**📄 Product Requirements Document (PRD)**

**Project Name: My Records**

**Platform: Android (Flutter, Dart, SQLite)**

**Version: 1.0**

**Default Mode: Dark Mode**

**1. Project Overview**

“My Records” is a personal folder-based record management mobile application where users can create folders and store categorized information.

All png are kept in D:\Anant\VSCodeProjects\My\_Records

Additionally, the existing project D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app will be analyzed only for reference (no modification) to reuse any reusable widgets, core structure, theme logic, database handling, android version or design considerations. You can also refer the project information via memory mcp.

**2. Objectives**

✔ Allow users to create, view, edit, duplicate, and delete folders.  
✔ Show total folder count under the section “Folder Lists”.  
✔ Provide Dark/Light mode toggle and Help section.  
✔ Use local SQLite database for storing folder records.  
✔ Default theme should be Dark Mode.  
✔ Use the provided image file ("") as App Icon (to be replaced with actual filename when provided).

**3. Core Features & Requirements**

**3.1 Home Screen (Refer to 1.png layout),** All png’s are kept in D:\Anant\VSCodeProjects\My\_Records

| **Component** | **Description** |
| --- | --- |
| AppBar | Title aligned to left: **“Folder Lists”** |
| Folder Count | Below title, display “Total: X Folders” dynamically from database |
| Theme Toggle | Light/Dark mode icon (Right side of AppBar) |
| Help Icon | On top right side (next to theme toggle), opens basic instructions dialog/page |
| Folder List View | List of folders in card/list format, each with folder name & icons for edit, duplicate, delete |
| Floating Action Button (FAB) | Positioned at bottom-right (“+” icon) to add new folder |

**3.2 Create New Folder (Refer to 2.png)**

When user clicks “+” button:

* Show Pop-up Window / Dialog titled **“Create New Folder”**
* Fields Required:
  + Folder Name (TextField)
  + Description (optional)
  + Creation Date auto-filled (but hidden or visible as per 2.png)
* Buttons:
  + **Save** – inserts into SQLite
  + **Cancel / Close**

**3.3 Folder Actions (after creation)**

Each folder row/card should display the following actions:

| **Icon** | **Functionality** |
| --- | --- |
| ✏ Edit | Opens pop-up to update folder name/description |
| 📄 Duplicate | Creates a copy of the folder with same details and “(Copy)” appended |
| 🗑 Delete | Shows confirmation pop-up before deleting from database |

**3.4 Dark/Light Mode**

* Default mode = **Dark Mode**
* Toggle icon in the AppBar for switching theme
* Theme settings saved using SharedPreferences or SQLite table settings
* Inherit reusable theme logic from **health\_records\_app**: check for themes, icons, helper functions.

**3.5 Help Section**

* Triggered via **Help Icon**
* Display pop-up or full screen with:
  + Purpose of the app
  + Instructions on creating folders and using app features
  + Usage of duplicate/edit/delete options
  + Theme switching info

**4. Database Design (SQLite)**

**4.1 Table: folders**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | INTEGER PRIMARY KEY AUTOINCREMENT |  |
| folder\_name | TEXT NOT NULL |  |
| description | TEXT |  |
| created\_at | TEXT (DateTime ISO format) |  |

**5. Reference Project Analysis (No Code Changes Made)**

Path: D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app

**Helpful reusable components identified:**  
✅ SQLite database helper structure  
✅ SharedPreferences theme management  
✅ Backup/restore logic – useful for future enhancements  
✅ UI theming & dark mode handling  
✅ Reusable widgets for dialogs and form validation

**Potential issues to avoid in My Records app:**

* Complex navigation & too many dependencies (keep structure modular)
* Avoid mixing business logic with UI widgets (use provider or simple MVC)
* Ensure folder duplication creates unique IDs and avoids primary key conflicts
* Optimize list rendering with ListView.builder to prevent UI lag

**6. App Icon**

* The provided icon image ("D:\Anant\VSCodeProjects\My\_Records\records.png") will be used as app icon
* To be processed into Android mipmap format using flutter\_launcher\_icons package

**7. Tech Stack**

| **Component** | **Technology** |
| --- | --- |
| Language | Flutter (Dart) |
| Database | SQLite using sqflite package |
| Theme | Light/Dark with Provider or ChangeNotifier |
| State Mgmt | Provider / GetX (simple approach preferred) |
| Local Settings | Shared Preferences |

**Incorporate the above requirements/fixes.**

* Use Sequential thinking and other MCP as required to achieve the objective.
* The UI should be stunning and easy to use.
* Improvise as required, Fill the missing requirements or gaps.
* Ensure this change/enhancement does not break any existing code or functionality.

**Check the Android build settings in the project**

**D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app**

**and apply the same settings to the current project to generate a debug APK.**

**Finally install the apk to my mobile.**

**Incorporate the above requirements/fixes.**

* Use Sequential thinking and other MCP as required to achieve the objective.
* The UI should be stunning and easy to use.
* Improvise as required, Fill the missing requirements or gaps.
* Ensure this change/enhancement does not break any existing code or functionality.
* **The handle icon shown in 1.png should allow users to move items up and down. Check how this is implemented in the project at  
  D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app  
  and apply the same functionality in the current project.**
* **Add fingerprint authentication to the app, similar to the implementation in the health\_records\_app project.**
* **When saving a folder name or optional description, automatically capitalize the first letter.**
* **The app name displayed below the app icon should be “My Records”, but it is currently showing as “my\_records”.**
* **Fix the “overflowed” error shown in 3.png in the requirements folder.**
* **Not working - Add fingerprint authentication to the app, similar to the implementation in the health\_records\_app project.**
* **On the Welcome Back screen (see 6.png in requirements folder), entering the PIN or using fingerprint authentication does not log in successfully.**
* **When I click on a folder item, it should open to Record screen which should have a + button on bottom right**
* **When + button is clicked it should open a popup screen which 2 fields Field Name, Field Value, the user will enter data in them and either cancel or create**
* **Add a + icon next to the Field Value input, allowing the user to add additional Field Value rows. The user should be able to add as many Field Values as needed.**
* **Also the date and time format on record should be like 17 Oct 2025, 4:40 PM**
* **The additional field(s) value added should show in same record not as different record.**
* **When displaying additional Field Values, they are currently shown with bullet points. Remove the dot (bullet point) before each value.**
* **Sort the records alphabetically by Field Name when displaying them.**
* Add a **Help screen** for the Record screen, similar to the one used for the Folder screen.
*  Add a **Search bar** on the Folder screen, below the folder count (refer 8.png using MCP).
*  Include a **filter dropdown** to search either in **All Folders** or within a **specific folder**.
  +  The dropdown list should display:
  + **“All Folders”** as the first and default option
  + Followed by all folder names, sorted alphabetically.
*  The search function should look for data **inside each folder’s records**, not by folder names.
*  If the same value is found in multiple records across different folders, show **all matching results along with their folder names**.
* The search result should show complete record of matching search phrase and folder name below it. No need to show search keyword, matched xxx in search record.
* When a search result is clicked, or when **“View Folder”** is selected, it should navigate to the respective folder and show that **specific record at its exact position** within the record screen for that folder.

Add a **Settings icon** on the top right of the Folder screen. When clicked, it should display two options: **Change PIN** and **Backup & Restore**.

* Implement both features similar to the **health\_records\_app** project located at D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app, with the only difference being that the PIN here should be **6 digits**.
* The backup file should be saved in the **my\_records** folder under **Android/Downloads**, with the filename format:  
  my\_records28Oct25\_0955AM
* All other functionalities—such as retaining the **last three backups**, providing **multiple share options**, and other related features—should work exactly as implemented in the health app.
* Add a **Help screen** for both features (**Change PIN** and **Backup & Restore**), similar to the one in the Health app.
* Finally, **verify that all features, functionalities** and **screen layout** match those in the Health app, **except for the specific exceptions mentioned** earlier.
* In the **Change PIN** screen, the text box should automatically gain focus so the user can start typing immediately without needing to tap inside it.
* When the user selects **Backup and Restore**, the screen should load and function exactly as it does in the **Health app**. Refer to **1.png** and **2.png** in the requirements folder for visual reference (use **Desktop Commander MCP** or any other MCP to view the PNG files).  
  The corresponding implementation can be found in the file:  
  D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app\lib\presentation\screens\backup\_restore\_screen.dart
* Also, **verify all functionalities** in the **Backup and Restore** screen for **buttons, Display values, dropdowns, and icons, etc.** — to ensure they are implemented exactly as in the **Health app, except for backup file name which should be** the filename format: my\_records28Oct25\_0955AM.
* Fix all errors listed in the **Problems tab**, including minor ones, and then **build and install the debug APK** on my mobile device to test all features.
* The backup should only be created if there are any folders, and its should create a backup file consisting of all folders and records within it and same will be used to restore at the later date.
* Check 4.jpg in requirements folder using **Desktop Commander MCP** or any other MCP to view the PNG files and fix the issue.
* When creating a **backup file**, first check if the folder **my\_records** exists in **Android/Downloads**. If it does not exist, **create the folder**, and then save the backup file inside it.
* Review how **folder permissions** are handled in the **Health app** to prevent errors such as:  
  Backup failed: Exception: Failed to create backup: PathAccessException: Cannot open file, path = '/storage/emulated/0/Android/Downloads/my\_records/my\_records28Oct25\_1100AM.json' (OS Error: Operation not permitted, errno = 1)
* Add a delete All icon between settings and help in folders screen, when user clicks it show warning message and once he confirms delete all folders along with its associated records
* On the **Splash Screen**, the text *“My Records – Secure, Organised, Accessible”* should be positioned **at the center or slightly above the middle** of the screen so that it remains **fully visible**, even when the fingerprint dialog appears (as the dialog currently hides part of the text).
* After fingerprint authentication, the screen initially appears blank (showing no folders) before the folders load. This delay should be **fixed** so that the folders are **displayed immediately** after successful authentication.
* Review all **Help screens** and update them if necessary to reflect the **new functionalities or changes added in the last two days**.
* Update the Readme.md based on project functionality as we have added **new functionalities or changes added in the last two days**.
* finally push to Github.
*  The **Automatic Backup** information is incorrect. Update/check the code if needed to reflect the correct details:
* Backup saved path (currently wrong)
* Device charging / idle requirement (currently described incorrectly)
* Number of retained automatic backups — it should be **3**, not 5.
*  The **Manual Backup** section is incorrect: it currently omits that **manual backup requires at least one existing folder**. Update the text and verify the code if necessary.
* In the **Backup and Restore** screen, make the **Backup panel’s background (Panel where the file name is shon) color** the same as the **Available Backups panel’s background color**.
* Additionally, set a **white border** for each backup panel where file name is shown.
* Do not make any other changes to the layout or functionality.
* The **Splash Screen** displaying the text *“My Records – Secure, Organised, Accessible”* briefly appears (for about one second) when interacting with certain UI elements such as the **search text box**, **dark/light mode toggle**, or during the **Change PIN** operation.  
  This issue should be **fixed** so that the splash screen only appears **once at app launch** and not during normal app interactions.
* **Build and install** the **debug APK** on my mobile device for testing and verification of all implemented features.
* In the **Health app** (D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app), the backup files are stored in an **encrypted format** with a .enc extension.
* Perform a **detailed review** of the implementation to understand how **encryption and decryption** are handled in that project.
* Apply the **same encryption logic** in the **current project**, ensuring that:
  + The **file name format** and **storage location** remain unchanged.
  + Only the **file content** is encrypted during **backup** and decrypted during **restore**.
* Invoke any **necessary MCPs** (e.g., *Desktop Commander MCP* or others) required to accomplish this task.
* **Additional Reference:**
  + Backup and Restore screen path:  
    D:\Anant\VSCodeProjects\Flutter\_Health\_Records\health\_records\_app\lib\presentation\screens\backup\_restore\_screen.dart
* The backup file extension should be .enc not .json
* Set default password 301976, no need to ask for regular or password protected
* During restore getting error “package: flutter/src/widgets/ framework.dart': Failed assertion: line 6065 pos 14: '\_dependents.isEmpty': is not true. See also: https://docs.flutter.dev/testing/errors” but data is getting restored.”
* Also the process should work smoothly with Auto backup settings of Daily and weekly?
* In Password-Protected Backup dialog
  + Remove - Remember your password! It cannot be recovered.
  + Some weird symbols at beginning of below 2 lines

ðŸ Folders backed up: 9

ðŸ\*, Records backed up: 11

* During restore its not asking for default password? And directly getting restored.
* Some weird symbols at beginning of below 2 lines, also Records backed up is showing twice, remove emojis

ðŸ Folders backed up: 9

ðŸ\*, Records backed up: 11

* The restore button doesn’t enable immediately once you enter the password? It takes 2 or 3 seconds?

• Remove the unwanted symbols appearing at the beginning of the following lines and eliminate emojis.  
• Ensure that **“Records backed up”** is displayed only once.

 “Folders backed up: 9”  
 “Records backed up: 11”

• Also, fix the issue where the **Restore** button takes 2–3 seconds to enable after entering the password — it should activate immediately once a valid password is entered.

• In the “Password-Protected Backup Created” dialog, the **Records backed up** count is shown as **11**, but the actual number of records across all folders in the app is **9**.  
• Please verify and fix the logic that calculates the record count during backup creation so it reflects the actual number of records.

* Remove the unwanted or junk characters that appear in place of bullet points in the **Help screen** of *Backup and Restore*.
* Update and enhance the Help content to include information about the **encryption process** used during backup and restore. Refer to the implementation code to ensure the details are accurate and consistent with actual functionality.

When we click on any record we get a dialog displaying field name/values, Created and Updated date, etc.

Add the following enhancement to the Dialog box

* If the Field value has a past date mentioned Example Anant DOB 30/12/1976 which is in date/Month/Year Add below it Age is x years, y months and z days, if Year value is zero just mention months and days, similarly if month value is 0 just mention days, If days value is 0 mention Expired in red color
* If the Field value has a future date mentioned Example Expiry 31/03/2026 which is in date/Month/Year Add below it Expires in x years, y months and z days, if Year value is zero just mention months and days, similarly if month value is 0 just mention days, If days value is 0 mention Expired in red color
* These are only for display purpose, no need to save the values in database
* These needs to be dynamically calculated which user clicks any record and in case it has a date mentioned alone or in the sentence
* Also cater for date variations like 01/03/2025, 1/3/25, 1 Jan 25, 1-Jan-2025, etc comibination but it will always be date first, Months next and finally year format

When a user clicks on any record, a dialog box currently displays the field names, values, and timestamps (Created and Updated dates), etc. Enhance this dialog box with the following functionality:

* **Dynamic Date Interpretation:**
  + If a **field value contains a past date**, e.g., *Anant DOB 30/12/1976*, display an additional line below it showing:  
    **“Age: X years, Y months, Z days”**
    - If the year value is 0, display only months and days.
    - If the month value is 0, display only days.
    - If the day value is 0, display **“Expired”** in **red**.
  + If a **field value contains a future date**, e.g., *Expiry 31/03/2026*, display an additional line below it showing:  
    **“Expires in X years, Y months, Z days”**
    - Apply the same formatting rules for 0 year, month, or day values.
    - If the day value is 0, display **“Expired”** in **red**.
* **Behavior & Data Handling:**
  + These calculated details are for **display only** and should **not** be stored in the database.
  + The values must be **calculated dynamically** each time the user opens a record.
* **Date Format Handling:**
  + The system should recognize and correctly interpret various date formats such as:
    - 01/03/2025
    - 1/3/25
    - 1 Jan 25
    - 1-Jan-2025
  + The format pattern will **always** follow the sequence: *Date → Month → Year*.

It currently works for standard date formats like **25/07/2025**, but does not detect formats such as **Sat 03 April 24** or **4 Mar 2025**.  
Enhance the logic to **analyze the field value**, automatically **identify any valid date pattern** within the text (including variations with day names, short/long month names, or short year formats), and then perform the corresponding **date interpretation and calculation** for those combinations.

* Inform users that **tapping on any record** will display its detailed information.
* Update the **Records Help** screen to guide users on **entering dates in supported formats** within the *Field Value*.
* Review the code to determine all **recognized date formats**, and include clear **examples** (e.g., 25/07/2025, 4 Mar 2025, 03-Apr-24) in the help section for user reference.
* Update the Readme.md file on important features added today and push to Github.

 When saving a record, if the **Field Name** contains the word **“email”**, allow the **Field Value** to start with either an **uppercase or lowercase letter**.

 When duplicating a folder, if it contains any records, ensure that **all associated records are duplicated** along with the folder.

* Not working – When saving a record, if the field value is a URL, it should be allowed even if it starts with an uppercase or lowercase letter. You may check whether the value begins with **“http”** or **“https”** to validate it correctly.
* Once done install build and install debug apk
* Enable record duplication, ensuring that all associated data are also duplicated along with the record.

**Enhance the existing record duplication functionality** as follows:

* If there is **only one folder** in the app, the **duplicate** action should work as it currently does.
* If there are **multiple folders**, clicking **Duplicate** should open a **popup window** with a **“Select Folder”** dropdown listing all folder names (sorted alphabetically).
  + The **current folder** should be selected **by default**.
  + Include **OK** and **Cancel** buttons in the popup to confirm or cancel the duplication action.
  + If the user select another folder from the dropdown than the duplicate record should happen in that folder.

The existing functionality—where clicking a **search result** or selecting **“View Folder”** navigates to the respective folder and displays the specific record at its exact position—is working correctly.  
However, in some cases, the record appears **partly hidden at the bottom** of the screen.  
Enhance the behavior so that the **selected record is positioned near the middle of the screen** for better visibility and user experience. Only exception to this is if record is the last one in the folder than showing it in the botton is ok. You need to see the searched keyword is clearly visible on the screen and not hidden below or above and the full record is visible.

Here’s the refined and professional version of your text:

The existing functionality—where clicking a **search result** or selecting **“View Folder”** navigates to the respective folder and displays the specific record at its exact position—is working correctly.  
However, in some cases, the record appears **partly hidden at the bottom** of the screen.

Enhance this behavior so that the **selected record is displayed near the middle of the screen**, ensuring that both the **searched keyword** and the **entire record content** are **fully visible** on the screen for better readability and user experience.

The **only exception** should be when the selected record is the **last one in the folder**—in that case, displaying it at the bottom is acceptable.

Yes — this kind of “scroll-to-centered record” behavior can be tricky in Flutter, and GitHub Copilot often struggles because it requires careful coordination between **ListView scrolling**, **item positions**, and **layout rendering timing**.

Here are some practical **tips and approaches** to help you implement it correctly:

**🧭 1. Use Scrollable.ensureVisible() — but with a twist**

If you’re already using Scrollable.ensureVisible(context), it only scrolls just enough to make the widget visible, not to center it.  
You can **extend this** by manually calculating an **offset adjustment** after the scroll.

**Example:**

final context = recordKey.currentContext;

if (context != null) {

await Scrollable.ensureVisible(

context,

duration: const Duration(milliseconds: 500),

);

// Optional: add a small delay, then adjust offset to center item

await Future.delayed(const Duration(milliseconds: 200));

final box = context.findRenderObject() as RenderBox;

final position = scrollController.position;

final itemOffset = box.localToGlobal(Offset.zero).dy;

final itemHeight = box.size.height;

final screenHeight = MediaQuery.of(context).size.height;

final targetOffset = position.pixels + itemOffset - (screenHeight / 2) + (itemHeight / 2);

scrollController.animateTo(

targetOffset.clamp(0, position.maxScrollExtent),

duration: const Duration(milliseconds: 400),

curve: Curves.easeInOut,

);

}

✅ This ensures the selected record appears roughly in the **middle of the screen**, unless it’s near the top or bottom.

**🧩 2. Use ScrollablePositionedList (if possible)**

If your records are displayed in a long list, consider replacing your ListView with the **ScrollablePositionedList** package — it provides **precise control** over which item to scroll to and how to align it.

**Example:**

ScrollablePositionedList.builder(

itemCount: records.length,

itemScrollController: itemScrollController,

itemPositionsListener: itemPositionsListener,

itemBuilder: (context, index) {

return RecordTile(record: records[index]);

},

);

Then, when showing a searched record:

itemScrollController.scrollTo(

index: targetIndex,

duration: const Duration(milliseconds: 400),

alignment: 0.4, // 0.0 = top, 0.5 = center, 1.0 = bottom

);

✅ The alignment parameter gives you **direct control** to position the record near the middle.

**🕐 3. Wait for UI to render before scrolling**

Sometimes Copilot’s implementation fails because it tries to scroll **before the widget tree finishes building**.  
Use a **post-frame callback**:

WidgetsBinding.instance.addPostFrameCallback((\_) {

scrollToRecord(targetIndex);

});

✅ This ensures the scrolling happens only **after** the list and layout are ready.

**🧠 4. Handle the “last record” exception**

Before applying the centering logic, check if the target record is the **last one**:

if (targetIndex == records.length - 1) {

scrollController.animateTo(

scrollController.position.maxScrollExtent,

duration: const Duration(milliseconds: 400),

curve: Curves.easeInOut,

);

} else {

// apply center-scroll logic

}

✅ Keeps the behavior natural and consistent with your exception rule.

 Add a **Search** field to the **Records** screen, similar to the one used in the **Folders** screen.

 A **filter field is not required**.

 The search should function **only within the currently opened folder**, allowing users to quickly find records based on their search input.

* When the user **navigates away** from the **Records** screen, the **search field** should be **automatically cleared** so that it resets when the screen is reopened.
* The **Search** field should be **displayed only** when the folder contains **more than one record**.
* If there is **only one record**, the search field should remain **hidden**.