FTP Server Functional Requirements

Group-3 (Sprint 2)

Customize File Transfer Protocol

Index

Chapter 1: Introduction.

The introduction of the software requirement specification provides an overview of the entire Software. The entire SRS with overview description purpose, scope, tools used and basic description. The aim of this document is to gather, analyse and give an in-depth insight into the FTP Server Functional Requirements by defining the problem statement in detail. The detailed requirements of the application are provided in this document.

1.1 Intended Audience

This document is intended to be read by the User.

1.2 Project Purpose:

The purpose of this project is to build and server client application which handles multiple clients. Where authorised clients can browse, update and download the files from server and anonymous clients can only upload the file to server.

1.3 Key project Objectives:

a. Server initialization message is displayed after starting server
b. Client connects message is displayed after clients connects to server
c. Authenticated and Anonymous user logins
d. Multiple clients connected to a single server
e. Client will have multiple options to browse, upload and download files from server
f. Clients show the successful and unsuccessful message after performing a particular operation

1.4 Project Scope:

The main aim of the project is to login a authenticated and anonymous users and do the required operations such as browse, list, upload, download and view present directory of the user, where as the anonymous user will not be having download privilege and whereas all the operations performed by anonymous user will only be effective in public directory.

1.5 Functional Overview: -

1.1.1 Following header files are included in the program:

- #include <sys/socket.h>
- #include <arpa/inet.h>
- #include <stdio.h>
- #include <stdlib.h>
- #include <unistd.h>
- #include <string.h>

1.6 Client site

1.6.1 Login

- Username
- Password

1.6.2 Run

- Download:
  Download files from server.

- Upload:
  Upload files on server in respective directory.

- Browse:
  Browse in present working directory.

- PWD:
  Check present working directory.

- Read:
  Read file in the present working directory.

- Exit:
  After the operation done exit from the server.

1.7 **Server site**

Client send the request to the server for translating the language.

2. Design Overview: -

- **Customize File transfer Protocols comprises of the following function in maintain database:**

Name of the Module	Browse choice
Handled by	Gokul
Description	The function is used to list the contents of the directory.

Name of the Module	Read choice
Handled by	Harshith
Description	This function is used to print the contents inside the file.

Name of the Module	PWD command
Handled by	Yuvraj
Description	The function is used to know present working directory

Name of the Module	Upload
Handled by	Sushant
Description	This function is used to upload file on the server

Name of the Module	Download
Handled by	Sandeep
Description	This function is used to download file from server

Name of the Module	Bye command
Handled by	Yuvraj
Description	This command is used to exit the client from the server.

2.1 Design Objectives:

1. Start the connection
2. Accept the connection
3. Check for black list IP
4. Different choices should be able to get excepted output.
5. Customize file transfer protocol able to send file.

2.2 Design Alternative: -

We have used fork instead of thread for independent multiple client handling.

2.3 User Interface paradigms: -

The Customize file transfer protocol should be able to allow the client to upload, download, read the contents file and browse directory. Multiple clients should be able to connect one server.

2.4 Validation: -

- The server first check for IP of incoming client check in black list IP file accordingly connects to client
- The server and Client should be able to establish a connection successfully.

- There are two types of users Anonymous and Authenticated User. For anonymous users 'username is any value other than valid username. But for authenticated users username and password credentials should match with the valid_user.txt file.

- Authenticate function load the user details from the valid_user.txt file give the user access accordingly.

2.5 Exceptional Handling: -

- If the admin gave invalid username or password, error message will be displayed
- If the user search for the file that is not present, error message will be displayed
- If the file will not open error message will be displayed.

2.6 Performance: -

The system will work on the user's terminal. The performance shall depend upon server.

3. SYSTEM ARCHITECTURE: -

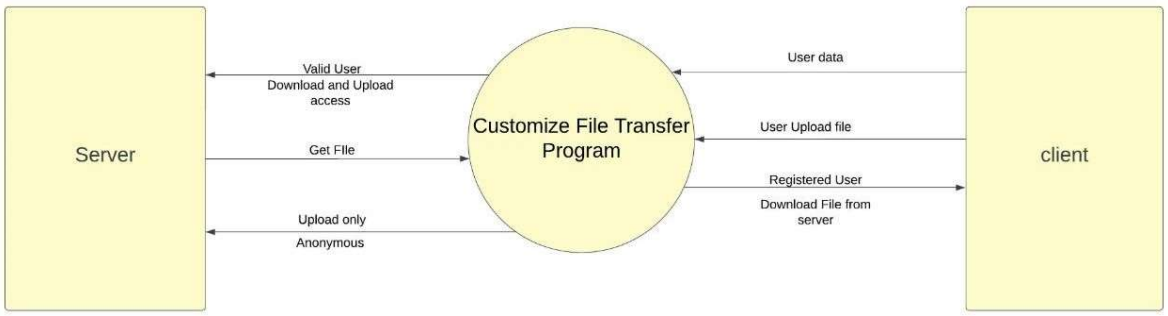
3.1. Database Architecture

The database used inside our program is the user.txt file. The user.txt file contains username and password separated by space. We have used strtok with a delimiter as space to store username and password. Also, for the present directory we can use username for the purpose.

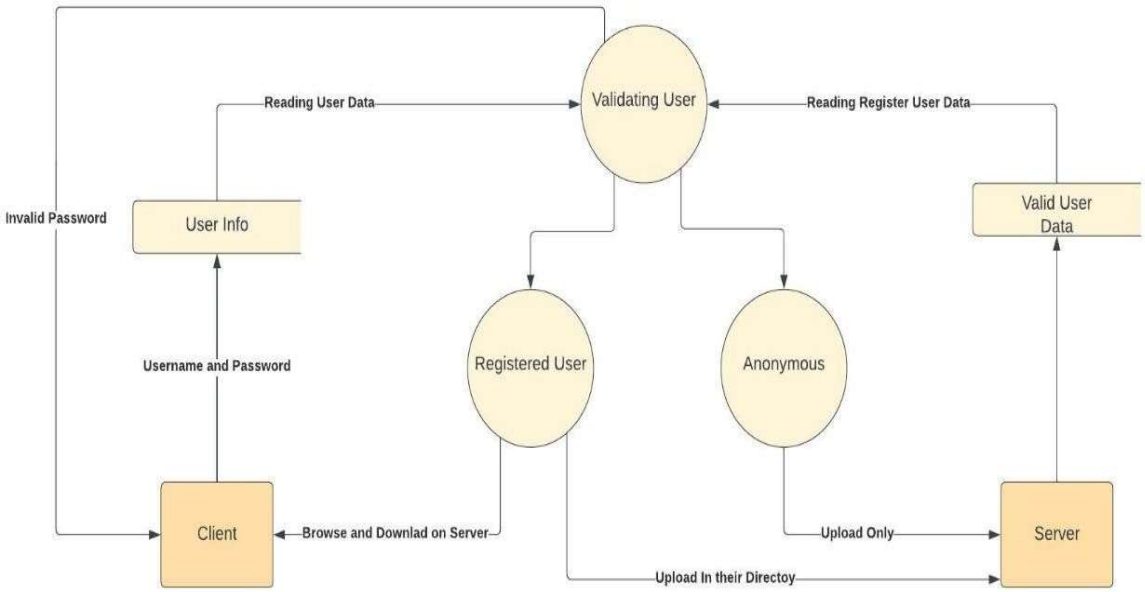
The architecture used in this system is like that we store all the users inside a data folder in the home directory. And all the information of the authenticated and anonymous user stored in user.txt files.

4. Detailed System Design:

4.1 DFD-0:



4.2: DFD-1:



5. Environment Description: -

5.1 Time Zone Support: -IST-Kolkata

5.2 Language Support: -English

5.3 User Desktop Requirements: -

- 64-bit processor, 1 GHz or faster
- At least 10 GB free hard drive space
- At least 1 GB RAM

5.4 Server Side Requirements: -

- 64-bit processor, 1 GHz or faster
- At least 2GB free hard drive space
- At least 1GB RAM

5.4.1 Deployment Considerations: -

- Local storage is used
- No network latency to consider
- To scale buy a bigger CPU, more memory, larger hard drive, or additional hardware 9

5.4.2. **Application Server Disk Space: -**
No such disk space is required as the program is fully functional on online IDE(s) as well. Local Operating System is required and two txt file to store the records of processes.

5.4.3. **Database Server Disk Space: -**
No such disk space is required as the program is fully functional on online IDE(s)as well. Local Operating System is required and two txt file to store the records of processes.

5.4.4. **Integration Requirements: -**

- Language: -C
- Tools: -Valgrind, Makefile ,splint, gcoverage
- Complier: -gcc
- Linux Environment

5.4.5. Network: -End to End

5.5 Configuration: -

5.5.1. Operating System: -Linux environment