Configuration Manual

MSc Research Project
Cloud Computing

Nan Wang
Student ID: 21185069

School of Computing
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …………………………Nan Wang…………………………………………………………

Student ID: ……………………………………………….21185069……………………………….……

Programme: …………MSc Cloud Computing……………………… **Year:** …………1…………..

Module: ……………Research Project………………………………………………………….………

Lecturer: …………… Rashid Mijumbi……………………………………………………….………

Submission Due Date: …………2 p.m. 11th August 2025……………………………………………….………

Project Title: ……………………… Automated Test Case Generation for Software APIs Using Reinforcement Learning….…………………………………………………

Word Count: ……………..926…………….. **Page Count:** ……………………5……….………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …………………………………Nan Wang…………………………………………………

Date: …………………………………9th August 2025………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nan Wang
21185069

1 Overview

This configuration manual accompanies the experimental pipeline of the master’s thesis titled: "Automated Test Case Generation for Software APIs Using Reinforcement Learning".

The system uses a hybrid LLM+RL framework to:

- Parse OpenAPI specifications;
- Generate test strategies;
- Train a reinforcement learning agent (PPO);
- Execute test cases on real APIs;
- Measure performance (coverage, fault detection, efficiency);

Target API: Swagger Petstore API v2
Framework: Python 3.12 + Stable-Baselines3 + Gymnasium
Key Metrics: Operation Coverage, Fault Detection Rate, Efficiency

This manual ensures full reproducibility of the experiment.

2 System Requirements

- Operating System: Windows 10/11 (64-bit), macOS, or Linux
- Hardware: Intel i5/i7 or equivalent, 16GB RAM minimum
- Python Version: 3.9 – 3.13
- Internet Connection: required (to access Petstore API)
- Disk Space: 500 MB free
- Tools: Command Prompt / Terminal, Python, VS Code (recommended)

This project runs on CPU only — no GPU required.

3 Project Structure

After setup, folder should look like this:

```
automated_test_rl/
├──── data/
│    └──── petstore_openapi.json      # OpenAPI spec for Petstore API
├──── models/
│    └──── ppo_api_tester.zip        # Trained RL model
```

```
├──── outputs/
│   ├──── experiment_results.json    # Raw test execution log
│   ├──── reward_plot.png          # Reward trend over time
│   └──── final_summary.txt         # Auto-generated result summary
├──── parse_openapi.py            # Parses OpenAPI spec
├──── generate_test_strategy.py      # Converts spec into test cases
├──── env_api_testing.py           # RL environment (custom Gym)
├──── train_rl_agent.py           # Trains the PPO agent
├──── run_experiments.py           # Runs trained agent
├──── visualize_results.py          # Generates reward plot
├──── analyze_stats.py            # Analyzes test outcomes
├──── generate_summary.py          # Summarizes final results
├──── requirements.txt            # Python dependencies
└──── README.md                # Project overview
```

4 Step-by-Step Setup Guide

Steps for reproduce the experiment.

4.1 Open Command Prompt in Project Folder

Open File Explorer;
Navigate to your project folder:

```C:\Users\YourName\Desktop\automated_test_rl```

Hold Shift + Right-click → Open PowerShell window here

You'll see:

```C:\Users\YourName\Desktop\automated_test_rl>```

4.2 Create Virtual Environment

Run:

```
cmd

python -m venv env
```

This creates an isolated environment so your project doesn't interfere with other Python projects.

4.3 Activate the Environment

Run:

```
cmd

env\Scripts\activate
```

You should now see:

```cmd

(env) C:\Users\YourName\Desktop\automated_test_rl>
```

The (env) prefix means the virtual environment is active.

4.4 Install Required Libraries

Run:

```
cmd

pip install torch stable-baselines3 requests pyyaml pandas numpy matplotlib seaborn gymnasium
```

Or just simply run the following command:
```pip install requirements.txt ```

Wait until installation completes (may take 2–5 minutes).

These libraries are used for:

- torch: Deep learning backend
- stable-baselines3: RL agent (PPO)
- requests: API calls
- matplotlib: Plotting results

4.5 Download the OpenAPI Specification

Download the Petstore API spec:

1) Go to: https://petstore.swagger.io/v2/swagger.json
2) Right-click → Save As
3) Save as: data/petstore_openapi.json
4) Create the data/ folder if it doesn't exist.

4.6 Running the Experiment

Now run the scripts in this exact order:

```
cmd
python parse_openapi.py
python generate_test_strategy.py
python train_rl_agent.py
python run_experiments.py
python visualize_results.py
python analyze_stats.py
python generate_summary.py
```

Each script performs a specific task:

- parse_openapi.py: loads and prints API endpoints
- generate_test_strategy.py: creates test cases from spec
- train_rl_agent.py: trains the RL agent, and after the training is completed, ‘ppo_api_tester.zip’ will be in models/.
- run_experiments.py: runs the trained agent, results are automatically saved to ‘outputs/experiment_results.json’
- visualize_results.py: saves reward plot – ‘reward_plot.png’
- analyze_stats.py: prints performance metrics
- generate_summary.py: generates final summary

4.7 Expected Outputs

After running all scripts, you should see:

Console Output (Example):

- Experiment finished in 17.62 seconds
- Total steps: 20, Total reward: 35.0

FINAL ANALYSIS

- Total Test Steps: 20
- Successful (200 OK): 5
- Faults Detected (4xx/5xx): 15
- Operation Coverage: 100%
- Fault Detection Rate: 75.0%
- Efficiency: 1.05 tests/second
- Total Reward: 35.0

Files Generated in outputs/

- experiment_results.json: a step-by-step test execution log
- reward_plot.png: the graph of reward over time
  X-axis: Test Step (1 to 20)
  Y-axis: Reward (+1 for success, +2 for fault)
- final_summary.txt: complete experiment summary

Shows the agent detecting faults across multiple endpoints

5 Key Configuration Instructions

5.1 Reward Rules (modifiable in env_api_testing.py):

- Successful Response (200): +1;
- Error Response (4xx/5xx): +2;
- Network Error: -1
- Test Skipped: 0

5.2 Training Parameters (train_rl_agent.py):

```
pythonmodel = PPO(
        "MlpPolicy",
        env,
        verbose=1,                          # display training
        logslearning_rate=0.001,       # learning
       raten_steps=2048               # number of steps per update
)
```

5.3 Testing Settings

- Timeout: 5 seconds (modifiable in env_api_testing.py)
- Testing Order: In the order of paths specified in the API specification

6 Frequently Asked Questions (FAQ)

- Model Fails to Load
  - Check if the `.zip` file exists in the `models/` directory
  - Verify you're using compatible versions of stable-baselines3

- API Test Timeouts:
  - Increase the `timeout` parameter (default: 5 seconds)
  - Check your network connection

- Visualization Not Displaying:
  - Ensure matplotlib is installed: `pip install matplotlib`
  - For Linux servers, set: `export MPLBACKEND=Agg`

- Advanced Usage
  - Custom Test Strategies: Modify test case generation logic in `generate_test_strategy.py`

- Multiple API Testing:
  - Add new OpenAPI specification files
  - Adjust parsing logic in `parse_openapi.py`

- Long-Running Tests
  - recommended command:
    ```bash
       nohup python run_tests.py > test.log 2>&1 &
    ```

7 Important Notes

- Start with small test batches (10-20 endpoints) for initial runs
- Conduct thorough validation before production deployment
- Maintain versioned copies of important test results

8 Conclusion

This configuration manual ensures that your thesis experiment is fully reproducible, transparent, and verifiable by examiners, reviewers, or industry partners.