SnapDrop Offline · Product Requirement Document

This is a Phase 1 product specification. Its requirements, proposed metrics and future roadmap are presented as written in the source.

هذه وثيقة متطلبات للمرحلة الأولى. تُقرأ المتطلبات والمقاييس المقترحة والخطة المستقبلية بوصفها محتوى الوثيقة الأصلية.

Read the original · 4 source pages · View the project

Page 1 · Offline sharing objective, sender and receiver, and session initiation

هدف المشاركة دون اتصال، المرسل والمستلم وبدء جلسة المشاركة

Product Requirement Document

Project: SnapDrop Offline (Tap-and-Send) • Phase 1 Core

1. Overview & Objective

SnapDrop Offline enables zero-cloud peer-to-peer file sharing directly from the native system share sheet. The sender taps "Share," spawning an ephemeral local hotspot, and broadcasts credentials via NFC. The receiver automatically connects and downloads the payload via a captive portal without any app installation.

2. User Personas

The Sender (Host)

Wants to send large files instantly without relying on active internet access, Bluetooth pairing friction, or cloud data limits. Initiates via Android native share menu.

The Receiver (Cross-Platform)

Wants to receive files seamlessly without the friction of app stores. Needs a pure native flow utilizing standard OS-level Wi-Fi and NFC triggers.

3. Functional Requirements

3.1 System Integration & Initiation

- **Share Sheet Target:** Register ACTION_SEND intent filters supporting */* MIME types to appear globally.
- **Transfer Overlay:** Display a modal dialog presenting the file thumbnail, name, extension, and formatted size.
- **Status UI:** Provide clear state indicators bridging "Starting hotspot..." to "Ready to Tap", alongside an active cancellation mechanism.

Source footer: SnapDrop Offline - PRD · Page 1 of 2

Page 2 · Local hosting, NFC handover, teardown and technical architecture

الشبكة المحلية والتسليم عبر NFC وإنهاء الجلسة والبنية التقنية

3.2 Local Network & File Serving

- **Ephemeral Hotspot:** Trigger Local-Only Hotspot generation on Android 8.0+, dynamically creating a randomized SSID and WPA2 network key.
- **Embedded Server:** Host a lightweight HTTP server bound strictly to the local interface (192.168.49.1:8080) to serve the payload.
- **Captive Portal:** Implement DNS redirection so the receiver's OS browser automatically opens the download landing page upon connecting.

3.3 NFC Handover & Connection

- **NDEF Configuration:** Emulate an NFC tag via HCE broadcasting the application/vnd.wfa.wsc Wi-Fi Simple Configuration record.
- **Payload Details:** Package the SSID, network key, and authentication type seamlessly into the NDEF record to trigger connection prompts.
- **Screen Fallback:** Prominently display a standard Wi-Fi QR code on the sender's screen for devices that restrict automated NFC network joins.

3.4 Teardown & Lifecycle

- **Completion Detection:** Monitor the server stream for successful delivery of the final byte. Display a confirmation on the host device.
- **Auto-Teardown:** Safely terminate the hotspot and HTTP server automatically 30 seconds post-transfer, or after 3 minutes of idle inactivity.

4. Technical Architecture

Component	Technology Stack	Responsibility
Share Interface	ShareTarget / BottomSheet	Intercepts standard system files and renders session UI.
Hotspot Engine	LocalOnlyHotspot API	Spawns unmetered, isolated local Wi-Fi without cell data.
Web Server	Ktor Embedded / NanoHTTPD	Hosts the HTML UI and streams binary files to client.
NFC Service	HostApduService (HCE)	Broadcasts Wi-Fi connection payload to tapping devices.

5. Constraints

Security & Privacy Imperatives

Generate unique WPA credentials for every session. The local server and file streams must be destroyed from system memory immediately following a successful transfer or timeout.

Page 3 · Connection and transfer failures, and the four-step interface flow

أخطاء الاتصال والنقل وخطوات الواجهة من المشاركة إلى الإغلاق

6. Error Handling & Edge Cases

Scenario	System Behavior	User Mitigation
Location Off	Hotspot API fails immediately (Android OS requirement).	Prompt user to enable device location services.
Port 8080 Blocked	Server fails to bind to the local interface.	Auto-retry binding on sequential ports (8081, 8082).
NFC Handover Fails	Receiver taps but no connection is initiated.	Fallback to on-screen QR code scanning.
Transfer Interrupted	Client drops off network before file finishes.	Keep server alive for 3 mins to allow reconnect and resume.
Storage Full (Receiver)	Browser download silently fails or truncates.	Pre-check file size and display warning on UI if possible.

7. User Interface Flow

1. The Share Menu

User selects a file from the gallery and taps "Share". SnapDrop appears as a native target alongside standard apps.

2. Initialization Overlay

A bottom sheet appears. A spinner indicates the OS is provisioning the unmetered Wi-Fi hotspot and binding the HTTP server.

3. Ready State

The screen turns green with a prominent NFC tap icon and a fallback QR code displaying the network credentials.

4. Delivery & Cleanup

Upon HTTP 200 OK completion, the UI updates to a checkmark. Resources are torn down and the overlay dismisses itself.

Page 4 · Success metrics, deferred telemetry and the future roadmap

مؤشرات القياس والنقل المؤجل للبيانات وخطة المراحل اللاحقة

8. Success Metrics & Telemetry

Because SnapDrop is an offline-first, zero-cloud application, telemetry is batched locally and sent only when the host device regains standard internet connectivity. Key metrics include:

- **Transfer Success Rate:** Percentage of initiated share intents that result in a completed HTTP download.
- **Time-to-Ready:** Average duration between tapping the share target and the hotspot successfully broadcasting.
- **Connection Vector:** Ratio of connections initiated via NFC tap versus QR code fallback.
- **Average Payload Size:** Helps dictate future chunking or multipart requirements for large video files.

9. Future Roadmap (Phase 2+)

The core Phase 1 implementation handles single-file 1:1 transfers. Subsequent iterations will expand the capabilities of the protocol:

- **Multi-File Queuing:** Support for ACTION_SEND_MULTIPLE , dynamically zipping payloads on the fly or serving a directory index page.
- **iOS Companion App:** An optional thin iOS client capable of intercepting proprietary NDEF payloads to bypass iOS native Wi-Fi restrictions.
- **Two-Way Transfer:** Updating the captive portal UI to allow the receiver to upload files back to the host device during the same session.

Open Source Release Candidate

This implementation will be documented and open-sourced under the MIT license, acting as a reference architecture for Android developers building localized, offline peer-to-peer sharing solutions.