Travel Genie: A Multi-Agent Collaborative Architecture for Autonomous Itinerary Synthesis and Real-Time Disruption Recovery

Sakthivell Kannan[1], Siti Azirah Asmai[1]*, Suhail S.M. Alqrinawi[1], Nur Diana Izzani Masdzarif[1], Muhammad Hafidz Fazli Md Fauadi[1]

[1]Fakulti Kecerdasan Buatan dan Keselamatan Siber, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia

DOI: https://doi.org/10.47772/IJRISS.2026.100900705

Received: 26 September 2026; Accepted: 01 October 2026; Published: 21 October 2026

ABSTRACT

Conventional trip-planning tools and pre-packaged online travel agency (OTA) bundles are ill-suited to a travel environment that changes in real time: they lock in a fixed budget, a fixed route, and a fixed set of assumptions about weather and transport, so a single delayed flight or a rained-out afternoon can invalidate an entire schedule. Feeding the same problem to a general-purpose large language model does not solve it either, since a single monolithic prompt tends to invent attractions that do not exist, misjudge travel times between stops, and ignore hard budget ceilings. This paper introduces TravelGenie, a production-oriented mobile platform that instead splits the planning task across seven cooperating, purpose-built AI agents responsible for destination research, itinerary drafting, logistics and lodging, budget enforcement, time-block scheduling, fatigue management, and live disruption recovery. The system is delivered through a three-tier stack (a React Native / Expo SDK 54 mobile client, a FastAPI backend, and an SQLite store acting as the single source of truth), with the agent pipeline built on the Agno orchestration framework and served by Groq-hosted LLM inference, supplemented by SerpAPI flight data and OpenWeatherMap forecasts. Itineraries can be exported as RFC 5545-compliant .ics files for direct import into Apple Calendar, Google Calendar, or Outlook. The evaluation combined REST API endpoint testing, three scenario-based tests of the agent pipeline, and user acceptance testing (UAT) with 15 participants on physical Android and iOS devices. All six API endpoints passed, and full seven-agent itinerary generation took 6.4 s on average. In the scenario tests, the system replaced rain-affected outdoor activities with indoor alternatives and reduced an over-budget Tokyo plan from RM 2,850 to RM 1,980, within its RM 2,000 limit. UAT participants rated overall satisfaction at 4.63 out of 5, with weather and disruption adaptation rated highest at 4.80.

Keywords— multi-agent systems; travel itinerary planning; large language models; dynamic rescheduling; calendar export; mobile computing; AI in tourism.

INTRODUCTION

Building a workable travel itinerary is a constraint-satisfaction problem in disguise: preferences, a budget ceiling, geography, opening hours, transport windows, and weather all have to be reconciled at once (Buhalis & Law, 2008). Most travelers still work through this manually, stitching together results from separate booking sites, blog posts, and OTA packages (Gretzel et al., 2015). That approach bakes in an assumption of stability that flights will land on time, prices will hold, and the weather will cooperate which rarely survives contact with an actual trip.

The rise of large language models has produced a wave of AI trip-planning tools (Xie et al., 2024), but asking a single prompt to do everything exposes four recurring failure modes. First, unconstrained generation invents landmarks that do not exist or produces schedules with physically impossible transit overlaps (Chang et al., 2024). Second, cost control is weak: a model asked to "stay under budget" will still recommend premium hotels or paid attractions that blow past the stated limit, because nothing in the prompt structurally enforces the inequality. Third, a model with no live data access cannot know about an active

storm warning, a fare spike, or a venue that just closed (Wang et al., 2024). Fourth, and most damaging in practice, a plan generated this way is static: a four-hour flight delay or an unexpected downpour leaves the traveler with no automated way to recover, since the system that produced the plan has no mechanism for revising it (Wooldridge, 2009).

A more promising direction, echoed across recent agentic-AI research, is to decompose the planning task across several narrower, cooperating agents rather than one generalist model (Wang et al., 2024; Wu et al., 2024; Hong et al., 2024). Dedicated sub-agents for budgeting, scheduling, and contingency handling can each enforce their own constraints strictly, and the resulting pipeline tends to be more resilient than a single prompt asked to juggle everything simultaneously.

This paper presents TravelGenie, an enterprise-grade mobile platform built around that idea. Given a traveler's departure point, destination, travel dates, style preferences, budget tier, hotel-star preference, and interests, the system runs a seven-agent pipeline that pulls live flight prices from SerpAPI (Google Flights) and live weather from OpenWeatherMap, resolves geographic coordinates for in-app navigation, and uniquely includes a built-in stress-testing suite that simulates rainstorms, flight delays, and attraction closures to verify that the itinerary can adapt automatically. The main contributions of this work are:

- A three-tier mobile architecture (React Native/Expo SDK 54, FastAPI, SQLite) with sub-second data operations and persistent server-side trip storage;
- A seven-agent collaborative pipeline built on the Agno framework and Groq inference;
- A budget-enforcement engine that enforces a hard spending ceiling whenever a feasible configuration exists, by auto-downgrading accommodation tiers and swapping paid attractions for free equivalents, and that explicitly flags the plan when no feasible configuration exists;
- A rule-based disruption-recovery engine that performs time shifts, indoor/outdoor activity substitutions, and route re-indexing in response to live weather, delay, or closure events; and
- A standards-compliant iCalendar (.ics, RFC 5545) export module for direct import into mainstream calendar applications.

LITERATURE REVIEW

Research on computer-assisted travel planning has moved through three broad phases. Early e-tourism systems focused on digitizing search and booking without addressing the underlying optimization problem (Buhalis & Law, 2008); the smart-tourism literature that followed emphasized personalization and context-awareness but still relied on largely static recommendation logic (Gretzel et al., 2015). The more recent wave of LLM-based planners inherits the flexibility of natural-language generation but also its weaknesses: hallucinated content, weak numeric constraint handling, and no access to live conditions unless explicitly tooled (Xie et al., 2024; Chang et al., 2024). In parallel, the autonomous-agents literature argues that decomposing a task across specialized, tool-using agents produces more reliable and auditable behavior than a single end-to-end prompt (Wang et al., 2024), which is consistent with classical multi-agent systems theory (Wooldridge, 2009).

Two strands of recent work are especially close to this study. The first treats travel planning as a test of real-world planning by language agents. Xie et al. (2024) introduced TravelPlanner, a sandbox benchmark with roughly four million data records and 1,225 curated planning queries that must satisfy both commonsense and hard user constraints. Even GPT-4-based agents achieved a final pass rate of only 0.6%, with failures traced to agents losing track of multiple constraints and misusing their information-gathering tools. This finding directly motivates the design choice in TravelGenie to move hard constraints (budget, timing, and disruption handling) out of free-form generation and into dedicated, rule-governed agents.

The second strand concerns general frameworks for LLM-based multi-agent collaboration. AutoGen (Wu et al., 2024) provides an infrastructure in which customizable, conversable agents combine LLMs, tools, and human input through programmable conversation patterns. MetaGPT (Hong et al., 2024) instead encodes standardized operating procedures into role-specialized agents arranged in an assembly-line workflow, so that intermediate outputs are checked before they are passed downstream, reducing cascading hallucination.

TravelGenie is closer to the MetaGPT pattern than to open-ended agent conversation: its seven agents run in a fixed sequence with structured hand-offs, and the constraint-critical stages are deterministic rather than conversational.

Disruption recovery can also be viewed as a dynamic rescheduling problem, which has been studied extensively in operations research. Liu et al. (2023) apply deep multi-agent reinforcement learning to the dynamic job-shop scheduling problem, where agents learn policies that make near-real-time sequencing decisions as new jobs arrive, and Singh et al. (2025) use an advantage actor-critic approach for energy-conscious flexible job-shop scheduling, producing feasible schedules quickly under multiple operational constraints. Although these studies target manufacturing rather than tourism, they show that learned policies can handle sequential re-planning decisions under changing conditions. TravelGenie deliberately uses predefined rules for this step to keep recovery predictable and auditable; learning-based adaptive rescheduling is discussed as a future extension in Section V.

TravelGenie sits at the intersection of these threads: it keeps the natural-language flexibility of LLM generation for the creative parts of planning (destination research, itinerary drafting) while pushing the parts that require hard guarantees (budget ceilings, timing, and disruption response) into deterministic, rule-governed agents rather than leaving them to free-form generation.

METHODOLOGY

System Architecture and Pipeline

TravelGenie is organized as a three-tier layered system. The presentation layer is a React Native (v0.81.4) mobile client built on Expo SDK 54 (Expo Developer Network, 2024), targeting both iOS and Android, and rendering seven interactive screens (Welcome, Auth, Dashboard, Creator, Result, Saved Details, and Disruption Simulator) with a 60 fps rendering target, with deep links out to Google Maps, Google Flights, and Booking.com. Beneath it, a FastAPI application layer running on Uvicorn handles REST state transfer, CORS policy, and Pydantic-based request validation, and coordinates the multi-agent workflow. The data layer combines the Agno multi-agent framework (Agno AI Framework Team, 2024), connected to Groq-hosted Llama 3 models (70B and 8B) for low-latency generation, with an SQLite 3 database that acts as the single source of truth for saved itineraries, user profiles, and audit history.

The Seven-Agent Pipeline

Rather than a single generation step, itinerary construction is executed as a sequential pipeline of seven specialized agents, each consuming the state produced by its predecessor and applying its own domain logic before handing an enriched state downstream. Table I summarizes each agent's role, inputs, and processing logic.

Table I Seven-Agent Collaborative Workflow

Agent	Responsibility	Key Inputs	Processing Logic	Output
1. Destination Research	Location & transport resolution	Origin, destination, departure date	Resolves IATA codes; queries SerpAPI for fares; pulls 5-day OpenWeatherMap forecasts	Airport codes, fare, weather alerts
2. Itinerary Synthesis	Daily schedule drafting	Destination, trip length, theme, interests	Prompts the Groq LLM with a curated list of verified landmarks to build daily activity lists	Draft multi-day itinerary (JSON)
3. Logistics & Hotel	Lodging & dining selection	Destination, star rating, budget tier	Queries a curated hotel/restaurant database; converts prices at live FX rates	Hotel & dining picks
4. Budget	Max-budget	Total cost,	Checks the budget inequality; downgrades hotel tier and/or	Budget-compliant

Optimization	enforcement	budget ceiling	swaps paid entries for free ones on violation	itinerary + audit log
5. Time Scheduling	Pacing & routing	Daily activity list	Assigns time blocks (e.g., 09:00, 12:30, 14:30, 18:00) preserving a sensible geographic sequence	Time-blocked schedule
6. Comfort & Fatigue	Overload mitigation	Number of daily activities	Computes a fatigue score; inserts a mandatory rest block when the day is overloaded	Fatigue-balanced schedule
7. Disruption Manager	Contingency handling	Live weather, delay hours, closures	Applies shift, swap, and re-index rules to keep the plan valid under real-world change	Adapted itinerary

Formal Model: Budget, Scheduling, and Fatigue

Total trip cost is modeled as

$$C_{total} = P_{flight} + (N_{days} \cdot P_{hotel}) + \Sigma C(a) \quad (1)$$

where P_{flight} is the ticket price, N_{days} the trip length, P_{hotel} the nightly room rate, and $C(a)$ the entry fee of activity a. The Budget Optimization agent enforces

$$C_{total} \leq B_{max} \quad (2)$$

and, on violation, applies a two-stage fallback: Stage 1 downgrades accommodation one tier at a time (Luxury → Mid-Range → Budget); if that alone is insufficient, Stage 2 replaces any activity priced above 50 MYR with a zero-cost public alternative. Daily fatigue is modeled separately as

$$F(d) = \Sigma \text{ Duration}(a) + \alpha \cdot N_{outdoor}(d) \quad (3)$$

with $\alpha = 0.5$ representing an outdoor/heat-exposure penalty. Whenever a day's activity count reaches four or F(d) reaches 6.0 hours, a one-hour, zero-cost rest block is inserted at 17:00, which keeps the schedule physically realistic rather than simply densely packed.

The budget guarantee is conditional on feasibility. Because the fallback can only lower accommodation and activity costs, the cheapest plan it can reach is

$$C_{min} = P_{flight} + (N_{days} * P_{hotel,min}) \quad (4)$$

where $P_{hotel, min}$ is the nightly rate of the cheapest available budget-tier hotel and all paid activities have been replaced by free alternatives. When $C_{min} \leq B_{max}$, the two-stage fallback is guaranteed to return a plan satisfying (2). When $C_{min} > B_{max}$ (for example, when the live airfare alone already exceeds the ceiling), no plan within the agent's action space can satisfy (2); the agent then returns the lowest-cost plan and flags it as over budget in the audit log rather than silently exceeding the limit. In the over-budget test reported in Section IV-B, a feasible configuration existed, and the agent brought the plan within the ceiling.

Disruption Recovery Engine

Live disruptions are represented as a vector $D = \langle \text{Condition, Severity, } h_{delay}, \text{ClosedAttraction} \rangle$ (5), and the Disruption Manager applies one of three deterministic recovery rules depending on which field is populated. A flight delay of h_{delay} hours shifts the start time of the first activity by the same amount, and automatically cancels and logs any Day-1 morning activity if the adjusted arrival falls after 14:00. A weather condition of Rain or Storm triggers a category swap: any outdoor activity is replaced with an indoor alternative drawn from museum/gallery-type venues. A closure event removes the affected attraction from the day's set and substitutes a pre-verified, comparably rated alternative from the regional database. Because these rules are

explicit rather than left to open-ended generation, recovery is both fast and predictable a property that matters when the traveler is relying on the plan mid-trip.

Calendar Export (RFC 5545)

Finished itineraries can be exported as a standards-compliant .ics file for import into Apple Calendar, Google Calendar, or Outlook, following the iCalendar specification (Desruisseaux, 2009). The exporter converts each activity's local time and day offset into a canonical YYYYMMDDTHHMMSS timestamp and applies the character-escaping RFC 5545 requires for commas, semicolons, backslashes, and line breaks, so that every VEVENT block with its UID, DTSTAMP, DTSTART/DTEND, SUMMARY, DESCRIPTION, STATUS, and LOCATION fields parses cleanly in third-party calendar clients.

Evaluation Protocol

The system was evaluated in three parts. **REST API testing:** each of the six backend endpoints was called in Postman with valid inputs, invalid payloads, and unauthorized requests. A test passed when the endpoint returned the expected HTTP status and response body; response time was taken from Postman.

Scenario testing: three scenarios were run end to end through the full seven-agent pipeline, using the Disruption Simulator in the mobile app where needed: (A) normal clear weather, (B) simulated heavy rain, and (C) an over-budget input. Scenario A passed if a complete day-by-day itinerary was produced within the stated budget; Scenario B passed if the weather-sensitive outdoor activities on the affected day were replaced with indoor alternatives while the plan stayed usable; Scenario C passed if the final estimated cost was at or below the user's maximum budget. Adaptation time is the time from submitting the request to receiving the updated itinerary.

User acceptance testing: 15 participants drawn from the target user groups (students, frequent travelers, and faculty members) used the app on physical Android and iOS devices through Expo Go. Each participant completed seven tasks: register or log in; enter preferences for a hypothetical trip; generate an itinerary; review the itinerary, budget breakdown, and weather information; test weather adaptation with the Disruption Simulator; save the trip and reopen it from Saved Trips; and export it as an .ics file. Participants then rated five criteria and overall satisfaction on a 5-point Likert scale (1 = very poor, 5 = excellent). The reported values are mean scores across the 15 participants.

RESULTS

TravelGenie was evaluated in three parts, following the protocol in Section III-F: (A) REST API endpoint testing, (B) scenario-based testing of the multi-agent pipeline, and (C) user acceptance testing with 15 participants. Section IV-D discusses the limitations of this evaluation.

REST API Endpoint Testing

All six backend endpoints passed functional testing (Table II). Data operations completed in 28–55 ms. The /plan/generate endpoint, which runs the full seven-agent pipeline with sequential Groq LLM calls, SerpAPI searches, and OpenWeather requests, took 6.4 s on average, within the 5–12 s target set for plan generation. Invalid payloads and unauthorized requests returned the expected 400, 401, and 404 status codes with descriptive error messages.

Table II Rest API Endpoint Test Results

Method	Endpoint	Expected Output	Resut (Time)
POST	/auth/register	200 OK (registration success)	Pas (42 ms)
POST	/auth/login	200 OK + user_id	Pass (38 ms)
POST	/plan/generate	200 OK + full multi-agent	Pass (6.4 s)

		plan	
POST	/plan/save	200 OK (plan saved)	Pass (55 ms)
GET	/plans/user/{id}	200 OK + list of saved plans	Pass (28 ms)
DELETE	/plans/{id}	200 OK (plan deleted)	Pass (31 ms)

Multi-Agent Scenario Testing

Scenario A (normal clear weather). A 3-day Singapore trip was planned with a Mid-Range tier, a 3-star hotel preference, a maximum budget of RM 3,000, and interests in outdoor sightseeing and local cuisine. The Research Agent returned attractions including Gardens by the Bay, Marina Bay Sands, and Sentosa Island; the Hotel Agent recommended Hotel Mono in Chinatown; and the Planner and Time Agents produced 3–4 sequenced activities per day. The Budget Agent estimated RM 2,450, within the RM 3,000 limit, and the forecast showed clear skies at 26–32°C.

Scenario B (simulated heavy rain). Heavy rain was applied to Day 2 of the same trip through the Disruption Simulator. The Disruption Manager identified the weather-sensitive outdoor activities (a walking tour of the Singapore Botanic Gardens and a visit to the Southern Ridges) and replaced them with the ArtScience Museum and the National Gallery Singapore. Weather alert flags were added to explain the change, and the revised plan was passed back through the Time and Budget Agents to update routes and costs.

Scenario C (over-budget input). A 4-day Tokyo trip was planned with a Mid-Range tier and a maximum budget of RM 2,000. The first draft cost RM 2,850, RM 850 over the limit. The Budget Agent triggered the fallback: a cheaper hotel that still met the 3-star requirement, local ramen shops and izakayas in place of two premium dining experiences, and free attractions such as the Imperial Palace East Gardens and Meiji Shrine in place of a paid activity. The revised plan cost RM 1,980, within the limit. Table III summarizes the three scenarios.

Table III Multi-Agent Scenario Test Results

Scenario	Test Focus	Outcome	Time
A – Normal weather (Singapore, 3 days)	Standard itinerary generation	Complete plan; RM 2,450 vs. RM 3,000 limit	6.4 s
B – Heavy rain, Day 2 (Singapore)	Weather disruption adaptation	Outdoor activities replaced with indoor venues	8.2 s
C – Over-budget (Tokyo, 4 days)	Budget constraint enforcement	Cost reduced from RM 2,850 to RM 1,980 (limit RM 2,000)	9.1 s

All three scenarios passed. Scenario C shows the budget fallback of Section III-C working in a case where a feasible lower-cost plan existed. Scenarios B and C took longer than plain generation because the revised plan is re-processed by downstream agents before it is returned.

User Acceptance Testing

All 15 participants completed the seven UAT tasks on their devices. Table IV shows the mean ratings.

Table IV User Acceptance Testing Results (N = 15)

Evaluation Criterion	Mean (out of 5)	Satisfaction Level
Mobile UI/UX aesthetics and ease of use	4.73	Very high
Itinerary personalization and relevance	4.60	High

Budget estimation accuracy	4.53	High
Weather and disruption adaptation capability	4.80	Very high
Mobile app speed and responsiveness	4.47	High
Overall system satisfaction	4.63	Excellent

Weather and disruption adaptation received the highest rating (4.80); participants valued seeing their plan adjust in the Disruption Simulator. The interface was rated 4.73, and personalization 4.60, with participants noting that the agent reasoning text explaining each recommendation made the plan easier to trust. Budget estimation was rated 4.53; participants found the itemized breakdown and within/over-budget indicator useful but noted that real prices may differ from estimates. Speed and responsiveness received the lowest score (4.47) because generation takes several seconds, although the loading indicator helped. Suggested improvements were dark mode, more detailed restaurant recommendations, offline access to saved itineraries, and direct booking.

Limitations and Threats to Validity

Several limitations qualify these results. First, the evaluation did not include a comparison with a single-agent or single-prompt baseline. The results therefore show that the seven-agent system works as designed, but they cannot isolate how much of its behavior comes from multi-agent decomposition rather than from live data access, deterministic rules, or the underlying model. Second, scenario testing covered three scenarios; the flight-delay and venue-closure rules described in Section III-D are available in the Disruption Simulator but were not benchmarked separately. Third, the UAT involved 15 participants planning hypothetical trips, and its scores reflect self-reported perceptions rather than objective measures of plan quality; the sample is small and drawn from a limited population. Fourth, costs are estimates, and actual prices may differ. Finally, response times depend on external LLM, search, and weather services and were measured under test conditions.

CONCLUSIONS

TravelGenie shows that decomposing itinerary planning into seven cooperating, narrowly scoped agents layered on a React Native/Expo, FastAPI, and SQLite stack, combined with live data and deterministic constraint rules, offers a practical way to address three weaknesses of static OTA packages and single-prompt LLM planners: hallucinated content, soft budget enforcement, and an inability to react to real-world disruption. In testing, all six REST endpoints passed, full itinerary generation averaged 6.4 s, the pipeline handled heavy-rain and over-budget scenarios correctly, and 15 UAT participants rated overall satisfaction at 4.63 out of 5. Itineraries can also be exported to standard calendar applications in RFC 5545 format. Planned extensions include chaining itineraries across multiple cities in a single trip, caching agent outputs for partial offline operation, and integrating live hotel-booking APIs so that recommended accommodation can be reserved directly from the app rather than only referenced.

Future evaluation will add a controlled comparison against single-prompt and single-agent baselines, a larger user study, and separate benchmarks for the flight-delay and venue-closure rules. Finally, the current Disruption Manager relies on predefined shifting, substitution, cancellation, and re-indexing rules. Future research could investigate learning-based adaptive rescheduling, for example multi-agent reinforcement learning (Liu et al., 2023) or actor-critic policies such as advantage actor-critic or proximal policy optimization (Singh et al., 2025; Schulman et al., 2017), trained on logged disruption episodes, while keeping the existing rules as a safety fallback.

ACKNOWLEDGMENT

The authors would like to thank the Faculty of Artificial Intelligence and Cyber Security (FAIX), Universiti Teknikal Malaysia Melaka (UTeM) for their assistance in this research.

REFERENCES

1. Agno AI Framework Team. (2024). Agno: Building multi-agent systems with memory, knowledge and tools [Technical documentation]. https://docs.agno.com
2. Buhalis, D., & Law, R. (2008). Progress in information technology and tourism management: 20 years on and 10 years after the Internet—The state of eTourism research. Tourism Management, 29(4), 609–623. https://doi.org/10.1016/j.tourman.2008.01.005
3. Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology, 15(3), Article 39, 1–45. https://doi.org/10.1145/3641289
4. Desruisseaux, B. (Ed.). (2009). Internet calendaring and scheduling core object specification (iCalendar) (RFC 5545). Internet Engineering Task Force. https://doi.org/10.17487/RFC5545
5. Expo Developer Network. (2024). Upgrading Expo SDK and React Native workflows [Documentation]. https://docs.expo.dev
6. Gretzel, U., Sigala, M., Xiang, Z., & Koo, C. (2015). Smart tourism: Foundations and developments. Electronic Markets, 25(3), 179–188. https://doi.org/10.1007/s12525-015-0196-8
7. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., & Schmidhuber, J. (2024). MetaGPT: Meta programming for a multi-agent collaborative framework. In Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024). https://doi.org/10.48550/arXiv.2308.00352
8. Liu, R., Piplani, R., & Toro, C. (2023). A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem. Computers & Operations Research, 159, Article 106294. https://doi.org/10.1016/j.cor.2023.106294
9. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347
10. Singh, S. S., Joshi, R., & Gupta, D. (2025). An advantage actor-critic approach for energy-conscious scheduling in flexible job shops. Journal on Artificial Intelligence, 7(1), 177–203. https://doi.org/10.32604/jai.2025.065078
11. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2024). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6), Article 186345. https://doi.org/10.1007/s11704-024-40231-1
12. Wooldridge, M. (2009). An introduction to multiagent systems (2nd ed.). John Wiley & Sons.
13. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2024). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In Proceedings of the First Conference on Language Modeling (COLM 2024). https://doi.org/10.48550/arXiv.2308.08155
14. Xie, J., Zhang, K., Chen, J., Zhu, T., Lou, R., Tian, Y., Xiao, Y., & Su, Y. (2024). TravelPlanner: A benchmark for real-world planning with language agents. In Proceedings of the 41st International Conference on Machine Learning (PMLR Vol. 235, pp. 54590–54613). https://proceedings.mlr.press/v235/xie24j.html