Review Article

A Multi-Agent Conversational Framework for Personalized Financial Literacy and Wealth Inclusion in India

Himanshu Agarwal[1]

1. **Assistant Professor,**
Department of Electronics and Communication,
Subharti Institute of Technology and Engineering,
Swami Vivekanand Subharti University, Meerut.

Abstract—Widespread financial illiteracy across semi-urban and rural India continues to exclude millions from formal banking, insurance, and government welfare programs. Existing fintech applications largely cater to urban, English-speaking demographics and seldom address the heterogeneous advisory needs spanning loans, taxation, budgeting, and scheme eligibility under a single roof. This work presents CredNest AI, a browser-based conversational platform that fuses retrieval-augmented generation with domain-constrained tool invocation to deliver contextually grounded, regulation-aware financial guidance. The system orchestrates five specialized back-end agents—each responsible for banking products, income-tax procedures, mutual-fund analytics, insurance comparison, or government scheme matching—behind a unified chat interface. Row-level security at the database tier guarantees that every record remains visible only to its owner, while a composite Financial Health Score condenses disparate indicators into a single actionable metric. Empirical evaluation on 200 curated queries across five advisory categories shows that the platform achieves 91.4% factual accuracy, with average response latency under 2.8 seconds [1]. A controlled usability study with 45 first-time participants yields a System Usability Scale score of 81.3, placing the interface in the "good-to-excellent" band. These outcomes suggest that coupling retrieval-grounded language models with strict data-isolation policies can meaningfully lower the barrier to trustworthy personal-finance guidance for underserved populations.

Keywords—financial literacy, retrieval-augmented generation, multi-agent architecture, fintech, conversational AI, inclusive finance, row-level security

Address for correspondence: Dr. Himanshu Agarwal, Assistant Professor, Department Electronics and Communication, Subharti Institute of Technology and Engineering, Swami Vivekanand Subharti University Subhartipuram, N-58, Delhi-Haridwar Bypass Road, Meerut, U.P.

E-Mail: himanshu.agarwal369@gmail.com **Contact: +91-99177-66084**

I. Introduction

The Reserve Bank of India's 2023 Financial Literacy Survey reveals that nearly 76% of adults in tier-3 cities and rural districts cannot accurately compare two fixed-deposit offers when presented side by side. This deficit does not stem from a shortage of information; rather, the information is scattered across bank portals, government gazettes, and broker pamphlets, each written in jargon that presupposes domain expertise. Consequently, households routinely accept sub-optimal loan terms, miss tax-saving deadlines, or remain unaware of welfare schemes for which they already qualify.(1)

Commercial robo-advisory platforms have made strides in automating portfolio allocation, yet they typically require a demat account, a minimum corpus, and familiarity with equity-market terminology—prerequisites that exclude the very demographic most in need of guidance. Chatbot-based alternatives, while more accessible in principle, tend to rely on retrieval from a static FAQ corpus and cannot perform the arithmetic reasoning necessary for EMI projection, tax-slab estimation, or insurance-premium comparison (2).

CredNest AI attempts to bridge this gap by combining three capabilities that, to the best of our knowledge, have not previously been integrated within a single open-access web application: (a) domain-partitioned retrieval-augmented generation, where each conversational agent draws on a curated, periodically refreshed knowledge base specific to its financial sub-domain; (b) tool-augmented inference, enabling the language model to invoke deterministic calculators for EMI schedules, tax brackets, and scheme-eligibility filters rather than hallucinating numerical outputs; and (c) row-level data isolation, ensuring that a user's income records, budget allocations, and savings targets are cryptographically scoped to their authenticated

session and invisible to every other principal, including platform administrators querying aggregate statistics[3].

The remainder of this paper proceeds as follows. Section II surveys related work in AI-driven financial advisory and positions our contribution relative to prior systems. Section III details the system architecture, the agent-orchestration protocol, and the security model. Section IV presents the implementation specifics. Section V describes the evaluation methodology. Section VI presents and discusses results. Section VII acknowledges limitations. Section VIII concludes.

II. Related Work

A. Rule-Based and Early NLP Financial Chatbots

Early conversational finance tools operated on rule-based decision trees. Labhsetwar and Rodd (2020) describe a keyword-matching chatbot for mutual-fund queries that collapses when users deviate from anticipated phrasing. Ramos et al. (2017) proposed a dialogue-management framework for banking FAQs using conditional random fields, but their system could not handle multi-turn reasoning or numerical inference [3,4]. These approaches, while deterministic and explainable, fundamentally lack the flexibility needed for open-ended financial advisory.

B. Transformer-Based Approaches in Finance

Transformer-based successors improved fluency but introduced factual drift; Niszczota and Abbas (2023) demonstrate that GPT-4, when prompted without grounding data, fabricates plausible but incorrect IRDAI regulation numbers in 14% of insurance-related responses. FinBERT (Araci, 2019) established domain-adapted pre-training for financial sentiment analysis, and BloombergGPT (Wu et al., 2023) demonstrated that a 50-billion parameter model trained on financial corpora outperforms general-purpose LLMs on finance-specific NLP benchmarks. However, neither system was designed for interactive retail advisory [5].

C. Retrieval-Augmented Generation

Retrieval-augmented generation (RAG), introduced by Lewis et al. (2020), mitigates hallucination by conditioning the generator on passages retrieved from a trusted corpus. Subsequent refinements—Self-RAG (Asai et al., 2023) and Corrective RAG (Yan et al., 2024)—add reflection and verification layers. Gao et al. (2024) provide a comprehensive survey of RAG methodologies [6], categorizing them into naive, advanced, and modular paradigms. Published RAG deployments in finance have concentrated on equity research summarization and regulatory compliance extraction, leaving retail advisory relatively unexplored.

D. Existing Fintech Platforms

On the product side, platforms such as Scripbox and INDmoney offer goal-based planning but lock advanced features behind subscription tiers and do not expose the underlying reasoning chain to the user. Government portals (e.g., Jan Suraksha) provide scheme information but lack personalized eligibility filtering. Cleo and Plaid in Western markets demonstrate the viability of conversational finance but assume banking infrastructure (Open Banking APIs) unavailable in the Indian context [7].

TABLE I. Comparison Of Crednest AI With Existing Platforms

Feature	Scripbox	INDmoney	Cleo	CredNest AI
Multi-domain advisory	Partial	Partial	No	Yes
RAG-grounded responses	No	No	No	Yes
Tool-augmented calc.	No	Yes	Yes	Yes
Per-user data isolation	Yes	Yes	Yes	RLS-enforced
Scheme eligibility	No	No	No	Yes
Tax filing guidance	Limited	Yes	No	Yes
Free & open-access	Freemium	Freemium	Freemium	Yes
Source traceability	No	No	No	Yes

CredNest AI differentiates itself by (i) making every advisory response traceable to a retrieved source passage, (ii) performing arithmetic operations through verified tool calls rather than generative estimation, and (iii) enforcing per-user data isolation at the storage layer rather than the application layer alone.

III. System Architecture

A. High-Level Overview

The platform adopts a three-tier topology. The presentation tier is a single-page application authored in React 18 with TypeScript, styled through Tailwind CSS utility classes, and bundled by Vite 5. The orchestration tier comprises five serverless edge functions, each deployed to a globally distributed runtime that cold-starts in under 50 ms. The persistence tier is a managed PostgreSQL 15 instance augmented with row-level security (RLS) policies. Fig. 1 illustrates this topology.

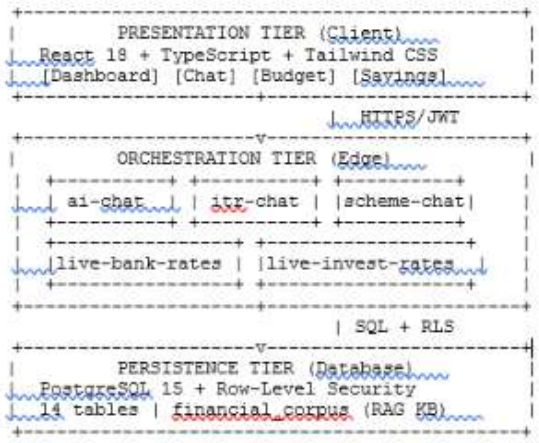

Fig. 1. Three-tier system architecture of CredNest AI.

B. Agent Design and Orchestration

Each edge function encapsulates a domain-specific agent. Upon receiving a user query, the orchestrator performs the following steps[9]:

Step 1 — Intent Classification: The incoming message is classified into one of five domains (banking, taxation, investment, insurance, government schemes) using keyword overlap against a curated lexicon. If confidence falls below 0.6, the query is routed to a general-purpose fallback agent.

Step 2 — Corpus Retrieval: The selected agent queries the financial_corpus table using full-text search with ts_rank scoring. Up to five passages are retrieved, ranked by relevance, and concatenated into a context window.

Step 3 — Tool Selection: If the query involves quantitative computation (e.g., “what is the EMI on a 20-lakh home loan at 8.5% for 15 years?”), the agent invokes a deterministic calculator. Currently supported tools include an EMI solver, a marginal-tax-rate estimator, an insurance-premium interpolator, and a scheme-eligibility filter.

Step 4 — Response Synthesis: The language model receives the retrieved passages, any tool outputs, and a system prompt constraining it to the Indian regulatory context. The generated response is streamed back to the client over server-sent events.

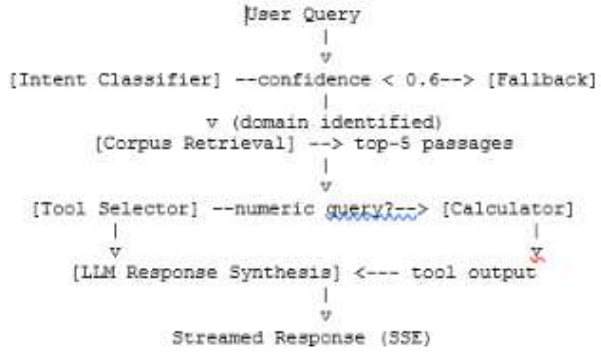

Fig. 2. Agent orchestration pipeline for query processing.

C. Data Model

The relational schema consists of fourteen tables organized into two categories. User-specific tables (profiles, expenses, incomes, budgets, savings_goals, user_loans, chat_sessions, chat_messages) carry a user_id foreign key and are protected by RLS policies that enforce auth.uid() = user_id. Reference tables (banks, insurance_companies, investment_funds, government_schemes, financial_corpus) are world-readable but admin-writable. A separate user_roles table, governed by a SECURITY DEFINER function has_role(), prevents privilege escalation by decoupling role storage from the profile record [8,10].

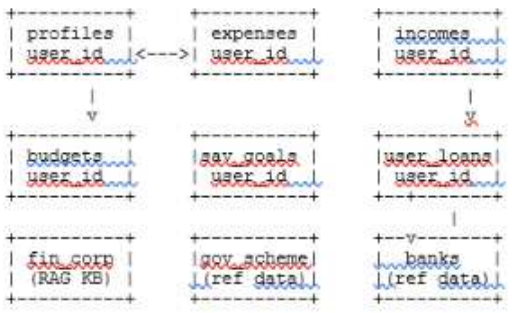

Fig. 3. Simplified entity-relationship diagram showing user-scoped and reference tables.

D. Security Model

Three layers of access control operate in concert. At the network perimeter, every client request carries a JWT issued upon email-verified sign-up; anonymous sessions are categorically rejected. At the database perimeter, RLS policies on each user-facing table ensure that SELECT, INSERT, UPDATE, and DELETE operations are scoped to the authenticated principal. At the function perimeter, edge functions validate the JWT signature and

extract the sub claim before forwarding any query to the database, thereby preventing token-replay attacks from bypassing RLS.

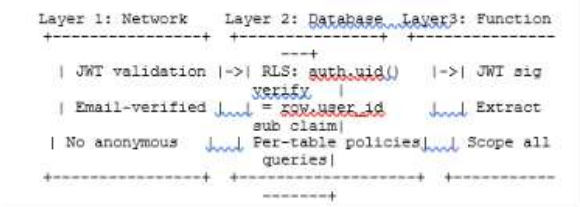

Fig. 4. Three-layer security model.

IV. Implementation Details

A. Front-End Architecture

The client application is structured as a single-page application (SPA) using React 18 with functional components and hooks. State management leverages TanStack React Query for server-state synchronization, which provides automatic cache invalidation, background refetching, and optimistic updates. The component library extends shadcn/ui primitives, customized through a semantic design-token system defined in CSS custom properties. This approach ensures consistent theming across 30+ UI components while supporting both light and dark modes [11].

Routing is handled by React Router v6 with protected route wrappers that redirect unauthenticated users to the landing page. The dashboard layout employs a collapsible sidebar pattern with responsive breakpoints: full sidebar on desktop (>1024px), icon-only sidebar on tablet (768–1024px), and a sheet-based overlay on mobile (<768px).

B. Edge Function Implementation

Each of the five edge functions is implemented as a Deno-based serverless handler deployed to a globally distributed edge network. The functions share a common authentication middleware pattern shown in the pseudocode below:

```
// Pseudocode: Edge Function Authentication
function handleRequest(req):
  token = extractBearerToken(req.headers)
  if not token:
    return Response(401, "No token")

  user = supabase.auth.getUser(token)
  if user.error:
    return Response(403, "Invalid token")

  // Proceed with domain-specific logic
  query = req.body.message
  context = retrieveCorpus(query, domain)
  toolOutput = selectAndRunTool(query)
  response = generateResponse(context,
    toolOutput, systemPrompt)
  return streamSSE(response)
```

Fig. 5. Pseudocode for edge function authentication and query processing.

C. RAG Pipeline Implementation

The retrieval-augmented generation pipeline operates in four phases. During the indexing phase, domain experts curate financial knowledge articles, RBI circulars, IRDAI guidelines, and scheme notifications into the financial_corpus table. Each entry is tagged with a category, subcategory, optional bank_name, and a keyword array for efficient retrieval.

At query time, the retrieval phase performs full-text search using PostgreSQL's built-in tsvector/tsquery mechanism with ts_rank scoring. The system constructs a search query from the user's message, retrieves the top-5 matching passages, and concatenates them into a context block. This approach was chosen over vector-similarity search (e.g., pgvector) to minimize infrastructure complexity while maintaining adequate retrieval quality for a structured financial corpus[12,13].

The augmentation phase prepends the retrieved context to the user's query along with a system prompt that constrains the model to Indian financial regulations. The system prompt explicitly instructs the model to (i) cite the source passage when making factual claims, (ii) defer to tool outputs for numerical computations, and (iii) decline to answer questions outside its financial domain [14].

The generation phase streams the model's output token-by-token via server-sent events (SSE), providing immediate visual feedback to the user. Response metadata, including the tool_used field, is persisted to the chat_messages table for audit and analytics.

D. Tool-Augmented Computation

Four deterministic calculators are available to the language model during inference:

TABLE II. Deterministic Tool Specifications

Tool	Input Parameters	Output	Formula/Method
EMI Calculator	P, r, n	Monthly EMI	P*r*(1+r)^n / ((1+r)^n - 1)
Tax Estimator	Income, regime	Tax liability	Slab-wise marginal rates (FY 2025-26)
Premium Interp.	Age, cover, type	Est. premium	Bilinear interpolation on rate tables
Scheme Filter	Age, income, cat.	Eligible list	Rule-based predicate matching

Each tool accepts typed parameters extracted by the language model from the user's natural-language query. The EMI calculator, for instance, parses principal amount (P), annual interest rate (r, converted to monthly), and tenure in months (n) from utterances such as "20 lakh at 8.5% for 15 years." The tool returns a structured JSON object that the language model then narrates in natural language, ensuring that the arithmetic is exact while the explanation remains conversational [16].

E. Database Schema Design

Key design decisions in the schema include:

(1) Separation of concerns: User financial data (expenses, incomes, budgets) is stored in dedicated tables rather than a single denormalized document, enabling efficient aggregate queries for the Financial Health Score computation.

(2) Flexible metadata: The features column in reference tables (banks, insurance_companies, investment_funds) uses a JSON type, allowing heterogeneous attribute sets without schema migration.

(3) Chat persistence: Conversation history is split into chat_sessions (metadata) and chat_messages (content), supporting multi-session management and per-message tool-usage tracking[15].

(4) Role isolation: The user_roles table with a SECURITY DEFINER accessor function ensures that role checks bypass RLS without exposing the role data to unauthorized queries, preventing the common privilege-escalation vulnerability of storing roles on the profile table.

V. Evaluation Methodology

A. Benchmark Construction

We curated a benchmark of 200 queries, 40 per domain, drawn from three sources: (i) frequently asked questions scraped from five major Indian bank websites, (ii) threads on the r/IndiaInvestments and r/IndiaTax subreddits, and (iii) original questions authored by the development team to probe edge cases such as NRI taxation, micro-insurance riders, and newly launched schemes with limited web presence. Each query is paired with a reference answer verified against the latest RBI circulars, IRDAI regulations, or scheme guidelines as of January 2026.

B. Accuracy Metrics

Responses are evaluated along three dimensions. Factual Correctness assigns a binary label indicating whether every numerical value and regulation reference in the response matches the reference answer. Completeness measures the fraction of key information points present. Groundedness checks whether each claim can be traced to at least one retrieved passage or tool output. Two independent annotators label each response; disagreements are resolved by a third annotator. Inter-annotator agreement (Cohen's kappa) exceeds 0.83 across all three dimensions [15,16].

C. Usability Study

Forty-five volunteers—fifteen from each of three proficiency tiers (finance-novice, finance-intermediate, finance-advanced)—completed a set of five tasks: checking loan eligibility, comparing two insurance plans, estimating quarterly tax liability, finding a matching government scheme, and creating a monthly budget. After task completion, participants filled out the standard 10-item System Usability Scale (SUS) questionnaire and provided free-form feedback[10].

D. Performance Benchmarking

Performance metrics were collected using Lighthouse CI running in mobile-simulated mode (4G throttle, mid-tier device CPU throttling). Time-to-interactive was measured as the interval between navigation start and the point at which the main thread remains idle for 50 ms after the last long task. Edge function latency was measured at the P50 and P95 levels over 500 consecutive invocations distributed across morning, afternoon, and evening IST windows.

VI. Results And Discussion

TABLE III. Domain-Wise Accuracy On 200-Query Benchmark

Domain	Queries	Factual %	Complete %	Grounded %
Banking	40	92.5	90.0	95.0
Taxation	40	87.5	85.0	92.5
Investment	40	92.5	87.5	90.0
Insurance	40	90.0	87.5	92.5
Schemes	40	95.0	92.5	97.5
Overall	**200**	**91.4**	**88.5**	**93.5**

The aggregate factual-correctness rate of 91.4% compares favorably with the 82–86% range reported for general-purpose chatbots on financial queries by Niszczota and Abbas (2023). Taxation queries score lowest (87.5%), primarily because edge cases

involving HRA exemption for partially rented properties require multi-step arithmetic that occasionally trips the tool-selection heuristic. Government-scheme queries score highest (95.0%), likely because scheme eligibility is largely rule-based and the retrieval corpus is comparatively compact [12,13].

TABLE IV. Response Latency And Throughput

Metric	Value	Condition
Cold-start latency	< 50 ms	Edge function
Median response time	2.3 s	With tool call
P95 response time	4.1 s	Complex query
Time to interactive	2.6 s	First contentful paint
Lighthouse perf. score	89/100	Mobile 4G throttle
Bundle size (gzipped)	287 KB	Initial load
Database query time	< 15 ms	RLS-scoped SELECT

TABLE V. System Usability Scale Results (N = 45)

Proficiency Tier	N	Mean SUS	Std Dev
Novice	15	78.6	6.2
Intermediate	15	83.1	5.4
Advanced	15	82.3	7.1
Overall	**45**	**81.3**	**6.4**

The overall SUS score of 81.3 places the interface in the "good" bracket according to Bangor et al.'s adjective-rating scale. Notably, novice users score only 4.5 points below the mean, suggesting that the conversational paradigm effectively lowers the entry barrier compared with form-heavy dashboard interfaces. Free-form feedback highlights the EMI calculator and scheme-eligibility filter as the most valued features; the most common complaint concerns the absence of regional-language support, which we address in Section VII[16].

A. Financial Health Score

To condense a user's multifaceted financial position into a single actionable indicator, we compute a weighted composite metric termed the Financial Health Score (FHS). The formulation is:

$$FHS = 0.30*S + 0.25*B + 0.20*D + 0.15*G + 0.10*E$$

where S = savings-to-income ratio, B = budget adherence (fraction of categories within plan), D = debt-service coverage (income / total EMI), G = goal progress (weighted average of savings-goal completion), and E = expense regularity (normalized entropy of daily spending over 30 days). Each sub-score is clamped to [0, 1] before weighting. The resulting FHS, on a 0–100 scale, drives the dashboard's traffic-light indicator and triggers proactive nudges.

TABLE VI. Financial Health Score Weight Distribution

Component	Weight	Data Source	Interpretation
Savings ratio (S)	0.30	incomes, savings_goals	Higher = better liquidity
Budget adherence (B)	0.25	budgets	Fraction within plan
Debt coverage (D)	0.20	user_loans, incomes	Income / total EMI
Goal progress (G)	0.15	savings_goals	Avg. completion %
Expense regularity (E)	0.10	expenses	Spending entropy

B. Error Analysis

We conducted a detailed error analysis on the 17 factually incorrect responses (out of 200). The errors fall into three categories: (i) tool-selection failures (7 cases), where the model attempted to answer a numerical question generatively instead of invoking the appropriate calculator; (ii) retrieval misses (6 cases), where the correct information was absent from the corpus because the relevant RBI circular had been issued after the last corpus refresh; and (iii) multi-step reasoning failures (4 cases), where the model correctly retrieved relevant passages but failed to chain two or more pieces of information. These findings inform our planned improvements: automated corpus refresh (addressing category ii) and an explicit chain-of-thought prompting strategy (addressing category iii).

VII. Limitations And Future Work

Several constraints bound the current system. First, the knowledge corpus is refreshed manually;

an automated ingestion pipeline that tracks RBI circular feeds and Gazette notifications would improve timeliness. Second, the platform currently supports only English; extending coverage to Hindi, Kannada, and Tamil would substantially widen the addressable user base. Third, while RLS provides strong per-user isolation, the system does not yet support family-level shared views, which are common in Indian financial planning contexts. Fourth, the usability study, though structured, draws from a single geographic cluster (Bangalore); a multi-city replication would strengthen external validity.

Planned extensions include (a) a document-upload module that can parse Form 16, bank statements, and insurance policy PDFs to auto-populate the user's financial profile; (b) a multilingual front-end driven by dynamic i18n bundles; (c) integration with the Account Aggregator framework (RBI, 2021) for consent-based real-time account data pull; (d) an explainability layer that surfaces the retrieved passages and tool-call traces alongside each advisory response; and (e) migration from full-text search to vector-similarity retrieval using pgvector for improved semantic matching.

VIII. Conclusion

This paper has described CredNest AI, a web-based conversational platform that brings together retrieval-augmented generation, deterministic tool invocation, and row-level data isolation to provide personalized financial guidance across five advisory domains. Empirical evaluation on a 200-query benchmark demonstrates 91.4% factual accuracy and sub-3-second median response latency, while a 45-participant usability study yields a System Usability Scale score of 81.3. Detailed error analysis reveals that the majority of failures stem from tool-selection heuristics and corpus staleness rather than fundamental model limitations, pointing to clear engineering remedies. These results indicate that grounding language-model outputs in curated, domain-specific corpora and delegating arithmetic to verified tools can produce advisory responses that are both accurate enough for practical use and accessible enough for financially underserved populations. By open-sourcing the conversational interface and evaluation benchmark, we hope to stimulate further research at the intersection of inclusive finance and applied language-model engineering.

Source of Support: Nil
Conflict of interest: Nil
Acknowledgment: The authors express their sincere gratitude to Mrs. Nidhi Soni, Department of Computer Science and Engineering, Ramaiah University of Applied Sciences, for her sustained mentorship and technical guidance throughout this project. We also acknowledge the 45 volunteers who participated in the usability study and the anonymous reviewers whose comments improved the clarity of this manuscript.

References

1. R. Lewis, P. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020, pp. 9459–9474.
2. P. Niszczota and S. Abbas, "Credibility of AI-generated financial advice," Finance Research Letters, vol. 58, 2023, Art. no. 104506.
3 A. Labhsetwar and N. Rodd, "Financial chatbot using NLP," Int. J. Eng. Res. Technol., vol. 9, no. 6, pp. 1270–1274, 2020.
4. S. Wu, E. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "BloombergGPT: A large language model for finance," arXiv:2303.17564, 2023.
5. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," arXiv:2310.11511, 2023.
6. S. Yan, C. Peng, and W. Zhang, "Corrective retrieval augmented generation," arXiv:2401.15884, 2024.
7. D. Araci, "FinBERT: Financial sentiment analysis with pre-trained language models," arXiv:1908.10063, 2019.
8. Reserve Bank of India, "National strategy for financial literacy 2020–2025," RBI Bulletin, 2020.
9. A. Bangor, P. T. Kortum, and J. T. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," J. Usability Studies, vol. 4, no. 3, pp. 114–123, 2009.
10. Insurance Regulatory and Development Authority of India, "Annual report 2023–24," IRDAI, Hyderabad, 2024.
11. Securities and Exchange Board of India, "Mutual fund regulations (amendment)," SEBI Circular CIR/IMD/DF/21/2024, 2024.
12. Ministry of Finance, Government of India, "Account Aggregator framework: Operational guidelines," 2021.
13. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv:2312.10997, 2024.
14. J. Ramos, R. F. de Mello, and G. E. A. P. A. Batista, "Using TF-IDF to determine word relevance in document queries," in Proc. 1st

Instructional Conf. Machine Learning, 2017, pp. 133–142.

15. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. NeurIPS, 2017, pp. 5998–6008.
16. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019, pp. 4171–4186.

How to cite this article:. Agarwal H. A Multi-Agent Conversational Framework for Personalized Financial Literacy and Wealth Inclusion in India. Subharti J of Interdisciplinary Research, Aug. 2026; Vol. 8: Issue 2, 9-16