COLLEGE CODE:3108

COLLEGE NAME: JEPPIAAR ENGINEERING COLLEGE

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: au23it022

ROLL NO: 310823205022

DATE: 16-5-25

Completed the project named as,

SMARTEVAC-A Smart Evacuation planning system

SUBMITTED BY:R.Durgashree

MOBILE NO:7200134132

Phase 4: Performance of the project

Title : SmartEvac – AI-Powered Evacuation Planning System

Objective:
The focus of Phase 4 is to enhance the performance of SmartEvac by refining its AI algorithms for more accurate evacuation planning, optimizing the system for scalability, and improving real-time data integration. This phase also aims to strengthen the chatbot's responsiveness, enhance disaster data processing, and reinforce data security, while preparing for multilingual and accessibility support.

1. AI Model Performance Enhancement

Overview:
The evacuation planning algorithms will be refined using feedback and data from Phase 3. The goal is to increase route accuracy, hazard prediction, and response adaptability under dynamic emergency scenarios.

Performance Improvements:

- **Route Accuracy Testing**:
  The evacuation planning algorithms will be refined using feedback and data from Phase 3. The goal is to increase route accuracy, hazard prediction, and response adaptability under dynamic emergency scenarios.
- **Model Optimization**: Optimization techniques (e.g., A*, Dijkstra tuning, and constraint mapping) will improve performance in both speed and real-time responsiveness.

Outcome:
By the end of Phase 4, the AI model will produce more precise evacuation routes, respond faster to changes in disaster conditions, and reduce planning errors such as unsafe or blocked paths.

2. Chatbot Performance Optimization

Overview:

The chatbot interface will be refined to provide quicker, more user-friendly interactions, especially under emergency stress. NLP capabilities will be improved for better comprehension of varied input styles.

Key Enhancements:

- **Response Time**: System tuning will ensure low-latency interactions even under high usage.
- **Language Processing**: NLP models will interpret diverse user inputs, with groundwork laid for multilingual support and voice commands.

Outcome:

The chatbot will deliver faster, clearer, and more accurate responses, improving accessibility and usability for diverse user groups during crises.

3. Real-Time Disaster Data Integration

Overview:

This phase will focus on seamless integration with real-time data sources, including government APIs, satellite feeds, and sensor networks, for live updates on hazards and infrastructure status.

Key Enhancements:

- **Live FeedProcessing**:SmartEvac will be optimized to process real-time updates (e.g., road blockages, weather alerts) instantly.
- **API Expansion:** Reliable disaster and infrastructure data sources will be integrated for higher situational awareness.

Outcome:

By the end of Phase 4,SmartEvac will dynamically adapt evacuation plans based on live data, greatly improving safety and response relevance.

4. Data Security and Privacy Performance

Overview:
Data security mechanisms will be strengthened to handle increased user data and maintain privacy standards under system scale-up.

Key Enhancements:

- **Advanced Encryption**: Scalable, end-to-end encryption protocols will be used for user location and identity data.
- **Security Testing**: Penetration tests and vulnerability scans will be conducted under simulated high loads.

Outcome:
The system will offer robust protection for all user data, even during peak usage, meeting disaster-response data privacy standards.

5. Performance Testing and Metrics Collection

Overview:
SmartEvac will undergo extensive performance testing to ensure scalability and reliability in real-world emergency conditions.

Implementation:

- **Load Testing**: Simulate large-scale disaster scenarios to validate system stability.
- **Performance Metrics**: Track response times, success rates, route generation accuracy, and error rates.
- **Feedback Loop**: Collect detailed feedback from a broad test group, including emergency personnel and local authorities.

Outcome:

SmartEvac will be capable of scaling to large populations with minimal lag, maintaining high accuracy and responsiveness.

Key Challenges in Phase 4

1. **Scalability Under Crisis Load:**
   - **Challenge**: Ensuring system stability and performance under massive user influx during emergencies.
   - **Solution**: Stress testing and infrastructure optimization through cloud-based scaling.
2. **Data Security at Scale**:
   - **Challenge**: Protecting user data during high traffic and data exchange volume.
   - **Solution**: Advanced encryption and ongoing security auditing.
3. **Real-Time Data Reliability**:
   - **Challenge**: Inconsistencies in live disaster data feeds may impact planning.
   - **Solution**:Use multiple validated sources and implement fallback protocols.

Outcomes of Phase 4

1. **Improved Route Accuracy:** The AI will generate safer, more efficient evacuation routes with quicker response to real-time updates.
2. **Enhanced Chatbot Performance:** The chatbot will support faster, more natural interactions with improved understanding of user input.
3. **Optimized Data Integration:** Live disaster updates will be processed with minimal delay, keeping evacuation plans up-to-date.
4. **Reinforced Data Security:** All user data will be encrypted and stored securely, even at scale, meeting emergency data standards.

Next Steps for Finalization

In the next and final phase, the system will be fully deployed, and further feedback will be gathered to fine-tune the AI model and optimize the overall user experience before the official launch. In the final phase, SmartEvac will be deployed in pilot locations. Real-world feedback will be gathered to fine-tune the AI and user interface. Final adjustments will ensure readiness for a full-scale launch.

Sample Code for Phase 4:

Main.py

```
    import tkinter as tk
from tkinter import filedialog, messagebox
from blueprint_processor import process_blueprint
from pathfinder import find_evacuation_path
from utils import draw_path
import os

def run_bot():
    disaster = disaster_entry.get()
    try:
        people = int(people_entry.get())
    except ValueError:
        messagebox.showerror("Invalid Input", "Number of people must be an integer.")
        return

    filepath = filedialog.askopenfilename(filetypes=[("Image files", "*.png;*.jpg;*.jpeg")])
    if not filepath:
        return

    graph, image, exits = process_blueprint(filepath)
    if not exits:
        messagebox.showerror("No Exits Found", "No exits were detected on the blueprint.")
        return

    start = (image.height // 2, image.width // 2)
    path = find_evacuation_path(graph, start, exits)

    if not path:
        messagebox.showerror("No Path", "No evacuation path found.")
        return
```

```
            output_image = draw_path(image, path)
            os.makedirs("outputs", exist_ok=True)
            output_path = os.path.join("outputs", "evacuation_plan.png")
            output_image.save(output_path)
            output_image.show()
            messagebox.showinfo("Success", f"Evacuation path saved to: {output_path}")

        # GUI
        root = tk.Tk()
        root.title("Smart Evacuation Planner")

        tk.Label(root, text="Disaster Type (e.g., fire, earthquake):").pack()
        disaster_entry = tk.Entry(root)
        disaster_entry.pack()

        tk.Label(root, text="Number of People:").pack()
        people_entry = tk.Entry(root)
        people_entry.pack()

        tk.Button(root, text="Select Blueprint and Generate Plan", command=run_bot).pack(pady=10)

        root.mainloop()
```

blueprint_processor.py

```
import cv2
import numpy as np
import networkx as nx
from PIL import Image

def process_blueprint(image_path):
    image = cv2.imread(image_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)

graph = nx.grid_2d_graph(*binary.shape)

for y in range(binary.shape[0]):
  for x in range(binary.shape[1]):
    if binary[y, x] == 0:
      if graph.has_node((y, x)):
        graph.remove_node((y, x))

exits = find_exits(binary)
pil_image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
return graph, pil_image, exits

def find_exits(binary):
height, width = binary.shape
exits = []
for x in range(width):
  if binary[0, x] == 255:
```

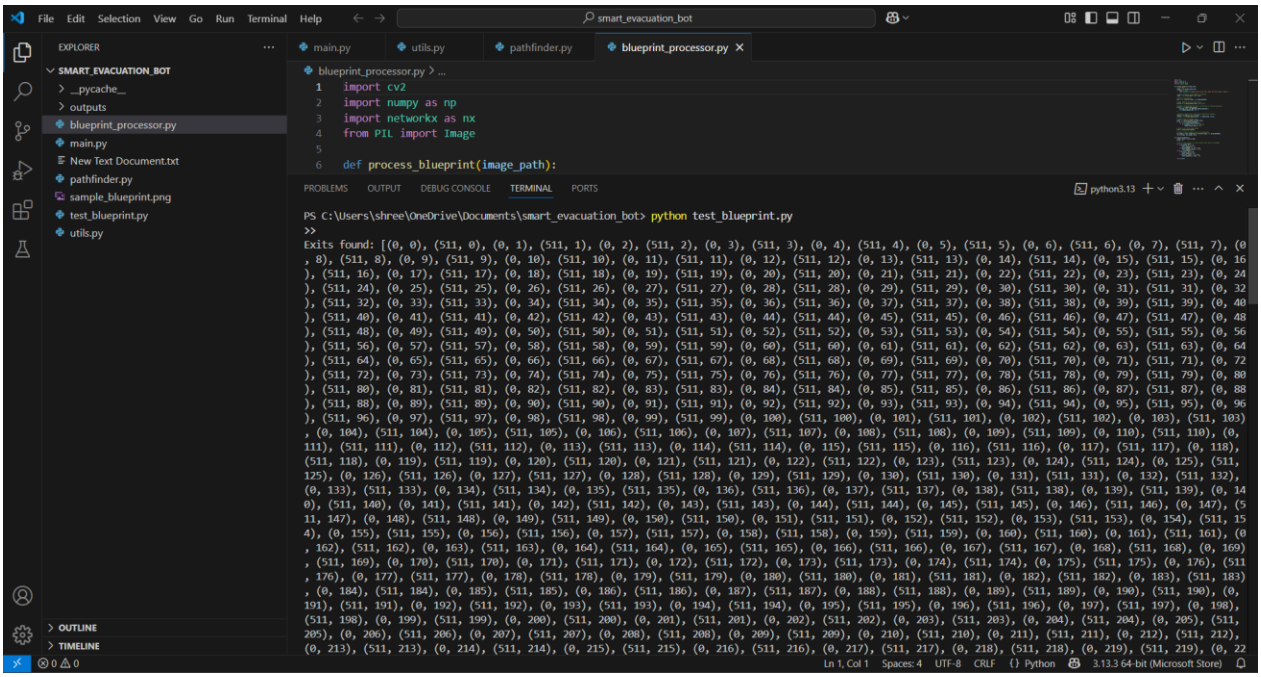
```
    exits.append((0, x))
if binary[height - 1, x] == 255:
    exits.append((height - 1, x))
            for y in range(height):
if binary[y, 0] == 255:
    exits.append((y, 0))
if binary[y, width - 1] == 255:
    exits.append((y, width - 1))
     return exits
```

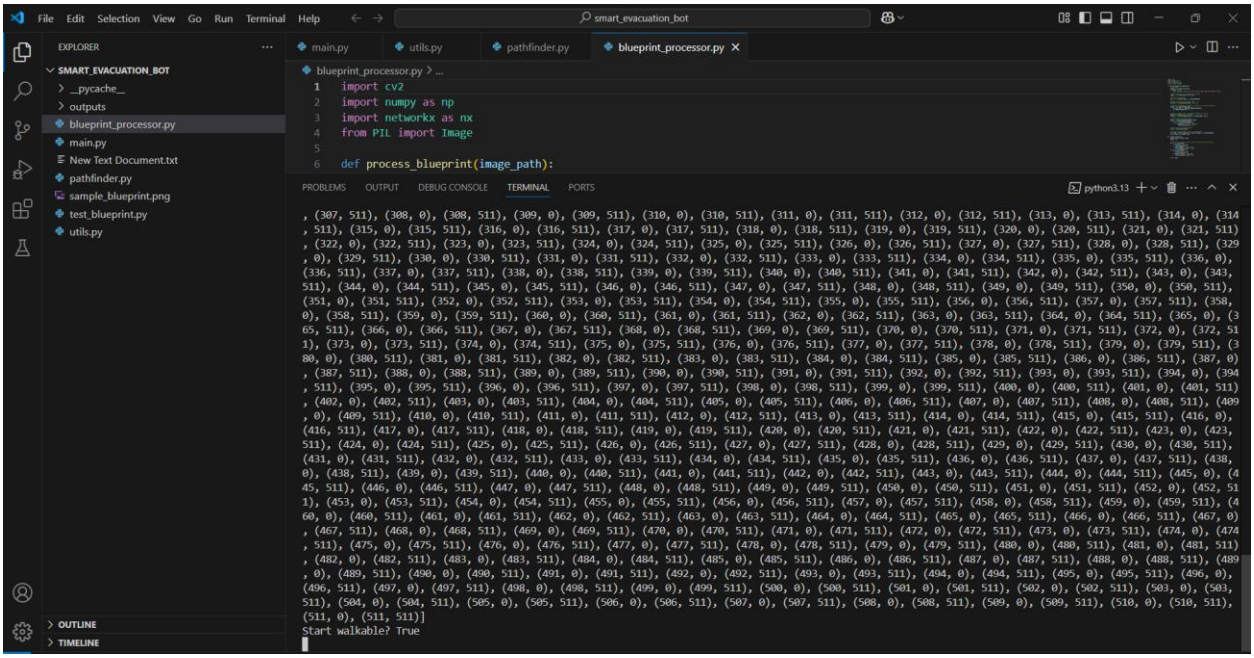
Pathfinder.py

```
import networkx as nx

def find_evacuation_path(graph, start, exits):
    shortest_path = None
    shortest_length = float('inf')

    for exit in exits:
        try:
            path = nx.shortest_path(graph, source=start, target=exit)
            if len(path) < shortest_length:
                shortest_path = path
                shortest_length = len(path)
        except (nx.NetworkXNoPath, nx.NodeNotFound):
            continue

    return shortest_path
```

utils.py

```
from PIL import ImageDraw

def draw_path(image, path):
    draw = ImageDraw.Draw(image)
    for point in path:
        draw.point((point[1], point[0]), fill=(255, 0, 0))  # red dot
    return image
```





Performance Metrics Screenshot for Phase 4:

Screenshots showing improved accuracy metrics, reduced latency in chatbot responses, and real-time IoT data collection should be included here