National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Balaji Srikanthan
Student ID: x23413263

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Balaji Srikanthan
Student ID:	x23413263
Programme:	Programme Name
Year:	2025
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	1607
Page Count:	10

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:	Balaji Srikanthan
Date:	10th December 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

Balaji Srikanthan
x23413263

1 Introduction

SimCloud is a lightweight trust evaluator framework that can be used to analyse cloud service behaviour in normal, adversarial (attack), and aging conditions. The system is used to assess the services based on four interpretable components:

- QoS-based scoring
- Green energy scoring
- Anomaly detection based on rules.
- Time-based trust decay

This manual contains the documentation on the installation, configuration, run and also the extension of the SimCloud Framework. It has been structured in such a way that it will take users through requirements, configuration files, architecture, and experiment workflows as well as troubleshooting.

2 Scope

The scope of the project is to design and test SimCloud, a lightweight trust system to select a cloud service. The system conducts the experiments on simulated datasets of QoS that simulates both standard and compromised cloud conditions. The trust model considers four actual components that are implemented such as Explainability, green-energy scoring, rule-based anomaly detection and time-based trust decay. This evaluation is limited to three simulated scenarios, including normal operation, single-service manipulation, and cohort attacks and compares SimCloud with two baselines, which are a random strategy and a QoS-only model.

3 Objective

This project is aimed at creating a model for cloud service selection based on trust by creating a lightweight and transparent framework that operates based on simulation. SimCloud will attempt to integrate several factors that influence trust including QoS, performance, ability to endure anomalies and recency of data to one scoring system that can be easily understood. The important goal is to test the behavior of the framework in

various simulated settings such as clean environments, single-service attacks, and cohort-level manipulation and to compare the ranking effectiveness of the framework with QoS-only and random baselines. Last but not the least, the project will ensure that trust decisions can be explained, reproduced and are computationally inexpensive; that is, robust trust assessment is not dependent upon the resources of complex machine-learning models or high-end infrastructure.

4 System Requirements

4.1 Hardware Requirements

- Minimum 2GB RAM
- Dual Core CPU
- 2 GB free disk space
- GPU (optional)

4.2 Software Requirements

- OS: Windows 10 or above
- Python 3.10 or above
- Package manager: Pip
- numpy, pandas, matplotlib, streamlit

5 Installation Instructions

1. **Clone the repository:**

   ```
   git clone <repository-url>
   cd <project-folder>
   ```

2. **Create and activate a virtual environment:**

   ```
   python3 -m venv venv

   # macOS/Linux:
   source venv/bin/activate

   # Windows PowerShell:
   venv\Scripts\activate
   ```

3. **Install all dependencies:**

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

5.1 Project Structure

```
JERRICHO/
|-- __pycache__/                  # Python cache files
|-- .venv/                        # Virtual environment
|-- .vscode/                      # VSCode settings
|
|-- data/                         # Simulation & evaluation datasets
|
|-- notebooks/                    # Jupyter notebooks (optional analysis)
|
|-- pages/                        # Streamlit dashboard pages
|   |-- evaluation.py             # Comparison: SimCloud vs baselines
|   `-- experiments.py            # Experiment visualisation UI
|
|-- src/                          # Core implementation
|   |-- __pycache__/
|   |-- anomaly_rules.py          # Rule-based anomaly detection
|   |-- data_loader.py            # Data loading & simulation helpers
|   |-- decay.py                  # Time-based trust decay module
|   |-- explainability.py         # Trust explainability generator
|   |-- green_scoring.py          # Sustainability scoring
|   |-- init.py
|   `-- trust_model.py            # Core trust computation pipeline
|
|-- tests/                        # Unit tests
|   |-- init.py
|   `-- test_trust_model.py       # Trust model test cases
|
|-- .gitignore
|-- app_streamlit.py              # Main Streamlit dashboard entry point
|-- experiments.py                # Backend: runs experiments for plots
|-- requirements.txt              # Python dependencies
|-- run_evaluation.py             # Runs full evaluation (Normal/Attack/Aging)
`-- run_phase2.py
```

5.2 Dataset File

- The simulated dataset is present in the simulated services.csv file.
- It is created by the load or generate function() inside the data loader.py file

6 Architecture Overview

SimCloud is designed using four independent components that collectively compute a usable and confident trust score. The first stage is the QoS preprocessing and scoring module which normalises core service attributes of availability, success rate and response time to

produce a standardised baseline QoS trust score. This is stretched by the Green Energy Scoring module that rewards services that exhibit energy efficient behaviour contributing a sustainability conscious dimension to trust assessment. The Rule-Based Anomaly Detection module then reviews the QoS submissions of impossible values or different metrics patterns to allow the system to report and punish manipulated or suspicious submissions using transparent rule-based offsets. Lastly is the Time-Based Trust Decay which imposes penalties on old or obsolete QoS information and the trust values should focus on recency and dynamic service behaviour. The Trust Aggregator mixes the results of all four modules and gives out the effective trust score in its final form and ranks the cloud services, respectively.

6.1 SimCloud Architecture Diagram

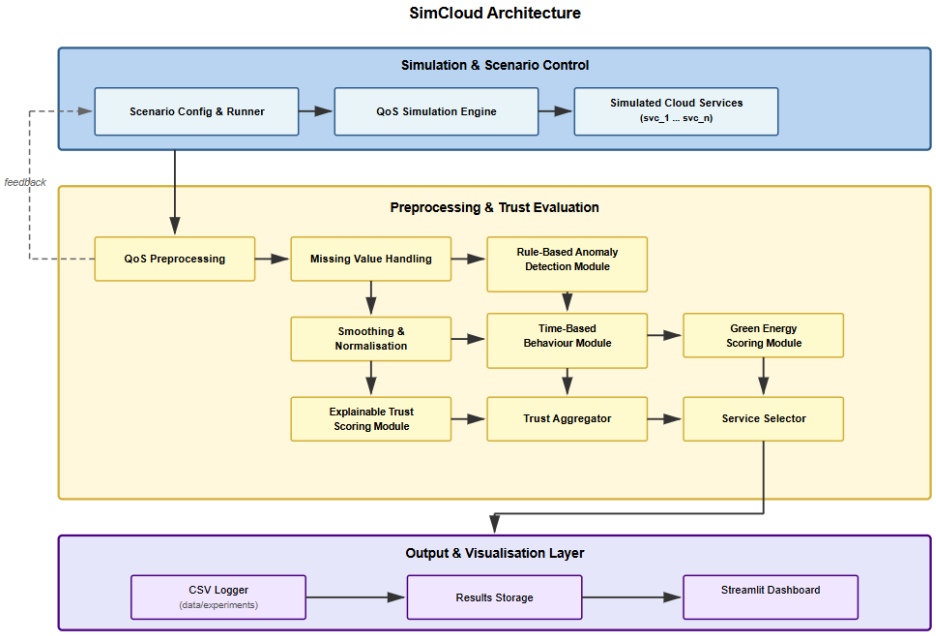

Figure 1: Architecture of the SimCloud Simulation Framework

6.2 Components of SimCloud

7 Simcloud Execution

7.1 Run Experiments for Trust Evaluation

To execute all simulation experiments, run:

```
python experiments.py
```

This script generates the following outputs:

- Trust Breakdown for each Service

- Trust Comparison charts Before/After
- Visuals including Factor Decomposition chart

Key Code Snippet:

```python
def exp_single_service_anomaly(svc_id: str = "svc_1"):
    """
    EXP 1: Single-service anomaly injection
    Goal: simulate a fake QoS submission (attack) and see its impact on trust and rank.
    """
    print("\n[EXP1] Single-service anomaly injection")
    ensure_outdir()
    base = load_or_generate("data/simulated_services.csv", n=N_SERVICES).copy()
    base, now = attach_recent_last_updated(base, max_days=10, seed=111)

    # BEFORE: normal data
    before = pipeline(base, now)
    r_before = rank_of(before, svc_id)

    # AFTER: inject fake QoS anomaly for this service
    attacked = base.copy()
    attacked.loc[attacked["service_id"] == svc_id, "response_time_ms"] = 2000.0
    attacked.loc[attacked["service_id"] == svc_id, "availability"] = 1.05
    after = pipeline(attacked, now)
    r_after = rank_of(after, svc_id)

    print(f"svc: {svc_id} | rank before: {r_before} | rank after: {r_after}")
```

Expected Output:

```
[EXP1] Single-service anomaly injection
svc: svc_1 | rank before: 6 | rank after: 12
Saved -> data\experiments\exp1_before_ranking.png
Saved -> data\experiments\exp1_after_ranking.png
Saved -> data\experiments\exp1_svc1_factors.png

[EXP2] Single-service aging / fixed decay
svc: svc_1 | rank before: 7 | rank after: 9
Saved -> data\experiments\exp2_before_ranking.png
Saved -> data\experiments\exp2_after_ranking.png
Saved -> data\experiments\exp2_svc1_factors.png

[EXP3] Cohort attack (~25% manipulated)
Anomaly detection rate: 25% (target: high)
Saved -> data\experiments\exp3_before_ranking.png
Saved -> data\experiments\exp3_after_ranking.png

All experiments complete. See data/experiments/ for outputs.
```

7.2 Run Evaluation Benchmark

To run the full evaluation benchmark across Normal, Attack, and Aging scenarios:

`python run evaluation.py`

This produces performance comparisons including:

- SimCloud vs. QoS-only vs. Random baselines
- Top-3 Ranking accuracy

- Anomaly detection rates (attack scenario)
- Green energy uplift (normal scenario)

Key Code Snippet:

```python
def scenario_setup(name: str, n: int, seed: int):
    """
    Creates simulated data for a given scenario:
      - Normal: clean and realistic QoS data.
      - Attack: adds fake/manipulated QoS values to simulate bad actors.
      - Aging: marks data as old to simulate outdated information.
    """
    rng = np.random.default_rng(seed)
    df = load_or_generate("data/simulated_services.csv", n=n).copy()
    now = datetime.utcnow()  # current time

    if name == "normal":
        # Normal scenario: recently updated data
        days_ago = rng.integers(0, 8, size=n)

    elif name == "attack":
        # Attack scenario: some services lie about their QoS
        days_ago = rng.integers(0, 8, size=n)
        idx = rng.choice(n, size=max(1, n // 4), replace=False)
        df.loc[idx, "response_time_ms"] = 2000.0     # unrealistically high
        if len(idx) >= 2:
            df.loc[idx[:2], "availability"] = 1.05   # impossible (>100%)
```

.

Expected Output:

```
(.venv) PS C:\Users\Balaji Srikanthan\OneDrive\Documents\jerricho> python .\run_evaluation.py
Saved -> data\eval\eval_summary.csv
Saved -> data\eval\eval_summary_mean.csv
Saved -> data\eval\normal_top3_acc.png
Saved -> data\eval\normal_green_uplift.png
Saved -> data\eval\attack_top3_acc.png
Saved -> data\eval\attack_anomaly_rate.png
Saved -> data\eval\aging_top3_acc.png
(.venv) PS C:\Users\Balaji Srikanthan\OneDrive\Documents\jerricho>
```

8 Experiments

8.1 Experiment 1: Single Service Anomaly Attack

This experiment tests the responsiveness of SimCloud to the submission of falsified QoS data by one of the services. Unrealistic values like 1.05 availability and excessive latency are also introduced in order to simulate a desired manipulation attempt. The aim is to note whether SimCloud is correctly detecting the anomaly and lowering the trust score of the service. The change in ranking is the result of the sensitivity of the system to explicit QoS attacks.

8.2 Experiment 2: Single Service Aging/Decay

This experiment investigates the role of outdated QoS information in the evaluation of trust. The chosen service is made to appear 35 days old, which imitates stale or rarely updated information. Recency is used to reduce the trust score by SimCloud using its time-based decay mechanism. The experiment confirms the assumption that the system correctly prioritises old and unreliable update QoS services.

8.3 Experiment 3: Cohort Attack

This experiment is an assessment of the strength of SimCloud under the coordinated manipulation of several services. False values of QoS such as unrealistic availability and extreme latency are allocated to around 25 percent of services. The rule-based anomaly detector of SimCloud is evaluated according to how it detects and punishes such manipulated entries. The most important is the rate of anomaly detection through the compromised cohort.

9 Experiments Outputs and Analysis

9.1 Experiment 1: Single Service Anomaly Attack

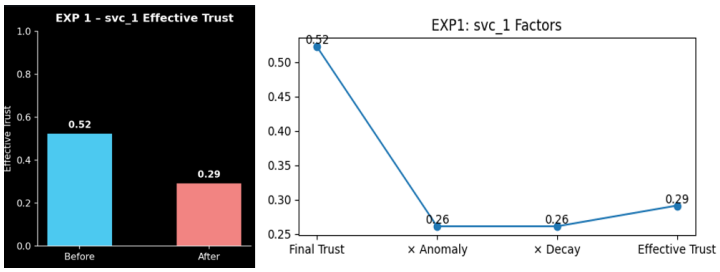

Figure 2: Effective trust dropping from 0.52 to 0.29 following the detection of anomaly injection (left) and factor breakdown for effective trust (right)

9.2 Experiment 2: Single Service Aging/Decay

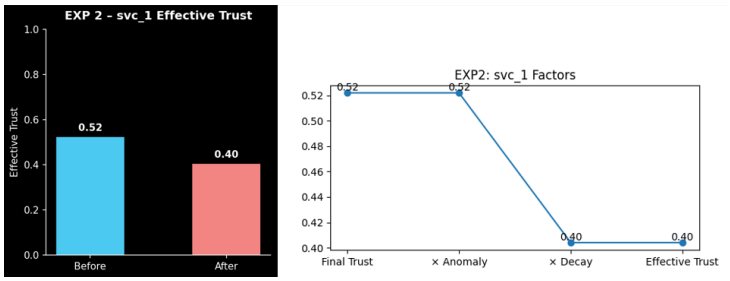

Figure 3: Effective trust dropped from 0.52 to 0.49 following the penalty for stale data

9.3 Experiment 3: Cohort Attack

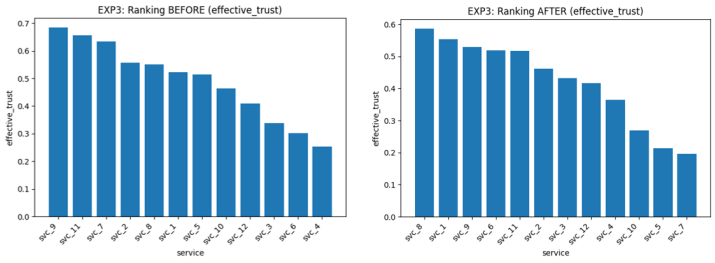

Figure 4: Baseline Trust Ranking of all services before manipulation (left) Baseline Ranking after 25 percent of services were manipulated (right). Compromised services show clear drop in trust ranking

9.4 Analysis of Results

In all the three experiments, SimCloud proved to be a stable, transparent, and flexible cloud in both benign and mutually hostile cloud environments. The single-service anomaly injection test demonstrated consistency in falsified QoS behaviour detection and high sensitivity and determinism of the rule-based anomaly module used in the framework because it lowered the trust score of the service significantly. The time-decay experiment confirmed the fact that SimCloud has the capability to punish stale or out of date non-renewed QoS information which will lower the trust scores, even though the actual QoS measurements remained constant. Here is the focus on recency and reliability of the system. SimCloud has high resilience to integrity attacks at a larger scale showing strong resistance to a cohort attack with 25 percent service manipulation survived by promoting compromised ones and keeping the honest services at place, as well as in the cohort attack scenario where 25 percent of services were manipulated. Combined, these results support the fact that SimCloud believes in prioritizing integrity, predictable when manipulated, and reflects temporal behaviour well-that proves its suitability of the evaluation framework as a lightweight, yet practical cloud service evaluation framework.

10 Summary

SimCloud provides a basic yet reasonable platform of trustful evaluation of the cloud service environments. The framework offers a clear and effective decision-making on both normal and adversarial environments with the modular framework made up of explainable scoring of trust, sustainability-conscious decision, time-based decay, and rule-based anomaly detection. The results of the experiments confirm that SimCloud can always ensure accurate rankings, penalize manipulated or stale QoS submissions, and ensure the integrity of trust without any complicated machine learning models or complex infrastructures. The further refinements are better quality anomaly analytics, more realistic behavioural modelling, multi-cloud capability, and automated policy optimization, all of which would provide a huge potential improvement to make SimCloud a new generation, intelligent, trust management environment capable of autonomously achieving reliability, fairness, and sustainability of all types of cloud-native workloads.

References