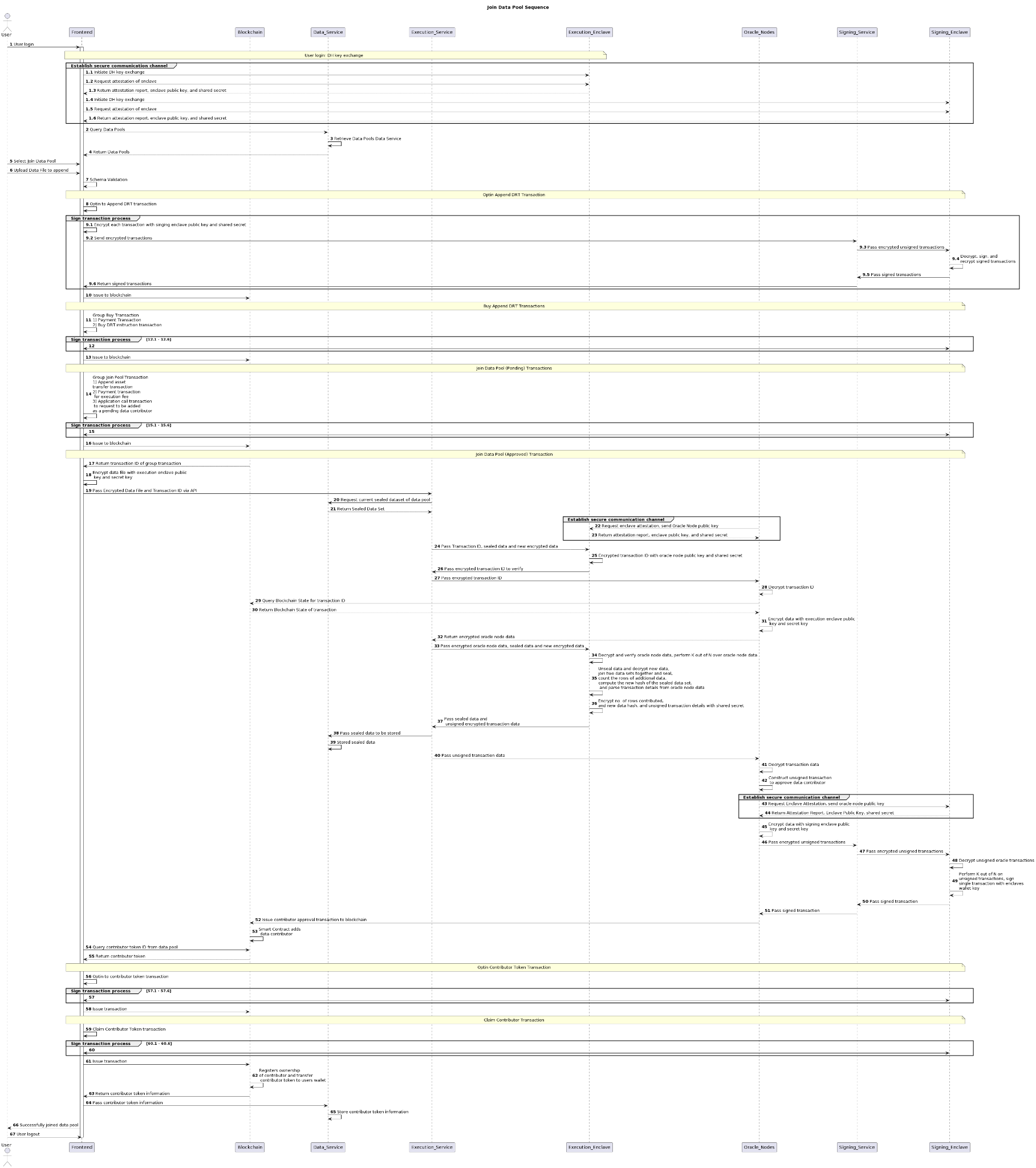
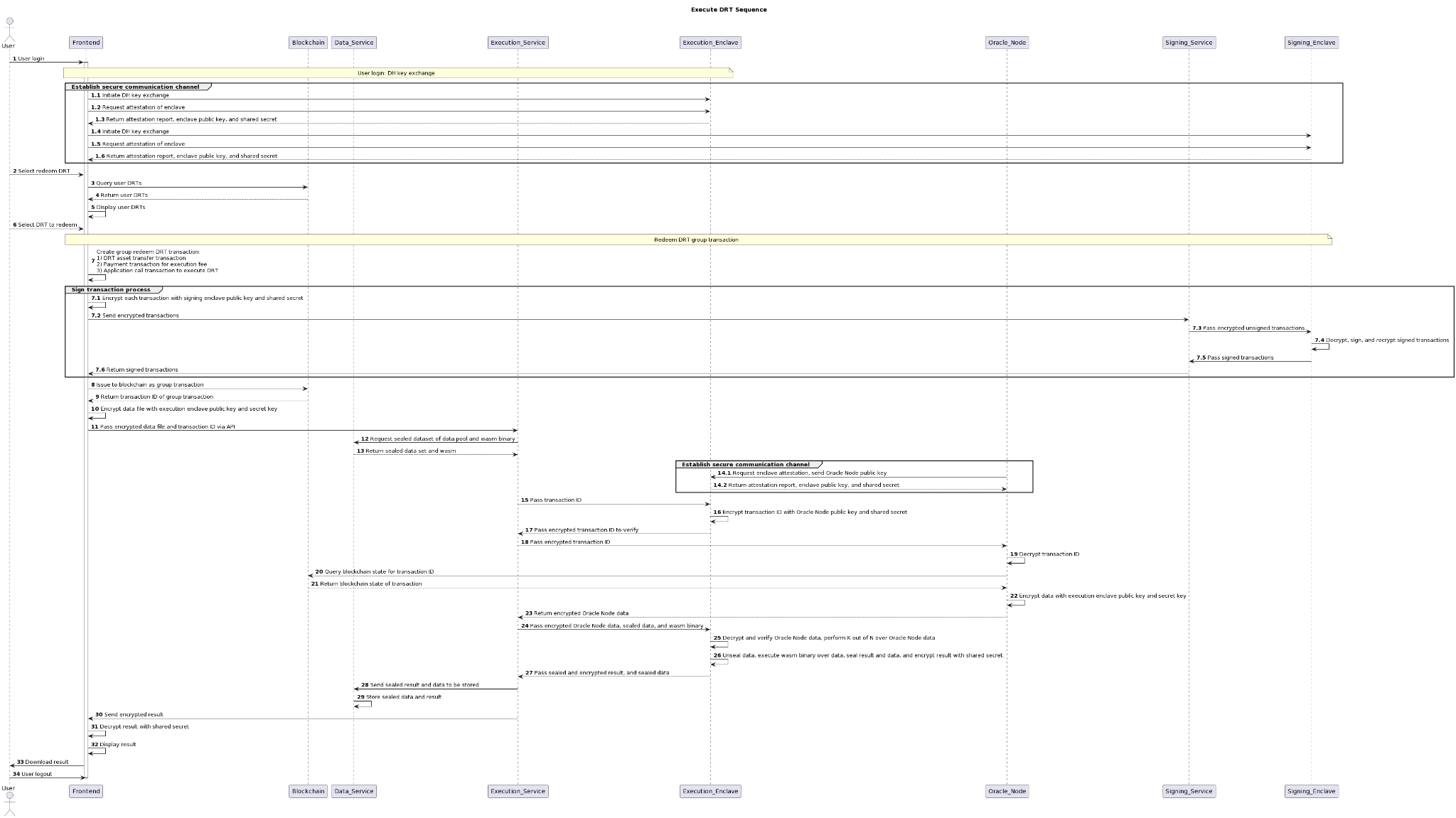
**NTC MVP: Component Analysis**

For the basis of this analysis I will use the two most complex Sequence Diagrams as they encompass all the components and their key functions. A zoomed in version of the Join data Pool sequence below is found [here.](https://drive.google.com/file/d/1W2pBaGLqkCMPSZDjjTFzpuk-zVIXmuXL/view?usp=drive_link) A zoomed in version of the Redeem DRT is found [here.](https://drive.google.com/file/d/1BEvL4AQZh3nGzIN1mhCP4gOqat2SvI0A/view?usp=drive_link)



Starting from the left on the components in the above sequence diagram. The following tasks are required for each component in order to complete the full flow of events from executing an enclave transaction and redeem DRT.

Already in use and completed

We have functionality but needs to be configured for component or waiting on component

There is some kind of implementation plan and slight development

There is minimal implementation plan

No implementation planning

## User/Frontend

{Need to add more frontend components}

1. User login  
   Establish secure user session with singing enclave only
   * Initiate DH key exchange with Execution\_Enclave
   * Request attestation of Execution\_Enclave, receive attestation report, enclave public key, and shared secret from Execution\_Enclave
   * Initiate DH key exchange with Signing\_Enclave
   * Request attestation of Signing\_Enclave, receive attestation report, enclave public key, and shared secret from Signing\_Enclave
2. Query Data Pools data from data service
3. Query DRTs from Blockchain per user
4. Select Join Data Pool
   * Upload Data File to append
     + Select data file from local computer
     + Encrypt data file and Transaction ID with execution enclave public key and secret key
   * Perform Schema Validation
5. Optin to Append DRT
   * create transaction
   * Sign Transaction Process
   * Issue to blockchain
6. Sign Transaction Process
   * Encrypt each transaction with signing enclave public key and shared secret
   * Send encrypted transaction with user pin and username
   * Receive back signed transaction and issue to blockchain
7. Group Buy Transaction:
   * Create Payment and Buy DRT Transactions
   * Sign Transaction Process for each transaction
   * Send to blockchain as group transaction
8. Group Join Pool Transaction:
   * Create append asset transfer transaction
   * Create payment transaction for execution fee
   * Create application call transaction to request to be added as a pending data contributor
   * Sign Transaction Process for each transaction
   * Send to blockchain as group transaction
   * Return Transaction ID
9. Pass Encrypted Data File and Transaction ID to Execution Service.
10. Optin contributor Token transaction
    * create transaction
    * Sign Transaction Process
    * Issue to blockchain
11. Claim Contributor Token transaction
    * create transaction
    * Sign Transaction Process
    * Issue to blockchain
    * Return contributor token information
12. Store contributor token information via Data service
13. User logout

## Execution\_Service

1. Establish secure communication using attestation and secret key generated from execution enclave with frontend
2. Receive encrypted data set, and transaction ID from frontend
3. Fetch current sealed data set from data service
4. Fetch wasm binary from data service
5. Pass transaction ID, sealed data set and new encrypted data set to enclave and initiate verification of data.
6. Send encrypted transaction data to oracle node and receive back oracle node data
7. Pass oracle node data, sealed data and new encrypted data to execution enclave and initiate join data pool transaction
8. Pass sealed data set, oracle node data and wasm binary into execution enclave to initiate wasm binary
9. Receive new sealed data and unsigned encrypted transactional data form execution enclave
10. Send sealed dataset to data service to be stored
11. Send unsigned transactional data to oracle nodes
12. Receive sealed result and send to frontend
13. Receive sealed result and send to data service to be stored

## Execution\_Enclave

1. Setup Communication channel with frontend using DH exchange and enclave public key.
2. Upon verification initiation:
   1. Decrypt transaction ID with shared secret and enclave private key
   2. Encrypt transaction ID with oracle node public key and shared secret
3. Pass out encrypted transaction ID to Execution\_Service
4. Upon Redeem DRT initiation:
   1. Receive oracle node data, sealed data and wasm binary
   2. Decrypt oracle node data
   3. Perform K out of N over oracle node data to verify
   4. Unseal sealed dataset
   5. Execute wasm binary over dataset
   6. Encrypt result
   7. Return to execution service
5. Upon Join Data Pool initiation:
   1. Receive oracle node data, sealed data and new encrypted data
   2. Decrypt oracle node data
   3. Perform K out of N over oracle node data to verify
   4. Unseal sealed dataset
   5. Decrypt new encrypted data set from shared secret with frontend
   6. Append new data and current dataset
   7. Count the no. of rows of new data
   8. Compute new hash of current dataset + new data
   9. Parse enclave transaction details from Oracle nodes data
   10. Encrypt no. of rows, new hash and unsigned transaction details with oracle node secret key
   11. Return to execution service

## Signing\_Service

1. Establish secure communication using attestation and secret key generated from execution enclave with frontend
2. Receive from frontend and pass encrypted unsigned transactions to signing enclave
3. Return signed enclave transaction to fronted to be issued
4. Establish secure communication using attestation and secret key generated from execution enclave with Oracle Nodes
5. Receive encrypted unsigned enclave transactions from oracle nodes and pass encrypted unsigned transactions to signing enclave and initiate enclave signing
6. Return signed enclave transaction to oracle node to issued

## Signing\_Enclave

1. Receive unsigned transaction from signing service encrypted from user session
   1. Decrypt with shared secret
   2. Grab users key from protectedFile system
   3. Sign with users key
   4. Re-encrypt with shared secret
   5. Return to Signing Service
2. Upon initiation of enclave wallet signing:
   1. Receive oracle node encrypted unsigned transactions
   2. Decrypt oracle node unsigned transactions
   3. Perform K out of N on oracle unsigned transactions
   4. Sign transaction with enclaves private key
   5. Return enclave signed transaction

## Data\_Service

1. Store data pool data per user into Azure Cosmos DB using mongoDB
2. Store sealed data pool datasets per data pool into Azure Cosmos DB
3. Store contributor token per user in Azure Cosmos DB
4. Retrieve data pool list per user from Azure Cosmos DB using mongoDB
5. Retrieve data pool sealed data set from Azure Cosmos DB using mongoDB
6. Pass data to execution service upon request
7. Pass data to frontend upon request
8. Store sealed data results per user

## Oracle\_Nodes

1. Establish secure communication with Execution Enclave via Execution Service
   1. Request execution enclave attestation
   2. Receive enclave public key and shared secret
2. Establish secure communication with Signing Enclave via Signing Service
   1. Request signing enclave attestation
   2. Receive enclave public key and shared secret
3. Upon verification of Transaction ID:
   1. Decrypt transaction ID with shared secret from Execution Enclave
   2. Query Blockchain state for transaction ID
   3. Encrypt blockchain state with execution enclaves public key and shared secret
   4. Return to Execution Service
4. Upon construction of enclaves unsigned transaction:
   1. Decrypt unsigned transactional data, no. rows, dataset hash, from execution service
   2. Compose unsigned transaction for enclave using the above data and AlgoSDK
   3. Encrypt the unsigned transaction with Singing Enclaves public key and shared secret
   4. Return encrypted unsigned transaction to Signing Service to be signed
5. Upon request to issue transaction to blockchain
   1. Receive signed transaction from Signing Service
   2. Issue to blockchain