

**FTP-Server Functional Requirements**

### Software Requirement Specification (SRS) Document

**Group-3**

### Batch: E2E-Manipal-Systems C Linux Programming PT Aug 11th Batch 2

**SPRINT 1**

**Project Timeline**

### October 13 to October 19

Contents

### INTRODUCTION

* 1. Purpose
  2. Intended audience
  3. Intended use
  4. Scope

### OVERALL DESCRIPTION

* 1. Assumptions and dependency.

### SYSTEM FEATURE AND REQUIREMENTS

* 1. **Functionality**

3.1.1CFTP\_01 Client Server Connection:

3.1.2 CFTP\_02 User Authentication:

3.1.3 CFTP\_03 Server Handshaking:

3.1.4 CFTP\_04 Server side Firewall Protection:

3.1.5 CFTP\_05 Server side Security:

3.1.6 CFTP\_06 Service Starting at Boot:

3.1.7 CFTP\_07 Anonymous user access:

3.1.8 CFTP\_08 Anonymous user upload access:

3.1.9 CFTP\_09 Authenticated User Access:

3.1.10 CFTP\_10 Server side Concurrency:

3.1.11 CFTP\_11 Client side program starting:

3.1.12 CFTP\_12 Client Side Connection request:

3.1.13 CFTP\_13 Client Side Authentication:

3.1.14 CFTP\_14 Client Side User Environment:

3.1.15 CFTP\_15 Client Side Environment Customization:

3.1.16 CFTP\_16 Client Side Browsing:

3.1.17 CFTP\_17 Client Side Download Transfer:

3.1.18 CFTP\_18 Client Side Upload Transfer:

3.1.19 CFTP\_19 Client Side Quit:

**3.2 Tools to be used.**

**3.3 System features.**

**4 Data flow diagrams.**

4.1 DFT level 0.

4.2 DFT level 1.

**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 complete **FTP-Server Functional Requirements** application by defining the problem statement in detail. The detailed requirements of the **FTP-server functional requirements** application are provided in this document.

**1.1 Purpose**: The purpose of this document is to show the requirements for the **FTP-server functional requirements** application, In which multiple clients can access the single server and authorized people can browse, upload and download the file and unauthorized people can only upload the file.

**1.2 Intended Audience:** This document is intended to be read by Client.

**1.3 Intended Use:**

* + - Development Team
    - Maintenance Team
    - Clients

**1.4** **Scope:** This project aims to create and development of a FTP-Server functional requirements application where multiple clients can access one server and can browse, upload and download the files depending upon there accessibility.

**Chapter 2: Overall Description**

In FTP-server functional requirement application multiple clients can access the single server simultaneously and can upload, download and browse the file depending upon their accessibility. First server will configure the clients IP address and then ask for user name and password if user name is correct and password dose not they he’s unauthorised client and he can only upload. If credentials are correct then client and browse, upload and download the files.

### 2.1 Assumptions and Dependency:

* + - System should have Ubuntu Linux installed.
    - System should have either 4GB or more RAM.

**Chapter 3: System Requirement and Features.**

### 3.1 Functionality.

### 3.1.1CFTP\_01 Client Server Connection: A connection between client and server must be established first.

### 3.1.2CFTP\_02 User Authentication: The server should authenticate the user from a poll of registered user data kept with appropriate data structure.

### 3.1.3 CFTP\_03 Server Handshaking: The user should get the customize message from server entering the user id but before providing the password by which the user get conformation that it got connected to the proper server.

### 3.1.4CFTP\_04 Server side Firewall Protection: The server should listen to port 8021 for client connection and make the client side computer access the server port.

### 3.1.5CFTP\_05 Server side Security: Before allowing a client connection the server should match the client IP with the blacklist IP specified in /etc/ftp/client\_blacklist .

### 3.1.6CFTP\_06 Service Starting at Boot: The server must configure to start theservice automatically at boot.

### 3.1.7CFTP\_07 Anonymous user access: Server should support an virtual user with name anonymous to have read only access the server’s /var/ftp/pub directory without any password.

### 3.1.8CFTP\_08 Anonymous user upload access: Server should support the same anonymous user to have upload only (no browsing & no download) access the server’s /var/ftp/upload directory.

### 3.1.9 CFTP\_09 Authenticated User Access: All the authenticated user should be allowed to have full access (Upload, download and Browse) to their home directory.

### 3.1.10CFTP\_10 Server side Concurrency: The server should support the concurrent client connection

### 3.1.11CFTP\_11 Client side program starting: The user should start the client program to connect the Server whenever he wants.

### 3.1.12CFTP\_12 Client Side Connection request: Client should request a connection to server on port 8021.

### 3.1.13CFTP\_13 Client Side Authentication: Client program should start with an authentication (default to anonymous).

### 3.1.14CFTP\_14 Client Side User Environment: On successful authentication the laced on his home directory as PWD & anonymous user PWD should be /var/ftp/pub.

### 3.1.15CFTP\_15 Client Side Environment Customization: User can use cd to change his remote PWD on client side.

### 3.1.16CFTP\_16 Client Side Browsing: User can use ls to browse the file list in the remote PWD.

### 3.1.17CFTP\_17 Client Side Download Transfer: User can use the get REMOTE FILENAME to download the file from the remote PWD.

### 3.1.18CFTP\_18 Client Side Upload Transfer: User can use the put LOCAL FILENAME to upload the file to remote PWD.

### 3.1.19 CFTP\_19 Client Side Quit: The user should use a sub command bye to terminate the Connection.

**3.2 Tools used:**

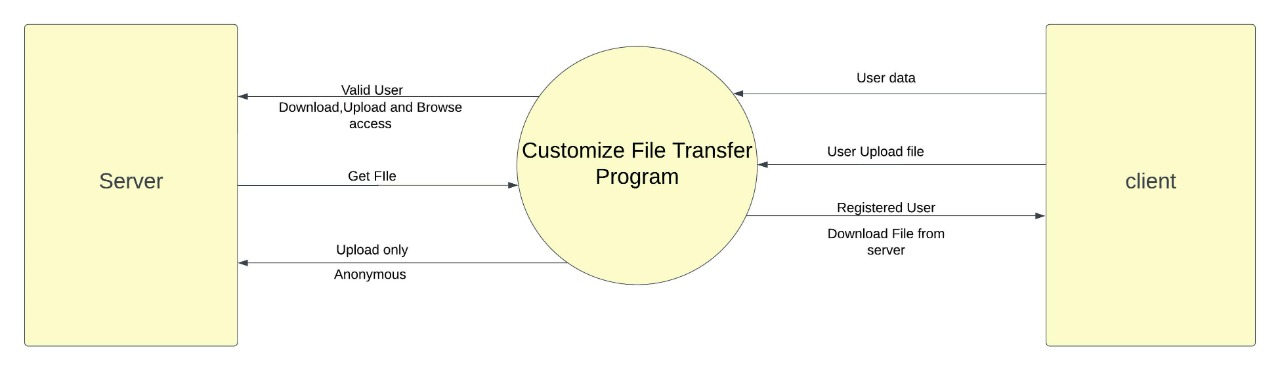
1. C language
2. POSIX Library is used for forking
3. File Handling is used to read user credentials.
4. Dynamic memory allocation is used for programming
5. CUnit, Valgrind, Make, and GDB are also used for making, analyzing and debugging the program
6. System Programming

### 3.3 System Features:

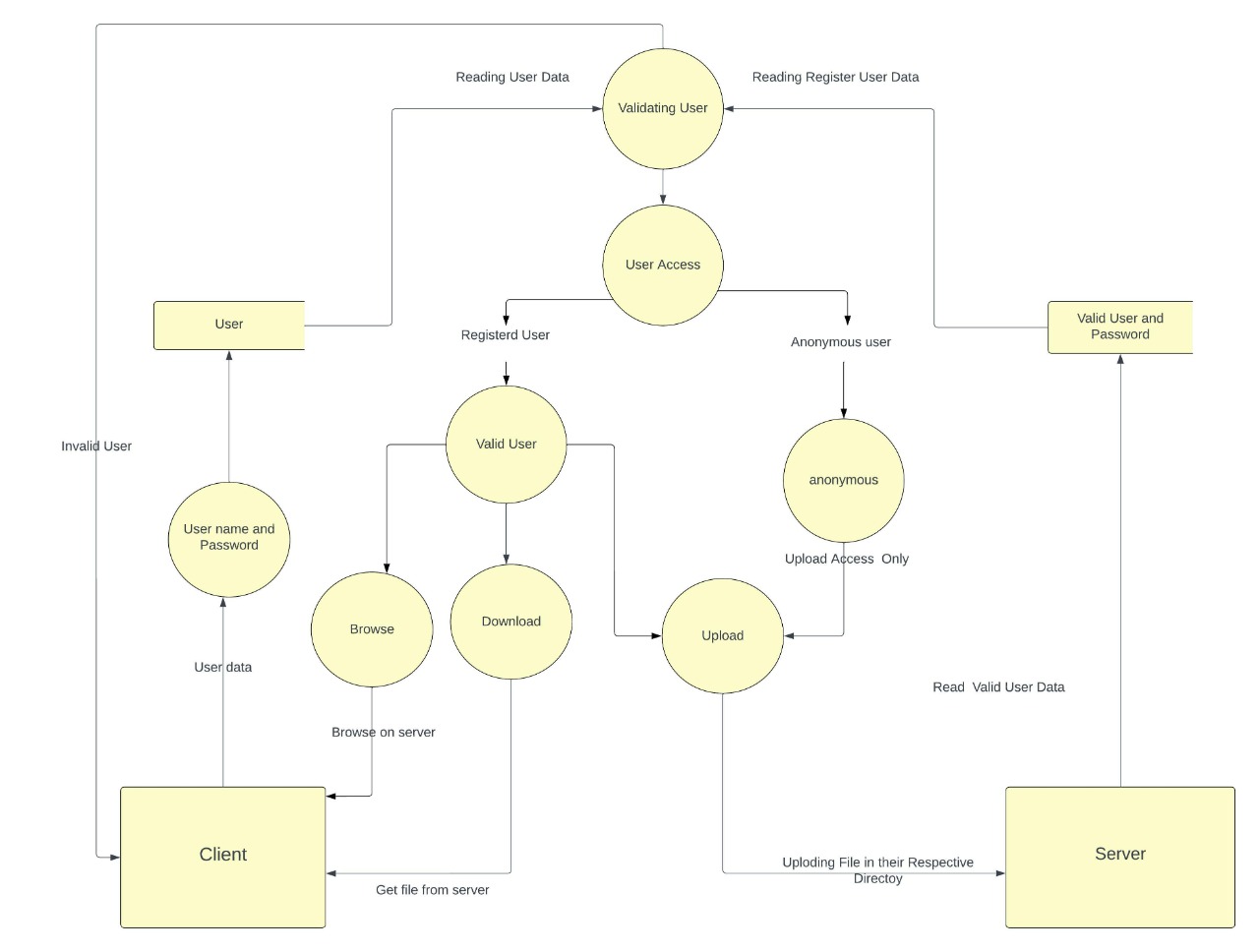
1. **Supportability** - The system is built using C language.
2. **Design Constraints** - The solution is built as a multi-file multi-directory solution. All codes are documented well and the coding standards are followed
3. **Usability** – This program can be used for assigning work to Engineers as per the requirements and the domain of knowledge. The system can read any number of defect files given in the format and assign the work to the employees as per the requirement.
4. **Reliability and Availability** – The system is available 24/7 and the user can use the system whenever he/she needs it. The user can avail of the functionalities of the program by providing the necessary files and also giving permission to read the employee data.
5. **Performance** – The system will work on the user terminal

**Chapter 4 Data flow diagram**

**DFT\_Level 0:**

****

**DFT\_Level 1:**

****