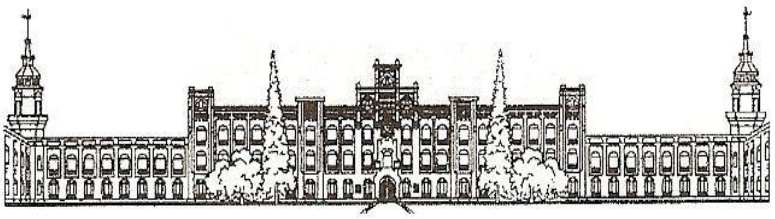

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» ім. Ігоря Сікорського

Звіт

Лабораторна робота №5

з дисципліни «Технології та інструменти розробки ПЗ»

«Відлагодження та тестування програмного забезпечення»

Виконав студент групи: КВ-32

Косарук Захар

Перевірив: Омельченко Андрій

Київ 2025

Мета роботи: отримати навички застосування інструментів відлагодження та

тестування програмного забезпечення.

Завдання:

1. На основі вимог (л.р. №2) скласти чеклист тестування розробленого додатку.

2. Обрати із чеклиста перевірку функціональності (рядок чекліста) та скласти для неї позитивний та негативний тесткейси.

3. Із використанням налагоджувача (debugger) або інтегрованого середовища розробки виконати налагодження складової частини яка реалізує обрану перевірку, а саме:

- додати точку зупинки (breakpoint) на рядок початку коду виконання;
- здійснити покрокове виконання програми від точки зупинки із аналізом змінних на кожному кроці із заходом в підпрограми та без;
- видалити точку зупинки.

В разі відсутності помилки змінити код таким чином, щоб помилка з’явилась.

4. Для виявленого дефекту (реального, або створеного штучно) створити відповідну картку в Kanban дошці GitHub Project із інформацією про:

- ідентифікатор дефекту;
- короткий опис проблеми;
- пріоритет;
- відповідального за усунення;
- очікуваний результат;
- фактичний результат.

5. Виправити помилку із відміткою в Kanban дошці GitHub Project.

6. Оформити звіт із лабораторної роботи. Зміст звіту:

- титульний аркуш із відомостями про виконавця;
- завдання;
- виконане завдання згідно варіанту: чекліст, позитивний та негативний тесткейси, скріншоти відлагодження, скріншоти картки Kanban дошки;
- висновки.

Хід роботи

Завдання 1

Таблиця 1 - Чеклист

№	Найменування (код/назва)	Статус (Pass/Fail)	Примітка (коментар)
1	**TCF_01**: Інсталяція та запуск програми	**Pass**	Програма успішно запускається з виконуваного файлу (.exe) після білду в Unity.
2	**TCF_03**: Вибір типу тригера (RS, JK, D, T)	**Pass**	У головному меню доступні 4 кнопки. При натисканні коректно встановлюється режим роботи.
3	**TCF_05**: Введення параметрів сигналів	**Pass**	Поля вводу приймають текстові рядки. Реалізовано багаторядкове введення для RS/JK тригерів.
4	**TCF_06**: Автоматичне обчислення сигналів	**Pass**	Результат обчислення відповідає таблицям істинності для всіх типів тригерів.
5	**TCF_07**: Відображення результатів	**Pass**	Вихідний сигнал відображається у відповідному текстовому полі після натискання "Calculate".
6	**TCF_08**: Збереження/експорт результатів	**Pass**	Результати успішно зберігаються у .txt файл у папці програми.
7	**TCF_04**: Зміна мови інтерфейсу	**Pass**	Перемикання Ukr/Eng змінює текст на кнопках та лейблах миттєво (через Unity Localization).
8	**TCN_01**: Зручність використання (UI)	**Pass**	Інтерфейс інтуїтивний, елементи не перекривають один одного, навігація працює.
9	**TCN_04**: Надійність (Валідація даних)	**Pass**	Система блокує розрахунок, якщо введено символи, відмінні від 0/1, або якщо довжини рядків не збігаються.

На основі технічного завдання та вимог, сформульованих у Лабораторній роботі №2, було складено чеклист для перевірки поточної стабільності додатку. Проведене первинне тестування показало, що всі

основні функціональні (TCF) та нефункціональні (TCN) вимоги реалізовані коректно, статус перевірок — «Pass».

Завдання 2

Для детального аналізу було обрано модуль UIManager, що відповідає за валідацію вхідних даних (вимога TCN_04). Розроблено два сценарії тестування:

- **Позитивний:** перевіряє роботу системи за нормальних умов (коректні дані).
- **Негативний:** перевіряє реакцію системи на помилку користувача (різна довжина рядків).

На момент початкової перевірки система коректно обробляє обидва сценарії, видаючи результат або повідомлення про помилку відповідно.

Таблиця 2 – Тесткейс (Позитивний)

Поле	**Значення**
Модуль:	UIManager (метод OnButton_Calculate_Click)
Мета:	Перевірка успішного розрахунку JK-тригера при введенні коректних даних однакової довжини.
Передумови:	Запущено додаток TriggerCalc. У головному меню обрано тип тригера "JK".
Сценарій:	Введення валідних бінарних послідовностей однакової довжини.

Таблиця 3 – Сценарій тесткейсу

№ кроку	**Дії (опис кроку)**	**Дані**	**Запланований результат**	**Статус**	**Фактичний результат**	**Коментарі**
1	Ввести сигнал J (перший рядок)	101	Дані введені у поле InputField	Pass	Дані відображаються	-
2	Ввести сигнал K (другий рядок)	010	Дані введені у поле InputField	Pass	Дані відображаються	-

3	Натиснути кнопку "Calculate"	-	Система обчислює результат та виводить його у поле Output	**Pass**	Результат 101 (приклад) відображено	Успішний розрахунок

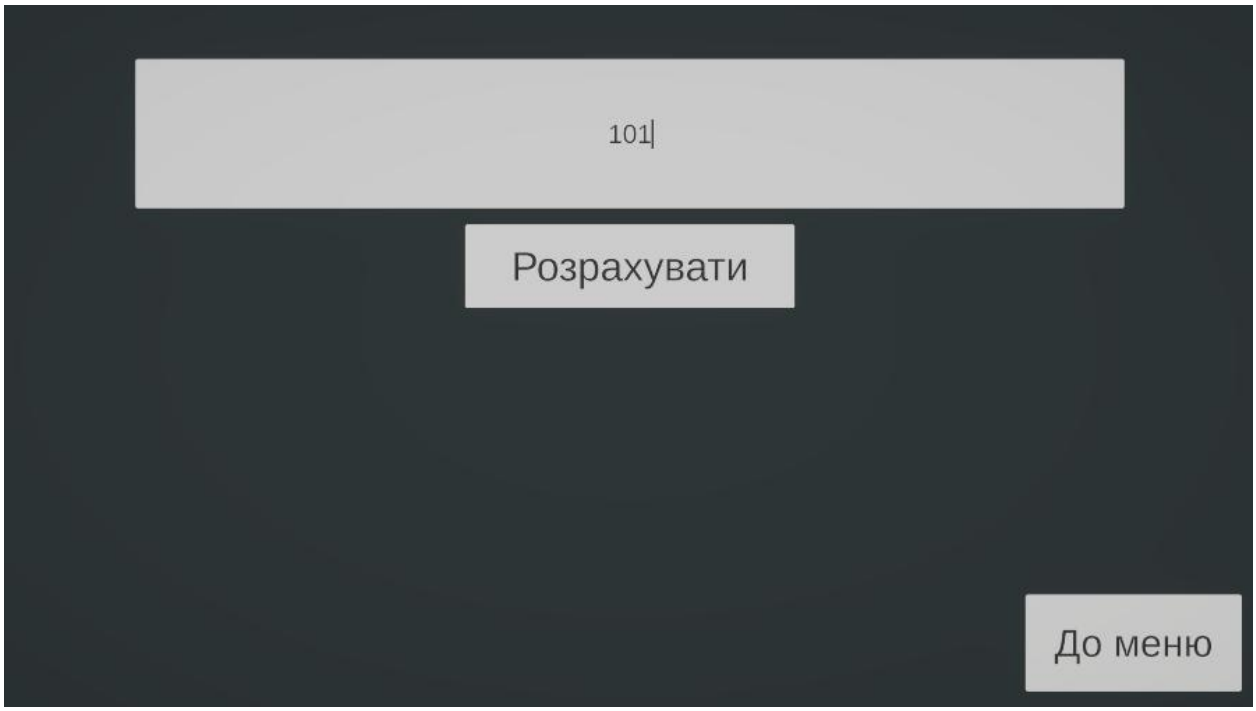

Рисунок 1 - Ввести сигнал J (перший рядок)

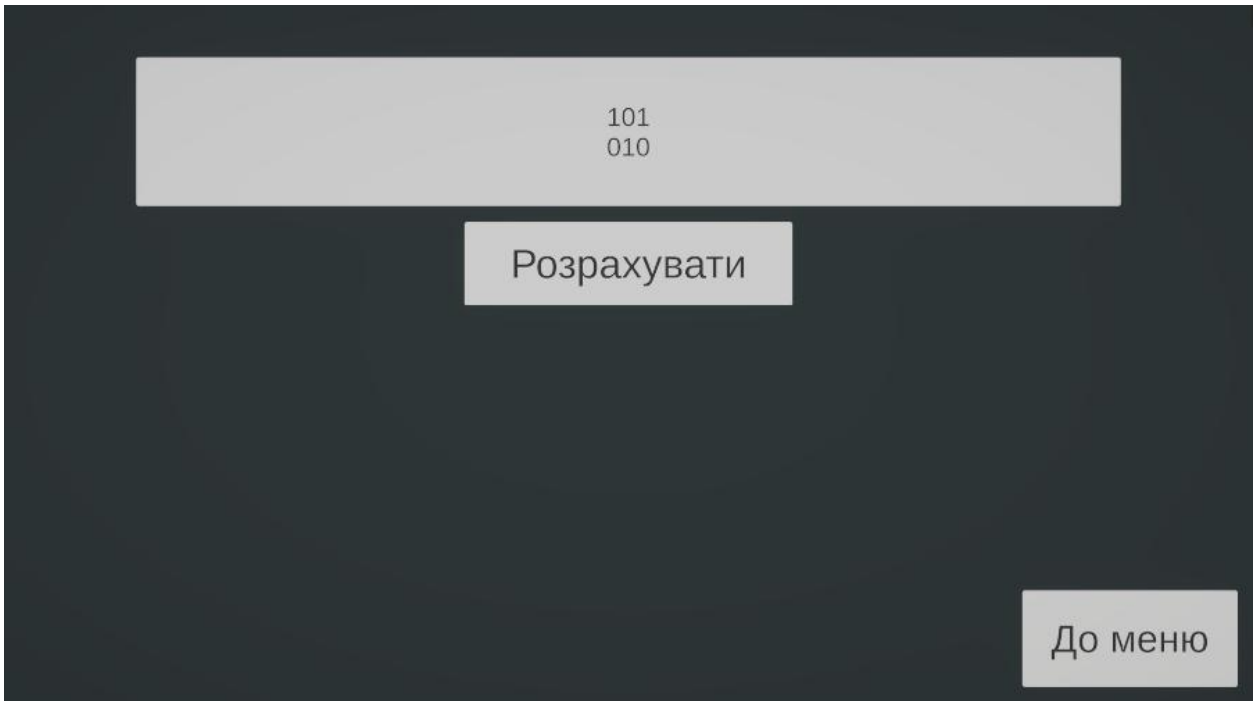

Рисунок 2 - Ввести сигнал K (другий рядок)

Рисунок 3 - Результат 101 (приклад) відображено

Таблиця 4 – Тесткейс (Негативний)

Поле	**Значення**
Модуль:	UIManager (метод OnButton_Calculate_Click)
Мета:	Перевірка спрацювання валідації (захисту) при введенні даних різної довжини.
Передумови:	Запущено додаток TriggerCalc. У головному меню обрано тип тригера "JK".
Сценарій:	Спроба розрахунку з вхідними послідовностями різної довжини.

Таблиця 5 – Сценарій тесткейсу

№ кроку	**Дії (опис кроку)**	**Дані**	**Запланований результат**	**Статус**	**Фактичний результат**	**Коментарі**
1	Ввести сигнал J (перший рядок)	11111	Дані введені (5 символів)	Pass	Дані відображаються	-
2	Ввести сигнал K (другий рядок)	00	Дані введені (2 символи)	Pass	Дані відображаються	-
3	Натиснути кнопку "Calculate"	-	Система **блокує** розрахунок і виводить повідомлення про помилку **error_length_mismatch**	**Pass**	Повідомлення про помилку відображено	Система коректно відхилила невалідні дані

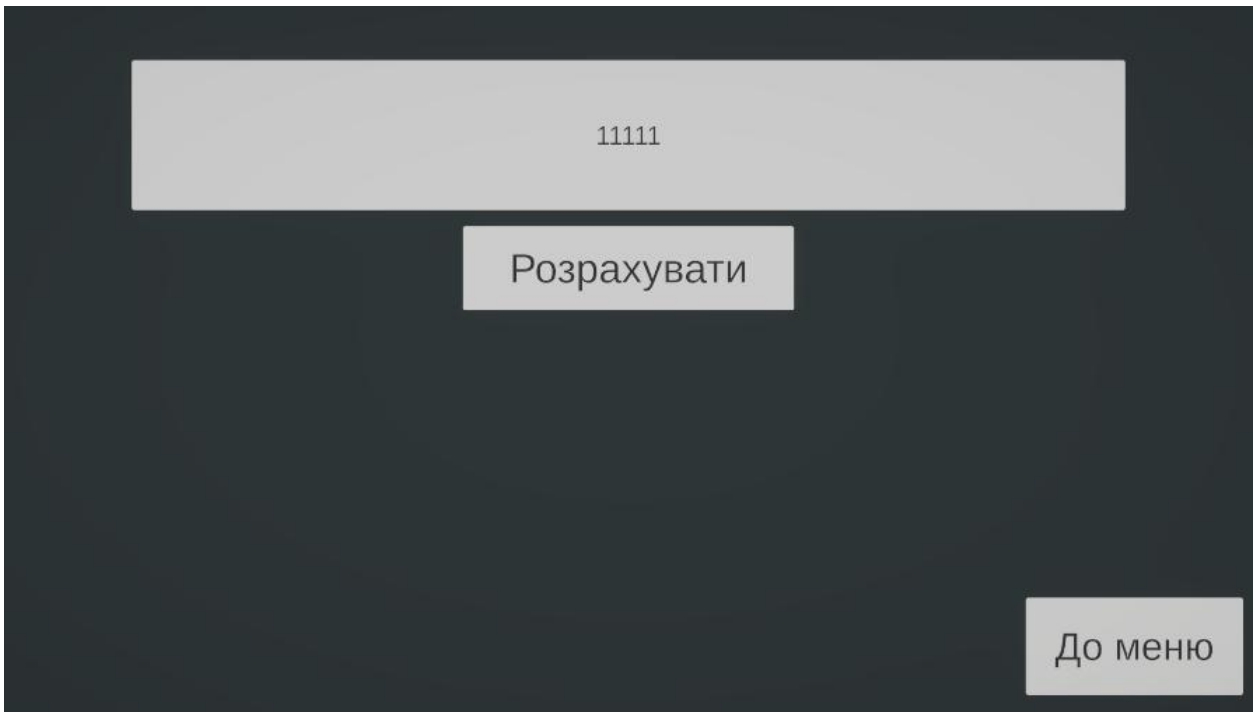

Рисунок 4 - Ввести сигнал J (перший рядок)

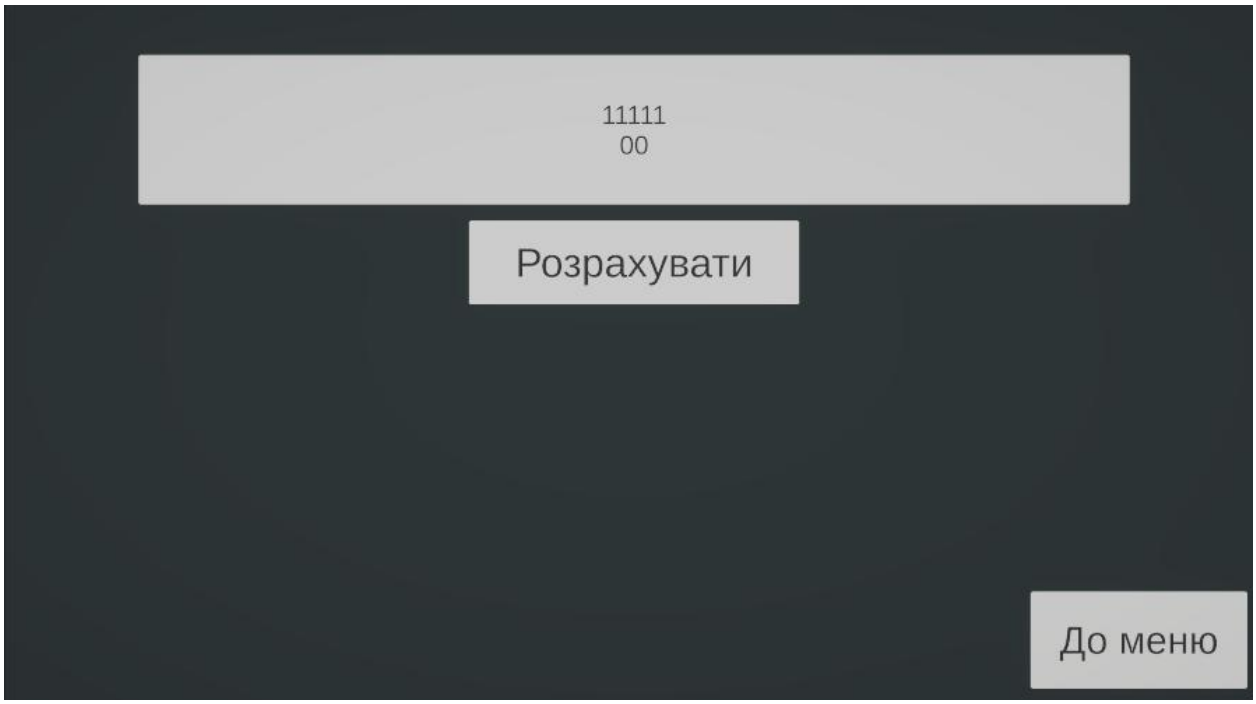

Рисунок 5 - Ввести сигнал K (другий рядок)

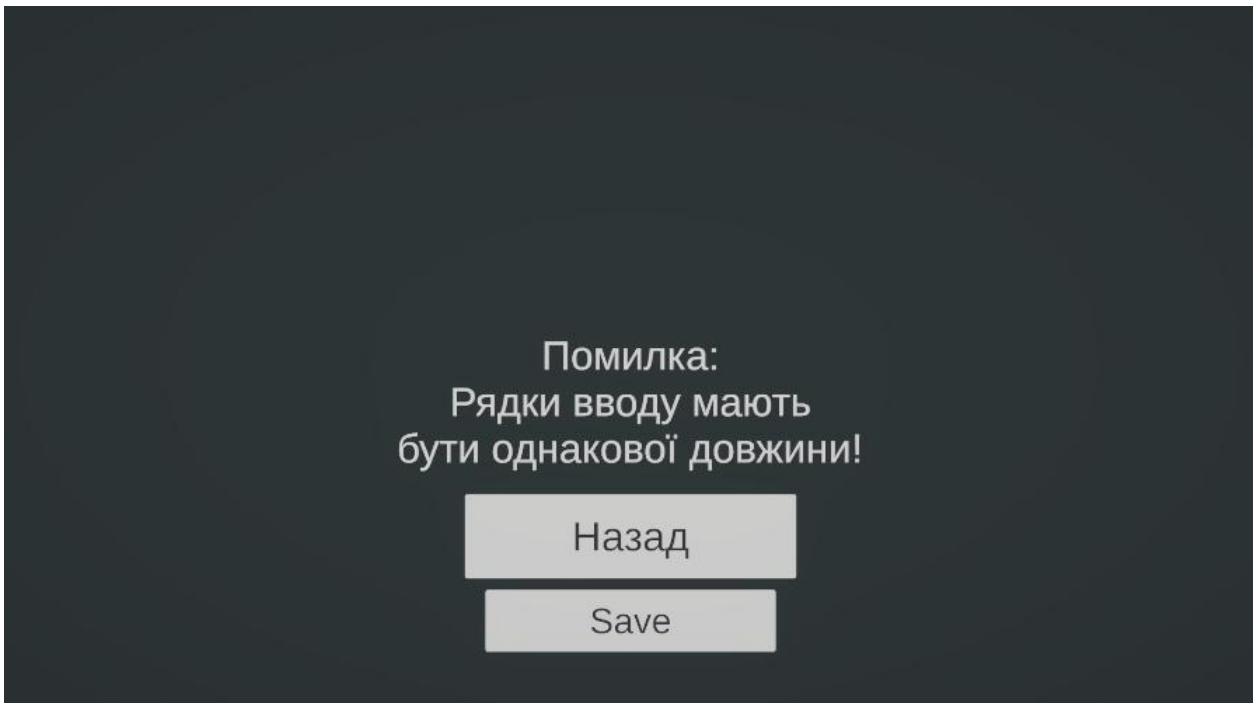

Рисунок 6 - Система **блокує** розрахунок і виводить повідомлення про помилку **error_length_mismatch**

Завдання 3

Для виконання завдання лабораторної роботи необхідно було змоделювати дефект. Для цього у методі OnButton_Calculate_Click було закоментовано блок коду, який відповідає за перевірку рівності довжин вхідних рядків (Bug Injection). Це дозволило пропустити некоректні дані далі в логіку обчислень.

```
0 references
public void OnButton_Calculate_Click()
{
    string inputText = inputField_1.text;
    SignalData inputData = new SignalData();

    string[] lines = inputText.Split(new[] { '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries);

    if (currentTriggerType == "RS" || currentTriggerType == "JK")
    {
        if (lines.Length < 2)
        {
            ShowError("error_rsjk_lines");
            return;
        }
        //if (lines[0].Trim().Length != lines[1].Trim().Length)
        //{
        //    ShowError("error_length_mismatch");
        //    return;
        //}

        inputData.inputSequence1 = ConvertStringToBoolArray(lines[0].Trim());
        inputData.inputSequence2 = ConvertStringToBoolArray(lines[1].Trim());
    }
```

Рисунок 7 - Створення штучної помилки (Bug Injection)

Процес відлагодження (Debugging) виконувався за допомогою інтеграції Visual Studio та Unity:

1. Встановлено **точку зупинки (Breakpoint)** на етапі парсингу даних.

2. За допомогою інспектора змінних (Locals) зафіксовано, що масиви inputSequence1 та inputSequence2 мають різну довжину (5 та 2 елементи відповідно), що є недопустимим для JK-тригера.

3. При продовженні виконання програми (Step Over/Into) відбувся виклик методу розрахунку, який завершився критичною помилкою System.IndexOutOfRangeException, оскільки цикл спробував звернутися до індексу, якого не існує в коротшому масиві.

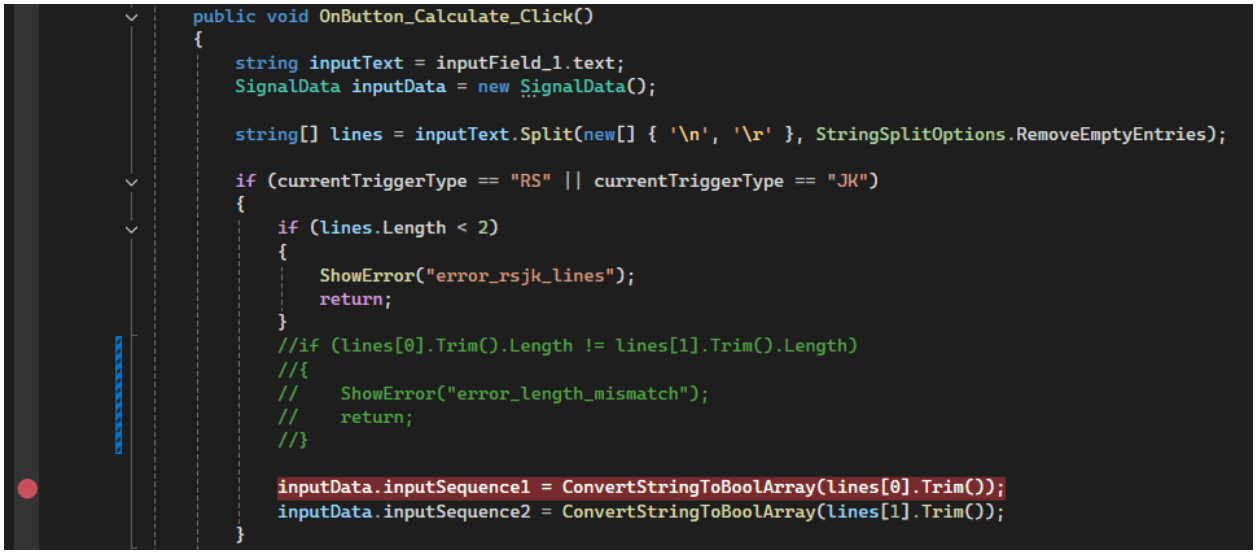

Рисунок 8 - Додавання точки зупинки (Breakpoint)

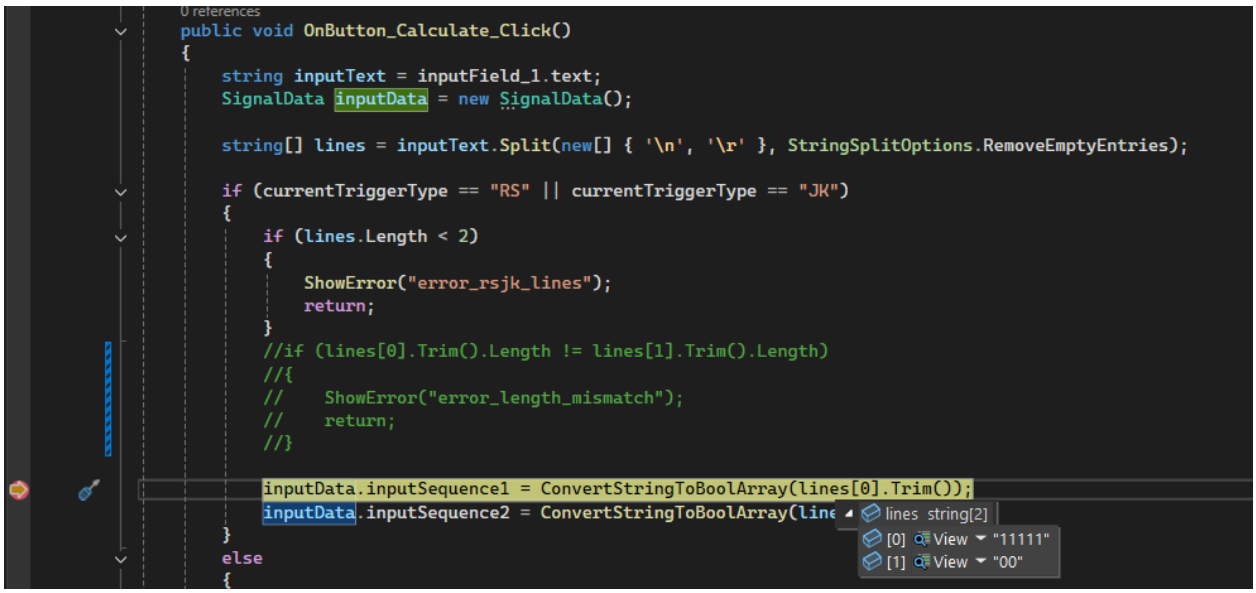

Рисунок 9 - У масиві є два рядки різної довжини

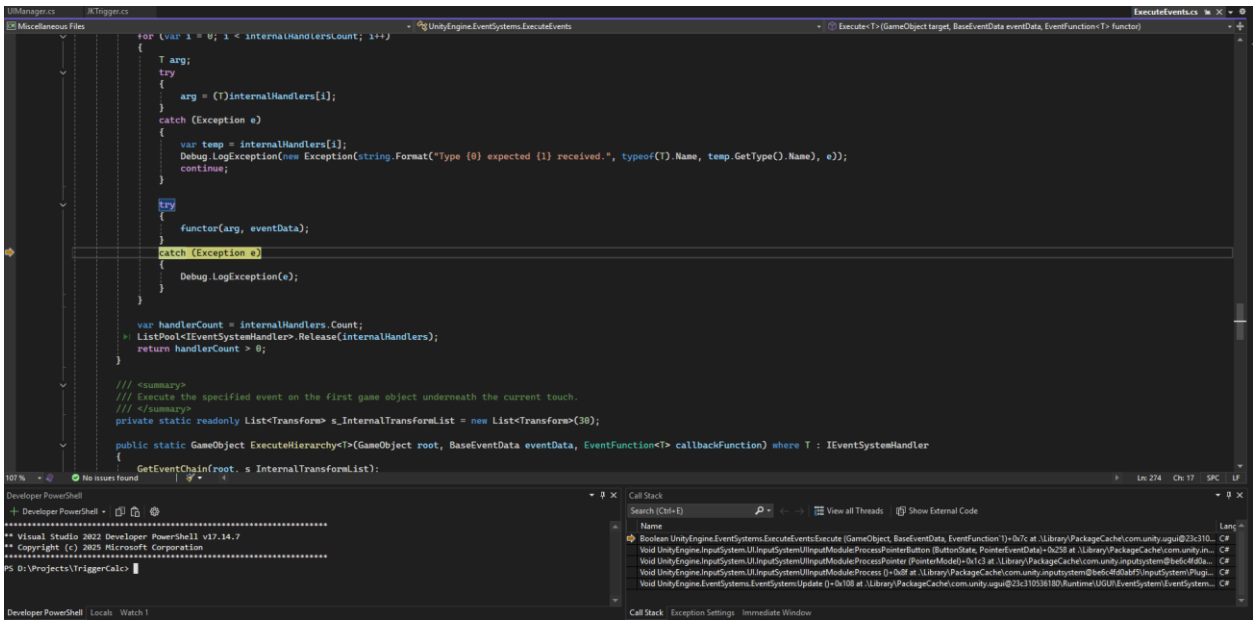

Рисунок 10 - **Exception Unhandled: IndexOutOfRangeException**

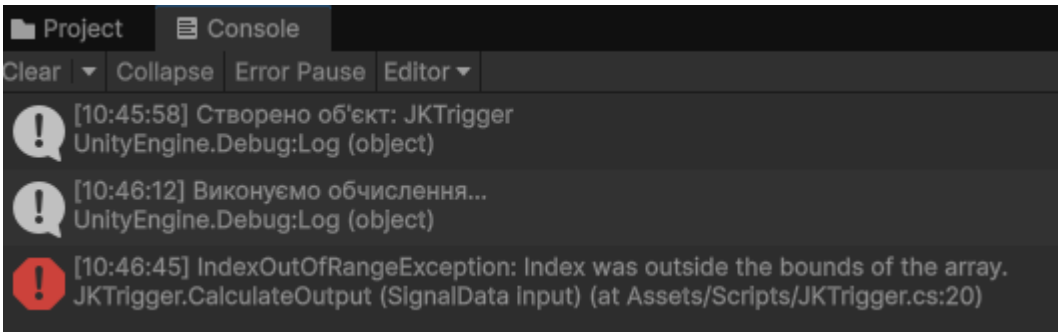

Рисунок 11 - Помилка

Завдання 4

Виявлений дефект було офіційно задокументовано у системі GitHub Issues. Картка містить повну інформацію для відтворення багу: опис проблеми, пріоритет (High), очікуваний та фактичний результати. Ця картка була автоматично додана до Kanban-дошки проєкту в колонку "Todo".

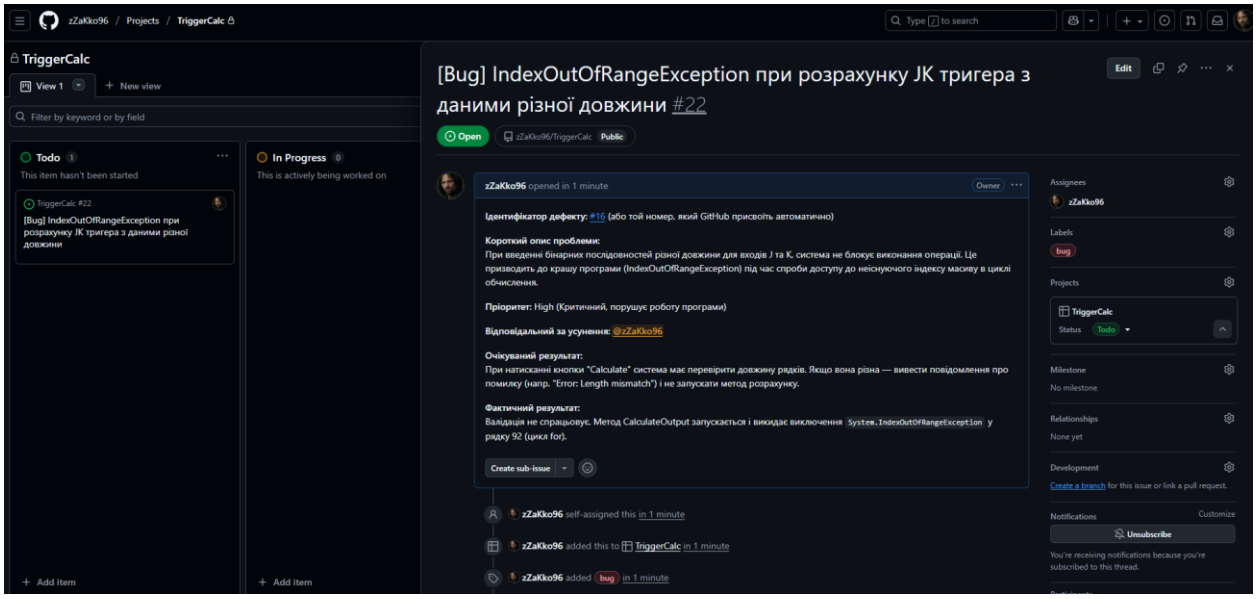

Рисунок 12 – Баг задокументовано

Завдання 5

Після відновлення коду валідації (розкоментування блоку перевірки) помилку було виправлено. Зміни зафіксовано у системі контролю версій Git. У повідомленні коміту використано ключову фразу Closes #22, що дозволило автоматично закрити Issue на GitHub та перемістити відповідну картку на Kanban-дошці у статус "Done".

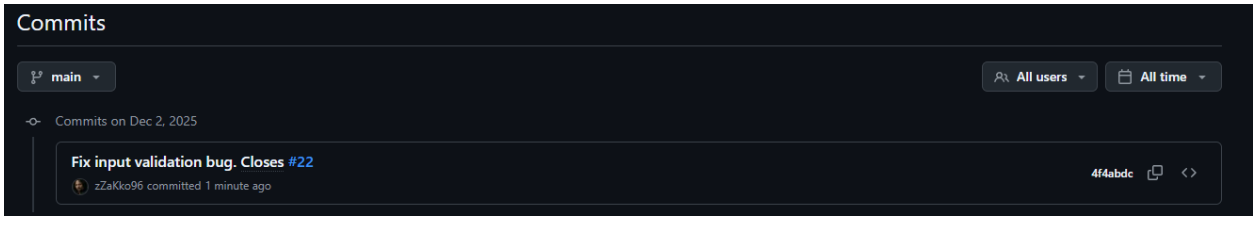

Рисунок 13 – Коміт з фіксом багу

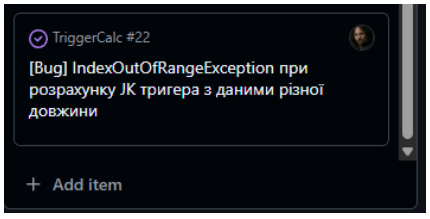

Рисунок 14 – Закритий таск

Висновки

У ході виконання лабораторної роботи №5 було успішно опановано методи та інструменти відлагодження (debugging) та тестування програмного забезпечення на прикладі проєкту «TriggerCalc».

Під час роботи було виконано наступні етапи:

1. **Планування тестування:** На основі вимог до ПЗ складено детальний чеклист та розроблено тестові сценарії (Test Cases) для перевірки валідації вхідних даних. Це дозволило структурувати процес перевірки якості коду.

2. **Робота з відлагоджувачем:** Отримано практичні навички роботи з Unity Debugger у середовищі Visual Studio. Використання точок зупинки (breakpoints) та покрокового виконання (step-by-step execution) дозволило проаналізувати стан змінних у реальному часі та виявити причину виникнення виключення IndexOutOfRangeException.

3. **Життєвий цикл дефекту:** Відпрацьовано повний цикл роботи з помилкою:
    - Штучне створення дефекту (Bug Injection) для навчальних цілей.
    - Документування багу в **GitHub Issues** із зазначенням пріоритету та кроків відтворення.
    - Виправлення коду та верифікація рішення.

4. **Автоматизація процесів Git:** Закріплено навички використання "розумних комітів" (Smart Commits). Використання команди Closes #ID продемонструвало ефективну інтеграцію між кодом та системою управління проєктами (GitHub Projects), де картка автоматично змінила статус на "Done".

Таким чином, лабораторна робота дозволила не лише перевірити працездатність розробленого ПЗ, а й на практиці застосувати професійні підходи до забезпечення якості (QA) та супроводу програмного коду.