PROJECT REPORT

GUESS A NUMBER

Take your learning journey beyond the syllabus

Name	Pulkit Yadav
REGISTRATION NO:	25BSA10078
TEACHER	Preetam Suman Sir

Introduction

Games are one of the simplest and most interactive ways to learn programming.
This project implements a **Guess the Number** game using Python, where:

1. **User guesses a number generated by the computer**, and
2. **Computer guesses the number selected by the user.**

This project helps understand loops, conditionals, input/output handling, random number generation, and simple algorithmic thinking.

Problem Statement

Design and implement a Python program where both the **user** and the **computer** try to guess a randomly chosen number. The system must provide hints and process user feedback in real time.

Functional Requirements

1: User Guess Module

- Computer selects a random number.
- User tries to guess it.
- System gives hints (“Too high”, “Too low”).

2: Computer Guess Module

- User thinks of a number.
- Computer guesses using narrowing (binary-search-like) logic.
- User provides feedback:
  - Too High (h)
  - Too Low (l)
  - Correct (c)

3: Input/Output Interaction

- Display prompts, results, and hints to user.
- Validate feedback.

Non-Functional Requirements

1: Usability

Program must be easy to use with clear instructions.

2: Reliability

Random numbers must be truly random in a given range.

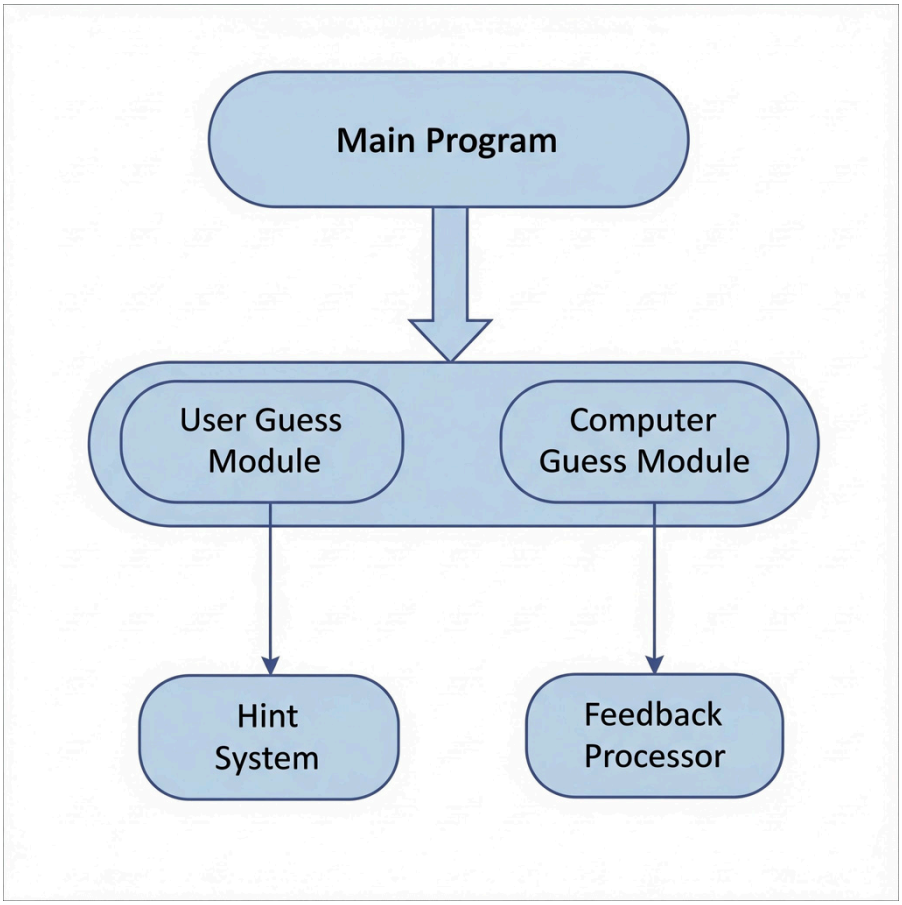
3: Performance

Game must respond instantly to user input.

4: Maintainability

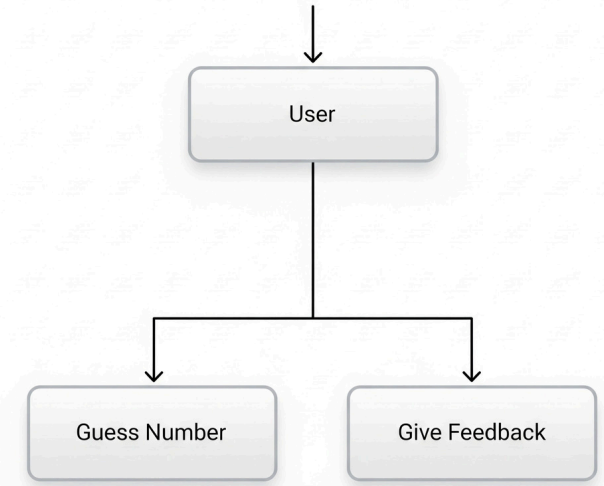
Code must be modular with functions for each task.

System Architecture

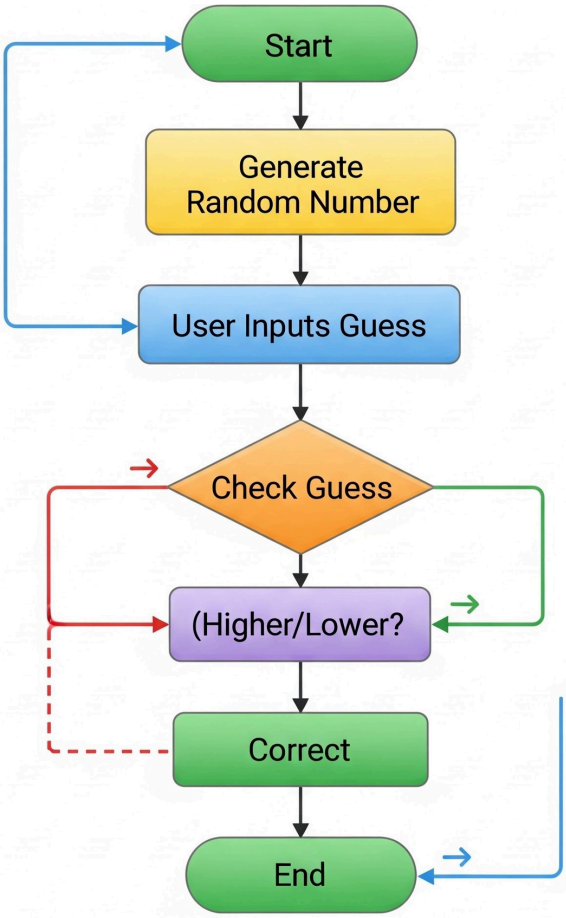


Design Diagrams

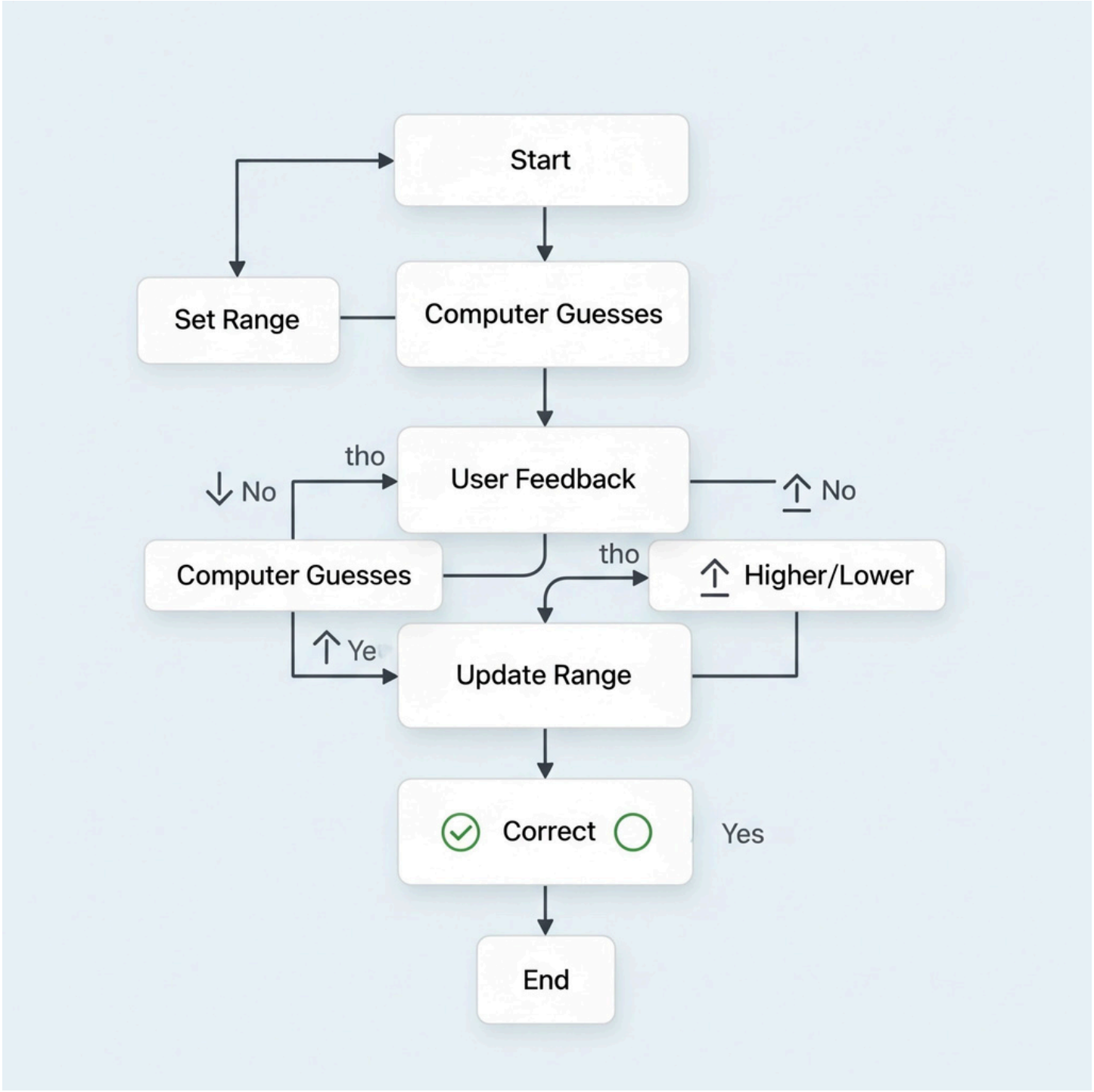
1.User Case Diagram



Workflow Diagram

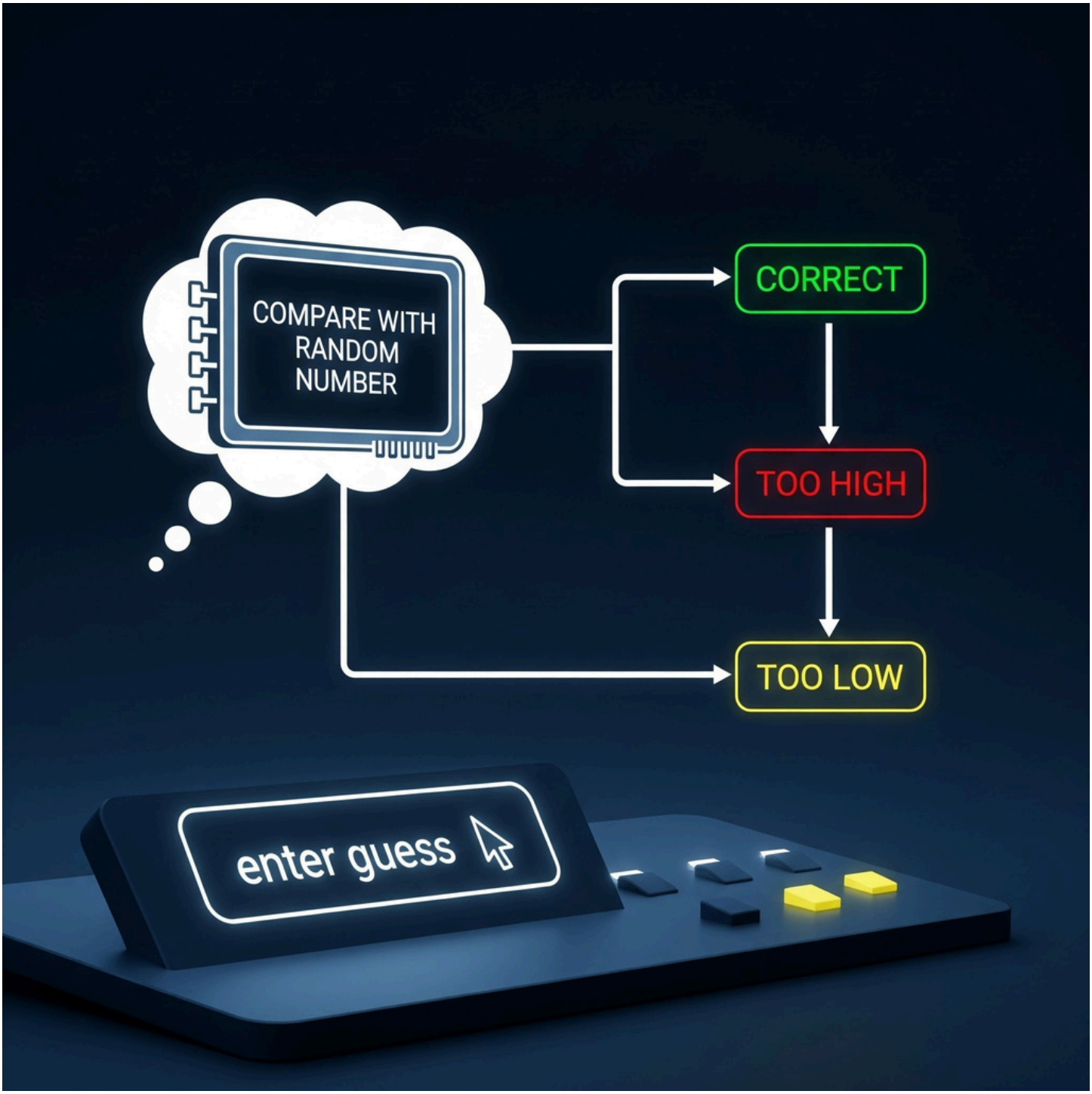


Computer Guessing Flow

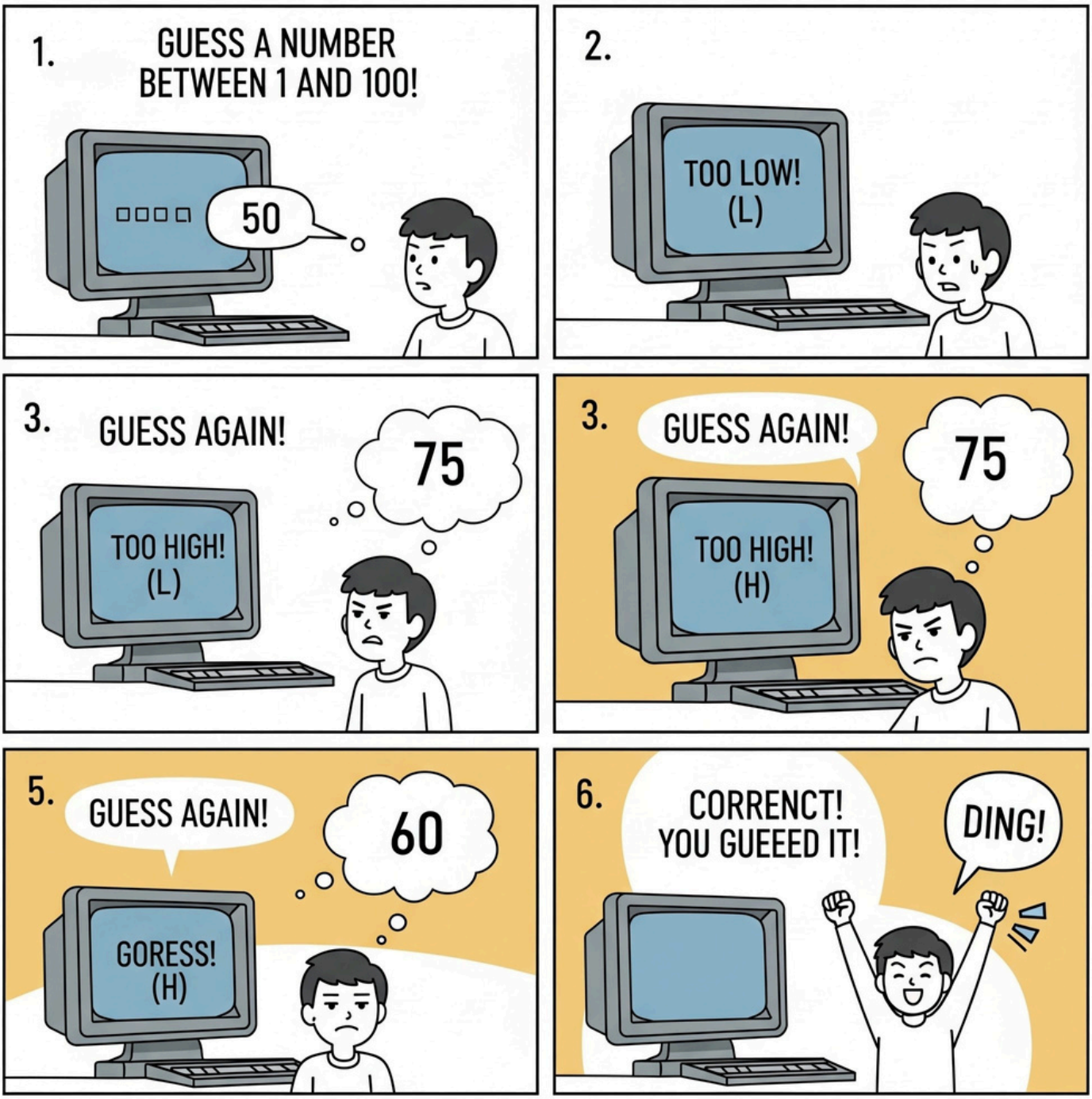


Sequence Diagram

User Guess



Computer Guess



Implementation

```python
import random

def guess(x):  1 usage
    random_number = random.randint( a: 1,x)
    guess =0
    while guess != random_number:
        guess = int(input(f'guess a number between 1 and {x}:'))
        if guess < random_number:
            print('sorry,guess again .Too low.')
        elif guess>random_number:
            print('sorry,guess again.too high.')


    print("congo you have guessed the number correctly")
guess(10)
```

```python
def computer_guess(x):  1 usage
    low =1
    high=x
    feedback = ''
    while feedback != 'c':
        if low !=high:
            guess= random.randint(low,high)
        else:
            guess = low
        feedback = input(f'is {guess} too high (h) , too low(l), or correct(c)')
        if feedback=="h":
            high = guess-1
        elif  feedback == "l":
            low = guess+1
    print(f"you got the number")

computer_guess(10)
```

OUTPUT

```
guess a number between 1 and 10:5
sorry,guess again .Too low.
guess a number between 1 and 10:6
sorry,guess again .Too low.
guess a number between 1 and 10:7
congo you have guessed the number correctly
```

```
guess a number between 1 and 10:5
sorry,guess again .Too low.
guess a number between 1 and 10:6
sorry,guess again .Too low.
guess a number between 1 and 10:7
congo you have guessed the number correctly
```

Testing Approach

- Input validation
- Boundary testing (1 and max value)
- Random value testing
- Feedback correctness testing (h/l/c)

Challenges Faced

- Handling invalid input
- Maintaining correct high/low boundaries
- Avoiding infinite loops when user enters wrong feedback

Learnings

- Use of loops and conditions
- Random module
- Designing simple algorithms
- User interaction handling
- Importance of debugging

Future Enhancements

- GUI using Tkinter
- Add scoring system
- Add difficulty levels
- Add sound or animations
- Store history of attempts

References

- Python Documentation
- Course Modules
- Random module reference