ZNotice — Business Model & Payments

Onboarding & access policy (Phase 13)

- **Existing accounts are admins.** Accounts created before Phase 13 keep the `admin` role and are never billing-locked.
- **New signups are regular users.** Registration hardcodes `role = 'user'` (a `role` field in the signup POST body is ignored), and every new account is **locked to the Billing and Account pages** until an admin confirms the first payment.
- **Locked means Billing + Account (plus the integration request form).** Every other `/dashboard/*` page redirects to the pending payment's pay page (`/dashboard/billing/pay/<id>`), or to `/dashboard/billing` when there is no pending payment, with an error flash explaining the account is not active yet. `/integration` is allowed because the integration request form (shown on the Billing page) creates a pending $20 payment, so it is part of the payment flow.
- **Unlock condition:** at least one `transactions.status = 'paid'` **or** a positive credit balance (`SUM(credit_ledger.delta) > 0`). Pending/failed transactions alone do **not** unlock the account.
- Admins are always unlocked and are exempt from the lock. `/api/v1/*` is unchanged (sends already fail with `402 insufficient_credits` at a zero balance).

1. Pricing

ZNotice is pay-as-you-go. There is no subscription or monthly fee; the optional sender-name add-on is purchased for a fixed period and renewed manually.

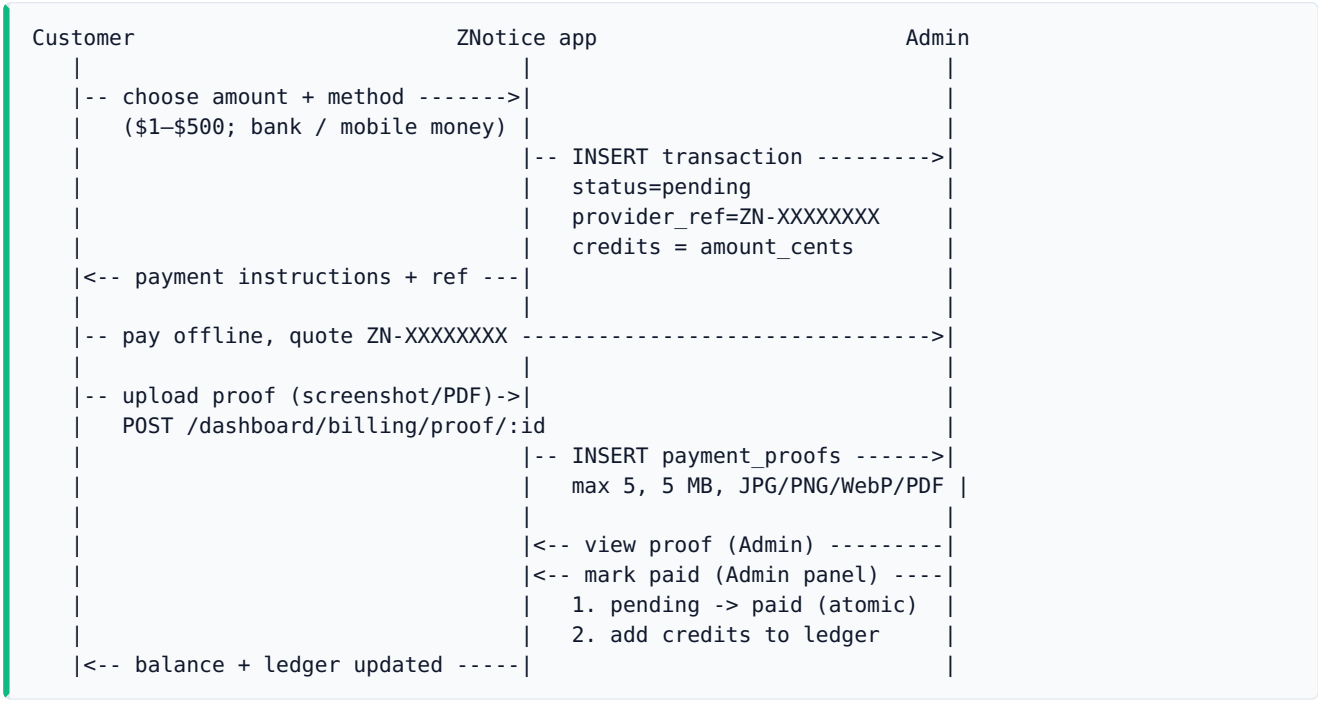
Item	Price	Unit	Notes
Message credits	**$0.01** each	per message	`pricePerCreditCents = 1`
Credit pack	**$10.00 = 1,000 credits**	pack	`pricePer1000CreditsCents = 1000`
Custom top-up	**$1.00 – $500.00**	per checkout	Credits always equal the amount in cents (100–50,000)
Dedicated sender name	**$10.00 / 30 days**	add-on	`sender_name_price_cents = 1000`, `sender_name_days = 30`; manual renewal
Integration service	**$20.00**	one-time	`integrationFeeCents = 2000`

- Credits are deducted **1 per successfully sent message**. Failed sends are automatically refunded (see §6).
- Credits do not expire and are non-transferable.
- Every movement is recorded in the `credit_ledger` (auditable, running balance). A balance is always `SUM(delta)` for the user.
- The **dedicated sender name** add-on is separate from message credits: it grants the ability to set the WhatsApp profile/push name recipients see. Messages are still billed at `$0.01` each; the add-on does not include credits. Attachments/media are **not** gated by the add-on.

- The integration fee, the per-1000 price and the add-on price/period are stored in the `settings` table and can be changed by an admin (see §5).

2. Payment workflow (credit top-up)

Credit top-ups are paid one of three ways: **manually** — the customer transfers funds offline and an admin approves (the default at launch, described below) — or **instantly** via IntouchPay mobile money (MTN MoMo / Airtel Money) or via KitovuPay mobile money (MTN MoMo), both of which are auto-confirmed with no admin approval and no proof upload (see the IntouchPay and KitovuPay subsections at the end of this section). Every path grants credits exactly once.



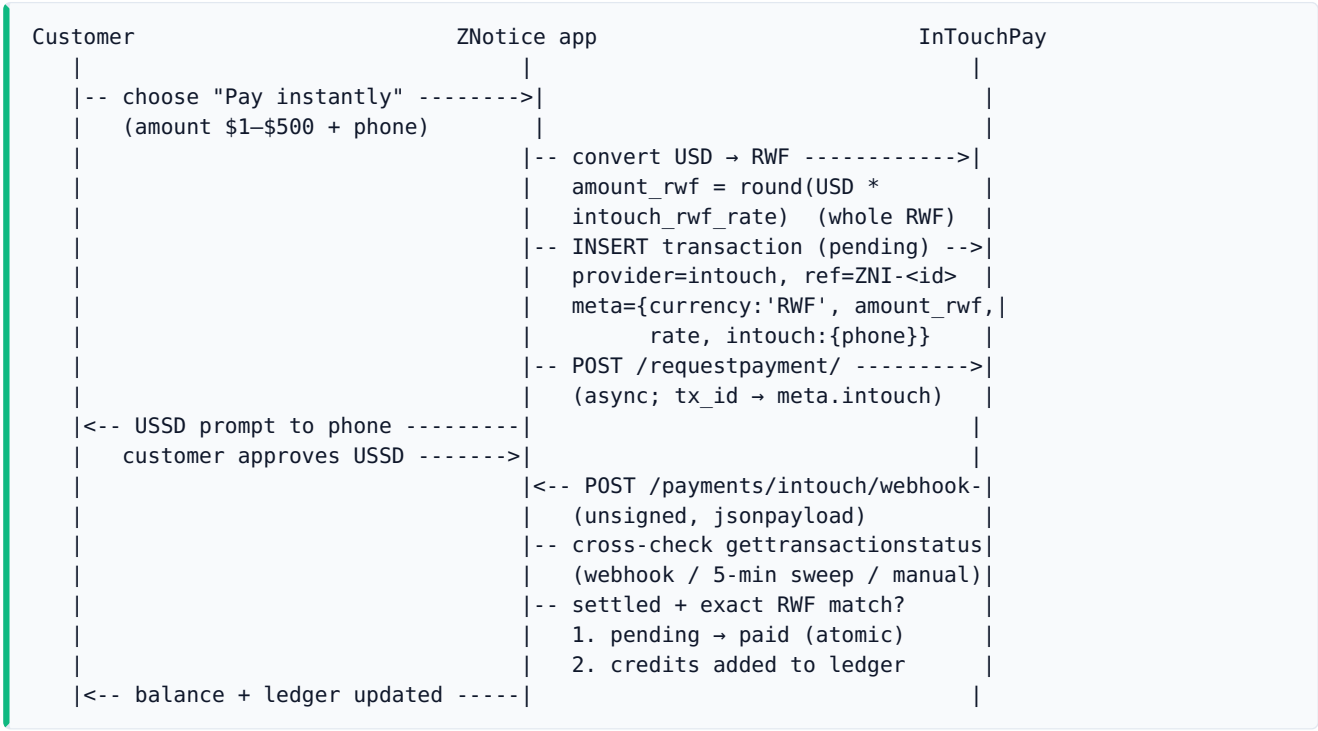
Steps in detail:

1. Customer opens **Dashboard → Billing**, enters an amount (or picks a preset: $10 / $25 / $50 / $100) and a method (bank transfer or mobile money), then continues to payment.
2. The app creates a `pending` transaction with a unique reference `ZN-XXXXXXXX` (8 unambiguous characters; no `0/O/1/I/L`) and shows the admin-configured payment instructions.
3. The customer transfers the exact amount offline, quoting the reference.
4. **The customer uploads proof of payment** on the same pay page (`POST /dashboard/billing/proof/:id`): a screenshot or receipt as **JPG, PNG, WebP or PDF, up to 5 MB**, with up to **5 proofs** per transaction. The file type is verified by magic bytes (the client MIME type and filename are never trusted) and proofs can be added, viewed or deleted only while the transaction is `pending`. Proofs stay readable by the owner and admins after settlement for audit.
5. An admin reviews the uploaded proof(s) from **Admin → Transactions** (the "Proofs (N)" column links each file) and marks the transaction **paid**.
   - The `pending → paid` transition is atomic and idempotent: a double-click cannot double-credit.
   - On success, credits equal to `amount_cents` are added to the ledger (`reason = topup`).
   - If crediting fails, the transaction is rolled back to `pending` for safe retry.
6. The customer's balance and ledger update; they can send immediately.

Customers can cancel a `pending` request themselves from the billing page (`POST /dashboard/billing/cancel/:id`). Pending requests are also visible in the admin overview.

IntouchPay instant payment (credit top-up)

Credit top-ups can instead be paid **instantly** with MTN MoMo or Airtel Money via **IntouchPay** (Rwanda). There is no offline transfer, proof upload or admin approval — the customer approves a USSD prompt on their phone and the app confirms the payment itself by cross-checking the gateway.



Steps in detail:

1. The customer opens **Dashboard → Billing**, picks the "Pay instantly with MTN MoMo / Airtel Money" card, and enters the amount (1–500) and their mobile number.
2. **USD → RWF conversion.** The app converts the USD amount to **whole RWF** at the admin-editable operator rate `intouch_rwf_rate` (float, default `1477.29` RWF per USD; see §5) and shows the RWF amount as a hint. The rate and RWF amount are written into the transaction `meta`; the transaction itself stays in USD (`amount_cents`), so the ledger and analytics remain USD-only.
3. The app registers the payment with IntouchPay (`POST /requestpayment/`, async) and stores the gateway `tx_id`. IntouchPay sends a **USSD prompt** to the customer's phone, which the customer approves to authorize the payment.
4. **Auto-confirmation (no admin, no proof).** The payment is confirmed by **cross-checking the gateway** — never by trusting the callback alone:
   - the **webhook** `POST /payments/intouch/webhook` (unsigned) triggers a `gettransactionstatus` cross-check;
   - a **5-minute sweep** (`server/services/intouchSweep.js`) cross-checks pending Intouch transactions older than 2 minutes and confirms settled ones; and
   - the customer (or an admin) can run a **manual status check** (`POST /dashboard/billing/intouch/:id/status`). Settlement requires the gateway status to be settled **and** the returned RWF amount + currency to match the transaction exactly (whole RWF, off-by-one blocked). Credits

equal to `amount_cents` are then added to the ledger exactly once, using the same atomic `pending → paid` gate as the manual flow.

5. The balance updates immediately; the customer can send right away.

Notes:

- **Credit top-ups only.** The dedicated sender-name add-on (§3) and the integration service (§4) stay on the **manual** workflow — there is no IntouchPay checkout for them.
- **No processor refunds.** IntouchPay exposes no reversal endpoint, so refunds are **local-ledger only** (an admin credit adjustment), exactly like manual payments (see §6).
- A still-pending payment can be re-requested (`POST /dashboard/billing/intouch/:id/start`); the same `POST /dashboard/billing/cancel/:id` route cancels it.

KitovuPay instant payment (credit top-up)

Credit top-ups can also be paid **instantly** with MTN MoMo via **KitovuPay** (Rwanda). There is no offline transfer, proof upload or admin approval — the customer approves the prompt on their phone and the app confirms the payment itself by cross-checking the gateway.

Steps in detail:

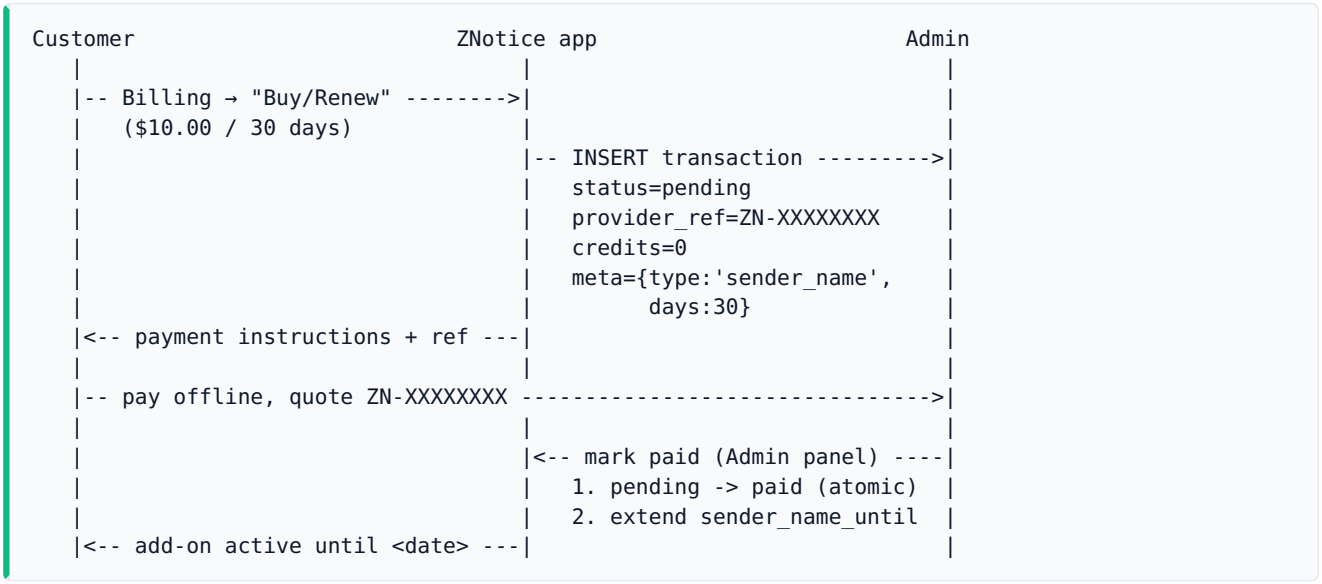
1. The customer opens **Dashboard → Billing**, picks the "Pay instantly with MTN MoMo (KitovuPay)" card, and enters the amount (1–500), their MTN MoMo number, and (optionally) the account name.
2. **USD → RWF conversion.** The app converts the USD amount to **whole RWF** at the admin-editable operator rate `kitovu_rwf_rate` (float, default `1477.29` RWF per USD; see §5) and shows the RWF amount as a hint. The rate and RWF amount are written into the transaction `meta`; the transaction itself stays in USD (`amount_cents`).
3. The app sends a signed collection request (`POST /merchant-transaction/merchant/collection/mobile_money`, HMAC-SHA256 signature, `payment_rail:'mtn'`) with a unique **6-digit `externalId`**, and stores the gateway transaction id. KitovuPay prompts the customer on their phone, which the customer approves to authorize the payment.
4. **Auto-confirmation (no admin, no proof).** The payment is confirmed by **cross-checking the gateway** — never by trusting the callback alone:
   - the **webhook** `POST /payments/kitovu/webhook` (unsigned) triggers a status cross-check;
   - a **5-minute sweep** (`server/services/kitovuSweep.js`) cross-checks pending Kitovu transactions older than 2 minutes and confirms settled ones; and
   - the customer (or an admin) can run a **manual status check** (`POST /dashboard/billing/kitovu/:id/status`). Settlement requires the gateway status to be `APPROVED` (and any webhook amount to match exactly). Credits equal to `amount_cents` are then added to the ledger exactly once, using the same atomic `pending → paid` gate as the manual flow.
5. The balance updates immediately; the customer can send right away.

Notes:

- **Credit top-ups only.** The dedicated sender-name add-on (§3) and the integration service (§4) stay on the **manual** workflow — there is no KitovuPay checkout for them.
- **No processor refunds.** Refunds are **local-ledger only** (an admin credit adjustment), exactly like manual payments (see §6).
- A still-pending payment can be re-requested (`POST /dashboard/billing/kitovu/:id/start`); the same `POST /dashboard/billing/cancel/:id` route cancels it.

3. Dedicated sender name add-on ($10.00 / 30 days)

The optional **dedicated sender name** add-on lets a customer set the WhatsApp profile/push name their recipients see (the Phase 9 feature). It is sold for a fixed period and uses the **same manual payment + admin approval** flow as a credit top-up.



Key behavior:

- **Price / period are server-side settings.** The customer cannot influence them: the dashboard route reads `sender_name_price_cents` (default `1000`) and `sender_name_days` (default `30`) and writes them into the transaction `meta`; `meta.days` is clamped to **1..365** at settlement.
- **Entitlement.** The paid period is stored as `users.sender_name_until` (UTC). The add-on is active while that instant is in the future. The WhatsApp sender-name form is gated on an active entitlement; while inactive or expired the card shows a Billing call-to-action and the form is disabled.
- **Renewal extends, never resets.** Settlement sets `sender_name_until = GREATEST(existing expiry, NOW()) + days`, so buying a renewal before expiry **adds** the new period to the remaining paid time and no paid time is lost.
- **One pending purchase at a time.** The checkout route locks the user row and re-checks for an existing pending `sender_name` transaction before inserting, so concurrent clicks cannot create multiple pending purchases.
- **No credits are involved.** The transaction carries `credits = 0`; approving it extends the entitlement instead of adding credits.
- **Dashboard-only.** There is no API v1 endpoint for the add-on and no change to message pricing ($10 = 1,000 credits).

If an admin approves a payment and the entitlement update fails, the transaction is rolled back to `pending` so it can be retried without losing the payment (see §2 for the same compensation on credit top-ups).

4. Integration service orders

The optional **$20 one-time integration service** is a done-for-you setup: endpoint wiring, webhooks/ delivery handling, test messages and handover.

Flow:

```
Customer submits requirements + contact channel + optional preferred stack
        |
        v
integration_orders row: status = requested
        |
        +--> pending $20 transaction (provider_ref ZN-XXXXXXXX, credits = 0)
        |
        v
Admin approves the $20 payment  --> order status = paid
        |
        v
Admin works the order            --> status = in_progress
        |
        v
Admin marks complete             --> status = delivered
```

Order status	Meaning
`requested`	Customer submitted the request; $20 payment pending.
`paid`	Payment approved; work has not started.
`in_progress`	Admin/team is building the integration.
`delivered`	Work completed and handed over.
`cancelled`	Order cancelled (set by an admin).

- The `$20` service fee is **not** a credit purchase: its transaction carries `credits = 0` and settling it advances the linked order instead of adding credits.
- Order status and `admin_notes` are managed at **Admin → Orders**.
- Customers can track order progress and the linked reference on the Billing page.

5. Editing payment instructions and pricing (admin)

Sign in as an admin and go to **Admin → Settings**. Relevant fields:

Setting	Purpose
`manual_payment_instructions`	Text shown to customers on the payment page. Supports `{{reference}}`, `{{amount}}` and `{{credits}}` placeholders.
`billing_email`	Shown on the payment page for questions.
`integration_fee_cents`	One-time integration fee in cents (default `2000` = $20.00).
`sender_name_price_cents`	Dedicated sender-name add-on price in cents per period (default `1000` = $10.00).
`sender_name_days`	Dedicated sender-name add-on period in days (default `30`; clamped 1–365 at settlement).
`intouch_rwf_rate`	RWF per USD used to convert credit top-ups to IntouchPay mobile money at checkout (float, default `1477.29`).

Setting	Purpose
`kitovu_rwf_rate`	RWF per USD used to convert credit top-ups to KitovuPay mobile money at checkout (float, default `1477.29`).
`default_country_code`	Country code used to normalize `0`-prefixed phone numbers.
`send_rate_per_min`	Per-key send rate limit.
`max_concurrent_sessions`	Gateway capacity guard.

Example `manual_payment_instructions`:

```
Bank transfer: Example Bank, Acct 1234567890, Name ZNotice Ltd.
Mobile money: +250 7XX XXX XXX (ZNotice).
Amount: ${{amount}} for {{credits}} credits. Reference: {{reference}}
Send proof to billing@consultbofa.com quoting {{reference}}.
```

Changes are validated and applied immediately (the settings cache refreshes).

6. Refunds, cancellations and suspension — policy notes

Automatic refunds (technical behavior)

- If the WhatsApp engine rejects a send, the app stores a `failed` message and writes a `+1` refund entry to the ledger (`reason = refund`, linked to the failed message). The customer is not charged for failed sends.

Manual refunds / adjustments

- Admins can adjust any balance at **Admin → Users → Credits** (positive to grant, negative to deduct). A deduction that would make the balance negative is rejected. These are recorded as `admin_adjust`.
- The `transactions.status` enum includes `refunded`, but there is currently **no UI or route that sets it** — manual monetary refunds are handled out-of-band and recorded via an admin credit adjustment. (See the discrepancy note in the project report.)

Cancellations

- A customer may cancel their own `pending` top-up; it becomes `failed`. The same cancel route applies to a pending sender-name purchase.
- Integration orders can be set to `cancelled` by an admin.
- Cancelling a pending sender-name purchase does not remove an already-active entitlement; it only discards the unpaid renewal.

Suspension

- Admins can suspend/activate users at **Admin → Users**. Suspended users cannot send (`403 account_suspended`) and API keys are rejected.
- Per the Terms of Service, ZNotice may rate-limit, suspend or terminate accounts that violate the Acceptable Use Policy, evade limits, or pose a risk to the platform. Credits are non-transferable.

- Because ZNotice uses an **unofficial WhatsApp automation client**, a number may be rate-limited, restricted or banned by WhatsApp. This risk is disclosed in `/terms` and is borne by the customer.

Billing contacts

- Default billing email: `billing@consultbofa.com` (editable in settings).
- Support / terms contact: `support@consultbofa.com`.