drop-customer-support-spec

Drop Customer Support System — Implementation Spec

Project: Drop Fintech App **MC Task:** #1187 **Created:** 2026-02-17 **Architecture:** Software Architect Agent

---

1. Overview

This spec defines a built-in customer support ticket system for Drop. No external dependencies (no Zendesk, no Intercom). MVP scope.

Scope:

- Support ticket creation (user-initiated)
- Ticket conversation thread (user + admin messages)
- Admin ticket management dashboard
- Integration with audit logging
- Email notifications when admin responds
- Link tickets to transactions when relevant

Non-goals (future):

- Live chat
- Chatbot / AI responses
- Multi-language support in UI (Norwegian only for MVP)
- File attachments (future enhancement)
- Public knowledge base / FAQ system

---

2. Database Schema

2.1 support_tickets Table

```
CREATE TABLE IF NOT EXISTS support_tickets (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  subject TEXT NOT NULL,
  description TEXT NOT NULL,
  category TEXT NOT NULL CHECK(category IN
('transaction_issue','account_access','verification','general','dispute')),
  status TEXT DEFAULT 'open' CHECK(status IN
('open','in_progress','waiting_user','resolved','closed')),
  priority TEXT DEFAULT 'normal' CHECK(priority IN ('low','normal','high','urgent')),
  transaction_id TEXT REFERENCES transactions(id), -- optional link to transaction
  created_at TEXT DEFAULT (datetime('now')),
  updated_at TEXT DEFAULT (datetime('now')),
  resolved_at TEXT,
  closed_at TEXT
);

CREATE INDEX IF NOT EXISTS idx_support_tickets_user ON support_tickets(user_id);
CREATE INDEX IF NOT EXISTS idx_support_tickets_status ON support_tickets(status);
CREATE INDEX IF NOT EXISTS idx_support_tickets_category ON support_tickets(category);
CREATE INDEX IF NOT EXISTS idx_support_tickets_created ON support_tickets(created_at);
CREATE INDEX IF NOT EXISTS idx_support_tickets_transaction ON support_tickets(transaction_id);
```

PostgreSQL version:

- Replace `datetime('now')` with `CURRENT_TIMESTAMP`
- No other changes needed (TEXT and CHECK constraints work in both)

2.2 ticket_messages Table

```
CREATE TABLE IF NOT EXISTS ticket_messages (
  id TEXT PRIMARY KEY,
  ticket_id TEXT NOT NULL REFERENCES support_tickets(id) ON DELETE CASCADE,
  sender_type TEXT NOT NULL CHECK(sender_type IN ('user','admin')),
  sender_id TEXT, -- user_id or admin user_id (optional, for audit)
  message TEXT NOT NULL,
  created_at TEXT DEFAULT (datetime('now'))
);
```

```
CREATE INDEX IF NOT EXISTS idx_ticket_messages_ticket ON ticket_messages(ticket_id);
CREATE INDEX IF NOT EXISTS idx_ticket_messages_created ON ticket_messages(created_at);
```

PostgreSQL version:

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

---

3. Status Flow

```
open → in_progress → waiting_user → resolved → closed
  ↓          ↓              ↓              ↓
  └──────────┴──────────────┴──────────────┘

          (can reopen if needed)
```

Status definitions:

- **open** — newly created, not assigned yet
- **in_progress** — admin is working on it
- **waiting_user** — admin replied, waiting for user response
- **resolved** — issue resolved, pending user confirmation
- **closed** — ticket closed (auto-close after 7 days resolved, or manual)

Priority:

- **urgent** — transaction blocked, account locked
- **high** — verification failed, payment issue
- **normal** — general inquiry, feature request
- **low** — informational

---

4. API Endpoints

4.1 User Endpoints

POST /api/support/tickets

Create new support ticket.

Request:

```
{
  "subject": "Transaction failed but money was deducted",
  "description": "I tried to send 500 NOK to Serbia but...",
  "category": "transaction_issue",
  "transaction_id": "tx_rem_123" // optional
}
```

Response (201):

```
{
  "data": {
    "id": "tkt_abc123",
    "subject": "...",
    "description": "...",
    "category": "transaction_issue",
    "status": "open",
    "priority": "normal",
    "created_at": "2026-02-17T10:30:00Z"
  }
}
```

Validation:

- subject: required, 5-200 chars, sanitized
- description: required, 20-2000 chars, sanitized
- category: must be one of enum values
- transaction_id: optional, must exist if provided

Auto-priority logic:

- category=dispute → priority=high
- category=account_access → priority=high
- category=transaction_issue → priority=normal
- category=verification → priority=normal
- category=general → priority=low

Side effects:

- Audit log: `support.ticket_created`
- Initial message in ticket_messages (sender_type=user, message=description)

GET /api/support/tickets

List user's tickets.

Query params:

- `page` (default: 1)
- `limit` (default: 10, max: 50)
- `status` (optional filter: open, in_progress, waiting_user, resolved, closed)

Response:

```
{
  "data": [
    {
      "id": "tkt_abc123",
      "subject": "Transaction failed...",
      "category": "transaction_issue",
      "status": "in_progress",
      "priority": "normal",
      "created_at": "2026-02-17T10:30:00Z",
      "updated_at": "2026-02-17T11:00:00Z",
      "unread_messages": 2 // count of admin messages since last user visit
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 5,
    "totalPages": 1
  }
}
```

GET /api/support/tickets/[id]

Get ticket detail + conversation thread.

Response:

```
{
  "data": {
    "ticket": {
      "id": "tkt_abc123",
```

```
      "subject": "Transaction failed...",
      "description": "I tried to send...",
      "category": "transaction_issue",
      "status": "in_progress",
      "priority": "normal",
      "transaction_id": "tx_rem_123",
      "created_at": "2026-02-17T10:30:00Z",
      "updated_at": "2026-02-17T11:00:00Z"
    },
    "messages": [
      {
        "id": "msg_1",
        "sender_type": "user",
        "message": "I tried to send...",
        "created_at": "2026-02-17T10:30:00Z"
      },
      {
        "id": "msg_2",
        "sender_type": "admin",
        "message": "Thank you for reporting. We are investigating...",
        "created_at": "2026-02-17T11:00:00Z"
      }
    ],
    "transaction": { /* if transaction_id is set */ }
  }
}
```

Authorization:

- User can only view their own tickets
- Return 404 if ticket belongs to another user

POST /api/support/tickets/[id]/messages

Add user message to conversation.

Request:

```
{
  "message": "I tried again and it worked this time"
}
```

Response (201):

```
{
  "data": {
    "id": "msg_3",
    "ticket_id": "tkt_abc123",
    "sender_type": "user",
    "message": "I tried again...",
    "created_at": "2026-02-17T12:00:00Z"
  }
}
```

Side effects:

- Update ticket.updated_at
- If status was 'waiting_user' or 'resolved', change to 'open'
- Audit log: `support.message_added`

4.2 Admin Endpoints

All admin endpoints require `requireAdmin()` middleware (TBD: define admin role or separate admin auth).

GET /api/admin/support/tickets

List ALL tickets with filters.

Query params:

- `page`, `limit`
- `status` (filter)
- `priority` (filter)
- `category` (filter)
- `sort` (created_at_desc, created_at_asc, updated_at_desc, priority_desc)

Response:

```
{
  "data": [
    {
      "id": "tkt_abc123",
      "user_id": "usr_demo1",
```

```
      "user_email": "amir@example.com",
      "subject": "Transaction failed...",
      "category": "transaction_issue",
      "status": "open",
      "priority": "normal",
      "created_at": "2026-02-17T10:30:00Z",
      "updated_at": "2026-02-17T11:00:00Z"
    }
  ],
  "pagination": { ... }
}
```

PATCH /api/admin/support/tickets/[id]

Update ticket status, priority, or admin notes.

Request:

```
{
  "status": "in_progress",
  "priority": "high"
}
```

Response:

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

Side effects:

- Update ticket.updated_at
- If status changed to 'resolved', set resolved_at
- If status changed to 'closed', set closed_at
- Audit log: `support.ticket_updated`

POST /api/admin/support/tickets/[id]/messages

Admin reply to ticket.

Request:

```
{
  "message": "Thank you for reporting. We have issued a refund.",
  "change_status": "resolved" // optional
}
```

Response (201):

```
{
  "data": {
    "id": "msg_4",
    "ticket_id": "tkt_abc123",
    "sender_type": "admin",
    "message": "Thank you for reporting...",
    "created_at": "2026-02-17T13:00:00Z"
  }
}
```

Side effects:

- Update ticket.updated_at
- If change_status provided, update ticket.status
- Change status to 'waiting_user' if not explicitly set
- Audit log: `support.admin_reply`
- Email notification to user (see section 5)

---

5. Email Notifications

When admin replies, send email to user:

Subject: "Svar på din support-henvendelse [#{ticket_id}]"

Body (plain text):

```
Hei,

Vi har svart på din support-henvendelse:

Emne: {ticket.subject}
Svar: {admin_message}
```

```
Logg inn på Drop for å se hele samtalen:
{NEXT_PUBLIC_APP_URL}/support/tickets/{ticket_id}

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

Vennlig hilsen,
Drop Support
```

Implementation:

- Use existing MCP email integration pattern (see ~/system/tools/manifest.md)
- No need to implement email in MVP — create draft and leave as TODO for now
- Future: use `email-to-ticket.js` pattern from ALAI tools

---

6. UI Pages

6.1 /support — Help Center (User)

Layout:

- Header: "Trenger du hjelp?"
- FAQ section (static, hardcoded for MVP):
  - "Hvordan sender jeg penger?"
  - "Hvorfor ble betalingen min avvist?"
  - "Hvordan verifiserer jeg kontoen min?"
  - "Hva er gebyrene?"
  - Each FAQ expands to show answer (accordion)
- CTA button: "Opprett support-billett"

Design:

- Match existing Drop UI (see mockups/figma-make-export/)
- Green accent color (#00E5A0)
- Dark background (#09090b)

6.2 /support/tickets — Ticket List (User)

Layout:

- Header: "Mine support-billetter"

- Filter tabs: All | Open | Resolved | Closed
- Ticket list:
  - Subject (clickable)
  - Category badge
  - Status badge (color-coded)
  - Created date
  - Unread indicator if admin replied

Empty state:

- "Ingen billetter ennå"
- "Opprett billett" button

6.3 /support/tickets/[id] — Conversation (User)

Layout:

- Ticket header:
  - Subject
  - Status badge
  - Category
  - Created date
  - Transaction link (if transaction_id set): "Relatert transaksjon: [link]"
- Message thread (chat-style):
  - User messages (left-aligned, gray background)
  - Admin messages (right-aligned, green accent)
  - Timestamps
- Reply form (if status != closed):
  - Text area (5 rows)
  - "Send svar" button
  - Disabled if status=closed with message "Denne billetten er lukket"

6.4 /support/new — Create Ticket (User)

Form fields:

- Subject (text input, required)
- Category (dropdown, required)
  - "Transaksjonsproblem"
  - "Tilgangsproblem"
  - "Verifisering"
  - "Generelt"
  - "Tvist"

- Transaction (optional, dropdown from user's transactions):
  - "Velg transaksjon (valgfritt)"
  - Show recent 10 transactions
- Description (textarea, 5 rows, required)
- "Opprett billett" button

Validation:

- Subject: 5-200 chars
- Description: 20-2000 chars
- Show character count below textarea

Success:

- Redirect to /support/tickets/[new_id]
- Toast: "Billett opprettet. Vi svarer så snart som mulig."

6.5 /admin/support — Admin Dashboard

Layout:

- Header: "Support-billetter"
- Filter bar:
  - Status dropdown (all, open, in_progress, waiting_user, resolved, closed)
  - Priority dropdown (all, urgent, high, normal, low)
  - Category dropdown (all, transaction_issue, account_access, verification, general, dispute)
  - Sort dropdown (Nyeste, Eldste, Oppdatert sist, Prioritet)
- Ticket table:
  - ID (clickable)
  - User (email)
  - Subject
  - Category badge
  - Status badge
  - Priority badge
  - Created
  - Updated

Click ticket:

- Navigate to /admin/support/[id]

6.6 /admin/support/[id] — Admin Ticket Detail

Layout:

- Ticket header:
  - Subject
  - User info: email, phone, KYC status
  - Status dropdown (editable)
  - Priority dropdown (editable)
  - "Lagre endringer" button
- Transaction link (if set): "Se transaksjon" button
- Message thread (same as user view)
- Admin reply form:
  - Text area (10 rows)
  - "Endre status til:" dropdown (optional)
  - "Send svar" button

Audit trail (bottom):

- Show ticket_messages with timestamps
- Show status changes from audit_log

---

7. Integration Points

7.1 Audit Logging

All support actions logged to `audit_log` table:

Action	Resource Type	Details
support.ticket_created	support_ticket	ticket_id, category, priority
support.message_added	ticket_message	ticket_id, sender_type
support.admin_reply	ticket_message	ticket_id, admin message preview
support.ticket_updated	support_ticket	ticket_id, old_status, new_status

Implementation: Use `auditLog()` helper from `@/lib/middleware`.

7.2 Transaction Linking

When user creates ticket with transaction_id:

- Validate transaction exists and belongs to user

- Display transaction summary in ticket detail:
  - Type (remittance / qr_payment)
  - Amount
  - Recipient / Merchant
  - Status
  - Created date
  - Link to /transactions/[id]

7.3 Email Notifications (Future)

Placeholder implementation:

- Create `src/lib/email-support.ts`:

```
export async function notifyUserTicketReply(ticketId: string, userId: string,
adminMessage: string) {
  // TODO: Integrate with MCP email (see ~/system/tools/manifest.md)
  console.log(`[Email] Notify user ${userId} about ticket ${ticketId}`);
}
```

- Call from admin reply endpoint
- Move to MCP email when ready

---

8. File List

8.1 Database Migration

File: `src/lib/db.ts` **Changes:**

- Add `support_tickets` and `ticket_messages` schemas to `SQLITE_SCHEMA` constant
- Add same schemas to `PG_SCHEMA` constant (replace `datetime('now')` with `CURRENT_TIMESTAMP`)
- No other changes needed

8.2 API Routes

User Endpoints

1. `src/app/api/support/tickets/route.ts` — GET (list), POST (create)

2. `src/app/api/support/tickets/[id]/route.ts` — GET (detail)
3. `src/app/api/support/tickets/[id]/messages/route.ts` — POST (add message)

Admin Endpoints

4. `src/app/api/admin/support/tickets/route.ts` — GET (list all)
5. `src/app/api/admin/support/tickets/[id]/route.ts` — PATCH (update status/priority)
6. `src/app/api/admin/support/tickets/[id]/messages/route.ts` — POST (admin reply)

8.3 UI Pages

User Pages

1. `src/app/support/page.tsx` — Help center (FAQ + create button)
2. `src/app/support/tickets/page.tsx` — Ticket list
3. `src/app/support/tickets/[id]/page.tsx` — Conversation view
4. `src/app/support/new/page.tsx` — Create ticket form

Admin Pages

5. `src/app/admin/support/page.tsx` — Admin dashboard (ticket table)
6. `src/app/admin/support/[id]/page.tsx` — Admin ticket detail

8.4 Components

1. `src/components/support/ticket-card.tsx` — Ticket list item (subject, status, date)
2. `src/components/support/message-bubble.tsx` — Chat message (user/admin differentiation)
3. `src/components/support/status-badge.tsx` — Status indicator (open, in_progress, resolved, closed)
4. `src/components/support/category-badge.tsx` — Category indicator
5. `src/components/support/priority-badge.tsx` — Priority indicator (urgent, high, normal, low)
6. `src/components/support/faq-accordion.tsx` — FAQ section (static hardcoded questions)

8.5 Types

File: `src/types/support.ts`

```
export interface SupportTicket {
  id: string;
  user_id: string;
  subject: string;
```

```
  description: string;
  category: TicketCategory;
  status: TicketStatus;
  priority: TicketPriority;
  transaction_id: string | null;
  created_at: string;
  updated_at: string;
  resolved_at: string | null;
  closed_at: string | null;
}

export interface TicketMessage {
  id: string;
  ticket_id: string;
  sender_type: 'user' | 'admin';
  sender_id: string | null;
  message: string;
  created_at: string;
}

export type TicketCategory = 'transaction_issue' | 'account_access' | 'verification' |
'general' | 'dispute';
export type TicketStatus = 'open' | 'in_progress' | 'waiting_user' | 'resolved' | 'closed';
export type TicketPriority = 'low' | 'normal' | 'high' | 'urgent';
```

8.6 Utilities

File: `src/lib/support-utils.ts`

```
import { TicketCategory, TicketPriority } from '@/types/support';

export function getAutoPriority(category: TicketCategory): TicketPriority {
  if (category === 'dispute') return 'high';
  if (category === 'account_access') return 'high';
  if (category === 'transaction_issue') return 'normal';
  if (category === 'verification') return 'normal';
  return 'low'; // general
}
```

```
export function getCategoryLabel(category: TicketCategory): string {
  const labels: Record<TicketCategory, string> = {
    transaction_issue: 'Transaksjonsproblem',
    account_access: 'Tilgangsproblem',
    verification: 'Verifisering',
    general: 'Generelt',
    dispute: 'Tvist',
  };
  return labels[category];
}

export function getStatusLabel(status: TicketStatus): string {
  const labels = {
    open: 'Åpen',
    in_progress: 'Under behandling',
    waiting_user: 'Venter på deg',
    resolved: 'Løst',
    closed: 'Lukket',
  };
  return labels[status];
}

export function getPriorityLabel(priority: TicketPriority): string {
  const labels = {
    urgent: 'Hastesak',
    high: 'Høy',
    normal: 'Normal',
    low: 'Lav',
  };
  return labels[priority];
}
```

---

9. Acceptance Criteria

9.1 Database

- ☐ support_tickets table created with all fields and constraints

- [ ] ticket_messages table created with foreign key to support_tickets
- [ ] Indexes created for performance (user_id, status, category, created_at, transaction_id)
- [ ] Schema works for both SQLite (dev) and PostgreSQL (staging)

9.2 API Endpoints

- [ ] POST /api/support/tickets creates ticket + initial message
- [ ] GET /api/support/tickets returns user's tickets with pagination
- [ ] GET /api/support/tickets/[id] returns ticket + messages
- [ ] POST /api/support/tickets/[id]/messages adds user message
- [ ] GET /api/admin/support/tickets returns all tickets (admin only)
- [ ] PATCH /api/admin/support/tickets/[id] updates status/priority (admin only)
- [ ] POST /api/admin/support/tickets/[id]/messages adds admin reply + sends email notification

9.3 Authorization

- [ ] Users can only view their own tickets (404 on unauthorized access)
- [ ] Admin endpoints require admin role (403 if not admin)
- [ ] CSRF protection via validateOrigin() middleware

9.4 UI Pages

- [ ] /support shows FAQ + create button
- [ ] /support/tickets shows user's tickets with status filter
- [ ] /support/tickets/[id] shows conversation thread + reply form
- [ ] /support/new shows create form with validation
- [ ] /admin/support shows all tickets with filters (status, priority, category)
- [ ] /admin/support/[id] shows ticket detail + admin reply form

9.5 Integration

- [ ] All support actions logged to audit_log
- [ ] Tickets can be linked to transactions
- [ ] Transaction summary displayed in ticket detail (if linked)

- [ ] Admin reply triggers email notification (placeholder implementation)

9.6 Validation

- [ ] Subject: 5-200 chars, sanitized
- [ ] Description: 20-2000 chars, sanitized
- [ ] Category: must be valid enum value
- [ ] Transaction ID: must exist and belong to user (if provided)
- [ ] Status transitions validated (open → in_progress → resolved → closed)

9.7 Edge Cases

- [ ] Closed tickets cannot receive new messages (UI shows message)
- [ ] Reopening resolved ticket changes status back to 'open'
- [ ] Empty ticket list shows "Ingen billetter" message
- [ ] Character count shown in textarea (5-200 for subject, 20-2000 for description)

---

10. Dependencies

10.1 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)

10.2 New Dependencies

None. All features use existing infrastructure.

10.3 Admin Role (TBD)

Current state: Users table has `role` field with values `user` or `merchant`.

Requirement: Add `admin` role for support staff.

Two options:

Option A: Extend existing role enum

```
ALTER TABLE users DROP CONSTRAINT users_role_check;
ALTER TABLE users ADD CONSTRAINT users_role_check CHECK(role IN ('user','merchant','admin'));
```

Option B: Separate admin table

```
CREATE TABLE IF NOT EXISTS admins (
  id TEXT PRIMARY KEY,
  email TEXT UNIQUE NOT NULL,
  password_hash TEXT NOT NULL,
  name TEXT NOT NULL,
  created_at TEXT DEFAULT (datetime('now'))
);
```

Recommendation: Option A (extend role enum) for MVP. Simpler, uses existing auth.

Implementation:

1. Add migration to extend role enum
2. Create `requireAdmin()` middleware:

```
export async function requireAdmin(request?: NextRequest) {
  const { user, error } = await requireAuth(request);
  if (error) return { user: null, error };
  if ((user as Record<string, unknown>).role !== 'admin') {
    return { user: null, error: jsonError('forbidden', 'Admin role required', 403) };
  }
  return { user, error: null };
}
```

3. Use in all admin endpoints

---

11. Testing Checklist

11.1 Unit Tests (Future)

- ☐ getAutoPriority() returns correct priority for each category

- [ ] getCategoryLabel() returns Norwegian labels
- [ ] Status validation logic

11.2 Integration Tests (Future)

- [ ] Create ticket → appears in user's ticket list
- [ ] Admin reply → user sees message in conversation
- [ ] Status change → audit log entry created
- [ ] Transaction link → transaction summary displayed

11.3 Manual Testing (MVP)

1. **User flow:**
   - Create ticket with subject, description, category
   - View ticket list (should show new ticket)
   - Open ticket detail (should show initial message)
   - Add reply message
   - Verify status changes to 'open' if was 'waiting_user'
2. **Admin flow:**
   - View all tickets dashboard
   - Filter by status, priority, category
   - Open ticket detail
   - Change status to 'in_progress'
   - Add admin reply
   - Verify user sees reply in conversation
3. **Authorization:**
   - User A cannot view User B's tickets (404)
   - Non-admin cannot access /admin/support (403)
4. **Transaction linking:**
   - Create ticket linked to transaction
   - Verify transaction summary shows in ticket detail

---

12. Implementation Order

Phase 1: Database + Types (Day 1)

1. Add schemas to `src/lib/db.ts`
2. Create `src/types/support.ts`
3. Create `src/lib/support-utils.ts`

4. Add `requireAdmin()` to `src/lib/middleware.ts`

Phase 2: User API (Day 1-2)

1. POST /api/support/tickets (create)
2. GET /api/support/tickets (list)
3. GET /api/support/tickets/[id] (detail)
4. POST /api/support/tickets/[id]/messages (reply)

Phase 3: User UI (Day 2-3)

1. /support (help center)
2. /support/new (create form)
3. /support/tickets (list)
4. /support/tickets/[id] (conversation)
5. Components: ticket-card, message-bubble, status-badge, category-badge, priority-badge, faq-accordion

Phase 4: Admin API (Day 3)

1. GET /api/admin/support/tickets (list all)
2. PATCH /api/admin/support/tickets/[id] (update)
3. POST /api/admin/support/tickets/[id]/messages (admin reply)

Phase 5: Admin UI (Day 4)

1. /admin/support (dashboard)
2. /admin/support/[id] (detail)

Phase 6: Integration (Day 4-5)

1. Audit logging for all actions
2. Transaction linking
3. Email notification (placeholder)

Phase 7: Testing + Refinement (Day 5)

1. Manual testing of all flows
2. Edge case validation
3. UI polish (spacing, colors, responsive)

---

13. Future Enhancements (Out of Scope)

1. **File attachments** — allow users to upload screenshots
2. **Multi-language support** — English, Bosnian translations
3. **Public FAQ system** — CMS-backed knowledge base
4. **Live chat** — real-time messaging via WebSocket
5. **AI chatbot** — auto-respond to common questions
6. **SLA tracking** — response time targets per priority
7. **Auto-close old tickets** — after 7 days in 'resolved' status
8. **Admin assignment** — assign tickets to specific support staff
9. **Internal notes** — admin-only notes not visible to user
10. **Ticket templates** — pre-filled forms for common issues

---

14. Notes for Implementation

14.1 Design Consistency

- Match existing Drop UI patterns (see mockups/figma-make-export/)
- Use green accent (#00E5A0) for primary actions
- Dark theme (#09090b background)
- Ensure mobile responsive (all pages must work on 375px width)

14.2 Security

- All text inputs sanitized via `sanitizeText()` from middleware
- CSRF protection via `validateOrigin()` middleware
- Audit logging for all support actions
- Users cannot view other users' tickets (authorization check)

14.3 Performance

- Pagination on all list endpoints (default 10, max 50)
- Indexes on user_id, status, category, created_at for fast queries
- Lazy load transaction details only when needed

14.4 Norwegian Text

All UI text in Norwegian (nb):

- "Opprett billett" (Create ticket)
- "Mine support-billetter" (My support tickets)
- "Trenger du hjelp?" (Need help?)
- "Send svar" (Send reply)
- "Under behandling" (In progress)
- "Venter på deg" (Waiting for you)

---

15. Spec Version History

Version	Date	Changes
1.0	2026-02-17	Initial spec — database, API, UI, integration

END OF SPEC

Revision #4
Created 2026-02-18 08:44:43 UTC by John
Updated 2026-06-07 20:00:20 UTC by John