PROCESS AUDIT

SNAPSHOT

Job-complete → invoice automation

> *This is a sample audit. Everything below is shown illustratively, including the data model, the screen wireframes, the integration detail, and the Phase-1 job and cost breakdown.*

PREPARED FOR

[Client name], NorthFlow HVAC

PREPARED BY

[Business Analyst], Fastw3b

DATE

[Date]

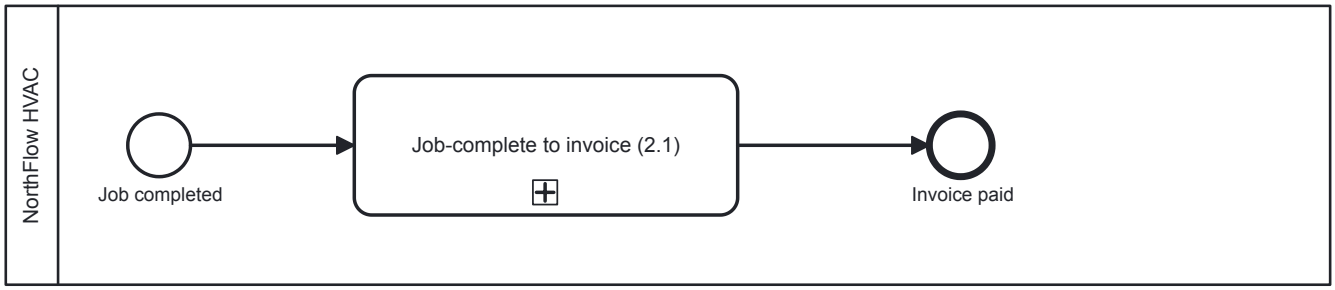
VERSION

v4.0

Contents

0. Solution landscape

The whole picture first. This automation covers a single process: a completed job becomes an invoice that goes out the same day and is marked paid when the money arrives. The detail is in section 2; what the current way costs is in section 3.



Roles and dependencies.

ROLE / SYSTEM	JOB-COMPLETE → INVOICE (2.1)	DEPENDS ON
Field Technician	records line-items and marks the job complete on site	the mobile job screen
Office Admin	reviews the draft, sends the invoice, records the payment	the invoice queue, the customer's billing email
System	auto-drafts the invoice and emails it (and matches the payment in Phase 2)	the price book, the accounting system, the email service

1. Executive summary

NorthFlow HVAC is a 25-person residential HVAC service company. Today, when a field technician finishes a job, the office re-types an invoice from the technician's paper notes each evening. Billing runs about 2 days behind, line-items get missed, and cash arrives late.

What this system does. It makes the **job-complete → invoice** step automatic. The moment a technician marks a job done in the mobile app, the system **drafts the invoice from the job's line-items**, **alerts the office to review and send**, and **tracks it through to paid**. The office stops re-typing invoices by hand.

The value it delivers. Invoices go out the same day instead of two days later, so cash arrives sooner. Nothing captured on site is dropped, so revenue stops leaking. And the nightly data-entry job disappears, freeing the office for work that needs a person.

Recommended approach. Build the **job-complete → invoice** slice first (Phase 1). The §3 opportunity ranking puts its three spots (capture at source, auto-draft, same-day send) at the top: together they remove the nightly re-typing and the billing lag with the least to build, and pay for themselves fastest. Payment-match and the Billing Board follow once that is live.

Business logic you're approving, at a glance.

Roles (who does what):

1. The **field technician** records the work and its line-items on site.
2. The **office** reviews each draft, sends it, and records the payment. (The manager is a permission level of the office role, with an all-jobs view plus export, not a separate workflow role. That keeps this to the 2 process roles of the Snapshot tier.)

The flow (start to finish):

1. A technician marks a job complete.
2. The system auto-drafts the invoice from the job's line-items.
3. The office reviews the draft and sends it to the customer.
4. A matched payment moves the invoice to paid.

Key rules (the guardrails):

1. A job cannot be completed with no line-items (BR-1).
2. There is exactly one invoice per job (BR-2).
3. Only the office can send or void an invoice (BR-4).
4. Sending requires a billing email on file (BR-5).
5. Unmatched payments are flagged for a person, never auto-applied (BR-7).

States (how records move):

1. A **Job** runs *scheduled → in-progress → completed → invoiced*.
2. An **Invoice** runs *draft → sent → paid*, with *void* reachable from draft or sent.

Edge handling (when things go sideways):

1. An offline completion queues on the device and syncs when back online.
2. A missing billing email blocks sending, but not drafting.
3. A re-opened job voids its invoice so nothing is double-billed.

Scope: Phase 1 delivers everything above. Payment-match, the Billing Board, and reporting are later phases.

2. Processes

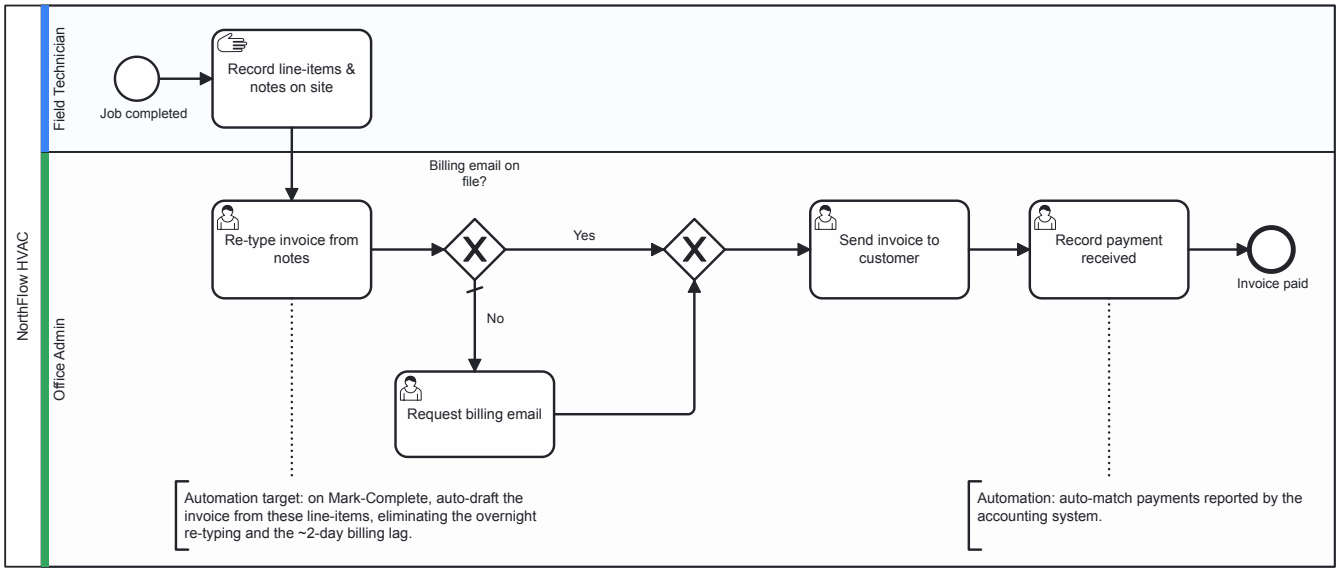
This operation is a single process; its current-state flow is below.

2.1 Job-complete → invoice

Roles (lanes): Field Technician (does the work, records line-items) and **Office Admin** (re-types the invoice, sends it, records payment). *(The automated target adds a **System** lane; see §4.)*

Stages (current state): (0) the technician completes a job and records line-items and notes on site → (1) that evening the office **re-types** an invoice from those notes → (2) the office sends it to the customer (chasing billing details first if they're missing) → (3) the office records the payment when it arrives. The **automation opportunities** are annotated on the diagram.

The process model below is a **real BPMN 2.0** diagram (pool with role lanes, typed start and end events, an exclusive gateway with a default flow, verb-noun tasks; the happy path runs straight across and the exception drops down). It renders interactively on the Process Audit page.



3. Current-state diagnosis + cost-of-status-quo

- **Manual re-entry.** Every invoice is typed from scratch in the evening from the technician's notes. That double-handles data already captured in the field.
- **Billing lag (about 2 days).** Jobs done today are invoiced a day or two later, and cash arrives later still, because the invoice waits for the evening re-typing before it can go out.

- **Missed line-items.** Parts and labour noted on site don't always make it onto the invoice, which leaks revenue.
- **No visibility.** There is no single view of "which completed jobs aren't invoiced yet" or "which invoices are unpaid"; it lives in one person's head and a spreadsheet.

Cost of the status quo. The figure below is built up from NorthFlow's own volume, so you can see where it comes from rather than take it on faith:

1. The nightly re-typing runs about 6 hours a week. At a loaded office rate of roughly $35/hr, that is about **$900/mo** spent re-keying data the technician already entered.
2. The roughly 2-day billing lag defers cash on around **$120k/mo** of invoicing, money earned but not yet collected, purely because the invoice went out late.
3. Missed line-items leak an estimated **1 to 2 percent of revenue**: work done on site but never billed.

Together that is on the order of **$1,400 to $2,000/mo** in lost time, deferred cash, and revenue leakage (illustrative, built on the volume assumptions in §9.2). This is the value the Phase-1 quote in §6 is measured against.

Where automation pays off most: the opportunity ranking. Before choosing what to build, here is where automating each step in the section 2.1 flow returns the most for the least. Effect is the value removed (a grounded volume from the cost-of-status-quo above, confirmed in §9.2); effort is the build size from the reference classes; leverage is the effect you get for that effort.

RANK	AUTOMATION SPOT (IN THE §2.1 FLOW)	EFFECT (VALUE REMOVED)	LEVERAGE, EFFORT & VERDICT
1	Auto-draft the invoice on Mark-complete (replaces the office "re-type invoice" task)	Removes about 6 hours a week of re-typing (about $900/mo) and the roughly 2-day billing lag on about $120k/mo of invoicing	Leverage: High Effort: Medium Phase 1
2	Capture line-items at source on the technician's mobile screen (the "record line-items" task)	Stops the 1 to 2 percent revenue leak from missed line-items, and ends the double data-entry	Leverage: High Effort: Medium Phase 1
3	Review queue, one-click Send and customer email (the office "send" task)	Invoices go out the same day, and the badge count makes a forgotten job impossible to miss	Leverage: High Effort: Medium Phase 1
4	Match payments from Sent to Paid (the "record payment" task)	Removes the manual reconciliation, but only once accounting exposes payment status (§7)	Leverage: Medium Effort: Low to medium Later (Phase 2)
5	Stale-draft and unpaid-invoice flags on the Billing Board	Surfaces what is unsent or unpaid, so nothing is chased from memory	Leverage: Medium Effort: Medium Later (Phase 2)
6	Reporting, digests and price-book self-service	Management insight and less day-to-day upkeep, but a refinement, not a pain remover	Leverage: Low Effort: Medium Later (refinement)

Reading. The top three spots are one connected slice (capture at source, auto-draft, same-day send), and that is exactly where the pain and the money are. They are the cheapest things to build that remove the most waste, which is why Phase 1 in §6 is that slice and nothing more. Spots 4 and 5 earn their keep only after invoices reliably go out, and spot 6 is refinement, so all three wait for a later phase. No spot here is padding: each traces to a step in the §2.1 flow, and a "later" verdict is a real decision about sequence, not a way to make the list look fuller.

3.5 Role views

This is a single linear flow with two roles, so each role's part is already visible in its own lane in the section 2 diagram; a separate per-role view would only restate that lane, so none is added here.

4. Target-state system specification

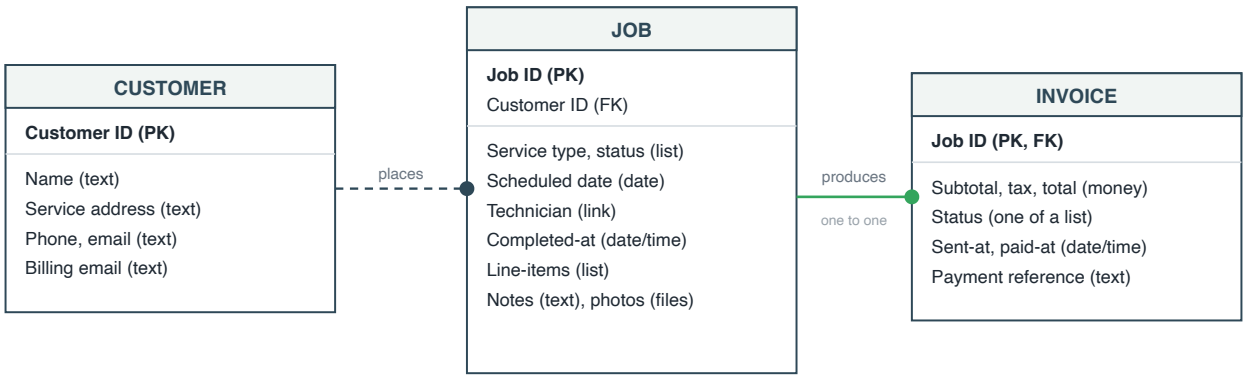
> ***At a glance:*** *3 kinds of information tracked · 5 screens · 6 automations · 9 rules · 1 integration.*

4.1 Information the system tracks: data model

Each field shows the kind of information it holds, in plain terms, so you can confirm it reads right.

- **Job.** The work performed: customer (link), service address (text), job type (one of a list), assigned technician (link), scheduled date (date), **line-items** (a list of parts and labour, each with quantity and price), on-site notes (text) and photos (files), completed-at (date and time), status (one of a list). *(One Job belongs to one Customer and produces at most one Invoice.)*
- **Customer.** Name (text), service address (text), contact phone and email (text), billing email (text).
- **Invoice.** The bill for a job: the copied line-items (a list), subtotal, tax and total (money), status (one of a list), sent-at and paid-at (date and time), payment reference (text). *(One Invoice belongs to one Job.)*

The three records and how they relate, as an **IDEF1X** data model. A solid line with a filled dot marks an **identifying** relationship (the one invoice per job that rule BR-2 guarantees, so an invoice cannot exist without its job); a dashed line marks a **non-identifying** reference (each job names the customer it was done for).



IDEF1X data model: each entity with its key fields, and the identifying (solid line, filled dot) versus non-identifying (dashed line) relationships between them.

4.2 Screens / views

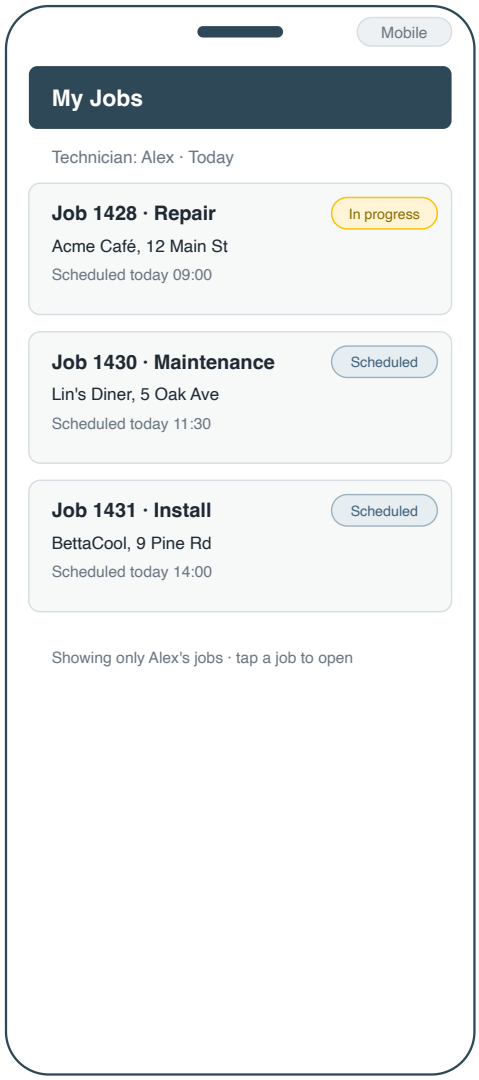
The system is five screens. The four that carry the Phase-1 flow are described and wireframed below, each with why the role needs it, what it shows, and how the person works with it. Every wireframe is drawn at a **fixed frame for its device**, the technician's on-site screens on a **mobile** frame (360×780), the office's screens on a **desktop** frame (1200×760), so screens of the same device read at one consistent size, and each shows the **actual fields from the §4.1 data model** (not a re-imagined layout). The fifth screen is listed under "Also in the system". Wireframes show layout and fields, not the final visual design.

Technician: My Jobs (mobile)

Why the technician needs it. It is the technician's way in: the list of the jobs assigned to them today and the door to each Job & Complete screen. Without it they have no on-phone starting point.

What they see. Their own jobs only, each showing the **customer**, **service address**, **job type**, **scheduled date** and **status** straight from the Job record.

How they work with it. They glance at the day's list and tap a job to open it; the status tells them what is still in progress versus done.



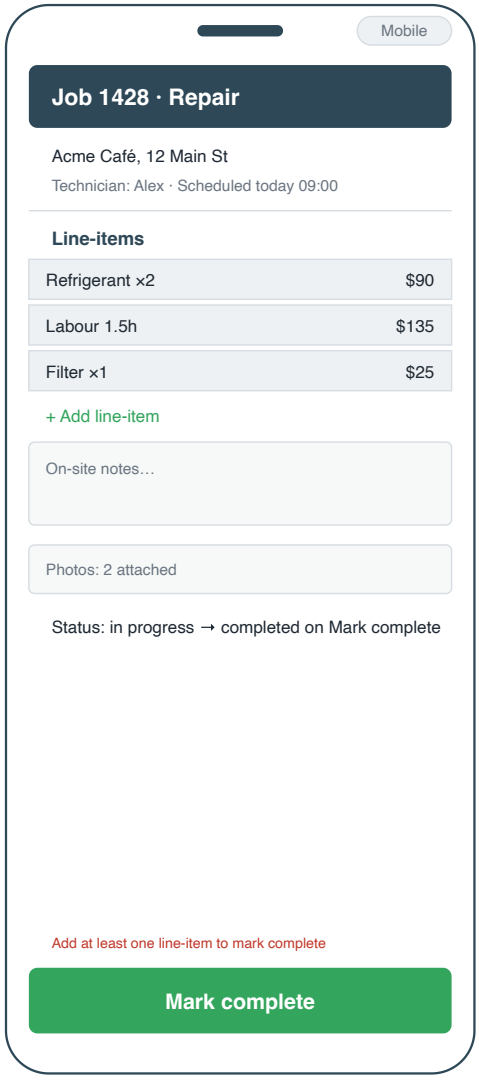
My Jobs (mobile, 360×780): the technician's own jobs for today, customer, address, type, scheduled time and status from the Job record. Tapping a job opens Job & Complete.

Technician: Job & Complete (mobile)

Why the technician needs it. This is where the job is recorded, and it is the trigger for the whole automation. Without it the technician keeps writing on paper and the office keeps re-typing in the evening. It is deliberately a phone screen, because the technician is on site, not at a desk.

What they see. The open Job's **customer**, **service address**, **assigned technician** and **scheduled date** at the top; the running list of **line-items** picked from the price book with quantities and price; an "add line-item" action; **on-site notes** and **photos**; and the job **status**.

How they work with it. They add each part and labour line from the price book, jot any notes, attach a photo or two, then tap **Mark complete**. That single tap sets the job's **completed-at**, moves its status to completed, and fires the auto-draft. If no line-item has been added, **Mark complete** is blocked (BR-1), so an empty job can never be billed.



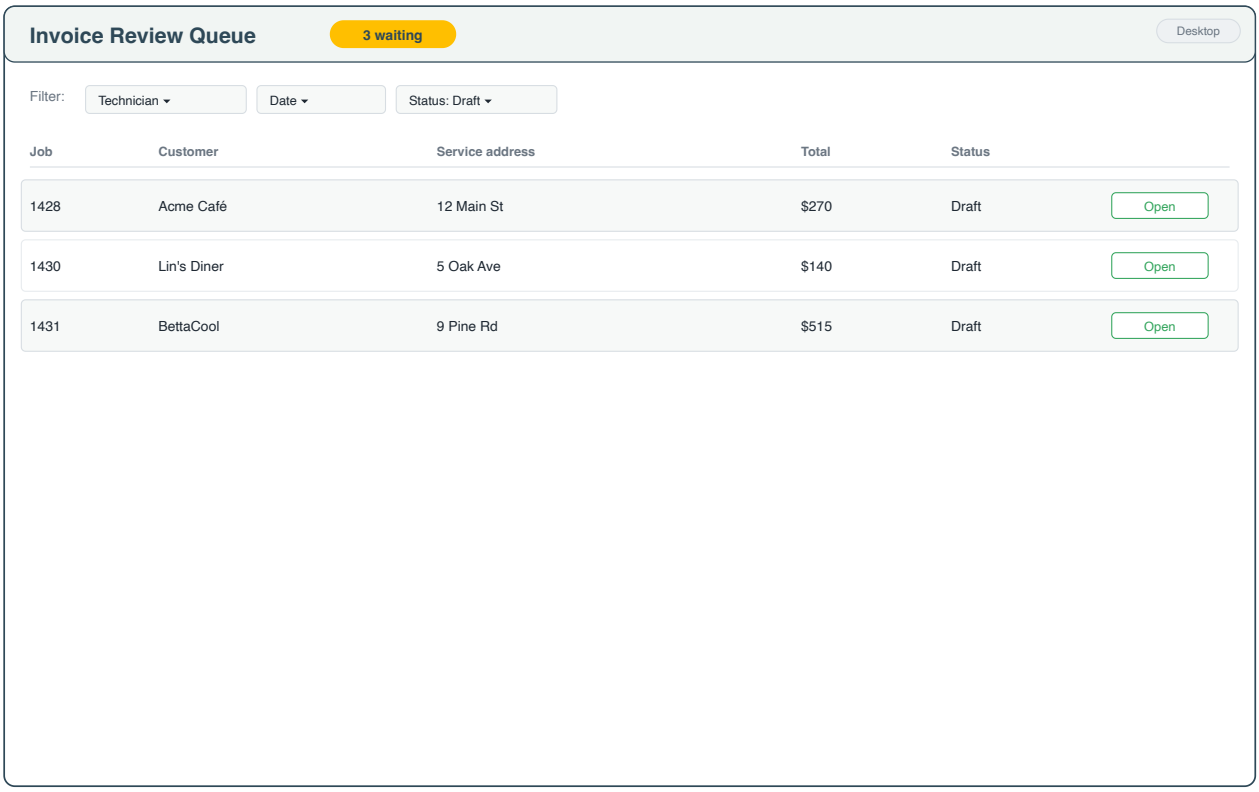
Job & Complete (mobile, 360×780): the technician's on-site screen and the trigger for the automation, carrying the Job record's fields.

Office: Invoice Review Queue (desktop)

Why the office needs it. It is the office's daily worklist: every freshly drafted invoice waiting for a human glance before it goes out. It replaces "trying to remember which jobs still need invoicing", which is where today's misses come from.

What they see. All draft invoices newest-first, each row showing its **job**, the **customer** and **service address**, the **total**, and the **status**, with a badge for how many are waiting. A filter narrows the list by technician, date, or status.

How they work with it. They scan the queue, open each draft to check it, and clear the list toward zero by sending. The badge count makes an unworked backlog impossible to ignore.



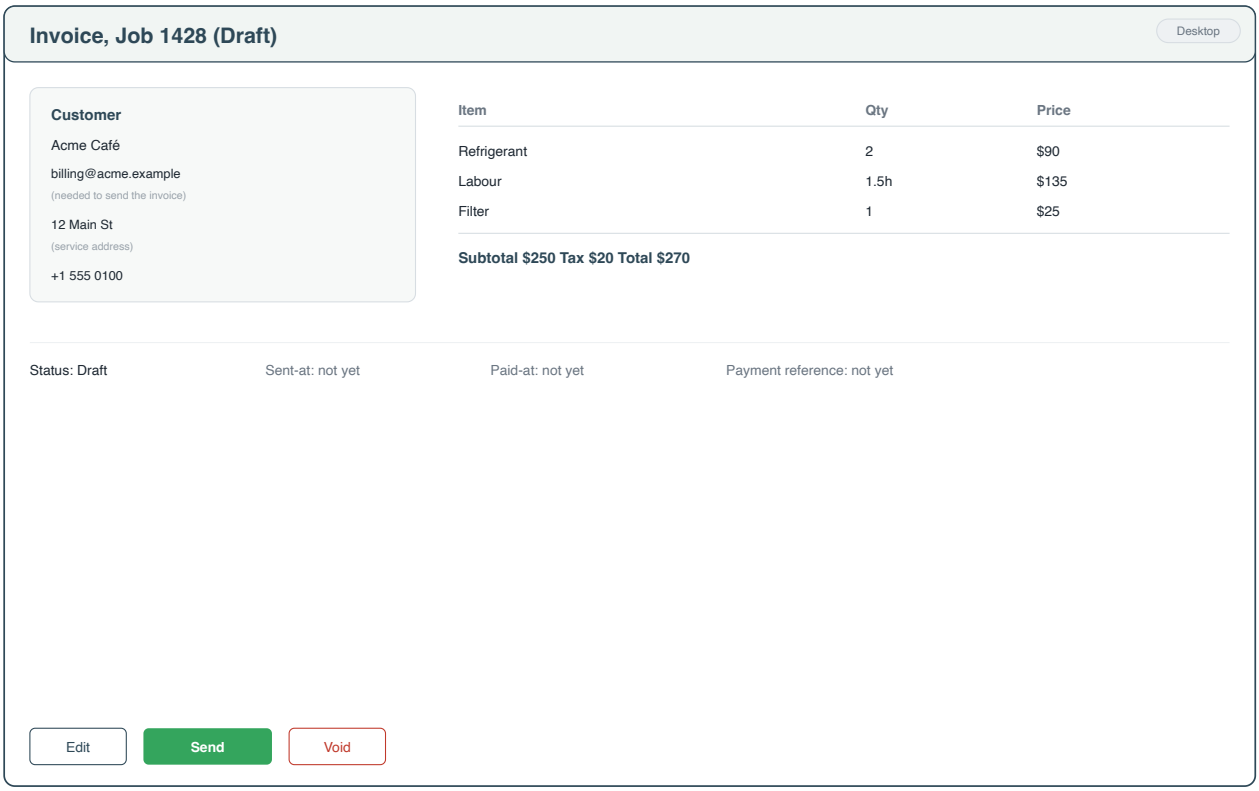
Invoice Review Queue (desktop, 1200×760): the office's worklist of drafts awaiting review, with a waiting-count badge and the key Invoice fields per row.

Office: Invoice Detail / Send (desktop)

Why the office needs it. This is where a draft becomes a sent invoice. It is the one place the office confirms the numbers and actually bills the customer, so it is the gate between "the system drafted this" and "the customer owes this".

What they see. The **customer** (name, billing email, service address) alongside the drafted invoice in full: each **line-item** with quantity and price, the **subtotal**, **tax** and **total**, and the invoice **status**, **sent-at**, **paid-at** and **payment reference** from the Invoice record.

How they work with it. They correct a line if something needs adjusting, then press **Send**, which emails the customer and moves the invoice to Sent in one click; or **Void** if the job should not be billed. Only the office can do either (BR-4), and **Send** is blocked when the customer has no billing email on file (BR-5).



Invoice Detail / Send (desktop, 1200×760): the customer plus the full Invoice record, line-items, totals, status, sent/paid timestamps and payment reference, where the office bills in one click.

What the customer receives. When the office presses **Send**, the customer gets an email like this:

> ***Subject:*** *Your invoice from NorthFlow HVAC (Job 1428)*
>
> *Hi [customer name],*
>
> *Thank you for choosing NorthFlow HVAC. Your invoice for the work completed on [service date] is ready.*
>
> ***Invoice 1428. Total due: $270.00*** *Refrigerant ×2, Labour 1.5h, Filter ×1 (subtotal $250, tax $20).*
>
> *You can view and pay your invoice here: [secure payment link]*
>
> *Questions about this invoice? Just reply to this email, or call the office at [office phone].*
>
> *NorthFlow HVAC*

Also in the system

1. **Office: Billing Board (Phase 2, desktop).** The manager's overview of every invoice by state (Draft, Sent, Paid), with flags for "completed jobs with no invoice yet" and "sent over N days, still unpaid", plus export. This is a later phase, listed here so the whole picture is visible.

4.3 Automations / functions

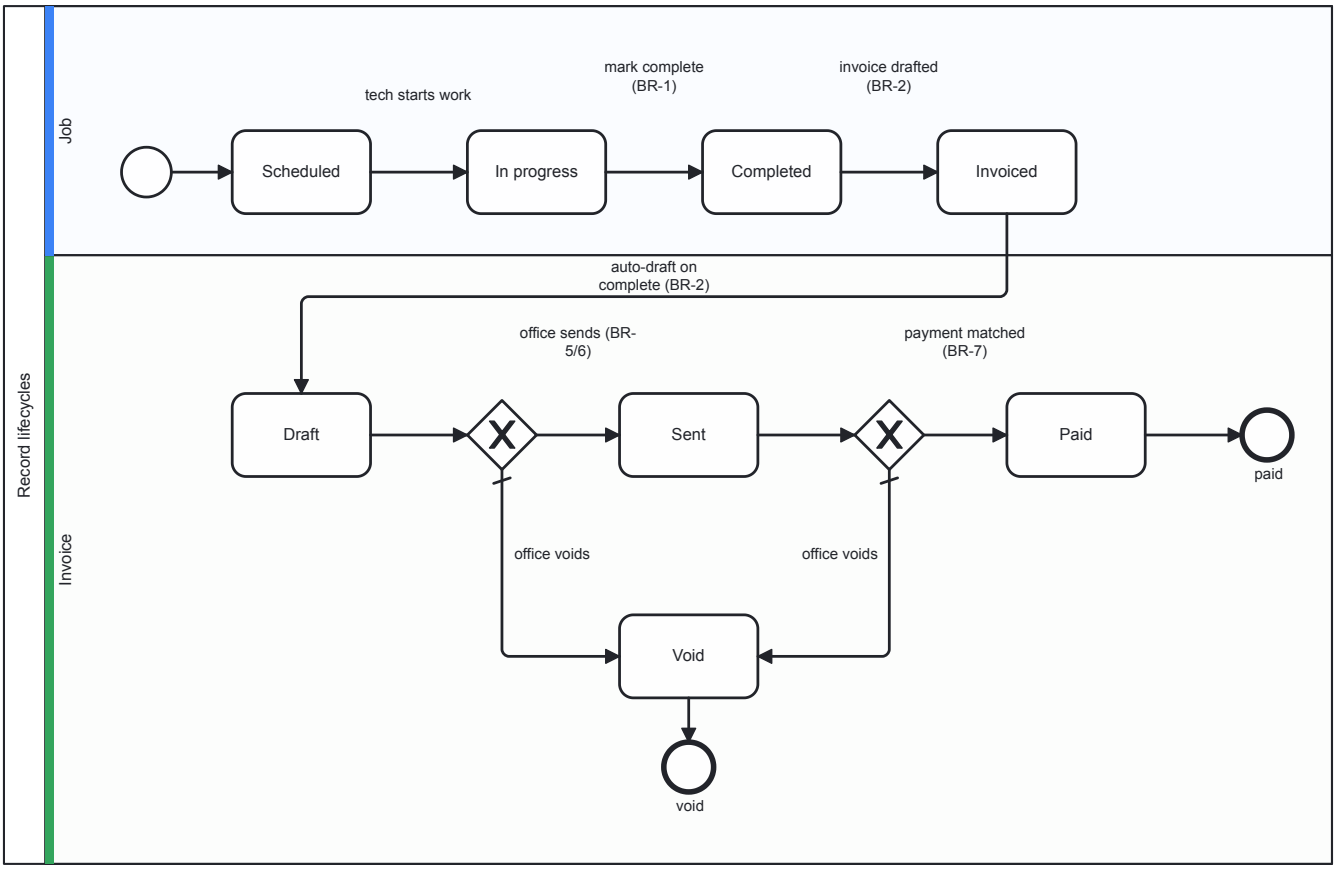
1. **Draft invoice on completion.** When a technician marks a job complete, the system creates a **draft invoice** populated from the job's line-items and price book (totals computed).
2. **Notify office of a new draft.** On draft creation, the system alerts the Invoice Review Queue (a badge, plus an optional email to the office).
3. **Send invoice.** When the office clicks Send, the system emails the invoice (PDF or link) to the customer's billing email and marks it **Sent**.
4. **Match payment.** When the accounting system reports a payment for an invoice, the system marks it **Paid** and records the reference.
5. **Stale-draft reminder.** Daily, the system flags invoices still in **Draft** after the cut-off (for example, same-day) so nothing sits unsent.
6. **Unpaid follow-up flag.** Daily, the system flags invoices **Sent** and unpaid beyond N days on the Billing Board, and optionally nudges the office to follow up.

4.4 Rules & conditions

ID	RULE
BR-1	A job can be marked complete only if it has at least one line-item.
BR-2	Marking a job complete always creates exactly one draft invoice (never a duplicate for the same job).
BR-3	An invoice total = sum(line-items) + tax; the office may adjust before sending.
BR-4	Only an office admin/manager can send or void an invoice; technicians cannot.
BR-5	An invoice can be sent only from **Draft**, and only if the customer has a billing email.
BR-6	Sending moves the invoice **Draft → Sent** and emails the customer.
BR-7	A payment match moves **Sent → Paid**; payments without a matching invoice are flagged, not auto-applied.
BR-8	A voided invoice is final (no re-send); a new one requires re-opening the job.
BR-9	"Same-day billing" target: a completed job's invoice should reach **Sent** the same day; misses surface on the Billing Board.

4.5 States / lifecycle

The two records that move through states (the **Job** and the **Invoice**) and what moves them, as one BPMN diagram (states as nodes in a lane per record). Completing the job is the hinge: it advances the job to *invoiced*, which spawns the draft invoice (BR-2); the invoice then runs its own *draft* → *sent* → *paid* lifecycle, with *void* reachable from draft or sent.



4.6 Roles & permissions: role × step × system matrix

CAPABILITY	FIELD TECHNICIAN	OFFICE ADMIN	OFFICE MANAGER
Record line-items / mark job complete	✓		
Review / edit draft invoice		✓	✓
Send / void invoice		✓	✓
See Billing Board + flags / export		✓ (own)	✓ (all)

> ***Role count (tier).*** *Two **process** roles, **Field Technician** and **Office**, fit the Snapshot 2-role ceiling. **Office Manager** is a permission level of the Office role (all-jobs view plus export), not a third workflow role. **Step by system:** the technician acts at complete (mobile app); the office acts at review, send, and record-payment (the office app, the **accounting system**, and **customer email**).*

4.7 Validations

- A job needs a customer with a billing email before its invoice can be **sent** (caught at send, BR-5).
- Line-items require a quantity > 0 and a known price-book item (or a manual price with a reason).
- Mark-complete requires at least one line-item (BR-1) and is blocked offline until the device re-syncs.

4.8 Integrations register & data contracts

- **Accounting system (one integration, two-way). Out:** on Send, the system pushes the invoice with customer name, billing email, line-items, subtotal, tax, total, invoice number, and sent-date. **In:** the accounting system reports **payment status** (invoice number, amount, paid-date, reference), which drives BR-7. (Drawn here as a sync with QuickBooks Online over its standard authorised connection; the exact product and field names are the kind of detail confirmed with the client before the build.)
- **Customer email:** the invoice is emailed to the customer on **Send** (a sample of that email is in §4.2).
- **Office notifications:** new-draft badge in the Review Queue; optional daily digest of unsent drafts.

4.9 Edge cases & exceptions

- **Offline completion:** a technician completes a job with no signal → the job + line-items queue on the device and sync (and draft) when back online.
- **No billing email:** the invoice drafts normally but **cannot be sent** until the office adds an email (flagged on the Review Queue).
- **Job re-opened after invoicing:** voids/blocks the existing invoice so billing can't double-count.
- **Payment with no matching invoice:** flagged for manual handling, never silently applied (BR-7).

4.10 Engineering coverage: the build-blocking decisions

This is a new build assessed from a description, not an existing system inspected for problems, so this section sets out the **baseline the build implies** rather than findings against current code. The work it describes is part of the §6 jobs (speed and security are built into each one); none of it is quoted as a separate engineering or architecture line.

- **Security & privacy.** Customer contact and billing data is visible only to office roles; technicians see only their own assigned jobs (ties to §4.6); payment references are office-only. *(See §8.)*
- **Architecture-significant decisions.** The **source of truth** for the invoice is *this* system (accounting receives a copy, not the master). The accounting link is **asynchronous and two-way**: Send pushes, and payment status returns on the accounting system's schedule (the match is event-driven, not a blocking call).
- **Build vs. buy.** Accounting stays **bought and integrated** (not rebuilt); email uses a standard sending service; the mobile capture and invoice flow is **built**, because it is the differentiator.
- **Operational expectations.** Cross-reference §8 (volume, speed, availability).
- **Integration contracts.** Cross-reference §4.8 (the per-integration data contracts).

Concrete specifics (shown here, not deferred to a later audit). Email goes out through a standard transactional email service; the accounting link is shown here as QuickBooks Online, which is an illustrative assumption we would confirm before build (see §9.2). On a failed push the system retries with back-off (for example after 1, 5, and 30 minutes) and then flags the invoice for manual attention rather than dropping it silently. Customer and billing data is kept for the active accounting period plus the statutory record-keeping window, then archived.

5. Acceptance criteria

- Completing a job with line-items produces exactly one draft invoice with correct totals (BR-1/2/3).
- A draft appears in the office Review Queue within moments of completion, with a badge count.
- Sending a draft emails the customer and moves it to **Sent** (BR-5/6); a customer with no billing email is blocked with a clear message.
- A payment reported by accounting moves the matching invoice to **Paid**; an unmatched payment is flagged.
- The Billing Board shows completed-but-uninvoiced jobs and sent-but-unpaid invoices.
- Technicians cannot send or void invoices (BR-4).

6. Phased build plan

Phase 1 (the slice that pays for itself) is the top of the §3 opportunity ranking: capture line-items at source, auto-draft the invoice on Mark-complete, and send it the same day from a review queue. That is the slice that removes the nightly re-typing and the billing lag, and it is sized to the Snapshot tier (a floor of 50 developer-hours; this Phase 1 is 145 hours, inside the Snapshot 50 to 150 hour range). The work is grouped into **three epics**, by significance; under each epic the **jobs** say what is built, why it matters, and how it is achieved; and under each job the **steps** are the actual build, **each one small** — no step runs longer than two days, so you see exactly what is being done and what it costs. The epic and job figures are simply the **sum of their steps**. The delivery work each job implies (its screen design, its tests, the shared setup it rides on) is folded into the steps, so the total is the whole shippable build with nothing left out. Duration is shown in working days; a developer does 4 to 6 productive hours in a day.

WORK AND STEPS	DURATION	COST
Epic 1. Capture and auto-draft	**~13-20 d**	**$8,000**
Technician mobile capture. The technician's two phone screens and the trigger for the whole automation, where line-items are picked, notes and photos added, and **Mark complete** fires the draft. It removes the paper-and-re-type loop at its source and stops missed line-items. *How we achieve it: build the two phone screens carrying the §4.1 Job fields, wire the line-item picker to the price book, enforce the no-empty-job rule on completion, then make it work offline and sync safely.*	**~8.5-13 d**	**$5,100**
My Jobs list The technician's **My Jobs** phone screen (§4.2): today's open jobs with address, customer and status, tap to open.	1-1.5 d	$600
Job & Complete screen — fields and layout The **Job & Complete** screen (§4.2) carrying the Job fields from §4.1, laid out for one-handed use on site.	1.5-2 d	$800
Line-item picker from the price book Add line-items on the job by picking from the price book (the integration in §7), quantity and price filled from the book.	1.5-2 d	$800
Notes and photos on the job Attach job notes and site photos on the same screen, stored against the Job (§4.1).	1-1.5 d	$600
Mark complete: block empty jobs, advance the state **Mark complete** enforces BR-1 (no complete without at least one line-item) inline and moves the Job to *completed* (§4.5), firing the auto-draft.	1-2 d	$700
Offline completion on the device Let a technician complete a job with no signal: queue the completion on the device so work is never lost (§4.9).	1.5-2 d	$800

WORK AND STEPS	DURATION	COST
Conflict-safe sync when back online Sync the queued completion when signal returns, reconciling without double-posting or overwriting (§4.9).	1.5-2 d	$800
Auto-draft invoice engine. The automation itself: on Mark-complete it creates exactly one draft invoice from the job's line-items, totals computed, and alerts the office. This is what ends the overnight re-typing and the roughly two-day billing lag. *How we achieve it: create one draft per completed job with a dedupe guard, compute the money, notify the office, and handle the re-open and missing-price cases cleanly.*	**~5-7 d**	**$2,900**
Create one draft per job (dedupe guard) On Mark-complete create exactly one draft invoice from the job's line-items, with a dedupe guard so a re-fire makes no second draft (BR-2), setting the Invoice to *draft* (§4.5).	1.5-2 d	$800
Compute subtotal, tax and total Total the draft from its line-items: subtotal, tax and total per BR-3.	1-1.5 d	$600
Notify the office review queue Tell the office a draft is ready by surfacing it in the Review Queue (§4.2).	1-1.5 d	$500
Void the draft when its job re-opens If a completed job is re-opened, void its draft so the numbers can't go out stale (BR-8).	1-1.5 d	$500
Handle a missing price-book entry When a line-item has no price-book match, draft it flagged for office review rather than failing the job (§4.9).	1-1.5 d	$500
Epic 2. Office billing	**~6.5-9.5 d**	**$3,800**
Office review, send and email. The office's two desktop screens that turn a draft into a sent invoice: the Review Queue worklist, and the Invoice Detail screen where the office checks the numbers and bills the customer in one click. It replaces "remembering which jobs still need invoicing" with a badge that counts down to zero. *How we achieve it: build the queue with its badge and filters, build the detail-and-send screen, restrict send to the office, then wire the send-and-email with its no-email guard.*	**~6.5-9.5 d**	**$3,800**
Invoice Review Queue with the count badge The office **Invoice Review Queue** worklist (§4.2) of drafts awaiting send, with the badge that counts down to zero.	1.5-2 d	$800
Queue filters and stale-draft surfacing Filter the queue and surface drafts sitting too long, so nothing is forgotten.	1-1.5 d	$600

WORK AND STEPS	DURATION	COST
Restrict send and void to the office Only the office role may send or void an invoice (BR-4); the technician cannot.	1-1.5 d	$500
Send: move to sent and email the customer **Send** moves the Invoice *draft to sent* (§4.5) and emails it to the customer (BR-5, BR-6).	1-2 d	$700
Block send when there is no billing email If the customer has no billing email, block the send and prompt the office to add one (§4.9).	0.5-1 d	$400
Epic 3. Data import and go-live	**~4.5-7 d**	**$2,700**
Data import and go-live. The shared work that serves every screen above and cannot fold into one of them: bring NorthFlow's real customers and price book into the system, then deploy and hand over. *How we achieve it: import each dataset behind a dry-run report and reconcile it, then turn the §5 checks into tests, deploy, and walk the office through the queue once.*	**~4.5-7 d**	**$2,700**
Import customers (dry-run + reconcile) Import NorthFlow's customers behind a dry-run validation report, reconciling the count before go-live (§7).	1-2 d	$700
Import the price book (dry-run + reconcile) Import the price book the same way, reconciling totals so the picker prices correctly (§7).	1-1.5 d	$600
Wire the acceptance checks as automated tests Turn the §5 acceptance checks into automated tests that run on every change.	1-2 d	$700
Go-live deploy and office walkthrough Deploy to production and walk the office through the queue once so they own it on day one.	1-2 d	$700
Total — Phase 1	**~24-36 d**	**$14,500**

The three epics above are the whole Phase-1 build: every job is broken into named steps, each priced and **none longer than two days**, so the plan is visible all the way down rather than hidden inside a job total. There is no separate engineering or architecture line: speed and security are part of each step, not a phase of their own, because there is no existing system to harden (this is a new build). Anything that would push past the slice (payment-match, the Billing Board, reporting) stays in the Phase 2 sketch below. The scope fits the Snapshot ceilings (2 roles, up to 8 steps, up to 2 integrations, up to 3 entities, up to 5 screens, up to 10 rules, a linear lifecycle), and is specified to build with no further discovery.

Phase 2 (sketch): close the loop and give the office its overview. Once invoices are reliably going out, the next phase matches payments and adds the management view. *What is suggested.* The payment-match automation that moves an invoice from Sent to Paid when the accounting

with flags for stale drafts and sent-but-unpaid invoices; and, building on those, reporting and email digests, a price-book screen the office maintains itself, and multi-technician dashboards. *How the system benefits.* It closes the loop from sent to paid without anyone reconciling by hand, gives the office one place to see and chase what is outstanding, and turns the captured data into management insight. *Why this is a later phase.* Payment-match depends on the accounting system exposing payment status (the dependency in §7), and all of it earns its keep only once Phase 1 has invoices going out reliably. It is sketched, not quoted: expect another Snapshot-scale slice, roughly comparable in size to Phase 1, shaped once Phase 1 is live.

7. Risks & dependencies

Each item below is something that could slow or block the build. For each one we have set out why we think it is a risk and what we would do about it, so the reasoning is visible rather than left as a note to ourselves.

- **Access to the accounting system's payment data.** Phase 2's payment-match automation can only mark an invoice paid if the accounting system will tell the system when a payment arrives. Not every accounting setup exposes that, and some only expose it on a paid plan or through a connector that has to be turned on. We flag this as a risk because the whole "Sent to Paid" step rests on it. If the data turns out not to be available, Phase 2 falls back to the office marking invoices paid by hand, which still works, just with a manual step. This is the first thing to confirm with NorthFlow before Phase 2 starts.
- **Quality of the existing price book.** The auto-drafted invoice totals are only as accurate as the price book they are computed from. If NorthFlow's current price list has gaps, stale prices, or duplicates, the drafts inherit those errors and the office ends up correcting every invoice, which defeats the point of the automation. That is why we treat a price-book clean-up as a likely prerequisite, folded into the data-import job in §6 rather than left as an afterthought.
- **Field connectivity.** Technicians finish jobs in basements, mechanical rooms, and rural sites where there is often no signal, so the capture screen cannot assume it is online. If a completion were lost because the device was offline, the office would never get the draft and the job would silently go unbilled. We have designed for this with the offline queue-and-sync handling in §4.9; the residual risk is making sure that sync is reliable on the technicians' actual devices, which the acceptance tests in §5 are there to exercise.

8. Non-functional expectations

- **Volume:** dozens of jobs/day (25-person company); the queue/board stay responsive at that scale.
- **Speed:** a draft is ready within moments of completion; sending is one click.

- **Availability:** the office works in business hours; the app stays available through the working day, with any maintenance taken in a quiet window.
- **Security/privacy:** customer contact + billing data is visible only to office roles; technicians see only their assigned jobs.
- **Failure handling and records:** a failed accounting push retries and then flags the invoice rather than dropping it (see §4.10); customer and billing data is backed up daily and kept for the active accounting period plus the statutory record-keeping window.
- **Maintenance:** the office manages the price book and customer emails; no developer needed for daily use.

9. What still needs confirming

These are the points still open before the build is a green light. None of them have been guessed into the specification above; the spec stays grounded and the uncertainty lives here, named, in three groups.

9.1 Open questions

1. Does NorthFlow want partial invoices for multi-visit jobs, or one invoice per job?
2. Does the accounting system expose payment status to the system, so the Phase-2 match automation can mark an invoice paid on its own?
3. What same-day cut-off time should the stale-draft reminder use?

9.2 Assumptions we made (to confirm)

Where the brief was silent we proceeded on a reasonable assumption rather than leaving a hole, so the picture is complete. Each is shown for confirmation, not asserted as fact.

1. We assumed the accounting system is QuickBooks Online and that the link is a two-way connection over its standard authorised access. If it is a different system the data contract in §4.8 stays the same; only the connector changes.
2. We assumed the loaded cost of the office re-typing time is about $35 an hour, which is what the cost-of-status-quo figure in §3 rests on.
3. We assumed NorthFlow invoices on the order of $120,000 a month, used to size the cash-flow leakage.
4. We assumed a retry schedule of about 1, 5, and 30 minutes on a failed accounting push before the invoice is flagged for manual attention.
5. We assumed "stale" means a draft left unsent past the same-day cut-off in question 3.

9.3 Requirements before we start

What NorthFlow needs to have in place before Phase-1 build begins, so neither side is held up on day one.

1. Access and credentials for the accounting system, so the invoice push can be set up and tested.
2. The current customer list and the price book, exported for the one-time data import in §6.
3. A billing email address for each customer (or agreement on how to backfill the ones that are missing), since the "send invoice" step needs somewhere to send it.
4. Sign-off on the invoice email wording and the same-day cut-off time.
5. A short window of the office manager's time to walk through the queue screen once it is ready, and to confirm the first few auto-drafted invoices before the team relies on them.