DFS - Distributed File System written in Java

GROUP 4

By: Aaron Ramos, Lakshmi Salini, Martin Le, Miguel Angel Braojos

For: SUMMER 2021, CS401-01

Prof. Christopher Smith

Table of Contents

06/29/2021

Design

Section 1 - Project Description

1.1 Project
Distributed File System

1.2 Description
The Distributed File System will allow connections between various computers to form a private cloud-storage where the connected systems can upload and download files to and from the cloud-storage that they have formed.

1.3 Revision History

Date	Comment	Author
06-29-2021		

Contents

Section 2 - Overview

2.1 Purpose

The purpose of this system is to allow systems within the cloud to safely store files as they are backed up onto all connected systems. The system also provides a means for sharing files easily over the formed cloud-storage. The main brain of the system (the server) will decide where the files go on each system so the user doesn't have to.

2.2 Scope

The system should aim to provide a way for a user in the system to download any file contained within the cloud-storage, and upload to it as well. The system should have a login module so only users with the right credentials have access.

2.3 Requirements

See Requirements Document

2.3.1 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement

SRS Requirement	SDD Module
Req 1	5.1.1 (link to module), 5.1.2 (link)

Section 3 - System Architecture

The user will first have to login to the DFS system using their computer, where their login credentials will be checked with the server. From there, the client will prompt the user with either the option to upload or download files from/to the system. These actions require the server to check the files on each node connected to the cloud-storage system, and then the nodes will interact with the requesting node. For example, if the client requested to download a file, the server would check each node for the file and then tell the node that has the file to send it directly to the requesting node.

Section 4 - Software Domain Design

4.1 Software Application Domain Chart

Describe / chart each major software application domain and the relationships between objects (UML, etc)

4.2 Software Application Domain

A Comprehensive high level description of each domain (package/object wherever it is better to start) within the scope of this module (or within the greater scope of the project if applicable)

4.2.1 Domain X

A high level description of the family of components within this domain and their relationship. Include database domain, stored procedures, triggers, packages, objects, functions, etc.

4.2.1.1 Component Y of Domain X

Define Component Y, describe data flow/control at component level

5.2.1.1.1 Task Z of Component Y1 of Domain X

Define Task Z, describe data flow/control at task level

Section 5 – Data Design

After selecting the “Upload file” option, prompt a standard folder/file upload window.

The Data contained within the system will consist of the files stored in the cloud-storage as well as all of the login credentials. Both will be hidden from the users.

Section 7 - User Interface Design

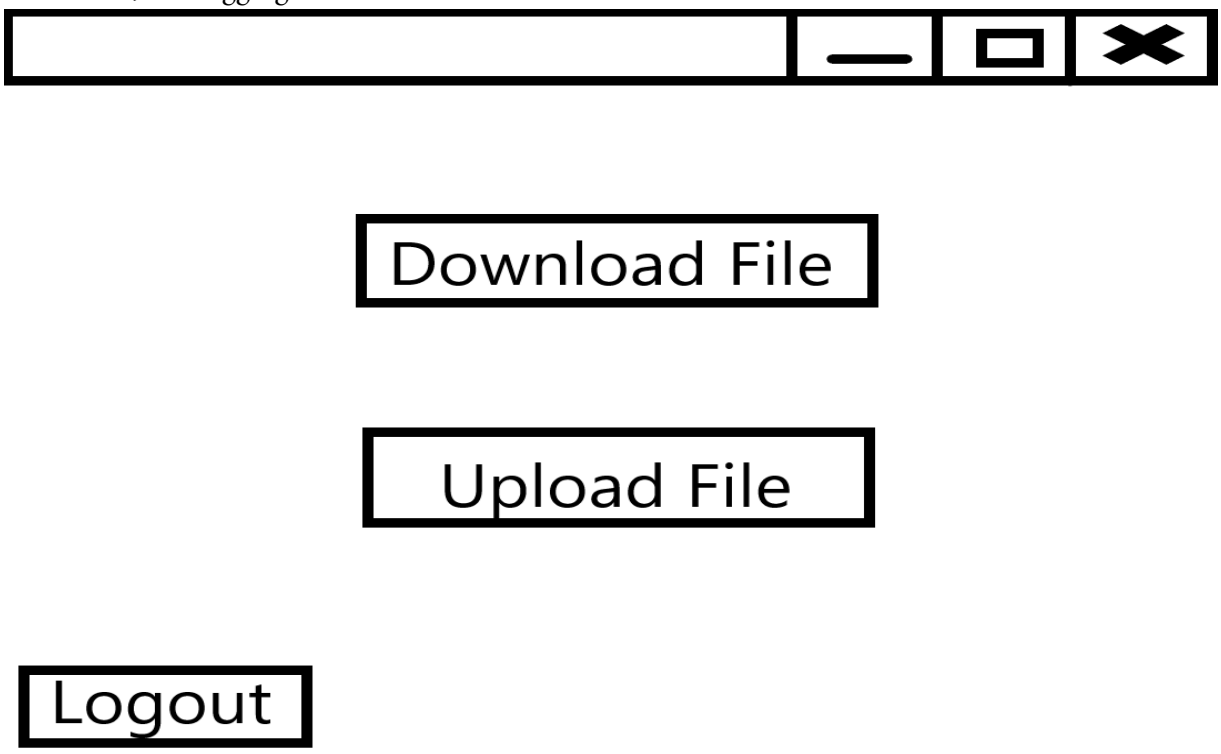
7.1 User Interface Design Overview

Pictures, high level requirements, mockups, etc.

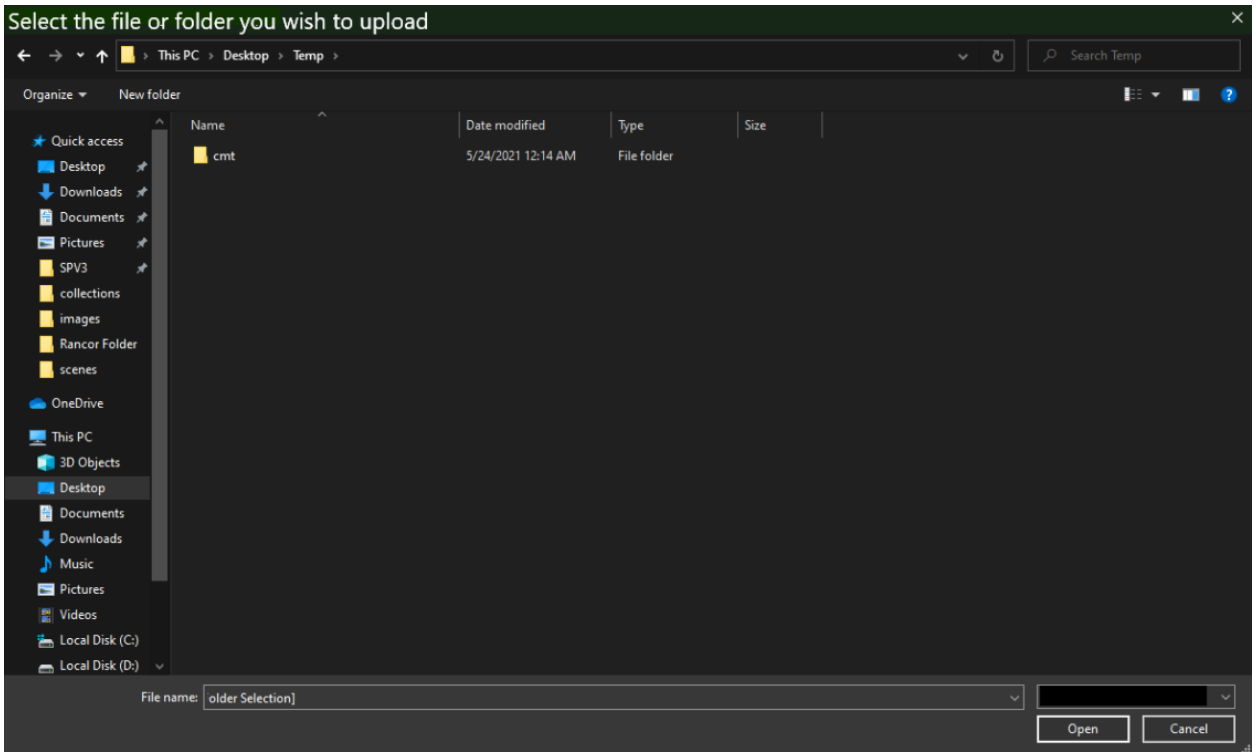
06/29/2021

The prompt popped up upon opening up the client software. Will ask for login credentials, with the option of entering a security question in case the user forgets their password.

The main menu upon logging in. Provides option for downloading files from the DFS, uploading files to the DFS, and logging back out.



After selecting the upload option, prompt a standard file/folder selection within the clients system to upload to the DFS System. In the case of download, prompt a standard file/folder selection of the contents of the DFS system to download onto the client's machine



7.2 User Interface Navigation Flow

Diagram the flow from one screen to the next

7.3 Use Cases / User Function Description

Describe screen usage / function using use cases, or on a per function basis

Section 8 - Other Interfaces

Identify any external interfaces used in the execution of this module, include technology and other pertinent data

8.1 Interface X

Describe interactions, protocols, message formats, failure conditions, handshaking, etc

Section 9 - Extra Design Features / Outstanding Issues

Does not fit anywhere else above, but should be mentioned -- goes here

Section 10 – References

- Requirements Document

06/29/2021

Section 11 – Glossary

Glossary of terms / acronyms

- Distributed File System (DFS) - A file system distributed across multiple locations
- Node - A computer system that is running part of the DFS software and is considered part of the network

06/29/2021

Design Document

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

Github Repository

---

https://github.com/madmartian8/DFS_Distributed-File-System

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

Client Meeting Notes

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

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?)
-

6/21/2021 - Client Notes

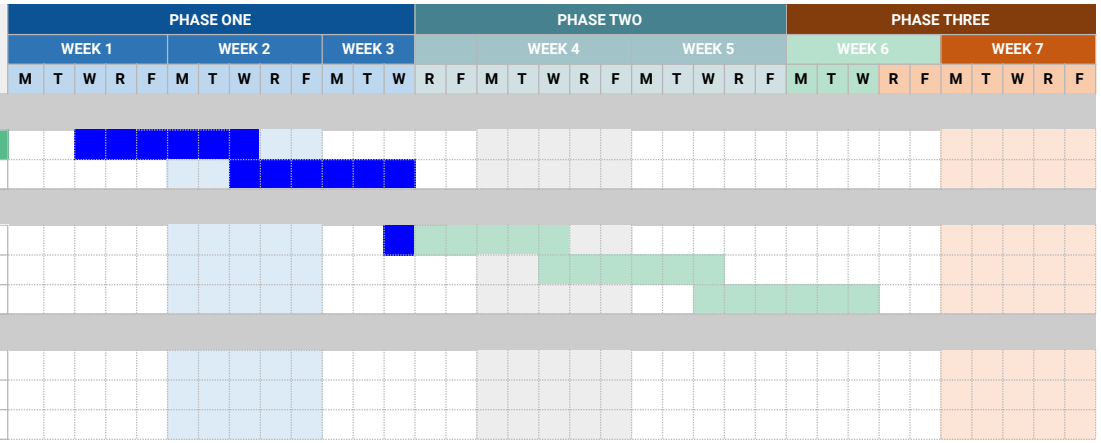
- What is the bare minimum required to make sense?
- What is the maximum we're going to do?
    - Offer the user a choice between the two
- Display stuff contextually
- Change passwords, fixed usernames
- Consider the use cases that are NOT normal. Where the users do the wrong thing, or users are purposely 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

DFS Gantt Chart Schedule

PROJECT TITLE	DFS - Distributed File System	**COMPANY NAME**	Group 4
PROJECT MEMBERS	Aaron Ramos, Lakshmi Salini, Martin Le, Miguel Angel Braojos	**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	6/30/21	7/7/21	7	0%
2.2	Testing	7/8/21	7/15/21	7	0%
2.3	Debugging	7/16/21	7/19/21	6	0%
3	**Project Performance / Polishing**				
3.1	Maintenance	7/14/21	7/16/21	6	0%
3.2	Demo setup	7/14/21	7/25/21	0	0%
3.2.1	Presentation		7/26/21	0	0%



DFS Project Tracking

PROJECT TITLE	DFS - Distributed File System	**COMPANY NAME**	Group 4
PROJECT MEMBERS	Aaron Ramos, Lakshmi Salini, Martin Le, Miguel Angel Braojos	**DATE**	6/9/21

PROJECT DETAILS								DELIVERABLES	
STATUS	**PRIORITY**	**START DATE**	**END DATE**	**DURATION**	**TASK NAME**	**ASSIGNEE**	**DESCRIPTION**	**DELIVERABLE**	**% DONE**
									0%
Not Yet Started	High	6/30/21	7/19/21	0	Login-Account System	Lakshmi			0%
Not Yet Started	High	6/30/21	7/19/21	0	Message System	Aaron			0%
Not Yet Started	High	6/30/21	7/19/21	0	File System	Martin			0%
Not Yet Started	High	6/30/21	7/19/21	0	Client-Server System	Miguel			0%
Demo & Presentation									**0%**
Not Yet Started	High	7/14/21	7/25/21	0	Blackbox testing	All group members			0%
Not Yet Started	High	7/14/21	7/25/21	0	Debugging	All group members			0%
Not Yet Started	High	7/14/21	7/25/21	0	Driver Demo Whitebox Testing	All group members			0%
Not Yet Started	High	7/14/21	7/25/21	0	Presentation	All group members			0%

Software Requirements Specification

Guide

Revision History

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
06/29/2021	1.2		

1. Purpose

This document outlines the requirements for the Distributed File System (DFS).

1.1. Scope

This SRS section will catalog the user, system, and hardware requirements for the DFS system. It will not, however, document how these requirements will be implemented.

1.2. Definitions, Acronyms, Abbreviations

List of any acronyms, terms etc. that need to be defined:

- DFS: Distributed File System
- UML: Unified Modeling Language
- Blockchain: a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain.
- Node: A computer system using the client software which is connected and interacting with the DFS System.

1.3. References

- Use Case Specification
- UML Use Case Diagrams
- Class Diagrams
- Sequence Diagrams

1.4. Overview

This Distributed File System (DFS), is designed to distribute files based on the uploads of each client connected to the server. For every client connected to the server, copies of uploaded files are distributed and stored on each when capacity is available.

2. Overall Description

2.1. Product Perspective

2.2. Product Architecture

The system will be organized into 3 major modules: the Node module, the Server module, and the File module.

Note: System architecture should follow standard OO design practices.

2.3. Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

2.4. Constraints

List of appropriate constraints.

- ➢ Client node systems have JVM 13 and Java SE 13 installed, available, and configured.
- ➢ Host client machine has enough memory space to download the client-server program.

2.5. Assumptions and Dependencies

List of appropriate assumptions

- ➢ Assumption 1: It is assumed that each client node has the capacity to store DFS files.
- ➢ Assumption 2: It is assumed that each client node will always be online and available.

3. Specific Requirements

3.1. Functional Requirements

3.1.1. Common Requirements:

Requirements that apply to all components

3.1.1.1 Runs over a network (TCP).

3.1.1.2 The system should provide an interface for users to interact with client-servers.

3.1.2. Node Module Requirements:

Module specific requirements

Example:

3.1.2.1 Users should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.3. Server Module Requirements:

Module specific requirements

Example:

3.1.3.1 Users should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.4. File Module Requirements:

Module specific requirements

Example:

3.1.4.1 Users should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.2. External Interface Requirements

Module specific requirements

3.2.1 The login system must provide an interface to the DFS client system administered by the DFS server system so that users can be automatically connected for the files in which they have uploaded. The interface is to be in a console formatted text displaying appropriate qualifying fields: login, register, change password, action. Where “action” is whether the student has added or dropped the course. The file will be exported nightly and will contain new transactions only.

3.3. Internal Interface Requirements

Module specific requirements

3.3.1 The system must process a data-feed from the accounts system such that user accounts are stored along with the client node identifier information. Data feed will be in the form of a interface file that is exported from the accounts system.
3.3.2 The system must process a data-feed from the file system that contains filepath records. The feed will be in the form of an array of type File and will be exported from the FileManager system with new Message packs. The fields included in the file are fileName, filePath, fileSize, fileDate, and fileType.

4. Non-Functional Requirements

4.1. Security and Privacy Requirements

4.1.1 The System must protect bare public access to DFS through login hurdle.

4.2. Environmental Requirements

4.2.1 System cannot require that any code interpreter/code base other than a JVM and JAVA SE 13 be installed on user computers.

4.2.2 System must make use of the Java SE 13 library methods.

4.2.3 System must be deployed on existing JVM supported machine.

4.3. Performance Requirements

4.3.1 System must render all console text display in proper format.

Use Cases

Use Case ID#001: UPLOAD
Use Case Name: Upload File
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told system which file and logged in successfully
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 compares the name of the file to be uploaded with the files inside the DFS system
4. File is valid and ready to be stored
5. DFS sends file to the node it wants to store it into
6. Display “File uploaded” message to client

Extensions or Alternate Flows:
Exceptions:

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

Related Use Cases:

Use Case ID#002: DOWNLOAD
Use Case Name: Download File
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told system which file and logged in successfully
Post-conditions: Server has file on their system
Basic Flow or Main Scenario:

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

Extensions or Alternate Flows:
Exceptions:

- Download is interrupted by network problems

Related Use Cases:

Use Case ID#003: UPLOAD_FILE_IN_SYSTEM
Use Case Name: Overwrite File
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told the system which file to upload to the DFS system and logged in successfully.
Post-conditions: DFS has overwritten an existent file to a node
Basic Flow or Main Scenario:

1. User presses “Upload file” button
2. Client sends networked message to DFS
3. DFS compares the name of the file to be uploaded with the files inside the DFS system
4. File is found out to be the same as another file already in the DFS system
5. Server sends networked message to Client
6. Display an option to Client prompting the overwriting of the existent file in exchange for the new one
7. Client accepts and decides to overwrite the existing file
8. DFS sends file to the node it wants to store the new file into
9. Display “File uploaded” message to client

Extensions or Alternate Flows:
Exceptions:

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

Related Use Cases:

Use Case ID#004: DOWNLOAD_FILE_NOT_IN_SYSTEM
Use Case Name: File Unavailable for Download
Relevant Requirements:
Primary Actor: Client
Pre-conditions: Client has told system which file and logged in successfully
Post-conditions: Server doesn't have file on their system
Basic Flow or Main Scenario:

1. User selects file from DFS system
2. Clients system sends networked message to DFS requesting the specific file
3. DFS looks up which node has the requested file
4. DFS fails to find the required file
5. DFS sends networked message to Client
6. Display an option to Client stating that the file selected is not in the system

Extensions or Alternate Flows:

Exceptions:

- Download is interrupted by network problems

Related Use Cases:

Use Case ID#005: LOGIN_CREATE_ACCOUNT

Use Case Name: Login create an account

Relevant Requirements:

Primary Actor: Client

Pre-conditions: First time using the DFS system

Post-conditions: Successful account creation

Basic Flow or Main Scenario:

1. User opens the program
2. Clicks the button to create an account
3. Enters their ID provided by the company beforehand
4. User enters a unique username
5. User enters a password
6. User selects a security question and enters the answer
7. Display that the account has been created successfully
8. User is prompted to the main menu where the User can login in with his new credentials

Extensions or Alternate Flows:

- If the username entered at the creation of the account is already being used by another person in the DFS system, then the User will be prompted to type in a different username

Exceptions:

- Assumption is made that client cannot forget their security question's answer
- Assumption that ID is correct and matches the ID's from the business

Related Use Cases:

Use Case ID#006: LOGIN_INCORRECT_PASSWORD

Use Case Name: Login with incorrect password

Primary Actor: Client

Pre-conditions: Client is logging into the DFS system but forgets its password

Post-conditions: Client logins successfully

Basic Flow or Main Scenario:

1. User clicks button to login into the DFS system
2. User enters correct username but invalid password
3. User fails to enter their password 5 times
4. After the 5th time, the program displays the security question to the User
5. User enters the correct answer to the security question
6. User is then shown their password or asked to change it if needed
7. User is logged in

Extensions or Alternate Flows:

Exceptions:

- Assumption is made that client cannot forget their security question's answer

Related Use Cases:

Use Case ID#007: LOGIN_SUCCESS
Use Case Name: Login successfully
Primary Actor: Client
Pre-conditions: Client wants to log into the DFS system
Post-conditions: Client logins successfully
Basic Flow or Main Scenario:

1. User clicks button to login into the DFS system
2. User enters correct username
3. User enters correct password
4. User is logged in and the menu is displayed

Extensions or Alternate Flows:
Exceptions:

- Assumption is made that client cannot forget their security question's answer

Related Use Cases:

Use Case ID#008: CHANGE_PASSWORD
Use Case Name: change password
Primary Actor: Client
Pre-conditions: Client successfully logs into the DFS system
Post-conditions: Client changes password successfully
Basic Flow or Main Scenario:

1. User selects change password button
2. User enters old password successfully
3. User enters new password twice and both password have to be the same
4. User is prompted back to the main menu

Extensions or Alternate Flows:
If the user fails to enter the old password then the user has unlimited attempts in order to change its password and an option is displayed for the user to enter their security question's answer to change the password.
Exceptions:

- Assumption is made that client cannot forget their security question's answer

Related Use Cases:

(UML) Unified Modeling Language

«interface»
Login

+ StartMenu()
+ DisplayLogin(params):returnType
+ DisplayRegister(params)
+ DisplayChangePw()

Client (Data Node)

+ IP address:String = 127.0.0.1
+ MessageStatus:MessageStatus = Ready

+Connect()

«enumeration»
FileStatus

-Available
-Corrupted
-Copied
-Deleted
-Missing

File

+FileName:String
+FileDate:Date
+Type:enumFileType
+FileSize:long
+Data:FileData
+FileLocation:FilePath

Account

-ID:String
-Password:String
-Security Question:String

AccountManager

+ SearchAccount(String id):Boolean(Success/Fail)
+ Register(String id, String pw):Boolean(S/F)
+ ChangePassword:Boolean(S/F)

LoginManager

+ Login(String id, String pw):Boolean(Success/Fail)
+ Register(String id, String pw):Boolean(S/F)
+ ChangePassword:Boolean(S/F)

«interface»
Accounts

+AccountSystem:Acount[ ]

+VerifyLogin(String id, String pw): Boolean]

Logger (History)

+ActionHistory[ ]
+ErrorHistory[ ]

+LogAction(MessageType):Boolean(S/F)
+LogError(MessageType):Boolean(S/F)
+GetLog()
+GetErrorLog()

FileManager

+ WriteFile(params):Boolean(Success/Fail)
+ ReadFile(params):FileStatus
+ OverWrite():Boolean(Success/Fail)
-GetSizeOfFile()
-GetDateOfFile()

«interface»
Files

+FileSystem:File[ ]

Message

+MsgType:MessageType
+Data

«enumeration»
MessageType

-Authentication
-ActionStatus
-Communication
-IP
-File

«enumeration»
MessageStatus

-Sent
-Sending
-Received
-Failed
-InQueue

<<entity>>
MessageHandler

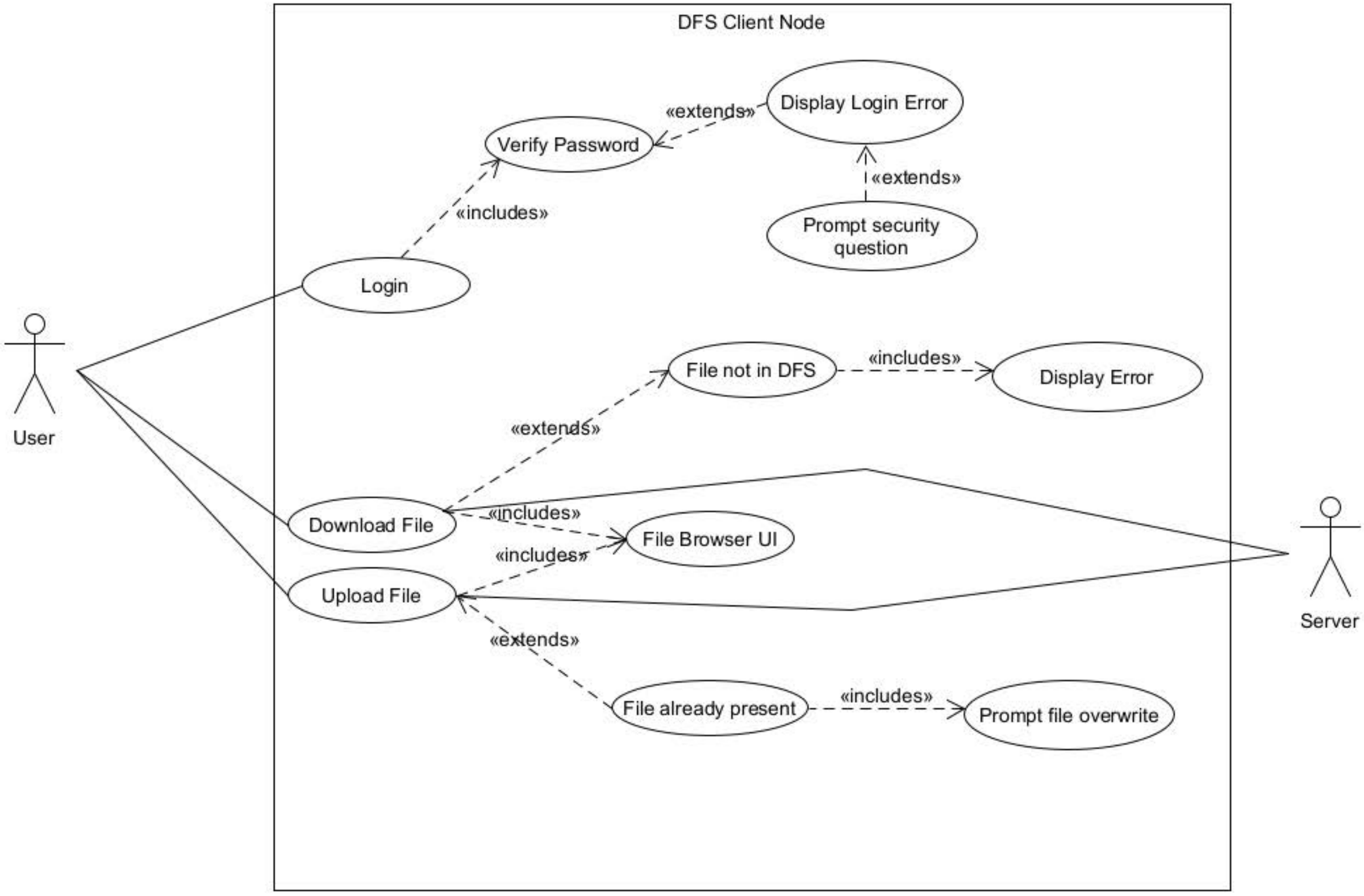
+ MessageQueue:Queue[ ]

+SendMessage(params)
+PackMessage()
-UnpackMessage()

Server (Name Node)

+Accounts()
+Logger()

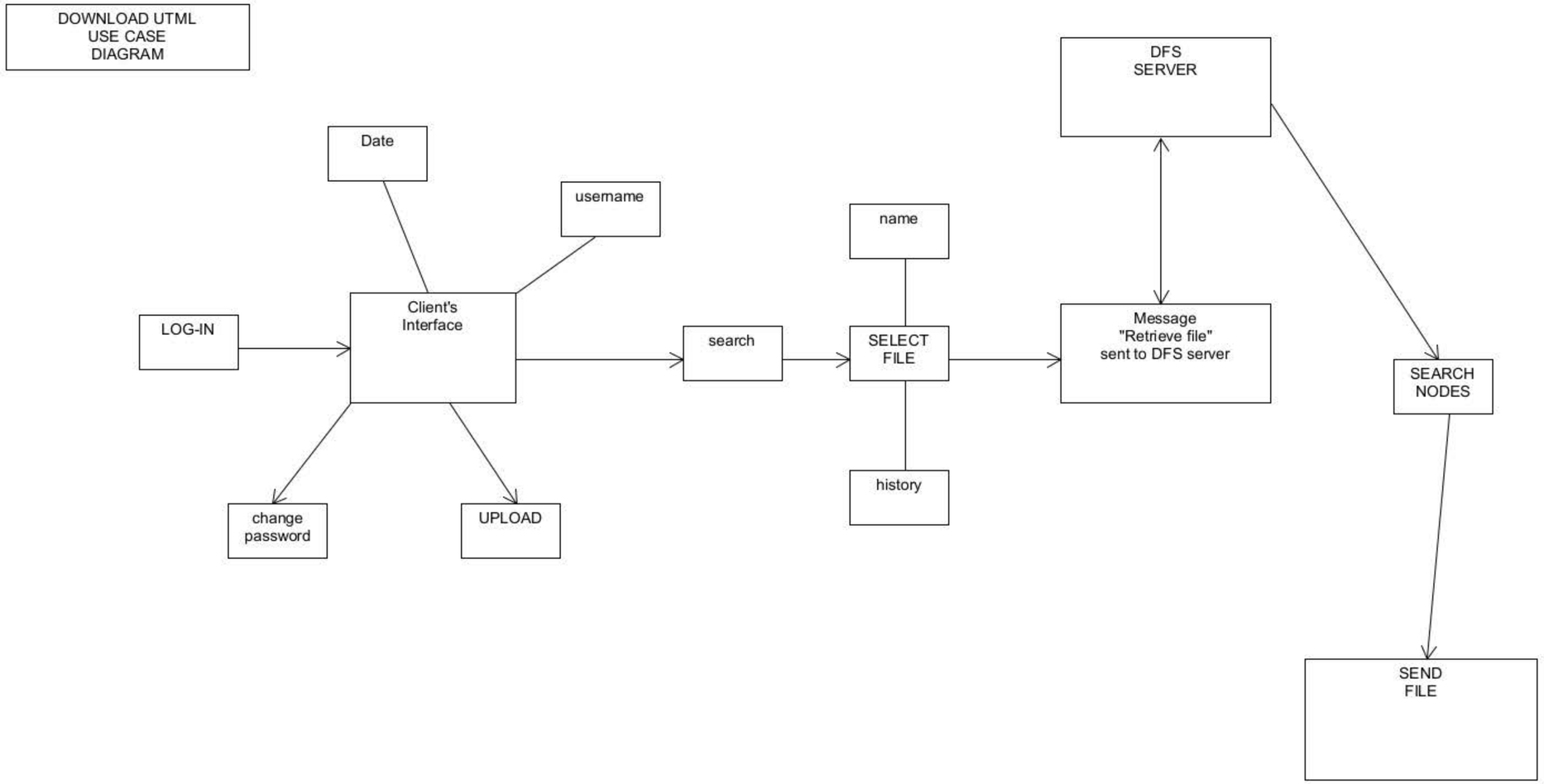
+FileManager()
+MessageHandler()


DFS Client Node
Display Login Error
«extends»
Verify Password
«extends»
«includes»
Prompt security question
Login
File not in DFS
«includes»
Display Error
«extends»
User
Download File
«includes»
File Browser UI
«includes»
Upload File
Server
«extends»
File already present
«includes»
Prompt file overwrite

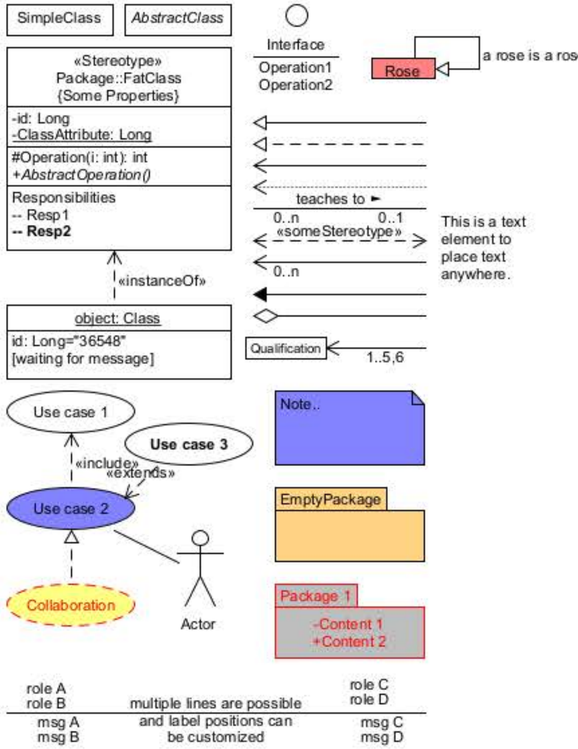
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

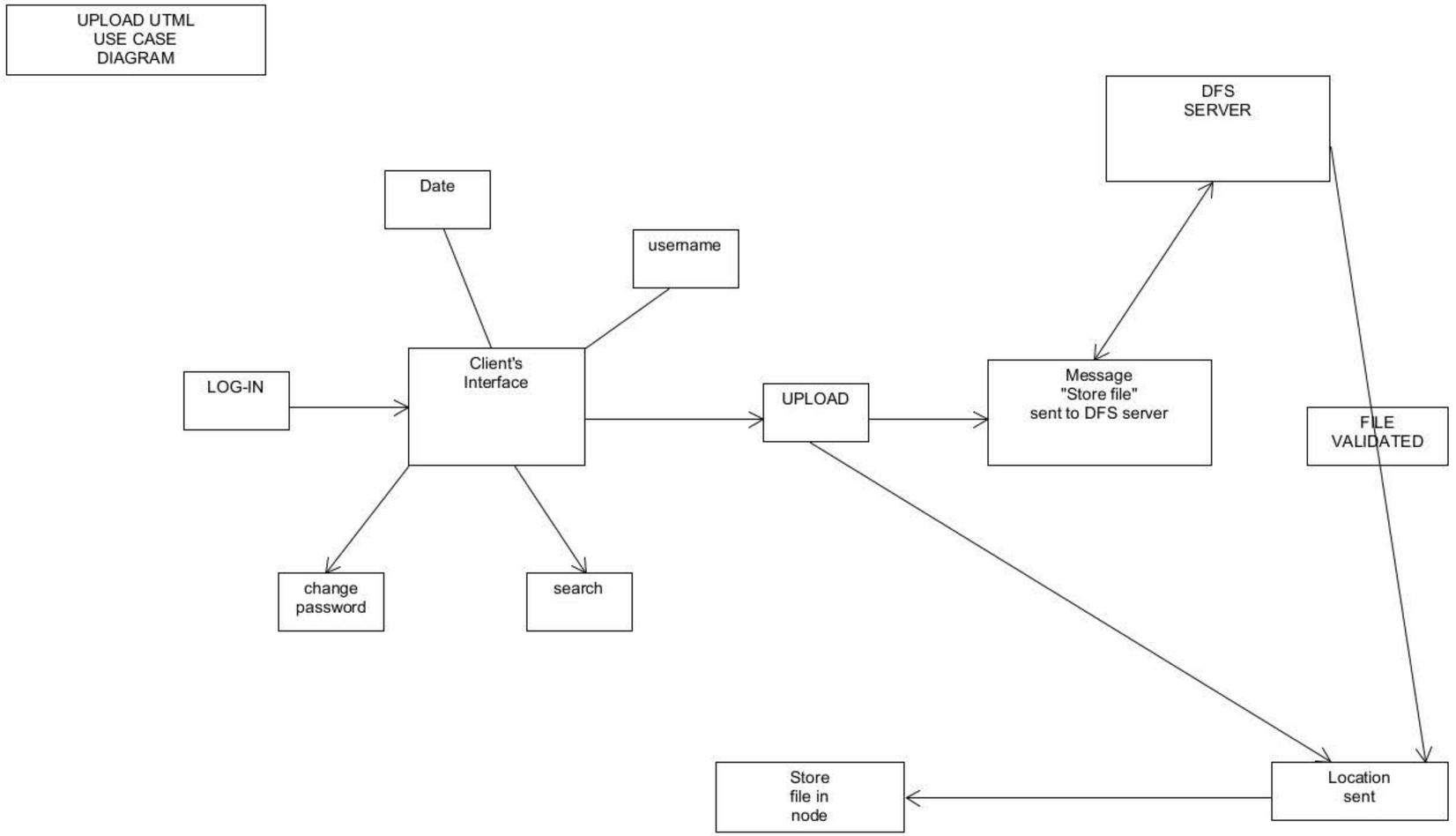
Type here to search

59°F Cloudy 8:20 AM 6/30/2021

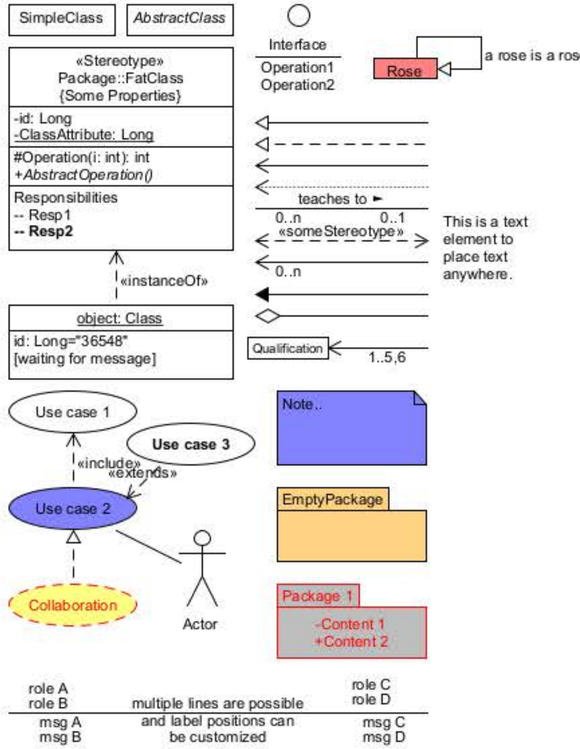
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

Type here to search

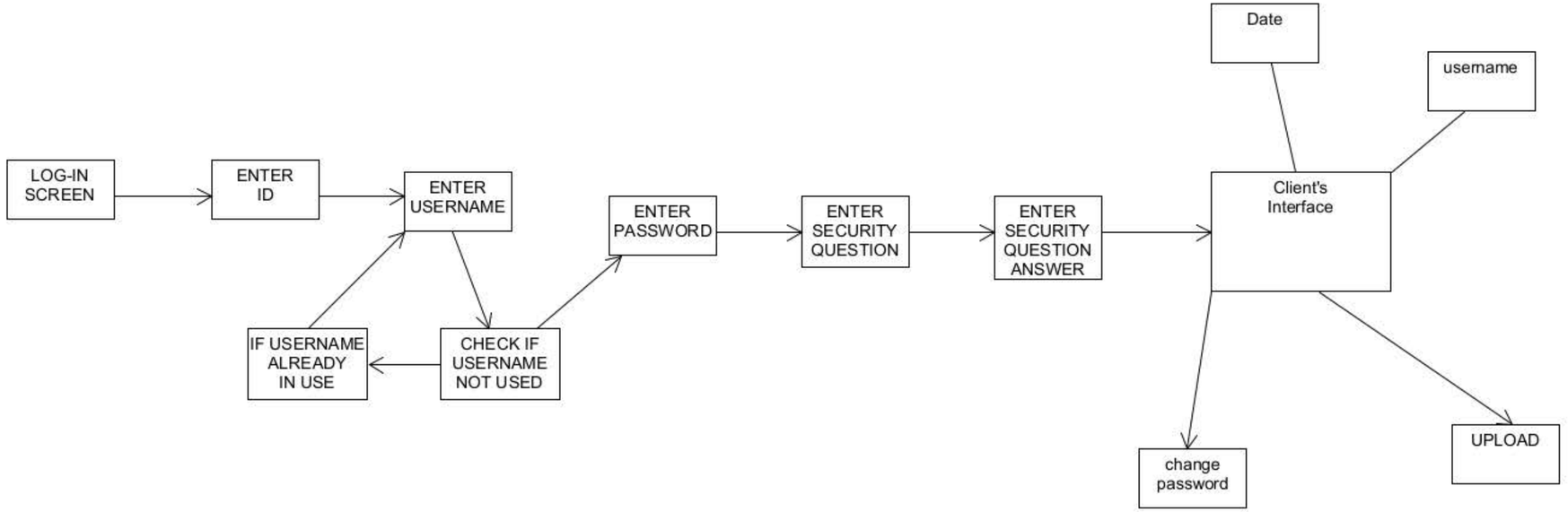
59°F Cloudy 8:22 AM 6/30/2021

UMLet - Free UML Tool for Fast UML Diagrams

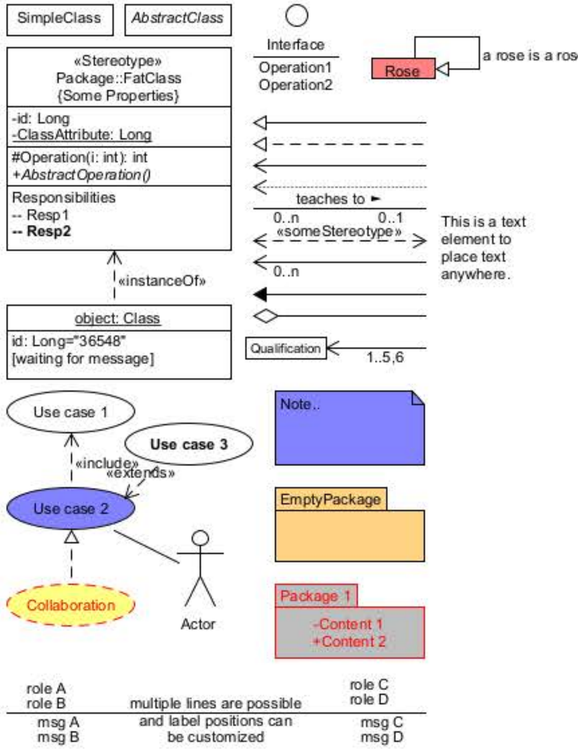
File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM

LOGIN
CREATE ACCOUNT
UTML
USE CASE
DIAGRAM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


////////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

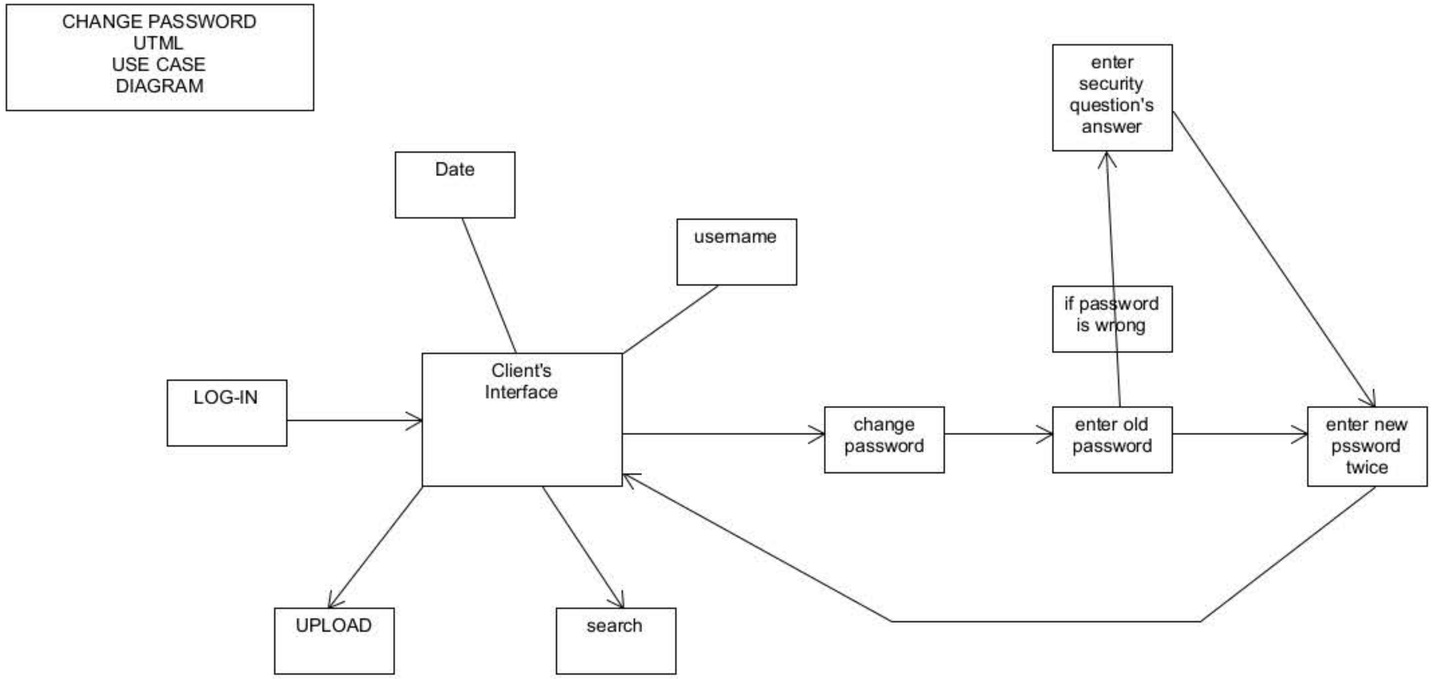
Type here to search

59°F Cloudy 8:21 AM 6/30/2021

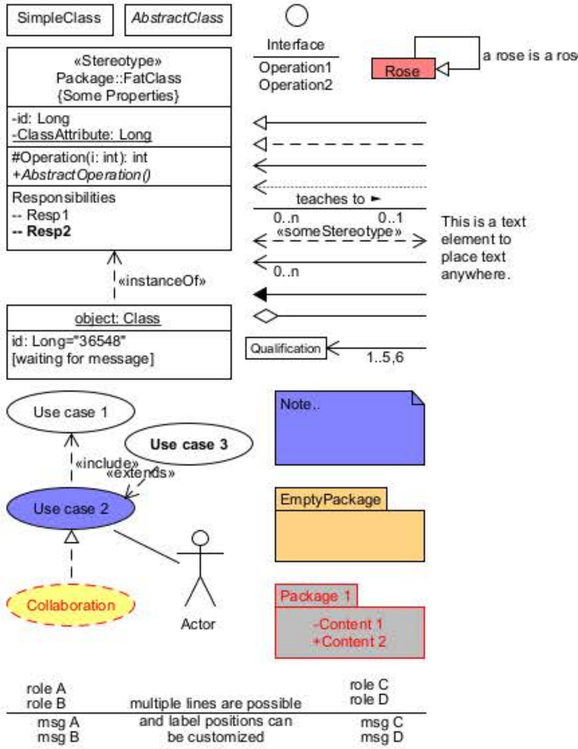
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements



Properties

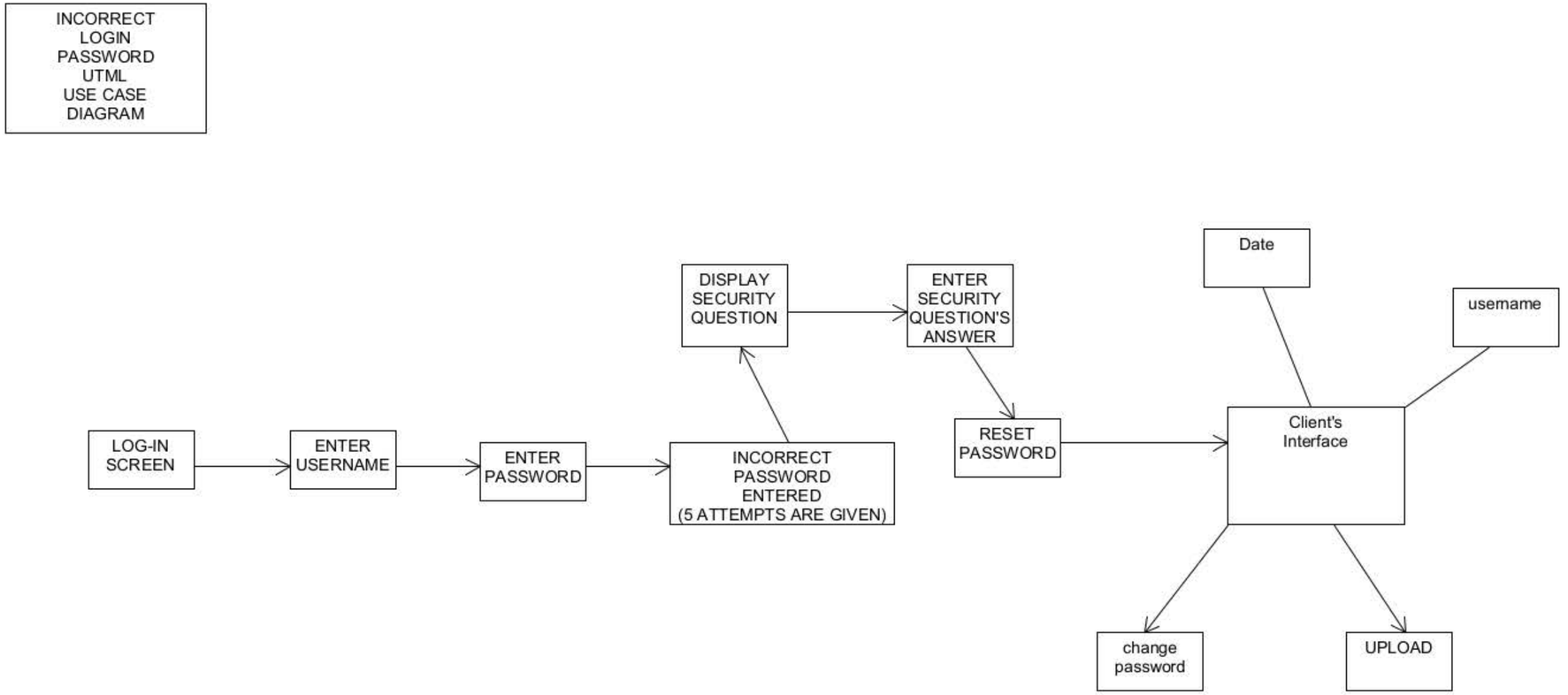
```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

Type here to search 59°F Cloudy 8:21 AM 6/30/2021

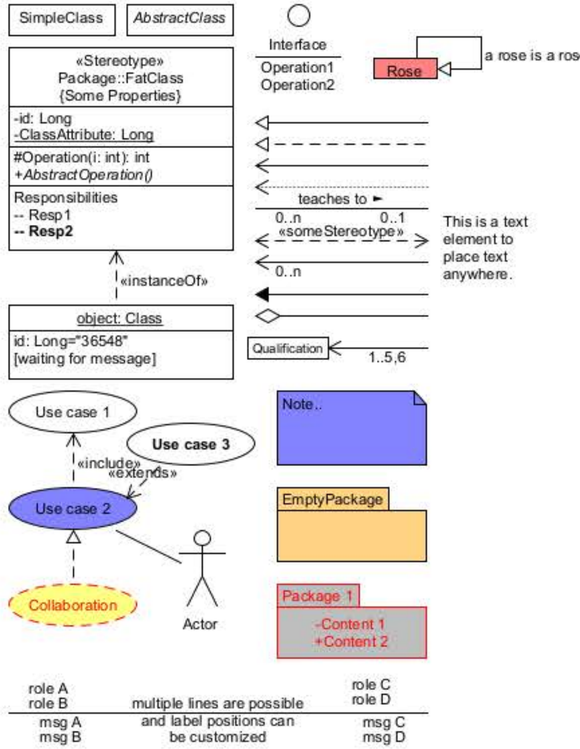
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

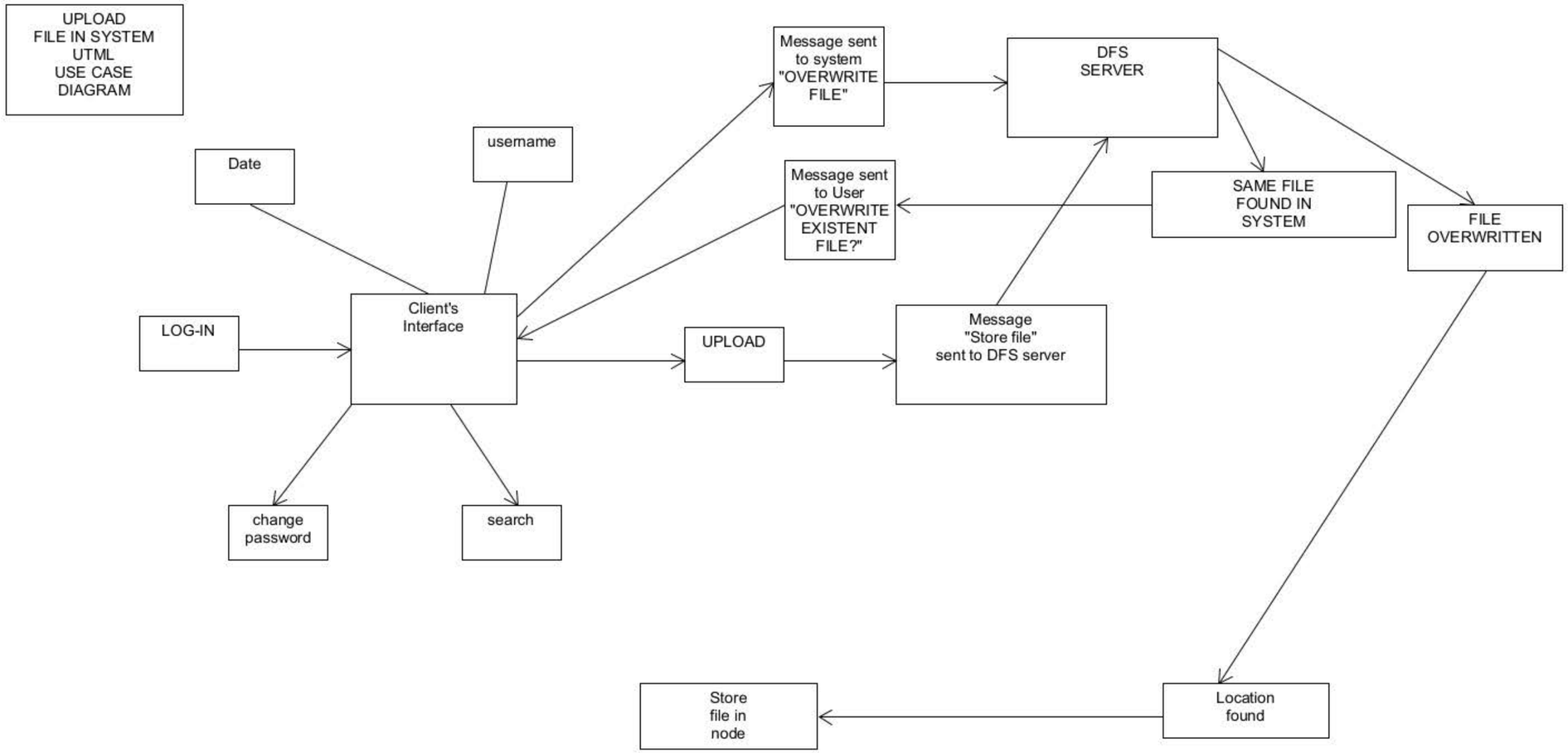
Type here to search

59°F Cloudy 8:21 AM 6/30/2021

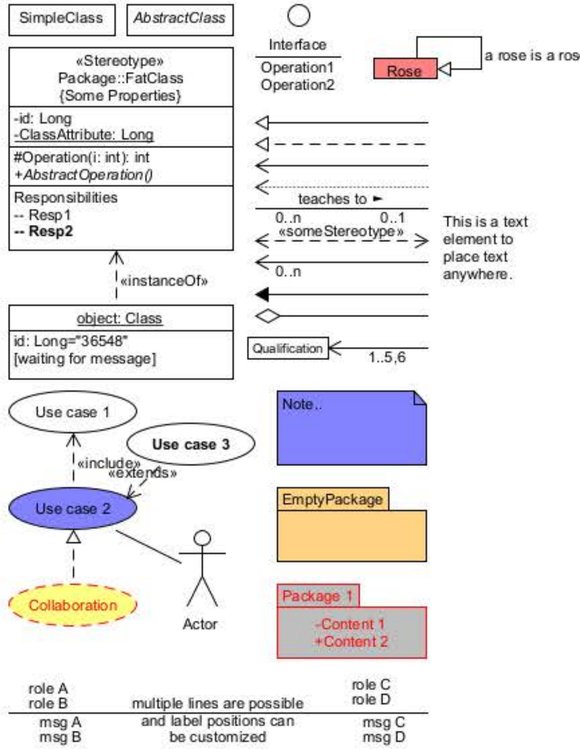
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

Type here to search

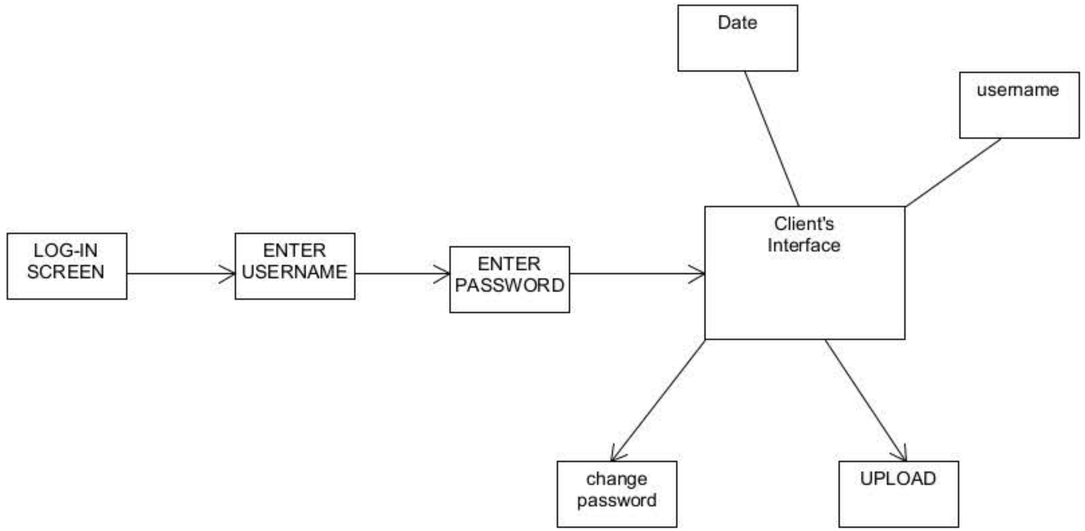
59°F Cloudy 8:22 AM 6/30/2021

UMLet - Free UML Tool for Fast UML Diagrams

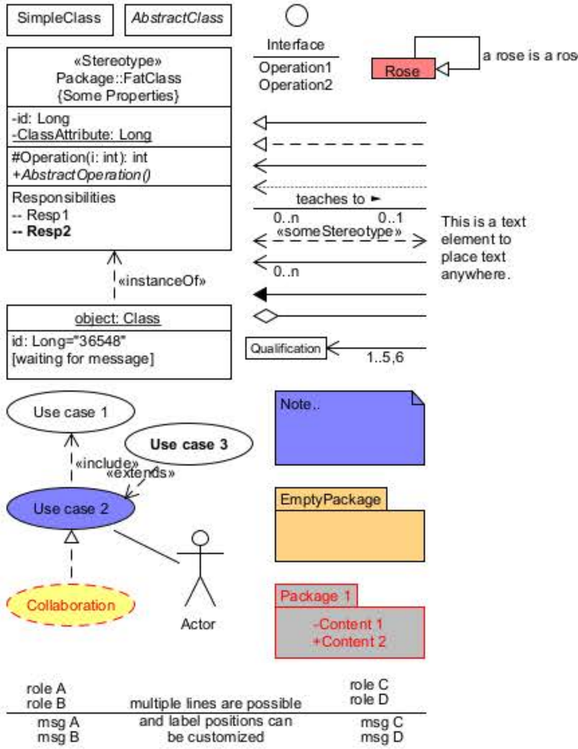
File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM

SUCCESSFULL
LOGIN
UTML
USE CASE
DIAGRAM



UML Common Elements



Properties

```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

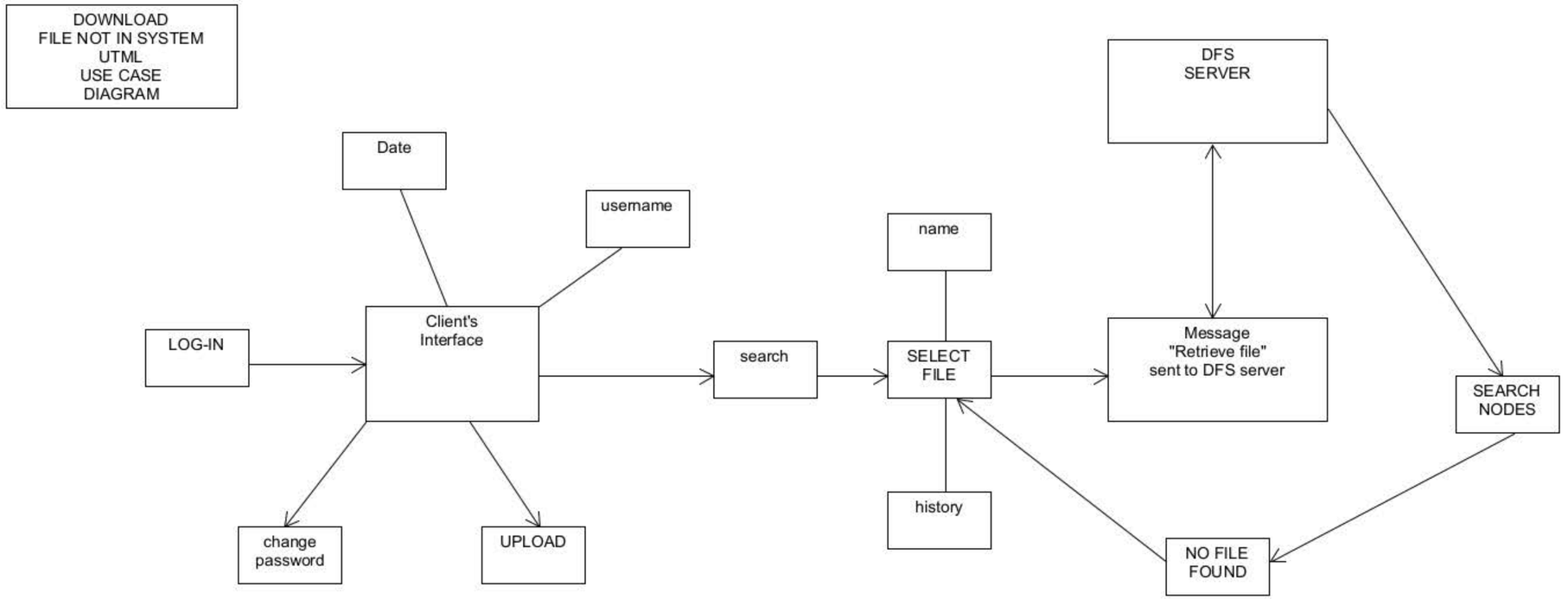
Type here to search

59°F Cloudy 8:21 AM 6/30/2021

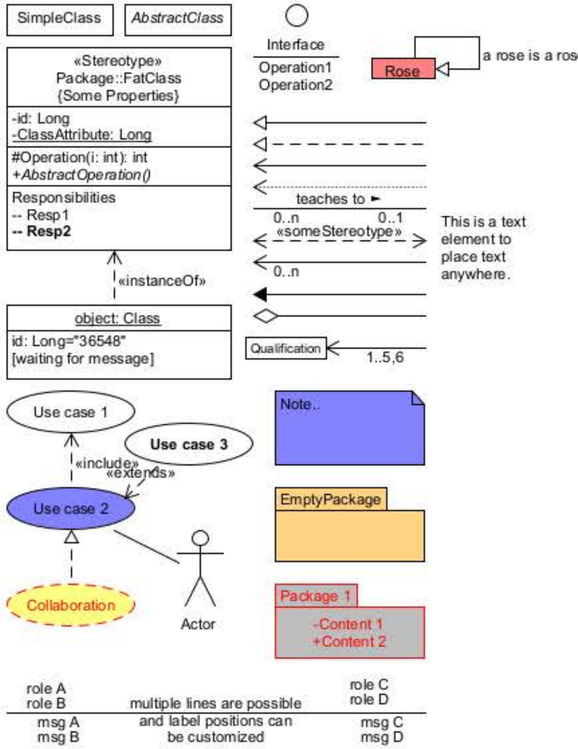
UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help Search: Zoom: 100% Mail diagram

CHANGE_PASSWORD | DOWNLOAD | DOWNLOAD_FILE_NOT_IN_SYSTEM | LOGIN_CREATE_ACCOUNT | LOGIN_INCORRECT_PASSWORD | LOGIN_SUCCESS | UPLOAD | UPLOAD_FILE_IN_SYSTEM



UML Common Elements

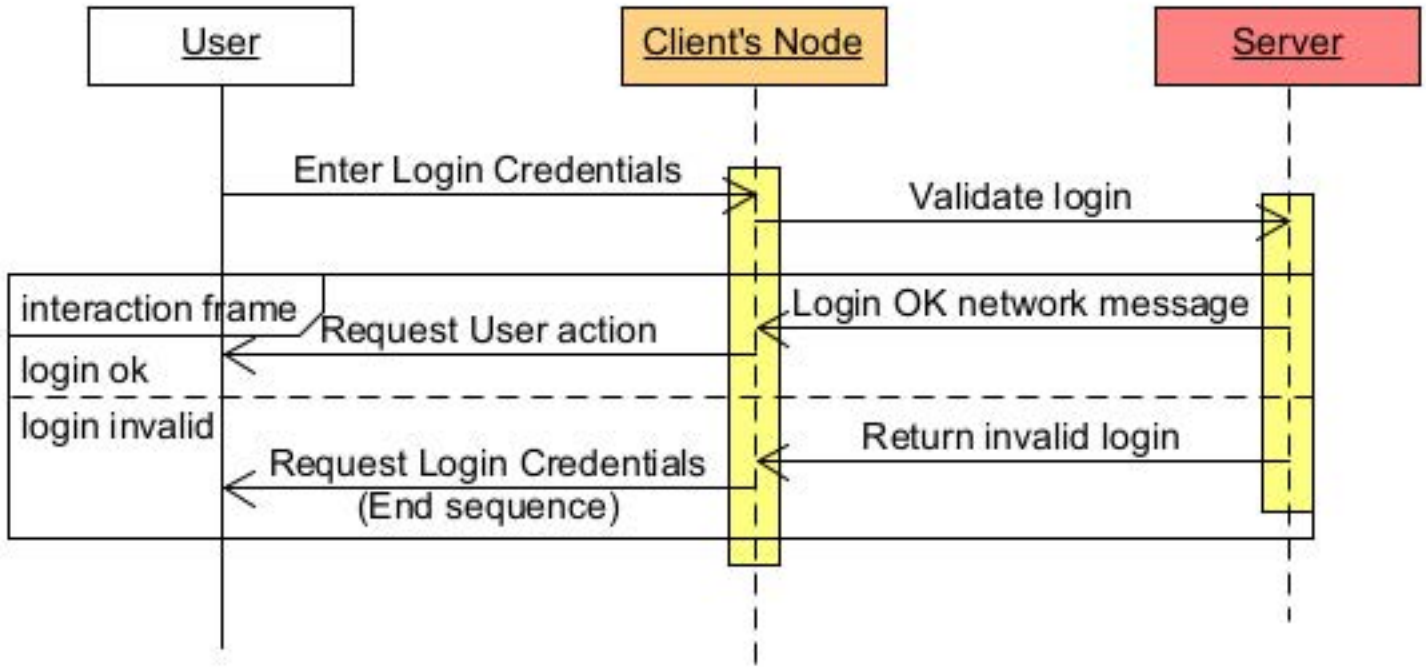


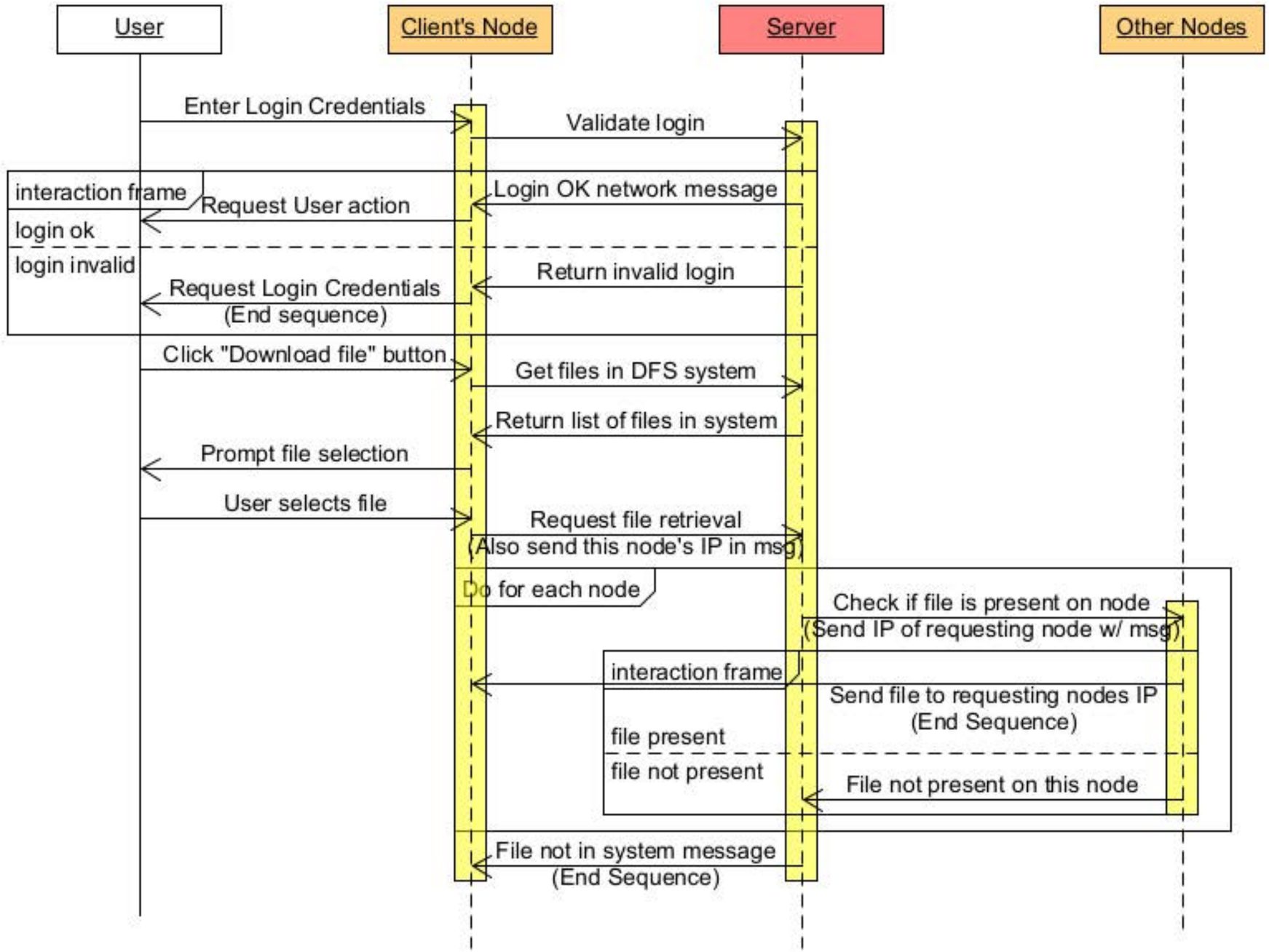
Properties

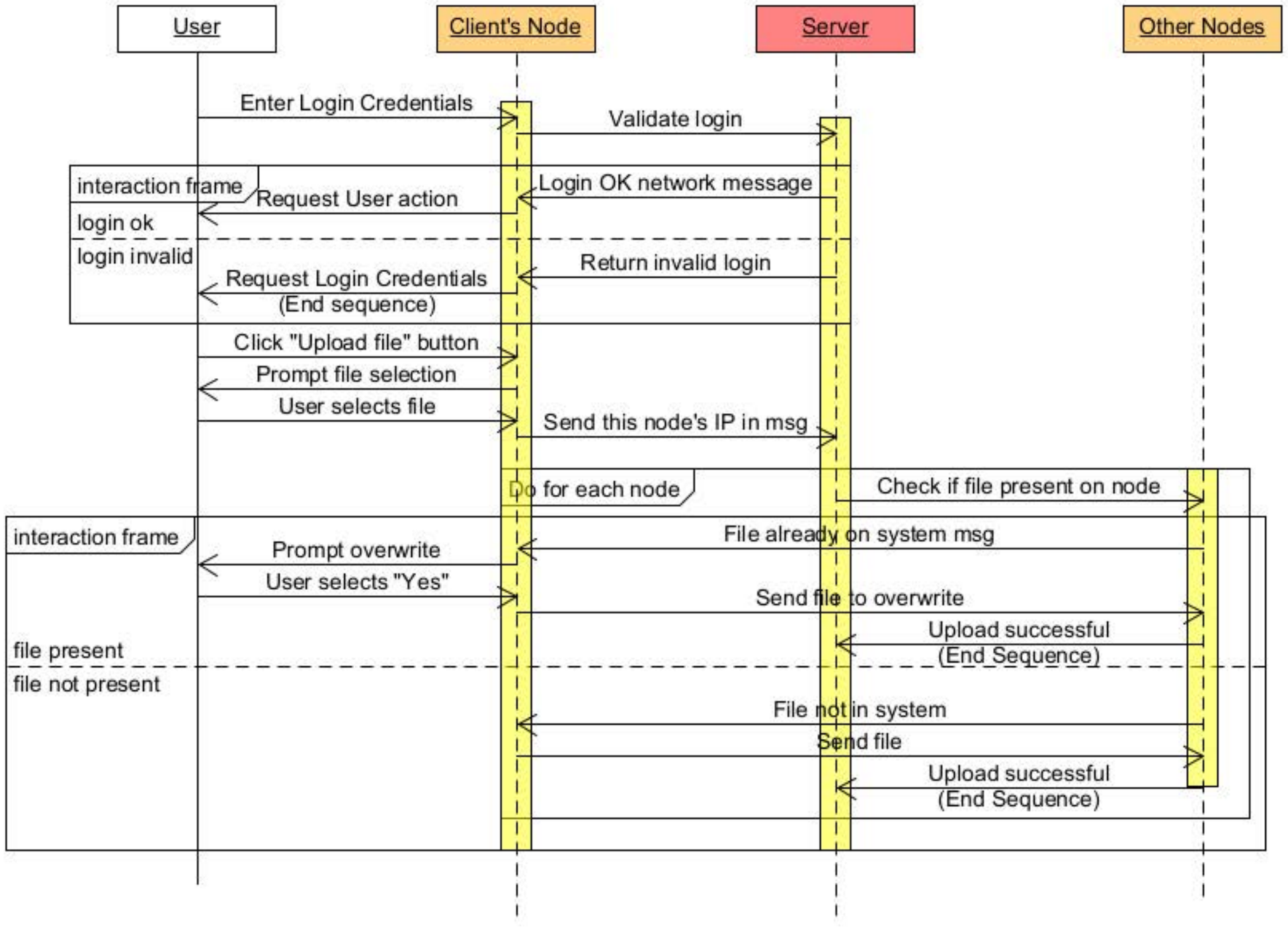
```
// Uncomment the following line to change the fontsize and f
// fontsize=14
// fontfamily=SansSerif //possible: SansSerif,Serif,Monospac


//////////////////////////////////////////////////////////////
// Welcome to UMLet!
//
// Double-click on elements to add them to the diagram, or t
// Edit elements by modifying the text in this panel
// Hold Ctrl to select multiple elements
// Use Ctrl+mouse to select via lasso
//
// Use +/- or Ctrl+mouse wheel to zoom
// Drag a whole relation at its central square icon
```

Type here to search

59°F Cloudy 8:20 AM 6/30/2021


User
Client's Node
Server
Enter Login Credentials
Validate login
interaction frame
Request User action
Login OK network message
login ok
login invalid
Return invalid login
Request Login Credentials
(End sequence)


User
Client's Node
Server
Other Nodes
Enter Login Credentials
Validate login
interaction frame
Request User action
Login OK network message
login ok
login invalid
Return invalid login
Request Login Credentials
(End sequence)
Click "Download file" button
Get files in DFS system
Return list of files in system
Prompt file selection
User selects file
Request file retrieval
(Also send this node's IP in msg)
Do for each node
Check if file is present on node
(Send IP of requesting node w/ msg)
interaction frame
Send file to requesting nodes IP
(End Sequence)
file present
file not present
File not present on this node
File not in system message
(End Sequence)


User
Client's Node
Server
Other Nodes
Enter Login Credentials
Validate login
interaction frame
Request User action
Login OK network message
login ok
login invalid
Return invalid login
Request Login Credentials
(End sequence)
Click "Upload file" button
Prompt file selection
User selects file
Send this node's IP in msg
Do for each node
Check if file present on node
interaction frame
File already on system msg
Prompt overwrite
User selects "Yes"
Send file to overwrite
Upload successful
(End Sequence)
file present
file not present
File not in system
Send file
Upload successful
(End Sequence)