

MyDropbox

Akshay Singla 2013EE10435
Nikhil Chaturvedi 2013CS50291
Rishabh Sanghi 2013CS0168

March 3, 2015

Abstract

A C++ and QT implementation of a file hosting service that offers storage on a server, file synchronisation and personal cloud.

1 Overall Design

The implementation consists of a server which stores all the data and enables a client to login into the system with a user identity and access his/her stored files.

The Desktop client enables users to copy/upload any file into a designated folder. The client synchronises with the server and the file is then automatically uploaded to the server and made available to any other of the user's computers and devices that also have the client installed. Users may also upload files manually through the Dropbox client. The desktop client uses Qt Creator for GUI.

The Dropbox client supports synchronisation and sharing along with personal storage. Users can even collaborate on projects on the DropBox.

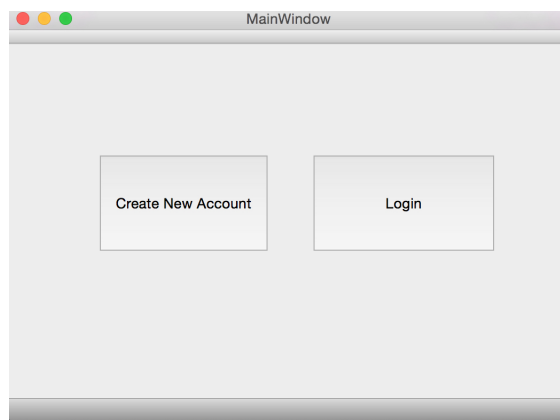


Figure 1: Main Window

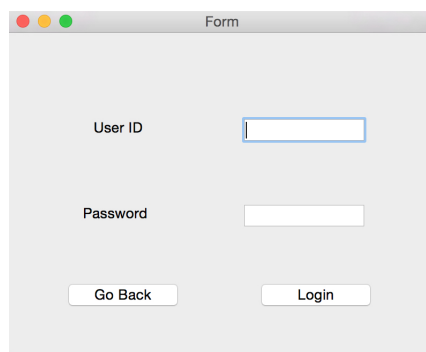


Figure 2: Login Screen

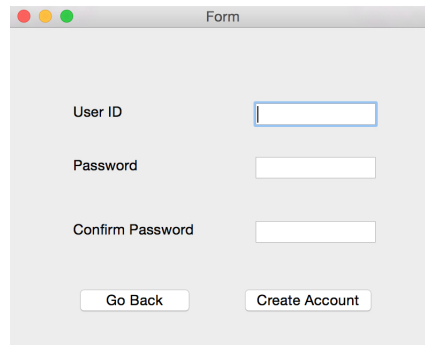
The image shows a graphical user interface window titled "Form". It has a standard macOS-style title bar with red, yellow, and green window control buttons. The form contains three text input fields arranged vertically, each with a label to its left: "User ID", "Password", and "Confirm Password". The "User ID" field is currently active, indicated by a blue border. Below these fields are two buttons: "Go Back" on the left and "Create Account" on the right. The entire form is set against a light gray background.

Figure 3: Create Account

2 Sub Components

2.1 GUI

- * The main window consists of a login option for existing users and a register option for new users.
- * As part of the registration, the user will be asked to enter a username, password and other essential information.
- * All this information is encrypted and stored in the server.
- * Post login, the user will have to enter his/her username and password.
- * After that, his whole dropbox folder will sync with the server recording any changes so that server and client are upto date.
- * The user can search for his files and also open his dropbox folder directly from the UI application.
- * The user name has the option to drag and drop files into the client or just save them in his local folder and they will be synchronised.
- * The users can also view the file hierarchy of his local machine and select multiple files and upload it.
- * The user can make changes in his local folder offline and these files will be synchronised to the server when user logs in the next time.
- * The user can upload files into a folder directly from the GUI.
- * The user can also share files to other users with the option of setting privileges , like, transfer or collaborate.
- * The user can also share files to other users with the option of setting privileges , like, transfer or collaborate.
- * The client can also search for other users in case he wants to share files with someone else.

2.2 Server

- * One computer acts as server
- * It has a database of all the users stored in encrypted text file for each user which contains his/her essential information and the unique IDs of all files stored on his account.
- * All the files are stored at the server side in compressed state.
- * Each file will have a unique ID associated with it and that ID also has sharing preferences of all files.
- * The server receives different types of signals from a client, like, to download a file from the user, to upload a file to user and if the client is logging in or registering and acts accordingly.
- * The server handles registration of new clients by making a new text file for them storing their essential data.
- * When user tries to log in, The server checks if the input password matches with the one stored on the server.
- * The server on receiving a file, creates a unique ID for the file and writes the information in the client's text file.
- * It displays all the file ID's written in the logged in user's text file to the user.
- * The server also has all the version history of all the files stored in its database and on receiving a request for a file, it decompresses the file, appends all the revisions onto it and then compresses it and sends it to the client.
- * The server on accepting a connection from a client, forks into two processes so that it is always listening to clients.

2.3 Client

- * Every user acts as a client. On registration, every user is assigned an unique username and password which will be required for all future log-ins.
- * The client chooses a folder which sync with the server whenever he logs in, if he chooses to, and the client has the option to selectively sync too.
- * The client is shown all the files on his server and the client can search for the files with a search option too.
- * The client can open a file directly from the UI application and he can set sharing privileges for all files, in case, he wishes to share a file with other users.
- * The client can also view the revision and edit history of each and every file in case he needs to go over some change.
- * The client can search for other users too and the client has the option of choosing whether other users can search for him or not.

- * The client has the option to work offline and can continue to make changes in the files on his home folder or add and delete files and the changes will be synced to the server when user logs in.
- * To upload a file, the user can simply drag a file into a predefined area in the GUI or select a file from the finder in GUI or simply copy files into his local folder.
- * All of client's personal information is stored in an encrypted format to ensure security.

2.4 Connection

The server is always online and is listening for clients. The connection is made using TCP(Transmission Control Protocol) . A socket of the type Unix Domain will be created for the Client and Server which will connect when they bind to the same port number. On connecting to a client, the server **forks** into 2 parts, one which caters to the client and the other, which checks the queue associated with the server for new incoming clients.

2.5 Security

The flow of data will be managed using a secure connection which will be implemented using OpenSSL.

- * A client attempts to connect to a server secured with SSL and it requests the server to identify itself.
- * The server sends the client a copy of its SSL Certificate which is then checked by the client to see whether it trusts it.
- * If so, it sends a message to the server .
- * The server sends back a digitally signed acknowledgement to start an SSL encrypted session.
- * Encrypted data is now shared between the client and the server.

2.6 Sharing

Each file is assigned an encrypted unique ID containing its location on the server, its location on the local folder of the user, and every user has a text file associated with him on the server which stores all the ID of files stored by the user. When a user decides to share a file with another user, that file ID is copied on the other user's text file and a marker is written inside the unique ID signifying that it's a shared file, and another marker associated by whom it was shared.

2.7 Sync and Persistent Storage

On logging in, the user's folder is automatically synchronised with the server and takes into account the offline changes made by the user. We are assuming that the client can login and use the application from different machines. On the client side, his time of last sync with the server is stored in a hidden file (on the machine) and data for when the server last synced with the client is stored in the database present on the server. There are possible cases:

- * The last sync on the server and machine from which the client logged in are the same. This means that the client last synchronised with the server using the same machine. In this case, the files' last updated time is checked with their counter parts present on the server and they are updated on the server if they had been changed on the machine after logout. In case the client deletes his file from his dropbox folder while offline, the file will also be deleted from his server in this case.
- * The last sync time of the client side is less than the last sync time on the server. This is the case where the user has logged onto the server from another machine after he logged in from the current one the last time. In this case, there is a two way transfer of files between the server and client based on whoever has the more updated file based on its last updated time. For same last updated time, no transfer of data takes place.

All files are stored on a local storage on the client's machine. In case the connection breaks, users can still work on their files. All files will be re-synchronised as soon as the connection is restored.

2.8 Minimal Data Usage

The programme implementation makes sure that minimal data is stored on the server by comparing file contents of files sent by the user to those present on the server. Not only have we implemented minimal data storage on the server, but minimal data transfer too as instead of comparing the files bit by bit, our application compares their size and sha-1 hash value which is much smaller than the actual size of the file. Even if different users store numerous files with different names, but the same content, only one instance of it will be created on the server. If one of the user updates his file, only then a new instance will be created to avoid any clashes.

3 Testing Method

The various components were tested by :

File Transfer and Connections :

- * Initially, a client-server connection was made.
- * Client connection was the connected and closed recursively to ensure server connection was not breaking and client is able to connect overtime.
- * An SSL session then established.

- * A message from the client terminal is passed onto the server.
- * A small text file was sent from the client and stored on the server in a folder and the contents and the byte size was checked.
- * This text file was then received from the server and contents and byte size were checked.
- * A video file, picture were the sent across the server and retrieved back.
- * Multiple clients were then tested simultaneously.
- * Registration system and logging in system was then checked and ensured only a person with a correct username and password is able to log in to his system.
- * Files in the server were stored and the unique ID's were written in the respected user's text file.

4 Interaction Of Sub components

4.1 GUI with Client and Server

The GUI, on opening, connects the Client with the Server and provides him the option to either login or register. Based on this choice, the information is sent to the server which then generates the info-file specific to that user or returns an error. Following the login screen, the GUI forms the connect between the client and the server assisiting him in uploading/downloading files to/from the server.

4.2 Client and Server

Client and Server interact using a basic TCP socket over an SSL secured connection. A server connection is established and it listens for the clients and on accepting a client, it **forks** into two processes and one of the process proceeds onto catering the client while the ohter process continues to listen for new clients. This ensures that a client doesn't have to wait. At a time, a server can queue upto 5 clients in the waiting list.

4.3 Client, Server and Version Control(File History)

On the server side, the last 10 difference files of the edited versions are stored in a hashed format and on accepting a request for a file from the client, the server decompresses the file, appends all the File History from the difference files onto the original file, compresses it and sends it to user. Also, to ensure minimal time and space is used, only 10 version histories are stored and if it exceeds that, then the oldest revisions are appended onto the original file to make a new original file.

The client can also view previous version by just using the UI to select the version he wants to view, which are sorted according to the edit date, and the server only appends the version history files upto that date.