InterGold - Junior Software Developer Technical Assessment

Introduction

Thank you for your interest in the Junior Software Developer role at InterGold. This technical assessment is designed to give us insight into your problem-solving approach, coding practices, and technical communication skills.

This is not a test with a single correct answer. We are most interested in understanding your thought process and seeing how you structure a solution. You may use any tools, references, or AI assistants you are comfortable with. If you do use external help, we encourage you to briefly note how it assisted you. Your ability to explain the reasoning behind your decisions is a key part of this evaluation.

Please aim to complete this exercise within **2-3 days** of receiving it.

Evaluation Criteria

We will be evaluating your submission based on the following:

- **Problem Analysis:** Your ability to identify flaws in existing code, particularly concerning security, maintainability, and best practices.
- **Code Quality:** The clarity, structure, and readability of your rewritten code. We prioritize clean, maintainable solutions.
- **Technical Knowledge:** Your application of fundamental programming concepts, including safe database query construction and conditional logic.
- **Communication:** The clarity and thoroughness of your written explanations.

Part 1: Analyze Legacy Code

Below is a simplified C# function representing a style of code that often needs modernization. It retrieves a customer's record from a database.

```
public DataTable GetCustomerInfo(string id)
{
    var dt = new DataTable();
    using (var conn = new SqlConnection("...")) // Connection string is l
    {
        conn.Open();
        var sql = "SELECT * FROM Customer WHERE id = '" + id + "'";
        using (var da = new SqlDataAdapter(sql, conn))
        {
            da.Fill(dt);
        }
    }
    return dt;
}
```

Task

1. In your own words, what is the primary purpose of this function?
2. Identify **at least three distinct problems** with this implementation. Consider aspects such as security, maintainability, and performance.
3. For each problem identified, briefly propose a specific improvement.

Part 2: Rewrite and Modernize

Rewrite the `GetCustomerInfo` function using one of the following modern stacks of your choice:

- **Go** (with the standard `database/sql` package)
- **Python** (with a library like `psycopg2`, `SQLAlchemy`, or `sqlite3`)
- **Modern C#** (.NET 6+ with Dapper or Entity Framework Core)

Note on Database Interaction

You are **not required to connect to a live database**. The objective is to see how you write code that safely interacts with a database library.

Your function should be written to:

1. Use a standard library for your chosen language.
2. Construct the SQL query with the correct placeholders for parameters.
3. Demonstrate how you would pass parameters to the library's execution function to prevent SQL injection.

You can comment out the actual database connection or query execution line and simply return a hardcoded struct, class, or dictionary that represents a customer. We are focused on the **code pattern**, not a live result.

Part 3: Extend with New Logic

Now, extend the functionality. The business requires the ability to filter customer records by their creation date.

Task

Add support for an optional date range filter to the query.

- The function should accept an optional `startDate` and `endDate`.
- If both dates are provided, the query must filter for records where the `created_at` column falls within that range (inclusive).
- If the dates are not provided, the original behavior (fetching by ID only) should be preserved.

Please show how you would implement this logic in **both**:

1. Your **modern, rewritten version** from Part 2 (as complete code).
2. The **original legacy C# code** from Part 1 (pseudocode or a brief code snippet is sufficient).

Part 4 (Optional): System and User Perspective

This part is optional and is intended to see how you think about the practical application of your code.

Briefly describe how a function like the one you built might be used by an employee on an internal web dashboard.

Consider the following points in your description or high-level sketch:

- What input fields would a user (e.g., a customer service representative) need on the screen to search for customer data?
- After the search is performed, what would the output look like on their screen? (e.g., a table, a profile card, a list of transactions).

Submission

Please package your response as a `.zip` archive, or a link to a public Git repository (e.g., GitHub). Your submission should include:

- A document with your written answers for all parts.
- The source code file(s) for your rewritten function.

Send your completed exercise to the provided application email address. We look forward to reviewing your work.