# ***Product Requirements Document (PRD)***

### 

### Title

Stateless Push Notification Relay Service for CA360 ESP32 Devices

### Objective

To ensure real-time, reliable push notifications from ESP32 hubs to user mobile devices — even when the app is backgrounded or terminated (especially on iOS) — by implementing a stateless and scalable backend service that acts as a proxy relay via Firebase Cloud Messaging (FCM).

This solution will bridge the gap caused by iOS/Android background execution restrictions that prevent WebSocket connections from delivering events when the app is not in the foreground.

### Scope

This change introduces the following capabilities:

* A secure backend API that receives event triggers from ESP32 devices over HTTPS.
* A stateless, in-memory or Redis-based mapping of device\_id ↔ FCM token to ensure quick routing without persistence overhead.
* A client-side registration mechanism for the app to send its FCM token and associated device ID.
* Push notification delivery via Firebase Cloud Messaging (FCM) to Android and iOS apps, even in background/terminated states.
* Lightweight authentication mechanism for ESP32 devices (API key or JWT).
* Horizontally scalable deployment on Cloud Run, Fly.io, or similar platforms to support tens of thousands of concurrent devices.

### Functional Requirements

| Feature | Description |
| --- | --- |
| /event Endpoint | Accepts device\_id and event data from ESP32. Resolves associated FCM token and relays notification to the app via FCM. |
| /register Endpoint | Allows the mobile app to register/update its device\_id and fcm\_token. Mapping is cached (in-memory or Redis). |
| FCM Integration | Utilizes Firebase Admin SDK to reliably push notifications to registered Android/iOS devices. |
| Stateless Design | Operates without a persistent database. Token mappings exist only in RAM or Redis cache with TTL for cleanup. |
| Event Proxying | Backend acts as a relay — it does not store events, only forwards them to FCM instantly. |
| Lightweight Authentication | ESP32 devices must send requests with a valid API key or JWT in headers. |
| FCM Token Refresh | App is responsible for re-registering FCM tokens when updated/rotated. |

### Security Requirements

* All backend endpoints must be protected with HTTPS.
* ESP32 hubs must authenticate via a shared x-api-key or JWT to prevent abuse/spamming.
* The /register endpoint may optionally include a user-level auth token if needed for multi-user devices.
* Rate limiting and basic request logging will be implemented to prevent abuse and identify unusual activity.
* Cached token data will have an automatic TTL (e.g., 24 hours) to clean up stale mappings.

### Validation & QA

| Test Case | Desdescription |
| --- | --- |
| Event Relay Flow | ESP32 → Backend → FCM → App (foreground, background, terminated) is tested end-to-end. |
| Invalid/Expired Tokens | Notification failures due to expired tokens are gracefully handled (logged, no retry). |
| Unregistered Devices | If no mapping is found for device\_id, event is silently ignored or logged (based on config). |
| iOS Specific Tests | iOS 13+ devices in background/locked/terminated states must receive push notifications reliably. |
| Volume Test | Simulate concurrent events from 10k–100k devices (with autoscaling) to verify latency and delivery. |
| Cold Start & Token Refresh | App relaunch or FCM token change triggers a new /register call, ensuring the latest token is cached. |

### 

### System Architecture

#### Core Flow

1. Mobile app registers FCM token via /register → cached in backend.
2. ESP32 hub sends event data via /event → backend looks up token and relays notification via FCM.
3. User device receives push notification even if the app is backgrounded or terminated.

#### Infrastructure Stack

| Layer | Tech |
| --- | --- |
| Backend | FastAPI (Python) or Express (Node.js) |
| Messaging | Firebase Admin SDK (FCM) |
| Cache | Redis (Upstash and AWS ElastiCache) |
| Hosting | Cloud Run (GCP), Fly.io, or AWS Lambda (Zappa/FastAPI) |
| Auth | API Key or JWT-based auth |
| Monitoring | Cloud Logging, Sentry, or BetterStack (optional) |

### Project Timeline

| Phase | Task Description | Duration |
| --- | --- | --- |
| Phase 1 | Backend setup with FastAPI/Node + FCM integration | 1 days |
| Phase 2 | Redis or in-memory cache integration, testing FCM token mapping | 1 days |
| Phase 3 | Deploy backend to Cloud Run/Fly.io + autoscaling + CI pipeline | 1 days |
| Phase 4 | Frontend integration (register FCM token) + ESP32 HTTP client changes | 1 days |
| Phase 5 (optional) | Logging, rate limiting, monitoring dashboards | 1 day |

Total Estimate: 5 days (with buffer)

### Deliverables

* /register and /event API endpoints with OpenAPI spec or Postman collection.
* Working FCM push integration for both iOS and Android.
* Deploy-ready codebase with Dockerfile + deployment docs.
* Load test scripts for simulating ESP32 event traffic.
* ESP32 sample HTTP client code for testing.
* CI/CD setup for deploys (GitHub Actions or similar).
* Optional alerting/logging for push failures or traffic spikes.

### Optional Enhancements (For Scale / Production Maturity)

* FCM token retry queue with exponential backoff for failed pushes (via Redis Streams).
* Token auto-refresh handler on mobile (auto-detect token rotation via Firebase Messaging).
* Multi-region cache sync if deploying in multiple data centers.
* Observability Dashboard (requests per minute, delivery success rate, errors).