

RAG Eval BDD - Tester Runbook

This file is a quick-start operational guide for running and validating the BDD evaluation framework.

For full architecture and implementation details, see `README.md` .

1) Quick Setup

Preferred (reuse existing repo venv):

```
cd rag_eval_bdd
source ../.venv/bin/activate
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

If `../.venv` does not exist, create a local one:

```
cd rag_eval_bdd
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

Set runtime env:

```
export BASE_URL="http://localhost:8000"
export OPENAI_API_KEY="your_openai_key" # required for live eval
# optional
export API_KEY=""
export MODEL="gpt-4.1-mini"
```

2) Standard Run Commands

Note: always run against `steps/` (not `features/`) because `pytest-bdd` scenarios are executed through Python step modules.

Run everything:

```
python -m pytest -c pytest.ini steps --alluredir=allure-results
```

Run deterministic smoke gate (stable, no live eval):

```
python -m pytest -c pytest.ini tests -m "smoke" -q
```

Or with Makefile (includes Allure output handling):

```
make smoke
```

Run live evaluation suite:

```
python -m pytest -c pytest.ini steps -m "live" --alluredir=allure-results
```

Makefile shortcut:

```
make live
```

Run live suite in notebook-parity mode (recommended for demo alignment with notebooks):

```
RAG_EVAL_NOTEBOOK_PARITY_MODE=1 python -m pytest -c pytest.ini steps -m "live" --allure
```

Makefile shortcut:

```
make live-notebook-parity
```

Run live evaluation as non-blocking:

```
python -m pytest -c pytest.ini steps -m "live" --alluredir=allure-results || true
```

Default behavior for each run (when `--alluredir` is provided):

- previous `allure-results/` and `allure-report/` are auto-cleaned
- fresh `allure-report/` is auto-generated
- report auto-opens in browser via local HTTP

Run Layer 1 only:

```
python -m pytest -c pytest.ini steps -m "layer1" --alluredir=allure-results
```

Run Layer 2 only:

```
python -m pytest -c pytest.ini steps -m "layer2" --alluredir=allure-results
```

Run selected metrics:

```
python -m pytest -c pytest.ini steps -m "faithfulness or completeness" --alluredir=allure-results
```

Run only Layer 2 feature set:

```
python -m pytest -c pytest.ini steps -m "layer2" --alluredir=allure-results
```

3) Generate and View Report

Normally this is automatic. Manual command (optional):

```
allure generate allure-results --clean -o allure-report
```

Preferred view command:

```
allure open allure-report
```

Optional toggles:

- `RAG_EVAL_AUTO_ALLURE_CLEAN=0` to keep previous Allure results
- `RAG_EVAL_AUTO_ALLURE_GENERATE=0` to disable auto generation
- `RAG_EVAL_AUTO_ALLURE_OPEN=0` to disable popup/open

4) What to Validate in Each Run

1. Scenarios pass/fail status.
2. Per-metric aggregate score vs configured threshold.

3. Pass rate per metric.
4. Per-question raw request/response attachments.
5. Trend artifacts for recent runs.

Result files:

- results/runs/<run_id>/results.json
- results/index.json
- results/trends/last5.json

5) Dataset Usage

Use inline table in feature files for small scenarios.

Use JSON/CSV datasets for reusable regression runs:

- data/datasets/layer1_questions.json
- data/datasets/layer2_questions.json

Required dataset fields:

- id
- question

Recommended fields:

- expected_answer
- category
- source_reference

6) Synthetic Dataset Generation

```
python -m rag_eval_bdd synthesize \  
  --input data/source/ \  
  --output data/generated/questions.json \  
  --num-questions 50
```

7) Config and Thresholds

All thresholds live in:

- `config/config.yaml`

Do not hardcode thresholds in feature files.

Token/cost controls are in `config/config.yaml` under `evaluation` and default to optimized values:

- model defaults to `gpt-4.1-mini`
- shorter retrieval context for evaluation prompts
- disabled metric reasons by default
- upload and ask caching enabled
- DeepEval retries reduced

Optional env overrides:

- `RAG_EVAL_INCLUDE_REASON=1`
- `RAG_EVAL_MAX_CONTEXT_CHUNKS=3`
- `RAG_EVAL_MAX_CONTEXT_CHARS_PER_CHUNK=1200`
- `RAG_EVAL_DEEPEVAL_RETRY_MAX_ATTEMPTS=2`
- `RAG_EVAL_CACHE_UPLOADED_DOCUMENTS=0`
- `RAG_EVAL_CACHE_ASK_RESPONSES=0`
- `RAG_EVAL_NOTEBOOK_PARITY_MODE=1` (fresh session/question + no trim + 0.5 thresholds + positional metric mapping)
- `RAG_EVAL_FRESH_SESSION_PER_QUESTION=1`
- `RAG_EVAL_DISABLE_CONTEXT_TRIMMING=1`
- `RAG_EVAL_METRIC_QUESTION_MAPPING_MODE=all|positional|row`

8) Tag Behavior

- `@smoke` -> deterministic unit/internals checks (CI gate)
- `@live` -> live backend + LLM evaluation scenarios
- `@layer1` -> contextual precision, recall, relevancy
- `@layer2` -> answer relevancy, faithfulness, completeness
- metric tags restrict the selected set
- both layers together -> union of metrics

9) Common Issues

ModuleNotFoundError :

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

Backend unreachable:

- Verify backend is running.
- Verify `BASE_URL` .

No metrics selected:

- Add `@layer1 / @layer2` or metric tags.

no tests ran :

- You likely executed `pytest ... features` .
- Run `python -m pytest -c pytest.ini steps --alluredir=allure-results` .

Allure empty:

- Ensure tests were run with `--alluredir=allure-results` .
- Ensure Allure CLI + Java are installed.
- If it keeps "Loading...", avoid opening `file:///.../allure-report/index.html` directly.
- Use `allure open allure-report` (or let the framework auto-open it).

10) CI Quick Notes

Jenkins pipeline example:

- `jenkins/Jenkinsfile`

TeamCity references:

- `teamcity/README.md`
- `teamcity/settings.kts`

11) Recommended Daily Flow

1. Pull latest code.

2. Activate venv.
3. Run smoke gate (`tests -m smoke`).
4. Run live suite (`steps -m live`) as needed.
5. Review the auto-generated/opened Allure report.
6. Check run JSON + trend JSON.
7. Raise defects with scenario name, metric, question id, and attachment links.