Orcta Naming Conventions & Software Engineering Style Guide

June 11, 2025

Contents

1 Core Philosophy

"A name should serve as documentation."

Names in code should clearly convey the purpose and behavior of the entity being named. Predictability and clarity across the Orcta codebase is non-negotiable. This guide aligns with the Orcta Engineering Playbook's Code Standards and the Architecture Guidelines' Domain-Driven Design principles.

2 Naming Conventions

2.1 JavaScript / TypeScript Code

Element	Convention	Example
Variables	`camelCase`	`userEmail`, `authToken`
Functions	`camelCase`	`getUserProfile()`, `fetchData()`
Constants	`SCREAMING_SNAKE_CASE`	`API_BASE_URL`, `MAX_RETRY_LIMIT`
Classes/Types	`PascalCase`	`InvoiceService`, `UserForm`
Components	`PascalCase`	`UserCard`, `AuthProvider`
Files/Folders	`kebab-case`	`invoice-service.ts`, `auth/login-form.tsx`

2.2 Python Code

Element	Convention	Example
Variables	`snake_case`	`user_email`, `auth_token`
Functions	`snake_case`	`get_user_profile()`, `fetch_data()`
Constants	`SCREAMING_SNAKE_CASE`	`API_BASE_URL`, `MAX_RETRY_LIMIT`
Classes	`PascalCase`	`InvoiceService`, `UserForm`
Files/Folders	`kebab-case`	`invoice-service.py`, `auth/login-form.py`

2.3 Go Code

Element	Convention	Example
Variables	`camelCase`	`userEmail`, `authToken`
Functions	`camelCase`	`getUserProfile()`, `fetchData()`
Exported Types	`PascalCase`	`InvoiceService`, `UserForm`
Packages	`lowercase`	`invoices`, `auth`
Files/Folders	`kebab-case`	`invoice-service.go`, `auth/login.go`

2.4 React Component Structure

Component Rule	Convention	Example
Component file	`PascalCase`	`UserCard.tsx`
Component folder	`kebab-case`	`user-card/`
Test file	`kebab-case + .test.tsx`	`user-card.test.tsx`

2.5 Folder Structure

See Figure 1 for a visual representation. Example:

```
src/
  auth/
    login/
      login-form.tsx
      login-form.test.tsx
    register/
      register-form.tsx
  dashboard/
    user-stats.tsx
    user-stats.test.tsx
  shared/
    hooks/
      use-auth.ts
    utils/
      format-date.ts
```

Figure 1: Folder Structure for Orcta Projects

2.6 Backend + APIs

Element	Convention	Example
Endpoints	`kebab-case`	`/user-profile`, `/get-token`
Params/Query	`camelCase`	`userId`, `includeMeta`
Controllers	`PascalCase`	`UserController`
Services	`PascalCase`	`PaymentService`

2.7 Database (SQL / NoSQL)

Element	Convention	Example
Table Names	`snake_case`	`user_profiles`
Field Names	`snake_case`	`created_at`, `user_id`

3 Naming Principles

1. **Clarity over Brevity** – Favor clear names (`getUserProfileFromToken`) over vague ones (`getData`).
2. **Avoid Redundant Context** – If the file path gives context, shorten the name.
   - Good: `auth/login/login-form.tsx`
   - Okay: `auth/login/form.tsx`
3. **Consistent Prefixes** – Use verbs for functions (`get`, `create`, `update`), `use` for hooks, `is`/`has` for booleans.
4. **Avoid Abbreviations** – `userProfile`, not `usrProf`.
5. **Avoid Type in Name** – `UserList`, not `UserListComponent`.

4 Test Naming

- Match the file/component under test.
- Use `.test.tsx` or `.test.ts` suffix for consistency.

```
components/
  user-card/
    user-card.tsx
    user-card.test.tsx
```

5 Tooling (Automated Enforcement)

- **JavaScript/TypeScript**:
  - **ESLint**: Use `eslint-plugin-naming-convention` to enforce case and patterns.
  - **Prettier**: Ensures consistent formatting.
  - **TypeScript Rules**: Catch incorrect usage via `tsconfig.json`.
- **Python**:
  - **Flake8**: Enforce PEP 8 naming conventions.
  - **Black**: Auto-format code for consistency.
- **Go**:
  - **golangci-lint**: Enforce naming and style rules.
- **Database**:
  - **sqlfluff**: Lint SQL queries for `snake_case` conventions.
  - **ORMs**: Use Sequelize (SQL) or Mongoose (MongoDB) with naming plugins.
- **Husky + Lint-Staged**: Run linters/formatters pre-commit.

6 Summary Table

Category	Convention	Tool Enforced
Variables (JS/TS)	`camelCase`	ESLint
Variables (Python)	`snake_case`	Flake8
Variables (Go)	`camelCase`	golangci-lint
Components	`PascalCase`	ESLint
Files/Folders	`kebab-case`	Lint-Staged
Constants	`SCREAMING_SNAKE_CASE`	ESLint, Flake8, golangci-lint
DB Entities	`snake_case`	sqlfluff, Sequelize/Mongoose

7 Final Word

Locking into these conventions ensures:

- High readability
- Easier onboarding
- Lower cognitive load

> Use names as design tools. Communicate. Don't abbreviate. Document via naming. Let's build excellent code, one clear name at a time.

8 Maintenance

Maintained by: Orcta Engineering Leadership
Version: 1.1
Last Updated: June 11, 2025
Versioning: This guide is version-controlled in Git, with a changelog maintained in Notion.