# drop-dispute-handling-spec

# Drop Dispute & Chargeback Handling — Implementation Spec

**Project:** Drop Fintech App **MC Task:** #1190 **Created:** 2026-02-17 **Author:** John (Software Architect Agent) **Status:** DRAFT — Awaiting Alem approval

---

## Executive Summary

This specification defines comprehensive dispute and chargeback handling for Drop's PSD2 pass-through payment system. Drop operates as a PISP (Payment Initiation Service Provider) — we initiate payments from users' bank accounts but never hold customer money.

**Unique challenges for PSD2 PISP:**

- **No merchant account** — Drop doesn't process card payments (no Visa/Mastercard chargebacks)
- **Bank-to-bank transfers** — Disputes go through banking system, not card networks
- **Regulatory framework** — PSD2 requires timely dispute resolution (1 business day for unauthorized, 8 weeks for authorized)
- **Limited control** — Bank makes final decision on refunds, not Drop
- **Trust is everything** — Users trust us with their bank access, disputes must be handled perfectly

**Core principles:**

1. **Transparency** — User always knows status of dispute
2. **Speed** — Respond within SLA (1 day unauthorized, 13 months time limit)
3. **Documentation** — Every dispute fully documented for compliance
4. **Bank coordination** — Work with user's bank and recipient bank
5. **User protection** — Unauthorized transactions refunded immediately (PSD2 requirement)

---

# 1. Regulatory Context (PSD2)

## 1.1 PSD2 Requirements for PISP Disputes

**Source:** Payment Services Directive 2 (EU 2015/2366), Articles 71-74

| Requirement | Value | Enforcement |
|---|---|---|
| **Unauthorized transaction refund** | Within 1 business day | Mandatory — user's bank must refund |
| **Dispute window** | Up to 13 months from transaction date | User can dispute any time within window |
| **Complaint response** | Within 15 business days | PISP must respond with decision |
| **Escalation to external body** | Finansklagenemnda (FinKN) | If user not satisfied with resolution |
| **Documentation retention** | 5 years | Audit trail of all disputes |

**Key differences from card chargebacks:**

- **No chargeback network** (no Visa/Mastercard dispute system)
- **Bank-initiated refunds** — PISP requests refund from user's bank, not from merchant
- **Strong Customer Authentication (SCA)** — If SCA was used, liability shifted to bank (not PISP)
- **No recurring transaction liability** — PISP not liable for subscription fraud if SCA used on initial transaction

## 1.2 Drop's Liability Model

**Unauthorized transactions (fraud):**

- User claims they did NOT authorize the payment
- **Drop's liability:** €0 if SCA (BankID) was used correctly
- **Bank's liability:** Refund user within 1 business day
- **Drop's action:** Facilitate dispute with bank, provide transaction proof (SCA logs)

**Authorized but disputed (service issue):**

- User authorized payment but recipient didn't deliver goods/service
- **Drop's liability:** €0 — commercial dispute between user and recipient
- **Drop's action:** Provide transaction evidence, recommend user contact recipient directly
- **Escalation:** User can contact Finansklagenemnda if unsatisfied

**Technical failure (Drop's fault):**

- Payment failed due to Drop system error (e.g., wrong amount sent)

- **Drop's liability:** 100% — immediate refund + compensation
- **Drop's action:** Refund from Drop's operational account, file incident report

---

# 2. Dispute Types

## 2.1 Classification

| Type | User Claim | Drop's Role | Time Limit | Outcome |
|---|---|---|---|---|
| **Unauthorized** | "I didn't make this payment" | Facilitate bank refund | 13 months | Bank refunds (1 day) |
| **Incorrect amount** | "Wrong amount was sent" | Investigate + refund if Drop error | 13 months | Refund if Drop fault |
| **Duplicate payment** | "Charged twice" | Check idempotency logs | 13 months | Refund duplicate |
| **Service not received** | "Recipient didn't deliver" | Provide evidence only | 60 days (informal) | Commercial dispute |
| **Technical failure** | "Payment stuck/failed but money gone" | Investigate + reconcile | 13 months | Refund if Drop fault |
| **Refund request** | "Recipient agreed to refund" | Facilitate reverse transfer | No limit | Process reverse payment |

## 2.2 Priority Levels

| Priority | Definition | Response SLA | Examples |
|---|---|---|---|
| **Critical** | Unauthorized transaction, large amount (>10,000 NOK) | 4 hours | Fraud, account takeover |
| **High** | Unauthorized <10k NOK, technical failure | 24 hours | Wrong amount sent, duplicate charge |
| **Normal** | Service not received, refund request | 5 business days | Merchant didn't deliver, user wants refund |
| **Low** | Informational, general inquiry | 15 business days | "How do I dispute?", status check |

---

# 3. Database Schema

# 3.1 disputes Table

```
CREATE TABLE IF NOT EXISTS disputes (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  transaction_id TEXT NOT NULL REFERENCES transactions(id),
  dispute_type TEXT NOT NULL CHECK(dispute_type IN (
    'unauthorized',
    'incorrect_amount',
    'duplicate',
    'service_not_received',
    'technical_failure',
    'refund_request'
  )),
  status TEXT DEFAULT 'submitted' CHECK(status IN (
    'submitted',       -- User filed dispute
    'under_review',    -- Drop investigating
    'evidence_requested', -- Need more info from user
    'bank_contacted',  -- Sent to bank (unauthorized cases)
    'resolved_approved', -- Dispute valid, refund issued
    'resolved_denied', -- Dispute invalid, no refund
    'escalated',       -- Sent to FinKN (external complaint)
    'withdrawn'        -- User withdrew dispute
  )),
  priority TEXT DEFAULT 'normal' CHECK(priority IN ('low','normal','high','critical')),

  -- Dispute details
  claimed_amount INTEGER NOT NULL, -- øre (amount user claims is wrong)
  actual_amount INTEGER NOT NULL,  -- øre (actual transaction amount)
  reason TEXT NOT NULL,            -- User's explanation (free text)

  -- Evidence
  evidence_files TEXT,             -- JSON array of file URLs (future)
  user_statement TEXT,             -- Detailed statement from user
  recipient_response TEXT,         -- If recipient contacted

  -- Resolution
  resolution_type TEXT CHECK(resolution_type IN (
    'refund_full',
```

```
    'refund_partial',
    'no_refund',
    'reversed_payment'
  )),
  refund_amount INTEGER,           -- øre (actual refund issued)
  refund_reference TEXT,           -- Bank reference or external_id
  resolution_reason TEXT,          -- Why resolved this way

  -- Timeline
  created_at TEXT DEFAULT (datetime('now')),
  updated_at TEXT DEFAULT (datetime('now')),
  responded_at TEXT,               -- When Drop first responded
  resolved_at TEXT,
  escalated_at TEXT,

  -- Compliance
  sla_deadline TEXT,               -- When response is due (24h for unauthorized)
  breach_sla INTEGER DEFAULT 0,    -- Did we miss deadline?
  external_case_id TEXT            -- FinKN case number if escalated
);

CREATE INDEX IF NOT EXISTS idx_disputes_user ON disputes(user_id);
CREATE INDEX IF NOT EXISTS idx_disputes_transaction ON disputes(transaction_id);
CREATE INDEX IF NOT EXISTS idx_disputes_status ON disputes(status);
CREATE INDEX IF NOT EXISTS idx_disputes_priority ON disputes(priority);
CREATE INDEX IF NOT EXISTS idx_disputes_sla ON disputes(sla_deadline, status);
CREATE INDEX IF NOT EXISTS idx_disputes_created ON disputes(created_at);
```

**PostgreSQL version:**

- Replace `datetime('now')` with `CURRENT_TIMESTAMP`
- No other changes needed

# 3.2 dispute_messages Table

```
CREATE TABLE IF NOT EXISTS dispute_messages (
  id TEXT PRIMARY KEY,
  dispute_id TEXT NOT NULL REFERENCES disputes(id) ON DELETE CASCADE,
  sender_type TEXT NOT NULL CHECK(sender_type IN ('user','admin','system')),
  sender_id TEXT,                  -- user_id or admin_id (NULL for system)
```

```
  message TEXT NOT NULL,
  created_at TEXT DEFAULT (datetime('now')),

  -- Attachments (future)
  attachments TEXT                -- JSON array of file URLs
);

CREATE INDEX IF NOT EXISTS idx_dispute_messages_dispute ON dispute_messages(dispute_id);
CREATE INDEX IF NOT EXISTS idx_dispute_messages_created ON dispute_messages(created_at);
```

**PostgreSQL version:**

- Replace `datetime('now')` with `CURRENT_TIMESTAMP`

# 3.3 dispute_actions Table (Audit Trail)

```
CREATE TABLE IF NOT EXISTS dispute_actions (
  id TEXT PRIMARY KEY,
  dispute_id TEXT NOT NULL REFERENCES disputes(id) ON DELETE CASCADE,
  action_type TEXT NOT NULL,       -- 'status_change', 'evidence_uploaded', 'bank_contacted',
etc.
  performed_by TEXT,               -- user_id or admin_id
  performed_by_type TEXT CHECK(performed_by_type IN ('user','admin','system')),
  details TEXT,                    -- JSON details of action
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE INDEX IF NOT EXISTS idx_dispute_actions_dispute ON dispute_actions(dispute_id);
CREATE INDEX IF NOT EXISTS idx_dispute_actions_type ON dispute_actions(action_type);
CREATE INDEX IF NOT EXISTS idx_dispute_actions_created ON dispute_actions(created_at);
```

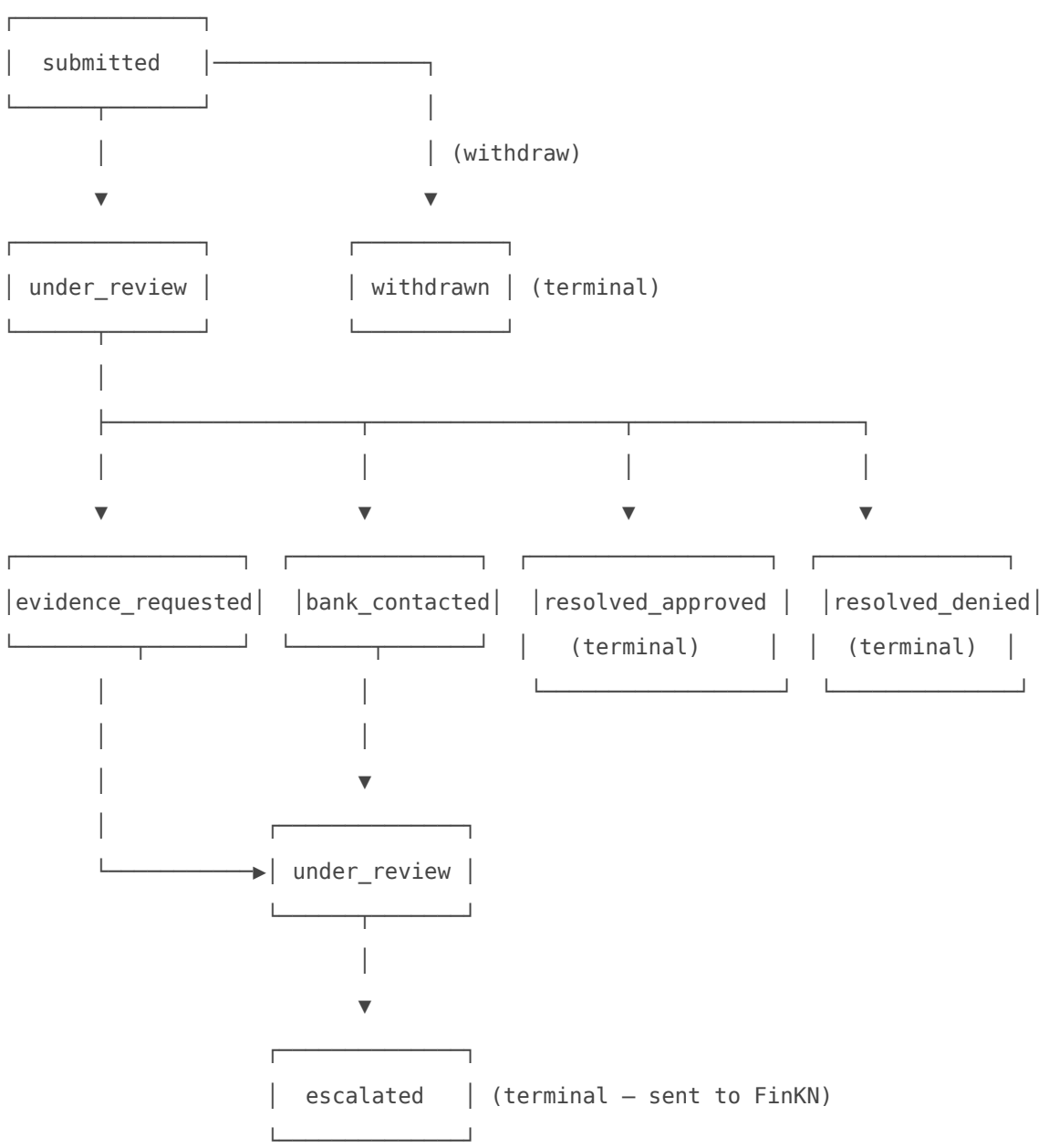
**PostgreSQL version:**

- Replace `datetime('now')` with `CURRENT_TIMESTAMP`

---

# 4. Dispute Lifecycle

## 4.1 State Machine




**Valid transitions:**

```
const VALID_TRANSITIONS = {
  submitted: ["under_review", "withdrawn"],
  under_review: ["evidence_requested", "bank_contacted", "resolved_approved",
"resolved_denied", "escalated"],
  evidence_requested: ["under_review", "withdrawn"],
  bank_contacted: ["under_review", "resolved_approved", "resolved_denied"],
  resolved_approved: [],  // terminal
  resolved_denied: ["escalated"],  // can only escalate after denial
  escalated: [],  // terminal
  withdrawn: [],  // terminal
};
```

# 4.2 SLA Deadlines

| Dispute Type | Response SLA | Resolution SLA |
|---|---|---|
| Unauthorized (critical) | 4 hours | 1 business day (bank) |
| Unauthorized (high) | 24 hours | 1 business day (bank) |
| Technical failure | 24 hours | 5 business days |
| Incorrect amount | 24 hours | 5 business days |
| Duplicate | 24 hours | 5 business days |
| Service not received | 5 business days | No formal SLA |
| Refund request | 5 business days | Depends on recipient |

**SLA calculation:**

```
function calculateSlaDeadline(disputeType: string, priority: string, createdAt: Date): Date {
  const now = new Date(createdAt);

  // Business hours: Mon-Fri 09:00-17:00 CET
  // Skip weekends and Norwegian public holidays

  if (disputeType === 'unauthorized') {
    if (priority === 'critical') {
      return addBusinessHours(now, 4);  // 4 hours
    } else {
      return addBusinessHours(now, 24); // 1 business day
    }
  }

  if (['technical_failure', 'incorrect_amount', 'duplicate'].includes(disputeType)) {
    return addBusinessHours(now, 24);   // 1 business day
  }

  return addBusinessDays(now, 5);       // 5 business days
}
```

---

# 5. API Endpoints

# 5.1 User Endpoints

## POST /api/disputes

Create new dispute.

**Request:**

```
{
  "transactionId": "tx_rem_123",
  "disputeType": "unauthorized",
  "reason": "I did not authorize this payment. My BankID was stolen.",
  "claimedAmount": 50000  // øre (500 NOK)
}
```

**Response (201):**

```
{
  "data": {
    "id": "dsp_abc123",
    "transactionId": "tx_rem_123",
    "disputeType": "unauthorized",
    "status": "submitted",
    "priority": "high",
    "claimedAmount": 50000,
    "createdAt": "2026-02-17T10:30:00Z",
    "slaDeadline": "2026-02-18T10:30:00Z"
  }
}
```

**Validation:**

- Transaction must exist and belong to user (404 if not found)
- Transaction must be completed (can't dispute pending/failed transactions)
- Dispute window: max 13 months since transaction (400 if expired)
- No duplicate dispute for same transaction (409 if exists)
- Reason: 20-2000 chars, sanitized

**Auto-priority logic:**

```
if (disputeType === 'unauthorized' && amount > 1000000) priority = 'critical';  // >10k NOK
else if (disputeType === 'unauthorized') priority = 'high';
```

```
else if (['technical_failure', 'incorrect_amount', 'duplicate'].includes(disputeType))
priority = 'normal';
else priority = 'low';
```

**Side effects:**

- Calculate SLA deadline based on type + priority
- Create initial message in dispute_messages (sender_type=user, message=reason)
- Audit log: `dispute.created`
- If unauthorized → auto-transition to `bank_contacted` + notify bank (see Section 6.1)

---

# GET /api/disputes

List user's disputes.

**Query params:**

- `page` (default: 1)
- `limit` (default: 10, max: 50)
- `status` (optional filter)
- `sort` (created_at_desc, created_at_asc, sla_deadline_asc)

**Response:**

```
{
  "data": [
    {
      "id": "dsp_abc123",
      "transactionId": "tx_rem_123",
      "disputeType": "unauthorized",
      "status": "under_review",
      "priority": "high",
      "claimedAmount": 50000,
      "createdAt": "2026-02-17T10:30:00Z",
      "slaDeadline": "2026-02-18T10:30:00Z",
      "breachSla": false,
      "unreadMessages": 2  // count of admin/system messages since last user visit
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 10,
```

```
    "total": 5,
    "totalPages": 1
  }
}
```

---

## GET /api/disputes/[id]

Get dispute detail + conversation.

**Response:**

```
{
  "data": {
    "dispute": {
      "id": "dsp_abc123",
      "transactionId": "tx_rem_123",
      "disputeType": "unauthorized",
      "status": "bank_contacted",
      "priority": "high",
      "claimedAmount": 50000,
      "actualAmount": 50000,
      "reason": "I did not authorize this payment...",
      "createdAt": "2026-02-17T10:30:00Z",
      "slaDeadline": "2026-02-18T10:30:00Z",
      "breachSla": false
    },
    "transaction": {
      "id": "tx_rem_123",
      "type": "remittance",
      "amount": 50000,
      "currency": "NOK",
      "recipientName": "Mama Jasmina",
      "createdAt": "2026-02-10T14:00:00Z",
      "completedAt": "2026-02-10T14:01:23Z"
    },
    "messages": [
      {
        "id": "msg_1",
        "senderType": "user",
        "message": "I did not authorize this payment...",
```

```
        "createdAt": "2026-02-17T10:30:00Z"
      },
      {
        "id": "msg_2",
        "senderType": "system",
        "message": "We have contacted your bank to initiate a refund. You should receive the refund within 1 business day.",
        "createdAt": "2026-02-17T10:31:00Z"
      }
    ],
    "actions": [
      {
        "id": "act_1",
        "actionType": "status_change",
        "performedBy": "system",
        "details": "{\"from\":\"submitted\",\"to\":\"bank_contacted\"}",
        "createdAt": "2026-02-17T10:31:00Z"
      }
    ]
  }
}
```


**Authorization:**

- User can only view their own disputes (404 if not theirs)

---

# POST /api/disputes/[id]/messages

Add user message (provide additional evidence).

**Request:**


```
{
  "message": "I have contacted my bank and they confirmed my BankID was compromised on Feb 9."
}
```


**Response (201):**


```
{
  "data": {
    "id": "msg_3",
```

```
    "disputeId": "dsp_abc123",
    "senderType": "user",
    "message": "I have contacted my bank...",
    "createdAt": "2026-02-17T12:00:00Z"
  }
}
```

**Side effects:**

- Update dispute.updated_at
- If status was `evidence_requested`, transition to `under_review`
- Audit log: `dispute.message_added`

---

## POST /api/disputes/[id]/withdraw

User withdraws dispute.

**Request:**

```
{
  "reason": "Resolved directly with recipient"
}
```

**Response:**

```
{
  "data": {
    "id": "dsp_abc123",
    "status": "withdrawn",
    "withdrawnAt": "2026-02-17T13:00:00Z"
  }
}
```

**Side effects:**

- Transition to `withdrawn` status (terminal)
- Audit log: `dispute.withdrawn`
- System message: "User withdrew dispute: [reason]"

---

# 5.2 Admin Endpoints

All admin endpoints require `requireAdmin()` middleware.

## GET /api/admin/disputes

List ALL disputes with filters.

**Query params:**

- `page`, `limit`
- `status` (filter)
- `priority` (filter)
- `disputeType` (filter)
- `breachSla` (boolean filter: show only SLA breaches)
- `sort` (created_at_desc, sla_deadline_asc, priority_desc)

**Response:**


```
{
  "data": [
    {
      "id": "dsp_abc123",
      "userId": "usr_demo1",
      "userName": "Amir Hadžić",
      "userEmail": "amir@example.com",
      "transactionId": "tx_rem_123",
      "disputeType": "unauthorized",
      "status": "bank_contacted",
      "priority": "high",
      "claimedAmount": 50000,
      "createdAt": "2026-02-17T10:30:00Z",
      "slaDeadline": "2026-02-18T10:30:00Z",
      "breachSla": false
    }
  ],
  "pagination": { ... },
  "summary": {
    "total": 15,
    "byStatus": {
      "submitted": 3,
      "under_review": 5,
      "bank_contacted": 2,
      "resolved_approved": 4,
```

```
      "resolved_denied": 1
    },
    "breachSla": 0
  }
}
```

---

# PATCH /api/admin/disputes/[id]

Update dispute status or priority.

**Request:**

```
{
  "status": "under_review",
  "priority": "critical",
  "notes": "Escalating due to large amount"
}
```

**Response:**

```
{
  "data": { /* updated dispute */ }
}
```

**Side effects:**

- Update dispute.updated_at
- If status changed to `resolved_approved` or `resolved_denied`, set resolved_at
- Audit log: `dispute.status_changed`
- System message added to dispute_messages

---

# POST /api/admin/disputes/[id]/messages

Admin reply to user.

**Request:**

```
{
  "message": "We have reviewed your case. Your bank has confirmed the refund will be processed within 24 hours.",
  "changeStatus": "resolved_approved"  // optional
}
```

**Response (201):**

```
{
  "data": {
    "id": "msg_4",
    "disputeId": "dsp_abc123",
    "senderType": "admin",
    "message": "We have reviewed your case...",
    "createdAt": "2026-02-17T13:00:00Z"
  }
}
```

**Side effects:**

- Update dispute.updated_at
- If changeStatus provided, update dispute.status
- Set dispute.responded_at (first admin response)
- Audit log: `dispute.admin_reply`
- Email notification to user (subject: "Svar på din dispute #{id}")
- Push notification: "Drop har svart på din dispute"

---

# POST /api/admin/disputes/[id]/resolve

Manually resolve dispute.

**Request:**

```
{
  "resolutionType": "refund_full",
  "refundAmount": 50000,  // øre
  "refundReference": "bank_ref_12345",
  "resolutionReason": "Bank confirmed unauthorized transaction. Refund processed.",
  "status": "resolved_approved"
}
```

**Response:**

```
{
  "data": {
    "id": "dsp_abc123",
    "status": "resolved_approved",
    "resolutionType": "refund_full",
```

```
    "refundAmount": 50000,
    "resolvedAt": "2026-02-17T14:00:00Z"
  }
}
```

**Side effects:**

- Update all resolution fields
- Set resolved_at
- Audit log: `dispute.resolved`
- System message: "Dispute resolved: [resolutionReason]"
- Email + push notification to user

---

# POST /api/admin/disputes/[id]/escalate

Escalate to Finansklagenemnda (FinKN).

**Request:**

```
{
  "reason": "User not satisfied with our decision to deny refund",
  "externalCaseId": "FINKN-2026-12345"  // optional, if already filed
}
```

**Response:**

```
{
  "data": {
    "id": "dsp_abc123",
    "status": "escalated",
    "escalatedAt": "2026-02-17T15:00:00Z",
    "externalCaseId": "FINKN-2026-12345"
  }
}
```

**Side effects:**

- Transition to `escalated` (terminal)
- Set escalated_at + external_case_id
- Audit log: `dispute.escalated`
- Email user with FinKN contact info

---

# 6. Integration with Banking System

## 6.1 Unauthorized Transaction Flow

**Scenario:** User claims they didn't authorize payment (fraud).

**Drop's actions:**

1. **Immediate response** (within 4 hours for critical, 24h for high):
   - Transition dispute to `bank_contacted`
   - Notify user's bank via API (if integrated) or email (if manual)
   - System message: "We have contacted your bank to initiate a refund."
2. **Provide evidence to bank:**
   - Transaction timestamp
   - SCA (BankID) authentication logs
   - IP address, user agent, device ID
   - User's account activity (login history)
   - Any previous disputes from this user
3. **Bank investigation:**
   - Bank verifies SCA was used correctly
   - If SCA valid → bank liable, refunds user within 1 business day
   - If SCA compromised → bank investigates further
4. **Drop receives bank decision:**
   - If refund approved → transition to `resolved_approved`
   - If refund denied → transition to `resolved_denied`
   - Update dispute with bank's resolution_reason

**API integration (future):**

```
// lib/services/bank-dispute.ts
export async function notifyBankOfDispute(params: {
  disputeId: string;
  transactionId: string;
  userId: string;
  bankId: string;
  evidence: {
    scaLogs: string;
    ipAddress: string;
    deviceId: string;
  };
}): Promise<{ caseId: string }> {
  // Call bank's dispute API (e.g., DNB, SpareBank1)
```

```
  // For MVP: send email to bank's fraud department

  const bankEmail = BANK_FRAUD_EMAILS[params.bankId] || "fraud@bank.no";

  await email.send({
    to: bankEmail,
    subject: `Drop PISP Dispute Notification - Transaction ${params.transactionId}`,
    template: "bank-dispute-notification",
    data: {
      disputeId: params.disputeId,
      transactionId: params.transactionId,
      userId: params.userId,
      evidence: params.evidence,
    },
  });

  return { caseId: `DROP-${params.disputeId}` };
}
```

# 6.2 Technical Failure Flow

**Scenario:** Payment failed due to Drop system error (e.g., wrong amount sent).

**Drop's liability:** 100% — immediate refund required.

**Actions:**

1. **Immediate acknowledgment** (within 4 hours):
   - Transition to `under_review`
   - System message: "We are investigating this issue."
2. **Investigation:**
   - Check transaction logs, audit trail
   - Verify actual vs claimed amount
   - Check if idempotency key was reused (duplicate)
   - Review PISP provider logs
3. **If Drop's fault confirmed:**
   - Transition to `resolved_approved`
   - Issue refund from Drop's operational account (not via bank)
   - Resolution type: `refund_full` or `refund_partial`
   - Create incident report in `admin_alerts` table
   - Notify ops team (Slack webhook)
4. **If not Drop's fault:**

- Transition to `resolved_denied`
- Explain to user what happened (e.g., "Bank declined payment, no money was charged")
- Provide evidence (transaction status logs)

**Refund implementation (MVP):**

```
// lib/services/refund.ts
export async function issueRefund(params: {
  disputeId: string;
  transactionId: string;
  amount: number;  // øre
  reason: string;
}): Promise<{ refundId: string; reference: string }> {
  // For MVP: Manual bank transfer
  // Future: Integrate with bank's refund API or PISP reverse payment

  const refundId = randomId("rfnd");

  // Create refund record
  await run(
    `INSERT INTO refunds (id, dispute_id, transaction_id, amount, reason, status, created_at)
     VALUES (?, ?, ?, ?, ?, 'pending', datetime('now'))`,
    [refundId, params.disputeId, params.transactionId, params.amount, params.reason]
  );

  // TODO: Call bank API to initiate reverse PISP payment
  // For MVP: Create admin task to process manual refund
  await createAdminTask({
    type: "manual_refund",
    priority: "high",
    title: `Process refund for dispute ${params.disputeId}`,
    description: `Amount: ${params.amount / 100} NOK\nReason: ${params.reason}`,
    assignee: "finance_team",
  });

  return { refundId, reference: `DROP-REFUND-${refundId}` };
}
```

# 6.3 Service Not Received Flow

**Scenario:** User authorized payment but recipient didn't deliver goods/service.

**Drop's liability:** €0 — commercial dispute between user and recipient.

**Actions:**

1. **Acknowledge** (within 5 business days):
   - Transition to `under_review`
   - System message: "We are reviewing your case."
2. **Provide transaction evidence:**
   - Recipient details (name, bank account, country)
   - Transaction timestamp and amount
   - Payment confirmation from PISP
   - Recommendation: "Contact recipient directly to request refund"
3. **Resolution:**
   - Transition to `resolved_denied`
   - Resolution reason: "This is a commercial dispute between you and the recipient. Drop cannot issue a refund as we do not hold funds. Please contact the recipient directly or seek legal advice."
   - Provide recipient contact info (if available)
   - Inform user of right to escalate to FinKN

**No refund from Drop** — user must resolve with recipient or pursue legal action.

---

# 7. UI Pages

## 7.1 /disputes — Dispute Center (User)

**Layout:**

- Header: "Mine tvister"
- Filter tabs: All | Active | Resolved | Escalated
- Dispute list:
  - Transaction summary (recipient/merchant, amount, date)
  - Dispute type badge
  - Status badge (color-coded)
  - SLA indicator (red if breached)
  - Created date
  - Unread indicator if admin replied

**Empty state:**

- "Ingen tvister"

- "Har du et problem med en betaling? Opprett en tvist"
- CTA button: "Opprett tvist"

---

# 7.2 /disputes/new — Create Dispute (User)

**Step 1: Select Transaction**

- Show user's completed transactions (last 13 months)
- Filter: Remittance | QR Payment
- Each transaction shows:
  - Recipient/merchant name
  - Amount + date
  - Status (completed)
  - "Tvist denne betalingen" button

**Step 2: Dispute Type**

- Radio buttons:
  - "Jeg autoriserte ikke denne betalingen" (unauthorized)
  - "Feil beløp ble sendt" (incorrect_amount)
  - "Jeg ble belastet to ganger" (duplicate)
  - "Jeg mottok ikke tjenesten/produktet" (service_not_received)
  - "Teknisk feil" (technical_failure)
  - "Jeg vil ha refusjon" (refund_request)

**Step 3: Details**

- Reason (textarea, 20-2000 chars, required)
- Claimed amount (prefilled with transaction amount, editable for incorrect_amount)
- Character count indicator
- Upload evidence (future — buttons disabled for MVP)

**Step 4: Review & Submit**

- Summary of dispute
- Transaction details
- Disclaimer (based on type):
  - Unauthorized: "Vi vil kontakte banken din umiddelbart. Du skal få refusjon innen 1 virkedag."
  - Service not received: "Dette er en kommersiell tvist mellom deg og mottakeren. Drop kan ikke refundere, men vi vil gi deg dokumentasjon."
  - Technical failure: "Vi vil undersøke dette og refundere hvis det er vår feil."
- "Send tvist" button

---

# 7.3 /disputes/[id] — Dispute Detail (User)

**Layout:**

- Dispute header:
  - Status badge + priority badge
  - Transaction summary (link to /transactions/[id])
  - Dispute type
  - Amount claimed
  - Created date
  - SLA deadline (if not resolved) — show countdown timer if < 24h
  - SLA breach indicator (if missed deadline)
- Timeline:
  - Initial submission
  - Status changes
  - Admin responses
  - Resolution (if applicable)
- Message thread (chat-style):
  - User messages (left-aligned)
  - Admin messages (right-aligned, green accent)
  - System messages (centered, gray)
  - Timestamps
- Action panel:
  - If status = `evidence_requested`: "Gi mer informasjon" button → message form
  - If status = `resolved_denied`: "Send til Finansklagenemnda" button (see Section 7.4)
  - If status = `submitted` or `under_review`: "Trekk tilbake tvist" button
  - If status = terminal: No actions

---

# 7.4 External Complaint (FinKN)

**Route:** `/disputes/[id]/escalate`

**Content:**

- Heading: "Send klage til Finansklagenemnda"
- Explanation:
  - "Hvis du ikke er fornøyd med vår avgjørelse, kan du sende klagen til Finansklagenemnda (FinKN)."
  - "FinKN er en uavhengig tvisteløsningsinstans for finansielle tjenester."
  - "Drop vil samarbeide fullt ut med FinKN i deres undersøkelse."
- FinKN contact info:
  - **Address:** Postboks 53 Skøyen, 0212 Oslo
  - **Website:** finansklagenemnda.no
  - **Email:** post@finkn.no

    - **Phone:** +47 23 13 19 60
- "Send til FinKN" button:
    - Transitions dispute to `escalated` status
    - Sends email to user with FinKN contact info + case summary
    - Creates admin alert for ops team

---

# 7.5 /admin/disputes — Admin Dashboard

**Layout:**

- Header: "Dispute Management"
- Summary cards:
    - Active disputes (count)
    - SLA breaches (count, red if > 0)
    - Resolved last 7 days (count)
    - Average resolution time
- Filter bar:
    - Status dropdown (all, submitted, under_review, etc.)
    - Priority dropdown (all, critical, high, normal, low)
    - Dispute type dropdown (all, unauthorized, etc.)
    - SLA breach toggle (show only breached)
    - Sort dropdown (Newest, SLA Deadline, Priority)
- Dispute table:
    - ID (clickable)
    - User (name + email)
    - Transaction (ID + amount)
    - Dispute type badge
    - Status badge
    - Priority badge
    - Created
    - SLA deadline (color-coded: green >6h, yellow 1-6h, red <1h or breached)
    - Actions: "View", "Resolve"

---

# 7.6 /admin/disputes/[id] — Admin Detail

**Layout:**

- Dispute header (same as user view)
- User info: email, phone, KYC status, account created date
- Transaction info: type, amount, recipient, bank account, completed date
- Dispute details: reason, claimed amount, evidence
- Status & Priority controls:
    - Status dropdown (editable)

    - Priority dropdown (editable)
    - "Lagre endringer" button
- Message thread (same as user view)
- Admin action panel:
    - Text area for admin reply (10 rows)
    - "Change status to:" dropdown (optional)
    - "Send svar" button
- Resolution panel (if not yet resolved):
    - Resolution type: dropdown (refund_full, refund_partial, no_refund, reversed_payment)
    - Refund amount: input (øre)
    - Refund reference: input (bank ref)
    - Resolution reason: textarea
    - "Resolve Dispute" button
- Escalation panel:
    - "Escalate to FinKN" button
    - External case ID: input (optional)
    - Reason: textarea
- Audit trail (bottom):
    - All actions from dispute_actions table
    - Expandable JSON details

---

# 8. Email Notifications

## 8.1 Dispute Submitted (Auto-Confirmation)

**Subject:** "Tvist opprettet - #{dispute_id}"

**Body (Norwegian):**

```
Hei,

Vi har mottatt din tvist angående transaksjon #{transaction_id}.

Tvisttype: {dispute_type_label}
Beløp: {claimed_amount} NOK
Status: Under behandling

{type_specific_message}

Du kan følge statusen her:
```

```
{NEXT_PUBLIC_APP_URL}/disputes/{dispute_id}

Forventet responstid: {sla_deadline}

Vennlig hilsen,
Drop Kundestøtte
```

**Type-specific messages:**

- **Unauthorized:** "Vi har kontaktet banken din umiddelbart. Du skal motta refusjon innen 1 virkedag."
- **Technical failure:** "Vi undersøker saken og vil svare deg innen 24 timer."
- **Service not received:** "Dette er en kommersiell tvist. Vi vil gi deg dokumentasjon, men kan ikke refundere direkte."

---

# 8.2 Admin Response

**Subject:** "Svar på din tvist #{dispute_id}"

**Body:**

```
Hei,

Vi har svart på din tvist:

{admin_message}

Logg inn på Drop for å se hele samtalen:
{NEXT_PUBLIC_APP_URL}/disputes/{dispute_id}

Hvis du har flere spørsmål, kan du svare direkte i tvisten.

Vennlig hilsen,
Drop Kundestøtte
```

---

# 8.3 Dispute Resolved

**Subject (approved):** "Tvist godkjent - Refusjon behandles" **Subject (denied):** "Tvist avslått - Se forklaring"

**Body (approved):**

```
Hei,

Din tvist har blitt godkjent.

Refusjon: {refund_amount} NOK
Referanse: {refund_reference}
Forklaring: {resolution_reason}

{refund_timeline_message}

Se detaljer:
{NEXT_PUBLIC_APP_URL}/disputes/{dispute_id}

Vennlig hilsen,
Drop Kundestøtte
```

**Body (denied):**

```
Hei,

Din tvist har blitt avslått.

Forklaring: {resolution_reason}

Hvis du ikke er fornøyd med avgjørelsen, kan du sende klagen til Finansklagenemnda (FinKN):
{NEXT_PUBLIC_APP_URL}/disputes/{dispute_id}/escalate

Vennlig hilsen,
Drop Kundestøtte
```

---

# 9. Integration with Existing Systems

## 9.1 Link to Transaction Failure Spec

**Spec:** `/Users/makinja/system/specs/drop-transaction-failure-spec.md`

**Integration points:**

1. **Transaction stuck detection:**
   - If transaction stuck in `processing` or `timeout` for >24h → auto-create dispute with type=`technical_failure`
   - Priority: high
   - Reason: "Transaction failed to complete after 24 hours"
2. **Failed transaction with money deducted:**
   - If transaction status=`failed` BUT bank debited user's account → user can dispute
   - Dispute type: `technical_failure`
   - Drop investigates via reconciliation-worker.ts
3. **Partial failure compensation:**
   - If transaction has `compensation_status=failed` → auto-create dispute
   - Dispute type: `technical_failure`
   - Priority: critical
   - Reason: "Refund failed after partial payment"

**Implementation:**

```
// In reconciliation-worker.ts (from transaction failure spec)
if (tx.status === 'processing' && hoursSinceCreated > 24) {
  await createAutoDispute({
    transactionId: tx.id,
    userId: tx.user_id,
    disputeType: 'technical_failure',
    priority: 'high',
    reason: 'Transaction failed to complete after 24 hours',
    claimedAmount: tx.amount,
  });
}
```

# 9.2 Link to Customer Support Spec

**Spec:** `/Users/makinja/system/specs/drop-customer-support-spec.md`

**Integration points:**

1. **Dispute from support ticket:**
   - If support ticket category=`dispute` → create dispute automatically
   - Copy ticket description to dispute reason
   - Link ticket to dispute (bidirectional)
2. **Support ticket from dispute:**
   - User can open support ticket from dispute detail page

- "Trenger du hjelp?" button → creates ticket with context

3. **Shared message thread:**
   - Admin can view both support tickets and disputes in unified inbox
   - User's conversation history visible to support team

**Implementation:**

```
// In support ticket creation (from customer support spec)
if (ticketCategory === 'dispute') {
  const dispute = await createDispute({
    transactionId: ticketData.transactionId,
    userId: ticketData.userId,
    disputeType: 'service_not_received',  // default, user can change
    reason: ticketData.description,
    claimedAmount: ticketData.amount,
  });

  // Link ticket to dispute
  await run(
    "UPDATE support_tickets SET dispute_id = ? WHERE id = ?",
    [dispute.id, ticketId]
  );
}
```

# 9.3 Link to Complaints System

**Table:** `complaints` (already exists in db.ts)

**Relationship:**

- Disputes are structured (transaction-specific, formal resolution)
- Complaints are unstructured (general feedback, service quality)

**Integration:**

1. **Complaint → Dispute:**
   - If complaint category=`transaction` → suggest creating dispute
   - "Opprett formell tvist" button in complaint detail page
2. **Dispute → Complaint:**
   - If user is unsatisfied with resolved_denied → can file general complaint
   - Complaint linked to original dispute for context

---

# 10. Acceptance Criteria

## 10.1 Database

- [ ] disputes table created with all fields and constraints
- [ ] dispute_messages table created with foreign key to disputes
- [ ] dispute_actions table created for audit trail
- [ ] Indexes created for performance (user_id, transaction_id, status, sla_deadline)
- [ ] Schema works for both SQLite (dev) and PostgreSQL (production)

## 10.2 API Endpoints

- [ ] POST /api/disputes creates dispute + calculates SLA deadline
- [ ] GET /api/disputes returns user's disputes with pagination
- [ ] GET /api/disputes/[id] returns dispute + messages + transaction + actions
- [ ] POST /api/disputes/[id]/messages adds user message
- [ ] POST /api/disputes/[id]/withdraw marks dispute withdrawn
- [ ] GET /api/admin/disputes returns all disputes (admin only)
- [ ] PATCH /api/admin/disputes/[id] updates status/priority (admin only)
- [ ] POST /api/admin/disputes/[id]/messages adds admin reply + sends email
- [ ] POST /api/admin/disputes/[id]/resolve resolves dispute with refund details
- [ ] POST /api/admin/disputes/[id]/escalate escalates to FinKN

## 10.3 Authorization

- [ ] Users can only view their own disputes (404 on unauthorized access)
- [ ] Admin endpoints require admin role (403 if not admin)
- [ ] Transaction ownership validated (can't dispute someone else's transaction)
- [ ] CSRF protection via validateOrigin() middleware

## 10.4 SLA Management

- [ ] SLA deadline calculated correctly based on dispute type + priority
- [ ] Business hours calculation (Mon-Fri 09:00-17:00, skip weekends + holidays)

- [ ] SLA breach flag set if deadline missed
- [ ] Admin dashboard shows SLA breaches prominently

# 10.5 UI Pages

- [ ] /disputes shows user's disputes with status filter
- [ ] /disputes/new shows multi-step dispute creation form
- [ ] /disputes/[id] shows conversation thread + action buttons
- [ ] /disputes/[id]/escalate shows FinKN contact info + escalation button
- [ ] /admin/disputes shows all disputes with filters (status, priority, type, SLA breach)
- [ ] /admin/disputes/[id] shows detail + admin action panel

# 10.6 Integration

- [ ] All dispute actions logged to audit_log table
- [ ] Disputes linked to transactions (bidirectional)
- [ ] Transaction detail page shows dispute status (if exists)
- [ ] Email notifications sent for dispute events (submitted, admin reply, resolved)
- [ ] Push notifications sent for status changes
- [ ] Auto-dispute created for stuck transactions (>24h)
- [ ] Support tickets can create disputes (if category=dispute)

# 10.7 Validation

- [ ] Transaction must be completed (can't dispute pending/failed)
- [ ] Dispute window: max 13 months since transaction
- [ ] No duplicate dispute for same transaction
- [ ] Reason: 20-2000 chars, sanitized
- [ ] Status transitions validated (finite state machine)

# 10.8 Edge Cases

- [ ] User can't create multiple disputes for same transaction
- [ ] Resolved disputes can't be reopened (only escalated if denied)
- [ ] Withdrawn disputes are final (can't un-withdraw)

- [ ] SLA deadline doesn't count weekends or Norwegian public holidays
- [ ] Email notifications handle missing user email gracefully

---

# 11. Implementation Order

## Phase 1: Database + Types (Day 1)

1. Add schemas to `src/lib/db.ts` (SQLite + PostgreSQL versions)
2. Create `src/types/dispute.ts`
3. Create `src/lib/dispute-utils.ts` (SLA calculation, status validation)
4. Add `requireAdmin()` to `src/lib/middleware.ts` (if not exists)

## Phase 2: User API (Day 1-2)

1. POST /api/disputes (create)
2. GET /api/disputes (list)
3. GET /api/disputes/[id] (detail)
4. POST /api/disputes/[id]/messages (add message)
5. POST /api/disputes/[id]/withdraw (withdraw)

## Phase 3: User UI (Day 2-3)

1. /disputes (list page)
2. /disputes/new (multi-step creation form)
3. /disputes/[id] (conversation view)
4. /disputes/[id]/escalate (FinKN escalation page)
5. Components: dispute-card, dispute-badge, message-bubble, sla-indicator

## Phase 4: Admin API (Day 3)

1. GET /api/admin/disputes (list all)
2. PATCH /api/admin/disputes/[id] (update status/priority)
3. POST /api/admin/disputes/[id]/messages (admin reply)
4. POST /api/admin/disputes/[id]/resolve (manual resolution)
5. POST /api/admin/disputes/[id]/escalate (escalate to FinKN)

## Phase 5: Admin UI (Day 4)

1. /admin/disputes (dashboard with filters)
2. /admin/disputes/[id] (detail + action panel)

# Phase 6: Integration (Day 4-5)

1. Audit logging for all dispute actions
2. Email notifications (submitted, admin reply, resolved)
3. Push notifications for status changes
4. Link to transaction failure spec (auto-dispute for stuck transactions)
5. Link to support ticket spec (dispute from ticket)

# Phase 7: Bank Integration (Day 5)

1. Unauthorized transaction flow (notify bank, provide evidence)
2. Technical failure flow (refund from Drop operational account)
3. Service not received flow (provide documentation, no refund)

# Phase 8: Testing + Refinement (Day 5-6)

1. Manual testing of all flows (user + admin)
2. Edge case validation
3. UI polish (spacing, colors, responsive)
4. SLA calculation accuracy test
5. Email template review

---

# 12. Dependencies

## 12.1 Existing Infrastructure

- Database: `src/lib/db.ts` (SQLite/PostgreSQL dual driver)
- Auth: `src/lib/auth.ts` (getCurrentUser, JWT validation)
- Middleware: `src/lib/middleware.ts` (requireAuth, sanitizeText, auditLog)
- Utils: `src/lib/utils-server.ts` (randomId)
- Email: Existing email service (to be determined)
- Push notifications: FCM (Firebase Cloud Messaging) or APNS

## 12.2 New Dependencies

None. All features use existing infrastructure.

## 12.3 External Services (Future)

- **Bank dispute API** — For automated refund requests (unauthorized transactions)
- **FinKN API** — For automated case escalation (if available)
- **SMS notifications** — For critical dispute updates (optional)

---

# 13. Testing Checklist

## 13.1 User Flows

1. **Create dispute (unauthorized):**
   - Select transaction
   - Choose "unauthorized" type
   - Submit reason
   - Verify auto-transition to `bank_contacted`
   - Verify email sent with SLA deadline
2. **Create dispute (service not received):**
   - Select transaction
   - Choose "service not received" type
   - Submit reason
   - Verify status = `submitted`
   - Verify email sent
3. **Add message to dispute:**
   - Open dispute detail
   - Add message
   - Verify status changes to `under_review` (if was `evidence_requested`)
4. **Withdraw dispute:**
   - Open dispute detail
   - Click "Trekk tilbake"
   - Confirm
   - Verify status = `withdrawn` (terminal)
5. **Escalate to FinKN:**
   - Open resolved_denied dispute
   - Click "Send til FinKN"
   - Verify status = `escalated`
   - Verify email with FinKN contact info

## 13.2 Admin Flows

1. **View all disputes dashboard:**
   - Filter by status, priority, type

- Sort by SLA deadline
- Verify SLA breach indicator

2. **Reply to dispute:**
   - Open dispute detail
   - Add admin message
   - Change status to `under_review`
   - Verify email sent to user
3. **Resolve dispute (approved):**
   - Open dispute detail
   - Fill resolution panel (refund_full, amount, reference, reason)
   - Click "Resolve Dispute"
   - Verify status = `resolved_approved`
   - Verify email sent to user
4. **Resolve dispute (denied):**
   - Open dispute detail
   - Fill resolution panel (no_refund, reason)
   - Click "Resolve Dispute"
   - Verify status = `resolved_denied`
   - Verify email sent to user with FinKN escalation option
5. **Escalate to FinKN (admin):**
   - Open dispute detail
   - Click "Escalate to FinKN"
   - Enter external case ID + reason
   - Verify status = `escalated`

# 13.3 Edge Cases

1. **Duplicate dispute:**
   - Try to create second dispute for same transaction
   - Verify 409 Conflict error
2. **Expired dispute window:**
   - Try to create dispute for transaction >13 months old
   - Verify 400 Bad Request error
3. **SLA deadline calculation:**
   - Create dispute on Friday 16:00
   - Verify SLA deadline is Monday 10:00 (skip weekend)
4. **Authorization:**
   - User A tries to view User B's dispute
   - Verify 404 Not Found
5. **Status transition validation:**
   - Try to transition from `resolved_approved` to `under_review`
   - Verify 400 Bad Request error

---

# 14. Future Enhancements (Out of Scope)

1. **File attachments** — Allow users to upload evidence (screenshots, receipts)
2. **Video evidence** — Record screen for fraud proof
3. **Multi-language support** — English, Bosnian translations
4. **AI dispute classification** — Auto-detect dispute type from user's description
5. **Automated refund triggers** — For specific patterns (e.g., duplicate transactions)
6. **Bank API integration** — Direct API calls instead of email for unauthorized disputes
7. **FinKN API integration** — Automated case filing
8. **Dispute templates** — Pre-filled forms for common issues
9. **Internal notes** — Admin-only notes not visible to user
10. **Dispute analytics** — Dashboard showing dispute trends, resolution rates, SLA performance

---

# 15. Compliance Notes

## 15.1 PSD2 Article 71 (Unauthorized Transactions)

**User's rights:**

- Report unauthorized transaction within 13 months
- Receive refund within 1 business day (from bank, not Drop)
- No liability if SCA (BankID) was compromised through no fault of user

**Drop's obligations:**

- Notify user's bank immediately (within 24 hours)
- Provide transaction evidence (SCA logs, IP, device ID)
- Keep audit trail for 5 years
- Do NOT delay refund process

**Liability shift:**

- If SCA was performed correctly → bank liable (not Drop)
- If SCA was not performed → Drop liable (refund from operational account)
- If user was grossly negligent (shared BankID) → user liable (no refund)

## 15.2 PSD2 Article 74 (Complaint Handling)

**Requirements:**

- Respond to complaint within 15 business days
- Provide clear explanation of decision
- Inform user of right to escalate to FinKN
- FinKN contact info must be easily accessible

**Drop's implementation:**

- SLA: 5 business days for normal, 1 business day for unauthorized
- Email notification with resolution reason
- Escalation button in UI after `resolved_denied`
- FinKN contact info on `/disputes/[id]/escalate` page

## 15.3 GDPR Compliance

**Data retention:**

- Dispute records: 5 years (regulatory requirement)
- User messages: 5 years
- Audit trail: 5 years
- After 5 years: Archive to cold storage or delete (per GDPR)

**User rights:**

- Right to access: User can view all disputes and messages
- Right to rectification: User can add messages to correct information
- Right to erasure: Limited (regulatory retention overrides)
- Right to data portability: User can export dispute data (future)

**Data minimization:**

- Only collect necessary information (reason, amount, transaction ID)
- No excessive evidence requests
- File attachments limited to 5MB each (future)

---

# 16. Monitoring & Alerting

## 16.1 Metrics to Track

| Metric | Threshold | Alert If |
|---|---|---|
| Active disputes (count) | 10 | > 50 |
| SLA breaches (count) | 0 | > 0 |
| Average resolution time (days) | 3 | > 7 |
| Unauthorized disputes (%) | 5% | > 15% |
| Dispute approval rate (%) | 70% | < 50% |
| Escalations to FinKN (count) | 1/month | > 5/month |

# 16.2 Dashboard Queries

**Active disputes:**

```
SELECT COUNT(*) FROM disputes
WHERE status NOT IN ('resolved_approved', 'resolved_denied', 'escalated', 'withdrawn');
```

**SLA breaches:**

```
SELECT COUNT(*) FROM disputes
WHERE breach_sla = 1
  AND status NOT IN ('resolved_approved', 'resolved_denied', 'escalated', 'withdrawn');
```

**Average resolution time:**

```
SELECT AVG(julianday(resolved_at) - julianday(created_at)) AS days
FROM disputes
WHERE resolved_at IS NOT NULL
  AND resolved_at > datetime('now', '-30 days');
```

# 16.3 Slack Alerts

**When to send:**

1. **SLA breach** (any dispute misses deadline)
   - Channel: #ops
   - Priority: high
   - Message: "Dispute #{id} missed SLA deadline (type: {type}, priority: {priority}, user: {email})"
2. **Critical unauthorized dispute** (>10k NOK)
   - Channel: #fraud
   - Priority: critical

- Message: "High-value unauthorized dispute created: #{id} ({amount} NOK, user: {email})"

3. **Escalation to FinKN**
   - Channel: #ops
   - Priority: normal
   - Message: "Dispute #{id} escalated to FinKN by {user_email}. Case ID: {external_case_id}"

---

# 17. Open Questions (For Alem)

## Q1: Refund Implementation

**Question:** How should we handle refunds for technical failures?

**Options:**

- Option A: Manual bank transfer (MVP) — Admin processes refund via bank UI
- Option B: PISP reverse payment (future) — Integrate with bank API for automatic refund
- Option C: Drop holds refund balance (NOT ALLOWED — breaks pass-through model)

**Recommendation:** Option A for MVP, migrate to Option B when bank APIs available.

---

## Q2: Admin Role

**Question:** Who should have admin access to dispute dashboard?

**Options:**

- Option A: CEO (Alem) only
- Option B: CEO + finance team
- Option C: CEO + finance + support team

**Recommendation:** Option C — support team needs access to respond quickly.

---

## Q3: FinKN Escalation Process

**Question:** Should we automate FinKN escalation or keep it manual?

**Options:**

- Option A: Manual (user clicks button, we send email)
- Option B: Semi-automated (user clicks button, we pre-fill FinKN web form)
- Option C: Fully automated (API integration if available)

**Recommendation:** Option A for MVP. Check if FinKN has API.

---

# Q4: Dispute Notification Channels

**Question:** Email + push notifications both? Or only one?

**Options:**

- Option A: Email only (simpler, no push infra needed)
- Option B: Push only (faster, modern)
- Option C: Both (redundancy, user preference)

**Recommendation:** Option C. Email is fallback if user disabled push.

---

# Q5: Dispute Evidence (Future)

**Question:** Should we allow file uploads (screenshots, receipts)?

**Options:**

- Option A: No (text only, simpler MVP)
- Option B: Yes (better evidence, but needs file storage + moderation)

**Recommendation:** Option A for MVP. Add file uploads in Phase 2 when we have S3/Cloudflare R2 storage.

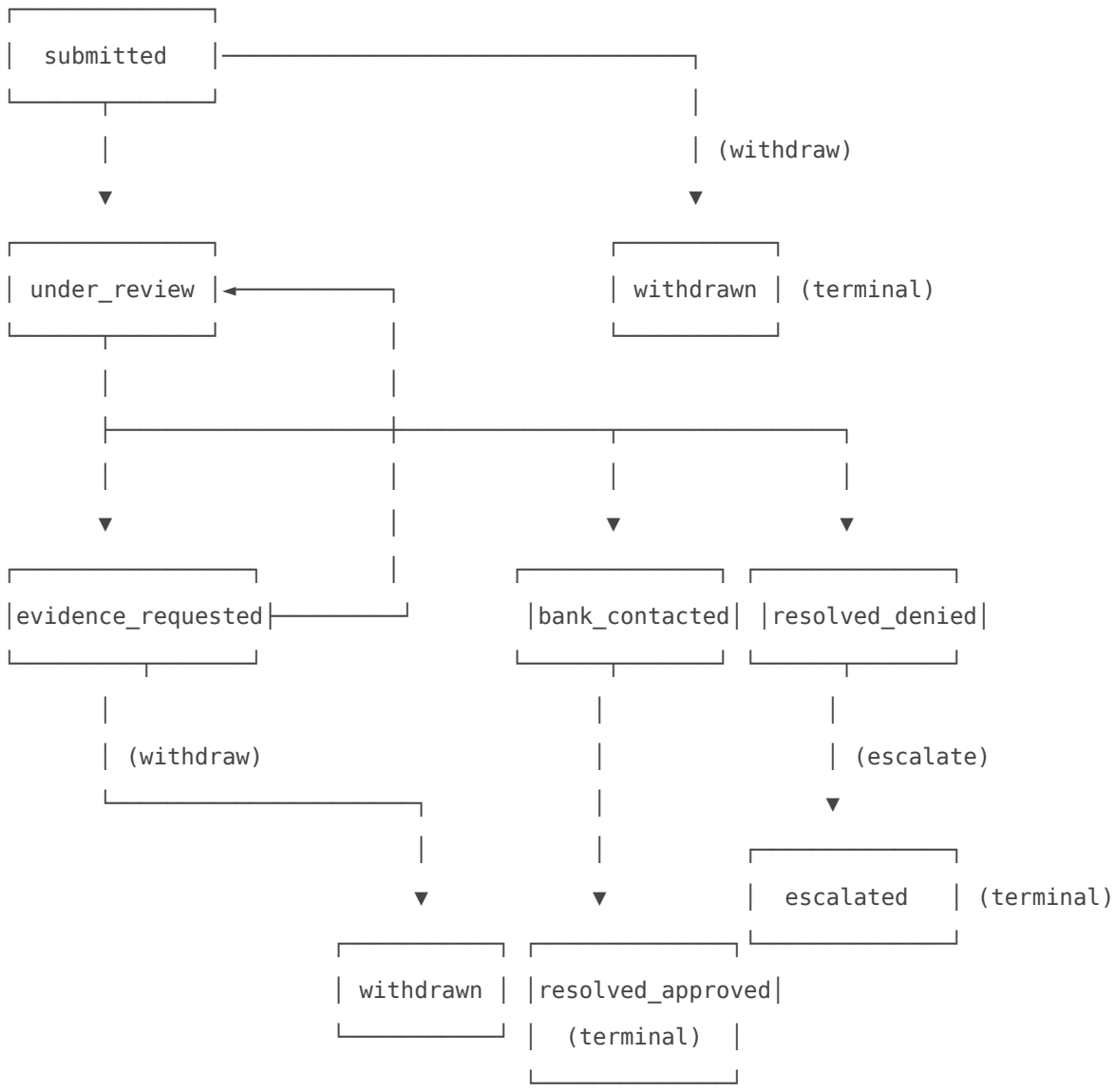
---

# 18. Next Steps

1. **Review this spec** with Alem
2. **Approve/reject sections** (or request changes)
3. **Answer open questions** (Q1-Q5)
4. **Prioritize phases** (which to implement first?)
5. **Assign to builder agent** (one phase at a time)
6. **Validation after each phase** (validator agent checks implementation)

**Estimated timeline:** 6 days for Phases 1-6, Phase 7-8 can run in parallel.

# Appendix A: State Transition Diagram (ASCII)




# Appendix B: Norwegian Translations

| English | Norwegian | Context |
|---|---|---|
| Dispute | Tvist | Formal complaint |
| Claim | Krav | Amount claimed |
| Unauthorized | Uautorisert | Fraud |

| English | Norwegian | Context |
|---|---|---|
| Chargeback | Tilbakeføring | Refund |
| Resolution | Avgjørelse | Final decision |
| Escalate | Eskalere | Send to FinKN |
| Evidence | Dokumentasjon | Proof |
| Refund | Refusjon | Money back |
| Complaint | Klage | General issue |

---

# Appendix C: PSD2 Regulatory Sources

**Primary sources:**

1. PSD2: Impacts and Compliance for Merchants
2. What is PSD2 everything to know for compliance - Adyen
3. The Payment Services Contract: PSD2 Requirements and PSD3 Perspectives - ILP Abogados
4. What is PSD2? How it Impacts Banks, Businesses & Chargebacks911
5. How European merchants can reduce chargebacks and protect revenue in 2026 | GR4VY

**Regulatory bodies:**

- **Finanstilsynet** (Norway) — Financial regulator
- **Finansklagenemnda (FinKN)** — External dispute resolution

---

**END OF SPEC**

---

Revision #4
Created 2026-02-18 08:44:44 UTC by John
Updated 2026-06-07 20:00:21 UTC by John