PROJECT TITLE :- ***RAPID GUARD***

NAMES OF THE MEMBERS :-

NAME :- ADITYA RAJ ROLL-NO:-92301733062

NAME :- MIHIR MALIKALI MITHANI ROLL-NO:-92301733025

NAME :- TANVI KAKKADIYA ROLL-NO:-92301733051

NAME :- SHIVANI AMBATI ROLL-NO:-92301733049

Project Explanation: "Rapid Guard"

Link for RapidGuard Google Site : ***RapidGuard*** *(Click to open.)*

The Rapid Guard project is a real-time incident reporting system designed to enhance community safety by quickly connecting available responders to incident locations. Using graph theory, specifically a ***Breadth-First Search (BFS)*** *algorithm for nearby member identification and a* ***bipartite graph*** *for assignment, this system efficiently matches incidents with the closest available responders.*

Project Abstract

The objective of the Rapid Guard project is to enable rapid responses during emergencies by leveraging graph theory algorithms to analyse member locations relative to an incident site. When an incident is reported through the Google site interface, the system identifies nearby members, assigns them to the incident based on proximity, and sends notifications to ensure a swift response. This project is a practical solution to reduce response times in emergencies, making it valuable for community safety and emergency services.

How the Project is Implemented with Python and Twilio

1. ***Incident Reporting****: A member reports an incident through a Google site, which captures details like type and location. This information is stored in a database.*

2. ***Member Matching Algorithm****:*

  - *The system uses **BFS** to find members near the incident location.*
  - *It applies a **bipartite graph algorithm** to pair incidents with available members, ensuring no member is assigned to multiple incidents simultaneously.*

3. ***Notification System**:*
  - *Once a match is made, the system uses Twilio's SMS API to notify the assigned members about the incident details, including location and response instructions.*

Algorithm for Graph-Based Incident Reporting

The following steps can add value if you want to understand relationships between incidents, centers, and members, such as identifying clusters, shortest paths, or coverage of centers.

Initialize the Graph Structure

Create an empty, undirected graph G using G = nx.Graph().

Add Nodes for Centers, Incidents, and Members

For each center in the member list (Book1.csv), add a node to G labeled with the center name.

For each incident in the incident report (Book2.csv), add a node to G labeled with the incident location.

For each member in the member list, add a node with the member's name or ID.

Optionally, include node attributes like type (e.g., center, incident, member) or location.

Add Edges Between Related Nodes

Incident to Center: For each incident, find the nearest center and add an edge between the incident node and the center node.

Center to Member: For each member, find their associated center and add an edge between the member node and their center node.

If you need more connections (e.g., members who can respond to multiple centers), add those connections accordingly.

Graph Analysis

Identify Clusters or Components: Use nx.connected_components(G) to find groups of nodes that are interconnected, indicating clusters of incidents and members who can respond within a particular center.

Calculate Shortest Paths (Optional): For optimized notification:

Use nx.shortest_path(G, source=incident, target=member) to find the shortest path from an incident to nearby members.

Use this to prioritize notifications to the closest members, potentially reducing response times.

Calculate Node Degree: Use G.degree(center_node) to determine the number of members connected to each center. Centers with low degree may need more member coverage.

Visualize the Graph: Plot the graph to view the network structure using matplotlib.

Graph-Based Notification Optimization (If required)

Use the shortest paths or cluster information to decide which members should be notified first.

Only notify members within the same connected component as the incident (if using connected components), reducing unnecessary notifications to members too far from the incident location.

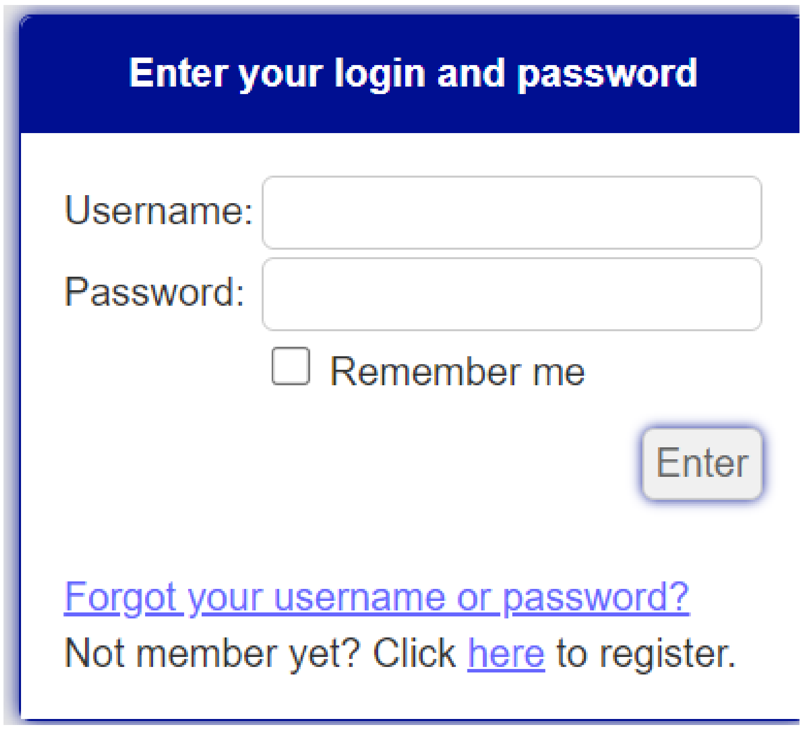
Display Graph Results or Output Analysis

After processing, update the UI to show analysis results, such as the number of connected components, clusters of incidents by center, or members with the shortest path to an incident.

```
# Initialize the graph

G = nx.Graph()


# Add nodes for centers, incidents, and members

for _, center in member_list['centre'].unique():

    G.add_node(center, type='center')


for _, incident in incident_report.iterrows():
    loc = incident['incident_location']
    centre = incident['nearest_centre']
    G.add_node(loc, type='incident')
    G.add_edge(loc, centre)  # Incident connected to nearest center

for _, member in member_list.iterrows():
    member_name = member['member_name']
    centre = member['centre']
    G.add_node(member_name, type='member')
    G.add_edge(member_name, centre)  # Member connected to their center

# Example: Find connected components
components = list(nx.connected_components(G))

# Example: Visualize the graph
import matplotlib.pyplot as plt
nx.draw(G, with_labels=True)
plt.show()
```

Flow Of The Project :-

Step 1 Login to the website with username and password assigned



Enter your login and password

Username:

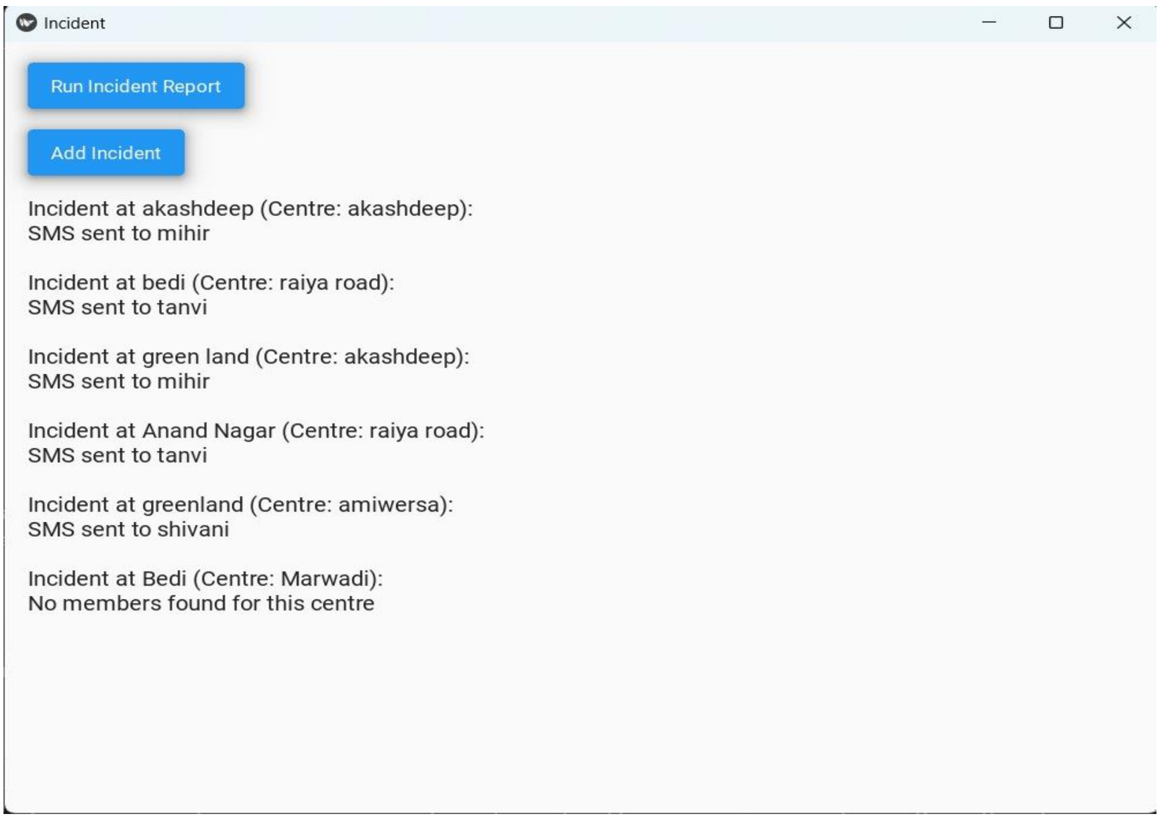
Password:

Remember me

Enter

Forgot your username or password?

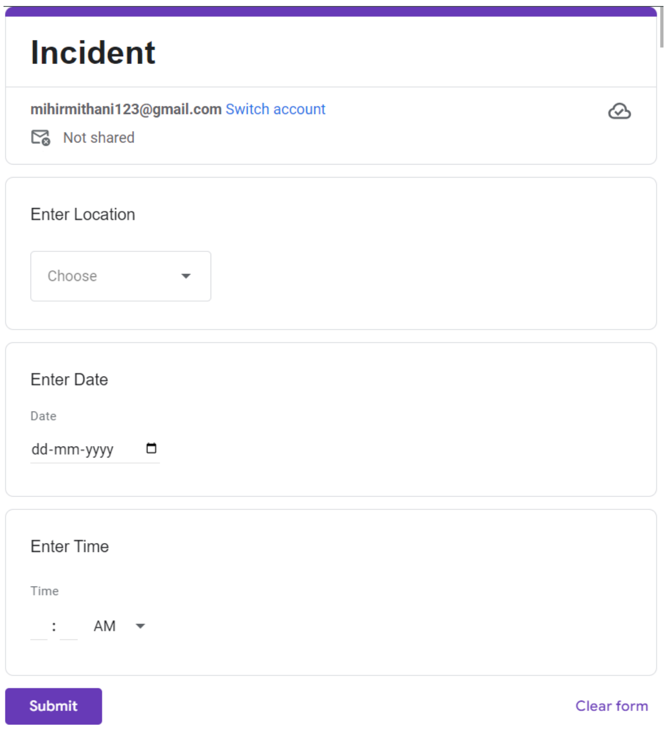
Not member yet? Click here to register.



Incident

```
Incident at akashdeep (Centre: akashdeep):
SMS sent to mihir

Incident at bedi (Centre: raiya road):
SMS sent to tanvi

Incident at green land (Centre: akashdeep):
SMS sent to mihir

Incident at Anand Nagar (Centre: raiya road):
SMS sent to tanvi

Incident at greenland (Centre: amiwersa):
SMS sent to shivani

Incident at Bedi (Centre: Marwadi):
No members found for this centre
```

Step 2 enter the details of the incident including location and nearest centre



Incident

mihirmithani123@gmail.com Switch account

Not shared

Enter Location

Choose

Enter Date

Date

dd-mm-yyyy

Enter Time

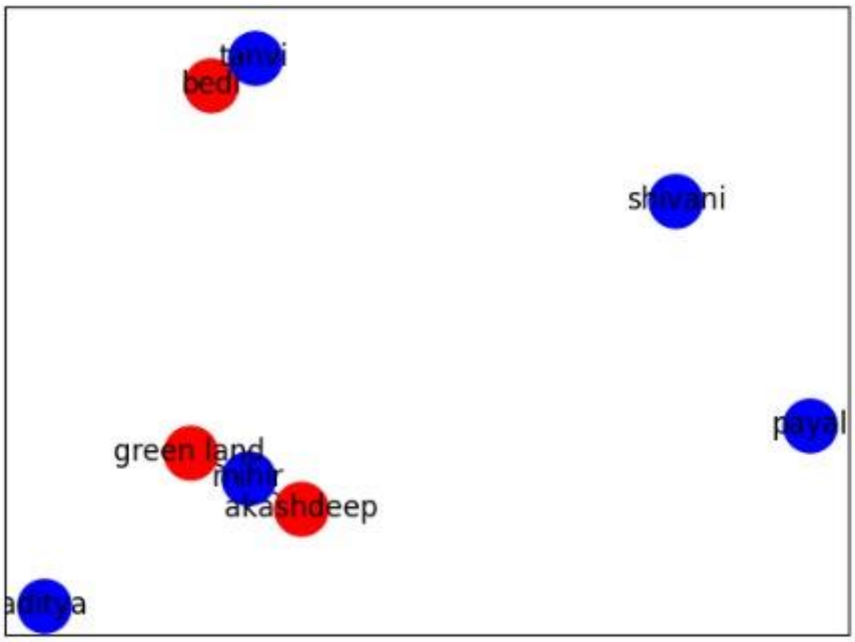
Time

: AM

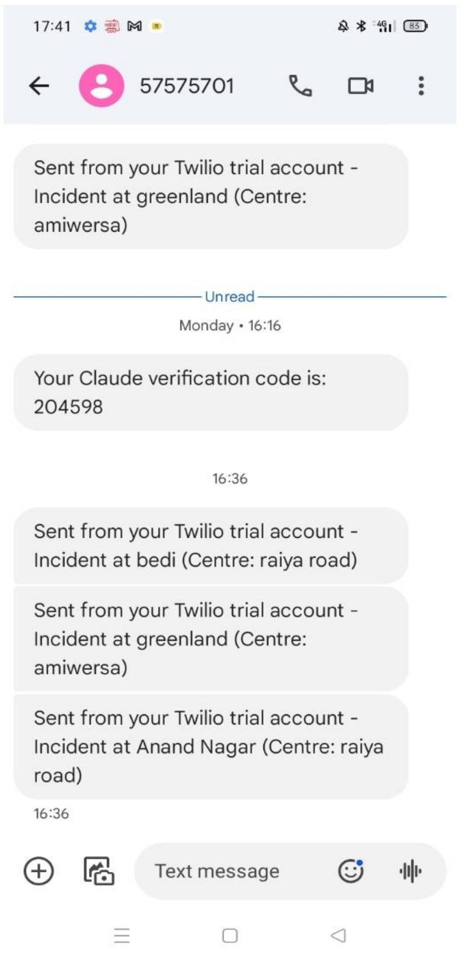
Submit

Clear form

Step 3: run the code and let the nearest member let know through sms(By RPM)



Step 5: Sending sms to the members assigned by the code.


17:41

57575701

Sent from your Twilio trial account - Incident at greenland (Centre: amiwersa)

Unread

Monday • 16:16

Your Claude verification code is: 204598

16:36

Sent from your Twilio trial account - Incident at bedi (Centre: raiya road)

Sent from your Twilio trial account - Incident at greenland (Centre: amiwersa)

Sent from your Twilio trial account - Incident at Anand Nagar (Centre: raiya road)

16:36