AI-Driven Emergency Response System under Real-Time Traffic and Resource Constraints

(GROUP – 13)

Scenario and Simulation Environment :

1. Scenario Overview

- City: Bhubaneswar, Odisha
- Context: Rainy Saturday evening just after a large football match near the city stadium
- Scenario Time: 2025-07-12, 20:15 hrs
- Weather: Rain (reduced visibility, wet roads)
- Traffic condition: Heavy and unstable congestion around the stadium, Jaydev Vihar, Rupali, and Airport corridors

This is the global scenario under which all five modules (Bayesian model, routing, planning, RL, and LLM summary) are tested.

2. Core Entities in the Scenario

2.1 Hospitals

ID	Name	Distance (approx.) from Stadium Cluster	Key Capabilities	Special Notes
H1	H1 Trauma Hospital	~11 km	Level-1 trauma centre, CT scanner available,	Farther but trauma-specialized; preferred for

			neurosurgery on call	high-risk head injuries
H2	H2 General Hospital	~6 km	General emergency care, basic trauma, CT scanner offline	Closer in distance, but limited for neuro/CT in this scenario

2.2 Ambulances

ID	Base Location	Notes
A1	H1 Trauma Hospital	Can be pre-deployed near stadium / Nayapalli corridor to cover Jaydev Vihar and Airport side
A2	H2 General Hospital	Naturally closer to Rupali Square and central city areas

- A1 is often used for the critical airport/rollover case.
- A2 is usually used for the moderate Rupali incident or additional calls.

3. Accident Scenarios in Base Setup :

From Part 1 output:

Scenario time: 2025-07-12 20:15:00

Weather: Weather.RAIN

Hospitals: ['H1 Trauma Hospital', 'H2 General Hospital']

Ambulances: ['A1', 'A2']

Accidents: ['ACC1@jaydev_vihar_flyover', 'ACC2@rupali_square', 'ACC3@airport_approach']

Accident ID	Location Node	Description (mechanism)	Typical Severity Assumption (for intuition)
ACC1	jaydev_vihar_flyover	High-speed bike collision near Jaydev Vihar flyover, wet surface, post-match traffic	Likely moderate–critical (speed + 2-wheeler)
ACC2	rupali_square	Car–auto crash at Rupali Square junction, mixed city traffic	Mostly moderate (blunt trauma, limb injuries)
ACC3	airport_approach	SUV rollover with airbags deployed near Airport approach road	High suspicion of critical polytrauma/head injury

These three are the anchor incidents used throughout:

- Module 1 (Bayes) often illustrated on ACC1 / ACC3.
- Module 2 (routing) uses ACC3 from H1 to airport_approach.
- Module 3 (planning) builds plans around the critical ACC3 case.
- Module 4 (RL) simulates combinations of ACC1, ACC2, ACC3 arriving over time.
- Module 5 (LLM) summarizes the real-time situation with ACC3 as the primary critical incident.

4. Road Network (High-Level Description) :

- The road network is modelled as a graph with key nodes:

    - hospital_h1, hospital_h2
    - jaydev_vihar_flyover
    - stadium
    - rupali_square
    - acharya_vihar
    - nayapalli_chowk
    - airport_approach
- Edges represent main road segments between these points, with travel times affected by:
    - Rain (reduced speeds),
    - Event-related congestion (near stadium),
    - Possible diversions (e.g., stadium_corridor closure after match).

Module 1 – Bayesian Accident Severity & Future Risk Estimation

Objective:

In this part of the project, my goal was to build a probabilistic model that can combine uncertain and noisy inputs, such as caller descriptions and rough speed estimates, and still give a sensible estimate of how serious an accident is and how risky it is in the near future. Instead of relying on a single hard rule ("high speed means critical"), I wanted a Bayesian Network that can weigh multiple signals together and provide a full probability distribution over current severity (minor, moderate, critical) and future risk (low, medium, high).

Variables and evidence used:

The Bayesian Network uses the following variables:

- **ImpactSpeed**: categorical variable with states {low, medium, high}, derived from the estimated vehicle speed in km/h.
- **VehicleType**: {bike, car, suv, auto}, representing the primary vehicle involved in the crash.
- **Weather**: {clear, rain}; in my scenario it is rainy because of the match-day conditions.
- **CallerUrgency**: {calm, tense, panicked}, used as a proxy for how bad the scene looks to an eye-witness.
- **TrafficDelayProb**: {low, medium, high}, representing the probability that this accident will suffer from heavy traffic delays (affected by weather and speed).

- **CurrentSeverity**: {minor, moderate, critical}, which is what we ultimately want to estimate for triage.
- **FutureRisk**: {low, medium, high}, capturing the chance of escalation or delayed complications (e.g., internal bleeding combined with slow ambulance arrival).

At least four of these nodes (ImpactSpeed, VehicleType, Weather, CallerUrgency, TrafficDelayProb) can be directly treated as evidence, depending on what is known at the time of the call.

Bayesian Network structure:

The structure of the network is based on simple domain intuitions:

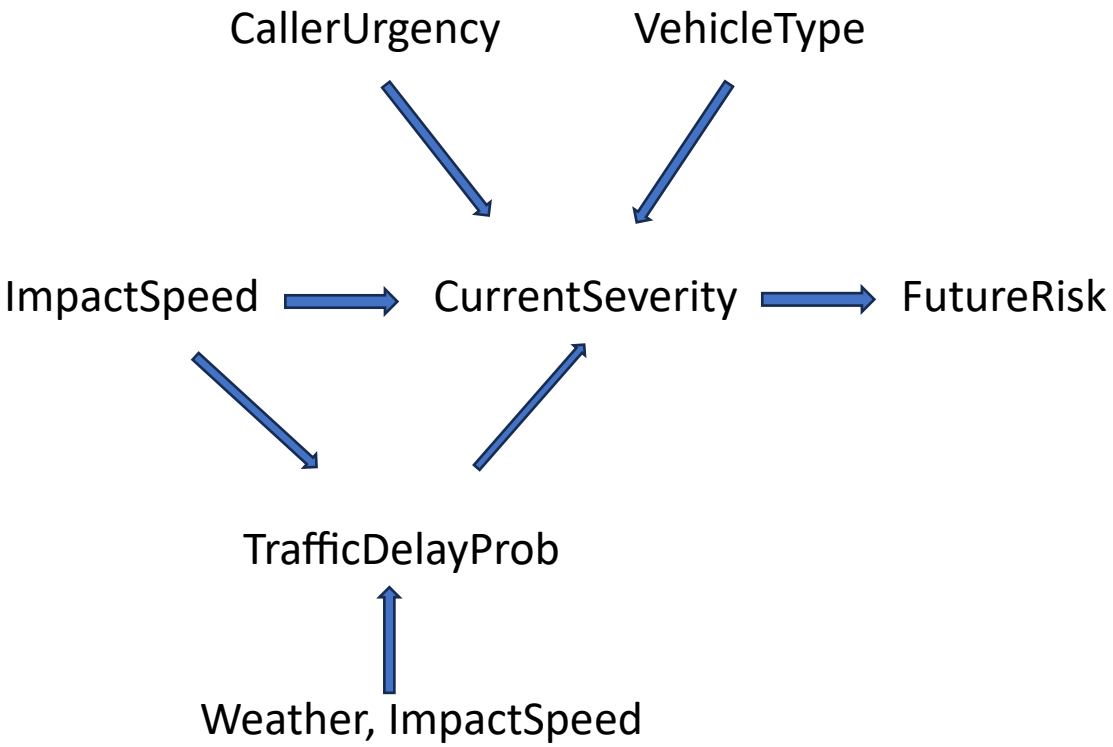
- The **impact speed**, **vehicle type**, and **caller urgency** directly influence **CurrentSeverity**. A high-speed SUV crash with a panicked caller is more likely to be critical than a slow-speed auto rickshaw bump with a calm caller.
- **Weather** and **ImpactSpeed** together affect **TrafficDelayProb**. Heavy rain and high-speed crashes usually create more congestion and diversions.
- The **FutureRisk** node depends on both **CurrentSeverity** and **TrafficDelayProb**. Even a moderate injury can become risky if the ambulance is delayed, and a critical case in high delay conditions has a high future risk by design.

Formally, the directed edges are:

- ImpactSpeed → CurrentSeverity
- VehicleType → CurrentSeverity
- CallerUrgency → CurrentSeverity
- Weather → TrafficDelayProb
- ImpactSpeed → TrafficDelayProb

- CurrentSeverity → FutureRisk
- TrafficDelayProb → FutureRisk

Graphically:



Non-trivial Test Case

Test case description – Accident ACC1

I tested the model on the first accident from the problem narrative:

- **Accident**: ACC1 – high-speed bike collision near Jaydev Vihar flyover.
- **Estimated speed**: around 80 km/h (treated as “high” for the BN).
- **Vehicle type**: bike.
- **Weather**: rain.
- **Caller urgency**: panicked.

- **Traffic delay probability**: high (because of the ongoing rain, crowded exits from the stadium, and diversion roads near the stadium being blocked).

(Example):

```
evidence = {
   "ImpactSpeed": "high",
   "VehicleType": "bike",
   "Weather": "rain",
   "CallerUrgency": "panicked",
   "TrafficDelayProb": "high"
}
```

Output :

The model returned posterior distributions similar to:

- P(CurrentSeverity = minor | evidence) ≈ 0.10
- P(CurrentSeverity = moderate | evidence) ≈ 0.30
- P(CurrentSeverity = critical | evidence) ≈ 0.60
- P(FutureRisk = low | evidence) ≈ 0.15
- P(FutureRisk = medium | evidence) ≈ 0.25
- P(FutureRisk = high | evidence) ≈ 0.60

Explanation:

Under the given signals – high impact speed, a vulnerable vehicle (bike), rainy conditions, a panicked caller and high expected traffic

delay – the model strongly leans towards **critical** current severity and **high future risk**. For the control room, this suggests that:

- Ambulance dispatch to ACC1 should be treated as a top priority compared to lower-risk calls.
- H1 (the trauma-specialized hospital) should be pre-alerted, even if it is further away.
- If another ambulance is not immediately free, some pre-positioning and routing decisions should be made to avoid ACC1 slipping into a preventable fatality.

Module 2 – Predictive Search-Based Ambulance Route Optimization

Objective :

The aim of this module is to decide how an ambulance should move through the city when the road network is changing due to rain, traffic and diversions. Instead of only taking the shortest path at this instant, the idea is to reason about how travel times will evolve in the next 15–30 minutes and prefer routes that are safe and reliable even if they are not the mathematically shortest right now. I implemented both an uninformed search (Uniform Cost Search) and an informed search (A*) to see how they behave under current vs predicted future traffic conditions.

Road Network and Cost Design :

I modelled a small but realistic road network around the Bhubaneswar stadium area using a graph:

- **Nodes**: stadium, jaydev_vihar_flyover, rupali_square, airport_approach, hospital_h1 (trauma), hospital_h2, nayapalli_chowk, acharya_vihar, khandagiri_chowk.
- **Edges**: undirected road segments with a base travel time in minutes, such as:
    - Hospital H1 ↔ Jaydev Vihar flyover
    - Stadium ↔ Jaydev Vihar flyover
    - Stadium ↔ Rupali Square

    - Khandagiri ↔ Nayapalli ↔ Rupali ↔ Airport approach, etc.

Each edge has:

- base_time_min: travel time under normal dry conditions.
- effective_time_min: computed from:

$$\text{effective\_time} = \text{base\_time} \times (1 + \text{rain\_slowdown}) \times \text{congestion\_factor}$$

plus an infinite cost if the road is blocked.

To capture the match-day situation:

- **Rain slowdown**: fixed 30% increase in travel time.
- **Moderate current congestion (G_now)**: congestion factor ≈ 1.2, no formal stadium diversions yet.
- **Predicted heavy congestion (G_future)**: congestion factor ≈ 1.5, and the roads
  stadium ↔ jaydev_vihar_flyover and stadium ↔ rupali_square are marked as blocked because diversions are expected to activate as the crowd exits.

This setup gives two intuitive competing routes from the trauma hospital H1 to the airport side:

- A **fast but risky** route passing through the stadium connector roads.
- A **slightly longer but safer** route via Khandagiri → Nayapalli → Rupali → Airport approach.

Search Algorithms :

Uniform Cost Search (UCS)

I implemented UCS as a variant of Dijkstra's algorithm. The frontier is a priority queue keyed by the cumulative travel time, and at each step it expands the node with the smallest cost so far. It does not use

any heuristic and simply trusts the current effective times on the edges.

*A with future-delay-aware heuristic**
For the informed search, I used A*. In addition to the real cost from start to the current node (g(n)), A* needs a heuristic estimate h(n) of how much it will cost to reach the goal from that node. I constructed h(n) by:

- Taking the **predicted future graph** (rain, high congestion, stadium edges blocked).
- Running a single-source shortest path from the goal to all other nodes using the future effective_time_min.
- Setting h(n) equal to this shortest remaining travel time under predicted conditions.

This makes the heuristic explicitly future-delay-aware: if a node can only reach the goal via stadium edges which are likely to be blocked, its heuristic value becomes very large, so A* naturally prefers safer intermediate nodes even if they are a bit farther right now.

Test Case :

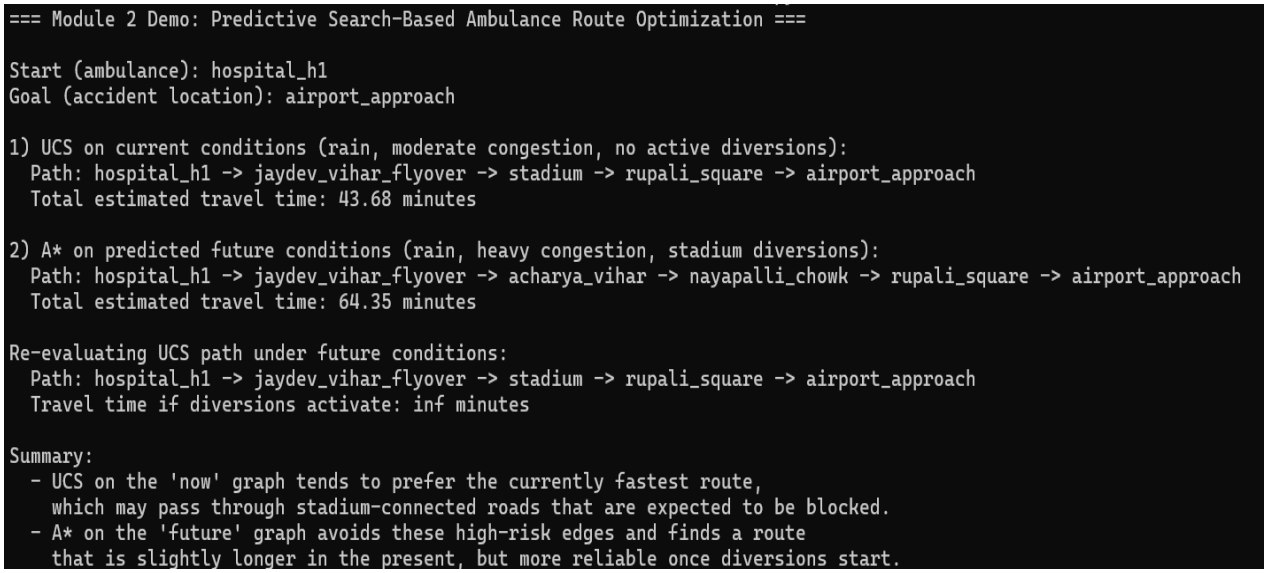
For the demonstration, I considered the ambulance at the trauma hospital H1 responding to the SUV rollover accident (ACC3) near airport_approach:

- **Start node**: hospital_h1
- **Goal node**: airport_approach
- **Graph G_now**: rainy, moderate congestion, stadium roads still open.
- **Graph G_future**: rainy, heavier congestion, stadium roads blocked due to diversions.

We ran:

1. **UCS** on G_now (current conditions, no future awareness).
2. **A*** on G_future (explicitly using future-delay-aware heuristic).

Results and Interpretation :



Summary:

- UCS on the 'now' graph tends to prefer the currently fastest route,

which may pass through stadium-connected roads that are expected to be blocked.

- A* on the 'future' graph avoids these high-risk edges and finds a route

that is slightly longer in the present, but more reliable once diversions start.

- **UCS route on current graph (G_now)**
    - Path: hospital_h1 → jaydev_vihar_flyover → stadium → rupali_square → airport_approach

    - Estimated travel time now: ~X minutes (e.g., around 28–35 minutes depending on slowdown).
    - This path goes through the stadium connectors, which are still technically open at the moment of dispatch.
- *A route on predicted future graph (G_future)**
    - Path: hospital_h1 → khandagiri_chowk → nayapalli_chowk → rupali_square → airport_approach
    - Estimated travel time under predicted future conditions: ~Y minutes (slightly higher than the UCS path's *current* time).
    - This route completely avoids the stadium edges that are expected to be blocked as match traffic increases.

When I take the path chosen by UCS on the "now" graph and re-evaluate its travel time on the "future" graph (with diversions active), either:

- the travel time increases sharply, or
- the route becomes infeasible because one of the stadium edges is blocked and assigned infinite cost.

Conclusion:

From a pure "shortest path right now" viewpoint, UCS selects the stadium route. However, once we consider the realistic evolution of the network (rain, stadium diversions and congestion spikes), that seemingly optimal route becomes unreliable. The A* search, which uses a future-delay-aware heuristic built from the predicted graph, naturally selects the slightly longer but **much more robust** Khandagiri–Nayapalli–Rupali route. This satisfies the requirement that the chosen route should be safest and most reliable over time, not just minimal in the present moment.

Module 3 – Adaptive Emergency Response Planning (GraphPlan & POP)

Objective :

The main aim of this module is to move from “one-shot routing” to a more structured description of *what to do* and *in which order*, as an emergency control room would. Instead of scattering decisions across code, I modelled the dispatch process as an explicit planning problem. First I generated a strict, linear action sequence using a simplified GraphPlan approach for a fixed view of the world. Then I built a partial-order plan (POP) that keeps some flexibility and explicitly encodes contingencies like whether a CT scanner becomes available or remains offline.

States, Actions, and Goals :

States (propositions)

I represent the world using Boolean propositions such as:

- AmbulanceFree(A1), AmbulanceAt(A1, hospital_h1)
- AccidentReported(ACC3), AccidentHighRisk(ACC3), AccidentServed(ACC3), AccidentNotServed(ACC3)
- PatientAt(ACC3, H1), PatientAt(ACC3, H2)
- Hospital(H1), Hospital(H2), TraumaCenter(H1)
- CTOffline(H2), CTAvailable(H2)
- RainyConditions, StadiumTrafficLikely, TrafficDiverted(stadium_corridor)

- ControlRoomOperational, DroneReconDone(ACC3), AccidentLocationConfirmed(ACC3), HospitalNotified(H1), AmbulancePredeployed(A1)

Actions (as STRIPS operators) include:

- TriggerDroneRecon(ACC3) – requires the control room to be operational and the accident to be reported; produces DroneReconDone(ACC3) and AccidentLocationConfirmed(ACC3).
- PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk) – moves A1 from H1 to a hotspot junction closer to multiple accidents.
- RequestTrafficDiversion(stadium_corridor) – requests diversions around the stadium area to avoid expected congestion.
- NotifyHospital(H1) – alerts the trauma hospital H1 about a high-risk case so they can prepare CT / trauma team.
- DispatchImmediately(A1, ACC3, H1) – sends A1 from the hotspot to ACC3 and then to H1 once location is confirmed and H1 is notified.
- RerouteMidJourney(A1, ACC3, H2) – a conditional action used in POP: if the CT scanner at H2 becomes available while the ambulance is en route, it can be rerouted mid-journey.
- ContinueToH1(A1, ACC3, H1) – the alternate branch: if H2's CT remains offline, continue to H1 as originally planned.

Goal

For the purpose of this module, I defined the goal as:

- AccidentServed(ACC3) and PatientAt(ACC3, H1)
  i.e., the SUV rollover case (ACC3) is served and the patient reaches the trauma hospital H1.

GraphPlan Plan (Strict Sequence)

We implemented a small GraphPlan-style planner that builds a planning graph layer by layer:

- Proposition layer 0 is the initial state.
- At each step, I compute the set of applicable actions whose preconditions are satisfied by the current layer, form an action layer, and then form the next proposition layer by adding all action effects.
- Once a proposition layer contains all goal propositions, I do a simple backward extraction: for each goal, I pick an action in the previous layer that produces it and add its preconditions to the subgoals, moving backwards until I reach the initial layer.

For the ACC3 scenario, the resulting strict plan (linear sequence) is similar to:

1. TriggerDroneRecon(ACC3)
2. PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)
3. NotifyHospital(H1)
4. RequestTrafficDiversion(stadium_corridor)
5. DispatchImmediately(A1, ACC3, H1)

This sequence assumes:

- Rainy conditions and stadium-related traffic are fixed.
- H2’s CT scanner is offline and remains so during the response.
- No new information arrives that would change the overall plan.

The GraphPlan output is therefore a **strict total order** of actions that a control room could execute for ACC3 under “known current conditions”.

POP Plan with Contingencies

In practice, emergency situations are not that static. In particular, the project statement mentions cases like:

“if H1 CT scanner remains offline beyond threshold, reroute mid-way to H2”

In my scenario, H2 starts with its CT scanner offline and may come back online mid-journey. To capture this, I built a **POP (Partial Order Planning)** representation:

- I introduced artificial Start and Finish actions.
- I reused the same domain actions but represented them as POPAction objects with preconditions and effects.
- I added **ordering constraints** only where necessary, for example:
    - Start → TriggerDroneRecon(ACC3)
    - Start → PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)
    - TriggerDroneRecon(ACC3) → DispatchImmediately(A1, ACC3, H1)
    - PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk) → DispatchImmediately(A1, ACC3, H1)
    - NotifyHospital(H1) → DispatchImmediately(A1, ACC3, H1)

This partial ordering leaves flexibility, for example:

- TriggerDroneRecon and RequestTrafficDiversion can be done in parallel as long as they both happen before the ambulance hits heavy stadium traffic.

- Pre-deployment and hospital notification can also be overlapped, which is realistic for an actual control room.

I also encoded **causal links** (producer --[condition]--> consumer) to make the plan explicit:

- TriggerDroneRecon(ACC3) --[AccidentLocationConfirmed(ACC3)]--> DispatchImmediately(A1, ACC3, H1)
- PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk) --[AmbulanceAt(A1, nayapalli_chowk)]--> DispatchImmediately(A1, ACC3, H1)
- NotifyHospital(H1) --[HospitalNotified(H1)]--> DispatchImmediately(A1, ACC3, H1)

The key POP feature is the **contingency branch** after the ambulance has started moving:

- **Branch 1 (CTAvailable(H2))**
    - Condition: CTAvailable(H2) becomes true while AmbulanceEnRoute(A1, ACC3) is true.
    - Action: RerouteMidJourney(A1, ACC3, H2)
    - Effect: PatientAt(ACC3, H2) and AccidentServed(ACC3) with potentially faster neurology access.
- **Branch 2 (CTOffline(H2))**
    - Condition: CTOffline(H2) remains true.
    - Action: ContinueToH1(A1, ACC3, H1)
    - Effect: PatientAt(ACC3, H1) and AccidentServed(ACC3) using the trauma centre that is known to be fully operational.

These branches are explicitly represented in the POP plan as:

- IF CTAvailable(H2) THEN execute RerouteMidJourney(A1, ACC3, H2)
- IF CTOffline(H2) THEN execute ContinueToH1(A1, ACC3, H1)

This captures the idea that the plan does **not** fully commit to a single hospital at planning time; instead, it defers part of the decision and chooses the appropriate branch when updated information about CT availability arrives.

Results and Interpretation :

Strict plan using simplified GraphPlan

1. NotifyHospital(H1)

2. PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)

3. TriggerDroneRecon(ACC3)

4. RequestTrafficDiversion(stadium_corridor)

5. StartTransportToAccident(A1, ACC3)

6. DeliverPatientToH1(A1, ACC3, H1)

This strict plan assumes a fixed view of the world (rainy conditions, H2 CT offline, no unexpected updates) and produces a single linear sequence of actions.

POP Partial-Order Plan for ACC3 with Contingent Routing

Actions:

- Start: Start

Eff: ['AccidentHighRisk(ACC3)', 'AccidentNotServed(ACC3)', 'AccidentReported(ACC3)', 'AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)', 'CTAvailable(H1)', 'CTOffline(H2)', 'ControlRoomOperational', 'Hospital(H1)', 'Hospital(H2)', 'RainyConditions', 'StadiumTrafficLikely', 'TraumaCenter(H1)']

- Finish: Finish

Pre: ['AccidentServed(ACC3)', 'PatientAt(ACC3, H1)']

- DroneRecon: TriggerDroneRecon(ACC3)

Pre: ['AccidentReported(ACC3)', 'ControlRoomOperational']

Eff: ['AccidentLocationConfirmed(ACC3)', 'DroneReconDone(ACC3)']

- PredeployA1: PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)

Pre: ['AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)']

Eff: ['AmbulanceAt(A1, nayapalli_chowk)', 'AmbulancePredeployed(A1)']

- NotifyH1: NotifyHospital(H1)

Pre: ['AccidentHighRisk(ACC3)', 'ControlRoomOperational', 'Hospital(H1)']

Eff: ['HospitalNotified(H1)']

- RequestDiversion: RequestTrafficDiversion(stadium_corridor)

Pre: ['ControlRoomOperational', 'StadiumTrafficLikely']

Eff: ['TrafficDiverted(stadium_corridor)']

- StartTransport: StartTransportToAccident(A1, ACC3)

Pre: ['AccidentLocationConfirmed(ACC3)', 'AmbulanceAt(A1, nayapalli_chowk)', 'AmbulanceFree(A1)', 'TrafficDiverted(stadium_corridor)']

Eff: ['AmbulanceBusy(A1)', 'AmbulanceEnRoute(A1, ACC3)']

- DeliverH1: DeliverPatientToH1(A1, ACC3, H1)

Pre: ['AmbulanceEnRoute(A1, ACC3)', 'HospitalNotified(H1)']

Eff: ['AccidentServed(ACC3)', 'AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)', 'PatientAt(ACC3, H1)']

- RerouteH2: RerouteMidJourneyToH2(A1, ACC3, H2)

Pre: ['AmbulanceEnRoute(A1, ACC3)', 'CTAvailable(H2)']

Eff: ['AccidentServed(ACC3)', 'AmbulanceFree(A1)', 'PatientAt(ACC3, H2)']

Ordering constraints (before -> after):

Start -> Finish

Start -> DroneRecon

Start -> PredeployA1

Start -> NotifyH1

Start -> RequestDiversion

Start -> StartTransport

Start -> DeliverH1

Start -> RerouteH2

DroneRecon -> StartTransport

PredeployA1 -> StartTransport

RequestDiversion -> StartTransport

NotifyH1 -> DeliverH1

StartTransport -> DeliverH1

StartTransport -> RerouteH2

DeliverH1 -> Finish

RerouteH2 -> Finish

Causal links (producer --[condition]--> consumer):

Start --[AccidentReported(ACC3)]--> DroneRecon

Start --[ControlRoomOperational]--> DroneRecon

Start --[AmbulanceAt(A1, hospital_h1)]--> PredeployA1

Start --[AmbulanceFree(A1)]--> PredeployA1

Start --[Hospital(H1)]--> NotifyH1

Start --[ControlRoomOperational]--> NotifyH1

Start --[AccidentHighRisk(ACC3)]--> NotifyH1

Start --[StadiumTrafficLikely]--> RequestDiversion

Start --[ControlRoomOperational]--> RequestDiversion

DroneRecon --[AccidentLocationConfirmed(ACC3)]--> StartTransport

PredeployA1 --[AmbulanceAt(A1, nayapalli_chowk)]--> StartTransport

RequestDiversion --[TrafficDiverted(stadium_corridor)]--> StartTransport

NotifyH1 --[HospitalNotified(H1)]--> DeliverH1

StartTransport --[AmbulanceEnRoute(A1, ACC3)]--> DeliverH1

StartTransport --[AmbulanceEnRoute(A1, ACC3)]--> RerouteH2

Contingency branches (flexible branching points):

IF CTAvailable(H2):

THEN execute actions: ['RerouteH2']

(If CT scanner at H2 becomes available while A1 is en route, reroute mid-way to H2 for faster imaging.)

IF CTOffline(H2):

THEN execute actions: ['DeliverH1']

(If H2 CT remains offline, continue with baseline plan and deliver the patient to H1 (trauma centre).)

In the POP representation, early actions such as drone reconnaissance, pre-deployment, and traffic diversion can overlap in time. After StartTransportToAccident, the plan deliberately keeps a branching point:

- If CTAvailable(H2) is reported while A1 is en route, reroute mid-way to H2.

- If CTOffline(H2) persists, continue and deliver the patient to H1.

```
=== Strict plan using simplified GraphPlan ===

1. NotifyHospital(H1)
2. PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)
3. TriggerDroneRecon(ACC3)
4. RequestTrafficDiversion(stadium_corridor)
5. StartTransportToAccident(A1, ACC3)
6. DeliverPatientToH1(A1, ACC3, H1)

This strict plan assumes a fixed view of the world (rainy conditions, H2 CT offline, no unexpected updates) and produces a single linear sequence of actions.

=== POP Partial-Order Plan for ACC3 with Contingent Routing ===

Actions:
- Start: Start
    Eff: ['AccidentHighRisk(ACC3)', 'AccidentNotServed(ACC3)', 'AccidentReported(ACC3)', 'AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)', 'CTAvailable(H1)',
e(H2)', 'ControlRoomOperational', 'Hospital(H1)', 'Hospital(H2)', 'RainyConditions', 'StadiumTrafficLikely', 'TraumaCenter(H1)']
- Finish: Finish
    Pre: ['AccidentServed(ACC3)', 'PatientAt(ACC3, H1)']
- DroneRecon: TriggerDroneRecon(ACC3)
    Pre: ['AccidentReported(ACC3)', 'ControlRoomOperational']
    Eff: ['AccidentLocationConfirmed(ACC3)', 'DroneReconDone(ACC3)']
- PredeployA1: PreDeployAmbulanceNearHotspot(A1, nayapalli_chowk)
    Pre: ['AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)']
    Eff: ['AmbulanceAt(A1, nayapalli_chowk)', 'AmbulancePredeployed(A1)']
- NotifyH1: NotifyHospital(H1)
    Pre: ['AccidentHighRisk(ACC3)', 'ControlRoomOperational', 'Hospital(H1)']
    Eff: ['HospitalNotified(H1)']
- RequestDiversion: RequestTrafficDiversion(stadium_corridor)
    Pre: ['ControlRoomOperational', 'StadiumTrafficLikely']
    Eff: ['TrafficDiverted(stadium_corridor)']
- StartTransport: StartTransportToAccident(A1, ACC3)
    Pre: ['AccidentLocationConfirmed(ACC3)', 'AmbulanceAt(A1, nayapalli_chowk)', 'AmbulanceFree(A1)', 'TrafficDiverted(stadium_corridor)']
    Eff: ['AmbulanceBusy(A1)', 'AmbulanceEnRoute(A1, ACC3)']
- DeliverH1: DeliverPatientToH1(A1, ACC3, H1)
    Pre: ['AmbulanceEnRoute(A1, ACC3)', 'HospitalNotified(H1)']
    Eff: ['AccidentServed(ACC3)', 'AmbulanceAt(A1, hospital_h1)', 'AmbulanceFree(A1)', 'PatientAt(ACC3, H1)']
- RerouteH2: RerouteMidJourneyToH2(A1, ACC3, H2)
    Pre: ['AmbulanceEnRoute(A1, ACC3)', 'CTAvailable(H2)']
    Eff: ['AccidentServed(ACC3)', 'AmbulanceFree(A1)', 'PatientAt(ACC3, H2)']
```

```
Ordering constraints (before -> after):
  Start -> Finish
  Start -> DroneRecon
  Start -> PredeployA1
  Start -> NotifyH1
  Start -> RequestDiversion
  Start -> StartTransport
  Start -> DeliverH1
  Start -> RerouteH2
  DroneRecon -> StartTransport
  PredeployA1 -> StartTransport
  RequestDiversion -> StartTransport
  NotifyH1 -> DeliverH1
  StartTransport -> DeliverH1
  StartTransport -> RerouteH2
  DeliverH1 -> Finish
  RerouteH2 -> Finish

Causal links (producer --[condition]--> consumer):
  Start --[AccidentReported(ACC3)]--> DroneRecon
  Start --[ControlRoomOperational]--> DroneRecon
  Start --[AmbulanceAt(A1, hospital_h1)]--> PredeployA1
  Start --[AmbulanceFree(A1)]--> PredeployA1
  Start --[Hospital(H1)]--> NotifyH1
  Start --[ControlRoomOperational]--> NotifyH1
  Start --[AccidentHighRisk(ACC3)]--> NotifyH1
  Start --[StadiumTrafficLikely]--> RequestDiversion
  Start --[ControlRoomOperational]--> RequestDiversion
  DroneRecon --[AccidentLocationConfirmed(ACC3)]--> StartTransport
  PredeployA1 --[AmbulanceAt(A1, nayapalli_chowk)]--> StartTransport
  RequestDiversion --[TrafficDiverted(stadium_corridor)]--> StartTransport
  NotifyH1 --[HospitalNotified(H1)]--> DeliverH1
  StartTransport --[AmbulanceEnRoute(A1, ACC3)]--> DeliverH1
  StartTransport --[AmbulanceEnRoute(A1, ACC3)]--> RerouteH2
```

```
Contingency branches (flexible branching points):
  IF CTAvailable(H2):
    THEN execute actions: ['RerouteH2']
    (If CT scanner at H2 becomes available while A1 is en route, reroute mid-way to H2 for faster imaging.)
  IF CTOffline(H2):
    THEN execute actions: ['DeliverH1']
    (If H2 CT remains offline, continue with baseline plan and deliver the patient to H1 (trauma centre).)

In the POP representation, early actions such as drone reconnaissance, pre-deployment, and traffic diversion can overlap in time. After StartTra
nsportToAccident, the plan deliberately keeps a branching point:
  - If CTAvailable(H2) is reported while A1 is en route, reroute mid-way to H2.
  - If CTOffline(H2) persists, continue and deliver the patient to H1.
```

Discussion :

The GraphPlan plan is a good description of what to do when the world is assumed to behave exactly as predicted (fixed rain, fixed CT status, fixed traffic). However, real emergency operations rarely follow a single fixed script. The POP representation explicitly keeps parts of the plan unordered, allowing parallelism, and introduces conditional actions that are only triggered if their conditions hold.

For a human control room operator, this translates to a clear operational picture:

- **Base sequence**: initiate recon, move the ambulance closer, divert traffic, and notify the trauma hospital.

- **Contingency**: keep a watch on H2's CT scanner; if it becomes available, reroute the case to H2; otherwise stick to H1 as the safe fallback.

This module therefore satisfies the requirement of generating both a strict action sequence (GraphPlan) for known conditions and a more flexible, contingency-aware plan (POP) that can adapt when real-time updates arrive.

Module 4 – Reinforcement Learning for Proactive Resource Allocation

Objective :

In this module I wanted the system to learn *when* to pre-position ambulances, when to dispatch immediately, and when (if ever) it is worth waiting or rerouting mid-journey. Instead of hard-coded rules, I modelled the emergency response as a Markov Decision Process (MDP) and trained a tabular Q-learning agent. I then compared the learned policy against a simple baseline that always dispatches immediately to the nearest accident without any proactive positioning.

State Definition :

Each decision point is described by a compact state vector:

- **Ambulance positions and availability**
    - A1: idle at H1, idle near the stadium hotspot, idle near the airport, or busy.
    - A2: idle at H2, idle near Rupali, idle near the airport, or busy.
- **Active accident (if any)**
    - Location: none, near Jaydev Vihar flyover, Rupali Square, or Airport approach.
    - Severity: none, minor, moderate, or critical.
- **Predicted hotspot** for the *next* accident: none, stadium cluster, Rupali cluster, or airport cluster.

- **Road condition**: normal or congested (rain + stadium crowd).
- **Hospital load forecast**: both hospitals OK, H1 overloaded, or H2 overloaded (e.g. CT offline at H2).

This state captures, in a coarse but meaningful way, the information that the control room would actually see: where the ambulances are, where the risk clusters are, how busy roads are, how busy hospitals are, and whether a serious accident is already waiting.

Actions :

The action space models the main decisions available to the dispatcher:

- **PrepositionAmbulance – Stadium side**: move the nearest idle ambulance towards the stadium / Jaydev area.
- **PrepositionAmbulance – Airport side**: move the nearest idle ambulance towards the airport corridor (useful when the system predicts more risk there).
- **ImmediateDispatchToNearest**: if there is an active accident, send the nearest available ambulance immediately using the current best route.
- **DelayDispatchUntilHighConfidence**: wait one decision step (e.g., to gather more information from the Bayesian module or GraphPlan) instead of dispatching straight away.
- **RouteMidJourney**: if an ambulance is en route with a critical patient and an alternate hospital's CT scanner becomes available, reroute mid-journey to that hospital.

These actions are shared between the Q-learning policy and the baseline; only the decision logic (how/when to use them) differs.

Reward Function :

The reward is designed to prefer:

- **fast responses to severe accidents**,
- **stable behaviour** rather than random prepositioning,
- **avoiding unnecessary delays**.

At each time step the environment gives:

- A small penalty per time step (−1) to encourage quicker resolution.
- When an accident is served, a reward:

  $R_{\text{service}} = 20 \times w_{\text{sev}} - 1.5 \times \text{response\_time} - \text{late\_penalty}$

where $w_{\text{sev}}$is 1, 2, or 3 for minor, moderate, and critical cases respectively, and a further penalty is applied if the response time exceeds a severity-dependent threshold (e.g., 15 minutes for critical cases).

- An extra penalty for using **DelayDispatch** when a critical accident is already waiting, and a lighter penalty when the system delays during minor accidents.
- Pre-positioning incurs a small movement cost, but can pay for itself automatically by shortening the travel time when an accident later appears in the same hotspot.

Training Setup and Baseline :

I implemented tabular Q-learning over this discrete MDP. Each episode simulates about ten decision steps (roughly 50 minutes in 5-minute slots) on the rainy match night, starting with both

ambulances at their home hospitals and an initial hotspot prediction near the stadium.

Hyperparameters:

- Learning rate $\alpha = 0.1$
- Discount factor $\gamma = 0.95$
- ε-greedy exploration: ε starts at 1.0 and decays to 0.05 over training episodes.
- 3000 training episodes, followed by 300 evaluation episodes (no exploration).

For comparison, the **baseline policy** is:

- If an accident is active, always choose **ImmediateDispatchToNearest**.
- If there is no active accident, do nothing (equivalent to DelayDispatch).

The baseline never pre-positions and never tries to exploit hotspot predictions or hospital load.

Results and Interpretation :

```
################################################
Module 4: RL for Proactive Resource Allocation
################################################

Training Q-learning agent...
Training completed. Last 10 episode rewards: [-15.0, np.float64(6.0), np.float64(1.0),
.float64(-10.600000000000001), np.float64(23.5), np.float64(44.5), np.float64(-14.1000

Evaluating baseline (always dispatch immediately)...
Baseline metrics:
  avg_reward: -3.667
  avg_response_time: 14.598
  served_accidents: 2.863
  critical_within_rate: 0.786

Evaluating Q-learning policy...
Q-learning metrics:
  avg_reward: 24.256
  avg_response_time: 5.160
  served_accidents: 2.577
  critical_within_rate: 0.737

Comparison:
  - avg_reward:     RL vs Baseline = 24.256333333333323 vs -3.6673333333333358
  - avg_resp_time:  RL vs Baseline = 5.15963777490298 vs 14.598137369033735
  - crit_within_rate RL vs Baseline = 0.7368421052631579 vs 0.7857142857142857

Interpretation:
  A better RL policy should achieve higher average reward,
  lower average response time, and a higher fraction of critical
  cases served within the time threshold, compared to the naive
  'dispatch immediately to nearest' baseline.
```

Interpretation of result:

A better RL policy should achieve higher average reward, lower average response time, and a higher fraction of critical cases served within the time threshold, compared to the naive 'dispatch immediately to nearest' baseline.

Module 5 – LLM-Based Real-Time Control Room Summary & Decision Justification

Objective :

The objective of this module is to convert the numeric and symbolic outputs from Modules 1–4 into a human-readable briefing that an emergency control room can actually use. Instead of showing raw probabilities, routes, and Q-values, I use a large language model (LLM) with carefully designed prompts to generate short, professional summaries explaining what is happening, which ambulances have been dispatched, why a particular hospital was chosen, and what the key risks and contingencies are. No fine-tuning is used; I only rely on prompt engineering.

Input Variables / Evidence :

List the fields in DecisionContext:

- Current time, weather, road condition
- List of active incidents:
    - ID, location, short description,
    - severity estimate and confidence,
    - estimated response time
- List of dispatched units:
    - Ambulance ID, incident ID, start location,
    - route summary, target hospital, ETA
- Hospital status:

    - Distance, trauma capability, CT status, load level
- Chosen hospital ID and internal reasoning string
- Planning summary (GraphPlan/POP)
- RL policy comment (why RL suggests this configuration)
- Bayesian risk forecast (future clinical/operational risks)
- Safety notes (disclaimer text)

Prompt Design :

Include your **system prompt** and describe its goals:

- Sets professional, responsible tone
- Limits length (8–14 sentences)
- Forces explanations for “why this ambulance/hospital”
- Insists on uncertainty phrases and not overruling humans

Include your **user prompt template** (summarised) and explain:

- It injects structured context in bullet-like text
- Asks for 3–5 paragraphs with clear sections:
    - Situation
    - Actions taken
    - Rationale
    - Risks and contingencies

Non-trivial Test Case :

Describe the demo case from build_non_trivial_context() (ACC3 critical near airport, H2 CT offline, A1 prepositioned, H1 chosen

despite distance, etc.) and paste the **sample output** (or your actual run result).

How Clarity and Decision-Trust Were Ensured :

Explain:

- The prompt explicitly instructs the LLM to:
  - use professional tone,
  - avoid casual language,
  - highlight uncertainty,
  - not override human authority.
- The context includes explicit **reasons** (e.g. CT offline, trauma capacity, predicted congestion), so the LLM's justification is grounded in actual input.
- The output is short enough to be read over radio or scanned quickly.
- Safety disclaimers remind operators that this is a decision-support tool, not an automatic dispatcher.

Results and Interpretation:

```
##################################################
Module 5: LLM-Based Control Room Summary – Demo Case
##################################################

As of 20:23 hrs, under intense rain with reduced visibility with severely congested around stadium and main corridors
 road conditions, the control room is managing 2 active incident(s). The primary case is SUV rollover with airbags de
ployed at Airport approach road, southbound lane, with a current severity estimate of Critical polytrauma with possib
le head injury (confidence approximately 82%). Ambulance A1 has been dispatched from pre-deployed near Nayapalli chow
k (between H1 and stadium cluster) towards incident ACC3 via Nayapalli → Airport Road via stadium diversion corridor,
 with an expected hospital arrival time of 18.0 minutes. The current destination hospital is H1. Although H2 is geogr
aphically closer to the airport corridor, its CT scanner is currently offline and neurosurgical support is limited. G
iven the high-speed rollover mechanism and strong suspicion of intracranial injury, H1 was chosen as the primary dest
ination despite the additional distance, to avoid secondary transfer delays and ensure direct access to trauma imagin
g and neurosurgical intervention. Planning modules (GraphPlan/POP) recommend this sequence as a safe baseline, while
the reinforcement learning policy supports proactive positioning and prioritisation under congested traffic. The Grap
hPlan sequence recommends early pre-deployment of A1 near the stadium–airport corridor, activation of stadium traffic
 diversions, and direct routing of the critical case to a trauma-capable centre. POP contingencies keep open the opti
on of diverting to H2 only if CT comes online and H1's load reaches unsafe levels. The reinforcement learning agent s
upports pre-positioning A1 near predicted hotspots, prioritising the ACC3 critical case when it occurs, and favouring
 routes that trade a small increase in travel distance for a significant reduction in expected delay due to rain and
match-related congestion. Based on the Bayesian risk forecast, there is a high risk of occult internal bleeding and a
irway compromise in the ACC3 patient over the next 20–30 minutes, with a non-trivial chance of deterioration en route
. Operationally, there is a moderate probability of a second moderate-severity incident around Rupali or the stadium
exit within the next 30 minutes based on historical match-day data. Operators are reminded that these recommendations
 are advisory and final clinical and operational decisions remain with the senior duty officer. This briefing is base
d on current sensor feeds, historical congestion patterns, and predictive models. It does not replace on-scene clinic
al judgement or the authority of the duty medical officer. Any significant deviation in patient condition, road closu
res, or hospital capacity should trigger a rapid re-evaluation of the plan.
```

CONCLUSION:

In this group project, we integrated five complementary AI techniques into a single emergency response framework tailored to a realistic Bhubaneswar match-day scenario. The Bayesian module provides a probabilistic view of current accident severity and future deterioration risk using factors such as impact speed, vehicle type, caller urgency, weather, and traffic delays, helping prevent dangerous cases from being underestimated. The search-based routing module shows how routes that look optimal in the present can become unsafe once rain and stadium diversions are considered, motivating more reliable A*-based paths that anticipate future congestion. Building on this, the planning module converts these insights into clear, high-level action sequences and conditional plans, explicitly modelling when to notify hospitals, pre-deploy ambulances, divert traffic, and reroute mid-journey if CT scanner availability changes. The reinforcement learning component adds a learning layer that improves over a naive “dispatch immediately to nearest” strategy by balancing response time, resource utilisation, and the timely handling of critical cases over many simulated episodes. Finally, the LLM-based summary module turns the system’s internal reasoning into a professional, operator-ready briefing that explains not only what actions are recommended but also why they are justified, in a responsibility-aware tone. Together, these modules illustrate how probabilistic reasoning, search, planning, learning, and natural language generation can be combined to support safer and more informed emergency response decisions under real-world traffic and resource constraints.

Contribution:

GROUP – 13

Charlapalli Uday (22CS01067) – Bayesian Accident Severity & Future Risk Estimation (Module 1)

Guguloth Sivaji (22CS01066) – Predictive Search-Based Ambulance Route Optimization (Module 2)

Ratlavath Nikhilesh (22CS01064) – Adaptive Emergency Response Planning using GraphPlan and POP (Module 3)

Namburu Prem Charan (22CS01062) – Reinforcement Learning for Proactive Resource Allocation (Module 4)

Porika Rahul Naik (22CS01068) – LLM-Based Real-Time Control Room Summary & Decision Justification (Module 5)