CCM — Continuous Delivery & Architecture

Music App — Cross-Platform (Flutter) · Clean Architecture · Firebase · CI/CD

CCM ⚡
CRAZY⚡ CRAZY MUSIC

1. Project Overview (Portfolio Summary)

Cross-platform music app (Android/iOS/Web-PWA) for the YouTube channel **CRAZY⚡ CRAZY MUSIC** — plays music via the YouTube IFrame Player in full ToS compliance · real-time chat (rooms/friends/DM) · VIP membership + COIN economy (buy gift items, convert to cash at 40%) · payment via **PromptPay QR + slip + admin approval** · real notification badges · i18n in 9 languages · Light/Dark theme · automated CI/CD on GitHub Actions

2. App Features & System Behavior

Feature	Details & How it works
🚀 **Splash + Onboarding + Login**	Splash loads ~2s → checks SharedPreferences (onboarded) → 3-page Onboarding (first launch only) → HomeShell with 6 tabs · login via **Google Sign-In → Firebase Auth** (GoogleAuthService as the central layer + switch/add accounts) · without login users can still play music, but chat/favorites/history are unavailable
🏠 **Home + Songs**	"Recommended" + "New songs" (last 10 days, with counts) are horizontally scrolling cards using YouTube thumbnails as backgrounds (purple gradient fallback) · song list split into All/Short/Live now/Live ended sections (empty sections hidden) · everything is a **Firestore Stream** — data changes update the screen instantly with no refresh
🎵 **Playback (YouTube embed)**	Tap a song → **YouTube IFrame Player** (bottom sheet / full-screen PlayerPage with YouTube API comments, top 20) · supports watch/youtu.be/shorts/embed/live/v links · audio-only mini player above the bottom nav · **no stream ripping / MP3 conversion** (ToS) · like/comment/live-chat buttons open the YouTube app (saves quota — API like code is ready but disabled at 50 units/call)
📱 **Shorts**	Songs in the short category use a vertical 9:16 grid → a YouTube Shorts-style page: video on top + description/👍 💬 🔧 bar below · swipe up/down to change songs · real description from the YouTube API · overlay mode is Web-only (Android/iOS use a native WebView overlay where touch events can't pass through to the video — a plugin limitation)
🔴 **Live**	Status checked via **YouTube Data API v3** automatically on load (no refresh button needed) + manual refresh button · badges ● LIVE / ended / upcoming · ended streams can be moved by the admin into "Live history" (and moved back) · live chat = button that opens YouTube (polling costs 5 units/call)
❤️ **Favorites + Watch History**	♥ button everywhere a song appears — stored per uid per Google account, switching accounts loads the new set automatically (auth-aware); not logged in = clears all · watch history: records every playback channel, **deduped by videoId**, newest first, limit 50 (Firestore users/{uid}/history)
💬 **Chat System (Rooms/Friends/DM)**	Real-time rooms: standard rooms cap 10 users · VIP rooms cap 20 + password (client-side check) · room creation restricted to admin+VIP · joining writes members/{uid}, leaving frees the slot · friends: request → accept → friendships · DMs stored in dms/{uidA_uidB}/messages (sorted chatId) with unlimited messages · read status via seenBy (arrayUnion, no duplicates) + timestamps · songs can be shared into chat (SongPicker + ▶ free preview) · **red badge = unread** (per friend + total)
🎁 **Gifts + Wallet (COIN economy)**	🎁 button in DM → pick one of 6 items (🌹 10 → 👑 2,500) → Firestore **runTransaction** deducts COIN from the sender (bought first → received) + adds received to the recipient + gift card in chat · recipient taps "Receive cash" to convert **40%** (kGiftCashRate, giftConverted prevents double redemption within the same transaction) · insufficient coins → opens the COIN purchase page
👑 **VIP Membership + COIN (Packages)**	2-tab popup (vip_packages_sheet): 👑 VIP membership ฿99/month or ฿899/year (perks: create 5 VIP rooms + badge + unlimited chat) · 🪙 COIN packs 100/500/1,200 with prices hardcoded in chat_repository (kVipPlans/kVipPackages — **verified by unit tests**)
💳 **PromptPay Payment + Slip**	Purchase → request doc vip_requests/{uid} (awaiting_slip) → full-screen page: PromptPay QR **generated in-app** (EMVCo payload + CRC16 with the amount embedded — fully unit-tested) → attach slip (image_picker Photo Picker → resized 540px/q30 → base64 ≤950KB in Firestore — no Storage) → tap "Send" (slip_sent) → approval popup (5–15 min) → slip is **locked, view-only** (full-screen zoom allowed)
🛡️ **Admin Approval + Audit**	Notification bar in chat + notifications page (tap item = slip dialog with approve/reject) · approving membership writes vips/{uid} · COIN increments wallets.bought · rejecting deletes the request · both cases: write a user noti + payment_history (admin history list + name/email search + rejected list)

🔔 **Notifications Page**	Bell 🔔 → NotificationsScreen · **real-time red number badge**: users = notis with read:false · admins = slip_sent requests not yet seen · tapping an item marks it read/seen and lowers the badge · users see request status + approval/rejection result (package+price+time)
🛡️ **Admin/VIP Status Shown Everywhere**	VipNameText (name + 🛡️ /🔥) · fancy UserAvatar: **spinning gradient ring** (SweepGradient, VIP gold-red / Admin blue-purple) + crown/shield above + VIP/ADMIN label + glow — resolved from streams (adminUids + vips) so any status change updates everywhere instantly · used in popups/rooms/DMs/friend list/You page
👤 **"You" Page + Settings**	Status+wallet cards with 4 real-time states (🛡️ Admin 🪙 ∞ / 🔥 VIP / ⏳ Pending / ✨ Regular) showing 🪙 total/bought/received/converted · menu: friends, my rooms, favorites, watch history, affiliate channels, song list (admin) · ⚙ full-screen settings: language (9 languages + search), theme, add song links (admin), help/terms/about
🌐 **i18n + Theme**	**9 languages** (Thai/EN/中文/日本語/한국어/ลาว/ဗမာ/Tiếng Việt/Melayu) — 250+ dictionary keys, S.t()/context.tr() throughout the app (no hardcoding), device default with Thai fallback, instant switching · Light/Dark theme follows system or manual choice — **every widget uses colorScheme** (CI scans for banned hardcoded dark colors)
🖥️ **Affiliate Channels + Song Rules (ToS)**	Only admins add songs · paste a link → title auto-fetched via oEmbed + auto type detection (shorts→Short, live→Live) + **no duplicates** (videoId check) · full-screen loading blocks taps · affiliate channels: add via any URL format validated through the API, no duplicates, disabling/deleting a channel hides its songs instantly (re-adding restores them) · video ownership validated only at add time

3. Architecture Structure — Clean Architecture with three layers, feature-first

```
lib/
├── main.dart                    # entry point: Firebase init, runApp
├── firebase_options.dart        # generated Firebase config
├── core/                        # cross-feature utilities
│   ├── app_config.dart            # API keys, PromptPay ID (single source of config)
│   ├── i18n.dart                  # dictionary i18n — 250+ keys × 9 languages, S.t()
│   ├── theme_controller.dart      # ThemeMode ValueNotifier (SharedPreferences)
│   ├── logger.dart                # ccsLog() — every error must be visible in the terminal
│   └── promptpay.dart             # PromptPay QR payload (EMVCo + CRC16, amount embedded)
├── models/                      # shared domain models
│   ├── song.dart                  # Song (videoId parser, thumbnails, category labels)
│   └── channel.dart               # Channel (handle, enabled, isOwner)
├── data/                        # data layer — Firebase lives only here
│   ├── repositories/
│   │   ├── song_repository.dart     # songs CRUD/streams, live history
│   │   ├── channel_repository.dart  # affiliate channels CRUD, owner seeding
│   │   └── chat_repository.dart     # rooms/messages, friends, DMs, VIP/COIN,
│   │                                # wallets (transactions), receipts, notis, badges
│   └── services/
│       ├── youtube_live.dart        # YouTube Data API (live, comments, validation)
│       ├── google_auth_service.dart # Google sign-in → Firebase, YouTube OAuth
│       ├── song_editor.dart         # add/edit/delete song + confirm dialog + isAdmin
│       ├── history_service.dart     # watch history per uid (dedupe by videoId)
│       ├── favorite_service.dart    # favorites per uid (auth-aware)
│       └── account_store.dart       # account switcher storage
└── features/                    # presentation layer, feature-first
    ├── shell/presentation/        # splash, onboarding, home shell (nav + top bar)
    ├── songs/presentation/        # home, category, shorts, songs, favorites,
    │   └── widgets/               #   history, live + song list/picker, mini player
    ├── chat/presentation/         # rooms, chat room, DM, friends, notifications,
    │   └── widgets/               #   VIP history + packages sheet, payment page,
    │                              #   status name text, fancy status avatar
    └── settings/presentation/     # you page, settings menu, channels, language,
                                   #   help, theme, account page
```

4. Architecture Decisions (7)

Decision	Rationale / Details
1. Stream-first data layer	Every real-time feature (songs, chat, rooms, notifications) is a Stream from the repository — screens use StreamBuilder with no manual refresh · screens never touch Firestore directly, only through repositories
2. Feature-first presentation	Four features (shell/songs/chat/settings) — cross-feature reuse only via models/ + data/, e.g. chat shares songs using the Song model + SongPicker without depending on the songs feature
3. Shared core/	config (single source of keys + PromptPay ID) · i18n dictionary · theme · logger · PromptPay generator — framework-free, callable from every feature

4. Transactions for Money	Gift sending / cash conversion uses Firestore runTransaction — deduct the sender's wallet (bought→received) and set giftConverted in one transaction, preventing negative balances / double redemption
5. Payment as a request workflow	No paid backend — a request is 1 doc (awaiting_slip → slip_sent), slips stored as base64 ≤950KB (validated before write) · approval writes vips/wallets + noti + payment_history — fully auditable from Firestore
6. Status identity via streams	UserAvatar/StatusIcon resolve uid → Admin (adminUids) / VIP (vips snapshot) then render a spinning ring + crown/shield + label — status changes propagate everywhere in real time
7. Defensive parsing + visible errors	Firestore reads via map access + ?? fallback (missing fields never throw) · errors always surface via ccsLog in the terminal + SnackBar · destructive actions require a confirm dialog

5. CD Pipeline — From Code to Users

1. Code scan analyze · format · secret · patterns · test → **2. Build** Web · APK (release) → **3. Deploy Web** Firebase Hosting + rules → **4. Release APK** GitHub Releases (ccm.apk) → **5. User Noti** install page gets the new version

6. CI — Code Scan / Clean-code Gate (5 stages, enforced on every push/PR)

Stage	What it checks	Fails when / fix
flutter analyze	static analysis across lib/	any single error/warning → fix per report
dart format check	every file must be properly formatted	`dart format lib` then re-commit
secret scan	private key / client_secret / API key outside app_config.dart	move to the single app_config or env
banned patterns	direct print() calls · hardcoded legacy dark colors (0xFF212121/2A2A2A)	use ccsLog() / colorScheme
flutter test	31+ unit test suites: PromptPay payload (EMVCo+CRC16) · Song videoId parsing · i18n (all keys verified in 9 languages) · chat constants (dmChatId/gifts/packages)	test failure → fix before merge

7. CD — Environment Deployment

Environment	URL	Source	Used by
Production Web	ccs-crazycrazymusic.web.app	auto-deployed from Develop/main	all users (PWA)
Android APK	github.com/ProgramZa2560/ccs-releases/releases/latest	auto-updated on every deploy	Android users
Dev	emulator Pixel_9_Pro · Xiaomi 2b3b02ff · iOS Sim	self-installed debug builds	development team

🔗 **Live links:** Install page https://ccs-crazycrazymusic.web.app/install.html · Web app (PWA) https://ccs-crazycrazymusic.web.app · APK https://github.com/ProgramZa2560/ccs-releases/releases/latest/download/ccm.apk

8. Rollback

- **Web**: Firebase Console → Hosting → Rollback version previous version (instant)
- **APK**: re-upload the old version to the GitHub Release
- **Rules**: deploy the rules files from the previous commit
- **Data**: Firestore is never rolled back — the schema must always be backward-compatible

9. Post-deploy Smoke Test

- Home loads songs · /install.html APK button works
- Google sign-in → send a chat message → bell/VIP status
- Check logs: Firebase Console / adb logcat

10. Tech Stack

Flutter / Dart	3 platforms from a single codebase
Firebase Auth	Google Sign-In
Cloud Firestore	real-time + transactions + security rules
Firebase Hosting	web app + install page
YouTube Data API v3	live status, comments, validation
GitHub Actions	CI/CD (scan → build → deploy)
qr_flutter · image_picker	PromptPay QR · slip upload (Photo Picker)

11. Required Pre-setup

✓	Item
✅	GitHub Actions ci.yml in the repo

✅	ccs-releases APK repo (public)
☐	FIREBASE_TOKEN secret (firebase login:ci)
☐	Real PromptPay ID (app_config.dart)
☐	Google Play Billing / App Store (real payments)

Known constraints: free Firebase Hosting (Spark) cannot host .apk → distributed via GitHub Releases · Firestore field limit 1MB (slips resized to 540px/q30) · iOS Simulator cannot play videos · stream ripping/MP3 conversion is prohibited (YouTube ToS) — playback via IFrame embed only

Next steps: Google Play internal testing (APK via the store) · add data/ layer tests (mocked Firestore — currently 31 unit suites: PromptPay/Song/i18n/chat constants) · Firebase App Check + API key → Cloud Functions · Privacy Policy + App Privacy labels · post-deploy notifications (Slack/LINE)