# Final Report

Owen Maynard - 10770644

## Introduction

In this report you will find explanations on the micro service, explanations on the design and how the project was planned out, I will outline the legal, social, ethical, and professional issues of the project. Finally, I will discuss the implementation and evaluate the effectiveness of the final product.

[Github link](https://github.com/Zedster82/Comp-2001)

[Swagger API Link](https://web.socem.plymouth.ac.uk/COMP2001/OMaynard/Swagger/index.html)

## Background

The micro service that I have implemented is the profile service, it is used use CRUD procedures on the profiles of the user, while also managing things such as followers and favourite activities.

This will all be done via the use of a Swagger interface to show the structure of the API.

I have controllers for: Activities, ArchiveUsers, FavouriteActivities, Follow, FollowersCount, FollowingCount, Login, Logout and finally Users.

The template that I used to create the project is a C# ASP.NET Web Core API , the purpose of this is to provide CRUD interactions on the database data such as the Users via a web interface. The project has vertical scaling and could be scaled easily. Currently the micro service doesn’t interact with any other services however it is designed with the capability to if I were to create other services. It features good security measures such as encryption and authentication. Errors are handled to provide feedback messages to the users. It is well tested and has been deployed onto a hosting server. Documentation could be improved in the future.

## Design

A white paper with black text

Description automatically generatedLogical ERD:

A diagram of a software company

Description automatically generated with medium confidenceUML Diagram of the controllers and how they interact with the tables.

|  |  |  |  |
| --- | --- | --- | --- |
| Controller Name and Type | Responsibility | Input Data | Return Data |
| Users - Get | GET All users data |  | A Json object of all users |
| Users – Get{id] | Get specific users data | Integer id | A Json object of one user |
| User – Post{data…} | Create A user | All user data such as String username, String password and String email | A confirmation result on the success status |
| User – Put{data…} | Edit user data | The new data that you want instead of the old data such as String newEmail | A confirmation result on the success status |
| User – Delete{id} | Delete user | The integer id of the user to delete | A confirmation result on the success status |
| Login - Get | Responsible for logging the user in using the authenticator API. | String email  String password | A confirmation result if the login was successful |
| Activities - GET, Get{id}, POST, PUT, DELETE | Responsible for CRUD interactions for the activities data. | Input data such as String activity\_id and  String activity\_type | Get requests return a json object of activities and POST,PUT and DELETE return a confirmation result. |
| Followers – POST, DELETE, GET{id}//Follower Count,  GET{id}//Following Count | Interactions for the followers such as follow unfollow and get follower/following counts. | Integer id for following and unfollowing, uses logged in id Aswell.  Integer id for follower/following count too. | Integer for following/follower count. Confirmation result on follow/unfollow. |
| Favourite Activities – POST, DELETE, GET,GET{id} | Interactions for the favourite activities such as favourite an activity and un favouriting it. Also able to get a list of favourite activities from all or single user. | Integer activity\_id  Integer id – Get list from specific user. | POST and DELETE are confirmation results. GET and GET id is a json object with a list of activities. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Legal, Social, Ethical and Professional (LSEP)

My implementation makes sure to implement information security, privacy, integrity and preservation. The way that it does this is via encryption, and the permissions that are given to users to help protect unauthorised access to sensitive information.

Encryption helps with information security by ensuring that sensitive data remains confidential and secure from unauthorized access.

It helps with privacy in that it provides users with control over their data as it is well safeguarded, and the user doesn’t have to trust other people with their data as much.

It helps with integrity as it helps to prevent unauthorised modifications of data.

It helps with preservation as it is less sought after and is less likely to be modified via unauthorised modifications.

The method that I used to implement the encryption is by double-hashing the passwords that the user enters, along with a randomly generated salt, the hashed password and salt are then stored in the database instead of a plain text password. When the user logs in, the hashed password and salt are retrieved, using the new password with the salt, and checking if the result is the same. This means that the user has entered the correct password.

Another way that my implementation ensures security is via my implementation of session timers that expire if the user has not performed an action within 10 minutes, this helps to make sure that a user’s account is secure and cannot be used by other potential bad actors.

My data items are designed in a way that enforces integrity, privacy and security as the controllers have limited access to sensitive information and is designed in a way that maintains data integrity via good use of normalisation and best practices in database management.

I have also mitigated some of OWASP top 10 using methods described here.

Legal – My programs legal issue would be the data protection act as I am storing sensitive information such as email and passwords. I mitigate this by not using real data and by encrypting the passwords. Another one could be the disability act however my program is using swagger which is already compatible for users.

Social - The social issue in my program is that there is currently no implementation for other languages, however this could be easily done as swagger has features that allow you to switch between languages easily.

Ethical - While the disability act is supposed to be for legal reasons, it also becomes an ethical issue when the application does not conform to the standards needed for some users to properly use the application. This again would not be that much of an issue due to swagger already having some tools for accessibility.

Professional - The professional issue that is present in my application is that there are no design choices based on where the API is being used, for example a different country, this however is not that much of an issue as the API would just be used for endpoints and therefore does not really need design.

## Implementation

### General Information

All of the actions in the API check if the user is logged in first, otherwise they are not allowed to perform the action.

All of these also have links to the code.

### Users

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L28) – Get all users, returns the main view on the database with followers and favourite activities shown

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L81) – Gets a specific user from main view

[POST](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L133) – Creates a user with the inputs provided, can be used without logging in

[PUT](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L192) – Edits the current logged in user, only admin can set usertype to admin

[DELETE](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L275) – Deletes (Archives) a user, admin only

### Login/Logout

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LoginController.cs#L25) – Logs in the user using the auth api and the database with the passwords

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LogoutController.cs#L14) – Logout, clears all login data

### Archive Users

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ArchiveUsersController.cs#L29) – Get all archive users, admin only

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ArchiveUsersController.cs#L79) – Get a specific archived user, admin only

### Follower/Counts

[POST](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FollowController.cs#L73) – Follow user

[DELETE](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FollowController.cs#L73) – Unfollow user

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/b0745e5c8a243dce34a1f5835f75aa4967b03c19/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FollowersCountController.cs#L88) – Get Follower count of a user

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FollowingCountController.cs#L90) – Get Following count of a user

### Activities

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ActivitiesController.cs#L28C35-L28C35) – Get a list of all activities

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ActivitiesController.cs#L81) – Get a specific activity

[POST](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/2e999949a792c581d12b4ad3c5e8e4004d5473c4/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ActivitiesController.cs#L134) – Create an activity, admin only

[PUT](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ActivitiesController.cs#L191) – Edit activity, admin only

[DELETE](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/ActivitiesController.cs#L248) – Delete activity, admin only

### FavouriteActivities

[GET](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FavouriteActivitiesController.cs#L28) – Get a list of all users and their favourite activities

[GET{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FavouriteActivitiesController.cs#L85) – Gets a list of all favourite activities from a specific user.

[POST{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FavouriteActivitiesController.cs#L136) – Favourites an activity

[DELETE{id}](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/FavouriteActivitiesController.cs#L191) – Unfavourite an activity

Logout timer after 10 mins – On login, the time is noted down +10 mins [here](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/177aa0bdb92656398626a85e75bfb78eeef258ba/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Login.cs#L66C17-L66C65).

Then, every time a user performs an action [this](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/177aa0bdb92656398626a85e75bfb78eeef258ba/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Login.cs#L28C9-L39C11) function is called to check if the session is expired. This returns true if the session is expired, if the session is not expired then the session time is reset, and it returns false.

This result is then used to check whether the function needs to be cancelled early, here is an example of the code used for that: [code](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/177aa0bdb92656398626a85e75bfb78eeef258ba/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/UsersController.cs#L35C13-L39C14)

Implementation of the encrypting of a password and the login into the authentication API.

First the authentication api is contacted via [this](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LoginController.cs#L28C13-L28C46) function from [this](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Login.cs#L40) class, which Is a simple http request using the email and password, if the result is true then we log in and set the variable Login.isLoggedIn to true, this is then used in the login controller to see if the login was successful.

After this we log into the database, this is done by getting the hashed password and salt from the api, code [here](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LoginController.cs#L41). The code then hashes the login password that has been entered, using the salt from the database too, it then [compares the result](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LoginController.cs#L68C29-L77C30) with the hashed password from the database. If they are the same then the user is allowed to login and the [rest of the data from the database is stored](https://github.com/Zedster82/Comp-2001-Owen-Maynard/blob/3315cbf93abc8ea5c95f6801771712fd9a720152/Assignments/Final%20Report/Comp%202001%20API/Comp%202001%20API/Controllers/LoginController.cs#L61C29-L65C94).

### New SQL Stored Procedures

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## Evaluation

An improvement that could be made is that instead of using double hashing to help remove clusters and reduce collisions I could have instead used uniform probing which is asymptotically equivalent to double hashing. [[1]](#_References) This would have an improvement that means the computational requirements would be a lot less.

Another thing that would be a massive improvement to the project would be to implement good documentation in the API, this would help people that want to use the endpoints be able to understand the data types and how to interact with the API a lot better.

I also could have created a global function to contact the database instead of using repeating code. However, this is not that important as different functions require different handling of the data and therefore the code is not as repetitive as it seems.

Finally, I could heavily improve on fault tolerance, the reason for this is that my program doesn’t handle errors in a perfect manor and could be improved to be clearer in what went wrong.

I think that despite the clear improvements that could be made the project is overall well implemented and is incredibly functional for the requirements needed.

### Testing

I have created a testing table to ensure thorough testing of the API.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What is being tested | Inputs | Result Expected | Actual Result | Changes/ Fixes |
| Login Controller - GET | Incorrect email and password | Does not login to user and does not allow user to perform other actions | Expected Result | N/A |
| Login Controller - GET | Correct email and password | Logs into system and allows user to perform other actions. | Expected Result | N/A |
| Login Controller – POST{data…} | New user data such as email password and username | User is successfully created into the database. | Expected Result | N/A |
| Login Controller – PUT{id…} | Changed user data such as newUsername, using the user\_id to edit | User data is successfully edited in the database. | Expected Result | N/A |
| Login Controller – DELETE{id} | Id of user to delete | Archive the user into the archive tables | Expected Result |  |
| User Controller- GET and GET{id} | Valid id input | Returns a JSON object of the users in the database | Returns a JSON object of the users in the database | N/A |
| Activity Controller – GET and GET{id} | Valid id input | Returns a JSON object of the activities in the database | Expected Result | N/A |
| Activity Controller – POST | String activityType | Creates new activity in the table | Expected Result | N/A |
| Activity Controller – PUT | Int activity\_id  String activityType | Edits the activity type | Expected Result | N/A |
| Activity Controller – DELETE | Int activity\_id | Deletes the activity | Expected Result | N/A |

#### Login

A screenshot of a computer

Description automatically generated

#### Logout

#### Get All Users

A screenshot of a computer

Description automatically generated

#### Get Specific User

A screenshot of a computer

Description automatically generated

#### Post User (Create)

A screenshot of a computer

Description automatically generated

New user is now present in database: A screenshot of a computer

Description automatically generated

#### Edit User

Before editing:

A screenshot of a computer

Description automatically generated

Editing Data:

A screenshot of a computer

Description automatically generated

After editing:

A screenshot of a computer

Description automatically generated

#### Delete User (Archive User)

Logged in as grace we are going to delete pat now.

A screenshot of a computer

Description automatically generated

 A screenshot of a computer

Description automatically generated

 A screenshot of a computer

Description automatically generated

#### Follow

A screenshot of a computer

Description automatically generated

D A screenshot of a computer

Description automatically generated

 A screenshot of a computer

Description automatically generated

#### Unfollow

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## References

### 1.

*Leo J. Guibas and Endre Szemeredi. 1976. The analysis of double hashing(Extended Abstract). In Proceedings of the eighth annual ACM symposium on Theory of computing (STOC '76). Association for Computing Machinery, New York, NY, USA, 187–191. https://doi.org/10.1145/800113.803647*