C# Developer Evaluation Test

Game Engine Developer III and IV

# Goal

The goal of this test is to evaluate candidates based on the technology we use in the math department at Bluberi.

# Limitations

This test should take around 4 hours to complete, so try to be as efficient as possible.

Generative AI can be used.

Any IDE can be used, but keep in mind that the test will be reviewed using Visual Studio 2022.

# Evaluation

Here are the major points this test will be reviewed on

* Architectural judgment
* Code design skills
* Understanding of how all the pieces work together
* Ability to work under constraints and make trade-offs
* Ability to explain and justify decisions
* Depth of their C# and .NET stack knowledge

# The ****Assignment****

## ****Scenario****

You're building a small internal **WPF tool** that receives real-time messages from a **gRPC backend**, routes them through **MediatR**, and displays them in a UI. Only messages that contain the word "test" should be displayed.

## ****Overview****

### Backend

1. Define a InformationMessage ProtoBuf message with fields: Id, From, To, Text, Timestamp.
2. Implement a **gRPC server** that sends messages (simulate streaming or periodic push)
3. Inject dependencies using **Microsoft.Extensions.DependencyInjection**.

### WPF Frontend

1. Create a WPF app with a InformationView.xaml:
   * A ListBox or DataGrid to show filtered messages.
   * A Start Listening button to initiate the gRPC client stream.
2. Use MVVM pattern.
3. Use **MediatR** to process each message (filter if it contains “test”, ignore all other messages).

## Unit Tests

Write xUnit unit tests for frontend logic only:

* Test MediatR message filtering behavior.
* Mock message reception (e.g., using a fake gRPC client).
* Use **FluentAssertions** for all assertions

# Deliverable

Either:

* A .zip with a solution (ChatMonitor.sln with /WpfClient, /GrpcServer, /Tests),
* Or a GitHub repo link