Updated Descriptive Summary of the Project

The project is an interactive system for generating random numbers and evaluating their statistical quality, built on a Flask backend and a modern English web UI using Bootstrap 5 on the client side. The system lets users choose among several different random generators-including a Python-based generator, a Java thread-scheduling-based generator, a system nano-time-based generator, and an ambient sound-based generator-and run a rich set of statistical tests on sampled sequences.

Users can configure an upper bound and sample size and run a specific test from a suite that includes: Frequency (Monobit), Runs, Chi‑Square (byte-level), Serial (pairs/triplets), Autocorrelation (lags 1/2), Poker (4/5-bit groups), and Maurer’s Universal (L=7). Test results are presented in a clear textual format with key metrics (e.g., p‑value, X², z‑score) and a PASS/FAIL decision, along with the name of the generator used to produce the sequence.

System components after the upgrade

* Fast interface for generating a single number instantly (direct.html): choose generator and upper bound, get an immediate value.
* interface for running generators and statistical analysis (full\_tests.html): select generator, test type, parameters (Samples, Upper Bound), run asynchronously in the background, view live progress (percent) and status updates, stop on demand, and display final results.
* Updated client side (app\_en.js): real-time validation for numeric inputs, toasts for success/error, tooltips, and complete task lifecycle management (Start/Stop/Status/Completion), including progress reaching 100% on completion.
* Server side (app\_updated\_en.py): endpoints for creating tasks, stopping them, and polling status/results; background execution of sampling + test; improved logging; user-friendly error handling with Jinja templates for 404/500.
* Generators ([generators.py](http://generators.py)):
  + PythonRandomGenerator (import random)
  + JavaRandomGenerator (spawns an external Java process and parses output)
  + NanoTimeRandomGenerator (based on time\_ns with a short random delay)
  + SoundRandomGenerator (requires microphone and PyAudio; proper resource management and close)
* Statistical tests module (tests\_module.py): implementations for all tests with dedicated error handling and logging; consistent result dictionaries (statistics, p‑value, passed).
* Automated tests (pytest):
  + test\_generators.py verifies ranges, basic variability, Java generator behavior under mock, and safe closing of the sound generator.
  + test\_tests\_module.py checks structural and statistical correctness of all test functions.

Key improvements in the upgraded version

* English LTR UI with Bootstrap 5, including English texts, toasts, and tooltips.
* Asynchronous test execution with detailed status and live progress that reaches 100% on completion.
* Generator display name included in final results for full transparency.
* Stronger client-side validation and unified server-side error handling.
* Jinja templates for 404/500 for friendly error pages.
* Clear, separate logs for application and generator errors.

Internet architecture and technologies

* Back end: Flask (Python) running test jobs in the background, endpoints for Start/Stop/Status, processing results and returning JSON.
* Front end: HTML templates (direct.html, full\_tests.html, base.html) with Bootstrap 5, modern JavaScript (app\_en.js) for interactivity, AJAX/JSON for asynchronous requests.
* Data storage: no central DB by default; status tracking via results.txt and logs for troubleshooting. Can be extended to a database (MySQL/MongoDB/Firebase) for history and analytics.
* Target platforms: modern desktop and mobile browsers; can be further enhanced for responsiveness with Bootstrap.

System requirements and notes

* Python 3.8+ with: flask, numpy, scipy, pyaudio, pytest.
* Java required for the Java generator; ensure java is available on the system PATH.
* Active microphone and PyAudio for the sound generator; browser/OS permissions required.
* Update the project directory path in app\_updated\_en.py, [generators.py](http://generators.py), and tests\_module.py according to the local environment.

Typical workflow

1. Quick generation: open the home page, select a generator and upper bound, immediately get a random number.
2. Statistical testing: navigate to /tests, choose generator and test, set Samples and Upper Bound, run and watch live progress reach 100%, then review detailed results.
3. Stopping a test: pressing Stop sends a stop request; stopping is cooperative and takes effect between iterations.
4. Troubleshooting: check logs (generator/server errors), fix microphone permissions/Java PATH/missing templates.

Recommended future extensions

* Fully responsive front end, add charts/visuals for result analysis.
* Add a database for history, reporting, and trend analysis.
* Add more generators/tests and deeper analyses (e.g., full NIST SP 800‑22).
* If needed for research, consider disabling generator-level “pattern-breaking” tweaks to keep “raw output.”

Clarifications and compatibility notes

* The Java generator depends on a proper Java environment and PATH.
* The sound generator depends on hardware/drivers/permissions; if no valid audio is available, 0 may be returned and an error logged.
* Jinja templates must exist under the templates folder with exact names (direct.html, full\_tests.html, 404.html, 500.html) to avoid TemplateNotFound errors.

Summary  
The upgraded system delivers a practical environment for demonstration, teaching, and research—supporting comparative evaluation of multiple randomness generators and the analysis of their quality-featuring a modern UX, live progress, detailed outputs, and comprehensive logging. Its modular structure makes it easy to extend: add a generator/test, update the UI, and leverage a clear, robust testing and learning framework.

1. app\_en.js
2. MyRandomProject.java

1. [script.py](http://script.py)
2. script\_1.py
3. test\_generators.py
4. test\_tests\_module.py
5. tests\_module.py
6. README.docx
7. tyvr-mylvly-prvyyqt-vgm-3-v8-mmbnh-dvkh.docx
8. 404.html
9. 500.html
10. base.html
11. direct.html
12. full\_tests.html
13. app\_updated\_en.py
14. generator\_errors.log

1. [generators.py](http://generators.py)