Assignment 2: COP 290: Design Document

Aniket Khandelwal(2013CS10209), Anupam Khandelwal(2013CS10212),
Ronak Khandelwal(2013CS50295)

February 20, 2015

Contents

1 Abstract

We will build an online file management system "MyDropBox" in which a server machine maintains the files of multiple users. The user will use a simple desktop application to login into the system and can access his/her files after authentication. We will have two modes:

- Offline mode: In this mode, the user will transfer files using the desktop application to the drop-box folder. Access to this folder will require authentication of the user
- On-line mode: We will extend the offline mode by including a server so that the client can store and access the data which he has stored on the server after getting connected to the server

2 Overall Design

- We will start the assignment by building different subcomponents like GUI, server and client back ends, sharing and handling of files and then we will do the testing of these subcomponents
- When we are done with testing of the subcomponents, we will link them together and get the offline and the online mode working
- We will then ensure that the data handling and data transfer is done in a secure way

3 Details of the Sub Components

3.1 User Interface

We will use QT Creator for designing the "client" Graphic User Interface. We will have multiple pages in the user interface which will open according to the options chosen by the user on the current page. These windows are as follows:

- First page will ask the user whether he is a new user or an existing one and the user will have to sign up if he is a new user or will sign in to move to the login page
- Second page is the login page. In this user will have to enter the user name and password for authentication and if both matches with the information given by the user at the time of sign-up, then he will be able to login into his account
- If the user is a new user, he will have to sign up and fill the required information to create his account on "MyDropBox" and then he can click on sign in to access his files and folders associated to his account on the server and the system
- If the user has forgotten the password, then we will ask him a security question, after authentication of the user name, which the user will give us at the time of creating his account. Then we will allow him to set new user name and password
- In the final page, the user will be able to see files on the system and can do various operations on the files like deleting the files, adding new files and then finally he needs to press a offline or online sync button to store the file on the system only or both on the system and the server. Also he can share the files with the other users while setting the appropriate permissions at the the same time

3.2 Server back end

- **Authentication.cpp :** It contains the file for authentication of user during signing in , storage of user data for signup and access to password using a security question if the user forgets his password.

Listing 1: Header file of the class Authentication

```
class Authentication
{
 private:
  unordered_map<string , user> table;
 public:
  Authentication();
  void sett(unordered_map<string , user> tab);
  unordered_map<string , user> gett();
  void adduser();
  void deluser();
  void isAuthenticated(string us , string pass);
  void sec();
  void verify();
};
```

- **Storage.cpp :** Stores the files that are present in server and client side as a data structure (hash map). It allows user to add files , delete files , share files , add user and delete user. It is the backbone of file management.

Listing 2: Header file of the class Storage

```
class Storage
{
 private:
  //list of all files in server
  unordered_map<string , int> allfiles;
  //user to file list mapping
  unordered_map<string , vector<filenames> > usertofilelist;
  //for name of files in server
  int nextnum;
 public:
  //constructor
  Storage();
  //set the list of all files
  void setaf(unordered_map<string,int>);
  //get all files list
  unordered_map<string,int> getaf();
  //set the user to file list map
  void setu(unordered_map<string,vector<filenames> >
      usertofilelist);
  //get the user to file list map
  unordered_map<string , vector<filenames> > getu();
```

```
21          //whether a file with same content already exists
22          string filexists(string filename);
23          // us -> username , info -> data that user fills while
                signup
24          void sharefile(string filename, string user1, string user2,
                bool permission);
25          void adduser(string us);
26          void addfile(string filename , string us);
27          void deletefile(string file, string us);
28          void deleteuser(string us);
29        };
```

- **Sync.cpp :** It allows the user to sync his files with the files of the server. It has a function that first checks which file is newer(server's or client's) and act accordingly which side to update.

Listing 3: Header file of the class Sync

```
1       class Sync
2       {
3        private:
4          string username;
5          //list of files of user sent by client
6          string* filelist;
7          //list of old files of users
8          string* oldfilelist;
9          //equate bases of time of 2 machines (required for
               persistent storage)
10         int timequate(string clientfile , string serverfile);
11         void getuser(string user);
12       public:
13         void Diff
14         //gets the action to be done as per modified time
15         vector<int> getaction();
16         //ouputs the blocks of difference ( for minimal data usage )
17         vector<string> diffplaces(string filename1, string filename2
               );
18         //syncs the file on both client and server side as per
               action
19         void syncfile(string us , string file);
20      }
```

3.3 Networking

3.3.1 Sharing

- We will use TCP(Transmission Control Protocol) for creating connection between server and the client

- Then after setting up the connection, the user can share files with the other users using Socket Programming. Also he can set the permissions to give certain level of access to the other user. In the back end, we will have the code which will assign the file and the permissions associated to it to the user to whom it has been shared

3.3.2 Security

- For secure connection between the client and the server, we will use OpenSSL.
- OpenSSL will help us to transfer files on the network in an encrypted way, so that the data cannot be tapped while it is getting synced between client and the server.
- The client uses the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem that an encrypted and authenticated connection cannot be established. If the server has requested client authentication (an optional step in the process), the client also signs another piece of data that is unique to this process and known by both the client and server. Both the client and the server use the master secret to generate session keys which are symmetric keys, used to encrypt and decrypt information exchanged during the SSL session. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

3.4 Persistent Storage

- The user will be allowed to sync the files between the server and his own folder and also from the desktop application to his folder on the system. This will ensure that he can access the files on any machine after logging into the application from any system

3.5 File Handling

3.5.1 Minimizing Data Usage

- If the two files have exactly same content, then we will store only one of the files irrespective of whether they are uploaded by different users or the same user
- We will compare the files bit by bit and then if every bit of one file matches with the corresponding bit of the other file, then the files are exactly same. In this case, we will upload only one of two files. This will minimize the data usage.

3.5.2 Handling Changes in the files

- If the user has changed the content of some file, then we will track down the changes in that file
- After that, we will store only the unchanged version of the old file combined with the changes
- We will combine the changes and unchanged version of older file by using diff command to detect the changes and the position of the changes

4 Testing of the program

4.1 Testing of Sub Components

4.1.1 User Interface Testing

- We will first test each form of the user interface by using different possible inputs
- After this we will do the testing of the whole interface by applying different possible combinations of input and seeing that the transitions between the pages and other things work correctly

4.1.2 Testing of sharing of files

- We will first do the unit testing of the classes that are needed to share the file. For this, we will test the constructors of each class, get and set functions of these classes.
- When we are done with the testing of the classes, we will test it by sharing files of different sizes first between the system folder and the application (Offline mode) and then we will share the files between two systems and check whether the files are send correctly by running them and checking their sizes

4.1.3 Testing of syncing of files

- We will first do the testing of the classes in the same as we will do it in testing of sharing of files
- Then we will check the syncing in the offline mode by adding new files, deleting files and adding different version of the same file. We will test whether we are able to add and delete files from the desktop folder and also whether we are able to tell which of the files having different versions is newer and whether we are to appropriately deal with this case
- After this, we will test the online syncing of the files on the same parameters which we have used in the offline mode

4.1.4 Testing of file handling

We will test the file handling by:

- First, we will sync two files with exactly same content with the same account and then check whether only one file is uploaded on the server. Similarly, we will test this by syncing two files with two different accounts
- Also, we will make the changes in the file, and then we will sync it. For testing, we will check the size fo the file saved on the server which should be less than the combined size of the older and the newer version

4.2 Overall testing

After doing the unit testing of each subcomponent, we will link all the subcomponents and then we will test the whole program starting from user interface, setting connection between server and client, sharing data securely,with other user, syncing the data using the parameters which has been used during unit testing. Also we will check whether the files are handled in a proper way i.e. the data usage is optimized and the changes in the file are handled in a proper way or not. We will also check whether the user is able to access the files from the local folder without downloading the files even after the connection is lost.

5 Interaction between Sub Components

5.1 User Authentication

- The user name and password entered by the user on the desktop application will be transfered to the server. Then it will check whether the user name and the password are correct or incorrect. This is how the user credentials are verified and the user is allowed to login to the system if both the user name and password are verified
- Also the data of the new user is linked to the server back end which stores the information about the new user

5.2 File transfer

- All the elements in the GUI that are entered by the user are transfered to the server back end with the help of file transfer from client to server PC. Also, the results which are evaluated in the server back end are transfered to the user interface. All this is done with the help of socket programming

5.3 File syncing and sharing

- The files that are synced by the authenticated user are transferred from the desktop application to the server and from the server to the folder in the system
- Also, the permissions set by the user for particular files are send to the server where the variables are assigned these permissions and are shared with other users according to the permission as specified by the user

5.4 Front end and Back end

- All the information filled by the user in the forms are send to the back end, where the server works on these information. It stores these information and verifies when they are again needed by the software application. After that, they are send to the front end according to which it takes action
- Files on the client and the server system are displayed on the window in the desktop application. The user can sync the files between the server and client system. This is done by sending the information from the desktop application to the server which is then processed and according to it, files are transfered from server to client and client to server

6 Extra Features

6.1 Encryption of User's Credentials

- We will store the information of the user (username and password) after encryption
- We will do it using RSA encryption algorithm
- This will ensure the safety of user's credentials as even if someone gets the access to the data on the server, he will not be able to decrypt it

6.2 Incomplete Downloading

- We will ensure that if the connection of the client with the server breaks, then he will not need to download the whole things again
- We will do it by storing a variable which indicates whether the file is synced or not, and another variable which will indicate the size of the file that has been synced.
- After this, when the client is again connected to the server, the download will start again from the file which is partially synced

6.3 Taking care of the direction of syncing

- Our code at the back end will ensure that the syncing process goes smooth. If there are two different versions of the same file on the system and the server, then we will check which one of the two files (the file on the server or the file on the system) is newer.
- We will select the newer file by taking the difference of the system time of the client and the server(say x). We will then take the difference between the value of x and the modified time of the one file. This is the final modified time of that file. Then the one with the higher value of modified time is the newer file. Then we will warn the user about the file which is newer and ask him to take appropriate action to deal with the file

6.4 Privacy of user's folder on client system

- We will protect the drop-box folder of the user with the password he uses to login to his account. This will ensure that even if many users use the same system, the data of one user will not be accessible by other users who use the same system