MASS Simulator - README Documentation (Web-Ready)

Project Overview

```
This project develops algorithms to efficiently explore complex state spaces for the Multi-Ammunitio
The goal is to simulate missile–ship engagements safely and cost-effectively, and to predict state s
```

Repository Structure

```
SEPSegFaults/
■■■ src/                # Source code
■   ■■■ algorithms/     # State space exploration algorithms
■   ■■■ simulation/     # Simulation engine and utilities
■   ■■■ __init__.py
■■■ tests/              # Unit and integration tests
■■■ notebooks/          # Jupyter notebooks for exploration
■■■ configs/            # Simulation parameters (YAML/JSON)
■■■ results/            # Experiment outputs
■■■ docs/               # Documentation
■■■ .github/workflows/ # GitHub Actions (CI)
■■■ requirements.txt   # Python dependencies
■■■ .gitignore         # Ignored files
■■■ LICENSE            # Project licence
■■■ README.md          # This file
```

Full Repository Layout

```
SEPSegFaults/
■
■■■ docs/               # Documentation (user guides, reports, design notes)
■■■ notebooks/          # Jupyter notebooks for experiments
■■■ src/                # Main Python source code
■   ■■■ mass_sim/       # Package folder (importable as `import mass_sim`)
■       ■■■ __init__.py
■■■ tests/              # Unit & integration tests
■■■ configs/            # JSON/YAML config files for simulation parameters
■■■ scripts/            # Helper scripts for running tasks
■■■ data/               # Sample datasets or inputs (small!)
■   ■■■ .gitignore      # ignore big/raw data
■■■ requirements.txt    # Python dependencies
■■■ setup.py            # (optional) installable package
■■■ README.md           # Overview & usage instructions
■■■ .gitignore          # Git ignore rules
```

Getting Started

```
1. Clone the repository
git clone <repo-url>
cd SEPSegFaults

2. Create and activate a virtual environment
python -m venv venv
source venv/bin/activate   # Linux/Mac
venv\Scripts\activate    # Windows

3. Install dependencies
pip install -r requirements.txt

4. Run tests
pytest

5. Start Jupyter
```

```
jupyter notebook
```

Workflow

```
- main → stable code only
- feature/ → new features or algorithms
- fix/ → bug fixes
- exp/ → experimental work (use Python Notebooks to explain ideas visually for teammates)

Rules:
- Contributors must not push directly to main
- All changes go through Pull Requests (PRs)
- PRs must pass CI tests and be reviewed by at least one teammate before merging
```

Continuous Integration (CI)

```
This repository uses GitHub Actions.

Every push or Pull Request automatically:
- Installs dependencies
- Runs all tests with pytest

If tests fail, the PR cannot be merged.
```

Results

```
Simulation outputs and experiment results are stored in results/.
These are tracked in Git to allow reproduction of past runs.

Note: Avoid committing files larger than 50MB.
```

Contributing

```
1. Fork or clone the repo
2. Create a feature branch (feature/)
3. Commit your work
4. Push and open a PR
5. Request a review
```

Licence