Software Implementation and Tests Specification

for

HE-DNS SERVER

Prepared by Yohai Mor-Yosef & Noy Meran

Table of Contents

1.1 Requirements

1. Check references.
2. Read about DNS
3. Read about Homomorphic Encryption
4. Create a git repo named “HE-DNS”, upload and work on that repo
5. Build project skeleton (define functions in files all files)
6. Add comments to skeleton
7. Implement server.py
8. Implement db_util.py
9. Implement he_util.py
10. Use Python 3+
11. Use the following packages:
    1. Flask - Server package
    2. Request - HTTP/S package
12. Use MySQL as Database
13. Work with Git
14. Use unittest to test your code
15. Write at least 3 tests for each function
16. Comment every 3 lines
17. HTTP/S communication should be based on REST API

2. Project Implementation Description:

Our project divides to three parts Server side part and Client-side part and test:

Server side *include 3 files:*

Run.py*- this class in charge to send a URL request to server and start the communication. First we imported the Flask class, an instance of this class will be our WSGI APPLICATION.*
Implementation includes 4 functions:

I Set() (Domain and IP)

II Print()

III Activate Server (listen to request)

Server.py- *this class is connecting to MySql and create a db_util instance and calling is functions.*

Db_util.py*- This class do SQL queries and by that get all the information of the domains and ip's. therefore, server class calling db-util funcrions. In order to interact with the database we use Cursor object in this class.*

Client side *include 1 file:*

Menu.py – *This part in the project guaranty our server works right we check so by applying requests to our server. Request means: we'll give him a URL name and once its exist in DB DNS table – IP will return to the client.*

Test *include 1 file:*

Tests.py – *Here we want to check if all functions work correctly by 6 tests. (18 in total).*

3. Test Implementation Description:

In our test we have tried to cover all kind of inputs.
Since we are dealing with hostnames and IPs we have to make sure our, given by client and admin server, data is correct under the conditions.
For example:
IP should be in special format: 0-255.0-255. 0-255. 0-255 so we make sure is fit to our need by using a regex format.
The next 4 tests belong to server-side part.

TEST1: create entry, via createNewEntry api function

3 kinds of hostnames were checked:

- *nana%20.com – which should be failed since "%" not allowed.*
- *..mako - which should be failed since ".." not allowed.*
- *ads.youtube.com – legal domain name – allowed.*

TEST2: Set IP, using setIp api function

8 kinds of IP were checked:

- *walla.co.il – illegal IP.*
- *45456 - illegal IP.*
- *12.45.3- illegal IP.*
- *12.26- illegal IP.*
- *312.12.1.44- illegal IP.*
- *12.451.3.10- illegal IP.*
- *123.1.257.3- illegal IP.*
- *10.43.20.21 – legal IP.*

TEST3: CREATE\CHECK existance domain via setDomain API function

Dns server table handle entity, which combine from IP and URL, the method, like "one-to-many" style, all the entities must be different in their domain name but could share same IP target. That's why we want to check that we are not allow to set 2 entities which sharing same domain name.

3 tests has made:

- *google.co.il – successfully added*
- *google.co.il – Error! Already exist*
- *walla.co.il - successfully added*

TEST4: printing DNS table

Server administrator can print the DNS table, so we want to make sure the entities successfully added and table printed correctly.

- *We first ask to print a table (option b) - which we'll be announced the table is empty.*
- *Second step will be to Set an entity (option a):*
  Hostname: walla.co.il IP: 10.43.121.3
- *Now we want to check if our entity correctly added.*

TEST5: get ip for non-existing domain

This test belong to client-side part, we want to make sure our server knows to deal with unfamiliar domain names.

- *We assume "google.co.il" not exist in our DNS table so we'll ask to translate his domain name. (should be Error)*

TEST6: get ip from existing domain

Now we want to check the basic option of our server, getting the IP of familiar domain name.

- *We ask to translate: walla.co.il and the return value will be: 10.43.121.3*

We have in total 18 tests – once we have 18/18 than our tests were succeed.

4. Build Project:

We use Anaconda Navigator environment using Spyder (Python 3.7)

Installed Packages: flask, threading, os, re, urllib, request

Step 1: *what would you like to execute? Test or server?*

For running the server (as server admin) make sure to open 3 files which described in page 2.

Step 2: *make sure variable "debug" is* ***off*** *(line 15 in run.pu):*

Editor - C:\Users\Yohai Mor\Desktop\DNSproject\New folder\LastVersion\run.py

temp.py | text.py | hw3.py | hw4.py | hw2.py | hw1

```
1 #!/usr/local/bin/python3.7
2 from flask import Flask #import the framework
3 import threading #handles the concurrency
4 import server #import the server functionality
5 import time
6 # ==========================================================
7 # #create flask app and routes
8 # ==========================================================
9 app=Flask("dns_server")
10 app.debug=False
11 app.use_reloader=False
12 # ==========================================================
13 #  this needs to be true if tests should be run
14 # ==========================================================
15 debug=False
16 # ==========================================================
17 # #this route is for getting ip from domain, takes 1 argumen
18 # ==========================================================
```

Step 3: *run the run.py file: you'll have such a manu:*

```
In [1]: runfile('C:/Users/Yohai Mor/Desktop/DNSproject/New f
folder/LastVersion')


Server-Adminstrator Menu:
a. Set new domain and IP
b. Print DNS table
c. Activate the server - listening to requests

What would you like to do? (a,b,c):
```

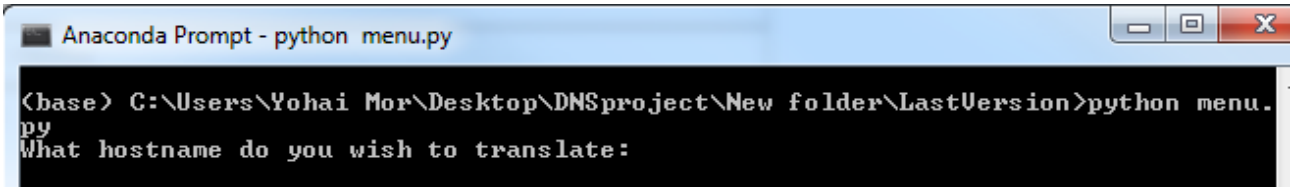
Step 4: *now, all what you need to do is choose one action, the default DNS table is:*

```
_________________________________
illegalcarrots.com : 76.211.123.2
hireusednapkins.uk : 92.122.65.1
notyomama.com : 76.165.22.8
howtotieyourshoes.com : 42.99.132.22
moonpenguins.hurrdurr : over 9000
t : 1
---------------------------------
```

Step 5: *Once you have set the desired IP & Hostnames you can activate the server by write "c" and than the server will be activate.*

Step 6: *Client server build: make sure the menu.py is in the same path of the 3 server files. Once you've done, please open python prompt and change directory to the specific path you've just chosen:*

And now write the next command: python menu.py



We succeed to communicate with the server and now we can ask for translations.

Steps – End.

5. Build Test:

Step1: *what would you like to execute? Test or server?*

For running the Test make sure:

- *Open 4 files which described in page 2. All files except the menu.py.*
- *Make sure the server is not in listening mode.*

Step 2: *make sure variable "debug" is* ***On*** *(line 15 in run.pu):*

Step 3: *build the tests.py and see the results. The expected result:*

```
TOTAL [True] TEST RESULT= 18/18
```

Steps – End.

Make sure: once you've build the tests, you'll need to change inputes directly to the code. For example: the default DNS table doesn't have: walla.co.il and once you build – it is set in the table so the next build it will show you ERROR cause you already have the host name in the table.