Inquizitive – Coding Standards Document

# 1. General Information

* **Project Name:** Inquizitive
* **Project Type:** Web application to manage student records, classes, and AI-generated quizzes.
* **Tech Stack:**
* **Frontend:** Next.js + React JS, Zod Schemas, Dashboard and component-based architecture.
* **Backend & Database:** Supabase (authentication, database, API).
* **AI Layer:** Grok + FastAPI (AI quiz generation & checking).
* **Deployment:** Vercel (auto-deploy on GitHub push).

# 2. Code Style

* **Language Standards:**
* JavaScript with ES6+
* Type-safe code using Zod.
* **Naming** **Conventions:**
* Variables & functions → camelCase.
* Components & Classes → PascalCase.
* Files & folders → kebab-case.
* **Indentation:** 2 spaces, no tabs.
* **Comments:** JSDoc style for functions, inline comments for complex logic.
* **File/Folder Structure:**
* /components → React components
* /schemas → Zod schemas
* /pages → Next.js routing
* /api → API routes & FastAPI services
* /utils → Helper functions
* /db → Supabase setup

# 3. Best Practices

* **Error** **Handling:** Use try/catch for async calls, return user-friendly messages.
* **Security:** Input validation via Zod, Supabase auth, secure env variables.
* **Performance:** Use caching, optimize queries, and avoid unnecessary re-renders.
* **Reusability:** Create modular components, follow DRY principle.

# 4. Frontend Standards

* React components must be functional with hooks.
* Global state management via context/hooks.
* Type safety enforced via Zod schemas.
* Accessibility: Semantic HTML, alt tags, ARIA roles.
* Responsive design (Laptop-first).

# 5. Backend & AI Standards

* Supabase for authentication, database, and APIs.
* Database tables follow snake\_case convention.
* API endpoints return consistent JSON responses.
* FastAPI as middleware for Grok AI integration.
* AI quiz generation & grading handled through isolated service functions.

# 6. Version Control

* **Repository:** GitHub
* **Branching** **Strategy:** Main branch only (protected).
* **Commits:** Format: feat: …, fix: …, docs: …, chore: …
* **Pull** **Requests:** Must include description, require one review.

# 7. Testing

* **Manual** **Testing:** Performed for all new features.
* **AI-assisted Testing:** AI tools simulate test cases.
* **Automated** **Testing:** Framework TBD (e.g., Jest/React Testing Library).

# 8. Deployment

* Hosted on Vercel.
* Auto-deployment on GitHub push.
* Environment variables managed in Vercel dashboard.
* Rollback available via Vercel UI.

# 9. Tools & Dependencies

* **IDE:** Visual Studio Code (primary).
* **Package** **Manager:** npm.
* **Linters** **&** **Formatters:** ESLint, Prettier.
* **Version** **Control:** GitHub.

# 10. Documentation & Communication

* **Documentation:** README, API docs, schema docs inside repo.
* **Task** **Tracking:** Trello (epics, tasks, sprint planning).
* **Communication:** Onsite meetings & online tools (Zoom/WhatsApp).