# Technical Terms & Simplified Explanations

## Salted SHA256 hashes

Original: All user identifiers are stored as salted SHA256 hashes. No personally identifiable information (PII) is retained.

Simplified: User IDs are turned into scrambled codes using SHA256 and extra random data, so they can't be traced back to a person, and we don't keep any real names or emails.

## Differentially private embeddings

Original: We extract embeddings on-device (differentially private) and send only high-level signals (e.g., sentiment scores, topic clusters) to the server.

Simplified: We process data on the user's device and add slight random noise, so the data keeps individual details private, and only summary information goes to our servers.

## AWS Lambda

Original: Incoming events first pass through a lightweight AWS Lambda (or similar) function that filters duplicates and aggregates minor events.

Simplified: A small cloud function runs automatically to clean and group most of the data before it moves on.

## Kinesis (or Pub/Sub)

Original: Buffers events in Kinesis (or Pub/Sub) with partition keys on app\_source.

Simplified: We store incoming data in a streaming service and organize it by which app it came from.

## Apache Flink

Original: Stream Processing (AWS Kinesis Data Analytics or Apache Flink): Performs real-time aggregation (e.g., share-rate counters, loop-alert thresholds).

Simplified: We use software that can process data as it's generated, instantly counting shares or detecting patterns.

## BERT-based classifier

Original: Sentiment Analysis: On aggregated embeddings, run a BERT-based classifier to tag clusters: “nostalgia,” “anxiety,” “grief,” etc.

Simplified: We use an AI model (BERT) to understand the emotional tone of messages and label them.

## Zero-shot GPT classifier

Original: Feedback Classification: Use a zero-shot GPT classifier to categorize Suggestions Box entries into feature\_request, bug\_report, or idea\_proposal.

Simplified: We use an AI that can sort feedback into categories without needing prior training examples.

## Pre-baked Docker images

Original: Use Docker images pre-baked with Android SDK, Flutter, and Unity CLI to reduce build times.

Simplified: We use container images that already have all necessary development tools installed, so building apps is faster.

## Kubernetes-based build farm

Original: Deploy a Kubernetes-based build farm (e.g., Jenkins agents or GitHub self-hosted runners on GKE). Configure auto-scaling based on queued jobs.

Simplified: We run our build servers in a container system that automatically adds more servers when needed.

## OAuth2

Original: Integrate with Apple HealthKit and Google Fit via secure oAuth2.

Simplified: We connect to Apple and Google health services using a secure standard that lets users give permission without sharing passwords.

## AES-256 & TLS 1.3

Original: Encrypted at Rest & In Transit: All data in S3, MongoDB, and Redis is AES-256 encrypted; communications use TLS 1.3.

Simplified: All stored data is locked with a very strong code, and data moving over the internet is protected by the latest secure protocol.

## GDPR & CCPA compliance

Original: GDPR & CCPA: User data deletion endpoint completely purges hashed IDs... Logs are anonymized.

Simplified: We follow European and California privacy laws by giving users the ability to delete their data and keeping records anonymous.

## SOC 2 Type II audit

Original: Annual SOC 2 Type II audit for cloud infrastructure.

Simplified: We have a yearly independent check to make sure our cloud setup meets strict security standards.

## UMAP + HDBSCAN

Original: Cluster micro-game titles, user feedback, and share captions using UMAP + HDBSCAN to form channels...

Simplified: We use techniques that find patterns in data to group similar items together automatically.