**Introduction:**

The design document describes program design of a bank server application, approach followed, and tradeoffs considered along with the performance impact.

**Design Approach:**

* The incremental design approach is implemented throughout the development of this application developed using socket communication and connection-oriented transport layer protocol TCP.
* Every client after successful connection with server first sends total no. of requests to the server and server sends its acknowledgement.
* Initially the client sends a transaction request (either withdraw/deposit) to a server. At the server end, all required validation checks are performed like sufficient balance check before processing a transaction.
* After successful validation, server performs the transaction and sends back the acknowledgement to the client. Every single transaction communication is logged at the server and client console terminals.
* The client waits before sending the next transaction request to the server and the time interval is calculated as the subtraction of the next transaction’s timestamp and current transaction’s timestamp and then multiplied by the time step provided by the client as a argument from the terminal.
* After implementing single client and server application, an incremental design consisting of multiple clients is implemented with multithreading.
* In case of single or multiple clients for each client trying to connect to a server, a new thread is created by the server using POSIX thread library and is handed over to a handler that processes all the incoming requests initiated by that particular client.
* In case of single or multiple clients, every transaction is taken care by mutex locks to provide exclusive access to a thread performing an operation and avoid deadlock that may result when multiple clients try to perform the same transaction on a same account.
* Every client calculates average time per transaction which is logged in a log.txt file.
* Also, periodically after a time interval of 5, the server adds interest amount to each account mentioned in a Records.txt

**Steps for Execution:**

* Makefile consists of following steps for executing an application:
* In order to clean all the compiled .out files “**make clean**” cleans out files present in current directory by executing the command rm \*.out
* To compile the server and client “**make compile**” compiles as:
* *gcc server.cpp -o server.out -lpthread*
* *gcc client.cpp -o client.out*
* To run a server executable “**make runserver**” runs
* *./server.out 8087*
* Similarly, in order to run a client “**make runclient**” runs the client
* *./client.out 127.0.0.1 8087 0.8 Transactions.txt*

**Design Tradeoff:**

* Processing overhead is one of the primary concerns in socket communication. To lower this overhead, for every active incoming request, separate function handler process requests handles it.
* The handler abstraction is providing a secure layer for inter-process communication.
* Also, once the request is handed over to the handler it performs further processing by creating its own local socket so that server’s parent socket can accept further incoming requests.

**Possible improvements:**

* To reduce the communication overhead and improve the performance time we can have our own fast socket design implementation.
* Efficient buffer management can improve interoperability.
* Efficient logging can help tracking any unexpected events.