**Backend Technology Assessment**

**Exercise 1**

* Low-Level Design
* Software Development Practices
* Hands-on Programming

**Currency Exchange and Discount Calculation**

**Description** Develop a Spring Boot application that integrates with a third-party currency exchange API to retrieve real-time exchange rates. The application should calculate the total payable amount for a bill in a specified currency after applying applicable discounts. The application should expose an API endpoint that allows users to submit a bill in one currency and get the payable amount in another currency.

**Requirements:**

1. **Third-Party API Integration:**
   * Integrate with a currency exchange API, such as [ExchangeRate-API](https://www.exchangerate-api.com/) or [Open Exchange Rates](https://openexchangerates.org/), to get real-time currency exchange rates.
   * Use the API key (replace your-api-key in the URL below) to access exchange rates.
     + Example endpoint: https://open.er-api.com/v6/latest/{base\_currency}?apikey=your-api-key
2. **Discounts and Currency Conversion Logic:**
   * Apply discounts as per the following rules:
     + If the user is an employee of the store, they get a 30% discount.
     + If the user is an affiliate of the store, they get a 10% discount.
     + If the user has been a customer for over 2 years, they get a 5% discount.
     + For every $100 on the bill, there is a $5 discount.
     + The percentage-based discounts do not apply to groceries.
     + A user can get only one of the percentage-based discounts on a bill.
   * Convert the bill total from the original currency to the target currency using the retrieved exchange rates.
   * Calculate the final payable amount in the target currency after applying the applicable discounts.
3. Authentication:
   * Implement authentication for the exposed endpoints.
4. **Endpoint Exposure:**
   * Expose an API endpoint (/api/calculate) to accept bill details including items, their categories, total amount, user type, customer tenure, original currency, and target currency.
   * The endpoint should return the net payable amount in the specified target currency after applying applicable discounts and currency conversion.
5. **Design and Testing:**
   * Use object-oriented programming principles to design the application.
   * Provide a high-level UML class diagram of all key classes in your solution.
   * Write unit tests to achieve good code coverage, utilizing mocking frameworks where applicable.
   * Ensure code simplicity and adherence to modern coding practices.
6. **Documentation:**
   * Maintain a clear README.md that explains how the code and tests can be run and how coverage reports can be generated. Also mention which endpoint you integrated with.

**Bonus Activities:**

* Create build scripts using Maven or Gradle to:
  + Build the project from the command line.
  + Run static code analysis such as linting.
  + Run unit tests and generate code coverage reports.
* Generate a SonarQube report for the code quality summary.
* Implement caching for exchange rates to reduce API calls.

**Submission:** Create a GitHub repository, ensure the name is generic and doesn’t contain any company names. Commit your code to the GitHub repository and share the link with us.

**Follow-up Discussion:** After completing the exercise, be ready for a 15-minute discussion on your key decisions, assumptions, and rationale for your implementation.