Developer Experience

Onboarding, offboarding, coding standards, local dev setup

- Developer Onboarding Guide: Drop — Fintech Payment App
- Developer Offboarding Guide: Drop — Fintech Payment App
- Coding Standards: Drop — Fintech Payment App
- Local Development Setup: Drop — Fintech Payment App
- Developer Onboarding Guide
- Developer Offboarding Guide
- Coding Standards
- Local Development Setup

Developer Onboarding Guide: Drop — Fintech Payment App

Developer Onboarding Guide: Drop — Fintech Payment App

> **Project:** Drop — Remittance + QR Payments **Version:** 1.0 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Approved **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial onboarding guide — AI-native team (Builder + Validator agents)

Welcome

Welcome to **Drop** — remittance + QR payments for everyone in Scandinavia.

Drop is built by an AI-native team at ALAI Holding AS. Our team structure is unusual: John (AI Director, Claude Opus) coordinates Builder agents (Claude Sonnet) and Validator agents (Claude Sonnet, read-only). Alem Bašić (CEO) provides direction and final approvals.

If you are a new Builder or Validator agent joining a Drop task, this guide will get you oriented quickly. If anything in this guide is unclear or out of date, please update it as you go.

Your first session at a glance:

- Step 1: Read this guide + `CLAUDE.md` in the project root
- Step 2: Set up local environment (Section 2) + verify tests pass

- Step 3: Read the architecture overview (Section 4)
- Step 4: Pick up your assigned Mission Control task

---

1. Prerequisites

Hardware Requirements

Component	Minimum	Recommended
CPU	4 cores	8+ cores (bcrypt is CPU-intensive in tests)
RAM	8GB	16GB
Disk	10GB free	SSD with 20GB free
OS	macOS (primary), Linux, Windows (WSL2)	macOS

Accounts You Need

Account	Why	How to Get Access
GitHub (alai-org)	Code repository	Ask John (AI Director)
Fly.io	Staging deployment (drop-app)	Ask John (AI Director)
Vaultwarden (`vault.basicconsulting.no`)	Secrets / credentials	Ask John (AI Director)
alai-talk.slack.com	Team communication	Ask Alem Bašić (CEO)
Mission Control (`node ~/system/tools/mc.js`)	Task management	Local system tool

Expected setup time for all accounts: 1 session (AI agents are provisioned per-task)

---

2. Development Environment Setup

2.1 macOS Setup

```
# Install Homebrew (if not already installed)
/bin/bash -c "$(curl -fsSL
```

```
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Node.js via nvm
brew install nvm
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"' >> ~/.zshrc
source ~/.zshrc

# Install Node 20 (Drop uses Next.js 16 which requires Node 20+)
nvm install 20
nvm use 20
nvm alias default 20
```

2.2 Linux Setup

```
# Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
source ~/.bashrc

# Install Node 20
nvm install 20
nvm use 20
```

2.3 Required Software & Versions

Software	Required Version	Install
Node.js	`20.x`	`nvm install 20`
npm	`10.x` (included with Node 20)	Included with Node
flyctl	Latest	`brew install flyctl` (for staging access)
Playwright browsers	Latest	`npx playwright install`

Check your versions:

```
node --version    # Should be v20.x.x
npm --version     # Should be 10.x.x
```

2.4 IDE Setup

Recommended IDE: VS Code

Required extensions:

Extension	Purpose
`dbaeumer.vscode-eslint`	ESLint inline linting
`esbenp.prettier-vscode`	Format on save
`bradlc.vscode-tailwindcss`	Tailwind CSS IntelliSense
`ms-playwright.playwright`	Playwright test runner
`vitest.explorer`	Vitest test explorer

2.5 Environment Variables

```
# Copy the example env file
cp src/drop-app/.env.example src/drop-app/.env.local

# Fill in your local values
# Retrieve from Vaultwarden: vault.basicconsulting.no
# CLI: bw get item "Drop Dev .env" --session $(cat /tmp/bw-session)
```

Key variables:

Variable	Value for Local Dev	Notes
`DATABASE_URL`	`./local.db` (SQLite)	Auto-created on first migration
`JWT_SECRET`	Any random 32+ char string	NEVER use in production
`NEXT_PUBLIC_SERVICE_MODE`	`mock`	Always `mock` for local dev
`BCRYPT_ROUNDS`	`10`	Use 10 locally (faster); production uses 12
`SUMSUB_API_KEY`	`mock`	Mocked in dev
`BAAS_API_KEY`	`mock`	Mocked in dev

Never commit `.env.local`: It's in `.gitignore`.

---

3. Repository Setup

```
# Step 1: Navigate to Drop app directory
cd ~/ALAI/products/Drop/src/drop-app

# Step 2: Install dependencies
npm install

# Step 3: Set up the database
npm run db:migrate

# Step 4: Seed development data
npm run db:seed

# Step 5: Configure environment (see 2.5)
cp .env.example .env.local
# Edit .env.local with your values

# Step 6: Start the development server
npm run dev

# Step 7: Open the application
open http://localhost:3000
# Expected: Drop landing page or login screen
```

---

4. Architecture Overview

High-level architecture: Drop is a PSD2 pass-through fintech app — it NEVER holds customer money.

CRITICAL — Read Before Writing Any Code:

> Drop uses an AISP/PISP pass-through model (ADR-003). The `users` table MUST NOT have a `balance` column. The `cards` table MUST NOT have `card_number` or `cvv` columns. These are tested in `db.test.ts` on every commit. Violating this is a P0 incident.

In brief:

- **Frontend:** Next.js 16 (App Router), React 19, TypeScript, Tailwind CSS — in `src/drop-app/`
- **Backend:** Next.js API Routes (26 routes) — in `src/drop-app/src/app/api/`
- **Database:** SQLite (dev) → PostgreSQL (Phase 1 production) — schema in `src/drop-app/db/`
- **Key external services:** Mock BaaS (AISP + PISP), Mock Sumsub KYC, exchange rates (static seed)
- **Deployment:** Fly.io (Stockholm region) for backend; Vercel for landing page
- **Auth:** JWT in httpOnly cookie, SameSite=Strict, 7-day expiry; bcrypt 12 rounds

Codebase tour (key directories):

Directory	Purpose
`src/drop-app/src/app/api/`	26 API routes (auth, transactions, rates, merchants, cards, health)
`src/drop-app/src/app/(app)/`	App pages (dashboard, send money, QR scan, history, profile)
`src/drop-app/src/components/`	Shared UI components (drop-logo.tsx, etc.)
`src/drop-app/db/`	SQLite schema, migrations, seed scripts
`src/drop-app/__tests__/`	14 test files (unit, integration, performance, E2E)
`mockups/figma-make-export/`	UI source of truth (10 screens — ALWAYS check here before UI changes)

---

5. Key Documentation Links

Document	Link	Purpose
Project overview	`CLAUDE.md`	Pass-through model rules, branding, features
API Reference	`docs/backend/API-REFERENCE.md`	All 26 API endpoints
Database Schema	`docs/backend/DATABASE-SCHEMA.md`	Table definitions, compliance rules
Architecture	`project/architecture/drop-architecture.md`	System design, ADRs
Testing Guide	`docs/testing/TESTING-GUIDE.md`	How to run Vitest + Playwright
Test Inventory	`docs/testing/TEST-INVENTORY.md`	All 14 test files documented
Coding Standards	coding-standards.md	How we write code
Definition of Done	`docs/templates-testing/definition-of-done.md`	What "done" means
Security Audit	`project/docs/security-qa-audit.md`	Known issues + status

Document	Link	Purpose
Roadmap	`ROADMAP.md`	Phases 0/0.5/1/2/3 + blockers

---

6. Coding Standards Reference

Standards document: coding-standards.md

Quick reference:

- **Language:** TypeScript — strict mode, no `any`
- **Commit messages:** Conventional Commits format (`feat:`, `fix:`, `test:`, etc.)
- **Branch naming:** `feature/MC-{task-id}-short-description`
- **PR size:** Aim for < 300 lines changed per PR
- **Testing:** Every PR must include tests for new code
- **SQL:** Parameterized queries ONLY — never string concatenation
- **Passwords:** bcrypt ONLY — SHA-256 is explicitly rejected in `auth.test.ts`

---

7. Git Workflow & Branching Strategy

```
# Create a feature branch (aligned to Mission Control task)
git checkout -b feature/MC-{task-id}-description

# Make changes, then commit using Conventional Commits
git add src/drop-app/...
git commit -m "feat(auth): add bcrypt rounds configuration"

# Push your branch
git push origin feature/MC-{task-id}-description

# Open PR to main branch
```

PR requirements:

1. Title follows Conventional Commits: `type(scope): description`
2. All Vitest tests passing (`npm run test`)
3. TypeScript compiles (`npm run type-check`)
4. ESLint passes (`npm run lint`)
5. Validator agent approved (read-only review)

8. First Task Checklist

- ☐ Development environment running locally (`npm run dev` at `http://localhost:3000`)
- ☐ Test suite passing: `npm run test` (40+ Vitest tests all green)
- ☐ E2E tests passing: `npx playwright test` (3 projects)
- ☐ Read `CLAUDE.md` in `~/ALAI/products/Drop/` — understand pass-through model
- ☐ Read `docs/testing/TESTING-GUIDE.md` — understand test structure
- ☐ Mission Control task assigned and marked `in_progress`
- ☐ First change includes appropriate tests
- ☐ Definition of Done checklist completed before marking task done

---

9. Team Rituals & Communication

Drop uses an AI-native async workflow:

Communication Type	Channel	Purpose
Task coordination	Mission Control (`node ~/system/tools/mc.js`)	All task assignments, status
Technical decisions	`comms/decisions/` in the repo	Architecture decisions, ADRs
CEO updates	MCP email (John → alem@alai.no)	Business milestones, blockers
Engineering	alai-talk.slack.com #drop	Engineering discussions
Security issues	alai-talk.slack.com #drop-security	Immediate escalation

No daily standups — async by default. John (AI Director) coordinates via Mission Control.

---

10. Key Contacts

Role	Name	Contact	Help With
AI Director / Tech Lead	John (Claude Opus)	Mission Control DM	Architecture, task assignment, code review
QA Lead	Validator Agent	Task coordination	Test review, compliance checks

Role	Name	Contact	Help With
CEO / Product Owner	Alem Bašić	alem@alai.no	Business decisions, UAT sign-off

---

Related Documents

- Local Development Setup
- Coding Standards
- Developer Offboarding Guide
- Definition of Done

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	Approved (AI)
Tech Lead	John	2026-02-23	Approved
CEO (Alem)	Alem Bašić	TBD	

Developer Offboarding Guide: Drop — Fintech Payment App

Developer Offboarding Guide: Drop — Fintech Payment App

> **Project:** Drop — Remittance + QR Payments **Version:** 1.0 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Approved **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial offboarding guide — AI-native team context

---

1. Offboarding Overview

Developer: {DEVELOPER_NAME} **Last Session:** {LAST_DATE} **Manager:** John (AI Director) **Offboarding Coordinator:** John (AI Director) **Security Review:** John (AI Director) + Alem Bašić (CEO)

Departure type: Agent session completion / Agent role change / Human developer departure

Drop offboarding context: Because Drop uses an AI-native team (Builder agents, Validator agents), most "offboarding" is agent session completion — no persistent access to revoke. Human developer offboarding is documented in full below for Alem Bašić or any future human team members.

Handoff started: {HANDOFF_START} **Access revocation deadline:** Same day for involuntary; planned for voluntary

2. Access Revocation Checklist

For AI agent team members (Builder / Validator agents): Agent sessions are ephemeral — no persistent credentials. Verify:

- [ ] Agent session terminated in Claude Code / Mission Control
- [ ] No API keys or secrets were stored in agent memory or session files
- [ ] All in-progress work committed or documented in Mission Control

For human developers (Alem Bašić or future hires):

Code & Version Control

- [ ] **GitHub (alai-org)** — remove from organization and all Drop repositories
- [ ] **SSH keys** — remove from GitHub: Settings > SSH and GPG keys
- [ ] **Personal access tokens** — revoke all tokens in GitHub settings
- [ ] **Fly.io** — remove from `drop-app` app team (fly.io dashboard → Members)

Cloud Infrastructure (Fly.io)

- [ ] **Fly.io team** — remove from `drop-app` org/team
- [ ] **Fly.io deploy tokens** — revoke any personal deploy tokens
- [ ] **SSH keys on Fly.io machines** — remove from `~/.ssh/authorized_keys` on app machines

Secrets & Credentials — CRITICAL FOR FINTECH

Drop handles financial data. All shared secrets must be rotated immediately on any departure:

- [ ] **Vaultwarden** (`vault.basicconsulting.no`) — remove user account; rotate all shared vault items they had access to
- [ ] `JWT_SECRET` — rotate immediately; rotate all active user sessions (deploy new secret)
- [ ] `BAAS_API_KEY` — rotate with BaaS partner (SpareBank1 / Swan) — Phase 2
- [ ] `SUMSUB_API_KEY` — rotate with Sumsub — Phase 2

- [ ] **Database URL / password** — rotate Fly.io PostgreSQL password (Phase 1+)
- [ ] **GitHub Actions secrets** — rotate any deploy keys / secrets in repo settings

All secrets known to this developer:

Secret	Location	Rotated	Rotated By
`JWT_SECRET`	Vaultwarden + Fly.io secrets	Yes / No	John
`BAAS_API_KEY`	Vaultwarden + Fly.io secrets	Yes / No	John
`SUMSUB_API_KEY`	Vaultwarden + Fly.io secrets	Yes / No	John
Vaultwarden master vault	Vaultwarden	Yes / No	Alem Bašić

Third-Party Services

- [ ] **alai-talk.slack.com** — deactivate account; remove from #drop, #drop-security channels
- [ ] **Mission Control** — remove any permanent task assignments

Access revocation completion signed off by: John (AI Director) + Alem Bašić (CEO) on {DATE}

---

3. Knowledge Transfer

Active Projects & Ownership Transfer

Project / Area	Current Status	New Owner	Handoff Complete
Drop Phase 0.5 security hardening	{STATUS}	Builder Agent (next session)	Yes / No
Drop Phase 1 BaaS integration	{STATUS}	John (AI Director)	Yes / No
Finanstilsynet registration prep	{STATUS}	John (AI Director)	Yes / No

Ongoing Work Documentation

Work Item	Mission Control Task	Status	Documentation	New Owner
{WORK_1}	MC-{ID}	{STATUS}	{LINK}	John
{WORK_2}	MC-{ID}	{STATUS}	{LINK}	John

Documentation written during knowledge transfer:

- [ ] All in-progress PRs reviewed and commented
- [ ] Active branches documented and either merged or closed
- [ ] Ongoing investigations/research notes written up in `comms/decisions/`
- [ ] Architecture decisions in progress documented as ADRs
- [ ] Pending operational tasks documented in `docs/OPERATIONS/`

Key Contacts & Relationships

Contact	Company / Role	Relationship	Transferred To
SpareBank1 BD contact	SpareBank1 (potential BaaS)	BaaS partnership pitch	Alem Bašić (CEO)
Swan.io contact	Swan (backup BaaS)	BaaS partnership pitch	Alem Bašić (CEO)
Finanstilsynet contact	Norwegian FSA	PSD2 registration	Alem Bašić (CEO) + Legal
Sumsub account manager	Sumsub (KYC provider)	KYC integration	John (AI Director)

Drop-Specific Tribal Knowledge Capture

Knowledge transfer sessions:

Topic	Date	Format	Notes Doc
Pass-through model ADR-003	2026-02-23	Written in `project/architecture/`	ADR-003
Security audit findings	2026-02-23	Written in `security/drop-security-rapport.md`	Security audit
BaaS mock implementation	2026-02-23	Code in `src/drop-app/lib/baas-mock.ts`	CODE-BAAS.md

Capture questions answered:

1. What breaks in production that only you know how to fix? → SQLite concurrent write limit (200 users); documented in NFR-S01
2. What shortcuts or workarounds exist? → Mock BaaS in `NEXT_PUBLIC_SERVICE_MODE=mock`; documented in CLAUDE.md

3. What external services have non-obvious quirks? → Sumsub webhook signature validation; documented in sumsub-integration.test.ts
4. What technical debt exists? → Documented in `docs/CROSS-CUTTING/tech-debt-log.md`
5. Upcoming risks? → BaaS partner not confirmed; SQLite concurrent limit; documented in risk-register.md

---

4. Code Ownership Transfer

CODEOWNERS Update

```
# Review current code ownership assignments
# (No formal CODEOWNERS file yet — John (AI Director) owns all Drop code)

# Transfer to new agent/developer:
# Update CLAUDE.md "Builder" and "Validator" role assignments
# Update Mission Control task ownership
```

- ☐ Mission Control task ownership transferred
- ☐ New builder/validator agents briefed on active tasks
- ☐ John (AI Director) notified of any in-flight architecture decisions

PR Review Reassignment

- ☐ Open PRs awaiting review: reassigned to Validator Agent (new session)
- ☐ In-progress PR review responsibilities communicated to John (AI Director)

---

5. Asset Return

Drop is AI-native — no physical hardware assets for agent team members.

For human developer offboarding:

Asset	Return By	Returned
Laptop (ALAI issued, if any)	Last day	Yes / No
Access cards / badges (N/A — remote)	—	—

IT coordinator: Alem Bašić (CEO) — contact@alai.no

---

6. Exit Interview Topics

For human developers leaving the Drop project:

Exit interview conducted by: John (AI Director) + Alem Bašić (CEO) (joint, async OK) **Format:** Written notes in `comms/decisions/YYYY-MM-DD-exit-{name}.md`

Topics to cover:

- What did you learn from working on a fintech pass-through payment system?
- Were there any technical decisions you disagreed with? (ADR feedback)
- What gaps exist in the documentation or test coverage?
- Any concerns about the codebase security or compliance you want to flag?
- What would you do differently in Phase 1?

Exit notes: Stored in `comms/decisions/` (confidential — CEO + AI Director access only)

---

7. Final Checklist Sign-Off

John (AI Director) Sign-Off

- ☐ All access revocation items completed
- ☐ `JWT_SECRET` and all shared secrets rotated
- ☐ Knowledge transfer complete (ADRs, decisions, tribal knowledge documented)
- ☐ Code ownership transferred in Mission Control
- ☐ All open PRs and tasks handed off
- ☐ Security audit log reviewed for last 30 days (no anomalies)

John (AI Director): John | **Date:** {DATE} | **Signature:** Approved (AI)

Developer Sign-Off

- ☐ All work documented and handed off to new owner
- ☐ No Drop production credentials retained on personal devices
- ☐ Exit interview/notes completed

Developer: {DEVELOPER_NAME} | **Date:** {DATE} | **Signature:** _____

CEO Sign-Off (Alem Baši?)

- ☐ Vaultwarden access revoked
- ☐ Fly.io team membership confirmed removed
- ☐ Shared BaaS/financial credentials confirmed rotated
- ☐ Business relationships (BaaS, Finanstilsynet, Sumsub contacts) transferred

Alem Bašić (CEO): | **Date:** {DATE} | **Signature:** _____

Related Documents

- Developer Onboarding Guide
- Coding Standards
- Security Audit Report

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	Approved (AI)
Tech Lead	John	2026-02-23	Approved
CEO (Alem)	Alem Bašić	TBD	

Coding Standards: Drop — Fintech Payment App

Coding Standards: Drop — Fintech Payment App

> **Project:** Drop — Remittance + QR Payments **Version:** 1.0 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Approved **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial standards — TypeScript strict, Next.js App Router, fintech security rules

---

Purpose

These standards apply to all code in **Drop**. When automated tools enforce a rule, the tool wins. When in doubt, optimize for readability — the next AI agent reading your code is trying to understand intent, not guess it.

Non-negotiable: All new code must pass TypeScript strict check, ESLint, and Prettier formatting before merge. No exceptions.

Fintech non-negotiable: Parameterized SQL everywhere. No stored balances. No stored card numbers or CVV. These are encoded as automated tests — failing these means P0.

---

1. Language-Specific Conventions

1.1 Naming Conventions

TypeScript (Drop's primary language):

Type	Convention	Example
Variables	`camelCase`	`userId`, `kycStatus`
Functions	`camelCase`	`hashPassword()`, `calculateFee()`
Classes	`PascalCase`	`TransactionService`
Interfaces	`PascalCase` (no `I` prefix)	`User`, `TransactionResult`
Types	`PascalCase`	`KycStatus`, `CurrencyCode`
Enums	`PascalCase`	`TransactionType`, `KycStatus`
Constants	`UPPER_SNAKE_CASE`	`REMITTANCE_FEE_RATE`, `MIN_AMOUNT_NOK`
Files — components	`PascalCase.tsx`	`SendMoneyForm.tsx`
Files — utilities	`camelCase.ts`	`calculateFee.ts`, `verifyJWT.ts`
Files — API routes	`route.ts` (Next.js convention)	`app/api/auth/login/route.ts`
Test files	`{name}.test.ts`	`auth.test.ts`, `db.test.ts`
E2E test files	`{name}.spec.ts`	`user-flows.spec.ts`

Drop-specific constants:

```
// Fee rates (business rules — change requires CEO approval + ADR)
const REMITTANCE_FEE_RATE = 0.005;  // 0.5%
const QR_MERCHANT_FEE_RATE = 0.01;  // 1%
const MIN_REMITTANCE_NOK = 100;
const MAX_REMITTANCE_NOK = 50_000;

// Supported corridors
const SUPPORTED_CURRENCIES = ['RSD', 'BAM', 'PKR', 'TRY', 'PLN', 'EUR'] as const;
type CurrencyCode = typeof SUPPORTED_CURRENCIES[number];
```

1.2 File Organization

```
src/drop-app/src/
├── app/
```

```
│   ├── api/                 # Next.js API Routes
│   │   ├── auth/
│   │   │   ├── login/route.ts
│   │   │   ├── register/route.ts
│   │   │   └── me/route.ts
│   │   ├── transactions/
│   │   │   ├── remittance/route.ts
│   │   │   └── qr-payment/route.ts
│   │   └── rates/
│   │       ├── route.ts           # GET /api/rates
│   │       └── [currency]/route.ts # GET /api/rates/RSD
│   └── (app)/               # Protected pages (dashboard, send-money, etc.)
├── lib/
│   ├── auth.ts              # JWT, bcrypt, session helpers
│   ├── db.ts                # SQLite/PostgreSQL client
│   ├── validate.ts          # Zod schemas for input validation
│   └── fee.ts               # Fee calculation functions
└── components/
    ├── drop-logo.tsx        # Brand logo component
    └── ui/                  # Shared UI components
```

Rules:

- One route handler per file (`route.ts` in Next.js App Router)
- Business logic extracted to `lib/` — never inline in route handlers
- Test files co-located with source files in `__tests__/`

1.3 Import Ordering

```
// 1. Node built-ins
import { cookies } from 'next/headers';

// 2. External dependencies (node_modules)
import { z } from 'zod';
import * as bcrypt from 'bcrypt';
import { SignJWT, jwtVerify } from 'jose';

// 3. Internal — absolute paths (@/ alias)
import { db } from '@/lib/db';
import { verifyJWT } from '@/lib/auth';
```

```
import type { User } from '@/lib/types';

// 4. Internal — relative paths
import { validateRemittanceBody } from './validate';
```

Enforced by: ESLint `import/order` rule

1.4 Error Handling Patterns

```
// PREFERRED — consistent HTTP error responses in Next.js API routes
export async function POST(request: Request): Promise<Response> {
  let body: unknown;
  try {
    body = await request.json();
  } catch {
    return Response.json({ error: 'Invalid JSON body' }, { status: 400 });
  }

  const validation = remittanceSchema.safeParse(body);
  if (!validation.success) {
    return Response.json(
      { error: 'Validation failed', details: validation.error.flatten() },
      { status: 422 }
    );
  }

  // Business logic in lib/ — throws typed errors
  try {
    const result = await processRemittance(validation.data);
    return Response.json(result, { status: 201 });
  } catch (error) {
    if (error instanceof InsufficientBalanceError) {
      return Response.json({ error: 'Insufficient balance' }, { status: 402 });
    }
    if (error instanceof KycRequiredError) {
      return Response.json({ error: 'KYC verification required' }, { status: 403 });
    }
    // Never expose internal errors
    console.error('Unhandled error in remittance:', error);
    return Response.json({ error: 'Internal server error' }, { status: 500 });
```

```
  }
}
```

Rules:

- NEVER return stack traces or DB error messages to the client
- NEVER use `any` type — TypeScript strict mode enforces this
- NEVER swallow errors silently
- Use 402 for Insufficient Balance (fintech convention), not 400
- Error messages in Norwegian for user-facing validation

---

2. Code Formatting

2.1 Formatter

Language	Tool	Config File	Enforced In
TypeScript / JavaScript	Prettier	`.prettierrc`	CI + pre-commit hook
JSON / YAML	Prettier	`.prettierrc`	CI + pre-commit hook

Key formatting rules:

- Indentation: 2 spaces
- Max line length: 100 characters
- Semicolons: required
- Trailing commas: `all` (ES2020)
- Quotes: single quotes

Auto-format on save: Enabled via `.vscode/settings.json` (see local-development-setup.md)

2.2 Linter

Language	Tool	Config	Rules Severity
TypeScript	ESLint + TypeScript-ESLint	`.eslintrc.json`	Error = blocks CI

Linting rules that are errors (block CI):

- `@typescript-eslint/no-explicit-any` — no `any` types
- `no-console` (except `console.error` in server code)
- `@typescript-eslint/no-unused-vars`
- SQL string concatenation (custom Drop rule)

Disable linting inline (sparingly):

```
// eslint-disable-next-line @typescript-eslint/no-explicit-any -- External library returns
untyped response
const rawResponse: any = await externalLibrary.call();
```

Every inline disable must have a comment explaining why.

---

3. Git Conventions

3.1 Commit Message Format

Standard: Conventional Commits

Format:

```
<type>(<scope>): <description>

[optional body]

[optional footer(s)]
```

Types:

Type	When to Use
`feat`	New feature
`fix`	Bug fix
`docs`	Documentation only
`style`	Formatting changes (no logic change)
`refactor`	Code change that neither fixes a bug nor adds a feature
`test`	Adding or updating tests
`chore`	Build system, dependency updates, CI changes
`perf`	Performance improvements
`security`	Security fix or hardening

Drop examples:

```
feat(auth): add bcrypt rounds configuration via env var
fix(transactions): prevent double-spend with per-user lock
test(db): add assertion that users table has no balance column
security(middleware): add CSRF protection to all mutating endpoints
chore: upgrade Next.js to 15.x patch
```

Breaking changes:

```
feat(api)!: rename /api/transactions/send to /api/transactions/remittance

BREAKING CHANGE: All clients must update API calls to new path
```

3.2 Branch Naming Convention

Type	Pattern	Example
Feature	`feature/MC-{task-id}-description`	`feature/MC-042-rate-limiting`
Bug fix	`fix/MC-{task-id}-description`	`fix/MC-051-double-spend`
Hotfix	`hotfix/MC-{task-id}-description`	`hotfix/MC-060-jwt-bypass`
Security	`security/MC-{task-id}-description`	`security/MC-070-csrf-fix`
Chore	`chore/MC-{task-id}-description`	`chore/MC-080-upgrade-deps`

Rules:

- Lowercase only; hyphens, not underscores
- Always include Mission Control task ID (`MC-{id}`)

3.3 PR Title & Description Format

Title format: Same as commit message: `type(scope): description`

Required PR description sections:

1. **What**: What does this PR do? (bullet points)
2. **Why**: Which Mission Control task / acceptance criterion?
3. **Pass-through model check**: Does this change touch DB schema? If yes, confirm no balance/CVV columns added
4. **Tests**: What tests were added/modified?
5. **Security**: Does this touch auth, payments, or input validation?

3.4 PR Size Guidelines

Size	Lines Changed	Status
Small	< 200	Ideal — review same session
Medium	200-400	Acceptable
Large	400-800	Needs justification — split if possible
Extra Large	> 800	Exceptional only — must be pre-approved by John

---

4. Code Review Guidelines

4.1 What to Look For (Validator Agent Checklist)

Reviewers must check:

- ☐ Pass-through model invariant: no `balance` column; no `card_number` or `cvv`
- ☐ Parameterized SQL — no string interpolation in any DB query
- ☐ bcrypt used for passwords — SHA-256 explicitly forbidden
- ☐ JWT secret not hardcoded; not in response body
- ☐ Rate limiting applies to new auth/transaction endpoints
- ☐ Input validation (Zod schema) on all new request bodies
- ☐ Error messages don't leak internal state
- ☐ Fee calculations correct (0.5% remittance; 1% QR)
- ☐ Tests cover the change and edge cases

Reviewers should NOT block on:

- Personal style preferences covered by Prettier/ESLint (the linter decides)
- Minor refactors not related to the PR's scope

4.2 Review Turnaround

PR Type	First review
Security / Critical fix	Same session
Standard feature	Within 1 session

4.3 Approval Requirements

Branch	Required Approvals
`main`	Validator agent approval + John (AI Director) sign-off
Feature branches	Validator agent approval
Hotfix	John (AI Director) or Alem Bašić (CEO) approval — async OK

4.4 Constructive Review Feedback

- `nit:` — Minor issue, not a blocker
- `question:` — Need clarification
- `suggestion:` — Improvement idea (not required)
- `blocker:` — Must be fixed before merge (e.g., SQL injection risk, bcrypt bypass)

---

5. Testing Standards

5.1 Test Naming Conventions

```
// Vitest format: describe → it('should [behavior] when [condition]')
describe('hashPassword', () => {
  it('should return a bcrypt hash starting with $2', async () => { ... });
  it('should reject SHA-256 hashes at verify time', async () => { ... });
  it('should reject passwords longer than 1000 characters', async () => { ... });
});

describe('calculateRemittanceFee', () => {
  it('should return 5 NOK fee for 1000 NOK amount', () => { ... });
  it('should reject amounts below 100 NOK', () => { ... });
});
```

5.2 Test Organization

```
src/drop-app/__tests__/
├── auth.test.ts              # Unit: bcrypt, JWT, password validation
├── validation.test.ts        # Unit: input validation, XSS, injection
├── transactions.test.ts      # Unit: fee calculation, transaction logic
├── rates.test.ts             # Unit: exchange rates
```

```
├── api-endpoints.test.ts     # Integration: all 26 API routes
├── db.test.ts                # Integration: schema compliance, FK constraints
├── middleware.test.ts        # Integration: rate limiting, auth, CSRF
├── api-benchmarks.test.ts    # Performance: bcrypt timing, DB queries
├── feature-flags.test.ts     # Unit: feature flag behavior
├── regression-suite.test.ts  # Regression: critical path smoke
├── sumsub-integration.test.ts # Integration: KYC webhook mock
├── cards-integration.test.ts  # Integration: cards (feature-flagged)
└── e2e/
    ├── user-flows.spec.ts    # E2E: registration, login
    ├── full-flows.spec.ts    # E2E: remittance, QR payment
    └── input-chaos.spec.ts   # E2E: 20+ validation edge cases
```

5.3 Mocking Guidelines

```
// PREFERRED: Mock BaaS at the module level (NEXT_PUBLIC_SERVICE_MODE=mock)
// The mock mode is configured via env var — no vi.mock() needed for BaaS

// PREFERRED for unit tests: vi.mock for db queries
vi.mock('@/lib/db', () => ({
  db: {
    query: vi.fn(),
    run: vi.fn(),
  }
}));

// For integration tests: use real SQLite in-memory DB
// (configured in vitest.config.ts — no mock needed)

// NEVER mock the pass-through model assertions in db.test.ts
// NEVER mock the bcrypt rejection tests in auth.test.ts
```

5.4 Coverage Requirements

Layer	Lines	Branches	Notes
Auth module	100%	100%	Fintech security — strictly enforced
Transaction logic	100%	100%	Fee calculation — strictly enforced

Layer	Lines	Branches	Notes
API handlers	≥ 80%	≥ 70%	
Input validation	≥ 90%	≥ 85%	Security-critical
DB layer	≥ 90%	—	Compliance assertions required
Overall minimum	≥ 80%	—	CI gate

---

6. Documentation Standards

6.1 When to Add Comments

```
// GOOD — explains WHY (not obvious from code)
// Bcrypt rounds set to 12 (not 10) per NFR-SEC02 fintech standard.
// Using 10 rounds would be faster but falls below security threshold.
const BCRYPT_ROUNDS = parseInt(process.env.BCRYPT_ROUNDS ?? '12', 10);

// GOOD — documents non-obvious business rule
// Fee is calculated on gross amount, not total debit (gross + fee)
// This is required by Drop's pass-through model (ADR-003)
const fee = amount * REMITTANCE_FEE_RATE;

// BAD — restates what code says
// Multiply amount by fee rate
const fee = amount * REMITTANCE_FEE_RATE;
```

Comment when:

- A fintech business rule is implemented (always cite the rule)
- A security decision was made (e.g., why bcrypt rounds = 12)
- A workaround exists for an external system quirk

6.2 ADR Format

Write an Architecture Decision Record when:

- Changing the database (SQLite → PostgreSQL)
- Changing the authentication mechanism (DOB → BankID)
- Modifying the fee model (new corridor, new fee rate)

- Adding a new BaaS partner

ADR location: `comms/decisions/YYYY-MM-DD-decision-title.md`

6.3 API Documentation

When adding a new API endpoint, update `docs/backend/API-REFERENCE.md`:

- Method + path
- Authentication required (Yes/No)
- Request body schema (Zod type → JSON example)
- Response schema (success + all error codes)

---

7. Security Coding Practices — Drop Fintech Standards

Practice	Rule
SQL queries	NEVER string interpolation. Always parameterized queries (better-sqlite3 `?` placeholders)
Password hashing	bcrypt ONLY with 12 rounds in production. SHA-256 hashes are REJECTED at login
JWT	`jose` library only. Secret via `JWT_SECRET` env var. Fail fast if missing
Input validation	Zod schemas on ALL API route request bodies. Server-side only — never trust client
Pass-through model	NEVER add `balance` column to users. NEVER add `card_number` or `cvv` to cards
Rate limiting	DB-backed rate limiter (not in-memory). Auth: 10/min; General: 60/min
CSRF	CSRF token required on all POST/PATCH/DELETE endpoints
Cookie settings	JWT: httpOnly, SameSite=Strict, Secure in production
Logging	NEVER log passwords, JWT tokens, card numbers, or full phone numbers
Error messages	NEVER expose DB errors, stack traces, or internal state to users
Dependencies	Run `npm audit` before adding any dependency. No HIGH/CRITICAL CVEs

Security review trigger: Any PR touching auth, payments, DB schema, or input validation must be reviewed by Validator agent AND flagged to John (AI Director).

---

8. Performance Coding Practices

Practice	Rule
Database queries	One bounded query per API call where possible. Use JOIN instead of N+1
Pagination	Never load all transactions in history — always paginate (default: 20 per page)
bcrypt	Password max length = 1,000 chars (validation before bcrypt to prevent DoS)
SQLite	Use WAL mode for concurrent reads. Serialize writes (one at a time)
Next.js	Use Server Components for static/cached data; Client Components for interactive UI
Bundle size	Run `npm run build` and check bundle analyzer before UI changes

---

Related Documents

- Developer Onboarding Guide
- Local Development Setup
- Test Strategy
- Definition of Done

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	Approved (AI)
QA Lead	Validator Agent	2026-02-23	Approved (AI)
Tech Lead	John	2026-02-23	Approved
CEO (Alem)	Alem Bašić	TBD	

Local Development Setup: Drop — Fintech Payment App

Local Development Setup: Drop — Fintech Payment App

> **Project:** Drop — Remittance + QR Payments **Version:** 1.0 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Approved **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial local dev setup — Next.js 16 + SQLite + Vitest + Playwright

---

1. System Requirements

Software	Required Version	macOS	Linux	Windows (WSL2)
Node.js	`20.x`	`nvm install 20`	`nvm install 20`	Via WSL2 + nvm
npm	`10.x` (with Node 20)	Included	Included	Included
Git	`2.30+`	Pre-installed	`apt install git`	Via WSL2
flyctl	Latest	`brew install flyctl`	fly.io/docs/flyctl/installing/	WSL2
Playwright browsers	Auto-installed	`npx playwright install`	`npx playwright install`	`npx playwright install`

Hardware minimum: 8GB RAM, 10GB free disk space (SQLite DB is small; bcrypt tests are CPU-intensive) **Recommended:** 16GB RAM, SSD

Not supported: Windows without WSL2 (Next.js dev server requires Unix-compatible environment)

---

2. Quick Start (5 Steps)

```
# 1. Navigate to Drop app
cd ~/ALAI/products/Drop/src/drop-app

# 2. Install dependencies
npm install

# 3. Configure environment
cp .env.example .env.local
# Edit .env.local — see Section 5 for required values

# 4. Set up database and seed
npm run db:migrate && npm run db:seed

# 5. Start development server
npm run dev
# Open: http://localhost:3000
```

Total time (fast path): ~5 minutes

If anything fails, read the full setup in Section 3 or check Common Issues.

---

3. Detailed Setup

3.1 Node Version Manager

```
# Install nvm (macOS with Homebrew)
brew install nvm
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"' >> ~/.zshrc
source ~/.zshrc
```

```
# Install Node 20 (Drop requires 20+ for Next.js 16)
nvm install 20
nvm use 20
nvm alias default 20

# Verify
node --version  # v20.x.x
npm --version   # 10.x.x
```

Auto-switching: The project root contains `.nvmrc` with `20`. If your shell is configured, it switches automatically on `cd` into the project.

3.2 Package Manager Setup

```
# Drop uses npm (not yarn or pnpm)
cd ~/ALAI/products/Drop/src/drop-app
npm install

# Verify installation
npm list --depth=0  # Should show direct dependencies
```

Lockfile: `package-lock.json` — committed to git. Run `npm install` (not `npm install --force`) to respect lockfile.

3.3 Database Setup

Drop uses SQLite for development and testing (no Docker required).

```
# Create/migrate database
cd ~/ALAI/products/Drop/src/drop-app
npm run db:migrate
# Creates: ./local.db (SQLite file)

# Verify migration
npm run db:migrate:status
# Should show all migrations applied

# Seed development data
npm run db:seed
# Inserts: test users, merchants, recipients, exchange rates
```

Verify database is set up:

```
# Check schema compliance
npm run test -- --testPathPattern=db.test.ts
# Should pass: no balance column, no card_number/cvv
```

Note: SQLite file is in `.gitignore`. Each developer has their own local DB.

3.4 Environment Variables

```
# Copy the example file
cp .env.example .env.local
```

Edit `.env.local` with your local values:

Variable	Value for Local Dev	Notes
`DATABASE_URL`	`./local.db`	SQLite file path (relative to app root)
`JWT_SECRET`	Any 32+ char random string	Use `openssl rand -base64 32`
`NEXT_PUBLIC_SERVICE_MODE`	`mock`	ALWAYS `mock` for local dev
`BCRYPT_ROUNDS`	`10`	10 locally (faster); production uses 12
`NEXT_PUBLIC_APP_URL`	`http://localhost:3000`	
`SUMSUB_API_KEY`	`mock-key`	Mocked in dev (no real API calls)
`BAAS_API_KEY`	`mock-key`	Mocked in dev (no real BaaS calls)
`RATE_LIMIT_MAX_AUTH`	`100`	Higher locally to avoid blocking dev flow

Never commit `.env.local`: It's in `.gitignore`. Use `.env.example` for sharing template values.

3.5 Playwright Browser Installation

```
# Install all Playwright browser binaries (required for E2E tests)
npx playwright install

# Or install only Chromium + WebKit (Drop E2E uses these)
npx playwright install chromium webkit
```

3.6 Seed Data

After `npm run db:seed`, these accounts exist locally:

Role	Email	Password	Notes
Consumer (Amir)	`amir@test.alai.no`	`TestDrop123!`	KYC approved, active
Merchant (Ahmet)	`ahmet@test.alai.no`	`TestDrop123!`	Merchant registered
Pending KYC	`pending@test.alai.no`	`TestDrop123!`	kyc_status = pending
Fresh user	Register via UI	—	Create during testing

---

4. Running the Application

Development Server

```
# Start Next.js dev server with hot reload
npm run dev

# Application available at:
# http://localhost:3000         — Landing page
# http://localhost:3000/login   — Login
# http://localhost:3000/api/health  — Health check (JSON)
```

What starts: Next.js dev server on port 3000 with hot reload for all routes (API + UI).

Useful Dev Commands

```
npm run dev           # Start development server
npm run build         # Build production bundle
npm run start         # Run production build locally
npm run lint          # ESLint
npm run type-check    # TypeScript type checking
npm run db:migrate    # Run pending migrations
npm run db:seed       # Seed test data
npm run db:reset      # Drop + recreate + seed (fresh state)
```

---

5. Running Tests

All Tests (Recommended)

```
# Run all Vitest tests (unit + integration + performance)
npm run test

# Expected: 40+ tests passing across 11 test files
```

Unit Tests

```
# Run all Vitest tests
npm run test

# Run in watch mode (re-runs on file save)
npm run test:watch

# Run specific test file
npm run test -- auth.test.ts

# Run tests matching pattern
npm run test -- --grep "bcrypt"
```

Integration Tests

```
# Integration tests use real SQLite DB (auto-created in :memory: or temp file)
npm run test -- api-endpoints.test.ts db.test.ts middleware.test.ts
```

E2E Tests (Playwright)

```
# Requires development server running
npm run dev &  # Or start in separate terminal

# Run all Playwright E2E tests
npx playwright test

# Run specific project
npx playwright test --project=user-flows
npx playwright test --project=full-flows
```

```
npx playwright test --project=input-chaos

# Run with browser visible (useful for debugging)
npx playwright test --headed

# Open Playwright UI mode
npx playwright test --ui
```

Performance Benchmarks

```
# Run api-benchmarks.test.ts
npm run test -- api-benchmarks.test.ts

# Expected outputs (NFR targets):
# bcrypt hash: < 1,000ms
# DB SELECT: < 10ms
# DB INSERT: < 20ms
# 50 concurrent rate limit checks: < 2,000ms total
```

Coverage Report

```
npm run test:coverage

# Report generated at: coverage/index.html
open coverage/index.html
```

---

6. Common Issues & Solutions

Problem	Likely Cause	Solution
`Module not found: next`	Dependencies not installed	Run `npm install`
`Database file not found`	Migrations not run	Run `npm run db:migrate`
`JWT_SECRET is required`	Missing env var	Copy `.env.example` → `.env.local`; set `JWT_SECRET`
`Port 3000 already in use`	Another process	`lsof -ti:3000 | xargs kill`
`Node version mismatch`	Wrong Node version active	`nvm use 20`
bcrypt tests very slow (> 2s each)	BCRYPT_ROUNDS=12 in env	Set `BCRYPT_ROUNDS=10` in `.env.local`

Problem	Likely Cause	Solution
Playwright tests fail: "browser not found"	Browsers not installed	`npx playwright install`
`database is locked`	Concurrent writes in SQLite test	Expected under high concurrency; run tests serially: `npm run test -- -- pool=forks`
`NEXT_PUBLIC_SERVICE_MODE=live` accidentally set	Wrong env var	Always use `mock` locally; `live` requires real BaaS
TypeScript error on `any`	Strict mode	Add proper type; never use `any`

Still stuck? Ask John (AI Director) via Mission Control or #drop Slack channel.

---

7. Project Structure Reference

```
~/ALAI/products/Drop/src/drop-app/
├── src/
│   ├── app/                    # Next.js App Router
│   │   ├── api/                # 26 API routes
│   │   │   ├── auth/           # register, login, logout, me, otp, pin
│   │   │   ├── transactions/   # remittance, qr-payment, list
│   │   │   ├── rates/          # exchange rates (6 corridors)
│   │   │   ├── merchants/      # register, me
│   │   │   ├── cards/          # list, get (feature-flagged)
│   │   │   └── health/         # health check
│   │   └── (app)/              # Protected app pages
│   ├── components/             # Shared UI components
│   ├── lib/                    # Utilities, helpers, auth, db client
│   └── middleware.ts           # Rate limiting, CSRF, auth middleware
├── db/
│   ├── migrations/             # SQL migration files
│   └── seeds/                  # Seed data
├── __tests__/
│   ├── auth.test.ts            # bcrypt, JWT, password validation
│   ├── api-endpoints.test.ts   # All 26 API routes
│   ├── db.test.ts              # Schema compliance (no balance, no CVV)
│   ├── middleware.test.ts      # Rate limiting, auth middleware
│   ├── validation.test.ts      # Input validation, XSS, injection
│   ├── transactions.test.ts    # Fee calculations, transaction logic
│   ├── rates.test.ts           # Exchange rates API
```

```
│   ├── api-benchmarks.test.ts   # Performance benchmarks
│   ├── feature-flags.test.ts    # Feature flag behavior
│   ├── regression-suite.test.ts # Regression tests
│   ├── sumsub-integration.test.ts # KYC mock integration
│   ├── cards-integration.test.ts  # Cards feature (feature-flagged)
│   └── e2e/
│       ├── user-flows.spec.ts   # Registration, login journeys
│       ├── full-flows.spec.ts   # Remittance, QR payment journeys
│       └── input-chaos.spec.ts  # 20+ edge case inputs
├── .env.example                 # Environment variable template
├── vitest.config.ts             # Vitest configuration
├── playwright.config.ts         # Playwright configuration (3 projects)
├── next.config.ts               # Next.js configuration
└── package.json                 # Scripts and dependencies
```

---

8. IDE Configuration

VS Code Settings (`.vscode/settings.json`)

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "typescript.tsdk": "node_modules/typescript/lib",
  "typescript.enablePromptUseWorkspaceTsdk": true
}
```

Recommended Extensions

Extension	ID	Purpose
ESLint	`dbaeumer.vscode-eslint`	Inline linting
Prettier	`esbenp.prettier-vscode`	Format on save
Tailwind CSS IntelliSense	`bradlc.vscode-tailwindcss`	Autocomplete for Tailwind
Playwright Test	`ms-playwright.playwright`	Run/debug Playwright tests

Extension	ID	Purpose
Vitest	`vitest.explorer`	Run/debug Vitest tests
TypeScript Error Lens	`usernamehw.errorlens`	Inline TypeScript errors

Related Documents

- Developer Onboarding Guide
- Coding Standards
- Testing Guide

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	Approved (AI)
Tech Lead	John	2026-02-23	Approved
CEO (Alem)	Alem Bašić	TBD	

Developer Onboarding Guide

Developer Onboarding Guide

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

Welcome

Welcome to **{{PROJECT_NAME}}**! This guide will help you go from zero to a fully working development environment and your first contribution.

If anything in this guide is unclear or out of date, please update it as you go — good documentation is a team responsibility. Your onboarding buddy is **{{BUDDY}}** — reach out to them anytime.

Your first week at a glance:

- Day 1: Set up your environment and get the app running locally
- Day 2-3: Read the architecture overview, explore the codebase
- Day 4-5: Work through your first small ticket with your buddy's guidance

---

1. Prerequisites

Hardware Requirements

Component	Minimum	Recommended
CPU	{{MIN_CPU}}	{{REC_CPU}}
RAM	{{MIN_RAM}}GB	{{REC_RAM}}GB
Disk	{{MIN_DISK}}GB free	SSD with {{REC_DISK}}GB free
OS	{{SUPPORTED_OS}}	

Accounts You Need

Account	Why	How to Get Access
GitHub / GitLab	Code repository	Ask {{GITHUB_ADMIN}}
{{CLOUD_PROVIDER}}	Cloud resources	Request via {{CLOUD_REQUEST_PROCESS}}
{{CI_CD_PLATFORM}}	CI/CD pipeline	Auto-provisioned on org invite
{{ERROR_TRACKER}}	Error monitoring	Ask {{SENTRY_ADMIN}}
{{PROJECT_MGMT}}	Tickets / tasks	Ask {{PM_ADMIN}}
{{COMMUNICATION}}	Team communication	Ask your manager
Vault / Secrets	Development secrets	Ask {{VAULT_ADMIN}}

Expected setup time for all accounts: {{ACCOUNT_SETUP_TIME}} days

Access Requests

Access Type	Contact	Process	SLA
GitHub organization	{{CONTACT}}	{{PROCESS}}	{{SLA}}
Cloud console (read-only dev)	{{CONTACT}}	{{PROCESS}}	{{SLA}}
VPN	{{CONTACT}}	{{PROCESS}}	{{SLA}}
Production read access	{{CONTACT}}	Requires security training first	{{SLA}}

---

2. Development Environment Setup

2.1 macOS Setup

```
# Install Homebrew (if not already installed)
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install required tools
brew install {{TOOLS_LIST}}
# e.g.: brew install git node nvm docker kubectl terraform

# Install nvm (Node version manager)
brew install nvm
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"' >> ~/.zshrc
source ~/.zshrc

# Install required Node version
nvm install {{NODE_VERSION}}
nvm use {{NODE_VERSION}}
nvm alias default {{NODE_VERSION}}
```

2.2 Linux Setup

```
# Update package index
sudo apt update && sudo apt upgrade -y

# Install required tools
sudo apt install -y {{LINUX_TOOLS}}

# Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v{{NVM_VERSION}}/install.sh | bash
source ~/.bashrc

# Install required Node version
nvm install {{NODE_VERSION}}
```

2.3 Windows Setup

```
# Enable WSL2
wsl --install
```

```
# After WSL2 is set up, follow the Linux setup instructions inside WSL2
# Recommended: Use VS Code with Remote WSL extension
```

2.4 Required Software & Versions

Software	Required Version	Install
Node.js	`{{NODE_VERSION}}`	`nvm install {{NODE_VERSION}}`
npm / yarn / pnpm	`{{PKG_MGR_VERSION}}`	Included with Node
Docker	`{{DOCKER_VERSION}}+`	docker.com/get-docker
Docker Compose	`{{COMPOSE_VERSION}}+`	Included with Docker Desktop
{{OTHER_TOOL}}	`{{VERSION}}`	{{INSTALL_CMD}}

Check your versions:

```
node --version    # Should be v{{NODE_VERSION}}
docker --version  # Should be {{DOCKER_VERSION}}+
```

2.5 IDE Setup

Recommended IDE: {{IDE}}

Required extensions:

```
# VS Code — install all recommended extensions
code --install-extension {{EXTENSION_1}}
code --install-extension {{EXTENSION_2}}
code --install-extension {{EXTENSION_3}}
# OR: Open project in VS Code → "Install recommended extensions" prompt will appear
```

Recommended extensions:

Extension	Purpose
{{EXT_1}}	{{PURPOSE}}
{{EXT_2}}	{{PURPOSE}}
{{EXT_3}}	{{PURPOSE}}

2.6 Environment Variables

```
# Copy the example env file
cp .env.example .env

# Fill in your local values
# Ask your onboarding buddy for any values marked {{SECRET}}
```

Key variables to set:

Variable	What to put	Where to get it
`DATABASE_URL`	`postgresql://localhost:5432/{{APP}}_dev`	Local setup
`{{SECRET_VAR}}`	Your dev secret	Ask {{BUDDY}}

---

3. Repository Setup

```
# Step 1: Clone the repository
git clone {{REPO_URL}}
cd {{REPO_NAME}}

# Step 2: Install dependencies
{{INSTALL_CMD}}  # npm install / yarn / pnpm install

# Step 3: Set up the database
{{DB_SETUP_CMD}}  # e.g., npm run db:create && npm run db:migrate

# Step 4: Seed development data
{{SEED_CMD}}  # e.g., npm run db:seed

# Step 5: Configure environment (see 2.6)
cp .env.example .env
# Edit .env with your values

# Step 6: Start the application
{{DEV_CMD}}  # e.g., npm run dev

# Step 7: Open the application
open http://localhost:{{PORT}}
```

---

4. Architecture Overview

High-level architecture: See deployment-architecture.md

In brief:

- **Frontend:** {{FRONTEND_STACK}} — located in `{{FRONTEND_DIR}}`
- **Backend:** {{BACKEND_STACK}} — located in `{{BACKEND_DIR}}`
- **Database:** {{DB_ENGINE}} — schema in `{{SCHEMA_DIR}}`
- **Key external services:** {{EXTERNAL_SERVICES}}

Codebase tour (key directories):

Directory	Purpose
`{{DIR_1}}`	{{PURPOSE_1}}
`{{DIR_2}}`	{{PURPOSE_2}}
`{{DIR_3}}`	{{PURPOSE_3}}
`{{DIR_4}}`	{{PURPOSE_4}}

---

5. Key Documentation Links

Document	Link	Purpose
Architecture	deployment-architecture.md	How the system is structured
API Documentation	{{API_DOCS_LINK}}	API reference
Design System	{{DESIGN_SYSTEM_LINK}}	UI components
Coding Standards	coding-standards.md	How we write code
Test Strategy	test-strategy.md	How we test
CI/CD Pipeline	cicd-pipeline.md	How deployments work
Tech Decisions (ADRs)	{{ADR_DIR}}	Why we made key decisions

---

6. Coding Standards Reference

Standards document: coding-standards.md

Quick reference:

- **Commit messages:** Conventional Commits format (`feat:`, `fix:`, `docs:`, etc.)
- **Branch naming:** `{{BRANCH_FORMAT}}` (e.g., `feature/TICKET-123-short-description`)
- **PR size:** Aim for < {{PR_SIZE}} lines changed per PR
- **Testing requirement:** All PRs require tests for new code

---

7. Git Workflow & Branching Strategy

```
# Create a feature branch
git checkout -b feature/{{TICKET}}-description

# Make changes, then commit
git add {{FILES}}
git commit -m "feat({{SCOPE}}): describe what you did"

# Push your branch
git push origin feature/{{TICKET}}-description

# Open a PR via GitHub/GitLab UI
```

Branching strategy: {{STRATEGY}} — see CI/CD Pipeline

PR requirements:

1. Title follows format: `type(scope): description`
2. PR description filled in (template auto-loaded)
3. Tests included for new code
4. CI pipeline passing before requesting review
5. At least {{REVIEW_COUNT}} reviewer(s) approved

---

8. First PR Checklist

Your first PR goal: A small, low-risk change to confirm everything works end-to-end.

- ☐ Development environment running locally
- ☐ Can run the full test suite: `{{TEST_CMD}}`
- ☐ All tests passing locally

- [ ] First branch created following naming convention
- [ ] Change made (start small — a doc fix or minor improvement is perfect)
- [ ] Tests written for any logic change
- [ ] PR opened with filled description
- [ ] CI pipeline passing on your PR
- [ ] Review requested from your onboarding buddy
- [ ] First PR merged □□

---

9. Team Rituals & Meetings

Meeting	Frequency	When	Purpose	Required
Daily standup	Daily	{{TIME}}	Status, blockers	Yes
Sprint planning	{{SPRINT_FREQ}}	{{TIME}}	Plan upcoming sprint	Yes
Sprint review	{{SPRINT_FREQ}}	{{TIME}}	Demo completed work	Yes
Retrospective	{{SPRINT_FREQ}}	{{TIME}}	Process improvement	Yes
Architecture review	Monthly	{{TIME}}	Tech decisions	Optional
Social / team coffee	{{SOCIAL_FREQ}}	{{TIME}}	Team building	Encouraged

---

10. Communication Channels

Channel	Platform	Purpose
`#{{GENERAL_CHANNEL}}`	{{PLATFORM}}	General team communication
`#{{DEV_CHANNEL}}`	{{PLATFORM}}	Developer discussions, code help
`#{{INCIDENTS_CHANNEL}}`	{{PLATFORM}}	Incident response
`#{{DEPLOYMENTS_CHANNEL}}`	{{PLATFORM}}	Deployment notifications
Direct message	{{PLATFORM}}	Quick questions to teammates
{{EMAIL_LIST}}`	Email	Formal announcements

Rule: Default to public channels over DMs. Questions asked in channels help the whole team.

---

11. Key Contacts

Role	Name	Contact	Help With
Onboarding Buddy	{{BUDDY}}	{{CONTACT}}	Everything in the first week
Tech Lead	{{TECH_LEAD}}	{{CONTACT}}	Technical decisions, architecture
Engineering Manager	{{ENG_MGR}}	{{CONTACT}}	Career, process, team
DevOps / Platform	{{DEVOPS}}	{{CONTACT}}	Infrastructure, deployment, CI
QA Lead	{{QA_LEAD}}	{{CONTACT}}	Testing questions

Related Documents

- Local Development Setup
- Coding Standards
- Developer Offboarding Guide

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Developer Offboarding Guide

Developer Offboarding Guide

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Offboarding Overview

Developer: {{DEVELOPER_NAME}} **Last Day:** {{LAST_DAY}} **Manager:** {{MANAGER}}
Offboarding Coordinator: {{COORDINATOR}} **Security Review:** {{SECURITY_REVIEWER}}

Departure type: Voluntary / Involuntary

Handoff started: {{HANDOFF_START}} **Access revocation deadline:** {{LAST_DAY}} by {{REVOCATION_TIME}}

---

2. Access Revocation Checklist

Code & Version Control

- ☐ **GitHub / GitLab** — remove from organization and all repositories
- ☐ **SSH keys** — remove from all servers and deployment systems
  - `~/.ssh/authorized_keys` on servers

- GitHub SSH keys: Settings > SSH and GPG keys

☐ **GPG signing keys** — revoke from keyserver if used for commit signing

☐ **Personal access tokens** — revoke all tokens in GitHub/GitLab settings

☐ **Webhooks using personal token** — identified and updated to service account

Cloud Infrastructure

☐ **{{CLOUD_PROVIDER}} IAM** — remove user from all IAM groups and roles

☐ **{{CLOUD_PROVIDER}} console access** — deactivate user account

☐ **SSH keys on cloud instances** — remove from all EC2/GCE/VM authorized_keys

☐ **Cloud access keys / credentials** — deactivate and delete

CI/CD & DevOps

☐ **{{CI_PLATFORM}}** — remove from organization (GitHub Actions / GitLab CI / CircleCI)

☐ **Container registry** — remove push/pull credentials

☐ **{{ARTIFACT_REGISTRY}}** — remove user access

☐ **Kubernetes** — remove kubeconfig entries, remove from RBAC

Secrets & Credentials

☐ **{{VAULT_TOOL}} (HashiCorp Vault / 1Password / Vaultwarden)** — remove user, rotate any shared secrets they had access to

☐ **All shared secrets/passwords known to the developer** — rotate immediately (DB passwords, API keys, etc.)

- Database passwords: {{DB_CREDS}}
- API keys accessed: {{API_KEYS}}
- Any others: {{OTHER_CREDS}}

☐ **Environment variables / .env files** — confirm no secrets taken/copied

VPN & Network

☐ **VPN** — revoke VPN certificate / remove user account

☐ **Bastion host** — remove from authorized users

☐ **IP allowlists** — remove their IP if personal device was allowlisted

Third-Party Services

- [ ] **{{SERVICE_1}}** (e.g., Sentry, Datadog, PagerDuty) — remove user
- [ ] **{{SERVICE_2}}** (e.g., Slack, Jira, Confluence) — deactivate account
- [ ] **{{SERVICE_3}}** (e.g., Stripe, AWS Marketplace) — remove user
- [ ] **Email / Google Workspace** — deactivate account, set out-of-office, forward to manager
- [ ] **Password manager (shared vaults)** — remove from shared vaults

Access revocation completion signed off by: {{SECURITY_REVIEWER}} on {{DATE}}

---

3. Knowledge Transfer

Active Projects & Ownership Transfer

Project / Area	Current Status	New Owner	Handoff Complete
{{PROJECT_1}}	{{STATUS}}	{{NEW_OWNER}}	Yes / No
{{PROJECT_2}}	{{STATUS}}	{{NEW_OWNER}}	Yes / No
{{PROJECT_3}}	{{STATUS}}	{{NEW_OWNER}}	Yes / No

Ongoing Work Documentation

Work Item	Ticket	Status	Documentation	New Owner
{{WORK_1}}	{{TICKET}}	{{STATUS}}	{{LINK}}	{{OWNER}}
{{WORK_2}}	{{TICKET}}	{{STATUS}}	{{LINK}}	{{OWNER}}

Documentation written during knowledge transfer:

- [ ] All in-progress PRs reviewed and commented
- [ ] Active branches documented and either merged or closed
- [ ] Ongoing investigations/research notes written up
- [ ] Architecture decisions currently being made: documented as ADRs
- [ ] Pending operational tasks documented in runbooks

Key Contacts & Relationships

Contact	Company / Role	Relationship	Transferred To

{{CONTACT_1}}	{{ORG}}	{{RELATIONSHIP}}	{{NEW_OWNER}}
{{CONTACT_2}}	{{ORG}}	{{RELATIONSHIP}}	{{NEW_OWNER}}

Undocumented Tribal Knowledge Capture

Knowledge transfer sessions scheduled:

Topic	Date	Format	Notes Doc
{{TOPIC_1}}	{{DATE}}	1:1 recording + notes	{{LINK}}
{{TOPIC_2}}	{{DATE}}	Pair programming	{{LINK}}

Capture questions to ask:

1. What breaks in production that only you know how to fix?
2. What shortcuts or workarounds exist in the codebase that aren't documented?
3. What external services have non-obvious quirks?
4. What technical debt exists that you've been meaning to address?
5. Are there any upcoming risks or time bombs in the codebase?
6. Are there any informal agreements or commitments with stakeholders?

---

4. Code Ownership Transfer

CODEOWNERS File Update

```
# Review current CODEOWNERS
cat CODEOWNERS | grep "{{DEVELOPER_HANDLE}}"

# Replace with new owner(s)
# CODEOWNERS update PR: {{PR_LINK}}
```

- ☐ CODEOWNERS file updated and PR merged
- ☐ New owners briefed on their additional responsibilities

PR Review Reassignment

- ☐ Open PRs awaiting their review: reassigned to {{REVIEWER_REPLACEMENT}}
- ☐ In-progress PR review responsibilities communicated to team

On-Call Rotation

- ☐ Removed from on-call rotation in {{ONCALL_TOOL}}
- ☐ On-call schedule updated and communicated
- ☐ On-call runbooks updated to remove their contact information

---

5. Asset Return

Asset	Serial / ID	Return By	Returned	Condition
Laptop	{{SERIAL}}	{{LAST_DAY}}	Yes / No	
Monitor	{{SERIAL}}	{{LAST_DAY}}	Yes / No	
Access card / badge	—	{{LAST_DAY}}	Yes / No	
{{OTHER_ASSET}}	—	{{LAST_DAY}}	Yes / No	

IT returns coordinator: {{IT_CONTACT}}

---

6. Exit Interview Topics

Exit interview conducted by: {{INTERVIEWER}} **Date:** {{DATE}} **Format:** {{FORMAT}}

Topics to cover:

- What did you enjoy most about working here?
- What could we improve for future developers?
- Were there any blockers or frustrations that weren't addressed?
- What did you learn? What skills did you develop?
- Would you recommend working here to others? Why / why not?
- Any concerns about the team or codebase you want to flag before leaving?

Exit interview notes: {{NOTES_LINK}} (confidential — manager access only)

---

7. Final Checklist Sign-Off

Manager Sign-Off

- [ ] All access revocation items completed
- [ ] Knowledge transfer sessions completed
- [ ] Code ownership transferred
- [ ] All projects handed off with documentation
- [ ] Assets returned
- [ ] Exit interview conducted
- [ ] Payroll and HR notified

Manager: {{MANAGER}} | **Date:** {{DATE}} | **Signature:** _____

Developer Sign-Off

- [ ] All work documented and handed off
- [ ] All personal assets retrieved (personal items, any personal accounts)
- [ ] No company data retained on personal devices
- [ ] Exit interview completed

Developer: {{DEVELOPER_NAME}} | **Date:** {{DATE}} | **Signature:** _____

Security Sign-Off

- [ ] All access revocation items verified independently
- [ ] Shared secrets rotated
- [ ] Audit log reviewed for last 30 days — no anomalies

Security Reviewer: {{SECURITY_REVIEWER}} | **Date:** {{DATE}} | **Signature:** _____

Related Documents

- Developer Onboarding Guide
- Coding Standards

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Coding Standards

Coding Standards

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

Purpose

These standards apply to all code in **{{PROJECT_NAME}}**. When automated tools enforce a rule, the tool wins. When in doubt, optimize for readability — the next person reading your code may be you in six months.

Non-negotiable: All new code must pass automated linting and formatting checks before merge. No exceptions.

---

1. Language-Specific Conventions

1.1 Naming Conventions

JavaScript / TypeScript:

Type	Convention	Example
Variables	`camelCase`	`userName`, `isLoading`

Type	Convention	Example
Functions	`camelCase`	`getUserById()`, `validateEmail()`
Classes	`PascalCase`	`UserService`, `PaymentProcessor`
Interfaces	`PascalCase` (no `I` prefix)	`User`, `PaymentResult`
Types	`PascalCase`	`UserId`, `ApiResponse<T>`
Enums	`PascalCase`	`UserRole`, `PaymentStatus`
Constants	`UPPER_SNAKE_CASE`	`MAX_RETRY_COUNT`, `API_BASE_URL`
Files — components	`PascalCase.tsx`	`UserProfile.tsx`
Files — utilities	`camelCase.ts`	`formatDate.ts`, `useAuth.ts`

Python:

Type	Convention	Example
Variables / functions	`snake_case`	`user_name`, `get_user_by_id()`
Classes	`PascalCase`	`UserService`
Constants	`UPPER_SNAKE_CASE`	`MAX_RETRY_COUNT`
Modules / files	`snake_case`	`user_service.py`
Private	`_single_leading_underscore`	`_internal_helper()`

1.2 File Organization

```
src/
├── {{MODULE_1}}/
│   ├── index.ts           # Public exports only
│   ├── {{MODULE_1}}.service.ts   # Business logic
│   ├── {{MODULE_1}}.controller.ts # HTTP handlers
│   ├── {{MODULE_1}}.types.ts     # Types/interfaces
│   ├── {{MODULE_1}}.test.ts      # Unit tests
│   └── {{MODULE_1}}.integration.test.ts
├── shared/                # Shared utilities, no business logic
│   ├── errors/
│   ├── utils/
│   └── types/
└── config/                # Configuration loading
```

Rules:

- One class/component per file
- File name matches the primary export name
- Test files co-located with source files (not in a separate `/test` directory)
- `index.ts` files only for re-exporting — no logic

1.3 Import Ordering

```
// 1. Node built-ins
import { readFileSync } from 'fs';
import path from 'path';

// 2. External dependencies (node_modules)
import express from 'express';
import { z } from 'zod';

// 3. Internal — absolute paths
import { UserService } from '@/services/user.service';
import { logger } from '@/shared/logger';

// 4. Internal — relative paths
import { validateRequest } from '../middleware/validate';
import type { CreateUserDto } from './user.types';
```

Enforced by: `eslint-plugin-import` / `isort` (Python)

1.4 Error Handling Patterns

Pattern: Return typed errors, never throw in business logic unless truly exceptional.

```
// PREFERRED — typed result pattern
type Result<T, E = Error> = { ok: true; value: T } | { ok: false; error: E };

async function createUser(dto: CreateUserDto): Promise<Result<User, ValidationError |
DatabaseError>> {
  const validation = validateUser(dto);
  if (!validation.ok) return { ok: false, error: new ValidationError(validation.message) };
  // ...
}

// ACCEPTABLE — throw only at application boundaries (HTTP handlers)
```

```
// FORBIDDEN — swallow errors silently
try {
  doSomething();
} catch (err) {
  // Never write this without handling the error
}
```

Async/await: Always use `try/catch` in async functions at the controller layer. **Promise chaining:** Avoid — prefer async/await. **Logging:** Log errors with `error` level at the boundary. Do NOT re-log in nested functions.

---

2. Code Formatting

2.1 Formatter

Language	Tool	Config File	Enforced In
TypeScript / JavaScript	{{FORMATTER}}	`{{CONFIG_FILE}}`	CI + pre-commit
Python	Black + isort	`pyproject.toml`	CI + pre-commit
JSON / YAML	Prettier	`.prettierrc`	CI + pre-commit

Key formatting rules:

- Indentation: {{INDENT}}
- Max line length: {{LINE_LENGTH}} characters
- Semicolons: {{SEMICOLONS}}
- Trailing commas: {{TRAILING_COMMAS}}
- Single quotes: {{QUOTES}}

Auto-format on save: Recommended — configure your IDE with the project's formatter settings.

2.2 Linter

Language	Tool	Config	Rules Severity
TypeScript	{{LINTER}}	`{{CONFIG_FILE}}`	Error = blocks CI, Warn = review
Python	Ruff / Flake8	`pyproject.toml`	Error = blocks CI

Linting rules that are errors (block CI):

- `no-unused-variables`
- `no-explicit-any` (TypeScript)
- `no-console` in production code
- Import ordering violations
- Type safety violations

Disable linting inline (sparingly):

```
// eslint-disable-next-line no-console -- debugging production issue, remove before merge
console.log('Debug:', data);
```

Every inline disable must have a comment explaining why.

2.3 Editor Config

```
# .editorconfig (root of repo)
root = true

[*]
indent_style = {{INDENT_STYLE}}
indent_size = {{INDENT_SIZE}}
end_of_line = lf
charset = utf-8
trim_trailing_whitespace = true
insert_final_newline = true
```

---

3. Git Conventions

3.1 Commit Message Format

Standard: Conventional Commits

Format:

```
<type>(<scope>): <description>

[optional body]
```

```
[optional footer(s)]
```

Types:

Type	When to Use
`feat`	New feature
`fix`	Bug fix
`docs`	Documentation only
`style`	Formatting changes (no logic change)
`refactor`	Code change that neither fixes a bug nor adds a feature
`test`	Adding or updating tests
`chore`	Build system, dependency updates, CI changes
`perf`	Performance improvements
`ci`	CI/CD configuration changes

Examples:

```
feat(auth): add magic link login flow
fix(cart): prevent double-charge on network retry
docs(api): update user endpoint examples
chore: upgrade typescript to 5.3
```

Breaking changes:

```
feat(api)!: rename user endpoint from /users to /accounts

BREAKING CHANGE: All clients must update API calls from /api/users to /api/accounts
```

Enforced by: commitlint + husky pre-commit hook

3.2 Branch Naming Convention

Type	Pattern	Example
Feature	`feature/{{TICKET}}-short-description`	`feature/AUTH-123-magic-link`
Bug fix	`fix/{{TICKET}}-short-description`	`fix/CART-456-double-charge`
Hotfix	`hotfix/{{TICKET}}-short-description`	`hotfix/SEC-789-xss-fix`
Release	`release/v{{VERSION}}`	`release/v2.4.0`
Chore	`chore/{{TICKET}}-short-description`	`chore/DEP-012-upgrade-react`

Rules:

- Lowercase only
- Hyphens, not underscores
- Include ticket number
- Keep description short (< 5 words)

3.3 PR Title & Description Format

Title format: Same as commit message format: `type(scope): description`

PR template: `.github/pull_request_template.md` — auto-loaded on PR creation

Required PR description sections:

1. **What**: What does this PR do? (1-3 bullet points)
2. **Why**: Why was this change made? (context, ticket link)
3. **How to test**: How should the reviewer verify it works?
4. **Screenshots**: Required for UI changes
5. **Checklist**: Pre-merge checklist (auto from template)

3.4 PR Size Guidelines

Size	Lines Changed	Status	Action
Small	< 200	Ideal	Review same day
Medium	200-500	Acceptable	Review within 24h
Large	500-1000	Needs justification	Split if possible
Extra Large	> 1000	Exceptional only	Must be pre-approved

Tips for keeping PRs small:

- One PR per feature/fix
- Separate refactoring from feature work
- Use feature flags to deploy incomplete features

---

4. Code Review Guidelines

4.1 What to Look For

Reviewers should check:

- [ ] Logic is correct and handles edge cases
- [ ] Tests cover the change (and edge cases)
- [ ] No obvious security issues (input validation, auth checks, SQL injection)
- [ ] Error handling is appropriate
- [ ] Performance: no N+1 queries, no unbounded operations
- [ ] Code follows naming and organization standards
- [ ] Documentation updated if behavior changed

Reviewers should NOT block on:

- Personal style preferences that aren't in the standards
- Minor refactors not related to the PR's scope
- Hypothetical future requirements

4.2 Review Turnaround Expectations

PR Type	First review	Re-review after changes
Critical / Hotfix	Within {{HOTFIX_SLA}} hours	Within {{HOTFIX_RESLA}} hours
Standard	Within {{STANDARD_SLA}} business hours	Within {{STANDARD_RESLA}} hours

4.3 Approval Requirements

Branch	Required Approvals	Notes
`main` / `develop`	{{MAIN_APPROVALS}} (including CODEOWNER)	
Feature branches	{{FEATURE_APPROVALS}}	
Hotfix	{{HOTFIX_APPROVALS}} (emergency: 1)	Post-merge full review

4.4 Constructive Review Feedback

Use these prefixes to set expectations:

- `nit:` — Minor issue, not a blocker: `nit: prefer const here`
- `question:` — Need clarification, not a change request: `question: why did you choose X over Y?`
- `suggestion:` — Improvement idea, not required: `suggestion: this could be simplified with X`
- `blocker:` — Must be fixed before merge: `blocker: this allows SQL injection`

Tone guidelines:

- Comment on code, not the author
- Explain the why: "This can cause N+1 queries, which will slow down under load" > "Wrong"
- Acknowledge good code: "Nice approach here — much cleaner than what we had"
- Be kind, be direct, be specific

---

5. Testing Standards

5.1 Test Naming Conventions

```
// Format: it('should [expected behavior] when [condition]')
it('should return 404 when user does not exist', async () => { ... });
it('should hash the password before storing', async () => { ... });
it('should throw ValidationError when email is invalid', async () => { ... });

// Describe blocks for grouping
describe('UserService', () => {
  describe('createUser', () => {
    it('should create user with valid input', async () => { ... });
    it('should throw when email already exists', async () => { ... });
  });
});
```

5.2 Test File Organization

```
src/
├── users/
│   ├── user.service.ts
│   ├── user.service.test.ts      # Unit tests
│   └── user.service.integration.test.ts  # Integration tests
```

5.3 Mocking Guidelines

```
// PREFERRED: Mock at the module level
jest.mock('../external/email-service');
```

```
// PREFERRED: Use test doubles that match the interface
const mockEmailService: EmailService = {
  send: jest.fn().mockResolvedValue({ id: 'msg-123' }),
};

// AVOID: Over-mocking internals
// AVOID: Tests that test mocks, not behavior
```

5.4 Coverage Requirements

Layer	Lines	Branches	Notes
Business logic	≥ {{BIZ_COV}}%	≥ {{BIZ_BRANCH}}%	Strictly enforced
Controllers / handlers	≥ {{CTRL_COV}}%	≥ {{CTRL_BRANCH}}%	
Utilities	≥ {{UTIL_COV}}%		
Overall minimum	≥ {{MIN_COV}}%		CI gate

---

6. Documentation Standards

6.1 When to Add Comments

```
// GOOD — explains WHY (not obvious from code)
// Retry up to 3 times because the payment API has transient failures on high load
for (let i = 0; i < 3; i++) { ... }

// BAD — restates WHAT (obvious from code)
// Loop 3 times
for (let i = 0; i < 3; i++) { ... }

// GOOD — documents non-obvious behavior
// Note: This function mutates the input array for performance. Callers must not
// pass arrays they intend to use afterwards.
function sortInPlace(arr: number[]): void { ... }
```

Comment when:

- The reason for a decision is non-obvious

- A workaround exists for an external library bug (include the bug link)
- The code is intentionally doing something that looks wrong

Don't comment when:

- The code explains itself
- The test explains the expected behavior

6.2 JSDoc / Docstring Format

```
/**
 * Creates a new user account and sends a verification email.
 *
 * @param dto - User creation data (email must be unique)
 * @returns The created user object, or a typed error
 * @throws {RateLimitError} If the creation rate limit is exceeded
 * @example
 * const result = await createUser({ email: 'user@example.com', name: 'Alice' });
 * if (!result.ok) handleError(result.error);
 */
async function createUser(dto: CreateUserDto): Promise<Result<User>> { ... }
```

6.3 README Requirements

Every repository/service MUST have a README with:

1. **What** — One sentence describing the service
2. **Quick start** — How to run it locally (3-5 commands)
3. **Key commands** — `dev`, `test`, `build`, `lint`
4. **Links** — Architecture doc, API docs, team channel

6.4 ADR Requirements

Write an Architecture Decision Record (ADR) when:

- Choosing a new major dependency
- Changing the tech stack
- Making a significant architectural change
- Explicitly choosing NOT to do something common

ADR location: `{{ADR_DIR}}/YYYY-MM-DD-decision-title.md` **ADR template:** See `{{ADR_TEMPLATE}}`

---

7. Security Coding Practices

Practice	Rule
Input validation	Validate ALL external input at system boundaries. Use schema validation (Zod / Joi / Pydantic)
SQL queries	NEVER use string interpolation in SQL. Always use parameterized queries / ORM
Authentication	NEVER implement auth from scratch. Use battle-tested libraries
Secrets	NEVER hardcode secrets. Use environment variables sourced from secret manager
Logging	NEVER log PII, passwords, tokens, or payment data
Dependencies	Review new dependencies for known CVEs before adding
Error messages	NEVER expose internal details (stack traces, DB errors) to users
HTML output	Always escape user-generated content. Use framework's built-in escaping
File uploads	Validate type, size, and content. Never execute uploaded files

Security review trigger: Any PR touching authentication, authorization, payments, or user data must be reviewed by {{SECURITY_REVIEWER}}.

---

8. Performance Coding Practices

Practice	Rule
Database queries	Avoid N+1 — use `include`/`JOIN` or batch loading. Profile with query analyzer
Pagination	Never load unbounded lists. Always paginate or stream
Caching	Cache expensive computations and external API calls. Define TTL explicitly
Async	Use async I/O for all I/O operations. Never block the event loop
Indexes	Add database indexes for any column used in `WHERE` or `ORDER BY` at scale
Large payloads	Compress API responses. Stream large files rather than buffering in memory

Related Documents

- Developer Onboarding Guide
- Local Development Setup
- Test Strategy
- Definition of Done

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Local Development Setup

Local Development Setup

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. System Requirements

Software	Required Version	macOS	Linux	Windows (WSL2)
{{RUNTIME}}	`{{VERSION}}`	`brew install {{RUNTIME}}`	`apt install {{RUNTIME}}`	Via WSL2
Docker Desktop	`{{DOCKER_VERSION}}+`	docker.com	docs	docs
Git	`2.30+`	Pre-installed	`apt install git`	Via WSL2
{{OTHER_TOOL}}	`{{VERSION}}`	`brew install {{OTHER_TOOL}}`	`{{LINUX_INSTALL}}`	{{WINDOWS_INSTALL}}

Hardware minimum: {{MIN_RAM}}GB RAM, {{MIN_DISK}}GB free disk space **Recommended:** {{REC_RAM}}GB RAM, SSD

Not supported: Windows without WSL2

---

2. Quick Start (5 Steps)

```
# 1. Clone
git clone {{REPO_URL}} && cd {{REPO_NAME}}

# 2. Install dependencies
{{INSTALL_CMD}}

# 3. Configure
cp .env.example .env
# Edit .env — see Section 5 for required values

# 4. Run (with Docker for dependencies)
{{START_CMD}}

# 5. Verify
open http://localhost:{{PORT}}
# Should see: {{EXPECTED_LANDING}}
```

Total time (fast path): ~{{QUICKSTART_TIME}} minutes

If anything fails, read the full setup in Section 3 or check Common Issues.

---

3. Detailed Setup

3.1 Runtime Version Manager

Tool: {{VERSION_MANAGER}}

```
# Install version manager
{{VM_INSTALL_CMD}}

# Install required runtime version
{{VM_USE_CMD}} {{RUNTIME_VERSION}}

# Set as default
{{VM_DEFAULT_CMD}} {{RUNTIME_VERSION}}

# Verify
{{RUNTIME}} --version  # Should output: {{RUNTIME_VERSION}}
```

Auto-switching: The project root contains `.{{VERSION_FILE}}` (e.g., `.nvmrc`, `.python-version`). If your shell is configured, it switches automatically on `cd` into the project.

3.2 Package Manager Setup

```
# The project uses {{PKG_MANAGER}}
# Install it if not available:
{{PKG_MANAGER_INSTALL}}

# Install project dependencies
cd {{REPO_NAME}}
{{INSTALL_CMD}}

# Verify installation
{{INSTALL_VERIFY_CMD}}
```

Lockfile: `{{LOCKFILE}}` — committed to git. Do NOT delete it. Run `{{INSTALL_CMD}}` (not `{{FORCE_INSTALL_CMD}}`) to respect lockfile.

3.3 Database Setup

Option A: Docker (recommended)

```
# Start database container
docker-compose up -d db

# Verify database is running
{{DB_CHECK_CMD}}

# Run migrations
{{MIGRATE_CMD}}

# Seed development data
{{SEED_CMD}}
```

Option B: Local database install

```
# Install PostgreSQL (example)
brew install postgresql@{{PG_VERSION}}  # macOS
# or: sudo apt install postgresql-{{PG_VERSION}}  # Linux
```

```
# Start database
{{DB_START_LOCAL_CMD}}

# Create database
createdb {{DB_NAME}}_dev

# Run migrations
{{MIGRATE_CMD}}

# Seed development data
{{SEED_CMD}}
```

Verify database is set up:

```
{{DB_VERIFY_CMD}}
# Should output: {{DB_VERIFY_EXPECTED}}
```

3.4 Environment Variables

```
# Copy the example file
cp .env.example .env
```

Edit `.env` with your local values:

Variable	Value for Local	Notes
`DATABASE_URL`	`{{LOCAL_DB_URL}}`	Update with your local DB creds
`REDIS_URL`	`redis://localhost:6379`	Default if using Docker
`PORT`	`{{PORT}}`	
`NODE_ENV`	`development`	
`{{SECRET_VAR}}`	—	Ask your onboarding buddy

Never commit `.env`: It's in `.gitignore`. Use `.env.example` for sharing template values.

3.5 SSL Certificates (If Needed Locally)

```
# Install mkcert
brew install mkcert    # macOS
# or: apt install mkcert  # Linux
```

```
# Install local CA
mkcert -install

# Generate cert for localhost
mkcert localhost 127.0.0.1 ::1

# Move certs to project (check .env for expected paths)
mv localhost+2.pem {{CERT_PATH}}
mv localhost+2-key.pem {{KEY_PATH}}
```

3.6 Seed Data Loading

```
# Load fixture data (fast, for unit/integration tests)
{{FIXTURE_CMD}}

# Load realistic development data (slower, ~{{SEED_SIZE}} records)
{{DEV_SEED_CMD}}

# Reset to clean state
{{RESET_CMD}}
```

Seeded accounts after running seed:

Role	Email	Password	Notes
Admin	`admin@dev.local`	`{{SEED_PWD}}`	Full access
User	`user@dev.local`	`{{SEED_PWD}}`	Standard user

---

4. Running the Application

Development Server

```
# Start development server (with hot reload)
{{DEV_CMD}}

# Application available at:
http://localhost:{{PORT}}
```

What starts: {{DEV_WHAT_STARTS}}

Watch Mode / Hot Reload

Hot reload is enabled by default in development. When you save a file:

- **Backend:** {{BACKEND_RELOAD}}
- **Frontend:** {{FRONTEND_RELOAD}}

Debug Mode

```
# Start with Node.js debugger
{{DEBUG_CMD}}

# Then attach your IDE debugger to port {{DEBUG_PORT}}
```

VS Code launch config: `.vscode/launch.json` — includes pre-configured debug targets.

Mobile Development (If Applicable)

```
# iOS Simulator (macOS only)
{{IOS_CMD}}

# Android Emulator
{{ANDROID_CMD}}
```

Requires: Xcode (iOS), Android Studio (Android)

---

5. Running Tests

Unit Tests

```
# Run all unit tests
{{UNIT_CMD}}

# Run tests in watch mode (re-runs on file save)
{{UNIT_WATCH_CMD}}
```

```
# Run specific test file
{{UNIT_CMD}} {{TEST_FILE_PATH}}

# Run tests matching pattern
{{UNIT_CMD}} --grep "{{PATTERN}}"
```

Integration Tests

```
# Requires database and Redis running (use Docker)
docker-compose up -d db redis

# Run integration tests
{{INT_CMD}}
```

E2E Tests

```
# Requires full application stack running
{{DEV_CMD}} &  # Or start in separate terminal

# Run E2E tests (headless)
{{E2E_CMD}}

# Run E2E with browser visible (debugging)
{{E2E_HEADED_CMD}}

# Run specific E2E test
{{E2E_CMD}} --grep "{{PATTERN}}"
```

Test Coverage Report

```
{{COV_CMD}}

# Report generated at: {{COV_REPORT_PATH}}
# Open in browser:
open {{COV_REPORT_PATH}}/index.html
```

---

6. Common Issues & Solutions

Problem	Likely Cause	Solution
`Module not found`	Dependencies not installed	Run `{{INSTALL_CMD}}`
`Database connection refused`	DB not running	Run `docker-compose up -d db`
`Port {{PORT}} already in use`	Another process using the port	`lsof -ti:{{PORT}} | xargs kill`
`Permission denied: .env`	File permission issue	`chmod 600 .env`
`Node version mismatch`	Wrong Node version active	`{{VERSION_MANAGER}} use {{NODE_VERSION}}`
Slow performance on Docker (macOS)	Docker Desktop volume mounts	Enable VirtioFS in Docker Desktop settings
`EACCES: permission denied, mkdir node_modules`	npm global prefix issue	See npm fix
Tests failing with timeout	Integration test DB not running	`docker-compose up -d`
`SSL: certificate verify failed`	Missing cert or mkcert not installed	Re-run mkcert setup (Section 3.5)
`Migration already applied`	Stale migration state	`{{MIGRATION_ROLLBACK_CMD}}` then re-run

Still stuck? Ask in `#{{DEV_CHANNEL}}` or ping your onboarding buddy.

---

7. Docker Setup (Alternative)

7.1 docker-compose.yml Reference

```
# Start all services
docker-compose up

# Start only backend services (db, redis, etc.)
docker-compose up db redis

# Rebuild after Dockerfile changes
docker-compose up --build

# Stop and remove containers
docker-compose down
```

```
# Stop and remove containers + volumes (reset state)
docker-compose down -v
```

Services defined in `docker-compose.yml`:

Service	Port	Description
`app`	`{{PORT}}`	Main application
`db`	`{{DB_PORT}}`	PostgreSQL database
`redis`	`6379`	Redis cache
`{{OTHER}}`	`{{PORT}}`	{{DESCRIPTION}}

7.2 Volume Mounts

Local Path	Container Path	Purpose
`.`	`/app`	Source code (hot reload)
`db-data`	`/var/lib/postgresql/data`	Database persistence
`node_modules`	`/app/node_modules`	Isolated from host

7.3 Port Mappings

Host Port	Container Port	Service
`{{PORT}}`	`{{PORT}}`	Application
`{{DB_PORT}}`	`5432`	PostgreSQL
`6379`	`6379`	Redis

---

8. IDE Configuration

8.1 Recommended Extensions (VS Code)

```
# Install all at once
cat .vscode/extensions.json | jq '.recommendations[]' | xargs -I{} code --install-extension {}
```

Extension	ID	Purpose
{{EXT_1}}	`{{ID}}`	{{PURPOSE}}

Extension	ID	Purpose
{{EXT_2}}	`{{ID}}`	{{PURPOSE}}
{{EXT_3}}	`{{ID}}`	{{PURPOSE}}
ESLint	`dbaeumer.vscode-eslint`	Inline linting
Prettier	`esbenp.prettier-vscode`	Format on save

8.2 Debug Configurations

The project includes `.vscode/launch.json` with these pre-built debug targets:

Configuration	What it does
`Debug: API Server`	Starts API with debugger attached on port {{DEBUG_PORT}}
`Debug: Tests`	Runs test suite with breakpoints supported
`Attach to Process`	Attach debugger to running process

8.3 Workspace Settings

`.vscode/settings.json` (committed to repo):

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "{{FORMATTER_ID}}",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "typescript.tsdk": "node_modules/typescript/lib"
}
```

Related Documents

- Developer Onboarding Guide
- Coding Standards
- Environment Configuration

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			