EVENT HUB

C PROGRAMMING REPORT

Group 34

Ryan Cleminson, Tylar Hart, Kayla Gelman, Jacob Gowing, Dee-Jay Logozzo

TABLE OF CONTENTS

EXECUTIVE SUMMERY

Our C Programming assignment required us to develop a professional program that will manipulate real world data. When deciding on our solution, our group decided to settle on a common issue that we could all relate to as university students, hence we decided to create an Events Management System called ‘Event HUB’.

Our group approached executives in university societies who expressed that a major issue that societies have is communicating with students about upcoming events. This problem was instigated after we asked an events coordinator how we (as university students) are meant to find out about different events, bringing about our problem statement which will can be found as a succinct sentence on the next page under ‘Scope of Project’.

Overall, our group worked well together, enabling us to create this final report which details the scope of our project, structure of our code, why we chose certain methods and what we were able to accomplish in these past 3 weeks.

SCOPE OF PROJECT

PROBLEM

Communication between event organisers, staff and patrons is often difficult when spread across multiple platforms.

If people are notified about events through platforms such as Facebook, email, discord, ActivateUTS, etc. It can be hard to update and manage all these events, making it hard to notify people about upcoming events in order to gain exposure. We aim to solve this issue by providing a platform designed for event coordinators and patrons where people can create or join events on a secure platform.

OVERVIEW OF OUR PROGRAM

Event HUB aims to provide a platform where anyone can create, edit or join an event. By creating an event, the user immediately becomes an event coordinator where they can specify the event name, whether or not they would like the event to be free, the type of event and the location of the event.
Any user can create an account and events, however, only the Admin is able to save and load database files. Additionally, when a user creates an account, their passwords are immediately encrypted and when the admin saves the database file, the whole file is encrypted.

Once the events have been created, the event coordinator is able to add staff or simply edit any important event details. If a user decides they want to join an event as a patron, then their details are added to the event.
It is important to note that whilst our idea has been inspired by the university society system, we have not designed our criteria, specifically for UTS as we decided to create a broad design for the benefit and use of everyone.

WHAT WE ACHIEVED

→ Using a Make File
Our code has been written using a Make file which complies the code to create the executable file ‘EventManager’. This was very convenient for working as a group as we could add working functions to the file instead of uploading code with bugs which could cause confusion for everyone.

→ Implementing Admin Mode
The admin views a separate menu to a regular user. To access this menu, when logging in the main program - username: Admin , password: Password

→ Account Creation and Logging in
We were able to implement an account creation function whereby a user can create an account and the program will validating their password and other crucial information such as date of birth in order to ensure that the information obtained is valid.

→ Encryption
We successfully implemented two Encryption methods; the XOR Cipher and Substitution Cipher. The Substitution Cipher was used for encrypting the password and the XOR Cipher was used to encrypt the whole database file.

→ File Creation
The Admin is able to successfully both save and view database files.

→ Run Time Modes
We successfully created 2 different run time modes. The first mode is the regular mode and the second is the 'Info mode' which will display information about how to use the program and what it is designed for. In order to run this mode, type './EventManager info' into the command line after compiling the program.

→ Debugging mode
When running the program in the command line with the word 'debug' our program enters debug mode which shows the process of encryption and runs through which values to enter.

→ Implementing Search Function
When joining an event, the user is able to search for a specific event that has been loaded into the database.

→ Linked Lists
Linked Lists have been implemented when joining an event as it links the users details to the event.

WHAT WE DIDN'T ACHIEVE

Overall, our group achieved a lot, however, there are a couple things that we were not able to implement in time.

→ Compression
We didn't implement a compression method, due to time restrictions and we believe that our program has reached the expected level of complexity. However, our report will discuss the compression method we had begun to code and why we chose it.

→ Automatic File Saving
One flaw that we were not able to change was the fact that once a user creates an event or a new account, their data isn't immediately saved to the file. Rather, the Admin has to log in and save the database manually. Whilst this doesn't affect the overall functionality of our program, automatic file saving is definitely a future development.

STRUCTURE CHARTS

NORMAL MODE

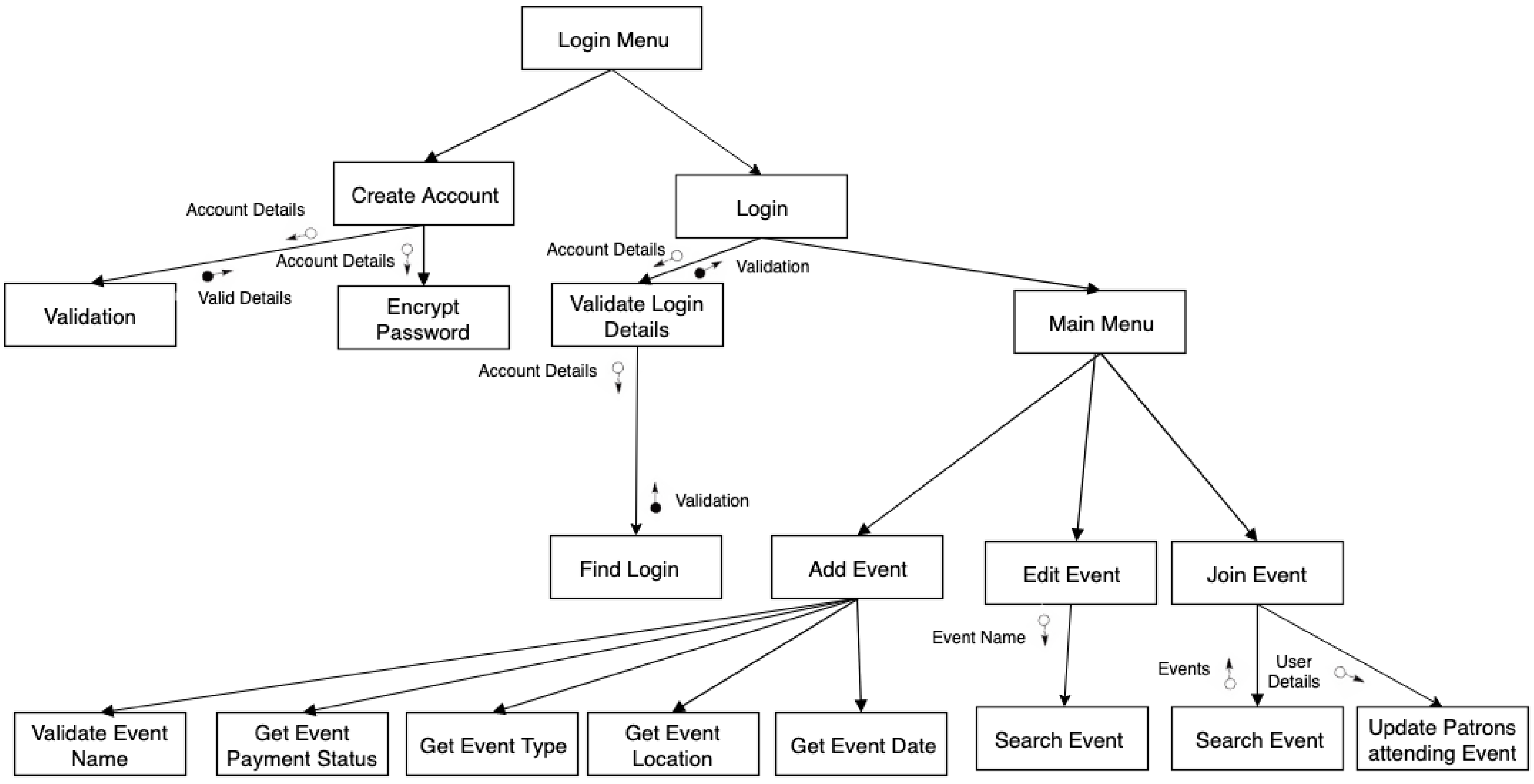


Figure 1. ‘Normal Mode’ Structure Chart

ADMIN MODE

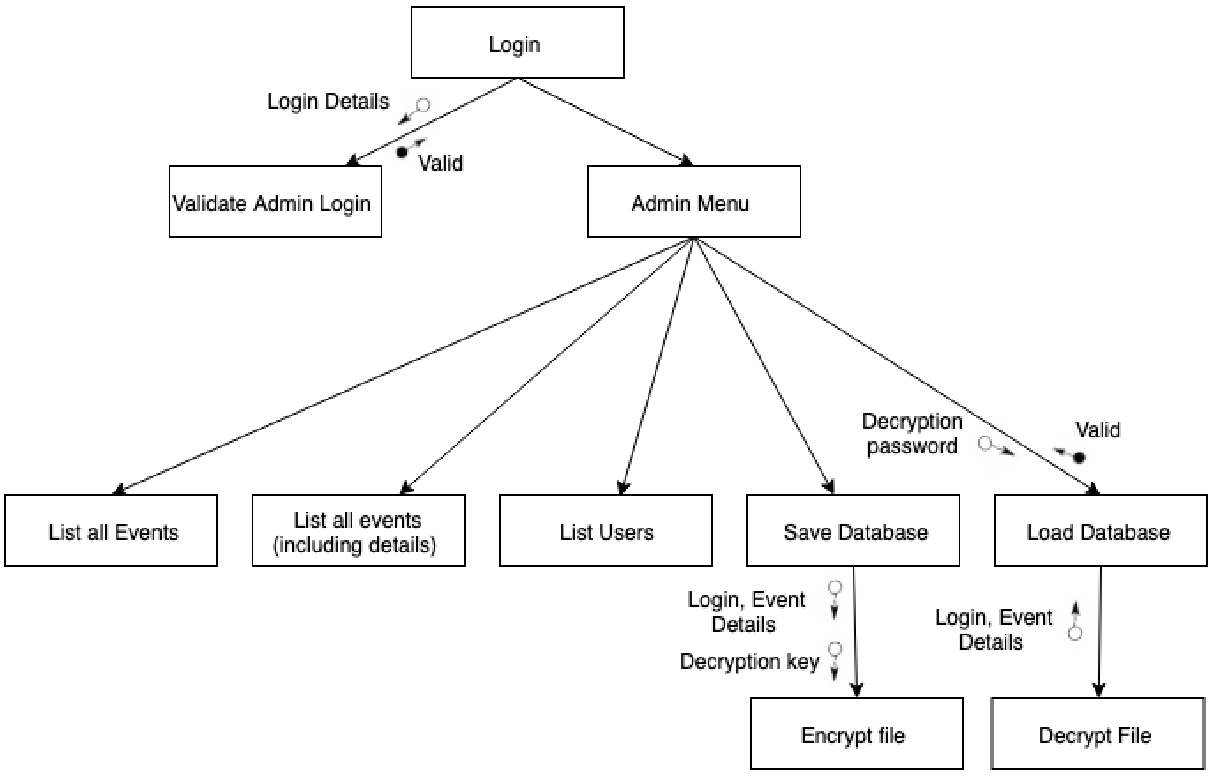


Figure 2. 'Admin Mode' Structure Chart

ACCESSING ADMIN MODE

In order to access the Admin mode, when logging into the main program, the following details should be used:

Username: Admin
Password: password

WHY OUR DESIGN IS EFFECTIVE

The premise of our design is that it provides a secure platform for users to create or join events. The security of user's details is ensured by our thorough encryption implementation, a further discussion about our encryption methods can be found on page 8, under 'Chosen Encryption Method'.

As illustrated in the figure 2. 'Admin Mode' structure chart, only the admin is able to update the database and save changes, allowing all changes to be confirmed by the admin. This is very important as the program is designed for businesses and university societies, where there is usually one person in charge of managing events.

Additionally, when joining an event, as illustrated in figure 1. 'Normal Mode' structure chat, users are able to search through current events and join them. When joining events, their information is saved to the event through the use of linked lists. After, when the admin saves the events database file, the data is encrypted so that only the Admin is able to view this information.

ALTERNATIVE DESIGN SOLUTIONS

Some possible design alternatives include:

- Modifications to 'Join Event' function, whereby users could search by the type of event, location or time. Currently, they care only able to search by the name of the event.

- Currently, everyone is able to edit an event, a future development would be to restrict this to the event creator. However, this may change the scope of our project as it is currently aimed at societies and businesses, implementing this function could change the scope of our project.

ENCRYPTION METHODS

METHOD 1 – SUBSTITUTION CIPHER

This method of encryption changes the value of one letter in a string at a time. More specifically, it substitutes one letter and a time with a predefined set of letters or symbols in order to create a new output. Usually, this type of cipher uses the plaintext alphabet, meaning that there are 26 different keys in which the true value can be. (University of Rhode Island).

Advantages	Disadvantages
- Easy to implement as each letter is substituted with a predefined letter.	- Only has 26 possible permutations and is therefore insecure and can be easily solved.

METHOD 2 – XOR CIPHER

The XOR Cipher requires a predefined XOR key and applies this key to the output. This method works by considering every output as a binary string. An XOR is then applied to both the output and predefined key to create the encrypted message.

Advantages	Disadvantages
- It is hard to decrypt by generating random keys - Fast computation	- Not secure enough on its own for complex systems

CHOSEN METHOD

We decided to use both the Substitution Cipher and the XOR Cipher. This was because, they both served good purposes for different functions. The whole database file is encrypted using the XOR Cipher because it is a good method of encryption, hard to decrypt and can handle large amounts of inputs without slowing the system down.

The reason for including the substitution cipher is, in the case that the file is decrypted, the passwords will still be secured with a different key, making It harder for any malicious activity to take place.

COMPRESSION METHODS

METHOD 1 – RUN-LENGTH ENCODING

This is a very simple method of lossless data compression in which sequences of repeated data are stored in a single data value with count, rather than being repeated multiple times.

Advantages	Disadvantages
- Easy to Implement - Does not require much CPU Horse Power	- Inefficient in comparison to Huffman coding - Only efficient with text files lots of repetitive data - Better for text files that have lots of spaces for indenting

METHOD 2 – HUFFMAN CODING

This is a popular algorithm for lossless compression of files based on the frequency of occurrence of a symbol in the file that is being compressed. It works by assigning variable-length code (bit sequences that have a distinct length) to input characters. The length of the variable will then depend on the frequency of the characters. So, symbols that occur a lot in a file are given a short sequence whilst others that aren't used as often get a longer bit sequence. (Huffman Coding, 2019).

Advantages	Disadvantages
- Efficient in compressing text files	- Not suitable for compressing images

CHOSEN METHOD

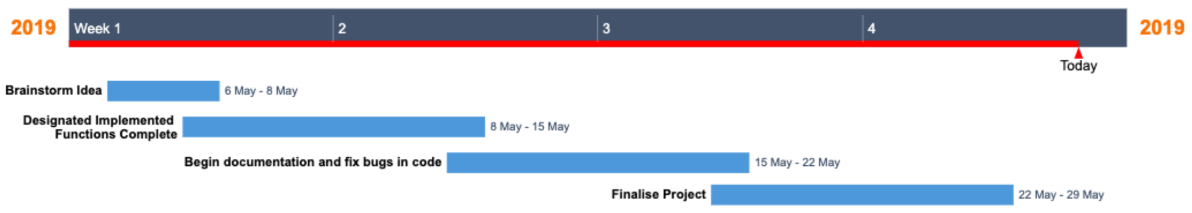
Whilst we didn't end up including compression in our final project, our group concluded that Huffman coding was the best method due to the efficiency of the algorithm to significantly reduce the file size of text files.

PROBLEMS ENCOUNTERED

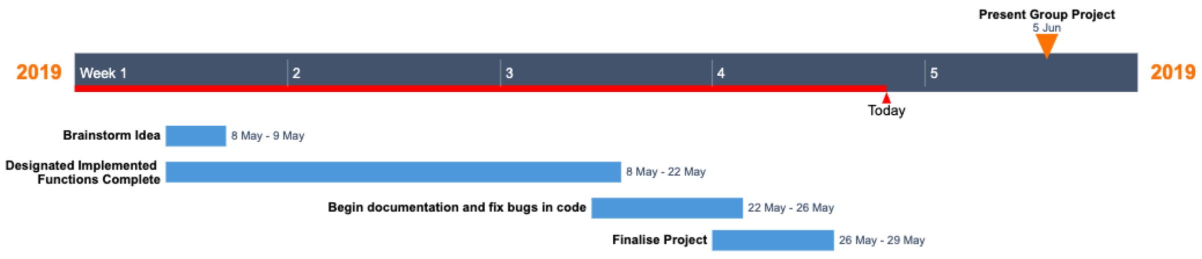
Time Management

Staying on track with implementing each function on time was an issue that reoccurred. Whilst our final program was finished on time, had we started coding earlier or prioritised it, we could have implemented more functions and expanded the scope of our project.

Expected Time Frame:



Actual Time Frame:



Communication

Communication with any group project can be difficult, we found this to be especially true for a group coding project. Since each group member has different coding styles, it was often difficult to decode someone else's function, especially as they grew more complex.

In addition, our group used GitHub to work on the project and we found it particularly perplexing when someone pushed through code with errors in it, meaning that we had to go through other functions to try and fix the errors. Whilst getting another coding professional to find errors can be a good debugging practice, it was often difficult and an inconvenience to fix errors that you were not expecting to occur. In saying this, it wasn't always clear whether someone else in the group had already fixed this error but hadn't pushed through their code yet, so this also got confusing at times.

Coding as a Group

Expanding on communication issues, coding in a group was a new process to a majority of our group members, meaning that it was important that all code was documented so that other members could understand what was being written. Thus, it was crucial that we all implemented good coding habits. Whilst brief explanations of each function were always provided, greater depth could have improved this process.

To work together as a group, we created a make file, so that we could all work on separate functions and document them separately after creating a function prototype template of the functions that we were all going to implement. This was a new form of coding for a majority of us and took some getting used to, a problem that 'scared' us away from the code for a bit as it was new and confusing. However, after playing around with the program and watching tutorials, this problem was overcome.

CONCLUSION

In conclusion, our group worked well together to overcome some major issues in this group project. We successfully created a program that allows event coordinator to create and edit and users/patrons to join them on a secure platform, everything that we had intended to in our problem statement. Whilst there were some minor edits that we could have made, we are happy with the overall functionality.

REFERENCES

University of Rhode Island, '*Classical Cryptography',* Viewed 28th May 2019, <https://www.cs.uri.edu/cryptography/classicalsubstitution.htm>

Wikipedia, 9th May 2019, '*Huffman Coding',* Viewed 26th May 2019, <https://en.wikipedia.org/wiki/Huffman_coding>