Coding Standards

Naming Conventions

- Functions, variables, and file names use camelCase.

```
const authenticateToken = (req, res, next) => {
```

- React components use PascalCase.

```
const handleSubmit = async (e) => {
  e.preventDefault();
  setError('');
  setLoading(true);
```

- React class names use kebab-case.

```
<div className="form-group">
  <label htmlFor="email">Email</label>
  <input
    type="email"
    id="email"
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    placeholder="you@example.com"
    required
  />
</div>
```

- Descriptive function and file names used across all components.

Indentation and Formatting

- We follow hierarchical indentation, where each nested block moves one level to the right. This ensures readability by clearly reflecting the logical structure of the code.
- Blocks within functions, conditionals, loops, and components are indented progressively to represent nesting.

This approach is applied consistently across JavaScript/React and SON/YAML files.

```
app.post("/api/register", async (req, res) => {
  try {
    const { name, email, password, role, inviteCode } = req.body;

    if (!name || !email || !password) {
      return res
        .status(400)
        .json({ message: "Name, email, and password are required" });
    }
```

File and Directory Structure

The main branch of our repository is structured as follows:

```
DevX360/
├── frontend/
│   └── (React files & subfolders)
├── devX360/
│   └── (Component folders)
│       ├── ai-analysis/
│       └── api/
├── documentation/
│   └── (Latest documents only)
├── testing/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── .github/
│   └── workflows/
├── assets/
│   └── (extra assets for personalizing repository)
├── README.md
├── package.json
```

Branching Strategy

The following branching strategy will be used, adopted from our Planning and Role Allocation document.

An adjusted/custom Git flow strategy will be used:

GIT FLOW BRANCH	PURPOSE
main	Stable and production ready code, will merge from release or the hotfix branch
develop	This branch will be used mostly for integration purposes and will often have the newest changes to the system that may not yet be ready
feature	This branch will be used specially for our system to store each decoupled module and section of our code. We will have a domain for each module such as a UI domain, API domain, JS-server domain, Testing domain etc. The following convention will be used: feature/UI domain/<related code and directories>
release	Likely will be used towards the end of the system development as the final QA tested code
bugfixes/hotfixes	Used for fixes when unexpected issues arises in the code. Will be merged into dev and or main branches. Bugfix will be used for non-critical bugs while hotfix will be used for urgent bugs.

Version Control Practices

- Commits must have a clear descriptive message
- Important commits must be reviewed by at least one team member