Contents

1 Chapter 1 — Exercises & Labs (Application Mode)

Reward design is now backed both by the closed-form objective (Chapter 1, #EQ-1.2) and by executable checks. The following labs keep theory and implementation coupled.

1.1 Lab 1.1 — Reward Aggregation in the Simulator

Goal: inspect a real simulator step, record the GMV/CM2/STRAT/CLICKS decomposition, and verify that it matches the derivation of #EQ-1.2.

```
import numpy as np
from zoosim.core import config
from zoosim.envs import ZooplusSearchEnv

cfg = config.load_default_config()
sim = ZooplusSearchEnv(cfg, seed=11)
_ = sim.reset()
action = np.zeros(cfg.action.feature_dim, dtype=float)
_, reward_value, _, info = sim.step(action)
print(f"Reward: {reward_value:.2f}")
print(info["reward_details"])
```

Output (representative):

```
Reward: 142.87
RewardBreakdown(gmv=112.7, cm2=22.5, strat=3.0, clicks=4)
```

Tasks 1. Recompute $R = \alpha\text{GMV} + \beta\text{CM2} + \gamma\text{STRAT} + \delta\text{CLICKS}$ using `cfg.reward` and confirm agreement with the printed value. 2. Perturb `cfg.reward.delta_clicks` (keeping `cfg.reward.alpha_gmv` fixed) until the assertion in `zoosim/dynamics/reward.py` fires, and document the smallest violation. 3. Push your findings back into the Chapter 1 text—this lab explains why the code enforces the same bounds as Remark 1.2.1.

1.2 Lab 1.2 — Delta/Alpha Bound Regression Test

Goal: keep the published examples executable via `pytest` so every edit to Chapter 1 remains tethered to code.

```
import pytest
from tests.ch01.test_reward_examples import (
    BusinessWeights,
    SessionOutcome,
    compute_reward,
    compute_conversion_quality,
)
```

```
def test_reward_section_examples():
    outcome_a = SessionOutcome(gmv=120.0, cm2=15.0, strat_exposure=1, clicks=3)
    outcome_b = SessionOutcome(gmv=100.0, cm2=35.0, strat_exposure=3, clicks=4)
    weights = BusinessWeights(alpha_gmv=1.0, beta_cm2=0.5, gamma_strat=0.2, delta_clicks=0.1)
    assert compute_reward(outcome_a, weights) > compute_reward(outcome_b, weights)
    assert compute_conversion_quality(outcome_a) > compute_conversion_quality(outcome_b)

pytest.main(["-k", "reward_section_examples"])
```

Output:

```
============================= test session starts ==============================
collected 1 item
tests/ch01/test_reward_examples.py .                                     [100%]
============================== 1 passed in 0.02s ===============================
```

Tasks 1. Extend `tests/ch01/test_reward_examples.py` with at least one new fixture representing a strategic exposure violation; show how the change propagates to this lab. 2. Tie the assertions explicitly to #EQ-1.2 and #REM-1.2.1 in your chapter text so MkDocs readers understand why the regression test matters.