AberLink - Creating a link between University and Discord accounts

CS39440 Major Project Report

Author: Joel Adams (joa38@aber.ac.uk)
Supervisor: Dr. Neal Snooke (nns@aber.ac.uk)

31st March 2021
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science and Artificial Intelligence (GG4R)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....Joel Luca Adams.....

Date .....

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....Joel Luca Adams.....

Date .....

Acknowledgements

I'd like to thank Dan Monaghan for giving me the original inspiration for this project and for helping me along the way with understanding key concepts.

I am grateful to have been working along side Dr. Neal Snooke for this project. Over the last year we have worked collaboratively on multiple Discord bots and improved the Uni's online Discord presence. I'd also like to thank him for allowing me to re-create and use the Discord bots that he created and have been the main motivation behind this project.

I'd like to thank Alun Jones for his continued support throughout this project and his resourcefulness in providing instructions on how to setup and use specific uni services.

I'd like to thank Leslie Johns for having setup an API endpoint that has been crucial in getting this project off the ground. He was very fast at setting this up and provided good documentation along with testing links which have helped in the testing section of this document.

Abstract

AberLink contains 2 components; this website which is used to link up Discord [1] accounts to students' accounts and a Discord bot called AberLink that is used for verifying students' Discord accounts. This bot is also responsible for marking students' attendance during practicals that have the flag of "Discord" on their student record.

This system was developed on the Linux distribution Debian 10 (Buster) [2]. It uses the open source HTTP server Apache2.0 [3] to web-host along with the Python framework Django [4] for the website. The website is locked behind the Open-ID Connect [5] login system that only authenticates users with Aber accounts. Once logged in a user can link multiple Discord accounts to their Aber account.

Information is saved to a back-end PostgreSQL [6] database using foreign keys to link the uni and Discord accounts together. A Discord bot (AberLink) using the python library Discord.py [7] then verifies Discord accounts by checking if that account is present in the database and then grant users access to the Discord server. AberLink also has an API endpoint for marking attendance on Student Record and uses the database to perform a reverse search on Discord users to get their uni information and send that to the endpoint.

The AberLink Discord bot is based on two previous bots called Aber Verify Bot [8] and I am here! [9]. These bots verified students' Discord accounts and helped to mark attendance during practicals respectively but have been greatly modified.

Contents

List of Figures

List of Tables

Chapter 1

Background & Objectives

1.1 Background

My main motivation behind this project was that I wanted to create another Discord bot like DemoHelper [10] that aids the university with their online presence. The main idea behind this project was however from Dan Monaghan when we were discussing a system to link student accounts up to Discord accounts. For this project I also wanted it to be more of a challenge so I decided to build a website where users could login and save their information to create this link. As I had very little experience working with websites and frameworks they were the primary focus of my research.

Please see the dev folder of my GitLab repository for spike work I completed at the beginning of the project. Below are subsections breaking down my choices and decisions I made with choosing parts of the system.

Note: This project is hopefully going to be deployed in the department so some choices have been made to accommodate ease of use for staff.

1.1.1 Web Hosting & Containers

When considering web hosting I decided to rely on the containers provided by the university along with Apache2 [3] mainly due to their convenience and ease of use. I did consider using Nginx instead of Apache2 but it didn't provide me with the level of documentation I wanted and there were far less tutorials on using the it.

1.1.2 Coding Languages

Over the summer of 2020 a new Discord bot appeared called DemoHelper [10] that was used to create and maintain a simple queue system in university practicals on Discord. Soon after it's initial creation I became involved in the project and learnt how to write and create my own Discord bots in Python based off of it that I slowly developed over the year.

This lead to me having solid foundation of Python and is the reason behind me choosing this as my primary coding language this project. It also had a knock on effect for choosing the database and web framework that worked well or used Python. I also used HTML, CSS and some Javascript to develop the website pages.

1.1.3 Database

PostgreSQL [6] was used for the back-end database due to it's flexibility and use of relational databases that I have previously worked with. It was also chosen due to it's support in Python and working with the website framework. I have considered other options such as MSSQL or NoSQL but settled on PostgreSQL as the university already is familiar with it and has database hosting already setup.

1.1.4 Website Frameworks

For the website building I went through a few different popular options such as Flask but decided against using them as they didn't scale well for my project's scope. Django [4] was the perfect framework as it has vast documentation, many tutorials so easy to learn and built in support for user handling or custom models. It is also useful as it has a prebuilt Admin page that allows admins to view all of the user accounts that are logged in so saves me the hassle of designing more webpages.

1.1.5 Discord bot

The Discord bot was an easy choice as I had already developed several bots using the Discord.py [7] framework and decided to stick with it.

1.2 Analysis

1.2.1 Objectives

After completing the spike work and prior research the next step was breaking the project down into sections and deliverables. Below is a list of the items that were used as milestones for the project.

- **Research & Discussion with IS/CS-support.**
  - Discuss how to access uni data and how to sign users in who are only on the uni network using OpenID Connect [5].
  - How to build the attendance API so that it is secure and only marks students for current practicals.

  - Setting up PSQL [6] and how to make it secure from outside attacks.

- **Version control, documentation and setup.**
  - Create a container on the uni network to remain secure and locked under VPN access.
  - Documentation and version control using git on the university's GitLab upon the departments request so that it can be easily redeployed as a complete service later on.
  - A blog (usually bi-weekly) to document the process and progress of this project.
- **Creation of Python back-end for website and the database.**
  - Build the website using the Python framework Django [4].
  - Establish a PSQL [6] database for data storage.
- **Re-writing 'AberVerify' and 'I am here' into single Discord bot.**
  - Recreate the two mentioned bots in Python instead of JavaScript.
  - Make the bot use a Discord users information to lookup their Aber ID in the database using relational keys.
  - Make the bot update attendance using the uni provided API endpoint.
- **Interface for lecturers and students on website.**
  - Create webpages for users to add Discord accounts, view the module servers they are in and the one's that they aren't.
  - Create Admin pages only visible to staff to view all the connected students, remove them and configure their roles in servers for which they have administrative privileges.
- **Resource links and further webpages.**
  - Create a Discord bot function to get information such as the Aber account that is linked to the Discord account.
  - Discord bot function to display other useful discord bots that can be added to the server. e.g. DemoHelper [10]
  - Create webpages to display information such as the 'privacy policy', 'ethics form', 'blog' and 'about this project'.
- **Further potential work**
  - Integrate DemoHelper Discord bot into AberLink
  - Add multiple language support

1.2.2 Possible Security Issues

Throughout this project I have attempted to minimise the amount of security risks that can occur. The main identified weakpoints are as follows:

- **Direct Database Access** - This is the first point of attack that could be used to get access to Discord and Aber account details. This has been secured by using a PostgreSQL [6] database that is located in Departments servers so it is already behind a very secure and up to date firewall. To access the database you also need to have a registered Aber account and be using the campus WiFi or the uni's VPN so that adds another layer of security. If the user however has access to both of these then brute forcing the PSQL login system is difficult as it has many security levels and incoming connections are monitored by the university.
- **Unauthorized Admin Access** - This is the possibility that the user will try and brute force access to the admin page of the website. Before they attempt this they would need to get past the website's OpenID Connect [5] authentication system that requires an Aberystwyth account to authenticate. The website also has another layer of security as it requires the user to be connected to the campus' internet or be logged in on the VPN. If they get past both methods there is no easy way to spoof the system to gain access to the administrator panel as the backend uses cookies to keep the user authenticated.
- **Accessing Database Credentials through Back-end Code** - All database credentials have been hidden in files that are not stored in the git repository and are loaded from JSON and .env files. This is good practice and creates a simple way for maintainers to setup and change variables such as database passwords easily instead of going through source code and manually editing it.
- **Accessing Data through AberLink Discord Bot** - The Discord bot doesn't contain any sensitive data, it merely queries the database using generic queries that are changed depending on the input variable. It can however be used to gain an understanding of the database model that is used.
- **Worst Case Scenario** - If the user someone gains unauthorized access to the data they will only be collecting a list of emails and linked Discord accounts. No password data is ever stored in the database because OpenID Connect [5] is used to authenticate Aber accounts and Discord accounts are linked using OAuth2 [11]. This means that no password data is ever exchanged with the database or website.

1.3 Process

You need to describe briefly the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should discuss any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don’t see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find a section addressing questions such as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Annotated Bibliography

[1] Discord Inc., "Discord," accessed January 2021. [Online]. Available: https://discord.com/

Discord is a voice, video and text communication platform.

[2] Debian, "Debian 10 (Buster)," accessed April 2021. [Online]. Available: https://www.debian.org/releases/buster/

Debian 10 (Buster) is a Linux Distro and is used throughout this project

[3] Apache Software Foundation, "Apache 2.0," accessed April 2021. [Online]. Available: https://httpd.apache.org/

Apache2.0 is a open source linux package for website hosting

[4] Django Software Foundation, "Django," accessed April 2021. [Online]. Available: https://www.djangoproject.com/

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

[5] OpenID Foundation, "OpenID Connect," accessed April 2021. [Online]. Available: https://openid.net/connect/

OpenID connect is a identity layer on top of OAuth 2.0 and is responsible for simple user verification. It provides a basic profile of the connected user to the client in a REST-like manner.

[6] PostgreSQL Global Development Group, "PostgreSQL," accessed April 2021. [Online]. Available: https://www.postgresql.org/

PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

[7] Danny (Rapptz) and Various, "discord.py," accessed January 2021. [Online]. Available: https://github.com/Rapptz/discord.py

Python API library for Discord currently available through Github.

[8] N. Snooke, "AberVerify," Sept. 2020, accessed April 2021. [Online]. Available: https://github.com/NealSnooke/Aber-Verify-Discord-Bot

Discord bot for Aberystwyth user verification in discord servers using email accounts (e.g. joa38@aber.ac.uk) and discord IDs (e.g. Joelin-Rome#4893). These accounts are then linked together using JSON objects. This project is currently available through Github.

[9] ——, "I am here," Sept. 2020, accessed April 2021. [Online]. Available: https://github.com/NealSnooke/IAmHere-Discord-Bot

Discord bot for marking Aberystwyth students' attendance during a practical. A student writes a message and the bot adds them to a list of students that have attended the practical and emails the lecturer at the end. This project is currently available through Github.

[10] Joel Adams, Nathan Williams, "Demohelper," accessed April 2021. [Online]. Available: https://github.com/AberDiscordBotsTeam/demoHelperBot

DemoHelper is a Discord bot created to manage demonstrating online for students on Discord.

[11] Blaine Cook, "OAuth 2.0," accessed April 2021. [Online]. Available: https://oauth.net/2/

OAuth 2.0 is the industry-standard protocol for authorization.

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

Appendix C

Code Examples

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Project Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
```

```
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
```

```
    *idum += IM1;
  }
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```