BY MENDY / 24 SEP 2026

Sichos Kodesh offline: the engineering plan

From source crawl to a bochur's pocket: archive, pack format, size math, app import, download site and yeshiva station, in the order they unblock.

The vision page said what we want: the Rebbe's farbrengens on a bochur's kosher phone, with no WiFi, ever. This is the engineering plan behind it. I read the deployed app, mapped both sources, ran the size math from the catalog's own numbers, and looked at how offline Wikipedia and Linux distributions solve the same problem. Then I planned, in the order things actually unblock each other.

> ⓘ **The one decision that shapes everything.** A month is not a zip. The unit of storage and updates is the individual file, named by its hash. A pack is just a plain zip that carries a selection of those files plus a signed manifest. Month packs are simply the popular selection, prebuilt. Delta updates become set difference, and a custom pack builds in seconds.

What the research found

The app today

Sichos Kodesh is a React PWA with a service worker. The entire catalog is compiled into the JavaScript bundle: 43 source years (5710 to 5752), 3,330 occasions, 6,454 audio tracks and 13,975 hanacha references pointing at 12,656 unique PDFs. Every track carries its duration, which is what makes the size math below exact rather than guessed.

All media flows through one Cloudflare Worker: audio from JEM's CloudFront and Ashreinu's S3, hanachos from Google Drive. Offline storage is the browser Cache API keyed by proxy URL, filled either by a cache-first service worker route or by the explicit download manager. The player is a plain audio element; the offline check is "is this URL in the cache". The download settings already model the taxonomy we need: audio, hanacha editions (Sichos Kodesh, Sefer, Toras Menachem) and the sections mugah, bilti mugah, maamorim, hagahos, hosofos.

The sources

- **mafteiach.app** renders its per-date pages client-side, so the existing indexer is the crawl path. Months run 01 to 12 with 06A and 06B for Adar in a leap year.
- **Ashreinu** is an Angular app over a Lambda API with years, dates, events, audio recordings, transcripts, links and playlists. Beyond audio there are transcripts and playlists worth archiving too.
- **Source bitrates** (sampled): Ashreinu files are a uniform 42 kbps; JEM files run 16 to 71 kbps, mostly 32 or 64. Everything is already lossy MP3, mostly speech.
- **PDFs** average about 430 KB, so the hanachos total roughly 6 GB.

> **Two data-quality findings.** 259 hanachos use legacy Drive IDs (starting with 0B) and the ones I sampled fail with a 502 through the proxy today. And about 9% of PDF references point at a file another occasion already uses, which is exactly why files should be stored once, by hash.

Real archive size

The catalog sums to 1,999 hours of audio. At the sources' mixed 32 to 64 kbps that is 35 to 55 GB, consistent with the "50 GB plus" figure. The vision page says 2,722 hours; the gap is one of the open questions below.

The size math

Farbrengens are speech. For voice, Opus at 24 to 32 kbps mono is the practical sweet spot, and because the sources are already 32 to 42 kbps MP3, re-encoding to 24 kbps Opus loses very little you can hear and halves the bytes. Sizes below include the month's hanachos at 430 KB each.

SCOPE	HOURS	AT 24 KBPS (DEFAULT)
MEDIAN SOURCE-MONTH	3.4	57 MB
AVERAGE SOURCE-MONTH	5.0	66 MB
TISHREI 5736	5.6	74 MB
A BIG MONTH (90TH PERCENTILE)	8.4	99 MB
LARGEST MONTH: TISHREI 5748	20.5	250 MB
WHOLE ARCHIVE, AUDIO ONLY	1,999	21.6 GB
WHOLE ARCHIVE WITH HANACHOS		28 GB

PROFILE	AVERAGE MONTH	WHOLE ARCHIVE
16 KBPS, COMPACT	48 MB	14.4 GB audio
24 KBPS, DEFAULT	66 MB	21.6 GB audio
32 KBPS, CLEAR	84 MB	28.8 GB audio
48 KBPS	120 MB	43.2 GB audio

The 115 MB month figure from the vision page corresponds to 32 kbps on a larger-than-median month; the 300 MB Tishrei matches 32 kbps on a late-5740s Tishrei. At 24 kbps the whole archive with hanachos is 28 GB, so a "kol hashanim" card for a yeshiva is realistic.

Codec choice

Opus, mono, variable bitrate, encoded in "audio" mode rather than "voip" (voip narrows the bandwidth and the farbrengens have nigunim and crowd sound). Container: Ogg, decoded natively by Android since 5.0 and by every Chromium WebView. The 572 tracks labelled Nigun get the next profile up automatically.

Hanachos

Ship the original bytes. Recompressing scanned Hebrew pages can destroy legibility for a 30% saving. When a pack is over its size target, trim by section (hosofos and hagahos first), never by re-encoding.

The pack format

- **Object:** one media file, identified by the SHA-256 of its bytes. Audio objects are the encoded Opus file; PDF objects are the original. Immutable, stored once.
- **Pack (.skpack):** a plain zip with uncompressed entries: `manifest.json` first, then `objects/<hash>`. Explorer opens it, Android streams it, and a half-copied pack simply fails verification. Big scopes split into numbered parts, each valid on its own.
- **Manifest:** catalog version, encoding profile, scope, and for every occasion the list of tracks and hanachos with their hash, size and duration. Signed with an Ed25519 key you hold; the app and the station carry the public key.
- **Delta:** the app writes `have.json`, the set of hashes it holds. The station reads it over USB and copies only what is missing. No network, no accounts, no server that knows about any device.

The plan, in dependency order

1. The canonical archive on shmuky.dev (about two weeks)

Fetch everything, not only what packs need. Go directly to CloudFront, S3 and Drive rather than through the Worker: two connections, about one request per second per host, backoff on errors, resumable range downloads, conditional requests on re-sync. The first pull of roughly 20,000 files is about a day.

```
/srv/sk/
  catalog/     one snapshot per crawl, plus a diff log
  raw/objects/ original upstream bytes, by hash, immutable
  raw/index.sqlite   url, hash, size, etag, fetched, source
```

```
derived/<profile>/  encoded audio, keyed on source hash + profile + encoder
packs/      built packs
keys/       the signing key (offline copy elsewhere)
```

Nightly: re-run the indexer, diff the catalog, fetch only what changed, stamp a date-based catalog version. Integrity: hash on ingest, a 2% random scrub weekly, a full scrub quarterly, backups to B2, and a mirror of derived files and packs on Cloudflare R2 (zero egress), which becomes the download site's origin. Fix the legacy Drive IDs here, because this is where truth lives from now on.

2. The pack pipeline (about two weeks)

One command-line tool: transcode (idempotent, parallel ffmpeg), build (scope, profile, sections, size cap), split, verify, and bundle-catalog for the app. Building is just concatenating existing objects into a zip, so a custom pack takes seconds. Prebuilt matrix: 524 source-months at two profiles, about 60 GB on R2, plus "this year's kvius", "every Tishrei", "every Yud Shevat".
Milestone: one hand-built Tishrei 5736 pack, about 74 MB.

3. The mobile app (about three weeks)

Four changes, none of which touch the way a sicha plays with its hanacha in front of you.

1. **A local object store and one indirection.** A resolver returns a local URL if the file is on the device, else the proxy URL. On Android, objects live in app-private storage and are served to the WebView through a custom scheme that honours range requests. Two things break if the native app keeps using the service-worker cache as its store: the route caps at 5,000 entries and will evict imported content (the full archive is 20,000 objects), and Chromium's quota on a 16 GB phone will refuse tens of gigabytes.
2. **Import.** Watch a SichosKodesh/import folder and Downloads. Since the APK is sideloaded, ask for all-files access, with the folder picker as fallback. On resume, on a button, and when the USB cable is pulled: verify the signature, stream each file through SHA-256 into the store, skip what is already held, delete the pack to reclaim space. Then write have.json and device.json for the station. Progress in Hebrew, one line: נוספו 31 שיחות, תשרי תשל"ו.
3. **The catalog as a versioned asset** in the APK, replaceable by a pack that carries a newer one. New sichos appear the moment their pack lands, without an app update.
4. **No network assumptions.** No update checks, no reachability probe, no analytics. Offline is the normal state, not an error.

Milestone: the APK plus the Tishrei pack, installed by copy and paste, with no station at all.

4. download.sk.shmuky.dev (about two weeks, in parallel with 3)

A static Hebrew site over a small Cloudflare Worker, R2 and a build queue. Because every object's size is in the index, the size estimate is exact and updates as you toggle year, month, occasion, date range, profile, sections and editions. Over the target: split into parts, or auto-trim in a fixed order (hosofos, then hagahos, then duplicate editions, then a lower profile, then whole occasions). Custom builds cached by selection hash; popular packs one click; a machine-readable index for the station.

5. The yeshiva station (about four weeks)

Windows first, because that is the office PC. Built with Tauri: a small binary, native access to the Windows Portable Devices API for the phone's USB storage, and the web UI reused from the app.

- **Overnight:** pull the index, download new packs for the yeshiva's scope, verify hashes.
- **Plug in:** recognise the device, read have.json, compute what is missing, show one button: טען תשרי (74 MB). Copy into the import folder; the app imports when the cable comes out. No app on the phone yet? Offer the APK.
- **Dumb players:** if the device is an MP4 player mounting as a drive, export plain MP3 into folders named by Hebrew date and occasion, straight from the manifest.
- **Later:** a sources plugin model so the same station serves other yeshiva content with the same manifest format.

Open questions

- **The mobile wrapper.** Capacitor, a Kotlin WebView or React Native? The object store and custom scheme depend on it.
- **The device matrix.** Which kosher phones exactly: Android version, storage, whether they are Play-certified, whether "all files access" is exposed. Two or three real devices are needed before phase 3 counts as done.
- **2,722 versus 1,999 hours.** Ashreinu content not yet on mafteiach, video, or a counting artifact? It moves the archive by about 10 GB.
- **The 259 legacy Drive IDs:** do resource keys exist, or should the crawler re-resolve them?
- **Key custody,** and whether a pack built by someone else should ever be trusted.
- **A listening test** of the same nigun-heavy sicha at 16, 24 and 32 kbps before 24 is locked as the default.

Risks

- **Android developer verification, starting this month.** On Play-certified devices an unverified sideloaded APK may require a "developer mode" flow with a one-day wait. Mitigation: register as a verified developer for this package, and confirm which kosher ROMs are certified at all; many are not and are unaffected. Settle this with the device matrix.
- **Cache eviction and quota** if the native app keeps the service-worker cache as its store. The 5,000-entry cap is live today.
- **Upstream fragility:** Drive IDs going dark, JEM or Ashreinu changing URLs. The canonical archive exists so downstream never depends on them again.
- **USB transfer on Windows** is slower and flakier than a plain drive. The station resumes and verifies rather than trusting a copy; hash-verified import makes partial copies harmless.
- **Old Android without Opus** is unlikely, but the MP3 profile is the fallback if it shows up.
- **Permission.** Redistributing JEM and mafteiach content as packs is a different act from proxying a stream. The archive's provenance index makes that conversation easy to document.

✓ **The sacred constraints hold.** Nothing on the phone ever opens a socket or asks for an account. Everything the bochur sees is Hebrew, right to left. And a sicha still autoplays with its hanacha in front of him, because the player only ever sees URLs, and the store just answers them locally.

The vision page **Sichos Kodesh today**