ChatGPT

Product Requirements Document (PRD)

1. Product Summary

Name: Archiver (working name)

Purpose: Provide a non-subscription iOS/iPadOS utility that allows users to compress and decompress files in a wide range of archive formats directly from their device. The product should be easy to use, offer advanced options (format selection, compression level, password protection) and integrate into the system share sheet and file browser so that users rarely need to open the app to perform basic actions.

Goal: Enable seamless file compression/decompression while maintaining a minimal, unobtrusive interface. Support iPhone and iPad with appropriate layouts and features.

2. Background and Context

Apple's built-in Files app supports compression and decompression of **ZIP** archives by long pressing on a file or folder and choosing *Compress* or *Uncompress* [1] [2]. Third-party utilities extend this capability to many other formats, including **bzip2, gzip, tar, xz and 7-Zip**, and offer features such as choosing compression level, splitting archives and encryption [3] [4]. There is a gap for a dedicated app that combines wide format support, a clean user experience and deep integration with the share sheet so users can compress or extract archives without launching a full UI.

3. Objectives

- Support compression and decompression of multiple archive formats: `.zip`, `.7z`, `.tar`, `.gz`, `.bz2`, `.xz` and others listed in the research [3].
- Provide a simple interface for compressing files and folders into archives and decompressing archives into files.
- Offer advanced options: select format, adjust compression level, split archives, set passwords and encryption.
- Integrate with iOS/iPadOS share sheet and Files extensions to allow compression and extraction without opening the main app.
- Use a consistent design across iPhone and iPad with accessible UI and minimal cognitive load.
- Ensure performance and reliability for large files and archives.

4. User Personas

- **Casual user:** needs to decompress downloaded archives (e.g., `.zip`, `.rar`) from email or web without understanding formats. Values simplicity and share-sheet integration.
- **Power user:** wants to compress multiple files, choose advanced formats and encryption, and control compression level. May compress large directories or share archives across platforms.
- **Developer/tester:** frequently packages logs or project folders; needs quick access to compress options from within the Files app and settings to customize defaults.

5. Core Features

5.1 Compression

1. **File selection:** Allow users to select single or multiple files/folders from the Files picker or by dragging them into the app. When compressing multiple items, they are packaged into a single archive (similar to the built-in Files app behavior for zip archives [5]).
2. **Archive format selection:** Provide a drop-down list of supported formats (ZIP, 7-Zip, TAR, GZIP, BZIP2, XZ, etc.).
3. **Archive name:** Text field to name the output archive.
4. **Compression level:** Slider offering pre-defined levels (Fast, Normal, Maximum) affecting speed vs. size.
5. **Encryption and password:** Toggle to encrypt the archive (supported formats only). When enabled, prompt for password. Use secure AES-256 encryption where possible.
6. **Split/combine archives (advanced):** Option to split large archives into segments of specified size (e.g., 100 MB parts) and to combine segmented archives when decompressing. This mirrors capabilities in some tools like Keka [4].
7. **Compression progress:** Show a progress bar and estimated time. Offer “Cancel” button.
8. **Completion actions:** On success show a summary with archive name, size and “Share” button to send via Mail/Messages or save to cloud.

5.2 Decompression

1. **Archive selection:** Let users pick an archive file via Files picker or share-sheet import. Support archives with multiple parts.
2. **Preview contents:** Display list of files/folders in the archive with file sizes and optional thumbnails. Permit selecting specific items or “Extract All.”
3. **Password entry:** If archive is encrypted, prompt for password. Validate password before extraction.
4. **Destination selection:** Let users choose destination directory. Offer quick options (same folder, iCloud Drive, On My iPhone/iPad).
5. **Extraction progress:** Show progress bar and estimated time. Provide “Cancel.”
6. **Completion actions:** After extraction, show list of extracted items with “Open” or “Share” actions.

5.3 Share Sheet and File Extension Integration

To minimise friction and keep the app unobtrusive, integrate deeply with iOS/iPadOS system interfaces:

1. **Compression via share sheet:** When a user selects files in the Files app or taps “Share” on any item, present a **Compress** action. This action shows a minimal sheet to choose archive format and encryption, then runs compression in the background and presents the resulting archive in the share sheet for saving or sending.
2. **Decompression via share sheet:** When a user opens the share sheet on an archive file (e.g., from Mail, Safari or Files), show an **Extract** action. This extracts the archive to a temporary location and opens a preview of the contents with options to save to a folder.
3. **Quick Actions (context menu):** Support long-press contextual menu in Files for “Compress” and “Extract” similar to built-in compress/uncompress actions [2] but extended to all supported formats.

4. **File provider extension:** Implement an extension so the app appears as a location in the Files app. Users can browse archives as if they were folders; the extension handles lazy extraction and supports Quick Look for preview.
5. **Shortcuts integration:** Expose compress and decompress actions to the Shortcuts app, enabling users to create automation workflows (e.g., compress camera photos daily).

5.4 Settings and Defaults

1. **Default archive format** (ZIP or 7-Zip, etc.).
2. **Default compression level** (Normal, Maximum, Fast).
3. **Default extract location** (e.g., iCloud Drive/Downloads). Users can set a default to avoid being asked each time.
4. **Theme** (light/dark/system).
5. **Clear recent history** and storage management options.

5.5 Recent/History

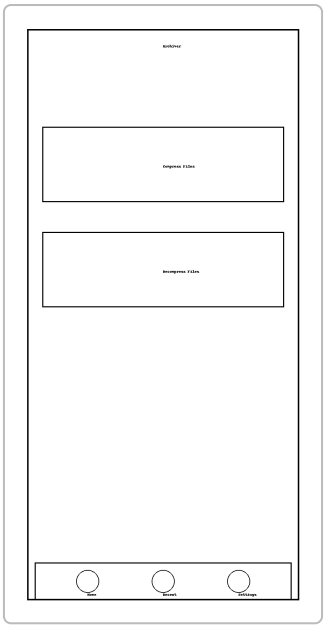
- Maintain a list of recently created archives and recently extracted archives. Each entry shows file name, format, date, size and action buttons (open, share, delete).

6. User Flows and Wireframes

The following flows correspond to the wireframes. Refer to the linked images for visual guidance.

6.1 Home Screen (iPhone)

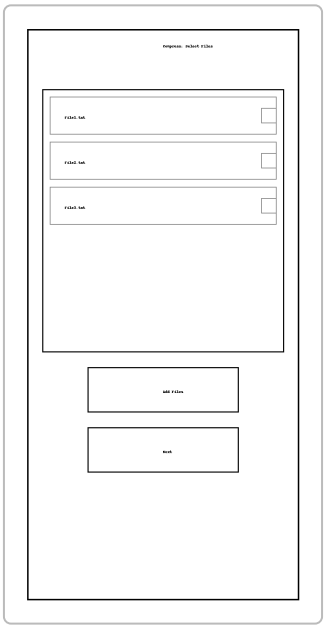
The home screen presents two prominent actions and a navigation bar:



1. User launches app.
2. Chooses **Compress Files** to start a compression flow or **Decompress Files** to extract an archive. The bottom nav bar gives access to the **Recent** list and **Settings**.

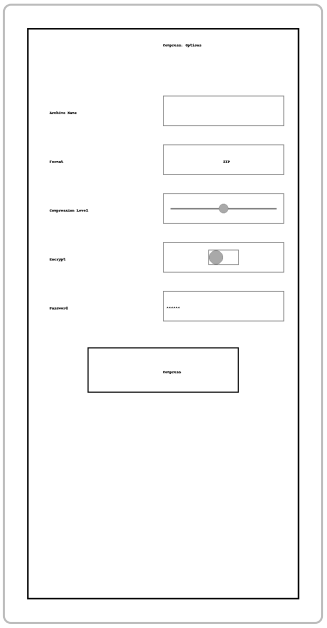
6.2 Compress Flow (iPhone)

Select files: The user picks files/folders from the device:



- Tapping **Add Files** opens the Files picker; selected items appear in the list. Checkbox icons allow removing items.
- Tapping **Next** navigates to the options screen.

Configure options: The user chooses archive name, format, compression level and encryption:



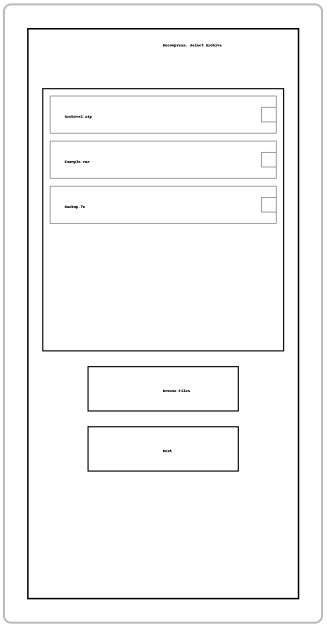
- Archive name field pre-fills with a suggestion (e.g., `Archive.zip`).
- Format dropdown shows supported formats.
- Compression level slider defaults to Normal but can be moved to Fast or Maximum.
- Encrypt toggle reveals password entry when enabled.

- Tapping **Compress** starts the process. A modal progress view appears with a spinner and Cancel button.

Upon completion, the app displays a sheet summarising the archive and offering **Share** and **Open in Files**.

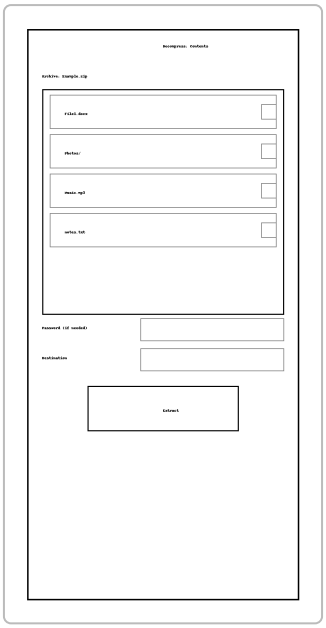
6.3 Decompress Flow (iPhone)

Select archive:



- The file list shows known archives; **Browse Files** opens the Files picker to select other archives.
- After selecting an archive and pressing **Next**, the app parses it.

View contents and extract:

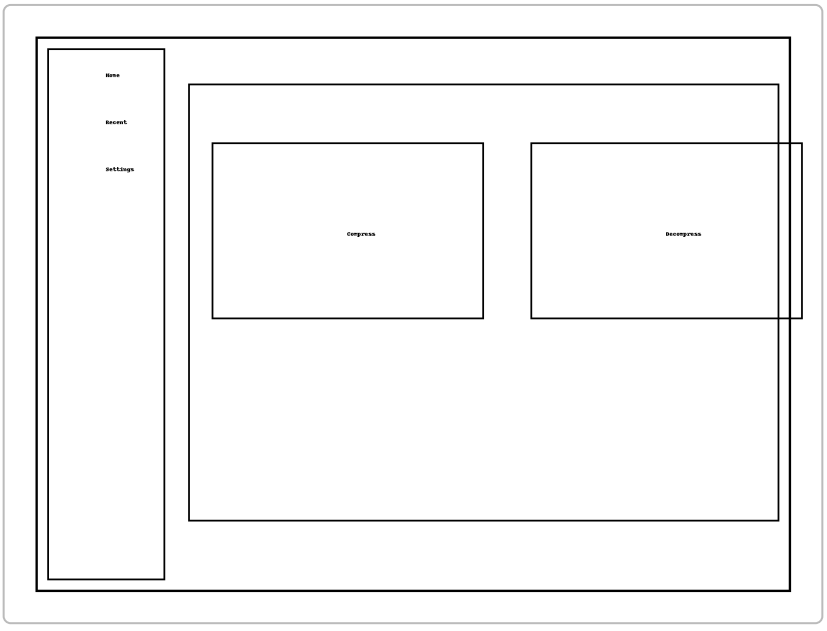


- Display contents with checkboxes. Folders are indicated with a trailing `/`.
- Password field appears when the archive is encrypted.
- Destination selector opens a folder picker (default is current folder).
- Pressing **Extract** begins extraction and shows progress. On completion, a summary with extracted files and options to open or share appears.

6.4 iPad Layouts

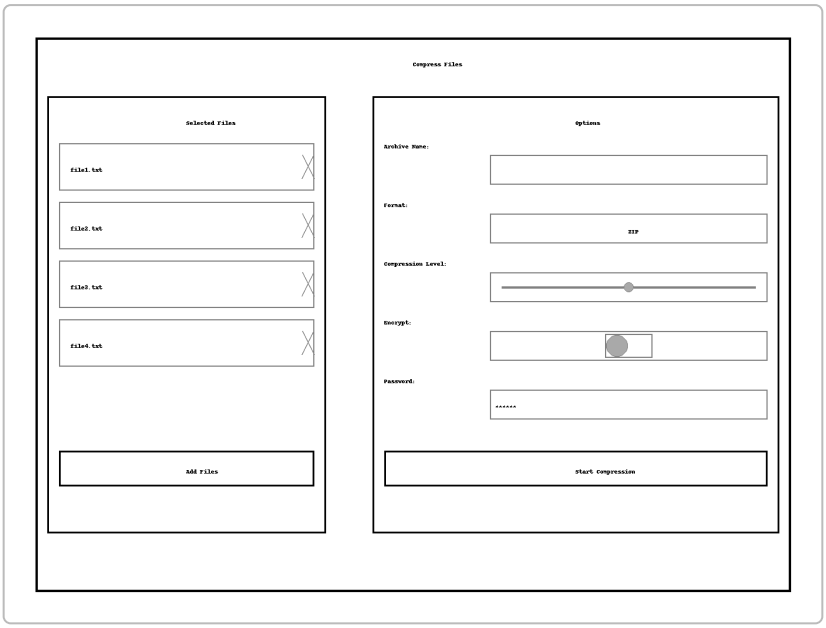
The iPad app uses split panes for efficiency.

Home / Recent:



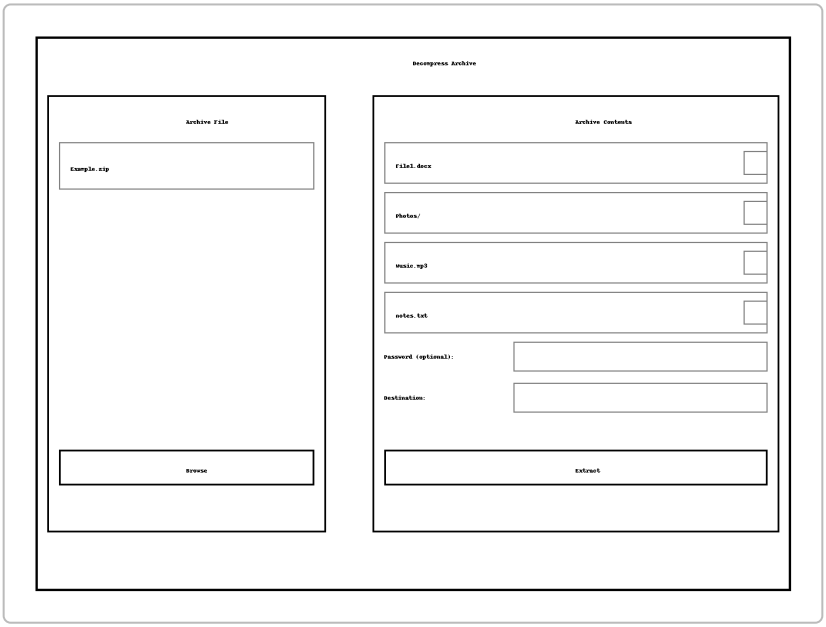
- Sidebar with Home/Recent/Settings; main pane with **Compress** and **Decompress** tiles.

Compress:



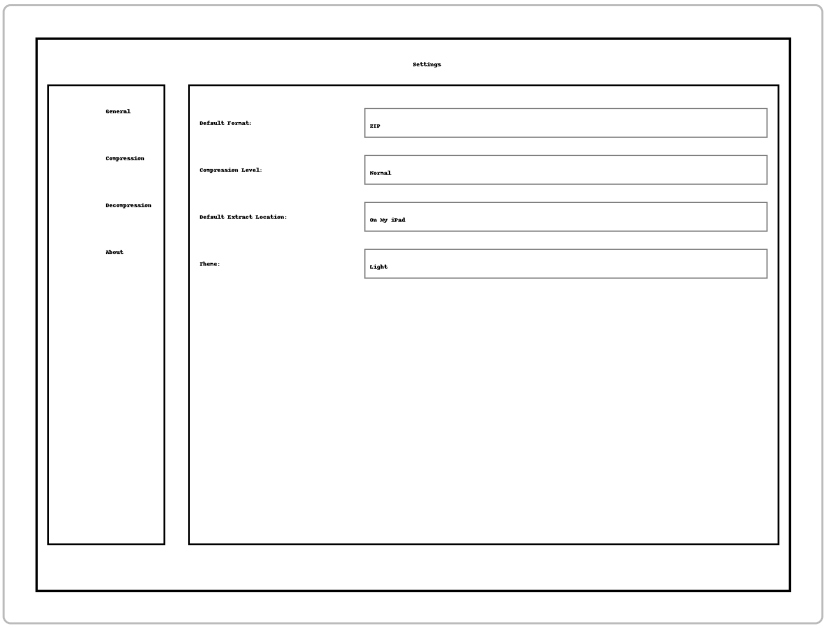
- Left panel lists selected files with remove buttons; right panel contains options (name, format, level, encryption).
- The bottom of the right panel has the **Start Compression** button.

Decompress:



- Left panel shows the selected archive and a **Browse** button; right panel lists archive contents and optional password/destination inputs.

Settings:



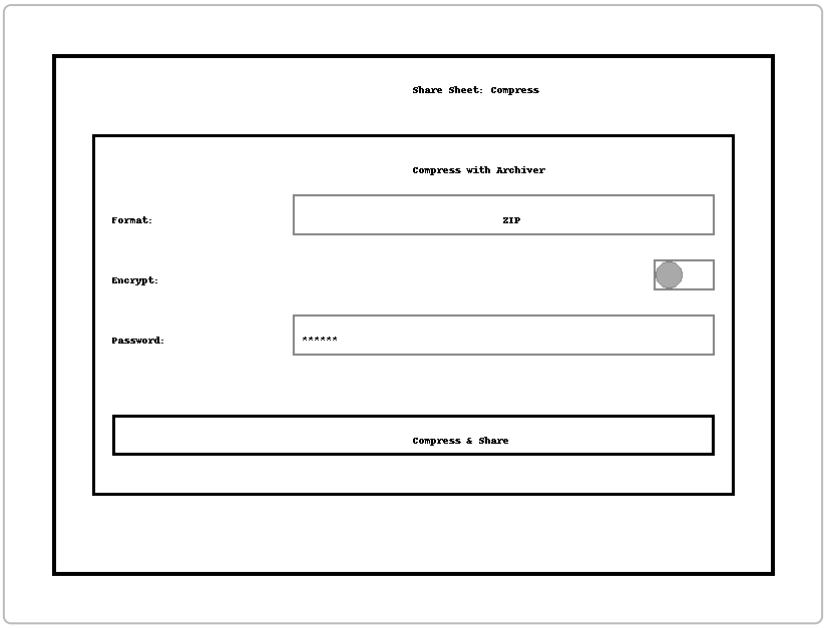
• Sidebar categories and right panel with default settings values.

6.5 Share Sheet & Context Menu Flows

To eliminate the need to open the full app for basic actions, the product must provide compact interfaces within the share sheet and context menus. When invoked from these contexts, the UI is reduced to only the necessary controls and runs as a system-provided extension. The following wireframes illustrate these flows.

Share sheet – Compress

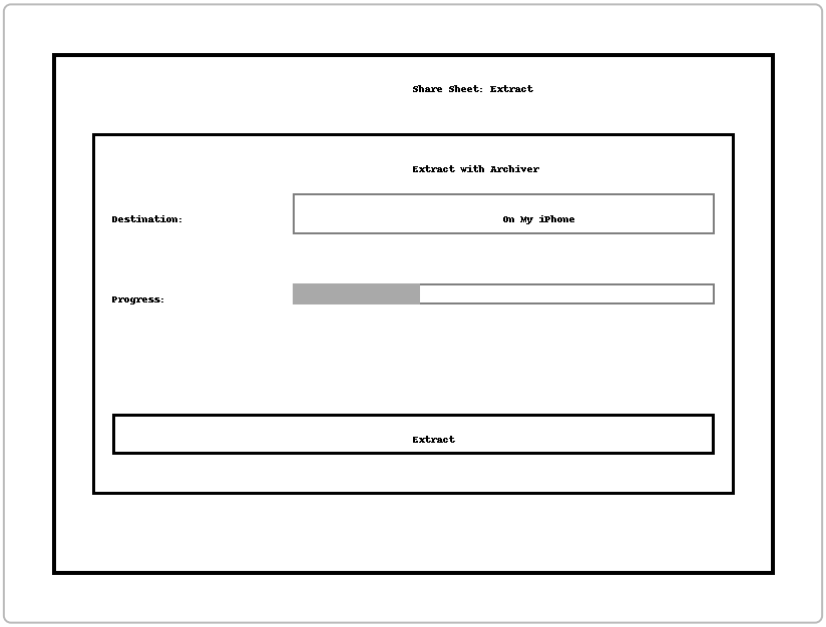
When the user selects one or more files in the Files app (or any other app that exposes a share sheet) and taps **Share**, the share sheet lists actions. Tapping **Compress with Archiver** launches our extension. A minimal modal appears showing just the essential options:



1. **Format selector:** Default to ZIP; drop-down for other supported formats.
2. **Encrypt toggle:** Off by default; if toggled on, a password field appears.
3. **Password field:** Hidden until encryption is enabled; uses secure input.
4. **Compress & Share button:** Starts compression. When complete, the resulting archive is passed back into the share sheet so the user can choose a destination (save to Files, AirDrop, send via Messages, etc.). The share sheet remains open, so the user never leaves the current app.

Share sheet – Extract

If the user shares an archive file (.zip, .7z, etc.), the share sheet displays an **Extract with Archiver** action. Selecting this brings up a minimal extraction panel:

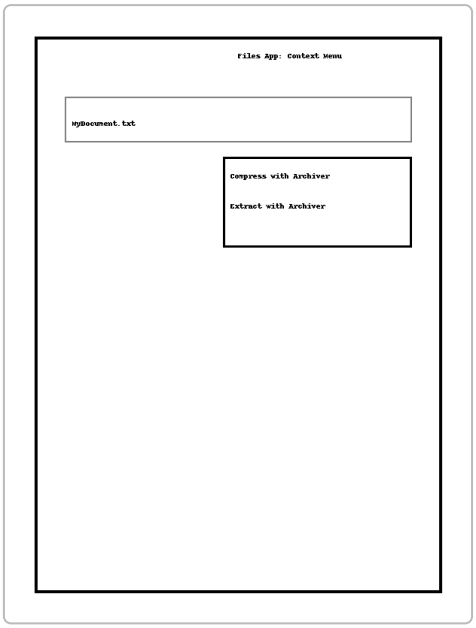


1. **Destination chooser:** Defaults to the current location (e.g., On My iPhone). A picker allows selecting other locations.

2. **Progress bar:** Indicates extraction status; this is necessary because extraction may take time. If the archive is encrypted, a password prompt appears before extraction begins.
3. **Extract button:** Executes extraction. When done, a summary with extracted items is displayed, and the user can choose to open them or return to the share sheet.

Context menu – Quick actions

Long-pressing a file in the Files app reveals contextual actions. The app should register two quick actions: **Compress with Archiver** for arbitrary files and **Extract with Archiver** when the selected item is a supported archive. These actions invoke the same extensions as the share sheet but bypass the initial share sheet UI. The context menu wireframe shows an example:



- When the user taps **Compress with Archiver**, the minimal compress panel (above) appears.
- When the user taps **Extract with Archiver**, the minimal extract panel appears.

These quick actions allow users to compress or extract files directly from the Files app with only one or two taps, meeting the objective of making the app as unobtrusive as possible.

7. Non-Functional Requirements

- **Performance:** Must handle archives up to several gigabytes. Use streaming compression and decompression to minimize memory usage.
- **Security:** Use Apple's Security and CryptoKit frameworks for encryption. Never store passwords in plain text.
- **Accessibility:** Support VoiceOver, Dynamic Type and large touch targets. Provide descriptive labels for share-sheet actions.
- **Localization:** Support multiple languages; all strings must be localizable.
- **Offline:** Core compression and decompression must work without internet.
- **Privacy:** Do not collect user data; comply with Apple's App Store guidelines.

8. Open Questions / Considerations

- **Cloud providers:** Should the app register as a file provider to support cloud services (Dropbox, Google Drive)? How will permissions be handled?
- **Background processing:** Is background execution needed for long compress/decompress operations? Should the app report progress via notifications?
- **Segmented archives:** Which formats support splitting and combining? Provide a simplified UI for this advanced feature.
- **Default format:** Determine the default format (likely ZIP) and whether to remember user choice.

9. Acceptance Criteria

- **Compression**
- Given files selected in the Files app share sheet, when the user taps “Compress” and selects a format, then the app produces an archive in the chosen format and presents it for sharing.
- Given the user enables encryption and provides a password, the resulting archive is password-protected and cannot be opened without the password.

- **Decompression**

- Given an archive in any supported format, when the user taps “Extract” from the share sheet, then the app prompts for password if needed and extracts contents into the chosen folder.
- Given the user selects only specific items within an archive, only those items are extracted.

- **Integration**

- The app appears as an action in the iOS share sheet for files and archives.
- Long-press context menu in Files shows “Compress with Archiver” and “Extract with Archiver” for all supported formats.
- The app is available as a location in the Files app (file provider extension) to browse archive contents.

- **Settings**

- The user can set default archive format, compression level, extraction location and theme from the settings page.
- These defaults persist across sessions and are used by share-sheet actions unless overridden.

- **Accessibility & Localization**

- All UI elements have accessibility labels and support Dynamic Type. The app is fully usable via VoiceOver.
- The app is localizable; at least English and German strings are provided for launch.

10. Implementation Notes

- **Platform:** Only iOS 18 and iPadOS 18 need to be supported. Use **Swift 5.9+** and **SwiftUI 5** for all on-device UI. SwiftUI's latest features in iOS 18 (including improved `Sheet` and `PresentationDetents`) allow building compact modals for share-sheet extensions.
- **File picker & share sheet:** Use `UIDocumentPickerViewController` (in SwiftUI via `DocumentPicker`) for selecting files in the main app. For share-sheet actions, register a `UIActivityExtension` (also called **Action Extension**) so that the app appears in the system share sheet. The extension's interface must be lightweight and built with SwiftUI, adhering to the memory/time limits imposed on share-sheet extensions.
- **Context menus & quick actions:** On iOS 18, define `UIAction` items using `UIMenu` and register them via the `UIContextMenuInteraction` API. In SwiftUI, this can be achieved with `.contextMenu` modifier. For Files integration, implement `NSFileProviderActionExtension` so quick actions appear when long-pressing files.
- **File provider:** To let users browse archives like folders, implement an `NSFileProviderExtension` with a custom domain. This allows the app to expose archived content as a file system view inside the Files app. iOS 18 adds improved performance for file provider extensions.
- **Compression library:** Use Apple's `Compression` framework for supported algorithms (LZMA, zlib). For formats beyond built-in support (e.g., 7-Zip), integrate a mature open-source library such as **libarchive** or **ZIPFoundation** (MIT licensed). These libraries handle streaming compression and decompression without loading entire files into memory.
- **Encryption:** Use `CryptoKit` for AES-256 encryption and secure random number generation. When encrypting archives, ensure keys and passwords never persist beyond the operation. Use a sealed box (AEAD) mode when format supports it (e.g., 7-Zip AES encryption).
- **Uniform Type Identifiers:** Employ the `UniformTypeIdentifiers` framework (UTType) to declare supported archive types and register the app to handle them. This ensures the share sheet and Files app know which files the app can open.
- **Quick look & thumbnails:** Use `QuickLookThumbnailing` to generate previews of archived files in the app's preview list and in the file provider view. This helps users identify files before extraction.
- **Background tasks:** Use `URLSessionTask` or `Task` concurrency to perform compression/decompression off the main thread. For long tasks triggered from the share sheet, use `BGProcessingTaskRequest` to continue work if the user switches apps.
- **Accessibility & theming:** Adopt SwiftUI's built-in accessibility modifiers (`.accessibilityLabel`, `.accessibilityHint`) and dynamic type support. Provide both light and dark appearance; adopt system accent colors.
- **Localization:** Use `Localizable.strings` and `SwiftGen` or similar tools to manage localizable strings. Support at least English and German.
- **Packaging & distribution:** Because the app is non-subscription, offer a one-time purchase or free model with optional IAP for additional formats. Comply with App Store Review guidelines regarding share-sheet extension length (must complete actions within ~30 seconds) and memory consumption.

---

[1] [2] [5] Zip and Unzip Files and Folders on iPhone and iPad - MacRumors
https://www.macrumors.com/how-to/zip-and-unzip-files-iphone-ipad/

[3] [4] Best file compression apps for macOS and iOS | AppleInsider
https://appleinsider.com/inside/macos/best/best-file-compression-apps-for-macos-and-ios