Kuripot App – Full Technical Documentation

Personal Offline Budget + Notes Tracker for Android (Kotlin + Jetpack Compose + SQLite Room)

🆔 App Info

- **Name**: Kuripot App 🪙
- **Platform**: Android
- **Min SDK**: 24 (Android 7.0)
- **Target SDK**: 35
- **Language**: Kotlin
- **UI Framework**: Jetpack Compose
- **Database**: SQLite (via Room)
- **Data Privacy**: Local-only, export as `.json`

📋 Summary

Kuripot App is an offline personal budgeting and note-taking app designed to manage freelance finances and journaling. It supports multiple categories, voice notes, and secure access via a 4-digit passcode. Data is exportable in JSON format and importable offline.

🧹 Features

🔐 Passcode Lock

- Required on app start
- Set and update in Settings
- 4-digit PIN (hashed locally)

📝 Notes

- Title + content
- Categorized (journal, plan, etc.)
- Optional voice note
- Editable/deletable
- Archived instead of hard deleted

💰 Budget Tracker

- Linked to non-deletable Budget category
- Add income/expenses
- Tabular format
- Monthly totals, balance
- Subcategories supported (e.g., Rent, Client A)

📂 Categories

- Default: Budget (undeletable)
- Add, edit, delete other categories
- Used to classify notes and budgets

📁 Archives

- Deleted notes & budgets are archived
- View, restore, or permanently delete
- Managed in Settings

🌓 Theme

- Light & dark mode toggle
- Stored in local DB

🎧 Voice Notes

- Record and attach .m4a or .ogg files to notes
- Playback supported in UI

🔄 Import / Export

- Export full app data to .json
- Import from .json to restore/merge data
- Export only available in Settings

🧠 App Structure

Packages:

- com.malikhain.kuripot_app
- ├── data/ // Room entities, DAOs, database
- ├── ui/ // Jetpack Compose screens
- ├── utils/ // Helper functions (theme, passcode, audio, JSON)
- ├── viewmodel/ // State & logic management
- ├── model/ // Data classes for JSON

- ├── service/   // Audio, import/export, archive
- └── MainActivity.kt

🗃️ Database Schema (Room + SQLite)

NoteEntity

- @Entity(tableName = "notes")
- data class NoteEntity(
-     @PrimaryKey(autoGenerate = true) val id: Int = 0,
-     val title: String,
-     val content: String,
-     val categoryId: Int,
-     val createdAt: String,
-     val voicePath: String?,
-     val isBudget: Boolean = false
-)

CategoryEntity

- @Entity(tableName = "categories")
- data class CategoryEntity(
-     @PrimaryKey(autoGenerate = true) val id: Int = 0,
-     val title: String,
-     val isDefault: Boolean = false
-)

BudgetEntryEntity

- @Entity(tableName = "budget_entries")
- data class BudgetEntryEntity(
-     @PrimaryKey(autoGenerate = true) val id: Int = 0,
-     val noteId: Int,
-     val date: String,
-     val description: String,
-     val entryType: String, // "income" or "expense"
-     val amount: Double,

- val subCategory: String?
-)

ArchiveEntity

- @Entity(tableName = "archives")
- data class ArchiveEntity(
- @PrimaryKey(autoGenerate = true) val id: Int = 0,
- val type: String,   // "note", "budget"
- val dataJson: String,   // Raw JSON for restoration
- val deletedAt: String
-)

SettingEntity

- @Entity(tableName = "settings")
- data class SettingEntity(
- @PrimaryKey val key: String,
- val value: String
-)

🔐 Passcode Security

- Passcode stored hashed using SHA-256 or similar
- Used on app startup and settings access
- No biometric yet (future feature)

🗺️ UI Navigation (Jetpack Compose)

- Notes Screen: List, filter, add/edit/delete

- Budget Screen: Table view with income/expense tabs
- Settings Screen: Theme, passcode, archive, import/export

🎨 Theme

- Dark/light stored in settings table
- Compose Material3 theme adapted dynamically

📤 Export & Import

Export

- Converts entire app DB into a `.json` string
- Saves to device storage (Downloads or sandboxed dir)

Import

- Parses `.json`
- Inserts data into DB
- Prevents duplicates via ID or timestamp checking

🔄 Undo & Archive Logic

- Deleting a note or budget → archive entry created
- Archive data includes full record JSON and type

- Archives view allows:
    - Restore
    - Permanent delete

🎧 Voice Note Support

- Uses Android `MediaRecorder` or similar
- Records .m4a or .ogg in app's private storage
- Path saved in `NoteEntity`

📁 JSON Export Format (Structure)

- {
-   "notes": [{ ... }],
-   "categories": [{ ... }],
-   "budget_entries": [{ ... }],
-   "archives": [{ ... }],
-   "settings": { "theme": "dark", "passcode": "hashed_pin" }
- }

🛠️ Dependencies Used

Library	Purpose

Jetpack Compose	UI
Room	SQLite ORM
Kotlinx Serialization	JSON import/export
Material3	UI components
MediaRecorder	Voice recording

🧪 Testing Suggestions

- Unit test Room DAOs
- UI test navigation + data state
- Instrumentation tests for passcode, theme, audio

📦 Build & Packaging

- Use `proguard-rules.pro` to exclude Room and audio classes
- Exported JSON saved to public or scoped storage via `Context.getExternalFilesDir()`

📆 MVP To-Do Checklist

Feature	Status
Room DB setup	✅ Planned
Jetpack Compose UI	⏳
Passcode setup	⏳
Notes & budget logic	⏳
Export/import JSON	⏳
Voice recording	⏳
Archive handling	⏳
Testing	⏳
Release build + icon	⏳