Get Familiar with FDB Source Code

Please complete the following sub-tasks (the order is just a recommendation, you could complete them in any order):

1. Download the FDB source code at https://github.com/apple/foundationdb and build it locally in your machine (either Linux or Mac; If you have Mac M1/M2, this sub-task is even more challenging). Prepare a document that lists the steps you did and the commands you executed.
2. From the source code built at Sub-task 1, launch the FDB server and use the fdbcli to connect to it. Issue simple KV operations (get, set, getrange, delete) via the fdbcli tool.
3. Use the FDB C library and write a C program (named the file basic_ops.c) to perform basic operations (get/set/getrange/delete) to the FDB server. The document for FDB C API could be found at: https://apple.github.io/foundationdb/api-c.html
4. Measure single getrange performance:
    a. Create a file single_get_range.c and use it for this sub-task
    b. Store 10k key-value pairs (key_i, val_i) in FDB
    c. Retrieve all 10k key-value pairs by executing a getrange on \x00 \xff
    d. Modify getrange to use different modes (WANT_ALL, EXACT, ITERATOR, etc..) and report the response time of each execution
5. Compare single getrange vs multiple getranges sent in parallel:
    a. Create a file single_vs_multi_ranges.c and use it for this sub-task
    b. With the 10k key-value pairs stored in sub-task 5, define 10 ranges R1, R2, ..., R10 such that executing getrange on each of those 10 ranges returns exactly 1k key-value pairs.
    c. Execute getRange() requests on these 10 ranges in parallel
    d. Repeat the execution with different streaming modes (WANT_ALL, EXACT, ITORATOR, etc..) and report the response time of each execution. How are the numbers compared with the case in sub-task 5 where we only issue one getrange on \x00 \xff ?
6. Understand read snapshot:
    a. Create a class read_snapshot.c and use it for this sub-task
    b. Store any arbitrary 4 key-value pairs in the database. Denote the keys as K1, K2, K3 and K4
    c. Start a transaction T1 and read several keys (let's say K1, K2, K3)
    d. On another thread, start a transaction T4 that updates the values of K2, K4
    e. Commit transaction T1. Would T1 be aborted? Explain why.
    f. Commit transaction T2. Would T2 be aborted? Explain why.
7. Understand transaction conflict:
    a. Create a class tx_conflict.c and use it for this sub-task
    b. Store any arbitrary 2 key-value pairs in the database. Denote the keys as K1 and K2.
    c. Start a transaction T1 to read K1 and update the value of K2
    d. Start a transaction T2 to read K2 and update the value of K1
    e. Commit T2. Would T2 be aborted? Explain why.
    f. Commit T1. Would T1 be aborted? Explain why.

Note:

- If you cannot complete sub-task 1, please install FDB server and client directly from here https://apple.github.io/foundationdb/client-design.html
- FDB stream modes: https://apple.github.io/foundationdb/api-c.html#c.FDBStreamingMode

Delivery:

- A detailed document for sub-task 1, 2 and report numbers for sub-task 4-7.
- Source code of the other sub-tasks (3, 4, 5, 6 and 7)
    - basic_ops.c
    - single_get_range.c
    - single_vs_multi_ranges.c
    - read_snapshot.c
    - tx_conflict.c