NUMBER GUESSING GAME

Mr. H . Sathish, Assistant Professor, Department of CSE

B. Jhansi, CH. Akshith , G. Karthik , M. Sanjay(UG Students, Department of CSE)

jhansibathini12@gmail.com, akshithch2004@gmail.com,

macharlasanjaygoud2004@gmail.com,karthikkoyalkar1@gmail.com.

Christu Jyothi Institute of Technology and Science, Telangana, India

1. Abstract

This paper focuses on implements an interactive Number Guessing Game using the Tkinter GUI toolkit. The game allows users to specify the number of digits in a hidden number and the maximum number of guesses allowed. The target number is randomly generated using nonzero digits, and players must guess individual digits within the set limit. Key features include: Customizable game settings (number of digits and attempts). Live feedback on guesses with a visual representation of progress. Score system starting from 100 points, deducting 10 points per incorrect guess .Dynamic themes for aesthetic customization. Interactive UI elements like blinking labels, input validation, animated widget shaking on invalid input, and a progress bar. Win/Loss detection with score display and a "Play Again" option. The game offers a visually engaging and user-friendly experience while demonstrating foundational principles of eventdriven programming and GUI design in Python.

2. Introduction

The graphical number guessing game developed using the tkinter library. It provides an interactive interface where players are challenged to guess a randomly generated number composed of a user-defined number of digits. At the start of the game, users input the number of digits in the target number and the maximum number of allowed guesses. The game selects a random sequence of digits (1–9), and the player must guess the digits one at a time. For each correct guess, the corresponding digit is revealed in its correct position. The game visually tracks the player's progress using a status label, attempt counter, score display, and a

progress bar. It also includes user-friendly features like input validation, blinking effects for emphasis, error prompts for invalid inputs, and an animated "shake" for incorrect guesses. Additionally, the game supports different color themes, enhancing the visual experience. When the player wins or exhausts all guesses, a message is displayed showing the result and score, along with the option to play again or exit. This program demonstrates core concepts in GUI development, user interaction, and control flow in Python.

3. Literature Review

This Number Guessing Game GUI embodies decades of educational and HCI research by turning a simple algorithmic puzzle into an engaging, interactive tool. In computer science education, “guess the number” exercises are a staple for introducing core programming concepts—loops, conditionals, randomization, and state management—because they encourage learners to think in terms of input–process–output cycles (e.g., Gaddis, 2015). By wrapping this logic in a Tkinter interface, the code leverages visual feedback—progress bars, blinking labels, error shakes—to make abstract constructs concrete, aligning with Papert’s constructivist view that learners build understanding by manipulating meaningful artifacts (Papert, 1980). From a usability standpoint, the program applies Nielsen’s heuristics (1994) and Shneiderman’s principles (2010): it provides immediate system status (through score and progress updates), prevents and handles errors gracefully (input validation with animated cues), and offers aesthetic customization via themes. Gamification elements—score deductions, hint options, and “play again” loops—draw on motivational design frameworks (Deterding et al., 2011), reinforcing how immediate, goal-oriented feedback can sustain user engagement. In sum, this project illustrates how combining pedagogical game design with proven HCI practices yields a small but powerful learning environment for both new coders and casual players.

4.Proposed System :

Graphical Interface:

Uses Tkinter to present a windowed application. Widgets (Labels, Buttons, Entry fields, Progressbar) replace console prompts.

- Rich Feedback & Animations:

Progress Bar visually shows how many guesses remain.

Blinking Status Label draws attention to the current progress. Fade-In Welcome and Shake on Error animations provide immediate, intuitive feedback.

Theming & Customization:

Four built-in color themes (light, dark, green, blue) let users choose an aesthetic that suits them. Additional themes can easily be added.

- Score Tracking:

Starts at 100 and deducts points for wrong guesses, encouraging efficiency. Score is displayed and updated live.

- Replay Loop:

After a win or loss, a dialog asks “Play Again?” so users can restart without rerunning the program.

- Input Validation with GUI Prompts:

Errors (“invalid digit,” “max guesses too low”) are shown via message boxes, and invalid entries trigger a shake animation to draw attention.

ADVANTAGES:

▪ More engaging and challenging

▪ Encourages strategic thinking

▪ Competitive and fun

▪ Encourages replayability

.

5.System Architecture and Design

- **Sequence Diagram**

  Describes the step-by-step interaction between objects like the user, GUI, controller, and logic during a specific operation. It captures the order of messages passed for function execution.

- **Uml diagram**:

  A UML (Unified Modeling Language) diagram is a standardized way to visualize the structure and behavior of a software system. UML provides a collection of diagram types

6. IMPLEMENTATION

The Number Guessing Game was implemented using Python with the Tkinter library, which offers builtin support for graphical user interfaces.

After implementing the number guessing game and performing unit testing, it's important to evaluate the overall success of the project, identify any areas for improvement, and consider future enhancements.

Here's a comprehensive postimplementation analysis:

6.1 HARDWARE REQUIREMENTS:

To implement the system, the following hardware requirements must be met:

• Processor: Minimum Core i5 processor

• RAM:2GB(minimum) or 8GB (recommended)

• Hard Disk Space: 50GB or more

6.2.SOFTWARE REQUIREMENTS:

The following software requirements must be met to implement the system:

• Programming Language: Python 3.12.4

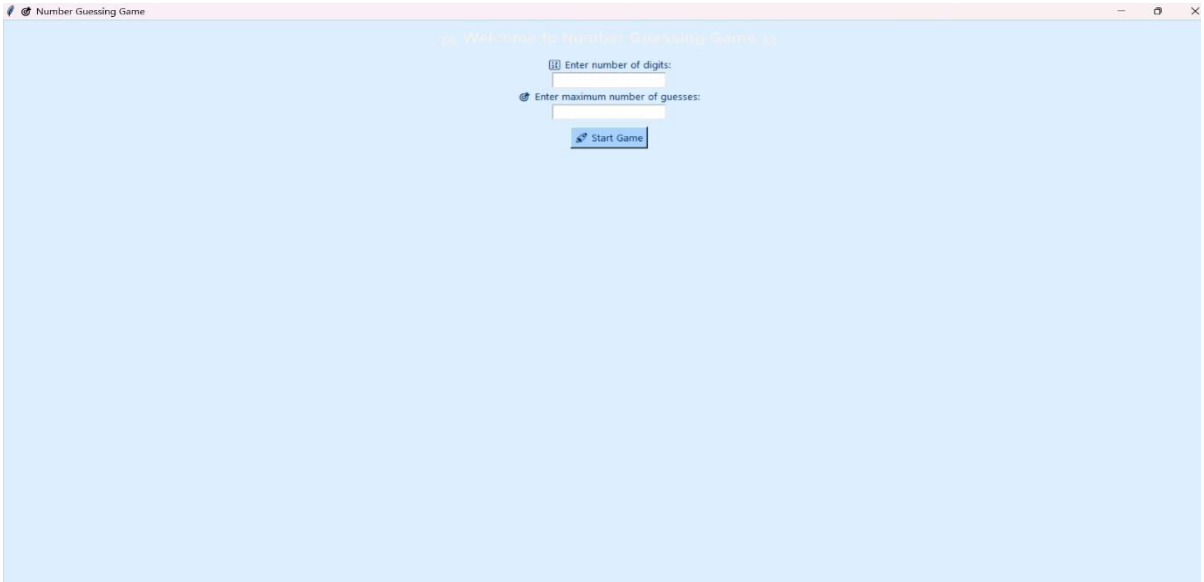
• Operating System: Windows 10 or later versions of Windows

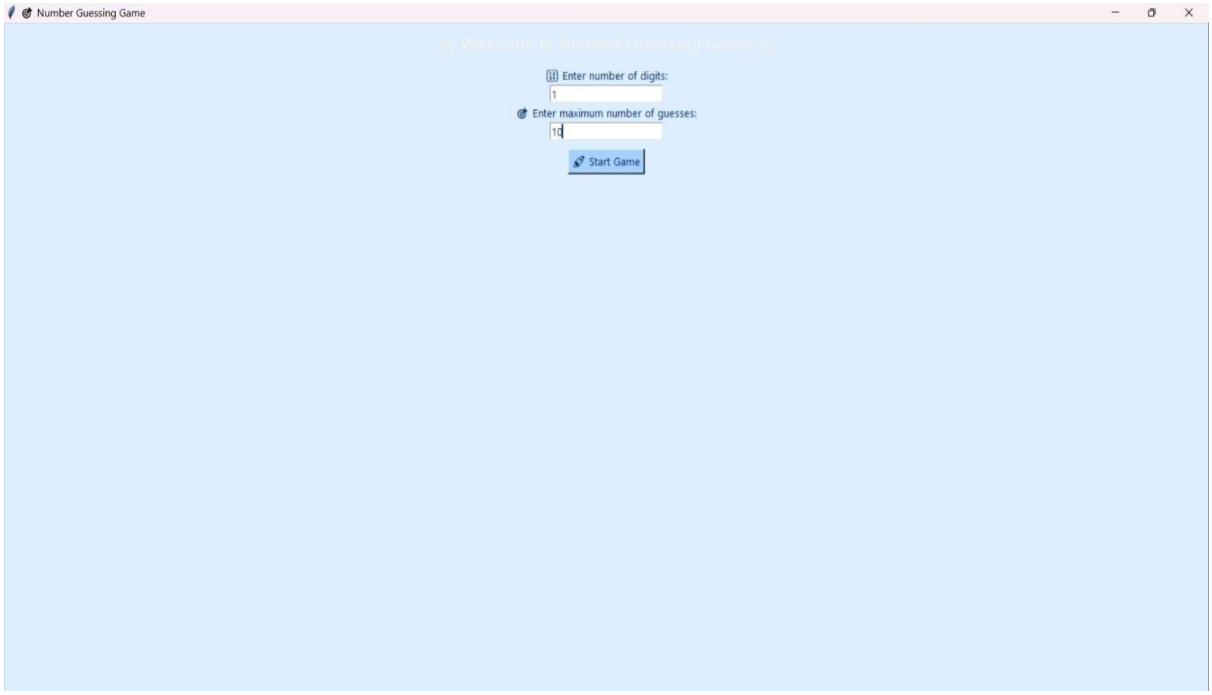
7. TESTING AND RESULTS

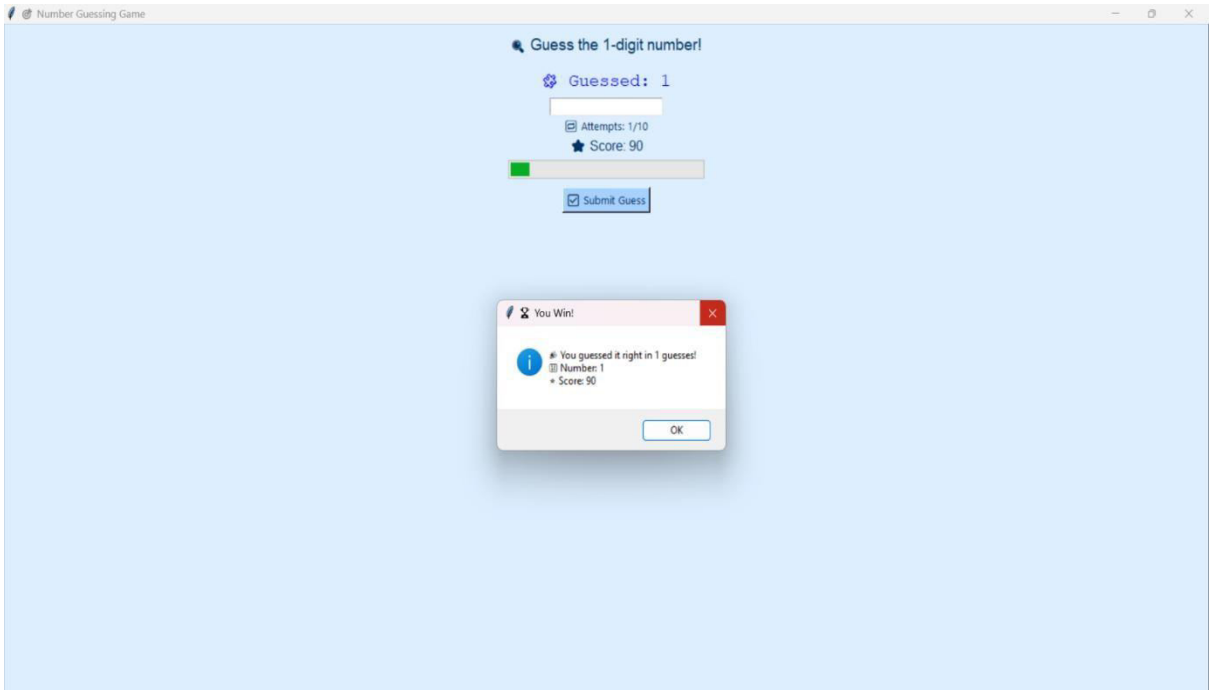
- Unit Testing: Tests individual functions like add or divide for correctness in isolation.
- Integration Testing: Ensures GUI components and backend logic interact correctly.
- Functional Testing: Verifies calculator functions work as expected from a user's view.
- System Testing: Tests the complete application's performance and stability in realworld conditions.
- Acceptance Testing: Confirms the calculator meets user requirements and is ready for release.
- White Box Testing: Tests internal code logic and paths for accuracy and efficiency.
- Black Box Testing: Tests input-output behaviour without looking at the internal code.

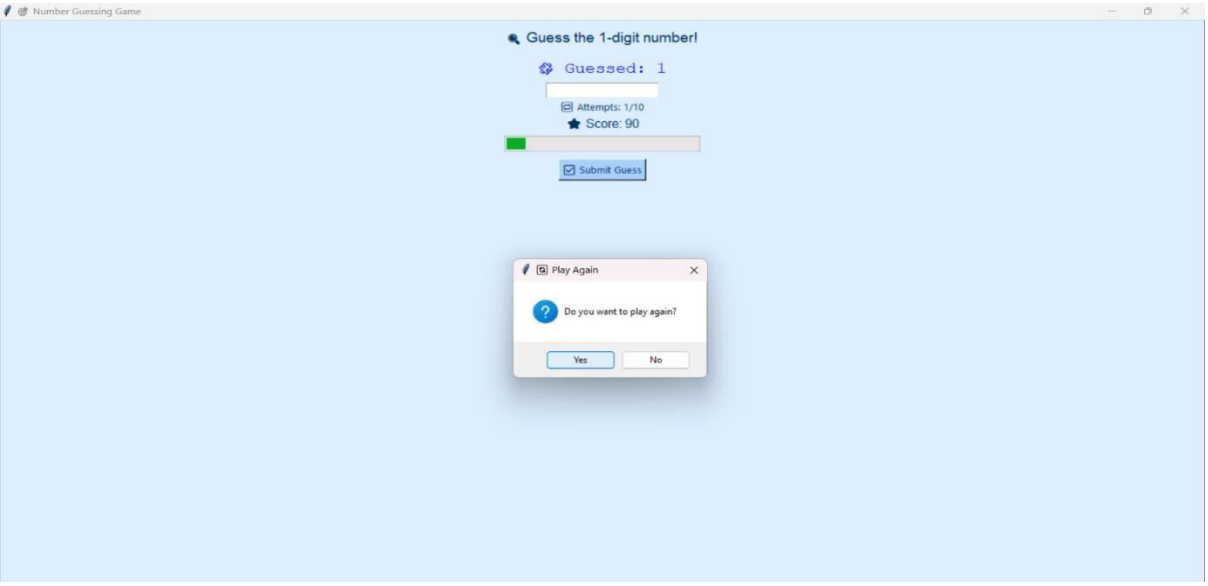
Result: All test cases passed successfully with no defects encountered.

8.OUTPUTSCREENS:




Number Guessing Game
Welcome to Number Guessing Game
Enter number of digits:
1
Enter maximum number of guesses:
10
Start Game


Number Guessing Game
Guess the 1-digit number!
Guessed: 1
Attempts: 1/10
Score: 90
Submit Guess
You Win!
You guessed it right in 1 guesses!
Number: 1
Score: 90
OK



9. CONCLUSION

The project successfully demonstrates how Python and Tkinter can be used to build a basic GUI-based Number Guessing Game. The implementation of the number guessing game in python serves as an excellent educational project for beginner programmers. This project successfully demonstrates fundamental programming concepts such as loops, conditionals, and random number generation, while also providing a fun and interactive way for users to engage with these concepts. Through this project, learners can gain practical experience in coding, improve their problem-solving skills, and better understand the basics of programming.

The game's simplicity and immediate feedback mechanism ensure that users remain engaged and motivated to improve their performance. The replay option further enhances the game's educational value by encouraging repeated practice. Additionally, the refactored code structure improves readability and maintainability, making it easier for learners to understand and modify the code.

10. FUTURE SCOPE

For the future development of this Number Guessing Game code, here are a few areas where you can focus:

1. Enhanced Game Features:

- **Multiple Players:**

Implement multiplayer functionality, where multiple players can take turns guessing the number. You can set it up so that each player has their own number to guess.

- **Hints:**

Allow the game to give hints after a certain number of incorrect guesses, such as telling the player whether their guess is too high or too low.

- **Difficulty Levels:**

Introduce different difficulty levels (e.g., Easy, Medium, Hard), where the number of digits, range of numbers, and maximum guesses change based on the difficulty level.

2. Input Validation & Error Handling:

- **Robust Input Checking:**

Improve input validation to handle edge cases better, such as checking for non-numeric inputs, blank inputs, or inputs that do not fit the allowed criteria (e.g., negative numbers, characters).

- **Multiple Guesses:**

Add an option for players to guess multiple digits at once, where they can try to guess the entire number in one go. If incorrect, they would lose a guess.

3.User Interface Improvements:

- **Sound Effects and Animations:**

Add sound effects for correct and incorrect guesses, or use simple animations to display the number being guessed. It will enhance the overall gaming experience.

4. Artificial Intelligence (AI) Integration:

You could introduce an AI opponent that guesses the number instead of a human. You can code it using a binary search or a strategy where the AI learns to make more accurate guesses over time.

11. REFERENCES

1. Bak, J. et al. (2014). *Teaching GUI programming with Python and Tkinter.*
2. Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience.* 3. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). *From Game Design Elements to Gamefulness: Defining Gamification.*
4. Gaddis, T. (2015). *Starting Out with Python.*
5. Malone, T. W. (1981). *Toward a theory of intrinsically motivating instruction.*
6. Nielsen, J. (1994). *10 Usability Heuristics for User Interface Design.*
7. Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.*
8. Shneiderman, B., & Plaisant, C. (2010). *Designing the User Interface.*
9. Werbach, K., & Hunter, D. (2012). *For the Win: How Game Thinking Can Revolutionize Your Business.*
10. Wood, D., Bruner, J. S., & Ross, G. (1976). *The role of tutoring in problem solving.*