



Cardiff University

School of Computer Science and Informatics

CM3203 - Individual Project Report

Cyber Security Education Portal

Supervisor: Dr Yulia Cherdantseva

Author: Alexander Jones

13/05/2022

Contents

CM3203 - Individual Project Report.....	1
Table of figures	3
1. Introduction	6
1.1 Motivation.....	6
1.2 Project scope.....	6
1.3 Project aims and objectives	7
Objective 1 – Review relevant services/literature.....	7
Objective 2 – Create and optimize a database suitable for the project	7
Objective 3 – Create a user-friendly UI.....	7
Objective 4 – Implement back-end functionality	7
Objective 5 – Successfully evaluate the prototype.....	7
Objective 6 – Create an implementation that is secure and complies with most modern security standards	7
Personal objectives	8
1.4 Key concepts	8
DoS and DDoS Attacks	8
DDoS Mitigation	9
SQL Injection	9
Defending against SQL injection	11
Cross-site scripting (XSS).....	12
Defending against XSS attacks	12
Input validation	12
Content Security Policy	12
1.5 Relevant services and standards.....	12
BBC Bitesize.....	13
Brainpop.....	19
Relevant standards and regulations	22
User personas	23
Persona 1	24
Persona 2	24
1.6 Approach.....	24
Waterfall methodology	25
Agile development methodology.....	26
Evaluation method.....	27
Achieving research aims and objectives	28

1.7 System design	28
Non-functional User Requirements	28
Functional User Requirements	29
Use cases.....	32
Low fidelity UI prototype	39
System Architecture Diagram	45
Frontend technologies	45
Backend technologies	45
1.8 Implementation	48
Role-based login and registration system.....	49
Security configuration and role-based permissions	54
Homepage.....	56
Uploading and viewing content	59
HTTPS Security	61
Overview of security features implemented:	61
2.1 Evaluation	62
Verifying HTTPS connection.....	72
Ensuring resistance against SQL injection.....	73
Verifying resistance to cross-site scripting (XSS) attacks.....	74
Achievement of deliverables and objectives	74
Deviations from the initial plan	75
Project limitations	76
Future work.....	76
Conclusion.....	77
Reflection	77
References	78

Table of figures

Figure 1: A diagram showing the basic concept of a DDoS attack [8]	8
Figure 2: Example SQL Query	10
Figure 3: Example injection statement	10
Figure 4: SQL Statement as a result of injection	10
Figure 5: Example of parameterized statements [9]	11
Figure 6: Explicit construction of SQL statement using string concatenation [9].....	11
Figure 7: Homepage of BBC bitesize	13

Figure 8: BBC bitesize page displayed after selecting education level	14
Figure 9: BBC bitesize subject selection page.....	15
Figure 10: BBC bitesize exam specification selection	16
Figure 11: BBC bitesize computer science topic selection.....	17
Figure 12: BBC bitesize sub-topic content	18
Figure 13: Brainpop homepage.....	19
Figure 14: Page displayed after selecting topic	20
Figure 15: Waterfall methodology [15]	25
Figure 16: Agile methodology flow [16].....	26
Figure 17: Prototype login page.....	39
Figure 18: Prototype sign up page	40
Figure 19: Homepage prototype.....	40
Figure 20: Admin view homepage prototype	41
Figure 21: Add new education level form prototype.....	42
Figure 22: Content selection prototype	42
Figure 23: Upload content form prototype	43
Figure 24: Content view prototype.....	43
Figure 25: Admin view prototype	44
Figure 26: System Architecture Diagram	45
Figure 27: Spring boot flow diagram [13]	46
Figure 28: Entity relationship diagram.....	48
Figure 29: Database connection in application.properties.....	50
Figure 30: User entity.....	50
Figure 31: Regex used to identify valid email address.....	51
Figure 32: User repository	51
Figure 33: User Service interface	52
Figure 34: Dependency injection within the UserServiceImpl class	52
Figure 35: User save method	53
Figure 36: Log in implementation	53
Figure 37: Thymeleaf to bind input with object	53
Figure 38: UserRegistrationController snippet	54
Figure 39: Web security annotation	54
Figure 40: HTTP configuration	55
Figure 41: Configuring access permissions based on role in SecurityConfiguration.java.....	55
Figure 42: Homepage.....	56
Figure 43: EducationLevelService.java.....	56
Figure 44: GetEducationLevelById method in EducationLevelServiceImpl.java.....	56
Figure 45: Delete and save education level methods.....	57
Figure 46: getContent() method within EducationLevelController.java.....	57
Figure 47: Thymeleaf for loop to display education levels	57
Figure 48: Controller responsible for displaying the correct content	58
Figure 49: Delete method within the education level controller	58
Figure 50: Show update form method.....	58
Figure 51: Update education level post method	59
Figure 52: Save content controller method	59
Figure 53: Thymeleaf snippet displaying content	60
Figure 54: Command to generate keystore	61
Figure 55: HTTPS configuration in application properties	61

Figure 56: PowerShell command used to verify SSL connection	72
Figure 57: Confirmation that SSL connection is active	73
Figure 58: Browser dev tools showing secure connection	73
Figure 59: Log in system rejecting SQL injection attempt	74
Figure 60: Script inserted into search box	74

1. Introduction

Society as we know it, revolves around technology, the technology industry is set to exceed \$5.3 trillion (about \$16,000 per person in the US) in 2022 [1]. People of all ages have access to various technology, whether it be smartphones, laptops or wearable tech such as smartwatches, these are all common in the average household. A staggering 6.648 billion people (83.89% of the population) have access to a smartphone. A study in 2019 showed that 49% of children in the UK aged between 8 and 11, owned a tablet with internet access [2]. As technology becomes cheaper, it will become more accessible and result in more users. Each user will have a different level of understanding of technology. Some more advanced users may use their increased knowledge of technology to take advantage of those with less knowledge. Cybercrime is up 600% since COVID [3] and is not showing any signs of slowing down. The total number of malware infections has been increasing for the last 10 years [3], in 2018 812.67 million devices were infected with malware [3], this is potentially hundreds of millions of people who suffered as a result of cybercrime. This is not helped by the fact that the world is currently experiencing a cyber security skill shortage, 64% of cyber firms have faced problems with technical cyber security skill gaps, either among existing staff or among job applicants [4].

1.1 Motivation

As technology advances and grows, we will share our data with more services. Attackers want access to our data and will often break the law and employ unethical techniques in order to gain access to it. As a result, we rely on cybersecurity professionals to implement secure methods to protect our data from these attackers. However, the problem lies within the fact that the demand for cyber security specialists is much higher than the supply. 51% of employers find it difficult to fill cyber roles [4], and this is in line with the global shortfall of 3.5 million cyber security jobs in 2021 [5]. The lack of cyber security specialists means that our data is at risk, if current security protocols cannot be updated and maintained at a fast enough rate to protect against the latest threats then the consequences would be devastating. For example, one of the most recent and major vulnerabilities discovered was within a service known as Log4J, the vulnerability meant that attackers could break into systems, steal passwords and infect other networks with malicious software. Fortunately, this was patched before any major damage could be done. But with the ever-increasing number of cyber-attacks and the shortage of cyber security experts, it is only a matter of time before a vulnerability gets exploited before cyber security specialists are able to patch it. It is therefore absolutely critical that the cyber security skill shortage is addressed as swiftly as possible. I believe that an online cyber security education portal would assist in increasing the amount of qualified cyber security specialists. The portal would contain material suitable for all ages and proficiency levels, this means even young children could utilize the portal. Being exposed to cybersecurity material at a young age could spark an interest which would later lead to a potential role in the cyber security industry. While we may not see an impact on the number of cyber security specialists immediately, it may still reduce the impact of cybercrimes. For example, phishing is the most common cybercrime [6] and involves tricking the victim into revealing sensitive information in which the attacker can then use for malicious purposes. These attacks are usually only successful if the victim is not cyber aware, therefore with the appropriate cyber education, the victim would be able to identify that it is a phishing attack.

1.2 Project scope

This project is concerned with creating a cybersecurity education portal that aims to reduce the cyber security skill gap that is present throughout the world, while also helping to reduce the

effectiveness of cyber-attacks by educating all age groups. The project will also be concerned with ensuring that all data will be processed in a GDPR compliant manner and that basic security methods are employed throughout the system. Cyber security material will not be created during this project, only the foundations of the web application will be implemented, it is up to the users to create and upload material for others to view. However, there may be cases where I deem it necessary to create a basic form of example material.

1.3 Project aims and objectives

The overall aim of this project is to implement an online cyber security education portal which will allow users to upload various types of content with the goal of educating others. The implementation should prove to be effective in reducing the cyber security skill shortage as well as decreasing the effectiveness of common cyber security attacks such as phishing. In order to achieve this, I have broken the project down into smaller goals.

Objective 1 – Review relevant services/literature

- Determine the strengths of relevant services and incorporate them into my own designs/implementation
- Identify the weaknesses of relevant services and determine how I can improve upon them
- Establish requirements for my own implementation based on relevant literature/services

Objective 2 – Create and optimize a database suitable for the project

- Based on functional requirements, optimize the database to allow the quickest possible query execution time

Objective 3 – Create a user-friendly UI

- Use inspiration from other services to determine UI design patterns
- Using these design patterns, sketch an initial wireframe of the UI
- Iteratively increase the fidelity of the UI sketches

Objective 4 – Implement back-end functionality

- Link the database with the web framework (hibernate+ spring boot)
- Use java in conjunction with thymeleaf + spring boot to add back-end functionality

Objective 5 – Successfully evaluate the prototype

- Create test cases
- Carry out test cases
- Establish if and why test cases failed
- Implement a fix for the failed test cases

Objective 6 – Create an implementation that is secure and complies with most modern security standards

- Identify cybersecurity vulnerabilities
- Review how to minimize these vulnerabilities
- Implement methods to mitigate the risk of cyber attacks

Personal objectives

The purpose of this project is to not only develop and implement a web application to combat the cyber security specialist shortage, but also to develop my own skills and knowledge of computer science/cyber security. I have compiled a list of personal objectives that I would like to complete during the course of this project.

- Familiarize myself with the most common cyber-attack methods on websites
- Learn to employ methods to defend against these cyber attacks
- Increase my proficiency in Java
- Improve my knowledge of Java web applications including web frameworks such as spring boot
- Gain experience developing and optimizing a MySQL database
- Gain a complete understanding of the evaluation procedure

1.4 Key concepts

It is important that this web application is resistant to some of the most common cyber-attacks. I shall go into detail explaining how these cyber-attacks work, as well as providing potential solutions to mitigate the risk of a cyber-attack occurring.

DoS and DDoS Attacks

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks focus on overwhelming a server with UDP and TCP packets. Due to the huge volume of packets, a server becomes overloaded and is unable to execute any processes. As a result, all services offered by that server become unavailable and the impact on the business can be substantial. Security surveys indicate that a DDoS attack costs a business between \$20,000 and \$40,000 an hour [7]. A DoS and DDoS attack both involve overwhelming a server with packets, however during a DoS attack, the packets are sent from one computer, whereas during a DDoS attack, multiple systems target the network simultaneously. Figure 1 shows how an attacker can control multiple systems and use these to all flood the same target device with packets.

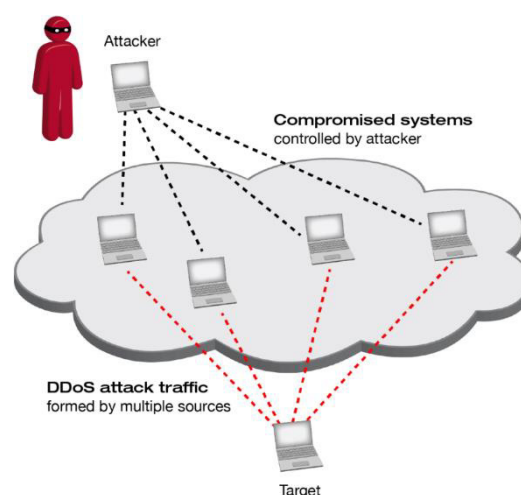


Figure 1: A diagram showing the basic concept of a DDoS attack [8]

There are multiple ways that a DoS attack can be used, the most common being the buffer overflow attack [8].

Buffer overflow attack – This attack involves overloading a network address with traffic. Each device has a unique network address, and this is used to determine the target device

ICMP Flood – The ICMP (Internet Control Message Protocol) is a protocol that network devices use to generate error messages when packets have not been delivered successfully. During an ICMP flood attack, an attacker uses unconfigured network devices to send spoof packets and pings every device within the target network.

Teardrop attack – A teardrop attack involves an attacker sending IP packet fragments with an incorrect offset value to a network. The network then attempts to recompile the fragments, however due to the incorrect offset values, this task is impossible and the server exhausts itself during the process.

SYN Flood – An SYN flood exploits the handshake process of a TCP connection, it involves sending a large volume of SYN packets to a server, the server responds to each connection request by sending a SYN/ACK packet back to each request. The server leaves an open port ready to receive the ACK packet (which never arrives). The attacker then continues to send more SYN packets, each arrival causes the server to open a new port connection, once all available ports have been utilized, the server is unable to function and the attack is complete.

DDoS Mitigation

Web applications can be protected from DDoS attacks by deploying a DDoS protection service such as Cloudflare. Cloudflare is a cloud-based provider which acts as a firewall for a web application. To mitigate a DDoS attack, the process is separated into four distinct stages

Detection – A large part of DDoS mitigation is identifying and distinguishing malicious packets from legitimate visitors. To identify a DDoS attack, packets are sampled, and this includes data such as the packet fields (source IP, source port etc), HTTP request metadata and HTTP response metrics such as error codes. Cloudflare uses rules to determine if a DDoS attack is occurring, if the attributes of the data collected during sampling matches a specific rule, the rule generates a unique signature. This signature corresponds to a specific type of DDoS attack and the relevant processes are then executed.

Response – After the system has detected a DDoS attack, the network drops confirmed bot traffic, while still allowing the legitimate traffic to be absorbed. This is known as ‘out of path’, which allows Cloudflare to asynchronously detect and mitigate attacks without impacting performance.

Routing – Routing the traffic separates traffic into smaller, more manageable portions. This can aid in preventing denial of service and maintains full functionality.

Adaptation – The network then analyses the traffic for patterns such as IP addresses that are repeatedly involved in DDoS attacks. The ability to adapt to these attack patterns can help improve its security against future attacks.

SQL Injection

The main idea behind SQL injections is to execute unauthorised statements on the database. This is achieved by inserting statements into forms or URL's. The effects could be devastating, if a DROP TABLE statement is successfully injected, then a large amount of data could potentially be lost if there are no up to date backups available. For example, there could be a form that is expected to take a user ID as an input from a user and then execute the query in figure 2.

```
SELECT * FROM users WHERE id = '23'
```

Figure 2: Example SQL Query

This works as intended if the user inputs their ID in the form, however during an SQL injection attack, a user will input an SQL statement in the form. For example, figure 3 shows a statement that an attacker could inject.

```
'23' OR '1'='1'
```

Figure 3: Example injection statement

This injected statement, would result in the perfectly valid statement shown in figure 4.

```
SELECT * FROM users WHERE id = '23' OR '1'='1';
```

Figure 4: SQL Statement as a result of injection

Since $1 = 1$ is always true, the inputted user ID does not have to be valid, this statement will return every data field for all users in the database. This is just one example of an SQL injection attack, different forms will execute different queries, giving the attacker more options when it comes to injecting SQL statements. There are three main categories of SQL injections, each with sub-types of their own.

In-band SQL injection:

In an in-band SQL injection attack, the attacker uses the same channel of communication for both launching attacks and gathering results. There are two sub-types of in-band SQL injection.

- Union-based SQL injection
 - o The attacker utilizes the UNION SQL operator which combines multiple SELECT statements to retrieve a single HTTP response. The response can contain data that the attacker can further utilize for more attacks.
- Error-based SQL injection
 - o The attack injects statements that cause the database to produce error messages, these error messages can contain important information about the structure of the database. This information can be leveraged by the attacker to create more attacks.

Inferential SQL injection:

In inferential SQL injection, to gain an understanding of the structure of the server, an attack will send data to the server and observe the response and behavior.

- Time-based
 - o An attacker sends an SQL query to the database using the method above, this makes the database wait before it can react. An attacker can determine whether a query is true or false based on the time it took for the database to respond.
- Boolean
 - o The attacker will inject an SQL query prompting the application to return a result. The result will depend on whether the query is true or false. The information within the HTTP response will either be modified or remain unchanged.

Out-of-band SQL injection:

Out-of-band injection can only occur if specific features are enabled on the database server. This type of attack is usually used as an alternative to the methods listed above. It is the opposite to the in-band injection in that it occurs if the attacker is unable to use the same channel to execute the attack and gather information.

Defending against SQL injection

The number one protection against SQL injection is parameterized statements. Parameterized statements make sure that the inputs passed into SQL statements are treated in the correct manner. Figure 5 shows parameterized statements that are resistant to SQL injection attacks.

```
// Connect to the database.
Connection conn = DriverManager.getConnection(URL, USER, PASS);

// Construct the SQL statement we want to run, specifying the parameter.
String sql = "SELECT * FROM users WHERE email = ?";

// Generate a prepared statement with the placeholder parameter.
PreparedStatement stmt = conn.prepareStatement(sql);

// Bind email value into the statement at parameter index 1.
stmt.setString(1, email);

// Run the query...
ResultSet results = stmt.executeQuery(sql);

while (results.next())
{
    // ...do something with the data returned.
}
```

Figure 5: Example of parameterized statements [9]

```
// The user we want to find.
String email = "user@email.com";

// Connect to the database.
Connection conn = DriverManager.getConnection(URL, USER, PASS);
Statement stmt = conn.createStatement();

// Bad, bad news! Don't construct the query with string concatenation.
String sql = "SELECT * FROM users WHERE email = '" + email + "'";

// I have a bad feeling about this...
ResultSet results = stmt.executeQuery(sql);

while (results.next()) {
    // ...oh look, we got hacked.
}
```

Figure 6: Explicit construction of SQL statement using string concatenation [9]

Figure 6 shows an unsecure method of using string concatenation to construct the SQL statements, this is extremely susceptible to SQL injection attacks.

Another approach to protecting against SQL injection is to remove the need to write SQL statements in the back-end code of the application. This can be done via object relational mapping. It allows you to manipulate data without the need for SQL statements. If there are no SQL statements present in the code, then the likelihood of a successful SQL injection attack is substantially reduced.

Cross-site scripting (XSS)

Cross-site scripting attacks involve injecting malicious code into a website, generally in the form of a browser side script. Since the browser has no way of determining that the script is malicious, the malicious script can access all cookies, tokens or any sensitive information stored by the browser for that session. The content of the malicious code will usually send this sensitive information back to the attacker. There are two main types of XSS attacks, persistent and reflected attacks.

During persistent attacks, the injected code is permanently stored on the target server, this could be in a comment, forum or even the database. This means that when a user requests the malicious comment, the script is then executed on the user's machine. Blind cross-site scripting is a subcategory of persistence attacks, the attacker's payload is stored on the target server and is then executed. The attacker would usually input this script into an input field within a form, the form sends the data to the server where the malicious script is executed.

Reflected attacks rely on the injected script being 'reflected' off the web server, usually during a search result. For example, if a user enters a script in a search box, and the application returns 'No results for: *insert script*', then this script is executed, it has been reflected from the URL of the search query.

Defending against XSS attacks

Input validation

Inputs should be validated to ensure that no malicious scripts are injected. For example, for email input fields, it should only accept strings that contain an '@' symbol, this dramatically reduces the number of scripts that can be injected. The same can be said for specifying minimum and maximum character lengths for inputs.

Content Security Policy

By adding the content-security-policy HTTP header to a web page, server administrators are able to specify the domains that the browser should consider to be trusted sources. This means that any scripts, including in-line scripts, will not be executed as the browser will detect that they come from a source that is not considered to be valid.

1.5 Relevant services and standards

To gain an understanding of both the design and functional requirements of the system I wish to implement, I reviewed the below relevant services that are currently available. In addition to reviewing relevant services, I also reviewed the security requirements for modern web applications as I want to ensure my prototype remains secure against the most common cyber threats.

BBC Bitesize

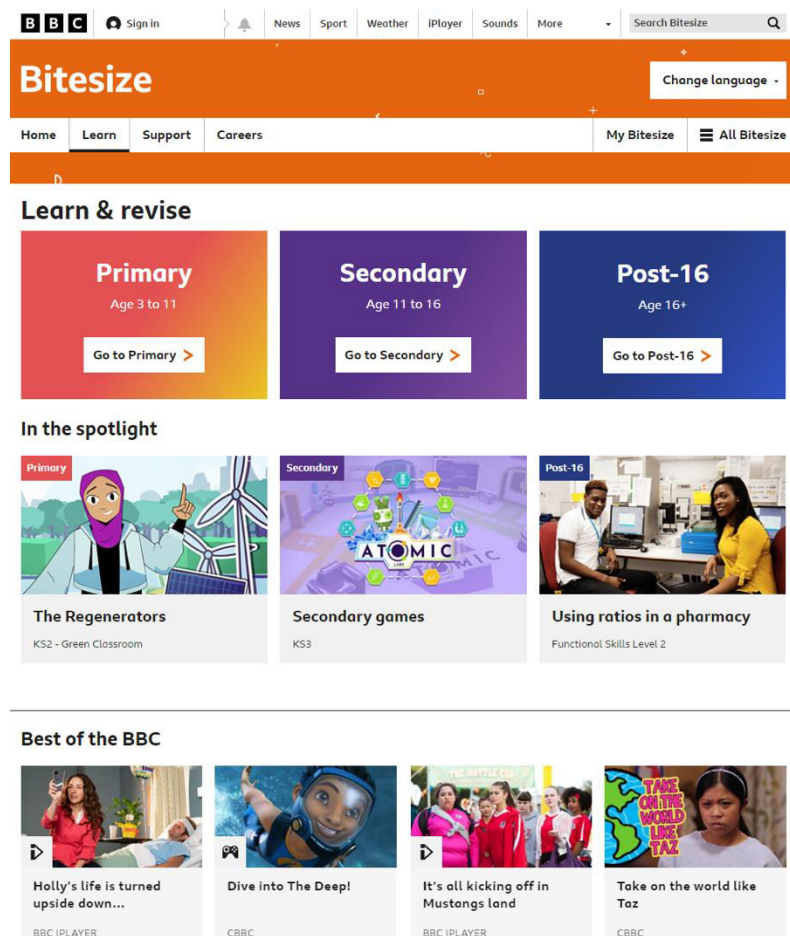


Figure 7: Homepage of BBC bitesize

While BBC bitesize is not solely a cyber security education platform, it hosts a wide range of material spanning multiple age groups and subjects. The concept is the same in that it aims to provide a centralized form of educational material for a variety of age groups. The homepage above shows a grid-like layout in which the user is able to select from 3 different age groups. The visual weighting of each element is appropriate, the eye is naturally drawn to the more important elements of the page (education level). The top navigation bar features a search and sign in function while also providing quick access to other services that the BBC provide.

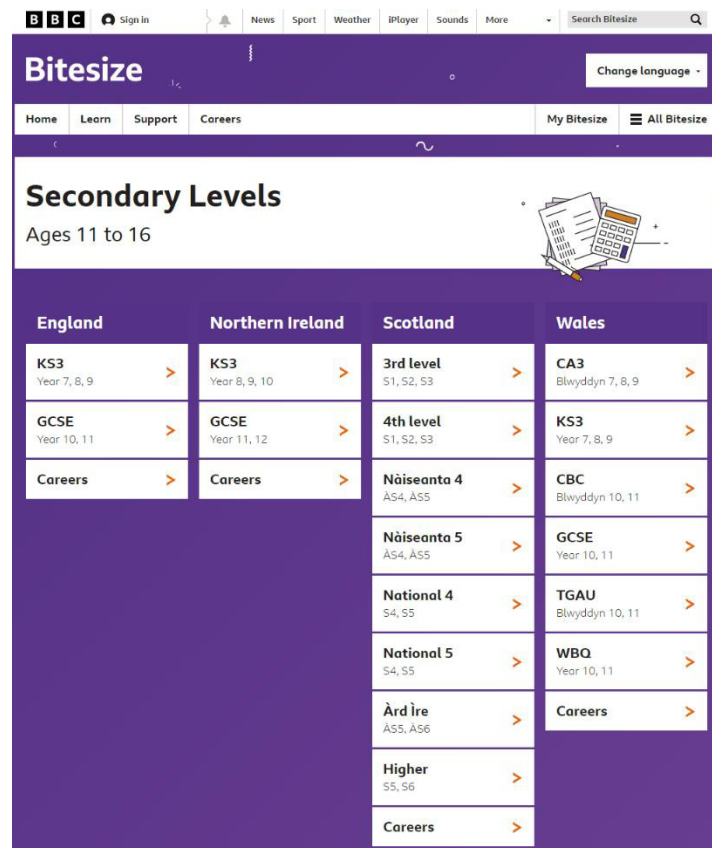


Figure 8: BBC bitesize page displayed after selecting education level

After selecting the 'secondary' level of education, the user is presented with the above page. It is a similar grid-like layout, displaying information such as the different countries within the UK and the structure of each countries education system. All text contrasts the background so it is clearly readable while also being accessible for those with colour blindness. The text is black/grey on a white background which is visible to all type of colour blindness. The two navigation bars are still present at the top of the page. So far, it is possible to navigate to this page without creating an account and signing in.

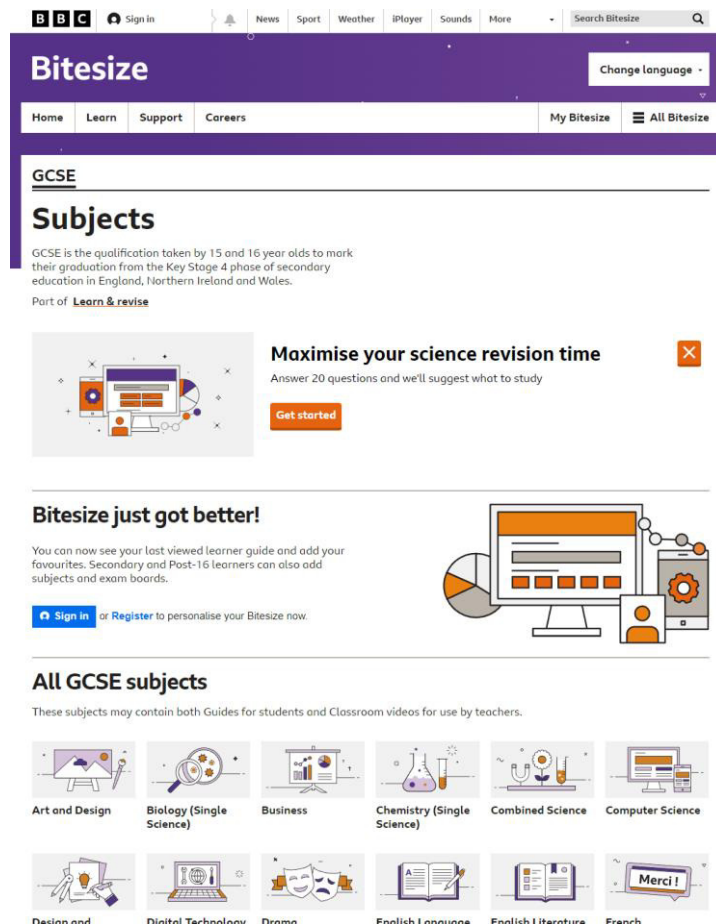


Figure 9: BBC bitesize subject selection page

Figure 9 shows the result of clicking on 'GCSE' under England in the previous page (figure 8). Again, a similar grid-like pattern is shown here, displaying the list of GCSE subject available to select. On this particular page, the visual weighting of each element is not as appropriate as those initially shown in figure 7. The centre of the screen is primarily focused on prompting the user to create an account to access more features. The content that a user originally navigated this page to access is right at the very bottom and requires the user to scroll down to even see past the second column of content. However, when signed in, this section is no longer displayed and a higher percentage of the screen is allocated to displaying the subjects.

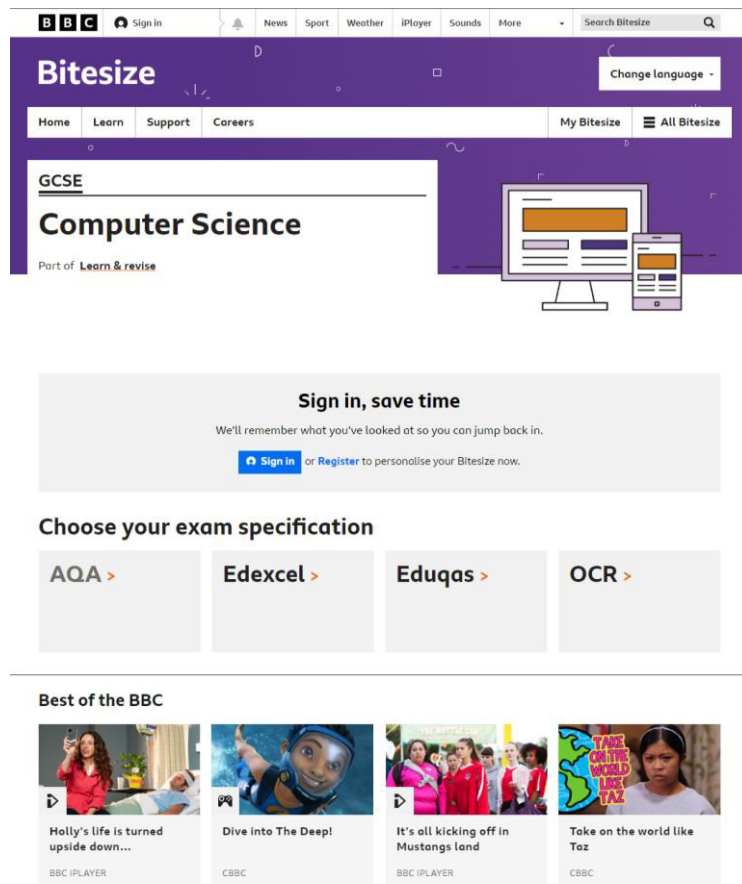


Figure 10: BBC bitesize exam specification selection

Figure 10 shows the page displayed after selecting the subject 'Computer science'. Again, the grid-like design pattern is continued here. It appears that BBC bitesize is more of an exam revision site, listing the specific exam specifications that are available throughout the country. The centre of the page is used to promote the user to register and create an account, but this is no longer displayed if the user is logged in. The footer titled 'best of the BBC' displays the most popular content that the BBC offer, however this is not relevant to the content that the user has selected.

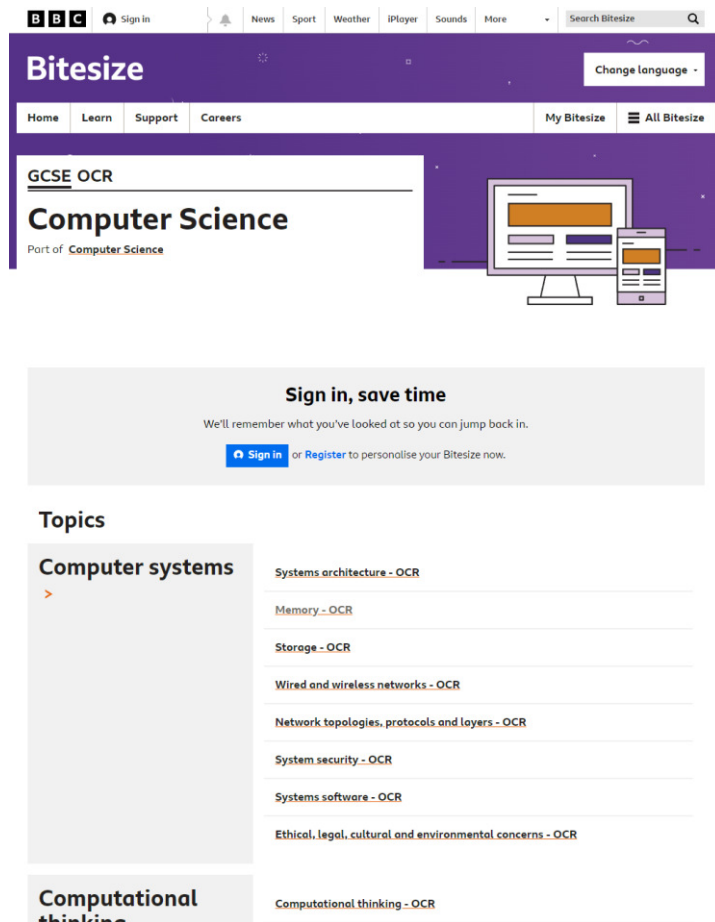


Figure 11: BBC bitesize computer science topic selection

Figure 11 shows the page that is displayed after selecting the exam board 'OCR'. It breaks the subject down into topics that are displayed in large, bold text on the left, and these are further broken down into sub-topics that are displayed on the right. The topics that are displayed here are unique to the exam board specification that was selected on the previous page.

Storage

Computers need to be able to store programs and data when the power is switched off. Secondary storage is used to hold data and programs when the computer is not in use.

Part of [Computer Science](#) | [Computer systems](#)

+ Add to My Bitesize



Share

Revise

Test

< 1 2 3 4 >

Secondary storage

Computers use primary **memory**, such as random access memory (**RAM**) and **cache** to hold **data** that is being processed. However, this type of memory is **volatile**, which means it loses its contents when the computer is switched off.

General purpose computers, such as personal computers and tablets, need to be able to store programs and data for later use.



Secondary storage is needed to keep programs and data indefinitely.

Secondary storage is **non-volatile**, long-term storage. It is used to keep programs and data indefinitely. Without secondary storage all programs and data would be lost the moment the computer is switched off.

There are many forms of secondary storage and each type of secondary storage device has its own characteristics. Because all devices are different, some are more suited to certain applications than others.

For example, a **hard disk drive** has a high capacity and reasonable access speed, making it suitable for everyday storage of programs and data. A **USB stick** is smaller in capacity, but extremely fast and portable, making it suitable for transferring files between computers.

Not all computers require secondary storage.

Embedded computers, such as those found in a digital watch or central heating system, do not need to store data when the power is turned off. The instructions needed to run them are stored in **ROM** and any user data is held in **RAM**.

< 1 2 3 4 >

More Guides

[Systems architecture - OCR](#) >

[Memory - OCR](#) >

Storage - OCR

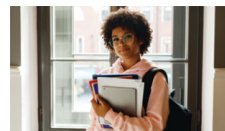
[Wired and wireless networks - OCR](#) >

[Network topologies, protocols and layers - OCR](#) >

[System security - OCR](#) >

[Systems software - OCR](#) >

[Ethical, legal, cultural and environmental concerns - OCR](#) >



Struggling to get your head round revision or exams?

Our tips from experts and exam survivors will help you through.

[Get advice here](#)

Links



[Personalise your Bitesize!](#)

Figure 12: BBC bitesize sub-topic content

Figure 12 shows the page displayed after selecting a sub-topic. The main information is displayed towards the centre of the screen, with navigation buttons present at both the top and bottom of the display. The navigation buttons feature arrows to enable the user to go to the next or previous page, while also providing the ability to instantly navigate to any page. The right-hand side of the page features quick links to various other sub-topics under computer science. The current selected sub-topic is highlighted in black.

Overall, BBC bitesize offers a wide range of features, I have summarized what I believe to be the strengths and weaknesses of this platform.

- + Log in/registration system
- + Content separated into different education levels
- + Search function
- + Ability to take tests
- No ability to upload content
- No file download ability
- Material complexity is limited to A-level

Brainpop

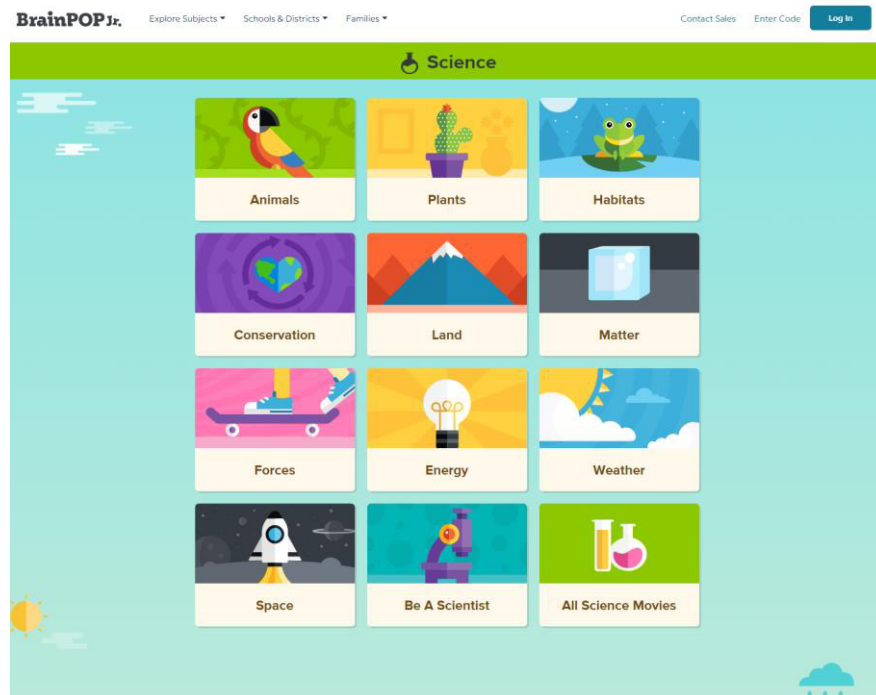


Figure 13: Brainpop homepage

Like BBC Bitesize, Brainpop does not exclusively provide content related to cybersecurity, however the concept is similar in that it aims to provide education to a wide range of age groups. The homepage in figure 13 features a card like design which are used to represent each category. A menu bar is present at the top of the page with options to log in. The 'BrainPOP' icon in the top left is clickable, returning the user to the homepage. The name of the topic selected is also presented at the top of the screen. All text is clearly visible as the text is a darker colour on a light background.

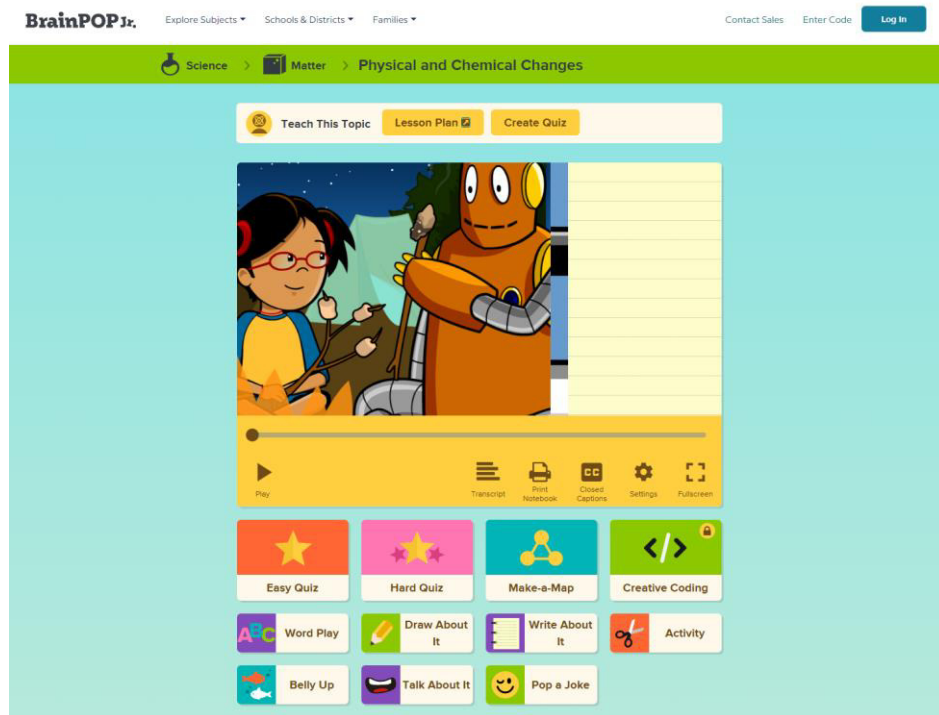


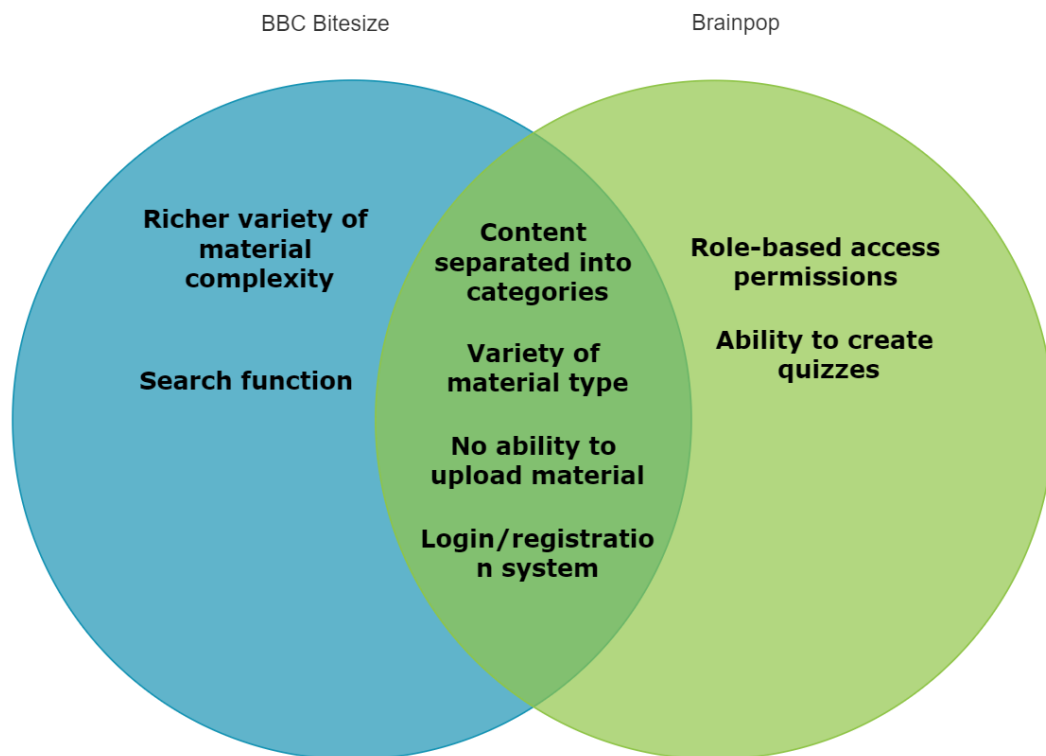
Figure 14: Page displayed after selecting topic

Figure 14 shows the page displayed after selecting the topic. It features an educational video in the centre of the screen, with playback options directly underneath. Again, the grid-like design is continued underneath, displaying further options such as the ability to take quizzes or make notes on the video. At this stage, an account is required to watch the videos or complete any of the activities. Those with teacher accounts are also able to make their own quizzes based on the content in the video. The design of the website is consistent across multiple pages, the same font and size is present and the location of elements is where I would expect them to be (log in button top right etc).

To summarise, Brainpop seems to focus on the younger age groups, this is clear from both the visual design of the website as well as the material present. Role based access permissions can be seen, for example only teacher accounts are able to make quizzes. Although, the permissions seem to be quite limited in that admin functions cannot be seen. However, it may be possible that the admin functions are not viewable by accounts without administrative permissions.

Summary of strengths and weaknesses:

- + Role based access permissions
- + Ability to create quizzes
- + Content separated into categories
- + Clear and consistent design
- + Variety of material type
- Material lacks variety in complexity
- No admin functionality visible
- No ability to upload new material



The above shows a Venn diagram comparing the features present in the two services are reviewed. There is only one weakness that they both share, and that is the ability to upload and share material made by the user. On both websites, there was no clear way to upload or view material uploaded by others. It appeared that the material on the site was made exclusively by the organisation which meant that the amount of material on the site was quite limited. Both websites had organised the material into categories either based on subject or education level. Additionally, they also both offered a variety of material type, ranging from quizzes to written material and videos. A log in and registration system was also present on both services. Upon logging in, more features were accessible.

The outcome from this analysis is that it is important for me to implement the features that they have in common into my own system. Both services offer these features for a reason, they help provide the necessary functionality of the website as well as improving ease of use. Separating content into categories means that users can spend less time searching for content, improving the usability of the system. From this chapter I was also able to identify the most common weaknesses of the relevant services, these include the lack of ability to upload content as well as the small variety in education levels. As a result of this, I have a good understanding of what features I wish to implement into my own prototype, either features I liked from other services, or improving on areas which I thought were lacking in the services I reviewed.

Relevant standards and regulations

The California Department of Education has set out some general standards for web applications [10]. These standards primarily focus on the usability and readability of the application.

- Must be easy and intuitive to use for the target audience.
- Must function in a logical manner for the target audience.
- Must use styles that are consistent throughout the application and within the associated website, these include:
 - o The use of capitalization
 - o The use of punctuation
 - o Error messages must appear in a consistent location and style (font type, size and colour)

The above requirements are more general standards; however the Open Web Application Security Project (OWASP) has set out some more specific guidance related to the security of a web application [11]. The OWASP top 10 is a list of the most critical web application security risks, it is constantly evolving in accordance with advancements and changes in the security market. It is recommended that web applications make clear attempts to protect against these risks. Below are presented the top 10 list and their explanations [23].

Broken Access Control

- Attackers may be able to get access to user accounts, or use the system as if they were an administrator
- For example, an attacker may be able to alter the URL to access pages they should not normally be able to access

Sensitive data exposure

- Data that is sensitive (passwords, credit card information etc) should be encrypted and passed only over secure methods of communication (HTTPS)
- Sensitive data that is not adequately protected can be easy targets for attackers
- Sensitive data should also be stored in a way that is compliant with the GDPR

Injection

- An attacker injects code into the application by inputting invalid data into forms, URL's etc. The attacker could alter the way the application functions, as well as access information they should not be able to access.
- To avoid SQL injection, statements must be parametrized, or object relational mapping should be used

Broken authentication

- Authentication functions can be exploited if not implemented appropriately, this can allow attackers to compromise passwords and even hijack user sessions.
- It is important to have a strong authentication method, such as email authentication

XML External Entities

- Web applications that use XML can be vulnerable to attackers uploading hostile commands within an XML document
- Static application security testing should be used to detect XXE in the code of the application

Security Misconfiguration

- A configuration error could result in a configuration weakness in which attackers are able to exploit
- All web applications should be configured correctly

Cross-Site Scripting (XSS)

- XSS attacks occur when a web application allows untrusted data to be uploaded, this untrusted data can include hostile scripts that the attacker has injected
- Tools should be used to detect any critical defects that allow XSS attacks

Insecure Deserialization

- Attackers can remotely execute code as a result of deserialization flaws. Deserialization is the process of reconstructing a data structure from a series of bytes in order to instantiate it for consumption.
- Penetration testing can be used to highlight any potential vulnerabilities

Using components with known vulnerabilities

- When developers choose to ignore warnings and use components with known vulnerabilities, attackers are able to sniff out these vulnerabilities and exploit them
- A web application should always use components with the fewest vulnerabilities possible

Insufficient logging and monitoring

- Logging provides important timelines of events such as failed login attempts, successful logins and other important events. Events that are not logged could potentially lead to a vulnerable application
- Logging should be present throughout the application

Reviewing the most common security vulnerabilities has developed my understanding of the need to ensure that web applications are resistance to security threats. Therefore, I need to develop my prototype to be secure, taking into account the vulnerabilities discussed above.

User personas

In order to gain a better understanding of the requirements and objectives of this project, I have created the following personas that I believe demonstrate the primary users of this system.

Persona 1

Name: Jason Smith

Description:

Jason is a 42-year-old computer science teacher from York. He has detailed knowledge of cybersecurity and wants to share his knowledge to those beyond his classroom. Jason has created a wealth of cybersecurity resources suitable for all age groups. He currently has no centralized resource to share his material with those interested in cybersecurity

Needs:

Jason would like a centralized application that allows him to share his educational resources to those not just in his classroom. He would like the ability to upload a variety of content suitable for different age groups to the same application, while separating them based on complexity/age group. As his content varies by type, from word documents to videos, he needs to be able to upload any file type. Jason wants his content to be consumed by those all over the world, and assist in reducing the cybersecurity skill gap that is ever increasing. He wishes that his content inspires people of all ages to learn about cybersecurity and potentially inspire some to pursue a career in cybersecurity.

Persona 2

Name: Ben Greene

Description:

Ben, age 12, has his eyes set on pursuing a career in cybersecurity. He currently struggles with finding cybersecurity material that is suitable for his age group. He is aware that the older students in the school are being exposed to cybersecurity material in their lessons but Ben is a few years away from that just yet. Ben asks his teachers if they know of any cybersecurity material available that he would understand, they tell him that they will send him over some material specifically made for him. After completing the material, Ben doesn't want to pester the teachers again and request more material.

Needs:

Ben wants to easily be able to find cybersecurity material suitable for his age group. He aims to find a variety of material, ranging from online games to videos, that he can consume on a regular basis. He needs to be able to locate said material without asking his teachers. Upon completion of the material, when he feels ready, he wants to be able to progress to the next education level and consume more complex cybersecurity material. He feels that this independent learning, especially learning of content that is not in the school curriculum, will give him an advantage over his peers when he eventually applies for a job in the cybersecurity sector.

1.6 Approach

This project aims to provide a centralized form of cyber security educational material that can be used to supplement material taught at all levels throughout the education system while also remaining secure against some of the most common cyber-attacks. To ensure that these aims are completed, it is important to choose a suitable software development methodology. There are a variety of development methodologies available, but the two most common include waterfall and agile [14]. I shall start by comparing the two, how their structures differ as well as the advantages and disadvantages of both.

Waterfall methodology

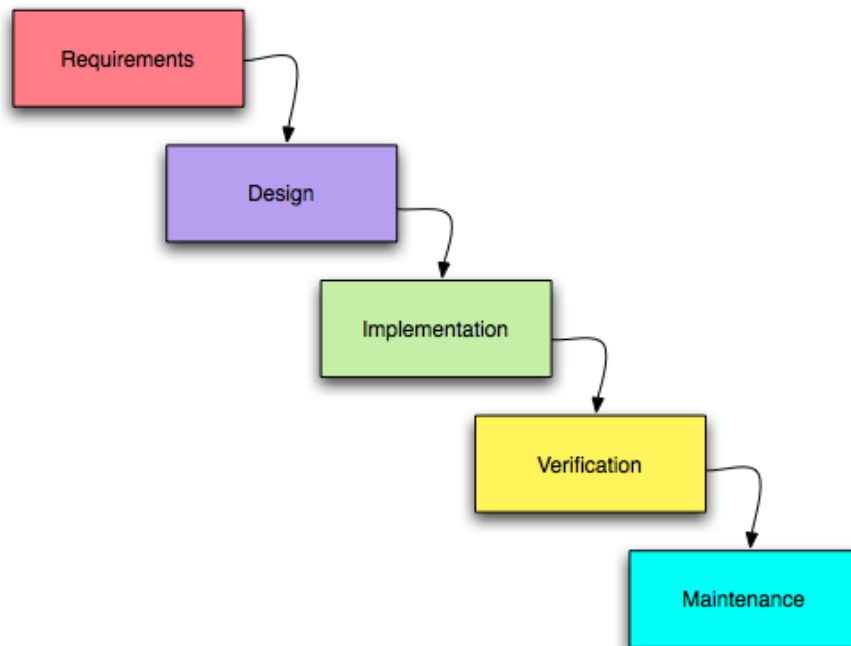


Figure 15: Waterfall methodology [15]

Figure 13 shows a diagram of the flow of the waterfall development methodology. The waterfall methodology is very much a linear design, one stage cannot be completed until all the previous stages have been. During the requirements stage, all requirements are determined ahead, including functional and non-functional. During this phase, it is important to get as detailed an understanding of the user's requirements as possible, this may involve data gathering/analysis or reviewing competitors' products. During the design phase, the requirements are used to design the system, what hardware is needed/suitable, specific software technologies required etc. The next phase is the implementation, this is when the programmers take the requirements and design into consideration and create the software. As a result of the previous phases, the programmers will have a clear understand of what is needed. After the software has been successfully implemented, it must be tested and verified that it meets the requirements, this is achieved in the verification phase. At this point, the product is released to the customer, and they provide feedback as to how well the product meets their requirements. The final stage is maintenance, this is an ongoing stage as it will likely last for the duration of the software's use. It involves fixing any bugs that the customer has found, implementing features that are missing, and general upkeep such as ensuring the software functions on the latest OS releases.

Due to its linear approach, the waterfall method has many advantages. As the system is designed before implementation is started, any design errors that have been caught before implementation mean that time is saved during implementation. Additionally, the waterfall method has a clear structure and follows a set of predefined steps. One step cannot be completed if the previous step is unfinished, this means that any problems encountered are brought to light quickly. As a result of the requirements phase, testing the product is easier as it can be completed by reference to the scenarios defined in the requirements phase. Moreover, since the approach is very structured and follows a set of predefined steps, progress is much easier to measure.

The waterfall methodology is not without its disadvantages, as the development process is very much separate from the customer, customer satisfaction is generally lower than products developed with an agile methodology. This can mean that the requirements are not met, resulting in the application having to be re-engineered. Due to the linear approach, re-engineering the application is time consuming and expensive, the new requirements simply cannot be added in during the implementation phase, the project will have to begin at the requirements stage. Moreover, the rigid structure of the approach means that it does not cater well for the possibility of a change in requirements during the development cycle.

Agile development methodology

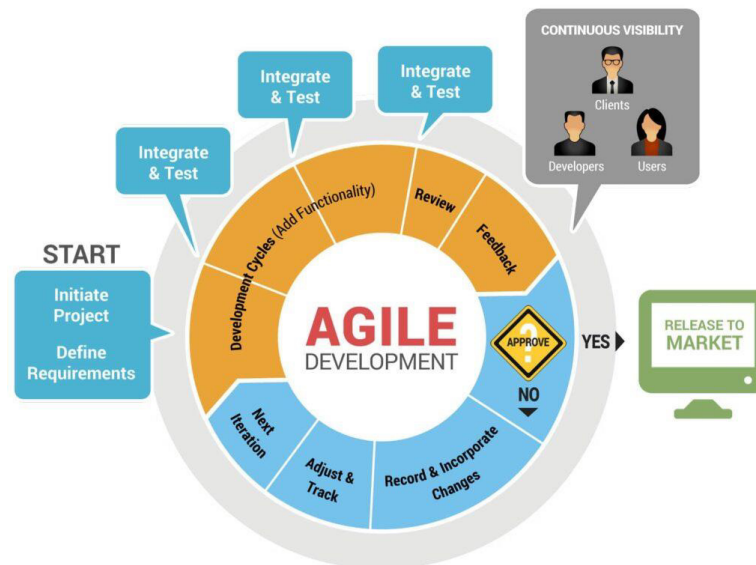


Figure 16: Agile methodology flow [16]

Figure 16 shows the flow of an agile development methodology approach, it is very much a circular approach and relies on continuous improvement. It is heavily focused on testing, adjusting and re-developing the product until release. Unlike the waterfall approach, the customer is heavily involved in the development process, customer input is welcomed at every stage of the process.

As a result of the heavy customer involvement, customer satisfaction is usually very high. If any requirements were somehow not met, due to the circular design approach, it is very easy to implement additional functionality. Again, as the customer is heavily involved in the development process, it is very difficult for unmet requirements to be released to the market as the customer will usually spot this during the cycle.

The agile development methodology is a heavily customer focused methodology, and therefore makes some sacrifices in other areas. For example, as the design is a circular approach, there is no clear end in sight for a 100% complete project. Therefore, it is very easy to go off track and do more cycles than required, which can end up being expensive. Again, this can make it difficult to measure overall progress as there is no finite end in sight. The incremental delivery can result in a fragmented output, teams work on each component in different cycles. This also makes it only suitable for larger projects.

Due to the fact that the agile development methodology is more suited to larger projects and relies on involving customers, I decided to implement the waterfall methodology. For this project, I need a clear definition of when the implementation is complete, and the disadvantages of the waterfall methodology are mainly associated with the lack of customer involvement, which does not affect

this project. Due to the relatively limited time scale for this project, measuring progress is vital to ensure I remain on track to deliver a functioning implementation by the deadline. Measuring progress during a waterfall approach is significantly easier as it follows a defined structure, allowing me to identify when I am falling behind and speed up implementation.

Evaluation method

There were two main types of evaluation techniques that I was considering, heuristic and functional testing. Heuristic evaluation focuses on the usability of the system, particularly Nielsen's 10 usability heuristics [17]. The list contains 10 usability characteristics that the product must have.

- Visibility of system status
 - o The system should always keep the user informed of the system status within a reasonable time
- Match between system and the real world
 - o The system should speak the user's language, with phrases and concepts familiar to the user
- User control and freedom
 - o Users will need an exit button to leave an unwanted mistake
- Consistency and standards
 - o The system should be consistent with other products and standards, red should signal delete
 - o Fonts/colours should remain consistent throughout the application
- Recognition rather than recall
 - o Options should be visible, a user should not have to remember information from one part of the application to the other
- Flexibility and efficiency of use
 - o The system should cater to both inexperienced and experienced users
- Aesthetic and minimalist design
 - o Only relevant information should be displayed to avoid cluttering the display
 - o The visual weighting of each element should be proportional to that elements importance
- Help users recognize, diagnose and recover from errors
 - o Error messages should be expressed in terms that an inexperienced user can understand
- Help and documentation
 - o Any additional information to assist the user in operating the system should be easy to search and list concrete steps to be carried out

Heuristic evaluation mainly focuses on the non-functional requirement that the system should be easy to use, achieving the desired task with little to no effort.

Functional testing on the other hand focuses on the functional requirements of the system, what the system can do. It helps determine if the application as met the functional requirements as set out in the earlier stages of the development methodology. The result of a functional test case is usually a 'pass' or 'fail', it generates discrete data rather than continuous. Discrete data is important when testing software as you cannot partially meet a functional requirement. Functional testing ensures the proper functionality of a system, whereas a heuristic evaluation could give a great score despite the system providing limited functionality. As a result of this, I decided to use functional testing to

evaluate my product. I opted to not use candidates in a study to evaluate my product as the ethical process can be time consuming and I can perform the functional evaluation myself.

Achieving research aims and objectives

The aims and objectives listed above were achieved by a combination of my own desires and the requirements specified by my supervisor. I used knowledge from previous modules to establish what would need to be done to deliver a successful implementation. For example, UX design taught me to make wireframes to help create a UI with a high usability. I then put this as an objective that needed to be completed. This was done for each step in the software development process.

1.7 System design

The below requirements have been determined by a combination of reviewing relevant websites, standards and regulations.

Non-functional User Requirements

Requirement 1 (must):

Requirement:

User friendly

Acceptance criteria:

- Text must be large enough to be easily read
- Text must be an appropriate colour based on the background
- Buttons must be in a consistent style and location
- UI must have the appropriate number of elements

Requirement 2 (must):

Requirement:

Sensitive data such as passwords must be stored in a secure manner (encrypted)

Acceptance criteria:

- Data is salted and hashed before being sent over a network to be stored in the database
- Plaintext values of passwords must not be visible

Requirement 3 (should):

Requirement:

Be resistant to SQL injection attacks

Acceptance criteria:

- Any SQL code injected must not be executed

Requirement 4 (should):

Requirement:

Be resistant to small scale DDoS attacks

Acceptance criteria:

- Web application remains fully functional when experiencing a DDoS attack

Requirement 4 (could):

Requirement:

Log all events

Acceptance criteria:

- Web application logs every event for every user

Requirement 5 (could):

Requirement:

Be HTTPS encrypted

Acceptance criteria:

- Data is not transmitted in plain text

Requirement 6 (could):

Requirement:

Mitigate XSS attacks

Acceptance criteria:

- Upon entering a script into an input field, or uploading an html file containing a script, the browser recognizes that it is not from a trusted source and the script is not executed

Functional User Requirements

Requirement 1 (must):

Requirement:

Have a log in/register system

Acceptance criteria:

- New accounts are created and stored in the database
- Application validates that credentials are correct and logs the user in

Requirement 2 (must):

Requirement:

Access permissions that vary by user role

Acceptance criteria:

- Different user roles
- Each role has different permissions

Requirement 3 (must):

Requirement:

Ability for educators and admins to upload new content to the site

Acceptance criteria:

- File upload ability
- New content can be viewed by all users

Requirement 4 (must):

Requirement:

Ability for all users to download content

Acceptance criteria:

- File upload ability
- New content can be viewed and downloaded by all users

Requirement 5 (must):

Requirement:

Specific content must be downloadable by all users

Acceptance criteria:

- Download button initiates a download of the content to the user's machine

Requirement 6 (must):

Requirement:

Allow users to log out of the session

Acceptance criteria:

- Once log out button clicked, the user's session is terminated
- User is returned to the log in screen
- User cannot return to a logged in state by pressing the browsers back button

Requirement 7 (must):

Requirement:

Allow filtering of material

Acceptance criteria:

- User can search for terms and relevant material is displayed

Requirement 8 (should):

Requirement:

Allow admins to manage user accounts

Acceptance criteria:

- Admins can disable user accounts
- Account info is no longer stored in the database

Requirement 9 (could):

Requirement:

Allow users to favourite material

Acceptance criteria:

- Users can select to favourite material
- This material will then be displayed in a favourites section that is unique to each user

Requirement 10 (could):

Requirement:

Allow users to create their own content within the site using a rich text editor

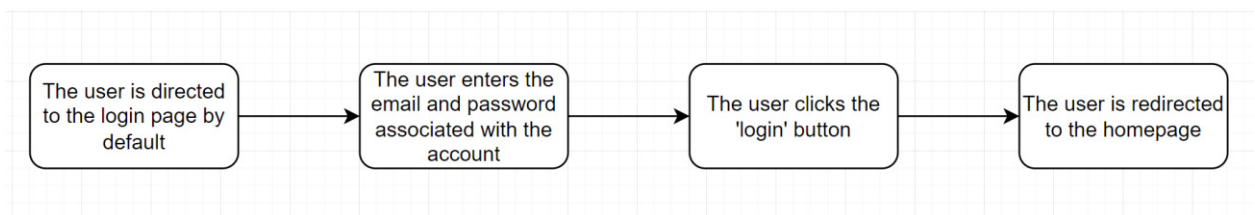
Acceptance criteria:

- Users can use a rich text editor to create content
- Content is uploaded to the database and displayed for all users

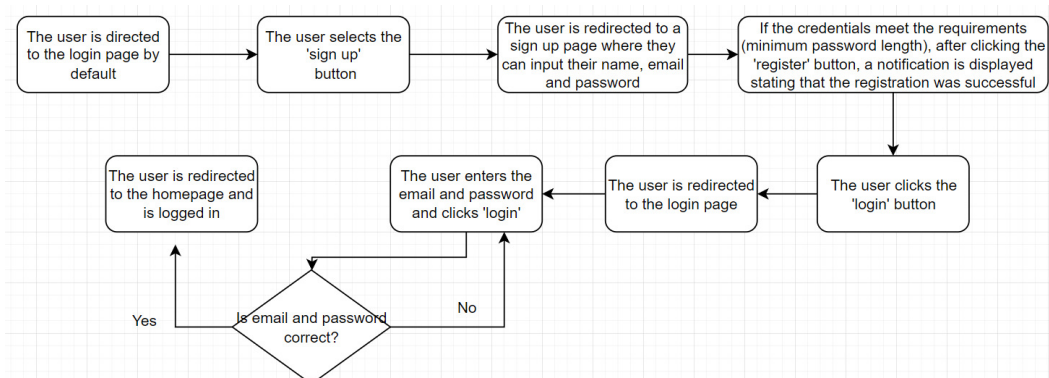
Use cases

Use case ID: 1 The users can log in and access the system (Must)	
Goal	The user must be able to create an account/log in to the system to be able to access the resources
Preconditions	The user may or may not have an account
Basic flow	1). The user is directed to the login page by default 2). The user enters the email and password associated with the account 3). The user clicks the login button 4). The user is then redirected to the homepage
Alternate flow	1). The user is directed to the login page by default 2). The user does not have an account so clicks the 'Sign up' button 3). The user is redirected to a sign up page where they input their first and last name, email and a password 4). The user clicks the sign up button, a notification is displayed stating that the registration was successful 5). The user navigates to the login page by pressing the log in button where they input the email and password 6). After clicking the log in button, they are redirected to the homepage

Basic flow for use case 1:

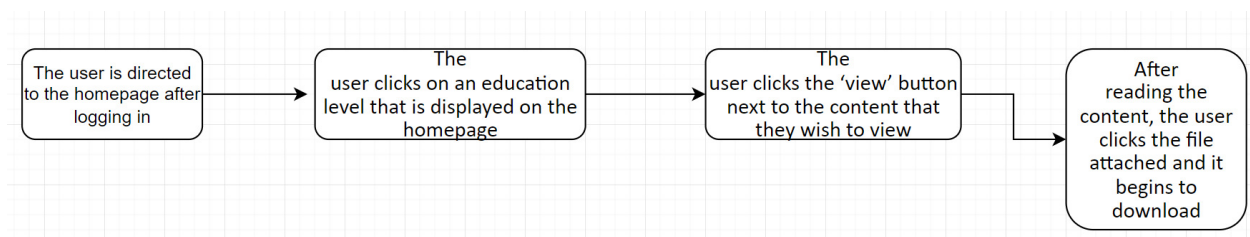


Alternate flow for use case 1:

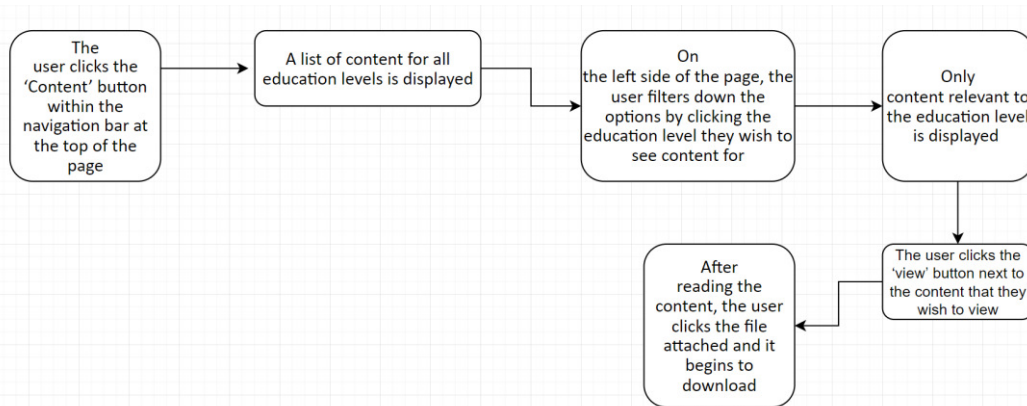


Use case ID: 2 The users can download material for a specific education level (must)	
Goal	The user must be able to click on an education level, see a selection of content for that education level, view the content and download any material attached
Preconditions	The user must be logged in as any role (User, Admin, Educator)
Basic flow	1). The user clicks on an education level that is displayed on the homepage 2). The user clicks the 'view' button next to the content that they wish to view 3). After reading the content, the user clicks the file attached and it begins to download
Alternate flow	1). The user clicks the 'Content' button within the navigation bar at the top of the page 2). A list of content for all education levels is displayed 3). On the left side of the page, the user filters down the options by clicking the education level they wish to see content for 4). Only content relevant to the education level is displayed 5). The user clicks the 'view' button next to the content that they wish to view 6). After reading the content, the user clicks the file attached and it begins to download

Basic flow for use case 2:

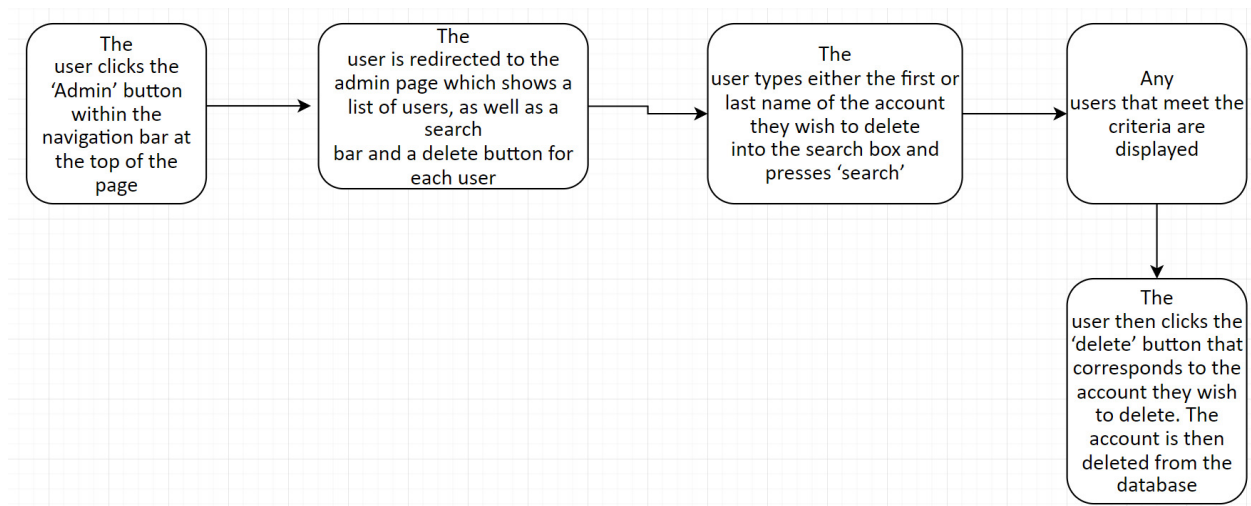


Alternate flow for use case 2:



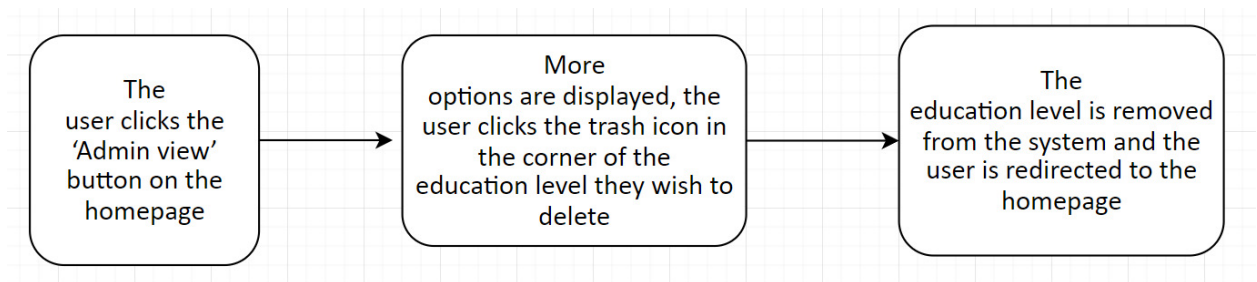
Use case ID: 3 Admins can delete accounts from the database (could)	
Goal	Admins can delete accounts from the database
Preconditions	The user must be logged in as an admin
Basic flow	1). The user clicks the 'Admin' button within the navigation bar at the top of the page 2). The user is redirected to the admin page which shows a list of users, as well as a search bar and a delete button for each user 3). The user types either the first or last name of the account they wish to delete into the search box and presses 'search' 4). Any users that meet the criteria are displayed 5). The user then clicks the 'delete' button that corresponds to the account they wish to delete. The account is then deleted from the database

Basic flow for use case 3:



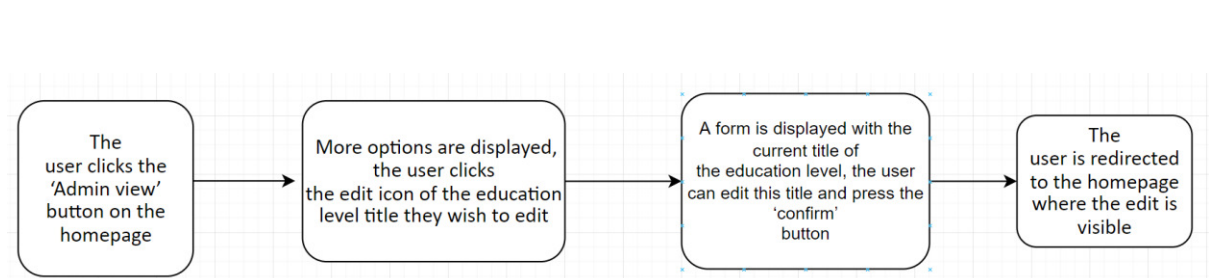
Use case ID: 4 Admins can delete education levels from the database (Should)	
Goal	Admins can delete education levels from the database
Preconditions	The user must be logged in as an admin and on the homepage
Basic flow	1). The user clicks the 'Admin view' button on the homepage 2). More options are displayed, the user clicks the trash icon in the corner of the education level they wish to delete 3). The education level is removed from the system and the user is redirected to the homepage

Basic flow for use case 4:



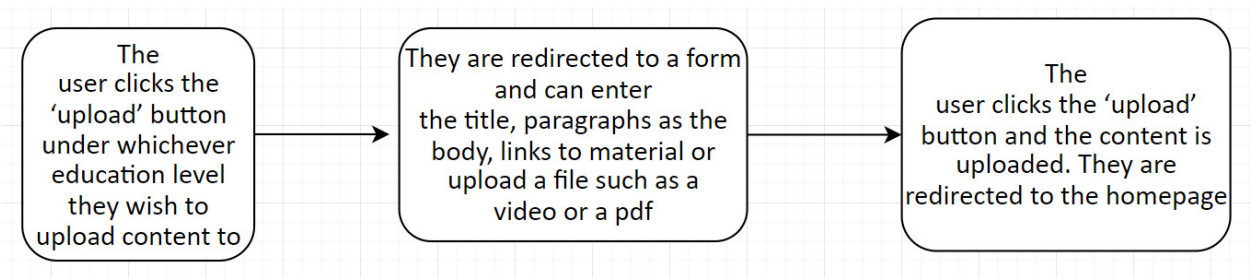
Use case ID: 5 Admins can edit the title of education titles (could)	
Goal	Admins are able to edit the title of education levels if they make a mistake when adding a new level
Preconditions	The user must be logged in as an admin
Basic flow	1). The user clicks the 'Admin view' button on the homepage 2). More options are displayed, the user clicks the edit icon of the education level title they wish to edit 3). A form is displayed with the current title of the education level, the user can edit this title and press the 'confirm' button 4). The user is redirected to the homepage where the edit is visible

Basic flow for use case 5:



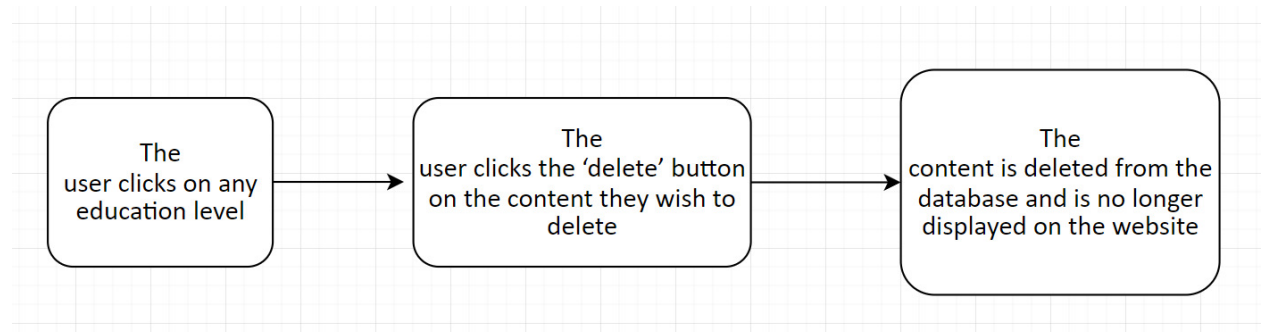
Use case ID: 6 Admins and educators can upload new content (must)	
Goal	Admins and educators are the only roles able to create and upload new content to the website
Preconditions	The user must be logged in as an admin or educator
Basic flow	1). The user clicks the 'upload' button under whichever education level they wish to upload content to 2). They are redirected to a form and can enter the title, paragraphs as the body, links to material or upload a file such as a video or a pdf 3). The user clicks the 'upload' button and the content is uploaded. They are redirected to the homepage

Basic flow for use case 6:

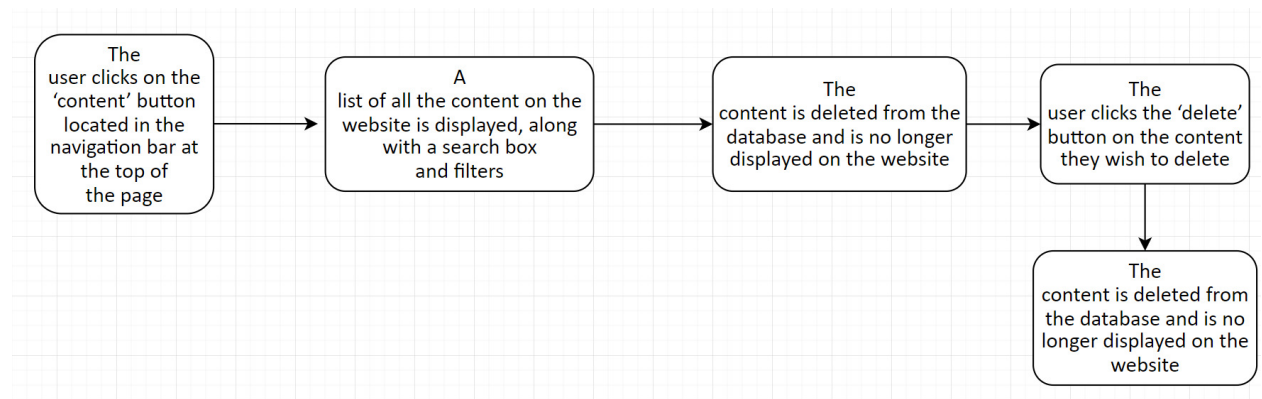


Use case ID: 7 Admins can remove content from the system (should)	
Goal	Only admins are able to delete content from the system
Preconditions	The user must be logged in as an admin
Basic flow	1). The user clicks on any education level 2). The user clicks the 'delete' button on the content they wish to delete 3). The content is deleted from the database and is no longer displayed on the website
Alternate flow	1). The user clicks on the 'content' button located in the navigation bar at the top of the page 2). A list of all the content on the website is displayed, along with a search box and filters. 3). The user clicks the 'delete' button on the content they wish to delete 4). The content is deleted from the database and is no longer displayed on the website

Basic flow for use case 7:

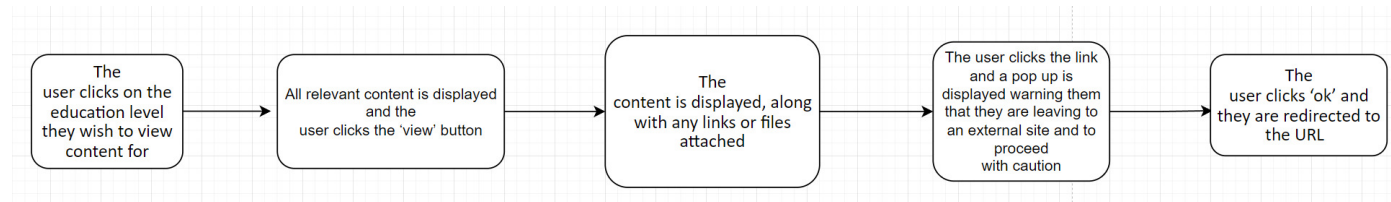


Alternate flow for use case 7:

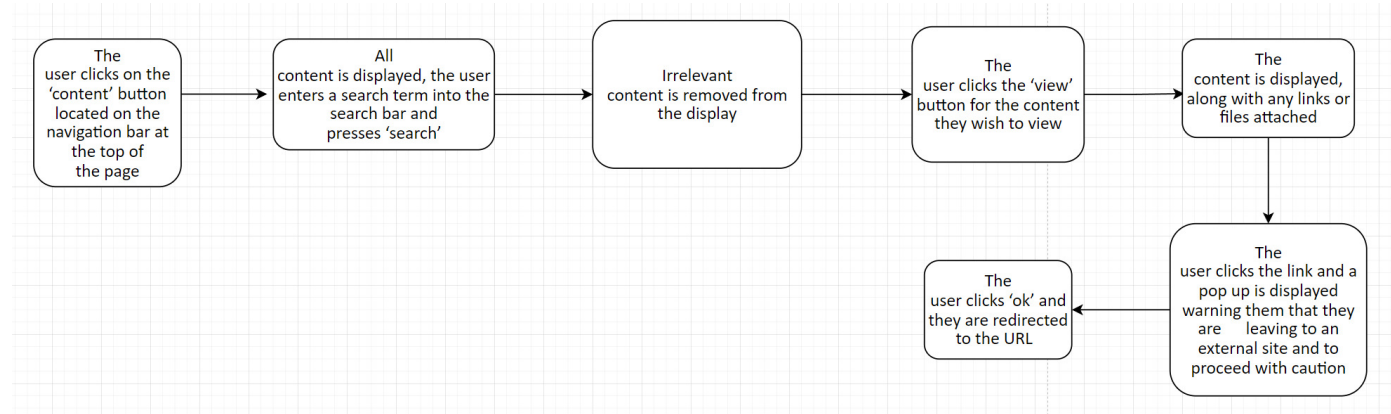


Use case ID: 8 Users are able to follow the links uploaded (must)	
Goal	All users are able to follow external links attached to the content
Preconditions	The user must be logged in and on the homepage
Basic flow	1). The user clicks on the education level they wish to view content for 2). All relevant content is displayed and the user clicks the 'view' button 3). The content is displayed, along with any links or files attached 4). The user clicks the link and a pop up is displayed warning them that they are leaving to an external site and to proceed with caution 5). The user clicks 'ok' and they are redirected to the URL
Alternate flow	1). The user clicks on the 'content' button located on the navigation bar at the top of the page 2). All content is displayed, the user enters a search term into the search bar and presses 'search' 3). Irrelevant content is removed from the display 4). The user clicks the 'view' button for the content they wish to view 5). The content is displayed, along with any links or files attached 6). The user clicks the link and a pop up is displayed warning them that they are leaving to an external site and to proceed with caution 7). The user clicks 'ok' and they are redirected to the URL

Basic flow for use case 8:

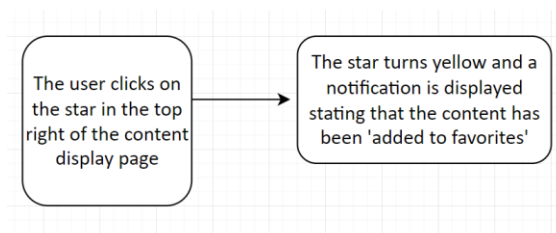


Alternate flow for use case 8:



Use case ID: 9 Users are able to add content to their favorites (could)	
Goal	Allow users to add content to a list of favorites
Preconditions	The user must be logged in and currently viewing content
Basic flow	1). The user clicks on the star in the top right of the content screen 2). The star turns yellow and a notification is displayed stating that the content has been 'Added to favorites'

Basis flow for use case 9:



Low fidelity UI prototype

While I am using functional testing to evaluate the application, it is still important to consider the usability of the system and make the UI visually pleasing. Below are the mock-up designs for the UI, this also helped me visualize the system and aided in development.



Figure 17: Prototype login page

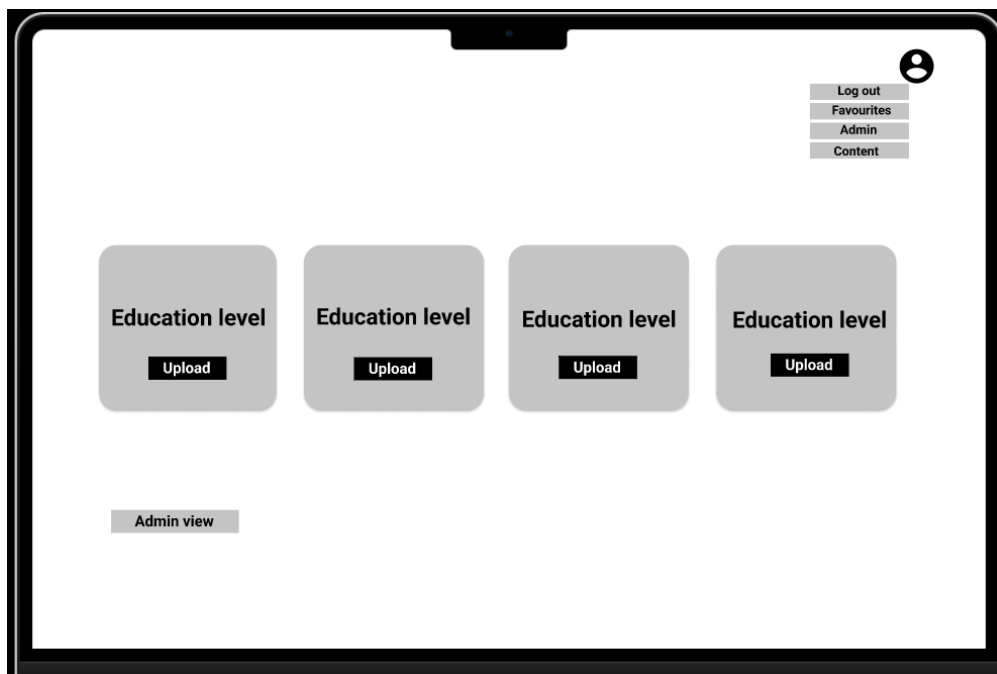
Figure 17 shows the mock-up design for the log in page. An email input field, along with a password field is displayed. Below that are the options to login, or navigate to a sign up page if the user does not have an account. This layout is consistent with the traditional approach to a log in page and is likely to be familiar with the user, allowing for increased usability.



A prototype of a sign-up page displayed on a tablet. The page has a white background with a black border. At the top center, the text "Sign up" is displayed. Below it, there are three input fields labeled "Email", "Password", and "Confirm password". At the bottom of the form, there are two buttons: "Sign up" and "Log in".

Figure 18: Prototype sign up page

Figure 18 shows the prototype for the sign-up page, it features an email, password and confirm password field. The same two buttons as shown on the previous page remain in the same location for improved consistency. The title at the top of the page clearly displays that the user is on the sign up page rather than the login page.



A prototype of a homepage displayed on a tablet. The page has a white background with a black border. In the top right corner, there is a user profile icon and a menu with the following items: "Log out", "Favourites", "Admin", and "Content". In the center of the page, there are four identical grey rounded rectangles, each containing the text "Education level" and an "Upload" button. In the bottom left corner, there is a button labeled "Admin view".

Figure 19: Homepage prototype

Figure 19 shows the prototype for the homepage, it features a grid like design with the education levels in cards towards the centre of the screen. The education levels take up the majority of the space on the screen, they have a large visual weighting which is proportional to the importance of them as an element. In the top right corner, more options are displayed, including an option to log out, navigate to the favourites page, admin or content page. In the bottom left, there is a button which switches to the admin view, allowing the user (who must have admin role) to see more options. On each education level card, there is an upload button that is clearly visible due to the contrasting colours.

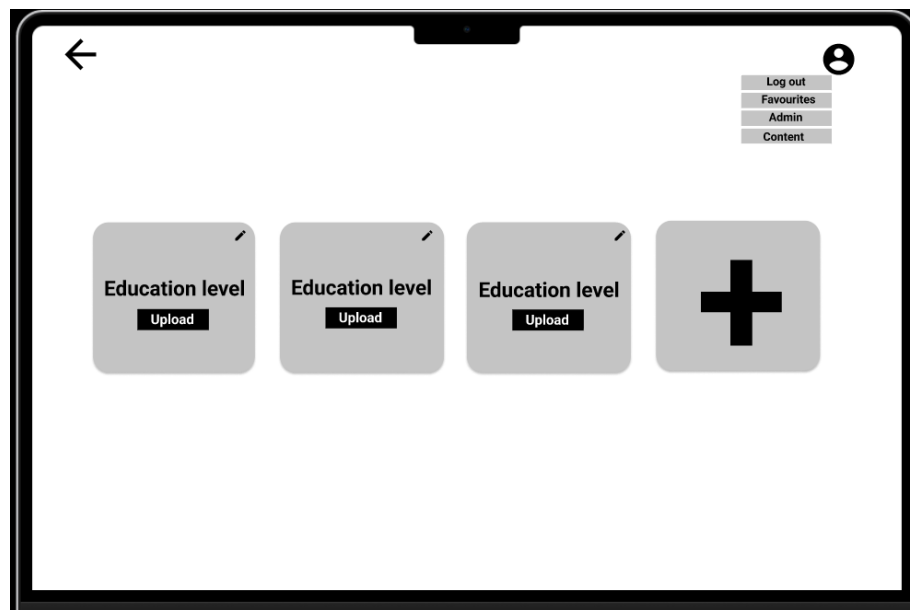


Figure 20: Admin view homepage prototype

Figure 20 shows the result of clicking the 'admin view' button on the homepage, more options are displayed that are exclusively accessible by admins. For example, there is an edit button in the top right of the cards, once clicked, this redirects the admin to a form for editing the title of the education level. The plus symbol on the right of the screen is universally recognized as the addition symbol, indicating that it is responsible for adding a new education level. The arrow pointing to the left can also be recognized as a 'return' or 'back' button as it is consistent with other applications available.

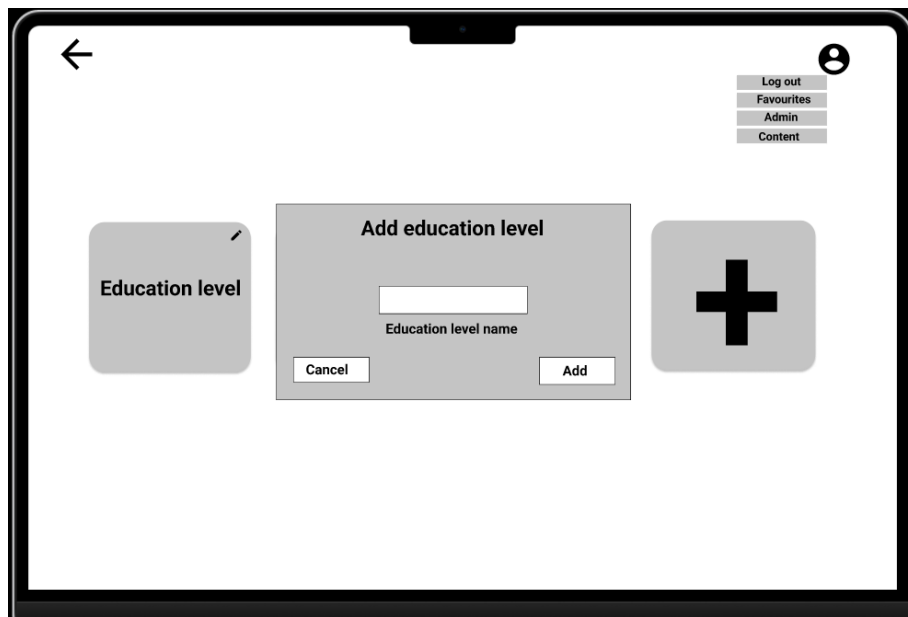


Figure 21: Add new education level form prototype

Figure 21 shows the result of clicking the '+' button on the admin view page. A form is displayed in the centre of the screen where the user is able to input an education level name. There is also an option to cancel, with a button in the bottom left, or an option to add in the bottom right. These options remain in these positions for the edit form to improve the consistency of the UI and increase usability.

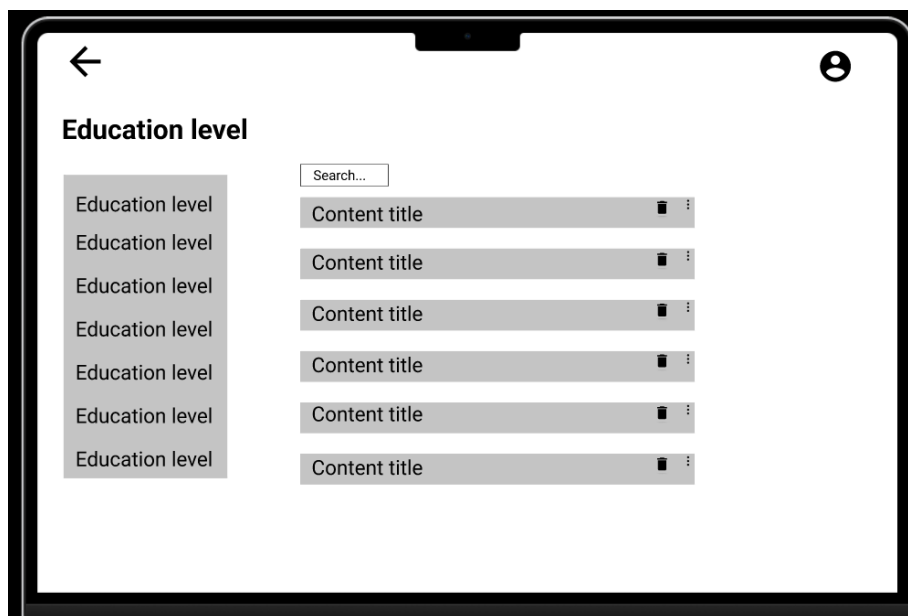


Figure 22: Content selection prototype

Figure 22 shows the prototype for the content selection view, this is displayed after selecting an education level. Only content that falls under the education level is displayed. On each content title, a 'trash' icon is present, this is well known as the symbol for delete and therefore should easily be recognizable by users. At the top of the list of content, a search bar is present. This will allow users to search for the title of specific content they wish to view. Additionally, if the user wishes to display

content that falls under a different education level, they may choose from a list of education levels on the left side of the screen. The education level that they are currently browsing is clearly displayed in large text above the list of education levels. Again, the arrow is present in the same position as in the other displays for consistency and usability reasons.

Education level

Upload material

Title

Link

Cancel Browse Upload

Figure 23: Upload content form prototype

Figure 23 shows the upload content form that is displayed after clicking the 'upload' button that is featured below each education level on the homepage. The styling is very consistent with the add and edit education level form, the cancel button remains in the same position. The form features two input fields for a title and a link. The link is not a required field as an educator may not wish to upload a link to their content. The 'browse' button opens the file explorer and allows the user to attach a file to their content. Again, the arrow is featured in the same position as the previous pages.

Content title

https://sampleLink/ file.pdf

Figure 24: Content view prototype

Figure 24 shows the content view page, the result of clicking 'view' on the content. The centre of the page showcases the text area, this is where the text that the educator has inputted is displayed. In the top left, in large, bold text, is the title of the content. Directly under text area for the content are buttons for the material attached. One button directs the user to the link that the educator has attached, while the other starts downloading the file. The file type is viewable before downloading the file encase the user does not want to download a video file as they are typically much larger than PDF's.

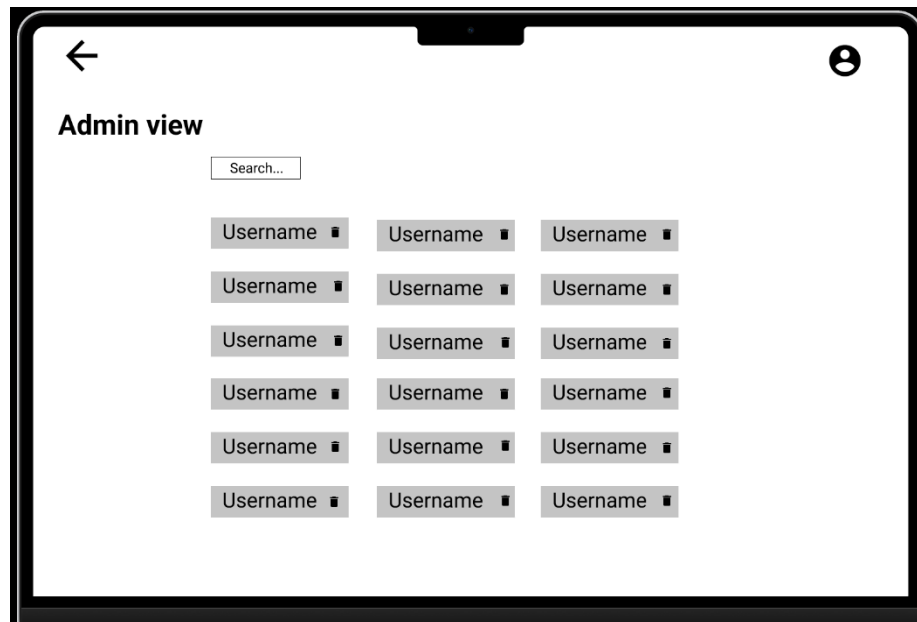


Figure 25: Admin view prototype

Figure 25 shows the admin view page, this page is only accessible by those with the admin role. It displays the users in a grid-like manor, with the same delete icon as featured on previous pages. A search bar is also present in the same position as with the other pages, and the title is clearly visible in large, bold text.

System Architecture Diagram

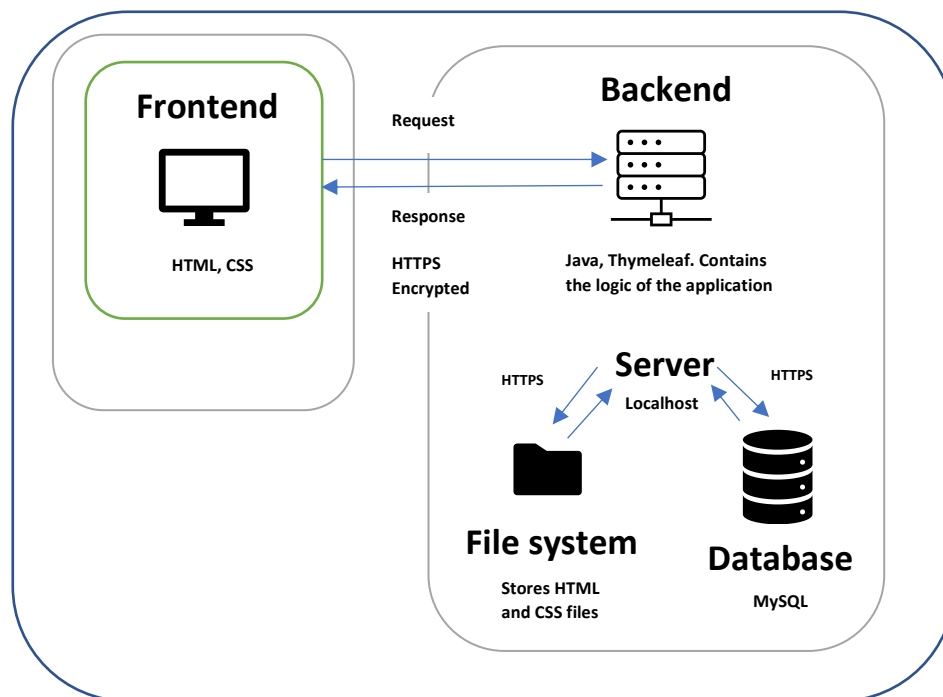


Figure 26: System Architecture Diagram

Figure 26 shows the System Architecture Diagram. There are 2 main components within the system architecture diagram, the frontend and the backend.

Frontend technologies

As this is a web application, HTML was chosen as the primary frontend technology, it is the standard markup language for web applications, supported by all browsers, is lightweight and the development process is very simple. Accessibility is a key value in web applications, as HTML is supported by all browsers, all users will be able to access the website, resulting in increased accessibility. The development process is fast as a program restart is not required to be able to see any changes. All it takes is a simple refresh of the browser window, the browser will fetch the updated HTML file from the web server and the new changes will be displayed. CSS is also a standard in the modern web, it allows more functionality and freedom in terms of the design of the frontend. I made the decision to not include JavaScript in this project as I believe the functionality of CSS is more than enough for the project's requirements. JavaScript may also hinder client-side security, as JavaScript code is visible to the user, it may be used for malicious purposes. Additionally, I also chose to not include JavaScript as a means of consistency. Different browsers interpret JavaScript differently, which may result in unwanted functionality across some browsers. On the other hand, CSS is interpreted the same across all modern browsers.

Backend technologies

The backend of an application is responsible for the logic of the application, processing data, responding to the requests of the frontend and providing it with functionality. There were a lot of options in terms of choosing the primary programming language, PHP, Python or Java are among the top 5 programming languages used in web applications [12]. Python has various web frameworks

available, the two options that I considered were Django and flask. I already had experience with flask after using it for multiple projects in the past, while Django was something new to me, I ultimately decided to not use python as I wanted to expand my web development skills beyond python and explore new methods in web development. Java was next in question, it is an object orientated programming language that I have previously had experience in, however I have never developed a web application with it. While a pure Java application is possible, a Java web framework can hugely increase the production speed. A web framework contains pre-written code that act as templates that can be used to create applications quickly and efficiently. Frameworks eliminate the need to create everything from scratch, which not only increases production speed, but is also much safer in terms of cybersecurity. The code snippets in frameworks have usually had years of development to ensure they are secure, if any vulnerabilities are discovered, a patch will automatically be rolled out. On the other hand, if I were to develop a pure Java application, maintenance against exploits would be much more of a manual job, sacrificing efficiency and ultimately the safety of the system. There are a variety of web frameworks available for Java, one of which I researched was Struts. Struts follows the Model-View-Controller model and extends the JSP API. An MVC model separates the application into three distinct components, the model, view and controller. Each component is responsible for different functions throughout the application. One of the main advantages of employing an MVC model is that it allows for parallel development. While this advantage is not as significant since this is an individual project, the MVC model helps to reduce code duplication, resulting in faster production times. Spring boot is an alternative Java web framework that is an extension of the popular Spring framework. Like Struts, Spring boot also implements an MVC model , although while Strut's modules are tightly coupled, Spring boot offers loosely coupled programming modules. In general, loosely coupled programming modules are better as they offer more flexibility and re-usability of the code as the modules are only dependent on interfaces, rather than classes. As a result of this, I came to the conclusion that I would use Spring boot as the web framework for my project. Additionally, there were more resources available for Spring boot when compared to those available for Struts.

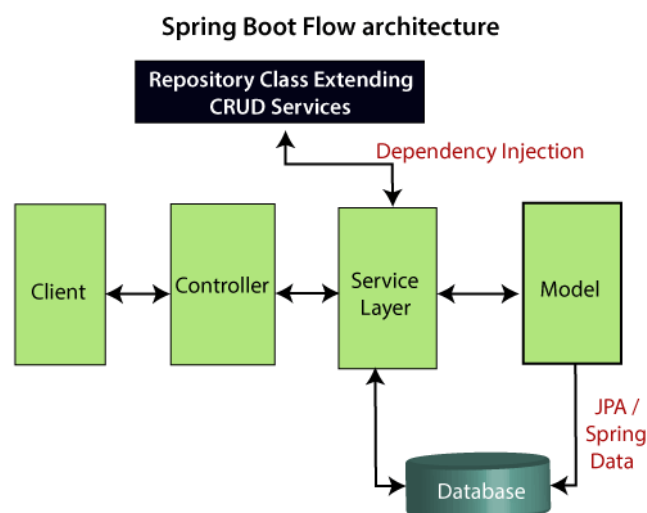


Figure 27: Spring boot flow diagram [13]

Figure 27 shows the basic architecture of a spring boot Java application, it consists of six main components. The model, or entity, defines the objects in the database and application. It sets out the attributes of the object, as well as its relationship between other objects. The use of the Java Persistence API (JPA) allows for automatic table generation based on the attributes and relationships defined within the models. Additionally, column names, data types and constraints can all be

specified in pure Java, with no need to write SQL commands to construct the database. This is one of the reasons why I chose spring boot. The repository is a mechanism which emulates a collection of objects, by extending CRUD services or JPA repositories, it allows CRUD operations to be completed on the objects contained within the repository. The objects can be injected into the service layer using the `@Autowired` annotation. This means that any methods defined in the repository class can then be used within the service layer. The service layer can be further broken down into two sub layers, the service and the service implementation. Firstly, the service class contains the definitions for the methods, while the service implementation contains the logic for these methods. The service implementation class implements the service class, while other dependencies can also be injected. The next component in the spring boot architecture is the controller layer. The controller layer is responsible for processing the incoming API requests, adding the appropriate objects to a model, and then returning this model as an HTML page. It is an intermediary layer between the client and service layer and prevents the client from directly interacting with the complex service layer.

Although the majority of the backend was implemented in Java, it was not the only language used. Thymeleaf played a major role in implementing the logic of the application. Thymeleaf is a Java template engine that is responsible for serving HTML at the view layer of MVC based projects. Additionally, it also assists in binding data that the client has inputted into a form, with objects defined in the model classes. It works in conjunction with the controller layer to output data stored in the database to the view which is then rendered as HTML on the clients side. JSP is another Java template engine, however I chose to use Thymeleaf as it utilizes 'natural templates' which look more like HTML and are easily readable compared to JSP files. While JSP is faster, it has a much steeper learning curve and does not support natural templates.

The application needs to be able to store data, so a database is required. The two most common types of databases are SQL and NOSQL. Some examples of SQL databases are MySQL and PostgreSQL. These are both relational database, they are made up of different tables, each consisting of rows and columns, the tables are linked via relationships. One of the main advantages of relational databases is that they can guarantee data accuracy. As all the tables are connected via primary and foreign keys, they are interrelated to each other, meaning that all the data store is non-repetitive. NoSQL databases are non-tabular databases and store information in JSON documents. They offer much higher performance than relational databases and are also much more scalable when employing techniques such as sharding. However, at the cost of scalability and performance, they often sacrifice Atomicity. MongoDB is one example of a NoSQL database that I have previously had experience with. In the end, I ultimately decided to use a relational database, in particular MySQL. I determined that for my application, atomicity is more important than scalability and performance. It is important to consider that while NoSQL databases are more easily scalable, it is not to say that relational databases are not scalable. My technology of choice was MySQL as this was something I had previous experience with, but also seemed to me the most appropriate choice.

1.8 Implementation

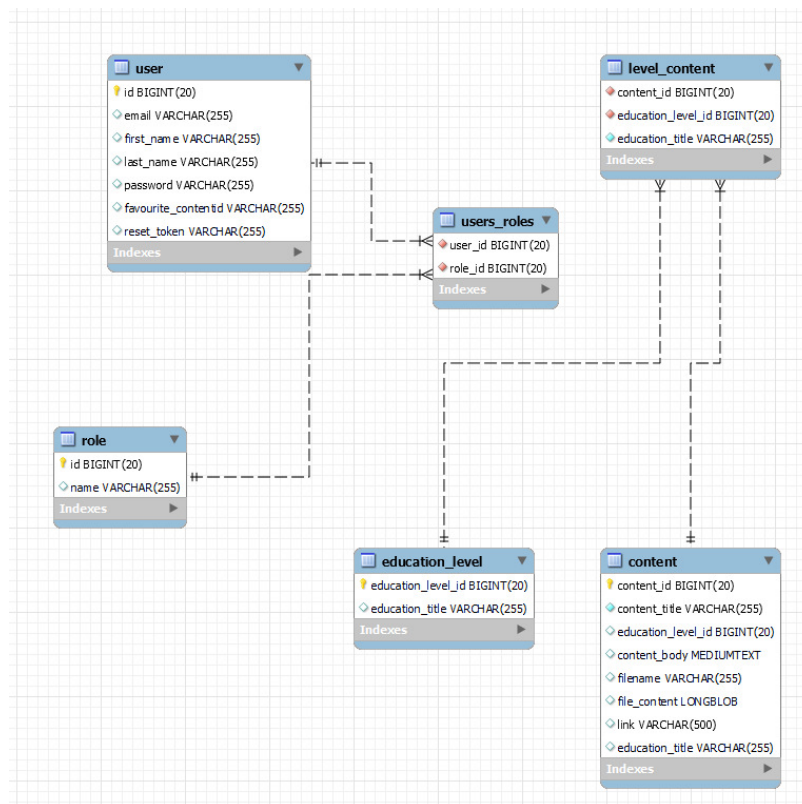


Figure 28: Entity relationship diagram

Figure 28 shows the entity relationship diagram of the MySQL database that I implemented. Each table has an integer as a primary key to identify the object in the database. The 'user' table features an email, first name, last name and password field with the VARCHAR datatype. For the password field, it was important that the database allowed large values as the password is not stored in plain text. The hash function converts the plaintext value inputted by the user, into a string of 56 characters. The string can contain any character, ranging from numbers and letters to special characters, so it was important that the database could handle these. The user table also contains two more fields, the 'favourite_contentid' and 'reset_token'. Unfortunately, I was not able to implement the functionality of these fields. The 'user' table is joined to the 'users_roles' table via a many to one relationship. I used a many to one relationship here to aid future development. In the future, it could be possible to add more roles than the original 3, perhaps granting educators new roles based on how much content they upload. This would be easier to implement as with the current system, a user is able to have more than one role, while roles are also able to be assigned to more than one user. The 'user_roles' table contains a list of all userID's paired to the ID of the role they currently have. I chose to join these two tables as one of the main advantages of a join is that it increases the performance of any queries being executed, instead of having the roles contained

within the 'user' table, I am able to separate them into two distinct tables. The 'education_level' table contains two fields, the education level ID and the education level title. The level ID is the primary key of the table and it is of a BIGINT data type. The primary key is automatically generated using JPA in spring boot and automatically selects the most appropriate data type and constraints. The education title is a VARCHAR, as the educator is able to choose a title of varying length containing a variety of different characters. The CHAR data type would not be suitable here as the character length of the education level title is unknown and varying in length. The 'content' table is made up of 8 attributes. The content_id is very much like the education_level_id in that it is a BIGINT and is automatically generated using JPA, it is the primary key of the table. The content_title is self-explanatory, it holds the title of the content. As this value can vary in length, I chose the VARCHAR datatype over the CHAR datatype. The 'content' table also contains a foreign key, the education level ID. Each content 'belongs' to an education level, the value of the education level ID within the content table determines which education level the content is displayed under. The 'content_body' field contains the main text for the content. When uploading content, an educator is able to enter text into a text area that is capable of containing several paragraphs of information. The MEDIUMTEXT data type can store up to 16,777,215 characters. I believe this was a suitable choice as the TEXT data type can only store 65,535 characters, while this will be sufficient for most cases, I did not want any educators to be limited in any way. The LONGTEXT data type is capable of storing up to 4,294,967,295 characters, which is much too large of a quantity. If somehow this capacity was regularly met by educators, then it would have a negative impact on the performance of the system. I believe the MEDIUMTEXT type is a compromise between the two and is therefore the most appropriate data type. The 'filename' field contains the name of the file for any files that the educator has attached. It has a data type of VARCHAR as with previous fields, the length of the data as well as the characters, is unknown. Additionally, the 'file_content' field, contains the bytes that make up the file. Without this field, the filename would exist and the user would be able to download it, however the file would not contain any content as the bytes that construct the file are not stored in the database. In order to store the file content, I have chosen a data type of LONGBLOB. BLOB stands for binary large objects, which is exactly what a file is. The LONG version of BLOB specifies a larger storage capacity of 4,294,967,295 bytes, whereas MEDIUMBLOB has a maximum capacity of 16,777,215 bytes. While this would be enough for PDF files etc, I want to allow educators to upload videos, since these are typically much larger, I decided that LONGBLOB would be more appropriate. Content is also able to contain a link to material outside of the website, as links can be very long, I opted for a VARCHAR with a max length of 500. While common search engines such as internet explorer support a max URL length of 2,083 characters, it is much more typical for URL's to be less than 120 characters long.

The 'content' and 'education_level' tables are joined via their primary keys to create a new table containing a list of education levels, along with the content that falls under each level.

Role-based login and registration system

In order to create a login and registration system, I first had to set up a connection with the database, this was done by adding some code to the application.properties file. This file contains the properties for the application, database connections can be added here, as well as configuring HTTPS connections etc.

```
spring.datasource.url=jdbc:mysql://csmysql.cs.cf.ac.uk:3306/c1932121_individual_project?useSSL=false
spring.datasource.username=c1932121
spring.datasource.password=HelloThere6401
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
```

Figure 29: Database connection in application.properties

Figure 29 shows how the database connection is established, it involves specifying a data source URL, username and password. The URL is the URL to the database, it features a host name and port followed by the name of the schema you wish to connect to. Hibernate is used to create the tables in the database from Java code, it is important to declare a hibernate dialect to use as this will determine the syntax of the SQL statements it generates. In this case, I am using MySQL 5.1, so I specify the MySQL5 dialect. Now the connection to the database has been made, it was time to construct the entities of the system. I started out with generating the 'user' and 'Role' entities. Each entity represents a table in the database, the table is automatically created upon execution of the application, this is achieved by utilizing the JPA. This technique is known as object relational mapping, each entity in the database is represented by an object in Java. ORM creates a layer between the programming language and the database, I can then query data in the database using Java. This is very important from a security point of view as SQL injection is not possible. SQL injection relies on extending on an SQL statement, but since there are no explicit SQL statements, the application is much less vulnerable to these types of attacks.

```
@Entity
@Table(name = "user", uniqueConstraints = @UniqueConstraint(columnNames = "email"))
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Email(message = "Email provided is not valid!", regexp = "(?:[a-z0-9!#$%&'*/=?")
    private String email;

    private String password;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.PERSIST)
    @JoinTable(
        name = "users_roles",
        joinColumns = @JoinColumn(
            name = "user_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(
            name = "role_id", referencedColumnName = "id"))

    private Collection<Role> roles;
```

Figure 30: User entity

Figure 30 shows a snippet of the user entity, I start off by declaring it as an entity by using the '@Entity' annotation. I am able to define the table name, as well as any unique constraints (primary key) I want to exist by using the '@Table' annotation. Every entity requires an ID, I have set the ID

using the '@Id' annotation, by changing the generation type to identity, for each user in the database, the ID is auto-incremented by 1. Additionally, in the user entity I am able to define the columns that are created in the database, I am able to set constraints as well as specifying data types. In this case, I wanted to add some constraints to provide multi-layer input validation. By adding the '@Email' annotation above the email variable, hibernate is made aware that this column should only contain data that matches the structure of an email address. However, by default, hibernate attempts to match the string with the regex expression of (.*). Regex is typically used to assist in matching patterns in a string, and with the correct expression, can be used to detect if a string matches a valid email pattern.

```
regexp = "(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\\. [a-z0-9!#$%&'*/+=?^_`{|}~-]+)*) *|\"(?:[\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21\\x23-\\x5b\\x5d-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f]) *\\") @ (?: (?: [a-z0-9] (?: [a-z0-9-] * [a-z0-9]) ? \\. ) + [a-z0-9] (?: [a-z0-9-] * [a-z0-9]) ? | \\. (?: (?: 25[0-5] | 2[0-4][0-9] | 01? [0-9] [0-9] ? ) \\. ) {3} (?: 25[0-5] | 2[0-4][0-9] | 01? [0-9] [0-9] ? | [a-z0-9-] * [a-z0-9] : (?: [\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x21-\\x5a\\x53-\\x7f]|\\\\[\\x01-\\x09\\x0b\\x0c\\x0e-\\x7f]) + ) \\. ) )"
```

Figure 31: Regex used to identify valid email address

Figure 31 shows the regex that I used to help validate the email field and ensure only valid email addresses can be stored in the database. Originally, I did not have regex implemented, instead I simply relied on front end input validation. However, when penetration testing my application I discovered a vulnerability. To ensure that the email entered contained an '@' symbol, the input type in the HTML was 'email'. Although, I discovered that the user could inspect the element using the developer tools of the browser and manually change the input type from 'email' to 'text'. This would remove the email constraint, allowing the user to create an account with an email of any series of characters provided it did not already exist. While I cannot prevent the user from inspecting the element, adding input validation on multiple layers means that changing the input type has no effect on the constraints as the constraints are defined in the back end as well.

Figure 30 also shows how I implemented a relationship between two entities. As for the reasons explained above, I decided to implement a many to many relationship between the user and role entities. I specified the fetch type to be eager which means that hibernate will fetch all elements of a relationship, not just the parent. The alternative, lazy, only fetches elements when needed. While lazy fetching offers better performance, the JDBC session must remain active when the entities need to be loaded, however this is not always possible. The session may end before the get() method has been called. By specifying a cascade type of all, when a user account is deleted from the system, the role ID that was associated with the user is also deleted. To create a table that contains the user ID's with the associated role ID's, I joined the role and user entities by the ID of each entity. The roles are then stored in a collection. The remainder of the user entity is made up of the appropriate constructors, getter, and setter methods.

As explained earlier, a repository emulates a collection of objects, and allows CRUD operations.

```
@Repository
public interface UserRepository extends CrudRepository<User, Long> {
    User findByEmail(String email);
}
```

Figure 32: User repository

Figure 32 shows a snippet of the user repository, the '@Repository' annotation declares the interface as a repository. The interface extends from the Crud repository which allows CRUD

operations to be performed on objects within the repository. I also define a method, 'findByEmail', this is equivalent to an SQL statement of 'SELECT u from User WHERE email =' '. This is where you can truly see the functionality of object relational mapping, there is no explicit SQL present for finding a user by an email, which means the application is much less vulnerable to SQL injection attacks.

```
public interface UserService extends UserDetailsService {  
    User save(UserRegistrationDto registrationDto);  
  
    List<User> listAll(String keyword);  
}
```

Figure 33: User Service interface

Figure 31 shows the UserService interface which extends upon the UserDetailsService. The UserDetailsService is a core interface within spring that retrieves the users authentication and authorization information. Within the UserService, there are two methods, a save method which is used to save the user info into the repository, and a list which is used to display a list of users. The purpose of these methods will become relevant later.

```
@Service  
public class UserServiceImpl implements UserService {  
    @Autowired  
    private UserRepository userRepository;  
  
    @Autowired  
    private BCryptPasswordEncoder passwordEncoder;  
}
```

Figure 34: Dependency injection within the UserServiceImpl class

Figure 34 shows a portion of the UserServiceImpl class, this is responsible for implementing the functionality of the log in and registration system. Here, the userRepository bean is injected, which allows the class to utilize some of the methods defined in the repository, as well as any CRUD operations as the repository extends from the CRUD repository. 'BCryptPasswordEncoder' is also injected which is a function that encrypts fields, typically passwords, by generating a random salt value to calculate the hash. This means that each time this function is called, a different hash value is generated even if the input is the same. As a result, the hash function is resistant to collision attacks as identical messages will always have different hash values.


```

@Override
public User save(UserRegistrationDto registrationDto){
    User email = userRepository.findByEmail(registrationDto.getEmail());

    User user = new User(registrationDto.getFirstName(),
        registrationDto.getLastName(), registrationDto.getEmail(),
        passwordEncoder.encode(registrationDto.getPassword()), Collections.singletonList(new Role( name: "ROLE_USER")));

    return userRepository.save(user);
}

```

Figure 35: User save method

Figure 35 shows how a user is saved into the repository and how the password is encoded into a secure form. I start off by declaring a new instance of user and use the get() methods set out in the user entity to get the attributes for the user from the input field in the frontend. At this stage, it was important to utilize the passwordEncoder function to hash the password value that was retrieved. I also assign the user the role 'ROLE_USER' by creating a new Role instance and adding it to the collection. I then call the userRepository save method and save the new user.

```

@Override
public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
    User user = userRepository.findByEmail(email);
    if(user == null ) {
        throw new UsernameNotFoundException("Invalid username or password.");
    }

    return new org.springframework.security.core.userdetails.User(user.getEmail(), user.getPassword(), mapRolesToAuthorities(user.getRoles()));
}

private Collection<? extends GrantedAuthority> mapRolesToAuthorities(Collection<Role> roles){
    return roles.stream().map(role -> new SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
}

```

Figure 36: Log in implementation

Figure 36 shows how the login system is implemented. The user object is returned to spring security where it validates the user object against the credentials entered by the user. It then fetches the role of the user as well as the authorities that come with that role.

```

<input id="firstname" class="input" type="text" th:field="*{firstName}" placeholder="Enter First Name " required />

```

Figure 37: Thymeleaf to bind input with object

Figure 37 shows how the user's input is bound to the user entity. By creating a thymeleaf field and specifying which attribute the input field corresponds to, provided the input meets all validation requirements, the input is added to the correct field in the database.

```

@GetMapping
public String showRegistrationForm() { return "registration"; }

@PostMapping
public String registerUserAccount(@ModelAttribute("user") UserRegistrationDto registrationDto) {
    User email = userRepository.findByEmail(registrationDto.getEmail());

    if (email != null ){
        return "redirect:/registration?error";
    }
    userService.save(registrationDto);
    return "redirect:/registration?success";
}

```

Figure 38: UserRegistrationController snippet

Figure 38 shows a portion of the UserRegistrationController. As mentioned earlier, a controller process incoming HTTP requests, adds objects to a model and then returns the model to the user in the form of HTML. The '@GetMapping' annotation handles the HTTP get requests, in this case, when the showRegistrationForm method is called, the registration page view is returned. The '@PostMapping' annotation handles the post requests, usually from input forms such as the registration form. By using @ModelAttribute, I am able to supply the User object to the controller. I define the 'email' variable equal to the result of the findByEmail method. This method is present in the user repository, I can get the value from the input field by calling the getEmail method. This sets the email variable equal to whatever the result of the query of the repository is. Therefore, if email is not equal to null, then the email was found to already exist in the user repository. This means that an account already exists with that email so a user should not be able to make a new account with that same email. To alert the user of this, I return a redirect to an error page which displays the appropriate error. On the other hand, if the result of the query is null, then the account does not exist and the data is saved into the repository using the userService. The user is then redirected to a success page where an alert is displayed stating that they have successfully registered.

Security configuration and role-based permissions

In spring boot projects, it is possible to configure the security of the system by implementing a security configuration class. Within this class it is possible to permit certain roles to access specific pages, add HTTP headers and add a host of other security features.

```

@Configuration
@EnableWebSecurity

```

Figure 39: Web security annotation

The '@EnableWebSecurity' annotation present in figure 39 automatically enables the web securities defined by the web security adapter within spring security. By enabling this, it provides protection against cross site forgery attacks. CSRF attacks work by forcing the user to submit a state-changing request such as changing a password or sending a message. If a user clicks a malicious URL that contains an unauthorized request for the target website. The browser of the unsuspecting victim then sends this request to the target website. While the malicious website does not have direct access to the target site, it has access to the users session cookies which can contain the credentials to the website. The target application, if open, will treat this malicious request as legitimate and respond accordingly. When a web application is vulnerable to CSRF attacks, the browser fails to distinguish if requests are coming from external websites. Enabling web security means that my

application is able to reject requests coming from external websites. It does this by generating a synchronizer token pattern. This ensures that every request contains a randomly generated token as an HTTP parameter. Upon receiving the request, the web server then compares the value to the expected result and rejects the request if there is no value or the value does not match the expected value. This means that the application is able to distinguish between legitimate requests by the user, and malicious requests from external web applications. Since I am using Thymeleaf, this token is automatically added to the form as a hidden input field.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .headers() .headersConfigurer<HttpSecurity>
        .xssProtection() .headersConfigurer<...>.XXsConfig
        .and() .headersConfigurer<HttpSecurity>
        .contentSecurityPolicy( policyDirectives: "form-action 'self'");
}
```

Figure 40: HTTP configuration

Figure 40 shows HTTP configuration of the application. I have added some security headers to the HTTP configuration. The 'xssProtection' header adds a layer of protection against cross-site scripting attacks. It does this by utilising the xss auditor in most modern browsers, adding the ability to tell the browser to not render any scripts attempted to be injected. However, not all browsers have the ability to do this as they cannot make use of the xssProtection header. Therefore, I added another layer of protection by configuring a content security policy header. All browsers can make use of the content security header, which allocate trusted sources and block any scripts from untrusted sources. In this case, I define the only trusted source to be the application itself.

```
.antMatchers( ...antPatterns: "/adminPage").hasRole("ADMIN")
.antMatchers( ...antPatterns: "/adminViewLevelSelect").hasRole("ADMIN")
.antMatchers( ...antPatterns: "/adminViewAddLevel").hasRole("ADMIN")
.antMatchers( ...antPatterns: "/index/{education_level_id}/uploadContent").hasAnyRole( ...roles: "ADMIN", "EDUCATOR")
.antMatchers( ...antPatterns: "/deleteContent/{education_level_id}").hasRole("ADMIN")
.antMatchers( ...antPatterns: "/update/{education_level_id}").hasRole("ADMIN")
.anyRequest().authenticated()
.and() HttpSecurity
.formLogin() FormLoginConfigurer<HttpSecurity>
.loginPage("/login")
.permitAll()
```

Figure 41: Configuring access permissions based on role in SecurityConfiguration.java

Figure 41 shows how the role-based access permissions are configured. This is done by using the antMatchers() method under the HttpAuthorizeRequests method. By defining the URL and adding the name of the role within the hasRole() method, I can permit certain roles to be able to access certain URL's. In the case where I want to allow multiple roles, I used the hasAnyRole method and inputted a list of the roles I want to be able to access the URL. The use of the anyRequest() and authenticated() methods restricts the access for any endpoint apart from the public URL, it also means that the user must be authenticated in order to access the URL's.

Homepage

Like the user and role entities, I made the education level an entity and created its own repository. The education level repository extended from the crud repository like with the previous examples.

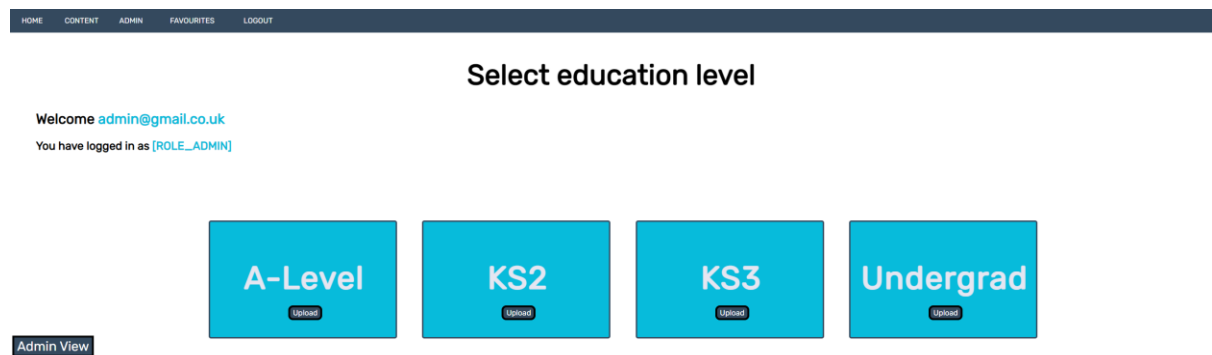


Figure 42: Homepage

Figure 42 shows the homepage, it features a list of education levels, as well as other options such as admin view or upload. I could have simply hard coded each education level into the HTML but since I wanted the admins to be able to add, edit and delete education levels, this approach would not work. Instead I would have to read the data from the database and display it. To allow for these operations, I declared the following methods in the education level service class.

```
public interface EducationLevelService {
    EducationLevel getEducationLevelByID(long education_level_id);
    void deleteEducationLevelByID(long education_level_id);
    void saveEducationLevel(EducationLevel educationLevel);
}
```

Figure 43: EducationLevelService.java

The first method defined allows me to search for an education level when given the ID, the second one deletes an education level by ID, and the final one saves a new instance of the education level object.

```
@Override
public EducationLevel getEducationLevelByID(long education_level_id) {
    Optional <EducationLevel> optional = educationLevelRepository.findById(education_level_id);
    EducationLevel educationLevel = null;
    if (optional.isPresent()){
        educationLevel = optional.get();
    } else {
        throw new RuntimeException(" Education Level not found!");
    }
    return educationLevel;
}
```

Figure 44: GetEducationLevelById method in EducationLevelServiceImpl.java

Figure 44 shows how the getEducationLevelById method is implemented. I use the optional type to convey to spring that the return may be empty, without using null. I use the findById method within the education level repository to search the repository for the ID that is passed in as a parameter. If there is no education level that matches the ID passed in as a parameter, then a run time exception

is thrown stating that the education level has not been found. However, if the optional variable is present and not empty, then it will fetch the education level object and return it.

```
@Override
public void deleteEducationLevelById(long education_level_id) {
    this.educationLevelRepository.deleteById(education_level_id);
}

@Override
public void saveEducationLevel(EducationLevel educationLevel) {
    this.educationLevelRepository.save(educationLevel);
}
```

Figure 45: Delete and save education level methods

Figure 45 demonstrates how the save and delete education level methods have been implemented. They simply utilize the save and delete methods featured in the education level repository by extending the crud repository.

```
@GetMapping("/")
public ModelAndView getContent(){
    ModelAndView modelAndView = new ModelAndView( viewName: "index");
    modelAndView.setViewName("index.html");
    modelAndView.addObject( attributeName: "EducationLevelList", educationLevelRepository.findAll());
    return modelAndView;
}
```

Figure 46: getContent() method within EducationLevelController.java

Figure 46 shows how the homepage view is returned along with the education levels as objects. It utilizes the MVC model of spring. By setting the view name as 'index' it tells spring boot to return that html page when the model and view is returned. In order for it to display the education levels, these must be added as objects to the view and this is done by querying the repository and calling the findAll() method. This adds all the objects to a list called 'EducationLevelList'. Then, in order to display the education levels in the HTML page, I utilize a thymeleaf for loop to iterate through the list of education levels.

```
<th:block th:each="EducationLevel : ${EducationLevelList}">
```

Figure 47: Thymeleaf for loop to display education levels

Figure 47 demonstrates the for loop mentioned above, this segment of code is equivalent to the pseudo code of 'for education level in EducationLevelList:'. Within the thymeleaf block that I have defined, I put the remainder of the HTML responsible for creating the grid like layout of education levels. Once clicked, each education level redirects to the same HTML page, but a different model containing the appropriate content. For example, if I were to click 'KS3', it would redirect me to a page that only displays content that falls under KS3. Figure 48 shows how this is achieved by using path variables.

```

@GetMapping("/{education_level_id}")
public ModelAndView findContentByID(@PathVariable(value = "education_level_id") long education_level_id){
    ModelAndView modelAndView = new ModelAndView("contentPlaceholder");
    modelAndView.setViewName("contentPlaceholder.html");

    modelAndView.addObject("EducationLevelList", educationLevelService.getEducationLevelByID(education_level_id));

    modelAndView.addObject("ContentList", contentRepository.findAllByeducationlevelid(education_level_id));
    return modelAndView;
}

```

Figure 48: Controller responsible for displaying the correct content

Once an education level has been clicked, the user is redirected to the URL '/index/education level' where education level is the ID of the education level they have clicked on. I then passed this ID into the method by utilizing the @PathVariable annotation, and assigning the value to be equal to 'education_level_id' which is an attribute of the education level entity. This variable is passed into the findAllByEducationLevelID method which in turn searches the content repository for all content with an education level ID that matches the path variable. All objects that match the ID are added to the model and view which is then returned. The view name has been assigned to 'contentPlaceholder.html' which means the 'contentPlaceholder' page is returned with the objects that have been added to the view.

```

@GetMapping("/deletelevel/{education_level_id}")
public String deleteLevel(@PathVariable("education_level_id") long education_level_id, Model model) {
    EducationLevel educationLevel = educationLevelRepository.findById(education_level_id)
        .orElseThrow(() -> new IllegalArgumentException("Invalid education level:" + education_level_id));
    educationLevelRepository.delete(educationLevel);
    return "redirect:/";
}

```

Figure 49: Delete method within the education level controller

Figure 49 shows the delete method present within the education level controller. Like the previous figure, I utilize the path variable annotation. The ID of the education level that the user wants to delete will be present in the URL. This value is then passed into the findById method and the education level repository is searched for an education level with the same ID. If an education level is not returned with the same ID then an illegal argument exception is thrown. When a matching education level has been found, the object is passed into the delete method and the instance is deleted from the repository. After deleting the object, I return the user to the homepage by redirecting them to the '/' URL.

```

@GetMapping("/edit/{education_level_id}")
public String showUpdateForm(@PathVariable("education_level_id") long education_level_id, Model model) {
    EducationLevel educationLevel = educationLevelRepository.findById(education_level_id)
        .orElseThrow(() -> new IllegalArgumentException("Invalid education level:" + education_level_id));

    model.addAttribute("educationLevel", educationLevel);
    return "updateEducationLevel";
}

```

Figure 50: Show update form method

In order to edit the title of an education level, the user is redirected to a form that allows them to input the new title of the education level. Figure 50 shows how the user is redirected to an update form for the specific education level that they wish to delete. Again, the path variable annotation is

utilized here. The ID of the education level that the user selects is passed to the URL, which in turn, is passed to the `findById` method. The education level that has been returned from this method is then added as an attribute to the model. Once the attribute has been added to the model, the view is returned to the user in the form of HTML and the update form is displayed.

```
@PostMapping("/update/{education_level_id}")
public String updateLevel(@PathVariable("education_level_id") long education_level_id, EducationLevel educationLevel,
                          BindingResult result, Model model) {
    if (result.hasErrors()) {
        educationLevel.setEducation_level_id(education_level_id);
        return "updateEducationLevel";
    }

    educationLevelRepository.save(educationLevel);
    return "redirect:/adminViewLevelSelect";
}
```

Figure 51: Update education level post method

Once the update form has been displayed, the form must have the required functionality of being able to save the update title of the education level. Figure 51 demonstrates how this is implemented. The post mapping annotation is used here because the form will utilize a post method. The path variable contains the ID of the education level, by using the ID, I can ensure that any changes made to the title are only reflected in the title of the level that was changed. The new instance of the education level object is then saved to the repository.

Uploading and viewing content

```
@PostMapping("/saveContent")
public String saveContent(@ModelAttribute("content") Content content, @RequestParam("file") MultipartFile file) throws IOException {

    String fileName = file.getOriginalFilename();
    content.setFileContent(file.getBytes());
    content.setFilename(fileName);

    contentService.saveContent(content);
    return "redirect:/";
}
```

Figure 52: Save content controller method

In order to save content to a repository, a form must first be displayed to allow the user to enter the data. This is achieved using a similar method to the education levels, utilizing the same path variable technique. Figure 52 shows how the content upload ability is implemented, it provides the backend functionality for the upload button. The use of 'MultipartFile' allows standard form data (content title etc) to be uploaded with the file attachment data in a single post method. The `getOriginalFilename()` method is found within spring's MultipartFile class, I set the filename using this method. The `getBytes()` method is also found in this class and is used to fetch the bytes of the file that has been uploaded. The `setFileContent()` method that is declared within the content entity is then used to assign it to the result of the `getBytes()` method. The same is done with the filename, setting it equal to the filename variable that is the result of the `getOriginalFilename()` method. Finally, the last step is to save the object using the `saveContent()` method set in the service class. Within the form in the HTML page, it is important to set the enctype to the following:

```
enctype="multipart/form-data"
```

In HTML forms, the data that forms the body of the request must be encoded. Specifying the encode type to be multipart means that the form is able to accept file uploads. It is important to add this encode type when the form contains an input of type 'file'.

When the file has been uploaded, it needs to be downloadable by all users. In order to display the unique content, I employ the same technique that was used to display the content titles for each education level. By taking advantage of springs MVC model, I can search the content repository by the content ID and add the content that was returned, to a list. This list is then added to the model which in turn is rendered in the HTML page.

```
<table border="2" id="content">
  <thead>
    <tr th:each="Content:${ContentBodyList}">
      <th class="contentHeader" th:text="${Content.content_title}">
    </th>

  </tr>
</thead>
<tbody>

<tr th:each="Content : ${ContentBodyList}">
  <td th:text="${Content.content_body}"></td>
```

Figure 53: Thymeleaf snippet displaying content

Figure 53 shows how the content is displayed in a table format. It iterates through the content list that was defined in the content controller and assigns the table header value to be equal to the title of the content. To do this, I specified the object along with the object attribute I want to be displayed, in this case 'content_title'. I applied the same concept to the content body below. However, there are still two more attributes that have not been displayed yet, the file and the link. Displaying the link is a very similar process to displaying any other attribute, but it needs to be made clickable.

```
<a th:target="_blank" class="file" th:href="${Content.link}"
th:text="${Content.link}" onclick="return confirm('You are leaving this
website and following an external link, are you sure you want to leave?')">
</a>
```

The above shows how I implemented a clickable link that is read from the database. It is all contained within the <a> tag as it is a link, the 'th:target='_blank'' ensures that once the link is clicked, the material opens in a new tab while keeping the existing one open. I added the 'th:href' element to ensure that the link was clickable, to fetch the link I specified the link attribute of the Content object. To aid in the security of the system, I wanted to implement a warning that would alert if the user clicked a link that led to an external site. To integrate this, the 'onclick' element was used to

return a confirmation box with a custom message. This message box contained two options, one to continue following the link and accepting the risk, or another to close the message box and remain on the site.

HTTPS Security

For the purpose of this project, I generated a self-signed certificate. In general, self-assigned certificates would not be used as they contain both the private and public keys within the same entity. Self-signed certificates are not trusted by browsers, causing the browser to display a message stating that the site is not secure. In order to generate an SSL certificate, I used keytool. Java keytool is a repository of security certificates and allows the creation of keystores.

```
keytool -genkeypair -alias cyber -keyalg RSA -keysize 4096 -storetype PKCS12 -  
keystore cyber.p12 -validity 3650 -storepass password
```

Figure 54: Command to generate keystore

Figure 54 shows the command I used to generate the keystore. I chose to generate a keystore in the PKCS12 format as that is the industry standard. The alternative is a JKS, which is specific to java, by using a PKCS12 format I can ensure the key can function with any language. The term 'genkeypair' signals to keytool that I want to generate a keystore. I assign the alias 'cyber' by using the 'alias' command, this is the name of the certificate. '-keyalg' specifies the algorithm used to generate the key pair, in this case I am using RSA with a key size of 4096 bits. To ensure I am generating a certificate in the PKCS12 format, I assign the store type to PKCS12. I am also able to specify how many days I want the certificate to be valid for, I chose 3650 days although a third party cannot issue SSL certificates for more than 13 months at a time due to security reasons. After executing the command and filling in the appropriate data that keytool requests, a file, cyber.p12 is generated.

In order for the application to use the HTTPS connection, spring boot must be configured to utilize it.

```
server:  
  ssl:  
    server.port=8443  
    server.ssl.key-store=classpath:cyber.p12  
    server.ssl.key-store-password=password  
  
    server.ssl.keyStoreType=PKCS12  
    security.require-ssl=true
```

Figure 55: HTTPS configuration in application properties





Figure 55 shows how spring is configured to use the HTTPS connection running on port 8443. The keystore, cyber.p12, is within the resources of the spring boot file which allows the application to access it.




Overview of security features implemented:


- Enabled web security
- Spring security to hash the password credentials
- Invalidate HTTP session after logging out
- Content security policy to defend against XSS and CSRF attacks



- Object-relational mapping and parametrized queries to defend against SQL injection
- HTTPS connection
- SSL connection to database
- Multi-layer input validation




2.1 Evaluation



Test Case ID: 1		Test Purpose: Sign up as user		
Preconditions: Must be on the cybersecurity portal website with an active internet connection.				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Click the 'Register here' button on the log in page	User is directed to the Sign-Up page.	User is directed to the Sign-Up page.	
2	User fills out the form; entering their first name, last name, email and password and clicks the 'Register' button.	The details entered are unique and do not match any existing account. A message pops up to confirm the registration was successful. Account details are added to database	The details entered are unique and do not match any existing account. A message pops up to confirm the registration was successful. Account details are added to database	
Alternate Flow: User enters credentials already associated with an account				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Click the 'Register here' button on the log in page	User is directed to the Sign-Up page.	User is directed to the Sign Up page.	
2	User fills out the form; entering their first name, last name, email and password and clicks the 'Register' button. The email will already be associated with an existing account	An error message displays stating that an account with that email already exists and the details are not added to the database	No error is displayed but the details are not added to the database	




Test Case ID: 2		Test Purpose: Sign in		
Preconditions: Must be on the cybersecurity portal website with an active internet connection.				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Enters the correct email and password combination that is stored in the database and clicks the 'Log in' button	User is logged in and is redirected to the home page where it says which role their account has	User is logged in and is redirected to the home page where it says which role their account has	
Alternate Flow: User enters incorrect credentials				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	User enters details that are not recognised in the system and presses the 'Log in' button	User is directed to the Sign Up page.	User is directed to the Sign Up page	
2	User fills out the form; entering their first name, last name, email and password and clicks the 'Register' button. The email will already be associated with an existing account	An error message displays stating that the email or password is invalid	An error message displays stating that the email or password is invalid	






Test Case ID: 3		Test Purpose: Content only associated with the education level selected is displayed		
Preconditions: Must be logged in and on the homepage				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks on an education level, in this case 'KS2'	User is redirected to a page that displays a list of content with an education level of 'KS2'	User is redirected to a page that displays a list of content with an education level of 'KS2'	




Test Case ID: 4		Test Purpose: Admins are able to delete content		
Preconditions: Must be logged in as an admin, previously selected an education level and currently on the content list view				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Under admin options, clicks the 'delete' button that corresponds to the content they wish to delete	Is redirected to the homepage and the content is deleted from the database	Is redirected to the homepage and the content is deleted from the database	
2	Clicks on the education level	The content that was deleted is no longer displayed in the list of content	The content that was deleted is no longer displayed in the list of content	




Test Case ID: 5		Test Purpose: Search for content		
Preconditions: Must logged in				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'content' button on the navigation bar at the top of the page	Is redirected to a page that displays all content on the website	Is redirected to a page that displays all content on the website	
2	Types in the name of the content in the search bar	Text appears in search bar	Text appears in search bar	
3	Presses the search button located next to the search bar	Content that does not match the search criteria is filtered out and only relevant content is displayed	Content that does not match the search criteria is filtered out and only relevant content is displayed	





Test Case ID: 6		Test Purpose: Viewing content		
Preconditions: Must be logged in and on homepage				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the desired education level	Is redirected to a page that displays a list of content relevant to the education level previously selected, as well as various other options	Is redirected to a page that displays a list of content relevant to the education level previously selected, as well as various other options	
2	Clicks on the 'View' button on the desired content	The title of the content is displayed at the top of the page, with the main body containing any text also being displayed. At the bottom of the page, any files that have been uploaded are displayed	The title of the content is displayed at the top of the page, with the main body containing any text also being displayed. At the bottom of the page, any files that have been uploaded are displayed	






Test Case ID: 7		Test Purpose: Downloading file from content view		
Preconditions: Must be logged in and on homepage				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the desired education level	Is redirected to a page that displays a list of content relevant to the education level previously selected, as well as various other options	Is redirected to a page that displays a list of content relevant to the education level previously selected, as well as various other options	
2	Clicks on the 'View' button on the desired content	The title of the content is displayed at the top of the page, with the main body containing any text also being displayed. At the bottom of the page, any files that have been uploaded are displayed	The title of the content is displayed at the top of the page, with the main body containing any text also being displayed. At the bottom of the page, any files that have been uploaded are displayed	
3	Locates the file and clicks on the name of the file	File starts to download and the user can open the file	File starts to download and the user can open the file	



Test Case ID: 8		Test Purpose: Uploading content		
Preconditions: Must be logged in as either an admin or an educator				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the upload button on the education level card that the user wishes to upload content to	A form is displayed that allows the user to enter the title of the content, the main body of the content, and upload a file	A form is displayed that allows the user to enter the title of the content, the main body of the content, and upload a file	
2	Presses the 'choose file' button	File browser is opened	File browser is opened	
3	Selects a file to be uploaded	File name is displayed in the form	File name is displayed in the form	
4	Clicks the 'Upload' button	Content is uploaded and is redirected to the home page	Content is uploaded and is redirected to the home page	
5	Selects the education level that the content was uploaded to	A list of relevant content is displayed, the content that was just uploaded is also displayed	A list of relevant content is displayed, the content that was just uploaded is also displayed	




Test Case ID: 9		Test Purpose: Adding new education level		
Preconditions: Must be logged in as an admin and on homepage				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the ‘admin’ button on the bottom left of the screen	Redirected to an admin view of the homepage with additional options	Redirected to an admin view of the homepage with additional options	
2	Presses the ‘add new level’ button	A form is displayed where the user can enter the education title	A form is displayed where the user can enter the education title	
3	Enters the title and selects the ‘add’ button	Returned to the homepage where the new level is visible	Returned to the homepage where the new level is visible	


Test Case ID: 10		Test Purpose: Admin deleting accounts		
Preconditions: Must be logged in as an admin				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'admin' button on the navigation bar at the top of the screen	Redirected to an admin view page that displays a list of accounts	Redirected to an admin view page that displays a list of accounts	
2	Enters a name in the search bar and clicks the search button	Users not matching the criteria are filtered out and the name searched is displayed	Users not matching the criteria are filtered out and the name searched is displayed	
3	Clicks the 'delete' button on the account details	Returned to the homepage and the account is deleted from the database. The user credentials cannot be used to log in to the website	Returned to the homepage and the account is deleted from the database. The user credentials cannot be used to log in to the website	

Test Case ID: 11		Test Purpose: Only admins can access services that require an admin account		
Preconditions: Must be logged in as either an educator or a user				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'admin' button on the navigation bar at the top of the screen	An error message displays stating that they 'do not have the necessary privileges to access this content'	An error message displays stating that they 'do not have the necessary privileges to access this content'	
Alternate flow 1: Attempts to access admin view on homepage				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks on the 'admin' button on the bottom left of the homepage	An error message displays stating that they 'do not have the necessary privileges to access this content'	An error message displays stating that they 'do not have the necessary privileges to access this content'	
Alternate flow 2: Attempts to delete content				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks 'content' in the navigation bar	All content is displayed as well as various other options such as view and delete	All content is displayed as well as various other options such as view and delete	
2	Clicks the 'delete' button under admin options in the table	An error message displays stating that they 'do not have the necessary privileges to access this content'	An error message displays stating that they 'do not have the necessary privileges to access this content'	

Test Case ID: 12		Test Purpose: Only admins and educators can upload content		
Preconditions: Must be logged in as a user				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'upload' button on an education level card	An error message displays stating that they 'do not have the necessary privileges to access this content'	An error message displays stating that they 'do not have the necessary privileges to access this content'	
Alternate flow 1: Logged in as educator				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'upload' button on an education level card	A form is displayed that allows the user to enter the title of the content, the main body of the content, and upload a file	A form is displayed that allows the user to enter the title of the content, the main body of the content, and upload a file	
2	Selects 'choose file'	File browser is opened	File browser is opened	
3	Selects a file to upload	File name is displayed on form	File name is displayed on form	
4	Clicks the 'upload' button	Content is uploaded to the database and is redirected to the homepage	Content is uploaded and is redirected to the homepage	

Test Case ID: 13		Test Purpose: Logging out		
Preconditions: Must be logged in				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'log out' button on the navigation bar	Is logged out of the session and redirected to the log in screen. A message is displayed stating that the log out was successful	Is logged out of the session and redirected to the log in screen. A message is displayed stating that the log out was successful	
2	Uses back button in browser to attempt to return to application in a logged in state	Remains on log in screen	Remains on log in screen	

Test Case ID: 14		Test Purpose: Editing education level		
Preconditions: Must be logged in as an admin				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the 'Admin view' button on the bottom left of the homepage	Is navigated to an admin view of the home page with additional options visible	Is navigated to an admin view of the home page with additional options visible	
2	Clicks the edit symbol below the trash icon on the education level to be deleted	A form is displayed with the name of the current education level	A form is displayed with the name of the current education level	
3	Enters new name and clicks the 'Confirm changes' button	Redirected to homepage and any changes are visible	Redirected to homepage and any changes are visible	

Test Case ID: 15		Test Purpose: Adding content to favourites		
Preconditions: Must be logged in and on the content display page				
Test Case Steps:				
Step No.	Procedure	Expected Output	Actual Output	Pass/Fail
1	Clicks the star in the top right of the display	Star turns yellow and notification is displayed stating that the content has been 'added to favourites'	No change in system state	

Verifying HTTPS connection

```
Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\Alex>
>> [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")
>> [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")
>>
>> $dialog = New-Object System.Windows.Forms.Form
>> $dialog.Text = "Enter SQL Server instance name:"
>> $dialog.Size = New-Object System.Drawing.Size(400,100)
>> $dialog.StartPosition = "CenterScreen"
>>
>> $check = New-Object System.Windows.Forms.Button
>> $check.Location = New-Object System.Drawing.Size(250,20)
>> $check.Size = New-Object System.Drawing.Size(75,23)
>> $check.Text = "Check"
>> $check.Add_Click({$x=$input.Text;$dialog.Close()})
>> $dialog.Controls.Add($check)
>>
>> $input = New-Object System.Windows.Forms.TextBox
>> $input.Location = New-Object System.Drawing.Size(40,20)
>> $input.Size = New-Object System.Drawing.Size(200,20)
>> $dialog.Controls.Add($input)
>>
>> $dialog.Add_Shown({$dialog.Activate()})
>> [void] $dialog.ShowDialog()
>>
>> $x
>>
>> #
>> $script = Invoke-Sqlcmd -Query "SELECT DISTINCT encrypt_option
>> FROM sys.dm_exec_connections WHERE session_id = @@SPID" -ServerInstance $input.Text
>> $wshell = New-Object -ComObject Wscript.Shell
>> $wshell.Popup($script.ItemArray,0,"Connection encryption enabled for instance " + $input.Text + ":")
```

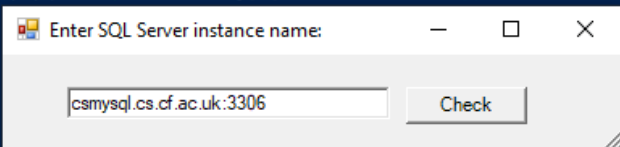


Figure 56: PowerShell command used to verify SSL connection

In order to check that the connection to the SQL server is secure, I used windows PowerShell. Figure 56 shows the command I used. Upon execution of the command, a window is displayed which allows me to enter the name and port of the SQL server.

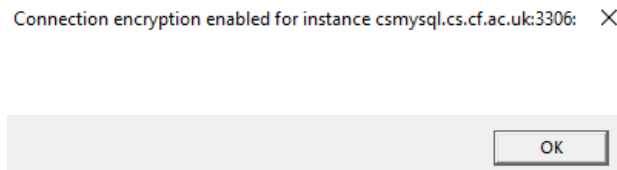


Figure 57: Confirmation that SSL connection is active

After inputting the SQL server name and pressing the 'check' button, confirmation is received that the SSL connection is active, as shown in figure 57.

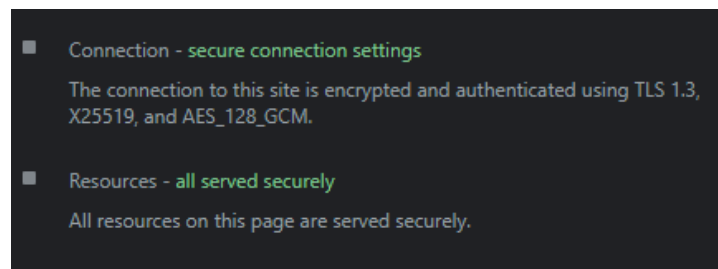


Figure 58: Browser dev tools showing secure connection

Figure 58 shows that chrome verifies that the connection is secure, proving that the HTTPS connection is active.

Ensuring resistance against SQL injection

In order to verify that my application is secure against SQL injection attacks, I must perform them myself. I will simulate an attack which will involve attempting to gain access to an admin account when the admin email is known but the password is not. Below is the SQL injection command I used.

" ' or 1=1 -- "

I then entered the admin email address into the email field and inputted the above command into the password field. If my application was vulnerable to SQL injection attacks, this would bypass the authentication system and allow me to log in as an admin without knowing the password. SQL injection can bypass the authentication system by extending on the explicit SQL statements present in the application. Sometimes the log in system may contain a similar SQL statement to the one below.

```
'SELECT U FROM Users WHERE email = 'admin@gmail.co.uk' AND password = 'admin123' '
```

If the SQL injection was successful, the application would perform the following query.

```
'SELECT U FROM Users WHERE email = 'admin@gmail.co.uk' OR 1=1--' AND password = 'admin123' '
```

However, due to the sequence of hyphens (--), the remainder of the statement is ignored. As 1=1 is always true, the statement does not check if the password entered is correct, resulting in the attacker bypassing the log in system.

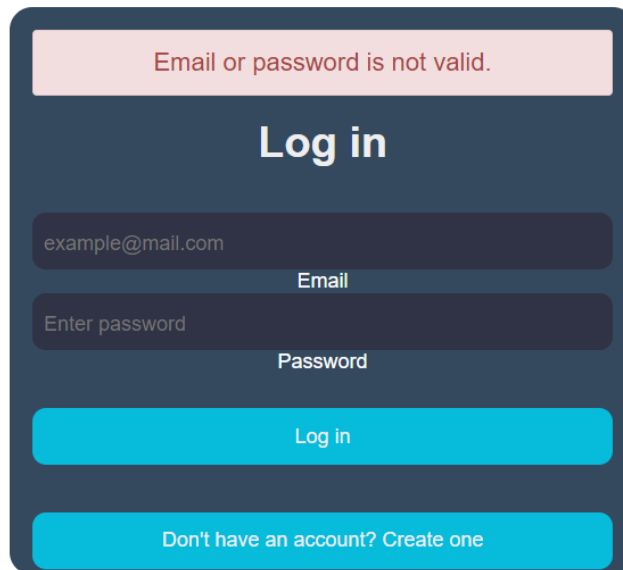


Figure 59: Log in system rejecting SQL injection attempt

Fortunately, this is not the case with my application. As it utilizes object-relational mapping, there is no explicit SQL statement to extent upon, meaning the password is identified to be invalid and prevents the attacker from entering the system. Figure 59 shows the error message that is displayed after attempting to log in with incorrect credentials.

Verifying resistance to cross-site scripting (XSS) attacks

Like with SQL attacks, in order to test my applications resistance to XSS attacks, I must conduct my own. XSS attacks can vary in complexity, if an application is vulnerable to a simple XSS script, it is also vulnerable to the most complex script. So for the sake of evaluation, I will conduct a simple, reflected XSS attack.

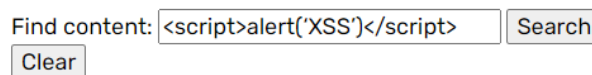


Figure 60: Script inserted into search box

Figure 60 shows the script inserted into the search box. Upon completion of a successful XSS attack, an alert would pop up in the centre of the screen, with the text 'XSS'. Fortunately, due to the content security policy and the web security annotation, the XSS attack was unsuccessful.

Achievement of deliverables and objectives

Based on the test cases above, I believe that the implementation fully satisfies the deliverables and requirements set out in the previous sections/initial plan. The majority of test cases have passed, with the only failures being additional features that were not deemed to be necessary. However, the success of the project cannot only be measured on the functionality that has been successfully implemented. It is important to assess whether the objectives have been met.

Objective 1 - Review relevant services/literature

I believe this objective has been met, I reviewed relevant services which involved describing the features that the service offers, as well as the downfalls of the service. I did this for multiple services and used this information to better define the requirements of my application. I incorporated the most common features of the services I reviewed but also improved upon areas where I think they

were lacking. However, due to the niche nature of the project, I struggled to find many relevant services and as a result the sample size for the services I reviewed was quite limited. Although I believe that with a larger sample size, I do not think the requirements of the application would have been wildly different. During the course of this project, I also reviewed the relevant web application regulations and standards. This process was extremely important as it helped me set out distinct protocols that I had to follow.

Objective 2 - Create and optimize a database suitable for the project

I believe this objective has partially been met as a database has been created, although I think it could be further optimized to improve both efficiency and scalability. Specifically, the relationships between tables as well as the constraints could be altered to improve the performance of the database.

Objective 3 – Create a user-friendly UI

By reviewing the general design standards and analysis relevant services, I believe I was able to implement a user-friendly UI. The use of wireframes allowed me to incorporate the design standards before implementing them in HTML/CSS. However, in order to properly assess whether the UI is easy to use, heuristic evaluation should be used.

Objective 4 – Implement back-end functionality that meets the “must” requirements

This success of this objective can be solely based on the result of the test cases. The results of the test cases show that the implementation has successfully met 100% of the “must” requirements, as well as the “should” requirements. Some of the additional, but not necessary, requirements have also been met, such as the ability for admins to manage accounts. Therefore, I believe this objective has been sufficiently met.

Objective 5 – Successfully evaluate the prototype

Test cases have been written that contain both the primary flow and alternate flows. They have also been completed and any fails have been highlighted. In order to assess the prototype in more detail, I performed cyber-attacks on the web application to identify any vulnerabilities and verify that the application is resistant to specific attacks. Overall, I believe this objective has also been met as the prototype was evaluated through test cases and more detailed methods such as performing cyber-attacks to exploit/identify any vulnerabilities.

Objective 6 - Create an implementation that is secure and complies with most modern security standards

By reviewing the OWASP top 10 security threats, as well as additional research into web application vulnerability, I believe I have developed a prototype that meets this objective. Having utilized the wealth of security features Spring Security offers, implementing an HTTPS connection as well as a variety of other security protocols and evaluating the effectiveness of these techniques, it is evident that the prototype is secure.

Deviations from the initial plan

In the initial plan, I highlighted that this application would be developed using Vaadin, a component-based web framework for Java. Developing with Vaadin eliminates the need to use HTML, but at a cost. By using Vaadin, I would have been limited to specific components and would not have been able to make my own elements using HTML. While Vaadin can be more efficient, there are limited

components available for use, and some of them require a pro subscription which is very costly. I therefore decided to not use vaadin and to develop the front end with HTML and CSS.

Another deviation from the initial plan is how I developed the database. In the initial plan, I specified that the database would be implemented after designing a UML class diagram. However, the use of hibernate meant that the database would develop as the web application developed. I simply had to create the entity classes and hibernate would execute the corresponding SQL statements. The use of object-relational mapping and JPA meant that creating the database was straightforward and was directly based off the Java code I had written.

Project limitations

As with most projects, the primary limitation that I experienced was time based. The amount of time allocated for the project had limited the amount of functionality I initially planned to implement. The amount of time not only limited the functionality of the prototype, but also limited the thoroughness of the testing and evaluation phase. Ideally, if I had more time, I would have recruited participants to help perform a heuristic evaluation and generate SUS scores to calculate the overall usability of the system. Additionally, I would have also interviewed participants to gather the requirements for the application. The time constraint also meant that I failed to implement some features I originally planned on implementing, such as the ability to add content to favourites.

Moreover, the small sample size of services I reviewed to determine the requirements is also a limitation. A small sample size means the features present in the services I reviewed might not be representative of the whole market. Although I believe that the requirements would not be vastly different, it is still important to consider that some additional services may have slightly different features.

The time constraint also meant that I could not deploy my application as originally planned. I wanted to deploy my application and employ DDoS mitigation efforts to protect against DoS attacks however the time constraint meant that I could not do so.

Future work

In the future, I would plan on implementing any unfinished functionality that I originally planned on including. Specifically, finishing the adding to favourites functionality. I would want users to be able to add any content they like to a list of favourites that is saved to the database. This list would only be accessible by the user who in turn would be able to access it from any device provided that they are able to log in to the system. Furthermore, I would like to pair user accounts to their content. This would allow a user to see the name of the educator who created the content, click on the name and then see a list of all content made by that educator.

I would also like to find a way to optimize the file reading from the database. This could be done by storing the files on a separate file server rather than in the MySQL database. This would improve the performance and scalability of the system and offer better data redundancy.

As I briefly mentioned earlier, I would like to recruit participants to assist in a heuristic evaluation of the prototype. I would set out specific tasks for the user to complete using the prototype, and then they would be required to fill out a survey based on their experience. The answer to these questions would then be used to determine the SUS (system usability score) for each user. After calculating the SUS for each user, I could calculate a mean average to get the SUS for the entire prototype.

Another objective that I would like to achieve in the future is the incorporation of a rich text editor. Currently, educators are able to upload material, and create very basic material within the site. I would like to implement the ability to create more detailed content within the site. This would be done by adding a rich text editor, allowing educators to choose different font sizes, insert images and format the text.

Finally, I would also like to deploy the web application and integrate DDoS protection. This is inline with the OSWAP top 10 security vulnerabilities and I feel this would be a great step towards improving the accessibility and security of the web application.

Conclusion

To conclude, the purpose of this project was to create an online cybersecurity education portal that would allow sharing of material that is suitable for all age groups in order to reduce the cyber skills gap. To achieve this, I reviewed relevant services and regulations to determine the requirements for the prototype. I feel that I have successfully implemented a solution, albeit with some less features than I originally intended. I believe that the evaluation successfully reinforces my verdict as the majority of the test cases passed. The research into cyber-attacks and security vulnerabilities proved to be exceptionally useful when developing my web application. I was able to implement a number of security features into the application which is again verified by the evaluation. However, without a large number of users and a long-time frame, it is difficult to identify if the prototype created will have any impact on the general cyber-security awareness of the population as well as on the cyber skills gap. The security of a system is often overlooked, but during this project I demonstrated a heavy focus on the security of the application. The wireframes of the UI that were created toward the beginning of the project proved to be useful in ensuring the UI abided by the design principles I reviewed during the literature review. Additionally, the wireframes also assisted in creating the flow of use cases. Having a detailed vision of what the prototype was going to look like very much assisted in the development process as I had a clear vision of what to implement on each page. Following the structure set out in the initial plan was a success, however I had underestimated how long it would take to implement specific features. One feature that took much longer to implement than I thought was the ability to upload content to a specific education level. Fortunately, some other features required less time to implement than allocated so the overall time to complete the project was the same. Overall, I believe my project was a success in that it met the objectives set out earlier in the report. It also sets out the foundations for more functionality to be incorporated, such as adding the ability to favourite content, which was something I had originally planned on including. While the objectives had been met, it is clear that additional functionality as well as quality of life improvements could be implemented.

Reflection

One of the primary skills that I have further developed during this project is my ability to manage time and workloads. By far, this is the single largest project I have undertaken individually in my life and required extensive time management skills. For example, having to balance workloads between this project and another module proved challenging. I had to work at a pace that meant the project would be finished before the deadline but without overworking myself. This was achieved by effective time management while also following the structure I set out in the initial plan. Time management is a priceless skill that will be transferrable to any projects I undertake in the future.

Another core skill that I developed during this project was my ability to communicate. At first, this may seem like a strange skill to develop considering this was an individual project. However,

communication with my supervisor was essential in ensuring that the project delivered on their requirements. The regular supervisor meetings with other students were crucial in ensuring my project remained on track and was headed in the correct direction. My ability to listen to feedback and act accordingly was tested as I wanted to ensure I had addressed the feedback given from my supervisor before our next meeting. A specific example of this was that my supervisor suggested to implement an HTTPS connection and I did so before our next meeting. It was also important to communicate to my supervisor any difficulties I was experiencing during the development of the prototype. Having good communication skills was essential in ensuring that my supervisor was able to understand my difficulty and assist in providing a solution. One example of this is when I was unsure of what data type to store the encrypted password values as in the database. At this time I was using a hashing algorithm from the `digestutils` class. My supervisor suggested to use spring securities own algorithm within the `Bcrypt` password encoder. This proved to be a more efficient and secure solution to what I had previously implemented. I believe this also showcases how my ability to adapt has developed throughout the course of this project. I was able to change my implementation relatively quickly to accommodate the changes recommended by my supervisor.

On top of the wide range of soft skills I developed, I have also improved my technical skills. Using spring boot extensively for several months has drastically increased my competency in Java. While I have used Java for previous projects, I have never used Java to build a web application. Using spring boot introduced me to a new concept, the MVC model. Having spent a lot of time using spring boot, I am now familiar with the MVC model. I believe my increased competency in Java and improved knowledge around web frameworks will benefit me in my future career. I found researching cyber-attack methods and vulnerabilities particularly interesting as this is a subject I wish to pursue further. Before I undertook this project, I had limited knowledge on cyber-attacks, but this project has developed my understanding of some of the most common attack methods such as SQL injection and XSS attacks. Overall, I am impressed with my ability to follow the work structure I set out in the initial plan, albeit with some minor deviations. I originally planned to make the homepage fully functional before implementing the admin page, but at one point I was encountering a bug on the homepage. I therefore decided that I did not want to slow my progress, so I began implementing the admin page. After the implementation of the admin page was complete, I was able to resume working on the homepage with a clear mind, this allowed me to fix the bug in a timely manner.

To conclude my reflection on learning, this project has allowed me to develop a wide range of core skills such as time management, communication, adaptation, and introduced me to new technical skills such as executing and protecting against cyber-attacks. This project also allowed me to build upon my Java, HTML and CSS skills.

References

- [1] Watters, A., 2022. 25 Crucial Information Technology Statistics & Facts to Know. [online] Default. Available at: <<https://connect.comptia.org/blog/information-technology-stats-facts#:~:text=Projections%20show%20the%20technology%20industry,CAGR%20of%2017.5%25%20by%202025.>> [Accessed 7 February 2022].
- [2] Half of UK 10-year-olds own a smartphone, BBC News, 2022. [Online]. Available: <https://www.bbc.co.uk/news/technology-51358192>. [Accessed: 07- Feb- 2022].
- [3] 2021 Cyber Security Statistics Trends & Data, PurpleSec, 2022. [Online]. Available: <https://purplesec.us/resources/cyber-security-statistics/>. [Accessed: 07- Feb- 2022].

- [4] Cyber security skills in the UK labour market 2020, GOV.UK, 2022. [Online]. Available: <https://www.gov.uk/government/publications/cyber-security-skills-in-the-uk-labour-market-2020/cyber-security-skills-in-the-uk-labour-market-2020>. [Accessed: 07- Feb- 2022].
- [5] The cyber security skills shortage is getting worse | OneFile, 2022. [Online]. Available: [https://onefile.co.uk/explore/the-cyber-security-skills-gap-is-getting-worse/#:~:text=The%20UK%20is%20suffering%20from,hitting%20cyber%20security%20the%20hard%20est.&text=And%20by%202021%2C%20it's%20predicted,affecting%20companies%20large%20and%20small](https://onefile.co.uk/explore/the-cyber-security-skills-gap-is-getting-worse/#:~:text=The%20UK%20is%20suffering%20from,hitting%20cyber%20security%20the%20hard%20est.&text=And%20by%202021%2C%20it's%20predicted,affecting%20companies%20large%20and%20small.). [Accessed: 07- Feb- 2022].
- [6] The 7 Most Common Types of Cybersecurity Attacks in 2021, Auth0 - Blog, 2022. [Online]. Available: <https://auth0.com/blog/the-7-most-common-types-of-cybersecurity-attacks-in-2021/>. [Accessed: 08- Feb- 2022].
- [8] DDoS Protection – understanding the complexity of multi-vector threats, Xantaro, 2022. [Online]. Available: <https://www.xantaro.net/en/tech-blogs/complexity-of-multi-vector-ddos-threats/>. [Accessed: 11- Feb- 2022].
- [9] Protecting Against SQL Injection, Hacksplaining, 2022. [Online]. Available: <https://www.hacksplaining.com/prevention/sql-injection>. [Accessed: 11 Feb 2022].
- [10] Web Application Development Standards - Web Site Information (CA Dept of Education), Cde.ca.gov, 2022. [Online]. Available: <https://www.cde.ca.gov/re/di/ws/appdevstandards.asp>. [Accessed: 13 Feb 2022].
- [11] Web Application Security Standards | Veracode, 2022. [Online]. Available: <https://www.veracode.com/security/web-application-security-standards>. [Accessed: 14 Feb 2022].
- [12] 2022. Best backend programming languages. [online] Available at: <https://blog.back4app.com/backend-programming-languages-list/> [Accessed 15 April 2022].
- [13] Mkoshops.com. 2022. Spring Boot Architecture - Tutorial And. [online] Available at: https://www.mkoshops.com/?product_id=131575206_72 [Accessed 16 April 2022].
- [14] Team, S., Team, S., Freeman, C., Cipot, B. and Rana, A., 2022. Top 4 software development methodologies | Synopsys. [online] Software Integrity Blog. Available at: <https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/> [Accessed 18 April 2022].
- [15] Umsl.edu. 2022. The Traditional Waterfall Approach. [online] Available at: <https://www.umsl.edu/~hugheyd/is6840/waterfall.html> [Accessed 18 April 2022].
- [16] Mezquita, T., 2022. Agile Development Methodology - CyberHoot. [online] CyberHoot. Available at: <https://cyberhoot.com/cybrary/agile-method/> [Accessed 18 April 2022].
- [17] Heurio.co. 2022. Nielsen's 10 Usability Heuristics - Heurio. [online] Available at: <https://www.heurio.co/niensens-10-usability-heuristics> [Accessed 18 April 2022].
- [18] Fadatare, R., 2018. Spring Boot User Registration and Login Module Tutorial. Available at: <https://www.javaguides.net/2018/10/user-registration-module-using-springboot-springmvc-springsecurity-hibernate5-thymeleaf-mysql.html> [Accessed 1 March 2022].
- [19] Almost Perfect Email Regex. 2022. Email Address Regular Expression That 99.99% Works. [online] Available at: <http://emailregex.com/> [Accessed 15 April 2022].

[20] Codebun.com. 2022. Spring Boot Upload and Download File Example using Thymeleaf. [online] Available at: <<https://codebun.com/spring-boot-upload-and-download-file-example-using-thymeleaf/>> [Accessed 8 April 2022].

[21] BBC Bitesize. 1998. Home - BBC Bitesize. [online] Available at: <<https://www.bbc.co.uk/bitesize>> [Accessed 22 February 2022].

[22] Jr.brainpop.com. 1999. [online] Available at: <<https://jr.brainpop.com/>> [Accessed 12 May 2022].

[23] Synopsys.com. n.d. What Is the OWASP Top 10 2021 and How Does It Work? | Synopsys. [online] Available at: <<https://www.synopsys.com/glossary/what-is-owasp-top-10.html>> [Accessed 14 February 2022].