DISTRIBUTED FILE SYSTEM IN HASKELL

LAURA RUNDLE || COMPUTER SCIENCE AND BUSINESS || 13321661

GITHUB REPOSITORY:
GITHUB.COM/RUNDLEL/DISTRIBUTEDFILESYSTEM

The initial steps in order to complete this project was to plan the system on paper and determine how each service would communicate with each other before beginning development. I committed code to my GitHub Repository regularly over the project period and this code is entirely my own work. In some cases, I requested help from the TA's or my classmates and some of the ideas were discussed between myself and my classmates.

The file system I aimed to implement is the upload/download style access model.

DATABASE

This file system operates off a MongoDB database. The initial steps of developing this system was to get the database in place and be able to insert things into and extract things from the database. MongoDB works off collections. Two collections were used in this system – "Files" and "Users"

USERS - Usernames and encrypted passwords were stored in this Collection. This collection was later updated to include the permissions that a user had eg. READ or READ WRITE

FILES – This Collection was intended to store data from text files

All data must be converted to BSON before it is stored in the MongoDB database.

COMMANDS:

To add a user to the database:

```
curl -X POST -d '{"username":"LauraRundle", "password":"securepass",
"permissions":"READ WRITE"}' -H 'Accept: application/json' -H 'Content-type:
application/json' http://localhost:8080/insertUser
```

To return a token:

```
curl -X POST -d '{"username":"LauraRundle", "password":"securepass"}' -H 'Accept:
application/json' -H 'Content-type: application/json' http://localhost:8080/returnToken
```

To insert a file into the database:

```
curl -X POST -d '{"fileContents": "This is a file"}' -H 'Accept: application/json' -H 'Content-
type: application/json' http://localhost:8080/postFile
```

FILE SYSTEM

This is the main functionality of the system. The postFile function allows the contents of a file to be stored in the database.

I also attempted to extract information from a .txt file and store it in the database. I was successful in my attempts to extract the data and manipulate it, for example, change the contents to uppercase. However, I was unable to store it in the database correctly.

SECURITY & AUTHENTICATION

Usernames and encrypted passwords were stored on the database in order to be used to verify if the user could access the file system. Once the user was authenticated they would be provided with a token in order to access the files.

The insertUser function takes in a username, password and permissions. The password is encrypted and a User data type is created. This data type is then converted to BSON and stored in the database.

The encryption used in this file system is an adaptation of the Caesar Cipher. A random number is generated which is then used to convert the text to an alternate ascii character in order to encode the message.

The encryption and decryption functions work with map and ord. The strings are converted to hexadecimal and then they are increased or decreased by the value of the key.

The encrypt and decrypt functions that I wrote do work with the random generated number however in order to maintain the state of the key, I set it to the constant 7 in order to move on with other parts of the system.

A token data structure was used to store the key from the encryption and some metadata. The meta data that I used was the permissions of the user. This was stored in the User collection. I didn't get this fully working with the permissions metadata and I intended to add additional meta data to provide more information such as how long the token or the key were valid for.

This token would then be encrypted with a second key.

The returnToken function uses the username an encrypted password to find the user in the database. This ensures that the user is authorized – i.e. the username and password match. The function then returns hard coded constant meta data.

In order to advance this function, I made a generateToken function which I was unable to get fully working. I intended to extract the permissions of a user from the database and use this data to create a token.

CACHING

This was the last feature that I had the time to discuss with classmates. I was unable to implement it within the time frame of the project. I intended to use the permissions of the user to enable them to download the files to their computer.

REPLICATION

I was unable to complete this feature. I intended to use a model based on primary copy. As I aimed to use an upload/download style model which doesn't require as much communication as other models e.g. NFS.

TRANSACTION

I was unable to implement this feature in the given time. I intended to try and implement the algorithm that was outlined in the project description in order to execute this feature.

LOCKS

I was unable to implement this feature in the given time.