FCT Form Management Tool (`gformgen`)

An admin-facing web app for **creating, sharing, and aggregating attendance Google Forms**.

- **No DB**: Google Drive + Google Forms are the source of truth
- **Operational UX**: create → share (link/QR) → aggregate → export (CSV/PDF)
- **Serverless-friendly**: designed to run on Firebase Hosting + Functions (or any Node host)

> The in-app “Manual” (説明書) is a user guide. This README is for engineers operating/deploying the system.

Features (product)

- **Programmatic Google Form creation** (meeting title, datetime, deadline, place, host, participant count)
- **Share via link and QR**
- **Response aggregation** (attendance, remarks, non-responders)
- **Exports**
  - CSV (Excel-friendly; multi-person fields are exported with **cell-internal newlines**)
  - PDF (print-ready; multi-person fields render as **multi-line cells**)
- **Admin operations**
  - “Close” (app-level closed state)
  - “Trash” (move to Drive trash)
- **Responsive UI** for mobile + desktop
- **In-app manual** as a separate page (cards → detail view)

Screenshots (optional, but recommended for ops handoff)

Put screenshots under `docs/screenshots/` and reference them here, e.g.:

- `docs/screenshots/form-create.png` (Form creation)
- `docs/screenshots/stats-toolbar.png` (Stats toolbar: open/closed + picker + actions)
- `docs/screenshots/exports.png` (CSV/PDF export)
- `docs/screenshots/manual.png` (In-app manual)

This repo intentionally keeps the README text-first so it remains usable in terminals/CI logs.

Tech stack

- **Frontend**: React (Vite), MUI, framer-motion, lucide icons
- **Backend**: Node.js (20) + Express (deployed as Firebase Function or standalone)
- **Google APIs**: OAuth2, Forms API, Drive API
- **PDF**: `jspdf`, `jspdf-autotable`
- **QR**: `qrcode.react`

Architecture (high level)

- The frontend talks to:
  - `/api/*` for application endpoints (forms list/summary/create/etc.)
  - `/auth/*` for OAuth (login/logout/me)
- Forms are **discoverable** without a database by tagging/metadata (Drive/Forms).
- **Single-admin model** (by design): tokens are treated as a single logical admin session.

Operational semantics: “Close” vs Google Forms “Stop accepting responses”

This app’s **Close** is intentionally **app-level**, not a Google Forms setting.

- **Close (app-level)**
  - Marks the form as “closed” in *this app* (so it moves to the “Closed” list)
  - Keeps the underlying Google Form usable unless you manually disable it in Google Forms
- **Stop accepting responses (Google Forms)**
  - A Google Forms setting that prevents any further submissions
  - Not controlled by this app (by design, to keep API scope and UX simple)

If you need true “hard close”, implement it explicitly via Forms API + additional scopes and update the manual/UX accordingly.

Authentication & session model (important)

The backend obtains Google OAuth tokens on callback and then:

- **Preferred (recommended for serverless)**: persists tokens in an **encrypted HttpOnly cookie** when `GF_SESSION_PASSWORD` is set.
- **Fallback**: in-memory token slot (cold start / instance swap ⇒ may require re-login).

This repo is intentionally not a multi-tenant auth server. If you need multiple independent admins, you should redesign token storage and request scoping.

Local development

Prerequisites

- Node.js **20**

Install

```
npm install
cd backend && npm install && cd ..
```

Frontend env

The frontend chooses the API/auth base via `src/lib/apiBase.ts`.

- **Local**
  - `VITE_RUNTIME=local`
  - `VITE_LOCAL_API_BASE=http://localhost:3000`

Copy the sample:

```
cp config/env.local.example .env.local
```

Backend env (local)

The backend reads `backend/.env.local` **only in local dev**.

Create `backend/.env.local`:

```
GF_GOOGLE_CLIENT_ID=...
GF_GOOGLE_CLIENT_SECRET=...
GF_FRONTEND_ORIGIN=http://localhost:5173
GF_CORS_ORIGIN=http://localhost:5173
GF_SESSION_PASSWORD=please_set_a_long_random_string_here_32chars_min
```

> Tip: we use **GF_*** keys to avoid collisions with firebase-tools auto-loading `backend/.env`.

Run

```
# backend
cd backend && npm run dev

# frontend (in another terminal)
cd .. && npm run dev
```

Production deployment (Firebase Hosting + Functions)

This repo includes `firebase.json` for a co-located deployment:

- Hosting serves `dist/`
- Functions exposes Express as the `api` function
- Rewrites route `/api/**` and `/auth/**` to the function

1) Build

```
npm run build
```

2) Configure secrets

Use Firebase Functions Secrets (recommended):

```
firebase use <your-project-id>

firebase functions:secrets:set GF_GOOGLE_CLIENT_ID
firebase functions:secrets:set GF_GOOGLE_CLIENT_SECRET
firebase functions:secrets:set GF_FRONTEND_ORIGIN
firebase functions:secrets:set GF_OAUTH_REDIRECT_URI

# Optional: lock down CORS (default is "*")
firebase functions:secrets:set GF_CORS_ORIGIN

# Recommended: enables encrypted cookie session across serverless
instances
firebase functions:secrets:set GF_SESSION_PASSWORD
```

Example values:

- `GF_FRONTEND_ORIGIN=https://<your-hosting-domain>`
- `GF_OAUTH_REDIRECT_URI=https://<your-hosting-domain>/api/auth/google/callback`
- `GF_CORS_ORIGIN=https://<your-hosting-domain>`
- `GF_SESSION_PASSWORD=<32+ chars random>`

3) Deploy

```
firebase deploy
```

Or:

```
npm run deploy:firebase
```

OAuth redirect URI (Google Cloud Console)

Add this to **Authorized redirect URIs**:

- `https://<your-hosting-domain>/api/auth/google/callback`

Caching notes (why you might see “old UI”)

SPAs can look “stuck” if `index.html` is cached while assets change. This repo configures Firebase Hosting headers so that:

- `index.html` is **not cached** (always revalidated)
- hashed `/assets/**` are cached long-term (immutable)

See `firebase.json`.

Security notes

- Never commit secrets. `.env` and `.env.*` are ignored at repo root, and `backend/.env` is ignored.
- Prefer **Firebase Secrets** over `.env` files in production.
- OAuth tokens are stored **encrypted** in an HttpOnly cookie when `GF_SESSION_PASSWORD` is set.

Repo layout

- `src/`: frontend
- `backend/`: Express backend (Firebase Function)
- `config/`: example env files for Vite