ChatGPT

Sahayak: Google Cloud–Firebase Deployment Architecture

Figure: Provided multi-agent teaching-assistant architecture (MCP host orchestrating multiple agents and external services).

Android App & Firebase Integration: The Sahayak Android app uses Firebase SDKs for user authentication, data sync, and storage. Teachers log in via **Firebase Authentication**, a secure identity solution (supporting email, Google Sign-in, etc.) [1]. Their preferences, past sessions, and classroom configs are stored in **Cloud Firestore**, a flexible, scalable NoSQL database built on Google Cloud [2]. Firestore offers realtime listeners and offline caching for responsive apps [3]. All Firestore data is protected by Firebase Auth and security rules [4]. (Static assets or admin dashboards can be served via **Firebase Hosting** if needed.) For file outputs (worksheets, slides, PDFs), we use **Firebase Cloud Storage**, which is backed by Google Cloud Storage (exabyte-scale, globally redundant object storage) [5]. By default Storage access is locked to authenticated users [6], so only the owning teacher can download generated files.

- **MCP Host (Orchestration):** Teacher input (prompt text/image/audio) is sent from the app via HTTPS to a central orchestrator (the "MCP Host"). This host runs on **Firebase Cloud Functions** or as a **Cloud Run** microservice (HTTP-triggered). The MCP Host uses **Cloud Pub/Sub** to publish task messages, decoupling processing and enabling asynchronous scaling [7]. For example, the MCP Host can publish a "generate worksheet" or "answer question" task to a topic.

- **Agent Microservices (Cloud Run):** Each agent (e.g. WorksheetAgent, ContentAgent, ArtAgent, ExamAgent, QA Agent) is implemented as a stateless microservice deployed on **Cloud Run**. Cloud Run is a serverless container platform that automatically scales instances up or down to handle incoming requests [8]. Agents pull tasks (e.g. via Pub/Sub subscriptions or direct invocation) and perform their work independently. Because Cloud Run scales to zero when idle, it minimizes cost during low usage.

- **AI and Tool Service Wrappers:** Agents call Google AI services through REST APIs (often wrapped in Cloud Run helper services). For example: the ContentAgent and QA Agent use the **Vertex AI** Gemini models to generate text or images; the ArtAgent uses Gemini/Image for illustrations; the WorksheetAgent can call **Document AI** (OCR) to extract text from uploaded images [9]. Each agent may also invoke **Cloud Text-to-Speech** for audio answers (converting text to natural-sounding speech) [10]. These service calls can be organized as "MCP server" wrappers (e.g. a Gemini-text server, a Document-AI server) so the agent logic simply requests a tool by name.

- **Real-Time Feedback:** To stream status updates back to the teacher, agents publish progress events. One approach is Cloud Pub/Sub: agents publish to a "status" topic, and a Cloud Function writes updates into Firestore or the **Firebase Realtime Database**. The Android app attaches a realtime listener to Firestore/RealtimeDB, so it sees live status (e.g. "ArtAgent generating image…") as soon as agents update it [3]. This gives teachers immediate feedback on agent activity. (Alternatively, agents could write directly to a Firestore "status" document that the UI listens to.)

- **Persistent Storage & Output Delivery:** Completed outputs are sent back to the app or saved as files. Short answers (single paragraphs or explanations) are returned inline via Firestore or direct JSON over HTTPS. If an agent produces a large worksheet, slide deck, or PDF, the **Combiner** service (another Cloud Run function) assembles the content and writes it to Firebase Storage (Google Cloud Storage) [5]. A secure download URL is stored in Firestore for the app to retrieve. Optionally, generated slide content could be pushed into a Google Slides deck via the **Google Slides API**, then shared. In all cases, file access is restricted by Firebase security rules so only the authenticated teacher can fetch it [6].

- **Scalability and Security:** All services are deployed in the India (asia-south1/Mumbai) region to minimize latency for local classrooms. Cloud Pub/Sub and Firestore automatically distribute data across zones for high availability. Cloud Run's autoscaling handles sudden load (e.g. many concurrent teacher requests) [8]. Firestore and Pub/Sub are designed for global scale [7] [2]. We use IAM and Firebase Security Rules to enforce access control: only the authenticated teacher's UID can read or write their Firestore documents or storage objects [4] [6]. This cloud-native design (serverless functions, managed databases, autoscaling containers) provides a secure, multi-tenant, and fault-tolerant setup with minimal operational overhead.

This architecture uses only Google Cloud and Firebase services, meets real-time update requirements, and is optimized for low-latency, production use in Indian classroom settings.

Sources: Google Firebase and Cloud documentation for Authentication, Firestore, Storage, Cloud Run, Pub/Sub, Document AI, and Text-to-Speech [1] [2] [4] [5] [7] [8] [9] [10].

---

[1] Firebase Authentication | Simple, multi-platform sign-in
https://firebase.google.com/products/auth

[2] [3] [4] Firestore | Firebase
https://firebase.google.com/docs/firestore

[5] [6] Get started with Cloud Storage on web | Cloud Storage for Firebase
https://firebase.google.com/docs/storage/web/start

[7] Architectural overview of Pub/Sub | Pub/Sub Documentation | Google Cloud
https://cloud.google.com/pubsub/architecture

[8] About instance autoscaling in Cloud Run services | Cloud Run Documentation | Google Cloud
https://cloud.google.com/run/docs/about-instance-autoscaling

[9] Document AI | Google Cloud
https://cloud.google.com/document-ai

[10] Text-to-Speech AI: Lifelike Speech Synthesis | Google Cloud
https://cloud.google.com/text-to-speech