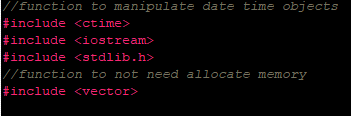
COMP2034  
C++ Programming  
  
Coursework 1 Report  
  
Healthcare Management System Using Blockchain  
  
  
  
Name: Soo Min Hao

Student ID: 20297525  
Module Convenor: Doreen Sim Ying Ying

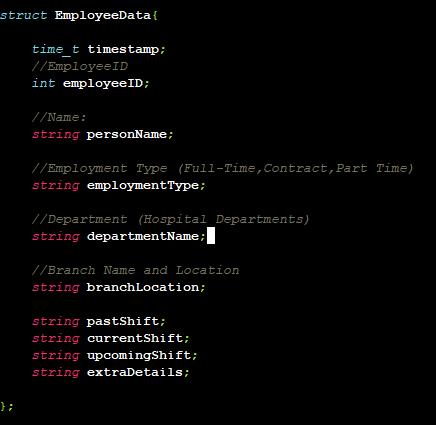
According to Yuan & Wang (2018), blockchain technology has been popularised since the introduction of Bitcoin. However, the technology itself has been introduced a long time ago in (Lee, 2019). The hype and rise of cryptocurrency in the 2010s proved that such technology has gained traction due to its security. Blockchain technology is described as a database that is a distributed database that makes it impossible to change or hack the system (Lee, 2019). Hence, today instead of only utilizing blockchain on cryptocurrency, we can also implement it in medical system which can yield several benefits especially during this covid-19 pandemic. Bell et al. (2018) stated that with blockchain technology, storing medical records can yield more secured data storage as well as preventing hackers from obtaining sensitive information of individuals through cyberattack on infrastructures especially hospitals. They can also be used for device tracking, clinical trials, pharmaceutical tracing, and health insurance (Hasselgren et al., 2020). Therefore, this report is a summary of what I have implemented on the medical record system through utilizing blockchain technologies and techniques through C++ programming.

There are several assumptions done during the making of this blockchain code. One is that this blockchain only stores employee data and not other patients’ data. Two is that the program is only used to show the employees data in the blockchain and that there is no addition function such as adding new employees to the chain, deleting employees from the blockchain as in the writing of this sentence. Three, this medical company only has 1 blockchain that only stores employee data rather than patients data and medicine which can be added.

(i) Full description of the developed C++ program  
  
 The C++ program written is called healthblockchain.cpp which is a type of medical record system that stores the employee’s data in a blockchain format. Currently, it is only able to display the block information of employees only and no patients. Some comments might not be in the screenshots of the code snippet but may appear in the actual code due to updates in comment in code. Pictures of Snippet of code will be included below together with explanation for easier understanding:

Starting with #include library function and why they are used  
  
  
#include <ctime> is used to generate a timestamp of blocks created.

#include <iostream> is used for handling strings, since structures are in string form, therefore this library is used.  
  
#include <stdlib.h> is a standard library of C++, therefore it is used here

#include <vector> Vector is used because it is a dynamic array and hence able to change its size dynamically. This is essential for developing a blockchain system as blockchain may contain thousands of blocks.  
  
  
  
  
This EmployeeData structure is used to as a structure on the information of employee to be stored in a block in the blockchain.

personName is the employee’s name.

employmentType employee’s employment type such as contract, full-time, intern and such…

departmentName – Hospital Departments such as Administration, and different departments for different specialists such as cardiology, radiology, orthopedia, renal, etc…

branchLocation – Which branch of hospital the employee is stationed at.

pastShift – Previous work time of employee

currentShift – Show the current working time of employee.

upcomingShift – Employee’s next working time

extraDetails – Extra details about the employee  
  
  
  
  
  
  
  
  
Text

Description automatically generatedThe following code snippet in the previous page is a Block class. The block class contains the block index, previous hash, and current hash as well as employee data. Within the class, it has a function that calls a block constructor (snippet of code in the following) that constructs the block. This block will also call generate hash function and get previous hash function to generate this block’s current and previous hash.   
  
Note that currentBlockHash is set to public to ensure that the genesis block’s previous hash equals to current hash.  
  
Then 3 get functions is implemented so that they can be called to display the employee’s information in the main function.  
  
  
Text

Description automatically generated

The code snippet above shows the constructor for Block class which takes the parameter of index as block index, previous hash, and employee information. To access previous block’s hash, **this->** keyword is used. Then, the current block will call the generateHash() function to generate a hash for the current block.

Text

Description automatically generated  
  
This code snippet this code represents the process of generating and managing hash values using blockchain implementation.  
  
**generateHash()**: This function is used to generate a hash value for a **Block** object by using **hash<string>** and **hash<size\_t>** functions from the C++ standard library. It generates two hash values for strings and one hash value for a **size\_t**. Then, it concatenates the **employeeID** and **timestamp** values of the **data** member variable of the **Block** object to create a string called **toHash**. It then calculates the hash value of this string using **hash1** and the hash value of the previous block using **hash2**. Finally, it adds these two hashes values and generates a final hash value using **finalHash**. This final hash value is then returned and since a hash value can be multiple digit, a modulus is used to set it to 10 digit instead.  
  
**getHash()**: This function will return the hash of current block object.  
  
**getPreviousHash**(): This function returns the hash value of the previous Block object in the chain.

Text

Description automatically generated

To create a blockchain, we must have a blockchain to begin with, hence we define a class called **Blockchain.** The we need to make a blockchain object hence, a constructor for this class is made. The Blockchain class is used to represent the data structure of employee in a chain. This Blockchain class has four member functions and one data member:  
  
createGenesisBlock(): This is a function that returns the first block of the blockchain, known as the genesis block. It is only accessible within the Blockchain class and this blockchain is made such that the first block is a genesis block but doesn’t contain any data. It will only output the data structure which will be seen in the screenshot of output folder.  
  
chain: Is a public vector of Block objects that represents the entire of this blockchain.  
  
Blockchain(): This is the constructor for the Blockchain class that creates the genesis block and adds it to the blockchain.  
  
addBlock(EmployeeData data): This function takes the whole structure of EmployeeData as input and creates a new block containing this data which the new block is then added to the end of the blockchain. However, the genesis block will not contain any information of the EmployeeData structure (following photos).

The constructor will initializes a new Blockchain object by creating the genesis block using the createGenesisBlock() function and pushing it onto the chain vector. The genesis block is the first block in the blockchain, and usually we set it manually.

Text

Description automatically generated

createGenesisBlock() is a private member function of the Blockchain class. The purpose of this function is to create the first block of the blockchain, (genesis block).

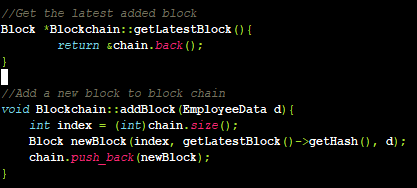
The genesis block has special characteristics, as it is the first block in the chain and does not have a previous block to reference. Therefore, its hash value is typically hardcoded or manually generated, in this case the previous hash of genesis block will be equal to the current hash of genesis block.

Since it is a genesis block, we manually include the data and, in this case, it’s an empty string. We start by creating an object of EmployeeData type named employee1, which represents the data to be stored in the genesis block. We set the employeeID, personName, employmentType, departmentName, and branchLocation member variables of employee1 to empty or default values and the timestamp variable of employee1 to the current time using the time() function from the <time.h> C++ library.

Next, the function uses the hash<string> function from the C++ standard library to generate a hash value for the string "genesis", which is a constant string that is typically used as the input for the genesis block's hash value. The hash value of "genesis" is stored in a string variable called hashStr. Then, the function calculates the hash value for hashStr using hash1 and stores it in the variable previousBlockHash. It then calculates the same hash value again and stores it in the variable currentBlockHash. This is because the genesis block's current hash value is set to the same value as its previous hash value in this implementation of blockchain program.

Finally, the function creates a new Block object named genesis using the Block constructor. The Block constructor will take three arguments: the block's index (which is 0 in this program for genesis block), the previous block's hash value (which is set to previousBlockHash, and the data to be stored in the block (which is set to employee1).

The function then sets the current hash value of genesis to be equal to currentBlockHash for this implementation of blockchain and returns the genesis object. Summary, this function creates a genesis block, and we must create it manually to establish the first set of data in the blockchain.

  
The purpose of these functions is to add new blocks to the blockchain and retrieve the latest block that was added.

getLatestBlock() returns a pointer to the latest block in the blockchain, which is the last element of the chain vector using the back() function of the vector class, that returns a reference to the last element of the vector. The & operator is used to return a pointer to this element.

addBlock() function takes an object of EmployeeData type as an argument and adds a new block to the blockchain with this data. It starts by getting the index of the new block, which is equal to the current size of the chain vector and then create a new Block object using the Block constructor to increase the block in the blockchain.

The constructor takes three arguments: the index of the new block, which is set the index of created block; the hash value of the previous block, which is obtained by calling getHash() on the latest block in the chain using getLatestBlock()->getHash(); and the data to be stored in the new block, which is data in the structure of EmployeeData.

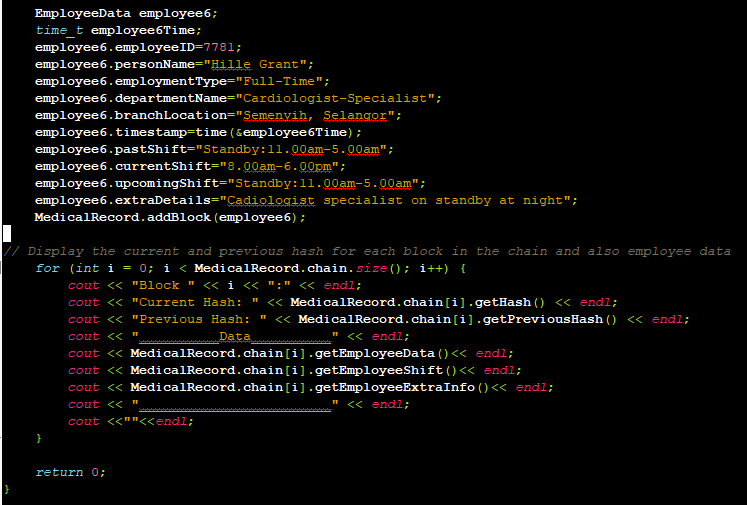
Finally, the newBlock object is added to the end of the chain vector using the push\_back() function of the vector class. In summary, these functions are needed to add new blocks to the blockchain and retrieve the latest block in the chain.  
  
  
  
  
  
  
  
  
Text

Description automatically generated

In int main(), the code creates a new blockchain object called MedicalRecord and then adds six blocks to the chain representing different medical employees with their associated data.

Text

Description automatically generated



Each block in the chain contains an object of the EmployeeData struct that represents an employee's information, including their employee ID, name, employment type, department, branch location, and timestamps for their past, current, and upcoming shifts. Additionally, the EmployeeData struct contains a string field for any extra details related to the employee's work.

The code then uses the MedicalRecord.addBlock() function to add each EmployeeData object to the blockchain as a new block. Each block has its own hash and a previous hash that links it to the previous block in the chain.

Finally, the program prints out the block index, current and previous hashes for each block in the chain, along with the employee data and shift details. A line of underscores is included to organize each block data by block.

(ii) Description of all the components, structures and data structures, basic and application functions involved in the developed C++ program;

This C++ program implements a simple blockchain using the "Block" and "Blockchain" classes. The program defines a struct "EmployeeData" that stores information about employees.   
  
The ‘Block’ class contains: a constructor to create a new block, a function to generate the hash of a block, and functions to access the current and previous hashes of a block. The ‘Blockchain’ class contains a constructor to create a new blockchain with a genesis block, a function to add a new block to the blockchain, and a function to get the latest block in the chain.

The program initially includes necessary C++ libraries such as ctime, iostream, and vector that is essential for this blockchain code program to run. The "using namespace std" statement is used so we do not need to to prefix standard library names with "std::".

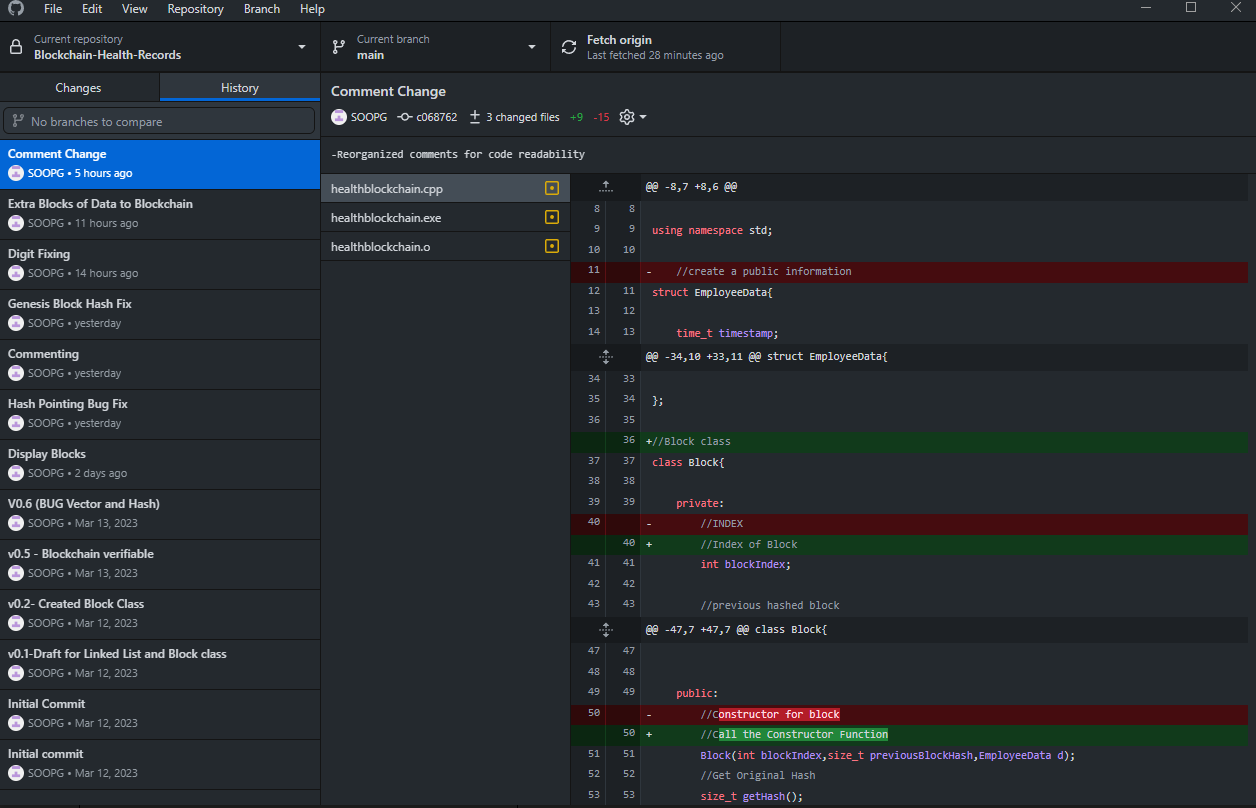
The EmployeeData struct has several fields, including a timestamp, employee ID, name, employment type, department name, branch location, and extra details.

The Block class has private fields for the block's index, previous block hash, and current block hash. The constructor takes an index of block object, a previous block hash, and an EmployeeData object as parameters. The Block class has public functions to get the current hash of a block, the previous hash of a block, the EmployeeData object associated with the block, and the employee's shift and extra information.

The Blockchain class has a private function to create a genesis block and a public vector of Blocks. The constructor creates a new blockchain by creating a genesis block and adding it to the chain. The addBlock function takes an EmployeeData object as a parameter and creates a new block with the appropriate index and previous block hash. It then adds the block to the chain. The getLatestBlock function returns a pointer to the latest block in the chain.

The main function creates a new Blockchain object, MedicalRecord. It then creates an EmployeeData object, employee1, and adds it to the blockchain. The employee1 object has various fields filled with information about an employee. The information is added to the genesis block of the chain. However this is not the same as employee1 in int main(). In the main function, six blocks are created using struct data employee and after adding them to the block, then they will display it block’s information which is block index, current and previous hash of block and the block’s employee information

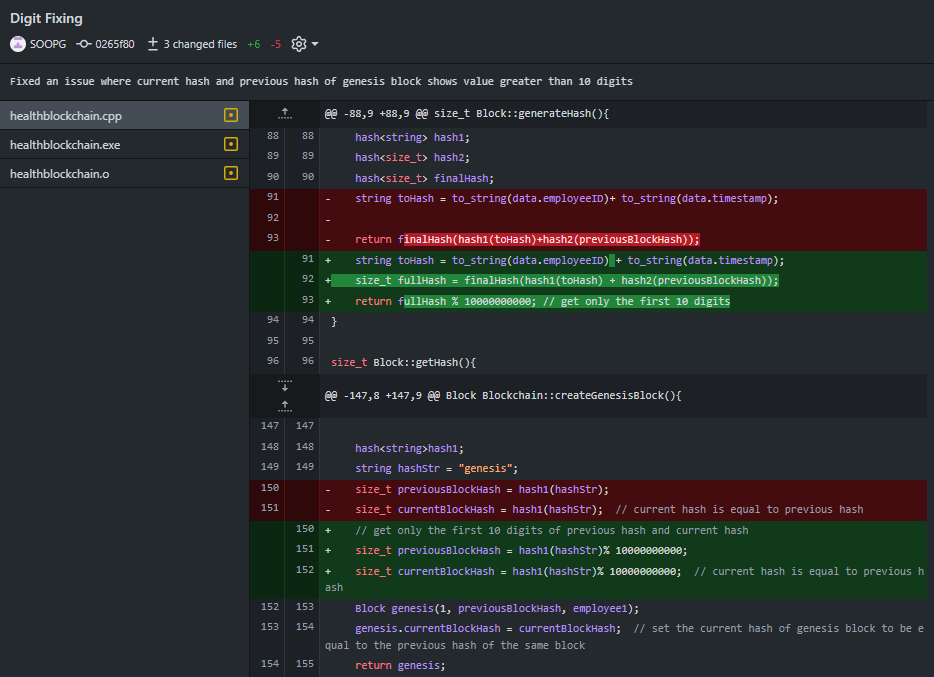
(iii) Overall Experience

My experience in theoretically speaking made it seem to be a very difficult coursework since it involved blockchain technologies, however through coding practically it is not the case. Blockchaining technology concept isn’t difficult to be grasp upon and neither does the code. However, during implementation, it may seem difficult due to the sequence to code, and steps required in creating a block. It is also slightly difficult in creating the hashing function due to hashing requires a certain formula to be used.  
  
This coursework also further helped me in utilizing and understanding the importance of using Github. Before starting this project, I have created a github repository for version control of the code which can be found in the following picture. This is important as it help me to keep track of the bugs and list of items to be fixed.  


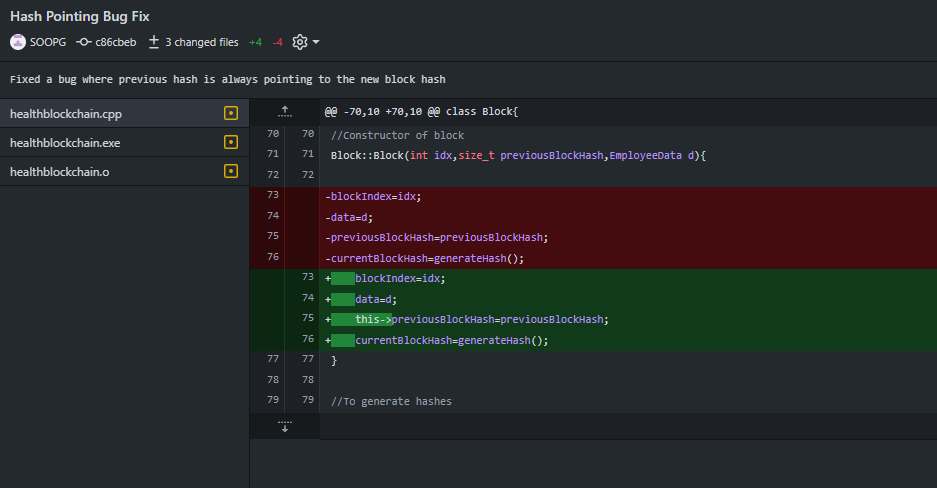
(iv) Difficulties Faced

1-Current hash equals to previous hash for genesis block  
This was difficult to implement because initially, I assigned current hash as a private data. However this caused the current hash value of genesis block to not be equal to previous hash of genesis block. In reality in blockchain technology, genesis block doesn’t have a previous hash as it is the first block. Hence a modification is made where the previous hahs value of genesis block should be its current hash value.  
  
Text

Description automatically generated

2-Setting Max the hash to digit 10  


The previous bug I faced was when displaying a current block hash, it will display random digits rather than 10 digits, in order to do so, I have to use a modulus % in the code to get only first 10 digits of a hash. It was difficult to debug as it involved hashing and, the formula of hashing was not defined clearly in terms of how many digits it will take, hence the difficulty.

3- Previous Hash of block is pointing to latest block rather than previous block  
  


Another difficulty I faced was the existence of a bug where:  
 assuming 1st block current hash is 1234567890.

2nd block previous hash will not show 1234567890 but instead a random number. It is difficult to trace this problem due to the code using pointers. However, I discovered that previousBlockHash should be changed to this->previousBlockHash to access that the actual previous block hash.

(v) Conclusion and discussions about doing this coursework

In conclusion, the coursework provided me with strong foundation in C++ programming as well as understanding how a blockchain works in general and in codebase. The code uses the hash function to generate the hash of the employee data and the previous block in the chain to ensure the integrity and security of the chain. The hash function used is based on the SHA-1 algorithm, which is a widely used cryptographic hash function.   
  
Overall, this code provides a basic implementation of a blockchain structure in C++ using a block class and a blockchain class. However, it is important to note that this implementation is limited as there is no graphical interface for users to interact with the records such as inserting new information, deleting new information which may prove to be desirable. Additionally, there is only one blockchain and not multiple blockchains and this is not realistic as it only shows employees’ information rather than patients’and medicinal information. It is only merely to show the implementation of blockchain technology in medical field. Finally, this coursework allowed me to realize and further emphasize on using GitHub as a version control.  
  
In future implementation, I plan to include a user interface for main menu and include features such as the ability to insert a new block and delete a block in the blockchain using a switch statement and adding other information such as medicines and patients’ information in the blockchain.

(vi) References   
  
Bell, L., Buchanan, W. J., Cameron, J., & Lo, O. (2018). Applications of Blockchain Within

Healthcare. *Blockchain in healthcare today*.

Lee, J. Y. (2019). A decentralized token economy: How blockchain and cryptocurrency can

revolutionize business. *Business Horizons*, *62*(6), 773-784.

Hasselgren, A., Kralevska, K., Gligoroski, D., Pedersen, S. A., & Faxvaag, A. (2020).

Blockchain in healthcare and health sciences—A scoping review. *International Journal of*

*Medical Informatics*, *134*, 104040.  
  
Yuan, Y., & Wang, F. Y. (2018). Blockchain and cryptocurrencies: Model, techniques, and

applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(9), 1421-1428.