# *Distributed Systems*

Amira Nabil Haiba (18)
Amr Mohamed Nasr (47)
Bishoy Nader Gendy (21)
Marc Magdi (55)
Michael Raafat Mikhail (57)

# Overview:

It was required to implement a file system with clients performing concurrent transactions while guaranteeing ACID properties and communicating with a main server. The data is partitioned and replicated on multiple replica Servers.

# Code Organization

# Main packages:

## Args:

- ClientArgs: wrapper class having directory of client.
- MasterArgs: wrapper class having Address, Port and Directory of master.
- ReplicaArgs: wrapper class having Address, Port and Directory of replica.

## Configurations:

- ClientConfigurations: wrapper class having Address, Port of master server needed by the client and a list of transactions.
- MainConfigurations: wrapper class having Address, Port, Directory, username and password of master server along with usernames, passwords and addresses of all clients.
- MasterConfigurations:wrapper class having Address of replica servers.

## Data:

- FileContent: wrapper file having the file name and list of file data
- FileMeta: class having a file name, all locations of it's copies on different replicas and it's main replica location.
- ReplicaLoc: wrapper class having Address, Port, ID of a replica and a boolean to indicate whether a replica is still alive or not.
- RequestType: Enum to indicate the request type whether read, write, commit, abort or begin.
- Transaction: wrapper class having a list of requests for a transaction.
- TransactionMsg: a message having transactionID, timeStamp, replica location and list of replicas.

## Exceptions:

- MessageNotFoundException

## Main

- Main: a main class that creates the Master server and the clients.

Request

- BeginTransactionRequest: wrapper class having file name and transaction number.
- CommitRequest: wrapper class having file name and transaction number.
- AbortRequest: wrapper class having file name and transaction number.
- ReadRequest: wrapper class having file name and transaction number.
- WriteRequest: wrapper class having file name, transaction number and data to be written.

RMI

- Client: each client tastransactions with each has a read, write and commit or abort requests.
- Master: check whether a replica is alive or not, manages the communication between the replicas and the client
- Replica: reads or writes to a file, commits or aborts a transaction, taking into consideration the concurrency control over a file edited by multiple clients.

Utils

- Entry: a data struction that has a replica and transaction id.
- FilesMetaManager
- ReplicasLocManager
- RMIUtils
- SSHConnection

# Main Functions:

- ReplicaServerClientInterface :
    - Read: reads a file
    - Write: writes intro a file
    - Commit: commits a transaction
    - Abort: aborts a transaction
- ReplicaServerMasterInterface:
    - isAlive: check whether a replica is alive or not
- ReplicaServerInterface:
    - Update_replicas: apply changes to all copies of a file
- MasterServerClientInterface:
    - Request_transaction: requests to start a transaction on a file

## How to start:

- We should run the main of the project that starts master and clients, and the master create all replica servers.
- You should change username and password in master main for starting replicas using ssh connection.
- You should adjust babies.info, files_meta, main.properties and client.info for each client.

## Assumptions:

- Each client has a file that contains master address and port, also has all requests that the client asks.
- Master has a file that contains all information abouts replicas.
- Master starts all replicas.
- Each transaction has only one filename.
- All replicas run on one machine.