THE ART OF CODE REVIEW
BALANCING PEOPLE AND CODE

Openmind tech

DON'T FORGET TO FILL THE EVALUATION FORM !

WHAT ABOUT ME ?

Personal Life

- 👪 Proud dad of 2 young kids
- 🤩 Foster family with the DYP
- 🏒 Proud fan of Montreal Canadiens… Go Habs Go !

Profession Journey

- 🏅 Bachelor degree in Software Engineering, Certified PSPO 1
- 🔥 Tech Lead at Openmind (7 years)
- 😇 Over 15 years of experience in web applications (PHP, .Net)
- 🎯 Roles: Intermediate Developer → Product Owner → Project Lead → Team Lead → Tech Lead

My passions

- 🌟 Passion for improvement and innovation
- 🌟 Building strong teams through collaboration and empathy

erik-beaulieu

matrix818181

WHO ARE WE?

Openmind Technologies : tech partner for your growth

- Established in 2005, a leading company specializing in **custom software and application development**.
- Experts in **process automation** for SMEs in **manufacturing, distribution, and construction**.
- 50+ employees and experts at your service.

Join our adventure!

OUR DISTINCTIONS

2017 PROFIT 500 CANADA'S FASTEST-GROWING COMPANIES

2018 GROWTH 500 CANADA'S FASTEST-GROWING COMPANIES

2019 GROWTH 500 CANADA'S FASTEST-GROWING COMPANIES

2020 GROWTH LIST CANADA'S FASTEST-GROWING COMPANIES

2019 Canada's Top Growing Companies REPORT ON BUSINESS

M THE MERCURIADES 2020

Certifié BonBoss

THE GLOBE AND MAIL 2020 REPORT ON BUSINESS CANADA'S TOP GROWING COMPANIES

Great Place To Work® 2018

Great Place To Work® 2019

Great Place To Work® 2020

Great Place To Work® 2021

Great Place To Work® 2022

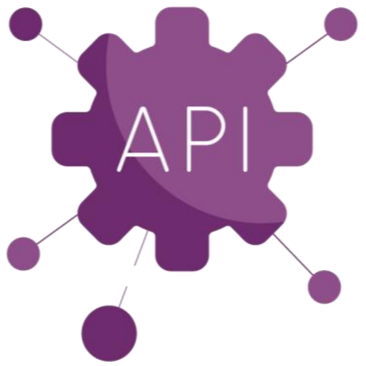
Great Place To Work® 2023

Great Place To Work® 2024

OUR EXPERTISE

Experts in digital automation, innovation, and software modernization, beyond existing solutions.

Automate.



Data connection, middleware integration, and APIs

Innovate.

Development of digital products

Modernize

Application modernization and software rewriting

< WHAT'S ALL ABOUT ?/>

Openmind tech

Today's Agenda

- What's that? Why should I do it ?
- How to... make yourself clearer ?
- I have code in front of me... What should I check ?
- Some examples !
- Let's wrap up !

LET'S GO !

< CODE REVIEW... WHAT'S THAT ? WHY SHOULD I DO IT ?/>

Openmind tech

Code what ?

A code review is a **systematic examination** of the source code carried out by someone other than the person who made the changes when adding or modifying an application.

During this activity, two people are involved:

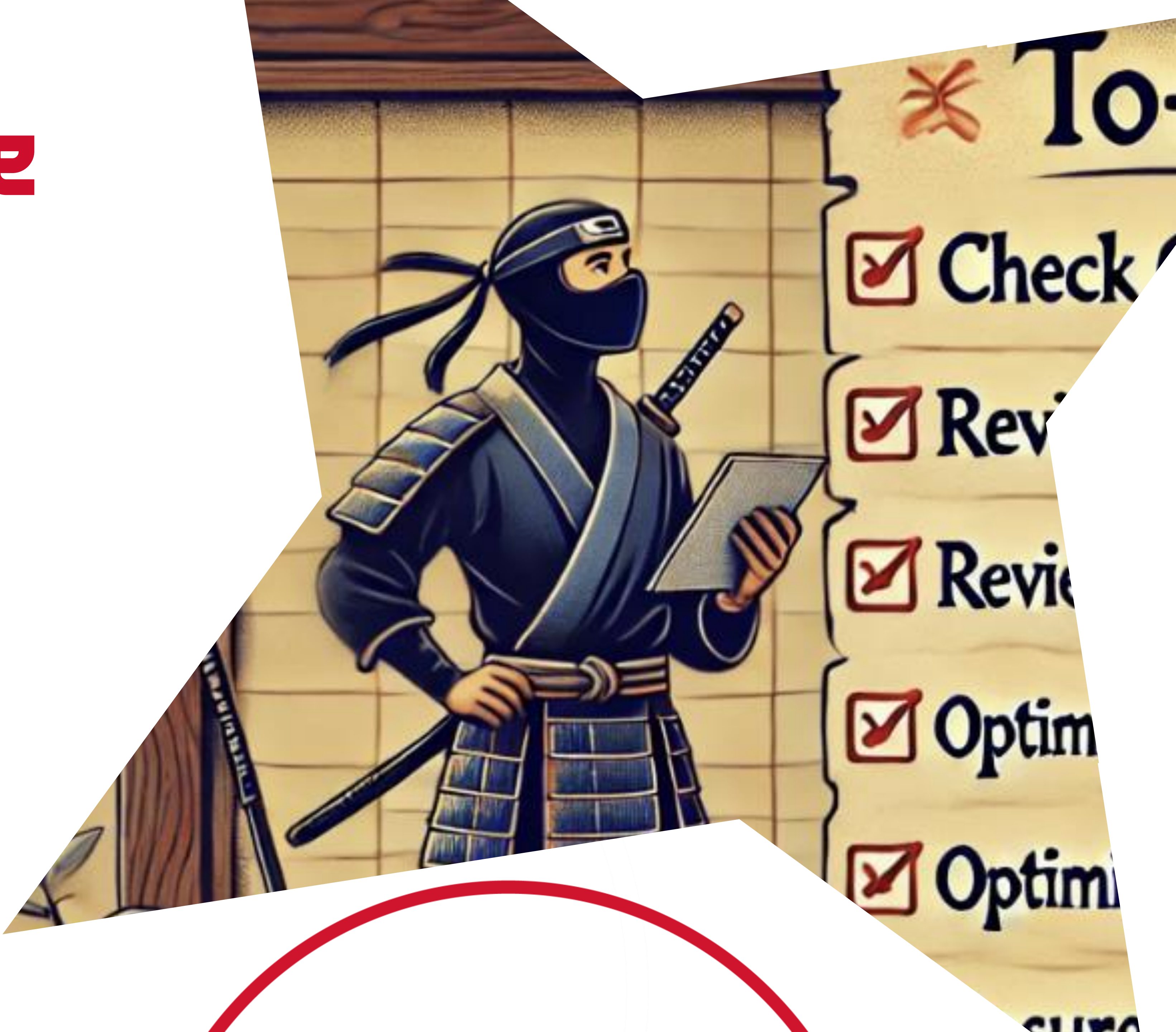
- the author (reviewee)
- the reviewer

WHY ?

- **Improve Code Quality:** Enhance quality and maintainability, including readability, consistency and understandability.
- **Early Bug Detection:** Identify and fix bugs, readability issues early in the code lifecycle to save costs.
- **Encourage Collaboration:** Share knowledge among developers, including:
  - Understand the source code (developer training)
  - Various approaches (apply standards)
  - Set quality expectations for both parties
- **Increase Accountability:** Ensure that all code changes are maintainable by the team
- **Find Better Solutions (Quick Wins):** Establish best practices that transcend the specific code at hand

What can be reviewed ?

- Code... **duh !?**
- Migrations
- Units tests
- Documentation
- Diagrams
- Scripts
- Configurations
- Anything... !



WHAT IS IN IT FOR ME ?

- **Senior Developer:** Ensure overall code quality and maintainability.
- **Mid-Level Developer:** Apply standards and improve skills.
- **Junior Developer:** Adopt best practices and improve code quality.
- **Intern:** Understand team standards and progress through feedback.
- **Software Architect:** Ensure architectural consistency and prevent technical debt.
- **Tech Lead:** Maintain code consistency and optimize the review process.
- **DevOps Engineer:** Ensure code compatibility with pipelines and production.

< HOW TO...
MAKE YOURSELF CLEARER ! />

Openmind tech

PROPER MINDSET 🧠

- **Set Your Ego Aside**

  When you receive feedback on your code, remember it’s the **code being critiqued, not you!** Most of your colleague’s comments should be seen as opportunities to improve and grow.

- **Be Rigorous and Take Your Time**

  If too much poor-quality code pass through, it may be a challenge **to establish the right way of doing right things**.

PROPER MINDSET 🧠

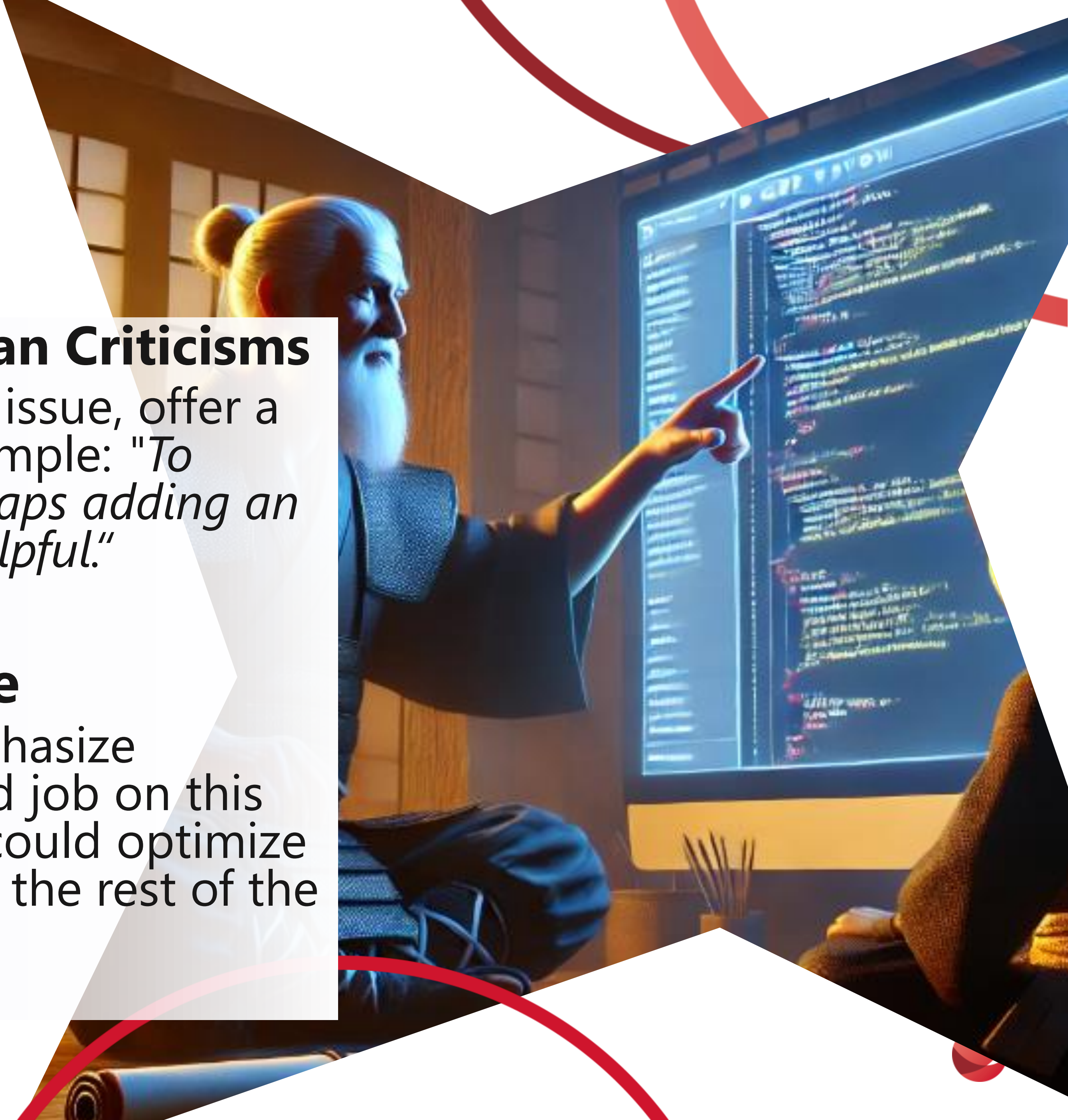
- **Don't Be Afraid**

  You might need to discuss a code review with your colleagues, to help you **understand** their thought process and the **reasons behind the solution they proposed**.

- **Alone We Go Faster, Together We Go Farther**

  An application or client **project is a team effort**. Relying solely on one person **may lead the project in the wrong direction**. Collaboration ensures that everyone understands the importance of code quality and its **long-term benefits**.

HOW TO BE A GOOD REVIEWER ?



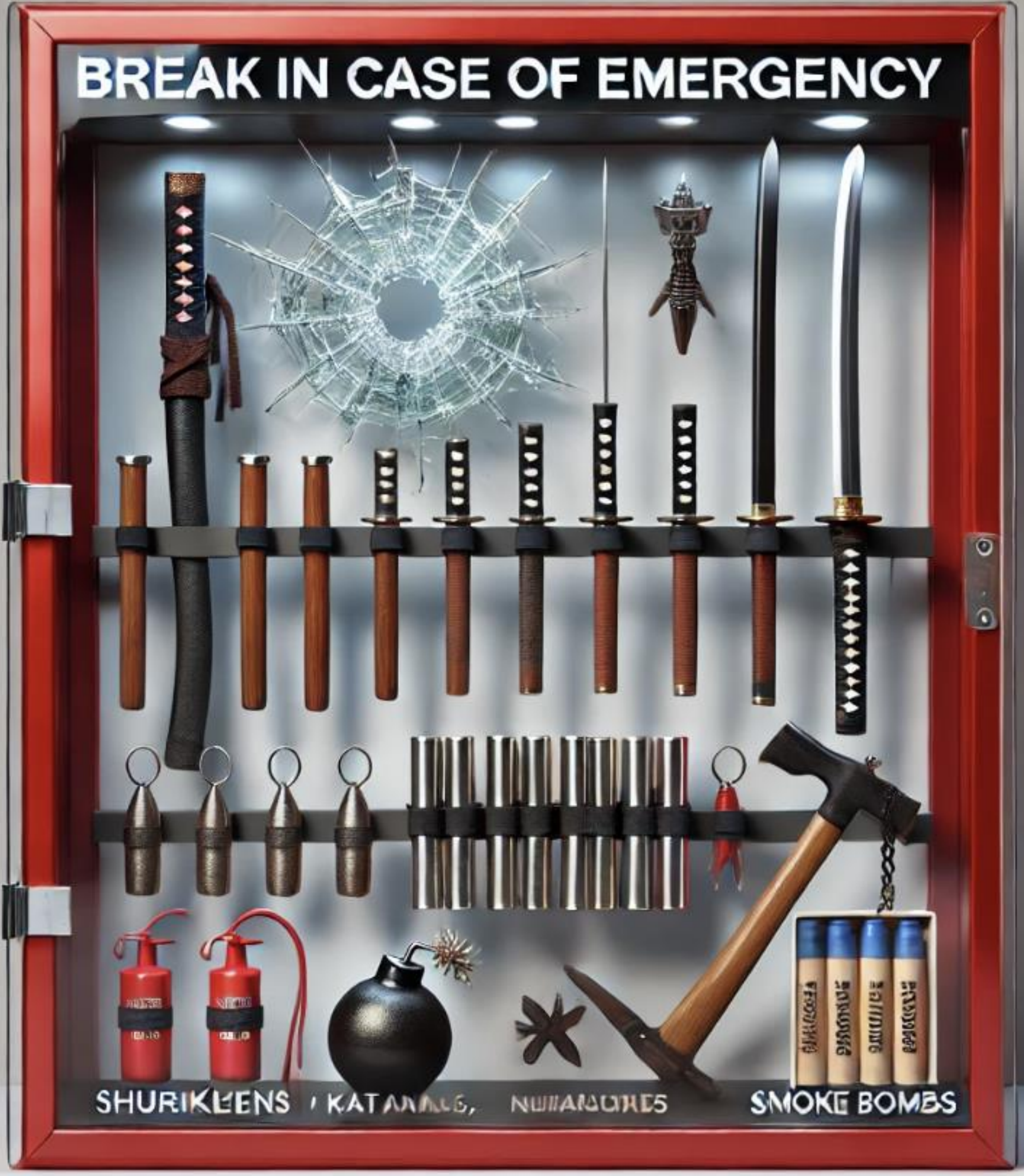
- **Make Suggestions Rather Than Criticisms**
  Instead of simply pointing out an issue, offer a solution or improvement. For example: "*To avoid this performance issue, perhaps adding an index to the database would be helpful.*"

- **Be Respectful and Motivate**
  Use a collaborative tone and emphasize improvement. For example: "Good job on this feature! There's a small thing we could optimize here to maintain consistency with the rest of the project."

How to be a good reviewer ?

- **Explain the Impact of Your Suggestion**
  Highlight the practical benefits, such as better maintainability or improved performance. For example: "*By replacing this loop with a native function, we could reduce execution time and make the code clearer.*"

- **Ask Questions to Avoid Attacks**
  Asking questions can open up a dialogue and reduce the chance of your comment being seen as a personal critique. For example: "*Have you considered using a utility function here? It might make the code more reusable.*"


BREAK IN CASE OF EMERGENCY
SHURIKLENS
KATANAAE,
NUIAMURES
SMOKE BOMBS

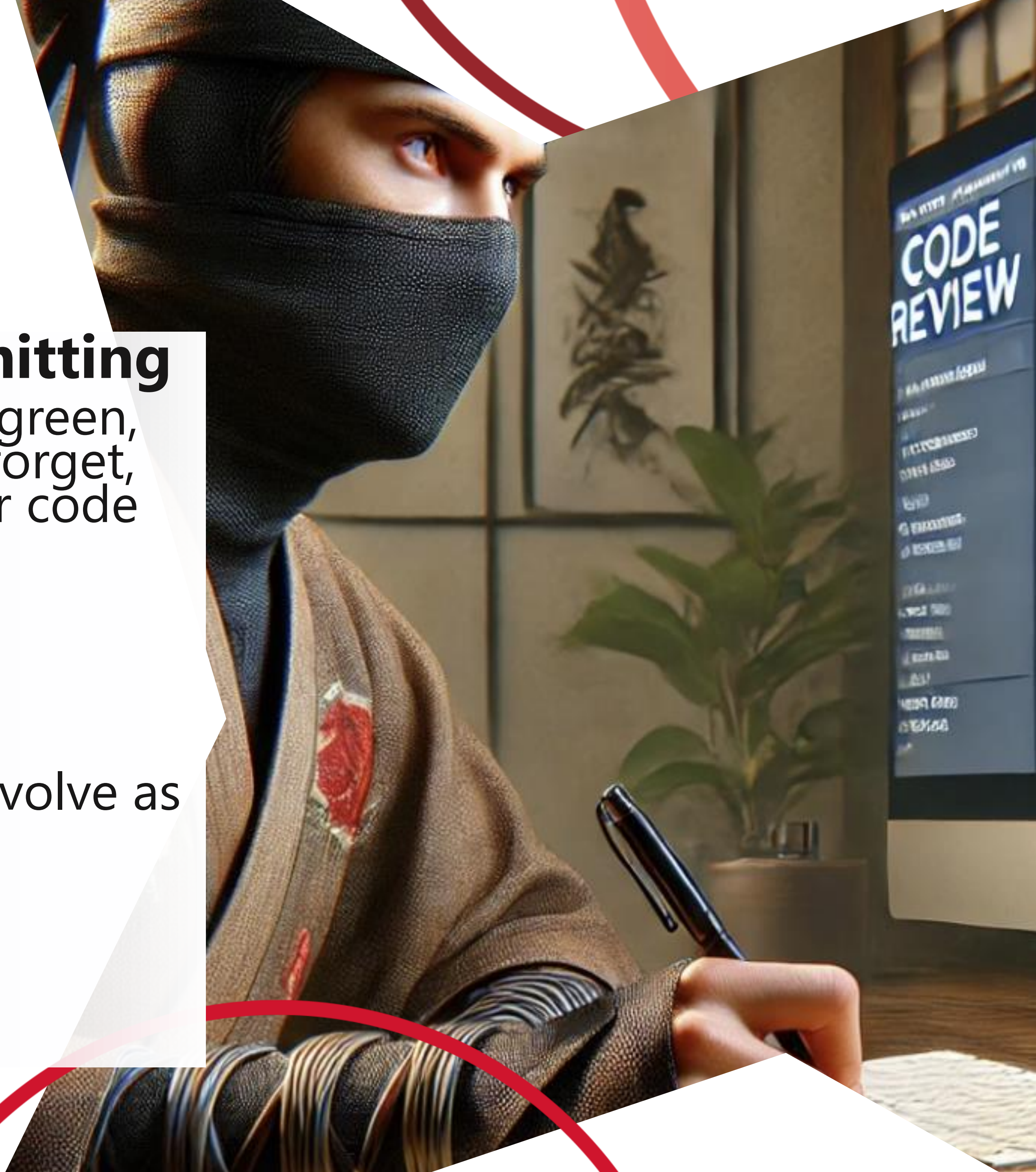
GO TALK TO YOUR COLLEAGE !
OR YOUR AI COMPANION

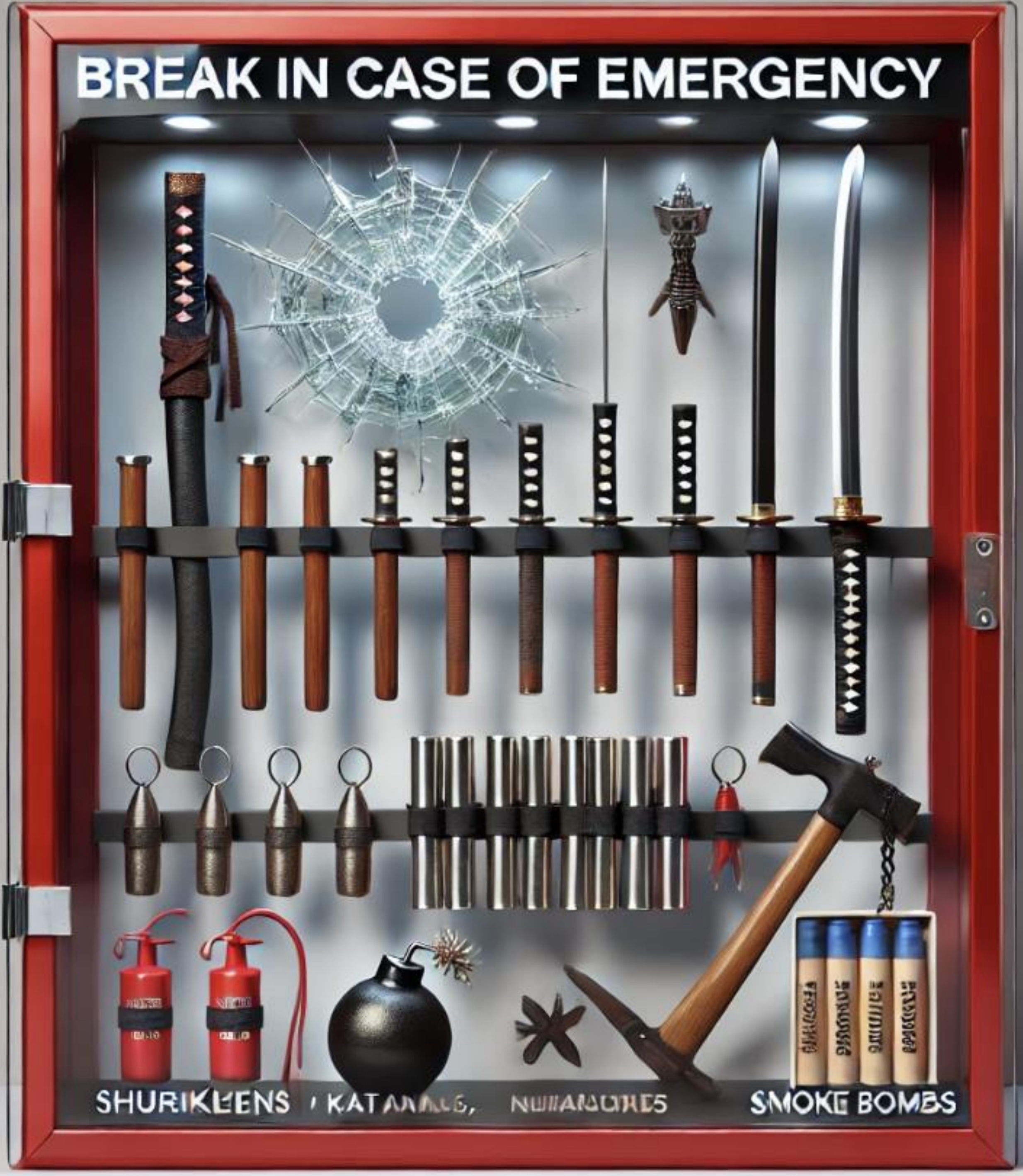
How to be a good reviewee ?

- **Prepare your code before submitting**
  Make sure it compiles, the tests are all green, no unnecessary TODO are lefts. Don’t forget, you can use your own PR to check your code before submitting.

- **Be open to feedback**
  Like a good team player, accepting constructive criticism will help you to evolve as a better developer.

- **But don’t forget…**




BREAK IN CASE OF EMERGENCY
SHURIKLENS
KATANAALE,
NUAANORES
SMOKE BOMBS

GO TALK TO YOUR COLLEAGE !
OR YOUR AI COMPANION

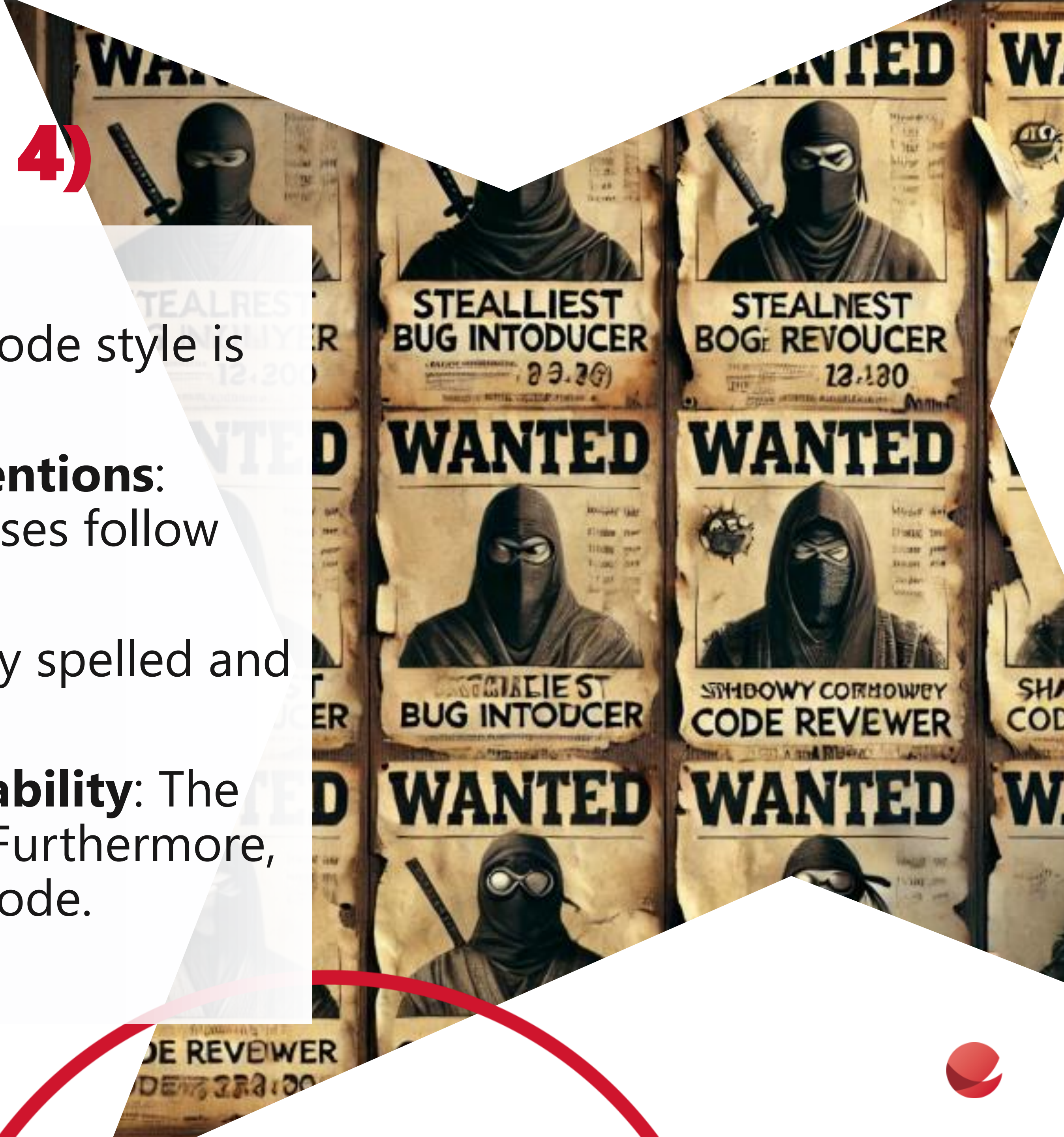
< I HAVE CODE IN FRONT OF ME... WHAT SHOULD I CHECK ? />

Openmind tech

THE NINJA LIST ! (1 of 4)

Readability and Consistency

- **Style Consistency**: Verify that the code style is consistent across the project.
- **Variable Naming / Naming Conventions**: Check if variables, functions, and classes follow proper naming conventions.
- **Spelling**: Ensure names are correctly spelled and align with the chosen language.
- **Clarity, Readability and Maintainability**: The code should be easy to understand. Furthermore, adding clear comments will benefit code.



THE NINJA LIST ! (2 of 4)

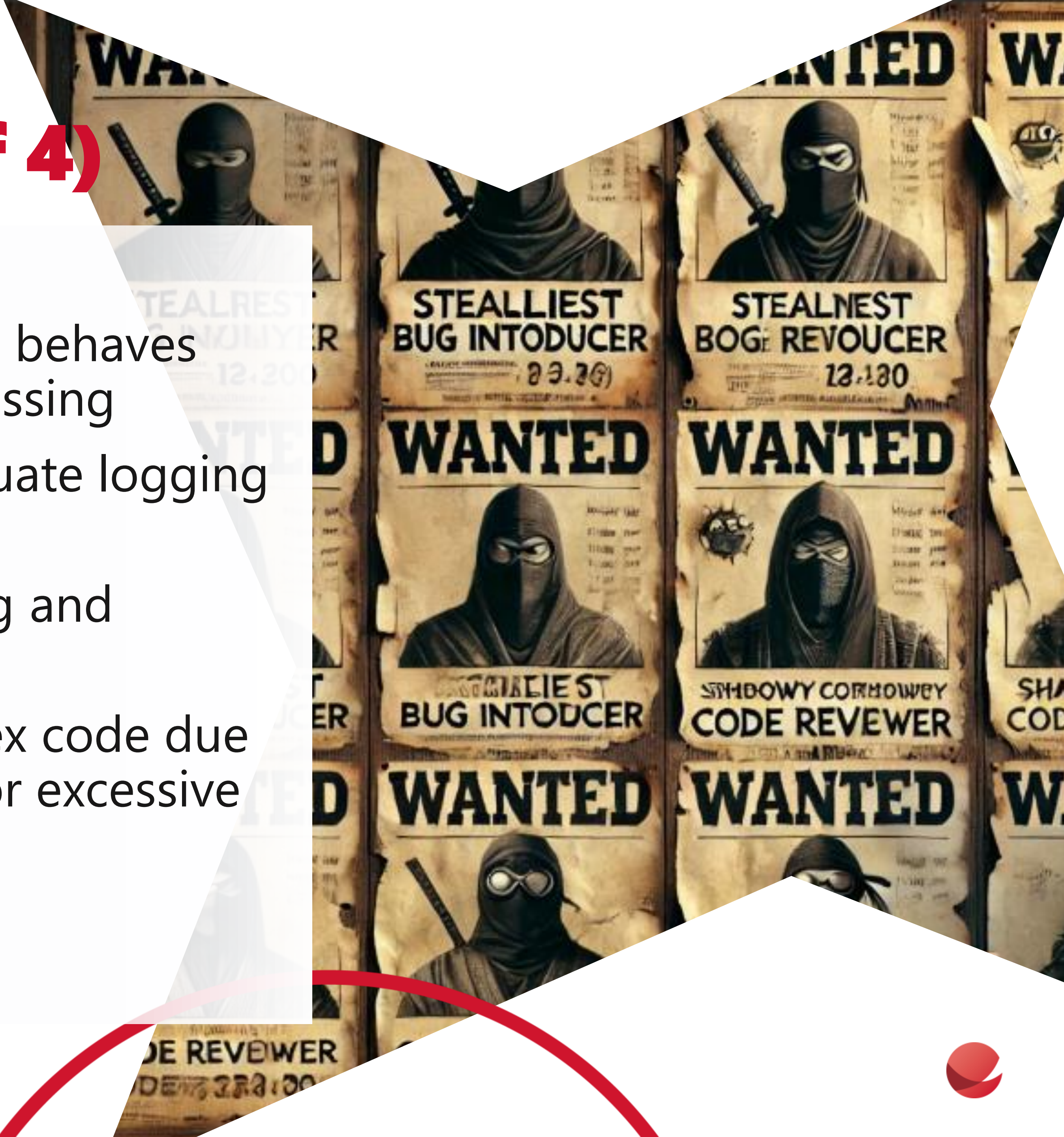
Code Quality and Best Practices

- **Programming Standards**: Ensure compliance with company and industry standards
- **Documentation**: Code is well-commented and avoids unnecessary duplication
- **Dead Code**: Remove unused variables, outdated TODOs, and unnecessary commented-out code
- **Approach Consistency / Don't reinvent the Wheel**: Ensure that existing libraries are used instead of unnecessary custom implementations and avoid redundant logic, don't forget to follow the DRY principle
- **Outdated Practices**: Identify and replace obsolete coding practices with modern alternatives

THE NINJA LIST ! (3 of 4)

Performance and Security

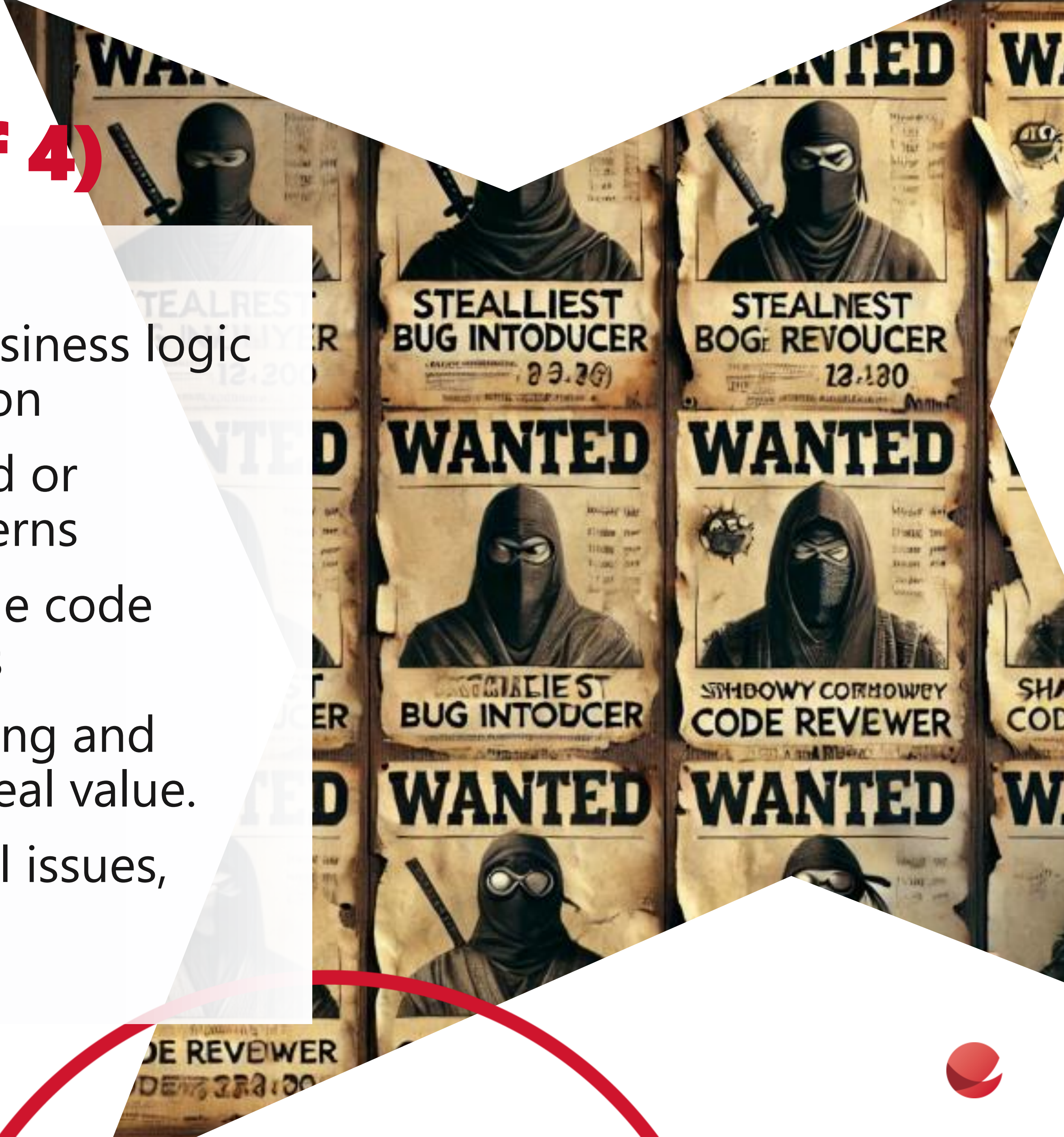
- **Performance**: Assess how the code behaves under heavy load or large data processing
- **Insufficient Logging**: Ensure adequate logging for debugging and monitoring
- **Edge Cases:** Validate input handling and boundary conditions
- **Complexity:** Identify overly complex code due to dependencies, lengthy methods, or excessive calls.



THE NINJA LIST ! (4 of 4)

Architecture and Design

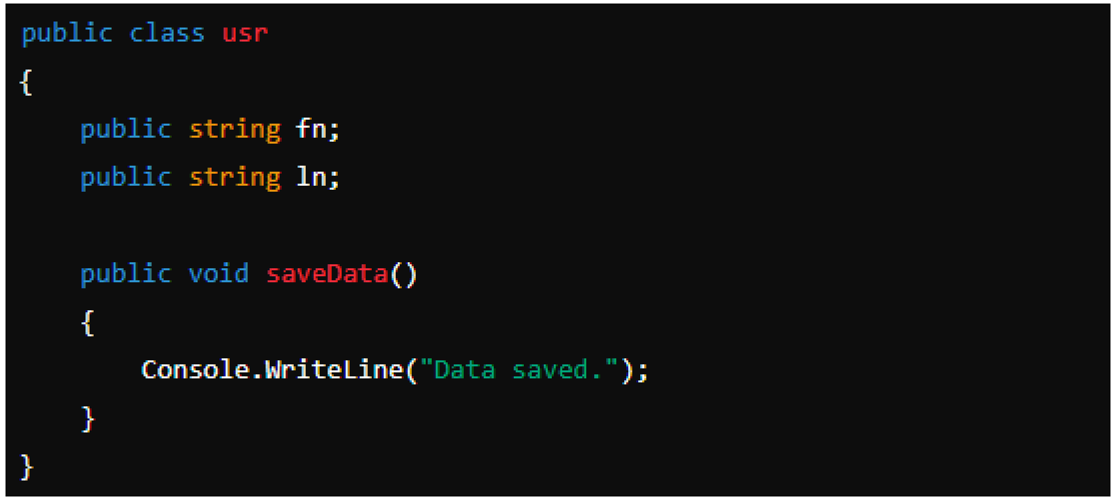
- **Architectural Issues:** Verify that business logic is in the correct layer in the application
- **Code Smells:** Flag poorly structured or incorrectly implemented design patterns
- **Misalign Requirements :** Ensure the code addresses the intended requirements
- **Gold Plating:** Avoid over-engineering and unnecessary flexibility that adds no real value.
- **Common Pitfalls:** Identify potential issues, misuse of libraries or frameworks



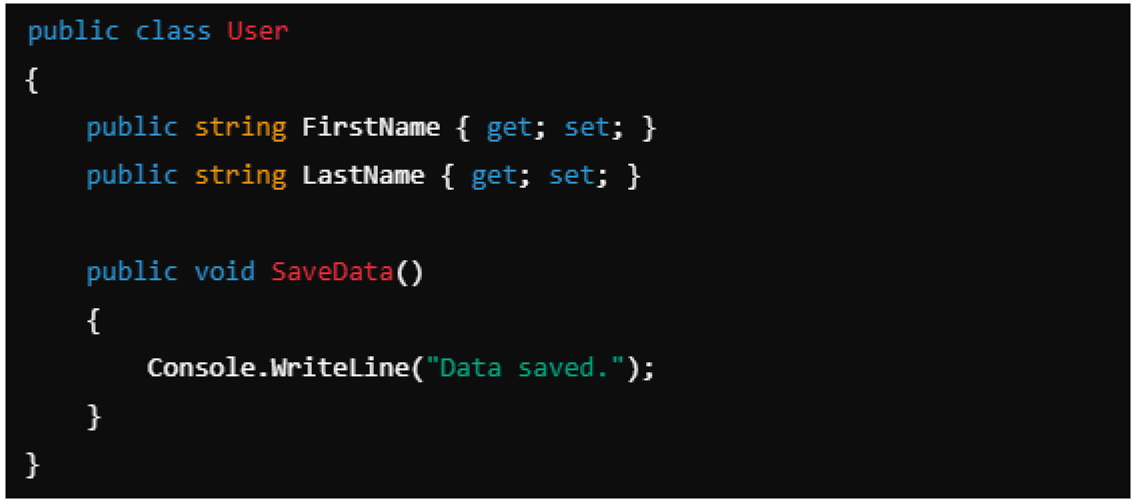
< LET'S SEE SOME... EXAMPLES ! />

The following examples are very basic and were put together for the conference.
The answers are subjective and there is not just one answer per situation.
Please use your best judgment in all cases !

Code smells



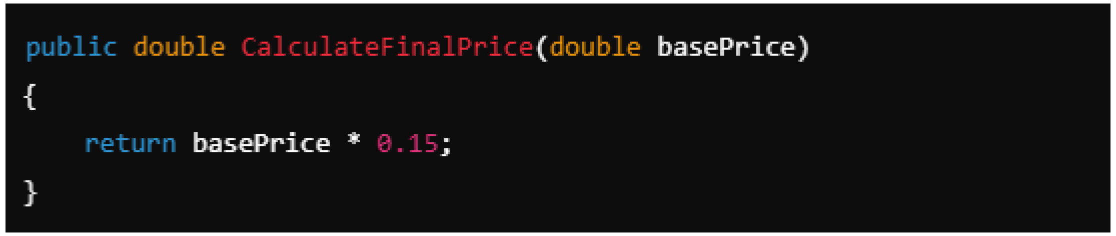
```
public class usr
{
    public string fn;
    public string ln;

    public void saveData()
    {
        Console.WriteLine("Data saved.");
    }
}
```



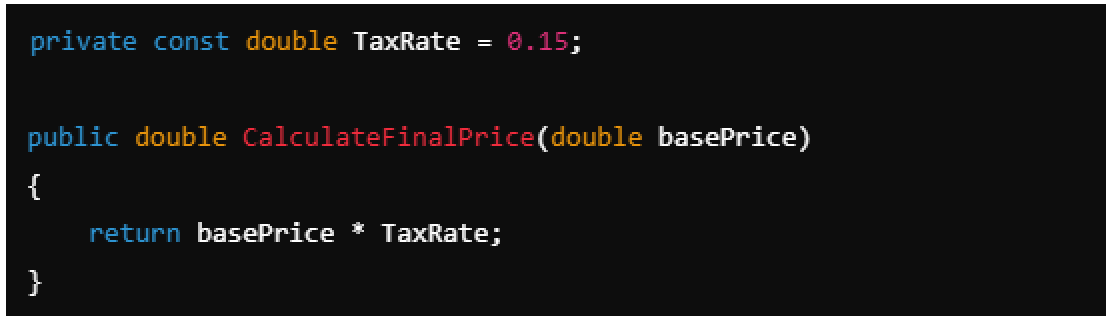
```
public class User
{
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public void SaveData()
    {
        Console.WriteLine("Data saved.");
    }
}
```

A. C'mon man ! You can do it better than that !
B. Plz fix your variables names.
✓ C. I'm not sure what mean *fn* and *ln*. Do you think we can improve those ?
✓ D. Maybe you should name your variables with complete names instead of abreviations. What do you think ?

Code smells



```
public double CalculateFinalPrice(double basePrice)
{
    return basePrice * 0.15;
}
```



```
private const double TaxRate = 0.15;

public double CalculateFinalPrice(double basePrice)
{
    return basePrice * TaxRate;
}
```

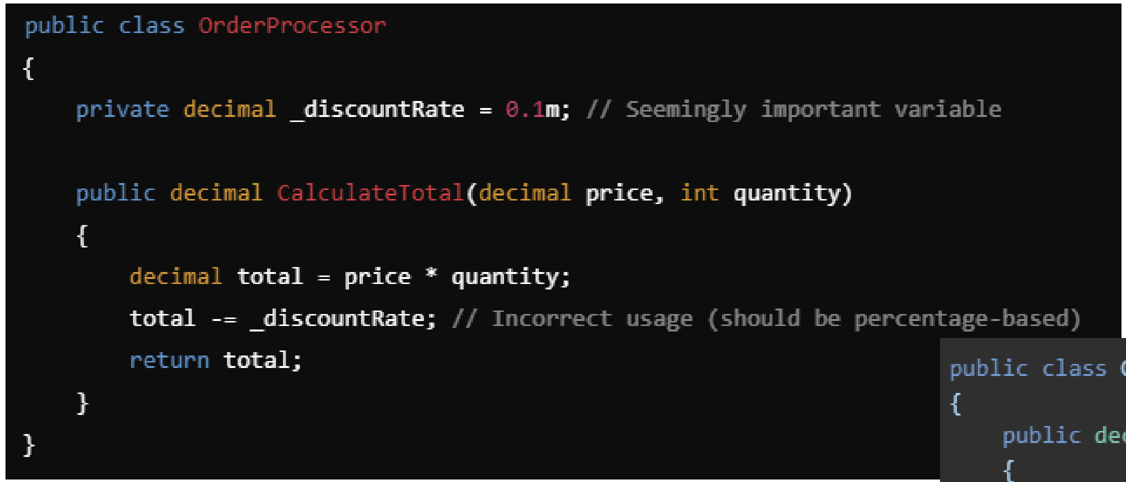
A. Nice ! Sleek code !

[✓ B.] What does 0.15 mean ? Do you think that everybody knows ?

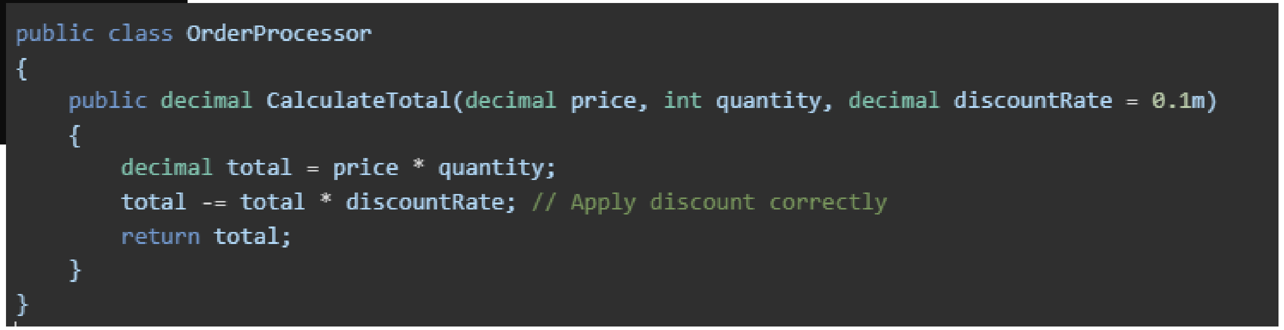
[✓ C.] I understand that 0.15 is the tax rate but do you think we should put it in a constant incase we need to change it later on ?

D. Magic number ? 🪄

Misalign Requirements



```csharp
public class OrderProcessor
{
    private decimal _discountRate = 0.1m; // Seemingly important variable

    public decimal CalculateTotal(decimal price, int quantity)
    {
        decimal total = price * quantity;
        total -= _discountRate; // Incorrect usage (should be percentage-based)
        return total;
    }
}
```



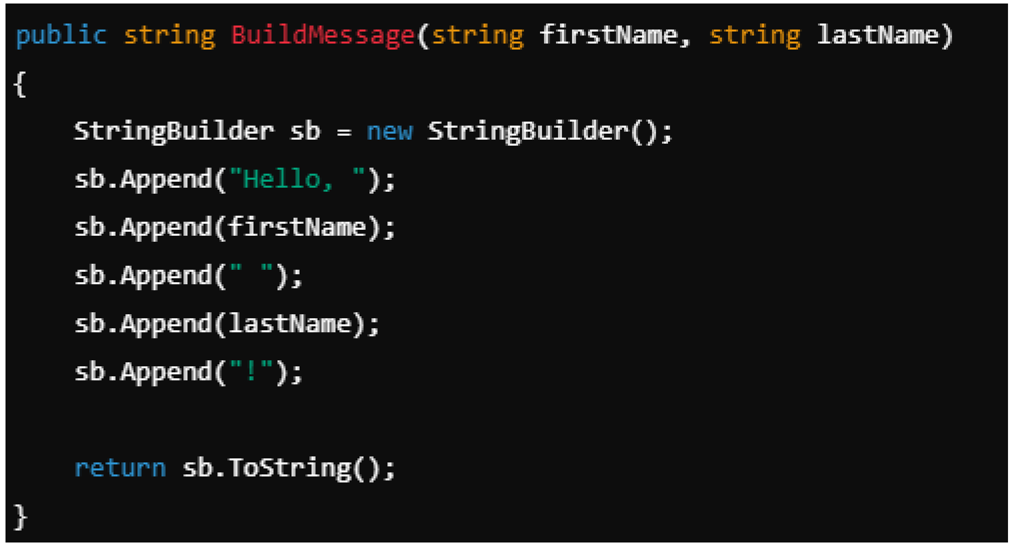
```csharp
public class OrderProcessor
{
    public decimal CalculateTotal(decimal price, int quantity, decimal discountRate = 0.1m)
    {
        decimal total = price * quantity;
        total -= total * discountRate; // Apply discount correctly
        return total;
    }
}
```

✓ A. I think the discountRate should be applied to the total. In addition, I think we could define the variable with a default value. What do you think?

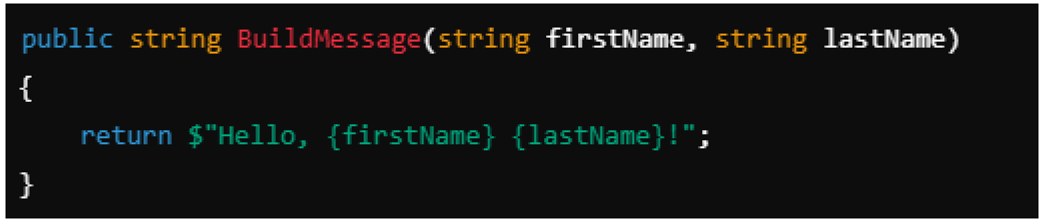
B. I don’t think your discountRate is properly applied.

C. Have you run your unit tests on this code?

Common Pitfalls / Outdated Pratices



```
public string BuildMessage(string firstName, string lastName)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("Hello, ");
    sb.Append(firstName);
    sb.Append(" ");
    sb.Append(lastName);
    sb.Append("!");

    return sb.ToString();
}
```



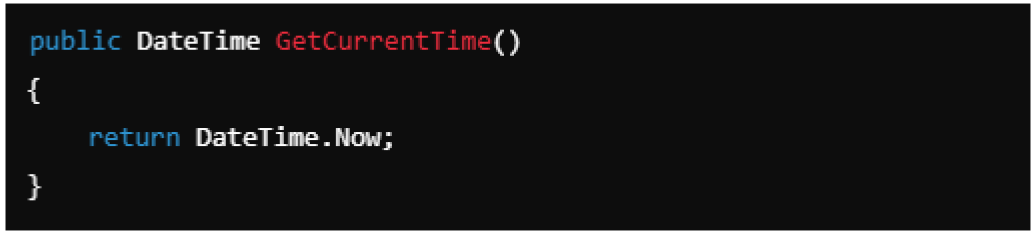
```
public string BuildMessage(string firstName, string lastName)
{
    return $"Hello, {firstName} {lastName}!";
}
```

A. Who use StringBuilder these days ?

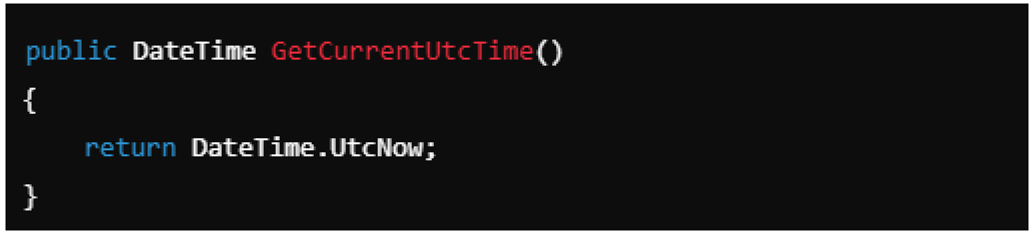
[✓ B.] I think you should use string interpolation. Go take a look at the doc [link to doc here...]

[✓ C.] I don't understand why you used a StringBuilder. Have you heard of string interpolation ?

Common Pitfalls / Outdated Practices



```
public DateTime GetCurrentTime()
{
    return DateTime.Now;
}
```



```
public DateTime GetCurrentUtcTime()
{
    return DateTime.UtcNow;
}
```

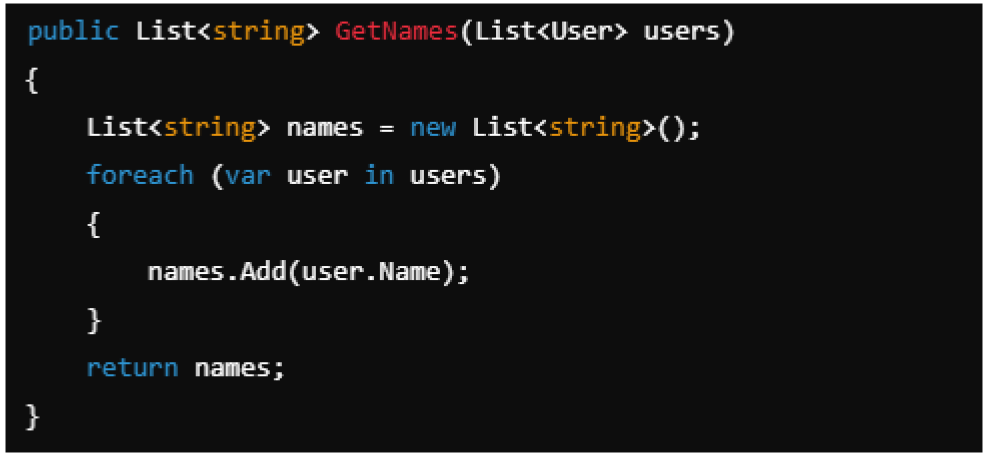
✓ A. What happens if we don't change the time zone to UTC on the server?

B. How many time did I tell you to set your Datetime in UTC ? 😡

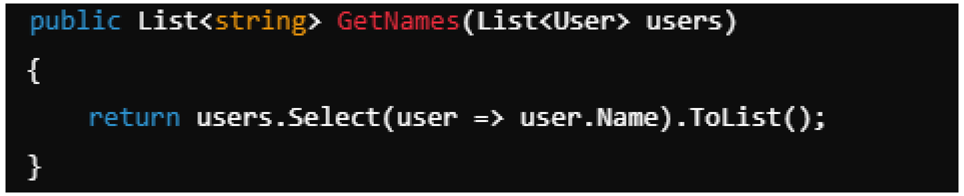
C. I think someone forgot to put his datetime in UTC. 🤣

✓ D. Do you know why we use UtcNow instead of Now ? If you don’t know, we can talk about it.

Common Pitfalls / Outdated Practices



```
public List<string> GetNames(List<User> users)
{
    List<string> names = new List<string>();
    foreach (var user in users)
    {
        names.Add(user.Name);
    }
    return names;
}
```



```
public List<string> GetNames(List<User> users)
{
    return users.Select(user => user.Name).ToList();
}
```

A. Your foreach works fine, but if LINQ were a rocket, this is a bicycle. 🚀

[✓ B.] Consider using LINQ instead of foreach for better readability and maintainability.

[✓ C.] Wouldn't using LINQ here, like `users.Select(user => user.Name).ToList();`, make the code more concise and easier to read?

D. Why loop manually when LINQ can do the heavy lifting? Save yourself the trouble!

< LET'S...
WRAP UP ! />

Openmind tech

Keep a proper mindset

Set Your Ego Aside

Take Your Time

Don’t Be Afraid

Be a good reviewee

Let the feedback flow in !

Be prepared before submitting

Be a good reviewer

Suggestions Rather Than Criticisms

Respectful and Motivating

Explain Why

Ask questions

Talk to the author in case of misunderstanding

Don't forget the ninja list

Practice makes perfect

Don't be narrow with code review

Small details can improve code in the long run

ANY QUESTIONS?

DON'T FORGET TO FILL
THE EVALUATION FORM !