CODE LEAP_

.NET Assessment - GEDAT

General Note

Should you find anything unclear in this assessment, please contact us for clarification so we can assist you in completing it under the best possible conditions.

Technology Stack

- Latest Stable Version of C# and .NET Core
- Package Manager
- You have the flexibility to select any backend architecture that you believe is most suitable for the project. (Three-Tier Architecture, Onion Architecture, Clean Architecture, DDD Architecture, Event-Driven Architecture, Workflow-Oriented Architecture...)
- No mandatory libraries required
    - All libraries are accepted

🔒 **All the above are mandatory and not open to interpretation**

Procedure

- Create a private GitHub repository and share it with the following GitHub users:
    - @Trang Huynh - quangtrang1111
    - @Hoang Dinh - hoangdinhminh93

- We expect you to maintain the repository's privacy out of fairness to all candidates. However, you are allowed to share this repository privately with other companies or recruiters to demonstrate your expertise
- Include a README.md file with instructions on how to build, run, and test the application.

Task

Context

The purpose of this project is to assess your ability to design, develop, and implement an application using C# and .NET Core. This will demonstrate your understanding of RESTful API development, and related technologies.

You are tasked with building a basic e-commerce system that handles operations like user authentication, product management, and order processing.

Evaluation

We will assess the functionality of your application, the quality of your source code, and the setup of tools to ensure project stability.

We recognize that with today's AI technology, creating such an application is more accessible. While we encourage the smart use of AI tools, be aware that merely copying and pasting code without understanding it, especially if detected, will result in immediate rejection of your application. We expect experts to use AI as a support tool, not as a substitute.

Here is a non-exhaustive list of evaluation criteria:

- Code quality, including readability, maintainability, and use of best practices
- Correctness and completeness of the implementation
- Effective use of RESTful APIs and proper HTTP methods
- Security implementation (JWT for authentication)
- Quality and completeness of documentation

The estimated completion time for this task is **8 to 10 hours**.

Functionalities

Core API Services

Implement the following core services with appropriate API endpoints:

- As a **user**, I want to be able to use basic authentication functionalities
    - API to create a user (email + password)
        - Very basic checks are expected
    - API to login as an existing user (email + password), returning a JWT token for authenticated requests
    - API to logout as an existing user (invalidate the user’s session)
    - API to retrieve basic information about the user

    > 💡 Existing services from cloud services can/should be used to ease this implementation

- As a **user**, I want to manipulate products
    - API to create a product with basic information
        - ID
        - Name
        - Description
        - Image URL (can use any online image)
        - Any other information you deem necessary
    - API to update a product
    - API to delete a product
    - API to retrieve list of products with the ability to search by product name.

Security and Authentication

- Implement **JWT-based authentication** to secure sensitive routes.
- Ensure JWT tokens are used for all protected endpoints.

Database Design

Choose a suitable **database** solution (SQL or NoSQL).

- Design tables/collections for **User**, **Product** entities.
- Ensure each entity is properly normalized (for relational DB) or structured (for NoSQL).
- Implement basic data validation at the database level (e.g., constraints, data types).
- Add seed data for testing (e.g., sample products, users).

Monitoring and Logging

- Ensure **error handling** and appropriate logging for exceptions (e.g., 500 Internal Server Error).

Testing

- Implement sample tests to demonstrate the approach and structure for **unit tests** or **integration tests** or both in the project.

Documentation

Basic documentation is expected to explain your decisions:

- As an **engineer**, I want an OpenAPI document must be generated for all APIs in the project, this document should fully describe all available endpoints. The OpenAPI document must include detailed information about the authentication mechanisms used.
- As an **engineer**, I want my project to be documented with one or more of the following resources:
    - Setup instructions
    - Architecture diagrams (e.g. Mermaid)
    - Brief explanation of your design decisions

Bonus

Please be aware that all the bonus tasks listed below are **optional**. You should only undertake them after completing all the mandatory tasks mentioned above.

Note that non-functional bonus features will be regarded in the same way as non-functional mandatory features.

We encourage you to focus on quality over quantity. It is better to do less work thoroughly than to attempt more in a superficial manner.

- As an **engineer**, I want to use a containerization solution to make the system easily scalable and enable simple load balancing.
- As an **engineer**, I want to set up a **CI/CD pipeline** (using GitHub Actions, Jenkins, or Azure DevOps) to automate build, test, and deployment processes
- As an **engineer**, I want to keep my code quality high by integrating a quality control analyzer (using SonarCloud, SonarQube)

Source Code

Do’s

- Ensure the code is well-structured
- Use naming conventions that align with the programming language
- Maintain a consistent coding style throughout (linter, my dear friend)
- Comment on your code for clarity
- Cite your sources if you replicate an algorithm or a significant piece of code
- Use meaningful commit messages
- Document your work

Don'ts

- Avoid copying and pasting without understanding the code
- Do not over-complicate the project

A final word

If you find yourself unable to implement all the mandatory features, prioritize documenting your work to thoroughly explain your process and the context of your test.

Feel free to continue working on your project after submission if you wish to enhance or complete aspects of it. However, we ask that you document any post-submission work in your commits or in the README.md (e.g., "After submission: ..." or "AS: ..."), including the additional time spent.

Do not stress too much over this assessment. Our goal is simply for you to showcase the skills you've honed over the years 🙂.