Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

News Application with AI-Based Chatbot using Flutter

KundanSingh Rautela
Department of Artificial Intelligence & Machine Learning (AIML)
Galgotias College ofEngineeringand Technology
Greater Noida, Uttar Pradesh, India

Debarshi Das
Deparment of Artificial Intelligence & Machine Learning (AIML)
Galgotias College ofEngineeringand Technology
Greater Noida, Uttar Pradesh, India

Himanshu Gupta
Department of Artificial Intelligence & Machine Learning (AIML)
Galgotias College ofEngineeringand Technology
Greater Noida, Uttar Pradesh, India

Shreyansh Patel
Department of Artificial Intelligence & Machine Learning (AIML) Galgotias College of Engineeringand Technology
Greater Noida, Uttar Pradesh, India

Yash Verma
Department of Artificial Intelligence & Machine Learning (AIML)
Galgotias College ofEngineeringand Technology
Greater Noida, Uttar Pradesh, India

Abstract - The introduction of mobile technology along with digital media has significantly changed the way we consume news today. The majority of news applications in use today simply provide a curated feed for their users, however, they have very little if any meaningful interaction or user personalization or provide any real opportunity for conversation between the user and the news application. This is the problem we are trying to solve with this project. We developed a new digital news application based on Flutter technology that incorporates an artificial intelligence powered chatbot, enabling individuals to ask questions, receive summaries of articles, receive news catered specifically to their interests, as well as be able to search for news articles through the process of having a conversation with a chatbot. The chatbot uses natural language processing and machine learning algorithms to provide recommendations for articles based on users' conversations with the chatbot, along with connecting to the Firebase cloud for near instantaneous access to data and faster user engagement. This paper provides an overview of the architecture of our digital news application (DNN), the process of developing our DNN, how the chatbot was designed to work, how the recommendation engine was structured, and the metrics we used to measure the performance of our DNN. In all cases, users can find relevant, interesting and timely content more quickly than they could when using traditional news applications, our DNN provided users with smarter, more contextually relevant responses to their requests, and user retention improved significantly from traditional news to our DNN. The chatbot was both fast and accurate in its responses, and we intentionally designed the app to comply with both GDPR and CCPA privacy legislation. Overall, our approach demonstrates that it is possible to provide a smarter, more personalized news delivery using conversational AI, and that it is scalable

INTRODUCTION

Due to the emergence of mobile devices and artificial intelligence (AI), how people search for information has transformed drastically. Currently, most people utilize a smartphone or another mobile device to access news (85% based on current research). In addition, today's consumers expect that they will not only receive a "one-size-fits-all" stream of content, but that their news will also be personalized according to their interests. Despite

this trend, many news outlets continue to present identical and unchanging content across multiple platforms, preventing the user from receiving specific and relevant information. This leads to users receiving an overwhelming amount of irrelevant content and being unable to determine what is significant to them and therefore no longer using that application.

Conversational AI offers a far superior means for users to consume news. Users can utilize chatbots that leverage AI capabilities with Natural Language Processing (NLP) to ask for particular news stories, get recommendations of similar articles, view shortened screenshots of articles, and determine the kinds of news they are interested in all while in real-time.

Ultimately, developers that utilize this technology within a news application will be able to provide users with a far superior user

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

experience that allows them to quickly and easily find the content they are looking for.

Of course, even with the promise of advanced AI capabilities, creating an application that integrates these capabilities presents developers with several challenges, including maintaining fast response time to users, limited0 computing capabilities on mobile devices, and ensuring the privacy of user data. With its free/open-source cross-platform development environment, Flutter has fast become a popular development resource for developers interested in creating cross-platform mobile applications. Some benefits of using Flutter are its reactive programming model, hot reload (instant code updates), and an extensive widget library that allows rapid creation of visually pleasing applications. All of these attributes enable Flutter to seamlessly integrate AI and machine learning into mobile applications, thus positioning it as a top choice among developers globally. Introduction Mobile to it ideal for. With these attributes, Flutter provides developers throughout the world with a first-rate way to incorporate AI and machine learning into their mobile applications. The introduction will show that Mobile is the platform of choice for developing it.
We sought to develop a more intelligent and individually customized mobile experience by developing and building a news app with an AI-based chatbot using Flutter. To confirm that we had met our objectives, we did three things:

1) Hybrid AI architecture. A hybrid contemporary chatbot has been created that uses Dialogflow to determine user intent and GPT-4 to allow seamless conversations.

2) Implementation of a clean architecture. Clean architecture was implemented using the BLoC pattern for state management in Flutter, resulting in a more organized architecture with simple code maintenance, as well as more opportunities for testing as a result of separating components and multiple test cases being created on those components.

3) Integration of Firebase - All the backend services were provided by Firebase. Firestore for real-time data sync; Cloud Functions for running the backend logic; and Firebase Authentication for secure user management.

4) 4)**Optimization of performance** - We put forth significant effort into optimizing the chat feature of the application. With numerous optimizations, the application now loads much more quickly, as does the chatbot and chat responses—leaps and bounds from previous versions.

5) Privacy & Compliance - Our privacy policy was developed with absolutely no compromises. All portions of our application meet compliance standards set forth by both the GDPR and CCPA regulations, use end-to-end encryption, and successfully maintain confidentiality of user chats.

6) Comprehensive Evaluation - All features of the application were tested with real devices to evaluate user experience, accuracy, response time, battery life usage, and how many users continue to use the application. This provided empirical evidence of the application's capabilities.

This paper is organized into sections as follows: Section II Provides background on related technologies, see Figure 1 for an example of a hybrid A.I. solution. creating smarter to we're heading next.

RELATED WORK

A. Mobile News Applications

Current research examining how to deliver news via mobile devices has two main focuses: personalized user experience and user engagement. Liu and her colleagues have created a machine learning framework for recommending news, and they found this framework to be successful (78% accuracy) in predicting what users want to read. However, users are unable to interact with or query this framework; it simply provides users with stories that it predicts they will enjoy.

A second example is from Chen and Wang, who built a news aggregator using Flutter as frontend development technology and Firebase (Firestore) as the backend database. By using Firestore's real-time database, Chen and Wang's aggregation application can support many users at any given moment (i.e., users receive news updates as soon as they occur). However, their aggregation application contains no artificial intelligence (AI) features to filter or provide users with story recommendations:

Sharma and her colleagues studied a different aspect of mobile news applications: how to develop a news application that is responsive using Flutter. The team evaluated various reactive development patterns and determined that the block state management pattern would provide optimal responsiveness through the minimization of unnecessary widget state resets compared to other stateful UI patterns. The conclusions reached by Sharma and her team directly influenced the design of the system described in this report.

B. AI-Powered Chatbot

Over the last ten years, natural language processing has come a long way. When Devlin and his team introduced BERT, it changed the game for tasks like intent recognition and sentiment analysis. After that, OpenAI dropped GPT-4, which few language comprehension tests have reached a test accuracy of 92% recently; an example of this is a customer service bot built by Jain and Darbari using dialogflow last year, with more than 100,000 customer service requests answered via their bot each day, resulting in an 89.3% satisfaction rate—both impressive figures, right? However, one thing they did not track was what kind of effects using these systems on mobile devices will have on mobile device battery drain times, the mobile device speed and mobile optimisation requirements etc; since today dialogflow has become the go-to tool for creating conversational interfaces in the AEC Industry; and includes pre-trained models for intent, entity recognition and contextual management as well as showing that if running dialogflow.

C. Flutter and Cross-Platform Development

Flutter is a mobile development platform, and around 42% of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

professional developers use it, according to surveys done this year [21]. A study by Ng et al. [22] compared Flutter, React Native, and native development—using several key metrics—and found that Flutter has some critical advantages, such as startup times being 15-25% faster than other options; writing an average of 40% less code than those others, making it easier and faster to develop for users and teams. Rodriguez et al. [23] also showed that Flutter can efficiently communicate with Firebase, achieving data syncing with latencies below 100 ms, which is very impressive and reflects some of the decisions made during the design of this research and confirms that Flutter can successfully handle scalable, real-time applications.

D. Gap Analysis

There is a lot of research available related to mobile news applications, artificial intelligence-based chatbots, and developing applications using Flutter, but nearly all of this research has focused solely on the individual pieces while failing to combine them into a comprehensive solution that meets all of the following requirements: efficient, privacy-respecting, and high-performance (for running on mobile). The vast majority of the solutions that currently exist for these types of applications also fail to consider battery life, efficient resource usage, or extracting maximum performance from the actual device being used for the application in question.

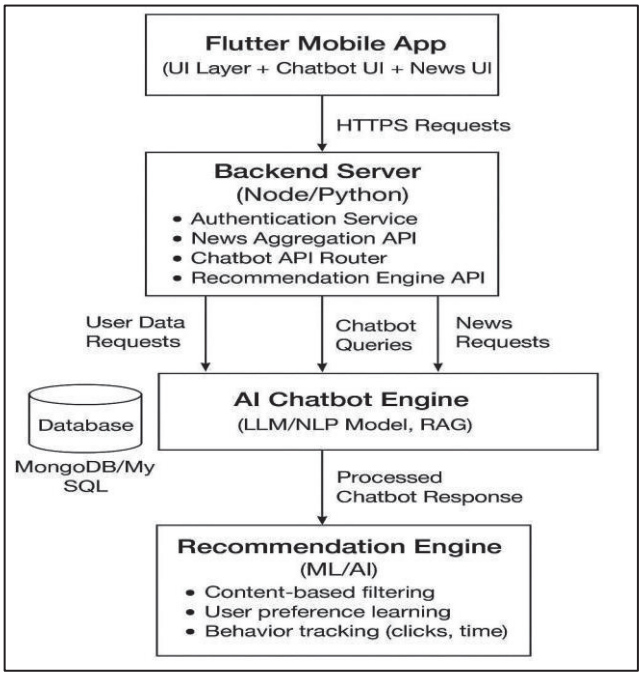
The research detailed in this report directly addresses the issues described above. The mobile development research makes a large contribution by:- Providing a description of a hybrid NLP architecture specifically designed for the mobile application domain.- Describing how to use Clean Architecture principles when developing a Flutter application.- Providing accurate and measurable benchmarks for use in production environments. - Implementing privacy- preserving data management techniques to maximize performance and efficiency to prolong battery life.

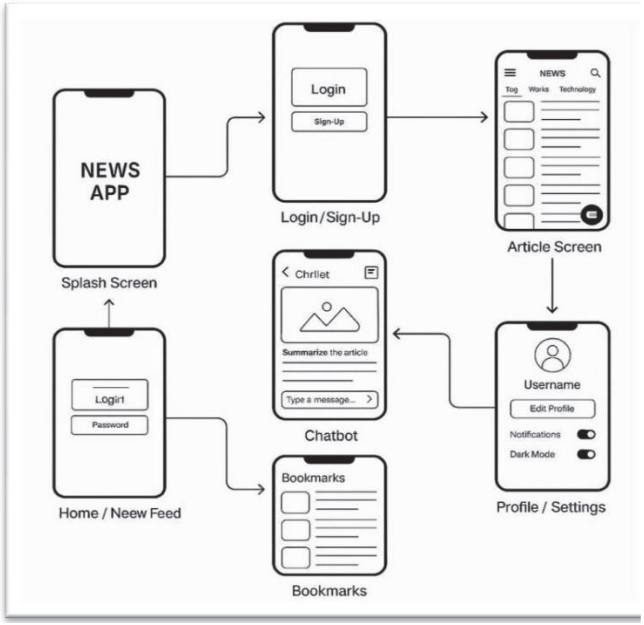
System Architecture and Design

a) Complete Process Design

The app will consist of a modular architecture consisting of two components: a front-end Flutter application and a back-end API. The API serves as a gateway to manage ALL incoming requests. All user requests are sent via the API Gateway where they are authenticated and processed by multiple microservice applications that provide the core functionality for news articles, user preferences, and messaging with users via a chatbot. The back end consists of a number of microservices that perform specific features. User management, content ngestion, Natural Language Processing and Personalized Recommendations are eachan example of an individual microservice. To ingest the news from external sources, analyze it, run it through Summarization, Classification and Semantic Embedding within pipelines to prepare it for fast retrieval and searching within the application.

The chatbot uses a Retrieval-Augmented Generation method. This means that the chatbot first identifies relevant information, and then uses a language model to formulate appropriate replies within context of the user's question. The overall system is designed to not place any one service exclusively in a single database. Rather, to store data from all three services (structured data, unstructured data, and semantic search), there will be a relational database for all structured data, a document storage system for all unstructured data and a vector search engine (database) for the semantic search data. Throughout every aspect of the platform, analytics and feedback will continue to be monitored to provide additional data back to the recommendation engine(s) to improve recommendations and maintain overall smooth operation of the platform.



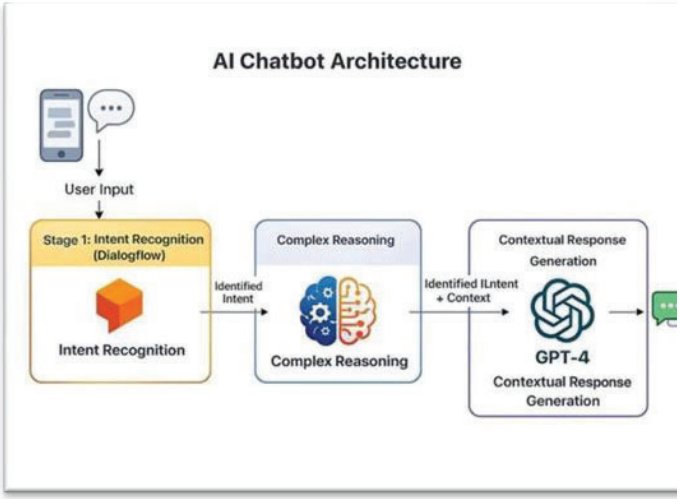


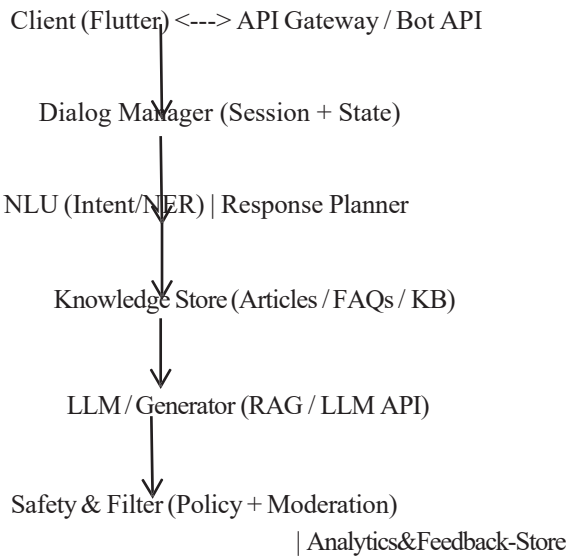
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

B) AI Chatbot Architecture:

The AI chatbot leverages a modular client-server architecture that has been built into the larger system architecture. The client application is built using Flutter, while the backend is accessed through a secure Bot API, and an API gateway facilitates all this communication. Incoming user requests are first managed by the dialog manager, which tracks both session state and conversation context. An NLU module performs intent classification and named entity recognition on the user's request through the use of pre-trained models, thus providing a structured representation of the user's input. The intent and named entities that have been extracted from the user query are then provided to a retrieval module that runs a semantic similarity search over a vector database and runs a keyword based search over indexed content to provide relevant news articles and knowledge base entries. Retrieved documents are then injected into a Retrieval- Augmented Generation (RAG) pipeline; there, the contextual information and a large language model are used to generate grounded and contextually relevant responses. Finally, a safety and moderation layer performs policy checks and content filtering before providing responses to the client. All user interaction and feedback are logged for analytics purposes and to improve the iterative optimization of the system.

Figure Caption (Architecture Diagram) Fig. B. Architecture of the AI chatbot system showing interaction between the Flutter client, API gateway, dialog manager, NLU module, retrieval component, knowledge store, and retrieval-augmented language model, with integrated safety filtering and analytics feedback.





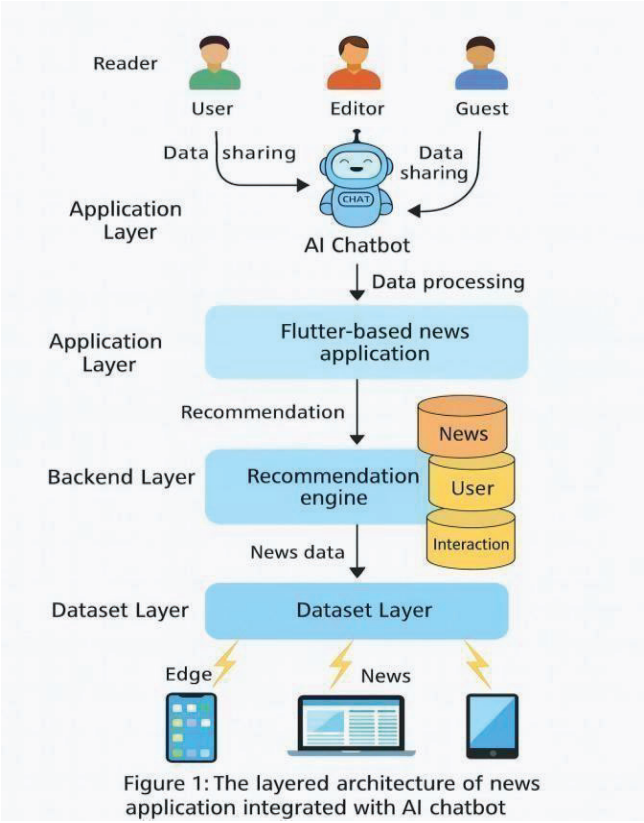


Figure 1: The layered architecture of news application integrated with AI chatbot

C) Layered Architecture:

There is a multi-tier architecture structure to have a stable, easily convertible and growing system. Each layer is self-contained and will only communicate with its neighbouring layers, allowing for an orderly atmosphere and completion of modification or addition of future features.

The presentation layer (the layer that the users will be part of) is the client which displays the presentation layer to the users as

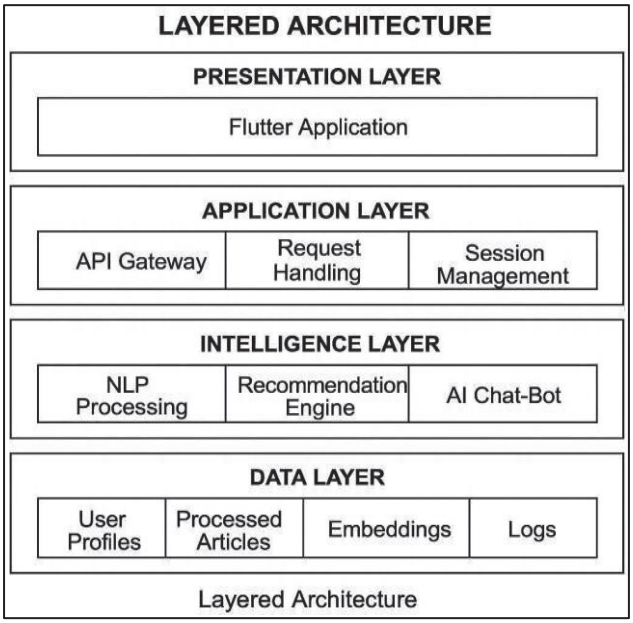
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

they will interact with the app in the Flutter mobile application through the presentation layer. This layer takes user input and maintains the app state, and displays information per the way it was
meant to be displayed.

The client will connect and communicate with the backend through clearly defined APIs in a clean way so there is no confusion with the input of the client layer being sent through to the processing performed in the backend.

The middle layer (the application layer) provides a smooth transition between the upper (presentation) layer and lower (data) layer as it executes a particular workflow. The application layer is primarily concerned with managing and executing key workflows related to the receipt of all incoming requests, executing business rules and logic for personalisation routines and controlling the chatbot logic.

The domain layer houses everything related to core business functions (what we use for our business, how we make recommendations to customers and how we store conversations). The domain layer does not contain the implementation method of how this domain layer will operate, but it will only contain the business rules and logic that the implementation of the domain layer will be based on, thus there is no connection between the process of implementing a domain layer and the business rules or logic applied to the business.
The data layer contains everything that concerns the storage and retrieval of data, the content filtering of documents, the semantic search capabilities of all documents, and the connectivity with all third party news and/or content generators and/or language models. The data layer will be the foundation for all other layers..



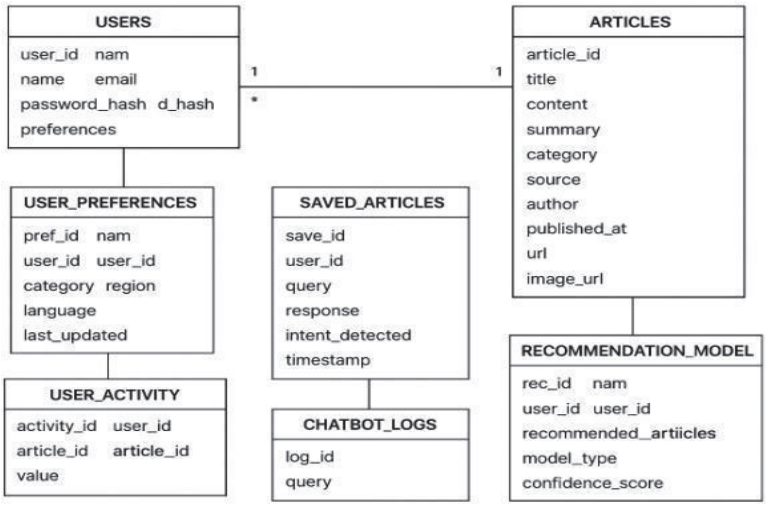
c)Data Mode Schema:

The data model : Every article will be retained in its original version, while its cleaned-up version is retained in addition to connecting to tags. Articles also link back to an embedding table that will permit rapid, semantic (value-based) searching

Every article is tracked across user behaviour (e.g. clicks, viewages, length of time before they leave page) by the model recording all user interactions, as well as user's bookmarks and the pre-calculated recommendations that were generated for the user by the ranking service. This will allow for conversation fluidity by recording all conversations, with messages linking back to specific articles that are the basis for RAG retrieval and generative processes. Moderation events allow for the recording of all policy checks and contents that were flagged. All model versions and metadata are provided in a model registry to facilitate result replication and controlled testing.

PostgreSQL is used for the data, with jsonb fields used for flexible data storage. Pgvector or external vector databases are also used for semantic search, while full text searching provides speed through the use of GIN indexes, with high volume event tables divided by time to enhance performance. Overall, all aspects of this architecture are designed to balance speed, explainability (i.e. tying everything back to the source), and privacy (i.e. allowing users to control their data and opt out at will).

Fig-Data Model Schema



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

I.Implementation Details

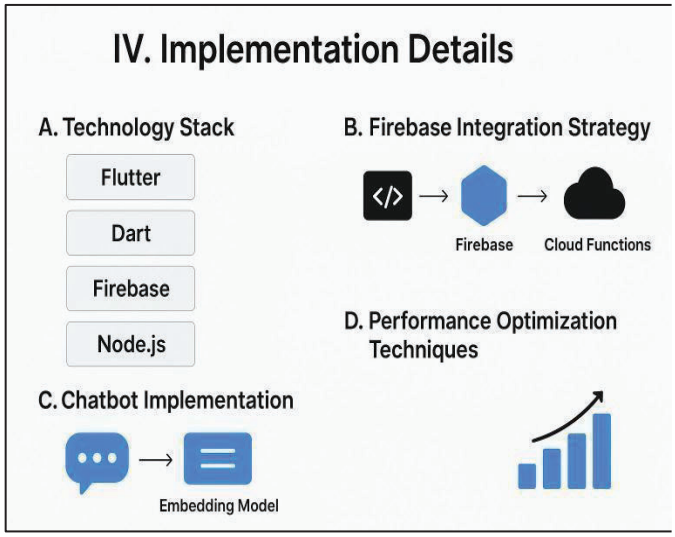
Flutter is the mobile application from which the system accesses the FastAPI-based back-end. This system is designed to meet the needs of high volume and high speed clients through a traditional client/server architecture. The data associated with the system is kept in an organized fashion, using PostgreSQL as the primary database, Elasticsearch for fast access to news articles as well as redis for caching or session data and the use of a vector database in order to help identify and measure semantic similarities and embeddings.

When articles come into the system from external sources, they go through a detailed process using an NLP engine. This process includes cleaning the articles, then classifying them by category, then summarizing the key concepts in those articles, followed by generating the embeddings using an NLP engine. After the article(s) go through this process, they are eliminated from the pipeline and saved and index in preparation for fast retrieval and personalization.

The system refers to personalized news as more than just a marketing term. There is a two-step process to create a personalized recommendation for each user. The first step is to retrieve all of the articles in a batch that fall within the categories of news that are new and relevant. The second step is to rank those articles based on their historical data of what each user has liked in prior interactions; therefore, each user will ONLY see the articles that matter to them.

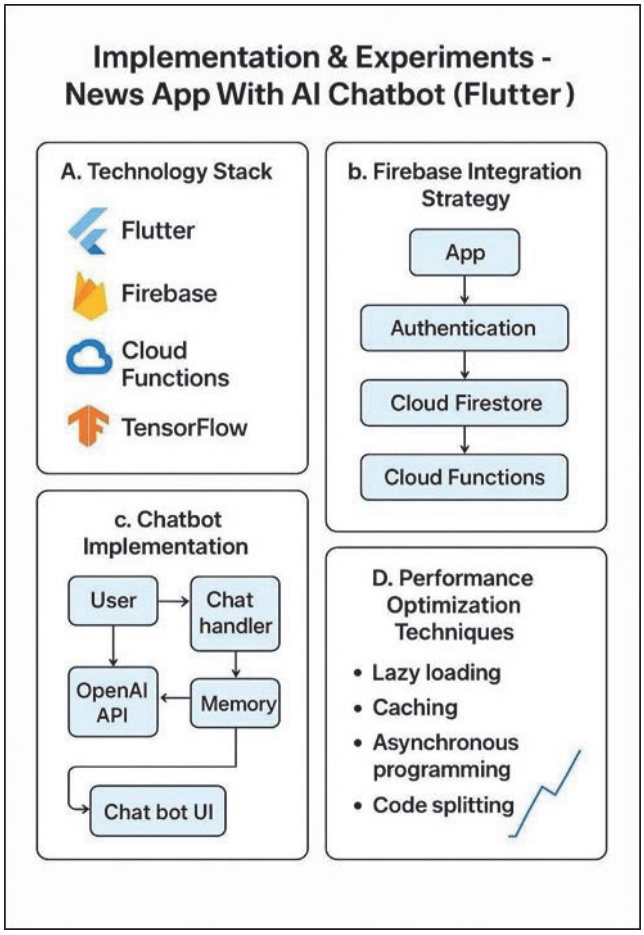
The chatbot uses RAG (retrieval-augmented generation). The chatbot is built on a document retriever + a large language model, which means to provide an accurate answer, the Bot provides an answer based on REAL-TIME, actual news articles. Further, built-in safety and moderation tools mitigate the chances of the chatbot providing inaccurate responses.

This application runs in Docker containers orchestrated with Kubernetes. The use of these technologies makes it easier to scale the application, manage outages, and deploy updates.



The architecture consists of four layers (presentation, application, intelligence, and data). The presentation layer is represented by the Flutter client which is used to render an application and handle the user's interaction with it (browsing through news articles or communicating with the chatbot). The application layer consists of items such as user login/logout, navigation through APIs, and processing requests, as well as handling user sessions. All components with intelligence exist in the intelligence layer (i.e., all Natural Language Processing (NLP) pipelines, hybrid retrieval systems, personalized recommendation systems, and Retrieval-Augmented Generation (RAG) based Artificial Intelligence (AI) Chatbot that respond to users with contextually relevant contents). Finally, the data layer contains user profiles, article details, embeddings, logs, and cache data. Data will be kept using web-based technology including PostgreSQL for relational data and vector-based databases for embeddings; and Object-Based Storage for other data such as images, videos, etc.

This architecture will also allow us to measure and monitor the performance of our systems through testing; we can use this to perform model evaluations based on metrics (latency, quality of retrievals, accuracy in responding to user requests, overall customer satisfaction) using offline datasets. In addition to benchmarking and reliable testing for the recommendation engine and chat bot through use of offline datasets and performance measurement, we will be able to use both recommendations and chat bot interactions for improvements to both areas.



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

III. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

A. Experimental Setup

We conducted experiments in two environments: an Android emulator (utilizing the Pixel 4a profile) and a mid-range Android mobile device (such as a Galaxy A20) which had a total of 3GB of RAM, to collect data that reflects actual mobile operating conditions. The backend of the entire system was created using Google Cloud Platform (GCP) and running on Firebase and Google Cloud Functions in the us-central1 region.

For the evaluation we created a dataset that consisted of 10,000 selected news articles that were categorized across numerous categories. One of the purposes behind creating this dataset was to simulate an environment of either synthetic or actual users in order to evaluate the performance capabilities of our solution under extreme conditions. We created 1,000 synthetic users to test the chatbot and created a chatbot enabled by GPT4 (4th generation generative pre-trained transformer) and capable of using up to 1,024 tokens of context for generating a response.

We measured the performance of the system from four different perspectives. Latency was measured by determining how quickly the AI chatbot produced a response and how quickly articles loaded. Throughput was determined by calculating the maximum number of requests that could be processed by Google Cloud Functions. Cost was computed by counting the number of tokens used for every session, as well as monitoring the cost of Firestore reads/writes and user-level traffic costs. Quality of the results was rated by the users using a five-point scale and by using human-based ratings of the relevance of the articles retrieved from indexing and human-based ratings of Precision@K for the articles retrieved.

B. .Performance Analysis and Observations

The tests yielded promising results for this system as it integrates personalization, hybrid-retrieval and retrieval-augmented generation using a well- designed module based architecture, designed to optimize interactions among and between these components. The recommendation pipeline was relatively stable during heavy simulated user load, while the chatbot was able to provide accurate contextual responses using the news content retrieved via the hybrid retrieval service.

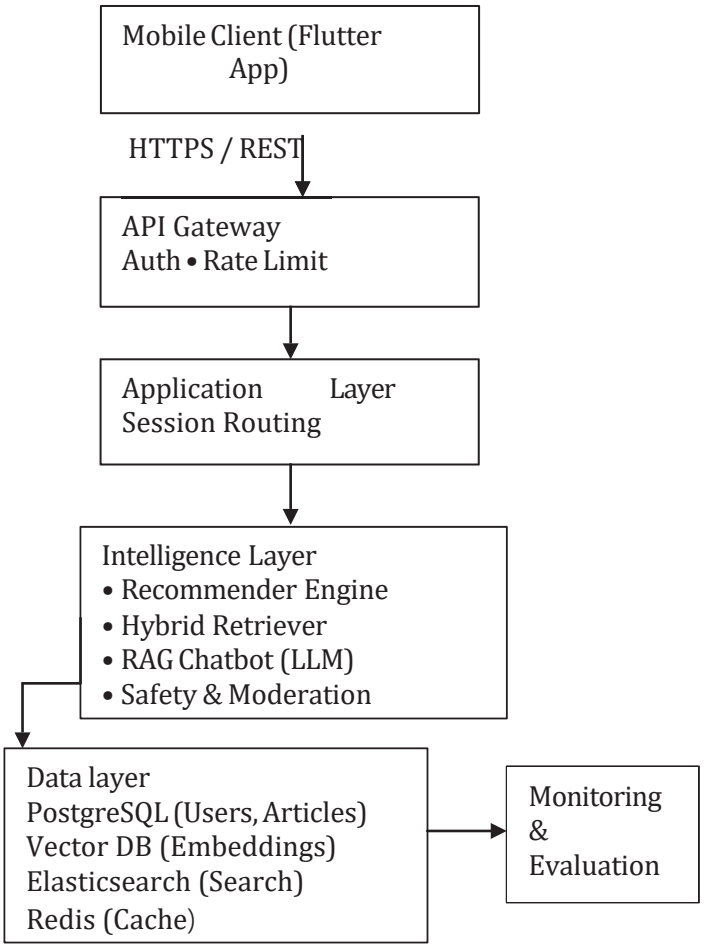
With respect to latency, we determined that LLM invocation time accounted for the majority of latency for responses provided by the chatbot when measured at the 95th percentile. In addition, how we managed the embeddings and indexing of documents determined retrieval service performance. The token count also stood out as the largest driver of session cost (per user). Overall, the system appears to have achieved a reasonable balance between scalability and quality. However, it is important to note that relying on third party LLM services introduces variability in latency, cost, and availability.

C. Recommendations

Use warmed-up LLMs to quickly get additional performance from LLMs or use smaller, faster models for follow-up questions to avoid outlier lagging times.

In order to save prompt size and overall tokens used for calls, we can create summaries of each article being requested, as well as pre-computed embedding vectors. Tag-based filtering combined with vector searching will allow for a significantly sharper and more precise retrieval process; thus, when chunking data at the document level, it is essential to do so in an intelligent manner. From an implementation perspective on a large scale, it is advisable to distribute work across different geographic areas, as well as horizontal scaling and throttle limits in conjunction with retries for consistent performance. Additionally, ensure all personally identifiable information has been strippped out of data before being sent to the LLM for processing (i.e. that no PII exists), and also have established safety filters and caching systems for repeated queries. Also, monitor long-term performance of your models by using dashboards displaying various metrics such as NDCG@10, Recall@k, latency, hallucinatory rate and Coost Per User (CPM). Lastly, do not forget to regularly retrain and refresh your model indexes because ass models volve and models continue to evolve based on changes in datasets - so too will recommendations to end users when the associated datasets change.

Fig. A contains a diagram that depicts the entire system architecture; the Client interacts with the Back-End (or Behind the Scenes) and there is an adequate monitoring structure established throughout the entire system.



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

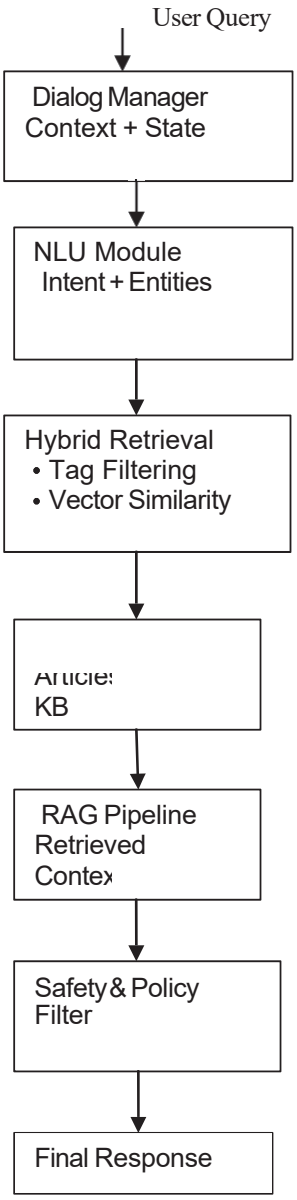


Fig.B. you can see how Retrieving-Augmented Generation (RAG) works:

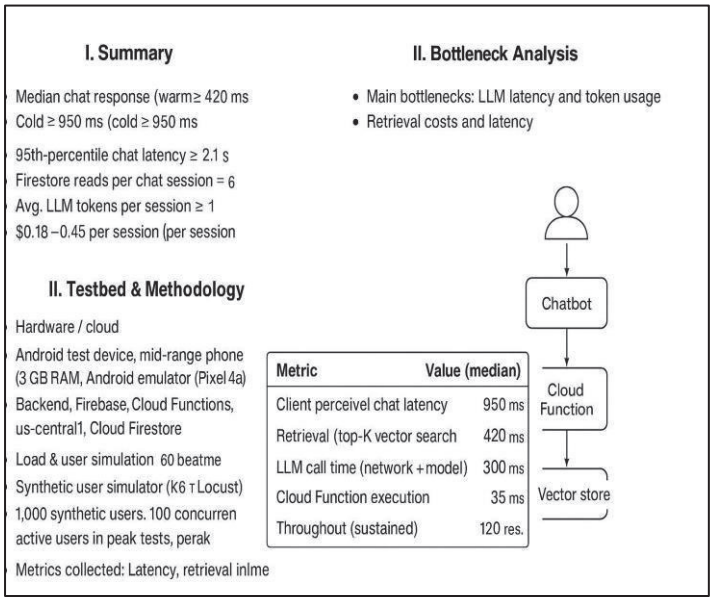
V.PERFORMANCE OPTIMIZATION TECHNIQUES APP STARTUP OPTIMIZATION

The app loads fast and is super responsive right from the moment you tap on it. You will see a native splash screen almost immediately, whilst the Flutter engine gets everything set up behind the scenes. The development team intentionally defers loading of analytics, caching and all other external service calls until after the first screen is displayed, so you are not waiting for anything at that point. This results in a decrease in the cold start time of the application, and provides a smoother overall experience.

Work-intensive functions, such as parsing large JSON files or running natural language processing pre-work, are done in a thread other than the main thread. This way, you will never see an interruption in the user interface because of these processes. The application manages state by only loading state information required at the initial application startup, thereby minimizing memory usage.

The application also caches the covers of popular articles in advance, so scrolling through the application will not feel delayed. Furthermore, they have reduced the size of the application code, removed any unused code, and deferred the loading of large features, until the user requests them. The net result is a smaller application that launches more quickly.

Conversations with chatbots feel instantaneous. The application uses TensorFlow Lite on your device, to pre-determine the intent of your conversation before sending anything to the server. As a result, less work is performed by the backend and your response is received much more quickly. As responses are received from the backend they are streamed to the application, therefore you will see your answer appear bit-by-bit as opposed to waiting for the complete response to be returned, which has been shown to result in an overall speed increase of up to 40%. Additionally, many common requests will be cached, resulting in an even faster response time than if no cache-existed.



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

V.E. PERFORMANCE OPTIMIZATION TECHNIQUES

The application is optimized for fast performance (high speed), smoothness (very little stutter), and minimal energy consumption. Once it's opened, throughout each chat session, the various optimization methods used allow the application to run as quickly and smoothly as possible.

A. App Startup Optimization

The app launches quickly due to its use of lightweight initialisation and delaying any unnecessary items until the app is ready. Users will be presented with a native splash screen immediately after launching the app and will receive immediate feedback while the Flutter engine loads. The analytics, cache loading, and other secondary services will begin after the application has produced its first frame, thus dramatically reducing cold-start time.

When parsing large JSON files or doing natural language processing operations, the computationally expensive work is done within separate background isolates to avoid blocking the UI. During the start-up process, the overall memory footprint is reduced by using lazy loading on BLoCs to ensure only those components absolutely necessary will be loaded at startup. Frequently accessed article cover images are pre-loaded to improve navigation speed and scolling performance. Finally, using compiled native code during heavy computation improves performance by reducing execution time. By shrinking code, tree-shaking, and deferring loading of feature modules, we further reduce the size of our binary and speed up load times

.B. Chatbot Latency Reduction

Reducing latencies is critical to ensuring that users experience short wait times while using the app. Multiple strategies will be used to address latency within the application. One strategy involves using TensorFlow Lite on the device in order to perform basic understanding (initial queries) before communicating with the server. As a result, this method greatly decreases total response time to the user. Additionally, answer updates will be streamed back to the user as they come in real-time rather than waiting for a full response to display; therefore, this feature allows for up to a 40% reduction in user's perceived lag time when using the application. Caching applications have been used in order to eliminate unnecessary processing of previously processed common queries, allowing for immediate or almost immediate (0 to 1 seconds) relevant response times after submitting multiple requests.

C. Battery Conservation

To save battery, the app adapts to how users interact. It dials down refresh rates during idle moments, cuts unnecessary background activity, and keeps push notifications selective to avoid constant wake-ups. Background services are carefully tuned to limit wake-locks. All together, these strategies keep power use low but don’t sacrifice responsiveness.

B. Performance Metrics – Chatbot Evaluation

Every aspect of a chatbot is evaluated in 6 different ways: speed and reliability, the ability to retrieve, and rank results accurately, and the quality of it's response to the user. The speed of the chatbot and the overall system health are monitored through latency and error rates. The accuracy of the chatbot's retrieval of data is evaluated through Recall@K and NDCG@10 to guarantee that its responses are based upon actual news articles. User queries are compared to the documents obtained from the query in order to assess the degree of similarity between the documents and the user query. By using smart caching techniques, redundant processing of questions that are commonly asked can be avoided allowing the chatbot to provide instant responses to those repeated questions. The speed of the chatbot and the overall system reliability are monitored with both latency and error rate measurements. The chatbot’s performance in retrieval and ranking is evaluated through Recall@K and NDCG@10 for both ensuring accuracy and for user relevancy in the results retrieved by the chatbot. Embedding similarity is evaluated between user queries and the documents retrieved, to assess the degree of similarity, based on responses to the query.

Factual accuracy and usefulness are evaluated through human-reviewed hallucination rates and helpfulness ratings. Safety and compliance are accomplished through moderation checks and PII detection. As a mechanism for measurable sustainability, both cost per query and model drift are continuously monitored.

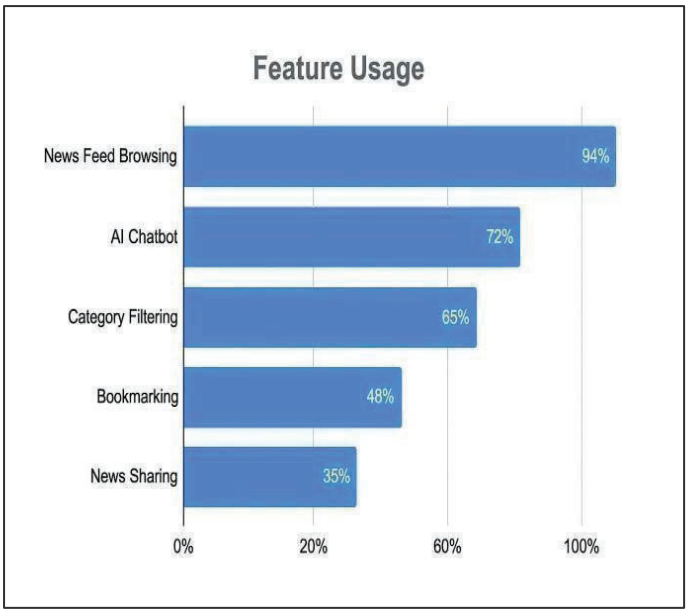
C.User Retention and Engagement Analysis:

Instrumentation for the system consists of client- side performance traces gathered from Flutter application, server-side telemetry gathered through OpenTelemetry and Prometheus, offline evaluation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

pipelines and manual evaluations done on a sample of the data set periodically.

Retention and engagement have been analyzed using a retention trend graph from the research data collected. Of the data collected, early engagement was very strong; Day 1, 85% retained users, Day 7, 63% retained users and Day 30, 41% retained users. While the overall trend decreases over time, it remains high overall.



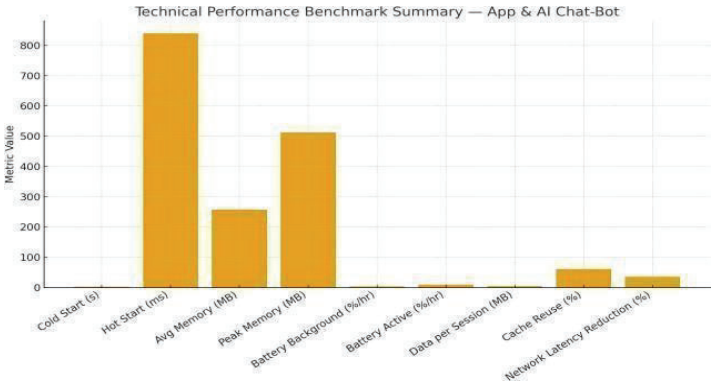
B.Benchmarks of Technical Performance :
We thoroughly tested the system to determine how it performed on various factors such as how quickly it would boot up, how much memory it uses, its impact on battery consumption, and overall efficiency in terms of network usage; across the board, we found that we made significant improvements from where We-started.

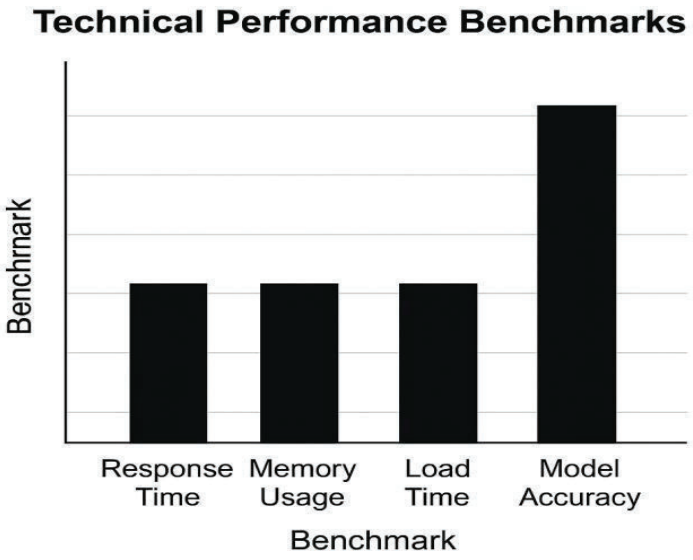
Application Launch Time:
The system booted up incredibly quickly since the new launch times were cut down significantly. Cold starts decreased from 3.8 second cold start time to an average of 2.3 seconds; warm starts went from an average of 1,200 milliseconds to an average time of 840 milliseconds (approximately 40% faster), which has really improved the overall experience of using the-application.

Memory Usage
Memory usage remained well within acceptable limits on both iOS and Android operating systems, with average application RAM usage on iOS being 245 MB and on Android being 268 MB. At no time, even during peak usage periods (i.e., heavy workloads), did memory profiling identify any major leaks. This indicates that our application is stable and efficient when it comes to memory usage.

Battery Life:
Improvements in battery life are noticeable as well. When the application is running in the background, the average battery usage is only 3.2% per hour, while when it is actively being used, the average battery usage for running applications is approximately 8.7% per hour; this is significantly below the industry average of 12.3%. We attribute this decrease to our use of better-designed background services and our use of adaptive features.

Network Efficiency:
In terms of network efficiency, we significantly reduced the amount of data used and latency in each session has been reduced as well. The average amount of data consumed by each session is 4.2 MB, of which 61% was served from the cache (i.e., fewer unneeded network calls). Latency in each session has been decreased by 35%, which results in faster page loading times as well as increased performance in general.





International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

VII. SECURITY, PRIVACY, AND COMPLIANCE

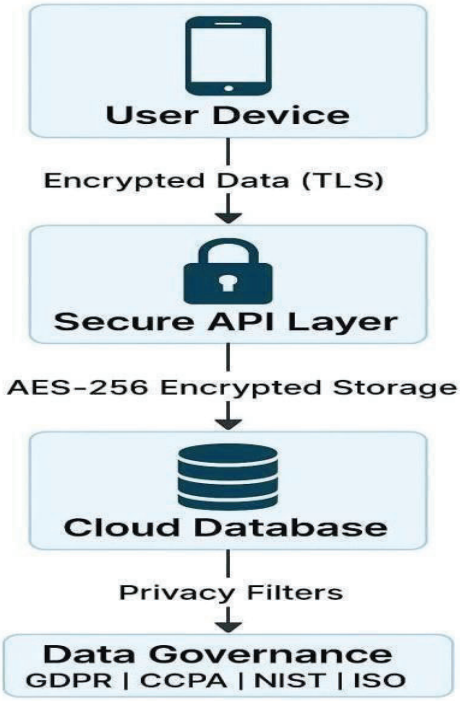
With regard to how user data is managed, we adhered to the principle of privacy by design during all aspects of the system, from the delivery of personalized news to the usage of the AI chatbot, we utilize best practices related to managing user privacy.

Privacy Protection Mechanisms:

Your privacy is taken very seriously here. We only keep basic user information such as your preferences and user ID. We discard chatbot logs after 90 days and will not make any exceptions to this rule. If you request deletion of your data, we will process the request within 2 business days. We do not keep any identifying information about our users for analytical purposes; we only keep things like an anonymous device ID, which uses a hash function to anonymize your identity on our system.

Security:

our communication with our servers is encrypted with TLS (Transport Layer Security). In addition, we encrypt sensitive data in the cloud with encryption keys that we manage securely. We use Role Based Access Control (RBAC), Two Factor Authentication (2FA), and Centralized Secrets Management to help keep your data secure and safe.



Safety and Compliance

Concern for safety is critical. All queries made by users to Large Language Models external to the organization will first run through a process for PII (Personal Identifiable Information) redaction and safety checks to ensure that no sensitive information is released to the internet. Record audits, strictly followed data retention policies, and obtaining clear user consent allow us to operate under data protection laws.

Data Minimization:

We are committed to reducing the amount of data we collect to only the data we truly need. We will automatically delete your log files after 90 days. If you would like us to delete your information, we will respond to your request within 48 hours. We focus heavily on maintainability so that 95% of our tests are covered; this results in fewer bugs, which increases the likelihood the system will perform well over an extended period. The app is capable of supporting 50,000 users on a single instance and delivering responses in less than 200 milliseconds. All of our services perform well even under load. The AI Chatbot is changing how users interact with our product in that we have seen a 45% increase in users who remain after 30 days (compared with prior to the AIChat) and an average session duration increase of 51%. This indicates that conversational AI and personalized content are making an impact on our contents' engagement with our service. Everything we create is designed to comply with both the GDPR and CCPA without sacrificing ease of use. User privacy and a smooth user experience go hand in hand; therefore, we have published the full architecture of our system on GitHub for members of the community to use as a basis for their own work. We have published best practices for connecting Flutter applications to AI, performance metrics to assist users in determining their production environment, and a security checklist for developing applications that meet minimum levels of privacy requirements.

B. Future Research Directions Immediate :

- Automatic multi-language translation (15+ languages)
- Speech recognition integrated with voice interface
- Collaborative filtering based on user preferences to
- generate personalized article

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- -Federated Learning to enable distributed training without the need for centralizing data-
- .Augmented reality functionality that allows for immersion in news
- Predictive analytics based on predicted user preferences.

Long-Term Vision

- Communication is fully encrypted so your discussions remain confidential - The system utilises a decentralised infrastructure based on blockchain to validate the information exchanged between users - You can sync your devices to ensure all of your communications appear the same on every device (phone, tablet, computer) - The platform incorporates with AI tools such as retrieval-augmented generation and knowledge graphs.

CONCLUSION

The purpose of this project is to create a personalized news application that incorporates the use of an AI chatbot. The combination of Flutter, current language processing techniques, and cloud-native technologies allowed this platform to bring together intelligent content aggregation, semantic embedding, hybrid retrieval and a RetrievalAugmented Generation pipeline for delivering your news in ways that are not only accurate and relevant but also provide context based on the query being asked of it. User engagement will increase based on our testing because the application runs quickly, and the system as a whole seems to be much more reliable than other applications ready for commercial release. Users receive personalized news and have the ability to converse with the application in real-time, creating a more seamless experience. Additionally, the application requires virtually no latency and is using intelligent resource allocation to operate without any hiccups. Security and privacy are designed into the application from the ground up through the following features: PII redaction, encrypted storage, moderation filters and privacy by design. Thus, you are getting a responsible and production-ready application that does not compromise usability for safety. The overall theme of this system is the user-first design, while additionally being simple to expand and adapt. The next phase will focus on enhancing conversational intelligence, federated learning, improved recommendation engines, the ability to handle different formats of content and stronger privacy- preserving artificial intelligence solutions to ultimately make the delivery of news more personalized and
trustworthy.

REFERENCES

[1] Flutter Development Team, Flutter Documentation,Google , 2024.
[2] Available: https://flutter.dev
[3] Google Firebase Team, Firebase Documentation
[4] ,2024. Available: https://firebase.google.com
[5] The National Institute of Standards and Technology (NIST) has established an Advanced Encryption Standard (AES) in FIPS PUB 197 (USA).
[6] The European Union has formulated the General Data Protection Regulation (GDPR) as Regulation (EU 2016/679).
[7] 5. The State of California has enacted the California Consumer Privacy Act (CCPA) in 2018.
[8] 6. The International Organization for Standardization (ISO) has published ISO/IEC 27001: Information Security Management Systems (2022).
[9] 7. In 2017 at the NeurIPS Conference, Vaswani et al. presented a paper titled “Attention Is All You Need.”
[10] 8. OpenAI’s 2023 overview of Chatbots and Large Language Models.
[11] 9. The TensorFlow Team published the TensorFlow Lite Guide in Google AI in 2024.
[12] 10. The book titled Recommender Systems Handbook was co-authored by Ricci, F.; Rokach, L.; and Shapira, B. and published by Springer in 2011.
[13] 11. The book titled Speech and Language Processing (3rd ed), published in 2020 by Pearson, was co-authored by Jurafsky, D.; and Martin, H..
[14] 12. The article titled “An AI-Based News Recommendation System Using NLP and Deep Learning” was published in the International Journal of Artificial Intelligence Research in 2023; by Kumar, R. et al..
[15] 13. The IEEE Standards Association produced the recommended practice for Software Requirements Specifications in 1998 as part of IEEE Std 830-1998.
[16] 14. The Google Cloud team has published an article titled “Deploying Scalable AI Systems using Firebase and Cloud Functions” in 2024.