Media Sharing Software

Introduction

The software provides a platform to share media files over the network. The framework has a server which is continuously running and has media files (Audio and Video). These files may be present initially or may be uploaded by the users of the system. These media files can be accessed by the users of the system which are called clients.

Functionality

- The User first needs to sign up to be able to use the system. A userid and a password is required for this. The sign up details of the user are stored on the server.
- A registered user then performs a login on the GUI. This login request would go to the server to verify if the user is registered. The confirmation whether the sign in was successful comes from the server. Once a user is logged in he can perform various actions which are described below. If the login fails, the session of the user is terminated.
- After a successful login, the user can view the files present on the server. These files would be displayed on the GUI for the client.
- The user can send a download request for a particular file item. The file can be a normal text file or a media file.
- On having received the media file, the user has the option to play the file from within the software. The media files would be played in the VLC media player.
- The user also has the option of uploading a file to the server. The main purpose of writing this software is to allow users to collaborate on the media content they have. Sharing it with others gives everybody a bigger media rich collection of files to be played.

Technical Challenges

- Everything that is send over the network is in the form of bytes. The main challenge is to read the files byte by byte or buffering a set of bytes before they are put onto a stream. On the other end of a connection over the network, the challenge is to read the stream byte by byte and and create a new file out of it.
- The readers available in java to read an inputstream are all non blocking. So if a client has requested a file and is waiting for the file to be put on the stream so that it can read it, it would not wait for the server to put the bytes on the stream. It would simple try to read bytes which it would not receive and hence close before receiving anything. The challenge is to make it blocking somehow to avoid this problem.

Technical Advantages

The files are being transferred over the network using the TCP through java sockets. A server is always listening for connections from the clients. Once a client requests a connection to the server, the server after verifying the authenticity of the client, spawns a new socket in a new runnable thread. This new thread is exclusive to the client which also consists an exclusive socket connection. The client on this socket connection explicitly mentions what is the next required action through our defined protocol. This action can be any out getting a file to uploading one.
The advantage of having a multithreaded system is that the server after having spawned a new thread for a requesting client becomes available to be requested further by new clients. Hence a second client after one is already for example downloading a file, can easily connect and proceed to complete its own actions.

Time Invested(by each team member)

- Time Invested on Design - 5 hours
- Time Invested on Coding - 15 hours
- Time Invested on Testing - 1.5 hours

Changes in Design

The client.gui package initially consisted of 7 panel classes which were extended JPanel and were designed for showing different windows like Sign Up window, Sign In Window etc. This design was made to separate different GUI functionality. But we later changed it to just one class MainWindow.java because of the following reasons-

- The button action listener function could not communicate amongst the various members that are an object of the MainWIndow.
- Swing provides CardLayout in which different screens can be toggled with a button click hence it is a smooth and convenient transition between different gui screens.

We did not implement the streaming functionality because the streaming algorithms and implementation were much more complex than we anticipated.

Sonar

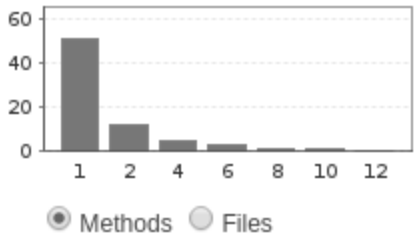
Lines of code
1,404
1,962 lines
533 statements
27 files

Classes
27
5 packages
79 methods
17 accessors

Comments
8.3%
127 lines
17.9% docu. API
55 undocu. API

Duplications
5.4%
106 lines
4 blocks
4 files

Complexity
1.8 /method
5.1 /class
5.1 /file
Total: 139



Violations
286

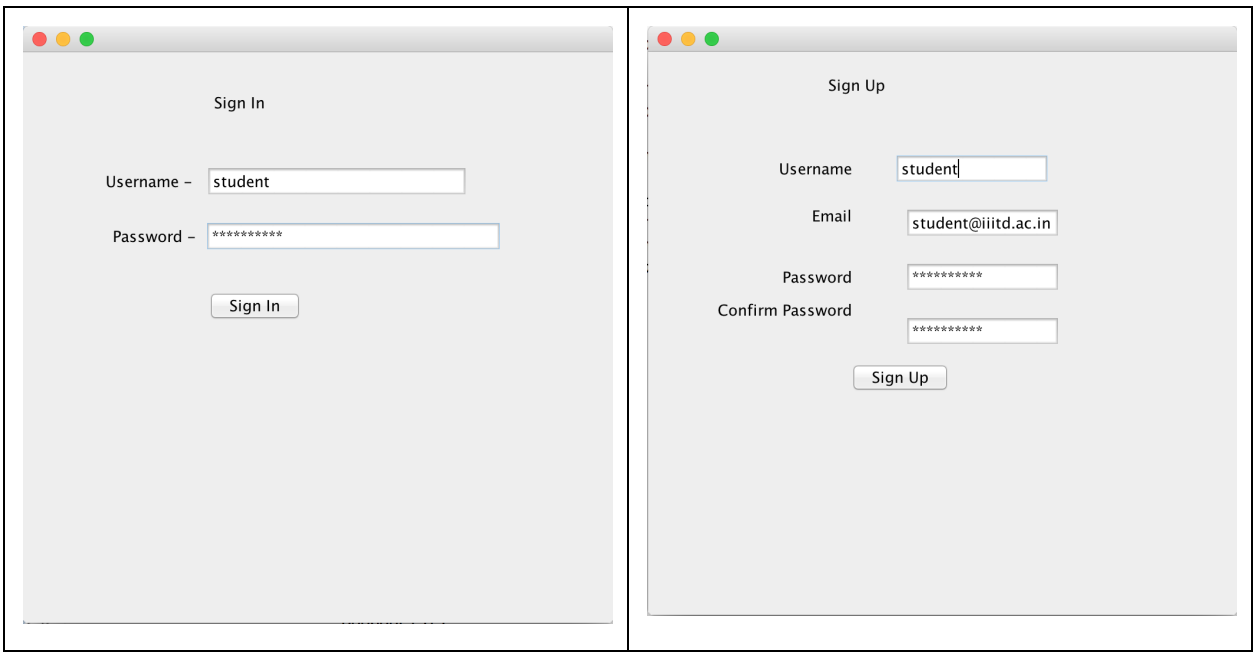
Rules compliance
61.7%

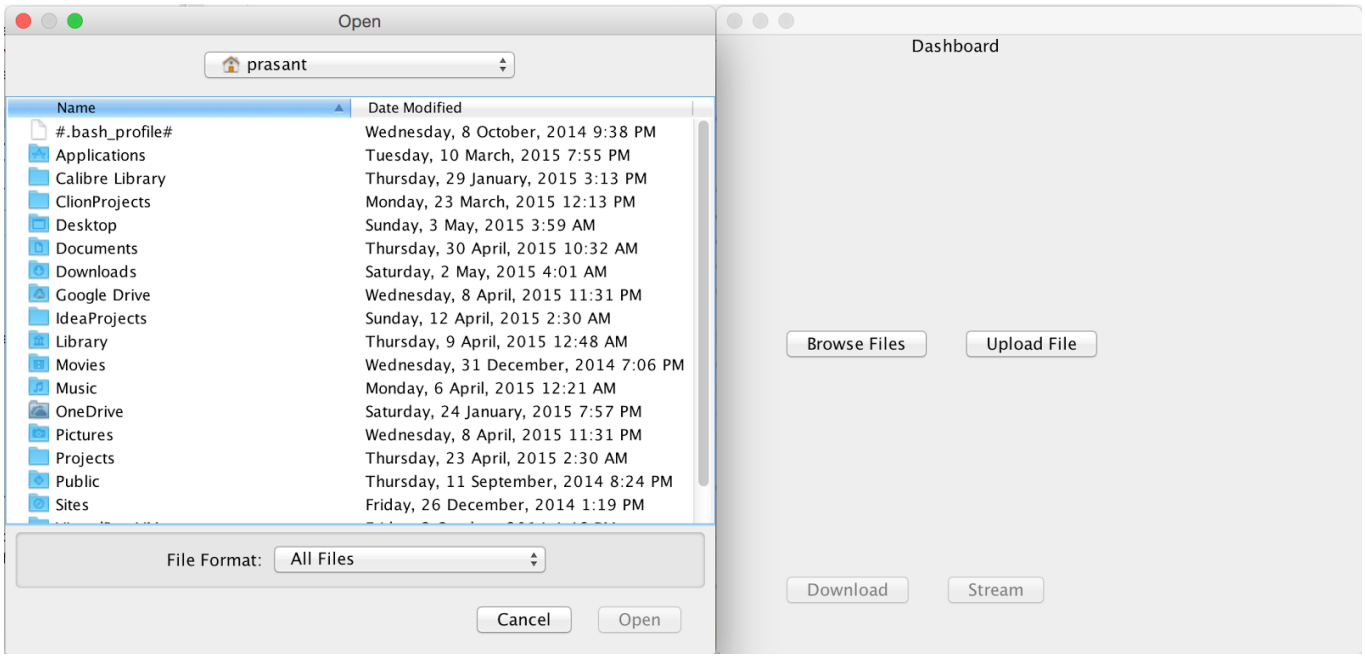
Blocker	0	
Critical	0	
Major	130	
Minor	148	
Info	8	

Unit tests coverage
-

Unit test success
0 tests

GUI


Sign In
Username –
student
Password –
Sign In
Sign Up
Username
student
Email
student@iiitd.ac.in
Password
Confirm Password
Sign Up


Open
prasant
Name
Date Modified
#.bash_profile# Wednesday, 8 October, 2014 9:38 PM
Applications Tuesday, 10 March, 2015 7:55 PM
Calibre Library Thursday, 29 January, 2015 3:13 PM
ClionProjects Monday, 23 March, 2015 12:13 PM
Desktop Sunday, 3 May, 2015 3:59 AM
Documents Thursday, 30 April, 2015 10:32 AM
Downloads Saturday, 2 May, 2015 4:01 AM
Google Drive Wednesday, 8 April, 2015 11:31 PM
IdeaProjects Sunday, 12 April, 2015 2:30 AM
Library Thursday, 9 April, 2015 12:48 AM
Movies Wednesday, 31 December, 2014 7:06 PM
Music Monday, 6 April, 2015 12:21 AM
OneDrive Saturday, 24 January, 2015 7:57 PM
Pictures Wednesday, 8 April, 2015 11:31 PM
Projects Thursday, 23 April, 2015 2:30 AM
Public Thursday, 11 September, 2014 8:24 PM
Sites Friday, 26 December, 2014 1:19 PM
File Format:
All Files
Cancel
Open
Dashboard
Browse Files
Upload File
Download
Stream