APP STORE OPTIMIZATION · BUILT ON GOOGLE SHEETS

The entire ASO stack, running inside one spreadsheet.

Keyword research, competitor scraping, AI metadata generation and bulk publishing to App Store Connect — for both the App Store and Google Play. No server, no subscription, no separate web app.

50	2	1	0
APP STORE LOCALES	APP STORES	CLICK TO PUBLISH	SERVERS TO RUN

Google Apps Script · App Store Connect API · Groq LLM · Screenshots show the live UI with sample data.

Why build ASO tooling in a spreadsheet?

Every serious ASO tool on the market is a web app with a per-seat subscription — you research keywords in one product, keep your metadata in a spreadsheet anyway, then copy it by hand into App Store Connect, locale by locale. The spreadsheet is already where the work lands. So the tooling was built there instead.

The usual workflow

Subscription tool → export CSV → paste into a sheet → re-type every field into App Store Connect for every language → repeat next release.

This workflow

Research, score, generate and publish from the same sheet. One click pushes finished metadata to all 50 App Store locales over Apple's API.

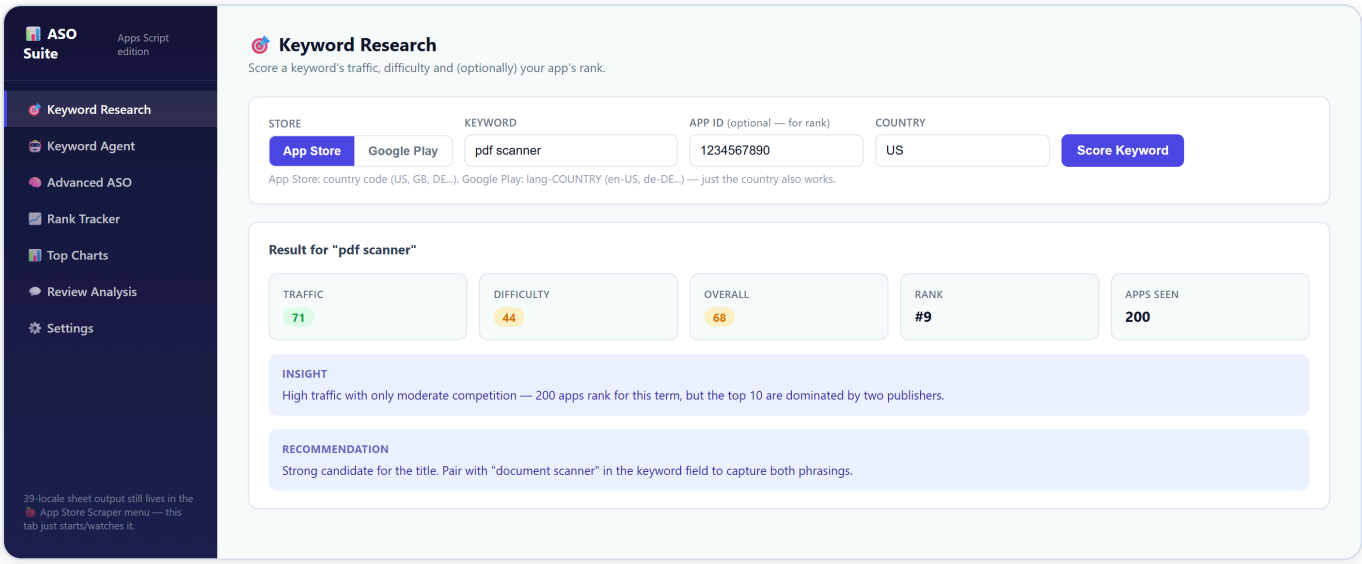
The four core modules

- **Keyword Research** — traffic, difficulty and live rank scoring across both stores.
- **Keyword Agent** — takes one keyword, finds the top competitor apps, and scrapes each across every store locale into per-language sheets.
- **Advanced ASO** — competitors → scored keywords → AI-generated metadata → review.
- **Bulk Upload** — authenticated publishing straight into App Store Connect, all locales at once.

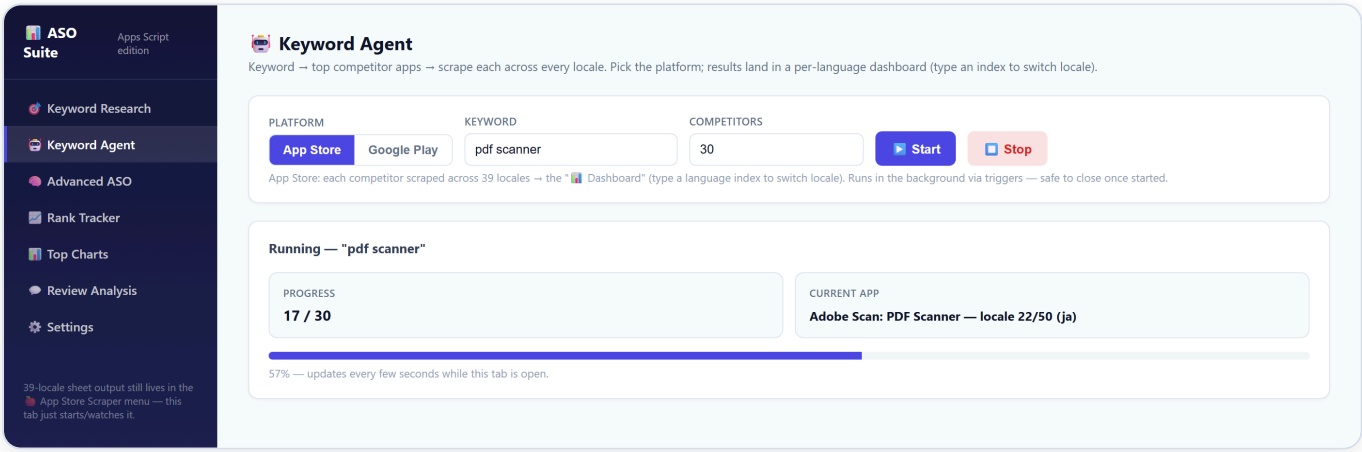
Rank tracking, top-chart snapshots and review analysis run alongside these on overnight triggers.

> **The constraint that made it interesting:** Apps Script has no BigInt, no ECDSA, a six-minute execution ceiling and no background workers. Every one of those had to be worked around to make real store publishing possible from a spreadsheet — see the last page.

Research a keyword, then let an agent do the digging



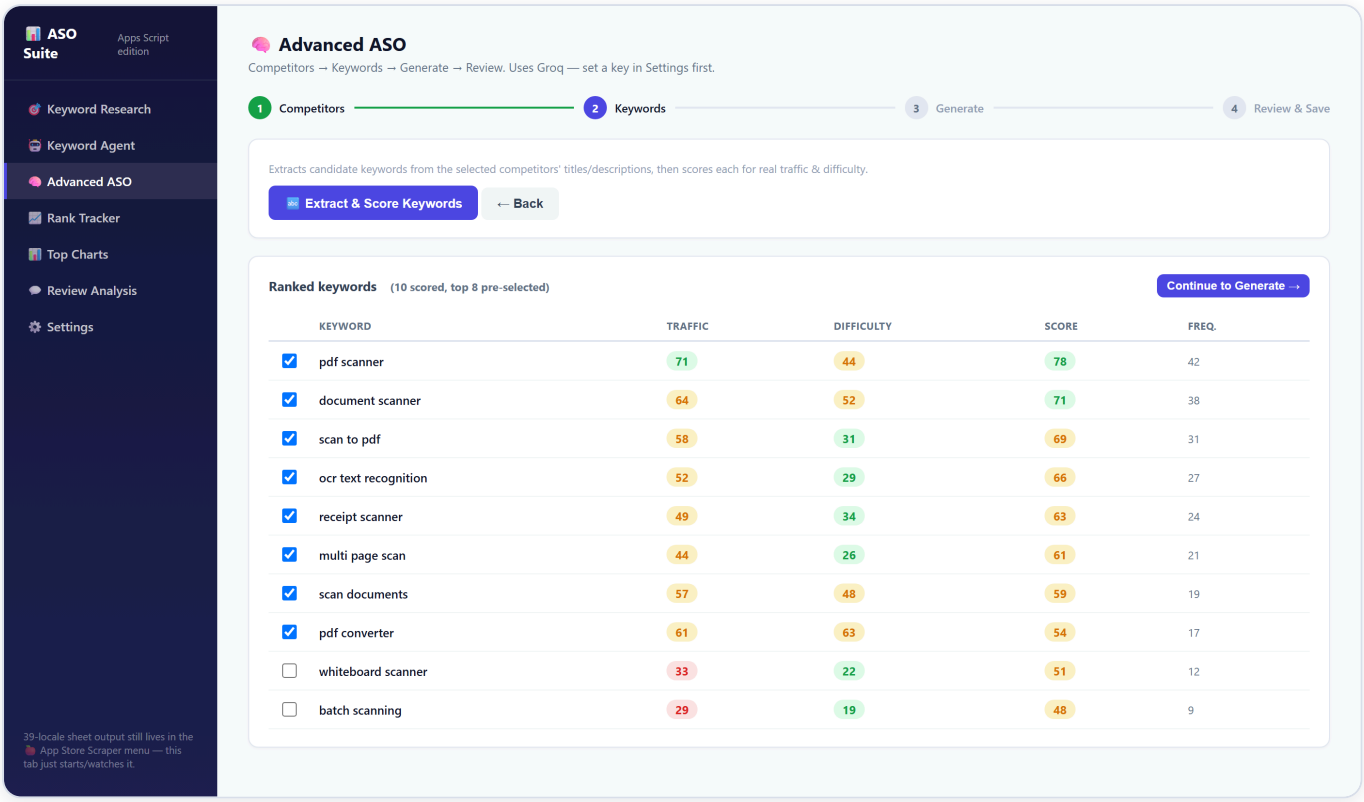
Keyword Research. Scores a keyword's traffic and difficulty, finds your app's current rank, and returns a plain-language insight and recommendation — for the App Store or Google Play, in any country or locale.



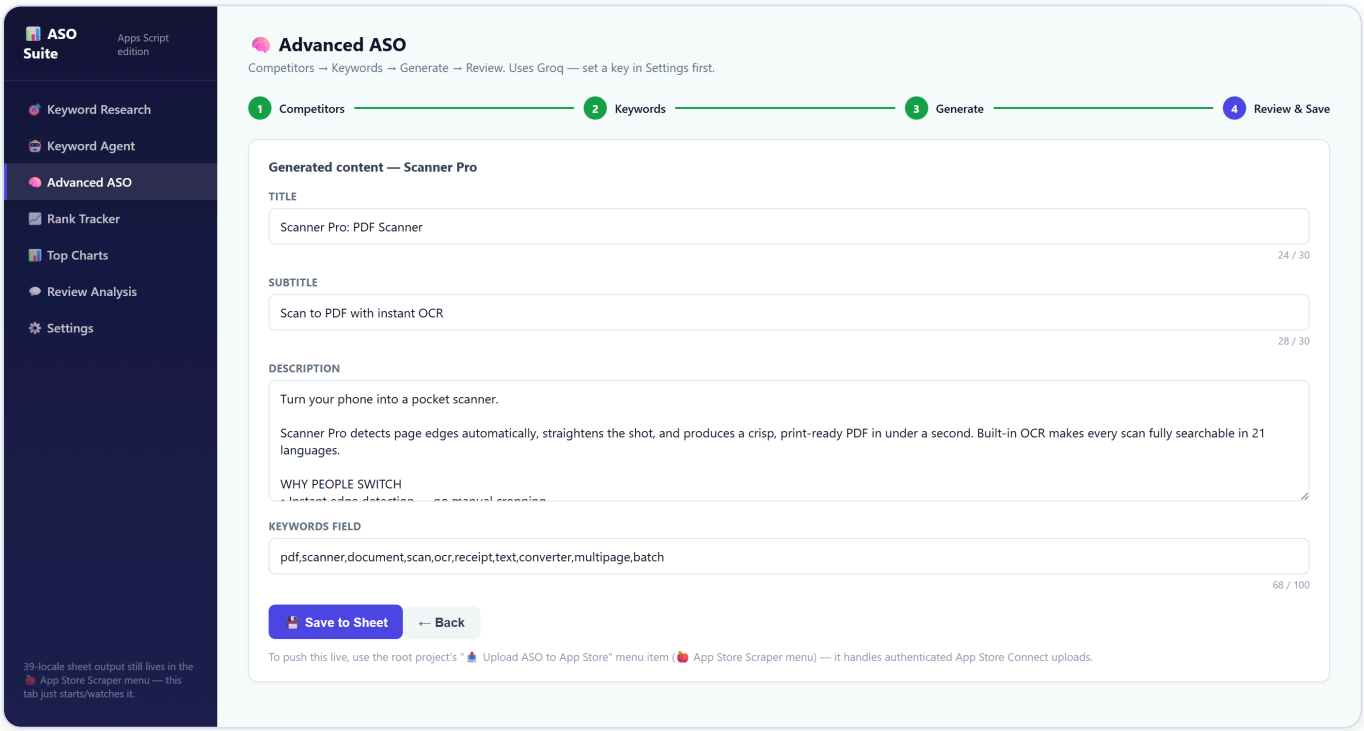
Keyword Agent. Give it a keyword and it finds the top competitor apps, then scrapes every one across all store locales — localized title, subtitle, description, category and rating — into per-language sheets. It runs in the background through chained triggers, so it survives Apps Script's six-minute execution limit and you can close the tab while it works.

From competitor vocabulary to publishable copy

Nothing here is a black box. Every keyword is shown with its numbers, and you choose what feeds the next step.



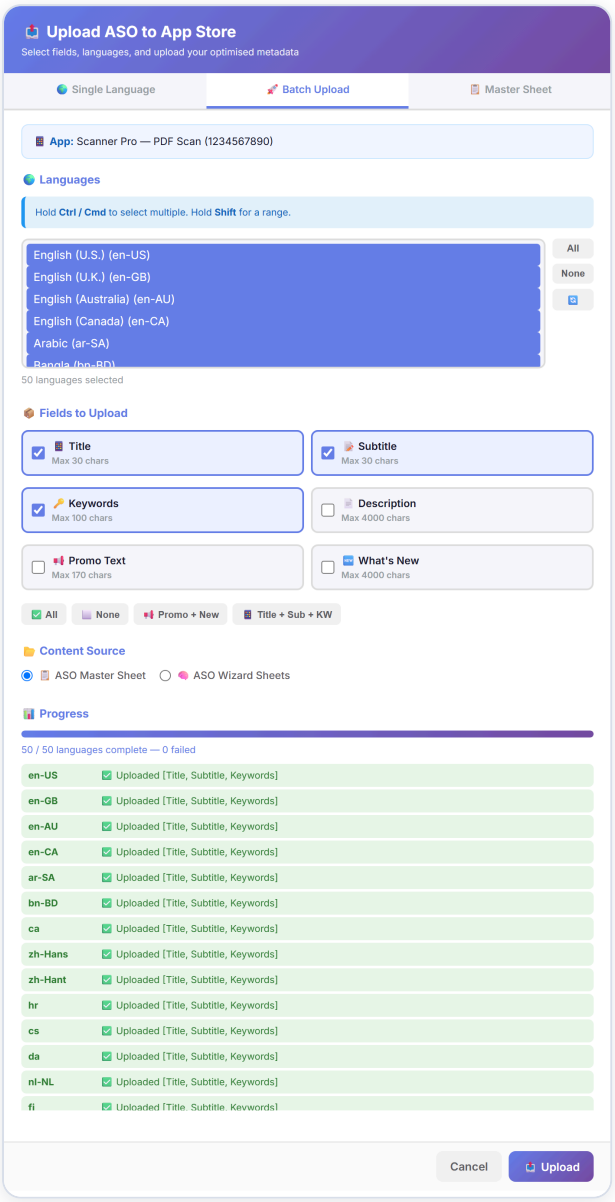
Scored keywords. Pulls the vocabulary out of the selected competitors' listings, filters it through a 29-language stop-word list, then scores each candidate individually for traffic and difficulty. The strongest are pre-selected.



Generated metadata. The chosen keywords go to an LLM, which drafts the title, subtitle, description and keyword field. Each one is editable in place with a live counter against Apple's real limits — 30 characters for title and subtitle, 100 for the keyword field — so nothing gets rejected at upload time for being too long.

One click, fifty locales, straight into App Store Connect

This is the part that normally eats an afternoon. Pick the fields, pick the languages, and the sheet publishes to Apple's API directly — authenticated, locale by locale, with a per-language result log.



Field-level control

Push only what changed. Presets cover the two common cases — "Promo + What's New" between releases, "Title + Subtitle + Keywords" for a metadata refresh — because Apple locks title and subtitle once a version is in review, while promo text stays editable.

Honest per-locale results

Every language reports its own outcome. A rejection on one locale never silently fails the rest, and Apple's actual error is surfaced rather than swallowed.

All 50 supported locales

Including the eleven Apple added recently — Bangla, Gujarati, Kannada, Malayalam, Marathi, Odia, Punjabi, Slovenian, Tamil, Telugu and Urdu.

Under the hood

Apps Script is a deliberately limited runtime. Publishing to Apple from inside it needed four problems solved that a normal backend would never hit.

ES256 signing, no crypto library

App Store Connect requires JWTs signed with ECDSA P-256. Apps Script offers only HMAC and RSA — and no BigInt. The curve arithmetic is hand-rolled on byte arrays, with RFC 6979 deterministic nonces standing in for the secure RNG the platform lacks.

Beating the 6-minute ceiling

Scraping dozens of competitors across 50 locales far exceeds one execution. The agent checkpoints progress and chains a fresh trigger to resume, so a long run finishes unattended.

Editable vs. live records

An app carries two metadata records at once — the live one and the one being prepared. Writing to the wrong one returns a confusing 409, so the uploader explicitly resolves the editable record before every write.

Pagination that actually matters

Apple caps a page at 50 results — exactly the locale count. Every localization fetch pages through to the end, so a 51st language can never be silently dropped.

Stack

Runtime	Google Apps Script (V8) — container-bound to the Sheet
UI	HTML service dialogs; one dashboard, no sheet-cell data entry
Store APIs	App Store Connect (JWT ES256), iTunes Search, Google Play endpoints
AI	Groq — metadata generation and review summarisation
Storage	Sheet tabs as tables; CacheService and Properties for tokens and state
Scheduling	Time-driven triggers — rank tracking and chart sync run overnight
Scale	~26,000 lines of Apps Script; ~10,400 stop words across 29 languages

Google Play data comes from undocumented endpoints and can change without notice; parts of Apple's are likewise unpublished. Both are monitored rather than assumed.