Major Project Report

“Random Number Generator Using File Entropy”

by

Krishna Kant Singh • Jay Bhatt

17th May 2023

Department of Computer Science & Engineering

Central University of Rajasthan

Ajmer, Rajasthan - 305817

Acknowledgements

We are extremely grateful for this opportunity to present our project idea and work on it as part of our major project.

We would like to thank Mr. Ravi Saharan sir for his governance and guidance, because of which our whole team was able to learn the minute aspects of a project work.

We are also thankful to our whole class and most of all to our parents who have inspired us to face all the challenges and win all the hurdles in life.

Abstract

Random numbers are widely used in cryptography and are used in many algorithms in network security. They are a sequence of numbers arranged in such a way that the values present in the sequence must be evenly distributed and independent of each other. What does it imply to have random numbers? Without understanding where a group of numbers came from, it is impossible to know if they were randomly generated. However, if the process to generate these numbers is truly understood, then the generated number is not a random number.

Today random numbers are used in various types of industries not limited to but including gaming, financial institutions, secure communication, cyber security etc.

There are two types of random number generators True random number generator uses seeds or entropy source that are already present in environment and not inventing them whereas Pseudo random number generator or Deterministic random number generator are those that does not depends on environment to produce their sequence.

In this project we tried to create a true random number generator which takes useless data i.e., text files, images, videos etc. as source to generate random numbers. Before generating, we ensure that the file is encrypted. Hence the generation of random number is not controlled by anyone.

The primary objective of this project is to develop a reliable and secure true random number generator that can be used in various industries. The proposed generator has the potential to enhance the security of sensitive data and improve the performance of various applications that rely on random numbers.

Table of Contents

1) Introduction 5
2) Literature Review 6
3) Problem Statement 8
    a) Logic 8
4) Experimental Setup 9
    a) Technologies Used 9
    b) Configuration and Installation 9
5) Features 10
    a) Models 10
    b) Form 10
    c) Upload 11
    d) Generate 12
    e) User Quota 13
    f) Scheduler 13
6) Code Snippets and Screenshots 14
    a) Token Generation 14
    b) Uploaded File Handler 15
    c) File Encryption 16
    d) Database Model 16
    e) Random Number Generation request handling API 17
    f) File upload handling API 18
    g) Random Number Generator function 19
    h) Test API Script 20
    i) Statistical Analysis Script 21
    j) Screenshots 22
        i) POST request (File upload) 22
        ii) POST request (API balance) 22
        iii) POST request (Number generation) 23
        iv) Front End (Index web page) 23
        v) Front End (Token generation web page) 24
        vi) Randomness Test (Diehard test) 24
        vii) Randomness Test (Statistical analysis) 27
7) Future Works 28
8) Conclusion 28
9) References 29

Introduction

Random number generator is a function which generates a sequence of numbers that cannot be reasonably predicted, other than the fact that each number has a random chance of being generated. Usually, pseudo random number generators (PRNGs) take a seed as input. If a person knows the process of generation and value of the seed, they can easily regenerate the sequence. In contrast, there are hardware random generators (HRNGs) which depend on environment to generate the sequence, or to fetch a seed value.

There are a lot of applications of random number generators in gambling, simulation, cryptography, shuffling, nonce, salt, one-time pads etc. These are also useful in Monte Carlo methods which require simulating a scenario multiple times with random starting situations.

Currently many random number generation algorithms exist such as blum blum shub, Xorshift, Advanced Randomization System etc. They are all PRNGs and can be compromised if the seed is predicted or if there is a weakness/pattern found in the algorithm itself.

TRNGs solve this by using random properties from nature such as thermal noise, atmospheric values, etc. but these noises are inherently biased towards specific ranges. They recover from this drawback usually by Software Whitening (attempts to reduce bias after number generation) or just extracting a value from environment and using it as seed with an existing PRNG.

Our Random Number Generator is centralized at server and uses files and encryption from various clients to generate random values. This way, the client cannot predict the sequence generated. We have taken several measures to ensure that clients provide data as much as they are using the service from server.

Literature Review

Django: Django is free and open-source high-level Python web framework that encourages rapid development and clean, pragmatic design.

A **model** is the single, definitive source of information about your data. It contains the essential fields and behaviours of the data we're storing. Generally, each model maps to a single database table.

The basics:

- Each attribute of the model represents a database field.
- With all of this, Django gives us an automatically generated database-access API

Django maps the fields defined in Django **forms** [1] into HTML input fields and handles three distinct parts of the work involved in forms: [2]

- preparing and restructuring data to make it ready for rendering
- creating HTML forms for the data
- receiving and processing submitted forms and data from the client

Migrations [3] are Django's way of propagating changes we make to our models (adding a field, deleting a model, etc.) into your database schema.

We have also used **Cryptography** [4] and **Threading** [5] modules in python to encrypt input data and distributing batch operation on another thread so that our main API does not get hinderance in its performance.

Advanced Python Scheduler [6] (APScheduler) is a Python library that lets us schedule our Python code to be executed later, either just once or periodically.

APScheduler has three built-in scheduling systems you can use:

- Cron-style scheduling (with optional start/end times) using **dateutils** [7]
- Interval-based execution (runs jobs on even intervals, with optional start/end times)
- One-off delayed execution (runs jobs once, on a set date/time)

Random Number Generator and Their Applications [8] gives us insight on different types of random number generator, importance of random number generator, and application of random number in our day-to-day life. Based on their paper we concluded that for a True Random Number generator we need

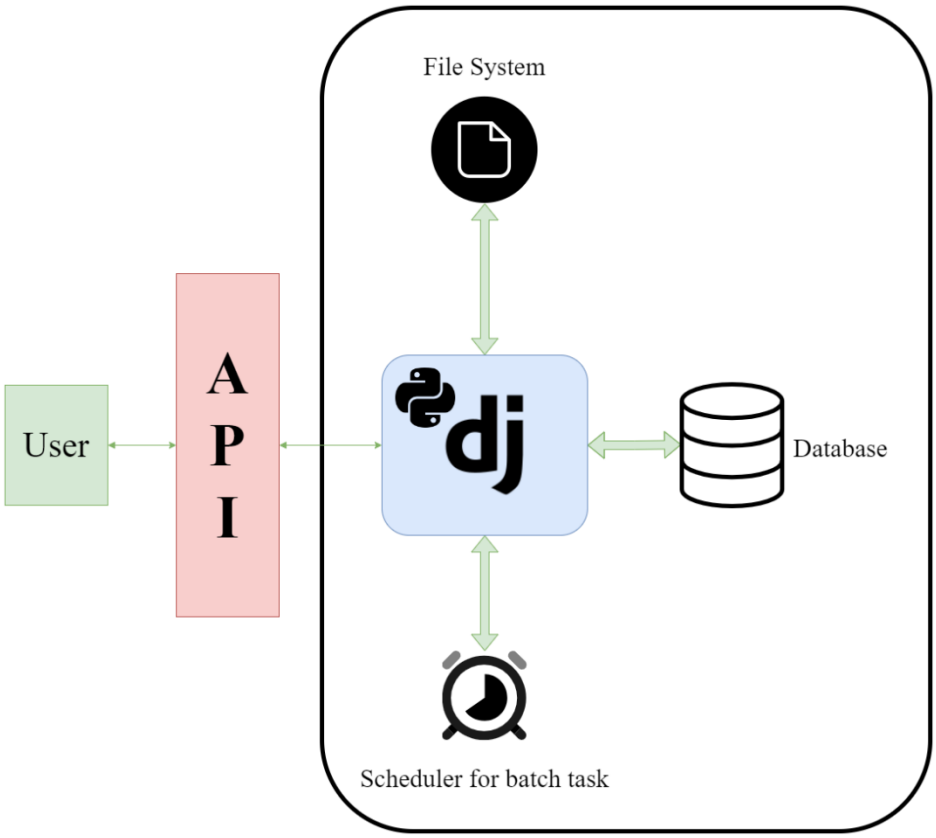
- Taking seed values from the entropy sources that are present in the physical environment.
- It consists of uniformity that means that bits are distributed uniformly overall in the sequence, the rate of occurrence of ones and zeros are roughly equal.
- Sequence should be unpredictable.

Problem Statement

To generate a reliable random number generator which can generate a random sequence which cannot be predicted by an attacker.

Logic

In our project we have created an **API** which takes files and **API token** as input from user and responds with a **random number** based on arguments supplied in API request call.



API talks to the Django application which first validates API token and then returns random number from the **queue**. There is also a **scheduler** for batch tasks like old file deletion and removing expired tokens.

Experimental Setup

Technologies Used

- **Django** as our backend framework.
- **Python** for backend request handling
- **SQLite3** as our Database Management System
- **JSON (JavaScript Object Notation)** to store and transmit data objects consisting of attribute value pairs and arrays
- **Python pip** as package manager
- **cURL and Python requests** module for testing API
- **Python venv** to create a virtual environment for running our app
- **Advanced Python Scheduler** for task scheduling

Configuration and Installation

- Install Python from https://www.python.org/downloads/
- (Optional) Switch to virtual environment by using the command

  python –m venv <environment_name>

  and use

  <environment_name>\Scripts\activate.bat

  to activate the virtual environment.

- Move complete project folder in <environment_name> directory.
- Switch to RNG directory and use the following command to install required packages:

  pip install –r requirements.txt

- Use command

  python manage.py makemigrations
  python manage.py migrate
  python manage.py runserver

  to initiate migrations and run server.

- Result of the previous command will show IP address and port on which server will be running. Now, we can easily access website and API by the given address and port.

Features

Models

- This project contains 1 Model – Token.
- Models are stored in sqlite3 database.
- Token contains various fields which are
    - token – A random number which is unique to every object. It is the primary key of Token class.
    - data – It indicates the number of bits that the token user have availability to generate. It is increased when user uploads a file and decreases when user generates a random number through the API.
    - exp – It represents the expiry date of the token. It is increased by 3 months whenever user interacts by either uploading a file or generating a random number.

Form

- It's configured such that requests other than specified for API purposes directs here.
- Here a user submits a captcha to verify himself/herself.
- After the verification, user is given a captcha/key which can be copied for use in their implementation.
- If verification failed, user is presented with Captcha Error message.
- If unused for more than 3 months, Tokens are deleted from the database by a scheduled task.

Upload

- url is located at “api/upload”.
- This is used for user to upload some data to the server, and hence increase the quota to generate more random data.
- It accepts a JSON request body which is in the following format:

```
{
    file: hex data of the uploaded file,
    token: token generated via form
}
```

- User can get an Error response depending on various situations. Error is of the format:

```
{
    error: error message,
    status: status of HttpResponse,
    message: detailed description of the error
}
```

- If successful, user gets a success response of the format:

```
{
    status: 200,
    message : “file successfully uploaded”
}
```

- Error responses are:
    - Token Not Found – token provided by request is not found in the database. It could’ve been expired or not valid. User should generate a new token in such a case.
    - Invalid JSON Request – This error is intended to tell user that his JSON request is not as per the described format. In this case, the user should edit his request to include file, token, and size fields.

Generate

- url is located at "/api/generate/".
- This is used by the user to generate random data of given size.
- It accepts a JSON request body which is in the following format:

```
{
    token: token generated via form,
    size: number of random bytes required
}
```

- User can get an Error response depending on various situations. Error Message format is as given below:

```
{
    error: error message,
    status: status of HttpResponse,
    message: detailed description of the error
}
```

- Error responses are:
    - Token Not Found – token provided by HttpRequest is not found in the database. It could've been expired or not valid. User should generate a new token in such a case.
    - Data Insufficient – In this case, token is found valid, but data/file uploaded by user is lesser than the required size of random number. User should do a request to add file to increase their data field.
    - Invalid JSON Request – This error is intended to tell user that his JSON request is not as per the described format. In this case, the user should edit his request to include file, token, and size fields.

User Quota

- url is located at “api/balance”.
- User Quota can be used by the user to know how much he can still generate from the API without adding new data via upload feature.
- It accepts a JSON request body which is in the following format:

```
{
        token: token generated via form
}
```

- User can get an Error response depending on various situations. Error Message format is as given below:

```
{
        error: error message,
        status: status of HttpResponse,
        message: detailed description of the error
}
```

- Error responses are:
    - Token Not Found – token provided by HttpRequest is not found in the database. It could’ve been expired or not valid. User should generate a new token in such a case.
    - Invalid JSON Request – This error is intended to tell user that his JSON request is not as per the described format. In this case, the user should edit his request to include file, token, and size fields.

Scheduler

- We have used Advanced Python Scheduler (APScheduler) for our scheduled scripts.
- This project has following tasks scheduled:
    - fast_crypt – This job takes all files in the “./Uploads/Temp” folder encrypts them and stores in “./Uploads/Encrypted” directory. It is scheduled to run each week.
    - delete_expired – This job scans the database to delete expired tokens. It is scheduled to run every 6 weeks.

Code Snippets and Screenshots

Token Generation

```
def keygen():
    api_token = uuid.uuid4()
    dt = datetime.date.today() + relativedelta(months=3)
    temp_token = Token(token=api_token, data=0, exp=dt)
    temp_token.save()
    return api_token


def form(request):
    if request.method == "POST":
        form = MyForm(request.POST)
        if form.is_valid():
            message = "Key: " + str(keygen())
        else:
            message = ""
    else:
        form = MyForm()
        message=''
    return render(request, generate/index.html', {'form': form, 'message':
message})
```

Uploaded File Handler

```
def writeFile(data):

    if data == "":
        return
    makeDir(UPLOAD_PATH)
    makeDir(DIR_PATH)
    makeDir(ENC_DIR_PATH)

    file_name = time.strftime("%Y%m%d-%H%M%S%p")
    file_path = DIR_PATH + file_name + ".dat"

    mode = ""

    if not os.path.isfile(file_path):
        mode = "wb"
    else:
        mode = "ab"

    with open(file_path, mode) as f:
        f.write(data.encode())
```

File Encryption

```
def encrypt_file(key, in_filename, chunksize=64*1024):

    file_name = in_filename.split('.')[0]
    out_filename = ENC_DIR_PATH + file_name + ".enc"
    in_filename = DIR_PATH + in_filename

    iv = os.urandom(16)

    encryptor = AES.new(key, AES.MODE_CBC, iv)

    if not os.path.isfile(out_filename):
        mode = "wb"
    else:
        mode = "ab"

    with open(in_filename, 'rb') as infile:
        with open(out_filename, mode) as outfile:
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    padding = b' ' * (16 - len(chunk) % 16)
                    chunk = chunk + padding

                enc_chunk = encryptor.encrypt(chunk)
                outfile.write(binascii.hexlify(enc_chunk))
```

Database Model

```
from django.db import models

class Token(models.Model):
    token = models.CharField(max_length=256, primary_key=True)
    data = models.BigIntegerField()
    exp = models.DateField()
```

Random Number Generation request handling API

```
@csrf_exempt
@require_POST
def generate(request):
    json_data = json.loads(request.body)
    try:
        token = json_data["token"]
        num_bytes = json_data["size"]
        try:
            obj = Token.objects.get(token=token)
        except:
            #token invalid -> return error / bad rng
            error = {
            "error" : "Token Not Found",
            "status" : "400",
            "message" : "Invalid token or token expired"
            }
            return JsonResponse(error)
        if num_bytes > obj.data:
            # not enough uploaded data -> return error
            error = {
            "error" : "Data Insufficient",
            "status" : "400",
            "message" : "Required size of random number too large"
            }
            return JsonResponse(error)
        randNum = randNumObj.generator(num_bytes*2)
        data = {"rnum" : randNum}
        obj.data -= num_bytes
        obj.save()
        return JsonResponse(data)
    except Exception as e:
        print(e)
        error = {
            "error" : "Invalid JSON Request",
            "status" : "400",
            "message" : "file, token or size field not found"
            }
        return JsonResponse(error)
```

File upload handling API

```
@csrf_exempt
@require_POST
def upload(request):
    json_data = json.loads(request.body)
    try:
        file = json_data["file"]
        token = json_data["token"]
        t = Thread(target=writeFile, args=[file])
        t.run()
        try:
            obj = Token.objects.get(token=token)
        except:
            #token invalid -> return error / bad rng
            error = {
            "error" : "Token Not Found",
            "status" : "400",
            "message" : "Invalid token or token expired"
            }
            return JsonResponse(error)
        # add file size to obj.data, increases time and saves
        obj.data += len(file)*8
        obj.exp =  datetime.date.today() + relativedelta(months=3)
        obj.save()
        success = {
            "status" : "200",
            "message" : "File successfully uploaded."
            }
        return JsonResponse(success)
    except Exception as e:
        print(e)
        error = {
            "error" : "Invalid JSON Request",
            "status" : "400",
            "message" : "file, token or size field not found"
            }
        return JsonResponse(error)
```

Random Number Generator function

```
class dataQueue:
    maxSize = 20971520*2 # 20 MB
    r, w = os.pipe() # creates a pipe which resides in memory
    currSize = 0

    def __init__(self):
        pass

    def fill(self):
        try:
            files = os.listdir(ENC_DIR_PATH)
        except Exception as e:
            print(f"\033[1;31;40m[!] File not exists: \033[0m{e}")
            return

        if len(files) <= 0:
            fast_crypt.entry()

        for file in files:
            file_path = ENC_DIR_PATH + file
            try:
                with open (file_path, "rb") as f:
                    self.currSize += os.write(self.w, f.read()) #
os.write(w, b) returns num of bytes which is added to current size
            except Exception as e:
                print("Error: ", e)
                pass
            self.renameFile(file)
            if self.currSize > self.maxSize: # exit condition
                break

    def generator(self, num_bytes):
        if self.currSize <= num_bytes: # fill if not availaible. Note: use
offset and multiprocessing if possible, we can pass writing file descripter
and it should work using os.fork(). Issue: updating currSize
            self.fill()
        try:
            self.currSize -= num_bytes
            num = os.read(self.r, num_bytes)
            num = int(num.decode(), 16)

        except Exception as e:
            print(f"\033[1;31;40m[!] Empty Stream: \033[0m{e}")

        return num
```

Test API Script

```
import requests
import sys

args = sys.argv
token = <Your token>

if len(args) > 1:
    if args[1] == "GEN" or args[1] == "gen":
        url = 'http://localhost:8000/api/generate/'

        myobj = {'token': token, 'size': int(args[2])}

        for i in range(int(args[3])):
            x = requests.post(url, json = myobj)
            print(f"Request no. {i} ", x.text)
    elif args[1] == "UPD" or args[1] == "upd":
        url = 'http://localhost:8000/api/upload/'
        data = <Text to upload>

        myobj = {'file': data, 'token': token}
        x = requests.post(url, json = myobj)

    elif args[1] == "BAL" or args[1] == "bal":
        url_bal = 'http://localhost:8000/api/balance/'
        bal = {'token': token}
        print(requests.post(url_bal, json = bal).text)

else:
    print(f"""
Options:
    {args[0]} [GEN/gen] <b> <n> : Generate n numbers of b bits.

    {args[0]} [BAL/bal] : Get balance & validity of your token.
    """)
```

Statistical Analysis Script

```
import statistics
from scipy.stats import norm

numbers = []

with open("test.csv", "r") as f:
    for data in f.readlines():
        num, count = data.strip().split(", ")
        for i in range(int(count)):
            numbers.append(int(num))


mean = statistics.mean(numbers)
std_dev = statistics.stdev(numbers)

z_scores = [(x - mean) / std_dev for x in numbers]
p_values = [2 * (1 - abs(norm.cdf(x))) for x in z_scores]
significant_values = [x for x in p_values if x < 0.05]

if len(significant_values) > 0:
    print("The numbers are not truly random.")
else:
    print("The numbers appear to be truly random.")
```

Screenshots

POST request (File upload)

```
(rng) PS E:\Programming\Projects\Major Project\rng\RNG> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 15, 2023 - 16:33:55
Django version 4.1.3, using settings 'rng.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[15/May/2023 16:35:06] "POST /api/upload/ HTTP/1.1" 200 59
```

```
PS E:\Programming\Projects\Major Project\rng\Test> python .\api_test.py upd
PS E:\Programming\Projects\Major Project\rng\Test>
```

POST request (API balance)

```
(rng) PS E:\Programming\Projects\Major Project\rng\RNG> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 15, 2023 - 16:33:55
Django version 4.1.3, using settings 'rng.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[15/May/2023 16:35:06] "POST /api/upload/ HTTP/1.1" 200 59
[15/May/2023 16:42:26] "POST /api/balance/ HTTP/1.1" 200 45
```

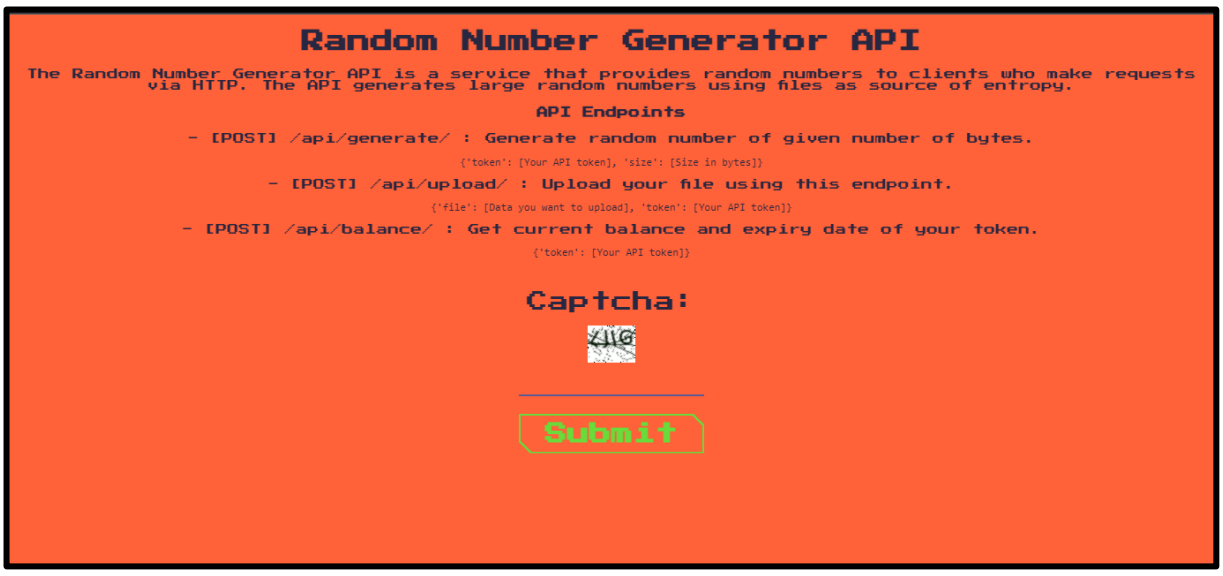
```
PS E:\Programming\Projects\Major Project\rng\Test> python .\api_test.py bal
{"remaining": 1848, "validity": "2023-08-15"}
PS E:\Programming\Projects\Major Project\rng\Test>
```

POST request (Number generation)

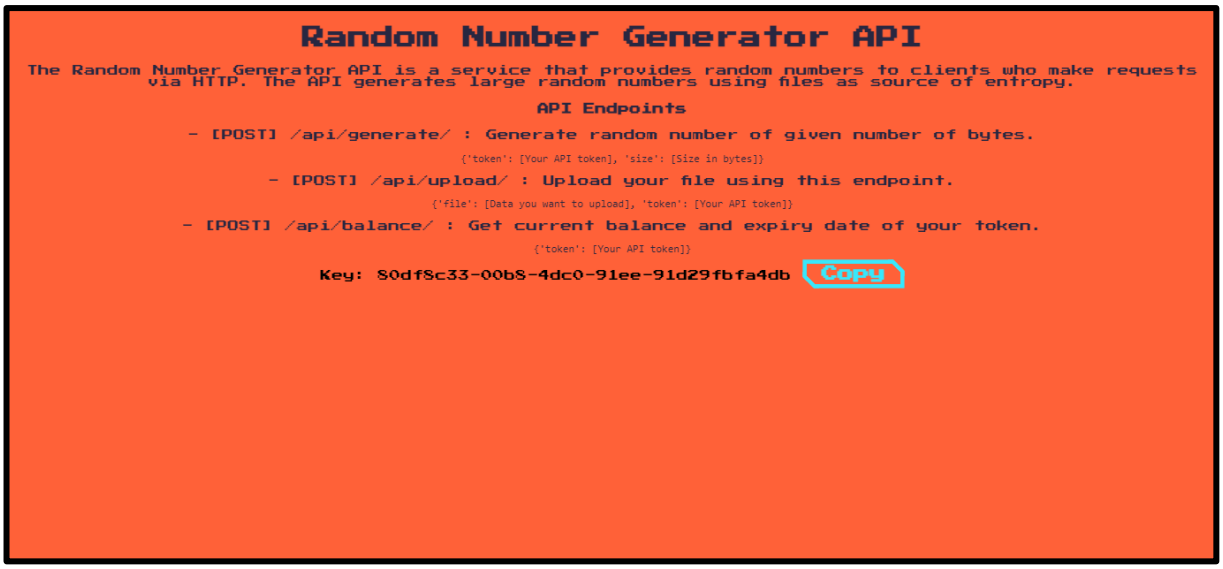
```
May 15, 2023 - 16:57:17
Django version 4.1.3, using settings 'rng.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[15/May/2023 16:57:26] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:28] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:30] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:32] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:34] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:36] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:38] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:41] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:43] "POST /api/generate/ HTTP/1.1" 200 20
[15/May/2023 16:57:45] "POST /api/generate/ HTTP/1.1" 200 20
```

```
PS E:\Programming\Projects\Major Project\rng\Test> python .\api_test.py gen 4 10
Request no. 0  {"rnum": 3677218910}
Request no. 1  {"rnum": 1208478405}
Request no. 2  {"rnum": 1292517022}
Request no. 3  {"rnum": 1049760540}
Request no. 4  {"rnum": 2585290780}
Request no. 5  {"rnum": 2851638293}
Request no. 6  {"rnum": 2670717658}
Request no. 7  {"rnum": 2246669796}
Request no. 8  {"rnum": 1339349709}
Request no. 9  {"rnum": 2655090156}
```

Front End (Index web page)



Front End (Token generation web page)



Randomness Test (Diehard test)

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
        mt19937|                        randnums|  1.13e+08  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
   diehard_birthdays|   0|       100|     100|0.98903492|  PASSED
      diehard_operm5|   0|   1000000|     100|0.27805415|  PASSED
  diehard_rank_32x32|   0|     40000|     100|0.00493638|   WEAK
    diehard_rank_6x8|   0|    100000|     100|0.06221906|  PASSED
   diehard_bitstream|   0|   2097152|     100|0.87967465|  PASSED
         diehard_opso|   0|   2097152|     100|0.72070984|  PASSED
         diehard_oqso|   0|   2097152|     100|0.12937783|  PASSED
          diehard_dna|   0|   2097152|     100|0.88964137|  PASSED
diehard_count_1s_str|   0|    256000|     100|0.86156593|  PASSED
diehard_count_1s_byt|   0|    256000|     100|0.31767414|  PASSED
 diehard_parking_lot|   0|     12000|     100|0.61013501|  PASSED
    diehard_2dsphere|   2|      8000|     100|0.64713335|  PASSED
    diehard_3dsphere|   3|      4000|     100|0.36591967|  PASSED
     diehard_squeeze|   0|    100000|     100|0.70004732|  PASSED
        diehard_sums|   0|       100|     100|0.05468930|  PASSED
        diehard_runs|   0|    100000|     100|0.98939918|  PASSED
        diehard_runs|   0|    100000|     100|0.91939977|  PASSED
       diehard_craps|   0|    200000|     100|0.34755313|  PASSED
       diehard_craps|   0|    200000|     100|0.57273927|  PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.87421561|  PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.31240526|  PASSED
```

sts_monobit	1	100000	100	0.04263310	PASSED
sts_runs	2	100000	100	0.45481317	PASSED
sts_serial	1	100000	100	0.52022066	PASSED
sts_serial	2	100000	100	0.26928342	PASSED
sts_serial	3	100000	100	0.35040781	PASSED
sts_serial	3	100000	100	0.87067406	PASSED
sts_serial	4	100000	100	0.17410933	PASSED
sts_serial	4	100000	100	0.29334201	PASSED
sts_serial	5	100000	100	0.01772750	PASSED
sts_serial	5	100000	100	0.53964735	PASSED
sts_serial	6	100000	100	0.00619718	PASSED
sts_serial	6	100000	100	0.49950722	PASSED
sts_serial	7	100000	100	0.12859867	PASSED
sts_serial	7	100000	100	0.13215410	PASSED
sts_serial	8	100000	100	0.00963406	PASSED
sts_serial	8	100000	100	0.14497138	PASSED
sts_serial	9	100000	100	0.02881192	PASSED
sts_serial	9	100000	100	0.40020557	PASSED
sts_serial	10	100000	100	0.08360199	PASSED
sts_serial	10	100000	100	0.77124075	PASSED
sts_serial	11	100000	100	0.16346752	PASSED
sts_serial	11	100000	100	0.93639996	PASSED
sts_serial	12	100000	100	0.04487777	PASSED
sts_serial	12	100000	100	0.20162880	PASSED
sts_serial	13	100000	100	0.53117136	PASSED
sts_serial	13	100000	100	0.29919353	PASSED
sts_serial	14	100000	100	0.50006415	PASSED
sts_serial	14	100000	100	0.03875559	PASSED
sts_serial	15	100000	100	0.12960749	PASSED
sts_serial	15	100000	100	0.36043107	PASSED
sts_serial	16	100000	100	0.33100165	PASSED
sts_serial	16	100000	100	0.99014112	PASSED

rgb_bitdist	1	100000	100	0.81283444	PASSED
rgb_bitdist	2	100000	100	0.49466130	PASSED
rgb_bitdist	3	100000	100	0.92006120	PASSED
rgb_bitdist	4	100000	100	0.27017585	PASSED
rgb_bitdist	5	100000	100	0.90672418	PASSED
rgb_bitdist	6	100000	100	0.63126914	PASSED
rgb_bitdist	7	100000	100	0.41853697	PASSED
rgb_bitdist	8	100000	100	0.44186700	PASSED
rgb_bitdist	9	100000	100	0.76545597	PASSED
rgb_bitdist	10	100000	100	0.80792114	PASSED
rgb_bitdist	11	100000	100	0.58850458	PASSED
rgb_bitdist	12	100000	100	0.85216864	PASSED
rgb_minimum_distance	2	10000	1000	0.38444238	PASSED
rgb_minimum_distance	3	10000	1000	0.87745305	PASSED
rgb_minimum_distance	4	10000	1000	0.97624893	PASSED
rgb_minimum_distance	5	10000	1000	0.48316781	PASSED
rgb_permutations	2	100000	100	0.15249103	PASSED
rgb_permutations	3	100000	100	0.53447729	PASSED
rgb_permutations	4	100000	100	0.96913753	PASSED
rgb_permutations	5	100000	100	0.99484469	PASSED
rgb_lagged_sum	0	1000000	100	0.53443714	PASSED
rgb_lagged_sum	1	1000000	100	0.96522427	PASSED
rgb_lagged_sum	2	1000000	100	0.82378979	PASSED
rgb_lagged_sum	3	1000000	100	0.69897518	PASSED
rgb_lagged_sum	4	1000000	100	0.83620654	PASSED
rgb_lagged_sum	5	1000000	100	0.86069419	PASSED
rgb_lagged_sum	6	1000000	100	0.92666139	PASSED
rgb_lagged_sum	7	1000000	100	0.94910790	PASSED
rgb_lagged_sum	8	1000000	100	0.57443723	PASSED
rgb_lagged_sum	9	1000000	100	0.81640788	PASSED
rgb_lagged_sum	10	1000000	100	0.83660125	PASSED
rgb_lagged_sum	11	1000000	100	0.98766511	PASSED

```
      rgb_lagged_sum|  12|   1000000|     100|0.76361008|  PASSED
      rgb_lagged_sum|  13|   1000000|     100|0.98200303|  PASSED
      rgb_lagged_sum|  14|   1000000|     100|0.99405584|  PASSED
      rgb_lagged_sum|  15|   1000000|     100|0.57309525|  PASSED
      rgb_lagged_sum|  16|   1000000|     100|0.05717457|  PASSED
      rgb_lagged_sum|  17|   1000000|     100|0.98396213|  PASSED
      rgb_lagged_sum|  18|   1000000|     100|0.46981825|  PASSED
      rgb_lagged_sum|  19|   1000000|     100|0.17764177|  PASSED
      rgb_lagged_sum|  20|   1000000|     100|0.96643880|  PASSED
      rgb_lagged_sum|  21|   1000000|     100|0.03200008|  PASSED
      rgb_lagged_sum|  22|   1000000|     100|0.58392175|  PASSED
      rgb_lagged_sum|  23|   1000000|     100|0.77611539|  PASSED
      rgb_lagged_sum|  24|   1000000|     100|0.74190386|  PASSED
      rgb_lagged_sum|  25|   1000000|     100|0.58756164|  PASSED
      rgb_lagged_sum|  26|   1000000|     100|0.68172237|  PASSED
      rgb_lagged_sum|  27|   1000000|     100|0.17262647|  PASSED
      rgb_lagged_sum|  28|   1000000|     100|0.20656423|  PASSED
      rgb_lagged_sum|  29|   1000000|     100|0.80285983|  PASSED
      rgb_lagged_sum|  30|   1000000|     100|0.02915791|  PASSED
      rgb_lagged_sum|  31|   1000000|     100|0.81004447|  PASSED
      rgb_lagged_sum|  32|   1000000|     100|0.16654965|  PASSED
     rgb_kstest_test|   0|     10000|    1000|0.75204349|  PASSED
     dab_bytedistrib|   0|  51200000|       1|0.97890120|  PASSED
             dab_dct| 256|     50000|       1|0.95486456|  PASSED
Preparing to run test 207.  ntuple = 0
        dab_filltree|  32|  15000000|       1|0.31143511|  PASSED
        dab_filltree|  32|  15000000|       1|0.54644297|  PASSED
Preparing to run test 208.  ntuple = 0
       dab_filltree2|   0|   5000000|       1|0.98584618|  PASSED
       dab_filltree2|   1|   5000000|       1|0.23778869|  PASSED
Preparing to run test 209.  ntuple = 0
        dab_monobit2|  12|  65000000|       1|0.26202204|  PASSED
```

Randomness Test (Statistical analysis)

```
PS E:\Programming\Projects\Major Project\rng\Test> python .\statistical_test.py
The numbers appear to be truly random.
```

Future Works

- Create a dedicated **File System** for handling encrypted file for optimized Read/Write.
- Implement the project as a micro service.
- Optimize generation time for random data.

Conclusion

In this report, we have covered the initial problem statement for our project. We have covered the technology used in our project with brief description. We have outlined the project features and experimental setup in detail. We have designed abstract logic flow diagram which gives us abstract view of our overall idea in very easy to understand form.

References

[1] “Working with forms,” [Online]. Available: https://docs.djangoproject.com/en/4.1/topics/forms/.

[2] N. Arora, “Django Forms,” [Online]. Available: https://www.geeksforgeeks.org/django-forms/.

[3] “Migrations,” [Online]. Available: https://docs.djangoproject.com/en/4.1/topics/migrations/.

[4] “Cryptography,” [Online]. Available: https://pypi.org/project/cryptography/.

[5] “Thread-based parallelism,” [Online]. Available: https://docs.python.org/3/library/threading.html.

[6] “APScheduler,” [Online]. Available: https://apscheduler.readthedocs.io/en/3.x/userguide.html.

[7] “dateutil,” [Online]. Available: https://pypi.org/project/python-dateutil/.

[8] Priyanka, H. I. K. and A. , “Random Number Generators and their Applications: A Review,” vol. 7, pp. 1777-1781, 2019.

[9] “Models,” [Online]. Available: https://docs.djangoproject.com/en/4.1/topics/db/models/.

[10] “Using django-simple-captcha,” [Online]. Available: https://django-simple-captcha.readthedocs.io/en/latest/usage.html.

[11] “Memory-mapped file support,” [Online]. Available: https://docs.python.org/3/library/mmap.html.

[12] “Miscellaneous operating system interfaces,” [Online]. Available: https://docs.python.org/3/library/os.html.

Thank You