**About FallingLetters…**

The FallingLetters is a game where random keys start falling down from the sky and the players have to press the correct key that belongs to the falling letter before it hits the spikes on the ground and gets destroyed.

Each player gets 3 main health points in the game. That means each player is allowed to make 3 mistakes and to allow falling letters to reach spikes 3 times.

However, the users can extend their health points to 6 by playing the banana game. After the main 3 health points get destroyed after a player makes 3 mistakes, upon making the next (4th) mistake, the player will be directed to the banana game. There, each player gets to answer the mathematical problem presented in the banana game within 10 seconds. If the player successfully answers the banana problem, they will be allowed to continue the game from where they stopped. Players can play the banana game for a maximum of three times for three mistakes they make after getting all the first 3 heart points destroyed.

Incase if the player provides a wrong answer or if the timer in the banana game runs out, the game will be over, and the player will be directed to the game over screen. The game over also occurs when a player makes a mistake after the player runs out of all three initial health points and all three banana attempts.

The level of the banana game will rise starting from score 20 for each score the player gets that’s divisible by 10 (30,40,50…). When the level goes up, the gravity of each falling letter increases. Hence increasing the difficulty.

**System Structure**

3 tier architecture is followed.

1. **Client**

C# code that is used in unity engine. Directly communicates with the server side and the banana API. Main game is run on this code. It contains 41 classes. Follows MVC (Model, View, Controller) architecture. Classes belonging to View are also separated into 8 folders based on the Unity Scene they work on. Model contains another folder known as intermediate which contains classes used for data conversions between C# objects and JSON.

1. **Server**

Python FastAPI server. Directly communicates with the client side and the database. Intermediary between client and the database. Separated into 6 different python files based on responsibility.

1. **Database**

Firebase database that directly only communicated with the server and also responsible for sending reset password emails to users.

**Functions in FallingLetters**

1. **Register**

User inputs will be validated upon entry and users will be only allowed to register if they enter valid inputs. Feedback from the server will be shown to users. (If the register was successful or not.)

1. **Login.**

User inputs will be validated upon entry and users will be only allowed to login if they enter valid inputs. Feedback from the server will be shown to users. (If the login was successful or not.)

1. **Reset Password.**

User inputs will be validated upon entry and users will only be able to reset password if they enter valid inputs (email). The firebase will send a password reset link to the user’s email.

1. **Leaderboard**

Displays all the users and their high scores along with their rank.

1. **FallingLetters game**

Main Game.

1. **Banana game**

Game to be played to extend the number of attempts within a single game.

1. **Score update**

When a user achieves a higher score their details in the database will be updated.

1. **Internet Checking**

Internet connectivity is checked in every scene of the game to assure flawless experience. Achieved by using a ping request.

1. **Error Handling**

Errors can occur during each communication and all these errors are handled to assure data consistency and good user experience.

**External resources used to develop FallingLetters game**

1. **C# Naming rules**

To make the code consistent by following naming rules.

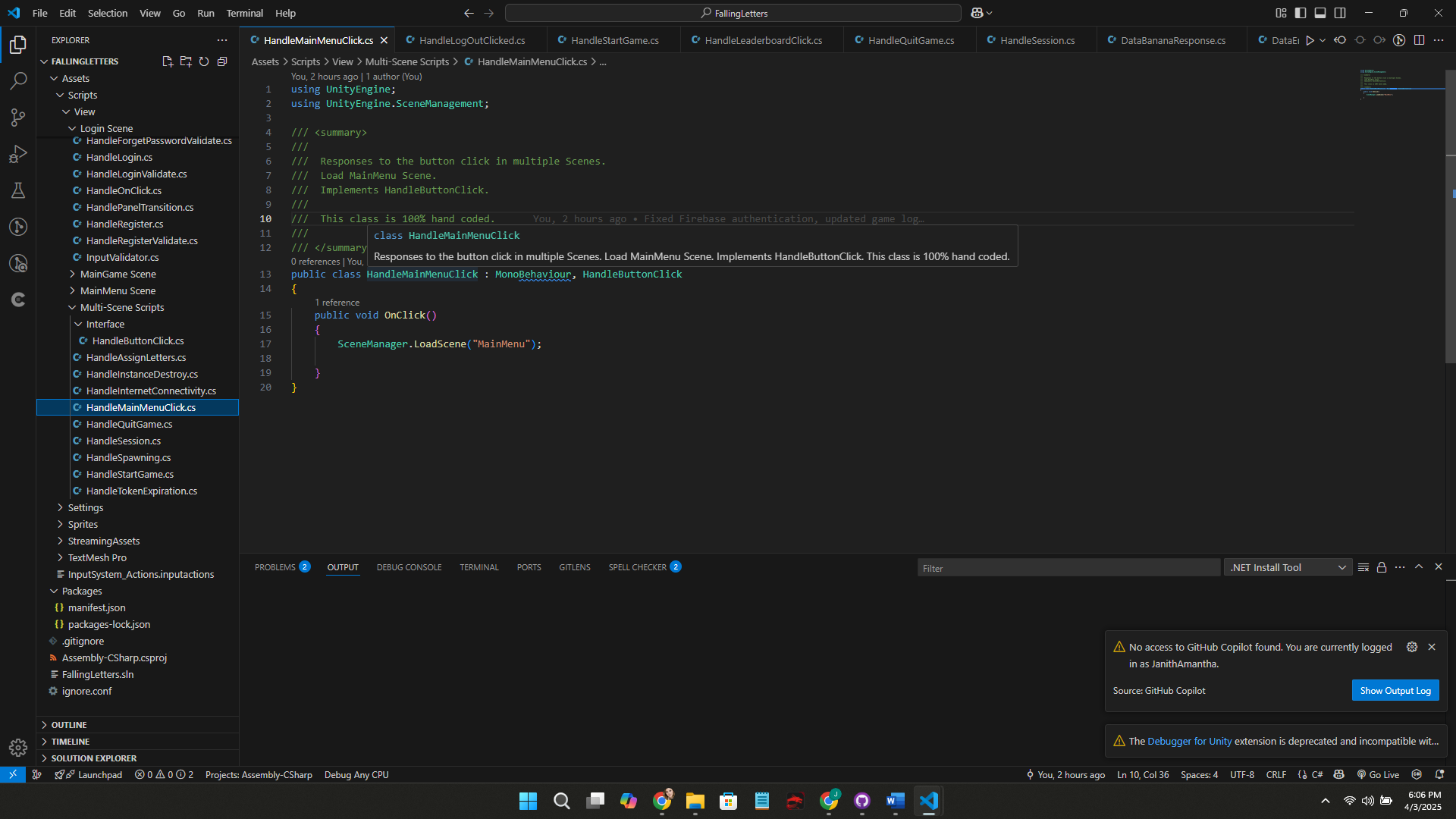
A screenshot of a computer

AI-generated content may be incorrect.

All the private variables start with “\_” while public attributes (Attributes are used as getters and setters instead of methods) and method starts with an Uppercase letter

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names>

1. **C# internal documentation**

****

Hovering over a class name or a method name will provide a handwritten description of the class or the method.

To document C# components such as classes and methods to briefly describe their responsibilities within the game.

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/documentation-comments>

1. **Python internal Documentation**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Hovering over a class name or a method name will provide a handwritten description of the class or the method.

To document Python components such as classes and functions to briefly describe their responsibilities within the game.

<https://peps.python.org/pep-0257/>

1. **Large Language Models (Code generators)**
   1. **ChatGPT**

Code generation and information.

<https://chatgpt.com/>

All the ChatGPT generated code lines are mentioned within the internal documentation (source code).

* 1. **Grok**

Problem analysis and Information.

<https://grok.com/>

1. **Stackoverflow**

To find solutions to certain problems (Firebase SDK not working…). No code lines are copied from it.

<https://stackoverflow.com/>

1. **Version Control**
   1. **Unity version Control**

Version control system provided by Unity.

<https://unity.com/solutions/version-control>

* 1. **GitHub**

GitHub is used to publish the system in a public repository after development.

<https://github.com/>

1. **YouTube** 
   1. **Unity Coding tips**

To understand how to arrange the unity code.

<https://www.youtube.com/watch?v=dLCLqEkbGEQ&t=941s>

1. **Udemy** 
   1. **Unity tutorial**

To learn how to use Unity Engine.

<https://www.udemy.com/course/learn-unity-and-c-sharp-programming-by-making-a-simple-2d-ga/>

1. **Students’ Knowledge**

A student in my batch gave his feedback on my system and advised me to separate the game into 3 tiers (client, server, database) rather than directly accessing the database from the client side.

**Nifadh Mohamed Nizar (2432947)** [BSc (Hons) Computer Science at University of Bedfordshire]