## Job Mailer — Technical Documentation

### 1) What this project is

Job Mailer is a Node.js automation tool + local web UI that helps send job application emails reliably and repeatedly without re-sending to the same recipient.

It supports **three sending modes**: - **UI single send**: fill a form, attach a resume (optional), and send. - **UI bulk send**: upload Excel (.xlsx) or paste a list of emails and send to many. - **Automation mode**: read `data/recipients.csv`, send to recipients not yet sent, and log results. This can run on a schedule and also auto-send when new rows are added.

It also includes: - **HR Finder**: discover HR/Talent emails for a company domain (Hunter / Apollo) with auto domain detection. - **Defaults management**: save SMTP settings, default subject/body, and a default resume via the UI. - **ATS score + optimization helper**: score resume text vs JD and produce an "optimized" downloadable PDF.

---

### 2) Tech stack

- **Runtime**: Node.js
- **Backend**: Express + Multer (uploads) + Nodemailer (SMTP) + XLSX (Excel I/O)
- **Automation**: node-cron (schedule), chokidar (file watcher), csv-parse (CSV parsing)
- **Frontend**: static HTML/CSS/vanilla JS served by Express

Key dependency confirmation: **Nodemailer is used** via `backend/src/mailer.js`.

---

### 3) Repository structure (important folders/files)

- **Backend (core)**: `backend/src/*`
  - `index.js`: starts watcher + cron schedule
  - `send.js`: sends pending recipients from CSV using `sent.json` as idempotency store
  - `watcher.js`: watches `data/recipients.csv` and sends newly added emails
  - `scheduler.js`: cron scheduler
  - `mailer.js`: SMTP transport creation + email send
  - `template.js`: default email template builder
  - `ui-server.js`: Express server for UI + API (send, bulk, HR, settings, ATS)
  - `recipients.js`: robust CSV parsing + validation
  - `sent-log.js`: idempotency and status tracking in JSON

  - `excel-log.js`: append/update `sent.xlsx` log
  - `utils.js`: atomic file writes + helpers
- **Frontend**: `frontend/public/*`
  - `index.html`: main UI (tabs: Send, HR Finder, Defaults)
  - `app.js`: UI behaviors + API calls
  - `styles.css`: UI styling
  - `login.html`: login page (if auth enabled)
- **Data directory**: `data/*` (project state)
  - `recipients.csv` (expected): list of recipients for automation mode
  - `sent.json` (created): idempotency + last status by email
  - `sent.xlsx` (created/updated): downloadable Excel log of sending
  - `ui-settings.json`: saved UI defaults (smtp/subject/body/etc)
  - `default-resume.pdf`: default resume uploaded via UI
  - `companies.json`: saved company names for the HR Finder dropdown

---

### 4) Configuration (.env)

Configuration is loaded from: - `.env` (preferred), else - `env.example` (fallback)

Main environment variables (names only; don't hard-code secrets in git):

#### SMTP / sender

- `SMTP_HOST`, `SMTP_PORT`, `SMTP_SECURE`
- `SMTP_USER`, `SMTP_PASS`
- `FROM_EMAIL`, `FROM_NAME`
- `RESUME_PATH`

#### Automation

- `SCHEDULE_CRON`, `TIMEZONE`
- `DELAY_MS_BETWEEN_EMAILS`
- `DRY_RUN`

#### UI

- `UI_AUTH_USER`, `UI_AUTH_PASS` (optional; enables login gate)
- `SENT_XLSX_PATH` (where to write the Excel log)
- `SUBJECT` (default subject)

#### HR Finder

- `HUNTER_API_KEY`
- `APOLLO_API_KEY` (+ optional `APOLLO_BASE_URL`, `APOLLO_ENDPOINT`)
- `APOLLO_REVEAL_PHONE_NUMBER`

---

### 5) Runtime modes (how to run)

Defined in `package.json` scripts: - `npm run start` → runs `backend/src/index.js` (watcher + cron) - `npm run send:now` → runs `backend/src/send-pending.js` (one-shot send) - `npm run watch` → runs `backend/src/watch-and-send.js` (watcher only) - `npm run ui` → runs `backend/src/ui-server.js` (web UI + API)

---

### 6) Email sending architecture (core logic)

**6.1 Transporter creation + SMTP reliability** `backend/src/mailer.js`: - Validates required SMTP config. - Creates a Nodemailer transport and calls `verify()`. - If the config uses **587 + STARTTLS** and the error looks like **network reachability**, it retries once using a **fallback 465 + SSL** configuration.

This is a real-world reliability improvement because some networks block outgoing SMTP submission ports.

**6.2 Email build + signature rules** Two ways content is produced:

1) **Default template** (`backend/src/template.js`):

- Reads default body from `data/ui-settings.json` (if present).
- Adds a signature (if the body does not already contain one).
- Builds both `text` and `html`.

2) **UI "override body" path** (`backend/src/ui-server.js`):

- If a custom body is provided, it builds a simple HTML wrapper and adds signature only if missing.

**6.3 Attachments**

- Attach resume from either:
  - uploaded file (UI), or
  - default resume path (UI settings), or
  - `RESUME_PATH` from env
- Error if resume path doesn't exist (CLI mode)

---

### 7) Idempotency + logging (avoid re-sending)

**7.1 JSON log for idempotency** `backend/src/sent-log.js` maintains `data/sent.json` keyed by email: - `status: "sent" | "error"` - timestamps

(`sentAt` / `errorAt`) - `messageId`, `response`, `source` (cron/watch/manual)

The "pending" list is computed by: `recipients.csv` minus emails already marked as sent.

**7.2 Excel log** `backend/src/excel-log.js` appends rows to `data/sent.xlsx` with columns: - email, name, subject, error

It also normalizes old headers (backward compatible).

---

### 8) Automation mode (cron + watcher)

**8.1 Cron schedule** `backend/src/scheduler.js`: - Validates `SCHEDULE_CRON`. - Schedules ticks in `TIMEZONE`. - On tick, calls `sendPending({source: "cron"})`.

**8.2 Watcher** `backend/src/watcher.js`: - Uses chokidar to watch `data/recipients.csv`. - Debounces changes (400ms). - Sends **only newly added emails** that are not already sent.

---

### 9) Web UI server (Express)

`backend/src/ui-server.js` is a separate runtime mode for interactive usage.

#### 9.1 Auth model

- If `UI_AUTH_USER` and `UI_AUTH_PASS` are set, the UI becomes protected.
- Simple cookie session (`jm_sid`) stored in-memory with 12h TTL.

Trade-off: in-memory sessions reset on server restart (acceptable for local tool).

#### 9.2 Key API endpoints (high level)

- **Auth**: `/api/login`, `/api/logout`
- **Settings**: `/api/settings`, `/api/settings/resume`
- **Sending**: `/api/send`, `/api/send-bulk`, `/api/send-list`
- **Downloads**: `/api/template.xlsx`, `/api/sent.xlsx`
- **HR Finder**: `/api/provider-status`, `/api/hr-lookup`
- **Company dropdown**: `/api/company-names`, `/api/company-suggest`
- **ATS**: `/api/ats-score`, `/api/ats-optimize`, `/api/ats-optimized.pdf`

---

### 10) HR Finder (Hunter / Apollo)

`/api/hr-lookup`: - Input: company name and/or domain, provider. - If domain not provided, the server tries to resolve a domain using Clearbit autocomplete. - Then: - **Hunter**: uses Hunter Domain Search and filters recruiting roles; falls back to all emails if none match. - **Apollo**: uses Apollo mixed people search, maps contact fields, optionally reveals phone.

Additionally: - Company names are saved to `data/companies.json` and shown in a UI dropdown. - Domain resolution retries simplified queries (e.g., stripping Pvt/Ltd) so dropdown selections behave like typing.

---

### 11) ATS scoring + optimized PDF generation

In `backend/src/ui-server.js`: - Accepts JD + resume (pdf/docx upload or pasted text). - Extracts text (uses system `pdftotext` or `unzip` for docx). - Computes a heuristic ATS score: - keyword match portion - structure score (sections, bullets, numbers) - "Optimize" iteratively appends suggestions and can generate a downloadable PDF.

Important operational note: - PDF merging uses `pdfunite` if available.

---

### 12) Security & privacy notes (what to say clearly)

- **Never commit real SMTP passwords / API keys**. Use `.env` and keep it out of git.
- The UI auth is **local / single-user** design. It's not intended as a hardened multi-tenant system.
- Email content + recipients are sensitive. Logging is local (`data/*`).

---

### 13) Extension ideas (future improvements)

- Replace SMTP with email API provider (Resend/SendGrid/Mailgun) for better deliverability and fewer network blocks.
- Persist sessions to disk (or disable auth in fully local mode).
- Add retries/backoff per recipient + per-provider rate limits.
- Add structured logging and a UI "history" view from `sent.xlsx`.
- Add tests for CSV/Excel parsing and HR mapping.