# RogueMon — MVP Contributor Guide

## Purpose:

Explains the current development approach, the reasoning behind minimal file creation during the MVP phase, and gives newcomers the context they need to contribute.

Why Minimal Files in MVP?

- Faster iteration

- Lower merge conflict risk

- Simpler onboarding

- Easier end-to-end testing

## Exit Criteria for splitting modules:

- MVP complete and stable

- Features justify modularity

- Multiple devs need to work in parallel

## Project Map (Current MVP Layout):

- MVP\_backend: battle\_engine.py, starters.py, backend prototype

- MVP\_frontend: screens (BattleScreen, MainMenu, WelcomeScreen), assets, utils, entry files

- package.json, vite.config.js, eslint config

## Core Concepts:

- Backend: battle\_engine.py → turn resolution, move effects

- Frontend: BattleScreen.jsx → renders state, sends choices

- Starters: starters.py + sprites

- Utilities: fetchAndCacheSprite.js

## Contribution Guidelines:

## Do:

- Add new moves/effects in battle\_engine.py

- Adjust UI flow in screens/

- Add sprites in assets/

## Don’t:

- Split modules prematurely

- Add meta systems (save, catching, leveling)

## Testing Strategy:

- Run backend Flask dev server

- Run frontend with Vite

- Validate starter → battle → win/lose flow

- Edge tests for accuracy and win/loss

## Commenting Guidelines:

- File headers with scope + future notes

- Section headers: Constants, Utilities, Core Loop, Effects

- Function docstrings (inputs/outputs, intent)

- Inline comments only where intent is unclear

- Example traces in battle logic

- TODO/FIXME for MVP exit refactors

## PR Template:

Context, Decision, Tradeoffs, Testing, Risks, Follow-ups

## References:

- Game Design Specifications

- Battle Logic Architecture

- UML Diagrams

- MVP Checklist

- Sprite Strategy