Fix It – Technical Plan

1) Overview

- Goal: Mobile app connecting users in Egypt with home-repair technicians (electrical, plumbing, carpentry, AC, etc.).
- Architecture: Clean Architecture (Presentation: Flutter+BLoC, Domain: Entities/UseCases/Repos, Data: Models/DataSources/Repo impls). DI via GetIt, networking via Dio/Retrofit, JSON serialization, local caching via shared_prefs/sqflite, secure storage via flutter_secure_storage. Backend REST+PostgreSQL. FCM for notifications. Arabic (RTL) first.
- Identity: Firebase Authentication only; backend verifies Firebase ID Tokens per request via Admin SDK; no backend JWTs, no refresh endpoints.

2) Authentication: Firebase Auth only

- Client (Flutter): Use Firebase Auth SDK (email/password; add phone sign-in as first-class for Egypt).
- Server (Backend):
  - Every protected request includes Authorization: Bearer <Firebase ID Token> over HTTPS.
  - Verify token on each request with Firebase Admin SDK verifyIdToken; extract uid for RBAC and data ownership.
  - Optionally cache decoded tokens briefly to reduce verification overhead on hot endpoints.
  - Use custom claims only for access control flags if needed; store all other user data in DB.
- Data model: Store firebase_uid UNIQUE on Users; email/phone optional to support phone-first onboarding for technicians.

3) Payments: Cash only (MVP)

- No payment gateway integration in MVP.
- UI: “Cash on completion” in booking confirmation.
- Backend:
  - Providers mark job completed; require client confirmation (two-step) to finalize cash status and reduce disputes/fraud; auto-complete after N hours with dispute_window flag for support intervention.
- DB:
  - Payments table remains for extensibility.
  - payment_method="cash"; transaction_status: pending→completed (after client confirmation)→refunded (manual if needed).

4) Proximity search (lat/lng + server-side distance filtering)

- Storage:
  - ServiceProviders: latitude, longitude (and last_location_updated_at).
  - Bookings: address_details_json includes latitude, longitude.
- API:
  - GET /providers?service_id=&search_query=&min_rating=&available_at=&lat=&lng=&radius_km=&max_price=&sort=distance|rating|price.
  - Response includes distance_km (rounded to 0.1km) and optional ETA_hint_minutes (simple heuristic, e.g., 25km/h intra-city) flagged as estimate_only.
- Query strategy:
  - Bounding box prefilter using lat/lng ranges, then Haversine to compute accurate distance, HAVING distance<=radius_km, ORDER BY distance when requested.
  - Keep a feature flag to switch to Postgres earthdistance or PostGIS later for spatial indexing/performance if volume grows.
    - earthdistance: ll_to_earth + earth_distance for simpler accurate queries.
    - PostGIS geography: ST_Distance and spatial indexes for scalability.

5) Core features (MVP)

- Users:
  - Browse categories/services; list nearby technicians with ratings/prices.
  - Book appointment: service, provider, date/time, address on map, notes.
  - Track booking status: pending, confirmed, in_progress, completed, cancelled.
  - Notifications for booking updates.
  - Rate provider after completion (verified booking).
- Providers:
  - Professional profile (profession, services, pricing, bio).
  - Coverage configuration (center and radius) and availability schedule.
  - Accept/decline jobs; update job status; view job address.
- Admin (light initially):
  - Manage categories/services.
  - KYC review and provider verification.
  - Monitor bookings, disputes, and timeouts.

6) Data model (PostgreSQL highlights)

- Users
  - user_id UUID PK, firebase_uid VARCHAR UNIQUE NOT NULL, full_name, email UNIQUE NULL, phone_number UNIQUE NULL, role ENUM('client','provider','admin'), profession, profile_picture_url, created_at, updated_at.[2]
- ServiceCategories
  - category_id UUID PK, name UNIQUE, icon_url.
- Services
  - service_id UUID PK, category_id FK, name, description, image_url, base_price.
- ServiceProviders
  - provider_id UUID PK (FK Users.user_id), bio, average_rating REAL default 0, total_ratings INT default 0, is_verified BOOLEAN default false, availability_json

JSONB, latitude REAL, longitude REAL, last_location_updated_at TIMESTAMP, coverage_center_lat REAL, coverage_center_lng REAL, coverage_radius_km REAL, kyc_status ENUM('pending','approved','rejected'), kyc_documents JSONB.
  - Indexes: composite (latitude, longitude); plus individual where helpful.
- ProviderServices (M2M)
  - provider_service_id UUID PK, provider_id FK, service_id FK, price REAL, UNIQUE(provider_id, service_id).
- Bookings
  - booking_id UUID PK, user_id FK, provider_id FK, service_id FK, status ENUM('pending','confirmed','in_progress','completed','cancelled','disputed'), booking_timestamp TIMESTAMP NOT NULL, scheduled_start_time TIMESTAMP NOT NULL, scheduled_end_time TIMESTAMP NULL, address_details_json JSONB NOT NULL (includes latitude, longitude), total_price REAL NOT NULL, payment_id FK NULL, user_rating_id FK NULL, provider_rating_id FK NULL, notes TEXT.
  - Audit/flow: expires_at TIMESTAMP, accepted_at, in_progress_at, completed_at, cancelled_at, cancelled_by ENUM('client','provider','system'), requires_client_confirmation BOOLEAN, client_confirmed_at TIMESTAMP.
  - Concurrency: prevent overlapping confirmed/in_progress bookings per provider via transactional checks; consider exclusion constraints with range types later.
- Payments (cash)
  - payment_id UUID PK, booking_id UUID UNIQUE FK, amount REAL, currency VARCHAR(3), payment_method ENUM('cash'), transaction_status ENUM('pending','completed','failed','refunded'), transaction_timestamp TIMESTAMP, external_transaction_id VARCHAR NULL.
- Ratings
  - rating_id UUID PK, booking_id FK NOT NULL, rater_user_id FK NOT NULL, rated_entity_id UUID (provider_id), rating_value INT CHECK 1..5, comment TEXT, rating_timestamp default now(), verified_booking BOOLEAN default true; UNIQUE(booking_id, rater_user_id).
- Notifications

    - notification_id UUID PK, user_id FK, title, body, type ENUM('booking_confirmation','review_request','booking_reschedule','app_update','special_offer','payment_success'), timestamp default now(), is_read BOOLEAN default false, target_data_json JSONB includes type, ids, version.
- AppSettings, FAQs as previously defined.

7) Indexes (key)

- Users: idx_users_firebase_uid, idx_users_email, idx_users_phone_number, idx_users_role.
- ServiceProviders: composite idx_providers_lat_lng(latitude, longitude), idx_providers_verification, idx_providers_rating.
- Services: idx_services_category_id, idx_services_name.
- ProviderServices: idx_provider_services_provider_id, idx_provider_services_service_id.
- Bookings: idx_bookings_user_id, idx_bookings_provider_id, idx_bookings_status, idx_bookings_scheduled_start_time, idx_bookings_expires_at.
- Payments: idx_payments_booking_id, idx_payments_transaction_status.
- Ratings: idx_ratings_rated_entity_id, idx_ratings_rater_user_id.
- Notifications: idx_notifications_user_id, idx_notifications_timestamp, idx_notifications_is_read.
- Optional later: Postgres earthdistance or PostGIS indexes for geo queries.

8) API contract (refined)

- Auth
    - All protected endpoints require Authorization: Bearer <Firebase ID Token>; backend verifies each call via Admin SDK verifyIdToken.[5][1]
    - GET /auth/me → returns current DB user resolved from token for client bootstrap convenience.
- Services
    - GET /service-categories
    - GET /services

    - GET /services/{service_id}
- Providers
    - GET /providers?service_id=&search_query=&min_rating=&available_at=&lat=&lng=&radius_km=&max_price=&sort=distance|rating|price → includes distance_km and availability_preview.
    - GET /providers/{provider_id}
- Bookings
    - POST /bookings → validates provider coverage and time-slot conflicts; creates booking status=pending; returns expires_at for acceptance window.
    - PUT /bookings/{id}/accept (provider), PUT /bookings/{id}/decline (provider).
    - PUT /bookings/{id}/start (provider), PUT /bookings/{id}/complete (provider).
    - PUT /bookings/{id}/confirm-completion (client) → finalizes cash completion.
    - PUT /bookings/{id}/cancel (client/provider/admin).
    - PUT /bookings/{id}/dispute (client/provider) within window.
    - GET /users/{user_id}/bookings?status=upcoming|past|all
    - GET /bookings/{booking_id}
- Payments (cash flow only)
    - PUT /bookings/{id}/complete-payment-cash → internally set pending client confirmation; finalization via confirm-completion.
- Profile & settings
    - GET/PUT /users/{user_id}/profile
    - GET/PUT /users/{user_id}/app-settings
- Notifications
    - GET /users/{user_id}/notifications
    - PUT /notifications/{notification_id}/read
- Ratings

- POST /ratings { booking_id, rated_entity_id, rating_value 1..5, comment? } → only for completed bookings (verified_booking=true).
- GET /providers/{provider_id}/ratings

9) Booking lifecycle and SLAs

- Acceptance window: provider must accept/decline within X minutes; otherwise system auto-cancels or reassigns; notify with FCM deep links.
- Status transitions:
    - pending → confirmed (accept) → in_progress (start) → completed (complete) → payment completed after client confirmation.
    - Any stage can move to cancelled with cancelled_by recorded; disputes allowed within a window.
- Audit timestamps for all transitions to support support workflows and analytics.

10) Availability and double-booking prevention

- availability_json schema:
    - Weekly blocks with day_of_week, start_time, end_time; exceptions list for blackout dates or special windows; backend validates schema and overlapping ranges.
- Double-book prevention:
    - Transactional query ensures no overlapping confirmed/in_progress bookings for the provider in the requested window.
    - Consider Postgres range types and exclusion constraints once stabilizing.

11) Provider verification, coverage, and safety

- KYC workflow: upload national ID and skill proof; kyc_status pending→approved/rejected; only approved providers are surfaced prominently.
- Coverage enforcement: coverage_center_lat/lng + coverage_radius_km; block bookings outside radius unless provider explicitly accepts (future: surcharge).
- is_verified badge and filter; emergency contact field; optional trade license upload.

12) Notifications and conventions

- target_data_json includes {type: "booking" | "rating" | "offer", booking_id?, provider_id?, version}.
- Scheduled jobs handle acceptance timeouts and auto-completions; all changes push FCM notifications with deep-links.

13) Security

- HTTPS everywhere; consider SSL pinning in app later.
- Server-side token verification on every protected call; never trust client-only claims.
- Store tokens securely in flutter_secure_storage; do not embed secrets.
- Input validation and RBAC on server.
- Structured audit logging for booking and payment transitions (actor_uid, IP/device if available).

14) Performance and scalability

- Frontend: const widgets, builder lists, image compression/lazy loading; pagination and caching.
- Backend: indexes as above; bounding-box prefilter before Haversine; optional upgrade to earthdistance or PostGIS when scale requires it.
- Token verification: consider short-lived cache of decoded tokens to reduce overhead while keeping correctness.

15) Testing and operations

- Unit tests: UseCases, Repositories, BLoCs with mocks; focus on booking lifecycle, availability validation, and provider search filtering.
- Widget tests for key flows; integration tests for end-to-end: auth→search→booking→accept→complete→confirm→rate.
- CI: run tests + static analysis on PRs; error logging via Crashlytics/Sentry for clients and structured logs on server.

- Admin panel (read-first, then write): KYC review, booking monitors, disputes, status overrides with audit logs.

16) Roadmap (6-week MVP)

- Week 1: Project setup; Firebase Auth integration; Users(firebase_uid) model; /auth/me; base services/categories.
- Week 2: Providers listing with proximity (bounding-box + Haversine), provider profile; composite lat/lng index.[7][8]
- Week 3: Booking flow; coverage and slot conflict checks; acceptance window and expires_at.
- Week 4: Provider dashboard (accept/decline/start/complete); FCM notifications; user bookings list.
- Week 5: Ratings (verified booking); profile/settings; KYC review basics; fraud controls (two-step completion).
- Week 6: RTL QA; security hardening (token verification paths, logging); performance; pilot release.

17) Implementation snippets (server-side proximity)

- Bounding box parameters:
  - dLat = radius_km/111; dLng = radius_km/(111*cos(radians(:lat))).
- Haversine select (km) example:
  - distance_km = 6371*acos(cos(radians(:lat))*cos(radians(latitude))*cos(radians(longitude)-radians(:lng))+sin(radians(:lat))*sin(radians(latitude))).
- Postgres alternatives:
  - earthdistance: earth_distance(ll_to_earth(:lat,:lng), ll_to_earth(latitude,longitude)).
  - PostGIS: ST_Distance(geography, ST_MakePoint(:lng,:lat)::geography) with spatial index if upgraded.