October 03 2024

CS61065 Theory and Applications of Blockchain

Assignment 4: Hyperledger Fabric

Date of Submission: **October 20, 2024 EOD**

Note that this is a group submission. Each group should submit one copy of the solution.

In this assignment, you will get familiar with Hyperledger Fabric, a permissioned blockchain framework. You will learn how to compile, deploy, and interact with smart contracts (called chaincodes) in Hyperledger Fabric.

Implement a blockchain network with two organizations on the Hyperledger Fabric platform. This assignment focuses on secure document management, where each organization can store and manage documents while maintaining the privacy of their data.

Part A: 40 Marks

Data to be stored on the Blockchain:

- **Public Data:**
  - **Account Balance:** The current balance of each organization's account.
- **Private Data:**
  - **Document Information:**
    - **Document ID**
    - **Document Title**
    - **Document Data Hash** (Hash of the Document Data)
    - **Document Data**
    - **Document Price**

Chain-code Functions (20 Marks):

1. **Add Money to Account:**
   - async AddBalance(amount)
   - **Functionality:**
     - Send the details as transient data.

        - Add the specified amount to the account balance of the client's organization.
2. **Add New Document:**
    - async AddDocument(docID, docTitle, docData, price)
    - **Functionality:**
        - Send the document details (id, title, data, price) as transient data.
        - Compute and store the document data hash (use SHA256).
        - Store the document details in the private data collection for the organization.
3. **Get Balance:**
    - async GetBalance()
    - **Functionality:**
        - Read the public account balance of the organization and return the details.
4. **Update Document Data**
    - async UpdateDocument(docID, newDocData, updateHash)
    - **Functionality:**
        - Send the new document data as transient data.
        - Check if the document with the specified docID exists in the private data collection of the organization.
        - Update the document's data with the newDocData.
        - If the updateHash boolean is true, compute a new hash for the newDocData (using SHA256) and update the stored document hash.
        - Store the updated document details in the private data collection for the organization.
5. **Get All Documents:**
    - async GetAllDocuments()
    - **Functionality:**
        - Return the details of all documents stored in the private data collection of the organization.
        - The returned details should include the Document ID, Document Title, Document Data, Document Data Hash, and Document Price for each document.
6. **Get Document Details:**
    - async GetDocument(docID)
    - **Functionality:**
        - Read and return the Document Title, Document Data, Document Data Hash, and Document Price from the private data collection for the organization based on the provided document ID.

Note: Ensure that one organization cannot access or infer the document information like Document ID, Document Title, Document Data, Document Data Hash, and Document Price of documents of the other organization. **If this is violated then only 50% of the marks will be awarded.**

Terminal-Based Application (20 Marks):

Create a terminal-based application for each organization to connect to the network. The application should support the following commands:

- ADD_MONEY <amount>: Invoke the AddBalance(amount) function.
- ADD_DOC <docID> <docTitle> <docData> <price>: Invoke the AddDocument(docID, docTitle, docData, price) function.
- QUERY_BALANCE: Invoke the GetBalance() function.
- UPDATE_DOC_DATA <docID> <newDocData> <updateHash> : Invoke the UpdateDocument(docID, newDocData, updateHash) function.
- GET_ALL_DOCS: Invoke the GetAllDocuments() function.
- GET_DOC <docID>: Invoke the GetDocument(docID) function.

Part B: 40 Marks

Support the trade of documents on the blockchain network with secure and validated document transfer. This part involves using events and event listeners and is a continuation of the previous part.

Data to be Stored on the Chain:

- **Marketplace:**
  - A collection of all the documents that organizations want to sell. This includes the Document ID, Document Title, Document Data Hash, Document Data, Document Price, and Seller. This data is public.
- **Purchase Record:**
  - A record of completed purchases, containing the Document ID, Seller, Buyer, Price and Document Data Hash. This data is public.

Note: Here buyer and seller attributes refer to the organizations.

Chain-code Functions (30 Marks):

1. **Add Document to Marketplace: 5 Marks**
   - Add a document to the marketplace with its Document ID, Document Title, Document Hash, Document Data, Document Price, and Seller.
   - Ensure that the document exists in the seller's private data collection.
2. **Buy Document from Marketplace: 15 Marks**
   - Deduct the document price from the buyer’s balance if there is sufficient balance.
   - Check if the hash of the data and data hash attribute matches. If not, refund the buyer and remove the document from the marketplace.

    - If the hashes match, then remove the document from the market place and add it to the buyer's private data collection.
    - If the purchase is successful then add an entry to the Purchase Record and add the document price amount to the seller's balance.
3. **Get All Documents in Marketplace: 5 Marks**
    - Return the details of all documents stored in the Marketplace.
    - The returned details should include the Document ID, Document Title, Document Data, Document Data Hash, Document Price and Seller for each document.
4. **Get All Records in Purchase Record: 5 Marks**
    - Return the details of all records present in the Purchase Record.
    - The returned details should include the Document ID, Document Data Hash, Document Price, Seller and Buyer for each document.

Application Functionality (5 Marks):

The application should support all the functionalities from the previous part and the following additional functionalities:

- **ENLIST_DOC Command:**
    - Enlist a document to the marketplace by specifying the document ID.
- **WISHLIST Command:**
    - Add a document to the wishlist by specifying the document ID.
- **ALL_DOCS Command:**
    - Display all the documents available in the Marketplace.
- **ALL_RECORDS Command:**
    - Display all the records present in the Purchase Record.

Additional Requirements (5 Marks):

1. **Wishlist Management:**
    - Maintain a local wishlist of desired documents for each client application. The user can add documents to the wishlist using a command.
    - Listen for events indicating a new document has been added to the marketplace, and automatically trigger a smart contract to purchase the document if it is on the wishlist.

Submission:

Submit a zip file containing the chain code and application code for both the parts in separate folders along with any other config files required to deploy your code and a README in which clearly mention the group details and instructions on how to run your code. The zip file should be named *<Roll_no_member_1>_<Roll_no_member_2>_Assignment_4.zip*.