AUTOMATIC EMAIL REPLY GENERATOR

High-level architecture

- Frontend: React + TypeScript SPA built with Vite. Styling via Tailwind CSS and shadcn-ui (Radix + cva). Theming with next-themes. Data fetching/caching with @tanstack/react-query. Charts via Recharts. Icons via lucide-react.
- Backend: Node.js Express server (ESM) exposing REST endpoints. Provider-adapter pattern for email ingestion:
    - Gmail adapter using googleapis + OAuth2 (readonly scope).
    - IMAP adapter using imapflow + mailparser.
    - Outlook is stubbed (hint for MSAL + Graph).
- Data ingestion approaches:
    - Demo CSV (PapaParse) for offline/local preview.
    - Live email via Gmail or generic IMAP.
- Email enrichment: Deterministic heuristics for priority, sentiment, entity extraction on both client (csvService.ts) and server (enrich.mjs).
- AI drafting: OpenRouter chat completions from client (llmClient.ts) with safe heuristic fallback (no key). Optional shortener flow. Notes warn against exposing production API keys; recommends server proxy in prod.
- Design system: CSS variables (HSL) define tokens for themes, priority/sentiment palettes; Tailwind consumes via config and utilities. shadcn UI primitives standardized via cva and Tailwind-merge.

Runtime flow

- App bootstraps with QueryClientProvider, ThemeProvider, Router.
- Index page:
    - Select source: csv | gmail | outlook | imap.
    - Checks Gmail auth; if needed, opens Google consent, then refresh.
    - Loads emails (via CSV or backend). Backend returns enriched emails; client normalizes dates.
    - Filters list by support-related keywords, displays table, detail panel with AI draft, and analytics.
    - Draft regeneration uses OpenRouter (multi-model fallback) or heuristic; “shorten” action uses rewriteDraftShort. “Send” simulates marking the email as sent.

APIs (server/index.mjs)

- GET /api/health: basic health.

- GET /api/auth/google: returns OAuth consent URL (requires GOOGLE_CLIENT_ID/SECRET).
- GET /api/auth/google/callback?code=…: handles token exchange (in-memory token storage for dev).
- GET /api/emails?provider=gmail|imap|outlook&max=&since=:
    - gmail: lists INBOX messages via Gmail API, extracts plain text (prefers text/plain, falls back to html-to-text), enriches, sorts.
    - imap: connects via ImapFlow, fetches and parses messages, enriches, sorts.
    - outlook: stub hint; returns empty with instructional message.

Tech stack inventory

- Core: React 18, TypeScript, Vite 5 (plugin-react-swc), SWC, path alias @ → src
- UI/UX: Tailwind CSS 3, tailwindcss-animate, shadcn-ui patterns, Radix UI primitives, class-variance-authority (cva), tailwind-merge, clsx, sonner, lucide-react, cmdk, vaul, react-day-picker, react-resizable-panels
- Theming: next-themes
- Data/State: @tanstack/react-query
- Charts: Recharts (with a small wrapper)
- Forms: react-hook-form, @hookform/resolvers, zod (available; used by UI primitives like form.tsx)
- CSV: papaparse (+ @types)
- Dates: date-fns
- Backend: express, cors, dotenv
- Email providers: googleapis (Gmail), imapflow (IMAP), mailparser, html-to-text
- Lint/Build: eslint 9, typescript-eslint 8, @eslint/js, eslint-plugin-react-hooks, eslint-plugin-react-refresh, tailwindcss, autoprefixer, postcss, typescript 5, vite
- Dev tooling: lovable-tagger (componentTagger plugin)
- Lockfile: bun.lockb present (Bun can be used, but npm scripts are provided)

Structure and per-file purpose

Root

- bun.lockb: Bun lockfile (optional; repo also supports npm).
- components.json: shadcn-ui config for component scaffolding.
- index.html: Vite app entry.
- package.json: scripts (dev, build, preview, server), deps (frontend + backend).

- postcss.config.js: tailwindcss + autoprefixer.
- eslint.config.js: TS + React lint config (hooks, refresh).
- tailwind.config.ts: Tailwind theme tokens, colors (priority/sentiment/sidebar), keyframes, darkMode, animate plugin.
- tsconfig.json / tsconfig.app.json / tsconfig.node.json: TS configurations for app/node builds.
- vite.config.ts: Vite + React SWC plugin, componentTagger in dev, alias @ -> src.
- README.md: project overview, techs, OpenRouter docs, dev and deploy guidance.

public/

- Demo.csv: offline demo data for CSV ingestion.
- favicon.ico, placeholder.svg, robots.txt: static assets.
- index.mjs: Express app; CORS setup; Google OAuth routes; /api/emails handler dispatching to providers; health route.
- providers/gmail.mjs: Gmail adapter.
    - OAuth2 flow via googleapis; in-memory tokens.
    - lists INBOX with optional “after” filter; fetches full message; extracts text via payload traversal; enriches + sorts.
- providers/imap.mjs: IMAP adapter.
    - Env-configured connection; mailbox open; search SINCE or last N; fetches message source; mailparser to parse; html-to-text fallback; enriches + sorts.
- utils/enrich.mjs: Heuristics shared on server (priority, sentiment, score, support filter, entity extraction: phone, email, order id, common issues).

src/ (frontend)

- main.tsx: Mounts React app to #root.
- App.tsx: Providers (QueryClientProvider, ThemeProvider, TooltipProvider), Toasters, Router with routes.
- index.css: Tailwind layers and extensive CSS variable design system:
    - HSL tokens for light/dark, priority (urgent/medium/low), sentiment (positive/neutral/negative), sidebar palette, gradients, shadows; utility classes (glass, card-modern, hover-lift, text-gradient), scrollbar styling.
- App.css: Default Vite scaffold styles (mostly unused).
- vite-env.d.ts: Vite types.

src/pages/

- Index.tsx: Main UI:
    - State: emails, filtered slice, selection, source “csv|gmail|outlook|imap”, loading/refresh flags, Gmail auth needs/URL.

    - Effects: check Gmail auth when source=gmail; load emails from chosen provider (CSV or server).
    - Handlers: filtering, refresh; AI regenerate (OpenRouter + heuristic fallback), shorten; send (marks sent).
-
    - Layout: Tabs (Inbox/Analytics/Settings); header with theme toggle and refresh; tables + details + analytics; settings for source + theme.
- NotFound.tsx: 404 route.

src/lib/

- types.ts: Core data models (Email, EmailEntity, EmailResponse, AnalyticsData; Priority, Sentiment, EmailStatus).
- csvService.ts: Loads Demo.csv via Papa; heuristic enrichment (priority, sentiment, score, filter, entities); generates initial AI response text for filtered emails; sorting and dedupe.
- emailService.ts: Client data sourcing:
    - checkGmailAuth(), getGmailAuthUrl(), fetchEmailsFromServer(provider), fetchEmails(provider) with CSV shortcut, normalized Date parsing.
- llmClient.ts: OpenRouter client + heuristic fallback.
    - buildSystemPrompt/buildUserPrompt.
    - generateSupportReply() with model fallback list and temperature/max_tokens tuning by verbosity/urgency/sentiment; returns modelUsed/errorMessage when applicable.
    - rewriteDraftShort() endpoint (OpenRouter or heuristic sentence compression).
- aiResponseService.ts: (empty placeholder).
- mockData.ts: Mock emails and analytics (not primary path; CSV/server are used).
- utils.ts: cn() wrapper (clsx + tailwind-merge).

src/hooks/

- use-mobile.tsx: responsive hook (used by sidebar).
- use-toast.ts: toast hook used across UI.

src/components/

- Analytics.tsx: KPI cards + Recharts pies and line chart with theme-aware styling.
- EmailDetail.tsx: Detail panel with metadata, entity chips, AI draft textarea, “More detail” switch, regenerate/send actions, model/error badges.
- EmailFilter.tsx: Keyword-based filter UI with select/clear all; emits filtered arrays.
- EmailTable.tsx: Searchable, prioritized/sorted table; shows badges for priority, sentiment, status.
- PriorityBadge.tsx: Badge variants for urgent/medium/low using icons.

- SentimentBadge.tsx: Badge variants for positive/neutral/negative.
- ThemeToggle.tsx, ThemeIndicator.tsx: Theme controls.

src/components/ui/ (shadcn + Radix-based primitives and wrappers)

- accordion.tsx, alert-dialog.tsx, alert.tsx, aspect-ratio.tsx, avatar.tsx, badge.tsx, breadcrumb.tsx, button.tsx, calendar.tsx, card.tsx, carousel.tsx (Embla carousel integration with keyboard and API context), chart.tsx (Recharts wrapper with theme-aware CSS variables), checkbox.tsx, collapsible.tsx, command.tsx (cmdk), context-menu.tsx, dialog.tsx, drawer.tsx (vaul), dropdown-menu.tsx, form.tsx (react-hook-form + zod ready), hover-card.tsx, input-otp.tsx, input.tsx, label.tsx, menubar.tsx, navigation-menu.tsx, pagination.tsx, popover.tsx, progress.tsx, radio-group.tsx, resizable.tsx (panels), scroll-area.tsx, select.tsx, separator.tsx, sheet.tsx, sidebar.tsx (stateful sidebar with cookie + mobile behavior; cva variants; lucide icon; integrated Tooltip), skeleton.tsx, slider.tsx, sonner.tsx (toast), switch.tsx, table.tsx, tabs.tsx, textarea.tsx, toast.tsx, toaster.tsx, toggle-group.tsx, toggle.tsx, tooltip.tsx, use-toast.ts.
    - Common traits: cva for variants, cn for class merge, Radix primitives for a11y, Tailwind tokens, and in some cases composable context providers.

public UX details

- Priority/sentiment/sidebars colors wired through Tailwind config to CSS variables in index.css.
- “glass”, “card-modern”, “hover-glow/lift” classes used broadly for a modern UI aesthetic.

Notable patterns and approaches

- Provider/Adapter pattern on backend to add more inbox sources cleanly.
- Deterministic enrichment heuristics mirrored on both client and server for consistency.
- Heuristic AI fallback to keep the app fully functional without OpenRouter keys.
- Theme tokens in HSL + Tailwind extension ensure consistent dark/light and semantic palettes for priority/sentiment.
- UI composition leans on shadcn patterns for predictable accessibility and styling ergonomics.

Completion summary:

- Mapped the full stack, data flow, and provider adapters.
- Cataloged technologies from package.json and usage in code.
- Provided a per-file purpose map, including UI primitives and backend module.