■ Coding Standards Document

1. Introduction

This document defines the coding standards for the Quiz Generation Platform, ensuring consistency, maintainability, readability, and security across the project.

Applies to:

- Frontend (Next.js + TypeScript)
- Backend (Python + FastAPI + Groq integration)
- Database (Supabase)
- API Testing (Postman, FastAPI Swagger UI)
- Project Tracking (Trello)

2. General Principles

- Code must be readable, modular, and consistent.
- Follow DRY, KISS, and YAGNI principles.
- Always validate inputs and sanitize outputs.
- Sensitive data must be stored in .env files (never hardcoded).
- All new work must be linked to a Trello card.

3. Project Structure

- Backend (FastAPI + Python): routers, services, models, db, tests, dependencies.
- Frontend (Next.js + TypeScript): components, pages, hooks, styles, utils.
- Documentation: backlog, architecture, reference files.

4. Naming Conventions

- Python: snake_case for files, functions, variables. PascalCase for classes. UPPER_SNAKE_CASE for constants.
- TypeScript/React: PascalCase for components, camelCase for variables/functions, UPPER_SNAKE_CASE for constants.

5. Formatting & Style

- Python: PEP8, Black (formatter), Flake8 (linter). Indentation 4 spaces, max line length 120.
- TypeScript: ESLint + Prettier. Indentation 2 spaces, max line length 100.

6. API Standards

- Endpoints must be RESTful and versioned → /api/v1/quiz.
- Use Pydantic models for request/response validation.
- Return consistent JSON format (status, data, error).
- Document APIs with Swagger UI.
- Verify endpoints with Postman before merging.

7. Database (Supabase)

- Tables: plural, snake_case.
- Columns: snake_case.
- Primary keys: id (UUID).
- Foreign keys: _id.
- Enable RLS (Row Level Security).

8. Error Handling & Logging

- Backend: try/except, structured error JSON.
- Frontend: toast/alert for users, console logs in dev mode.

9. Testing

- Backend: Pytest.
- Frontend: Jest + React Testing Library.
- API: Postman + Swagger UI.
- Coverage: Minimum 70% before merging.

10. Version Control (GitHub)

- Branch naming: feat/..., fix/...
- Commit messages: feat: ..., fix: ...
- Pull Requests: Must reference Trello card, require 1 reviewer approval.

11. Trello Workflow

- Columns: Backlog → To Do → In Progress → In Review → Done.
- Each card includes description, assignee, due date, checklist.

12. Security

- Use .env for secrets (never commit).
- Sanitize and validate inputs.
- Enable Supabase RLS.
- Use JWT/OAuth for auth (future).
- HTTPS in production.

13. Code Review Checklist

- Naming conventions followed.
- Linter/formatter applied.
- Proper error handling.
- Tests written & passing.
- API docs updated.
- Trello card linked.

14. Tools & Automation

- Linters/Formatters: ESLint, Prettier, Black, Flake8.
- Testing: Jest, Pytest, Postman.
- Docs: Swagger UI, Postman Collections.
- Tracking: Trello.
- CI/CD: GitHub Actions.