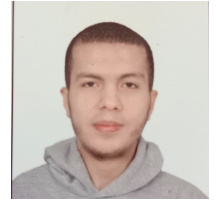


YOSF ALY MOHAMED

Web Developer · Front-End & Serverless · Arabic-First Products

Beni Suef, Egypt · El-Wasta | +20 109 060 1462
yussefali21@gmail.com | Live: noor-86c.pages.dev



PROFILE

Self-taught web developer who ships complete, production-ready products end to end: responsive interfaces, offline-first PWAs, serverless backends and Telegram bots. I work in vanilla JavaScript to keep apps fast and dependency-free, and I specialise in Arabic (RTL) products — from typography and audio pipelines to caching and deployment. Two live products are used daily: an Islamic education platform for children and a multi-branch institute management bot.

TECHNICAL SKILLS

Front-end	HTML5, CSS3 (Grid/Flexbox, RTL-first layouts), Vanilla JavaScript (ES6+), responsive & mobile-first UI, PWA, Web APIs (Service Worker, Cache, Web Audio, Notifications)
Back-end / Serverless	Cloudflare Workers, Cloudflare Pages, Cloudflare KV, REST APIs, Telegram Bot API, webhooks, cron-scheduled jobs
Data & Auth	localStorage state design & migrations, JSON data modelling, Firebase (optional auth layer), CSV/Excel export
Automation (Python)	ReportLab (Arabic PDF generation), Pillow, FFmpeg audio pipelines (loudness normalisation, concatenation), build scripts
Practices	offline-first & caching strategy, performance budgets, SEO (meta/OG, sitemap, schema), Git, versioned builds, manual QA & scripted test harnesses

SELECTED PROJECTS

Noor Al-Sighar — Islamic Learning Platform for Children

Live PWA · Cloudflare Pages · Vanilla JS

- Designed and built a complete children's platform as a single self-contained app: Qur'an reader, stories of the prophets, adhkar, prayer times, quizzes and games.
- Built an Arabic letters module (28 letters) with recorded teacher audio, letter forms and three games, plus a star-and-badge reward system.
- Engineered offline-first behaviour with a custom Service Worker (runtime caching, audio pre-caching) so the app works with no connection after first load.
- Produced 30+ audio assets through a scripted pipeline (generation → loudness normalisation → even playback level) and shipped a per-user progress report for parents.
- Deployed as an installable PWA and packaged a Trusted Web Activity Android build with Bubblewrap.
- Built a white-label demo build with partner branding and an isolated storage namespace, so an institution can preview without touching real user data.

Farqan Institute Bot — Multi-Branch Management on Telegram

Cloudflare Worker · Telegram Bot API · KV

- Wrote a ~2,600-line serverless Worker handling the full lifecycle: bot webhook, menus, callbacks and scheduled jobs — no server, no database server, no monthly hosting cost.
- Implemented student registration with identity validation, per-year segmentation and admin approval flows.
- Built homework submission in three formats (text, photo, voice note) with a manual review queue, feedback and points awarded by teachers.
- Delivered admin tooling: targeted broadcasts with preview & confirmation, manual reminders, CSV export, PDF reports and a leaderboard.
- Automated weekly digests and honours broadcasts with Cloudflare cron triggers, plus session-safe KV access and idempotent point claiming.

Arabic Document & Media Tooling

Python · ReportLab · FFmpeg

- Built RTL Arabic PDF generators (contracts, price offers, staff guides) with custom layout, correct bidirectional text and reusable design systems.
- Built audio post-processing scripts that normalise dozens of voice files to a single perceived loudness without clipping, and mux recordings onto video with frame-accurate timing.
- Built image generation for social posts (Arabic typography, brand palette, logo composition).

HOW I WORK

- Ship, then verify. Every change is checked with scripted syntax and logic tests plus a live smoke test before handover.
- No unnecessary dependencies. Vanilla JS and platform features over frameworks when they do the job.
- Owner-friendly handover. Every delivery ships with a short guide: what changed, how to deploy, how to roll back.

EDUCATION & LEARNING

Self-taught — continuous learning by shipping real products and reading official documentation (MDN, Cloudflare Workers/Pages, Telegram Bot API). No formal CS degree; the live products above replace it.