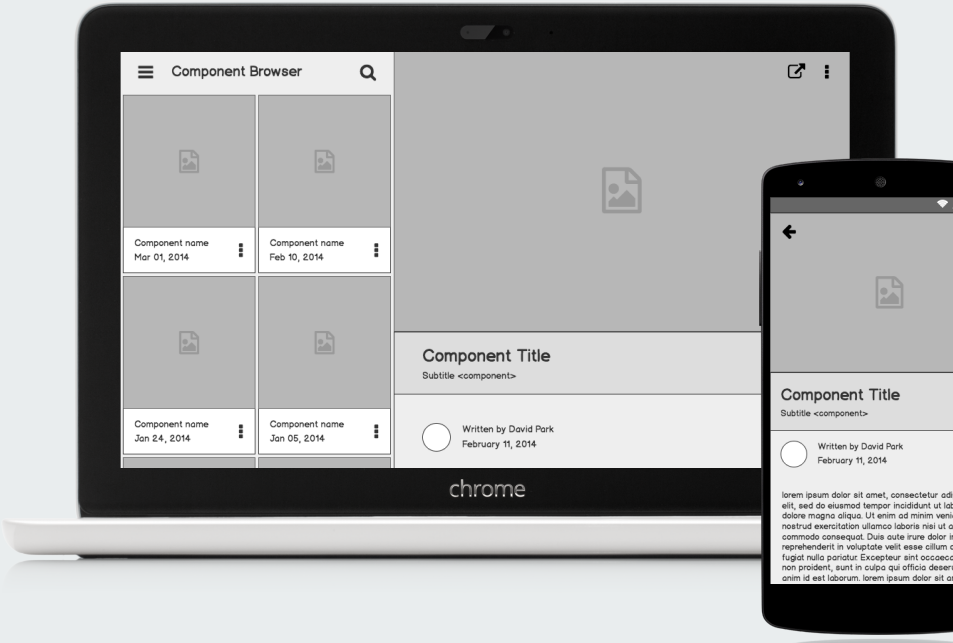
Customized File transfer Protocol



Topics

The Problem statement

Solution

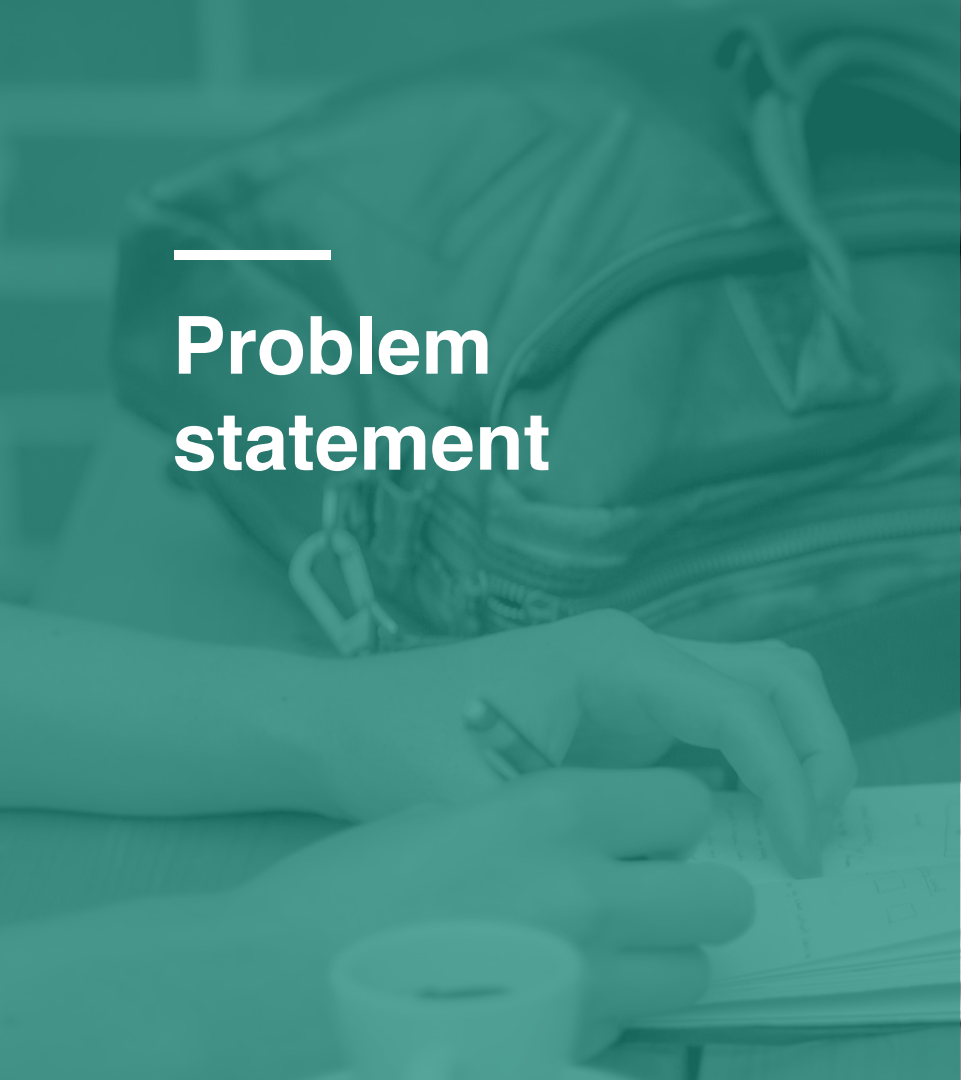
Program Flow

WorkFlows

Server Side Functions

Client Side Functions

Testing - Unit Testing,Integration Testing, Valgrind, Gcov



Problem statement

The main agenda is to create a customized File Transfer protocol that supports user authentication and can handle multiple client connections concurrently and independent of each other .

Solution

Server and client

Multiple client handling

Multiple Directory

Modular

Workflow

Program Flow

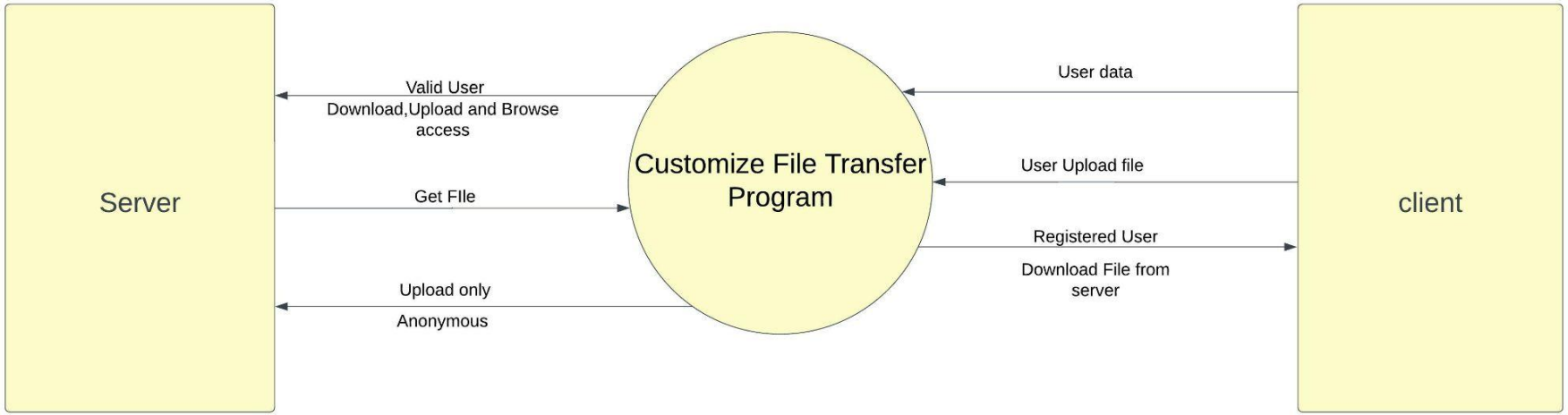
DFD Level 0

DFD Level 1

Program Flow

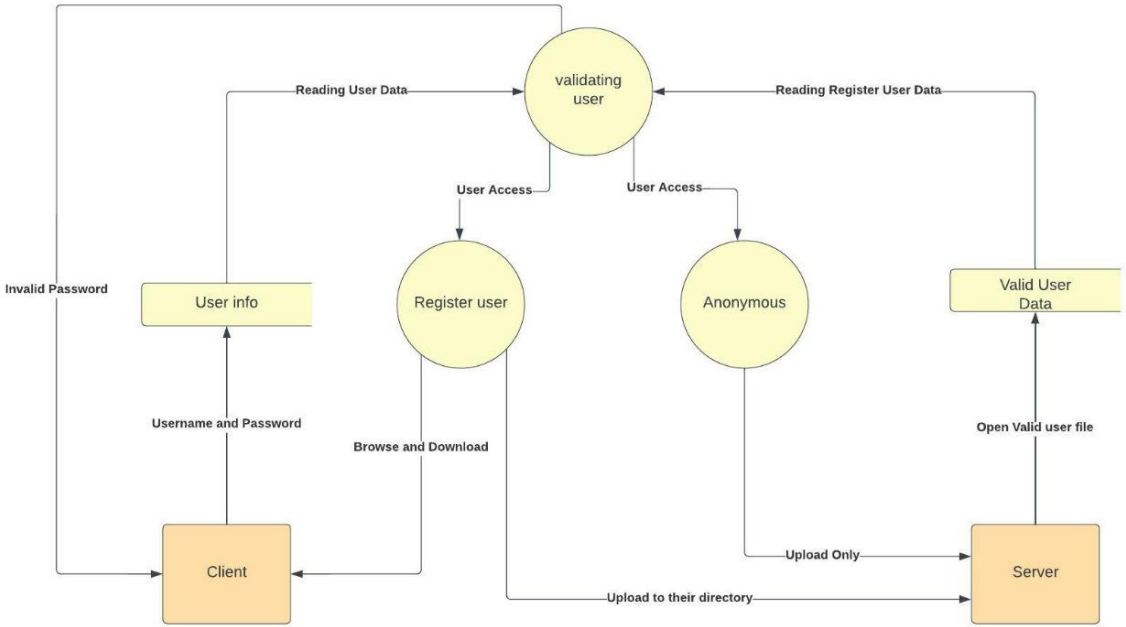
Data Flow Diagram

DFD Level 0



Data Flow Diagram

DFD Level 1



Server side

Server

Main Functions

server.c

1. handle_client()
2. authenticate()
3. check_BlackList()
4. get_ip()
5. upload()
6. download()
7. view()

main()

server.c

- Create socket descriptor variable called sockfd, and new_sock for server and client respectively
- Socket creation : AF_NET, SOCK_STREAM
- Error handle the socket creation
- Specify address and Assign the PORT to the socket
- BIND the socket to the specified address and PORT and do the require error handling
- Put the server in passive mode using Listen function where clients approach to the server
- Accept clients using accept function and check the ip address of the client
- If it is a valid ip address send a proper message to the client and check if we have reached maximum clients.
- Create a new child process and call handle_client function to handle the new clients requests.

get_ip()

- Extracts the ip address from sockaddr_in structure and converts it into a string and store it in variable named ip using sprintf
- Returns the ip in string format

check_BlackList()

- Opens a file called invalid_list where we store all the black_list ip address.
- Compares the clients ip address with every black listed ip present in the file and returns a int value.
- If the client ip is present in our black-listed ip addresses data base then we return 1
- Else we return 0

View()

server.c

- Popen the file by which function was called in read mode
- In an infinite while loop scanf each line in the file and send it to the client until EOF is reached
- This function is used to view the content of the files

handle_client()

- Check whether the client is authenticated.
- Authenticated user will be placed in respective home directive.
- Anonymous user placed in a public directory (/var/ftp/pub).
- Access allowed for authenticated user
  - Browsing - “ls”
  - Download - “get Remote Filename”
  - Upload - “put Local Filename”
  - Quit - “bye”
- Access allowed for anonymous user
  - Only Access to Upload - “put”
  - Can View Files - “cat Filename”
  - Quit = “bye”

authenticate()

- This function is used to check whether the user is authenticated or not.
- Authenticated users data is stored in **“users.txt”** file.
- If the user name is found in users.txt file then consider him as authenticated user else anonymous user.
- Returns type : string
  - Authenticated
  - Not Authenticated

upload()

- ➔ This function is invoked / called when the client wants to download a file from the server's working directory.
- ➔ Receives a string of form <file_name > put.
- ➔ Opens the file in read mode and transfers each line to client.

download()

- ➔ This function is invoked / called when the client wants to upload a file to the server's working directory.
- ➔ Receives a string of form <file_name > get.
- ➔ Opens the file in write mode and receives each line from client.

Client side

Main.c

CLIENT

Client side

Main.c

- In client side we'll create socket. If socket created successfully then server will send the acknowledgement.
- After successful connection auth function gets invoked.
- Depending upon the accessibility client gets upload, download and browse options.

Main Functions

1. upload()

1. Download()

1. pipe_data()

1. auth()

Upload()

Description:

- ➜ Takes the location of the file.
- ➜ Using file pointer it will read every line from file is any error occurs while reading the line it will send message.
- ➜ Else it will send the file to server.

Download ()

- Receive filename from client
- Create file with filename in client side
- Open that file in write mode
- Receive data from server
- Copy that data into file and close the file
- If receiving data length is 0 then stop receiving from server and close the file.

Pipe_data ()

- This function invokes by main()
- When user enter choice for ls,pwd and cat command
- It receive data from server save in buffer array
- Print that buffer array on client terminal
- If receiving data length is 0 then stop receiving from server and break that loop

auth()

- This is used for the authentication of user
- Get username and send to server
- Receive data from server
- If data not matches to “ Anonymous” then get password from client and send to server
- If it matches change its flag to 1 to limit the access.

MakeFile

Client MakeFile

```
run: app
	../bin/client.exe

app: client.o
	gcc -o ../bin/client.exe ../obj/client.o

client.o: ../src/client.c
	gcc -o ../obj/client.o ../src/client.c -c

clean:
	rm ../obj/*.o ../bin/*.exe
```

Server MakeFile

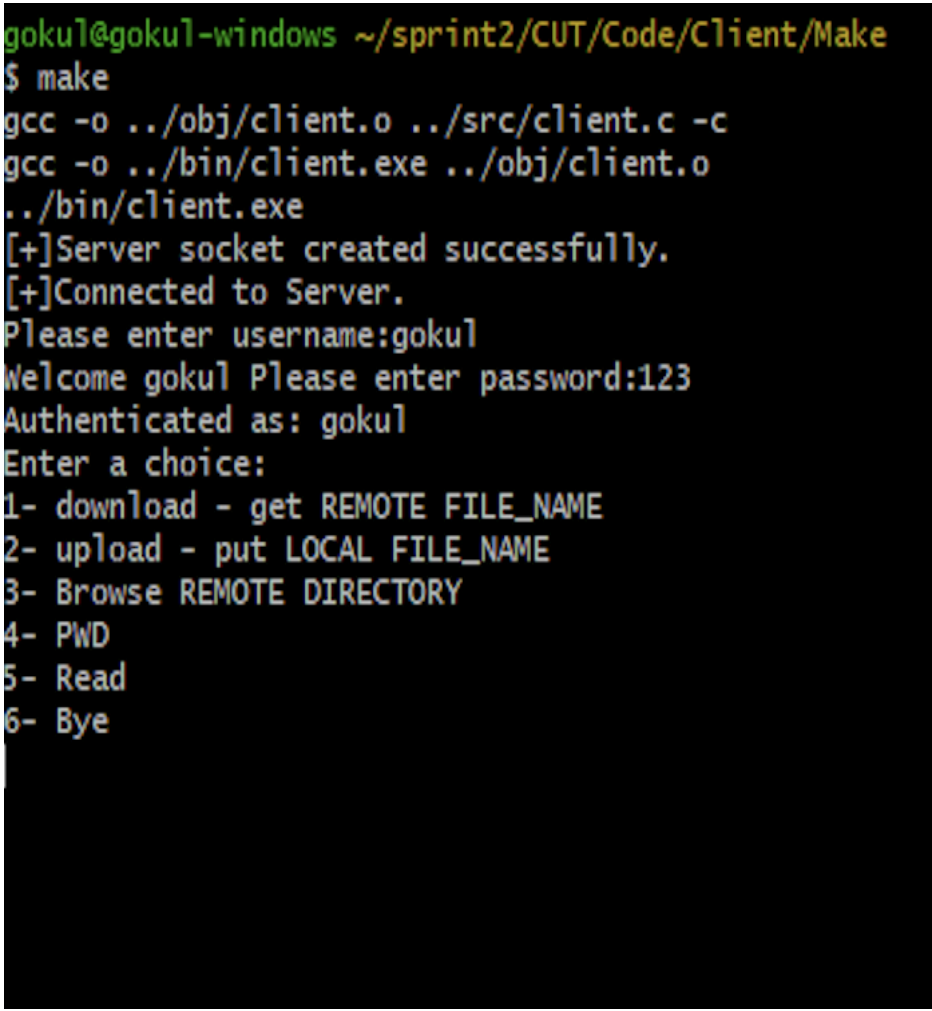
```
run: app
	../bin/server.exe

app: server.o
	gcc -o ../bin/server.exe ../obj/server.o

server.o: ../src/server.c
	gcc -o ../obj/server.o ../src/server.c -c

clean:
	rm ../obj/*.o ../bin/*.exe

test: ../../../Tools_Report/CUnit/test_program.c  ../../../Tools_Report/CUnit/func.c
	gcc -o ../../../Tools_Report/CUnit/test.exe ../../../Tools_Report/CUnit/
test_program.c ../../../Tools_Report/CUnit/func.c -lcunit
	../../../Tools_Report/CUnit/test.exe > ../../../Tools_Report/CUnit/Cunit_Report.txt
```

Integration testing and output

Test cases that we've Consider in this project

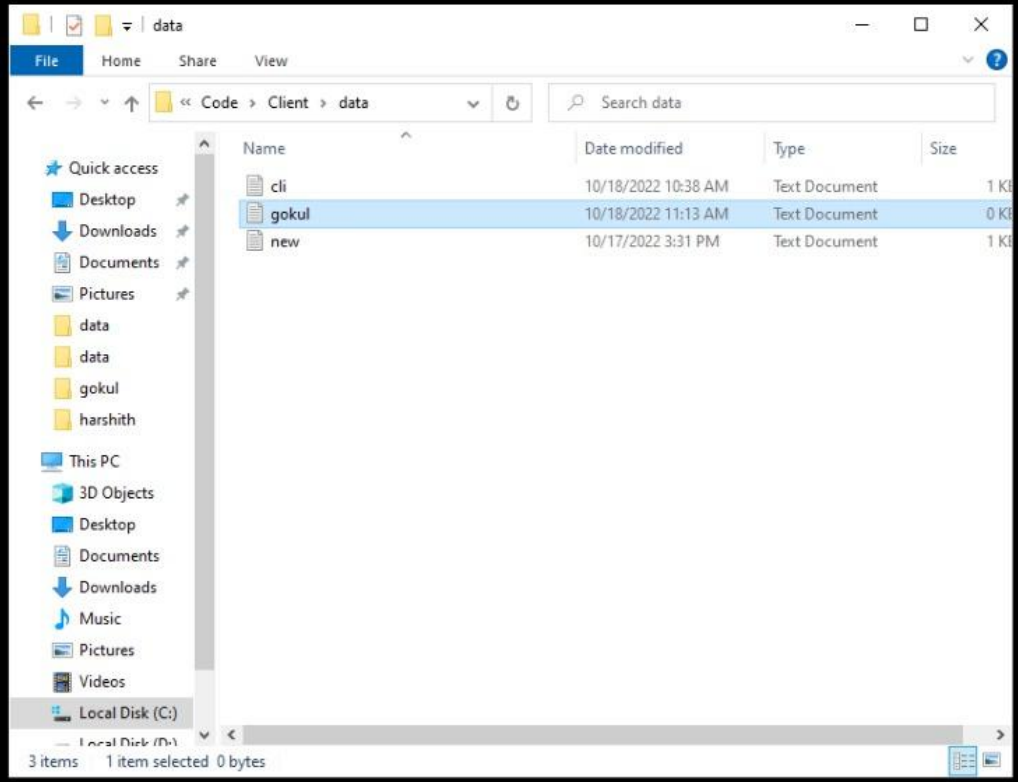
- If client provided correct user ID or not
- If client is authorised or anonymous
- If client is authorised then he can upload, download and browse.
- If client is anonymous then he can only upload.

Terminal output



~/sprint2/CUT/Code/Client/Make

```
gokul@gokul-windows ~/sprint2/CUT/Code/Client/Make
$ make
gcc -o ../obj/client.o ../src/client.c -c
gcc -o ../bin/client.exe ../obj/client.o
../bin/client.exe
[+]Server socket created successfully.
[+]Connected to Server.
Please enter username:gokul
Welcome gokul Please enter password:123
Authenticated as: gokul
Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
1
enter the file to download:gokul.TXT
Downloaded Successfully

Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
```



~/sprint2/CUT/Code/Client/Make

```
gokul@gokul-windows ~/sprint2/CUT/Code/Client/Make
$ make
gcc -o ../obj/client.o ../src/client.c -c
gcc -o ../bin/client.exe ../obj/client.o
../bin/client.exe
[+]Server socket created successfully.
[+]Connected to Server.
Please enter username:yuv
Authenticated as: Anonymous
Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
1
No Permission to execute

Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
3
cli.txt
read.TXT

Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
5
enter the file to read:read.TXT
reading file from public
Enter a choice:
1- download - get REMOTE FILE_NAME
2- upload - put LOCAL FILE_NAME
3- Browse REMOTE DIRECTORY
4- PWD
5- Read
6- Bye
```

Tests Covered

Sunny Test Cases:

`"127.0.0.20"`

`127.34.213.1`

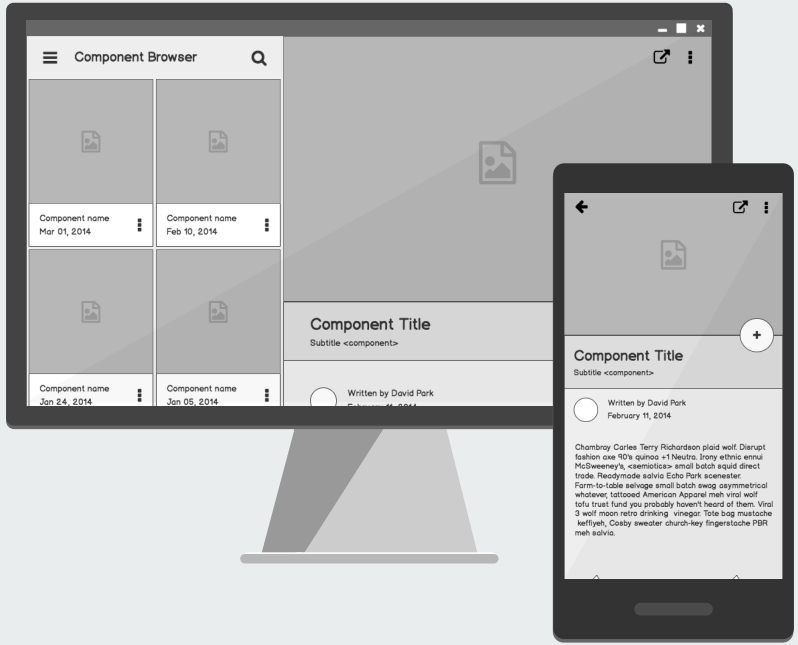
`127.0.0.100`

Rainy Test Cases:

`12.204.255.255`

`127.0.0.10`

`127.0.0.1606`



Unit Testing for Check_BlackList Function:

```
     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/


Suite: Testing_Suite1
  Test: Testing Sunny Cases ...


passed
  Test: Testing Rainy Cases ...


passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      2      2      2      0        0
             asserts      6      6      6      0      n/a

Elapsed time =    0.000 seconds
```

check_BlackList():

1. Open file at location: ../etc/ftp/client_blackList/filename using fopen in “r” mode
2. Scanf every ip in the file and if it is equal to the ip address of any client return 1
3. Else in any other case return 0

Valgrind Report

Client Side Report

```
==16892== Memcheck, a memory error detector
==16892== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==16892== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==16892== Command: ./a.out
==16892==
[+]Server socket created successfully.
[-]Error in socket: Connection refused
==16892==
==16892== HEAP SUMMARY:
==16892==     in use at exit: 0 bytes in 0 blocks
==16892==   total heap usage: 5 allocs, 5 frees, 3,064 bytes allocated
==16892==
==16892== All heap blocks were freed -- no leaks are possible
==16892==
==16892== For lists of detected and suppressed errors, rerun with: -s
==16892== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Valgrind Report

Server Side Report

```
==16835== Memcheck, a memory error detector
==16835== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==16835== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==16835== Command: ./a.out
==16835==
[+]Server socket created successfully.
[+]Binding successfully.
[+]Listening....
^C==16835==
==16835== Process terminating with default action of signal 2 (SIGINT)
==16835==    at 0x4949AB3: accept (accept.c:26)
==16835==    by 0x10B498: main (in /home/cg83-user20/CGSprint2/capg2/capg/CUT/Code/Server/src/a.out)
==16835==
==16835== HEAP SUMMARY:
==16835==     in use at exit: 0 bytes in 0 blocks
==16835==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==16835==
==16835== All heap blocks were freed -- no leaks are possible
==16835==
==16835== For lists of detected and suppressed errors, rerun with: -s
==16835== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

GCOV()

Gcov is a source code coverage and statement-by-statement profiling tool.

Server Report : -

- File 'server.c'
- Lines executed:77.78% of 153
- Creating 'server.c.gcov'

Client Report : -

- File 'client.c'
- Lines executed:84.72% of 144
- Creating 'client.c.gcov'

Conclusion:

This is a software which allows multiple clients to access there directories on server and make necessary changes.

Challenges faced:

- ➔ Concurrent connection of multiple clients
- ➔ Maintaining data flow between server and multiple clients
- ➔ Error handling

Thank You

https://github.com/HarshithReddy15/CGSprint2

Team CG83-Group3

- Gundra Harshith Reddy
- Gokul Krishna
- Egumamidi Sandeep Reddy
- Yuvraj C gadad
- Sushant Kumar Singh