TherapyAI - Development Guide

Version: 1.0
Date: 2024-03-10

Table of Contents

- Networking (Retrofit, OkHttp, API Service)
- Security (HIPAAKeyManager, AESUtil, EphemeralPrefs)
- UI Adapters
- Background Services (Recording, Notifications)

6. Feature Implementation Details
    - 6.1 Therapist Workflow
        - Session Card Management (CRUD, Ordering, Filtering)
        - Category Management
        - Starting & Conducting Sessions (SessionHostActivity Navigation)
        - Audio Recording Flow
        - Note Taking & Summary
        - Session Review & Submission
        - Searching Profiles & Sessions
        - Viewing Profile Details (Charts & Syncing)
        - Viewing Session Details (Charts & Syncing)
        - Inbox & Transcript Review/Editing
    - 6.2 Patient Workflow
        - QR Code Generation
        - Viewing History & Profile
    - 6.3 Common Features
        - Settings (Theme, Notifications)
        - Navigation (Bottom Nav, ViewPager, Nav Component)
7. Adding New Features
    - Example: Adding a New Session Type (e.g., VR)
    - General Steps
8. Testing

9. Coding Conventions & Best Practices
10. Troubleshooting
11. Appendix

1. Introduction

- **Purpose of the Guide:** This document serves as the primary technical guide for developers working on the TherapyAI Android application. It outlines the project's architecture, key components, data flow, and provides guidance on how to maintain, modify, and extend the application.
- **Application Overview:** TherapyAI is a dual-role application assisting therapists and patients. Therapists manage session templates, conduct sessions (currently audio-focused), review processed data, and track patient history. Patients generate secure check-in codes and view their history. The application aims for security (HIPAA considerations) and a streamlined user experience. The data is processed by AI models which sit on an azure cloud.

2. Project Setup

- **Prerequisites:**
    - Android Studio (Latest stable version recommended)
    - Git
    - Android SDK (Check build.gradle for minSdk and targetSdk)
    - Emulator or Physical Android Device
- **Unpacking the ZIP file:**
- Extract files at a desired directory.
- cd to the folder.

- **IDE Setup:** Open the cloned project directory in Android Studio. Allow Gradle to sync dependencies automatically.
- **Dependencies:** Key dependencies include:
    - AndroidX Libraries (AppCompat, RecyclerView, ViewPager2, Lifecycle, Navigation, Room, Biometric)
    - Material Components
    - Retrofit & Gson (Networking & JSON parsing)
    - OkHttp (HTTP Client, including TokenAuthenticator)
    - Glide (Image loading)
    - MPAndroidChart (Charting)
    - ZXing (QR Code Generation/Scanning)
    - Firebase Messaging (Push Notifications)
    - viewpagerdotsindicator (UI)
    - *(Review app/build.gradle for the complete list)*
- **API Configuration:**

    - **Base URL:** The base URL for the backend API is defined in ApiServiceProvider.java. **Action:** Update BASE_URL to point to the correct deployment environment (development, staging, production).
    - **API Keys:** If the API requires keys (e.g., for Firebase, other services), ensure they are configured correctly, ideally using local.properties and build.gradle to avoid committing secrets. Placeholder needed.
- **Running the App (Mock vs. Real Data):**
    - The application initialization in TherapyAIApp.java's initRepositories method uses a useMockData flag.
    - **Action:** Set useMockData to false to connect to the real backend API. Set it to true to use the mock implementations within TherapyApiImpl.java for offline development or testing UI flows without a backend.

3. Architecture Overview (MVVM)

The project generally follows the Model-View-ViewModel (MVVM) architectural pattern, promoting separation of concerns, testability, and maintainability.

- **Model:** Represents the data.
    - **Local:** Room Database entities (CardItem, CategoryItem), SharedPreferences data. Located in data.local.models.
    - **Remote:** Data Transfer Objects (DTOs) used by Retrofit for API communication (e.g., LoginResponse, ProfileResponse, SessionSummaryResponse). Located in data.remote.models.
- **View (Activities & Fragments):** Responsible for displaying the UI and capturing user input. They observe LiveData from ViewModels and delegate user actions to the ViewModel. They should contain minimal business logic.
    - *Examples:* MainActivity, TherapistSessionFragment, AudioRecordFragment, ProfileActivity. Located in ui.* packages.
- **ViewModel:** Acts as a bridge between the View and the Repository. It prepares and manages UI-related data (using LiveData), handles user actions, and interacts with the Repository to fetch or save data. ViewModels survive configuration changes.
    - *Examples:* CardViewModel, SessionViewModel, DataViewModel, TranscriptDetailViewModel. Located in ui.viewmodels.
- **Repository:** Abstracts the data sources (local database, remote API, mock data). ViewModels interact with Repositories to get data, without needing to know *where* the data comes from. Repositories handle the logic of fetching from cache, DB, or network.
    - *Examples:* CardRepository, CategoryRepository, AuthRepository, SearchRepository, RecordingRepository, ProcessedDataRepository, NotificationRepository. Located in data.repository.
- **Data Sources:**
    - **Local:**
        - Room Database (AppDatabase): Persistent storage for structured data like Session Cards and Categories. Accessed via DAOs (CardDao, CategoryDao). Located in data.local.db and data.local.dao.

        - SharedPreferences:
            - LocalStorageManager: For non-sensitive app settings (theme, notification preferences).
            - EphemeralPrefs: For sensitive session data (auth tokens, user info), encrypted using Android Keystore. Located in data.local.
    - **Remote:**
        - Retrofit (ApiServiceProvider, TherapyApiService): Handles network requests to the backend API. Includes TokenAuthenticator for automatic token refresh. Located in data.remote.
        - TherapyApiImpl: Implementation layer that directs calls to either the real TherapyApiService or mock data methods based on the useMockData flag.
- **Dependency Management:** Primarily uses manual dependency injection via Singleton patterns for Repositories and utility classes (SessionManager, EphemeralPrefs, etc.), initialized in TherapyAIApp.

4. Project Structure (Key Packages & Files)

```
therapy-ai-android
├── build.gradle.kts                # Top-level build configuration (plugin management)
├── settings.gradle.kts             # Gradle project settings (repositories, modules)
├── gradle.properties               # Project-wide Gradle configuration (AndroidX, JVM settings)
├── gradlew                         # Gradle wrapper executable (Unix)
├── gradlew.bat                     # Gradle wrapper executable (Windows)
├── local.properties                # Local SDK configuration (auto-generated)
├── log.out                         # Application logs
└── app/                            # Main Android application module
    ├── build.gradle.kts            # App-level build configuration (dependencies, compile settings)
    ├── proguard-rules.pro          # Code obfuscation and optimization rules
    └── src/
        ├── main/
        │   ├── AndroidManifest.xml                 # App manifest (permissions, activities, services)
        │   ├── java/com/example/therapyai/
        │   │   ├── TherapyAIApp.java                # Application class (initialization, repositories)
        │   │   ├── BaseSignedActivity.java         # Base authenticated activity (session management, security)
        │   │   ├── BaseSessionActivity.java        # Base session activity (HIPAA compliance, secure recording)
        │   │   ├── data/                             # Data Layer
        │   │   │   ├── AudioGraphPoint.java        # Audio visualization data model
        │   │   │   ├── local/                      # Local Data Management
        │   │   │   │   ├── EphemeralPrefs.java     # Encrypted secure preferences (HIPAA-compliant storage)
        │   │   │   │   ├── LocalStorageManager.java  # Theme and app settings management
        │   │   │   │   ├── SessionManager.java     # User session, authentication, and timeout management
        │   │   │   │   ├── dao/                    # Room database access objects
        │   │   │   │   ├── db/                     # Room database configuration and converters
        │   │   │   │   └── models/                 # Local entity models (cards, categories)
        │   │   │   ├── remote/                      # Remote API Layer
        │   │   │   │   ├── TherapyApiImpl.java     # Main API implementation (with mock data support)
        │   │   │   │   ├── TherapyApiService.java  # Retrofit API interface definitions
        │   │   │   │   ├── ApiServiceProvider.java  # API service configuration and client setup
        │   │   │   │   ├── NotificationService.java  # Firebase Cloud Messaging service
        │   │   │   │   └── models/                 # API request/response models
        │   │   │   └── repository/                 # Repository Pattern Implementation
        │   │   │       ├── AuthRepository.java     # Authentication operations (login, password reset)
        │   │   │       ├── CardRepository.java     # Session card CRUD operations
        │   │   │       ├── CategoryRepository.java   # Category management operations
        │   │   │       ├── NotificationRepository.java # FCM device registration and management
        │   │   │       ├── ProcessedDataRepository.java # Therapy session data processing
        │   │   │       ├── RecordingRepository.java  # Audio session upload and submission
        │   │   │       └── SearchRepository.java    # Patient/session search functionality
        │   │   ├── ui/                             # User Interface Layer
        │   │   │   ├── MainActivity.java           # Main app activity (navigation, user type routing)
        │   │   │   ├── SplashActivity.java         # App startup and auto-login
        │   │   │   ├── adapters/                   # RecyclerView adapters for lists and UI components
        │   │   │   ├── browse/                     # Browse Module (Therapist Features)
        │   │   │   │   ├── AccountActivity.java    # User account management
        │   │   │   │   ├── InboxActivity.java      # Processed session inbox
```

```
│   │   │   │   ├── AboutActivity.java        # App information and credits
│   │   │   │   ├── SupportActivity.java      # Help and support features
│   │   │   │   ├── ProcessedDataDetailActivity.java # Session transcript review and
editing
│   │   │   │   └── NotificationSettingsActivity.java # Push notification preferences
│   │   │   ├── dialogs/                      # Custom dialog implementations
│   │   │   ├── patientQR/                    # QR code patient identification system
│   │   │   ├── search/                       # Search and Data Module
│   │   │   │   ├── ProfileActivity.java      # Patient profile viewing
│   │   │   │   └── DataActivity.java         # Session data browsing and analytics
│   │   │   ├── sessions/                     # Therapy Session Module (Core Business
Logic)
│   │   │   │   ├── FormActivity.java         # Session metadata form
│   │   │   │   ├── ScanQRActivity.java       # Patient QR code scanning
│   │   │   │   ├── QRResultActivity.java     # QR scan result processing
│   │   │   │   ├── defaultAudio/              # Audio Recording Subsystem
│   │   │   │   │   ├── AudioRecordActivity.java # Audio recording interface
│   │   │   │   │   ├── RecordingService.java # Real-time encrypted audio recording
service
│   │   │   │   │   └── SendActivity.java     # Session upload and submission
│   │   │   │   └── session/                  # Session Management
│   │   │   │   └── SessionHostActivity.java # Main session orchestration activity
│   │   │   ├── viewmodels/                   # MVVM ViewModels for UI state management
│   │   │   ├── views/                        # Custom view components
│   │   │   └── welcome/                      # Authentication Module
│   │   │       ├── WelcomeActivity.java      # App onboarding and login flow
│   │   │       └── LoginActivity.java        # User authentication interface
│   │   └── util/                             # Utility Classes
│   │       ├── AESUtil.java                      # HIPAA-compliant encryption
utilities (file/data encryption)
│   │       ├── AppStateTracker.java              # Application state monitoring
│   │       ├── AudioFormatUtil.java              # Audio processing and format
conversion
│   │       ├── DateInputMask.java                # Date input formatting
│   │       ├── DialogUtils.java                  # Common dialog utilities
│   │       ├── Event.java                    # Event-driven architecture support
│   │       ├── HIPAAKeyManager.java              # Android Keystore management
(encryption keys)
│   │       ├── InAppNotificationManager.java # In-app notification system
│   │       ├── NotificationHelper.java           # System notification
management
│   │       ├── ProfilePictureUtil.java           # Profile image loading and
caching
│   │       └── SentimentChartHelper.java         # Therapy session sentiment
visualization
│   └── res/                                  # Android Resources
│       ├── layout/                           # XML layout files for all activities and
fragments
│       ├── drawable/                         # Vector graphics, icons, and visual
assets
│       ├── values/                           # Strings, colors, dimensions, and themes
│       ├── mipmap-*/                         # App icons for different screen densities
│       ├── anim/                             # Animation definitions
│       ├── color/                            # Color state lists and selectors
│       ├── menu/                             # Navigation and context menus
│       ├── navigation/                       # Navigation graph definitions
│       └── xml/                              # Configuration files
├── androidTest/                              # Instrumented tests (UI and integration
tests)
└── test/                                     # Unit tests
└── gradle/                                   # Gradle wrapper configuration
    └── wrapper/                              # Gradle wrapper JAR and properties
```

Key Technologies & Architecture:

- Android (Java): Native Android app with Material Design 3
- MVVM Architecture: ViewModels, LiveData, and Repository pattern
- Room Database: Local SQLite database for cards, categories, and caching
- Retrofit + OkHttp: REST API communication with therapy-ai-backend
- Firebase Cloud Messaging: Push notifications for session updates
- Android Keystore + AES-GCM: HIPAA-compliant encryption for audio and data
- Real-time Audio Encryption: Streaming encryption during recording to prevent plaintext storage
- Biometric Authentication: Device credential verification for sensitive operations
- ViewPager2 + Navigation: Tab-based navigation with fragment management

Security & HIPAA Compliance:

- All sensitive data encrypted at rest using Android Keystore
- Real-time audio encryption prevents plaintext audio files
- Session timeout and automatic logout (15-minute absolute timeout)
- Screen capture prevention (FLAG_SECURE on all activities)
- Secure key deletion on logout and app termination
- User authentication required for key access (12-hour validity)

User Roles & Core Workflow Overview (see next section for more details):

- THERAPIST WORKFLOW:
    - Login → MainActivity with therapist-specific navigation
    - Session creation: Scan patient QR code or manual entry
    - Real-time encrypted audio recording with amplitude visualization
    - Session metadata collection (notes, categories, patient info)
    - Encrypted audio upload to backend with session data
    - Backend processes: STT transcription + sentiment analysis
    - Notification when processing complete → Transcript review in Inbox
    - Edit and finalize transcripts → Submit final session data
    - Analytics: View patient progress and sentiment trends
- PATIENT WORKFLOW:
    - Login → MainActivity with patient-specific navigation
    - QR code generation for therapist identification
    - View own session history and progress
    - Access final session summaries and progress reports
    - Analytics: Personal therapy progress visualization

- DUAL-MODE FEATURES:
    - Real-time encrypted recording service (background operation)
    - Session state management (pause/resume/timer tracking)
    - In-app notifications for session updates
    - Profile management and theme selection
    - Secure logout with key cleanup

Backend Integration:

- REST API communication with therapy-ai-backend Flask server
- Mock data support for offline testing and development
- Automatic device registration for push notifications
- Session upload with metadata (patient info, notes, timestamps)
- Processed data retrieval (transcripts, sentiment scores)
- Search functionality for patients and sessions

Development Features:

- ProGuard configuration for release builds
- Comprehensive logging and diagnostic tools
- Modular architecture for maintainability
- Mock data system for testing without backend
- Material Design 3 theming with dark/light mode support

5. Core Components & Modules

- **Authentication Flow:**
    - SplashActivity: Checks EphemeralPrefs via AuthRepository.loginFromMemory. Navigates to MainActivity on success, WelcomeActivity otherwise.
    - WelcomeActivity: Entry point for non-logged-in users. Leads to LoginActivity.
    - LoginActivity: Handles email/password input, Terms agreement, Biometric/Device Credential authentication (BiometricPrompt, KeyguardManager), calls AuthRepository.login, stores tokens/user info in EphemeralPrefs via SessionManager, navigates to MainActivity.
    - AuthRepository: Makes API calls (TherapyApiImpl) for login, password reset.
    - SessionManager: Manages user session state (validity, timeouts), provides access to logged-in user details from EphemeralPrefs.
    - EphemeralPrefs: Stores sensitive session data (tokens, user ID, etc.) encrypted using AESUtil and a master key from HIPAAKeyManager.
    - BaseSignedActivity: Base class for activities requiring login. Enforces FLAG_SECURE, handles session timeout warnings (SessionTimeoutListener), and re-authentication prompts (ReAuthListener).
- **Session Management (SessionManager):**
    - Handles inactivity timeouts (14 min warning via SessionTimeoutListener, 15 min force logout).
    - Tracks background time and forces logout if app is backgrounded for > 15 min.
    - Provides methods to get current user info (getUserId, getUserType, etc.).
    - Central point for initiating forceLogout.
    - Handles re-authentication requests via ReAuthListener.
- **Data Storage (Local):**
    - Room: (AppDatabase, CardDao, CategoryDao) Used for persistent storage of therapist's Session Cards and Categories. Operations are performed via

CardRepository and CategoryRepository on a background thread (databaseExecutor). Default entries are added in TherapyAIApp.

    - LocalStorageManager: Standard SharedPreferences for non-sensitive settings like theme mode and notification preferences.
    - EphemeralPrefs: Encrypted SharedPreferences for tokens and user session data, tied to the Android Keystore key.

- **Networking:**
    - Retrofit: Used for defining and executing API calls (TherapyApiService). Instance provided by ApiServiceProvider.
    - OkHttp: Underlying HTTP client. Configured in ApiServiceProvider with TokenAuthenticator and TLS specs.
    - TherapyApiService: Interface defining all API endpoints (@GET, @POST, etc.).
    - TherapyApiImpl: Acts as a facade, deciding whether to call the real TherapyApiService or return mock data based on the useMockData flag. Contains ApiCallback for handling responses.
    - TokenAuthenticator: OkHttp Authenticator that intercepts 401 responses, calls the /auth/refresh endpoint using EphemeralPrefs.getRefreshToken(), stores new tokens, and retries the original request.
- **Security:**
    - HIPAAKeyManager: Manages a master encryption key within the Android Keystore (AndroidKeyStore). This key requires user authentication (biometric/device credential) to use and has a validity duration (e.g., 12 hours). It's used by EphemeralPrefs.
    - AESUtil: Provides utility methods for AES/GCM encryption/decryption (used by EphemeralPrefs) and file encryption/decryption (used by RecordingRepository). Includes a secureDelete method for overwriting files before deletion.
    - EphemeralPrefs: Encrypts all stored data using the key from HIPAAKeyManager. Handles UserNotAuthenticatedException by requesting re-authentication via SessionManager.

    - BaseSignedActivity/BaseSessionActivity: Apply WindowManager.LayoutParams.FLAG_SECURE to prevent screenshots and screen recording of sensitive screens.
    - SessionManager: Enforces inactivity and background timeouts.
- **UI Adapters:** Located in ui.adapters. Standard RecyclerView.Adapter implementations for various lists (Session Cards, Categories, Search Results, Transcript Entries, Notes, etc.). Adapters often include interfaces for click/interaction listeners. Pay attention to adapters handling complex states like SessionCardAdapter (edit mode, selection, drag/drop) and FinalTranscriptAdapter/ProfileSessionAdapter (highlighting, payloads for partial updates).
- **Background Services:**
    - RecordingService: A foreground service responsible for handling audio recording using MediaRecorder. Manages recording state (recording, paused), provides amplitude updates, handles notifications, and manages the lifecycle of the recording file. Communicates with AudioRecordFragment via binding (ServiceConnection).
    - NotificationService: A FirebaseMessagingService that receives push notifications from FCM. It parses the message payload, determines the notification type (e.g., SESSION_READY), builds an appropriate PendingIntent (e.g., to open InboxActivity or ProcessedDataDetailActivity), and uses NotificationHelper to display the notification according to user preferences (LocalStorageManager). It also handles FCM token refreshes by calling NotificationRepository.updateDeviceToken.

6. Feature Implementation Details

6.1 Therapist Workflow

- **Session Card Management (CRUD, Ordering, Filtering):**
    - **View:** TherapistSessionFragment
    - **ViewModel:** CardViewModel, CategoryViewModel
    - **Repository:** CardRepository, CategoryRepository
    - **Adapter:** SessionCardAdapter
    - **Dialog:** SaveCardDialogFragment
    - **Flow:**
        - Fragment observes CardViewModel.cardListLiveData and CategoryViewModel.categoryListLiveData.
        - Filtering (Category/Search): CategoryAdapter click calls onCategorySelected -> Fragment updates selectedCategory -> calls CardViewModel.loadCards(category). Search view listener calls SessionCardAdapter.filter(query).
        - Add/Edit: FAB/+ button or card edit icon -> shows SaveCardDialogFragment. Dialog collects data -> calls CardViewModel.saveCard(card). ViewModel calls CardRepository.saveCard -> Repo updates DB -> ViewModel reloads cards for the current category (loadCardsInternal) -> LiveData updates UI.
        - Delete: Swipe action (if configured) or Edit Mode delete icon -> shows confirmation dialog -> calls CardViewModel.deleteCard or deleteSelectedCards -> Repo updates DB -> ViewModel reloads cards.
        - Reorder (Edit Mode): ItemTouchHelper in Fragment detects drag -> calls SessionCardAdapter.swapItems (updates adapter list visually) -> clearView in ItemTouchHelper callback is called on drop -> calls CardViewModel.updateCardOrder(currentOrder) -> Repo updates positions in DB.
- **Category Management:**

- **View:** TherapistSessionFragment
- **ViewModel:** CategoryViewModel, CardViewModel
- **Repository:** CategoryRepository, CardRepository
- **Adapter:** CategoryAdapter
- **Flow:**
  - Add: "+" button click -> shows add category dialog -> calls CategoryViewModel.addCategory(name) -> Repo checks existence, inserts if new -> ViewModel reloads categories.
  - Edit: Long press category -> shows edit/delete menu -> select Edit -> shows rename dialog -> calls CategoryViewModel.renameCategory(oldName, newName) -> ViewModel coordinates with CardRepository (renameCategoryInCards) and CategoryRepository (updateCategory) -> ViewModel reloads categories.
  - Delete: Long press category -> shows edit/delete menu -> select Delete -> shows confirmation -> calls CategoryViewModel.deleteCategoryAndAssociations(category) -> ViewModel coordinates with CardRepository (removeCategoryFromCards) and CategoryRepository (deleteCategory) -> ViewModel reloads categories.

- **Starting & Conducting Sessions (SessionHostActivity Navigation):**
  - **Trigger:** Tapping a card in TherapistSessionFragment.
  - **Host:** SessionHostActivity
  - **ViewModel (Shared):** SessionViewModel (scoped to SessionHostActivity)
  - **Navigation:** session_nav_graph.xml defines the flow (Form -> Record -> Summary -> Review). NavController within SessionHostActivity manages transitions.
  - **Data Passing:** Initial CardItem is passed as an argument to the NavGraph/SessionViewModel. Subsequent data (patient info, file paths, notes) is stored and shared via the SessionViewModel.
- **Audio Recording Flow:**

    - **View:** AudioRecordFragment
    - **ViewModel:** SessionViewModel (Shared)
    - **Service:** RecordingService
    - **Flow:** Fragment binds to RecordingService. User taps Start -> Consent -> startActualRecording called -> Service starts MediaRecorder, runs as foreground service, provides amplitude updates. Pause/Resume toggles service state. Stop -> Confirmation -> stopActualRecordingAndProceed called -> Service stops, Fragment gets file path -> stores path in SessionViewModel -> navigates to next step (AddSummaryFragment). Notes added via dialog call SessionViewModel.saveOtherNote. Back press during recording triggers cancellation confirmation.
- **Note Taking & Summary:**
    - **Views:** AudioRecordFragment, AddSummaryFragment, ReviewSubmitFragment
    - **ViewModel:** SessionViewModel
    - **Adapter:** NoteCardAdapter
    - **Flow:** AudioRecordFragment adds timestamped/generic notes to SessionViewModel.otherNotes. AddSummaryFragment updates SessionViewModel.summaryNote. ReviewSubmitFragment observes SessionViewModel.combinedNotes (MediatorLiveData combining summary and others) and displays them using NoteCardAdapter. Editing notes in ReviewSubmitFragment updates the respective LiveData in the ViewModel.
- **Session Review & Submission:**
    - **View:** ReviewSubmitFragment
    - **ViewModel:** SessionViewModel
    - **Repository:** RecordingRepository
    - **Flow:** Fragment displays combined notes. User taps "Send Session" -> SessionViewModel.requestSubmit called -> triggerSubmit LiveData fires -> SessionHostActivity observes triggerSubmit -> calls uploadDataBasedOnSessionType. Activity gets file path(s) and notes from ViewModel -> calls RecordingRepository.uploadRecordingSession -> Repo encrypts file (AESUtil, HIPAAKeyManager), prepares metadata/multipart

request -> calls TherapyApiImpl.uploadSessionData. On success/failure, callback updates ViewModel/UI -> Activity shows dialog -> cleans up files (AESUtil.secureDelete) -> finishes. Delete flow is similar via triggerDelete.

- **Searching Profiles & Sessions:**
    - **View:** TherapistSearchFragment
    - **Repository:** SearchRepository
    - **Adapter:** SearchAdapter (handles multiple view types - headers, profile, session)
    - **Flow:** User enters query -> performSearch called -> SearchRepository called for both profiles and sessions -> ApiImpl makes API calls -> Repo converts responses (ProfileResponse, SessionSummaryResponse) to local models (Profile, SessionSummary) -> Fragment receives results via callbacks -> updateUnifiedList builds List<SearchItem> -> SearchAdapter displays results. Tapping results navigates to ProfileActivity or DataActivity.
- **Viewing Profile Details (Charts & Syncing):**
    - **View:** ProfileActivity
    - **Repository:** SearchRepository (called within ProfileActivity.fetchProfile)
    - **Adapter:** ProfileSessionAdapter
    - **Chart Util:** SentimentChartHelper
    - **Flow:** Activity fetches profile and associated sessions using SearchRepository. Data (Profile, List<SessionSummary>) is displayed. SentimentChartHelper configures and populates the BarChart. RecyclerView.OnScrollListener and BarChart.OnChartGestureListener implement synchronization logic: scrolling one scrolls the other (syncChartToRecyclerViewScroll, syncRecyclerViewToChartScroll). Tapping items (OnChartValueSelectedListener, adapter click) calls setHighlight which updates highlights in both chart and list (ProfileSessionAdapter.setHighlightPosition) and scrolls the *other* view (scrollToSessionIndex, centerChartToSessionIndex). isProgrammaticScroll flag prevents infinite loops during sync.
- **Viewing Session Details (Charts & Syncing):**

    - **View:** DataActivity (Host), SessionOverviewFragment, SessionTranscriptFragment
    - **ViewModel:** DataViewModel (Scoped to DataActivity)
    - **Repository:** SearchRepository
    - **Adapters:** FinalTranscriptAdapter
    - **Chart Util:** SentimentChartHelper
    - **Flow:** DataActivity fetches FinalSessionDetail using DataViewModel -> SearchRepository. ViewModel exposes data via LiveData. SessionOverviewFragment observes detail, displays text, populates its chart. SessionTranscriptFragment observes detail, populates FinalTranscriptAdapter and its chart. Chart/List synchronization logic is similar to ProfileActivity but uses DataViewModel.requestHighlight to coordinate highlights between the two fragments when a chart bar is tapped in either fragment or when navigating from Overview chart to Transcript tab.
- **Inbox & Transcript Review/Editing:**
    - **Views:** InboxActivity, ProcessedDataDetailActivity, TranscriptDetailsFragment, TranscriptEditorFragment
    - **ViewModels:** InboxViewModel, TranscriptDetailViewModel (Scoped to ProcessedDataDetailActivity)
    - **Repository:** ProcessedDataRepository
    - **Adapters:** ProcessedDataAdapter, TranscriptPagerAdapter, TranscriptAdapter
    - **Flow:** InboxActivity observes InboxViewModel.pendingItems -> ProcessedDataRepository fetches pending list from API. Tapping item navigates to ProcessedDataDetailActivity with dataId. ProcessedDataDetailActivity uses TranscriptDetailViewModel to load TranscriptDetail via ProcessedDataRepository. TranscriptPagerAdapter manages "Details" and "Transcript" tabs. TranscriptEditorFragment displays editableTranscriptItems from ViewModel using TranscriptAdapter. Edits update the list in the ViewModel (notifyTranscriptChanged). Speaker changes/swaps modify items directly in ViewModel list. Swipe-left deletes items (with Undo). "Send Revised" button calls ViewModel.submitData ->

ProcessedDataRepository.submitFinalTranscript -> API PUT request. Back press checks hasUnsavedChanges LiveData.

6.2 Patient Workflow

- **QR Code Generation:**
    - **View:** PatientQRFragment
    - **Util:** AESUtil
    - **Flow:** User accepts terms -> button enabled -> tap button -> gets user details from SessionManager -> concatenates data + current date -> encrypts using AESUtil.encrypt with a randomly generated 6-digit passcode -> generates QR bitmap using ZXing library -> displays QR and passcode in a dialog.
- **Viewing History & Profile:**
    - **View:** PatientSearchFragment
    - **Repository:** SearchRepository
    - **Adapter:** SearchAdapter
    - **Flow:** Fragment calls SearchRepository.getOwnProfile and getOwnSessions. Repo calls specific API endpoints (/profiles/me, /sessions/me). Fragment displays results similar to Therapist search, but only for the logged-in patient. Tapping items navigates to ProfileActivity or DataActivity.

6.3 Common Features

- **Settings (Theme, Notifications):**
    - **View:** BrowseFragment, ThemeSelectionDialog, NotificationSettingsActivity
    - **Storage:** LocalStorageManager
    - **Flow:** BrowseFragment reads preferences from LocalStorageManager to display current settings. Theme selection dialog calls LocalStorageManager.setApplyTheme, which applies theme using AppCompatDelegate and saves preference. Notification settings toggle switches update preferences in LocalStorageManager. NotificationHelper reads these prefs when constructing notifications.
- **Navigation:**

- **Main:** MainActivity uses ViewPager2 with MainViewPagerAdapter synchronized with BottomNavigationView. Adapter determines which fragment (Therapist/PatientSessionFragment, Therapist/PatientSearchFragment, BrowseFragment) to show based on user type and selected tab.
- **Session:** SessionHostActivity uses the Navigation Component (session_nav_graph.xml) managed by NavHostFragment and NavController. Transitions triggered programmatically based on session type and user actions.
- **Data/Profile:** Simple Intent-based navigation between Search -> Profile -> Session Detail (DataActivity). DataActivity uses TabLayoutMediator to link TabLayout and ViewPager2.
- **Browse:** Simple Intent-based navigation from BrowseFragment to various settings/info activities.

7. Adding New Features

- **Example: Adding a New Session Type (e.g., VR):**
    1. **Define Type:** Add "VR" as a constant or enum value where session types are referenced (e.g., potentially in CardItem or constants file).
    2. **Update UI (Card Dialog):** Modify SaveCardDialogFragment (SESSION_TYPES array) to include "VR" as an option in Step 1.
    3. **Create Fragments:** Create new fragments for the VR session flow (e.g., VrSetupFragment, VrRecordingFragment, VrReviewFragment) similar to the audio flow fragments.
    4. **Update Navigation Graph:** Add these new fragments and define navigation actions between them (and from FormFragment) in res/navigation/session_nav_graph.xml.
    5. **Update ViewModel:** Modify SessionViewModel to handle VR-specific data (e.g., add LiveData for VR data file paths: _vrDataPath). Add logic to clear this data appropriately.
    6. **Update Host Activity:** In SessionHostActivity.navigateToNextStep, add a case "VR": to navigate to your new starting fragment (e.g., action_formFragment_to_vrSetupFragment).
    7. **Update Submission:**
        - In SessionHostActivity.uploadDataBasedOnSessionType, add a case "VR":. Get the VR data path(s) from the SessionViewModel.
        - Add a corresponding method in RecordingRepository (e.g., uploadVRSession) to handle encrypting VR data files and calling the appropriate API endpoint.
        - Add the new API endpoint definition in TherapyApiService.
        - Add the implementation (mock and real) in TherapyApiImpl.
    8. **Backend:** Ensure the backend API has corresponding endpoints and logic to handle the new VR session type and its associated data format.
- **General Steps:**
    1. **Identify Scope:** Determine which layers (UI, ViewModel, Repository, API, DB) are affected.

2. **Model:** Define any new data models (local entities, API DTOs).
3. **API:** Define new endpoints in TherapyApiService, implement in TherapyApiImpl (mock/real).
4. **Repository:** Add new methods in the relevant Repository to interact with the API or local DB.
5. **Database:** If needed, update Room entities (@Entity), DAOs (@Dao), and AppDatabase (increment version, add migrations).
6. **ViewModel:** Add LiveData to hold new data, methods to interact with the Repository, and logic to prepare data for the UI.
7. **UI (Fragments/Activities):** Create new layouts, observe ViewModel LiveData, handle user input, call ViewModel methods. Update navigation graphs or adapters as needed.
8. **Testing:** Add unit, integration, and UI tests for the new feature.

8. Testing

- **Unit Tests:** Test ViewModels (logic, LiveData updates), Repositories (mocking dependencies), utility classes (AESUtil, SentimentChartHelper). Use JUnit, Mockito/MockK.
- **Integration Tests:** Test Room database interactions (DAO tests), API calls (mocking server or using test endpoints), interactions between ViewModel and Repository. Use AndroidX Test libraries.
- **UI Tests:** Test user flows, UI state changes, adapter interactions. Use Espresso, UI Automator.

9. Coding Conventions & Best Practices

- **Language:** Primarily Java (consider migrating to Kotlin for new features).
- **Architecture:** Adhere to MVVM principles. Keep logic out of Views. Repositories abstract data sources.
- **Naming:** Follow standard Java/Android naming conventions (camelCase for variables/methods, PascalCase for classes). Be descriptive (e.g., CardViewModel vs. VM).
- **Logging:** Use Android's Log class consistently. Use different levels (Log.d, Log.i, Log.w, Log.e) appropriately. Avoid logging sensitive data.
- **Error Handling:** Handle potential errors gracefully (network issues, DB errors, permission denials). Use callbacks (ApiCallback, repository callbacks) or potentially Kotlin Coroutines/Flow with structured error handling for asynchronous operations. Provide user-friendly error messages.
- **Threading:** Perform database and network operations on background threads (Repositories use ExecutorService). Update UI only on the main thread (runOnUiThread, LiveData.postValue).
- **Resource Management:** Close database connections, input/output streams properly (using try-with-resources where applicable). Unbind services (RecordingService) when Fragments/Activities are destroyed. Cancel background tasks/coroutines when the associated scope is cancelled.
- **Security:** Be mindful of sensitive data (PHI). Use EphemeralPrefs with Keystore encryption for tokens/session data. Use FLAG_SECURE. Avoid hardcoding secrets. Validate input.
- **Code Style:** Maintain consistent formatting (use Android Studio's auto-formatter).
- **Comments:** Add comments to explain complex logic, public APIs, or non-obvious code sections.

10. Troubleshooting

- **Login Fails:** Check network connection, API base URL, credentials. Examine Logcat for specific error messages from AuthRepository or TherapyApiImpl. Verify TokenAuthenticator isn't stuck in a loop. Ensure device credential/biometric setup is correct.
- **Data Not Loading (Cards, Sessions, Inbox):** Check network connection. Verify API calls succeed in Logcat. Check repository callbacks are being triggered. Ensure LiveData observers are active (correct LifecycleOwner). If using mock data, check mock implementations in TherapyApiImpl. Check Room DB using App Inspection if necessary.
- **Recording Fails:** Check RECORD_AUDIO permission. Ensure RecordingService is binding correctly. Check Logcat for MediaRecorder errors (e.g., prepare failed, start failed). Ensure sufficient storage space.
- **Upload Fails:** Check network connection. Verify authentication token is valid (EphemeralPrefs). Examine Logcat for errors from RecordingRepository or TherapyApiImpl. Check file encryption/permissions. Verify backend endpoint is expecting the correct MultipartBody parts and metadata format.
- **Crashes:** Check Logcat for the full stack trace. Identify the component (Activity, Fragment, ViewModel, etc.) and line number where the crash occurred. Debug step-by-step.
- **UI Issues (Layout, Synchronization):** Use Layout Inspector. Debug adapter logic (onBindViewHolder, item counts). Add extensive logging in chart/list synchronization callbacks (OnScrollListener, OnChartGestureListener) to trace isProgrammaticScroll flag and sync calls.

11. Appendix

- **Glossary:**
    - **Card:** Therapist's template for a session type.
    - **Session Card:** UI representation of a CardItem in TherapistSessionFragment.
    - **Session Summary:** Overview data for a past session (used in search/profile lists).
    - **Session Detail:** Full processed data for a past session (including transcript, notes).
    - **EphemeralPrefs:** Encrypted SharedPreferences for sensitive session data.
    - **LocalStorageManager:** Standard SharedPreferences for app settings.
    - **Keystore:** Android's secure key storage system.
    - **PHI:** Protected Health Information.