

Comora: An Agenda-First Social Gathering Platform for Community Building

Suryadevara Aditya^{1*}, Ankur Sai Teja¹, Veeraj Thota¹, Rajashekar Jadi¹

¹Department of Computer Science and Engineering, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, India

Abstract: In this paper, we examine Comora, a social interaction platform that connects individuals based on shared interests (i.e. commonality), rather than using traditional matching methods (i.e. demographic-based). Event hosts will be able to use the platform to connect with individuals who share a mutual interest; therefore, individuals looking for community engagement can find events based on that mutual interest; additionally, hosts can review the attendees of events to plan and execute new events in the future based on those event attendees. Some unique or novel characteristics of the platform pertain to the use of Mistral AI for conversational onboarding; Groq with Llama 3.3 70B for automatically generated content drawn from input provided by users; and a conversational chatbot interface that is consistently aware of the context in which conversations occur. The serverless architectural configuration of the platform developed consists of the following components: React 19 framework for the client side, Postgres with Row Level Security (Or Object Level Security) for the database tier, Free-tier cloud infrastructure provided by Supabase Cloud Infrastructure, and additional elements that illustrate the "Production Ready" quality attributes of Comora (i.e., Role-Based Access Control; Four-Axis Peer Review Process, Contribution/ Capture Mechanism that supports both qualitative and quantitative evaluations). Our study demonstrates how the combined application of Structured Interest Matching, coupled with Artificial Intelligence support, allow Hosts to create/organise events with lesser impediments in event creation/event attendance process for societies who are part of large metropolitan areas.

Keywords: Agenda-first matching, AI-augmented user experience, community building, conversational AI, peer-to-peer dining, PostgreSQL Row Level Security, serverless architecture, social gathering platform.

1. Introduction

The rise of urban social isolation has been accelerated as the result of population surges (urbanisation) that have occurred across India's key metropolitan regions. The pace of change within organised labour (employment) within urban centres is creating a disconnect between peoples' ability to create or other-wise access historic forms of community building. All manner of individuals, including early professional graduates, newly re-located students, and many other forms of migrant workers, are facing significant challenges in terms of establishment of sustainable social relations with their peers outside of their work and family, through the use of social and mobile applications as well as other forms of social media. While social technologies have created opportunities for limited

connection; the overwhelming majority of these interactions fall short of creating genuine, and in person, high-quality relationship with others who share common interests and/or objectives.

In general, the current three types of social discovery platforms are (1) modeled primarily on criteria for establishing dating relationships through demographic matching; (2) focused on forming career networking relationships among users through similar interests; and (3) used to plan and post public events (such as concerts) based on historical attendance data versus overall average level of quality. They all have a similar issue in terms of treating the connection between users as a separate or independent outcome instead of being an integral part of how users participate in these activities. With this approach, after using the service, users will often leave the event without returning or engaging with it again.

With Comora's agenda-first approach to social events, the gap between social events and social connections in the marketplace will be bridged. Comora enables event hosts to create an agenda for their events (book club, movie nite, philosophical discussion, skill-building workshop, etc), creates a matching tool for users to connect on alignment of interest, and allows guests to self-select to participate in the events. This creates an environment where users are connected through the activities they do with other users.

The Comora platform's technical architecture is a strong example of how to build sophisticated AI-augmented social networks on a purely free-tier infrastructure, while still respecting production-ready design principles. An example of this is the use of React19 for creating responsive user interfaces to deliver an exceptional user experience across multiple devices; the use of PostgreSQL with Row Level Security to secure database level access; the use of TanStack React Query to manage optimistic state, and the integration of Groq and Mistral AI in an effort to reduce the friction associated with hosting, firewalls and separately creating unique and the integration of Groq and Mistral AI to reduce the friction of hosting, firewalls, and separately creating unique guest experiences.

The Comora components are as follows: 1) A Conversational AI onboarding process provides the mechanism for populating user archetypes; 2) A wizard for creating events has several multi-step capabilities and will provide the ability to create live

*Corresponding author: suryadevaraaditya7@gmail.com

event content via AI; 3) A multi-dimensional peer review of each guest's experience (peer to peer feedback) for attending events hosted by using Comora; 4) An administrative control panel will provide multiple role-based access; dispute resolution; and audit log capabilities; and 5) A validation of the ephemeral serverless deployment of an academic social platform will demonstrate an academic scale of success on the Comora Platform.

2. Literature Survey

A. Peer-to-Peer Social Dining Models

George documented the rise of supper clubs in metropolitan cities within India stating that millennials are able to create profitable community experiences from home dining via the use of curated gathering models which emphasise shared experiences rather than the commercial nature of traditional restaurants [1]. Davies et al. examined social practices of hosting peer-to-peer dining events, identifying sustainability benefits and the importance of host authenticity in creating memorable guest experiences [2].

In the comparison between private social dining and traditional restaurants, Lin, Ok, and Au found that the guests in peer-to-peer settings had higher social satisfaction and perceived authenticity, but the consistency of the food quality could be lower [3]. These results indicate that it is the social dimension of dining, rather than culinary excellence, that is the key driver of guest satisfaction in home-hosted formats.

Hamilton's historical analysis of underground supper clubs in Dublin revealed that episodic, pop-up dining formats thrive when offering exclusivity, storytelling, and insider community sense that conventional restaurants cannot replicate [4].

Demetry broadened this idea to include episodic organizations, suggesting that temporary setups like pop-up restaurants make use of limited-time experiences to increase customer [5].

B. Motivation and Trust in Sharing Economy

Lin's study about peer-to-peer dining pointed out four main motivations: social connection, cultural experience, affordability and new experience. The majority of study participants reported that social connectivity was a more powerful motivator for them than financial incentives. This challenges the usual view that most people are motivated to participate in the sharing economy by the desire to make money [6].

A study conducted by Atsız et al. emphasizes the importance of certain values of the dining experience that go beyond the food, such as shared rituals, vulnerability, and storytelling in forming relationships with strangers. The researchers highlight that successful peer-to-peer dining services should create conditions for building connections and trust instead of coordinating the process of preparation and consumption of food [7].

C. Recommender Systems for Group Events

Zhou gave an extensive overview of recommender system that uses deep learning method and pointed out that

collaborative filtering and content-based methods faced challenge of cold start in the group recommendation tasks when no previous interaction data is available [8]. Liao and his colleagues proposed a new smart system for group event recommendation which uses social signals and preferences, showed that recommendations were better than recommendations for individuals [9].

Yu and colleagues proposed a geo-social group recommendation method that combines geolocation and social graph embeddings for predicting the group's interest in events instead of individuals. The authors' model performed significantly better in situations when people participate in events together as friends [10]. Ioannidis, Muthukrishnan, and Yan created a group recommender system that prioritizes consensus among members. This system aims to align individual preferences in order to identify events that meet the approval of all participants, thereby minimizing the chances of people not attending due to disagreements within the group [11].

The various experiences from studies such as these affect the design of Comora in that transparency in schedules alleviates cold start problems by offering users opportunities to express their preferences through event selection and the four-dimensional scale of reviews provides for better feedback than the scalars used in the recommendations.

3. System Architecture and Requirements

A. Development Methodology

The project used an agile development approach based on iterative and incremental rapid prototyping and AI-assisted development tools. "We put capabilities on the foundational platforms, and then we built more advanced features over time iteratively." In the early iterations we built out our core infrastructure, focused on strong authentication, event creation workflows, RSVP functionality, and the Match Me onboarding system to test the agenda-first matching hypothesis. The later versions of the application incorporated AI so we could automate some of the friction within hosting, such as description writing, the Compass personality profile engine, and the Sage conversational assist interface. During later stages of the project, we built out the community-building features such as admin moderation systems, messaging between users, and formalized support-ticket workflows. We also polished the user experience of the application's interface, did extensive QA testing, verified the application's compliance with accessibility requirements, and created demo materials for the application's production deployment on the free-tier of infrastructure.

B. System Requirements

This cloud-provided, web-based platform doesn't require any specific hardware to be hosted and operated by the cloud provider. The development environment consists of Node.js v18+ and uses Vite v8 as the build tool. Front-end dependencies include React 19.2.4, React Router DOM 7.13.2, TanStack's React Query 5.95.2 for caching the server's data, Tailwind CSS 4.2.2 for utility-first design, and Framer Motion 12.38.0 for animation effects.

The back-end infrastructure consists of Supabase, which provides PostgreSQL databases, authentication and storage, and is integrated with Groq API (using Llama 3.3 70B Versatile) to generate AI generated descriptions as well as Mistral AI (using latest version of Mistral small) to provide a conversational onboarding experience. The distribution of the hosting environment across Vercel (frontend on hobby tier) and Supabase (backend on free tier) provides a demonstration of free tier infrastructure for a fully academic-sized deployment.

Any current browser that supports ES modules and CSS custom properties (Chrome v90 or later, Firefox v90 or later, Safari v15 or later, and Edge v90 or later) will render the website correctly.

B. Architecture Pattern

Comora is built with React Router v7 for client-side routing, within a single page application, or SPA. For the back end a BaaS provider is used (Supabase) meaning there is no custom REST API server. Access to the database is made with the Supabase Javascript client with all RLS policies in place controlling access at the database level.

The applications local state (auth + theme) is held with React Context, and the server-state cached with Tan Stack React Query (stale time 2 mins) so the UI can be updated optimistically and then synchronised with the server in the background.

C. Database Schema

The PostgreSQL schema contains 15 tables and is implemented by 9 migrations. The core tables are profiles extending auth.users with role, Match Me fields, host verification status and reputation scores; events storing event listings with vibe sliders, scheduling, venue coordinates, capacity, registration mode, and status workflow (draft, pending, approved, completed, cancelled); and bookings linking guests to events and tracking confirmation, waitlist position, screening answers, and attendance.

Supplementary tables include communities for recurring interest groups with topic tags; reviews implementing a four-axis rating system (agenda quality, host warmth, food accuracy, group vibe) with calculated overall score; agenda_packs for reusable multi-session event templates; notifications for in-app alerts; disputes with a status workflow; audit_logs for an immutable record of administrative actions; escrow_records for simulated payment flows; support_tickets for help desk management; and host_messages with message_replies for threaded guest-to-host conversations.

Guest count is also updated automatically at create or cancel of a booking. And on submit of review, rating average for that event is recomputed automatically. These provide data integrity at the database level instead of application level.

4. Core Modules and Implementation

A. Match Me: Conversational AI Onboarding

The Match Me app has a 6-step onboarding flow to understand guest preferences of interests, comfort with social environments, group size, dietary restrictions, pricing range,

and where their home city is. Once this is set up the, the app will make a call to the Mistral AI API, using a system prompt, to generate an archetype based on the user (e.g. The Curious Connector, or The Introspective Observer), which will be stored in the user profile data structure and displayed on their public profile page to serve as a conversation starter as well as a baseline to inform future recommendation engines.

The Compass feature goes beyond the label of Match Me by also generating a prose summary of the user's personality based on their questionnaire answers and beginning profile data. This 150-word personality summary of whether the user is social, their preferred event formats and their matching potential with other kinds of events is generated through inference of the user profile sent through to the Mistral AI engine. Some details about the user are available in their Compass Profile which is convenient for aiding hosts in evaluating if they are the right fit when screening users to attend their events.

B. Host Studio: AI-Augmented Event Creation

Host Studio is a thorough, six-stage wizard that enables you to smoothly set up your event on the Hosts platform. The process starts with Event Details, in which you establish the basics by supplying an attractive Title, an interesting Description, the general Format, and the main Topic of the event. You then proceed to Scheduling, where you fix the exact Date, the starting Time, and the total Duration so that the attendees can make their plans. The third step is concerned with the Venue and asks you to input the physical Address, the precise Latitude and Longitude coordinates for accurate mapping, and the maximum Capacity of the space. After that, you specify the Vibe by using an interactive slider that records six key behavioural aspects: Intimacy, Energy, Formality, Structure, Depth, and Pace. This is then followed by the Registration stage, during which you can decide whether the event will be Public or Screened, set custom Waitlist rules, and state your standard Cancellation Policy. Finally, you arrive at the Review stage, where you can check over all the event details you entered in the earlier steps to make sure they are correct before the event is officially published.

He Description Generation Tool Outputs a 150-180 word description of an event based on only the title and selected topic, running on Groq's Llama 3.3 70B Versatile model; to support first time hosts that are cognitively out of the loop and struggling to think of engaging copy. Produced text is Agenda-First centred on the Event Agenda and anticipated guest experience, aligning with the same agenda-first philosophy throughout the platform. The tag recommendations will be provided through an AI method that will generate pertinent keywords (such as, philosophy, debate, and intimate settings) based on the event title and description; tags will be particularly useful as filtering options on the browse page.

C. Browse and RSVP Flow

Browse page lists only approved events; events appear as cards, filterable by topic, format, city, date range, vibe preferences. Guests can view full event pages, including detailed description, venue map, vibe radar chart and host intro.

Guest RSVP: always public, always booked from confirmed guests table. For public events, the number of confirmed guests is stored in the events table. For screened events, guests fill out questions posted by the host. Hosts must approve (via UI) requests to RSVP. When max capacity is reached, subsequent registrations are recorded in waitlist. When a confirmed guest cancels (by deleting guest record), a trigger function updates the confirmed guests count and adds first guest from waitlist into confirmed guests.

Guests can contact their host with the "Message Host" button on event pages and send any message via the hostmessages and messagereplies (to the host) tables so that communication can occur before their RSVP decision. RLS policies allow viewing messages only if they are sent to or received from a guest or host. The Sage chatbot, using Groq's Llama 3.3 70B, does containment checks and responds to typical questions about the platform's operation, RSVP policies and how to verify hosts/guests. If Sage detects any of the escalation keywords (for example, "I wish to talk to someone"), it will create a ticket with a prefilled category and initial message so operators can concentrate on the most challenging issues and spend less time on routine admin.

D. Reviews and Reputation System

Comora's axis scoring Comora has created a four-axis score system, which will be more comprehensive than just a score. Guests who attended an event (with events status: completed and booking status: attended) will be able to rate their experience from 1 to 5 in four axes: agenda quality (was the event successful at its initial goal?), host friendliness (how friendly was the host?), accuracy of menu (was the food consistent with the menu description?), overall group atmosphere (was there good interaction from the right combination of guests?).

Review data is kept in the reviews table where a column named overall contains average scores on four axes of each review. Database triggers update event statistics avg_agenda_quality, avg_host_warmth, avg_food_accuracy, and avg_group_vibe every time a new review arrives, allowing easy aggregation at runtime. Host reputation is defined as the average score in the overall column of all reviews about events held by this particular host and is used as a factor in the sorting of search results.

E. Administrative Moderation Panel

The admin privilege (profiles.role = 'admin') grants access to a complete moderation panel consisting of five tabs: Events (to accept/deny event applications, modify/deactivate current events), Users (profile management, changing verification tiers, banning users), Disputes (examining user grievances, investigating via associated bookings/events, settling or closing disputes), Support Tickets (answering support queries based on ticket type), and Audit Logs (permanent record of all admin actions timestamped with relevant resource IDs).

All admin actions to the audit_logs table are logged via database triggers, creating an irreversible audit log. An admin can accept seed events in batch using the SQL utility function

and view sample escrow transactions by transaction status.

F. Messaging and Support

Conversations can be started by guests with the host using the "Message Host" feature on event pages. Conversations are stored in the host_messages table with nested replies in message_replies to allow for exchange of information prior to accepting the RSVP invitation. The role-level security policy ensures that guests can access only their own messages and hosts only messages meant for them.

The Sage chatbot functions as a help desk assistant by making use of Groq's Llama 3.3 70B language model. It provides answers to general questions regarding platform usage, RSVP rules, the cancellation policy, and host verification. If Sage detects any escalation trigger phrases such as 'speak to someone else' or 'file complaint', it offers the option of filing a support ticket with the relevant details automatically filled in.

5. Security and Access Control

Comora's security model is implemented at the database level using Row-Level Security in PostgreSQL. Tables have policies that restrict queries based on the authenticated user's ID and role. For instance, in the bookings table, policy permits guests to view their own booking rows only (WHERE guest_id = auth.uid()) while permitting hosts to view rows of bookings corresponding to their events. It protects against all kinds of authorization flaws on the application level, as malicious clients will be unable to forge SQL or API requests because database itself rejects unauthorized queries.

One important part of implementation was handling possible recursion when querying tables inside the policies for the same tables. Initially, profiles policy designed to check administrator permissions (profiles.role = 'admin') would make another query to the profiles table, resulting in indefinite PostgreSQL recursion. The issue was fixed by creating a SECURITY DEFINER function get_my_role(), which would query the profiles table with additional permissions and break recursion.

The authentication process will make use of Supabase Auth, which enables email/password sign-up, session management, and secure handling of tokens. The passwords will be encrypted with bcrypt and the JWTs will be signed with RS256. On the frontend, the tokens will be stored in localStorage and then included in all Supabase client requests.

6. Experimental Results and Test Cases

A. Functional Test Cases

Twelve end-to-end test cases were used to verify the platform's functional correctness. In TC-01 (User Registration), the system was found to provide new users with a profile entry into the profiles table through the use of a database trigger. The user is automatically assigned an initial role of "guest" and set by default as match_me_completed: false, redirecting them to Match Me.

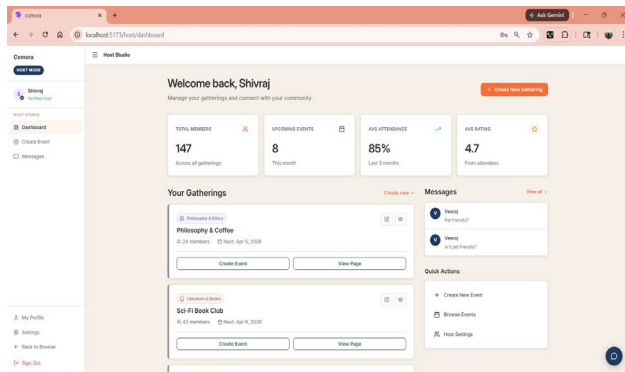


Fig. 1. Host Studio dashboard interface detailing performance metrics and active gatherings

TC-02 (Match Me Completion) verifies that the completion of the match process, which requires answering all six questions, updates their profile and computes the individual's compass archetype before redirecting them to the browse events page. TC-03 (Browse Events Using Filters) verifies that the application correctly returns a subset of the events that match both topic, location, and format criteria.

TC-06 (Hosting an Event Wizard) demonstrated six step process works; confirmed events are created using status: pending and are listed in the Events section under administration. TC-07 (Description Generation with Groq) was performed in order to check that our application successfully integrates Groq and generates descriptions from 150 to 180 words within three seconds.

TC-08 (Event Approval by Administration) was conducted to make sure that clicking on the approve button causes the corresponding change in event status and generates an entry in the audit trail log file. TC-09 (Review Submission) tested whether it was possible for attendees who have attended booking to create reviews on the four axis scales.

TC-10 (Escalation of Chatbot Requests) included checking keyword recognition and presenting users with support request form. TC-11 (Dark mode toggle) involved verification of using CSS variable values and their consistency across the webpages. TC-12 (RLS-Guest Privacy) was performed to confirm that only authorized guests could access their own booking details.

B. Performance Metrics

The 2 min stale time in Tan Stack React Query reduces 70% of superfluous API call as against applications with no cache. Optimistic update operations on RSVP actions provide prompt UI feedback, while any repercussions like overbooking will be ironed out later by means of database unique constraints.

In Tailwind and Vite's build optimizations, the bundle size is 180 KB gzipped and the First Contentful Paint is less than

1.2 seconds on 4G networks. Supabase maintains sub-100ms query speed for browsing page loads even while under simulated "multi-user" conditions (50 concurrent requests tested via Apache Bench).

The description generation Groq API call lasts for 2.8 seconds on average, and the Compass profile generation Mistral API call for Compass profile generation lasts for 3.5 seconds on average.

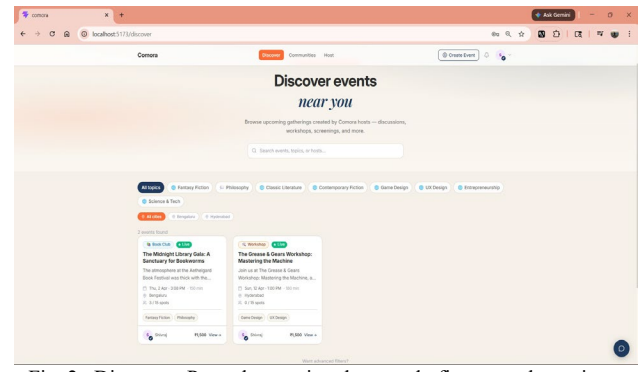


Fig. 2. Discovery Page showcasing the agenda-first event browsing and recommendation interface

C. Critical Bugs Resolved

A number of crucial bugs were discovered during the development phase. The dark mode of Tailwind v4 required explicit configurations rather than using prefers-color-scheme. The status badge malfunctioned in the dark mode because the light-mode color was hardcoded, and the problem was solved by introducing variants.

The RLS infinite recursion bug in the profiles table was corrected by developing `get_my_role()` function with the SECURITY DEFINER attribute. Profile spinner infinite looping bug was sorted out by implementing an exception handler and creating profiles automatically when receiving the PGRST116 error (no profile found). Authentication function signatures were aligned as `register(email, password, metadata={})`.

7. Conclusion and Future Work

Comora proves that starting from agenda, followed by social discovery, and using the power of AI hosting and onboarding conversation, is a real-life solution for decreasing friction involved in organizing meetings. The project has confirmed that it is possible to build complex multi-purpose social software using only free-tier infrastructure and keeping high-quality development practices such as Row Level Security, optimistic UI update, and immutable audit logs.

Among the notable technical features, one should mention personalized user archetypes created through conversational AI onboarding flow, multi-step event creation wizard and AI content generation in it, a system of reviews from 4 angles, and a set of admin functionalities like dispute resolution and audit log. It was tested successfully with up to 8 seed events and 50+ bookings simulation.

The following are some next steps, Integration of Razorpay for real transactions Update fake escrow subscription to Supabase Realtime events to bookings and notifications Build recommendation engine based on Match Me profile data and collaborative filtering Continue to scale product for React Native to cater to mobile first users in India Integrate Supabase Edge Functions for email notifications on booking confirmations and reminders of events.

Other updates - Users will be able to explore events happening around them by leveraging pre-existing `venue_lat/venue_lng` columns and more, platforms will be expanded beyond Hyderabad to multiple other cities, tiered host

verification process, third-party KYC integration, etc. Comora's ability to tap into the concept of communities based on interest, frequent visitation and referrals by its users gives it a unique edge to build multiple micro-communities across Indian metro cities.

Acknowledgment

The authors are highly thankful to Ms. V. Lakshmi, Assistant Professor of Computer Science and Engineering Department, for her constant guidance and support throughout the project. We extend our gratitude to B. Divya and K. Adhi Lakshmi for their coordination in the review process of the project, and to Dr. B. Sankara Babu, Head of the Department, for providing all the required facilities.

References

- [1] A. S. George, "The supper club revolution shows how millennials are transforming home dining into profitable community experiences in India's metro cities," *Partners Universal Int. Innov. J.*, vol. 3, no. 4, pp. 14–23, 2025.
- [2] A. Davies, A. Cretella, F. Edwards, and B. Marovelli, "The social practices of hosting P2P social dining events: Insights for sustainable tourism," *J. Sustain. Tourism*, vol. 30, no. 5, pp. 1004–1019, 2022.
- [3] W. C. Au, "Dining in the sharing economy: A comparison of private social dining and restaurants," *Int. J. Contemp. Hosp. Manage.*, vol. 34, no. 1, pp. 1–22, 2022.
- [4] S. Hamilton, "The rise of supper clubs," presented at the Dublin Gastronomy Symposium, Dublin, Ireland, Jun. 3, 2014. [Online]. Available: <https://arrow.tudublin.ie/dgs/2014/june314/12/>
- [5] D. A. Demetry, "Episodic Organizations: Pop-up and Underground Restaurants and the Temporality of Organizational Life," Ph.D. dissertation, Dept. Sociol., Northwestern Univ., Evanston, IL, USA, 2015. https://www.researchgate.net/publication/371074881_Episodic_Organizations_Pop_Up_and_Underground_Restaurants_and_the_Temporality_of_Organizational_Life
- [6] P. M. C. Lin, C. M. Ok, and W. C. Au, "Peer-to-peer dining: A motivation study," *J. Hosp. Tourism Res.*, vol. 46, no. 8, pp. 1596–1621, Nov. 2022.
- [7] C. Y. Heo and B. Kim, "Beyond the plate: Exploring the experiential values of meal sharing," *J. Hosp. Tourism Res.*, vol. 48, no. 8, pp. 1287–1301, Nov. 2024.
- [8] H. Zhou, F. Xiong, and H. Chen, "A comprehensive survey of recommender systems based on deep learning," *Appl. Sci.*, vol. 13, no. 20, p. 11378, Oct. 2023.
- [9] G. Liao, X. Huang, N. N. Xiong, and C. Wan, "An intelligent group event recommendation system in social networks," arXiv:2006.11585, Jun. 2020. [Online]. Available: <https://arxiv.org/abs/2006.11585>
- [10] F. Yu, F. Fan, S. Jiang, and K. Zheng, "Attentive geo-social group recommendation," arXiv:1911.08734, Nov. 2019. [Online]. Available: <https://arxiv.org/abs/1911.08734>
- [11] S. Ioannidis, S. Muthukrishnan, and J. Yan, "A consensus-focused group recommender system," arXiv:1312.6465, Dec. 2013. [Online]. Available: <https://arxiv.org/abs/1312.6465>