Folder Structure Creation Commands for intelliDGAforge
==========

This document outlines all terminal commands required to replicate the folder structure for the **intelliDGAforge** project. Commands are grouped logically, and brief comments explain the purpose of each step. Run these commands in a Unix-like shell (macOS or Linux) to create the same directory layout.

1. Create the project root and navigate into it
```
# Navigate to the directory where you want to create the project (e.g. your home directory)
cd ~

# Create the root folder and enter it
mkdir intelliDGAforge
cd intelliDGAforge

# Create a top-level README file for the project overview
touch README.md
```

2. Create global configuration and metadata files
```
# Create common dotfiles and an example environment file
# These files help configure editors, linting rules and environment variables.
touch .editorconfig .env.example .gitignore .eslintignore .prettierrc

# Create the .github directory for CI/CD workflows
mkdir -p .github/workflows
```

3. Assets directory
```
# Create the assets directory with subfolders for fonts, icons and images
mkdir -p assets/{fonts,icons,images}

# Add a README to describe the purpose of the assets folder
touch assets/README.md
```

4. Backend API
```
# Create the backend directory and its environment template
mkdir backend
cd backend
touch README.md .env.example

# Create the src directory with common backend subfolders
mkdir -p
src/{config,controllers,middleware,models,routes,services,validators,utils,seeds,jobs,uploads,logs,tests}

# Add README files to each subfolder for documentation
for d in config controllers middleware models routes services validators utils seeds jobs uploads logs tests; do
    touch "src/$d/README.md"
done

# Return to the project root
cd ..
```

5. Configuration files per environment

```
# Create the config directory for environment files and documentation
mkdir config
cd config
touch README.md

# Example environment templates for different stages
for env in development production staging testing; do
    touch "${env}.env.example"
done

# Return to the project root
cd ..
```

6. Database structure
```
# Create database directory with migrations and seeds subfolders
mkdir -p database/{migrations,seeds}

# Add README files to describe how to write migrations and seed data
touch database/README.md database/migrations/README.md database/seeds/README.md
```

7. Project documentation
```
# Create the docs directory and subfolders for API, architecture and UML
diagrams
mkdir -p docs/{api,architecture,uml}

touch docs/README.md docs/uml/README.md
```

8. Docker configuration
```
# Create a folder for Docker and docker-compose files
mkdir docker
```

9. Frontend application
```
# Create the frontend root folder and key files
mkdir frontend
cd frontend
touch README.md index.html package.json vite.config.js

# Set up the src directory with entry points and subfolders
mkdir -p
src/{assets,components,context,hooks,layouts,pages,redux,router,services,styles,
tests,utils}

touch src/App.jsx src/main.jsx src/index.css

# Add README files to each subfolder in src
for d in assets components context hooks layouts pages redux router services
styles tests utils; do
    touch "src/$d/README.md"
done

# Return to the project root
cd ..
```

10. Logs and scripts

```
# Create a logs directory with separate folders for backend and frontend runtime logs
mkdir -p logs/{backend,frontend}

# Create a scripts folder for deployment and maintenance scripts
mkdir scripts
cd scripts
touch README.md
cd ..
```

11. Deployment configurations
```
# Create a deployment directory with separate folders for different deployment strategies
mkdir -p deployment/{k8s,nginx,pm2,terraform}

# Add README files to explain the purpose of each deployment subfolder
touch deployment/README.md
touch deployment/k8s/README.md deployment/nginx/README.md deployment/pm2/README.md deployment/terraform/README.md
```

12. Test suites
```
# Create a top-level tests directory with subfolders for different test types
mkdir -p tests/{backend,frontend,integration,e2e,unit}

touch tests/README.md
```

13. Notes and write-up

- **Root files**: `.editorconfig`, `.env.example`, `.gitignore`, `.eslintignore` and `.prettierrc` configure your editor, linting rules and environment variables. The root `README.md` gives a high-level overview of the project.

- **assets/**: Holds static files such as fonts, icons and images. A `README.md` explains how to organise and reference assets.

- **backend/**: Contains your Node/Express API. The `.env.example` file lists environment variables (e.g. database connection strings). The `src` subdirectories organise the code: `config` for configuration files, `controllers` for route handlers, `middleware` for authentication and validation, `models` for Mongoose schemas, `routes` for route definitions, `services` for business logic, `validators` for request validation schemas, `utils` for helpers, `seeds` for seed data, `jobs` for scheduled tasks, `uploads` for user-uploaded files, `logs` for application logs, and `tests` for unit and integration tests.

- **config/**: Stores environment templates (`development.env.example`, etc.) and a `README.md` describing configuration guidelines.

- **database/**: Houses database migrations and seed scripts. The subfolders `migrations` and `seeds` each have a `README.md` explaining how to run them.

- **docs/**: Contains project documentation, API docs, architecture diagrams and UML diagrams.

- **docker/**: Placeholder for Dockerfile and docker-compose configurations. Use this folder to containerise your application.

- **frontend/**: Holds the React client built with Vite and Tailwind. The `src`

folder contains entry files (`App.jsx`, `main.jsx`, `index.css`) and subdirectories such as `assets`, `components`, `context`, `hooks`, `layouts`, `pages`, `redux` (for state management), `router`, `services`, `styles`, `tests` and `utils`. Each subfolder includes a `README.md` describing its purpose.

- **logs/**: Split into `backend` and `frontend` so runtime logs don’t clutter the source directories. These folders can be mounted into a logging service or aggregated in production.

- **scripts/**: Contains helper scripts for tasks such as deployment, setup and cleanup. The `README.md` should document each script.

- **deployment/**: Organises deployment resources by strategy:
  - `k8s/` for Kubernetes manifests (deployments, services, ingress, config maps).
  - `nginx/` for NGINX configuration files used as a reverse proxy.
  - `pm2/` for PM2 process manager configuration files and deployment scripts.
  - `terraform/` for infrastructure-as-code definitions using Terraform.

- **tests/**: A top-level folder to organise your automated test suites. Subfolders such as `backend`, `frontend`, `integration`, `e2e` and `unit` hold tests of different types.

By following these commands, you will replicate the entire **intelliDGAforge** folder structure. Keep each `README.md` up to date with relevant documentation to help future contributors understand the purpose of each directory.