AI Empower Hackathon 2025: Enhancing Egyptian Tourism

Problem Definition

The Egyptian tourism sector, a cornerstone of the national economy and a vital source of foreign currency, is ripe for digital innovation. Our solution is designed to address critical inefficiencies and challenges that hinder a truly world-class tourist experience. The problems we are tackling are not just inconveniences for travelers; they represent significant barriers to sustainable growth for the industry as a whole.

Our project aims to solve the following core issues:

- **Overcrowding & Congestion:** Tourist hotspots in cities like Cairo and Luxor frequently suffer from high foot traffic during peak hours, leading to long queue times, a degraded visitor experience, and potential stress on historical sites. Our solution optimizes hoping for **reduced queue time** and **a more even distribution of visitors throughout the day**, ensuring a smoother flow and preserving the integrity of archaeological treasures.
- **Lack of Personalization:** The market is saturated with generic, one-size-fits-all itineraries that fail to capture the unique interests of modern travelers. This leads to tourist dissatisfaction and a missed opportunity to create memorable experiences. We are optimizing for **enhanced tourist experience**, as personalized plans are more likely to be followed and enjoyed.
- **Underused Destinations ("Hidden Gems"):** Many culturally rich and historically significant sites are overshadowed by major attractions, leading to an unbalanced tourism footprint. By strategically promoting these "hidden gems," we can distribute economic benefits more equitably across communities and reduce the strain on popular sites.
- **Poor On-site Experience:** Tourists often face practical hurdles, from navigating complex urban transport to dealing with unfair pricing practices for local goods. These issues can be frustrating and tarnish the overall perception of the destination. We are optimizing for **reduced instances of overpaying for goods** and a **seamless, stress-free travel experience**, which directly correlates with improved satisfaction and positive word-of-mouth recommendations.

Motivation: The Importance for Egypt's Tourism

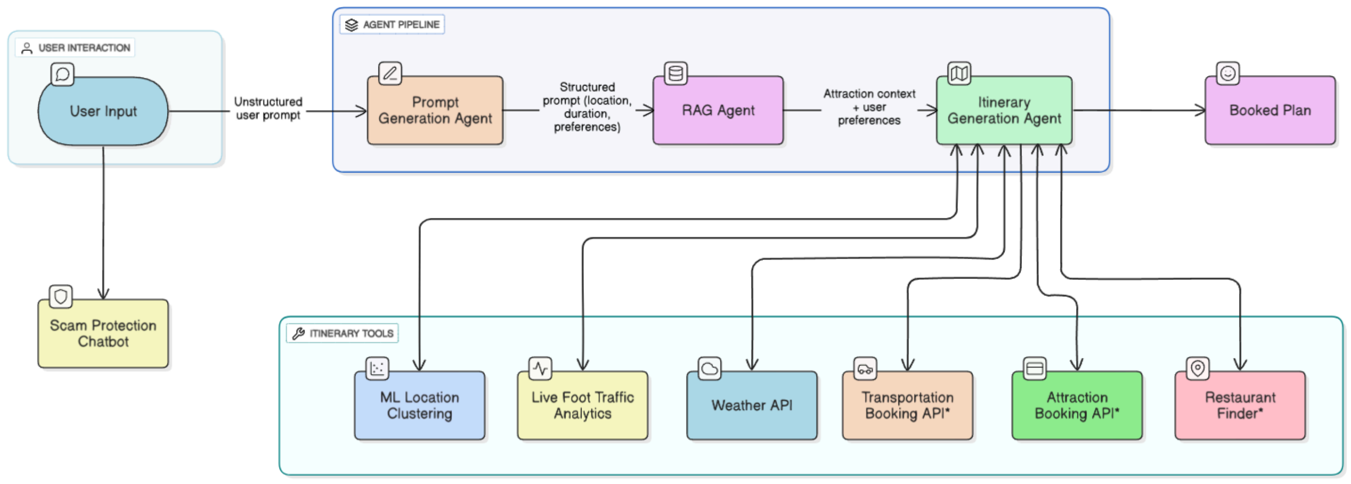
This solution is particularly vital for Egypt at a time when the government is actively pursuing digital transformation and sustainable development. Our project directly aligns with the strategic objectives outlined in **Egypt's National AI Strategy 2025-2030** and **Egypt’s Vision 2030** for sustainable tourism.

- **Economic Growth and Diversification:** By enhancing the tourist experience and promoting a wider range of sites, our solution can help increase the length of stay and tourist spending. This drives economic growth not just in the major hubs but also in local communities surrounding "hidden gems," fostering a more diversified and resilient tourism economy.
- **Global Competitiveness:** In an increasingly competitive global travel market, providing a highly personalized, intelligent, and seamless experience is a key differentiator. Our solution leverages AI to elevate Egypt’s reputation as a technologically advanced and visitor-centric destination, helping it attract new demographics and maintain its leadership in the Middle East and Africa.
- **Sustainability and Preservation:** Overcrowding is not just an inconvenience; it can pose a serious threat to fragile historical sites. By using AI to intelligently manage visitor flow and mitigate congestion, we are directly contributing to the long-term preservation of Egypt's invaluable cultural heritage for future generations.

Our Solution: A Multi-Agent Architecture

Our solution is a multi-agent system designed to create a dynamic, personalized, and efficient tourism experience. It is composed of three interconnected modules, each powered by specialized AI agents working collaboratively to solve a specific problem.

The agents are designed to communicate and share data, creating a holistic system that can adapt to real-time changes and user preferences.



Module 1: The Intelligent Itinerary Planner

This module is the core of the system, responsible for generating a highly personalized and optimized travel plan.

Workflow:

1. **User Query:** The process begins when a user provides a natural language query, for example: "Schedule an itinerary in Cairo from 14/8/2025 to 16/8/2025, I am mostly interested in archaeological sites."
2. **Query Refinement:** This step processes and cleans the input query to extract only the relevant entities required for next stages.
3. **Information Agent (RAG):**
   - This is a Retrieval-Augmented Generation (RAG) system which processes the refined query.
   - It is fine-tuned on a comprehensive dataset from the website of the Egyptian Ministry of Tourism and Antiquities.
   - It retrieves a list of tourist sites in the specified location (Cairo) that match the user's preferences (archaeological sites).
   - **Bias for "Hidden Gems":** The agent is programmed with a strategic bias to include and promote sites with a low count of visitor reviews, helping to distribute tourist traffic and introduce visitors to lesser-known attractions.
4. **Data Fetching & Clustering:**
   - The generated list of sites is sent to a dedicated tool that fetches real-time data.
   - **Tooling:** This tool integrates with APIs (such as those for weather forecasts

    and location-based services , namely BestTime.app) to gather weather and crowdedness data for each site over the entire duration of the trip, forming a comprehensive dataset having all needed fields for subsequent steps.
    - **Clustering:** The sites are intelligently clustered into daily groups. The clustering algorithm ensures that no day is overcrowded (exceeding a max_hours_per_day threshold) or overly not filled, maintaining a balanced and enjoyable pace for the traveler.
5. **Planning Agent:** This agent orchestrates the final schedule and logistics, with two sub-agents handling specific tasks.
    - **Scheduling Sub-Agent:**
        - It assigns each daily cluster of sites to a specific date. The assignment is optimized to minimize the traveler's exposure to extreme high temperatures, particularly for outdoor sites.
        - Within each day, it schedules the visits. The scheduling logic avoids popular peak hours for each site, further mitigating overcrowding and enhancing user's experience.
    - **Booking Sub-Agent:**
        - Based on the final schedule, this agent books and pre-plans Uber rides to ensure seamless transportation between sites.
        - It also suggests nearby restaurant recommendations for meal breaks, intelligently filling gaps in the schedule with dining options close to the last site visited.

Module 2: Itinerary Maintenance Module

This module ensures the itinerary remains optimal and responsive to real-time conditions.

Workflow:

1. **Scheduled Monitoring:** A scheduled cron job periodically checks the crowdedness of the next site on the itinerary.
2. **Live Foot Traffic Tool:** It utilizes a "Live Foot Traffic" tool (integrating with services , namely BestTime.app) to compare the current crowdedness with historical data.
3. **Dynamic Rescheduling:** If the current foot traffic is detected to be higher than usual, the system can automatically trigger a re-planning of the rest of the day including cancelling Uber rides and booking the rides needed for the new schedule. This dynamic rescheduling accommodates for the unexpected crowdedness, while still optimizing for the original parameters of avoiding extreme heat and popular hours.

Module 3: Reasonable Pricing Module

This mini-module acts as a valuable on-site tool for the traveler, protecting them from overpricing.

Workflow:

1. **User Input:** The user provides the name of an item they are considering purchasing.
2. **AI Agent & LangChain:** An AI agent uses a framework like LangChain to perform a live web search.
3. **Price Fetching:** It fetches the top 5 search results related to the item's price.
4. **Price Range Generation:** The agent analyzes the search results to determine two distinct price ranges:
   - **Reasonable Price Range:** A fair price for the item in a standard market.
   - **Touristic Price Range:** A higher but still reasonable price range to expect in a tourist area, providing the user with a realistic negotiation baseline.

Novelty Highlights:

- **Holistic Optimization:** Unlike single-purpose apps, our solution doesn't just offer an itinerary; it optimizes the entire trip based on multiple, often conflicting, real-world constraints simultaneously. The system intelligently balances the need to avoid heat and crowds with the logical flow of a daily schedule.
- **Adaptive Maintenance Loop:** The Itinerary Maintenance Module is a key differentiator. The use of a scheduled cron job and a "Live Foot Traffic" tool to dynamically re-plan an itinerary in real-time is a novel feature in tourism planning. It anticipates problems before they happen and provides a responsive, stress-free experience for the traveler, especially in unpredictable environments.
- **"Hidden Gems" Promotion:** The strategic bias built into the RAG-based Information Agent actively addresses the problem of underutilized sites. By recommending lesser-known, low-review sites alongside popular ones, our solution promotes a more sustainable and equitable tourism ecosystem.
- **On-site Financial Protection:** The "Reasonable Pricing Module" is a useful, real-world utility that provides a tangible benefit to the traveler. It directly addresses a common pain point of being overcharged, building trust and enhancing the overall on-site experience.

Technical Overview

The Dataset

Using web scrapping, we collected a dataset from the website of the Egyptian Ministry of Tourism and Antiquities. A single row in this dataset represents a touristic site and includes the following fields:

- `url`: A string containing the URL to the site's official page.
- `name`: A string with the name of the site.
- `location`: A string indicating the city or region.
- `Outdoors`: A boolean value (`"True"` or `"False"`) to specify if the site is an outdoor venue.
- `Latitude` and `Longitude`: Numerical values for the geographical coordinates.
- `hours_needed`: Number of hours needed to fully explore the touristic site.
- `rating` and `reviews`: Numerical values for visitor ratings and the total count of reviews.
- `description`: A long string with a detailed description of the site's history and features.
- `opening_hours`: A nested object containing the operating hours for each day of the week.
- `prices`: A string detailing the ticket prices for different groups (e.g., foreigners, Egyptians/Arabs).
- `services`: A list of strings for any additional services offered.

Notebook: `RAG/RAG.ipynb`

This notebook provides a technical implementation of the Information Agent from the "Intelligent Itinerary Planner" module. It demonstrates a Retrieval-Augmented Generation (RAG) workflow to generate a list of tourist sites based on a user's query and preferences.

The key components of the notebook are:

- **RAG Setup:** The code initializes a vector store (`Chroma`) with document embeddings (`HuggingFaceEmbeddings`). These documents are loaded from JSON files (e.g., `museums.json`, `monuments.json`) representing a knowledge base of Egyptian tourist sites which were scrapped as described earlier.

- **Intent Refinement:** A prompt is used with a large language model (`ChatDeepSeek`) to extract and structure the user's intent from a natural language query. It identifies the user's location, preferences, and desired duration of the trip.
- **Document Retrieval and Filtering:** The refined user intent is used to query the vector store. A custom function (`filter_with_threshold_limit`) is then applied to the retrieved documents. This function is crucial for implementing the "hidden gems" strategy, as it prioritizes a limited number of sites with fewer reviews while ensuring a sufficient number of popular sites are also included in the final list.
- **Output:** The final output is a list of curated tourist sites, ready to be passed to the Planning Agent for itinerary scheduling.

Notebook: `agents/WeatherCrowdednessAggregator.ipynb`

This notebook is a data aggregation tool within the Itinerary Planner module.

- **Input:** A list of touristic sites, each with a name, location (latitude/longitude), and a specified date range for the trip.
- **Output:** A structured output for each site, augmented with essential data for the planning process. This includes hourly foot traffic forecasts, popular hours, and hourly temperature data for every day in the requested date range.

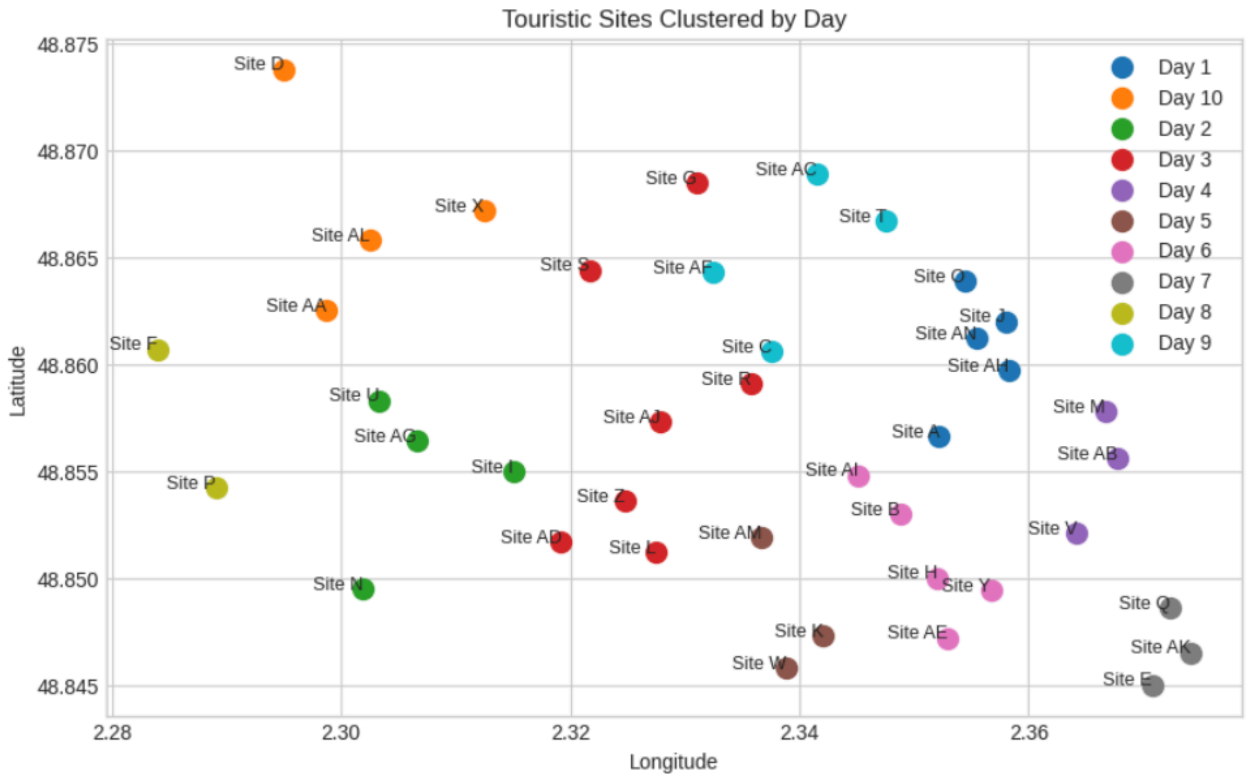
Notebook: `clustering/ClusteringLocations.ipynb`

This notebook demonstrates the intelligent clustering and day-planning functionality of the Itinerary Planner module. It takes a list of touristic sites and groups them into logical clusters, with each cluster representing a single day of the trip.

The main functions and libraries used are:

- **K-Means Clustering:** The core of the notebook's logic, used to group tourist sites based on their geographic location.
- **Optuna:** This library is used for hyperparameter tuning. It optimizes the clustering by finding the best weight to apply to the number of hours needed per site, ensuring a balance between geographical proximity and the total time spent per day.
- **Haversine:** The Haversine formula is used to calculate the great-circle distance between two points on a sphere, which is essential for accurate distance-based clustering.

- **Output:** The notebook outputs a final, balanced itinerary, with a list of sites and their attributes for each day, ready for the next stage of the planning process.



Notebook: `agent/DayDateAssigningAgent.ipynb`

This notebook implements the "Scheduling Sub-Agent" of the Itinerary Planner module. Its primary responsibility is to assign clustered groups of touristic sites (each representing a day) to specific dates within the trip.

The key components are:

- **Optimization Model:** The core logic uses a scoring function that penalizes assignments based on two main criteria:
    1. **Heat Exposure:** Outdoor sites are penalized more heavily if assigned to dates with high average temperatures.
    2. **Crowd Density:** Sites are penalized for being scheduled on days with high average crowd levels.
- **Brute-Force Optimization:** The notebook generates all possible permutations of assigning the clustered days to the available dates and calculates a total optimization score for each permutation. The assignment with the lowest score is selected as the most optimal schedule.
- **LangChain Agent:** The logic is wrapped in a LangChain agent. This allows the

system to not only find the optimal schedule but also to reason about *why* a particular schedule was chosen, providing a clear and transparent explanation of the trade-offs between crowd and temperature penalties.

Notebook: `agents/IntraDayPlanningAgent.ipynb`

This notebook implements the Intra-Day Planning Agent, a sub-agent responsible for creating an optimal hourly schedule for a pre-clustered list of sites for a single day.

The key components are:

- **Scoring Function:** A comprehensive scoring function evaluates all possible time slots for each site based on multiple criteria:
    - **Crowd Levels:** Time slots with lower crowd levels receive a higher score.
    - **Temperature:** For outdoor sites, time slots with temperatures in an optimal range (e.g., 20°C–28°C) receive a bonus, while extreme temperatures incur a heavy penalty.
    - **Time of Day:** The model incorporates bonuses for visiting sites during less crowded periods (according to aggregated data).
- **Greedy Scheduling Algorithm:** The notebook uses a greedy algorithm to assign the highest-scoring time slots to monuments first. This prioritizes the most optimal visits while leaving space for other sites.
- **LangChain Agent Integration:** The scheduling logic is embedded within a LangChain agent. This allows the system to not only output the final schedule but also to provide rich, human-readable insights and reasoning for its choices.

Notebook: `agents/ReschedulingAgent.ipynb`

This notebook implements the core logic of the Itinerary Maintenance Module, specifically the dynamic rescheduling of an itinerary in response to real-time events. It acts as a more advanced version of the intra-day planner, with the added capability to handle unforeseen changes like unexpected crowdedness.

The key components are:

- **Dynamic Rescheduling Logic:** The notebook's main function `find_optimal_schedule` takes an existing itinerary and a "current time" as input. It discards any activities that have already passed and then re-optimizes the remaining schedule from the current time forward.
- **Penalty-Based Optimization:** The scheduling decision-making is driven by a penalty-based scoring function. It calculates a penalty for each potential time

slot, heavily weighting factors such as extreme temperatures for outdoor sites and high crowd levels. The goal is to find the schedule with the minimum total penalty.

- **LangChain Agent:** The entire process is orchestrated by a LangChain agent. This enables the agent to not only perform the reschedule but also to provide a detailed, human-readable summary of the changes, including the reasoning behind the new schedule and a comparison of the original vs. new plan.

Notebook: `agents/ReasonablePrice.ipynb`

This notebook implements the "Reasonable Pricing Module" using a LangChain agent to provide a price range for a given item. The agent leverages web search to gather pricing information, helping tourists avoid overpaying.

The key components are:

- **LangChain Agent:** The notebook sets up a LangChain agent using a large language model (`ChatDeepSeek`) and a web search tool (`TavilySearch`). The agent is given instructions to find and aggregate pricing information for a specified item.
- **Web Search:** The `TavilySearch` tool is configured to perform a web search and return a limited number of results (e.g., the top 5). This focuses the search on the most relevant information.
- **Price Range Aggregation:** The agent processes the search results to identify a price range. The output is a clear, human-readable summary that presents a reasonable price range for the item based on the aggregated data.

`scripts/estaurants.py`

- **Input:** Latitude, longitude, and a search radius in meters.
- **Output:** A list of nearby restaurants with their names and coordinates.

`scripts/rental.py`

- **Input:** Pickup and drop-off coordinates, dates, and times.
- **Output:** Details of the most optimal rental car option, including the car's name, price, supplier, rating, and pick-up/drop-off information.