HamroKam — End-to-End Data Flow

Prepared: 2025-08-13 09:57

This document describes the data movement and storage practices from user creation to payment completion.

HamroKam — End-to-End Data Flow (User → Login → Task → Payment)

Prepared: 2025-08-13 09:57

This document explains how data moves through the HamroKam app from account creation to login, task posting, offers, ID verification,
escrow payment, completion, and payout — with an emphasis on **not storing bank or ID documents** on HamroKam servers.

Principles
- Payments are handled by **Stripe Connect** (marketplace), so card and bank data never touch HamroKam's servers.
- Identity verification is handled by a third-party (e.g., **Stripe Identity** or **Australia Post Digital iD**).
- HamroKam stores only **non-sensitive references** (IDs/tokens from providers) and the minimum personal info needed.
- Backend is a stateless API with JWT-based auth; data is persisted in MongoDB Atlas.

Key Entities & External IDs

HamroKam Entities
- **User**: { id, name, email, phone, roles[poster|tasker], status, verificationStatus, createdAt }
- **Task**: { id, title, description, category, budget, location, images[], createdBy, status, createdAt }
- **Offer**: { id, taskId, offeredBy, amount, message, status[pending|accepted|rejected], createdAt }
- **Payment**: { id, taskId, posterId, taskerId, amount, fee, status, providerRefs, history[] }
- **Message**: { id, taskId, from, to, text, createdAt }

External/Provider References (stored by HamroKam)
- **Stripe**:
  - `customerId` for posters
  - `connectedAccountId` for taskers (payouts)
  - `paymentIntentId` for escrow
  - `transferId` (or balance transaction IDs) for payouts/fees

- **Identity Provider**:
  - `verificationSessionId` (status only, no images kept)

1) User Creation (Sign Up)

Flow
1. User submits name, email, password (hashed with bcrypt on server) and accepts Terms/Privacy.
2. Server creates a User record; sends email verification (optional but recommended).
3. (If tasker role selected) show “Connect your bank” and “Verify ID” as next steps, but these can be completed later.

Data stored
- User profile (no bank, no card, no ID scans)
- Password hash + salts
- Email verification token (short-lived)

2) Login & Session Management

Flow
1. User logs in with email + password.
2. Server verifies hash and issues a short-lived **access JWT** (e.g., 15min) and a **refresh token** (httpOnly cookie).
3. Client stores access token in memory; uses refresh endpoint to get new tokens when needed.
4. Dashboard is **role-aware**: “Hire” (Poster) or “Work” (Tasker).

Data stored
- JWT is not stored server-side (stateless), but refresh token identifiers can be kept to allow revocation.
- Basic device/session logs for security auditing.

3) Task Posting (Poster)

Flow
1. Poster fills title, description, category, budget, location; optionally uploads images (to Cloudinary/S3).
2. Server creates Task with status `open`.
3. Task appears in browse/search for taskers.

Data stored
- Task document with poster’s userId; image URLs only (no raw files on app server).

4) Offers & Acceptance

Flow
1. Tasker browses tasks and submits an **Offer** (amount + message).
2. Poster reviews incoming offers and **accepts** one.
3. On acceptance, server prepares **payment** (escrow) via Stripe.

Data stored
- Offer document (links taskId and taskerId, price, status).
- Minimal audit log for offer actions.

5) Tasker Identity & Payout Setup (Third-Party)

Flow
- From the Tasker profile, user clicks **Verify ID** → redirect to identity provider's hosted flow.
- Provider captures document/selfie; on completion returns a **verification status** (pass/fail).
- For payouts, Tasker connects bank via **Stripe Connect hosted onboarding**; HamroKam receives only a `connectedAccountId`.

Data stored
- `verificationStatus` (unverified|pending|verified) and provider session ID (reference only).
- `connectedAccountId` for payouts.

Not stored
- No bank numbers, no photos/scans of IDs.

6) Payment Authorization / Escrow (When Poster Accepts Offer)

Flow (Stripe Connect - Destination Charges)
1. Server creates a **PaymentIntent** for the total task amount (e.g., $100).
2. Poster pays (card/Apple Pay/Google Pay). Stripe handles card details and 3DS if required.
3. On success, PaymentIntent status becomes `requires_capture` or `succeeded` (depending on flow). Funds are effectively **held in escrow** until completion.

Data stored
- Payment record with: taskId, posterId, taskerId, amount, fee %, `paymentIntentId`, status=`in_escrow`.

Not stored
- No card numbers; Stripe tokens/IDs only.

7) Work Completion & Funds Release

Flow
1. Tasker marks work submitted; Poster reviews and clicks **Mark Complete**.

2. Server **captures** the PaymentIntent and instructs Stripe to **split funds**:
  - **Tasker payout** goes to `connectedAccountId`.
  - **Platform fee** (e.g., 5–10%) goes to HamroKam platform account.
3. Stripe schedules **payouts** to tasker's bank account (typically 1–2 business days).

Data stored
- Update Payment status to `released` with Stripe transfer/balance references.
- Update Task to `completed`; enable mutual reviews.

8) Disputes, Cancellations & Refunds (Outline)

Flow
- If job is cancelled before completion: server can **cancel** or **refund** the PaymentIntent (full/partial).
- If dispute after completion: create a Dispute record and pause payout capture if still pending; otherwise process refund/adjustments per policy.
- Maintain an audit trail with timestamps and actor IDs (poster/tasker/admin).

Data stored
- Dispute record with evidence URLs, messages, outcomes, timestamps.

9) Data We Store vs Data We Never Store

We store (in MongoDB)
- User profiles (name, email, phone), password hashes
- Roles/permissions, verification status flags
- Task/Offer/Message documents
- Provider references: `customerId`, `connectedAccountId`, `paymentIntentId`
- Audit logs (who did what, when)

We never store
- Card numbers, CVC, expiry
- Bank BSB/account numbers
- Raw ID documents or selfies
- Plaintext passwords

10) Security Measures (Summary)

- **Transport:** HTTPS only (HSTS), secure cookies for refresh tokens.
- **Auth:** Short-lived access tokens; refresh rotation; optional 2FA for poster/tasker.
- **Input Validation:** Zod/celebrate; sanitize user inputs; strict content-type checks.
- **Rate limiting & WAF:** Prevent brute force and abuse; IP throttling.
- **Secrets:** Managed via env vars; never in repo; rotate periodically.
- **Backups:** Encrypted DB backups; test restore.

- **Monitoring:** Error tracking (Sentry), uptime checks, anomalous activity alerts.
- **Least privilege:** Separate service accounts/keys; restrict CORS origins.

11) Text Sequence Diagrams (Reference)

Signup → Login
User → API: POST /auth/signup (name,email,password)
API → DB: create User (hash password)
API → User: 201 Created
User → API: POST /auth/login (email,password)
API → DB: verify hash
API → User: accessToken + refreshCookie

Post Task → Offer → Accept
Poster → API: POST /tasks
Tasker → API: POST /tasks/:id/offers
Poster → API: POST /offers/:id/accept
API → Stripe: create PaymentIntent (amount)
User (Poster) ↔ Stripe: confirm payment (escrow)
API → DB: Payment status=in_escrow

Complete → Release → Payout
Tasker → API: POST /tasks/:id/submit
Poster → API: POST /tasks/:id/complete
API → Stripe: capture PaymentIntent + split (destination charge)
Stripe → Tasker: payout to bank
Stripe → HamroKam: application fee transfer
API → DB: update Payment status=released