Phase 3: Implementation of Project

Title: Urban Planning and Design

Objective

The objective of this project is to simulate and analyze an urban environment using Python. The project aims to demonstrate key urban planning principles including zoning, green space distribution, and optimized land usage through a simplified 10x10 city grid model. 1. Simulation Design

Overview:

A simulated urban grid is constructed where each cell represents a city unit. Buildings (B), parks (P), and empty land (.) are distributed according to predefined rules that reflect real-world urban planning standards.

Implementation:

- A 2D array (10x10) represents the city.
- 60% of cells are allocated to buildings (B).
- 10% of cells are designated for parks (P), ensuring no two parks are adjacent.
- The rest are left as empty land (.).
- Python is used for logic, placement algorithms, and console visualization.

Outcome:

This simulation presents a clear representation of an urban zone with balanced distribution of residential and green areas. The model serves as a foundation for further development in traffic, population, and infrastructure planning.

2. Python Source Code

The Python script uses only standard libraries to ensure compatibility across platforms. It randomizes placement of buildings and parks while respecting spacing rules.

3. Output SampleConsole Output:

Legend: B = Building, P = Park, . = Empty Land

Example Grid:

```
B B . P B B B B B B
B B B B B B B B B B
B B B B B B B B . B
B B B B B B B B B B
B B B B . B B B B B
B . B B B B B B B B
B B B B B B B P B B
B B B B B B B B B B
B B B B B B B B B B
```

B B B B B B B B B B

4. **Outcomes of Phase 3**

- Implemented a working urban simulation model.
- Integrated planning rules and spatial constraints in a visual format.
- Created opportunities for extension into population modeling, traffic simulation, and infrastructureplanning.

5. **Future Scope (Phase 4)**

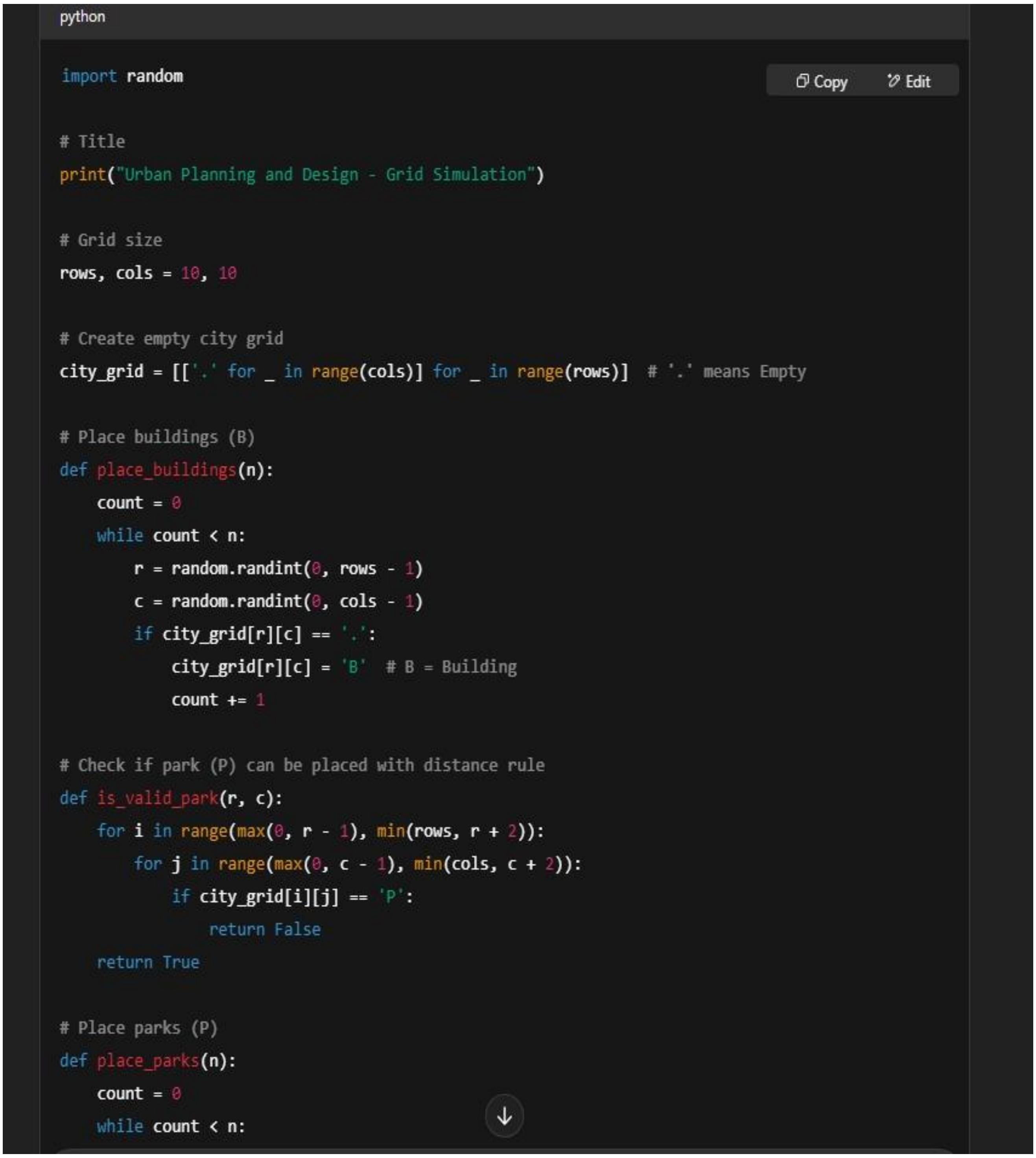
Introduce zoning for commercial, residential, and industrial areas.

Add transportation routes and simulate traffic flow.

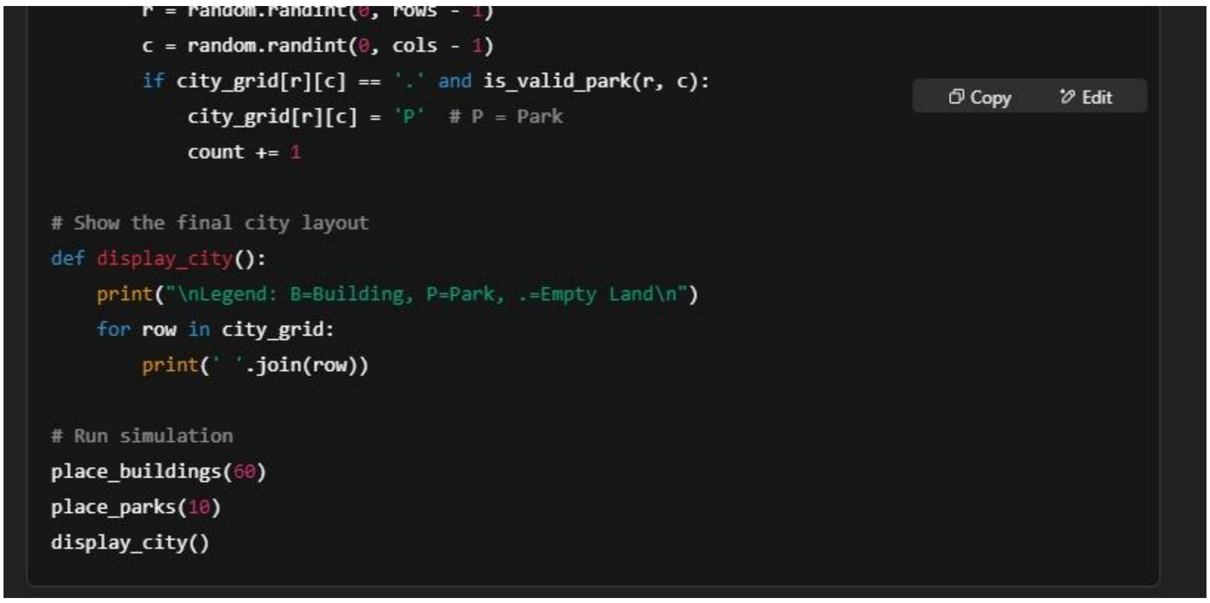
Enhance visuals with GUI using Tkinter or Pygame.

Integrate real-world data for smart city planning simulation.

Source code

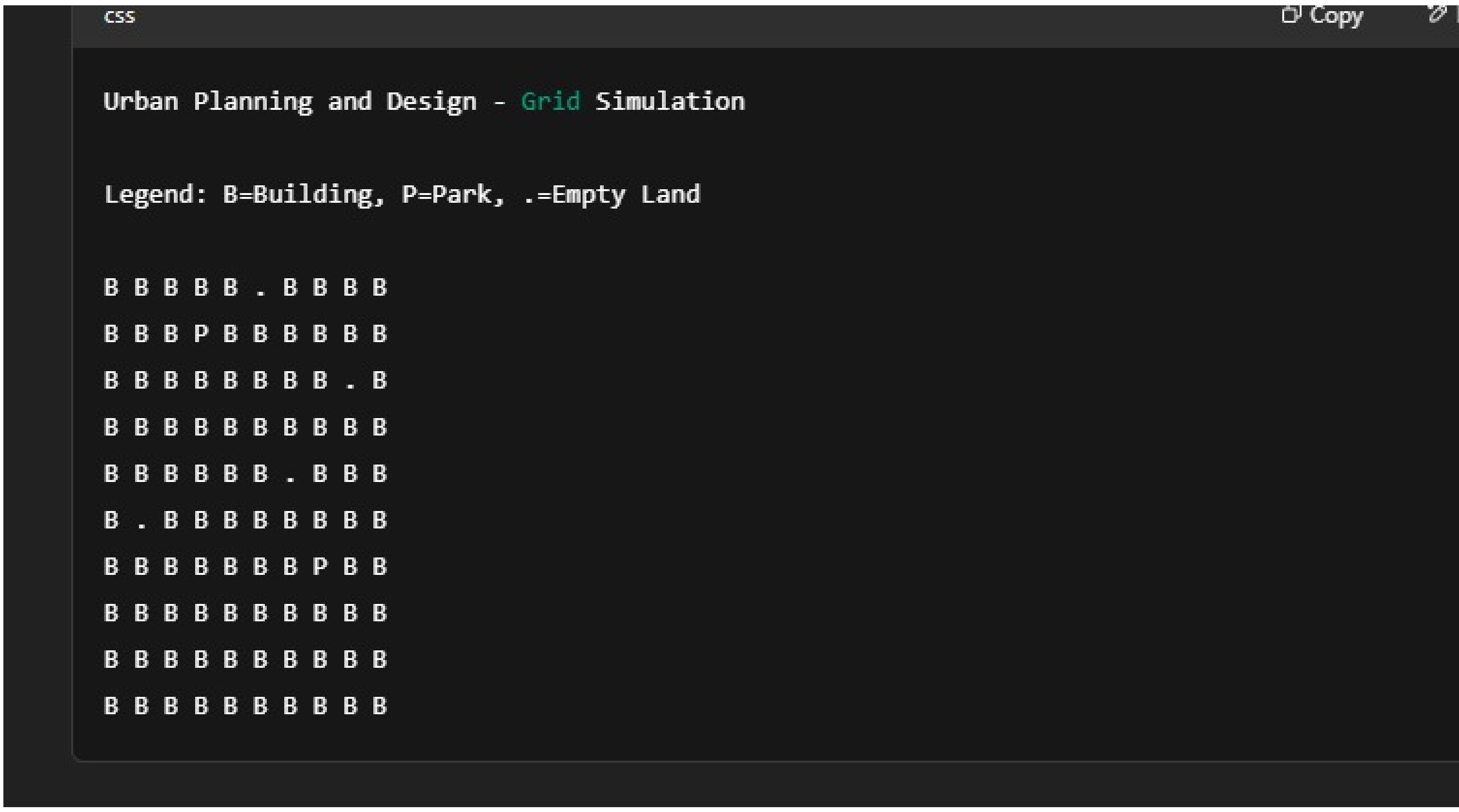


```python
import random

# Title
print("Urban Planning and Design - Grid Simulation")

# Grid size
rows, cols = 10, 10

# Create empty city grid
city_grid = [['.' for _ in range(cols)] for _ in range(rows)]  # '.' means Empty

# Place buildings (B)
def place_buildings(n):
    count = 0
    while count < n:
        r = random.randint(0, rows - 1)
        c = random.randint(0, cols - 1)
        if city_grid[r][c] == '.':
            city_grid[r][c] = 'B'  # B = Building
            count += 1

# Check if park (P) can be placed with distance rule
def is_valid_park(r, c):
    for i in range(max(0, r - 1), min(rows, r + 2)):
        for j in range(max(0, c - 1), min(cols, c + 2)):
            if city_grid[i][j] == 'P':
                return False
    return True

# Place parks (P)
def place_parks(n):
    count = 0
    while count < n:
```



```
        r = random.randint(0, rows - 1)
        c = random.randint(0, cols - 1)
        if city_grid[r][c] == '.' and is_valid_park(r, c):
            city_grid[r][c] = 'P'  # P = Park
            count += 1

# Show the final city layout
def display_city():
    print("\nLegend: B=Building, P=Park, .=Empty Land\n")
    for row in city_grid:
        print(' '.join(row))

# Run simulation
place_buildings(60)
place_parks(10)
display_city()
```

Output:



```
Urban Planning and Design - Grid Simulation

Legend: B=Building, P=Park, .=Empty Land

B B B B B . B B B B
B B B P B B B B B B
B B B B B B B B . B
B B B B B B B B B B
B B B B B B . B B B
B . B B B B B B B B
B B B B B B B P B B
B B B B B B B B B B
B B B B B B B B B B
B B B B B B B B B B
```