Agent Anti-Hallucination Rules

> Last Verified: 2026-02-17 | Owner: John

Agent Anti-Hallucination Rules

Source: FjordConsulting simulation test (2026-02-06) **Findings:** 75/100 accuracy — 25% hallucinated details

---

Rule 0: VERIFY BEFORE CLAIMING (NAJVAŽNIJE PRAVILO)

NIKAD ne tvrdi da nešto postoji ili ne postoji bez da PROVJERIS.

Prije nego kažeš "fajl ne postoji" → pokreni `ls` ili `cat`. Prije nego kažeš "tool ne radi" → pokreni ga. Prije nego kažeš "nema komunikacije" → provjeri kanal.

```
# ISPRAVNO: provjeri pa tvrdi
ls ~/path/to/file.txt          # postoji? → ONDA reci "postoji"
cat ~/path/to/config.json      # čitljiv? → ONDA reci šta piše

# POGREŠNO: tvrdi bez provjere
"Taj fajl ne postoji"          # ← KAKO ZNAŠ? Jesi li pokrenuo ls?
"Sistem nije konfigurisan"     # ← KAKO ZNAŠ? Jesi li provjerio?
```

Primjer greške (2026-02-06): Edita tvrdila da 5 fajlova "ne postoji nigdje u Dropboxu" — `ls` na istoj mašini je pokazao da SVI postoje. Nije provjerila filesystem prije nego je donijela zaključak.

Pravilo: Ako nisi pokrenuo komandu koja DOKAZUJE tvoju tvrdnju — ne tvrdi.

Rule 1: TBD > Hallucination

If you don't know a specific value, write `TODO:` or `TBD` instead of inventing it.

NEVER invent:

- Cryptographic hashes, keys, or salts (write `// TODO: generate with bcrypt/openssl`)
- API names or algorithm names you're not 100% sure exist (write `// TODO: verify API exists`)
- Market prices, hourly rates, or budgets without source data (write `TBD — requires market research`)
- Framework-specific APIs without verification (write `// TODO: verify available in iOS X+`)
- Company names, contact emails, or legal citations without verification

ALWAYS mark uncertainty:

```
// TODO: verify — this API may not exist in iOS 17
// TBD: Norwegian developer hourly rate (estimate 800-1500 NOK/h, needs verification)
// PLACEHOLDER: replace with real bcrypt hash before deployment
```

Rule 2: Cross-File Consistency

Before writing code that references another file, READ that file first.

Mandatory checks:

- If you write a DB query with a status value → READ the schema CHECK constraint
- If you add a package.json dependency → IMPORT it somewhere or don't add it
- If you write an API spec endpoint → IMPLEMENT the route handler
- If you reference a type/enum → VERIFY it exists in the type definitions

Anti-pattern (caught in test):

```
// websocket.ts writes status='online'
// BUT schema.sql only allows: 'active', 'away', 'offline', 'deactivated'
// Result: runtime crash
```

Rule 3: No Phantom Dependencies

Every dependency in package.json / Package.swift MUST be:

1. Actually imported in at least one source file

2. Actually used (not just imported)

If you add a dependency for future use, comment it:

```
// "ioredis": "^5.4.2"  // Phase 2: session caching — NOT YET IMPLEMENTED
```

Rule 4: Health Checks Must Be Real

Never hardcode health check responses. Always verify:

```
// BAD (caught in test):
return { database: 'up', redis: 'up' }; // lies

// GOOD:
const dbOk = await pool.query('SELECT 1').then(() => true).catch(() => false);
return { database: dbOk ? 'up' : 'down' };
```

Rule 5: Spec-Implementation Parity

If you write a specification (API spec, design doc), track implementation status:

```
## Endpoints

| Endpoint | Status |
|----------|--------|
| POST /auth/login | IMPLEMENTED |
| POST /auth/register | IMPLEMENTED |
| GET /groups | NOT YET IMPLEMENTED |
| POST /push/register | PHASE 2 |
```

Never present unimplemented features as done.

Rule 6: Budget and Timeline Reality Checks

When estimating costs or timelines:

- State assumptions explicitly ("assuming Norwegian senior developer at 1000-1200 NOK/h")
- If you don't have market data, write: "TBD — requires market research for [country/region]"
- Add 30-50% buffer to timeline estimates
- Never present optimistic estimates as realistic

Rule 7: Placeholder Code Must Scream

Mock implementations must be OBVIOUSLY fake:

```
// BAD (caught in test): looks like real encryption but is base64
func encrypt(_ data: Data) -> Data {
    return data.base64EncodedData() // This is NOT encryption
}

// GOOD: impossible to miss
func encrypt(_ data: Data) -> Data {
    fatalError("PLACEHOLDER: Real encryption not yet implemented. Integrate libsignal-client-swift before shipping.")
}
```

Rule 9: Config/Schema Changes Require Docs (dodano 2026-02-12)

ROOT CAUSE: John mijenjao hooks format u settings.json bez čitanja docs. Haiku validator potvrdio pogrešan format.

PRAVILA:

1. **NIKAD** mijenjaj config format (hooks, settings, schema) bez čitanja oficijelne dokumentacije
2. **WebFetch/WebSearch OBAVEZAN** za: hooks format, API schema, config migration, breaking changes
3. **Validator agent NIJE rubber stamp** — mora imati docs URL ili spec file kao input
4. **Anti-pattern:** "Provjeri da sam dobro uradio" bez davanja izvora istine agentu

Primjer greške (2026-02-12):

```
// POGREŠNO (John napisao):
"matcher": {}        // objekt — Claude Code expects string

// ISPRAVNO (iz docs):
"matcher": "*"       // regex string — matcha sve toolove
"matcher": "Bash"    // regex string — matcha samo Bash
```

Workflow za config promjene:

```
1. WebFetch oficijelnu docs stranicu
2. Pročitaj schema/format
3. Tek onda mijenjaj fajl
4. Validator dobije docs URL kao input za nezavisnu provjeru
```

Hallucination Examples from Test

What agent wrote	Reality	Root cause
"Sesame Algorithm"	Does not exist	Invented name for concept
`contentSecureLevel` iOS 17 API	Does not exist	Invented API
bcrypt hash `$2b$12$vJ4...`	Invalid hash	Generated random string
150 NOK/h developer rate	Norway is 800-1500 NOK/h	No market context
Redis "up" in health check	Redis never connected	Hardcoded response
`status = 'online'`	DB enum has no 'online'	Didn't read schema
ReportCo AS "WON client"	Demo data, firma ne postoji	Test data bez is_test flag
FitLife AS "50K deal"	Demo data, firma ne postoji	Lead→WON bez ugovora
FjordConsulting "200K proposal"	Kontakt ne postoji u contacts.db	Phantom pipeline entry
Revenue plan 774K Q1	Realno 544K (230K phantom)	Auto-generated bez review

---

Rule 8: Demo/Test Data Protection (dodano 2026-02-11)

ROOT CAUSE: John kreirao demo podatke (FitLife, ReportCo) 2026-02-04 za AIOS testing. Bez oznake. Propagirali kroz leads→contacts→invoices→revenue plan kao REALNI klijenti 7 dana.

PRAVILA:

1. **NIKAD** kreiraj test/demo podatke u production bazama bez jasne oznake
2. **Flag marking:** Svaki test zapis MORA imati polje `is_test=true` ili ekvivalent
3. **Name prefix:** Test firme MORAJU imati prefix `TEST:` ili `DEMO:` u imenu
4. **Email:** Test kontakti koriste `@test.local` domenu, NIKAD realne domene
5. **Human gate:** Lead koji ide u WON stage MORA imati:
   - Potpisan ugovor, ILI
   - Eksplicitna potvrda od Alema
6. **Revenue plan:** NIKAD auto-generate i publish — uvijek draft + human review
7. **Invoice kreacija:** Zahtijeva referencu na potpisan ugovor ili Alem approval
8. **Contacts baza:** Novi kontakt MORA imati verificiran email i dokumentovan source
9. **Cross-check:** Prije uključivanja firme u revenue plan, verificiraj da postoji u contacts.db SA realnom komunikacijom

Ako kreiraš test data za demo:

```
# ISPRAVNO:
node contacts.js add "TEST: DemoFirma" "test@test.local" --notes "DEMO DATA for testing only"

# POGREŠNO:
node contacts.js add "ReportCo AS" "cfo@reportco.no"  # izgleda realno, propagira
```

Revision #4
Created 2026-02-17 22:14:44 UTC by John
Updated 2026-07-05 20:00:45 UTC by John