📝 Coding Guidelines

These guidelines ensure **clean, maintainable, and consistent code** across projects.

---

1. Project Structure

- `src/` → main source code
- `tests/` → automated tests
- `docs/` → architecture diagrams, case studies, documentation

---

2. Naming Conventions

- **Classes & Interfaces** → `PascalCase`
- **Methods & Properties** → `PascalCase`
- **Variables & Parameters** → `camelCase`
- **Constants** → `UPPER_CASE`
- **Folders & Files** → `kebab-case` or `PascalCase` (depending on language)

---

3. C# / .NET Guidelines

- Use **Dependency Injection** for services.
- Follow **Clean Architecture** principles (Controller → Service → Repository → Database).
- Apply **async/await** for database and network calls.
- Write **unit tests** for business logic using xUnit.

---

4. Vue 3 / Frontend Guidelines

- Use **Composition API** and **TypeScript**.
- Organize components under `components/` and views under `views/`.
- Use **Pinia** or Vuex for state management.
- Follow **atomic design** principles where possible.

---

5. Git & Version Control

- Main branches: `main` (stable), `develop` (active development).
- Feature branches: `feature/<name>`.
- Commit messages:
  - `feat: add new login endpoint`
  - `fix: correct null reference in repository`
  - `docs: update case study`

---

6. Testing Strategy

- **Unit tests** for services and repositories.
- **Integration tests** for API endpoints.
- **Frontend tests** with Jest or Vitest.
- Aim for **>70% coverage** in critical modules.

---

7. Code Quality Tools

- Use **SonarLint / SonarQube** for static code analysis.
- Apply **Prettier + ESLint** in frontend projects.
- Run code reviews before merging to `main`.