DIY SDK & CS Dashboard

2025 Highlights & 2026 Roadmap

Thanks to

Gani,
Rahul Karthik Naligala,
Hari Prasadh,
Prasath,
Revanth Venkata,
Suram Vishnu Vardhan Reddy,
Anbarasan Felix,
Bhanu Prakash,
Kavya,
Maroof Ansari,
Nithish,
Rohith Perala,
Gowtham Reddy

2025 Highlights (What did we ship?)

SDK

- Built and released the BLE SDK for both Android and iOS, which enables client apps to interact with Gen 3.0 YCUs.
- Integrated the SDK into Ops, OEM, and Yuma Partner apps.
- Added Jitsu Analytics to track swap sessions in real time, helping with monitoring and issue debugging.
- Developed OEM Manual Swap, where users scan the charged battery they received, ensuring the database is updated and the swap token is completed correctly.
- Consolidated the SDK's network architecture from 4 fragmented API clients into a unified system with dedicated OEM separation, improving performance and making it easier to add new API calls.

2025 Highlights (What did we ship?)

CS DASHBORAD - Alohomora

- Real-time Dashboard with State of Charge (SoC)
- Cloud Connect Integration for Customer Support.
- Remote Control System for Door Access and Other Operations
- Battery Type Count Monitoring
- Live Location Tracking for Different User Roles
- One-click Actions: Reset, Enable, Disable, Mark Faulty
- Easy call back for Missed calls

The Good, The Bad, The Ugly (Retrospective on code & culture)

The Good

SDK:

- BLE communication is more stable and reliable
- Auto-submit flow simplified the swap experience
- SDK is customisable and supports multiple development environments

CS Dashboard:

- Real-time updates via WebSocket

The Bad

SDK:

- Code is still tightly coupled and hard to reuse

CS Dashboard:

- No strict TypeScript

The Ugly

SDK:

- Swap availability check was done in frontend with hardcoded SOC

CS Dashboard:

- No error tracking/monitoring

Process Check (Where are we slowing down? How do we fix it?)

SDK

- When testing or integrating, failures were hard to debug because real-time logs were not visible in Metabase
- Understanding how users use new features was dependent on client apps releasing the latest SDK, which takes longer than expected

CS DashBoard

- Need automation in testing
- No pre-commit hooks, bad code gets committed. Fix: install husky and lint staged and setup pre commit

Under the Hood (Tech stack status, major refactors needed)

Tech Stack Status

SDK:

- Language: Kotlin 2.1.0 (Stable), Swift 5.
- BLE Engine: Native Android BLE APIs (Classic/Stock implementation).
- Networking: Retrofit 2.9.0 + OkHttp 4.9.1.
- Event Handling: SharedFlow (kotlin), Combine (swift) for yumaResponse event stream
- Architecture: Singleton ServiceLocator (Manual DI).

CS Dashboard:

- React + TypeScript (modern, scalable frontend stack)
- Real-time communication using Socket.IO (critical for live ops visibility)

Major Refactors Needed

SDK:

- Migrate to Modern DI: Replace manual ServiceLocator with Hilt or Koin to resolve God Object issues and improve testability.
- Initialization Safety: Remove Force Unwrap (!!) risks in core service initialization to prevent startup crashes.
- Set up a Jira project for the BLE SDK to better track updates.

CS Dashboard:

- Enable TypeScript strict mode to catch issues early and reduce runtime errors
- Break down large/monolithic components and standardize the API client layer to improve maintainability and ease of refactoring.

The Path Forward

SDK:

- Support multiple battery swaps across multiple YCUs
- Provide a complete UI through the SDK
- Make the codebase modular and easier to maintain
- Enable client apps to track and respond to door open and door close events, allowing clear, real-time user feedback in the UI.

CS Dashboard:

- Robust State Management and Testing to Improve System Stability and Release Confidence
- Performance and Scale Optimization - to support growing real-time operational usage.
- Breaking Down Large Single Files into Smaller, Reusable Components

One Big Idea

AI-First Customer Support

An AI agent that helps users complete swaps and escalates to human support only when required.