# Practical Guide: Working with the Evacuation System

## Getting Started

You can run this project either directly on your machine or using Docker. Choose the method that works best for you.

### Option 1: Local Installation

### Phase 1: Setup and Familiarization

1. Clone the repository:

```
git clone <repository_url>
cd zombie_playgroung
```

2. Create and activate a virtual environment:

```
python -m venv .venv
source .venv/bin/activate  # On Linux/Mac
# or
.venv\Scripts\activate  # On Windows
```

3. Install dependencies:

```
pip install -r requirements.txt
```

PROF

### Phase 2: Understanding the System

1. Run a sample simulation with the default policy:

```
python3 run_simulation.py
```

- Observe the visualization
- Study the event log
- Note the success/failure conditions

2. Examine the core interfaces in `public/lib/interfaces.py`:

- `CityGraph`
- `ProxyData`
- `ResourceTypes`
- `PolicyResult`

3. Look at the example implementation in `public/examples/random_policy.py`

### Phase 3: Data Exploration

1. Run multiple simulations to understand variability:

```
python3 run_bulk_simulations.py --skip-city-analysis
```

2. Study the generated data in:

```
data/policies/EvacuationPolicy/experiments/exp_<timestamp>/
├── summary.json              # Overall results
├── cities/                   # Individual scenarios
│   └── city_<id>/
│       ├── definition.json   # City layout
│       ├── proxy_data.json   # Environmental data
│       └── mission_results.json  # Outcomes
└── visualizations/           # Analysis plots
```

3. Analyze proxy data patterns:

- Node indicators
- Edge indicators
- Correlations with outcomes

PROF

### Phase 4: Development Cycle

1. Create your policy in `public/student_code/solution.py`

2. Test single scenarios:

```
python3 run_simulation.py
```

- Use for quick feedback
- Debug specific situations
- Understand failure cases

3. Run bulk tests:

```
# Without detailed analysis (faster)
python3 run_bulk_simulations.py --skip-city-analysis

# With full analysis (more information)
python3 run_bulk_simulations.py
```

4. Analyze results:

- Check success rates
- Study resource usage
- Review event logs
- Examine visualizations

5. Iterate and improve based on data

### Phase 5: Advanced Analysis

1. Create custom visualizations:

- Extend `public/visualization/city_analysis.py`
- Add new metrics to `public/visualization/bulk_analysis.py`

2. Add custom logging:

- Track additional metrics
- Create new analysis plots
- Generate custom reports

3. Experiment with different scenarios:

- Modify city sizes
- Adjust number of runs
- Change random seeds

PROF

### Phase 6: Performance Optimization

1. Profile your solution:

- Time taken per decision
- Resource efficiency
- Path optimality

2. Run large-scale tests:

```
# Increase number of simulations
python3 run_bulk_simulations.py --n-runs 100
```

3. Generate comprehensive reports:

- Success rates across conditions
- Resource usage patterns
- Environmental correlations

## Useful Commands

### Basic Usage

```
# Single run with visualization
python3 run_simulation.py

# Bulk testing without city analysis
python3 run_bulk_simulations.py --skip-city-analysis

# Full bulk testing with all analysis
python3 run_bulk_simulations.py
```

### Additional Options

```
# Set specific random seed
python3 run_simulation.py --seed 42

# Change city size
python3 run_simulation.py --nodes 50

# Custom experiment name
python3 run_bulk_simulations.py --experiment-name "test_run_1"
```

## Data Locations

PROF

### Simulation Results

```
data/policies/EvacuationPolicy/experiments/
└── exp_<timestamp>/
    ├── summary.json
    ├── cities/
    │   └── city_<id>/
    │       ├── definition.json
    │       ├── proxy_data.json
    │       ├── mission_results.json
    │       └── visualizations/
    └── visualizations/
```

### Analysis Outputs

```
data/policies/EvacuationPolicy/
└── experiments/
    └── exp_<timestamp>/
        └── visualizations/
            ├── success_rates.png
            ├── resource_efficiency.png
            ├── proxy_correlations.png
            ├── time_distance.png
            └── resource_impact.png
```

## Development Tips

1. Use version control for your policy implementations
2. Keep notes on what you learn from each experiment
3. Create systematic test cases
4. Document your custom analysis code
5. Back up important experiment results

### Option 2: Using Docker

**Prerequisites**

- Docker installed on your system
- Docker Compose installed on your system

**Quick Start with Docker**

1. Build the Docker image:

```
docker-compose build
```

PROF

2. Run a single simulation:

```
docker-compose run zombie-sim
```

3. Run with different commands:

```
# Run bulk simulations
docker-compose run zombie-sim python3 run_bulk_simulations.py

# Run with skip city analysis
docker-compose run zombie-sim python3 run_bulk_simulations.py --skip-city-analysis
```

```
# Run with specific parameters
docker-compose run zombie-sim python3 run_simulation.py --nodes 50 --seed 42
```

### Development with Docker

The Docker setup includes volume mounts that allow you to:

- Edit code on your local machine and see changes immediately
- Preserve data between runs
- Access visualization outputs locally

1. Start an interactive session:

```
docker-compose run --rm zombie-sim bash
```

2. Run commands inside the container:

```
root@container:/app# python3 run_simulation.py
root@container:/app# python3 run_bulk_simulations.py
```

3. Access generated data:

- All data will be available in your local `data/` directory
- Visualizations can be viewed directly from your local filesystem

### Docker Tips

1. Clean up containers:

PROF

```
docker-compose down
```

2. Rebuild after requirements change:

```
docker-compose build --no-cache
```

3. View container logs:

```
docker-compose logs
```

4. Run with specific environment variables:

```
docker-compose run -e PYTHONPATH=/app zombie-sim python3 run_simulation.py
```