ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 5 Issue 4 October 2025
Paper Id: JETA-V5I4P120 / Page No: 138-144

Original Article

Smart Dining: An AI-Powered Personalized Restaurant Recommender System

Dilip Mehendra Sharma
North American University, Stafford-TX.

Received Date: 05 December 2025 Revised Date: 12 December 2025 Accepted Date: 19 December 2025

Abstract: *It is difficult to agree on a restaurant to eat at, especially in groups which make it time-consuming. An intelligent recommendation system is presented in this work which leverages machine learning and the genetic algorithms for restaurant selection based on user behaviour. The system has been executed like a web-based app that makes recommendations for the individual and group alike. It also helps to reduce the choice overload of the target audience or user group. Because the approach is data-driven it is better than random. After testing, its efficacy is proven in making decision making easy for one and all.*

Keywords**:** *Smart Dining, Restaurant Recommender System, AI.*

I. INTRODUCTION

The world today is rife with digital personalization. Yet, restaurant recommendation services in Thailand still only utilize fixed filters. Variety of tastes and other contextual elements make group dining decisions complicated. This often happens when users must balance conflicting options without any sort of custom help. An AI based recommendation framework was built using user behavior and stated preferences to combat these challenges. Machine learning modules were incorporated to deduce personal preferences from explicit and implicit signals. A genetic algorithm was mixed in to maximize global satisfaction through a candidate set of restaurants. A web interface was chosen to make it platform independent and easy to access The expectations based on experience are a lower decision time and greater satisfaction for the solo and group dining forms.

II. RELATED WORK

Personalized recommender systems have been shown to improve engagement in e-commerce and content platforms. In comparison, apps for Thai restaurants mainly used fixed listings that did not adapt[1]. There is evidence that the ratio of explicit feedback (based on ratings) to the implicit feedback (based on interaction log) affect recommendation accuracy and recommendation data abundance[2].Content-based filtering methods use metadata about items to compare with user profiles although they can tend to be limited in diversity. Collaborative filtering methods rely on user–item matrices with similarity metrics such as Pearson correlation and cosine distance, and can be memory- or model-based [3]. To solve the cold-start and sparsity problem, hybrid approaches have been proposed to leverage the content-based and collaborative signals. Research on group recommendations has explored which way to aggregate individuals' preferences to form a single collective recommendation. In the context of multiple objectives, genetic algorithms have been used to improve recommendations subsets under complex constraints.[4].

III. METHODS

A three-tier architecture was implemented: a browser-based front end, a server-side API, and a database for persistence [1]. User profiling involved the collection of credentials, explicit ratings, and implicit interaction metrics (clickstreams, dwell times). Explicit ratings were used to calibrate initial preference models, while implicit signals enabled continuous adaptation [2]. Content-based filtering computed similarity scores between user profiles and restaurant metadata (cuisine, price, location) to generate personalized lists. Collaborative filtering applied both user- and item-based memory methods, using Pearson correlation and cosine similarity on the user–item matrix [3]. Hybridization combined the outputs of content based and collaborative streams via weighted aggregation to enhance robustness. Group preferences were derived by applying average-score, least-misery, and most-pleasure heuristics to individual scores, producing a consolidated ranking. A genetic algorithm iteratively optimized the top-N recommendation set: candidate sets were encoded as chromosomes, fitness was defined by aggregate satisfaction, and standard selection, crossover, mutation, and termination criteria were employed [5].

IV. TECHNOLOGIES

A. Front–End Technologies

Web pages use HTML. It is a coding language. It helps in a video, image, and text, and other interactive materials and it is annotated semantically. JavaScript makes it possible to create dynamic behaviors, manipulate the Document Object Model, and interact with real-time users. Integrating TypeScript into Go code promotes more maintainable code through static type checking. With React, one can build declarative user interfaces using components. State management is easy with virtual DOM.

B. Back–End Technologies

Node.js is a software platform that allows JavaScript to be run on the server side. Express is a web framework developed on Node.js. It is helpful for JavaScript code that deals with HTTP requests. Flask is a small framework which makes use of the programming language called python to present REST APIs which work as useful entities during data analysis and model inference. Programming professionals extensively use Python for data processing and creating machine learning algorithms. JupyterLab allows you to experiment and prototype quickly.

C. Communication Protocols

HTTP is employed as the client–server protocol for re source exchange, with standard methods (GET, POST, PUT, DELETE) indicating requested actions on resources. Stateless ness is addressed through HTTP cookies, which carry session identifiers and authentication tokens to preserve user context across isolated requests.

D. Data Storage

MongoDB is chosen as the document-oriented database to accommodate evolving schemas and high-volume data storage. Mongoose is employed to define application data models, enforce schema validation, and streamline query construction within the Node.js environment.

E. Deployment

Google Cloud App Engine is used to host web services in a fully managed, serverless environment, enabling automatic scaling and simplified deployment workflows. MongoDB Atlas is configured as a managed cloud database service, providing high availability, automated backups, and seamless connectivity with application components.

F. External Services

The Facebook Graph API is consumed to ingestrestaurant metadata and user-generated tags for enrichment of the recommendation dataset. Google Cloud Storage is utilized to host and serve static assets (e.g., restaurant images), offloading storage and bandwidth demands from application servers. Longdo Map Search is integrated to provide geolocation and place-search capabilities tailored to Thai locales, supporting coordinate retrieval and keyword queries.

V. COMPETING SOLUTIONS

A. Entrée

Entree presents food imagery from nearby restaurants via ´ a swipe-based interface, learning individual preferences over time and offering contextual suggestions for occasions such as birthdays or date nights. Integration with Foursquare provides review data and allows in-app Uber booking where supported. Recommendation is limited to single users across 30 global cities and relies on visual appeal rather than explicit group preference modelling.

B. Eatsee

Eatsee enables preference specification through initial set tings followed by a swipe interface to save favored dishes. Available in select Australian and U.S. cities, Eatsee focuses on item-level recommendations without behavioural adaptation, and does not aggregate group preferences or prioritise restaurant-level suggestions.

C. Tinder

Tinder employs a swipe paradigm to match users based on profile attributes and photographic cues, refining recommendations via ongoing interaction. Although analogous in its algorithmic approach, Tinder optimizes individual social matching rather than dining experiences, and does not address group consensus or location-specific restaurant data.

D. Tourism Recommender Framework

An e-Tourism platform employs the Generalist Recom mender System Kernel (GRSK) to deliver both individual and

group travel suggestions by aggregating personal ratings, demographics, and past visits. Aggregation mechanisms such as intersection and average-score are used, but domain focus remains on tourist attractions rather than dining venues, and external restaurant databases are not integrated.

E. Genetic Algorithm–Based Music Recommender

A music recommender applies a genetic algorithm with BLX- crossover to evolve track feature sets based on user favourites. Initial populations are generated via content-based filtering, and iterative evolution maximizes fit to user profiles. While offering insights into GA application for personalized suggestions, the system targets music features rather than multi-criteria restaurant satisfaction or group dynamics.

VI. METHODOLOGY AND DESIGN

A. Functional Requirements

Account management: registration, login, logout, and optional guest access.

- Profile configuration: multi-step setup including general information and food-preference selection.
- Individual recommendation: location-aware retrieval, swipe-style feedback, and final selection capture.
- Group recommendation: pin-based group formation, con current preference ranking, and consensus selection.
- Persistent favorites: saving and retrieval of preferred restaurants.
- Preference editing: revision of stored taste profiles at any time.

B. Use Cases

Authentication is handled via a login form requiring valid credentials; invalid submissions trigger form re-display or redirection to registration. Registration entails a two-stage process: personal details and food-preference capture, with validation at each step. Individual recommendation is initiated by location consent, followed by iterative presentation of restaurant options until confirmation of choice. Group recommendation begins with group creation and sharing of a secure code, after which members simultaneously rank options and a final group choice is computed. Preference editing and favorites management are accessible from the profile interface at any point.

C. System Components

App Client serves the user interface in a browser, handling user input and presenting recommendations. App Server ex poses RESTful endpoints, orchestrates authentication, user and group state, and delegates recommendation requests. Recom mender implements machine learning and genetic-algorithm routines via a Python/Flask service, consuming user and restaurant data to generate ranked suggestions. Data Service ingests and preprocesses external restaurant metadata via API scripts. Database persists user profiles, preferences, restaurant records, and session data in a document store.

D. System Architecture

Inter-component communication is conducted over HTTP with stateless requests augmented by session cookies. Deploy ment targets include Google Cloud App Engine for App Client, App Server, Recommender, and Data Service, and MongoDB Atlas for database hosting. External integrations comprise Facebook Graph API for restaurant metadata ingestion, Google Cloud Storage for static asset delivery, and Longdo Map Search for Thailand-focused geolocation services.

VII. USER INTERFACE

A. Authentication and Onboarding

A login screen prompts for username and password, with validation feedback and a guest-access option. Registration is presented as a two-page flow: initial credential capture followed by optional profile and food-preference forms. Error states are indicated inline, and progression is gated by field completion and format validation.

B. Recommendation Workflow

Individual recommendations are displayed as sequential cards summarising restaurant details. Interactive controls allow “like” to confirm a choice or “dislike” to request alternatives. Filters and informational overlays are accessible via dedicated controls. Group recommendation mirrors this interface, pre ceded by group-code entry and concurrent ranking until a consensus result is presented.

C. Profile and Preference Management

A profile dashboard lists saved favorites, past selections, and current preference settings. Editable fields include general user information and category-based food tastes. Privacy and account settings are collated in a unified section, with logout

available at all times.

VIII. DATABASE STRUCTURE

The persistence layer is implemented as a document oriented database, with five principal collections storing ap plication data.

A. Collections

Five collections were defined:

- Restaurants: stores restaurant records, including identifying metadata, profile attributes, location coordi nates and external links.
- Users: contains authentication credentials, demographic fields and food-preference settings.
- Recommendations: logs each recommendation session, recording participant IDs, interaction histories and geo-context.
- Categories: maintains a catalog of cuisine or restau rant types, reducing redundancy across restaurant docu ments.
- Favorites: maps user IDs to lists of bookmarked restaurant ObjectIds.

B. Schema Definitions

Mongoose schemas enforce a uniform data format without preventing heterogeneous inserts. Mandatory fields include:

- Restaurants: name, profile.categories (Ar rayObjectId¿), profile.price_range (Number), profile.rating (Number), profile.likes (Number), address, location.coordinates (Array Number), link.
- Users: authentication (token object), username, password, profile.gender, profile.preference (Object). profile.birthdate,
- Recommendations: users (ArrayObjectId¿), histories (Array of subdocuments with restaurant (ObjectId), is_love (Boolean), location.coordinates. timestamp),
- Categories: name, is_common (Boolean).
- Favorites: user (ObjectId), restaurants (Array ObjectId¿).

C. Indexing

Indexes were created on frequently queried fields: restaurants.location.coordinates (2dsphere) for geoqueries; users.username (unique) for authentica tion; recommendations.users for session favorites.user for rapid favorites retrieval.

IX. DATA COLLECTION AND PREPROCESSING

A. Restaurant Data Ingestion

Restaurant metadata was retrieved from a Thai governmental API providing names and coordinates. Records were enriched via the Facebook Graph API to obtain categorical tags, price ranges, user ratings, engagement metrics and image URLs. Duplicate entries were resolved by applying a fuzzy name-similarity threshold of 0.65 combined with a 200 m spatial proximity filter; the record with the most complete metadata was retained.

B. Restaurant Dataset Overview

The final dataset comprises 11 175 unique restaurants within Bangkok. Spatial density peaks in central districts. Category tags are dominated by “Restaurant,” “Thai Restaurant,” “Food & Beverage” and “Fast Food,” while 50% of records include price-range data (levels 1–4, mid-range most common). Image URLs are present for 30 % of entries, highlighting a need for additional visual resources.

C. User Interaction Data Collection

A custom web tool captured demographic profiles, initial food-preference settings and interactive ratings on 20–60 restaurants per participant. Three recruitment methods (in person, virtual interview, online survey) yielded 143 respon dents and 5 102 interaction records. Non-Bangkok participants were excluded to align with the restaurant coverage.

D. User Dataset Overview

Gender distribution was approximately equal; age clustered between 20 and 25 years. The mean interactions per user were 36. Category preference scores (0–100 scale) peaked for Japanese, Thai and Shabu Shabu cuisines and were lowest for Chinese cuisine. Rating distribution centered on neutral values.

E. User Feedback

Qualitative feedback indicated that high ratings correlated with prior visits and visual appeal. Recommendation relevance was affirmed by 99 % of interview participants, who also requested more restaurant imagery and menu previews. Price range labels were clarified following reports of user confusion.

F. Training Data Utilization

Initial collaborative-filtering experiments were conducted using the collected dataset. Following a pivot to a genetic algorithm–driven recommendation approach in the second semester—requiring no pre-training—the dataset was retained for exploratory analysis and potential future model refinement.

X. RESULTS

A. User Journey Refinement

Modifications to the user journey were implemented across four key workflows: login and registration, home screen navigation, individual recommendation presentation, and group recommendation interaction. Mandatory food-preference cap ture was introduced in the registration path, eliminating guest access and skip-flow options. A central home screen was added to simplify navigation between feature modules. Recommendation interfaces were redesigned to display multiple candidate restaurants simultaneously, replacing the prior one-at-a-time swipe model. In the group recommendation workflow, a fixed set of ranked options was adopted, and participants were instructed to submit ordered preferences rather than iterative binary decisions.

B. System Performance

System performance under load was assessed via a 100- concurrent-request benchmark. An average end-to-end response latency of 4 747.69 ms was recorded. The recommendation engine exhibited the highest tail latency at 11 243 ms when servicing concurrent suggestion requests. The genetic algorithm service emitted up to 873.33 KB/s of outbound data due to the transmission of expanded restaurant candidate lists.

Table 1 : Load Test Results (100 Concurrent Requests)

Metric Value Average response time	4 747.69 ms
Maximum response time (Recommender)	11 243 ms
Maximum outbound data rate (Recommender)	873.33 KB/s

C. Recommendation Evaluation

User satisfaction was evaluated via structured surveys of 41 end users. In the individual recommendation context, 88 % of respondents indicated that suggested restaurants aligned with their culinary preferences, and 76 % reported accurate price-range matches. Group recommendation feedback (n = 14) yielded an 82 % agreement rate for both food and price alignment. Additionally, 63 % of users reported that decision making was expedited compared to unaided processes.

Table 2 : User Satisfaction Metrics

	Food Match	Price Match
Individual Recommendation	88%	76%
Group Recommendation	82%	82%

D. Dataset Summary

An initial harvest of 11 175 restaurant records from governmental and social-media APIs underwent deduplication via spatial (200 m) and name-similarity (0.65 threshold) heuristics, removing approximately 2 000 duplicate entries. Post processing yielded 9 175 unique records. Price-range metadata was present in 50 % of records, while 30 % included image URLs, of which approximately one-third represented relevant restaurant visuals.

Table 3 : Restaurant Dataset Characteristics

Original records retrieved	11 175
Duplicate entries removed	2 000
Final unique restaurants	9 175
Records with price range	50%
Records with images	30%

XI. CONCLUSION

A personalized restaurant recommendation system was developed and evaluated, integrating content-based, collaborative, and genetic-algorithm techniques within a webapplication framework. The design addressed both individual and group dining

scenarios, overcoming decision fatigue through tailored candidate sets and ranking aggregation. Empirical assessments demonstrated system robustness un der load, with acceptable latency for prototype deployment and high user satisfaction in matching food and price preferences. The revised user journey and interface improvements contributed to a streamlined experience, as reflected in 90 %+ ease-of-use ratings and a 95 % perceived recommendation relevance.

Remaining limitations include incomplete image coverage, occasional category inaccuracies from user-generated meta data, and residual duplicate entries beyond automated filtering thresholds. Group discussion support was identified as an area for enhancement to reduce waiting-state confusion during consensus formation.

Future work will focus on enriching visual content via supplemental APIs, incorporating real-time group chat for collaborative decision making, and tuning genetic-algorithm parameters through A/B experiments. Continued data collection and performance monitoring will guide iterative refinements toward production readiness and expanded geographic coverage.

XII. REFERENCES

[1] M. Rouse, “What Is Web Application (Web Apps) And Its Benefits,” Available at https://searchsoftwarequality.techtarget.com/definiti on/We b-application-Web-app, [Online; accessed 28-November-2020].

[2] Wikipedia, “Recommender system,” Available at https://en.wikipedia .org/wiki/Recommender system#Content-based filtering, [Online; accessed 20-November-2020].

[3] ——, “Collaborative filtering,” Available at https://en.wikipedia.org/w iki/Collaborative filtering, [Online; accessed 21-November-2020].

[4] ——, “Genetic algorithm,” Available at https://en.wikipedia.org/wiki/ Genetic algorithm, [Online; accessed 16-March-2021].

[5] V. Mallawaarachchi, “Introduction to Genetic Algorithms — Including Example Code,” Available at https://towardsdatascience.com/introdu ction-to-genetic-algorithms-including-example-code-e396e98d8bf3, [Online; accessed 16-March-2021].

[6] F. Isinkaye, Y. Folajimi, and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” Egyptian Informatics Journal, vol. 16, no. 3, 2015.

[7] E. S. Team, “What is Machine Learning?” Available at https://expertsyst em.com/machine-learning-definition/, [Online; accessed 26-November 2020].

[8] S. Singh, “Algorithms for Discrete Optimization,” Available at https: //mech.iitm.ac.in/nspch52.pdf, [Online; accessed 16-March-2021].

[9] Wikipedia, “Mathematical optimization,” Available at https://en.wikip edia.org/wiki/Mathematical optimization, [Online; accessed 16-March 2021].

[10] MDN, “HTML: HyperText Markup Language,” Available at https://de veloper.mozilla.org/en-US/docs/Web/HTML, [Online; accessed 26-November-2020].

[11] ——, “JavaScript,” Available at https://developer.mozilla.org/en-US/do cs/Web/JavaScript, [Online; accessed 26-November-2020].

[12] Microsoft, “TypeScript,” Available at https://www.typescriptlang.org/, [Online; accessed 27-November-2020].

[13] O. Foundation, “Introduction to Node.js,” Available at https://nodejs.d ev/learn, [Online; accessed 27-November-2020].

[14] P. Hunt, “Why did we build React?” Available at https://reactjs.org/bl og/2013/06/05/why-react.html, [Online; accessed 27-November-2020].

[15] DA14, “TOP 10 ADVANTAGES OF USING REACT.JS,” Available at https://da-14.com/blog/its-high-time-reactjs-ten-reasons give-it-try, [Online; accessed 27-November-2020].

[16] Tutorialspoint, “Python Tutorial,” Available at https://www.tutorialspoi nt.com/python/index.htm, [Online; accessed 27-November-2020].

[17] O. Bahaieva, “Top 7 Reasons Why You Need to Learn Python as a Data Scientist,” Available at https://towardsdatascience.com/top-10-reasons -why-you-need-to-learn-python-as-a-data-scientist-e 3d26539ec00, [Online; accessed 27-November-2020].

[18] JupyterLab, “JupyterLab Overview,” Available at https://jupyterlab.rea dthedocs.io/en/stable/getting started/overview.html, 6-December-2020]. [Online; accessed 6-December-2020].

[19] C. Creative, “What Is Figma? a 101 Intro,” Available at https://desi gnshack.net/articles/software/what-is-figma-intro/, [Online; accessed 6-December-2020].

[20] MDN, “An overview of HTTP,” Available at https://developer.mozilla. org/en-US/docs/Web/HTTP/Overview, accessed 27-November 2020]. [Online; accessed 27-November 2020].

[21] ——, “HTTP request methods,” Available at https://developer.mozilla. org/en-US/docs/Web/HTTP/Methods, accessed 28-November 2020].

[22] I. StrongLoop, “Express,” Available at https://expressjs.com/, [Online; accessed 28-November-2020].

[23] T. P. Projects, “Flask,” Available at https://palletsprojects.com/p/flask/, [Online; accessed 28-November-2020].

[24] I. MongoDB, “What is NoSQL?” Available at https://www.mongodb.co m/nosql-explained, [Online; accessed 28-November-2020].

[25] Guru99, “What is MongoDB? Introduction, Architecture, Features & Example,” Available at https://www.guru99.com/what-is-mongodb.html, [Online; accessed 28-November-2020].

[26] Mongoose, “mongoose,” Available at https://mongoosejs.com/, [Online; accessed 28-November-2020].

[27] Google, “Google App Engine Documentation,” Available at https://clou d.google.com/appengine/docs, [Online; accessed 17-November-2020].
[28] I. MongoDB, “MongoDB Atlas,” Available at https://docs.atlas.mongo db.com/#service-fullname, [Online; accessed 18-November-2020].
[29] Facebook, “Graph API,” Available at https://developers.facebook.com/ docs/graph-api/, [Online; accessed 18-November-2020].
[30] S. Gopinathan, “Entree - choose from a world of dishes with a single ´ touch,” Available at https://techweek.com/entree-choose-dishes-single-t ouch/, [Online; accessed 29-August-2020].
[31] A. Store, “Eatsee - See it. Eat it,” Available at https://apps.apple.com/u s/app/id1202110881, [Online; accessed 30-August-2020].
[32] Wikipedia, “Tinder (app),” Available at https://en.wikipedia.org/wiki/Ti nder (app), [Online; accessed 30-August-2020].
[33] I. Garcia, L. Sebastia, and E. Onaindia, “On the design of individual and group recommender systems for tourism,” Expert Systems with Applications, vol. 38, no. 6, 2011.
[34] C. I, “Build a user-based collaborative filtering recommendation engine for Anime,” Available at https://towardsdatascience.com/build-a-user-b ased-collaborative-filtering-recommendation-engine-for-anime-92d35 921f304, [Online; accessed 30-August-2020].
[35] MDN, “Manipulating documents,” Available at https://developer.mozill a.org/en-US/docs/Learn/JavaScript/Client-side web APIs/Manipulating documents, [Online; accessed 30-April-2021].
[36] Wikipedia, “Stateless protocol,” Available at https://en.wikipedia.org/w iki/Stateless protocol, [Online; accessed 30-April-2021].
[37] GoodFirms, “What is a Web Framework?” Available at https://www.go odfirms.co/glossary/web-framework/, [Online; accessed 30-April-2021].
[38] H.-T. Kim, E. Kim, J.-H. Lee, and C. W. Ahn, “A recommender system based on genetic algorithm for music data,” ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings, vol. 6, 01 2010.
[39] F. Developer, “Whatever mans,” Available at https://play.google.com/st ore/apps/details?id=com.DEC.Eatwhat, [Online; accessed 27-November 2020].