**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**

**COURSE NAME: IT PROJECT**

**-----🙞🙞🙞🙞🙞-----**

****

**IT PROJECT REPORT**

**Project name:** Guess Number Game

**COURSE CODE:** *232PROJ215879E*

**SEMESTER:** 2

**ACADEMIC YEAR:** 2023-2024

LIST OF STUDENTS

**PROJECT TEAM**

SEMESTER 2 ACADEMIC YEAR 2023-2024

|  |  |
| --- | --- |
| **ID** | **Full name** |
| 21110811 | Võ Hoàng Tùng |

**Professor’s comment**

Ho Chi Minh city, May …, 2024

Grading

**CONTENTS**

[**I.** **PROJECT DESCRIPTION** 1](#_Toc166326676)

[**1.** **Introduction** 1](#_Toc166326677)

[**2.** **What is the software used for?** 1](#_Toc166326678)

[**3.** **Input data** 1](#_Toc166326679)

[**4.** **User Interface** 2](#_Toc166326680)

[**II.** **TASK ASSIGNMENT** 2](#_Toc166326681)

[**III.** **DESIGN** 3](#_Toc166326682)

[**1.** **Classes catalog** 3](#_Toc166326683)

[**2.** **Methods in RoundedButton class** 3](#_Toc166326684)

[**3.** **Methods in GuessingGame class** 3](#_Toc166326685)

[**4.** **Methods in Form1 class** 5](#_Toc166326686)

[**IV.** **TEST CASE** 7](#_Toc166326687)

[**V.** **CONCLUDE** 10](#_Toc166326688)

[**VI.** **Reference** 10](#_Toc166326689)

1. **PROJECT DESCRIPTION**
2. **Introduction**

The **Guess the Number** game is a simple mind puzzle game suitable for all ages. Players will take turns entering numbers, the computer will compare them with the secret number that was chosen beforehand and give hints like "higher", "lower" or "congratulations, you guessed right!". The goal of the game is to find the secret number in as few guesses as possible.

1. **What is the software used for?**

The **Guess the Number** can be used in a variety of contexts, including:

* **Entertainment:** This is the primary purpose of the software. Guess the Number is a simple but fun game that is suitable for all ages and can provide moments of relaxation and entertainment after hours of study and work.
* **Education:** Guess the Number can be used as an educational tool to help students train their logical thinking, concentration, and reflexes. For example, teachers can use the game in the classroom to reinforce knowledge of math or to teach students how to use the binary search algorithm.
* **Training:** Guess the Number can also be used in training programs to help trainees develop problem-solving and critical thinking skills. For example, companies can use the game in training sessions to help employees train their ability to make quick and accurate decisions in conditions of information scarcity.
* **Research:** Guess the Number can be used in research on artificial intelligence and machine learning. For example, researchers can use the game to evaluate the effectiveness of new search algorithms or to develop artificial intelligence systems that can learn and adapt to new environments.

1. **Input data**

Input data: The number entered by the player.

1. **User Interface**

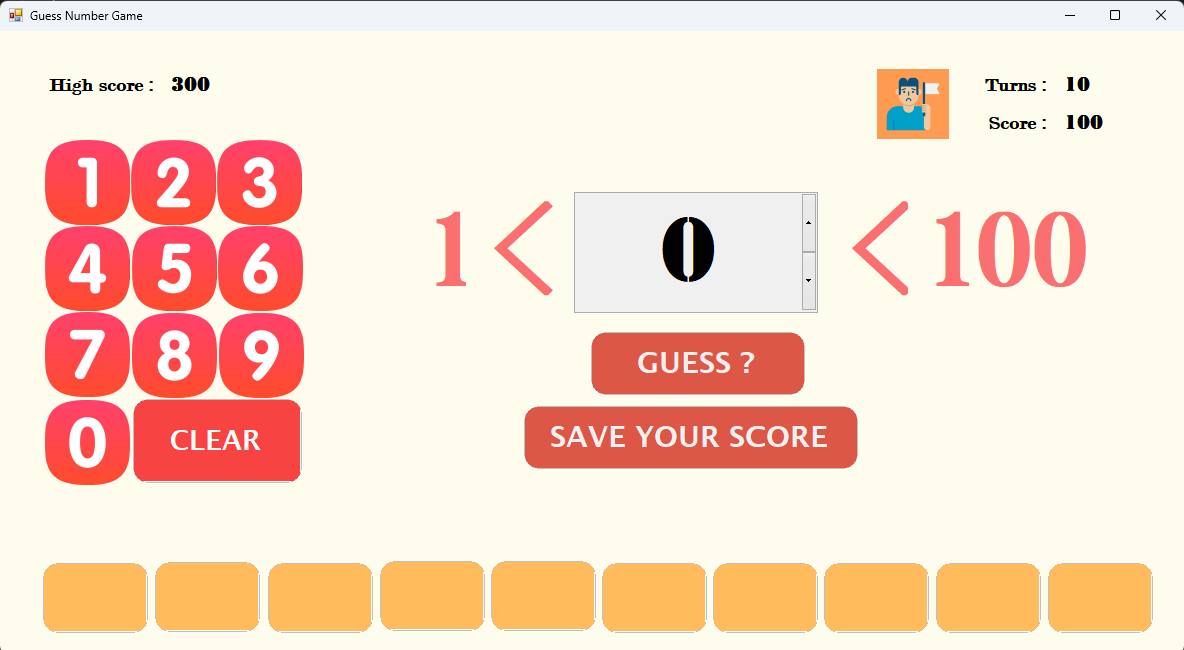


Figure 1 User Interface

1. **TASK ASSIGNMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Name of student | Task | Contribution percentage |
| 1 | Võ Hoàng Tùng | Analyze the request | 100% |
| 2 | Võ Hoàng Tùng | Game design | 100% |
| 3 | Võ Hoàng Tùng | Game programming | 100% |
| 4 | Võ Hoàng Tùng | Complete and deploy | 100% |

1. **DESIGN**
2. **Classes catalog**

Student in charge: Võ Hoàng Tùng

|  |  |  |
| --- | --- | --- |
| Order | Name of class | Purpose |
| 1 | RoundedButton | Button design with rounded corner  Source: [Rounded edges in button C# (WinForms) - Stack Overflow](https://stackoverflow.com/questions/28486521/rounded-edges-in-button-c-sharp-winforms) |
| 2 | GuessingGame | To encapsulate the core logic of the Guess the Number game. |
| 3 | Form1 | To handle the UI and interact with the GuessingGame class to create the Guess the Number game experience. |

1. **Methods in RoundedButton class**

Student in charge: Võ Hoàng Tùng

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Method | Purpose | File name, line number containing declaration |
| 1 | GetRoundPath(RectangleF Rect, int radius) | To create a GraphicsPath object that defines the shape of a rounded rectangle. | RoundedButton.cs (14) |
| 2 | OnPaint() | Drawing the visual appearance of the button, including its rounded corners. | RoundedButton.cs (34) |

1. **Methods in GuessingGame class**

Student in charge: Võ Hoàng Tùng

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Method | Purpose | File name, line number containing declaration |
| 1 | GuessingGame() | Initializes a new game instance. Sets the initial number of turns (turns to 10) and calls the GetRandomNumber method to generate a secret number to be guessed (numNeedToGuess). | GuessingGame.cs (11) |
| 2 | CheckValidAnswer(int answer) | Validates if the provided user guess (answer) is within the allowed range (1-100). Returns true if valid, false otherwise. | GuessingGame.cs (17) |
| 3 | CheckTheAnswer(int value, out bool isTooHigh) | Evaluates the player's guess (value). Sets the isTooHigh output parameter to true if the guess is higher than the secret number, false otherwise. Returns true if the guess is correct, false otherwise. It also updates the game state:   * Increases score by the remaining turns if the guess is correct. * Decreases score and remaining turns (`turns`) for incorrect guesses. | GuessingGame.cs (19) |
| 4 | GetRandomNumber() | Generates a random integer between 1 and 100 | GuessingGame.cs (37) |
| 5 | AddScore(int score) | Updates the player's score | GuessingGame.cs (43) |
| 6 | GetRemainingTurns() | Returns the number of turns remaining for the player. | GuessingGame.cs (48) |
| 7 | NumNeedToGuess | Exposes the secret number to be guessed | GuessingGame.cs (49) |

1. **Methods in Form1 class**

Student in charge: Võ Hoàng Tùng

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Method | Purpose | File name, line number containing declaration |
| 1 | Form1\_Load() | Sets up the initial UI state | Form1.cs (35) |
| 2 | btnClear\_Click() | Clear the user input field in the number text box. | Form1.cs (44) |
| 3 | Numeric Pad Click events (pic1\_MouseClick, pic2\_MouseClick, ...) | Capture user input from the number pad buttons and update the number text box. | Form1.cs (50) |
| 4 | btnGuess\_Click() | * Validates user input and checks for repeated guesses. * Calls the CheckTheAnswer method from GuessingGame to evaluate the guess and update the game state (score, turns). * Displays messages based on the guess outcome (correct, too high/low, out of turns). * Updates the UI elements (turns, score) and potentially reveals the answer | Form1.cs (111) |
| 5 | resetAnswerButtons() | Reset the answered buttons in the game. | Form1.cs (177) |
| 6 | assignTheAnswer() | Assigns text to buttons to indicate previous guesses that were "too high" or "too low" compared to the secret number. | Form1.cs (185) |
| 7 | ResetGameState() | Resets the game state by clearing variables like number, guessedNumbers, and creating a new GuessingGame object. | Form1.cs (226) |
| 8 | ResetUI() | Resets the user interface by clearing text fields, hiding/showing buttons, and updating labels with information retrieved from the GuessingGame object. | Form1.cs (233) |
| 9 | playAgainBtn\_Click() | * Starts a new game by creating a new GuessingGame instance. * 9Resets the UI elements and game state. * Optionally allows the player to continue with the accumulated score if they haven't finished the previous game. | Form1.cs (244) |
| 10 | giveupbtn\_Click() | Reveals the secret number and allows the player to start a new game. | Form1.cs (255) |
| 11 | LoadHighScore() | Loading the high score from a text file | Form1.cs (262) |
| 12 | SaveScore() | Saving the current score to a text file | Form1.cs (276) |
| 13 | saveScorebtn() | To save the current game score and reset the game for a new round. | Form1.cs (290) |
| 14 | Form1\_FormClosing() | To ensure the current game score is saved before the application exits. | Form1.cs (300) |

1. **TEST CASE**

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Test case | Purpose | Explain decision |
| 1 | **Valid Guess - Win**   * Input: Player enters a number within the valid range (1-100) that matches the secret number. * Expected Result: * The game should identify the guess as correct. * A congratulatory message should be displayed. * The score should be updated based on the remaining turns (higher score for more remaining turns). * The game should end. | This test case verifies the game behaves correctly when the player enters a valid guess that matches the secret number. | This is a core functionality test. I want to verify the game can identify a correct guess, congratulate the player, update the score based on remaining turns, and end the game gracefully. |
| 2 | **Invalid Guess - Out of Range**   * Input: Player enters a number less than 1 or greater than 100. * Expected Result: * The game should display an appropriate error message indicating the number is outside the valid range. * The guess should not be processed. * The game should continue, allowing the player to enter a new guess. | This test case checks how the game handles invalid user input that falls outside the allowed range (numbers less than 1 or greater than 100). | This tests input validation. The game should prevent invalid user entries outside the allowed range (1-100) and provide informative error messages. |
| 3 | **Invalid Guess - Repeated Guess**   * Input: Player enters a number that they have already guessed previously. * Expected Result: * The game should display a message indicating the number has already been guessed. * The guess should not be processed. * The game should continue, allowing the player to enter a new guess. | This test case verifies the game prevents the player from entering a number that has already been guessed. | This ensures the game doesn't allow players to submit the same guess twice. It discourages repetitive guesses and encourages players to adjust their strategy. |
| 4 | **Incorrect Guess - Too High**   * Input: Player enters a number that is higher than the secret number. * Expected Result: * The game should indicate that the guess is "too high." * The score should be decremented by a penalty. * The remaining turns should be decreased by 1. * The game should continue, allowing the player to enter a new guess. | This test case checks how the game handles an incorrect guess where the player's guess is higher than the secret number. | These test cases verify how the game handles incorrect guesses in both directions (higher or lower than the secret number).  We want to ensure the game provides appropriate feedback ("too high" or "too low"), penalizes the score, and deducts a turn. |
| 5 | **Incorrect Guess - Too Low**   * Input: Player enters a number that is lower than the secret number. * Expected Result: * The game should indicate that the guess is "too low." * The score should be decremented by a penalty. * The remaining turns should be decreased by 1. * The game should continue, allowing the player to enter a new guess. | This test case checks how the game handles an incorrect guess where the player's guess is lower than the secret number. |
| 6 | **Out of Turns**   * Input: Player enters guesses until they have used all their allotted turns (10 in this case). * Expected Result: * The game should end after the last turn is used. * A message should be displayed indicating the player is out of turns. * The correct answer (secret number) should be revealed. * The final score should be displayed. | This test case verifies the game ends when the player runs out of turns. | This tests the game's behavior when the player exhausts their turn limit.  I want to verify the game ends, reveals the secret number, displays the final score, and allows the player to start a new game. |
| 7 | **Play Again**   * Input: Player clicks the "Play Again" button after finishing a game. * Expected Result: * A new game should be started. * The secret number should be regenerated. * The score should be reset to the initial value (100) unless the player continues with the accumulated score. * The remaining turns should be reset to 10. * The UI elements (number input, score, turns) should be updated to reflect the new game state. | This test case checks how the game behaves when the player chooses to play again after a win or loss. | This tests how the game restarts after a win or loss.  I want to ensure a new game begins with a regenerated secret number, potentially preserved score (optional), reset turns, and updated UI elements. |
| 8 | **Give Up**   * Input: Player clicks the "Give Up" button. * Expected Result: * The correct answer (secret number) should be revealed. * The game should end, allowing the player to start a new game. | This test case verifies the behavior when the player chooses to give up during the game. | This tests the functionality of the "Give Up" button.  We want to verify the game reveals the secret number, ends the current game, and allows the player to start a new one. |

1. **CONCLUDE**
2. **Assess the level of achievement of the goal**

Comply with the requirements: 100%

1. **Project development ideas**

* Add a scoring function: 100%
* Add high score function: 100%
* Add different difficulty levels: 0%

1. **Reference**

[Rounded edges in button C# (WinForms) - Stack Overflow](https://stackoverflow.com/questions/28486521/rounded-edges-in-button-c-sharp-winforms)