NoxVision Development Standards

Guidelines for AI/ML and Web Application Teams

1. Development Environment

1.1 Virtual Environment (Python)

- Use **Python 3.10** for AI/ML features (critical for library compatibility).
- Isolate dependencies with venv or conda.
- Document all packages in requirements.txt with exact versions.

1.2 Frontend Setup

- Use **Node.js LTS (v18.x)** and **npm**.
- No global CSS frameworks; **Tailwind CSS** is allowed.

2. Version Control (GitHub)

2.1 Branching Strategy

- main: Stable/production-ready code.
- dev: Integration branch for testing.
- **Feature branches**: feature/[name] (e.g., feature/fire-detection).

2.2 Commit Practices

- **Prefix commits** by domain:
    - [AI] for machine learning (e.g., [AI] Add YOLOv8 training script).
    - [WEB] for frontend/backend (e.g., [WEB] Fix API authentication).
- **Update GitHub regularly** (no fixed schedule; aim for daily progress).

2.3 Pull Requests (PRs)

- Require **at least one review** before merging to dev.
- Link PRs to GitHub Project Board tasks.

3. Tech Stack Boundaries

3.1 AI/ML Team

Component	Allowed	Restrictions
Language	Python 3.10	No older Python versions
ML Frameworks	PyTorch, TensorFlow Lite	Avoid deprecated libraries
API Development	FastAPI	No Flask/Django

3.2 Web Application Team

Component	Allowed	Restrictions
Frontend Framework	React.js (Functional)	No class components
Styling	Tailwind/CSS Modules	No Bootstrap
State Management	Redux Toolkit	Avoid Context API alone

4. Coding Standards

4.1 General Rules

- **Modularize code**: Split logic into reusable functions/components.
- **Document**: Add comments for complex logic (no snippets; describe purpose).
- **File structure**:
  - ai/ for models/training.
  - web/ for frontend/backend.

4.2 Frontend-Specific

- Use **JSX** with functional components.
- Store styles in src/styles/ (CSS Modules or Tailwind).

4.3 Backend-Specific

- Use **type hints** and docstrings in Python.
- Validate inputs/outputs in APIs.

5. Tools & Design

5.1 Mandatory Tools

- **IDE**: VS Code (team-wide for consistency).
- **Design**: Figma/Canva (AI-generated wireframes allowed).

5.2 Optional/Future Tools

- **CI/CD**: GitHub Actions (if deployment needed).
- **Testing**: Postman (APIs), Jest (React).

6. Documentation

6.1 README Files

- **Root README**: Project overview, setup, and contributors.
- **Per-directory READMEs**: Purpose and key files (e.g., ai/README.md).

6.2 Dependency Files

- **Python**: requirements.txt with pinned versions.
- **Frontend**: package.json with exact versions (npm install --save-exact).

7. Workflow & Collaboration

- **Daily updates**: Push code frequently (no fixed schedule).
- **Weekly syncs**: Discuss blockers (flexible timing).
- **Design collaboration**: Share Figma/Canva links in docs/.

---

8. Flexibility for Future Tech

- New tools/libraries require **team consensus**.
- Document additions in docs/tech_updates.md.

Note: This document is **adaptive**—revise as needed via team discussions.