Distributed File System Project Report

David Clince | Computer Science & Business | 13319802

Github Repsository: https://github.com/clinced94/Distro_project

Introduction

In this report I will be describing my progress the distributed file system assignment, detailing what I managed to complete, along with some of the challenges I encountered. I worked with some classmates to develop parts of the project, while other parts I developed by myself.

I spent a lot of time at the beginning of this assignment trying to get up to speed with the Haskell language as it, and functional programming itself, were things I had no experience with. Distributed systems in general were still quite new to me as well as this was my first module on the subject, which added to the learning required. It took quite a long time to get up to a reasonable ability in Haskell, and even then I was never very comfortable with the language. I also spent some of this time planning out on paper how I would construct this project, what order I should build them in, and how they would all interact with each other. My priorities were to work on the authentication and database/file system.

The system I created is made up of the following sub-systems:

- Database
- Authentication system
- File system
- Caching system

Database

The first thing I implemented was a MongoDB database. Inserting and extracting from a database are core parts of this system so I wanted to get them working early on. I used two ‘collections’ in this project, Users and Files. Users stored usernames and encrypted passwords, Files stored data on the files being used.

Authentication

This is the section of the project where I made the most progress. The encrypt and decrypt functions allowed me to encrypt passwords or files and then decrypt them again. These functions used a simple Caesar Cipher made with by using the Map function along with Ord in order to convert the characters into hexadecimal values, manipulate them, and then convert them back to characters.

The addUser function took in a User, encrypted the password associated with the user and then called insertUser in order to insert the username and encrypted password to the database.

I added a client to my project later on in the development to try and test some extra functionality such as retrieving certain users, but I never had time to implement it fully.

I was also working on returning a token to each user, which would either grant them Read, or Read/Write permissions. My difficulties with Haskell slowed down my progress quite a lot as I found it difficult to find what was going wrong in my code, which required a lot of debugging on almost every function I wrote.

File/caching System

The saveFile function here took in a file and saved it to the Files collection of the database, this was developed alongside my implementation of the database. I was planning on adding my Caesar cipher functions to this section as well in order to encrypt the files before saving them, but I ran out of time before I could do it.

I spent a bit of time trying to get a getFile function working, where the user could access the database and save the contents locally, but I was unable to get it working.

Evaluation

I think that if I had started learning Haskell earlier on in the year I would have been more successful with the project. Having to learn the basics of Haskell alone took quite a bit of time, not to mention the more advanced knowledge I needed to even understand the code that was provided by the lecturer. I ended up with a learning curve that was a bit too steep for me to be confident with what I was doing and how everything was interacting witch each other.