Group 4 - Lakshmi Salini, Martin Le, Miguel Angel Braojos, Aaron Ramos

Github Repository

https://github.com/madmartian8/DFS-Client

6/7/2021 - Client Notes

- DFS is a system allowing sharing of data to other users.
- DFS “Server”
- Every client stores their system
- Client-server model
- Every connection to the server becomes a part of the DFS, and the server may store some files on the client that runs that software
- Includes a GUI
- Option for Client to select which files to push out to the Server
- DFS has no storage
- DFS decides which files to store to which client
- Client can see all the files across all systems
- host server on one machine(edited)
- Peer-to-Peer distribution
- Clients do not know what files they are storing
- Do not have to fragment the files
- Clients assumed to always be online

Group 4 - Lakshmi Salini, Martin Le, Miguel Angel Braojos, Aaron Ramos

6/9/2021 - Client Notes

- Create a Requirements Document
- Requirements
    - Network application – runs over a network (TCP, message example)
    - Server – server application will allow connections from clients over the network
    - Server has no permanent storage space
    - Username and password, then setup a session
        - Client would then look through all the files on the DFS, then download it
        - Server will then find that file from which system that file was on, then transfer it to the requestee
- Server receives saved file, then picks from a node and transfers it to them
- Needs a GUI for the Client

Group 4 - Lakshmi Salini, Martin Le, Miguel Angel Braojos, Aaron Ramos

6/14/2021 - Client Notes

- We can choose for a file to go to the server before going to a client or directly to the client (?). Ideally the latter after a request for the server to send it.
- Server is handling all the choices. Don't want any client to know where the files are.
- Server is the ONLY ledger. Server tracks all the locations of the files, client doesn't know.
- 1 Message has only 1 request. 3 files will be 3 transactions (maybe queue'd?)
-

Group 4 - Lakshmi Salini, Martin Le, Miguel Angel Braojos, Aaron Ramos

6/21/2021 - Client Notes

- What is the bare minimum required to make sense?
- What is the maximum we're going to do?
    - Offer the user a choice between the two
- Display stuff contextually
- Change passwords, fixed usernames
- Consider the use cases that are NOT normal. Where the users do the wrong thing, or users are purposely trying to break the program.
- Add security question prompt for password to requirements document
- Creation of new accounts
    - Admin account that creates accounts, with user, password

    Self-subscription, user creates account if does not exist

Design Document

- Server class
- Node class
- File manager class
- Client UI class
- File storage class
- Login class
- History Log Class

Guide

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it
- Types of Requirements
    - Requirements set out what the system should do and define constraints on its operation and implementation
    - Functional requirements set out services the system should provide. These are mandatory for system functionality
    - Non-functional requirements constraints the system being developed and specifies non-functional characteristics
    - User requirements are high-level statements of what the system should do

Date	Version	Description	Author
06/09/2021	1.0	Initial Creation	Aaron, Lakshmi, Martin, Miguel
06/23/2021	1.1	User actions such as security ?'s, changing passwords, etc.	Aaron, Lakshmi, Martin, Miguel

DFS Requirements

- Functional requirements:
    - Active Directory
    - Runs over a network (TCP)
    - File retrieving
    - File storage for clients
        - Path to stored file
        - Disk management
    - Use storage to store files on nodes (clients)
    - All client are assumed to be online at all times
    - Error Handling
        - Finding missing files

- Non-functional requirements:
    - Server has no permanent storage
    - Garbage disposal
    - A login system-Username and password for clients
        - Ability to change passwords
        - Fixed usernames
    - Security questions
    - New account creation
    - A GUI for clients
        - A file browser
        - Information about modification/deletion below file

      FIFO Method

- User Requirements
    - Internet connection
    - Knowledge of valid login credentials
    - JVM installed
    - Sufficient Storage Space to download DFS Client

Use Cases

Use Case ID: UPLOAD
Use Case Name: Upload File
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told system which file
Post-conditions: DFS has uploaded file to a node
Basic Flow or Main Scenario:

1. User presses “Upload file” button
2. Client sends networked message to DFS
3. DFS sends file to the node it wants to
4. Display “File uploaded” message to client

Extensions or Alternate Flows:

Exceptions:

- File is already in system
- File is corrupted (?)
- Upload is interrupted by network problems

Related Use Cases:

Use Case ID: DOWNLOAD
Use Case Name: Download File
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told system which file
Post-conditions: Client has file on their system
Basic Flow or Main Scenario:

1. User selects file from DFS system
2. User tells system which file they want
3. Clients system sends networked message to DFS requesting the specific file
4. DFS looks up which node has the requested file
5. Node’s system sends file to requesting node

Extensions or Alternate Flows:

Exceptions:

- File no longer exists
- File is corrupted (?)
- Download is interrupted by network problems

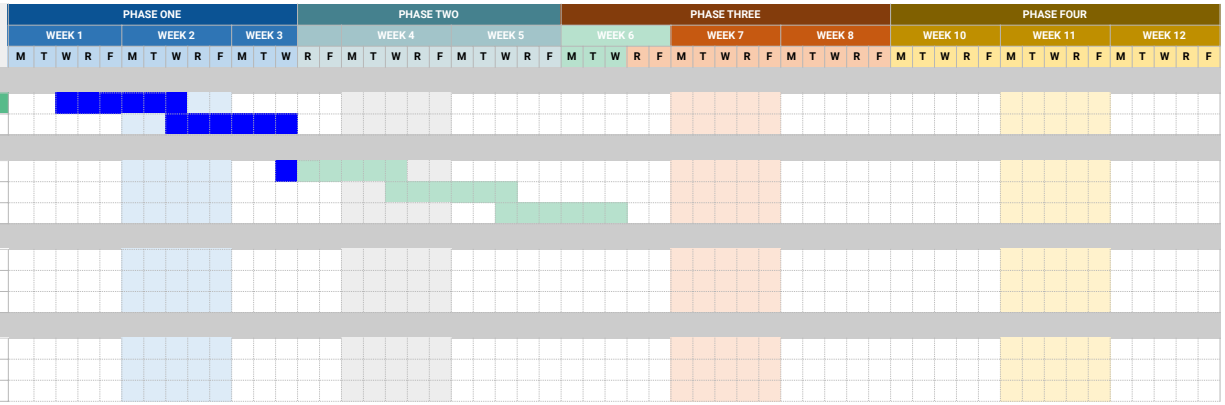
Related Use Cases:

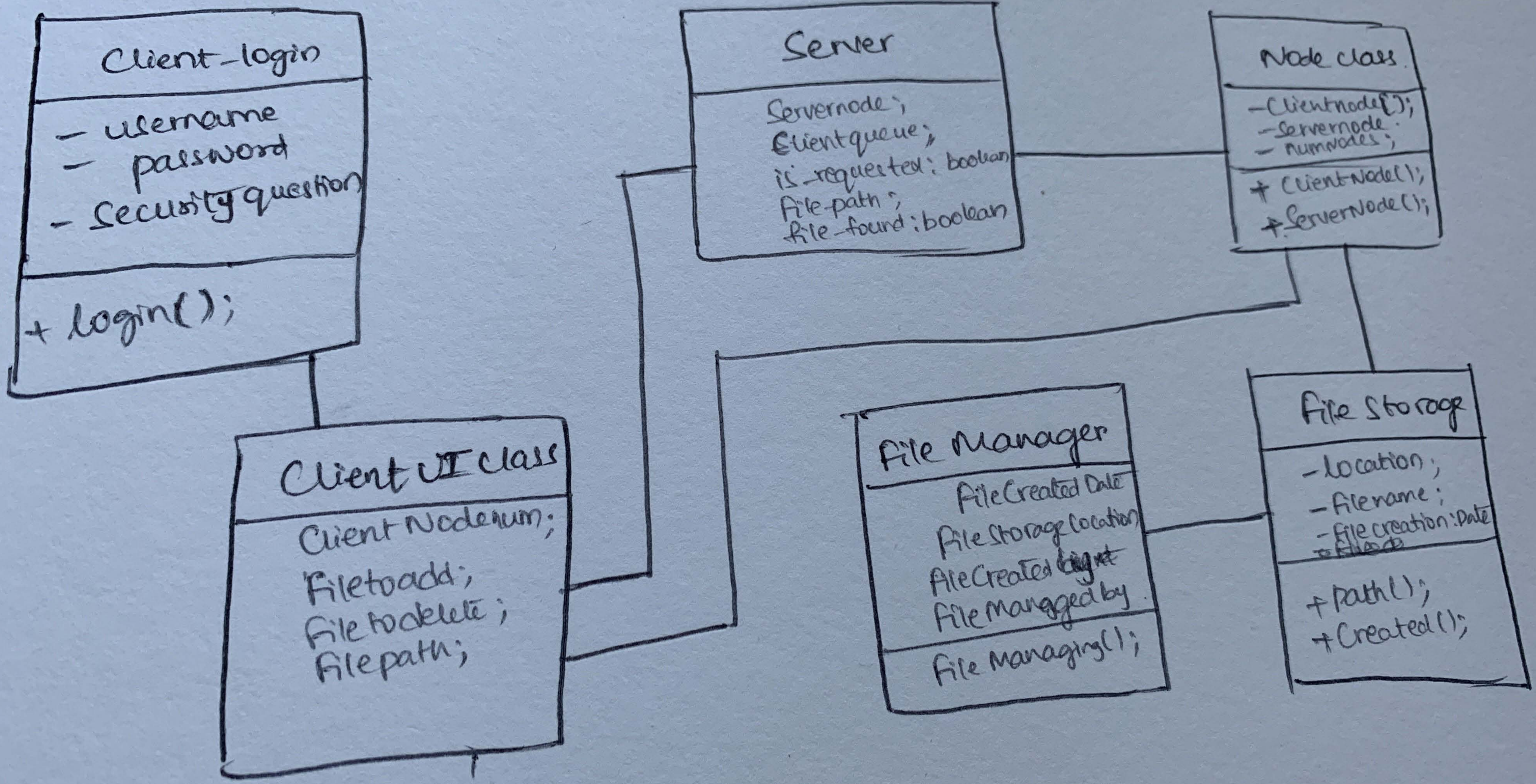
GANTT CHART TEMPLATE

Smartsheet Tip ➜ A Gantt chart's visual timeline allows you to see details about each task as well as project dependencies.

PROJECT TITLE	DFS Project	**COMPANY NAME**	Group 4
PROJECT MANAGER	Laksmi, Aaron, Miguel, Martin	**DATE**	6/9/21

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Project Conception and Initiation**				
1.1	Requirements Document	6/9/21	6/16/21	7	90%
1.1.1	Design Document	6/17/21	6/23/21	6	0%
2	**Project Coding**				
2.1	Implementation	6/24/21	6/30/21	6	0%
2.2	Testing	7/1/21	7/7/21	6	0%
2.3	Maintenance	7/8/21	7/14/21	6	0%
3	**Project Conception and Initiation**				
3.1				0	0%
3.2				0	0%
3.2.1				0	0%
4	**Project Performance / Monitoring**				
4.1				0	0%
4.2				0	0%
4.3				0	0%




Client_login
- username
- password
- Security question
+ login();
Server
Servernode;
Clientqueue;
is_requested: boolean
file_path;
file_found: boolean
Node class
-Clientnode[];
-Servernode;
-numnodes;
+ ClientNode();
+ ServerNode();
Client UI class
ClientNodenum;
Filetoadd;
Filetodelete;
Filepath;
File Manager
FileCreated Date
FileStorage(location)
FileCreated by
FileMangged by
File Managing();
File Storage
-location;
-filename;
-file creation: Date
+path();
+Created();