests to the respective service. Internally, services like the Credit Scoring Service and Data Collection Service interact directly using RESTful APIs to exchange data. For real-time communication and updates, we use Kafka that allows services to respond instantly to changes, such as new data for credit score calculations.

What are the benefits and drawbacks of using an API Gateway?

Benefits: Centralized management of security and authentication, simplified client-side communication, and efficient request routing and load balancing.

Drawbacks: Introduces a single point of failure, potential bottleneck if not properly managed, and can increase complexity in debugging.

What databases did you choose for this project and why?

For this project, we chose MySQL as the primary database for all microservices because of its reliability and strong support for complex queries. We're also considering MongoDB for future updates but as of now we are using MySQL only for all the services.

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 the Credit Score Analysis Tool project, we ensured data consistency by using the same MySQL database for all microservices, which helped keep our data uniform and accessible. We also used Kafka to update data across services in real time and implemented Redis to store and quickly access frequently used data like credit scores, and hence making sure the information is consistent everywhere it's used.

What are the security measures in your project Credit Score Analysis Tool? Can you detail the authentication and authorization strategies?

In this project, we made sure security was tight by using OAuth for user authentication. This means when bank 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. We also used Spring Cloud Gateway, which checks these tokens before allowing any requests to go through to the services and hence adding an extra layer of security.

What challenges did you face while integrating external APIs for credit data retrieval and how did you overcome them?

We majorly faced the challenges due to data formats, in the responses of different external APIs, there might have different data formats and mapping those was the challenge. Currently we are using a dedicated service file in the data collection microservice to map these data but in future we might create a different dedicated microservice for the same

What is the main role of kafka in your project?

The main role of Kafka in our project is to manage real-time data updates between different parts of the system. It helps our services communicate quickly and efficiently, ensuring that new financial data is processed right away for accurate credit score calculations and keeping data consistent across the system.

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

Some best practices are monitoring Kafka's performance regularly, ensuring data is partitioned effectively across the Kafka cluster, and using appropriate consumer groups to maximize throughput and fault tolerance.

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 Service after being authenticated. They use this token for all their requests to access different services. The Spring Cloud Gateway checks if the token is valid before allowing the request to reach the right service and ensuring secure access throughout the system.

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

To prevent unauthorized access to user data, we implemented OAuth for secure authentication, used Spring Cloud Gateway 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 algorithms did you consider for calculating credit scores? (Part of a different team)

This was handled by a different team, but they considered algorithms based on statistical analysis and machine learning models to predict creditworthiness.

What machine learning libraries or tools did you use in calculating the credit score? (part of a different team)

Again, as handled by a different team, but the common tools they use are Python libraries such as scikit-learn for machine learning models.

How do you handle the generation of large reports without impacting system performance?

To handle large report generation without impacting system performance, we use asynchronous processing with a dedicated Report Service. Reports are generated in the background, and users are notified upon completion. We also leverage caching with Redis to store and quickly access frequently requested data, and we use pagination to manage and display large datasets efficiently.

What formats are available for the credit reports, and how are they secured?

Credit reports are available in PDF and HTML formats. They are secured through encryption and are only accessible through secure authenticated sessions.

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 this project, we determine which data to cache by identifying frequently accessed and computationally expensive data, such as credit score calculations, external API results, and reference data.

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.

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 Credit Score Analysis Tool?

One particularly challenging test case I developed for our Credit Score Analysis Tool involved ensuring the system accurately recalculated scores under concurrent user queries. To simulate this, I used Mockito to mock concurrent access to the scoring service. The test involved initiating multiple threads to fetch and update scores simultaneously and then verifying that all updates were processed correctly without any data corruption or loss. This test was critical for confirming the robustness of our system in a high-load, real-time environment.

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.