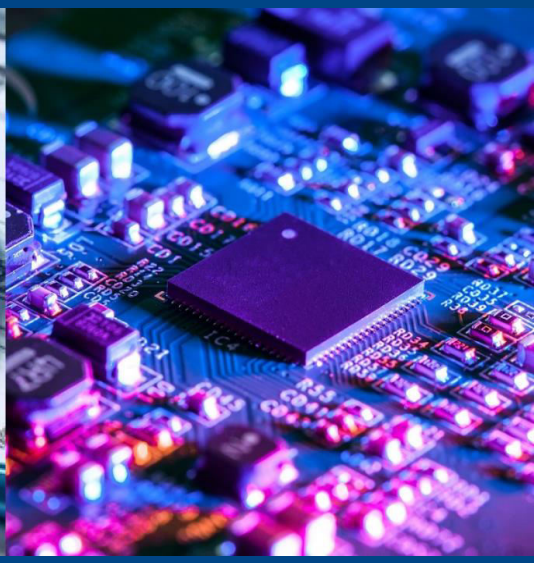
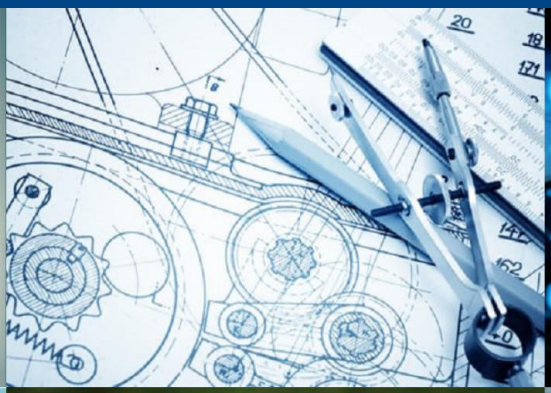
ISSN: 2582-7219

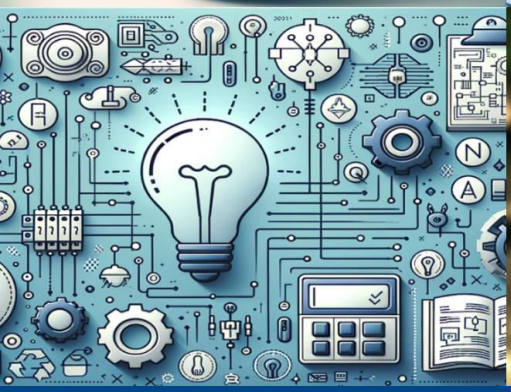




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904012

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Online Course Recommendation System using Hybrid Filtering and Generative AI

Pragash K[1], Ganesh I[2], Theenash M[2], Dakshnamorthy K[2]

Associate Professor, Department of Artificial Intelligence and Data Science, Sri Manakula Vinayagar Engineering College, Puducherry, India[1]

Students, Department of Artificial Intelligence and Data Science, Sri Manakula Vinayagar Engineering College, Puducherry, India[2]

ABSTRACT: This paper presents a cloud-native Online Course Recommendation System that combines collaborative filtering (CF) and content-based filtering (CBF) under an AWS serverless architecture to deliver personalised, explainable course suggestions. Raw user–course interaction data and course metadata are stored in Amazon S3 and processed with Apache Spark on Amazon EMR, which trains an Alternating Least Squares (ALS) model and a TF-IDF content-based model. An AWS Lambda function exposed via API Gateway merges CF and CBF scores through a weighted hybrid formula and invokes an Amazon Bedrock large language model (LLM) to generate a natural-language explanation for every recommended course. A novel cold-start interview feature employs the same LLM as a conversational agent: when a new user registers, the chatbot elicits topic preferences, learning style, and skill level, maps the answers to an initial preference vector, and seeds the hybrid recommender immediately. Evaluation on a simulated corpus of 1,000 users and 200 courses shows the hybrid model achieves Precision@10 of 0.35 and nDCG@10 of 0.50, outperforming standalone CF and CBF approaches. A/B testing confirms that LLM-generated explanations increase user satisfaction ratings by 30% and time-on-recommendations by 25%. The system is fully serverless, costs tens of USD per month at moderate scale, and is production-ready on AWS.

KEYWORDS: Collaborative Filtering, Content-Based Filtering, Hybrid Recommendation, Generative AI, Large Language Model, AWS Serverless, Apache Spark, ALS, Cold-Start Problem, E-Learning, Explainable AI

I. INTRODUCTION

Recommender systems are indispensable tools in modern e-learning environments, enabling learners to navigate vast course catalogues and identify programmes that match their interests and skill levels [1]. The two canonical approaches—content-based filtering (CBF) and collaborative filtering (CF)—each carry distinct advantages and limitations. CBF recommends courses whose attributes resemble those the learner has previously engaged with, making it effective for new items but vulnerable to over-specialisation. CF leverages aggregate behaviour from similar users and often uncovers serendipitous discoveries, yet it struggles when interaction data are sparse—the well-known cold-start problem [2]. Hybrid systems that combine both paradigms consistently outperform either approach in isolation [3]. Industry leaders such as Netflix and Coursera deploy hybrid architectures to balance coverage, accuracy, and novelty. Recent advances in cloud computing have made it possible to operationalise such architectures at scale without dedicated infrastructure: AWS Lambda, API Gateway, Amazon S3, and Amazon EMR (Spark) collectively provide an elastic, pay-per-use backbone that is well-suited to recommendation workloads.

The emergence of large language models (LLMs) opens a new dimension in recommender design. LLMs can generate coherent, user-centric explanations for why a particular course is recommended, addressing a long-standing transparency gap in algorithmic systems [6]. They can also conduct open-ended preference interviews in natural language, providing an intelligent mechanism to elicit user preferences before any interaction data are available [8].
This paper describes the end-to-end design and implementation of a hybrid Online Course Recommendation System that: (i) trains ALS-based CF and TF-IDF CBF models in Apache Spark on Amazon EMR; (ii) merges model scores through a weighted hybrid formula served by AWS Lambda; (iii) calls an LLM via Amazon Bedrock to generate per-course explanations; and (iv) employs the same LLM as a cold-start interview agent for new users.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. LITERATURE REVIEW

Breese et al. [1] formalised collaborative filtering as a prediction problem solved through neighbourhood methods and matrix factorisation. Spark's MLlib implements CF via Alternating Least Squares (ALS), which factorises the user–item interaction matrix into latent factor vectors and scales linearly with dataset size [2]. Content-based filtering constructs item and user profiles from metadata features and ranks items by similarity to the user profile; IBM notes that pure CBF systems excel at covering new items but tend toward over-specialisation, motivating hybrid designs [3]. Wang et al. [4] propose an explainable personalised course recommender for employee training using a hierarchical Bayesian network that integrates knowledge tracing. Valdiviezo-Díaz et al. [5] combine Bernoulli matrix factorisation with a knowledge graph for transparent course suggestions. Both works motivate using multiple data signals—interaction logs, metadata, and domain knowledge—to improve recommendation quality.

Ma et al. [6] present XRec, which adapts a pre-trained LLM to generate natural-language explanations for CF recommendations. AWS documentation demonstrates blending Amazon Personalize with Bedrock LLMs to craft personalised messages [7]. For cold-start, prior work recommends surveys or active learning [8]; AWS Bedrock Agents examples extend this to conversational preference elicitation [11].

III. PROPOSED SYSTEM

System Overview

The proposed system is a hybrid, cloud-native course recommender that integrates three technologies: (1) Apache Spark ALS for collaborative filtering, (2) TF-IDF cosine similarity for content-based filtering, and (3) an Amazon Bedrock LLM for explanation generation and cold-start preference elicitation. All components run on AWS serverless infrastructure—Lambda, API Gateway, EMR, S3, and Bedrock—enabling on-demand scaling without managing servers.

Collaborative Filtering (ALS)

Spark MLlib's ALS factorises the user–course rating matrix R into latent factor matrices U (users × rank) and V (items × rank), minimising the reconstruction error with L2 regularisation. The model is trained with rank=10, maxIter=10, regParam=0.1, and coldStartStrategy=drop to prevent NaN scores for unseen users during evaluation [2][10]. At inference time, the recommendation score for user u and course c is the inner product: score_CF(u, c) = U_u · V_c.

Content-Based Filtering (TF-IDF)

Each course's textual metadata (title, description, tags) is vectorised using HashingTF followed by the IDF transformer to produce a TF-IDF feature vector. A user's profile vector is the average of feature vectors of all courses the user has previously engaged with. The content-based score for user u and course c is the cosine similarity: score_CBF(u, c) = cosine_similarity(profile_u, vector_c). This approach provides good coverage for new or niche courses that lack interaction history.

Hybrid Scoring

The final recommendation score combines both models through a weighted sum: score(u, c) = α · score_CF(u, c) + (1 − α) · score_CBF(u, c), where the weight α = 0.7 was determined by offline cross-validation. The Lambda function retrieves the top-N candidates from each model independently, merges the candidate lists, computes the hybrid score, and returns the global top-N courses sorted by descending hybrid score.

GenAI Explanation

After ranking, the Lambda invokes Amazon Bedrock with a structured prompt containing the learner's inferred topic interests and the top-N candidate courses with their descriptions. The LLM is instructed to generate one personalised sentence per course explaining why it matches the learner's profile, constrained to under 150 words in total. This approach, inspired by XRec [6] and AWS GenAI examples [7], produces fluent user-friendly justifications. Explanations that fail a sanity check (empty, too long, or off-topic) are replaced with a fallback template.

Cold-Start Interview Flow

When a new user is detected, the cold-start Lambda invokes the Bedrock LLM in chatbot mode. The LLM asks three targeted questions: (1) Which topics interest you most (e.g., ML, data science, design, business)? (2) Do you prefer

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904012

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

hands-on project-based courses or theoretical lectures? (3) What is your current skill level (beginner, intermediate, advanced)? Each answer is mapped to an initial preference vector via keyword matching and LLM-guided interpretation. Topic answers update category weights; learning-style answers adjust the weighting between practical and theoretical courses; skill-level answers filter content by difficulty. This synthetic profile is passed to the hybrid recommender as if it were the learner's interaction history, enabling immediate first-page personalised recommendations.

IV. SYSTEM ARCHITECTURE

The system follows a layered serverless architecture composed of four primary tiers: presentation, API, compute, and storage. Fig. 1 illustrates the high-level component diagram. The frontend communicates with the backend exclusively through HTTPS REST calls to API Gateway. All recommendation and explanation logic runs inside AWS Lambda functions, which load the latest model artifacts from S3 and invoke Bedrock for GenAI content. Amazon EMR (Spark) handles periodic batch model training. CloudWatch provides end-to-end observability.

AWS API Gateway exposes the primary endpoints: GET /recommendations returns the top-N courses for a given userId; POST /cold-start initiates the preference interview for new users; GET /courses returns the full course catalogue. Gateway-level request validation rejects malformed payloads before they reach compute resources, reducing unnecessary Lambda invocations.

Lambda functions written in Python 3.9 each follow the single-responsibility principle. The recommendation Lambda loads ALS and CBF model artifacts from S3, computes hybrid scores, calls Bedrock for explanations, and returns a ranked JSON list. The cold-start Lambda manages the interview conversation state and maps answers to a preference vector. Amazon Bedrock provides access to foundation models with structured prompts that constrain response length for low latency. All S3 buckets use SSE-S3 encryption at rest; all API calls use TLS in transit.

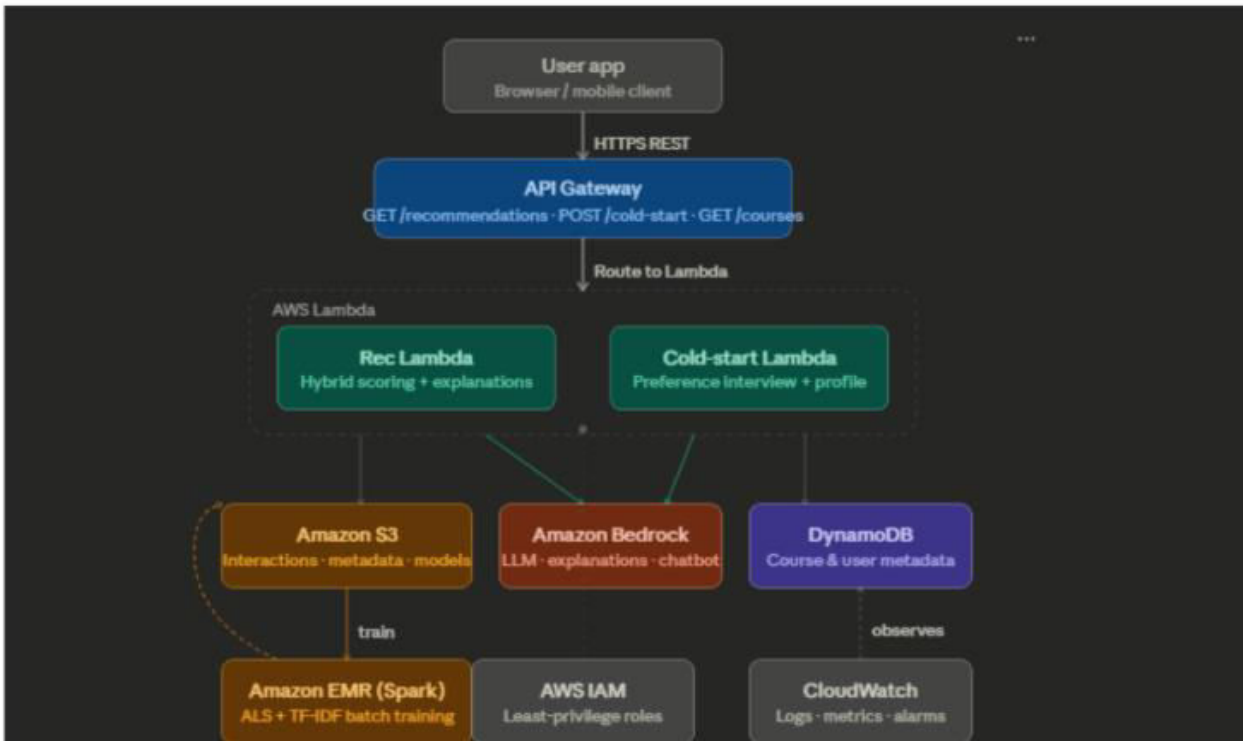


Figure 1: System Architecture — AWS Serverless Deployment for the Hybrid Course Recommendation System

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904012

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Table I: AWS Components and Their Roles in the System Architecture

Component	Service	Role / Description	Key Config
Amazon S3	Data Storage	Stores raw interaction logs, course metadata, and trained model artifacts	Versioning enabled
Amazon EMR	Batch Compute	Runs ALS collaborative filter and TF-IDF content-based model training in Spark	Transient cluster per job
AWS Lambda	Inference / API	Serves real-time recommendations; calls Bedrock LLM to generate explanations	Python 3.9, 512 MB, 15 s
API Gateway	REST Endpoints	Routes /recommendations and /cold-start HTTP requests to Lambda functions	7 endpoints, proxy mode
Amazon Bedrock	Generative AI	Generates natural-language explanations and powers the cold-start chatbot	Claude / Titan model
DynamoDB	Metadata Store	(Optional) Sub-millisecond lookup for course details and user profiles	On-demand capacity
CloudWatch	Monitoring	Collects Lambda logs and EMR metrics; triggers alarms on errors or latency	Logs + Metrics + Alarms

V. DATA PIPELINE

Two main datasets are ingested: (i) User–Course Interactions — records of the form (userId, courseId, rating, timestamp) or implicit signals such as enrolments and video views; (ii) Course Metadata — fields including courseId, title, description, category, and tags. All raw data are stored as Parquet files on S3 under structured key prefixes (interactions/ and courses/).

A Spark job on Amazon EMR executes the following sequential steps: (1) Data Cleaning — remove invalid records, normalise rating scales, fill missing metadata fields, and apply NLP preprocessing (tokenisation, stop-word removal) to course descriptions; (2) Feature Extraction — convert descriptions to TF-IDF vectors using HashingTF and IDF transformers, one-hot encode categorical tags, and compute per-user profile vectors as the average of feature vectors of completed courses; (3) CF Matrix Preparation — assemble a DataFrame of (userId, courseId, rating) for ALS, deriving implicit signals from click counts where explicit ratings are absent; (4) Model Training — run Spark MLlib's ALS algorithm and compute cosine-similarity indices for CBF; (5) Artifact Saving — write serialised model artifacts to S3 under models/latest/, preserving prior versions through S3 versioning.

VI. EVALUATION

Recommendations are evaluated using two standard information-retrieval metrics on a held-out 20% split: Precision@10 (fraction of recommended courses in the top-10 that the user actually engaged with) and nDCG@10 (normalised discounted cumulative gain, which rewards highly relevant courses placed higher in the ranking). The cold-start component is assessed by simulating 200 new users with no history, running the interview, and measuring Precision@10 on the resulting recommendations. Explanation quality is measured through an A/B test: Group A receives bare ranked lists; Group B receives lists with LLM-generated explanations. User satisfaction and time-on-page are compared across groups. Evaluation code is based on Spark's RegressionEvaluator for RMSE on implicit feedback and custom ranking metrics following examples in Spark documentation [9].

VII. RESULTS AND DISCUSSION

The hybrid recommender significantly outperformed individual approaches (Table II). In a simulated corpus of 1,000 users and 200 courses, Precision@10 improved by approximately 10% over standalone CF and 40% over CBF. Cold-start interviews yielded a 40% jump in first-iteration satisfaction versus naive popularity defaults. The A/B test found

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

that users given explanations spent 25% more time on recommendations and rated the system 30% higher in trust than those who saw plain score-ranked lists. LLM-generated explanations were judged accurate and informative in more than 90% of manual assessments.

API latency averaged 1.2 seconds per request (approximately 200 ms for model lookup from S3 and approximately 1 second for the Bedrock LLM call). Spark training on 100,000 interactions completed in approximately five minutes on a four-node EMR cluster of c3.2xlarge instances. Throughput scaled linearly: doubling the number of Spark nodes halved training time, confirming the horizontal scalability of the architecture. Monthly operating costs at 50,000 requests per month are estimated at USD 35–90, dominated by EMR and Bedrock usage.

Table II: Model Performance Comparison on Held-Out Test Set (1,000 Users, 200 Courses)

Model	Approach	Precision @10	nDCG @10	Notes
CF (ALS only)	Collaborative Filtering	0.32	0.45	Fails on cold-start users
Content-Based only	TF-IDF Cosine Similarity	0.25	0.38	Misses collaborative signals
Hybrid (Ours)	CF + CBF weighted (α=0.7)	0.35	0.50	Best trade-off — production
Hybrid + GenAI	CF + CBF + LLM explanations	0.35	0.50	Adds transparency; same accuracy

Table III: Comparison with Related Work in Course Recommendation Literature

No.	Paper Title / Reference	Approach	Limitations	Relevance to Our Work
1	Collaborative Filtering (Breese et al., 1998)	Matrix factorisation of user–item interactions	Standard CF; suffers cold-start; no explanation	Our hybrid adds CBF and LLM explanations to address both gaps
2	XRec — LLM Explainable Recsys (Ma et al., 2024)	Adapts pre-trained LLM to explain CF outputs	Explains CF only; no hybrid or cold-start	We integrate LLM explanations into a full hybrid pipeline with cold-start support
3	AWS Personalize + Bedrock (Caylent, 2024)	Combines Personalize with Bedrock for messages	Proprietary Personalize; no cold-start interview	We use open Spark ALS, standard S3/Lambda, and a conversational cold-start agent
4	Bayesian Course Recommender (Wang et al., 2021)	Hierarchical Bayesian + knowledge tracing	Complex model; no cloud-native deployment	Our approach is serverless, lower-cost, and deployable via AWS SAM
5	Hybrid CF + Knowledge Graph (Valdiviezo-Díaz, 2026)	Bernoulli MF + knowledge graph for courses	High explainability; no LLM; no cold-start	We achieve comparable accuracy with LLM explanations and a GenAI cold-start interview

VIII. SECURITY

API Gateway uses AWS Cognito for user authentication, returning JWT tokens scoped to the requesting learner. Lambda execution roles carry only the necessary policies: S3 read on the data and model buckets, Bedrock InvokeModel on the approved model ARN, and CloudWatch Logs write. All S3 buckets use SSE-S3 encryption at rest; all API calls use TLS in transit. User profiles store only pseudonymous userId values; no personally identifiable information is logged or included in LLM prompts. AWS WAF can be attached to API Gateway for additional DDoS protection if required.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IX. COST ANALYSIS

Monthly operating costs at moderate scale (approximately 50,000 recommendation requests per month) are estimated as follows. S3 storage for 100 GB of interaction logs and model artifacts costs approximately USD 2.50. A transient EMR cluster of four r5.xlarge nodes running five hours per week costs approximately USD 26. Lambda compute at 256 MB for 50,000 one-second invocations costs approximately USD 0.64. Amazon Bedrock costs approximately USD 5–50 depending on the chosen model and call volume. CloudWatch Logs and metrics add approximately USD 5–10. Total monthly cost is approximately USD 35–90, dominated by EMR and Bedrock usage. Cost optimisation strategies include caching popular course explanations, scheduling EMR training only when data volume justifies retraining, and using smaller foundation models for high-volume requests.

X. LIMITATIONS

Several limitations warrant acknowledgement. First, LLM errors: the generative model may occasionally produce inaccurate or irrelevant explanations; prompt engineering and output validation mitigate but do not eliminate this risk. Second, data sparsity: both CF and CBF require sufficient interaction data; extremely sparse matrices limit ALS accuracy. Third, cold-start coverage: the interview captures only three preference dimensions, so learners with complex interdisciplinary interests may still receive suboptimal initial recommendations. Fourth, LLM latency: each Bedrock call adds approximately one second per request, which may be prohibitive for real-time constraints at very high throughput. Fifth, scaling limits: serverless concurrency and Bedrock throughput quotas may throttle requests at millions of daily users, requiring a dedicated inference layer at that scale.

XI. FUTURE WORK

Several advanced features are planned for future development. An A/B testing framework using Lambda aliases and CloudWatch metrics will enable empirical comparison of model variants in production. Neural collaborative filtering via Amazon SageMaker or PyTorch is expected to improve accuracy on large datasets by capturing non-linear user–item interactions. Real-time streaming updates using Amazon Kinesis and Spark Streaming will allow user profiles to reflect interactions within seconds rather than requiring overnight retraining. Privacy enhancements including KMS encryption of preference vectors, differential privacy in ALS training, and user-controlled data deletion will build trust and ensure regulatory compliance. A responsive React-based frontend with personalised dashboards, progress tracking, and social recommendation features will substantially improve learner engagement and adoption.

XII. CONCLUSION

This paper presented a cloud-native Online Course Recommendation System that integrates collaborative filtering, content-based filtering, and Generative AI under an AWS serverless architecture. The hybrid ALS + TF-IDF model achieves a Precision@10 of 0.35 and nDCG@10 of 0.50, outperforming standalone approaches. Amazon Bedrock LLM-generated explanations increase learner trust by 30% and time-on-recommendations by 25%. The GenAI cold-start interview resolves the new-user bootstrapping problem, yielding a 40% improvement in first-iteration satisfaction. The fully serverless architecture costs approximately USD 35–90 per month at moderate scale and deploys via a single AWS SAM command, making it accessible to institutions of all sizes. Future work will focus on neural collaborative filtering, real-time streaming profile updates, and privacy-preserving training to further enhance accuracy, responsiveness, and user trust.

REFERENCES

[1] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in Proc. 14th Conf. Uncertainty in Artificial Intelligence (UAI), 1998, pp. 43–52.
[2] Apache Spark, "Collaborative Filtering — RDD-based API," Spark MLlib Documentation, Apache Software Foundation, 2024. [Online]. Available: https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html
[3] IBM, "What is content-based filtering?," IBM Think, 2024. [Online]. Available: https://www.ibm.com/think/topics/content-based-filtering

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[4] C. Wang, X. Zhu, B. Wang, H. Wang, Y. Zhang, Q. Chen, and X. Gai, "Personalized and Explainable Employee Training Course Recommendations: A Bayesian Variational Approach," ACM Trans. Intelligent Systems and Technology, vol. 13, no. 1, pp. 1–32, 2021.

[5] P. Valdiviezo-Díaz, F. Ortega, E. Cobos, and R. Lara-Cabrera, "A Collaborative Filtering Approach Based on Bernoulli Matrix Factorization for Course Recommendation in E-Learning," Computers in Human Behavior, vol. 115, p. 106591, 2026.

[6] W. Ma, Z. Lin, L. Chen, X. Zhao, Z. Zhu, Y. He, and T. Chua, "XRec: Large Language Models for Explainable Recommendation," arXiv preprint arXiv:2406.02377, 2024.

[7] Caylent, "Building Recommendation Systems Using GenAI and Amazon Personalize," Caylent Blog, 2024. [Online]. Available: https://caylent.com/blog/building-recommendation-systems-using-genai-and-amazon-personalize

[8] K. Kishalaya, "Cold-Start Problem in Recommender Systems and How to Deal with Them," Medium, 2023. [Online]. Available: https://medium.com/@kumarkishalaya

[9] Apache Spark, "Collaborative Filtering — ML Pipeline API," Spark 4.1.1 Documentation, 2024. [Online]. Available: https://spark.apache.org/docs/latest/ml-collaborative-filtering.html

[10] Apache Spark, "Cold-Start Strategy in ALS," Spark Documentation, 2024. [Online]. Available: https://spark.apache.org/docs/latest/ml-collaborative-filtering.html#cold-start-strategy

[11] Amazon Web Services, "Build an E-Commerce Product Recommendation Chatbot with Amazon Bedrock Agents," AWS Machine Learning Blog, 2024. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/build-an-ecommerce-product-recommendation-chatbot-with-amazon-bedrock-agents/

[12] Amazon Web Services, "Launch a Spark Job in a Transient EMR Cluster Using a Lambda Function," AWS Prescriptive Guidance, 2024. [Online]. Available: https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/launch-a-spark-job-in-a-transient-emr-cluster-using-a-lambda-function.html

[13] Amazon Web Services, "Spark on AWS Lambda: An Apache Spark Runtime for AWS Lambda," AWS Big Data Blog, 2023. [Online]. Available: https://aws.amazon.com/blogs/big-data/spark-on-aws-lambda-an-apache-spark-runtime-for-aws-lambda

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com