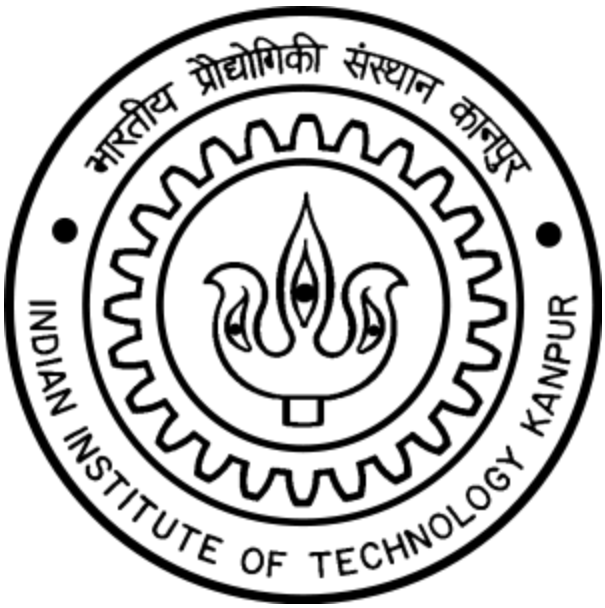
**CS455 - Software Engineering**
**IIT Kanpur**

**Project Report**

# GroupShare




**Submitted By:-**
**1} Devashish Kumar Yadav (13240)**
**2} Sourav Anand(13709)**

## Abstract and project idea:

Traditionally there have been various ways of sharing files with other people over the network and within an intranet. The simplest of the method includes uploading to an outside cloud storage and downloading from it. This requires an internet connection and consumes the essential bandwidth. Other method includes running a central server within the intranet (such as DC++) which keep tracks of the users and their shared files and serve as a point of information for each client. This requires a dependency on a central server (hub) and if the server goes offline the connection is lost and hence file sharing is affected. One another method is running a simple client server architecture like python SimpleHTTPServer but it very basic and lack rich features.

We attempt to create a robust peer to peer network based application “GroupShare” which doesn’t require a central server, is tolerant to peers connection and disconnection and provides additional functionality like sharing in groups, secure sharing without revealing directory path to peers and dynamically adding/removing peers and shared files.

## Architecture

The application is based on client-client interactions. There is no central server required. As seen from the process view below, each peer have several components such as Flask microframework, sqlite DB, file system and browser. The main component of a peer is flask microframework. This contains the main logic of the application. Flask microarchitecture can be further classified into 3 parts, Server, Client App Module and Inter App Communication module. The role of server is to interact with the server of the other peer to exchange the important data such as file list, availability of the server, connected peers etc. Also, server is responsible for transfer of file. The file requested by a peer is sent by the server, which is then downloaded by the browser of the peer requesting the file. The Client App Module contains the logic and important things required for execution of the application. It is connected the the sqlite database and the file system for retrieval of files/group data. The client app module is also connected to the inter app communication module which contains the logic for updating of file list, removal of client etc.

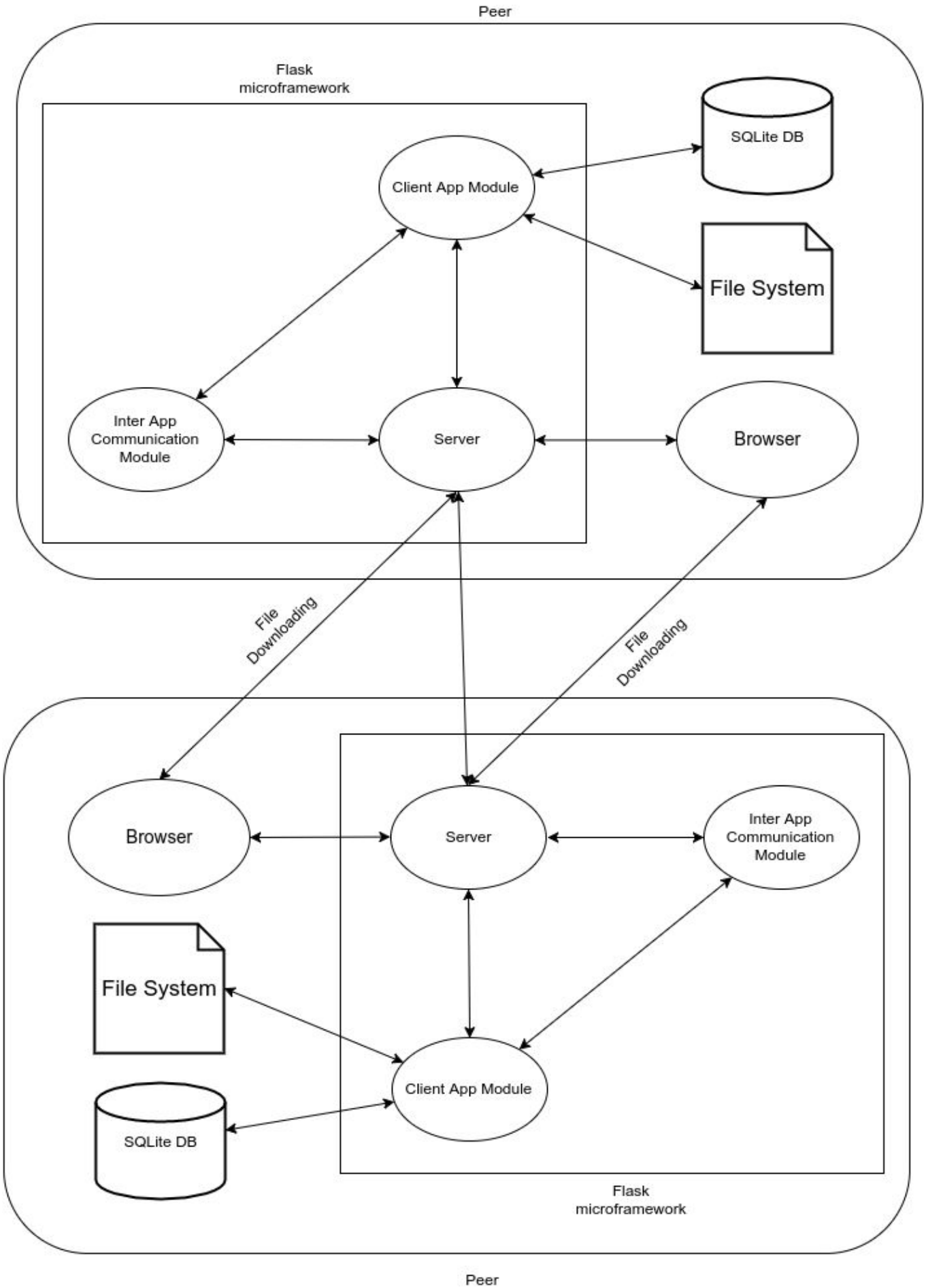



Fig: Process View of the application

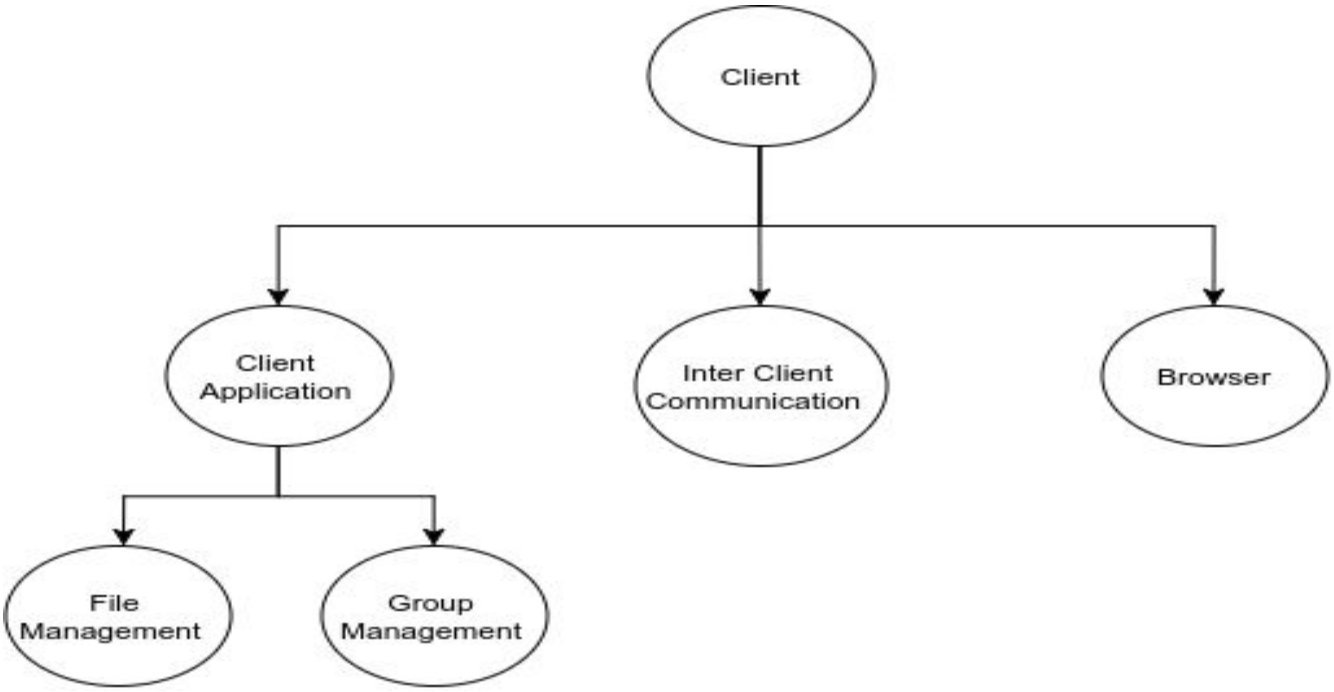



Fig: Logical View of the application

## Working of the Application:

The steps involved in the working of the application are:

1. A peer starts a group (let us call this a leader peer) and shares his ip:port and group name with the group he want to share files with (through mediums such as facebook, mail etc)
2. A new peer wants to connect to the group. It does this by requesting to join the group using the ip:port of the leader and the group name. The leader accepts the connection and both the peers exchange their list of shared files.
3. Now, when a new peer wants to join the group, he can do so by connecting to the ip:port of any of the two peers in the group. When he connects to one of the peer, he is provided with the connection information for connecting to the other peer. Now, the new peer uses this information to connect to the other peer.
4. The group keeps on expanding in this way. At any arbitrary point of time, a peer can connect to any one active member of the group and that member replies with the data of all other active members. This way, a new peer is able to connect to all the peers in the group by having knowledge of just one peer.
5. When a peer leaves the group (i.e. closes the application), other peers identify this disconnected peer and removes him from their list of active peers.

## Functionalities

The Key functionalities of the application are:

- Once a group is formed, it can remain existent till every peer from the group leaves.
- The group organically increases independent of who started the group. An incoming peer need not to connect to the leader(i.e the peer who started the group). It can send connection request to any of the peer in the group and can become a part of the whole group.
- On disconnection of a Peer from the group, that Peer is removed from the connected Peers list of each client. The application keeps checking the connected peers at fixed intervals (say 15s) and if any peer is found to be unavailable, that peer is removed
- On updation of file list by a peer, all other peers of the group are notified by that peer that he/she is updating the shared files list. The peers in the group, on receiving the notification, send a request to the peer to get the updated file list
- The application keeps track of groups of a peer and hence, each instance of the application is able to support more than one group
- The application have security measures in place which allows the peer to download only the file shared in the connected group. Also, to protect the privacy of the the users, only filename and a token is shown to members of the group. The token is basically a hashing of the directory of the file and only the peer sharing the file has the map of the token.

## Tools and Technologies Used

We used Flask, a python microframework for developing our client side app and interface. We chose flask  as compared to developing a native interface in PyQt or JavaFx as it provided flexibility in communicating with the user interface and with other clients. The user interface and frontend logic uses HTML, JQuery and Bootstrap. The Interface is generated using jinja. The database used is sqlite3 which provides persistence and stores all user data so even if the client closes the application and restarts he/she can rejoin the groups.