**Saar Weinberg** סער וינברג תז:209401934, email: saarwai@gmail.com

**Randomness Testing Project**

**Overview**

This project is a cross-platform randomness testing suite that generates random numbers using different sources: including Python, Java threads, system nano-time, and microphone input and evaluates their statistical quality with a rich set of tests. It features a Flask-based backend server, auto-tests (Pytest), and an extensible structure.

**Main Features:**

* Supports multiple random generators (Python, Java, time-based, and sound/microphone)
* Wide collection of randomness tests: Frequency, Chi-Square, Runs, Serial, Autocorrelation, Poker, Maurer’s Universal Test and more
* Asynchronous task handling, status updates, and error logging
* Simple web interface, with extendable potential for React/Bootstrap frontends
* Full automated testing (Pytest)

**Structure & Files**

|  |  |
| --- | --- |
| File | Description |
| [**app.py**](http://app.py) | Flask web server: exposes endpoints, manages tests/background threads, connects user requests to generators/tests. |
| [**generators.py**](http://generators.py) | Implements all random number generators (Python, time, sound, and Java) and provides a Factory for easy instantiation. Handles resource management and errors. |
| **MyRandomProject.java** | Java-based random number generator using thread scheduling competition for entropy. Used by the Python backend via subprocess. |
| **tests\_module.py** | Implements all randomness statistical tests. Each test returns pass/fail and details on bit patterns/statistics. |
| **test\_generators.py** | Pytest test suites for automatic testing of all generator and randomness logic. |
| **test\_tests\_module.py** | Pytest module for thorough validation of all randomness tests in tests\_module.py. |
| **generator\_errors.log, flask\_app\_errors.log** | Log files tracking generator and server errors for debugging. |
| **direct.html, full\_tests.html**  static/cubes.jpg | These files are HTML templates for the user interface pages of the Flask application.  **direct.html** A page that allows the user to run a random number generator and immediately see a generated value, selecting the desired algorithm and specifying an upper bound. Suitable for quick and interactive checks.  **full\_tests.html** A page designed for comprehensive statistical testing of randomness sequences from the various generators. The user can choose a generator, set the type of test (such as Frequency Test, Runs Test, and more), define parameters like sample size or upper bound, execute any test, and view results in the interface after the test run is completed.  Picture of cubes. |

**Installation & Setup**

**Prerequisites:**

* Python 3.8+
* pip packages: flask, numpy, scipy, pyaudio, pytest (see requirements.txt)
* Java (for Java generator)
* Microphone (for sound generator functionality)
* Node.js, npm (if you wish to build a React/Bootstrap frontend)

**Setup:**

1. Clone/download the repository and place all files in a working directory.
2. Make sure paths in generators.py, tests\_module.py, and app.py match your project location.
3. Install required Python packages:

pip install flask numpy scipy pyaudio pytest

1. Ensure Java is installed and accessible in your PATH.

**Running the Project**

1. **Start the server:**

python app.py

* + Access the web UI at http://localhost:5000/
  + Try the generators and then click button to go to the generators tests page

1. **Testing the generators:**
   * Via web UI, select generator, test type, number of samples, and upper bound.
   * View and filter results for statistical evaluation.
2. **Run Pytest automated tests:**

pytest test.py  
pytest test\_generators.py  
pytest test\_tests\_module.py

**API Endpoints (Flask)**

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| / | GET/POST | Main direct random generator page |
| /tests | GET | Full randomness test suite UI page |
| /start\_test | POST | Start a randomness test (background thread) |
| /stop\_test | POST | Stop a running test task |
| /status/ | GET | Get status and results for a specific test task |

**Statistical Tests Implemented**

* **frequency** (Monobit)
* **runs**
* **chi\_squared\_full\_test**
* **serial\_test** (pairs/triplets)
* **autocorrelation\_test** (lags)
* **poker\_test**
* **maurer\_universal\_test** (compressibility)

Each test returns pass/fail decision and statistical details.

**Extending the Project**

* **Frontend Upgrade:** To further modernize the UI, migrate full\_tests.html and direct.html interfaces to React/Bootstrap based components, using the Flask API for backend logic.
* **Adding Generators:** Implement new generator classes in generators.py, add them to generator\_factory mapping and update UI accordingly.
* **Expanding Tests:** Add new statistical tests to tests\_module.py with clear APIs for result formatting.

**Troubleshooting**

* **Error Logs:** See generator\_errors.log and flask\_app\_errors.log for Python/Java/server tracebacks.
* **Java Generator Issues:** Ensure Java is installed and project folder paths are correct; errors are logged if subprocess fails.
* **Sound Generator:** Requires microphone access; test hardware before use.