Coding Standards Document
Taxi Tap by Git It Done

Contents

1 Introduction

This document outlines the coding standards, conventions, and practices adopted for the development of the project. Adhering to these standards ensures:

- **Uniformity**: All developers follow the same style.
- **Clarity**: Code is easy to read and maintain.
- **Reliability**: Reduce bugs caused by inconsistent practices.
- **Efficiency**: Improve development speed through predictable patterns.

2 Repository Structure

The project repository follows a clear and organized structure:

```
project-root/
|-- .github                     % Workflow files
    |-- workflows
        |-- platform.yml
|-- assets                      % All project images
    |-- images
|-- docs/                       % Documentation
|-- platform                    % The app
    |-- app                     % Frontend
    |-- assets                  % Fonts and images
    |-- components
    |-- constants
    |-- contexts
    |-- convex                  % Backend
    |-- hooks
    |-- tests/                  % Unit and integration tests
    |-- .eslint.config.mjs      % ESLint configuration
    |-- package.json            % Project dependencies and scripts
|-- .prettierrc                 % Prettier configuration
|-- README.md                   % Project description
```

3 Coding Conventions

3.1 Language and Framework

The project uses:

- Programming Language: `TypeScript/JavaScript`
- Frontend Framework: `React Native`
- Backend Framework: `Convex`

3.2 Naming Conventions

- **Files and directories**: camelCase and PascalCase
- **Variables**: camelCase
- **Constants:** `UPPER_SNAKE_CASE` and camelCase
- **Classes/Components**: camelCase and PascalCase
- **Functions**: camelCase

3.3 Code Style

- Indentation: 2 spaces
- Maximum line length: 100 characters
- Use semicolons at the end of statements
- Use single quotes for strings
- Always use curly braces for blocks

4 Tooling and Configuration

4.1 Prettier

Prettier is used for consistent code formatting.

```
npm install --save-dev prettier
```

Example `.prettierrc` configuration:

```
{
  "semi": true,
  "singleQuote": true,
  "tabWidth": 2,
  "printWidth": 100,
  "trailingComma": "all"
}
```

4.2 ESLint

ESLint is used for linting and enforcing coding standards.

```
npm install --save-dev eslint
```

5 Testing Standards

- All code must be covered with unit tests using `Jest`.
- Use descriptive test names.
- Mock external dependencies where appropriate.
- Integration tests.

6 Documentation Standards

- Use clear and concise comments where necessary.
- Keep the README updated with setup and usage instructions.

7 Pull Request Guidelines

- Ensure code passes all linting and tests before submission.
- Provide clear commit messages.
- Perform peer reviews.
- Do not merge code with failing tests.

8 Continuous Integration

- We are using GitHub Actions to automate linting and tests on every pull request.