Bangladesh University of Engineering and Technology

# Project Report on *ISP Censorship Analyzer*

Course Code: CSE 326

Course Name: Information System Design Sessional

| *Author:* | *Student ID:* |
|---|---|
| Mahim Mahbub | 1505022 |
| Md Hasan Al Kayem | 1505023 |
| Md. Imran Hasan | 1505026 |

September 3, 2020

## Acknowledgements

We would like to express our gratitude and appreciation to our respected teachers

- Dr. Anindya Iqbal Sir
  Associate Professor
  Department of CSE, BUET
- Mahjabin Nahar Madam
  Lecturer
  Department of CSE, BUET
- Nafis Irtiza Tripto Sir
  Lecturer
  Department of CSE, BUET

# Contents

# List of Figures

# 1 Introduction

People all over the world has different opinion regarding censorship in any communication medium. Internet being the most popular communication medium of this age, has enhanced this dispute. At one end of the spectrum, countries like China and Russia censor almost all internet traffic. On the other end, nations like United states and Japan have traditionally been observed as supporters of internet freedom. In between, countries like Bangladesh are attempting to apply internet censorship over the last few years. According to Freedom on the Net, Bangladesh is among the list of countries that have partly free internet. However, the project failed to identify the mechanisms of internet censorship in Bangladesh. Therefore, we aim to investigate the existing internet censorship mechanism in Bangladesh.

Internet censorship in any country is generally controlled by the country's government while the country's public and private internet service providers implement the censorship. First step of our project will be the collection of blocked websites from Bangladesh government. With required permissions, Various governmental organizations like Access to Information, Bangladesh Telecommunication Regulatory Commission, and Ministry of Information Technology can help us collect this information.

The next step of our project will be to investigate whether all the major internet service providers enforce the government provided censorship in their traffic. In that regard, one could use existing tools like open observatory of network interference (OONI), BlockCheck, and cachebrowser but these tools are subjected to high percentage of false positives and false negatives. Therefore, we will build an automated system from bottom up to verify results ourselves. This will be a desktop application and could be used by anyone. For major internet service providers, we will be testing with mobile networks such as Banglalink, Grameenphone, Robi, etc and try to understand the behaviour of these mobile networks with respect to censored websites. Using our system, we will try to understand how major internet service providers enforce the censorship. Different techniques to enforce the censorship include DNS blocking, TCP/IP packet filtering, HTTP filtering, Active probing, etc. Among these techniques, the effort to implement and elude the censorship varies widely. Moreover, these techniques have other side effects like unexpected traffic blocking and sudden internet throttling and shutdown. Therefore, it is of much importance to study the approaches adopted by major internet service providers.

## 2 System overview

Our system has 6 major subsystems. Brief descriptions of these major subsystems are listed below.

1. **Authentication Subsystem** : This subsystem simply makes a user login to his/her account and if account does not exist, then he/she can register and make an account using email address. A simple email verification also exists.
2. **DNS Blocking Check Subsystem**: This subsystem, in short, checks whether a given url is censored with respect to DNS blocking censorship technique.
3. **HTTP Filtering Check**: This subsystem, in short, checks whether a given url is censored with respect to HTTP filtering censorship technique.
4. **TCP Filtering Check**: This subsystem, in short, checks whether a given url is censored with respect to TCP filtering censorship technique.
5. **Censorship Tracking**: This subsystem, in short, checks whether a given url is censored and finds out which method is used. Above subsystems work as kind of independent checking whereas this is kind of a sequential checking starting with DNS checking, then TCP filtering checking and finally ending with HTTP filtering checking.
6. **General Facility**: This subsystem is for the interaction with the user. Users can check status using ping, tracert etc. Users can also review their history.

The actors of our system include:

- **Logged User**: This is a primary business actor. A user must first have to be registered and logged in to have access to the system.
- **Admin**: This is a primary system actor. Admin has the right to delete or modify any data stored in the database of system server.
- **System Server**: External Server Actor. Stores data of the user and the reports generated.
- **Local Server (ISP)**: Another External Server Actor. Responds to dns query request. Either sends an IP (or a list of IPs) or does not send back anything.
- **Proxy Server (eg. TOR Browser)**: Another External Server Actor. Responds to dns query request. Sends an IP (or a list of IPs) or does not send anything if a url is invalid.
- **Web Server**: Another External Server Actor. System tries to initiate a TCP 3-way handshake.

## 3 Use case diagram

### 3.1 Authentication Subsystem

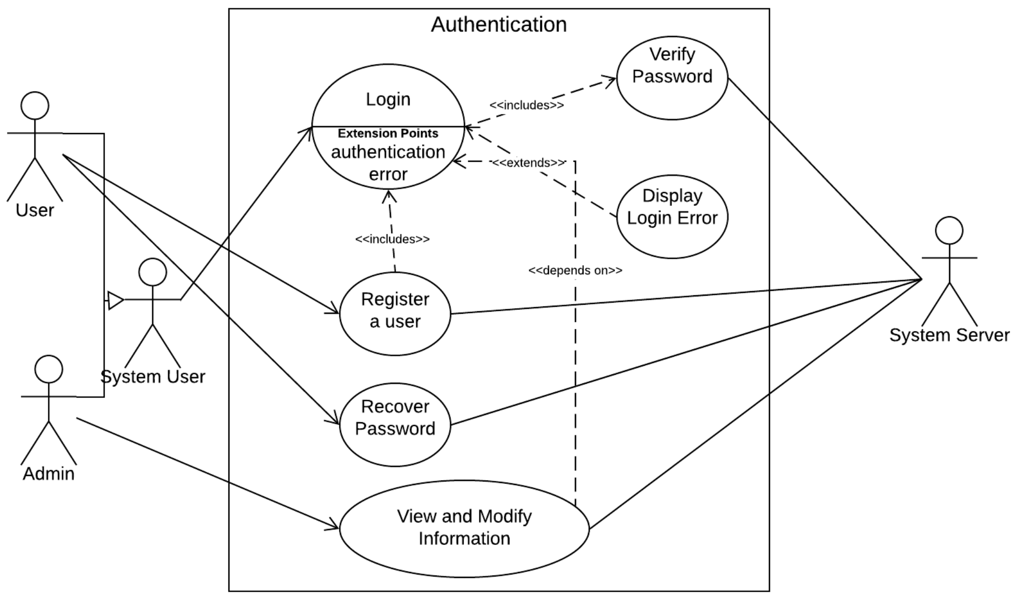


Figure 1: Use Case of Authentication Subsystem

In authentication subsystem, a user can register in this system and log in to use full feature of our system. Admin can view and modify information of whole system server.

## 3.2 DNS Blocking Check

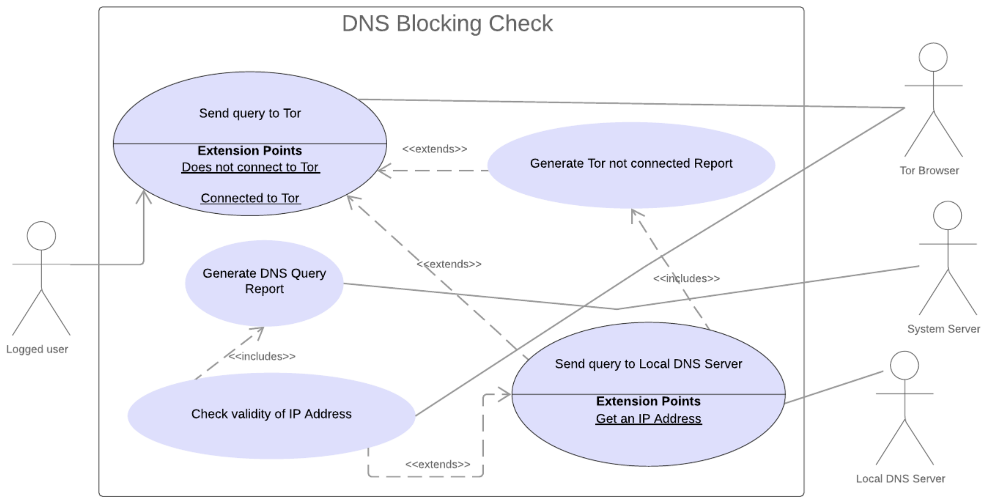


Figure 2: Use Case of DNS Blocking Check Subsystem

For DNS Blocking Check subsystem, We will try to figure out if a given url is blocked using DNS Blocking or DNS poisoning techniques. First to verify the validity of url, we will send a query to Tor Browser (used for proxy server). If it does not connect then the url is invalid and we will generate invalid url report (Does not connect to Tor report). If it does connect, from Tor browser, we shall retrieve IP(s) against this url and do the same query (nslookup) with the local server (ISP). If we do not get any IP address, then we can conclude that DNS related censorship was used. Otherwise, using Tor again, we shall check the validity of the IP(s) got from Local Server(ISP) and generate the report accordingly.

## 3.3 HTTP Filtering Check

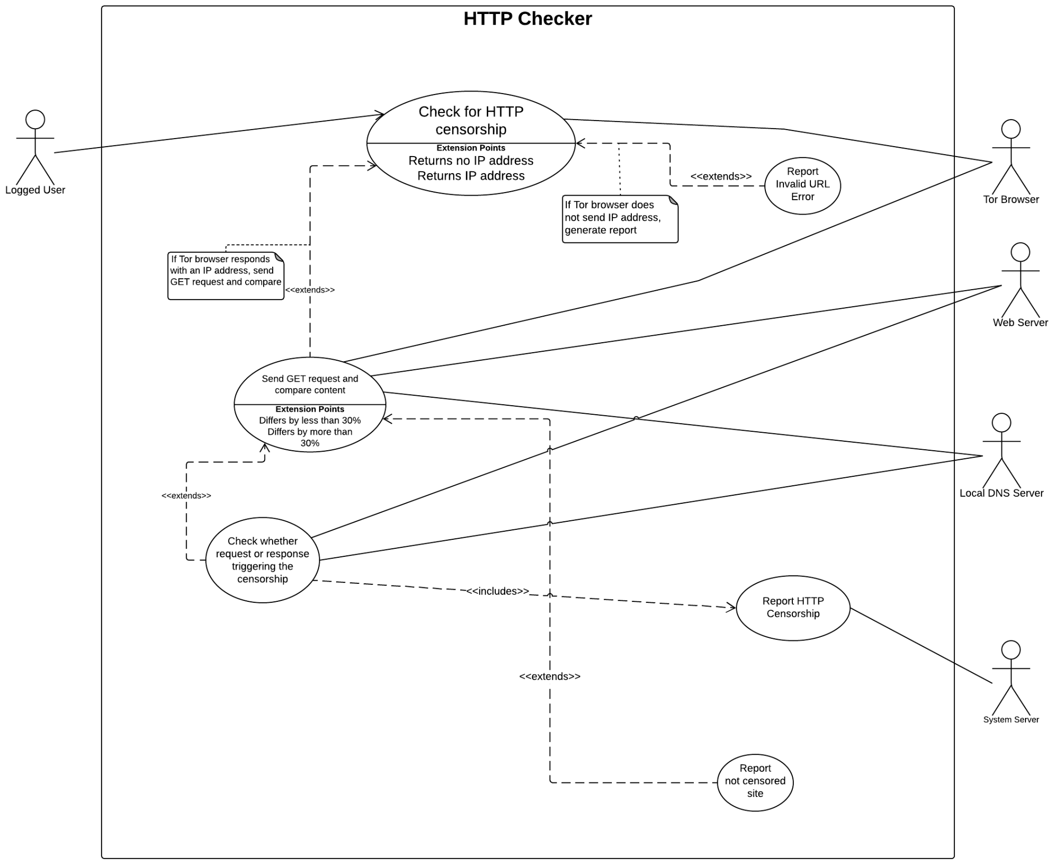


Figure 3: Use Case of HTTP Filtering Check Subsystem

For HTTP Filtering, we will assume that our local server does return IP(s) for a given url successfully (i.e. no DNS related censorship for this url). So, after initial validation of url (i.e. checking if proxy browser eg. TOR browser does connect to the url), we shall use this IP(or list of IPs) to send HTTP GET request(s) and retrieve the content from proxy browser. Then by doing nslookup using local server, we shall send the same GET request(s) using IP(s) obtained from local server to retrieve this content. We shall then compare the two contents and if their mismatch lies beyond a certain threshold (which shall be experimentally determined), we would conclude HTTP related censorship and form a report accordingly for users to analyze.

## 3.4 TCP Filtering Check

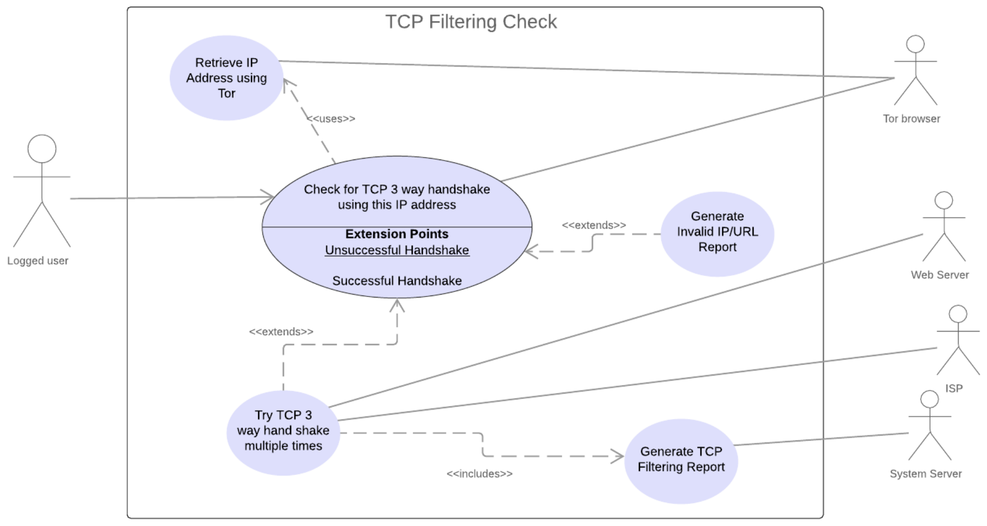

TCP Filtering Check
Retrieve IP Address using Tor
<<uses>>
Check for TCP 3 way handshake using this IP address
Extension Points
Unsuccessful Handshake
Successful Handshake
<<extends>>
Generate Invalid IP/URL Report
<<extends>>
Try TCP 3 way hand shake multiple times
<<includes>>
Generate TCP Filtering Report
Logged user
Tor browser
Web Server
ISP
System Server

Figure 4: Use Case of TCP Filtering Check Subsystem

For TCP filtering as well, we shall assume no DNS related censorship has occured (since our three methods could be checked independently and in conjunction as well). With this assumption, we shall retrieve IP address(or list of IPs) from a proxy browser(eg. TOR) and try to establish a TCP three-way handshake with the proxy server. This is done to ensure that the website is not temporarily down or has not been moved temporarily when we are running our system. Then by getting IP(s) from local server, we shall try to establish three-way handshake multiple times with the website after regular intervals. If all of the attempts are unsuccessful, we would conclude that there may be TCP filtering done. User may try again after a while to confirm. This is probabilistic since TCP filtering simply does not allow three way handshake which has the same effect as dropping a packet in route to the destination. Dropping of a packet could be for various reasons such as congestion control, flow control, etc. So, for now, we shall try establishing 3-way handshake multiple times at regular intervals to do this check. If another novel approach does arrive, we may implement that approach.

## 3.5 Censorship Tracking Use Case

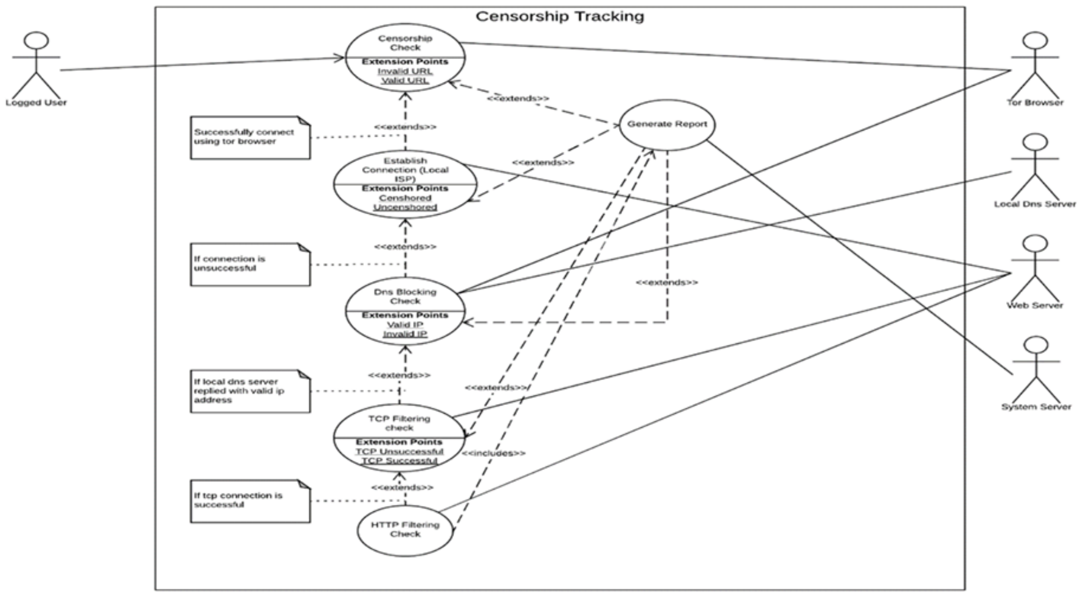


Figure 5: Use Case of Censorship Tracking

In this subsystem, we will implement the whole Censorship checking tests that we previously implemented independently. First we will check for DNS censorship, if we cannot conclude that DNS censorship exists, then we will check for TCP filtering censorship. Failing to conclude that, we will finally test for HTTP related censorship. The techniques used here are very much similar to the techniques used in the independent subsystems.

## 3.6 General Facility Use Case

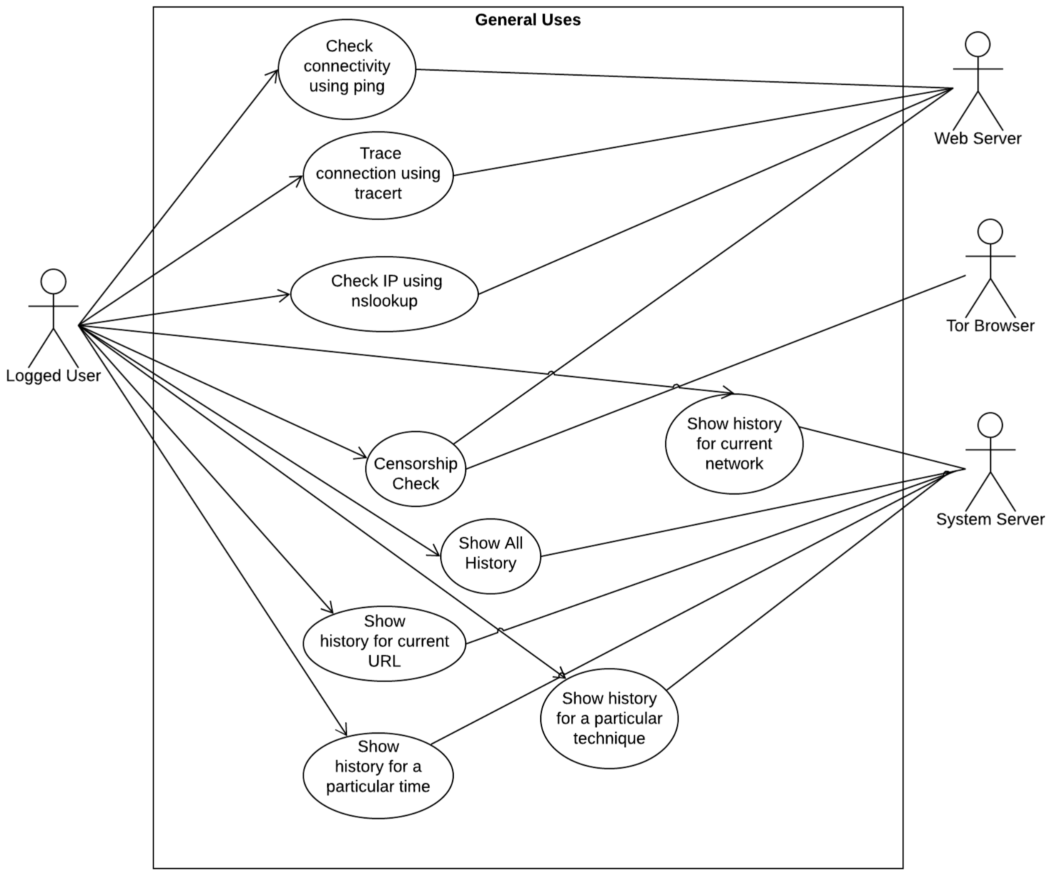

General Uses
Check connectivity using ping
Trace connection using tracert
Check IP using nslookup
Censorship Check
Show history for current network
Show All History
Show history for current URL
Show history for a particular technique
Show history for a particular time
Logged User
Web Server
Tor Browser
System Server

Figure 6: Use Case of General Facility Subsystem

This is for the users to view their history of the reports generated by the methods described above. Also, users may run normal tools like nslookup, tracert, ping from our system to explore themselves.

## 4 Entity Relationship Diagram

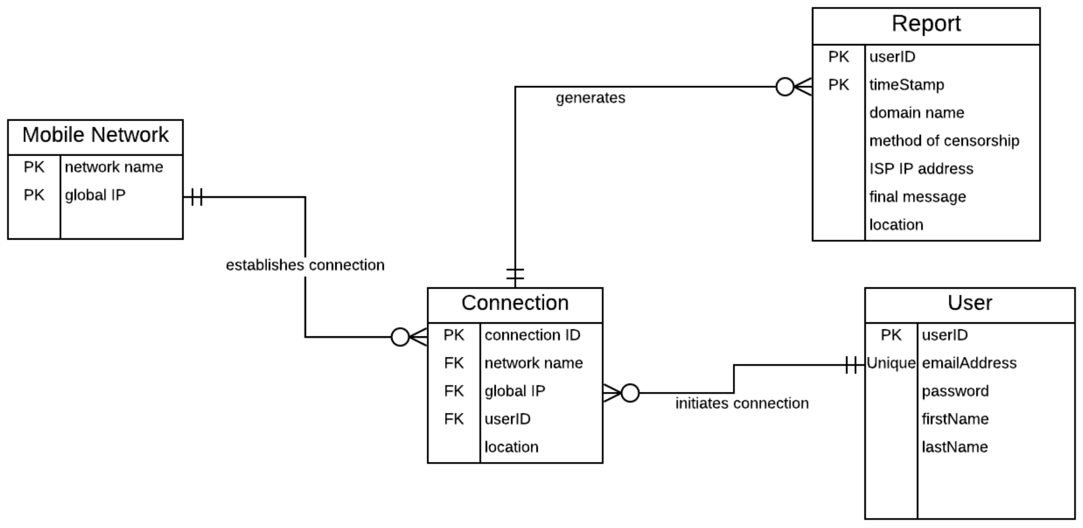


Figure 7: Entity Relationship Diagram for Our System

# 5 Class diagrams

## 5.1 DNS Blocking Check Subsystem

This is the class diagram of the DNS Blocking Check Subsystem. We have followed Model-View-Controller design pattern for this subsystem.

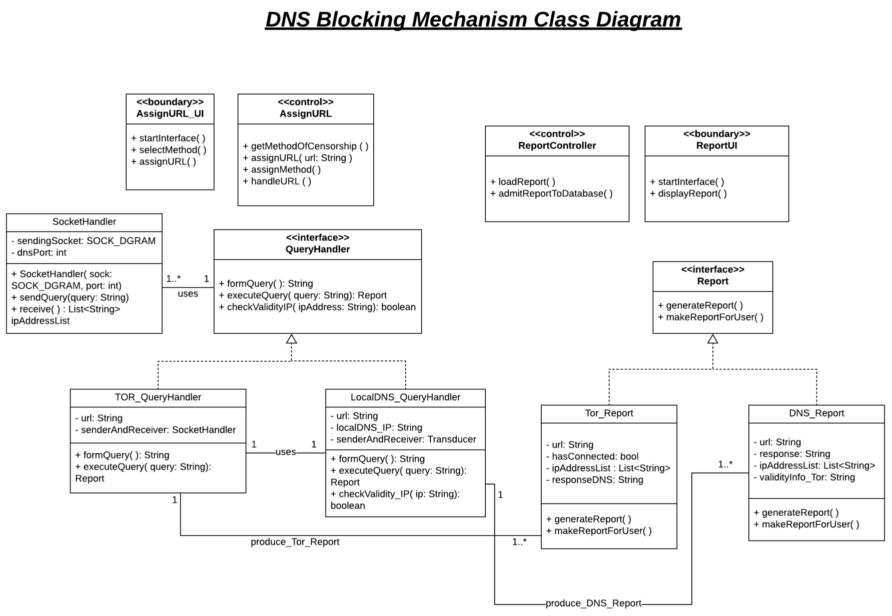

DNS Blocking Mechanism Class Diagram
<<boundary>> AssignURL_UI
+ startInterface( )
+ selectMethod( )
+ assignURL( )
<<control>> AssignURL
+ getMethodOfCensorship ( )
+ assignURL( url: String )
+ assignMethod( )
+ handleURL ( )
<<control>> ReportController
+ loadReport( )
+ admitReportToDatabase( )
<<boundary>> ReportUI
+ startInterface( )
+ displayReport( )
SocketHandler
- sendingSocket: SOCK_DGRAM
- dnsPort: int
+ SocketHandler( sock: SOCK_DGRAM, port: int)
+ sendQuery(query: String)
+ receive( ) : List<String> ipAddressList
1..*
1
uses
<<interface>> QueryHandler
+ formQuery( ): String
+ executeQuery( query: String): Report
+ checkValidityIP( ipAddress: String): boolean
<<interface>> Report
+ generateReport( )
+ makeReportForUser( )
TOR_QueryHandler
- url: String
- senderAndReceiver: SocketHandler
+ formQuery( ): String
+ executeQuery( query: String): Report
1
uses
1
LocalDNS_QueryHandler
- url: String
- localDNS_IP: String
- senderAndReceiver: Transducer
+ formQuery( ): String
+ executeQuery( query: String): Report
+ checkValidity_IP( ip: String): boolean
Tor_Report
- url: String
- hasConnected: bool
- ipAddressList : List<String>
- responseDNS: String
+ generateReport( )
+ makeReportForUser( )
DNS_Report
- url: String
- response: String
- ipAddressList: List<String>
- validityInfo_Tor: String
+ generateReport( )
+ makeReportForUser( )
1
produce_Tor_Report
1..*
1
1..*
produce_DNS_Report

Figure 8: Class Diagram of DNS Blocking Check Subsystem

## 5.2 HTTP Filtering Check Subsystem

This is the class diagram of the HTTP Filtering Check Subsystem. We have followed Model-View-Controller design pattern for this subsystem as well.

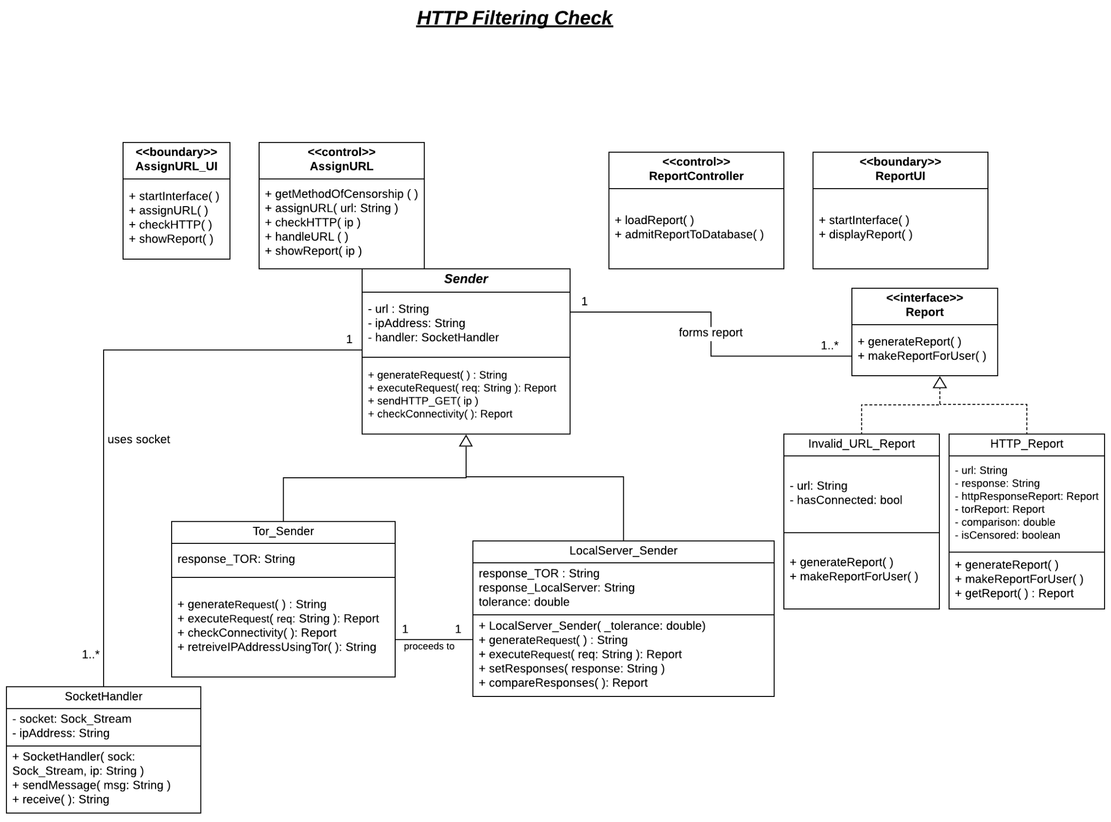


Figure 9: Class Diagram of HTTP Filtering Check Subsystem

## 5.3 TCP Filtering Check Subsystem

This is the class diagram of the TCP Filtering Check Subsystem. We have followed Model-View-Controller design pattern for this subsystem as well.

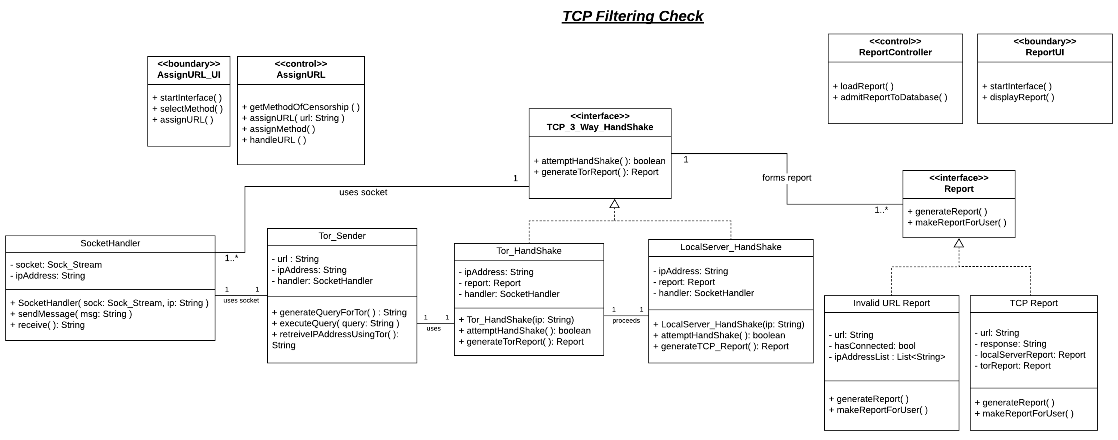


Figure 10: Class Diagram of TCP Filtering Check Subsystem

# 6 Sequence diagram

## 6.1 URL Validity Check Sequence Diagram

In this subsystem, first the control class starts up the initial UI window of the ui class. Then the Logged user assigns a url using assignURL() which upon receiving by the control class, it calls the getIPAddressList method using its instance of the TOR_Sender class which instantiates a new object of Socket_Handler class and calls the getIP_set method. Then control is then finally returned to the user and the CheckURL_UI class displays whether the url is valid or not to the user.

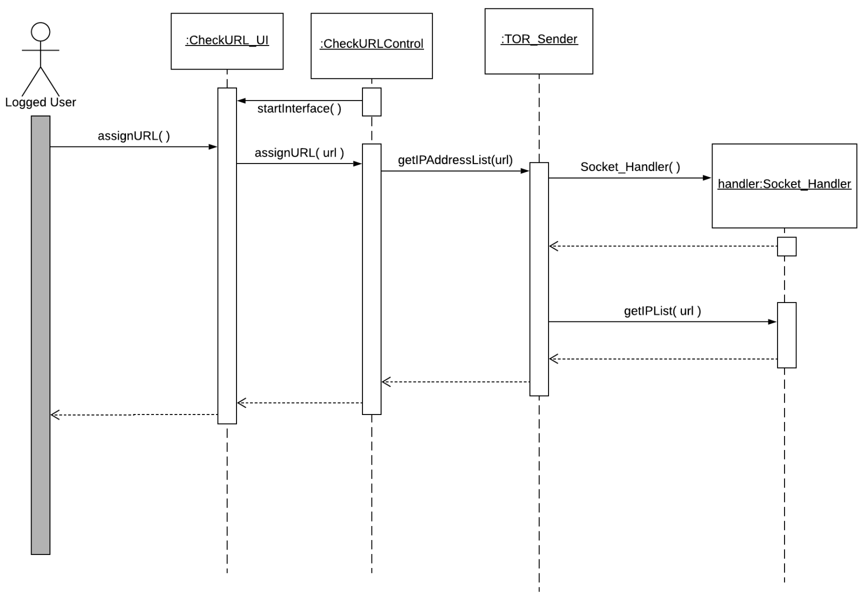


Figure 11: Sequence Diagram of URL Validity Check Subsystem

## 6.2 HTTP Filtering Check Sequence Diagram

In this subsystem, just like the previous one, first the control class starts up the initial UI window of the ui class. Now the logged user checks for whether HTTP filtering censorship is applied to this url. Control class calls the sendHTTP method of its instance of TOR_Sender class. Now the instance of TOR_Sender class instantiates a new object of Socket_Handler class and executes HTTP GET request (and keeps the content of the request) via proxy browser (TOR Browser). Then the TOR_Sender class's instance instantiates a new object of LocalServer_Sender class. This does the same thing as the instance of TOR_Sender class but now content is retrieved by sending HTTP GET request to local server (ISP) rather than proxy browser (TOR browser). Finally compareResponses method is called and a new report object is instantiated of the Report class and control is returned back to the user and the report is displayed. To further display the report again, the user may call the showReport method.

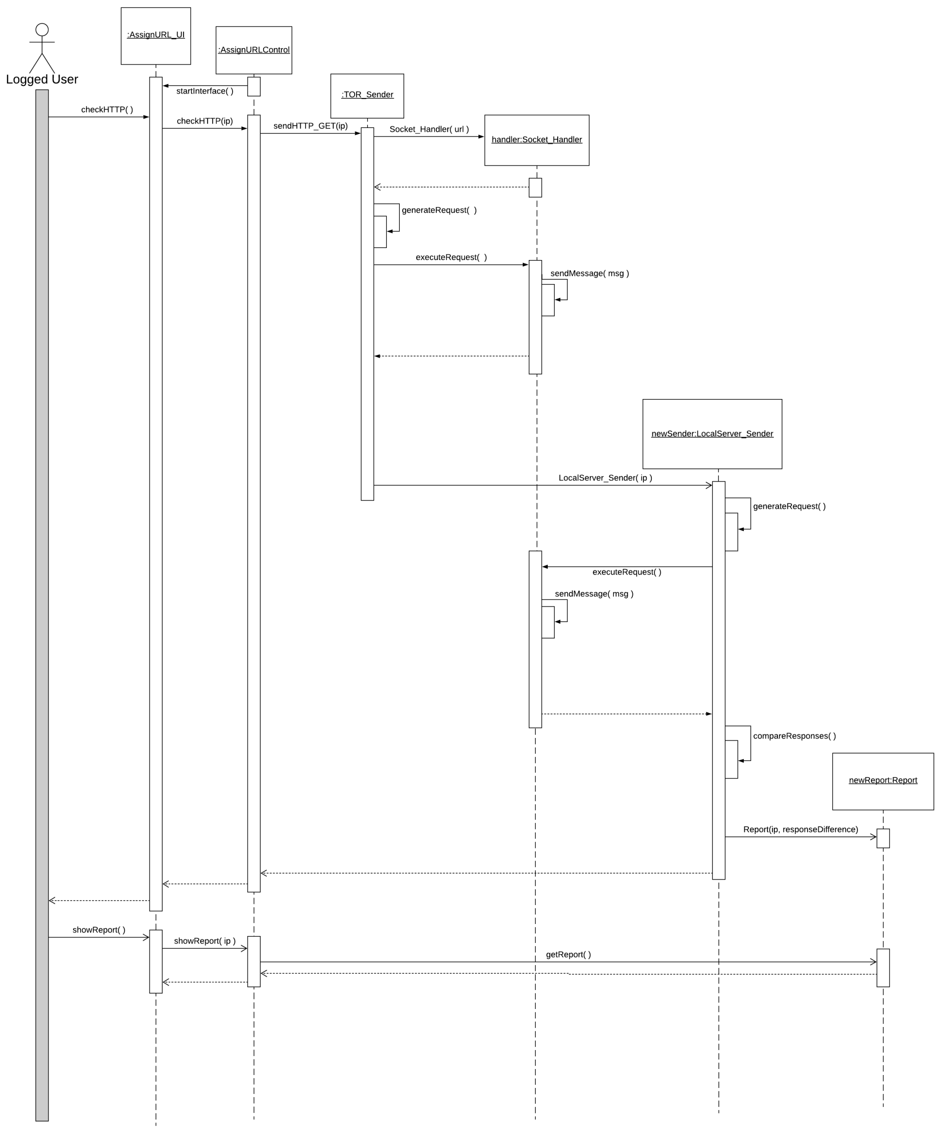


Figure 12: Sequence Diagram of HTTP Filtering Check Subsystem

# 7 Data flow diagram

## 7.1 DNS Blocking Subsystem

In this subsystem, Logged User (entity) sends a url to the subsystem. Subsystem then does a dns query to Local DNS Server (Query Message), then it sends back IP address list (IP Address Set).If no IP is returned, Generate Report process is called (Null IP Report data flow) and corresponding report is stored in DNS Report Master data store. On the other hand if IP Address set is returned, using this IP Address set, first we try to connect to TOR Browser. Depending on the response, a suitable report is generated and stored in DNS Report Master data store.

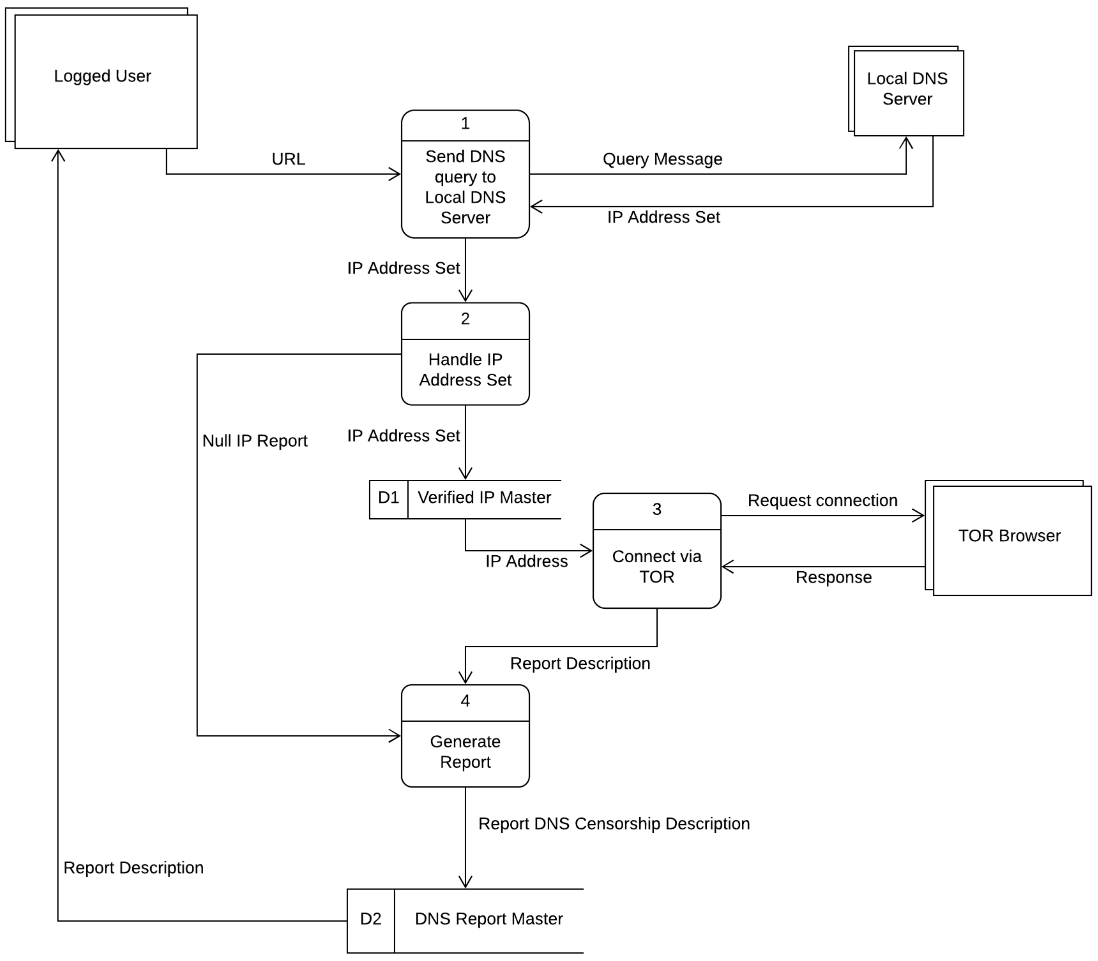


Figure 13: Data Flow Diagram of DNS Blocking Subsystem

## 7.2 TCP Filtering Check Subsystem

In this subsystem, Logged User (entity) sends a url to the subsystem.

Subsystem then sends a query to Proxy Server, TOR Browser in this case (Query Message data flow), then it sends back IP address list (IP Address Set). Just like the previous subsystem, if no IP is returned, Generate Report process is called (Null IP Report data flow) and corresponding report is stored in DNS Report Master data store.

On the other hand if IP Address set is returned, first it is stored in a Verified IP Master data store, then this IP Address set is used to establish TCP 3 way hand shake using proxy browser.

If unsuccessful after multiple tries, this IP Address set is stored in IP Address Website Down Report Master data store and Generate TCP Report process is called and required report is formed and stored finally in TCP Report Master data store(D3).

However, if proxy browser does connect, then subsystem tries to esablish TCP 3-way handshake using local server (Process 6) and generates required TCP Report using Process 5(Generate TCP Report). Final report is stored in TCP Report Master data store (D3).

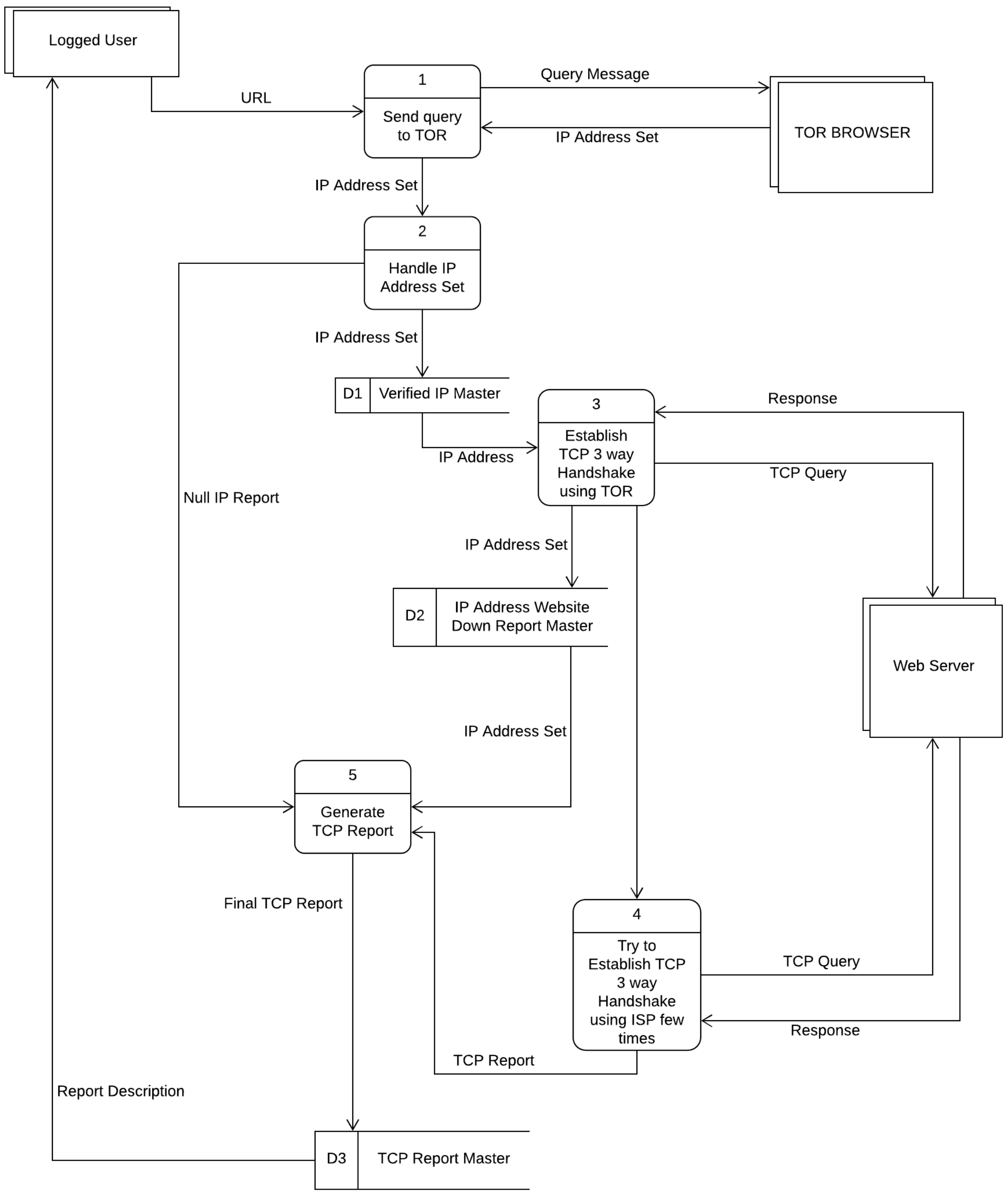


Figure 14: Data Flow Diagram of TCP Filtering Subsystem

## 8 Gantt chart

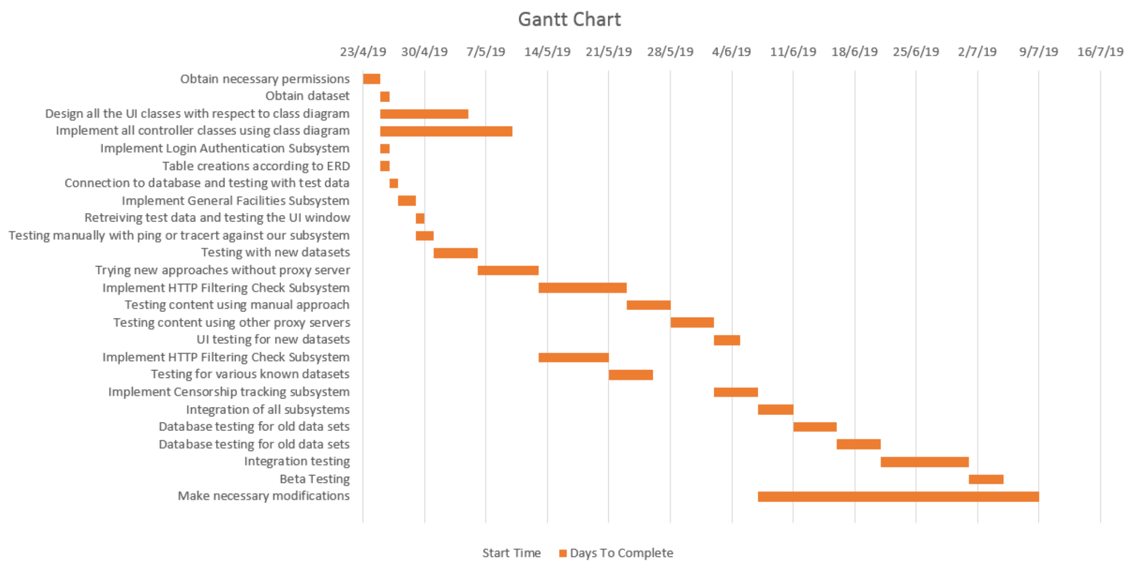


Figure 15: Gantt Chart of ISP Censorship Analyzer

## 9 Implementation Example

Our implementation is for the DNS Blocking Check Subsystem. Initial UI screen is simple. User may enter a single url, or by choosing a file, may enter a file where there can be multiple urls. After clicking the "Click me" button, the dns checking program runs and user can see in real time various outputs of various checkings and the final output.

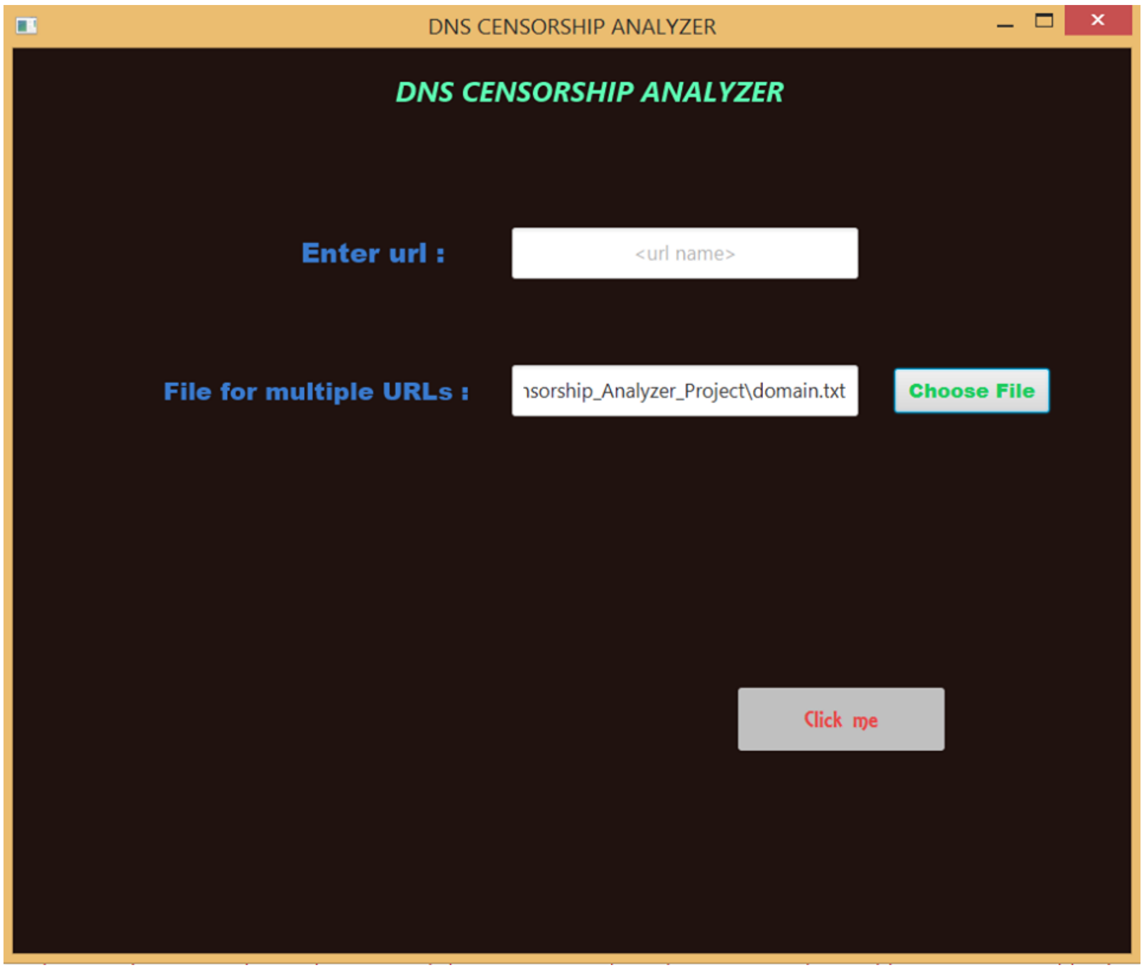
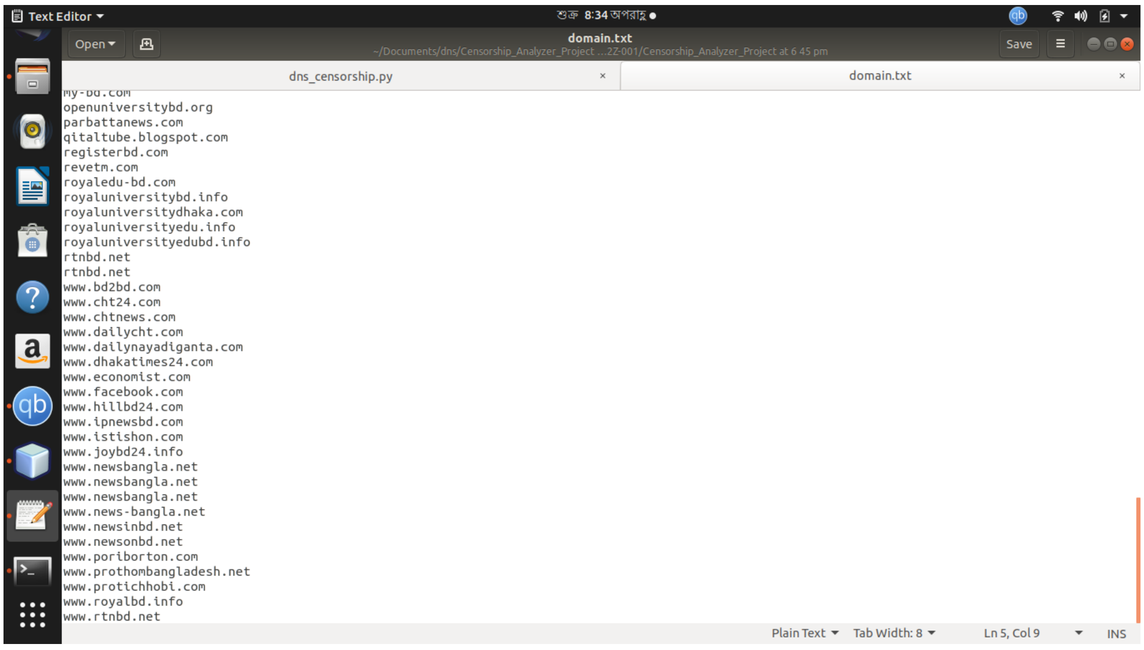


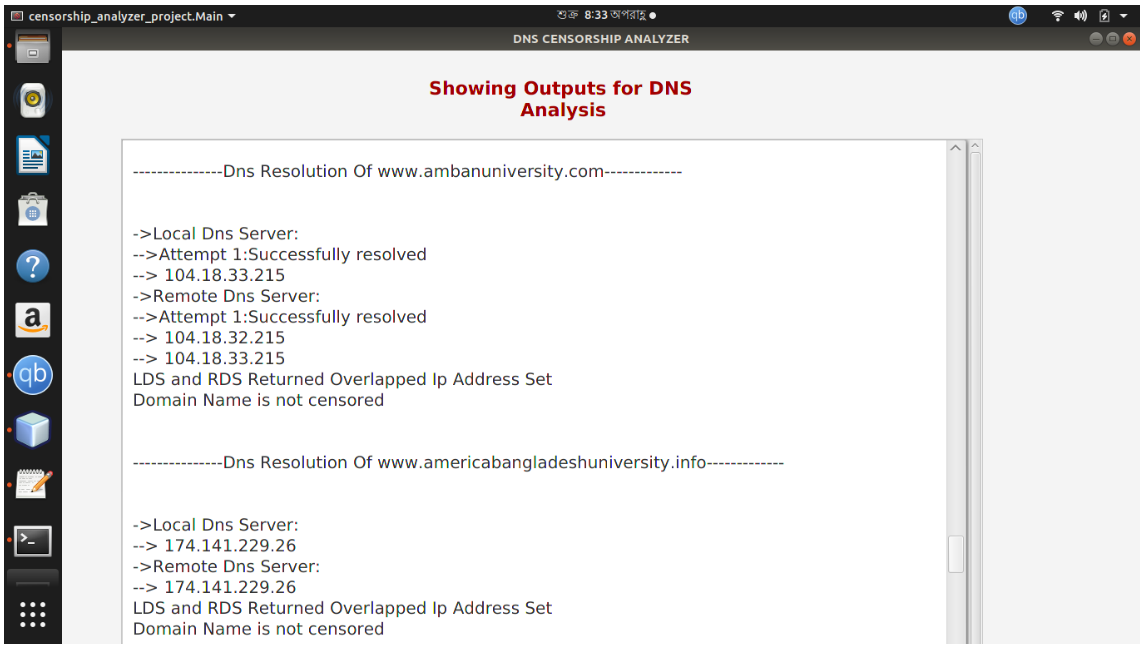
Figure 16: Input UI Window



```
my-bd.com
openuniversitybd.org
parbattanews.com
qitaltube.blogspot.com
registerbd.com
revetm.com
royaledu-bd.com
royaluniversitybd.info
royaluniversitydhaka.com
royaluniversityedu.info
royaluniversityedubd.info
rtnbd.net
rtnbd.net
www.bd2bd.com
www.cht24.com
www.chtnews.com
www.dailycht.com
www.dailynayadiganta.com
www.dhakatimes24.com
www.economist.com
www.facebook.com
www.hillbd24.com
www.ipnewsbd.com
www.istishon.com
www.joybd24.info
www.newsbangla.net
www.newsbangla.net
www.newsbangla.net
www.news-bangla.net
www.newsinbd.net
www.newsonbd.net
www.poriborton.com
www.prothombangladesh.net
www.protichhobi.com
www.royalbd.info
www.rtnbd.net
```

Figure 17: Sample input file

This is a sample input file which contains multiple urls for users to test whether dns blocking method is used for censorship or not.

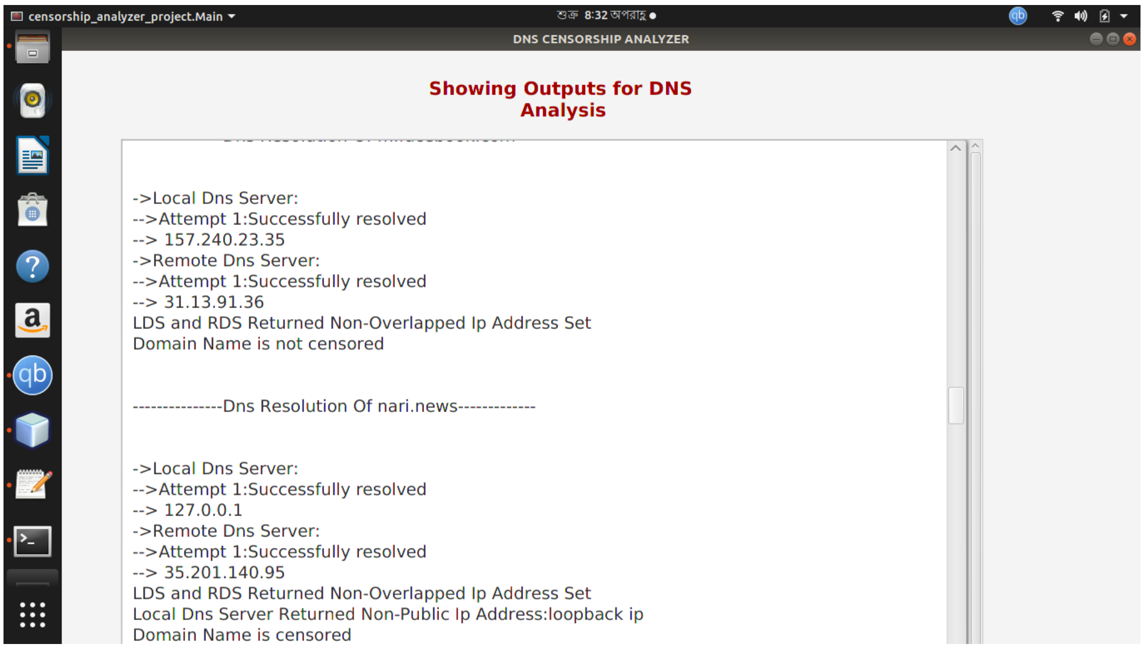


**Showing Outputs for DNS Analysis**

```
---------------Dns Resolution Of www.ambanuniversity.com-------------


->Local Dns Server:
-->Attempt 1:Successfully resolved
--> 104.18.33.215
->Remote Dns Server:
-->Attempt 1:Successfully resolved
--> 104.18.32.215
--> 104.18.33.215
LDS and RDS Returned Overlapped Ip Address Set
Domain Name is not censored


---------------Dns Resolution Of www.americabangladeshuniversity.info-------------


->Local Dns Server:
--> 174.141.229.26
->Remote Dns Server:
--> 174.141.229.26
LDS and RDS Returned Overlapped Ip Address Set
Domain Name is not censored
```

Figure 18: DNS Censorship report (Both are uncensored)

In this sample output of two such urls, www.ambanuniversity.com and www.americabangladeshuniversity.info, we observe that Local Server(LDS) and Remote DNS Server(proxy server) has overlapping IP address set, 104.18.33.215 for www.ambanuniversity.com and 174.141.229.26 for www.americabangladeshuniversity.info, so we can conclude that DNS related censorship is not applied to either of these urls.

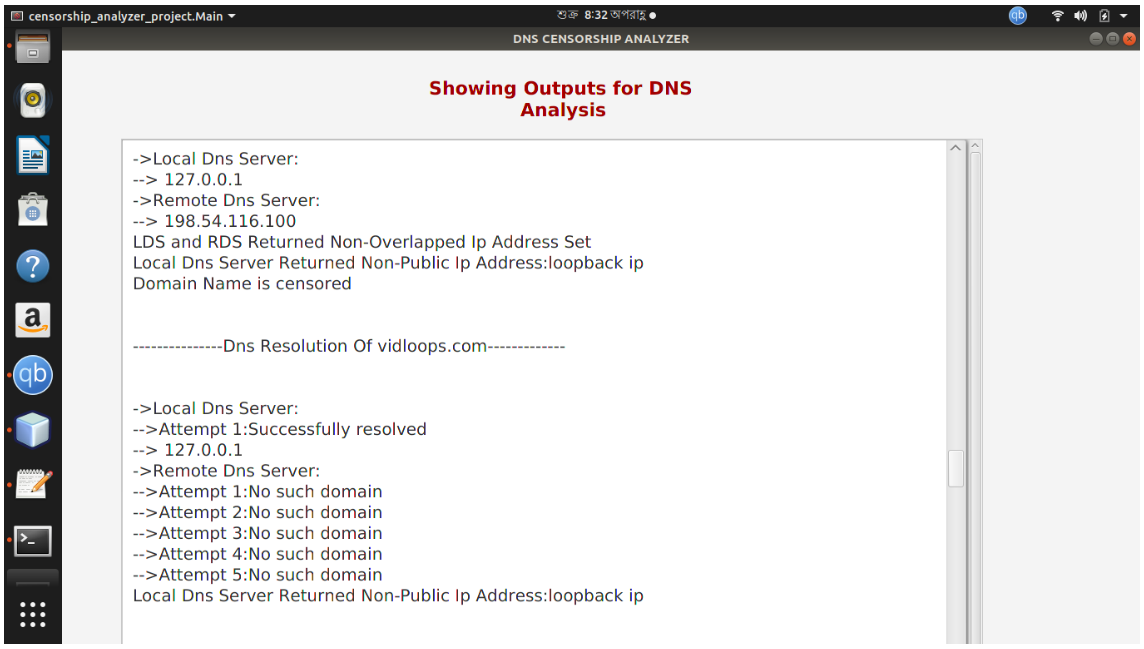


**Showing Outputs for DNS Analysis**

```
->Local Dns Server:
-->Attempt 1:Successfully resolved
--> 157.240.23.35
->Remote Dns Server:
-->Attempt 1:Successfully resolved
--> 31.13.91.36
LDS and RDS Returned Non-Overlapped Ip Address Set
Domain Name is not censored


---------------Dns Resolution Of nari.news-------------


->Local Dns Server:
-->Attempt 1:Successfully resolved
--> 127.0.0.1
->Remote Dns Server:
-->Attempt 1:Successfully resolved
--> 35.201.140.95
LDS and RDS Returned Non-Overlapped Ip Address Set
Local Dns Server Returned Non-Public Ip Address:loopback ip
Domain Name is censored
```

Figure 19: DNS Censorship report (DNS Blocking due to Loopback IP)

In this sample output for nari.news, local server provides a loopback ip so dns blocking can be concluded.



Showing Outputs for DNS Analysis

```
->Local Dns Server:
--> 127.0.0.1
->Remote Dns Server:
--> 198.54.116.100
LDS and RDS Returned Non-Overlapped Ip Address Set
Local Dns Server Returned Non-Public Ip Address:loopback ip
Domain Name is censored


---------------Dns Resolution Of vidloops.com-------------


->Local Dns Server:
-->Attempt 1:Successfully resolved
--> 127.0.0.1
->Remote Dns Server:
-->Attempt 1:No such domain
-->Attempt 2:No such domain
-->Attempt 3:No such domain
-->Attempt 4:No such domain
-->Attempt 5:No such domain
Local Dns Server Returned Non-Public Ip Address:loopback ip
```

Figure 20: DNS Censorship report (DNS Blocking due to Loopback IP)

In this sample output for www.vidloops.com, local server provides a loopback ip so dns blocking can be concluded.

# 10 Conclusion

All in all, we wish to build such a system that can detect censored websites and techniques used to censor them, with as much certainty as possible, rather than relying on existing tools such as OONI(open observatory of network interference), BlockCheck etc. We wish to add as many checking as possible for as many corner cases we can figure out. We also wish to make the system as user friendly as possible. Many benefits may come from such a system like detecting whether a url is actually censored or not when it should be. Or whether it is censored by an ISP, when it should not be. Also normal users who are eager to learn about computer networking and censorship techniques may explore this area. We hope our system may provide a starting point for such users.

## 11 References

[1] How OONI detects HTTP Filtering? https://ooni.torproject.org/nettest/web-connectivity/.
[2] HTTP 1.1 RFC 2616. https://tools.ietf.org/html/rfc2616.
[3] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Sharma, Sambuddho Chakravarty.
Where The Light Gets In: Analyzing Web Censorship Mechanisms in India
https://arxiv.org/pdf/1808.01708.pdf