Warranty Wallet

Never Lose a Warranty Again

App Description 2
App Overview **2**
Key Features **2**
1. Smart Receipt Scanning with OCR 2
2. Automatic Warranty & Return Policy Lookup 2
3. Organized Warranty Dashboard 3
4. Detailed Warranty Information Views 3
5. Receipt Image Management 3
6. Data Organisation 4
Screenshots & Visuals **4**
Screenshot 1: Home Screen / Dashboard 4
Screenshot 2: Add Item View - Camera Selection 5
Screenshot 3: Add Item View - Receipt Capture 6
Screenshot 4: Item Detail View - Main Information 6
Screenshot 5: Warranty Detail View 7
Screenshot 6: Return Policy Detail View 7
User Instructions **8**
Getting Started 8
Managing Warranties 11
Sharing Options 12
Technical Details **12**
Technology Stack 12
App Architecture 13
Core Data Model 16
Data Flow 17
API Integration 18
Performance & Constraints 18
System Extensions **19**
1. Share Warranty Card Extension 19
Purpose: Allow users to share formatted warranty information 19
2. Action Extension - Add from Photos 20
Purpose: Quickly add receipts from Photos app without leaving the app 20
3. Future Consideration: Siri Shortcuts (Not Implemented because we didn't have apple developer account) 20
What's Actually Possible with Current Code **21**
Conclusion **21**

App Description

WarrantyWallet is your personal warranty management assistant. We know how frustrating it is to lose receipts or forget when your warranty expires. That's why we built WarrantyWallet - an intelligent iOS app that turns your receipt photos into organized warranty records.

Simply take a photo of your receipt, and our advanced AI technology automatically extracts the item name, store, price, and purchase date. We then search the web to find the exact warranty period and return policy for your product, all while keeping everything organized in your personal wallet.

With WarrantyWallet, you'll never wonder "Does my warranty cover this?" or "Can I still return this item?" again. Easily access your warranty conditions with one tap, and find return policy information when you need it most.

App Overview

App Name: WarrantyWallet
Category: Business & Utilities
Platforms: iOS (iPhone)
Minimum Requirements: iOS 26.0

Key Features

1. Smart Receipt Scanning with OCR

- Uses Apple Vision Framework for text extraction from receipt images
- Powered by OpenAI's advanced language model (GPT-4o-mini)
- Automatically extracts:
    - Item name
    - Store name
    - Purchase price
    - Purchase date (in DD-MM-YYYY format)
- Works with receipts from any retailer
- Clean-up and correction of OCR errors by AI

2. Automatic Warranty & Return Policy Lookup

- Uses OpenAI API with web search integration
- Searches for Australia-specific warranty policies
- Returns structured data including:
    - Warranty period in months
    - Warranty conditions and exclusions

    - Return window in days
    - Return policy conditions
    - Evidence URLs to policy pages
- Two-step process: Initial web search, then structured parsing

3. Organized Warranty Dashboard

- View all warranties at a glance with real-time status
- Status indicators: Active, Expiring Soon, Expired
- Summary statistics:
    - Total items count
    - Active warranties count
    - Expired warranties count
- Search functionality to filter by item or store name
- Sorted by creation date (newest first)

4. Detailed Warranty Information Views

Main Detail View Shows:

- Receipt image with pinch-to-zoom capability
- Purchase details (store, price, purchase date)
- Summary cards for Warranty and Return information
- Return window status with days remaining
- Additional information (warranty period in months, return window in days)

Warranty Detail View Shows:

- Warranty status (Active/Expired)
- Days remaining until expiration
- Expiration date
- Warranty conditions (if provided)
- Link to full warranty policy

Return Policy Detail View Shows:

- Return window status (Active/Expired)
- Days remaining for returns
- Return expiration date
- Return conditions (if provided)
- Link to full return policy

5. Receipt Image Management

- Store receipt photos with warranty records
- Full-screen image viewer
- Pinch-to-zoom capability for detailed viewing
- Binary storage in Core Data

6. Data Organisation

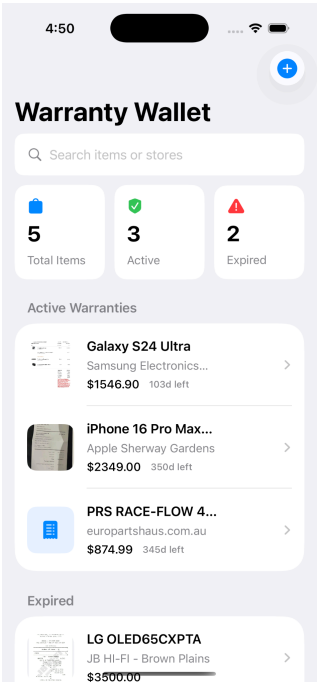
- Warranty items stored in Core Data
- Persistent storage across app sessions
- Each warranty includes:
    - Item details (name, store, price)
    - Purchase date
    - Warranty information (period, conditions, URL)
    - Return policy information (window, conditions, URL)
    - Receipt image
    - Creation and update timestamps

Screenshots & Visuals

Screenshot 1: Home Screen / Dashboard

Description: Main screen showing warranty collection

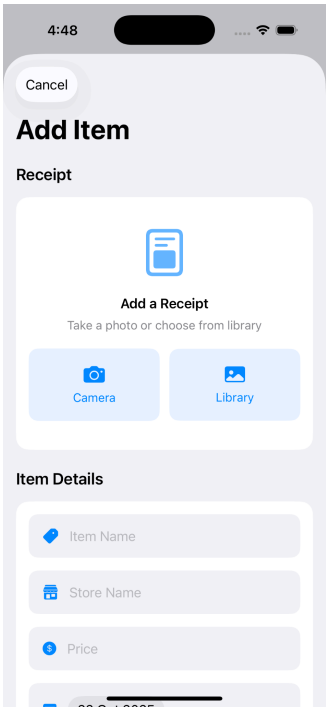
- Navigation header: "Warranty Wallet" with Add button (+)
- Summary cards showing:
    - Total items count
    - Active warranties count
    - Expired warranties count
- Search bar to filter warranties
- Warranty items listed with:
    - Receipt thumbnail or placeholder icon
    - Item name
    - Store name
    - Price
    - Days remaining
    - Warranty status badge (Active/Expired)



Screenshot 2: Add Item View - Camera Selection

Description: Initial screen for adding new warranty

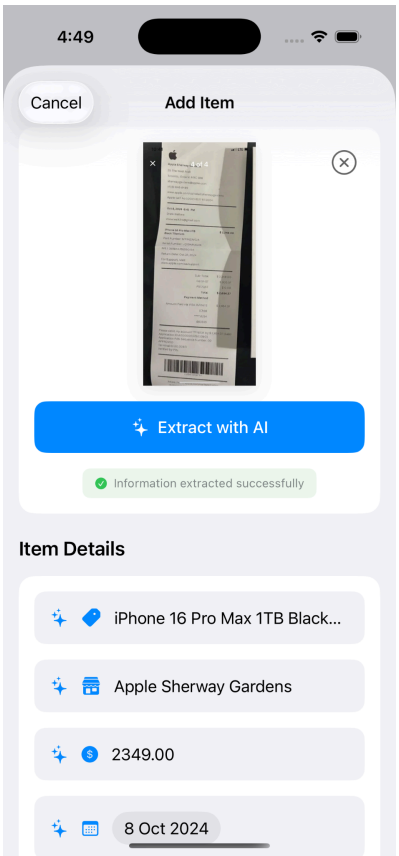
- Large "Add a Receipt" section with:
  - Receipt icon
  - Title and subtitle
  - Two buttons: "Camera" and "Library"
- Navigation with Cancel button



Screenshot 3: Add Item View - Receipt Capture

Description: After selecting receipt image

- Receipt image preview
- "Extract with AI" button
- Success message: "Information extracted successfully"
- Item Details section with fields:
    - Item Name
    - Store Name
    - Price
    - Purchase Date picker
- Warranty & Returns section:
    - Warranty Period (months)
    - Return Window (days)
    - "Find Policy Online" button



Screenshot 4: Item Detail View - Main Information

Description: Comprehensive warranty details page

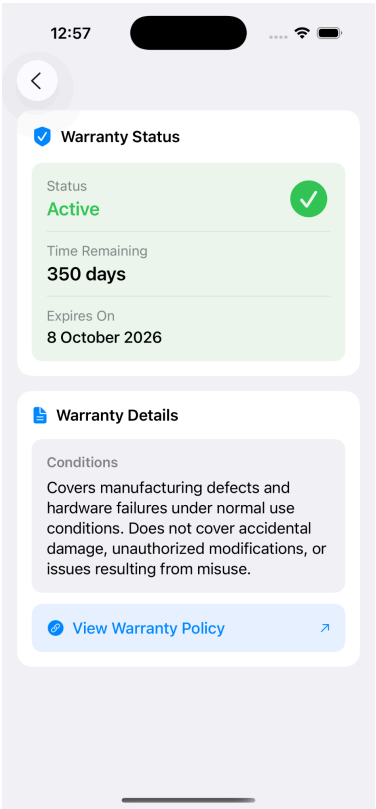
- Item name as title with Edit button
- Receipt image at top (tap to expand)

- Purchase Details card:
    - Store: europartshaus.com.au
    - Price: $874.99
    - Purchase Date: 2 October 2025
- Two side-by-side summary cards:
    - Warranty card (expires date)
    - Return card (expires date)
- Return Window card with status
- Additional Information section

Screenshot 5: Warranty Detail View

Description: In-depth warranty information

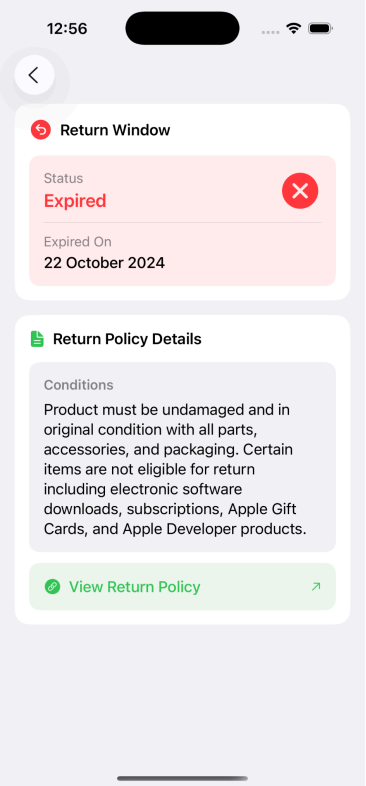
- Back button for navigation
- Warranty Status card showing:
    - Status (Active/Expired)
    - Time Remaining (days)
    - Expires On (date)
- Warranty Details card showing:
    - Conditions text
    - View Warranty Policy link
- Clean, readable layout



Screenshot 6: Return Policy Detail View

Description: In-depth return policy information

- Back button for navigation
- Return Window card showing:
  - Status (Active/Expired)
  - Time Remaining (days)
  - Expires On (date)
- Return Policy Details card showing:
  - Conditions text
  - View Return Policy link
- Consistent layout with warranty view



User Instructions

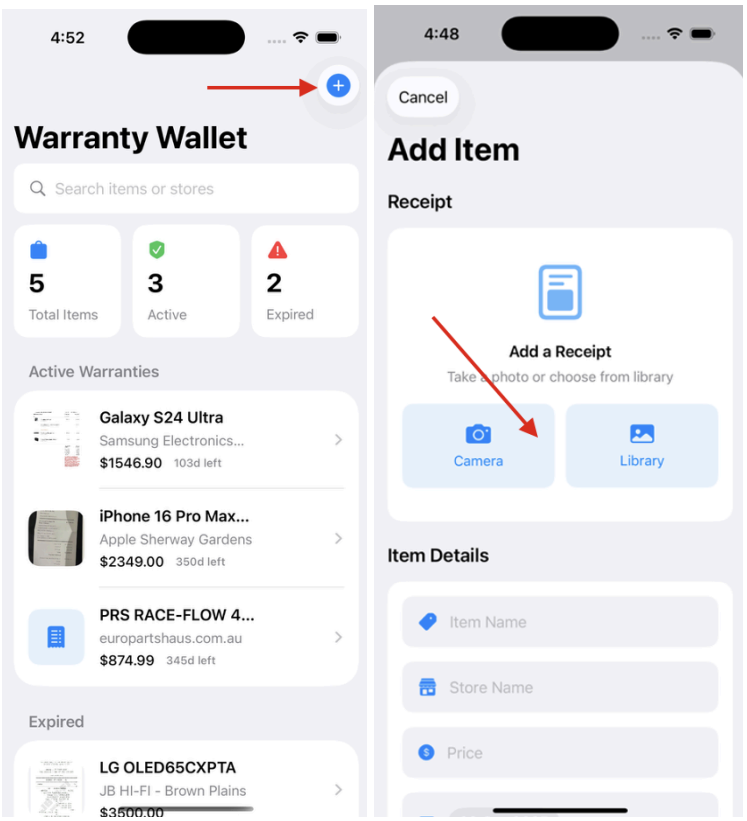
Getting Started

1. Download & Install

- Git clone WarrantyWallet
- Install on your iPhone (iOS 26.0+)
- Grant camera and photo library permissions when prompted

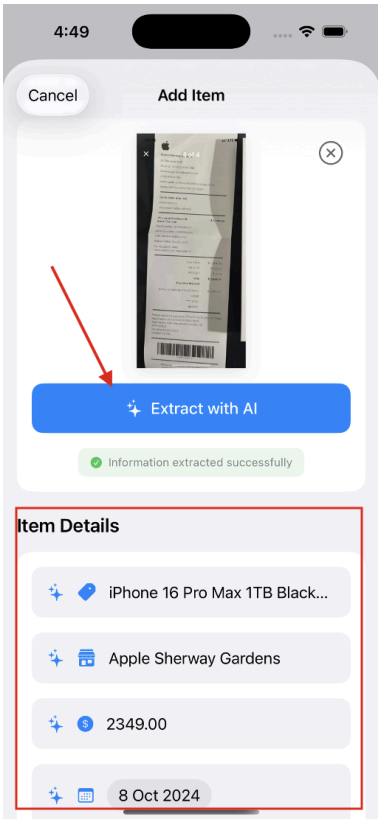
2. Add Your First Warranty

- Tap the "+" button in the top-right corner of the home screen
- Choose either:
    - **Camera:** Take a new photo of your receipt
    - **Library:** Select an existing receipt from Photos
- Ensure the receipt is clear and well-lit
- Tap "Extract with AI" to scan the receipt



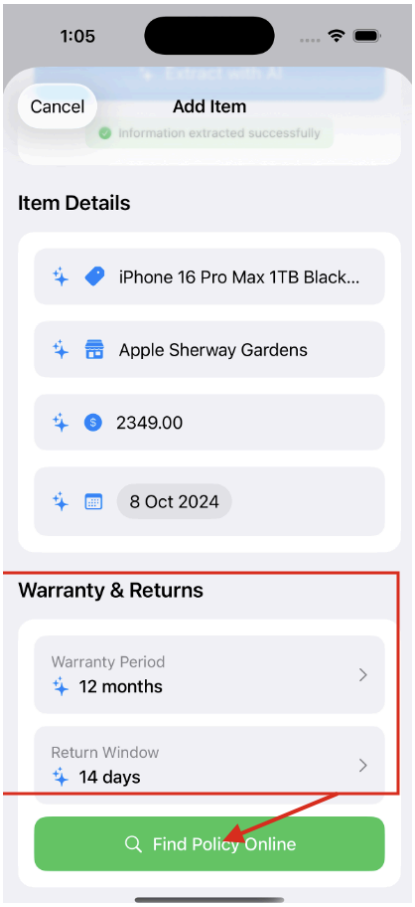
3. Review Extracted Information

- The app automatically extracts:
    - Item name
    - Store name
    - Price
    - Purchase date
- Verify all fields are correct
- Edit any fields manually if needed by tapping on them



4. Find Warranty Information

- Tap "Find Policy Online" button
- The app searches for:
    - Warranty details for your item
    - Return policy for the store
- Warranty period and return window are automatically populated
- Review warranty and return conditions



5. Save Your Warranty

- Once all information is correct, tap "Save Item"
- Your warranty is now stored and organized

Managing Warranties

Viewing Warranty Details:

- From home screen, tap any warranty item
- View purchase details card
- Tap the "Warranty" card to see:
    - Current warranty status
    - Days remaining
    - Warranty conditions
    - Link to full warranty policy
- Tap the "Return" card to see:
    - Return window status
    - Days remaining for returns
    - Return conditions
    - Link to return policy

Viewing Return Window:

- On main detail view, see "Return Window" section
- Shows current status (Active/Expired)
- Displays time remaining and expiration date

Searching Warranties:

- Use search bar on home screen
- Type item name or store name
- Results filter in real-time
- Clear search with X button

Viewing Receipt:

- On detail view, tap receipt image
- Opens full-screen viewer
- Use pinch-to-zoom to see details
- Tap X to close

Deleting Warranties:

- On home screen, swipe warranty item left
- Tap Delete button
- Or use Edit mode to select multiple items

Sharing Options

Share Warranty Card:

- Tap Share button on warranty detail view
- Select destination (Messages, Mail, etc.)
- Warranty information is formatted and sent

Add from Photo:

- Open Photos app
- Select receipt image
- Tap Share button
- Select "Add to WarrantyWallet"
- App opens with image pre-loaded

Technical Details

Technology Stack

Frontend:

- **SwiftUI** - Modern UI framework for iOS

- **iOS 26.0+** - Minimum deployment target
- **Core Data** - Local data persistence

Backend & APIs:

- **OpenAI API** - GPT-4o-mini model for text processing
- **Vision Framework** - Apple's OCR for text extraction
- **Web Search Integration** - Search warranty policies

Core Services:

1. **OCRService**
    - Extracts text from receipt images using Vision Framework
    - Implements VNRecognizeTextRequest
    - Supports accurate text recognition level
    - Language: English
    - Returns raw extracted text
2. **OpenAIService**
    - Processes extracted text through OpenAI API
    - Uses fetchJSONResponse with responses API
    - Can call web_search for policy lookup
    - Parses responses into structured JSON
    - Two main functions:
        - extractReceiptData() - Gets item details
        - processWarrantyInfo() - Gets warranty details
        - processReturnPolicyInfo() - Gets return policy
3. **WarrantyService**
    - Manages Core Data operations
    - Creates warranty items with all details
    - Calculates warranty end dates
    - Calculates return end dates
    - Determines warranty status (Active/Expiring Soon/Expired)
    - Determines return status (Active/Expiring Soon/Expired)

App Architecture

WarrantyWallet/

├── Models/

│ └── ReceiptData.swift

│ - itemName, storeName, price, purchaseDate

│ - formattedPrice computed property

```
│       - parsedDate computed property
│
├── Services/
│   ├── OCRService.swift
│   │   - extractTextFromImage (Data or UIImage)
│   │   - Vision Framework integration
│   │
│   ├── OpenAIService.swift
│   │   - extractReceiptData()
│   │   - processWarrantyInfo()
│   │   - processReturnPolicyInfo()
│   │   - fetchJSONResponse() generic method
│   │
│   └── WarrantyService.swift
│       - createWarrantyItem()
│       - processReceiptImage()
│       - getWarrantyStatus()
│       - getReturnStatus()
│       - calculateWarrantyEndDate()
│       - calculateReturnEndDate()
│
├── Views/
│   ├── ContentView.swift
│   │   - Home screen with warranty list
│   │   - Summary cards
│   │   - Search functionality
```

```
│   │
│   ├── AddWarrantyItemView.swift
│   │   - Camera/library selection
│   │   - Receipt image display
│   │   - Form fields for warranty details
│   │   - AI extraction button
│   │   - Find policy online button
│   │
│   ├── WarrantyItemDetailView.swift
│   │   - Main warranty detail view
│   │   - Purchase details
│   │   - Warranty/Return summary cards
│   │   - Return window status
│   │   - Delete functionality
│   │
│   ├── WarrantyDetailView.swift
│   │   - Warranty status card
│   │   - Warranty conditions
│   │   - Warranty policy link
│   │
│   ├── ReturnPolicyDetailView.swift
│   │   - Return window status card
│   │   - Return conditions
│   │   - Return policy link
│   │
│   └── EditWarrantyDetailsView.swift
```

```
│       - Edit warranty period
│       - Edit return window
│       - Edit conditions
│       - Edit evidence URLs
│
├── Utils/
│   ├── Config.swift
│   │   - OpenAI API key configuration
│   │   - Default warranty/return values
│   │   - App configuration constants
│   │
│   ├── Secrets.swift
│   │   - API key storage (empty placeholder)
│   │
│   └── Persistence.swift
│       - Core Data setup
│       - PersistenceController singleton
│
└── WarrantyWalletApp.swift
    - App entry point
    - Core Data context setup
```

Core Data Model

WarrantyItem Entity:

- `id` (UUID) - Unique identifier
- `itemName` (String, optional) - Product name
- `storeName` (String, optional) - Retailer name
- `price` (Double) - Item price
- `purchaseDate` (Date, optional) - Purchase date
- `receiptImageData` (Binary, optional) - Receipt photo
- `extractedText` (String, optional) - Raw OCR text
- `warrantyLengthMonths` (Int16) - Warranty period in months
- `warrantyEndDate` (Date, optional) - Calculated expiration
- `warrantyConditions` (String, optional) - Warranty terms
- `warrantyEvidenceUrl` (String, optional) - Policy link
- `returnWindowDays` (Int16) - Return period in days
- `returnEndDate` (Date, optional) - Calculated return deadline
- `returnConditions` (String, optional) - Return terms
- `returnEvidenceUrl` (String, optional) - Policy link
- `createdAt` (Date, optional) - When added
- `updatedAt` (Date, optional) - Last modified

Data Flow

Receipt Scanning:

1. User selects receipt image (camera or library)
2. Image converted to Data/UIImage
3. OCRService.extractTextFromImage() called
4. Vision Framework extracts text
5. Text returned to WarrantyService

Receipt Data Extraction:

1. Extracted text sent to OpenAIService.extractReceiptData()
2. OpenAI API parses text into structured data
3. Returns: ReceiptData (itemName, storeName, price, purchaseDate)
4. Form fields auto-populated

Warranty & Return Lookup:

1. User taps "Find Policy Online"
2. WarrantyService.findWarrantyInfo() called
3. OpenAIService.processWarrantyInfo() with web_search
4. Returns: WarrantyInfo (warrantyMonths, conditions, evidenceUrl)
5. WarrantyService.findReturnInfo() called

6. OpenAIService.processReturnPolicyInfo() with web_search
7. Returns: ReturnPolicyInfo (returnDays, conditions, evidenceUrl)
8. Fields updated in UI

Saving Warranty:

1. WarrantyService.createWarrantyItem() called
2. New WarrantyItem created in Core Data
3. Warranty end date calculated (purchaseDate + warrantyLengthMonths)
4. Return end date calculated (purchaseDate + returnWindowDays)
5. Item saved to Core Data
6. View context refreshed, item appears on home screen

API Integration

OpenAI API Responses Format:

```
struct ReceiptData: Codable {

    let itemName: String?

    let storeName: String?

    let price: Double?

    let purchaseDate: String? // "dd-MM-yyyy"

}


struct WarrantyInfo: Codable {

    let warrantyMonths: Int?

    let conditions: String

    let evidenceUrl: String?

}


struct ReturnPolicyInfo: Codable {

    let returnDays: Int?

    let conditions: String

    let evidenceUrl: String?
```

}

Performance & Constraints

Defaults (from Config):

- Default warranty: 12 months
- Default return window: 30 days
- Warranty expiring soon threshold: 30 days
- Return expiring soon threshold: 7 days

Image Handling:

- JPEG compression quality: 0.8
- Variable based on file size requirements
- Stored in Core Data binary field

API Timeouts:

- OpenAI API calls depend on network
- Web search adds 5-8 seconds to lookup
- Two-step process for warranty lookup

System Extensions

1. Share Warranty Card Extension

Purpose: Allow users to share formatted warranty information

Concept:

- User completes warranty entry
- Taps Share button
- Custom share extension generates formatted card
- Shares via Messages, Mail, or Files

Share Format:

WARRANTY CARD

Item: [Item Name]

Store: [Store Name]

Price: $[Price]

Purchase Date: [Date]

Warranty Information:

- Warranty Period: [Months] months

- Warranty End Date: [Date]

- Warranty Conditions: [Conditions]

- Evidence: [URL]

Return Information:

- Return Window: [Days] days

- Return End Date: [Date]

- Return Conditions: [Conditions]

- Evidence: [URL]

Reference: Apple Share Sheet Documentation

2. Action Extension - Add from Photos

Purpose: Quickly add receipts from Photos app without leaving the app

How It Works:

1. User opens receipt in Photos
2. Taps Share button
3. Selects "Add to WarrantyWallet" from actions
4. Image passed to app via app groups
5. AddWarrantyItemView opens with image pre-loaded
6. User continues normal workflow

Implementation Steps:

1. Create Action Extension target
2. Set up App Groups container
3. Handle image passing via shared container
4. Launch main app with image loaded

Reference: Apple App Extensions

3. Future Consideration: Siri Shortcuts (Not Implemented because we didn't have apple developer account)

Potential Commands:

- "Add warranty" - Open add screen
- "Show my warranties" - Open home screen
- Could automate warranty lookups if Siri integration added

Reference: Apple Shortcuts Documentation

What's Actually Possible with Current Code

Current Capabilities:

- Complete receipt-to-warranty workflow
- OpenAI API integration with web search
- Vision Framework OCR
- Core Data persistence
- Beautiful, organized UI
- Warranty status calculation
- Return window tracking
- Receipt image storage and viewing
- Manual warranty detail editing

Not Yet Implemented (Future):

- Push notifications
- Cloud sync/backup
- Share extensions
- Siri shortcuts
- Multiple accounts
- Dark mode optimization
- Export functionality
- Bulk operations
- Edit warranty information

Conclusion

WarrantyWallet combines AI technology with thoughtful design to solve a real problem: keeping track of warranties and return policies. By integrating the OpenAI API with web search capabilities and Apple's Vision Framework, we've created an app that intelligently processes receipt photos and helps users manage their warranties effectively.

The app is built with modern Swift and SwiftUI practices, uses Core Data for reliable local storage, and provides a clean, intuitive interface for managing warranties specific to the Australian market.

Download WarrantyWallet today and take control of your warranties.