Middleware Design Document

Middleware Design Document

> **Project:** Drop **Version:** 0.1.0 **Date:** 2026-02-23 **Author:** Platform Architect (AI)
> **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Platform Architect (AI)	Initial draft from source code analysis

---

1. Overview

Drop has two middleware layers:

1. **`src/lib/middleware.ts`** — The active middleware used by all API routes. Provides `requireAuth`, `requireMerchant`, `rateLimit`, `getClientIp`, `jsonError`, CSRF protection, and session revocation.
2. **`src/lib/middleware/`** — A modular middleware library with `auth-middleware.ts` (Bearer token for mobile), `error-handler.ts` (AppError class), and `validation.ts` (input sanitization functions).

Both layers are used in production. Routes import from `@/lib/middleware` (auth, rate limiting) and `@/lib/middleware/validation` (input validation).

---

2. Active Middleware (`lib/middleware.ts`)

2.1 `requireAuth(request?)`

Source: `middleware.ts:42–80`

Authenticates the current request via cookie-based JWT.

Returns: `{ user: User, error: null } | { user: null, error: NextResponse }`

Steps:

1. **CSRF origin check** — if `Origin` header present, must match allowed origins (`NEXT_PUBLIC_APP_URL`, `http://localhost:3000`, `http://localhost:3001`)
2. **Cookie extraction** — reads `drop_token` from request cookies
3. **JWT verification** — validates HS256 signature and expiry using `jose` library
4. **User lookup** — loads user from `users` table by `userId` from JWT payload
5. **Session revocation check** — verifies at least one non-revoked session exists for this user

Usage:

```
const { user, error } = await requireAuth(request);
if (error) return error;  // Returns NextResponse with JSON error
// user is guaranteed non-null here
```

Error responses:

- 401 `unauthorized` — missing cookie, invalid JWT, expired token, user not found, all sessions revoked

---

2.2 `requireMerchant(request?)`

Source: `middleware.ts:101–108`

Extends `requireAuth` with a merchant role check.

```
const { user, error } = await requireMerchant(request);
if (error) return error;  // 401 if not authenticated, 403 if not merchant
```

Returns 403 **`forbidden`** if user exists but `role !== 'merchant'`.

Applied to: `GET /api/merchants/dashboard`, `GET /api/merchants/qr`, `GET /api/merchants/transactions`

2.3 `rateLimit(ip, limit, windowMs?)`

Source: `middleware.ts:7–31`

Persistent IP-based rate limiter using the `rate_limits` database table.

Parameter	Default	Description
`ip`	—	Client IP address
`limit`	—	Max requests per window
`windowMs`	60,000ms	Window size in milliseconds

Returns: `boolean` — `true` if request is allowed, `false` if rate limited.

Implementation:

- Uses `runUpsert` for atomic counter creation/update
- Cleans expired entries on each call (removes rows where `expires_at < now`)
- Counter stored in `rate_limits` table: `(key, count, expires_at)`

Rate limit table schema:

```
CREATE TABLE rate_limits (
  key TEXT PRIMARY KEY,      -- IP address
  count INTEGER DEFAULT 1,
  expires_at INTEGER         -- Unix timestamp (ms)
);
```

Usage:

```
const ip = getClientIp(request);
if (!(await rateLimit(ip, 10))) {  // 10 req/min
  return jsonError("rate_limited", "Too many requests", 429);
}
```

Applied limits:

Endpoint	Limit	Window
`/api/auth/bankid/initiate`	10/min	60s
`/api/auth/bankid/callback`	10/min	60s
`/api/auth/register` (deprecated)	10/min	60s
`/api/auth/login` (deprecated)	10/min	60s

Endpoint	Limit	Window
`/api/transactions/remittance`	10/min	60s
`/api/transactions/qr-payment`	10/min	60s
`/api/rates`	120/min	60s
`/api/rates/[currency]`	120/min	60s

2.4 `getClientIp(request)`

Source: `middleware.ts:33–35`

Extracts the client's real IP address from the `x-forwarded-for` header (first IP in the chain — the originating client). Falls back to `'127.0.0.1'` if header not present.

Note: When behind App Runner (AWS managed proxy), `x-forwarded-for` is set automatically with the real client IP.

2.5 `jsonError(error, message, status, details?)`

Source: `middleware.ts:37–39`

Creates a standardized JSON error `NextResponse`.

```
return jsonError("validation_error", "Validation failed", 422, ["Email required"]);
// Response body: { "error": "validation_error", "message": "Validation failed", "details":
["Email required"] }
```

2.6 `revokeAllSessions(userId)`

Source: `middleware.ts:83–85`

Sets `revoked=1` on all sessions for a user. Called by `POST /api/auth/logout`.

```
UPDATE sessions SET revoked = 1 WHERE user_id = $1;
```

2.7 `generateCsrfToken() / validateCsrf(request, token)`

Source: `middleware.ts:88–99`

CSRF token generation (32 random bytes hex-encoded) and validation via `x-csrf-token` header.

Status: Implemented but not actively required on any route. CSRF protection is handled via:

- BankID OIDC state parameter (login flow)
- Origin header validation (in `requireAuth`)

---

3. Middleware Library (`lib/middleware/`)

3.1 Error Handler (`middleware/error-handler.ts`)

AppError class:

```
class AppError extends Error {
  constructor(
    public code: string,
    message: string,
    public status: number = 500,
    public details?: unknown
  ) {}
}
```

Predefined error constructors:

Constructor	Code	HTTP Status
`Errors.unauthorized(msg?)`	`UNAUTHORIZED`	401
`Errors.forbidden(msg?)`	`FORBIDDEN`	403
`Errors.notFound(resource)`	`NOT_FOUND`	404
`Errors.badRequest(msg, details?)`	`BAD_REQUEST`	400

Constructor	Code	HTTP Status
`Errors.conflict(msg)`	`CONFLICT`	409
`Errors.tooManyRequests(msg?)`	`RATE_LIMIT_EXCEEDED`	429
`Errors.internal(msg?)`	`INTERNAL_ERROR`	500

Error response format:

```
{
  "error": {
    "code": "BAD_REQUEST",
    "message": "Amount must be between 100 and 50000 NOK",
    "details": "validation_error"
  }
}
```

Production masking: `createErrorResponse()` masks internal error messages in production — only returns `"An unexpected error occurred"` for 500 errors.

3.2 Auth Middleware (`middleware/auth-middleware.ts`)

Alternative auth middleware for **mobile clients** using Bearer token pattern.

`requireAuth(request)`:

- Extracts JWT from `Authorization: Bearer <token>` header
- Verifies JWT signature + expiry
- Returns `userId` from payload

In-memory rate limiter (for Bearer token routes):

- `DEFAULT_RATE_LIMIT`: 100 req/min
- `STRICT_RATE_LIMIT`: 10 req/min
- Auto-cleanup every 5 minutes
- Rate limit headers: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`

`getClientIP(request)`: Checks `X-Forwarded-For` → `X-Real-IP` → falls back to `'unknown'`.

3.3 Validation (`middleware/validation.ts`)

Input validation functions — no external dependencies, all custom implementations.

Function	Description	Rules
`validatePhone(phone)`	International phone	Starts with `+`, 8–15 digits
`validateAmount(amount)`	Positive monetary amount	`> 0`, max 2 decimal places
`validateIBAN(iban)`	European IBAN	Country code + alphanumeric, mod-97 checksum
`validatePIN(pin)`	Card PIN	Exactly 4 digits
`validateEmail(email)`	Email address	Basic `x@y.z` pattern
`validateCurrency(currency)`	ISO 4217 code	Whitelist: EUR, USD, GBP, BAM, CHF, PLN, NOK, RSD, TRY, PKR
`validateDateISO(date)`	ISO 8601 date	Parseable by `Date.parse()`
`validateName(name)`	Name field	1–100 chars, at least one letter, XSS-safe
`validateLanguage(lang)`	Language code	Whitelist: nb, en, bs, sq
`sanitizeText(text, maxLength?)`	Text sanitization	Strips HTML tags + control chars, trims, enforces max length (default 500)
`validate(condition, msg)`	Assert helper	Throws `AppError` (400) if false
`required(value, name)`	Required field check	Throws `AppError` (400) if null/undefined

Security notes:

- `validateName` checks for: `<script`, `javascript:`, `onerror=`, `onclick=` — blocks XSS injection in name fields
- `sanitizeText` removes HTML tags via regex, strips control characters
- `validateIBAN` implements full mod-97 checksum algorithm
- `validateAmount` rejects `NaN`, `Infinity`, negative values

---

4. Security Headers (Next.js Config)

Applied to all responses via `next.config.ts`:

Header	Production Value	Development Value	Purpose
`Content-Security-Policy`	`default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; img-src 'self' data: blob:; connect-src 'self'; frame-ancestors 'none'`	Adds `'unsafe-eval'` + `'unsafe-inline'` for HMR	XSS protection

Header	Production Value	Development Value	Purpose
`X-Frame-Options`	`DENY`	`DENY`	Clickjacking prevention
`X-Content-Type-Options`	`nosniff`	`nosniff`	MIME sniffing prevention
`Referrer-Policy`	`strict-origin-when-cross-origin`	Same	Referrer leakage prevention
`Permissions-Policy`	`camera=(self), microphone=(), geolocation=(self)`	Same	Feature restriction
`Strict-Transport-Security`	`max-age=63072000; includeSubDomains; preload`	Same	Force HTTPS (2-year HSTS)

---

5. Middleware Usage Matrix

Route	Rate Limit	`requireAuth`	`requireMerchant`	Feature Flag	Validation Functions
GET `/api/auth/bankid`	10/min	No	No	No	—
GET `/api/auth/bankid/callback`	10/min	No	No	No	state cookie
GET `/api/auth/me`	No	Yes	No	No	—
POST `/api/auth/logout`	No	Yes	No	No	—
POST `/api/auth/refresh`	No	Yes	No	No	—
GET `/api/transactions`	No	Yes	No	No	—
POST `/api/transactions/remittance`	10/min	Yes	No	No	`validateAmount`
POST `/api/transactions/qr-payment`	10/min	Yes	No	No	`validateAmount`
GET `/api/rates`	120/min	No	No	No	—
POST `/api/recipients`	No	Yes	No	No	`validateName`, country whitelist
POST `/api/merchants/register`	No	Yes	No	No	`validateName`, orgNumber
GET `/api/merchants/dashboard`	No	Yes	Yes	No	period whitelist

Route	Rate Limit	`requireAuth`	`requireMerchant`	Feature Flag	Validation Functions
GET `/api/notifications`	No	Yes	No	`notifications`	—
PATCH `/api/notifications`	No	Yes	No	`notifications`	ID format, max 100
PATCH `/api/settings`	No	Yes	No	No	currency/language whitelist
POST `/api/cards/[id]/physical`	No	Yes	No	`physicalCards`	address min 10 chars
POST `/api/cards/[id]/pin`	No	Yes	No	`cardPin`	`validatePIN`
GET/PUT `/api/cards/[id]/limits`	No	Yes	No	`spendingLimits`	limitType whitelist

---

6. Error Spike Detection

Implemented in `src/lib/alerts.ts` as a middleware-adjacent concern:

- Every HTTP 5xx response triggers `trackError()` (called in `jsonError()` middleware for 500 errors)
- Rolling 1-minute window of error timestamps maintained in-memory
- When count > 5 in 60 seconds → sends critical Slack alert to `#drop-ops`
- 10-minute cooldown per alert title prevents spam

Limitation: Error counter is in-memory only — resets on application restart. Redis-backed counter planned for v1.0.

Related Documents

- Backend Architecture
- API Reference
- Source: MIDDLEWARE.md

Approval

Role	Name	Date	Signature
Author	Platform Architect (AI)	2026-02-23	
Reviewer			
Approver	Alem Bašić		

Revision #4
Created 2026-02-18 08:44:22 UTC by John
Updated 2026-05-31 20:01:58 UTC by John