SYNCER: A FILE HOSTING SERVICE

Syncer: A File Hosting Service

Course: Software Project Lab II (SE 505)

Submitted by

Team Yazilim

Moumita Asad (731)

Rafed Muhammad Yasir (733)

Supervised by

Dr. B M Mainul Hossain

Assistant Professor

IIT, DU

Institute of Information Technology

University of Dhaka

10th May, 2017

Table of Contents

List of Figures

List of Tables

Chapter 1: Project Definition

This chapter describes the project definition of our project.

1.1 Introduction

Often we need to work on a document from different places. But for that, we have the hassle of downloading the file before use and uploading it after each modification. To tackle this issue we have created this program called Syncer that will synchronize files in multiple pcs. If a change is made to a file in one pc, the files in other pcs will also be updated. If necessary a user can also share a file with others.

1.2 About the Project

Syncer is a file hosting service. At first, users have to install the client program. Only registered users can use this program. After registration, an admin approves new accounts. Admin may also remove existing users. Each user has a username and password. Whenever a user wants to login, he/she must be authenticated by the server. When a user logs in for the first time, a directory named “Syncer” is created in his home directory. From then, whenever a user logins, Syncer starts syncing files/folders with the server in the Syncer directory. Any files or folders placed in that directory is uploaded to the server. Whenever the client program detects any change, it notifies the server. The server then updates files accordingly and notifies other users who have access to these files. Thus the server file contents is always in sync with the client. A user can install the client program on multiple computers. In that case when he changes a file from one computer, it is also updated in other computer.

A file can be shared publicly or with other users. A user must approve before any shared file is stored in their directories.

A user can also view which users are online now.

A history table is maintained for all files. Any changes made is recorded. Information such as last modified date, modified by, size are stored.

This system can be used by anyone. Teachers may use it to share their lectures with students. In an office where many people are working on a project can use this system for managing their project.

1.3 Definitions

Server: A server is a computer program that provides services to other computer programs (and their users) in the same or other computers [1]. Though any computer running special software can function as a server, usually server references very large, high-powered machine that functions as the pump pushing and pulling data across the internet.

Client: A client is a computer that retrieves information from or uses resources provided by the server or main computer [2].

Client and server may run on the same machine and connect via inter-process communication techniques. They may also run on different machine and communicate via socket.

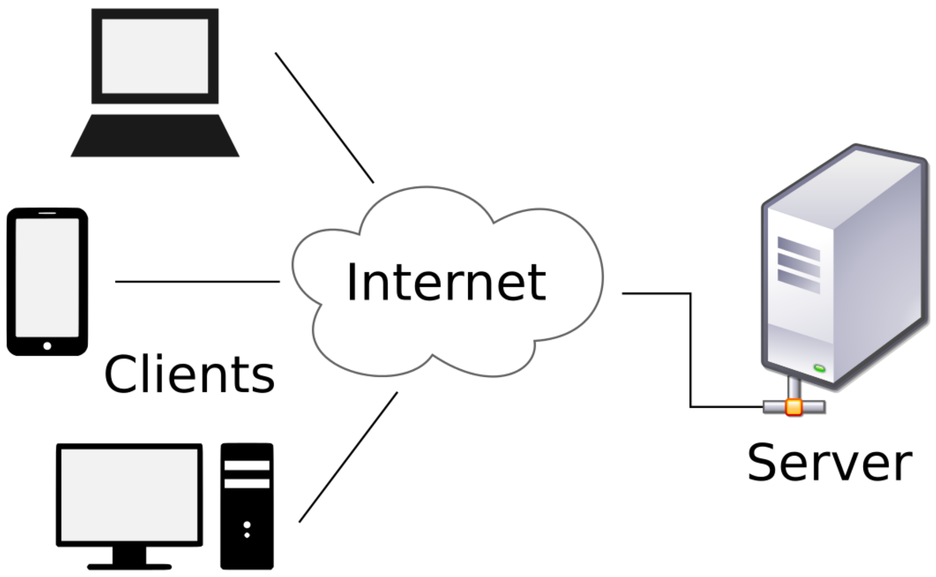


Figure 1 Client-Server Architecture

Port number: A port number is a way to identify a specific process to which an Internet or other network message is to be forwarded when it arrives at a server [3]. Port number is a 16-bit integer.

1024 well-known port numbers are reserved by convention to identify specific service types on a host.

IP address: An IP address is a binary number that uniquely identifies computers and other devices on a TCP/IP network [4]. Version 4 of the Internet Protocol (IPv4) defines an IP address as a 32-bit number. However, because of the growth of the Internet a new version of IP (IPv6), using 128 bits for the IP address has been developed.

Socket: A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to [5]. A socket is a combination of an IP address and port number.

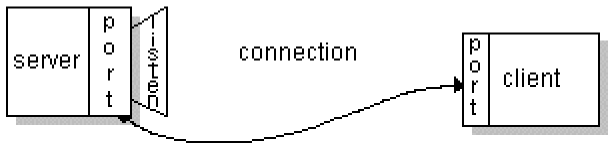


Figure 2 Socket

Synchronization: Process of making two or more data storage devices or programs in the same or different computers having exactly the same information at a given time [6].

File hosting service: A file hosting service is an Internet hosting service specifically designed to host user files. It allows users to upload files that could then be accessed over the internet from a different computer, tablet, smart phone or other networked device, by the same user or possibly by other users, after a password or other authentication is provided [7]. It can be used for various purpose.

- keeping back-up copies of file
- allowing users to create special folders on each of their computers so that it appears to be the same folder regardless of which computer is used to view it.

Chapter 2: Implementation Overview

This chapter discusses what technology was used, how the product was implemented and what the constraints were.

2.1 Technology used

Most of the source code is written in Java. A minor portion is written in Python. For the database MySQL has been used.

Some Java APIs were used to make development easier. Watchevent API was used for change detection, JavaMail API for sending mail and Swingx API for improving UI.

2.2 Scope and Assumptions

Requirements of developing a file synchronizing system is quite big. It cannot be implemented within a short timespan. So, to make things simpler we made our project scope smaller and avoided handling some exceptional and critical cases.

Scope

The scope of our project is:

I. The program will run only for Linux distributions.

II. File modification change will be only detected if it was modified by gedit.

III. Users are only allowed to create new files or modify existing in the Syncer directory while they are offline.

IV. Only files can be shared.

V. It is not possible to share file to a group.

VI. Users are not allowed to rename shared files.

VII. If a user cancel sharing a file, it will not be deleted from client.

VIII. Deleted files cannot be recovered.

Assumptions
The assumptions we have made are:

I. The underlying network is completely reliable. No communication failure will occur.

II. Users will not create/modify/delete files very rapidly or in a very short interval of time.

III. Two users will not modify a file exactly at the same time.

IV. Users will not delete or rename a file in the Syncer directory while they are offline.

V. Users will not place their files inside Shared directory.

VI. Only one user will use Syncer from a computer.

VII. File names will not contain comma (,).

2.3 Implementation Description

The program written has two portions- the server side code that is run on the server, and the client side code which is run by the users of the program. The total implementation procedure in short is as following:

I. Made a change detector that detects change made to a file.

II. Created a multi-threaded server that stores file of users and communicates with the clients.

III. Created a client program that connects with the server.

IV. Created a protocol that the server and client uses to communicate and exchange files and folders.

V. Created a database that keeps information about users and files that are exchanged and shared with other users.

VI. Created a mechanism for registration and authentication

VII. Created a mechanism for sending Emails when an admin approves a user.

VIII. Made a file compression/extraction mechanism that is used when files are transferred, thus reducing network overhead and increasing performance

2.3.1 Protocols

The server and client communicates through some messages. The message sets that client can send to a server are:

I. Registration/Authentication messages

1. "cookie", cookie number
2. "register", name, username, password, email
3. "authenticate", username, password
4. "logout"
5. "close"
6. "forgot-pass", username
7. "change-pass", new password

II. File transfer messages

1. "folder-created", path, filename
2. "file-created", path, filename
3. "file-modified", path, filename
4. "file-deleted", path, filename
5. "file-renamed", path, from filename, to filename

III. Admin messages

1. "user-approval", username, decision
2. "user-delete", username

3. “pending-users”

IV. User activity messages

1. “who-are-online”
2. “file-history”, path, filename

V. Private share messages

1. “view-pending-requests”
2. “view-my-shares”
3. “shared-with-me”
4. “private-share-add”, share with, path, filename
5. “approve-share”, shared by, file id, approval message (accept/reject)
6. “share-file-privately”, path, filename, share with

VI. Public share messages

1. “public-shares”
2. “public-share-add”, path, filename
3. “public-download”, file id

VII. Other messages

1. “cancel-share”, share type, file id, shared with
2. “initial-sync”
3. “userlist”

Depending on the messages of the client, the server respond. The server can send the following set of messages:

I. File transfer messages

1. “folder-created”, path, filename
2. “file-created”, path, filename
3. “file-modified”, path, filename
4. file-deleted”, path, filename
5. “file-renamed”, path, from filename, to filename

II. Acknowledgement messages

1. “OK” - the client request is successful/client can send the file
2. “NO” - the client request is unsuccessful/client cannot send the file
3. “DONE” - the server has completed the initial sync

Depending on a message a server or client gives a reply and performs an action.

2.4 Constraints

There were enormous challenges that we faced during the implementation phase of the project. The constraints we faced are mention below.

1. Our knowledge about server client architecture was definitely not enough to implement this project. We learned as we went. But this learning curve was very steep and time consuming. This slowed down the pace of development. Also, solutions to some problems were made by ourselves as we did not find them. At times even these solutions proved to be faulty.
2. Debugging was near to impossible. Debugging when multiple threads are running are so difficult to keep track of. On top of that, multiple clients were connected making it quite impossible to use a debugger. All debugging was done manually which was a very laborious work. We also experienced heisenbugs (heisenbug is a classification of an unusual software bug that disappears or alters its behavior when an attempt to isolate it is made [8]).

3. Managing multiple connections was a big difficulty. Socket streams often collided giving unexpected results.

4. The protocol we designed failed at some cases which gave rise to new sets of messages in the protocol. With the increase in protocol messages, things got more complicated.

5. For change detection we used a WatchEvent API that comes with Java 7. This API does not work properly for Linux distributions. And as we were working for Linux distributions it failed us. We had to create our own change detector modifying this API.

All the above problems that we faced were big impediments for our project. It slowed down the development pace to a great extent and completion of the project seemed impossible at times.

Chapter 3: Source Code Description

The program is written in object oriented style. As a result the source code is formed by many classes. The classes that were made and their methods are described below.

3.1 Server side classes

The server side program consists the following classes.

I. **Authentication:** Activities related to login events are done in this class. It's methods are-

Table 1 Authentication class

No	Method Name	Method Description
1	createCookie()	creates a unique cookie for a user
2	verifyUser()	matches the username and password for a user
3	generateRandomPassword()	generates a random password for a user when a user prompts that he has forgotten his password

II. **Registration:** Activities related to registering a user are done in this class. It's methods are-

Table 2 Registration class

No	Method Name	Method Description
1	checkIfUsernameExists()	checks during registration whether the name supplied by the user is unique or not
2	registerUser()	registers a new user who is to be approved by the admin later so that he can use syncer
3	updateUser()	it can update any attribute of a user (name, email, password)
4	notifyAdmin()	sends admin a list of registered users who are yet to be approved to use syncer

III. Server: The server class is the starting point of the server program. It mainly listens for inbound connections so that clients can connect. It's methods are-

Table 3 Server class

No	Method Name	Method Description
1	listenForConnections()	accepts connection from a client
2	synchronizeFilesWithOnlineUsers()	when a change is made to a file, this method sends the file to available online users who are supposed to receive this file
3	compareFileDates()	checks for the modification date of a file and decides which is the latest
4	updateSessionTables()	this method keeps track of users who are online currently and updates the database accordingly
5	sendToOwners()	Sync files with owners who are online
6	sendToSharedUsers()	Sync files with shared users who are online
7	run()	listening for connections is required to run in a thread, so the server class had to implement the Runnable interface so that it could be run in a thread

IV. Database: The wrapper class for CRUD operations (create, retrieve, update, delete) on the database. It has the following methods.

Table 4 Database class

No	Method Name	Method Description
1	addUser()	adds a new user in the database

2	removeUser()	removes an existing user from the database
3	updateUser()	updates an attribute of a user
4	retrievePendingUsers()	returns a list of users who are not yet approved by the admin
5	addSession()	creates a login session for a user
6	deleteSession()	deletes a login session for a user
7	checkIIfSesssionExists()	checks if a session is still valid for a user
8	updateLastOnline()	updates the last online time of a user
9	checkIfFilesExists()	checks if a file sent by a user already exists on the server
10	addFileEntry()	adds information about a newly created file in the database
11	deleteFileEntry()	deletes info about a file
12	updateFileEntry()	updates attributes of a file when they are changed
13	updateForFolderRename()	this method is called when a folder is renamed and thus the path all the files and folders under this folder needs to be renamed
14	addHistory()	records the modification event that occurred on a file
15	retrieveFileHistory()	gets the edit history of a file
16	addShare()	when a user shares a file this method is called to do it
17	deleteShare()	deletes a share of a user
18	updateShare()	called when a user with whom the file has been shared approves the share
19	retrieveSharedWithMe()	returns a list of files shared with a user
20	addPublicShare()	makes a file available publicly for download

21	removePublicShare()	makes a file unavailable for downloading by public users
22	retrievePublicShare()	returns a list of file publicly shared by a user
23	retrieveMyShares()	returns a list of file shared by a user
24	getUsernamesForSync()	returns a list of users with whom a file should be synced
25	getUsernames()	retrieves all approved users
26	matchCredentials()	matches the username and password of a user
27	retrieveOnlineUsers()	returns a list of users that are currently online
28	checkIfUserExists()	checks if a username is unique or not
29	getUsernameFromCookie()	gets the username corresponding a unique cookie
30	getFileIdFromPath()	gets the unique file id of a file from its path
31	getPathAndFileFromFileId()	returns id of a file from its path and name
32	getEmail()	retrieves a user's email address
33	initializeConnections()	gets the connection to the database
34	closeConnection()	closes the connection to the database

V. **Email:** This class sends 2 types of mails.

1. Confirmation/ rejection mail
2. Forgot password mail

Table 5 Email class

No	Method Name	Method Description
1	sendEmailToClient()	sends client a confirmation email. It composes mail using MimeMessage class and sends mail through Transport class.

3.2 Client side classes

The client side program consists the following classes.

I. **User:** This class uses ConnectionHandler class to bring information from the server.

Table 6 User class

No	Method Name	Method Description
1	login()	logs in the user
2	logout()	logs out the user
3	viewFileHistory()	gets the edit history of a particular file
4	viewOnlineUsers()	returns a list of online user.
5	viewPublicDocuments()	view public files that are available for download
6	viewSharedFiles()	view the files that a user has shared with othersr
7	changePassword()	does the necessary procedures to change the password of a user
8	downloadPublicFiles()	downloads a publicly available document
9	receiveFile()	downloads a file for user
10	viewFileSharingRequests()	shows the file sharing requests that others have made to a user
11	approveFileSharingRequest()	approve the file sharing request
12	cancelSharing()	unshare a shared file

13	forgotPassword()	creates a random password for a user and emails him
14	shareFile()	shares a file with a user
15	getUserList()	provide suggestion to user for sharing

II. **Admin**: This class extends User.

Table 7 Admin class

No	Method Name	Method Description
1	receiveNotification()	gets the list of users who are not approved yet
2	approveAccount()	verify a user and allow him to use syncer
3	removeUser()	remove a user from the system

III. **ClientProgram**: This class manages the client side program.

Table 8 ClientProgram class

No	Method Name	Method Description
1	initializeEnvironment()	initiates the InitialSync and ChangeDetector class
2	retrieveCookie()	retrieves cookie (if exists)
3	connectWithServer()	connects with the server using ConnectionHandler class
4	validateFieldsAndSend()	validates registration form fields and submits the form if valid
5	createDirectory()	creates Syncer, Shared and Public directories

IV. ChangeDetector: This class is responsible for detecting changes.

Table 9 ChangeDetector class

No	Method Name	Method Description
1	run()	this class implements the Runnable interface so that it can be run in a thread
2	detectChange()	main method for detecting a change
3	onFileCreate()	whenever a file/ folder is created, this method is called to handle the event
4	onFileDelete()	whenever a file/ folder is deleted, this method is called to handle the event
5	onFileRename()	whenever a file/ folder is renamed, this method is called to handle the event
6	addThread()	adds a new thread to a directory that requires listening to changes
7	startThread()	starts an added thread
8	stopThread()	stops a thread when a directory is deleted or renamed or program is stopped

V. ChangeHandler: This class queues the detected changes and sends it to the server.

Table 10 ChangeHandler class

No	Method Name	Method Description
1	run()	this class implements the Runnable interface so that it can be run in a thread
2	addEvent()	add a new event to the queue that was detected
3	handleFolder()	searches recursively for files in a directory

4	getModifiedPath()	transforms the absolute path to the relative path before sending to the server
5	notifyChange()	send server the message about the change detected

VI. InitialSync: Whenever the program is executed, this class performs the initial synchronization.

Table 11 InitialSync class

No	Method Name	Method Description
1	synchronizeFiles()	synchronize files with the server when the client logs in
2	getModifiedPath()	transforms the absolute path to the relative path before sending to the server
3	getFiles()	returns all file stored inside Syncer directory

Classes for Graphical User Interface

Our project has the following classes for graphical user interface:

1. **AccountRequestPanel:** Displays pending account requests to admin.
2. **ChangePasswordPanel:** Takes new password as input from user.
3. **CustomizedPanelForAccountRequest:** Used for organizing pending account requests.
4. **CustomizedPanelForOnlineUser:** Used for better presentation of online username.
5. **CustomizedPanelForShare:** User for organizing shared files to improve user experience.
6. **CustomizedPanelForSharingRequest:** User for organizing pending sharing requests.
7. **CustomizedPanelForUserManagement:** User for organizing existing username.
8. **ForgotPasswordPanel:** Takes username as input from user.
9. **LoginPanel:** Takes username and password as input from user.
10. **MyFilesPanel:** Displays files inside Syncer directory.

11. MySharingsPanel: Displays files shared by user (for both private sharing and public sharing).

12. OnlineUserPanel: Displays online users.

13. PublicSharePanel: Displays publicly available files.

14. RegistrationPanel: Takes name, username, password, email address as input.

15. SharedWithMePanel: Displays files shared with a user.

16. ShareRequestPanel: Displays pending sharing requests.

17. UserManagementPanel: Displays existing users to admin.

18. UserSelector: Used for providing suggestion to user at the time of sharing file.

3.3 Common classes

Apart from the above classes mentioned, there are classes that were needed in both the server and client side. These classes are-

I. **ConnectionHandler:** This class is responsible for the transfer of messages and files between the server and the client. It has the following methods.

Table 12 ConnectionHandler class (Client side)

No	Method Name	Method Description
1	sendMessage()	sends a message to the connected computer
2	sendReply()	sends reply to the connected computer
3	receiveMessage()	receives message from the connected computer
4	sendFile()	sends a file
5	receiveFileThroughServer()	receives a file through server socket
6	receiveFileThroughClient()	receives a file through client socket
7	receiveList()	receives list sent by server
8	run()	message receiving was needed to be run in a thread. Thus this class had to implement the runnable interface

9	isConnected()	checks if the connection is open or not
10	receiveReply()	when an event is triggered the computer connected is sent a message. The sender then waits for the receiver for a message on what to do about thie event
11	deleteFiles()	deletes file/folder

Table 13 ConnectionHandler class (Server side)

No	Method Name	Method Description
1	sendMessage()	sends a message to the connected computer
2	sendReply()	sends reply to the connected computer
3	receiveMessage()	receives message from the connected computer
4	getFiles()	retrieves all file of a client for initial syncing
5	sendFileThroughServer()	sends a file through server socket
6	sendFileThroughClient()	sends a file through client socket
7	receiveFile()	receives a file
8	run()	message receiving was needed to be run in a thread. Thus this class had to implement the runnable interface
9	isConnected()	checks if the connection is open or not
11	deleteFiles()	deletes file/folder
12	adminFileDelete()	deletes file of a user removed by admin

II. **Zipper:** This class is responsible for compressing a file before sending and uncompressing the file after receiving. It has the following methods.

Table 14 Zipper class

No	Method Name	Method Description
1	zipFile()	zips a file, given it’s path
2	unzipFile()	unzips the contents of a zipped file

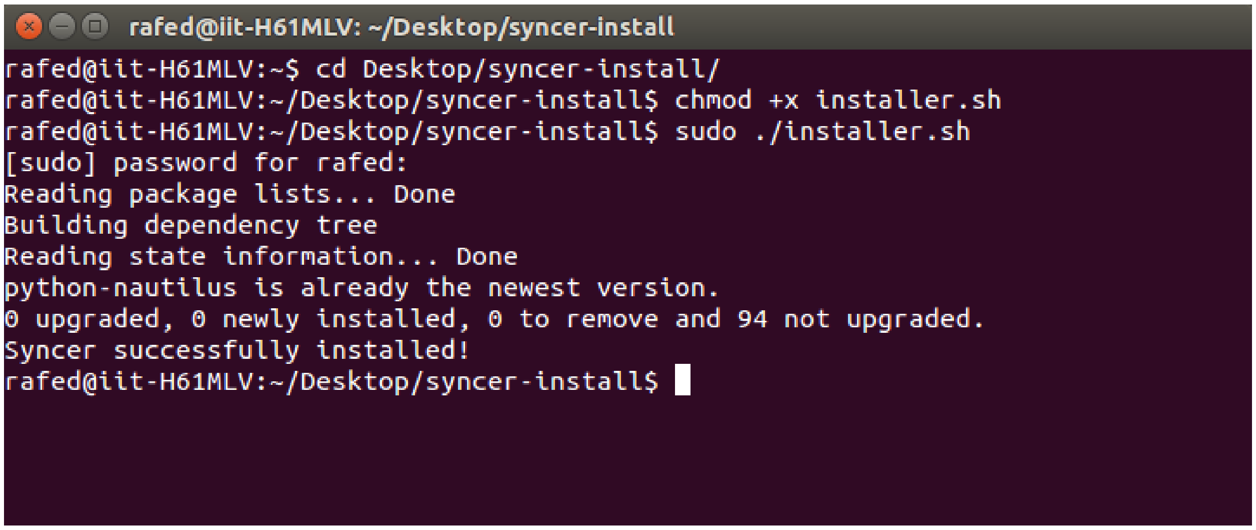
III. Utils: Contains some miscellaneous methods needed throughout the program.

Table 15 Utils class

No	Method Name	Method Description
1	hashPassword()	returns the MD5 hash of a text
2	randomString()	returns a string of random characters; the length of the string can be passed as a parameter

Chapter 4: User Manual

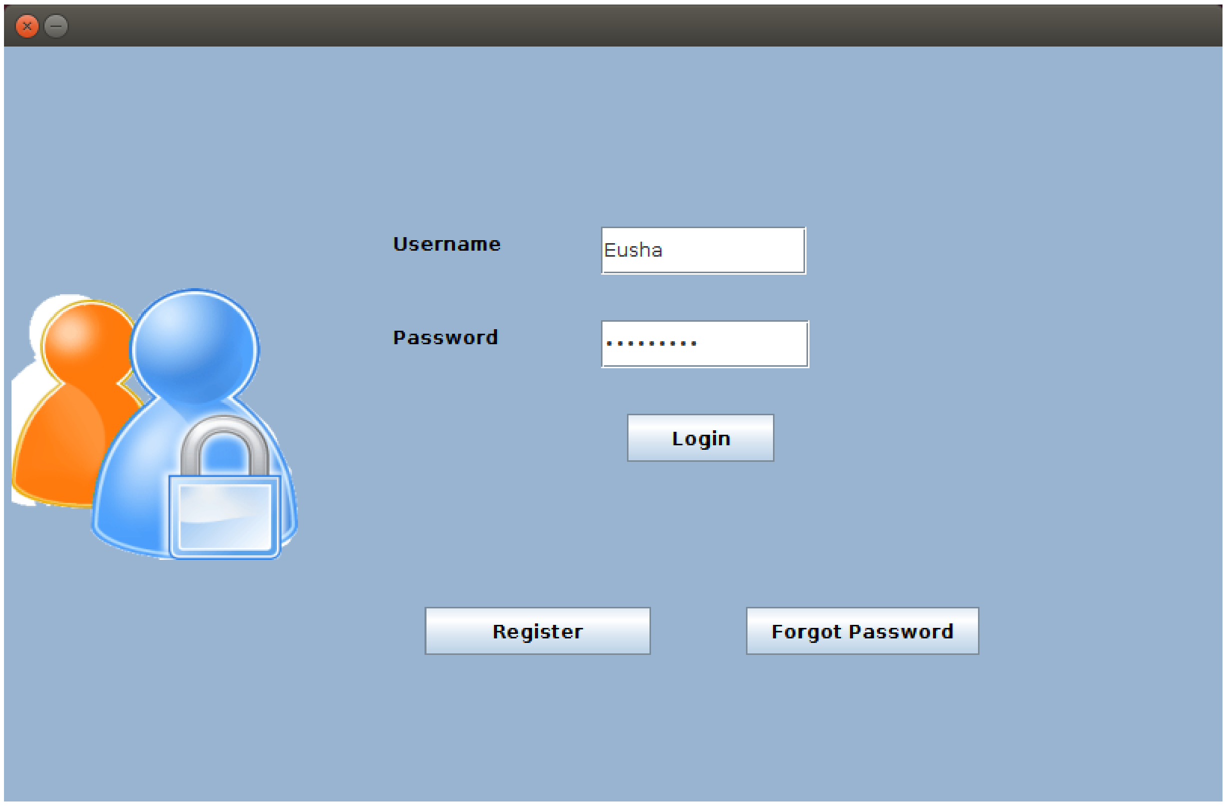
1. At first navigate to the syncer-installer directory and run the installer script as sudo.



```
rafed@iit-H61MLV: ~/Desktop/syncer-install
rafed@iit-H61MLV:~$ cd Desktop/syncer-install/
rafed@iit-H61MLV:~/Desktop/syncer-install$ chmod +x installer.sh
rafed@iit-H61MLV:~/Desktop/syncer-install$ sudo ./installer.sh
[sudo] password for rafed:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-nautilus is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 94 not upgraded.
Syncer successfully installed!
rafed@iit-H61MLV:~/Desktop/syncer-install$
```

Figure 3 Syncer Installer

2. Run the jar and you will see a login panel.

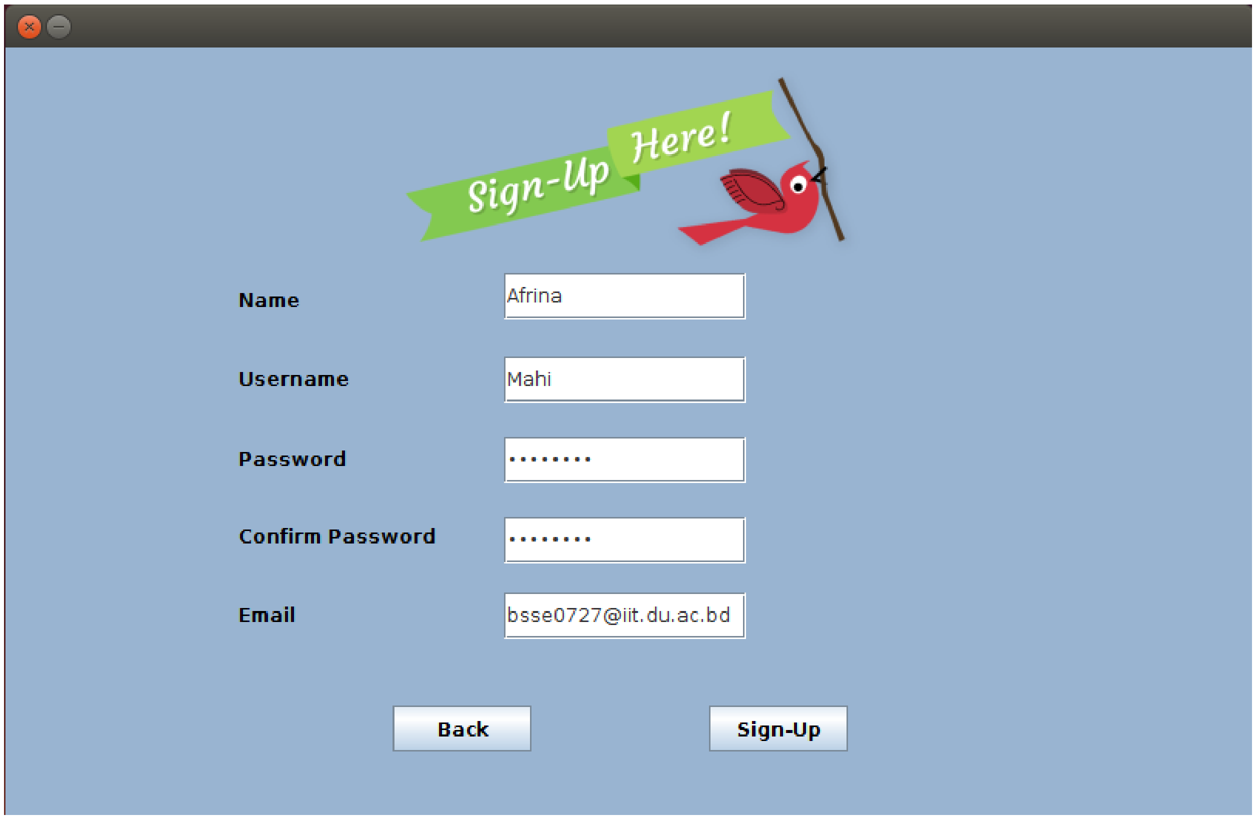


Username: Eusha

Password: ·········

Figure 4 Login Panel

3. If no user account has been created then register for one.



Name	Afrina
Username	Mahi
Password	••••••••
Confirm Password	••••••••
Email	bsse0727@iit.du.ac.bd

Figure 5 Registration Panel

4. When the registration is successful an email will be sent to your account. Now login using your credentials.

5. You will see a Syncer directory created in your desktop as you login. Place files or folders in it and they will be automatically uploaded.

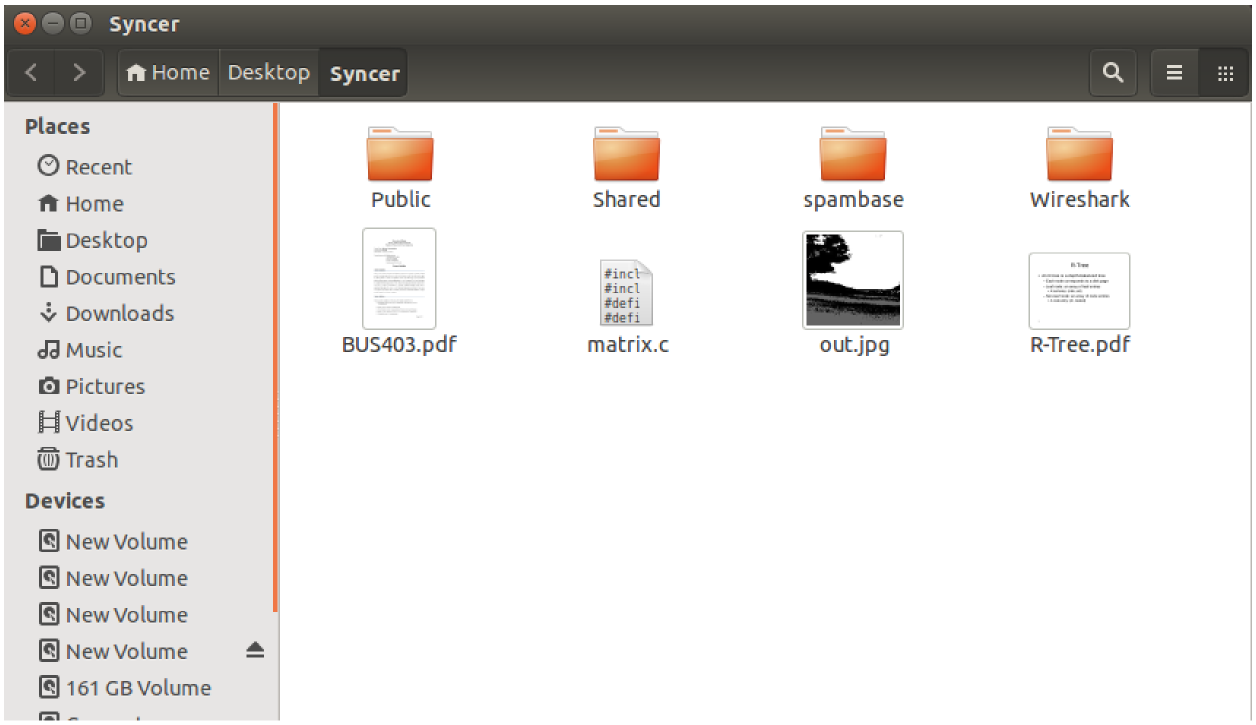


Figure 6 Syncer Directory at Desktop

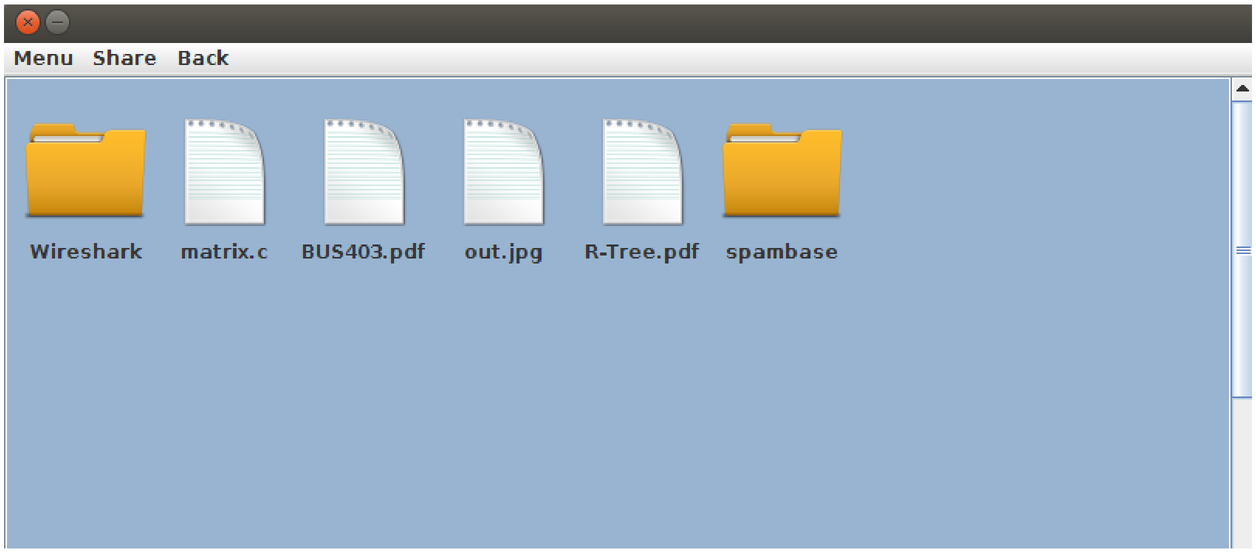


Figure 7 My Files Panel

6. You may share a file with another user. Click on “share” at the top and click on share file. Type in the username with whom you want to share. Or if you want it to share it publicly the click on “publicly share.” You can also see the history of a file.

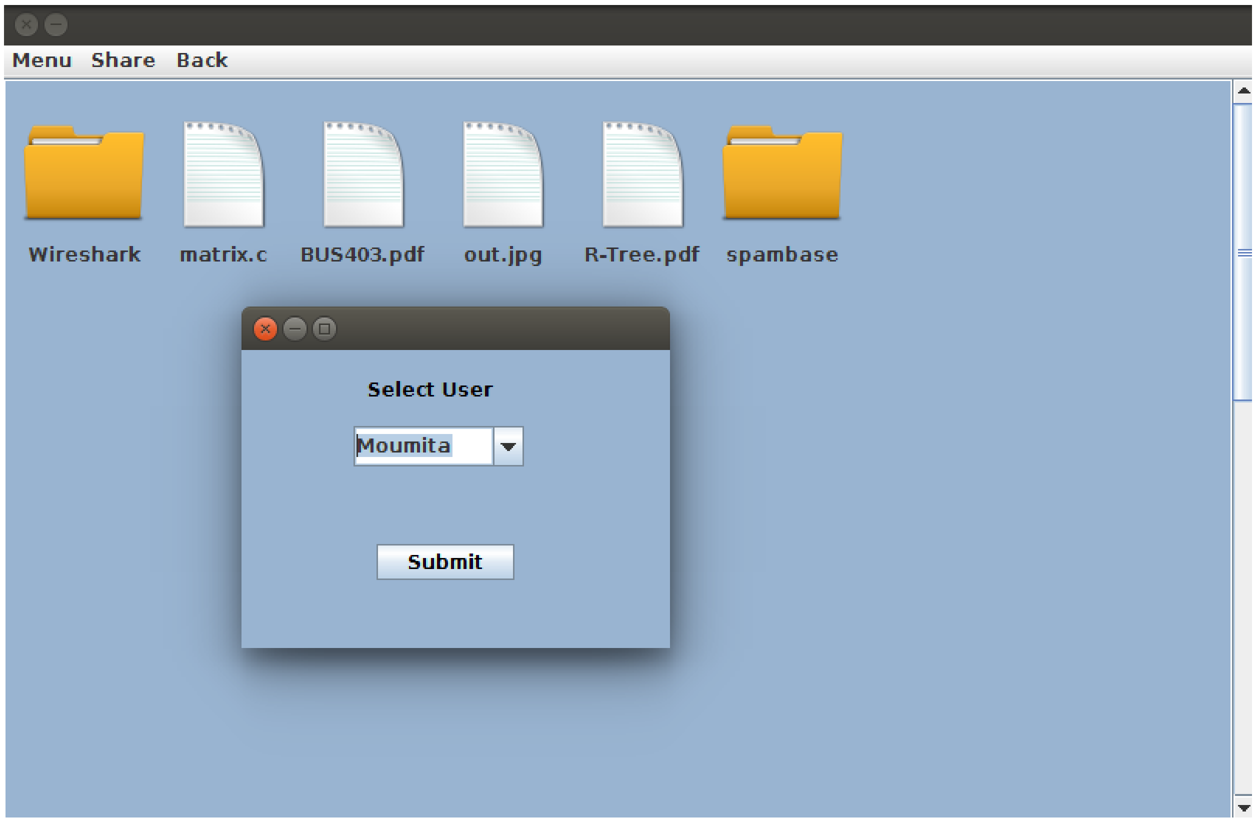
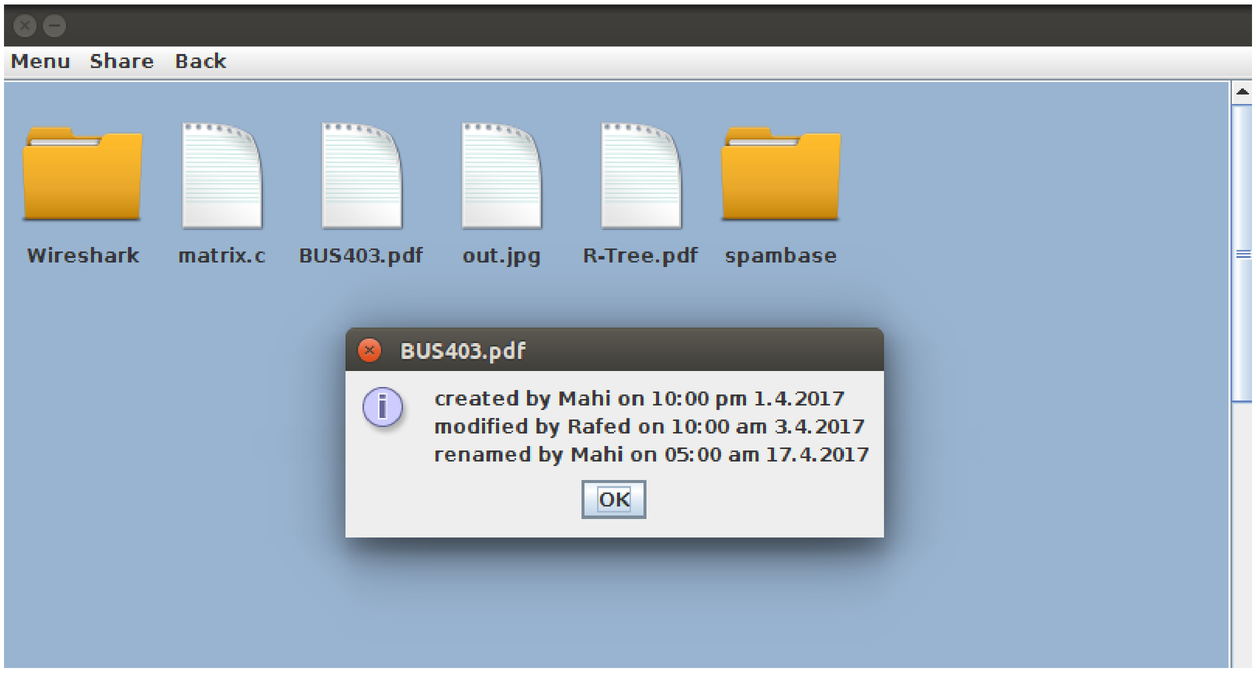


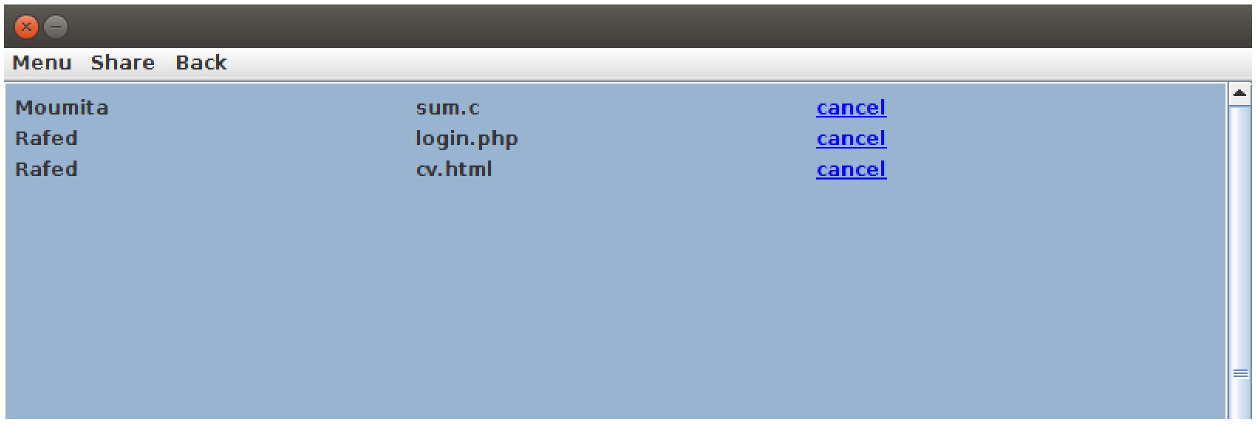
Figure 8 Sharing Files



created by Mahi on 10:00 pm 1.4.2017
modified by Rafed on 10:00 am 3.4.2017
renamed by Mahi on 05:00 am 17.4.2017

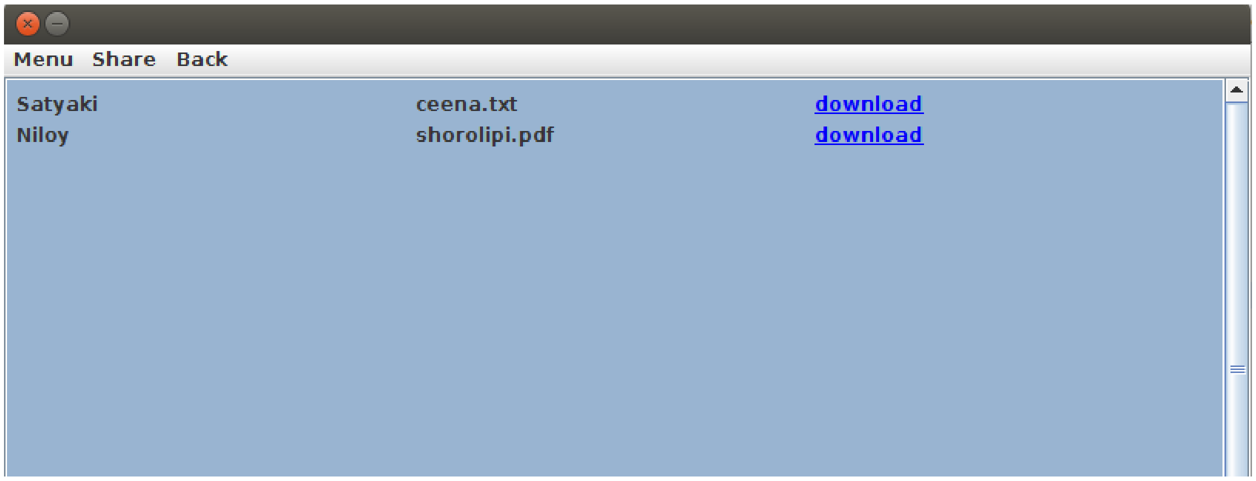
Figure 9 File History

7. You may cancel shares from the “view shared files” and “view public shares.”



Moumita	sum.c	cancel
Rafed	login.php	cancel
Rafed	cv.html	cancel

Figure 10 My Shared Files



Satyaki	ceena.txt	download
Niloy	shorolipi.pdf	download

Figure 11 Publicly Shared Files

8. In case you want to change your password, go to menu and change password.

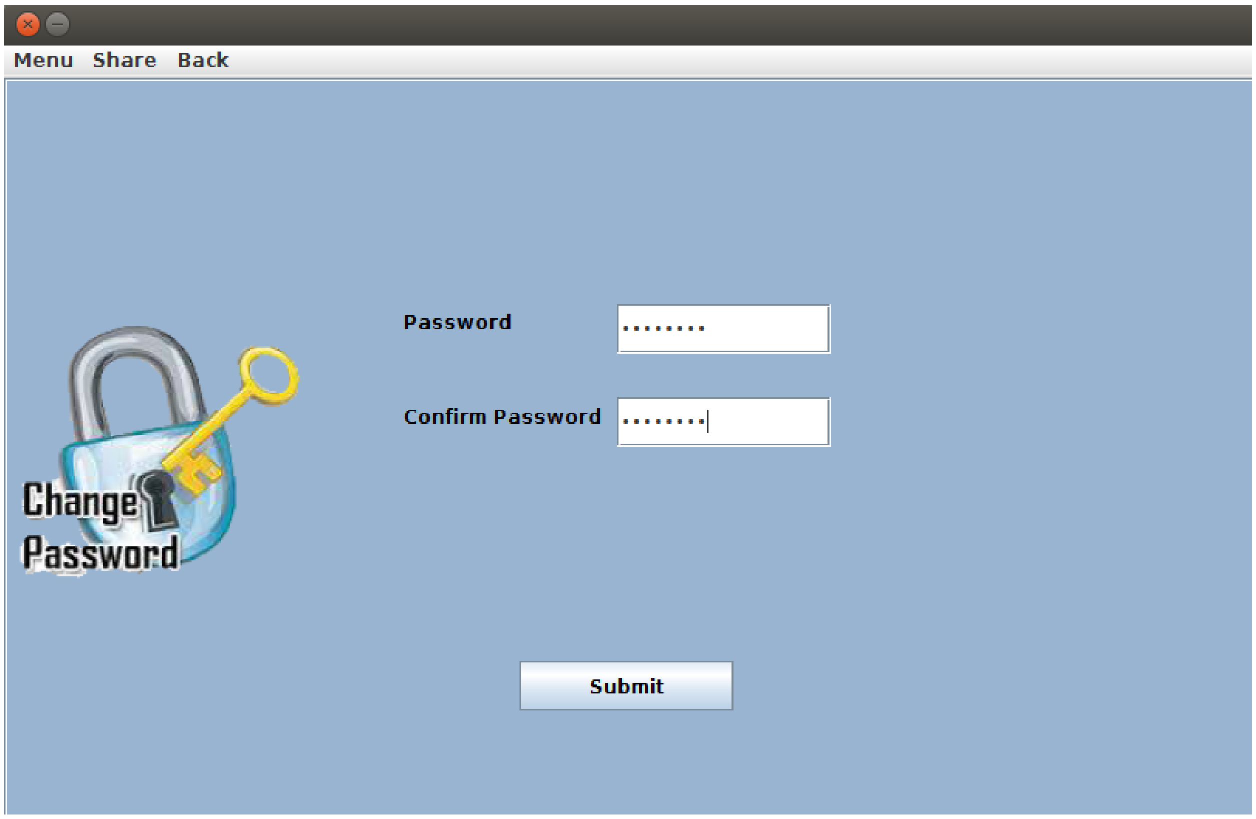


Figure 12 Change Password Panel

9. Logging out is not necessary. If not logged out, the program will automatically log in the next time you start the computer.

10. If you have forgot your password, you can get new password. Simply type your username in the input box and submit. You will receive new password through the email address you provided during registration.

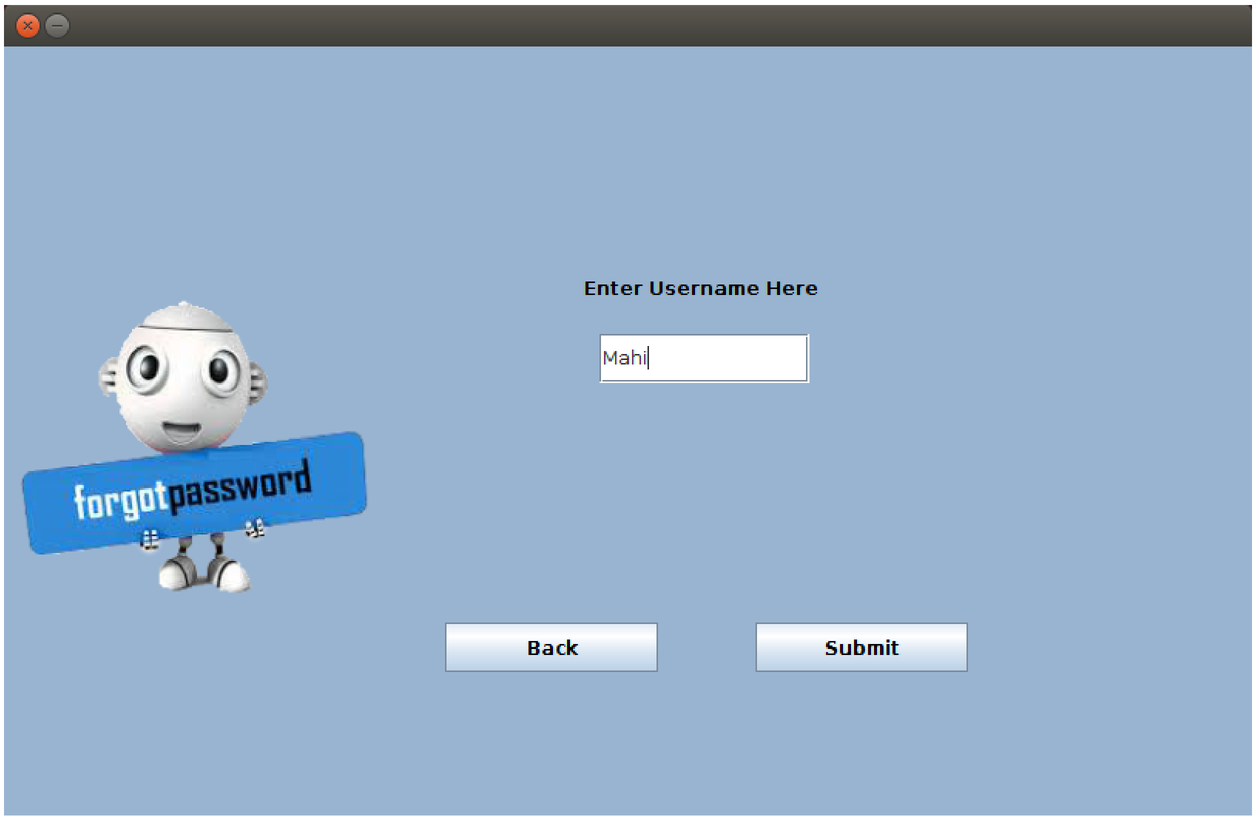
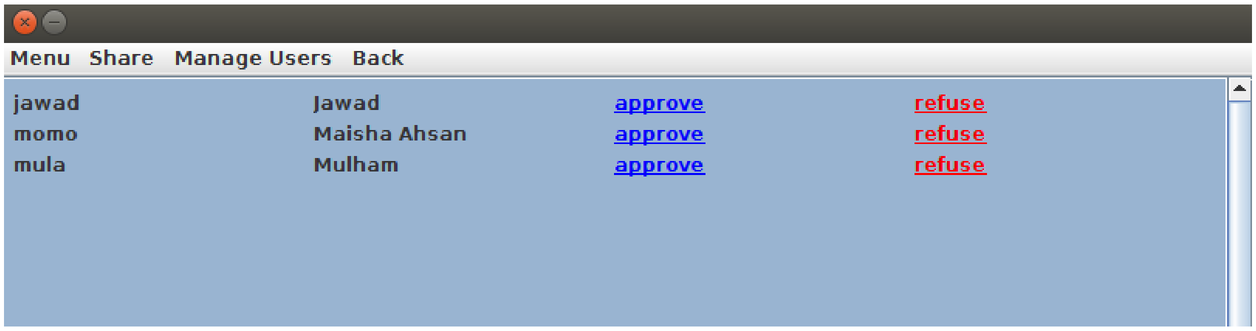


Figure 13 Forgot Password Panel

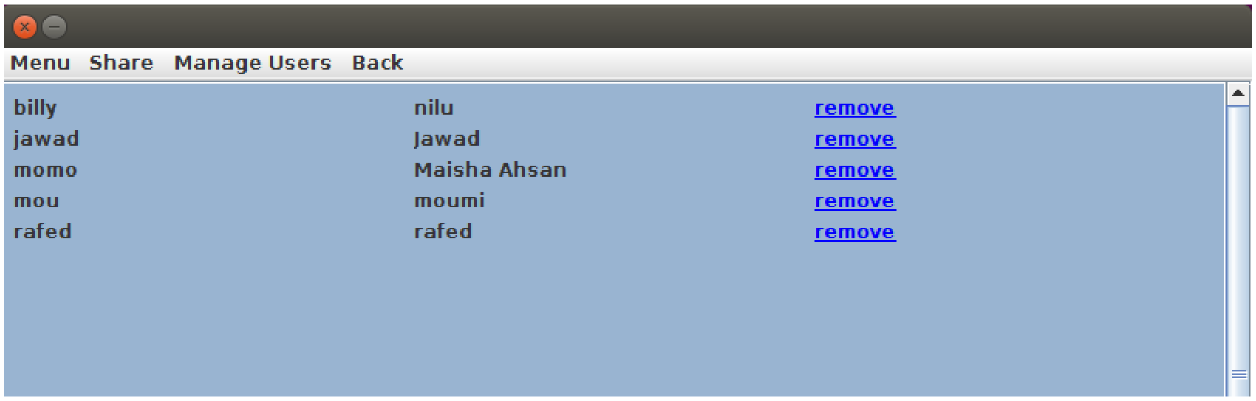
11. If you are an admin you can see which users have registered and are yet to be approved (from the Manage Users menu). Clicking on approve will allow them to use Syncer If they are refused their registration will not be complete and thus they cannot use Syncer. An automated email will be sent to them when you approve or refuse.



jawad	Jawad	approve	refuse
momo	Maisha Ahsan	approve	refuse
mula	Mulham	approve	refuse

Figure 14 Approving Account Panel

12. Admin also has the power of removing an existing user. If they are removed all their information and files will be removed from the system.



billy	nilu	remove
jawad	Jawad	remove
momo	Maisha Ahsan	remove
mou	moumi	remove
rafed	rafed	remove

Figure 15 Removing Existing Users Panel

That's it! Now you are ready to go and use Syncer!

Chapter 5: Conclusion and Implementing Limitation

The difficulty of this project was beyond our imagination. It seemed impossible at times with the given time constraint. Not only were problems occurring frequently but they were very difficult as well.

However, despite these difficulties, we have successfully completed this project. We enjoyed every moment of our work. We learned a lot about multi-threading, server client architectures, locking mechanisms and operating systems. Our programming skill with Java also increased by a fair margin. We are very happy with what we have completed considering the time limit of the project. The difficulties we faced only gave us more insight about this topic. If we are to design this sort of program again, we are sure to be better at our design and more confident than before.

We have plans to improve this project in the future. As this was a classroom project we had a defined scope and we also had some assumptions. Hopefully, we will make Syncer more complete by adding the features which we currently skipped. In the next version we have plans for handling renames for shared files, increasing performance and security and improving the graphical user interface.

References

[1] Techtarget, http://whatis.techtarget.com/definition/server, last accessed on 17.4.2017

[2] Computerhope, http://www.computerhope.com/jargon/c/client.htm, last accessed on 17.4.2017

[3] Searchnetworking, http://searchnetworking.techtarget.com/definition/port-number, last accessed on 17.4.2017

[4] Lifewire, https://www.lifewire.com/what-is-an-ip-address-818393, last accessed on 17.4.2017

[5] Oracle, https://docs.oracle.com/javase/tutorial/networking/sockets/-definition.h-tml , last accessed on 17.4.2017

[6] Businessdictionary, http://www.businessdictionary.com/definition/synchronization.html, last accessed on 17.4.2017

[7] Wikipedia, https://en.wikipedia.org/wiki/File_hosting_service, last accessed on 17.4.2017

[8] Webopedia, http://www.webopedia.com/TERM/H/heisenbug.html, last accessed on 22.4.2017