Coursework 2 Report

Steven Pyper
40319882@napier.ac.uk
Edinburgh Napier University – Web Tech (SET08101)

Contents

1. Introduction

The main scope of this coursework will be to build a website that allows users to create accounts and send messages to other users (or themselves) which will be stored in a database ciphered. This data will then be deciphered when it is being sent to the recipient allowing them to view it in plaintext. There will be two main sections, the server routing and setup and then there will be the database section. As I know my design skills are lacking just due to the way I think and see things I will be focusing on adding as much functionality as I can. I will implement a system that allows users to create an account, login, allow them to update their password if they are logged in and to logout manually. Additionally, I will allow them to send messages to themselves and to other people based on usernames and then view these messages on a different page. These messages will also allow for lots of different methods of ciphering(which since the other user will see it in plaintext anyway is more to allow the user to feel like they have a choice) the methods the user will be allowed to use are a Caesar cipher, a direct substitution cipher and Morse code. As an additional goal if there is extra time there will be an additional button which allows viewers to see their messages in ciphered text.

1.2 Server Side

The server side must allow the user to route around the website in a way that makes sense, it must allow them to login or register a account, then it should show them pages that they could only see if they were logged in, if they are not logged in they must be restricted in what they can(in reality there would be static pages such as the first coursework but these will be excluded as they will no longer fit with the website although if there is enough time they will be added in).once a user has logged in they will see a slightly different website, in theory it should be possible to change the navigation bar to then give them the links to see there messages, to send messages and to log out if they wish. The server will also have to deal with logged out users trying to access pages that they are not allowed, the simplest way to do this would be to redirect to the login page as it is likely they are a user if they have those links saved. The server will have to have a system to track which users are logged in and identify who each request comes from. This can be done with cookies which will be made unique to the user and to when the server was started.

1.3 Database

The database will have to contain a table that will keep track of users such as their username and password. Another table will keep track of messages showing who sent them, who they are for and what the message itself contains. This will allow the server to interact with the tables and join them in a way to get the correct information for the specific user. The user's password will eventually have to be stored in a encrypted way, so they are not stored in plaintext as this would be a massive security issue.

2. Software Design

The initial design for the server will be to simple allow users to register and login and send plaintext messages to each other. To do this I will need a database system in place, I have chosen to use sqlite3 as we used it In the lab and there is a lot of information online showing how to use it, also as I already know SQL this is another advantage as I don't need to learn a new language as well.

2.1 Database Interaction

There will be a lot of interaction with the database as most things will need to be verified before allowing the user to access anything. The database section will not contain information about encrypting the data as I have decided that is more of a server-side programming issue than a database side (even though the database will store the encrypted password). There will be two tables. A User table which will contain Usernames, Passwords and Cookies. This will allow a user to be validated. The second table will include a Message ID as there is no nice way to have a unique ID excluding a automatically incrementing number. it will also store the Sender of the message, the Recipient, the message itself, as well as a way to show which cipher was used to cipher the messages.

2.1.1 Register Account

When a user creates an account, they will enter a username and password that they would like to use for there account. Due to this there will have to be a check to make sure that the username is already free and not in use by another user. This will consist of a select statement which will look for that username, if a result is returned that means that the username is already in use. Therefor the user should be informed of this and asked to choose a different name. once they have a username that is not already in use a database command will run that inserts the username, the password and the cookie that will be used to keep them logged in.

2.1.2 Login to Account

Logging into an account will require the user to enter their username and password. This then will start a select statement looking for that username, if the username is found that means the user exists and then you can compare the password in the database and password that they entered. If the passwords match, they will be logged in. If the user isn't found or the passwords do not match the user will not be logged in. when a user successfully logs in the cookie in the database will be updated to the newest cookie.

2.1.3 Update Password

This should be fairly simple as the user will already be logged in, so the same statement that gets used to validate the user is logged in and the cookie they use will be used, then if they are validated a simple statement that will update the users row I the database with the new password can be used.

2.1.4 View Messages

This will be a slightly more complicated as the user is only allowed to see messages that is directed to them. This means

there will have to be a statement that can figure out what user is logged in and use them as the recipient and get every single message from them from the database. The best way to do this would be to either do a foreach or an all database statement, these will both work in very similar ways but will slightly change the code. using the join statement would allow for there to only have to be one search to the database to get this information back which is important as this is the most likely to be used functionality of the website and as such should be more efficient than the rest of the website. Once this information is read form the database it should be sorted so that the most recent messages are shown first, and I might limit them just as a way to keep it tidier looking for now.

2.1.5 Remove Messages

Allowing users to delete their messages is an additional feature that would be useful but not necessary. For this you would have to do a statement o validate the user that is logged in, then another statement to make sure that the message they are trying to remove is their own. Finally, you would then run the delete statement removing that message from the database.

2.1.6 Send Message

Again you will have to validate the user who is logged in as they will be the sender, then a statement will be required to make sure that the user they are trying to send the message too actually exists because sending messages to no one will take up space and give no benefit to the user. Finally, an insert statement will be used that will insert the sender, recipient, the message content that the user has entered, and anything required for the ciphered messages to be deciphered.

2.2 Server-Side Coding

2.2.1 Routing Logged Out Users

As logged out users will be very limited in what they could do on the website they will have very limited abilities on the website, all they should be allowed to access is the login and register page, this means that the navigation bar should be changed to only let them see these.

2.2.2 Routing Logged Out Users

Logged in users have a lot more choices and as such will have to be validated often whenever doing tasks that would require the user to be logged in. they should also be shown a different nav bar that will show them links to get to there messages, sending messages, updating there passwords and logging out.

2.2.3 Passwords And Cookies

As passwords should not be stored in plaintext they will instead be stored in the database using bcrypt, this will allow them to be encrypted before the database and never store a plaintext version, when a user tries to login the plaintext password they send will then be encrypted and that compared ot the one in the database. Once a user has successfully logged in they will be given a cookie which will contain the time the server started and a signature of there username which should help to keep them more unique and less guessable by malicious attacks. These cookies will be stored in the database when the user is logged in allowing them to be quickly compared to the database helping to speed up the validation of that user. Theses cookies will either last only for a day or until the server is restarted.

2.2.4 Ciphering/deciphering the Text

Ciphering the text will be more complicated due to the fact that I have multiple ciphers with different keys(as well as substitution technically having two keys) what this will mean is that when the user is selecting how they wish to cipher the text they will have to choose these ciphers and keys. Then server will have to deal with all them individually and make sure to only look for the correct information, which is dependent on ciphers and methods, then the text will be ciphered using a modified version of the ciphers that were implemented during the last coursework. This will return the text which can then be sent to the database. Deciphering the text should be relatively simple once the ciphers have been implemented as the keys will just be reversed.

The Ciphers will have to modified from the original coursework to allow them to return the text that they have converted. There will also need to be a good way of dealing with all of these ciphejrs, as such a middle ground method will be used that will be used to simplify what is passed into the ciphers, there will be one of these for each of the methods used, this will allow you to only have to pass in the type(encrypt/decript) the key/s and the content, this will mean that it will be able to deal with things for you as well as making it much easier to decipher the text from the database. Modifications to the database will also be required as there is a need to now store the method, the key/s and the cipher so that it can be deciphered later.

2.2.5 Logging in

The login page should only be visible to users who have not logged in and as such a redirect will be used to take them to the messages page. However if a use is actually trying to log in they will be given a pug fill which will have a form that will allow the user to create a post request to pass along there username and password.

2.2.6 Register

Registering will be similar to logging in, the user will be redirected if they are logged in and if they are not, they will be given a pug file that will allow them to enter their information but with an additional check to make surethat the account is not already in use.

2.3 Additional Features

An additional feature I would like to implement is an admin account which will be allowed to remove messages and users from the database. This should be simple as you would just do a check that the cookie lines up with the admin username in the table then delete statements depending on what is being removed. If a user is removed all messages that are sent to them should also be removed.

3. Actual Implementation

All the planned features have been implemented. Features: A user is allowed to create an account, sign in, update password, send messages, read messages sent to the, delete those messages and log out. As well as there is an admin account which has permission to delete any user account and any message based on the username or ID.
The quality of the visual design is of course lacking however I did once again try to get a website that did not look to plain yet was still very focused on giving the user as much functionality as I could. The user has a good experience with the website as it just works. There is no extras that they do not need that takes up space on the screen that would be wasted and give no additional benefit to the user.

See Appendix A – E for pictures of the website
I specifically aimed to keep the amount on information on the screen to a minimum as my last coursework had way to much information and was just looking at a screen of text, by aiming to keep the minimum on each page this keeps the screen much cleaner and easier to find exactly what you are looking for.
It also takes a very low number of clicks to be able to login and see your messages (as you are redirected to the message page once you have successfully logged in)

Users can send messages containing punctuation and the server should be able to deal with it in a way that does not cause a problem to the end users however there are still some situations that the server will crash if the user enters certain characters.

The server feels like it is responding very quickly (although it is local host) and in theory should be able to manage with a decent number of users running at the same time although very large amounts of users could lead to issues.

The server routes the users in a way that makes sense as to whether they are logged In or not, it tries to keep the amount the user has to do to a minimum to try and improve user experience. most of the time when a user does something wrong the server will tell them why. There is a lot of redundancy in the code especially around areas of security such as passwords

4. Critical Evaluation

4.1 Comparison to Requirements

I believe that my Coursework not only meets the Requirements set by the Descriptor but exceeds them in most cases. I would also like to make it clear again that I followed the original specification before we were allowed to use additional packages and I tried to follow this for the entire coursework even when we were allowed to use extras, although this may have led to some implementations not being perfect I am glad I did this as I believe it led me to have a much better understanding of what I was doing

4.1.1 The Client side

allows users to sign up for an account, write messages which are then encoded and saved in the database, which the recipient can then use to read those messages in plaintext as they will be deciphered, but it also allows them to update there own account password if they wish to change it as well as send messages to themselves not only other people. Also they are allowed to choice which cipher they used which at the point of writing may not specifically change a lot for the user but it would be very quick to implement a system that allows the user to get the message back still ciphered and manually decipher it as part of a game.(which if I implement will be included in part 6)

4.1.2 Server Side (+Database included and pugs)

The server-side deals with all of the user's requests laid out previously as well as storing all of the formation using a CRUD Approach. Registering is Create, Logging in is Retrieve, changing password is an update, and removing messages is delete. The server manages to route users in a way that I think is nice to use as it takes you to the page you are most likely to want to use quickly. I also do like the very plain visual design; it is a definite improvement over the last coursework although I do think I went to far the other way. With the aim of having the page be very clear and only have information that the user is likely to want it has led to a lot of the pages being very empty and not making use of space. Also the page that shows messages will have trouble dealing with having to many messages which is why I implemented an artificial limit of 20, this is not an effective solution as this means a lot of data stored on the database can never be viewed again as it has been more than 20 messages after them for that user.

4.1.3 Additional Features

I am very happy with the many additional features of the website as this was a design choice to focus on features over design as I know that is where my strengths lie. There is an administrative account which is allowed to remove users and messages in a very quick manner simply by giving there id or username, this system could still be improved but does give administrative control over the website. the website is also completely restricted to anyone who is not logged in, in reality there should have been static pages(such as the previous coursework) however those were left out as even though they would be very quick to add I felt they would not add a lot to the website(if early on I had implemented a system that allowed the user to see the ciphered text I would definitely have done this as it would allow them to go manually decipher it and see what it doing but it was an afterthought). Another additional feature that I felt was good implementing was the ability to change passwords, even though it is a relatively small addition I feel like it is still a nice feature to have implemented for the users sake as it means that they can change it any time they want as long as they can log in. Also another important feature was password encryption, this was a must as a security student as it is frustrating when you find websites that can send you your password back when you forget it as that means they have stored it unsafely, therefor storing passwords in the database using bycrypt with salt was a must, this allows the user to have peace of mind that if they have chosen a strong password(not one that can be easily brute forced) they should

have all of their message save unless someone can get there hands on the database. Another feature that I like is the ability to kick of all users form the server with a restart, as it changes the date key it means that any old cookies are immediately invalid and as such users will have to log in again.

4.2 Possible Improvements

There are a lot of improvements that could be made to the website specifically around design but also around making sure that the text that users enter has been cleared and will not cause problems to the database structure. This would have bneen easier to implement had I just used a additional node package such as sanitize however as I had already completed so much of the coursework to the original specification I wanted to finish it that way and use no additional packages outside of what we additionally allowed, in reality I should have just followed the newer rules as this would have allowed a better quality implementation of a lot of different areas. However I choose not to and still managed to implement a lot of features so there really wasn't to many issues.

4.2.1 Design Improvements

Some of the pages on the site feel empty, this was intended as a way to keep the pages feeling clean, but they are to empty now. This could have been improved by centring things more to the middle of the page, without adding content this would still have made it have a feeling of having less whitespace without there actually being any less. There is also a odd colour effect once a page expands a lot I believe this is due to the way I have done the gradient however I left it as it was because I think it looks somewhat interesting although I would have preferred had I been able to flip it around so that the pattern flowed smoother. The colour scheme works quite well with the blue background and white text however I feel like the font needs to be changed as sometimes it feels very small or to large. Also the forms that are used to allow the user to enter there information's are not lined up in any way and for having a page so clean to have them be unaligned is very frustrating even though it makes no change to the actual input having a different way to layout the page would have led to a definite improvement.

4.2.2 Functionality Improvements

Currently users are only allowed to look at the 20 most recent messages that they have received. This is an issue as if there is an important message that is older than 20 other messages for that recipient, they can no longer see that message, this is done to save the page from being covered with tons of messages. An improvement to this would be to allow the user to either see all messages and sort them by sender, or split the messages they receive into differentiate pages having a limit on how many per page, this way the user would be able to see every message they have in a much tidier way. Another improvement would be to allow the user to send messages to multiple recipients rather than only allowing one, this would have been fairly simple to implement but I did not think of it at the time of implementation. There is still an issue with not sanizing user input which is a major issue for something like this as specific use of punctuation could lead to a broken input which In theory may crash the server. This is something that should have been dealt with in a much better way however as I did not want ot use additional modules this led to me not really having time to implement a system to sanitise every user input. This could be a very quick fix with a node install of a sanise module.

An implementation that allows the user to view the messages still ciphered would have been a nice addition to the user experience, theoretically this would not be a large amount of code.

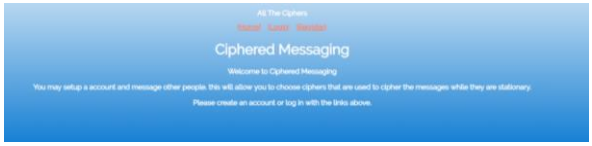
5 Personal Evaluation

There were many issues faced during this coursework as it felt like the starting point was very vague due to learning a lot of new systems but not having a lot of time to mess around with them before starting. Setting up the server and getting it to route properly was fairly easy but getting it to act how you want all the time was more difficult as it was fairly finicky. Having to convert the ciphers over to work with the server took some time but was fairly simple as the cipher was already created so it was just tweaking. The most difficult implementation was sqlite3. There are things that have to be done that that make no sense compared to the rest of it such as having to use `'${variablename}'` just to get it to behave as expected , this took the majority of the time of the coursework as it was just lots of little things such as this that took time. I did learn a lot specifically about routing users around sites as well as the getting the database stuff to work effectively keeping. Getting help from the internet in this coursework was much more difficult as a lot of the help didn't seem to work or be really vague or be for a different version of software which led to the best way ot learning being trial and error, try set something up and log what happened.

Overall, I think I did quite well on the coursework, again I am unhappy with the design although not as bad as the previous coursework as that really was awful. In terms of functionality I believe that this is where the website really shines, there is a lot of little quality of life features that did not have to be implemented but I wanted to implement anyway the website gives a lot of options for the user and I believe that it is decent quality for the coursework.

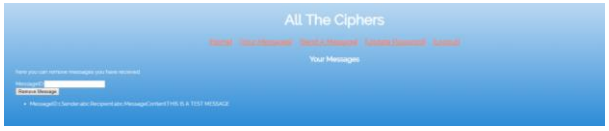
Appendix

Appendix A (home Page)



Appendix B (Login Page Similar to register Page)

Appendix C (User Messages)



Appendix D (Sending Messages)

Appendix E (Change Password)