Development & Onboarding Guide

This document is the single source of truth for developing, maintaining, and safely operating the True Competency platform.

It is written for:

- Current and future developers
- Auditors / reviewers who need to understand *how* the system is operated safely

True Competency follows a database-driven, security-first architecture.

---

1. High-Level Architecture

- Frontend: Next.js (App Router), TypeScript, Tailwind
- Backend: Supabase (PostgreSQL, Auth, RLS, RPC)
- Deployment: Vercel
- Auth & Security: Supabase Auth + Row Level Security (RLS)

---

2. Environments Overview

There are three environments, each with a strict purpose.

Environment	Purpose	Database *(Supabase)*	Domains
Local Dev	Development on local machine	Staging	http://localhost:3000/
Staging	Testing, QA, Demos	Staging	https://staging-true-competency.vercel.app/
Production	Real users	Production	https://truecompetency.com/

Local development must always point to staging, never production.

---

3. Environment Variables

Local Development

Local development uses .env.local only.
`.env.local` template:

```
NEXT_PUBLIC_ENV=staging
```

```
NEXT_PUBLIC_SUPABASE_URL=https://<STAGING_REF>.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=<STAGING_ANON_KEY>
```

Vercel

- Staging domain → staging Supabase keys
- Production domain → production Supabase keys

Never reuse keys across environments.

---

4. Database Structure (Conceptual)

Core Data

- `profiles`
- `countries`
- `competencies`
- `competency_questions`
- `question_options`

Progress / Runtime Data

- `competency_assignments`
- `student_answers`
- `student_competency_overrides`

Governance & Proposals

- `competencies_stage`
- `competency_questions_stage`
- `competency_question_options_stage`
- `committee_votes`
- `committee_question_votes`

---

5. Roles & Permissions

Roles

- Trainee — takes tests, sees own progress
- Instructor — assigns & approves competencies
- Committee — proposes competencies & questions, votes
- Admin — elevated governance, not a frontend role

Security Model

- RLS enabled on every table
- Writes are always validated by:
  - auth.uid()
  - role checks via profiles
  - or SECURITY DEFINER RPCs

---

6. SQL, Migrations & Schema Changes

All schema changes must live in `supabase/migrations/`

Workflow (Without Docker)

1. Create a new migration file
2. Apply it manually in Supabase SQL Editor (staging)
3. Test staging
4. Apply same SQL to production
5. Commit migration file

Order matters! Never skip staging.

- *Production schema was captured as a baseline migration*
- *All future changes build on that*

---

7. Analytics

- Vercel Analytics is used
- Privacy-preserving
- No PII tracking
- No Google Analytics

---

8. Git & Branching

- main → production
- develop → staging

Rules:

- All features go to develop first

- Only tested code reaches main
- Conventional commits required

Example:

```
chore(db): add baseline schema migration
feat(committee): allow proposing test questions
```

---

9. What NOT To Do

- Do not bypass RLS for convenience
- Do not add frontend-only security
- Do not test on production
- Do not keep temporary scripts or CSVs
- Do not reuse prod keys locally
