Project Report: Number Guessing Game

Project Overview

The **Number Guessing Game** is a simple Python program designed to engage users by having them guess a secret number within a specified range. The program randomly generates a secret number between 1 and 1000, and the user has up to 10 attempts to guess the correct number. After each guess, the program provides feedback to the user, telling them whether their guess was too high, too low, or correct.

The program incorporates error handling to ensure that the user inputs valid numbers and stays within the specified range. If the user guesses the correct number, they are congratulated, and the number of attempts it took them is displayed. If they fail to guess the number within the allowed attempts, the program reveals the secret number and ends the game.

Duration Taken to Complete the Project

The development of this project would typically take a few hours, depending on the programmer's familiarity with Python and the game's logic. A rough estimate for completion is:

- **Initial Setup**: 50 minutes – Setting up the project structure, defining the random number generation, and preparing for user input.
- **Coding and Debugging**: 2-3 hours – Implementing the core functions (like the get_guess and check_guess functions), as well as incorporating error handling and loops.
- **Testing and Refining**: 40 minutes – Ensuring that the game works as expected, such as validating the number input and checking edge cases (e.g., invalid input or out-of-range guesses).

The total time taken for the completion of the project would be around **4-5 hours**, considering testing and any adjustments required.

Outcome

The outcome of the project is a functional number guessing game where the user can attempt to guess a randomly generated number. The game provides the following features:

- **User-Friendly Input**: The program ensures that only valid numbers within the specified range (1-1000) are entered.
- **Feedback System**: After each guess, the game informs the user whether their guess is too high, too low, or correct.
- **Attempt Tracking**: The game tracks the number of attempts the user makes and ends when the user either guesses correctly or exhausts their attempts.
- **Error Handling**: The game gracefully handles incorrect inputs, ensuring smooth gameplay without crashing.

The game successfully meets the objective of providing an interactive and educational experience to users while demonstrating basic Python concepts such as loops, conditionals, and functions.

Challenges Faced

During the development of the game, several challenges were encountered, including:

1. **Input Validation**: Ensuring the user inputs a valid number within the range was challenging, especially when dealing with non-numeric inputs or out-of-range guesses. This was addressed by using a try-except block and ensuring that the input is both a valid integer and within the valid range.
2. **Attempt Management**: Handling the logic for the number of attempts was slightly tricky at first, especially when ensuring that the user could not continue guessing beyond the allowed number of attempts. This was managed using a simple counter that incremented with each guess.
3. **User Experience**: Making the user experience smooth and engaging required careful attention to the feedback messages. The messages had to be clear and concise so that the user would always understand the status of their guess.
4. **Testing**: While testing the code, ensuring that edge cases (like guessing the minimum or maximum number) were handled properly took some time. It was important to ensure that the feedback system worked correctly under all conditions.

Conclusion

In conclusion, the **Number Guessing Game** project is a simple but effective exercise in applying basic programming concepts such as random number generation, conditionals, and loops. The project was successfully completed within a few hours, and it provides an enjoyable interactive experience for users.

By tackling challenges such as input validation, attempt tracking, and ensuring smooth user interaction, the game works well and serves as an educational tool for learning Python.

The project could be expanded in the future to include additional features like difficulty levels (adjusting the range or the number of attempts), a graphical user interface (GUI), or storing high scores.

This project serves as a good starting point for beginners to get comfortable with Python while building something interactive and fun.