Google Cloud Firestore Use Case

Use Case: Building Real-Time Applications with Google Cloud Firestore

GENERAL CHARACTERISTICS

Intent	To enable real-time data synchronization and seamless scalability using Google Cloud Firestore.
Scope	Implementation of a document-oriented NoSQL database for high-performance, real-time applications.
Level	System-level.
Client	Confidential (Social Media Startup).
Last Update	03/12/2024
Status	Finalized.
Stage	Development and Deployment.

ACTORS

Primary Actor	Application Developer.
Secondary Actors	Mobile Developers, Backend Engineers, QA Team.

PREREQUISITES

Static Preconditions	- Google Cloud Project set up with Firestore API enabled. - Database schema and document structures designed based on application requirements.
Dynamic Preconditions	- Firestore security rules defined to manage access control. - Backend integrated with Firestore for data queries and updates.
Assumptions	- Client requires real-time updates for user-facing features. - Scalability is critical to handle increasing user traffic.

TRIGGERS

Trigger Event	The client needed a real-time database solution to enable live updates for user profiles and activity feeds.

EXPECTED OUTCOME

Success Postcondition	- Real-time updates are delivered seamlessly to user-facing features. - Firestore scales automatically to handle

	user growth.
Failed Postcondition	- Delays or inconsistencies in real-time updates impact user experience.

OPERATIONS AND CONCEPTS

Operations	1. Designed document structures for efficient querying and storage. 2. Implemented Firestore security rules to restrict data access. 3. Integrated Firestore SDK with the client application for real-time updates. 4. Created Firestore triggers using Cloud Functions for automated workflows. 5. Configured indexes to optimize query performance. 6. Monitored Firestore usage and adjusted scaling configurations.
Concepts	- Firestore: A document-oriented NoSQL database with real-time synchronization. - Security Rules: Define access permissions based on user roles. - Cloud Functions: Automate workflows triggered by Firestore events.

MAIN SUCCESS SCENARIO

Step 1	Analyzed the client's application needs for real-time data synchronization.
Step 2	Designed a document schema to optimize data retrieval and updates.
Step 3	Integrated Firestore with the application backend and mobile SDKs.
Step 4	Defined Firestore security rules for secure data access and updates.
Step 5	Created Firestore triggers for automated workflows using Cloud Functions.
Step 6	Monitored database performance and scaled configurations to handle traffic growth.
Step 7	Delivered a scalable and real-time data solution meeting client expectations.