

Phase 3: Implementation of Project

Title: AI Driven Evacuation Planning System

Objective

Phase 3 focuses on implementing the AI-Driven Evacuation Planning System's core components, including an AI blueprint checker, chatbot interface, IoT integration, and data security measures. The system leverages machine learning and optimization to generate optimal evacuation routes and shelter allocations during emergencies, considering factors like population density and disaster severity to minimize response time and casualties.

1.AI Model Development

Overview:

The Evacuation Planning System utilizes machine learning and optimization techniques to generate optimal evacuation routes and shelter allocations during natural disasters or emergencies. The system integrates data on population density, infrastructure, disaster severity, and resource availability to minimize response time, reduce casualties, and ensure individual safety.

Implementation:

- **Data Collection:** Gather data on population density, infrastructure, disaster severity, and resource availability.
- **Data Preprocessing:** Clean and preprocess data for model training.
- **Model Training:** Utilize machine learning algorithms (e.g., graph-based models, reinforcement learning) to train the system.
- **Optimization:** Implement optimization techniques (e.g., linear programming, genetic algorithms) to generate optimal evacuation routes and shelter allocations.
- **Integration:** Integrate the system with real-time data feeds and GIS mapping.

Outcome:

By the end of this phase, The Evacuation Planning System optimizes emergency responses by generating efficient evacuation routes, allocating shelters effectively, and minimizing response times, ultimately saving lives and enhancing safety. The AI model is expected to provide basic Evacuation planning for the provided blue print and the population.

2.Chatbot Development Overview:

The AI will be made accessible through a chatbot interface that allows users to communicate with the system easily. The chatbot will serve as the front-end interface where users enter their blueprint and receive an appropriate evacuation planning.

Implementation:

- **NLP Integration:** Integrate NLP capabilities to understand user queries and provide relevant responses.
- **Data Integration:** Connect to databases containing information on population density, infrastructure, disaster severity, and resource availability.
- **Route Optimization:** Utilize algorithms to generate optimal evacuation routes based on user location and destination.
- **Shelter Allocation:** Provide information on nearby shelters, including capacity, location, and resource availability.

Outcome

At the end of Phase 3, the chatbot will be functional and capable of providing users with personalized evacuation plans through a simple conversation interface. It uses real-time data to suggest optimal routes and nearby shelters, ensuring quick and safe evacuations.

3.Data Security

Overview

Given the critical nature of user location and emergency data, SmartEvac will implement essential security measures to protect user privacy and system integrity. In Phase 3, foundational data protection strategies will be applied, focusing on encryption and controlled access.

Implementation

- **Encryption:** All user-submitted data, including location, personal details, and blueprint information, will be encrypted during transmission and storage using industry-standard encryption protocols.
- **Secure Storage:** Data will be stored in protected databases with role-based access controls, ensuring only authorized personnel or systems can access sensitive information.

Outcome

By the end of Phase 3, SmartEvac will have secure data handling processes in place, ensuring that sensitive evacuation and user information is protected through encryption and access control mechanisms.

4. Testing and Feedback Collection

Overview

Initial testing of the SmartEvac system will be conducted in this phase to assess its route planning accuracy, response time, and overall user experience.

Implementation

- **Test Groups:** A selected group of users will interact with the chatbot, simulating different emergency scenarios by submitting locations and blueprints. The system's route suggestions, shelter recommendations, and interface usability will be evaluated.
- **Feedback Loop:** Users will provide feedback on system performance, ease of use, clarity of instructions, and the relevance of evacuation plans.

Outcome

The feedback collected in Phase 3 will inform refinements in Phase 4, focusing on improving evacuation accuracy, optimizing AI responses, and enhancing the overall user interface and experience.

Challenges and Solutions

1. Data Integration and Accuracy

- **Challenge:** Inconsistent or incomplete real-time data (e.g., disaster updates, population density) may affect the accuracy of evacuation plans.
- **Solution:** Implement data validation protocols and prioritize integration with reliable and verified data sources to improve system reliability.

2. Route Optimization Under Dynamic Conditions

- **Challenge:** Rapidly changing emergency conditions (e.g., blocked roads, new hazards) can make route suggestions outdated quickly.
- **Solution:** Incorporate real-time updates and re-routing capabilities into the system to ensure evacuation paths remain accurate and safe.

3. Scalability During Emergencies

- **Challenge:** The system may experience a surge in user activity during large-scale disasters, potentially affecting performance.
- **Solution:** Utilize scalable cloud infrastructure and load balancing to maintain responsiveness and service availability under high demand.

Outcomes of Phase 3

By the end of Phase 3, the following milestones should be achieved:

1. **Basic AI Evacuation Model:** The AI system will be capable of generating basic evacuation routes and shelter recommendations based on user input and available data.
2. **Functional Chatbot Interface:** A working chatbot will allow users to submit locations or blueprints and receive evacuation guidance in a conversational format.
3. **Data Integration Framework:** Initial connections to key data sources—such as disaster feeds, population density, and shelter databases—will be established.
4. **Data Security Measures:** User and location data will be encrypted and stored securely, with access restricted to authorized systems or personnel.
5. **Initial Testing and Feedback:** Early testing will be completed with test users, and feedback will be collected to guide improvements in the next development phase.

Next Steps for Phase 4

In Phase 4, the team will focus on:

1. **Improving the AI's Accuracy:** Based on Phase 3 feedback, the evacuation algorithms will be refined to improve the accuracy and relevance of route planning and shelter recommendations.
2. **Expanding Multilingual Support:** The chatbot will be enhanced to support multiple languages and voice-based interactions, making it more inclusive and user-friendly during emergencies.
3. **Scaling and Optimizing:** Infrastructure will be optimized to support higher user volumes and complex evacuation scenarios, ensuring consistent performance under stress.

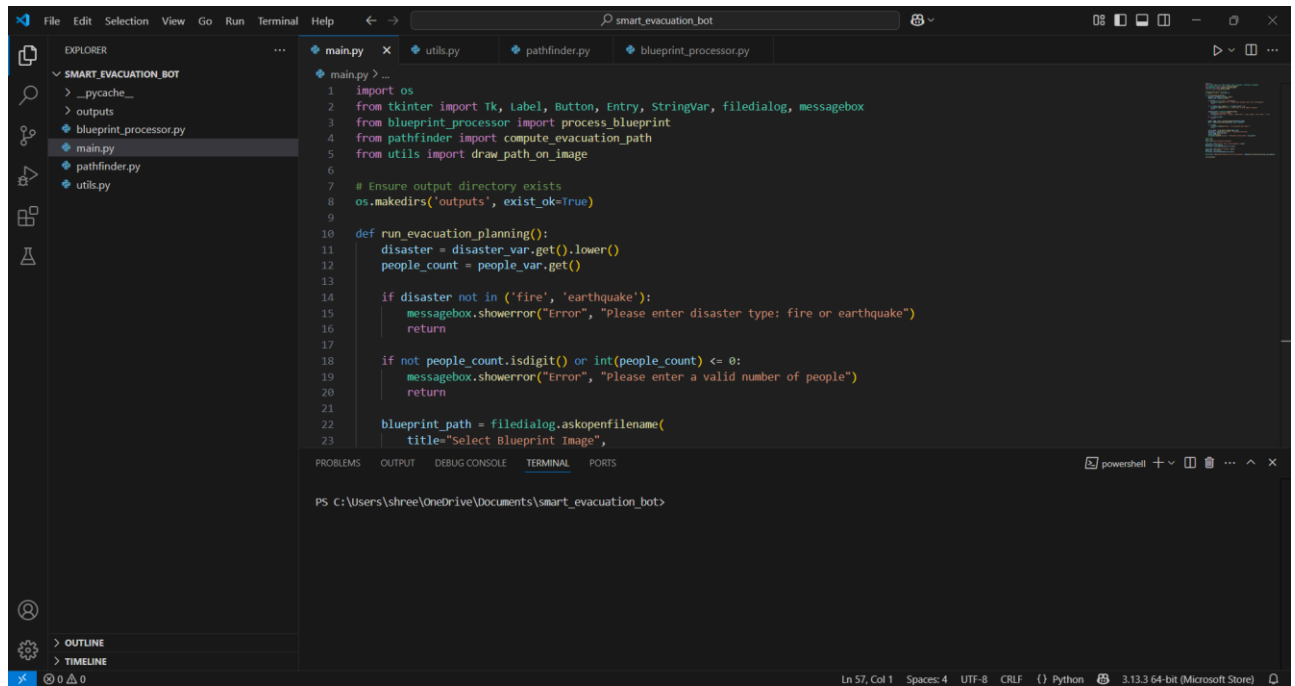
SCREENSHOTS OF CODE and PROGRESS – MUST BE ADDED HERE FOR PHASE 3

Blueprint_processor.py

The screenshot displays a Jupyter Notebook environment with the following components:

- File Explorer (Left):** Shows the project structure with files like `main.py`, `utils.py`, `pathfinder.py`, and `blueprint_processor.py`.
- Code Editor (Center):** Contains the Python script for `blueprint_processor.py`. The script performs the following steps:
 - Imports `cv2`, `numpy as np`, and `networkx as nx`.
 - Imports `Image` from `PIL`.
 - Defines a function `process_blueprint(image_path)` that:
 - Reads the image.
 - Converts it to grayscale using `cvtColor`.
 - Thresholds the image to get a binary image using `threshold`.
 - Creates a grid graph for the blueprint size using `nx.grid_2d_graph`.
 - Removes nodes that correspond to walls/distances (black pixels) from the graph.
- Terminal (Bottom):** Shows the command prompt with the file path `PS C:\Users\valve\OneDrive\Documents> smart_evaluation_bot`.

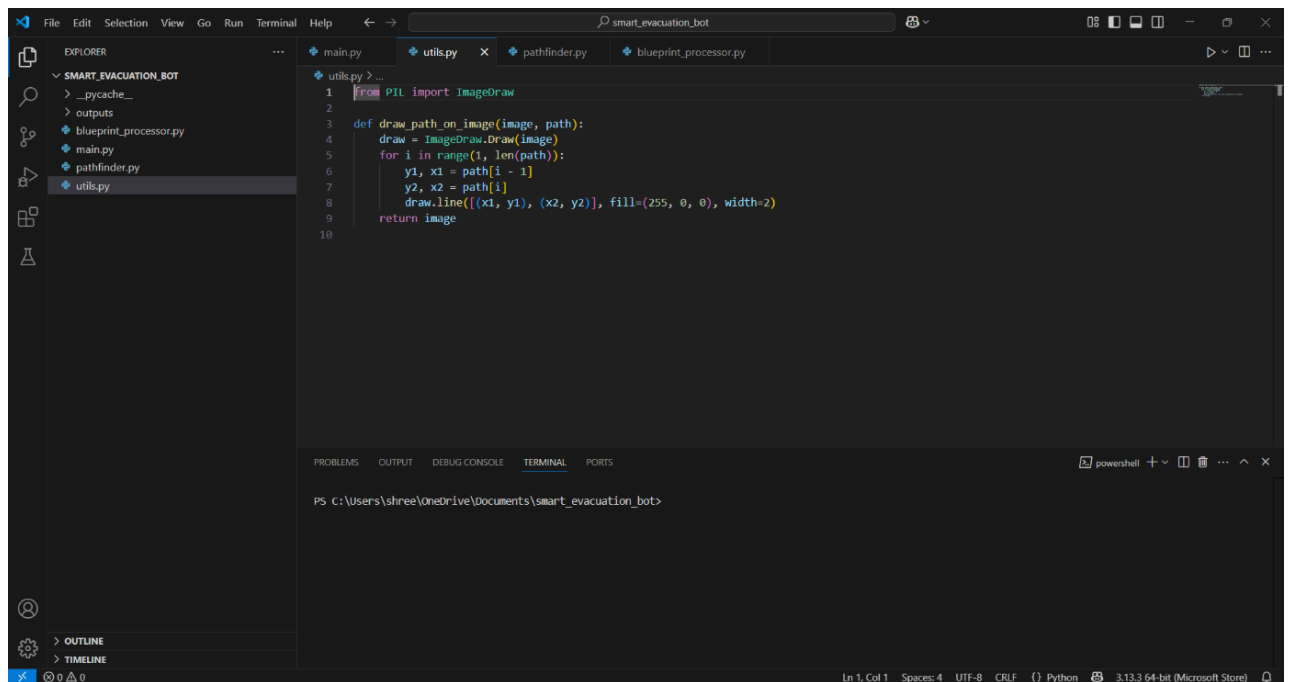
main.py



```
1 import os
2 from tkinter import Tk, Label, Button, Entry, StringVar, filedialog, messagebox
3 from blueprint_processor import process_blueprint
4 from pathfinder import compute_evacuation_path
5 from utils import draw_path_on_image
6
7 # Ensure output directory exists
8 os.makedirs('outputs', exist_ok=True)
9
10 def run_evacuation_planning():
11     disaster = disaster_var.get().lower()
12     people_count = people_var.get()
13
14     if disaster not in ('fire', 'earthquake'):
15         messagebox.showerror("Error", "Please enter disaster type: fire or earthquake")
16         return
17
18     if not people_count.isdigit() or int(people_count) <= 0:
19         messagebox.showerror("Error", "Please enter a valid number of people")
20         return
21
22     blueprint_path = filedialog.askopenfilename(
23         title="Select Blueprint Image",
```

PS C:\Users\shree\OneDrive\Documents\smart_evacuation_bot>

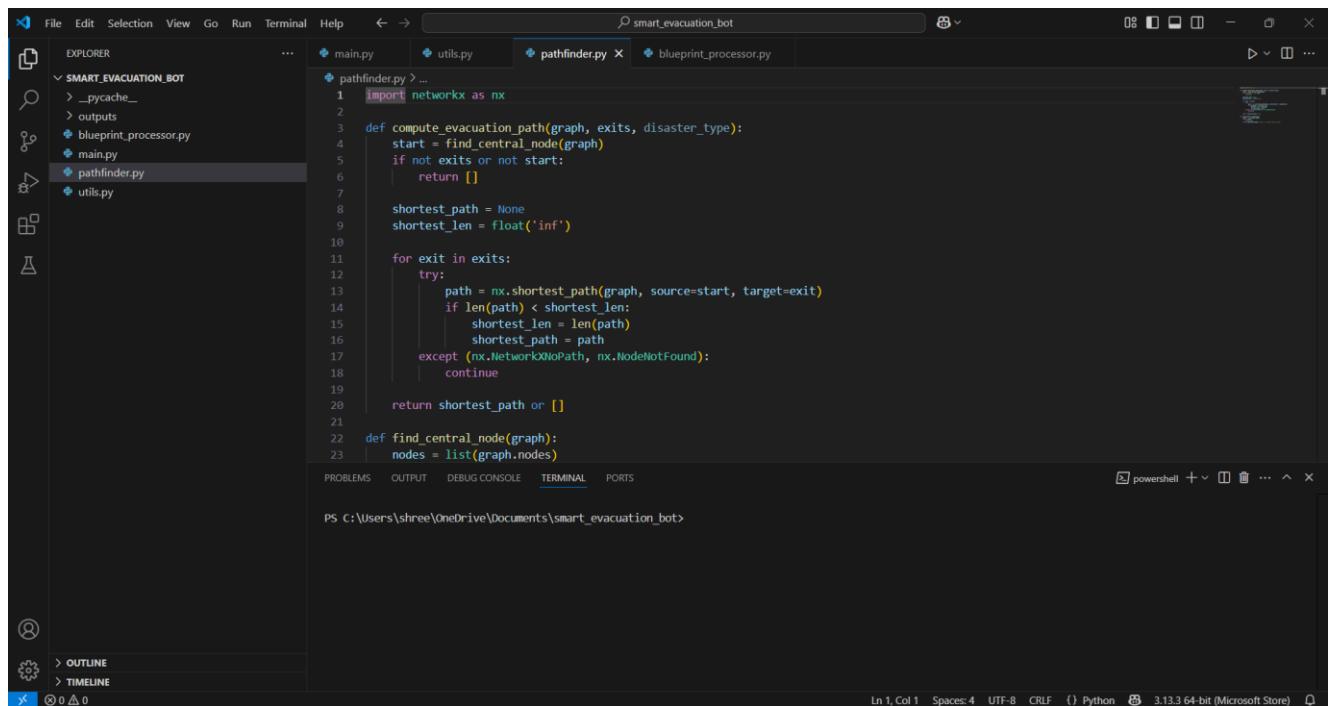
utils.py



```
1 from PIL import ImageDraw
2
3 def draw_path_on_image(image, path):
4     draw = ImageDraw.Draw(image)
5     for i in range(1, len(path)):
6         y1, x1 = path[i - 1]
7         y2, x2 = path[i]
8         draw.line([(x1, y1), (x2, y2)], fill=(255, 0, 0), width=2)
9     return image
10
```

PS C:\Users\shree\OneDrive\Documents\smart_evacuation_bot>

Pathfinder.py

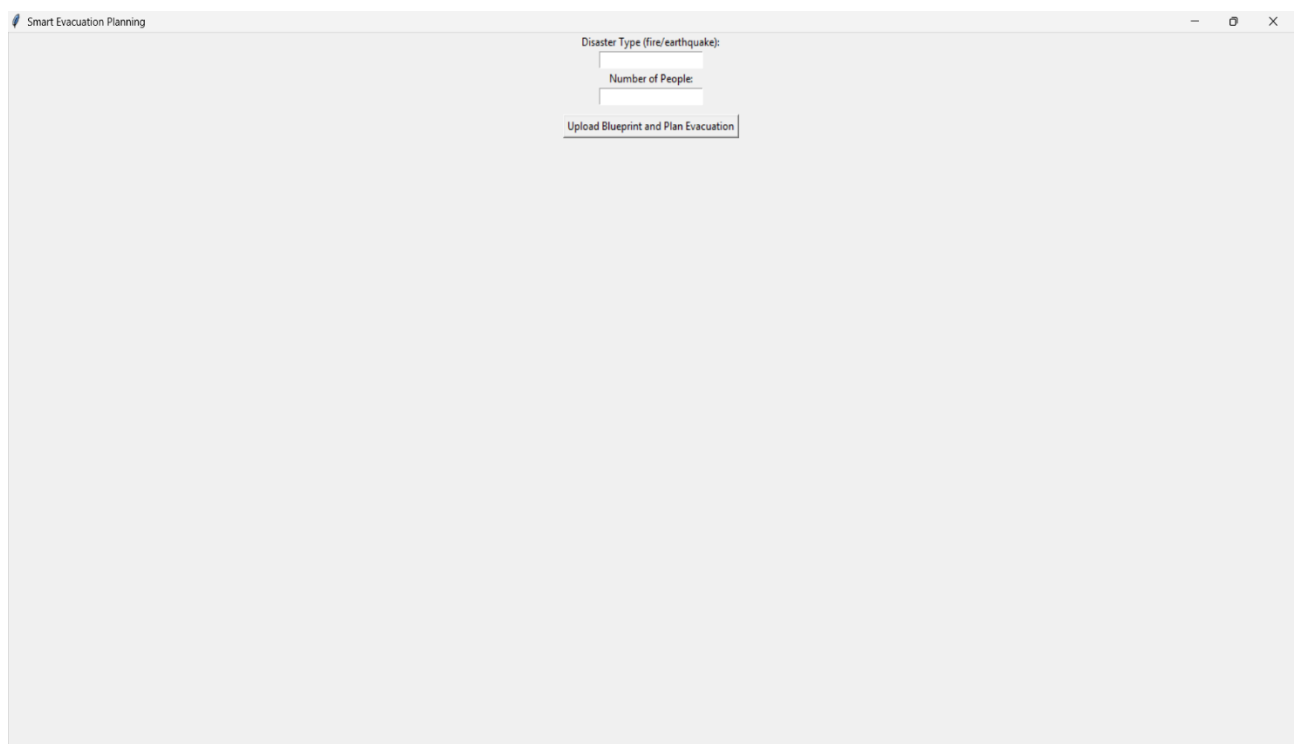


```
1 import networkx as nx
2
3 def compute_evacuation_path(graph, exits, disaster_type):
4     start = find_central_node(graph)
5     if not exits or not start:
6         return []
7
8     shortest_path = None
9     shortest_len = float('inf')
10
11     for exit in exits:
12         try:
13             path = nx.shortest_path(graph, source=start, target=exit)
14             if len(path) < shortest_len:
15                 shortest_len = len(path)
16                 shortest_path = path
17         except (nx.NetworkXNoPath, nx.NodeNotFound):
18             continue
19
20     return shortest_path or []
21
22 def find_central_node(graph):
23     nodes = list(graph.nodes)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shree\OneDrive\Documents\smart_evacuation_bot>

Sample output:



Smart Evacuation Planning

Disaster Type (fire/earthquake):

Number of People: