ARTIFICIAL INTELLIGENCE PROJECT

AI-Driven Travel Recommendation System Under Real-Time Disruptions

Submitted By: Group-6

- Soham Jain (22CS01007)
- Anshul (22CS01009)
- Harshil Singh (22CS01015)
- Suprit Naik (22CS01018)
- Kumar Utkarsh (22CS01032)
- Adarsh Dhakar (22CS01040)
- Om Prakash Behera (22CS01041)
- Sai Nikhita Palisetty (22CS01050)
- Sayali Khamitkar (22CS01052)

Introduction

Modern travel during monsoon season is highly uncertain due to severe weather alerts, sudden train/flight delays, congestion surges, road blockages, and last-minute cancellations. A traveller moving across multiple cities must constantly adapt to changing conditions, reevaluate alternatives, and choose routes that remain safe, feasible, and cost-efficient in real time.

The goal is to build **AI-Driven Travel Recommendation System** that assists travellers under dynamic disruptions, recommend end-to-end multimodal travel plans and update them as conditions change.

GitHub link: https://github.com/adarshdhakar/AI_Project_Group_6

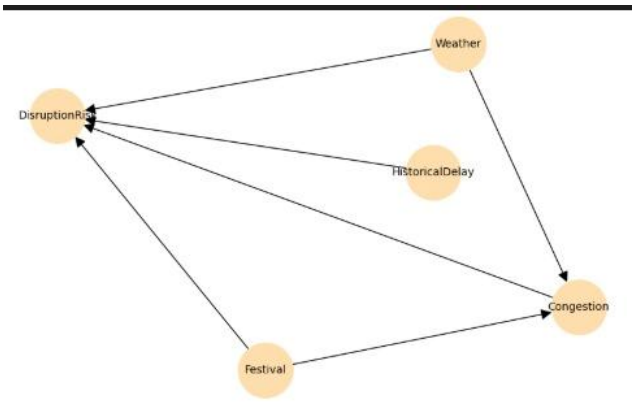
Module 1: Bayesian Disruption Risk Estimation

1.1 Objective

To design a Bayesian Network (BN) that estimates **DisruptionRisk = {LOW, MEDIUM, HIGH}** for a traveler's route during monsoon season, by combining uncertain factors like weather, historical delays, congestion, and festivals in a principled probabilistic way.

1.2 Bayesian Network Structure

This captures how monsoon weather and festivals increase congestion, and how these factors jointly drive the final disruption risk.



1.3 Variables & States

- **Weather:** NONE, MODERATE, SEVERE
- **HistoricalDelay:** LOW, MEDIUM, HIGH
- **Festival:** NO, YES
- **Congestion:** LOW, MEDIUM, HIGH
- **DisruptionRisk:** LOW, MEDIUM, HIGH

1.4 CPT Summary

All CPTs are exported to **bn_cpts.json**.

- **Priors**

Variable	States	Probabilities
Weather	NONE / MODERATE / SEVERE	0.60 / 0.30 / 0.10
HistoricalDelay	LOW / MEDIUM / HIGH	0.50 / 0.30 / 0.20
Festival	NO / YES	0.85 / 0.15

- **Congestion CPT (Weather, Festival → Congestion)**

Weather	Festival	P(LOW)	P(MED)	P(HIGH)
NONE	NO	0.53	0.30	0.17
NONE	YES	0.44	0.30	0.26
MODERATE	NO	0.35	0.30	0.35
MODERATE	YES	0.26	0.30	0.44
SEVERE	NO	0.17	0.30	0.53
SEVERE	YES	0.08	0.30	0.62

- **DisruptionRisk CPT (Excerpt)**

Example entries (full table in JSON):

Weather	HistDelay	Congestion	Festival	P(LOW)	P(MED)	P(HIGH)
NONE	LOW	LOW	NO	0.891	0.082	0.027
NONE	LOW	HIGH	YES	0.500	0.318	0.182

1.5 Test Case & Output (Example A – Non-trivial Scenario)

Scenario: *Severe monsoon, historically delay-prone route, festival day; congestion not directly observed.*

Evidence:

Variable	Value
Weather	SEVERE
HistoricalDelay	HIGH
Festival	YES

Posterior DisruptionRisk:

State	Probability
LOW	0.0313
MEDIUM	0.0917
HIGH	0.8770

Posterior Congestion

State	Probability
LOW	0.0800
MEDIUM	0.3000
HIGH	0.6200

1.6 Interpretation:

Under severe weather, historically high delays, and a festival day, the model estimates **~88% probability of HIGH disruption risk** and a **62% chance of high congestion**. This means the route is very likely to be disrupted and should be avoided or backed with strong alternative plans in downstream modules.

Module 2: Search-Based Multi-Modal Route Exploration

2.1 Objective

To model a realistic **multi-modal travel network** (flight, train, cab, metro) as a weighted graph and use:

- **Uniform-Cost Search (UCS)** — uninformed optimal search
- **A*** — informed search with an admissible time-based heuristic

to find the best route from **A → E** before and after disruptions.

2.2 Graph & Edge Model

The **MultiModalGraph** stores edges of type **Edge(src, dst, mode, base_time, base_cost, delay_prob, delay_extra, comfort, safety)**.

Each edge encodes:

- **Mode:** flight / train / cab / metro
- **Base travel time and fare**
- **Delay probability & expected extra delay**
- **Comfort, safety scores ∈ [0, 1]**

The **expected travel time** is:

$$t_{\text{exp}} = t_{\text{base}} + p_{\text{delay}} \times \text{extra delay}$$

2.3 Cost Function

A single scalar **multi-criteria cost** is used in both UCS and A*:

$$\text{Cost}(e) = w_t t_{\text{exp}} + w_m \cdot \text{fare} + w_d(p_{\text{delay}} \cdot \text{extra delay}) + w_c(1 - \text{comfort}) + w_s(1 - \text{safety})$$

Weights used in the demo:

- time = 1.0, money = 0.5, delay = 1.0
- comfort = 40, safety = 60

This heavily penalizes **uncomfortable** and **unsafe** legs even if they are cheap or fast.

2.4 Search Algorithms

Uniform-Cost Search (UCS)

- Dijkstra-style expansion by lowest cumulative cost g(n).
- Uninformed (no heuristic).
- Always finds the optimal path but may explore more states.

A* Search

- Uses f(n) = g(n) + h(n), where, g(n) = cost so far, h(n) = optimistic lower bound on remaining time

Heuristic:

- Pre-computed minimum base travel time from each node to goal using reverse Dijkstra, then: $h(n) = w_{\text{time}} \cdot \text{minBaseTime}(n \rightarrow \text{goal})$
- This is **admissible** (never overestimates), so A* still finds the **same optimal path** as UCS but more efficiently.

2.5 Dynamic Disruptions

After computing initial routes, the graph is modified to simulate real-time disruptions:

- **Flight B→E (flight)**: delay probability ↑ and extra delay ↑
- **Train C→E (train)**: delay probability ↑ and extra delay ↑
- **Metro A→D (metro)**: base_time scaled by 1.7 (severe slowdown)

These updates change edge costs, forcing the planner to re-evaluate the best route under new conditions.

2.6 Example: Routes Before vs After Disruption

Start: A **Goal:** E
Weights: {time=1.0, money=0.5, delay=1.0, comfort=40, safety=60}

- **Before Disruption**

Both UCS and A* pick the same route and cost:

- **UCS (cost = 202.800)**
- **A* (cost = 202.800)**

Route:

- A --[metro, 90.0 min, ₹15] --> D
- D --[cab, 40.0 min, ₹30] --> E

Expected time: 130.7 min
Total cost: ₹45.0

- **After Disruption**

After increasing delays and slowing metro A→D:

- **UCS (cost = 265.800)**
- **A* (cost = 265.800)**

Route (still optimal under new conditions):

- A --[metro, 153.0 min, ₹15] --> D
- D --[cab, 40.0 min, ₹30] --> E

Expected time: 193.6 min
Total cost: ₹45.0

2.7 Interpretation

- Both UCS and A* always agree on the **optimal path** (same cost) hence heuristic is **admissible**
- After disruptions, the **cost** and **expected time** rise significantly (from ~130.7 to ~193.6 minutes), reflecting worsened conditions.
- Even with disruptions on metro and competing edges, A→D→E via **metro + cab** remains the **safest/most comfortable multi-modal route** according to the weighted cost model.

Module 3: Adaptive Travel Plan Generation (GraphPlan + POP)

3.1 Objective

To model travel as a **planning problem** and automatically generate feasible itineraries using **GraphPlan** with a **POP-style extractor**. The planner should build a planning graph, select non-mutex actions, and replan when some actions fail.

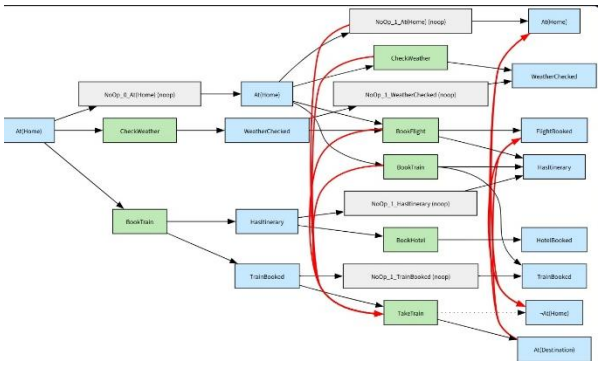
3.2 Domain Overview

Initial state: At(Home); **Goals:** At(Destination), HotelBooked

Actions (Core Set) (NoOp actions preserve literals across levels.)

Action	Preconditions	Add effects	Delete	Cost	Time	Risk
CheckWeather	At(Home)	WeatherChecked	—	1	5	0.01
BookTrain	At(Home)	TrainBooked, HasItinerary	—	10	10	0.02
BookFlight	At(Home), WeatherChecked	FlightBooked, HasItinerary	—	80	15	0.08
BookHotel	HasItinerary	HotelBooked	—	30	5	0.01
Fly	FlightBooked	At(Destination)	At(Home)	200	120	0.12
TakeTrain	TrainBooked	At(Destination)	At(Home)	40	360	0.05

3.3 GraphPlan Diagram:



3.4 Replanning Test (Failure Handling)

Test: test_replanning_avoids_failed_action

"CheckWeather" is marked **failed**. Planner expands 4 levels and extracts a plan.

Result:

- A valid plan is returned.
- The failed action **does not appear** in any layer.
- Extracted plan switches to the **train itinerary**:
    - BookTrain → BookHotel → TakeTrain → achieve goals.

3.5 Interpretation:

The planner successfully reroutes around failed actions and still reaches the goals, showing its ability to **adapt travel plans under disruptions**.

Module 4: Reinforcement Learning for Robust Travel Adaptation

4.1 Objective

To model the travel process as a **Markov Decision Process (MDP)** and train a **Q-Learning agent** that adapts to disruptions in real time; by choosing actions such as confirming, switching, waiting, or rebooking to safely complete the journey under uncertainty.

4.2 MDP Formulation

State representation: (Progress, Disruption, Urgency, CostLeft)

Progress ∈ {NOT_STARTED, EN_ROUTE, ARRIVED, CANCELLED}
Disruption ∈ {NONE, DELAY, CANCEL}
Urgency ∈ {LOW, MEDIUM, HIGH}
CostLeft ∈ {0, 20, 40, 60, 80, 100}

Actions:
ConfirmRoute, SwitchRoute, Wait, CancelAndRebook

Transition model & rewards are implemented in transitions() and encode realistic travel effects:

- **ConfirmRoute:** progress → EN_ROUTE, with chances of NONE / DELAY / CANCEL
- **Wait:** urgency increases; delays or cancellations may occur
- **SwitchRoute:** reduces cost but has cancellation risk
- **CancelAndRebook:** resets trip with penalty

Rewards encourage finishing the trip and penalize risky or costly decisions:

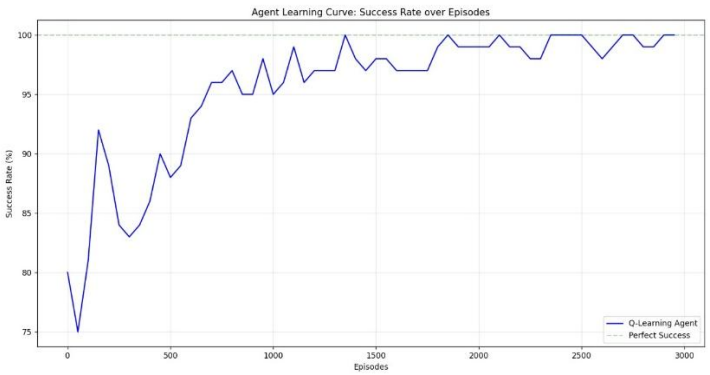
- +30 on safe arrival
- Moderate bonuses for good confirmations (+5…20)
- Penalties for delays (−5…−10), cancellations (−20…−30), and unnecessary waiting

4.3 Q-Learning Setup

- Episodes: 3000
- Exploration decay: ε → 0.05
- Learning rate α = 0.1
- Discount γ = 0.95

The agent learns an optimal policy mapping states to best travel actions despite uncertainty.

Learning Curve (Agent Performance)



The success rate starts around **75–90%**, then climbs steadily. After ~1500 episodes, the agent consistently achieves **97–100% success**, indicating strong convergence and stable behaviour under disruption dynamics.

4.4 Detailed Example Run (Using Learned Q-Table)

From our simulation output:

Step	State (Progress, Disruption, Urgency, Budget)	Action	Reward
1	('NOT_STARTED','NONE','LOW',100)	Wait	−1
2	('NOT_STARTED','NONE','MEDIUM',100)	ConfirmRoute	+5
3	('EN_ROUTE','NONE','MEDIUM',80)	ConfirmRoute	−5
4	('EN_ROUTE','DELAY','MEDIUM',80)	ConfirmRoute	+20

Result: SUCCESS — ARRIVED SAFELY
Final State: ('ARRIVED','NONE','MEDIUM',80)
Total Reward: 19

4.5 Interpretation

- The agent first **waits** to reduce risk, then confirms at a safer moment.
- Even after encountering a **delay**, its learned strategy still pushes it toward arrival.
- The policy balances urgency, cost, and disruption risk effectively.

Module 5: LLM-Based Traveler Advisory & Explanation Generation

5.1 Objective

To generate a **clear, trustworthy traveller advisory** using an LLM (Gemini). The model explains the final itinerary, why it is safe/feasible, what risks were avoided, and what fallback options exist—without exposing chain-of-thought.

5.2 Inputs to the LLM

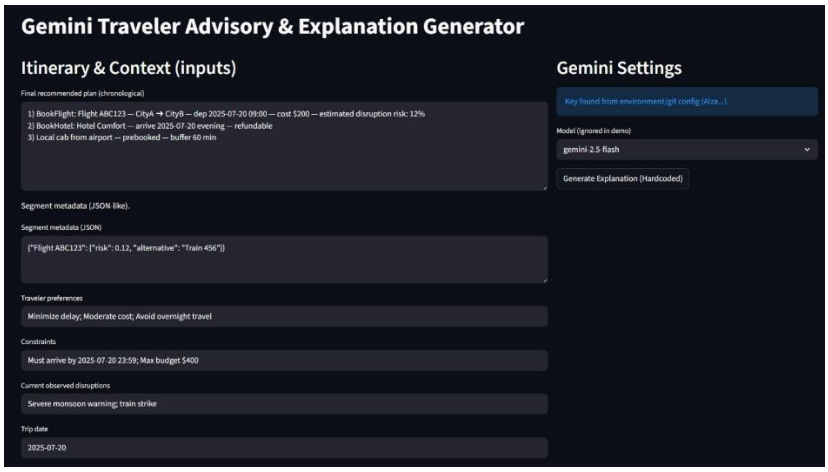
- **Final itinerary:** Flight ABC123 (risk 12%), hotel (refundable), pre-booked cab (60-min buffer).
- **Traveler preferences:** minimize delay, moderate cost, avoid overnight travel.
- **Constraints:** arrive by 2025-07-20 23:59, budget ≤ $400.
- **Disruptions:** monsoon warning, train strike.
- **Segment metadata:** e.g., alternative train option “Train 456”.

5.3 Prompt Structure

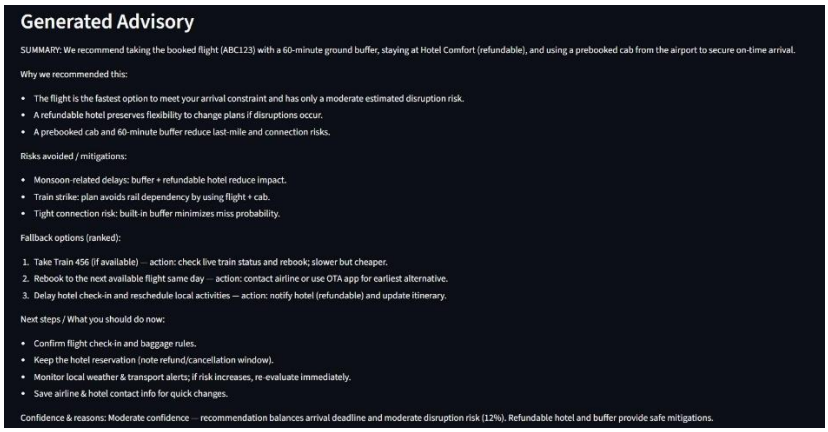
Two-part prompt:

1. **System prompt**
    - Defines style and format.
    - Requires headings:
      SUMMARY, *Why we recommended this*, *Risks avoided*, *Fallback options*, *Next steps*, *Confidence & reasons*.
    - Forbids chain-of-thought.
2. **User prompt**
    - Provides constraints, preferences, disruptions, itinerary, and metadata.
    - Requests a traveller-facing advisory following the fixed template.

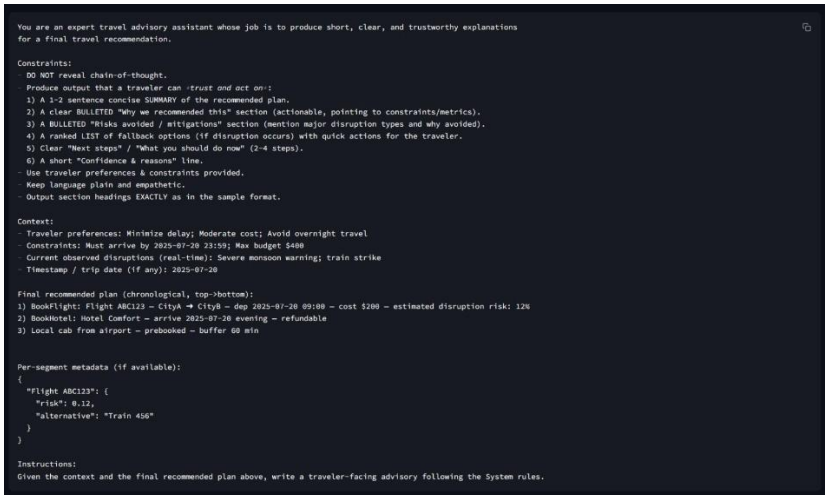
Input:



The Generated Advisory as Output:



The Respective Prompt is given here.



```
You are an expert travel advisory assistant whose job is to produce short, clear, and trustworthy explanations
for a final travel recommendation.

Constraints:
- DO NOT reveal chain-of-thought.
- Produce output that a traveler can *trust and act on*:
  1) A 1-2 sentence concise SUMMARY of the recommended plan.
  2) A clear BULLETED "Why we recommended this" section (actionable, pointing to constraints/metrics).
  3) A BULLETED "Risks avoided / mitigations" section (mention major disruption types and why avoided).
  4) A ranked LIST of fallback options (if disruption occurs) with quick actions for the traveler.
  5) Clear "Next steps" / "What you should do now" (2-4 steps).
  6) A short "Confidence & reasons" line.
- Use traveler preferences & constraints provided.
- Keep language plain and empathetic.
- Output section headings EXACTLY as in the sample format.

Context:
- Traveler preferences: Minimize delay; Moderate cost; Avoid overnight travel
- Constraints: Must arrive by 2025-07-20 23:59; Max budget $400
- Current observed disruptions (real-time): Severe monsoon warning; train strike
- Timestamp / trip date (if any): 2025-07-20

Final recommended plan (chronological, top->bottom):
1) BookFlight: Flight ABC123 — CityA → CityB — dep 2025-07-20 09:00 — cost $200 — estimated disruption risk: 12%
2) BookHotel: Hotel Comfort — arrive 2025-07-20 evening — refundable
3) Local cab from airport — prebooked — buffer 60 min


Per-segment metadata (if available):
{
  "Flight ABC123": {
    "risk": 0.12,
    "alternative": "Train 456"
  }
}

Instructions:
Given the context and the final recommended plan above, write a traveler-facing advisory following the System rules.
```

5.4 Purpose

Module 5 converts the system's internal reasoning into a **clean, structured explanation** a traveller can trust, completing the pipeline from planning → risk estimation → final human-readable advisory.