# TalentPatriot Messages Page (Google-only)

**Objective:**
Update the Messages page to match the new mockup UI/UX and add **Google Calendar + Google Meet** support only. Do **not** add Microsoft or Zoom yet. Generate a safe, idempotent **SQL script** for Supabase that I can copy/paste into the SQL editor.

---

## 0) Guardrails (follow strictly)

- ❌ Do **not** add Microsoft 365, Teams, Outlook, Zoom, or any non-Google code.
- ❌ Do **not** remove existing app code or break current routes; only add or minimally modify.
- ❌ Do **not** print or commit secrets. Reference env names only.
- ✅ Make the SQL **idempotent** (safe to run multiple times).
- ✅ Keep changes **scoped** to Messages + Google Calendar/Meet.
- ✅ Provide a short **diff/summary** of files created/modified at the end.

---

## 1) Environment & Dependencies (describe; do not print secrets)

Document these env vars (don't print values):

- `SUPABASE_URL`, `SUPABASE_SERVICE_ROLE_KEY`
- `GOOGLE_CLIENT_ID`, `GOOGLE_CLIENT_SECRET`, `GOOGLE_REDIRECT_URI`
- `APP_JWT_SECRET` (random string)

- `TOKEN_ENCRYPTION_KEY` (32-byte base64; note how to generate)

List required deps (no versions): express, cookie-parser, @supabase/supabase-js, zod, jsonwebtoken.

---

## 2) Supabase SQL (generate script text only, no execution)

Output a single **SQL script** the user can copy/paste that:

- Ensures `pgcrypto` is enabled.
- Creates (if not exists) `public.connected_accounts` with fields: `id, user_id (fk auth.users), provider=('google'), scopes text[], access_token_enc, refresh_token_enc, expires_at, created_at, updated_at` and unique `(user_id, provider)` + index.
- Creates (if not exists) lightweight messaging tables if absent: `public.message_threads`, `public.messages`.
- Creates (if not exists) `public.calendar_events` to store Google event references (`provider=('google')`, `provider_event_id`, `summary`, `start_at`, `end_at`, `conference_url`, `attendees jsonb`, `status`, `created_by`, timestamps).
- Creates/updates a `public.connected_accounts_public` **view** that exposes only non-sensitive cols.
- Adds/updates a generic `set_updated_at()` trigger and applies to mutable tables.
- Notes (as comments) where to enable RLS later; do not enable RLS in this script.

**Deliverable:** Print the SQL block clearly labeled “Copy into Supabase SQL editor”.

---

## 3) Backend structure (create files; keep code minimal & modular)

Create/modify files (no code printed here—just implement):

- `/server/lib/supabase.ts`
  - Supabase **admin** client (service role).
- `/server/lib/crypto.ts`
  - AES-256-GCM helpers for encrypt/decrypt token strings (uses `TOKEN_ENCRYPTION_KEY`).
- `/server/lib/google.ts`
  - Google OAuth helpers: auth URL builder, state signing/verification, token exchange, token refresh, upsert tokens (encrypted) into `connected_accounts`, `getValidAccessToken(userId)` with refresh.
  - Scopes: `calendar.events`, `calendar.readonly`, plus `openid email profile`.
  - Use `access_type=offline` and `prompt=consent` to ensure refresh tokens.
- `/server/routes/auth-google.ts`
  - `GET /auth/google/login` → redirects to Google consent. Requires a current user id (accept temp `?user_id=` or `X-User-Id` header with TODO note to wire real auth).
  - `GET /auth/google/callback` → exchanges code, stores encrypted tokens, redirects to `/settings/integrations?google=connected` (or a provided `redirect_to`).
- `/server/routes/google-calendar.ts`
  - `POST /api/google/meet` → creates a Calendar event on the recruiter's **primary** calendar with `conferenceData` (Google Meet) and returns `{ meetUrl, eventId }`. Accepts JSON body: `summary`, `start`, `end`, `attendees: string[]` (optional). Persists a minimal record in `calendar_events`.
  - `GET /api/google/freebusy` → proxy to Calendar FreeBusy for the primary calendar. Accepts `start`, `end` (ISO) and returns raw free/busy; keep response

small.

- `/server/index.ts`
    - Mount the two route modules and standard middleware.

**Security notes to implement:**

- Never log access/refresh tokens or PII.
- Encrypt tokens at rest using `crypto.ts`.
- Add 60s headroom to `expires_at` refresh logic.
- Keep CORS/settings consistent with the app.

---

## 4) Frontend: Messages page alignment (minimal change; no library bloat)

Update the Messages page to match the mockup:

- **Tabs:** Internal · Email · Client Portal.
- **Email mode:** fields `From` (read-only, connected account email if available), `To`, `Subject`, `Body`.
- **Chips/Controls:**
    - **Propose Times** → opens right-side drawer; call `/api/google/freebusy` and (for now) just log results; show a placeholder grid overlay if trivial, otherwise text note that availability fetched successfully.
    - **Add Calendar Invite** → keep as a non-functional placeholder for now (no backend yet).
    - **Create Video** button with dropdown:

        - **Google Meet** → call `POST /api/google/meet`, then append returned `meetUrl` to the body.
        - Keep **Zoom** and **Microsoft Teams** as UI items that insert **placeholder** links only (no backend).
- **Thread timeline:** show example items with badges `Email`, `Invite`, `Meet`. No backend changes required; hardcode sample.
- **Filters row:** `Has open invite · Needs reply · Upcoming meetings` — UI only.
- Ensure component uses existing design tokens / shadcn components and does not break current styles.

**Do not** introduce heavy scheduling libraries. Keep the drawer and overlay simple.

---

## 5) Testing & Validation (add to output as a checklist)

- **OAuth flow:**
    - Hit `/auth/google/login?user_id=<testUserId>` → consent → verify row in `connected_accounts`.
- **Meet creation:**
    - `POST /api/google/meet` with test body → returns `meetUrl` and persists `calendar_events`.
    - Confirm event appears in the connected Google Calendar.
- **FreeBusy:**
    - `GET /api/google/freebusy` with 7-day window → returns busy blocks.
- **UI:**
    - In Email mode, selecting **Google Meet** inserts the **real** join link into the body.

    - Propose Times drawer opens and fetches availability without errors.

---

## 6) Deliverables (print at end)

- List of files created/updated with a one-line purpose each.
- The **SQL script** (clearly separated and titled) for Supabase.
- Short setup notes: envs required and where to configure Google API scopes.
- Manual test instructions (from §5).

---

## 7) Future hooks (notes only; no code now)

- Gmail send + threading.
- Calendar event creation for selected slots.
- Microsoft/Teams and Zoom providers mirroring these endpoints.
- RLS policies for `connected_accounts`, `calendar_events`, `messages`, `message_threads`.