Eric Gathinji

Programming in C# CST-150-0500

Grand Canyon University

13th July 2025

Activity 5 part 2 & 3

Github link: https://github.com/Ericgathinji444/GCU

Video link: https://youtu.be/_FCD9Dap5NY?si=x6Dwn_zQc2Pbd7Wt

FLOWCHART

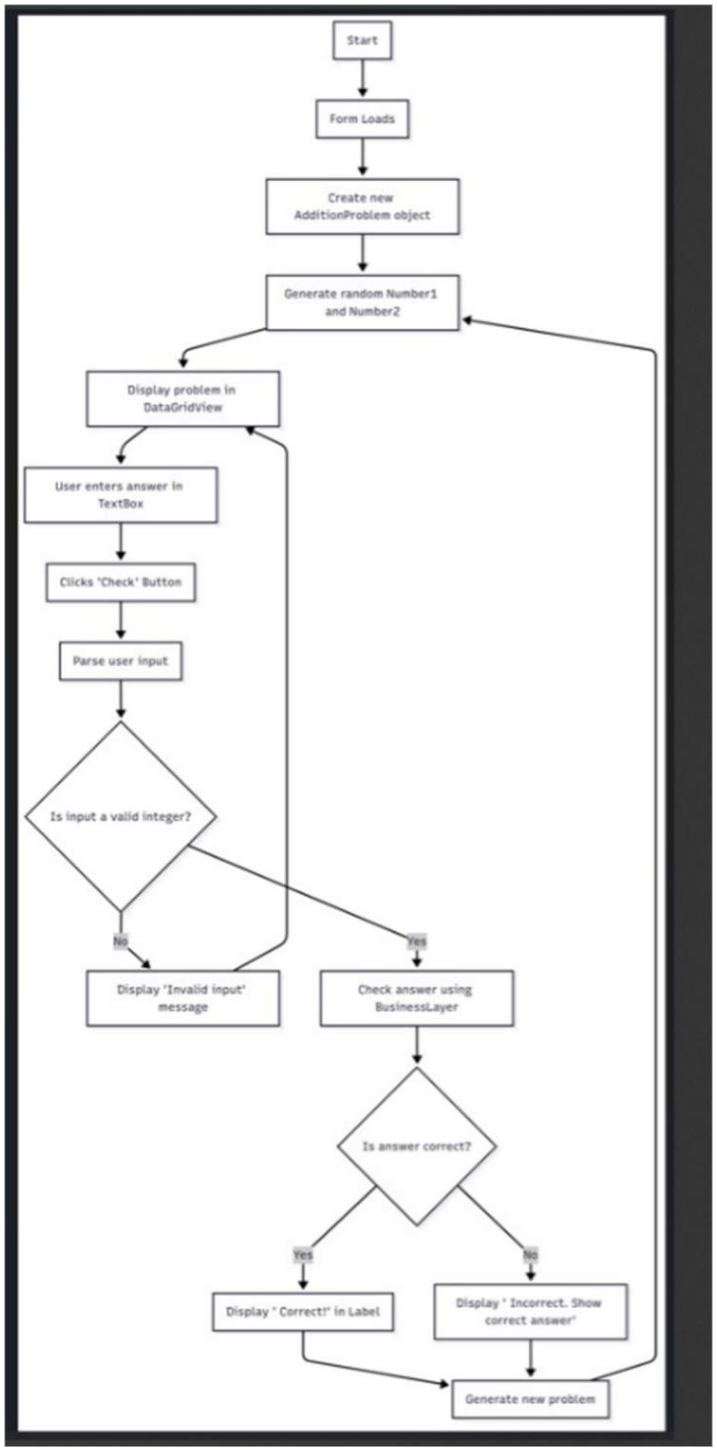
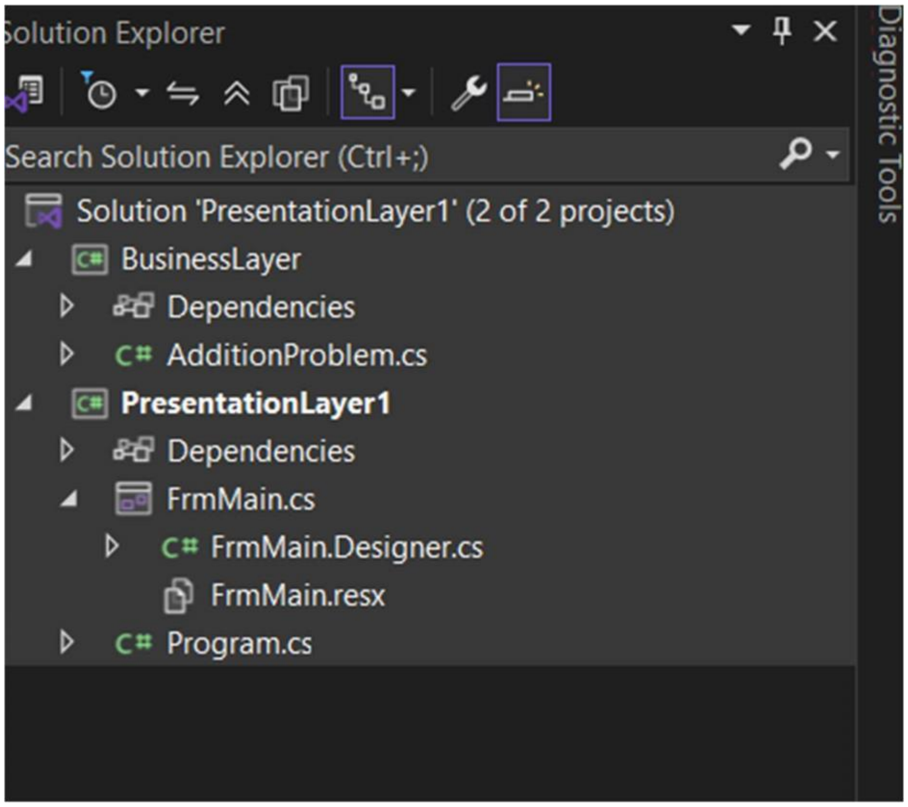

Start
Form Loads
Create new AdditionProblem object
Generate random Number1 and Number2
Display problem in DataGridView
User enters answer in TextBox
Clicks 'Check' Button
Parse user input
Is input a valid integer?
No
Yes
Display 'Invalid input' message
Check answer using BusinessLayer
Is answer correct?
Yes
No
Display " Correct!" in Label
Display " Incorrect. Show correct answer"
Generate new problem

Figure 1:Flowchart showing the control flow of the Addi on Tutor application.

“The flowchart shows how the user interface interacts with the business logic class. On form load, an Addi onProblem object is created and random numbers are generated.

After the user submits an answer, the app checks if the input is valid, then verifies the answer. Feedback is shown accordingly, and a new problem is generated.”

SCREENSHOT OF SOLUTION EXPLORER

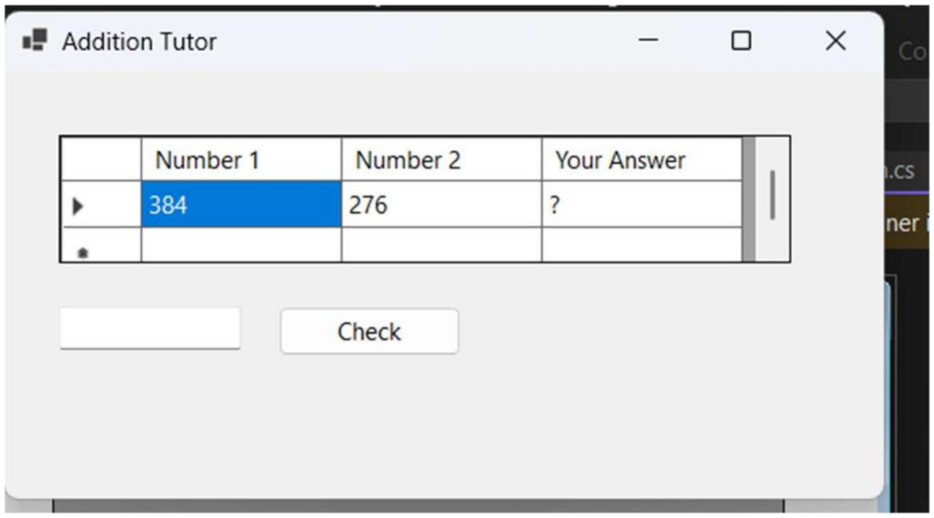


```
Solution 'PresentationLayer1' (2 of 2 projects)
◢ BusinessLayer
    ▷ Dependencies
    ▷ AdditionProblem.cs
◢ PresentationLayer1
    ▷ Dependencies
    ◢ FrmMain.cs
        ▷ FrmMain.Designer.cs
          FrmMain.resx
    ▷ Program.cs
```

Figure 2: Solution on Explorer showing the N-layer project structure.

The screenshot of the Solu on Explorer shows two projects: Presenta onLayer and BusinessLayer. The form resides in the presentation layer, while all processing logic resides in the business layer. The presentation layer references the business layer to access the logic it needs.

SCREENSHOT OF FORM BEFORE BEING POPULATED



	Number 1	Number 2	Your Answer
▸	384	276	?
*			

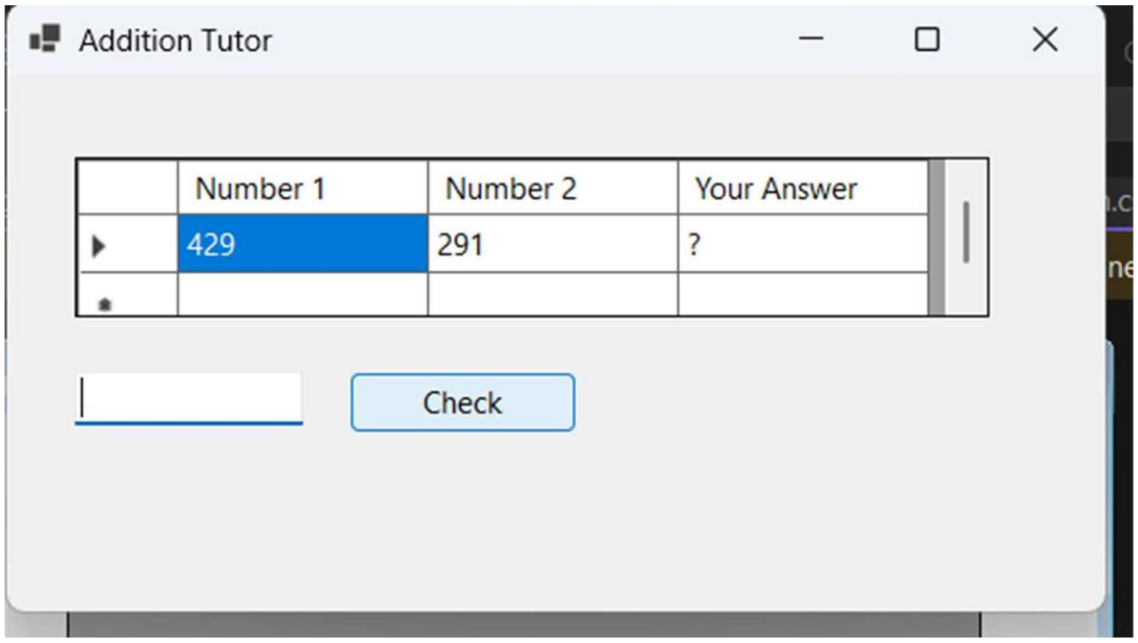
Figure 3: Form ready to display the first math problem.

This shows the state of the form when the application starts.

A DataGridView is set up to display the random numbers and the user's answer.

The text box is empty, waiting for user input.

SCREENSHOT AFTER BEING POPULATED



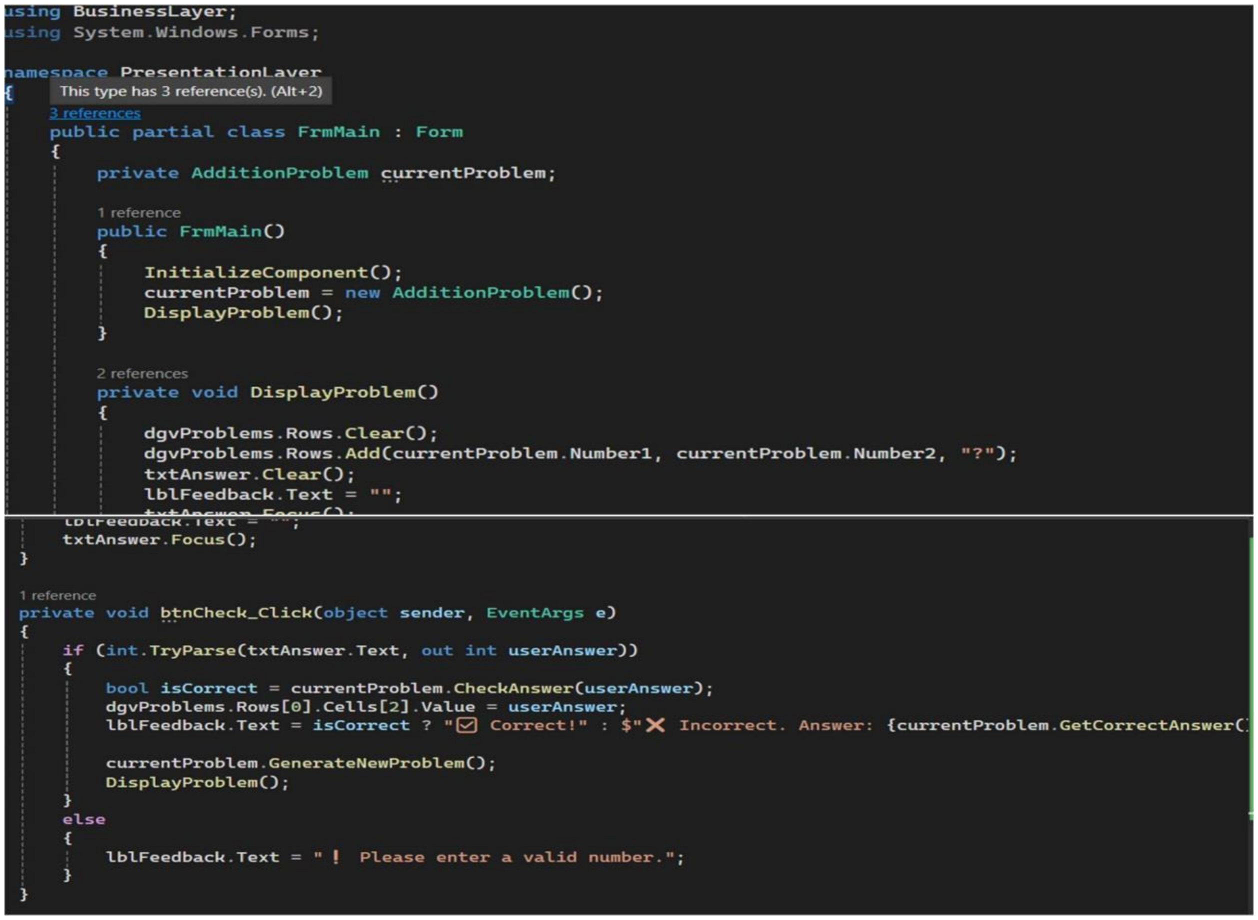
	Number 1	Number 2	Your Answer
▸	429	291	?
*			

Figure 4: Form after the user submits an answer.

This shows the form after the user has entered an answer and clicked the ‘Check’ button.

The feedback label displays whether the answer was correct or incorrect, and the problem updates with new random numbers.

SCREENSHOT BEHIND FrmMain



```
using BusinessLayer;
using System.Windows.Forms;

namespace PresentationLayer
{    This type has 3 reference(s). (Alt+2)
    3 references
    public partial class FrmMain : Form
    {
        private AdditionProblem currentProblem;

        1 reference
        public FrmMain()
        {
            InitializeComponent();
            currentProblem = new AdditionProblem();
            DisplayProblem();
        }

        2 references
        private void DisplayProblem()
        {
            dgvProblems.Rows.Clear();
            dgvProblems.Rows.Add(currentProblem.Number1, currentProblem.Number2, "?");
            txtAnswer.Clear();
            lblFeedback.Text = "";
```

```
    lblFeedback.Text = "";
    txtAnswer.Focus();
}

1 reference
private void btnCheck_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtAnswer.Text, out int userAnswer))
    {
        bool isCorrect = currentProblem.CheckAnswer(userAnswer);
        dgvProblems.Rows[0].Cells[2].Value = userAnswer;
        lblFeedback.Text = isCorrect ? "☑ Correct!" : $"✕ Incorrect. Answer: {currentProblem.GetCorrectAnswer(

        currentProblem.GenerateNewProblem();
        DisplayProblem();
    }
    else
    {
        lblFeedback.Text = "❗ Please enter a valid number.";
    }
}
```

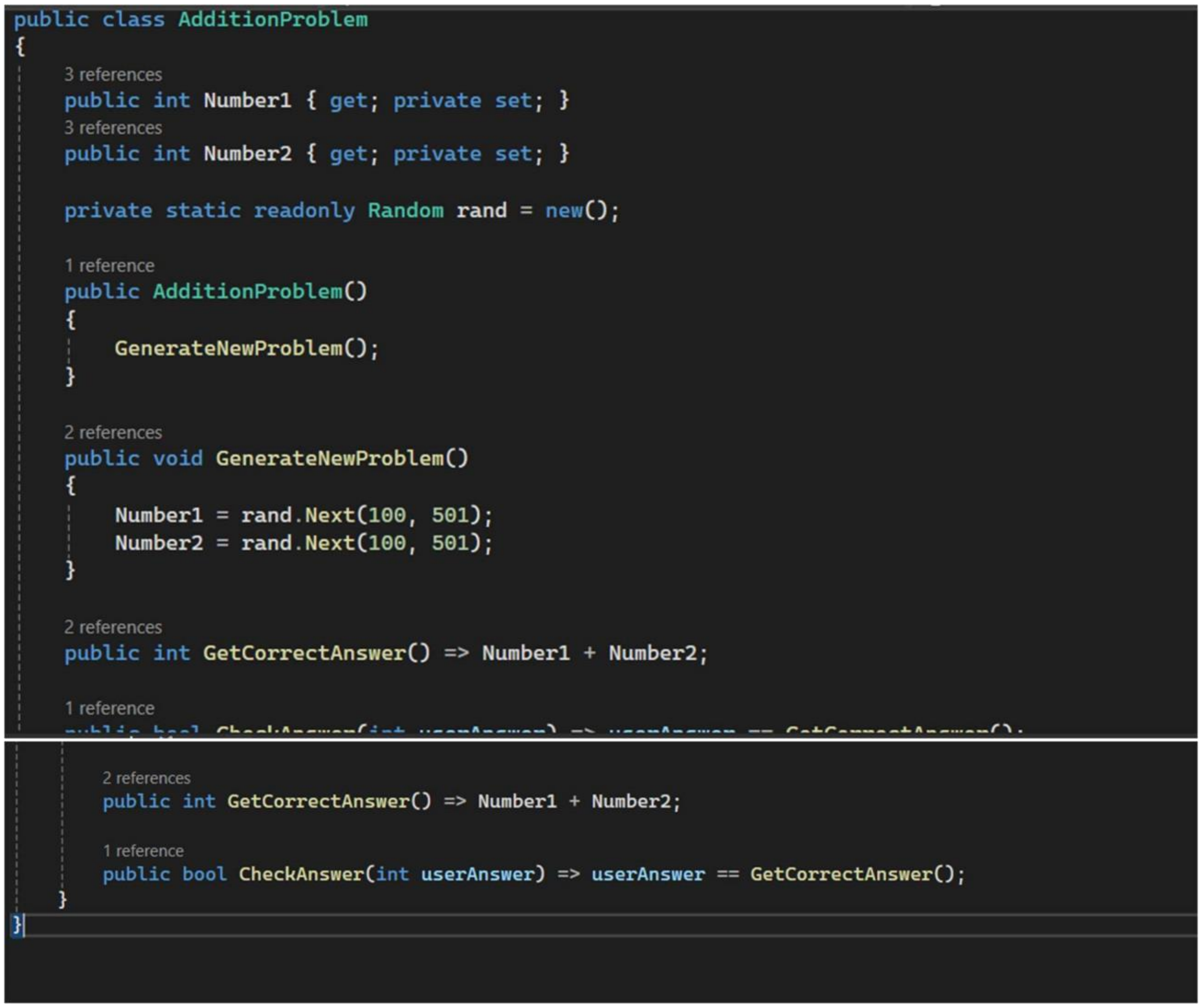
Figure 5:FrmMain.cs showing UI logic calling business layer.

This code handles the UI logic only.

It creates an instance of the business logic class, calls methods to generate problems, and updates the DataGridView.

This design keeps the UI clean and maintainable.

SCREENSHOT OF CODE FOR DICE CLASS



```
public class AdditionProblem
{
    3 references
    public int Number1 { get; private set; }
    3 references
    public int Number2 { get; private set; }

    private static readonly Random rand = new();

    1 reference
    public AdditionProblem()
    {
        GenerateNewProblem();
    }

    2 references
    public void GenerateNewProblem()
    {
        Number1 = rand.Next(100, 501);
        Number2 = rand.Next(100, 501);
    }

    2 references
    public int GetCorrectAnswer() => Number1 + Number2;

    1 reference
```

```
        2 references
        public int GetCorrectAnswer() => Number1 + Number2;

        1 reference
        public bool CheckAnswer(int userAnswer) => userAnswer == GetCorrectAnswer();
    }
}
```

Figure 6: Addition Problem.cs showing encapsulated logic.

This class lives in the business layer and encapsulates all the logic for generating random numbers and checking answers.

The form does not know how the logic works internally it

simply uses this class like a tool.

PART 3

Flowchart

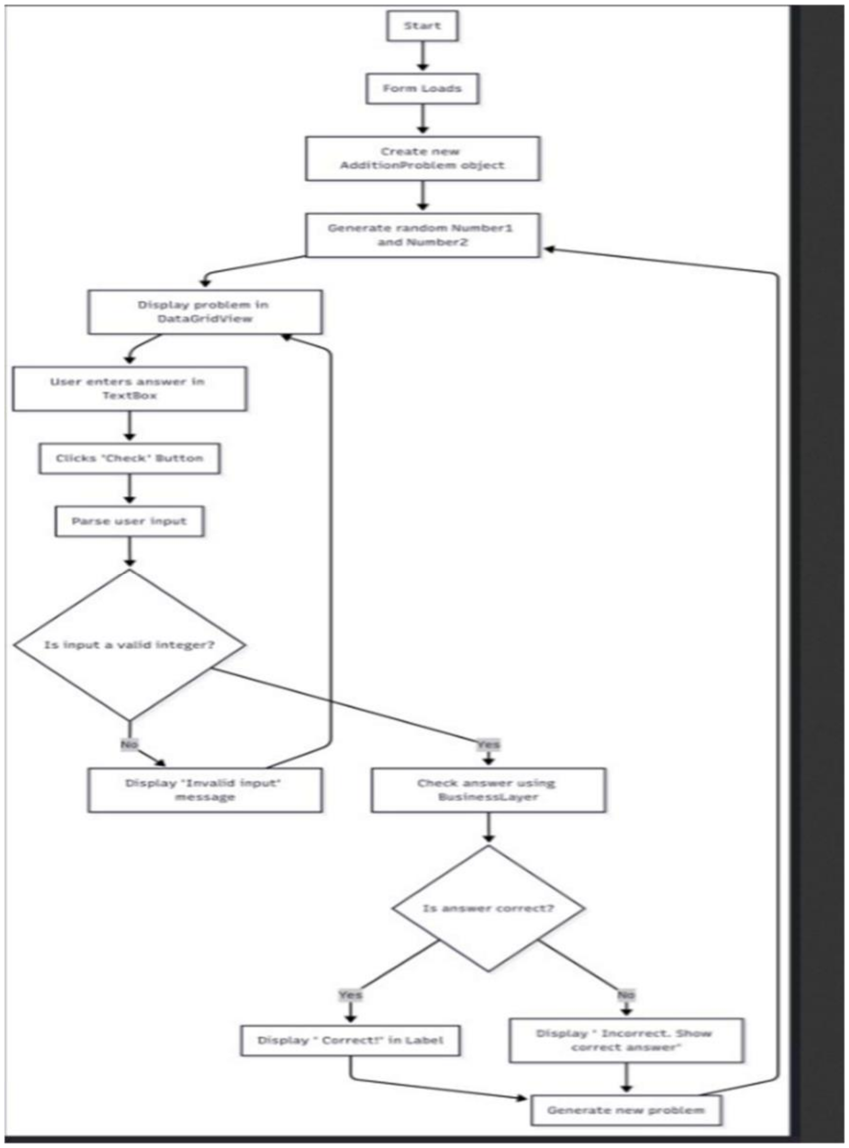

Start
Form Loads
Create new AdditionProblem object
Generate random Number1 and Number2
Display problem in DataGridView
User enters answer in TextBox
Clicks "Check" Button
Parse user input
Is input a valid integer?
No
Yes
Display "Invalid input" message
Check answer using BusinessLayer
Is answer correct?
Yes
No
Display " Correct!" in Label
Display " Incorrect. Show correct answer"
Generate new problem

Figure 7:Flowchart summarizing of activity 5

This flowchart is divided into two subgraphs: one for the inventory system and the other for the addition tutor.

Both apps follow event-driven flow, using business layer logic to process data and send output back to the form. This highlights the clean separation of concerns provided by the N-layer design.”

Follow-up questions

1. What was challenging? Maintaining a clean design while providing appropriate connection between the presentation and business levels
2. What did you learn? I learned how crucial it is to divide up issues in application design to improve scalability and maintainability.
3. How would you improve the project? By adding more reliable error-handling and user-feedback systems, I would make the project better.
4. How can you use what you learned on the job? I can create more effective and maintainable software solutions for the workplace by utilizing the concepts of N-layer architecture and clean code practices.

ADD ON

Monday 7th July

Start: 8:00 am End: 2:00 pm Activity: 5 part 1

Tuesday8 th July

Start: 8:00 End: 2:30 pm Activity: 5 part 2 &3

Wednesday 9th July

Start:2:00 am End: 4:00 am Activity: 5 part 1

Saturday 12th July

Start 12:00 am End: 4:00 am: Activity 5 part 2 & 3