An E-Learning Approach to Cyber Security Awareness and Defence: Mitigating Threats in the Digital Landscape

by
Harry Campbell
BSc (Hons) Cyber Security and Forensic Computing

Project code: *PJE40*
Supervisor: Oluwatobi Fajana
May 2024

✓	I give permission for my project to be published in the University library and/or be made available to other students as examples of previous work.
✓	I confirm that I have read and understood the University Rules in respect of plagiarism and student misconduct.
✓	I declare that this work is entirely my own. Each quotation or contribution cited from other work is fully referenced.

Date: 01/05/2024

Acknowledgements

I would like to thank my friends and family for their constant support and encouragement, throughout my time in university.

I would also like to thank project supervisor Tobi for continued support and guidance throughout my final year.

Abstract

on a new page and centred
about 100 words
should not appear in contents list.
can be coloured if you wish.

Table Of Contents

Table Of Figures

Table of Tables

Chapter 1
Introduction

1.1 Project Background/Overview

In an era characterised by rapidly evolving digital threats, the need for effective cyber security education has never been more apparent. The increasing sophistication and frequency of cyber-attacks such as phishing, ransomware, malware infection, social engineering and identity theft highlight the gap in the general public's knowledge and ability to combat these threats. E-learning approaches across all subjects have proven their effectiveness in delivering information, leveraging the flexibility and accessibility of digital platforms to deliver a lower-cost learning experience with consistently better grades (Odhaib, 2018).

Across the UK, cybercrime has perceptibly grown, and threats are escalating at an alarming rate. A report by IT Governance UK (2023) found that in the year 2023, over 8 million records were breached across 2,814 separate incidents. It is not just corporations which are experiencing this however: following the COVID-19 pandemic, UK cybercrime grew rapidly, taking advantage of those in vulnerable positions throughout lockdowns, through pet scams, banking scams and vaccination scams - with a large proportion of victims being the elderly (Sikra et al., 2023).

While cost is a significant barrier to accessing cybersecurity education, there are additional challenges with existing platforms beyond financial constraints. For instance, many platforms, like 'cyberawareness.co.uk,' primarily target organizations, making their resources inaccessible to the general public who also require essential cybersecurity information. Furthermore, even platforms like 'TryHackMe.co.uk' or 'Udemy.com,' which offer more accessible options, often provide content that is either too advanced for beginners or comes at a substantial cost, with basic courses priced upwards of £60. Additionally, the lack of tailored content for various skill levels and learning styles can further hinder effective learning experiences. Therefore, there is a pressing need for a platform that not only addresses cost barriers but also offers accessible, comprehensive, and tailored cybersecurity education suitable for diverse audiences.

Addressing these limitations, this project introduces a fully featured online learning platform designed to offer up-to-date, interactive, and user-friendly education in cyber security, to bridge the gap in the public's general ability to

identify and protect themselves from cyber threats. By providing accessible education on critical aspects of cyber security, it is hoped that this will result in a significant reduction in the vulnerability of individuals to cyber-attacks. The platform will be particularly beneficial for users who are not technically aware and for small businesses who are unable to justify the premium costs of in-house training.

1.2 Project Scope

The scope of this project encompasses the development of an interactive e-learning platform focussed on online safety education. With a broad target audience, reaching out to those who are unaware of the threats posed against them, this could include students, professionals, and most everyday internet users - the platform will cover a variety of essential topics from strong password creation to recognising phishing attacks, aiming to educate users on how these attacks are performed in order to best prepare them to defend themselves online.

1.3 Aims and Objectives

The primary aim of this project is to create an effective and extensible e-learning platform, which can easily be continued long after the completion of this project. The platform will effectively educate users about online safety through a variety of instructional videos, real-life scenarios, and quizzes.

The objectives throughout the project are as below:

- To develop a platform for the dissemination of cyber-security knowledge
- To curate content which will comprehensively cover the key aspects of current cyber threats
- To incorporate real-world scenarios and quizzes which will reinforce learning and assess users' understanding of the content.
- To evaluate the platform's effectiveness at improving users’ awareness and online safety practices.

1.4 Project Constraints

Due to its nature as a university final year project, this project has three notable constraints, outlined below, which necessitate careful consideration and adjustments to the project's development:

Completion Deadline: The strict deadline imposes pressure on project management and necessitates a focus on efficient time management and task prioritization. To address this, agile project management methodologies will be employed to ensure timely completion of project milestones and deliverables. Regular progress assessments and adjustments to the project plan will help mitigate the risk of schedule overruns.

Budget Constraints: Limited financial resources constrain the project's ability to invest in advanced technologies and high-quality content creation. In response, cost-effective solutions and open-source technologies will be prioritized to optimize resource utilization. Additionally, leveraging available university resources, such as software licenses and collaboration opportunities, will help mitigate the impact of budget constraints on project outcomes.

Scalability Constraints: Initially focusing on creating a functional platform, scalability considerations for accommodating future growth require additional resources and planning. To address this, the project will adopt a modular architecture and design principles that allow for scalability without significant rework. Scalability features will be incorporated iteratively, ensuring that the platform's infrastructure can adapt and expand as user demand grows.

Acknowledging the project constraints outlined above will inevitably lead to the identification of certain requirements that may not be feasible within the given limitations. To address this, the project will utilize the MoSCoW prioritization technique to categorize requirements based on their importance and feasibility. This approach allows for the identification of essential features (Must-haves) that align with project goals and constraints, while also recognizing that some requirements may need to be deferred (Could-haves) or excluded entirely (Won't-haves) due to time, budget, or scalability constraints.

By recognizing the limitations imposed by the completion deadline, budget, and scalability concerns, it is possible to prioritize efforts, allocate resources efficiently, and deliver a viable solution that meets both immediate needs and future scalability requirements.

1.5 Legal and Ethical Issues

Addressing legal and ethical issues ensures this project's compliance with relevant laws and regulations, protects intellectual property, and upholds user trust and safety. It also guarantees that the platform is accessible and inclusive, providing accurate and unbiased educational content.

Legal Issues:

- Intellectual Property Rights: Ensuring that content, software, and materials used or created for the platform do not infringe upon copyrights, trademarks, or patents is crucial. Proper attribution and permissions are necessary.

- Data Protection and Privacy Laws: Compliance with data protection and privacy laws, such as GDPR is paramount. Any personal or sensitive data collected, processed, or stored by the platform must be handled in accordance with regulations.

Ethical Issues:

- User Consent: Ethical concerns arise when collecting user data for research or analysis. Ensuring that users are fully informed and have consented to data collection and usage is essential.

This project maintains legal and ethical standards, including respecting intellectual property rights and complying with data protection laws like GDPR, by adhering strictly to the university policies. Following the ethical procedures outlined in the university's guidelines, as detailed in the appendix.

1.6 Project Approach

To combat the project’s constraints such as its deadline and budget, whilst still accomplishing the aims and objectives, the project will be split into sections for development utilising tools such as Kanban tables to manage workload. Firstly, the platform will be developed and tested as a generic content-delivery platform, and then once fully operational, content will be created to fulfil the objectives. Approaching the project this way will avoid confusion and bottlenecks in production, tackling small chunks at a time to eventually tie together a fully functional platform.

1.7 Project Management

For this project, the Agile methodology is the most suitable approach along with the implementation of a Software Development Life Cycle. Several key factors from the project align well with the Agile methodology. Firstly, cyber security itself is a rapidly growing field with new threats emerging daily meaning that the content may need to update and change swiftly. Secondly, the project will require close contact with end-users: Agile promotes continuous communication and collaboration, fostering a deep understanding of user requirements and enabling the incorporation of feedback promptly. Lastly, the modular nature of Agile development will enable the project to be broken down into manageable components, prioritising features and content that deliver the most value early in the development process.

1.8 Summary

In summary, this project emerges as a timely and critical response to the escalating cyber threats prevalent in today's digital landscape. By spearheading the development of an innovative online learning platform, it endeavours to meet the surging demand for comprehensive cybersecurity education. Whilst overcoming various constraints, including time, budget, and scalability, the project remains committed to the creation of an effective online safety platform tailored to the needs of users. In doing so, it not only aims to safeguard individuals in the digital age but also contributes significantly to global endeavours aimed at enhancing cybersecurity awareness, equipping users with the knowledge and skills necessary to navigate cyberspace securely and confidently.

Chapter 2
Literature Review

2.1 Introduction

This literature review will examine the extent of cyber security threats in the modern digital landscape, illustrating the growing rate of attacks and a population-wide deficit of appropriate cyber security awareness. It will also review best practices for effective e-learning and approaches to teaching. In addition, existing approaches will be reviewed and analysed to identify the deficits in the current market.

2.2 Search Methodology

In conducting this literature review, a variety of carefully selected search strings were employed to explore academic papers and articles across multiple databases, primarily focusing on Google Scholar.

Google Scholar gives access to almost every research paper and article ever published online, simply filtering for relevant texts using search strings is not enough. By restricting the results to content published after 2002, it was ensured that the information retrieved reflects current insights and developments, thus maintaining its relevance to contemporary contexts. Additionally, sorting the results by relevance further enhanced the quality of results, facilitating the identification of the most pertinent and informative resources. This approach ensures that the information presented is both up-to-date and closely aligned with the subject matter, optimizing the utility of the search results.

Articles found from these strings, and the filter criteria were. further analysed to extract key points and ensure suitability and quality – for example pertinence to the UK and EU more than other countries; relevance to the current threat landscape and ensuring the accuracy of the content.

Listed in the table below are some of the key search strings used in this literature review and the number of results found from them, the search strings were designed to produce many relevant papers to answer the question of how to build an effective e-learning solution for cyber security issues but also to provide articles specific to each section of the literature review such as the overview of threats; public perceptions; teaching methodologies and best design practices.

Search String	**Number of Results**
“Cyber Attacks”	104,000 results
“Modern Cyber Threats”	27,500 results
“Recent Cyber Attacks”	48,300 results
“Cyber Threat Landscape”	3,770 results
“Public Awareness Cyber Threats”	40,800 results
"Public understanding of cyber risks"	54,800 results
"Cybersecurity literacy survey"	17,300 results
“Phishing awareness survey”	19,700 results
“Modern day malware”	19,500 results
“Malware infection Rates”	18,100 results
"Cybersecurity education lag"	17,100 results
“Common Cyber Scams”	18,000 results
"Gap in cyber threat knowledge"	35,600 results
“Cyber Security Education Trends”	76,600 results
“Cyber Security Training Tools”	103,000 results
“History of Cyber Security”	105,000 results
“Global Cyber Attack Patterns”	28,000 results
“UK Cybersecurity Statistics 2023”	16,900 results

“Digital Tools in Cybersecurity Training”	20,300 results
“Understanding Cyber Threats”	41,700 results
“Why Certain Teaching Methods are Effective”	17,800 results
“Effective teaching methods”	6,720,000 results
“The psychology behind adult learning”	17,800 results
“Pedagogical Approach vs Andragogical”	17,600 results
“Cyber security insights report”	49,300 results
“Multimedia learning effectiveness”	17,800 results
“Role of video and text in education”	405,000 results
“Interactive elements in learning processes”	27,800 results
“Effectiveness of E-learning”	488,000 results
“E-learning design”	419,000 results
“UK Phishing Scams”	14,600 results
“E-learning effectiveness”	102,000 results
“Digital Literacy and Cyber Security”	21,800 results
“Cost of cyber security training”	122,000 results
“Cyber breaches report”	19,100 results
“Demographics of Cybersecurity Users”	17,500 results
“COVID-19 Phishing scams”	6,180 results
“Effective web design for e-learning”	64,300 results
	7,832,620 total articles

Table 1 - Literature Search Strings

2.3 An Overview of Cyber Threats in the Modern Digital Landscape

In the modern world, society is increasingly technologically dependent and whilst the digital era has brought about countless benefits, it has also opened the door to a whole world of new threats, not just for larger corporations but also for everyday users. Cyber-crime is becoming one of the largest challenges for law enforcement agencies, and with relevant laws and legislations taking their time to pass into legislation, there are very few laws to protect users (Zhang et al., 2011).

As suggested by Zhang et al. (2011), there are two different approaches attackers use to commit cyber-crimes: they can use cyber-crime as a tool for activities such as online piracy and copyright violations, or they can use it as a weapon to target a computer or network. The concept of attack classification is further examined by Uma & Padmavathi (2013), where it is stated that attacks can be distinguished by their purpose. Firstly, there are reconnaissance attacks, typically focussed on gaining information from a person to use later; these include packet sniffers, DNS queries and port scans. Secondly, there are access attacks which are the most common type of attack and aim to steal valuable information; these can be dictionary attacks, man-in-the-middle attacks or most commonly, social engineering and phishing attacks. The third and final type of attack is a denial-of-service attack and is most commonly targeted towards businesses with the intention of deliberately taking down systems.

Social engineering attacks, carried out through phishing, are one of the most prevalent threats to general internet users. Attackers use psychological manipulation to trick users into revealing sensitive information, and although some are easily identifiable, unfortunately a significant number of these attacks succeed. The most common type of phishing seen is via email where attackers send out emails pretending to be companies who need users to login, often with believable looking content and variations of legitimate email addresses such as ‘@appl3.com’ or ‘@g00gle.com’ (Andaluri et al., 2023). There are more methods of phishing than just email attempts, however. Spear phishing is a more targeted phishing attack often aimed at certain businesses; whaling attacks are a variation of phishing attack which is directly aimed at high level executives within a company who have valuable data to expose; SMS phishing is growing more popular recently thanks to smart-phone innovations, allowing scammers to send SMS texts which mimic 2 factor authentication requests, or false messages which request users to sign in through the malicious link supplied (Andaluri et al., 2023).

Attackers often utilise current events when constructing their attacks, typically focussing on natural disasters, major political events, and economic concerns (Andaluri et al., 2023). The COVID-19 pandemic in the year 2020 brought a plethora of new attacks as explored by Lallie et al (2021). The increased anxiety caused by the pandemic drastically increased the likelihood of a cyber-attack succeeding, and the pandemic also fuelled an increase in cyber-attacks overall. A report by Office for National Statistics (2022) found that in the year ending March 2022, 61% of fraud incidents were cyber related and suggests a direct relation to the COVID-19 pandemic and the “behavioural changes” caused by it. Lallie et al. (2021) highlights the technological dependence brought about by the pandemic and lockdown efforts and its direct correlation with an increase in cyber-attacks.

A growing trend in the digital landscape is the use of generative AI models like ChatGPT, DallE or Google Bard - these AI tools however have some more sinister relatives. Falade (2023) explores the use of AI tools to revolutionise cyber-attacks, adding to existing threats and ushering in new risks - from deepfake technology which can impersonate authoritative figures, to AI driven social engineering exploits. AI tools can now be used to automate a variety of cyber-attacks by generating malicious code and creating undetectable malware, and even creating believable phishing emails for attackers with almost no existing technical ability or knowledge. This paradigm shift allows anyone to immediately start committing cyber-crimes, aligning with the suggestions made by Lallie et al. (2021), that more and more people have started turning to cyber-crime to support themselves.

In conclusion, the widespread impact of cyber threats in today’s digital era is undeniable, affecting both individuals and businesses. The COVID-19 pandemic has only intensified these threats, creating an environment where dependency on technology and heightened societal anxiety have given rise to a surge in cyber-attacks. These attacks are not only becoming more sophisticated but also increasingly easy to execute.

2.4 Unravelling the Disconnect: Public Perception vs. Cyber Reality

Cyber-attacks are becoming increasingly more common, and as security mechanisms are put in place, researchers suggest that the most fatal flaw in any security chain will always be humans (Ricci et al., 2018). Whilst reports from security companies like Kaspersky show a trend in over 55's susceptibility to cyber threats (Kaspersky Lab 2016), Symantec (2016) actually suggests that millennials, with their overconfidence and naivete can often be more vulnerable online, accounting for the majority of cyber-crime victims in 2015. This suggests that the cyber security awareness gap is endemic across all age groups and all online users.

According to a safety insights report (Norton, 2022), a vast majority of people want to protect their privacy online, but in the UK, 49% of survey participants stated they did not know how to go about increasing their online security. 67% of UK participants also stated that they are more alarmed now than ever before about the threats they face. In the year ending March 2020, UK instances of computer misuse increased by 89% and 61% of all fraud was cyber-related fraud, almost half of all cyber-related fraud incidents being phishing attacks. With close to half of the adults having experienced a form of computer or mobile virus in the previous 12 months (Norton, 2022) and cyber-related fraud increasing year on year, it is clear that the threat landscape is ever-growing, and security awareness and education is falling behind.

As explored in the previous chapter through the findings of Lallie et al. (2021), the COVID-19 pandemic has been a catalyst for cyber security threats. Ringström (2023) explores this concept further to evaluate the security deficits brought by the Work From Home (WFH) model. The shift from the established corporate security infrastructure to home offices can have a drastic effect on employee behaviour; this is further explored by Esfahani & Mohit, (2023) who introduced a new behaviour terminology - "Cyberloafing" - which refers to employees using company internet and devices for non-work-related activities. The overall findings suggest that the shift to WFH has made it increasingly difficult for companies to secure assets, with valuable data now stored on employees' devices, effectively turning employees with unprepared and insecure home IT infrastructures into easy targets for cyber-attacks (Ringström 2023).

With social engineering and phishing attacks becoming so common and increasingly successful, it is important to consider why users so often fall for them. An experiment by Loyola University tested 584 participants with a

phishing email. Out of this group, 28.3% clicked on the email, 13.6% of whom clicked on the link contained within it. This experiment also considered the participants psychological characteristics, finding that of those who opened the email, 61% scored low on anxiety scales (Moran & Hart, 2023). A similar concept is reviewed in a much greater depth in a study by Alhaddad et al., (2023) in which it is proposed that certain personality traits are far more susceptible to social engineering attacks, and therefore should be focussed on for security awareness training. Those with internet anxiety, who feel a compulsion to reply to or check spear phishing emails which others ignore, proved to be most susceptible.

There are other factors however which can increase the effectiveness of phishing attacks, one being their ever-growing complexity. In recent years a growing market for AI tools and generative models has led to their significant use in social engineering attacks, whether they are used to create believable emails, deep-fake video of audio, or even generating malicious scripts. With research suggesting a 135% increase in attacks which leverage generative AI, its role within social engineering is understood to have significantly increased both the complexity of attacks and the likelihood of users falling victim (Falade 2023).

The increases in both cyber-attack frequency and complexity are compounded by the lack of security awareness and education people receive. Outside corporate training, there are few ways for users to educate themselves on current threats, but as adults consistently admit to risky online behaviours such as using the same password multiple times, using unsecured networks at home, and even willingly sharing passwords with others, there have never been more clear signs of the growing need for appropriate cyber education (Ricci et al., 2018).

2.5 Effective Teaching Methodologies

For many in the modern world, upon reaching school leaving age, all forms of education cease and the only way to further knowledge is to undertake self-study. Muresan (2014) argues that continuous education is a main pillar of knowledge society, however the implementation of continued adult education has its challenges. An American scholar, Malcolm Knowles, was one of the first to suggest that there is a significant difference between the education of children and the education of adults, introducing a term called “andragogy” stemming from its Greek origin of “man-leading” or adult education, Knowles suggests that adult learners are “a neglected species” (Knowles, 1984). Since the term was introduced, many have continued this concept of pedagogy vs andragogy.

Unlike teaching children, where a teacher transmits predetermined syllabuses of knowledge in the most efficient method possible, the goal of adult education, and the aim of the andragogical approach, is to supply resources and procedures for learners to acquire information and skills themselves. Adult education should be done through self-actualisation, involving the whole emotional, psychological, and intellectual being (Holmes & Abington-Cooper, 2018). A few key principles of andragogy suggest that adults need to learn through trial and error, with what they learn being immediately relevant to their lives and they learn far better through problem-based learning as opposed to content-based (Fornaciari & Lund Dean, 2013).

Considering the challenges of the andragogical approach, advancements in technology and the advent of e-learning platforms have introduced a learning model with unlimited access, ease of use and self-paced learning which adults with busy schedules want to take advantage of (Cercone 2008). It is also argued by Cercone, (2008) that whilst the andragogical approach suggests adults are self-directed learners, some may need more structure to assist them in their learning which is where e-learning platforms fit in - they can provide a framework of self-paced online tasks and contextualise them to support one of the main principles as suggested by Fornaciari & Lund Dean, (2013), to make the content relevant to their current lives.

Utilising e-learning platforms also provides access to a variety of new content to aid educators, one of the most relevant being multimedia technologies. Modern technologies strip away the traditional blackboard and chalk, introducing instead virtual classroom environments; student-student communications; online assessments and most importantly the ability to complete learning at any time of the day (Lau et al., 2013). Multimedia learning refers to the combination of various digital media types including text, images, sound, and video. Put

together, all of these features create a well-rounded learning experience which is significantly better than simple text-based learning (Shank, 2005). A study carried out by Zhang et al., (2006) sought to examine the effectiveness of e-learning methods. Whilst they did find that overall interactive video along with individual control to content could lead to increased academic performance, simply implementing video into e-learning environments may not be sufficient and this supports the idea that multimedia learning must be a combination of digital media types to guarantee its effectiveness (Shank, 2005).

Whilst it is clear that e-learning and digital media can improve teaching, especially aligning with the self-paced nature of the andragogical approach, many suggest it could have some serious drawbacks. There are concerns that with e-learning creating an easily distributable platform for learning, companies and educators will turn to an “earn-to-learn” approach, replacing the concept of learners as students with the concept of students as clients (Cernat, 2013) (Fry, 2001). Incidents of this have already been seen in the Spiru Haret University of Romania, where online distance learning saw an unprecedented boom in the number of students enrolling, eventually leading to a point where for every 1 teacher there were 206 students (Cernat, 2013). Similar concerns are raised with the effectiveness of e-learning entirely in some contexts, for example nursing students who need to be shaped into critically thinking, fast responding care givers, cannot gain the necessary skills from ticking boxes on online courses (Muirhead, 2007). Finally, the largest issue with online-learning, intensified in recent years by the COVID-19 pandemic, is itself a pandemic of online cheaters in higher education (Jenkins et al., 2022). Moving all aspects of university online from the year 2020 saw what Jenkins et al., (2022) described as a major increase in cheating, with three quarters of the interviewed sample admitting to having cheated on assessments during COVID-19 online assessments.

In summary therefore, it is clear that whilst developments in online learning platforms utilising multimedia content provide a great andragogical approach for adult learners, e-learning can have its drawbacks especially when used for higher education and therefore any approach should be designed appropriately to counteract the issues approached within this section.

2.6 Best Practices in UI Design for Innovative Learning Approaches

Modern day learning has made significant strides in both the breadth of knowledge and the effective dissemination, with chalk and blackboards left in the past, virtual classroom environments, which are able to support more advanced student-teacher and teacher-teacher interactions, are being introduced (Lau et al., 2013).

The shift from “standard” in person learning to online teaching has also enabled more effective methods and a variety of content types to be used. Driven by emerging web technologies such as HTML, CSS3 and WebGL, multimedia technologies allow for a far more complex dissemination of knowledge (Lau et al., 2013). Utilising technologies such as social media in learning has been seen to significantly increase motivations to learn, while game-based learning (GBL) has also been seen to improve learning attentiveness and effectiveness.

Simply uploading existing traditional content such as worksheets and textbooks online is not conducive to a more effective learning environment. Online learning content must be effectively designed to facilitate the desired learning outcomes (Brown & Voltz, 2005). There are several key components to the effective implementation of e-learning. Firstly, there is the activity: students should be provided with a rich learning activity which allows them to actively choose what to learn instead of forcing them down a specified pathway. Secondly there is the scenario that in order for the learning to be memorable and valuable, learners need to contextualise it and be motivated to learn. Thirdly is feedback: without reflection and feedback, knowledge growth is stunted therefore the use of effective feedback enables users to learn from their mistakes. Finally, delivery method is one of the more crucial aspects of e-learning effectiveness: e-learning content should maximise engagement with learners and allow for the implementation of rich media content (Brown & Voltz, 2005).

As suggested by the anagogical approach, adult learners generally require study to be self-paced and be able to continue when they have free time. It also needs to be relevant to their social role to improve readiness to learn (Cercone, 2008). Some examples of effective design which can propagate the anagogical adult learning approach are highlighted by Cercone, (2008) and include: large, easily readable fonts; a variety of graphics, images and tables, clear menu structures, search and find functionality; practice and feedback options; frequent exit and entry points; help functions and use of graphic organisers.

In summary, as learning environments evolve to be online, the content itself is also required to evolve to support more rich, interactive, and effective learning especially in the context of adult learning.

2.7 Existing Solutions

Cyber security threats are already a well-established issue and as such, multiple organisations already have proposed solutions for online security e-learning. Analysing these existing applications will aid in the identification of key requirements and features whilst also identifying any failings these platforms may have.

2.7.1 Certified Secure Safe Internet Course

Certified Secure (Certified Secure, 2010) is an online platform which provides a free, safe internet training module and certification, alongside several other cyber security-related training courses both paid for and free. Their platform aims to encourage and fulfil growing interests in IT security skills.

The web platform itself is outdated, however it is still easy to navigate and find the content you are looking for. The safe internet module consists of two online examinations, a simple quiz and a “Don’t Click” challenge to aid in identifying malicious emails. At the footer of the webpage, there are links to several recommended training videos and some supporting PDFs for learners to utilise.

Core Features:

- User login system: this provides a means for tracking course progression and allows users to come back later and pick up where they left off. Having a user account also affords users the ability to purchase a premium option.

- Video and PDF content: an array of video and text-based content is provided to users and can help them to understand the key concepts of IT security. The videos are all around 2 minutes long and are broken down into smaller sections which allows users to be self-paced.

- Quiz/Assessments: providing an assessment on the platform allows users to solidify and test their knowledge, however a lack of feedback on quiz results significantly lowers users’ ability to reflect on and learn from their mistakes.

Overall, as a free to access platform, it provides a good base for users who want to take an interest in IT security, however its layout, with videos linked at the very bottom of the page, makes it more of a repository of information than a guided educational module and a lack of reinforcing feedback will make it difficult for users to advance their learning.

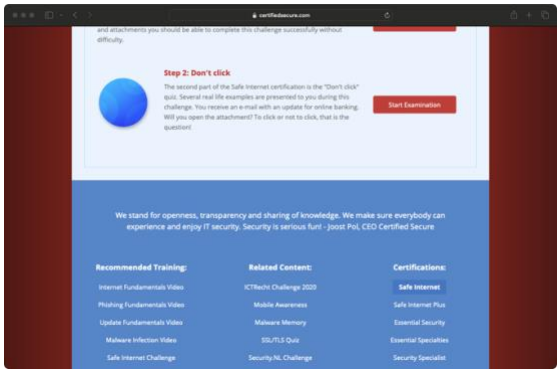

Figure 1a - Certified Secure Videos

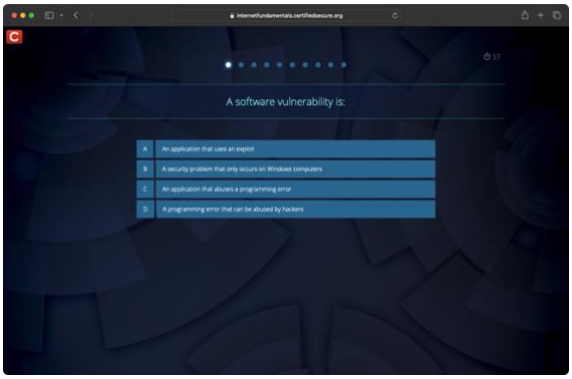

Figure 1b - Certified Secure Quiz

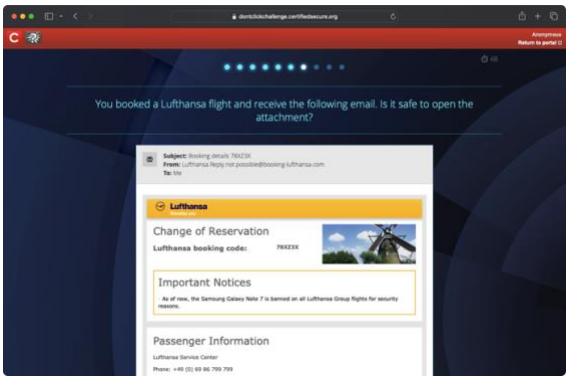

Figure 1c - Certified Secure Real-World Scenarios

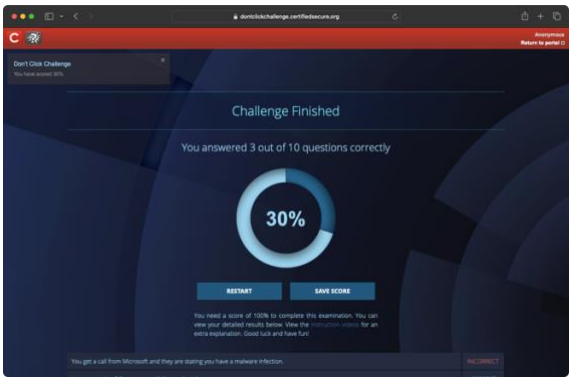

Figure 1d - Certified Secure Feedback

2.7.2 GCF Global Internet Safety Course

GCF Global (GCF Global, 2014) is a large repository of self-paced courses on topics ranging from using Microsoft Office products to core skills such as maths and English and perhaps most importantly, basic internet usage. Their Internet safety course is free to access and contains a linear course approach to teach users how to understand and avoid the risks they face on the Internet.

The Internet safety course covers a range of topics, starting with simple concepts such as password security and browser security, and eventually covering more complex topics such as malware protection and social engineering. Throughout the course, sections are comprised of short chunks of text-based information with supporting videos embedded within the webpage.

Core Features:

- User accounts: having a user account to track your progress allows users to step away from the content and come back at a time which suits their needs; this is especially important considering adult learners' busy lifestyles.

- Structured and staged content: by splitting the content across 15 separate sections, the learning has been broken down into more manageable chunks which allow learners to continue at their own pace and pick back up from where they left off. Learners are not however forced into unlocking sections as they go and can swap from section to section as they please.

- Infographics: The use of small graphics to solidify some of the concepts within each section is effective at breaking up the content to make it easier to digest and emphasises the important concepts to the learner.

- End of module quiz: the use of a quiz at the end of the module allows learners to validate their understanding of the content and test their knowledge. Without giving users feedback, however, the quiz fails to enforce the concept of learning from your mistakes and doesn’t allow improvement.

Overall, this internet safety course is a very effective tool to be used to learn the basics of internet safety and security, despite the drawbacks of having no feedback on mistakes made in the quiz. It improves on the drawbacks of Certified Secures (Certified Secure, 2010) platform; however, it does not provide a complete solution for meaningful cyber security awareness and education.

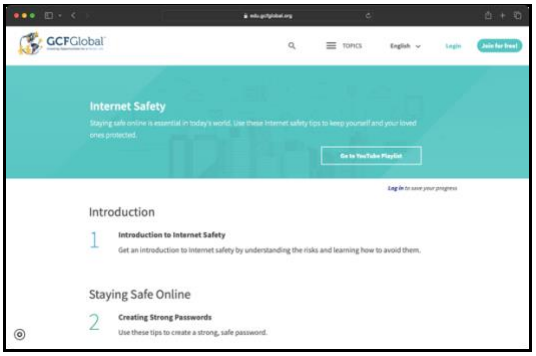

Figure 2a – GCF Global Safety Course.

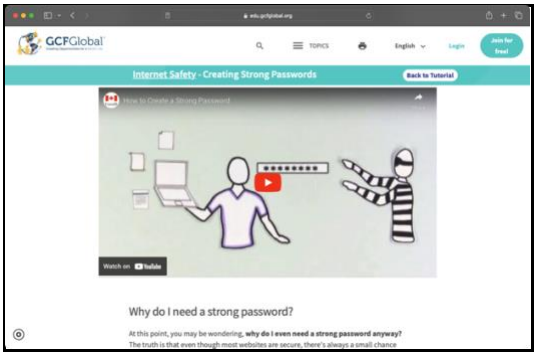

Figure 2b – GCF Global Videos.

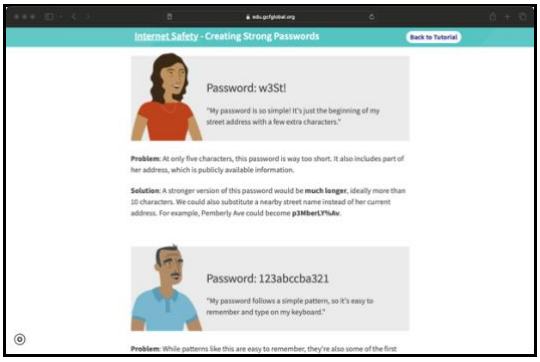

Figure 2c – GCF Global Text Content.

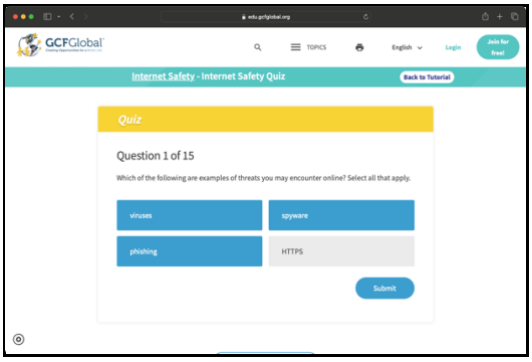

Figure 2d – GCF Global Quiz.

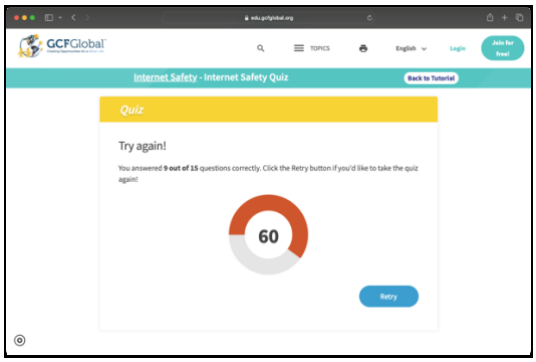

Figure 2e – GCF Global Quiz Feedback.

2.7.3 Phishing Box Phishing Test

The phishing box phishing IQ test (Phishing Box, 2022) is an online tool used to test your susceptibility to an attempted phishing attack, provided by Phishing Box - a company which designs corporate cyber security training courses for education of employees. Although the actual courses are very costly and designed for employees and not consumers, their phishing IQ test is a valuable resource in testing the ability to detect phishing emails by their patterns.

The test is accompanied by a 4-minute video covering the basics of phishing attacks and their recognisable characteristics to prepare users for the test itself.

Core features:

- Video Content: use of training videos can be a very easy method to provide a large amount of information in a short amount of time. To be effective however, they need to be engaging and the videos provided by phishing box are simply voice over a slideshow which lessens their effectiveness.

- Phishing Test: a quiz which uses realistic simulations of what a phishing email would look like provides learners with the necessary context and skills to identify them afterwards. With little feedback on the users' results however, the actual chance of solidified learning is reduced.

Overall, whilst not a complete solution to cyber security awareness, this short video and online test is a great starting point for what would be classed as effective online security training when paired with concepts seen in other applications.

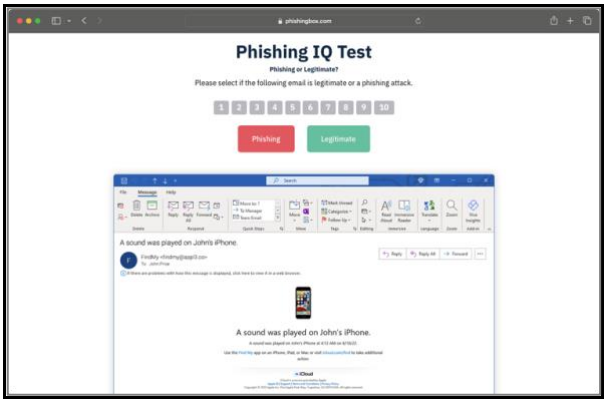

Figure 3a – Phishing IQ Test.

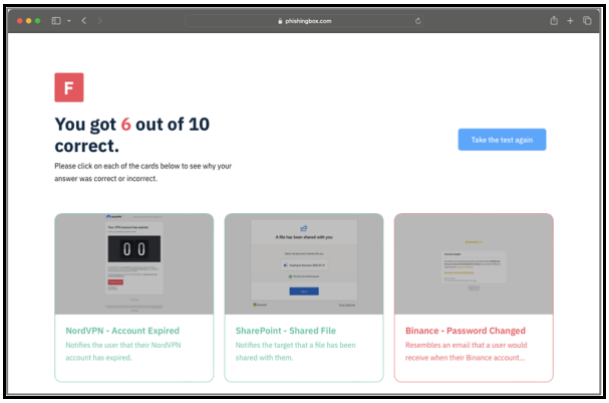

Figure 3b – Phishing Simulator

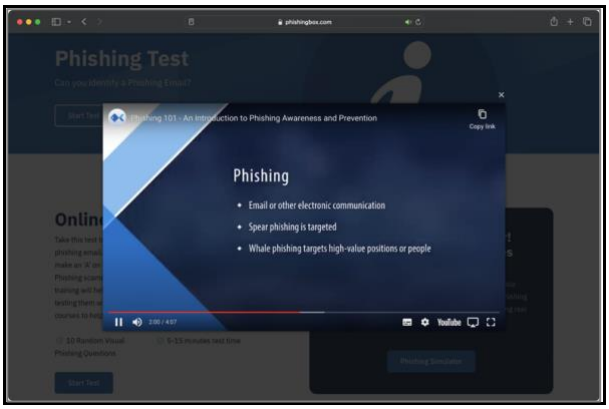

Figure 3c – Phishing Video Content

2.7.4 Comparing Existing Solutions

Feature	Certified Secure	CFC Global Internet Safety	Phishing Box attack Simulation
Login System & Progress Tracking	Yes	Yes	No
Free Online Safety Module	Yes	Yes	Limited (Only free quiz)
Online Quiz Activities	Yes	Yes, end of module quiz	Yes
Feedback on quizzes	No feedback provided	No feedback provided	Limited feedback
Video Content	Provided linked below but not contextualised	Informative videos provided, backing up the text content.	No
Text Content	Provided linked below but not contextualised	Modules are mostly made of text content	No
Website Usability	Old, hard to navigate, lacks structure	Structured and staged learning	N/A
Learning Approach	Focus on certifications and examination	Module by module, not entirely linear	N/A
Use of Graphics	Used in some of the PDF content and videos	Used throughout to help users understand information	Quiz includes graphics of phishing emails
Feedback Mechanisms	No feedback on quizzes	No feedback in final quiz	Limited feedback
Overall Content Accessibility	Hard to find content and navigate website	Well-structured web pages easy to navigate	N/A

Table 2 - Existing Solutions Comparisons

2.8 Summary

This literature review has delved into many aspects of cyber security threats and effective e-learning strategies which can be used to educate on them. It examines the recent escalation of cyber threats, emphasising their growing sophistication and success rates through tools like AI. This review also highlights the gap between public perception and the actual severity of cyber threats to contextualise the requirement for a platform as proposed in this project.

This review also explores concepts of adult education, focussing on the effectiveness and challenges of e-learning platforms and multimedia learning, continuing to assess a selection of existing cyber security education platforms, identifying their features and limitations.

Through this comprehensive analysis, the review has established a strong foundation for the development of the project going forwards to effectively enhance users' cyber security awareness and fill the gaps in the public's understanding.

Chapter 3
Project Methodology

In this chapter, various software development methodologies will be explored, explaining each methodology in detail; providing advantages and disadvantages for each approach; giving justification for the final methodology used within the project and highlighting the key resources and tools required for its successful implementation to manage the project effectively.

3.1 Waterfall Methodology

The waterfall methodology, a traditional approach to software development, is characterized by its linear progression through defined phases, including Analysis, Design, Coding, Testing, and Acceptance (Adenowo & Adenowo, 2013). While this method offers simplicity and structure, its rigid nature prohibits revisiting completed phases, making it challenging to accommodate evolving project requirements (McCormick, 2012). In essence, the waterfall model demands that the scope and all possible requirements be determined upfront, leaving little room for adaptation or response to changing needs without starting anew (McCormick, 2012).

Given the dynamic nature of this project and the likelihood of evolving requirements, the waterfall model's inflexibility would be inadequate. Instead, a more flexible approach is imperative to effectively address the project's ever-changing landscape and ensure successful outcomes within the given constraints.

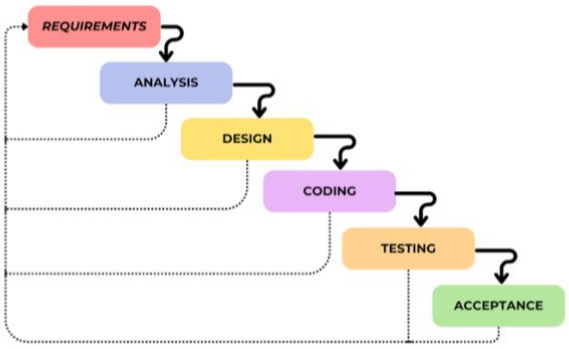

Figure 4 - Waterfall Methodology

3.2 Agile Methodology

The Agile model, named so for its fast-moving nature, is a software development methodology focussed on an iterative and incremental model (McCormick, 2012). Agile software development suggests that production should start off with simple and predictable approximations for the final requirements and adapt through incremental refinement across the life of development (Kumar & Kumar, 2012).

While Agile is commonly associated with collaboration in larger teams, its principles can be effectively adapted for individual projects (Agile Manifesto, 2019). However, it's crucial to acknowledge the challenges that may arise, especially for a solo project with a short deadline. One notable drawback is the emphasis on working software over comprehensive documentation, which may pose difficulties for a solo developer in maintaining thorough documentation while focusing on rapid development. Additionally, the iterative nature of Agile can introduce uncertainty regarding project timelines and deliverables, especially when initial requirements evolve significantly throughout development.

There are 4 key concepts to the agile process as outlined in the Agile Manifesto, (2019), and these are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change by following a plan

software and fostering adaptability Agile streamlines the production process and proves to be an invaluable framework for driving efficient development and ensuring project success within the constraints of a single individual and a short timeframe.

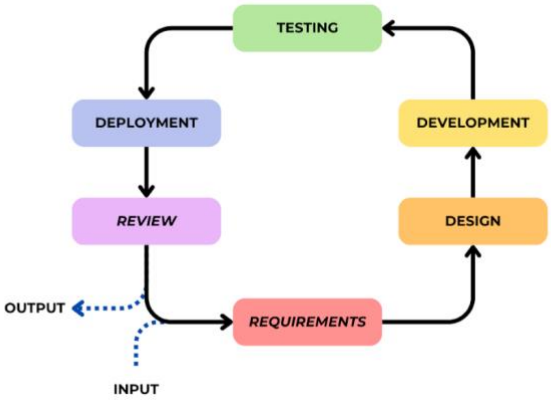

Figure 5 - Agile Methodology

3.2.1 Kanban Method

The Kanban method, Kanban meaning “Signboard,” is traditionally a manufacturing scheduling system, however, in 2004 David J. Anderson proposed it as a software development method which would enable teams to visualise the workflow, limit work in progress and measure cycle time (Ahmad et al., 2013).

The just-in-time approach to manufacturing, as allowed by the Kanban method, is growing in popularity, and gaining attention, replacing the concept of ‘push’ systems where new jobs push along others - Kanban introduces a ‘pull’ system, where each stage’s demand depends on the previous stage (Huang & Kusiak, 1996).

The Kanban method has 4 key principles - visualise work; limit work in progress; focus on flow and continuous improvement - these principles are what maintain efficiency in the system and reduce waste. Visualisation relies on Kanban cards, a vital component of the system (Wakode et al., 2015).

Whilst the Kanban method was developed with manufacturing in mind, the basic principles merge well over to software, allowing for work items to be visualised and task completion to be monitored. Paired with the Agile methodology, a Kanban board can aid in the breaking down of the project into manageable chunks (work items) which will independently go through the key stages of design, development, testing, etc.

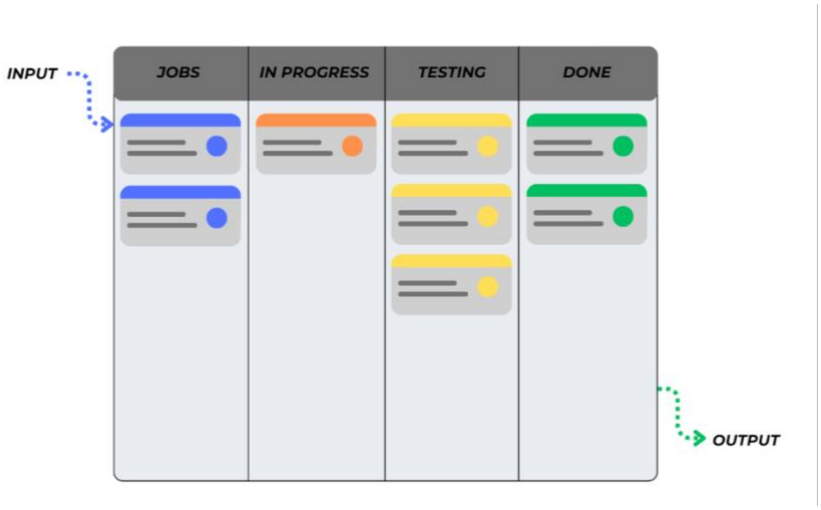

Figure 6 - Kanban Table

3.3 Chosen Methodology Justification

To align with the project's objectives and optimize its development process, the Agile methodology has been selected over the traditional waterfall approach. Agile's flexibility and adaptability offer inherent advantages. As a solo endeavour, the rigid structure of the Waterfall model could hinder responsiveness to unforeseen challenges or opportunities. Agile's iterative approach allows for continual refinement and enhancement, ensuring that the project remains aligned with evolving goals and user needs. Utilising Agile features like Kanban charts enables organized task management, maximizing efficiency and focus. This Agile framework not only streamlines development but also promotes incremental progress, enhancing the project's overall effectiveness and timeliness.

3.4 Methodology Implementation

To implement the Agile methodology effectively, a structured approach will be adopted, using specific tools and techniques to streamline production and ensure timely delivery. Four iterations of programming will be completed, each comprising of task breakdown, implementation, and testing phases. These iterations will serve as incremental cycles of development, allowing for continuous refinement and enhancement of features.

1. **Task Breakdown**: At the start of each iteration, tasks will be identified and broken down into smaller, manageable units, ensuring efficient progress throughout the iteration.

2. **Implementation**: Following task breakdown, the implementation phase will focus on coding and building the features outlined. Maintaining good coding standards, utilising version control, and integrating effectively with existing components to maintain code quality and coherence.

3. **Testing**: Upon completion of implementation, unit testing will be conducted to validate the functionality and reliability of the developed features.

Throughout each iteration, Agile tools such as Kanban boards, task trackers, and version control systems will be used to manage tasks and monitor progress. By adhering to this iterative approach and leveraging Agile principles and tools, productivity can be maximised, whilst maintaining flexibility, and deliver high-quality results within the defined deadlines.

3.4.1 Project Timeline

Utilising a Gantt chart will significantly enhance task management and execution by offering a visual tool that provides a comprehensive view of the project timeline. This enables efficient planning and monitoring of progress against milestones. With a strict deadline in place, adopting a clear strategy for project execution facilitates effective time management and enables the project to be segmented into manageable sections with well-defined start and end dates.

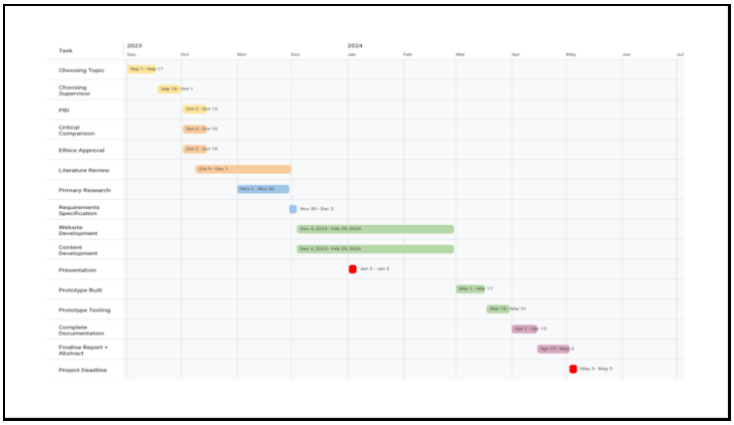

Figure 7 - Gantt Project Timeline

3.4.2 Kanban Board

Incorporating a Kanban chart into the project will also greatly improve task management and efficiency. While a Gantt chart offers a broad perspective of the timeline and project status, a Kanban chart offers real-time insights into work in progress. By categorizing tasks into "To Do", "In Progress", "In Testing", and "Completed" on a large table, it facilitates a smooth and steady workflow of manageable tasks. This becomes particularly valuable when certain tasks encounter obstacles which can have a bottleneck effect on production.

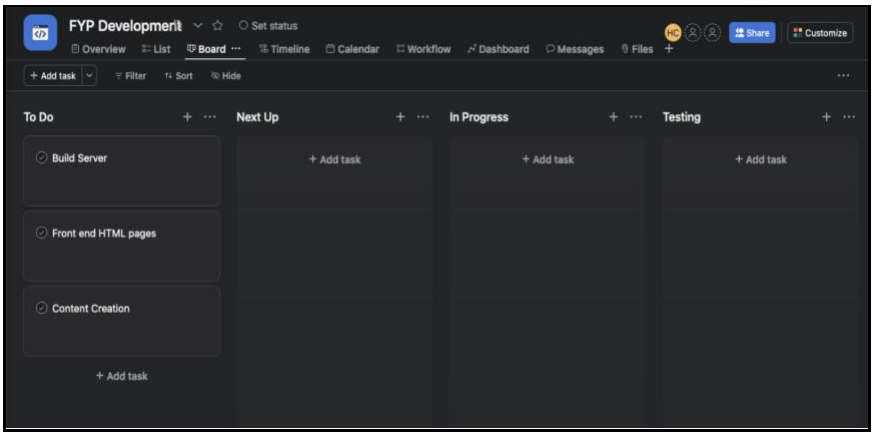

Figure 8 - FYP Kanban Board

3.4.3 Source Control (Git)

Utilisation mechanisms such as Git and GitHub for version control is industry standard in the development of web application programs, allowing for large teams to collaborate on one singular code base using branches to safely develop and test new code functions whilst maintaining code integrity and platform availability. Although this project is being completed individually, many of Git's tools still prove vital for secure and efficient code development, GitHub enables the creation and merging of branches, as well as the ability to revert to previous versions, thereby minimizing the risk to the project's deadline.

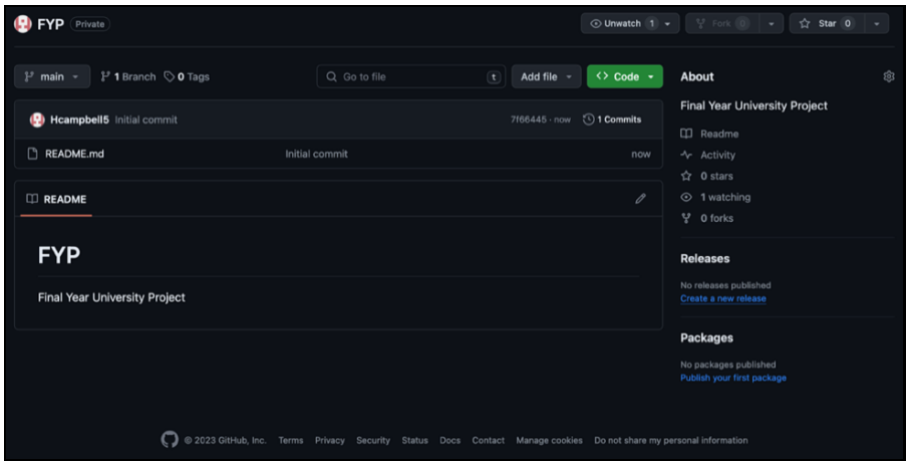

Figure 9 - FYP Github Page

3.5 Summary

In summary, though both Agile and Waterfall methodologies have their merits, the ultimate selection of Agile is justified by this project's demand for flexibility, adaptability toward evolving requirements and a dynamic and iterative development process.

The implementation of Agile methodology throughout the project, including features such as Gantt charts, Kanban tables and Git Source control will allow for effective task visualisation and workflow management leading to a more efficient and speedy development cycle with redundancy measures in place for the inevitable bottlenecks certain tasks will create.

Chapter 4
Requirements

This section will cover the definition and ranking of this project's requirements, considering the necessities of the project itself. It will start with the elicitation of requirements at a foundational level before proceeding to explore more intricate and specific requirements.

4.1 Requirements Elicitation

The requirements for this project were established largely through the research undertaken in Chapter 2, in which several existing systems were analysed for key features and their effectiveness, as well as through a literature review in which many papers proposed essential elements of effective e-learning systems.

4.1.1 Research & Existing Systems

The analysis of existing solutions has identified several key requirements. A paramount need is for a modern intuitive interface which will ensure easy navigation and content accessibility. A further need is content diversity; blend of engaging video and text-based materials is necessary for a comprehensive learning experience, and materials should be contextually rich and interactive. Interactive elements such as quizzes and assessments are also vital for reinforcing learning, with detailed feedback to enable self-improvement.

Alongside these requirements, more general features are also identified such as progress tracking, account creation, statistics and more which will be incorporated into the tables in the next section.

4.2 Requirements

Utilising the information gained in the previous section, requirements elicitation, 18 functional requirements have been identified, and 6 non-functional requirements were also identified.

Requirements will be ranked using a feature of the agile methodology: MoSCoW prioritisation, which establishes a hierarchy of requirements labelled as Must-haves; Should-haves; Could-haves and Won't-haves.

Must-Have - A core requirement which is necessary for the application to meet the project's goal and be considered a success.

Should-Have - A high-priority requirement and attempts should be made to implement these requirements if possible.

Could-Have - A lower priority requirement, which would be beneficial but is not necessary or potentially unfeasible given time constraints.

Won't-Have - Most likely will not be implemented but is considered for future development.

By ranking requirements this way, work prioritisation can easily be shifted toward Must-Have requirements rather than focusing on less function-critical requirements.

4.2.1 Functional Requirements

Functional requirements are those which require the system to do something. Not meeting a functional requirement should cause a system to fail (Gorbachenko, 2021).

ID	Name	Description	Priority	Justification
FR1	Login Authentication	The system should be protected by login authentication. Users must sign in and have permission to access certain aspects of the application. This not only ensures integrity of course content but allows users to save their progress in courses/lessons.	Must Have	Essential for ensuring the security and integrity of the platform. By implementing login authentication, only authorized users can access the application, safeguarding sensitive course content and allowing users to save their progress.
FR2	Account Creation	Users need to be able to sign up and create an account to access the platform.	Must Have	Account creation is a fundamental feature that enables users to access the platform and participate in courses. Without account creation, users would not have personalized access to course content and progress tracking features.
FR3	Account Customisation	Users must be able to update their account details such as name, contact details, role.	Must Have	Allowing users to customize their account details is essential for providing a personalized experience and ensuring user satisfaction. Users should have the flexibility to update their information such as name, contact details, and role as needed.

FR4	User Role Management	Users must be given roles to specify their permissions such as admin, teacher, and Student.	Must Have	User role management is critical for assigning permissions and regulating access to different features of the platform. Admins, teachers, and students must have distinct roles to ensure proper management and security of course content.
FR5	Course Enrolment	Users must be able to enrol in certain courses which interest them, which will be the focus of their learning.	Must Have	Course enrolment is necessary for users to select and participate in courses that align with their interests and learning objectives. Without this feature, users would not have access to course materials and learning opportunities.
FR5	Course Creation	Admins and teachers need to be able to create new content and upload it to the platform in a user-friendly way.	Must Have	Admins and teachers must have the capability to create new content and upload it to the platform to ensure the availability of up-to-date and relevant course materials. This feature is essential for maintaining the platform's educational value.
FR5	Course Management	Admins and teachers need to be able to update content on the platform to correct mistakes or update new information.	Must Have	Admins and teachers require the ability to manage and update content on the platform to ensure accuracy and relevance. This feature enables corrections of mistakes and updates of new information.

FR6	Database System	The application will require a database system to manage users as well as the content on the platform itself in a secure way.	Must Have	A robust database system is essential for securely managing user data and course content. This requirement ensures data integrity, scalability, and efficient management of platform resources.
FR7	Content Delivery Mechanisms	There should be several content delivery mechanisms such as video integration, graphics, text pages.	Must Have	Providing multiple content delivery mechanisms such as videos, graphics, and text pages enhances the learning experience and caters to different learning styles. This feature promotes engagement and comprehension of course material.
FR8	Assessment Mechanisms	There should be assessment mechanisms such as quizzes to reinforce learning.	Must Have	Assessment mechanisms such as quizzes are essential for reinforcing learning and evaluating user comprehension. This feature allows users to gauge their understanding of course material and track their progress.
FR9	Progress Tracking	There should be a progress tracking feature to promote learning.	Should Have	Progress tracking enhances the learning experience by allowing users to monitor their advancement through courses. While not essential, this feature promotes accountability and motivates users to achieve their learning goals.

FR10	Course Feedback	There should be a mechanism for users to give feedback based on courses.	Should Have	Incorporating a mechanism for users to provide feedback on courses enables continuous improvement and ensures course content remains relevant and effective.
FR11	Course Statistics tool	There should be a mechanism for viewing course statistics such as completion rate, average time taken and user feedback.	Should Have	A course statistics tool provides valuable insights into course performance, completion rates, and user feedback. While not critical, this feature can inform decision-making and optimize course content.
FR12	Notifications and Alerts	There could be a notifications tool, to convince users to come back and finish their learning.	Could Have	Implementing notifications and alerts can encourage user engagement and prompt users to return to the platform. While beneficial, this feature is not essential for basic functionality.
FR13	Real-Time Communications	There could be a real-time communication feature, allowing students and educators to actively discuss elements of their learning.	Could Have	Real-time communication features facilitate active discussions between students and educators, enhancing collaboration and engagement. While desirable, this feature is not essential for core functionality.
FR14	Streak Feature	There could be a streak-feature to encourage repeated daily learning and reflection.	Could Have	A streak feature incentivizes daily learning and fosters consistent engagement with the platform. While motivational, this feature is not critical for basic functionality.

FR15	Certificates / Badges	There could be a certificate/badge system to reward users for continued development and completion of modules/courses.	Could Have	A certificates/badges system rewards users for their achievements and promotes continued learning. While motivational, this feature is not essential for core functionality.
FR16	Time Tracking	There could be a system for tracking the time users take to complete modules or courses - used to suggest how much time it takes on average to new starters.	Could Have	Time tracking provides insights into user Behavior and learning habits, allowing for personalized recommendations. While informative, this feature is not essential for basic functionality.
FR17	Game-ified Learning	There could be elements of the e-learning encapsulated within small flash games.	Won't Have	While gamified learning can enhance engagement, it is not feasible for implementation within the scope of this project.
FR18	Live Tutorials	There could be live tutorials scheduled for certain times which could include guest lecturers.	Won’t Have	Live tutorials require additional resources and infrastructure that are not available for this project. Therefore, they are not feasible for implementation within the current scope.

Table 3 - Functional Requirements

4.2.2 Non-Functional Requirements

Non-functional requirements are those which describe how the system will work. They do not have an impact on the functionality of the system but do impact its performance and usability – failing to meet functional requirements could lead to users being frustrated with the system (Gorbachenko, 2021).

ID	Name	Description	Priority	Justification
NFR1	Compatibility	The application must be able to run on multiple web platforms and across multiple device formats such as desktop, mobile and tablet.	Must Have	Ensuring compatibility across multiple web platforms and device formats is essential for maximizing user accessibility. This requirement guarantees that users can access the application seamlessly regardless of their preferred device or platform, enhancing user experience and platform usability.
NFR2	Security	All personal data should be secured in accordance with GDPR, and critical systems must be protected from abuse.	Must Have	Security is a critical aspect of any application, particularly one handling personal data. Compliance with GDPR regulations will safeguard user privacy and maintain integrity. Implementing robust security measures is paramount to develop user trust and prevent data breaches.
NFR3	Availability	The application should be available for use 24/7 for multiple users at once.	Should Have	Whilst 24/7 availability is desirable, it is not always feasible due to maintenance or unforeseen circumstances. However, ensuring high availability for multiple users at once enhances user satisfaction and ensures uninterrupted access to the platform's services, improving overall user experience.

NFR4	Performance	The application should be able to respond correctly to user input with a low fail/error rate.	Should Have	Maintaining optimal performance with a low fail/error rate is essential for providing a smooth and seamless user experience. While not as critical as security or compatibility, ensuring reliable performance minimizes user frustration and enhances platform usability, ultimately contributing to user satisfaction.
NFR5	Usability	The user interface should be intuitively designed and easy to use.	Should Have	The user interface plays a crucial role in user engagement and satisfaction. Designing an intuitively designed and easy-to-use interface improves user satisfaction. While usability is important, it may not be as critical as security or compatibility.
NFR6	Scalability	The system should be designed with future expansion in mind leaving room for development and growth to a larger user base.	Could Have	While scalability is desirable for accommodating future growth and expansion, it may not be immediately necessary depending on the current user base and projected growth rate. However, designing the system with scalability in mind ensures flexibility and adaptability for future enhancements, supporting long-term sustainability and growth.

Table 4 - Non-Functional Requirements

4.3 Summary

This section has successfully outlined the essential requirements for a cyber security e-learning platform, derived from an in-depth analysis of existing e-learning systems and relevant academic literature. The requirements are categorized into functional and non-functional groups, prioritized using the MoSCoW method.

Functional requirements form the core of the system's operation, focusing on user interaction and content management aspects such as login authentication, account creation, user role management, course enrolment, and diverse content delivery mechanisms. Key features include interactive quizzes with detailed feedback, progress tracking, and course feedback mechanisms.

Non-functional requirements address the operational quality of the platform, emphasizing compatibility across devices, GDPR-compliant security, continuous availability, performance efficiency, user-friendly design, and scalability for future growth.

This chapter sets a comprehensive foundation for the platform's development, balancing a user-centric approach in functionality with high operational standards, ensuring the platform is not only effective but also adaptable to future advancements.

Chapter 5
Design

In this chapter, all aspects of the design phase will be discussed and analysed to determine how the final product will both look and function. The project itself will be a web-based e-learning platform, therefore requiring a keen focus on usability and cross device functionality. To develop this platform for web use, a collaboration of suitable programming languages and platforms will be required and outlined in this chapter. By the end of this chapter there will be a clear understanding of how the product will be created, how it will look and how it will function behind the scenes.

5.1 System Design

5.1.1 System Architecture

For the development of this project, there are several options to consider when it comes to the systems' architecture. Taking into consideration the requirements found in the previous section, which identify usability, scalability, and availability as key non-functional requirements, a hybrid architecture is suggested.

By integrating a traditional client-server model with cloud-based principles optimal efficiency, scalability, and reliability can be ensured. The client-server model is foundational, allowing for effective interaction between users (clients) and a platform's server - enabling efficient processing of user requests including content access and interactive activities. The integration with cloud-based solutions for server hosting will capitalise on the strengths of cloud computing, allowing for dynamic resource allocation with a robust infrastructure which can support a platform's growing demand. Not only will the use of cloud services increase efficiency, but it will also allow for cost-effective server hosting. As mentioned in the project's constraints, there is insufficient budget to host a physical server - especially one which would match the functionality of cloud-hosting.

Therefore, this hybrid system architecture strategy, which combines client-server with cloud computing, will offer a balanced solution ensuring the platform's robust functionality, user experience and operational efficiency whilst also positioning it for potential future expansion.

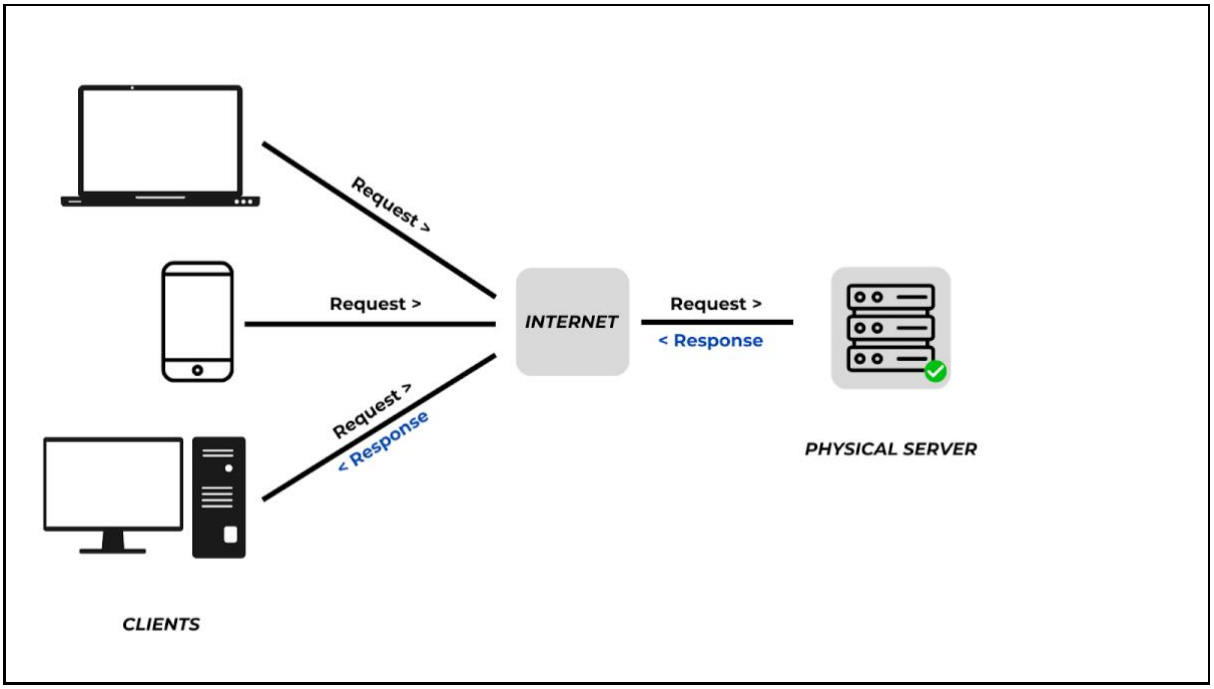

Figure 10 - Single Server Diagram

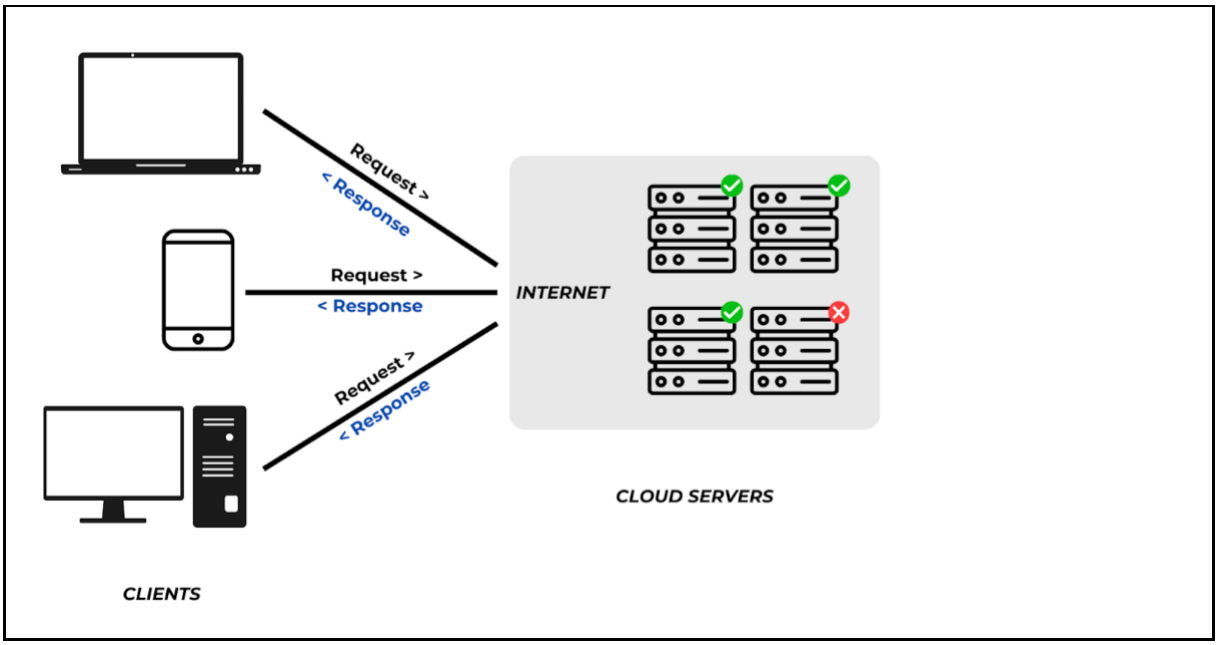

Figure 11 - Cloud Server Diagram

The above figures demonstrate the necessity for multiple cloud servers, which can actively respond to multiple clients with redundancy and backup measures in place for server failures. In Figure *10*, the single server connection proves to be a bottleneck in client-server responses which would slow down the platform as a whole.

5.1.2 Web Architecture / Flutter

The Worldwide Web itself operates on the foundational principles of client-server architecture, a critical concept which will be adopted into the e-learning platform. In this architecture, when a learner interacts with the platform via a web browser (client), they are initiating requests for content or services which are transmitted over the internet using protocols such as HTTP and HTTPS, to the servers. In the context of this platform, said servers will be hosted in a cloud environment which forms the backbone of the service. The cloud server will be responsible for processing these requests, managing course content, user data, and providing dynamic responses such as interactive web pages or multimedia content. The server's response is then rendered by the client's browser, delivering to the user the requested content. This client-server interaction, bolstered by the scalability and flexibility of cloud computing, ensures that the platform can efficiently handle varying user loads and extensive data, whilst maintaining high performance and continuous availability. The integration of these technologies not only facilitates a seamless learning experience but also ensures the platform's capability to remain adaptive and secure, aligning with evolving educational demands and technological advancements.

5.1.3 Flutter Codebase

This project will utilise Flutter, an open-source UI software development kit created by Google which operates using the Dart programming language. Flutter is known for its ability to enable the development of natively compiled applications for mobile, web, and desktop from a single codebase.

Opting to use Flutter in this project offers distinct advantages over using a traditional stack of Node.js with raw HTML, JavaScript, and CSS, particularly in terms of development efficiency and cross-platform consistency. While Node.js paired with HTML, JavaScript, and CSS is a powerful combination for building web applications, it typically requires separate codebases to create equivalent native applications for iOS and Android. In contrast, Flutter's cross-platform nature allows for a single codebase to develop applications that run natively on multiple platforms, reducing development time and effort.

Furthermore, Flutter's comprehensive widget library and its own rendering engine will simplify the creation of visually appealing and highly responsive UIs without complex CSS styling or HTML layout design. The 'hot reload' feature of Flutter will also enhance productivity by enabling real-time previews of changes, which can be more cumbersome in traditional web development. These features make Flutter a far superior choice for developing an e-learning platform, ensuring faster development, uniformity across different operating systems, and a more engaging user interface.

5.1.4 Security

The security of a web application is never more clearly paramount than in a web application built solely for cyber security education. As explored in Chapter 2, in today's digital landscape, where cyber threats are rampant, safeguarding user data and platform integrity is essential to ensure user trust and maintain confidentiality.

Utilizing Flutter for development offers a few inherent security advantages. Flutter's compiled nature provides protection against common vulnerabilities such as cross-site scripting (XSS) and injection attacks, as the code is compiled into native machine code, reducing the risk of exploitation. Additionally, Flutter's widget-based architecture encourages a structured approach to UI development, minimizing the likelihood of UI-related vulnerabilities.\

By integrating Firebase as the database and authentication provider, this project further increases its security measures. Firebase's NoSQL database offers built-in security features such as data validation rules and user authentication, mitigating risks associated with SQL injection attacks and unauthorized access. The real-time database capabilities of Firebase ensure data consistency and integrity, crucial for maintaining a secure environment.

Firebase's authentication services allow users to securely sign in to the platform using various methods, including email/password, phone number, or third-party providers such as Google or Facebook. Firebase handles the complexities of user authentication, including password hashing and encryption, ensuring that user credentials are protected from unauthorized access.

Furthermore, Role-Based Access Control (RBAC) is implemented to regulate user access and permissions within the platform. RBAC assigns roles such as student, teacher, and admin - each with specific permissions, ensuring that users can only access functionalities relevant to their role. This prevents unauthorized access to sensitive information and reduces the risk of data breaches.

In summary, the combination of Flutter's inherent security features, Firebase's robust database security, and authentication services, and the implementation of RBAC ensures that this project is equipped with comprehensive security measures to protect against common vulnerabilities and safeguard user data.

5.2 Database Design

In order to efficiently manage data in a web application, a database is critical. It acts as the central hub for storing and retrieving data, whether accessed through a web browser or mobile application. This setup will allow for a seamless, synchronised learning experience across multiple devices. The database, hosted in the cloud, ensures that the educational content is always available, scalable, and backed up. Using Flutter and Dart, the client side can dynamically display content and record user interactions, which are then efficiently managed by the server-side database, providing a robust and interactive e-learning environment. The database's role will also extend to managing user access and progress, essential for both personalised learning and administration.

5.2.1 Firebase

The platform will utilise Firebase for its database requirements. Firebase is a platform developed by Google which has many advanced features for real-time data management including immediate updates and synchronisation. The scalability of Firebase will support any platform growth either in terms of user numbers or data from course content. Firebase's straightforward integration with Flutter is ultimately the reason for its use, simplifying the development process. Firebase also offers robust security features like authentication and data encryption, and its cloud-hosted nature also means that there is less need for extensive server management. These aspects contribute to Firebase being a practical choice for the project, considering factors like real-time data handling, scalability, and ease of integration.

5.2.2 Database Schema

The Entity-Relationship Diagram (ERD) graphically represents the database layout and the relationship between each table (or collection in the context of Firebase Fire store) and the information stored within.

The main application will be hosted on a cloud platform, and it will manage data for various courses, users, lessons, quizzes, and other related entities. This necessitates an organised structure to efficiently store and retrieve data relevant to each entity. The schema is designed with a focus on modularity and scalability.

- **Users Collection**: Represents the users of the platform. It includes information like user ID, username, email, password hash, role, date of creation, and last login.
- **Courses Collection**: Each document in this collection represents a course, detailing its ID, title, description, category, instructor ID, creation date, and active status. The **`Courses`** collection links to the **`Users`** collection via **`InstructorID`**.
- **UserCourses Collection**: Acts as a junction between **`Users`** and **`Courses`**, managing user enrolments in courses. It stores details like user course ID, user ID, course ID, progress, enrolment and completion dates, and completion status.
- **Lessons Collection**: Contains individual lessons associated with each course. Each lesson document includes a lesson ID, course ID, title, content, and order index.
- **Quizzes Collection**: Stores quizzes related to each lesson, with details like quiz ID, lesson ID, title, and description.
- **Questions Collection**: Represents questions in each quiz, containing question ID, quiz ID, question text, and the correct answer.
- **UserQuizAttempts Collection**: Manages the attempts made by users on quizzes, storing attempt ID, quiz ID, user ID, attempt date, and score.
- **UserAnswers Collection**: Records answers given by users in quizzes. It includes user answer ID, attempt ID, question ID, and the answer given.
- **Resources Collection**: Stores resources associated with lessons, including resource ID, lesson ID, type, URL, and description.
- **Discussions Collection**: Manages discussion threads for courses, with each document containing a discussion ID, course ID, user ID, message, and date posted.
- **Announcements Collection**: Contains announcements for courses, including announcement ID, course ID, title, content, and date announced.
- **Feedback Collection**: Stores user feedback on courses, including feedback ID, course ID, user ID, comment, rating, and date provided.

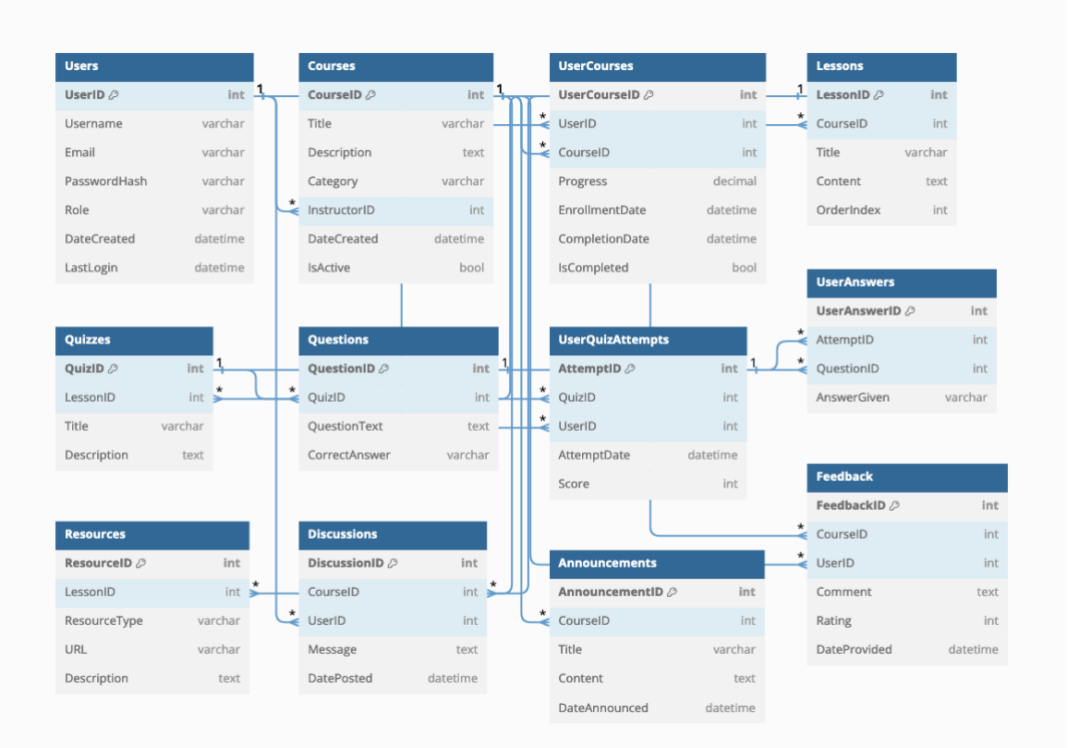

Figure 12 - Database Schema

Firebase uses JSON because it is designed as a NoSQL database, which is different from traditional SQL databases. NoSQL databases like Firebase are built to handle a variety of data types and structures, and they work best with JSON, a flexible and straightforward format. To allow for this, the information within the tables above is converted into a JSON format which can be imported into Firebase to create its “collections”.

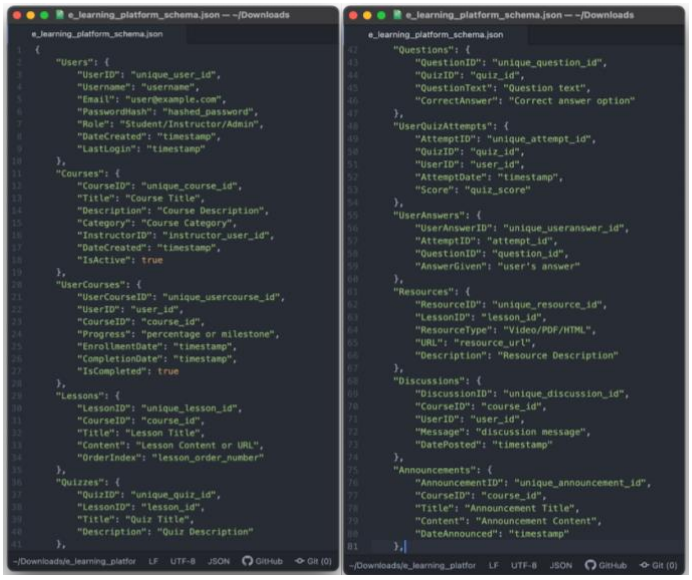

Figure 13 - Database Schema JSON Conversion

5.2 Use Cases

Use case diagrams for software and system design offer a clear and concise method of visualising the interactions between users and the system. They help in identifying and organising the systems' requirements and users' interactions with the system through use-cases. Creating a use-case diagram allows a clear visualisation of the different ways the system will be used and interacted with allowing for a focus on user experience and essential features. This user-centric approach will ensure that the final product aligns with both the requirements identified and user needs and expectations, ultimately leading to a more user-friendly system. In addition, utilising use-case diagrams will allow for clear prioritisation of tasks during the development process.

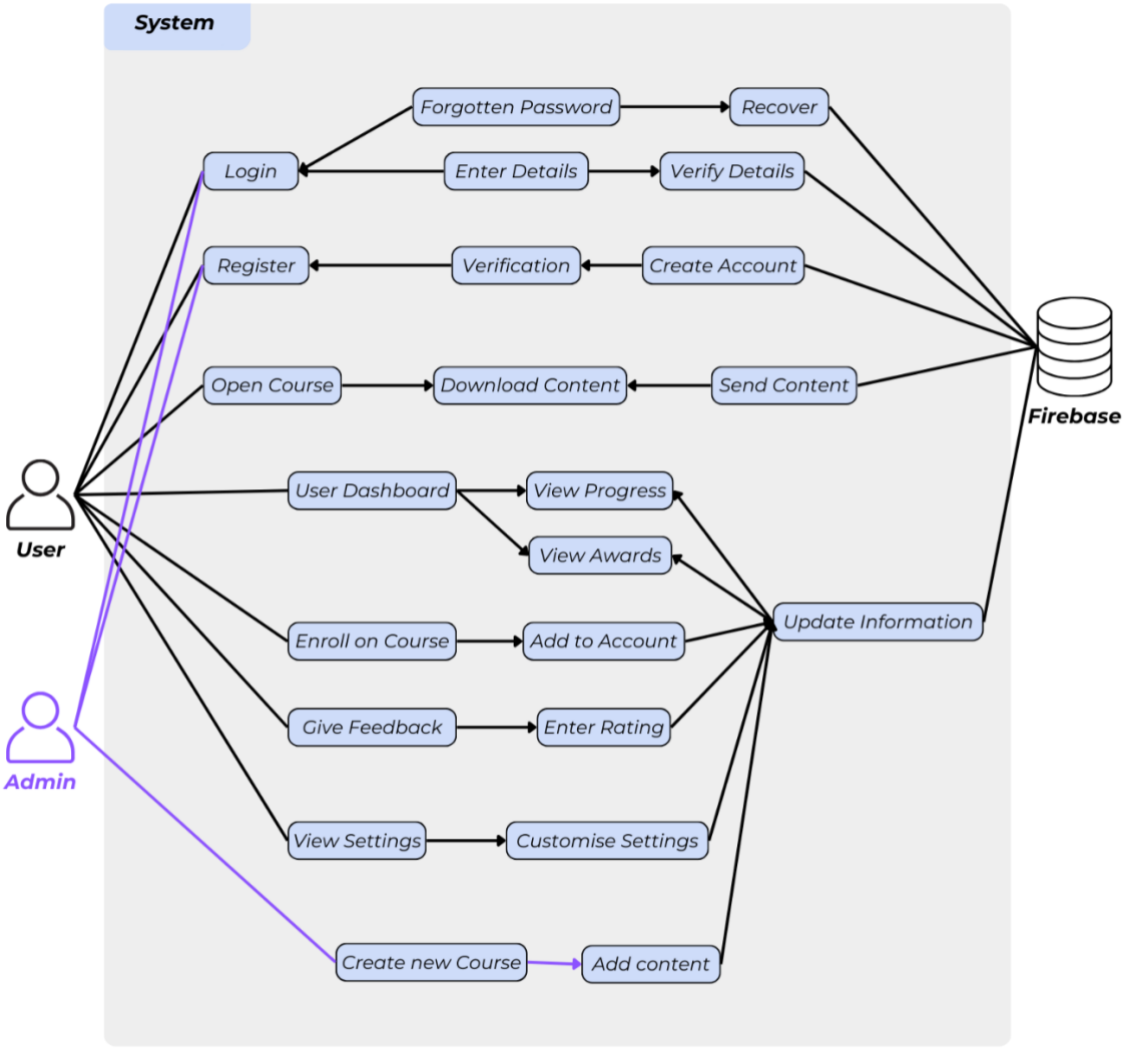

Figure 14 - Use Case Diagram

5.3 User Interface

5.3.1 Colour Scheme

For this cyber security platform, the colour scheme chosen consists of a mostly blue-centric palette, complemented by contrasting shades for both strategic and symbolic reasons. The colour blue is widely recognised in colour psychology for its associations with security, trust, order, and productivity (O'Connor, 2011) these attributes align with the theme of cyber security. The use of varied and contrasting shades also serves a specific function to provide visual depth and focus as well as a more modern interface - facilitating user navigation and enhancing the readability of key elements.

Consideration for colour blindness is also paramount to ensure accessibility as mentioned in the requirements. Colour blindness is a visual impairment which affects users' ability to distinguish certain colours and can significantly impact a user's interaction with digital content. The most common type of colour blindness is red-green - affecting between 6-10% of men (Gordon, 1998). Therefore, the use of a blue-centric and contrasting colour palette will not create barriers to accessibility for those with colour blindness.

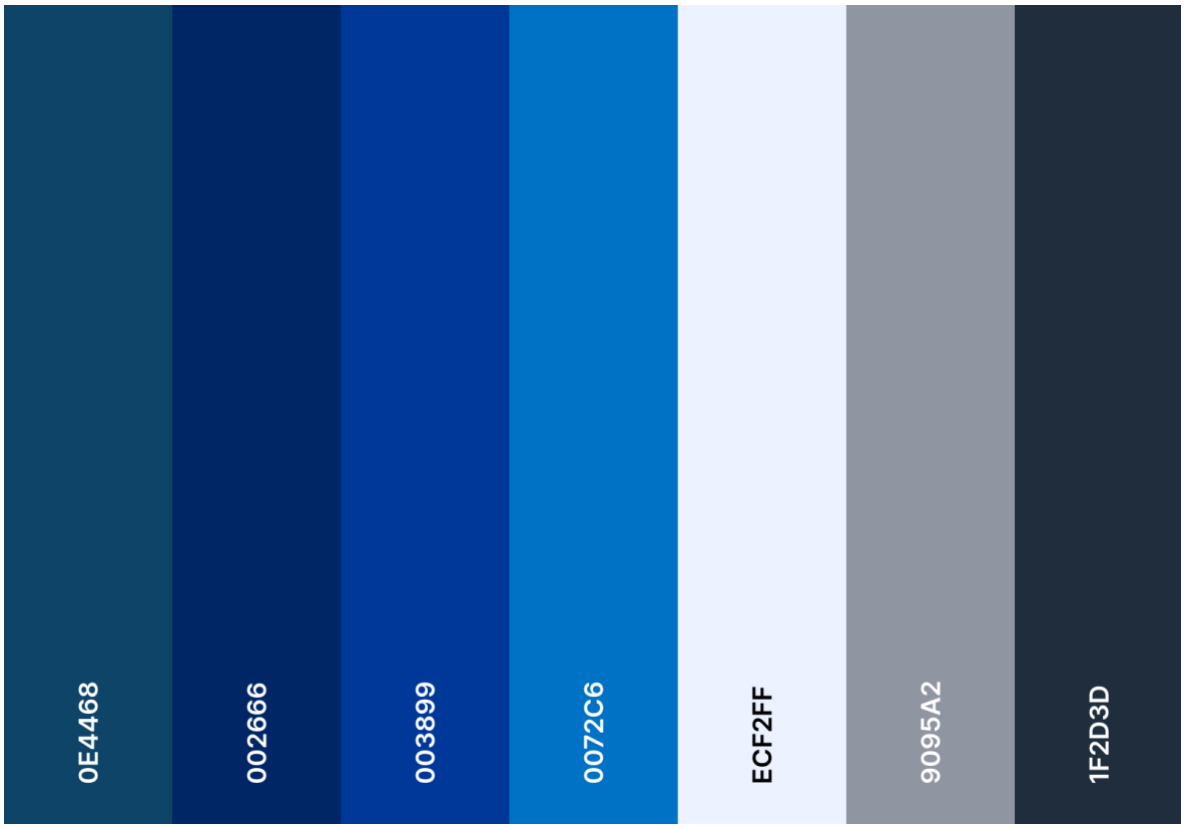

Figure 15 - Project Colour Scheme

5.3.2 Font

The font selected for the platform is ‘Montserrat’, a sans-serif font created by Google, it comes under the open font licence meaning it is free to use. It is a clean, modern font which ensures excellent readability. Its geometric, sans-serif style offers a contemporary feel, making it versatile for various web contexts, and its large library of styles and weights allows for flexible and creative typography. Montserrat is already used in over 20,000,000 websites across the world. Paired with the chosen colour scheme, this font will aid in usability and accessibility of the platform, whilst giving it a clean and modern appearance.

Figure 16 - Project Font

5.3.3 UI Mock-up

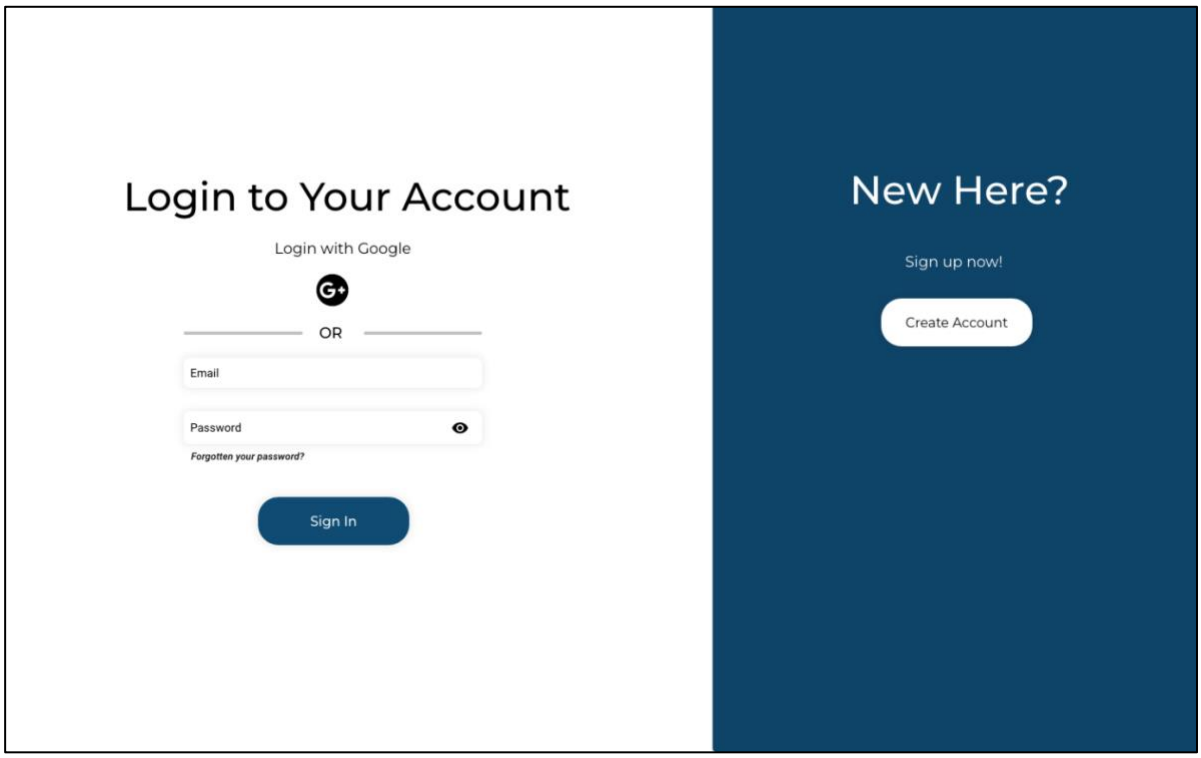

Figure 17 - Login Page UI

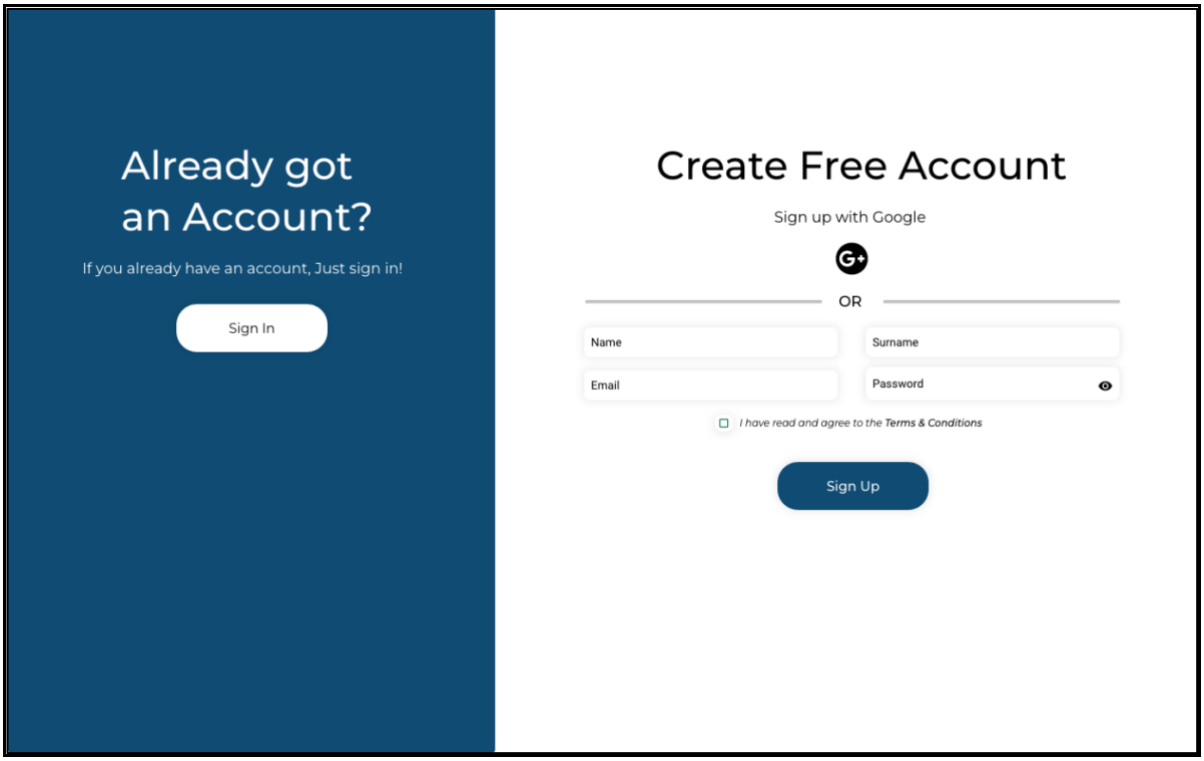

Figure 18 - Sign Up Page UI

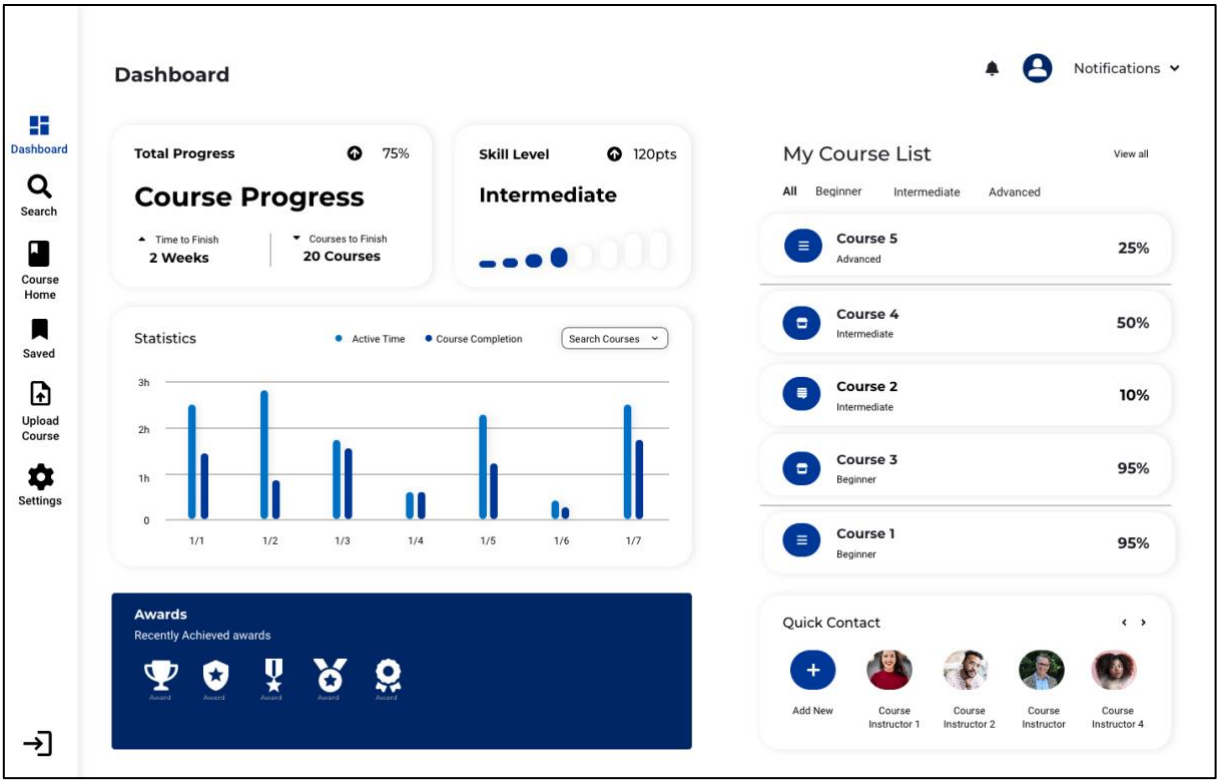

Figure 19 - Dashboard UI

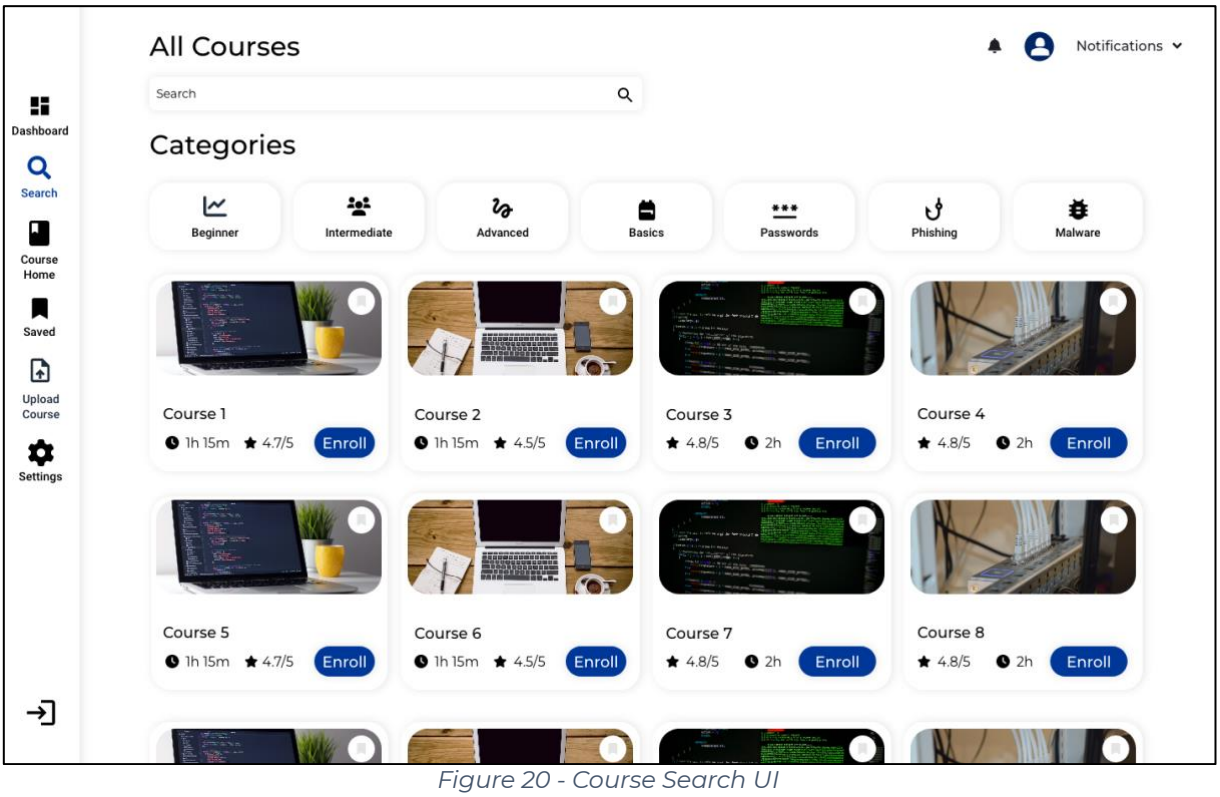

Figure 20 - Course Search UI

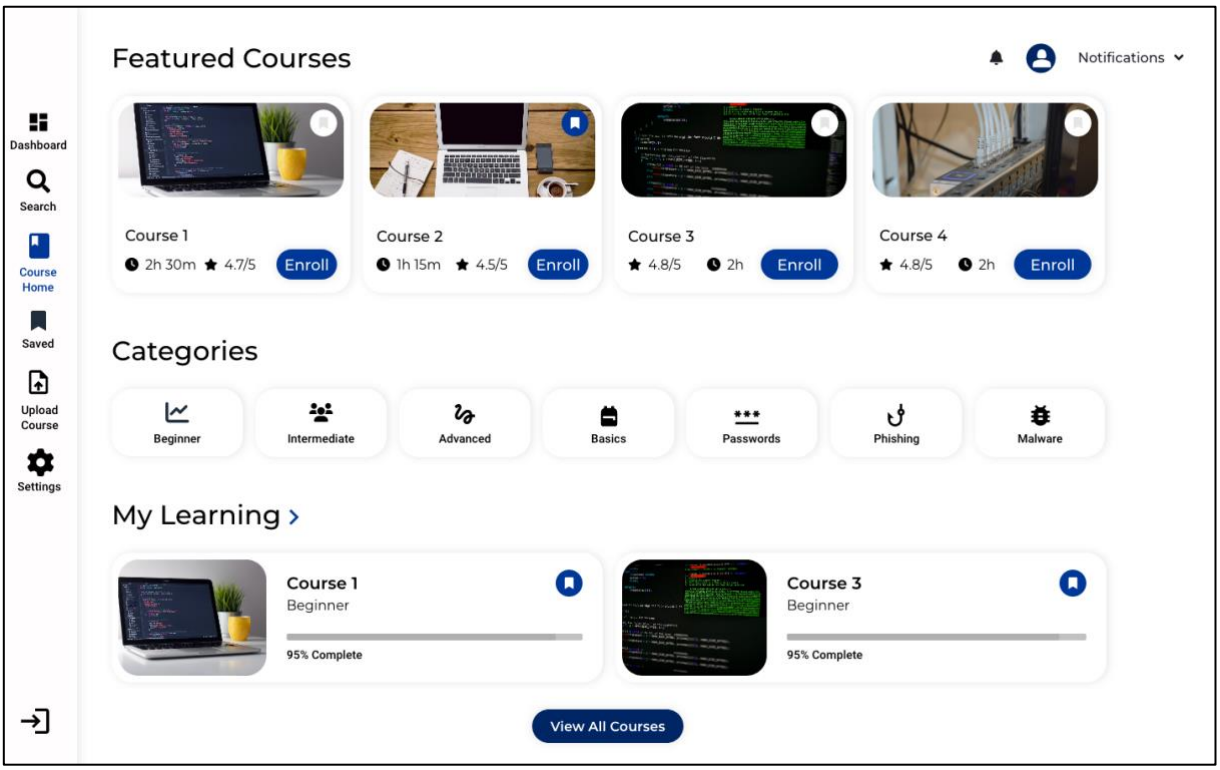

Figure 21 - Course Home UI

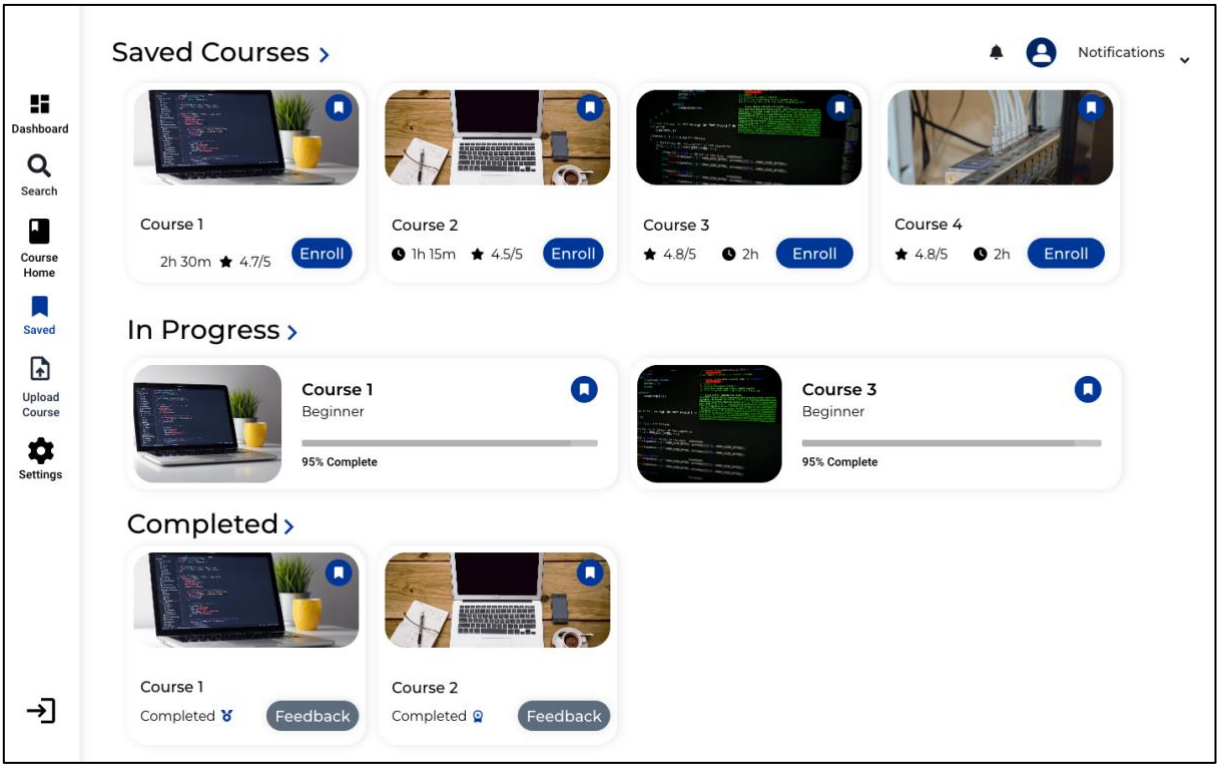

Figure 22 - Saved Courses UI

Figure 23 - Text Lesson UI

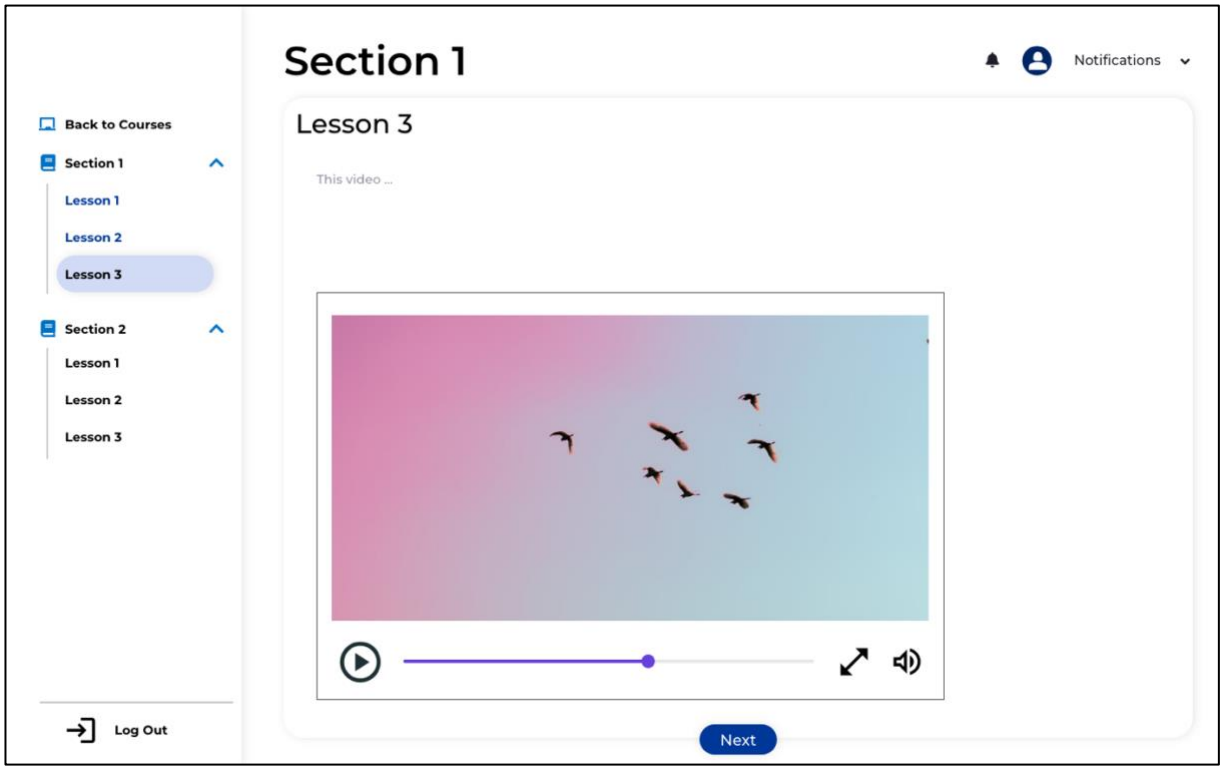

Figure 24 - Video Lesson UI

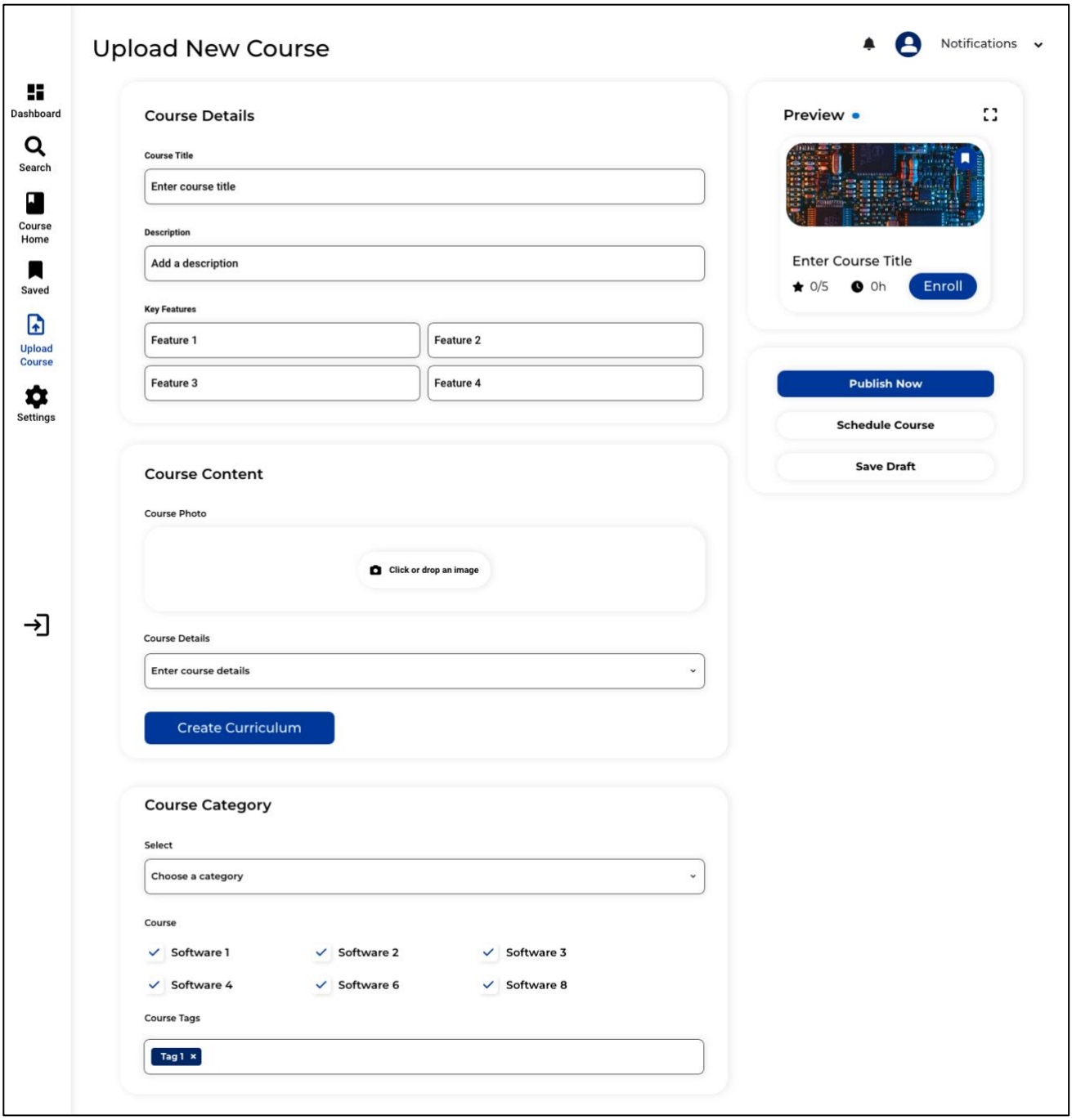

Figure 25 - Course Upload UI

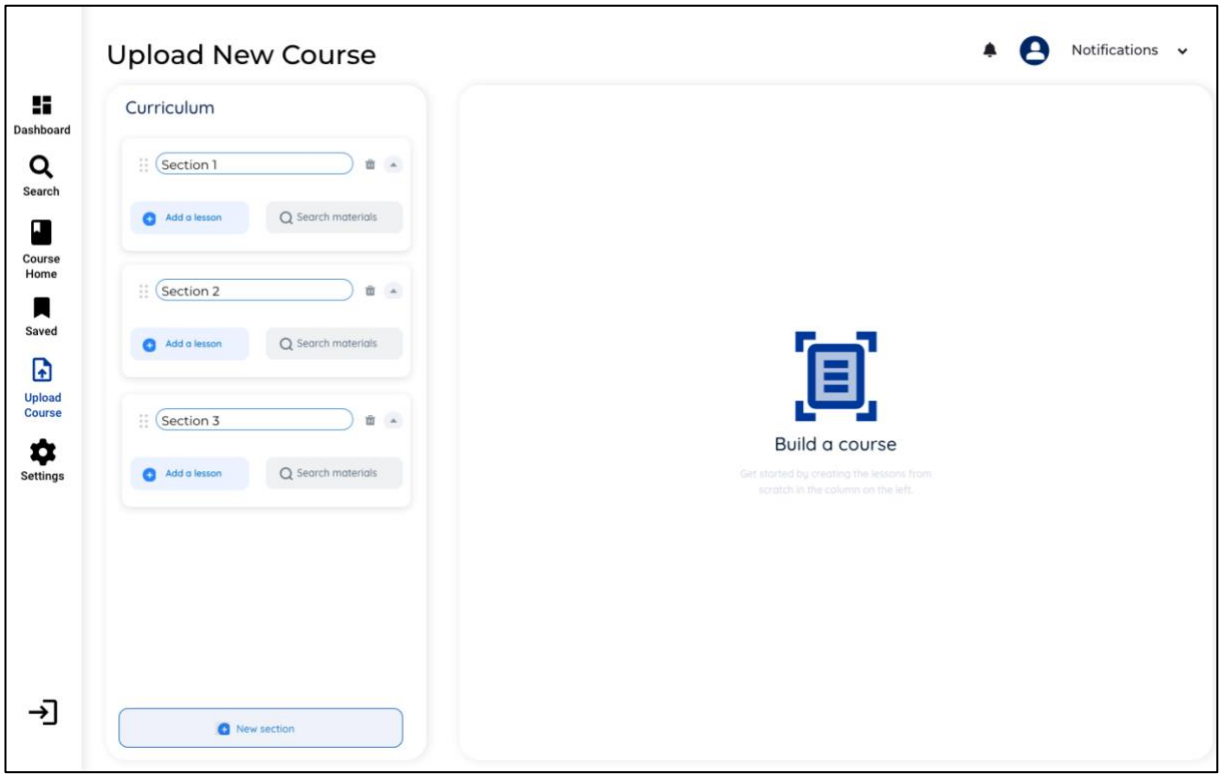

Figure 26 - Add Curriculum UI

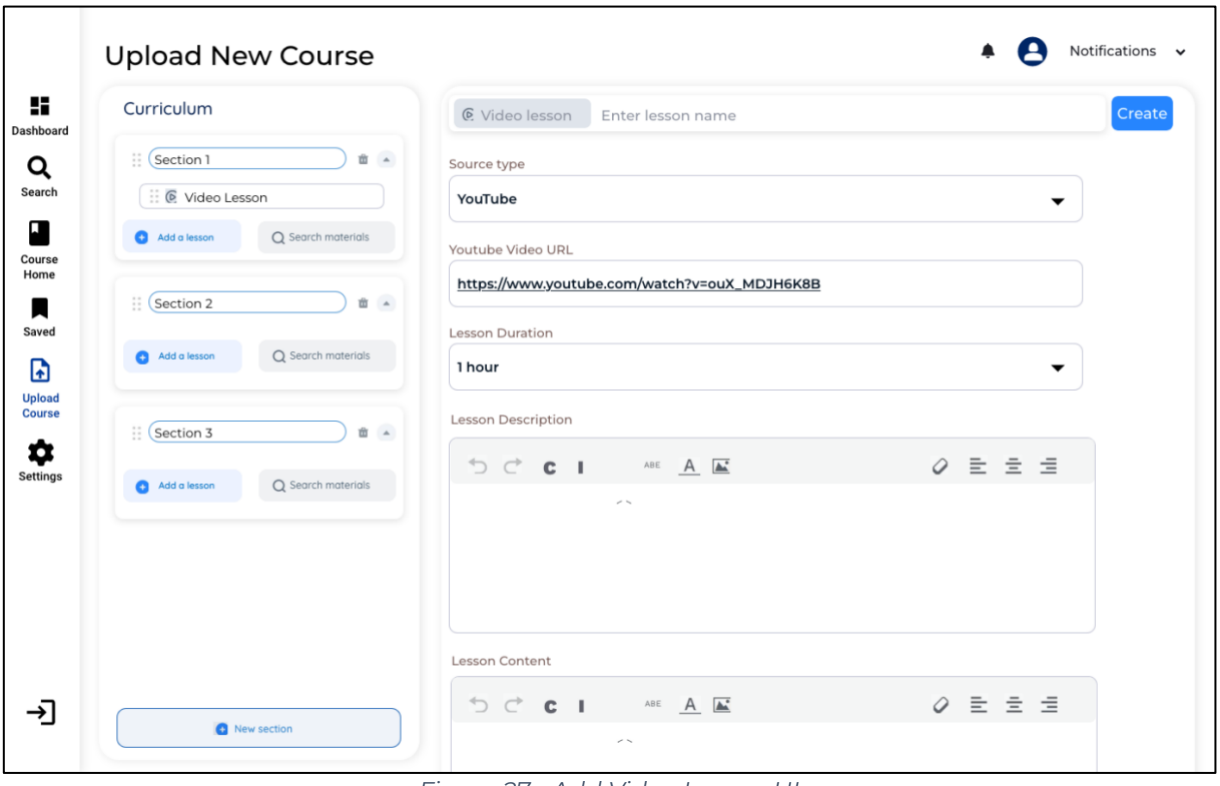

Figure 27 - Add Video Lesson UI

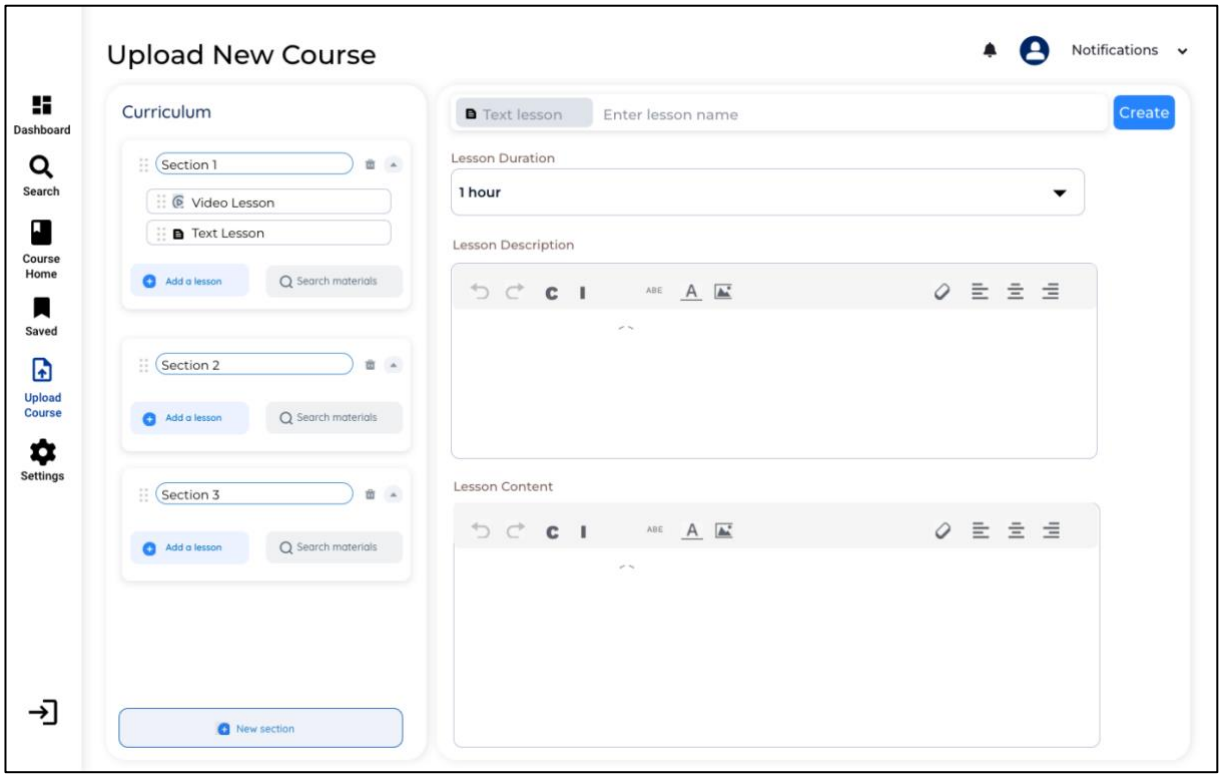

Figure 28 - Add Text Lesson UI

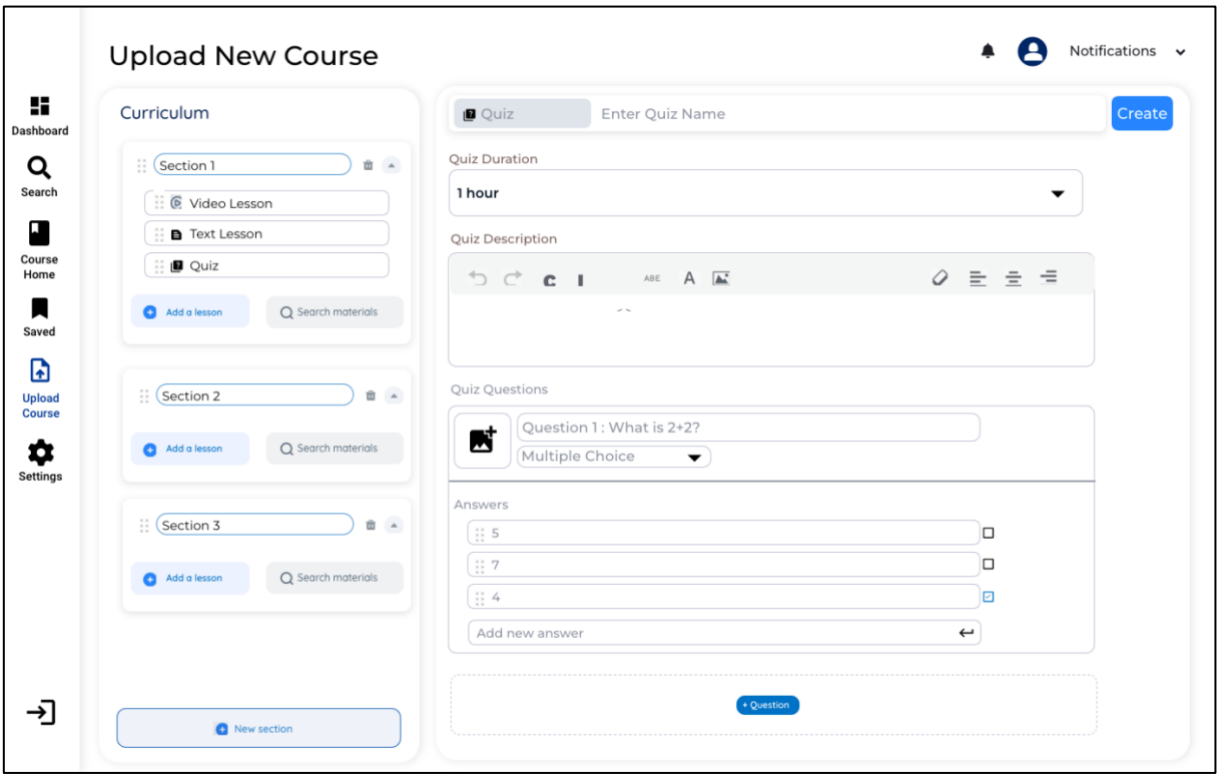

Figure 29 - Add Quiz UI

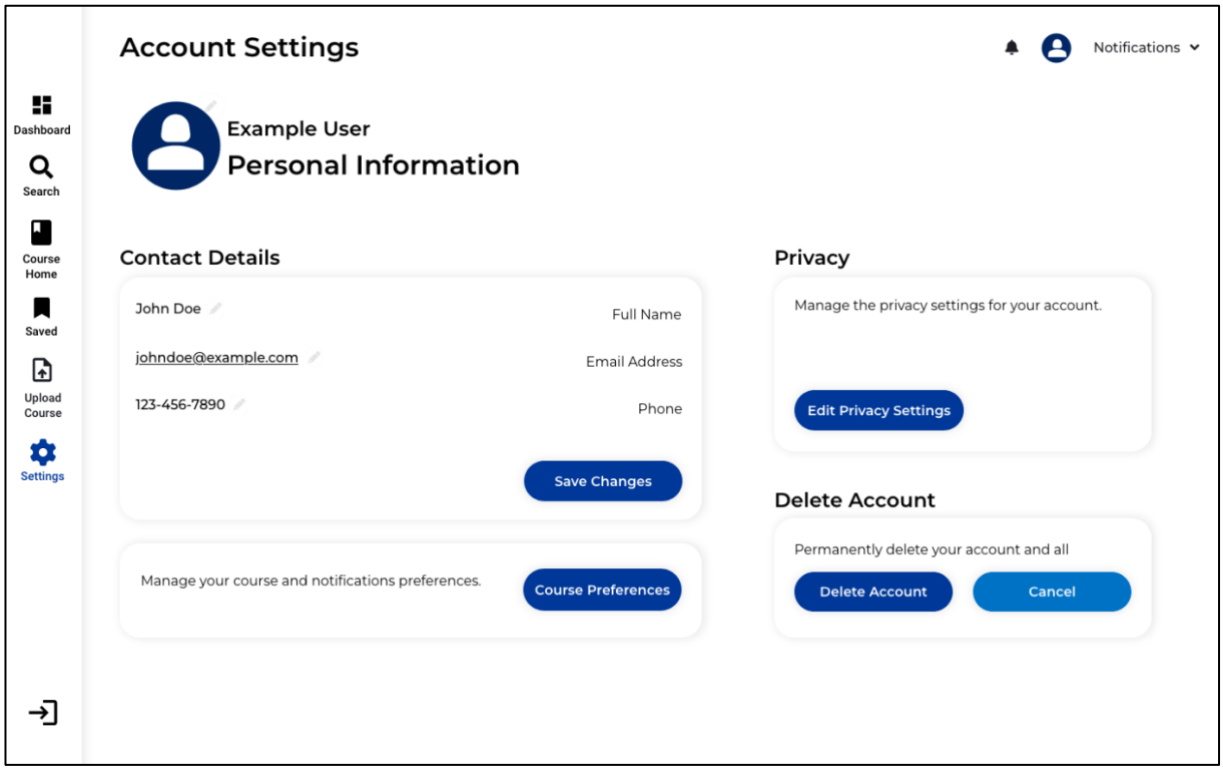

Figure 30 - Account Settings UI

These UI mock-ups were created using the online tool Uizard, allowing for a detailed rendering of what the final platform will look like and how it will function through user interactions. There has been a great focus on usability throughout this section when selecting key components such as font and colour scheme and this has also been followed in the design of the UI to make it as user-friendly as possible.

During the development process, these mock-ups will be greatly useful as templates from which to build the final platform without having to consider usability and other non-functional requirements in the coding environment. Uizard even has the ability to export custom CSS for elements within the UI designs.

5.4 Summary

In this chapter, all elements of the design for a web-based cyber security e-learning platform are comprehensively outlined, providing clarity on its visual, functional, and technological composition. The chapter introduces a hybrid system architecture which blends traditional client-server models with cloud computing elements to cater towards scalability, reliability, and budget constraints. The adoption of Flutter as the UI toolkit is a strategic choice, enabling the development of natively compiled applications for multiple platforms from one codebase. The database design is centred around Firebase, another Google product selected for its capabilities in real-time data management and scalability, ensuring an interactive and seamless learning experience for users.

Through both the use-case diagrams and UI design mock-ups, there is a clear direction for the functional layout of the application which has been designed with user centric principles to allow for an involving and productive learning environment.

Overall, this chapter has laid a solid foundation for the e-learning platform to now be built from, ensuring not only an operationally efficient and robust platform but also a friendly, adaptable, and user-centric platform.

Chapter 6
Implementation

This chapter focuses on the implementation stage of the project, detailing each iteration's process, decision-making, and rationale. Following the requirements established from chapter 4, features have been divided into iterations, each involving planning, execution, modifications, testing and review. The goal of this chapter is to implement the 'must have' features identified in chapter 4, whilst aligning with the design principles discussed in chapter 5 such as a user centric approach.

As mentioned in chapter 3: Methodology, several features of Agile will be utilised throughout development for task breakdown and prioritisation. Each iteration's features are broken down into individual tasks on a Kanban board to monitor progress and ensure timely completion. Each iteration will also include various tests on the features implemented to identify any errors which may occur regularly or irregularly; any usability issues and also any performance issues. Testing in this stage will be conducted manually, with the inbuilt web debugger identifying any additional issues. The next chapter will cover testing of the system as a whole with a wider range of variables considered and using automated tools.

6.1 Iteration 1

6.1.1 Features to Implement

- Login Authentication (FR1) (NFR2)
- Account Creation (FR2)
- Account Customisation (FR3)
- Google OAuth 2.0 Authentication (FR1) (NFR2)
- Navigation Pane (NFR5)

6.1.2 Task Breakdown

The task has been broken down into actionable tasks to be used throughout the development process.

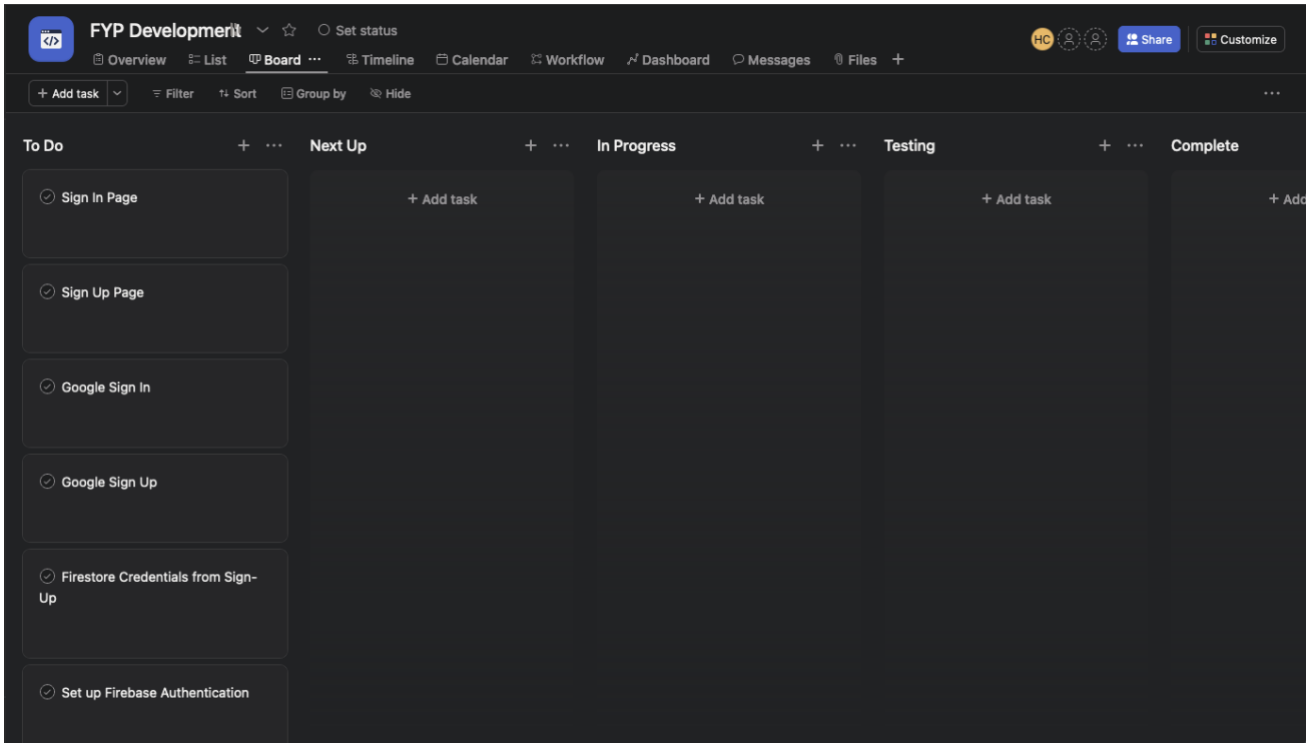

Figure 31 - Iteration 1 Task Breakdown

6.1.3 Design

The design of these features will incorporate features from Figures 17, 18 and 30 in the design section, for the Sign in, Sign up and Settings page. These features will also require the use of Firebase and Firestore for authentication and user account storage.

6.1.4 Implementation

Login Authentication

In order to implement many features of the e-learning platform, users will need to have an account with which they can be uniquely identified for progress tracking, notifications, streak status and awards/certificates. Upon opening the application, users will be greeted with a sign in page, which offers email sign in or sign in with Google - there is also an option to be taken to another page for sign up. Upon credential verification, the user will be taken to a home page - if the current user is signed in already, they should be taken straight to the home page.

```dart
Future<bool> signInWithEmailAndPassword(String email, String password) async {
  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    return true; // Sign-in successful
  } catch (e) {
    print('Error during sign in: $e');
    return false; // Sign-in failed
```

Figure 32 - Sign In [Code]

Using the Firebase auth package to sign in with email and password, this ensures a high-level of security as plaintext passwords are never revealed.

```dart
      child: ElevatedButton(
        onPressed: () async {
          bool success = await signInWithEmailAndPassword(
            emailController.text,
            passwordController.text,
          );
          if (success) {
            Navigator.pushReplacementNamed(context, '/home'); // Navigate to Home Screen
          } else {
            // Display an error message
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(content: Text('Login failed. Please try again.')),
            );
          }
        },
        style: ElevatedButton.styleFrom(
          backgroundColor: Color.fromARGB(255, 15, 75, 115), // Background color
          foregroundColor: Colors.white,
        ),
        child: Text('Sign In'),
      ), // ElevatedButton
```

Figure 33 - Sign In [Code]

```dart
Future<bool> signInWithGoogle() async {
  try {
    final GoogleSignInAccount? googleUser = await GoogleSignIn().signInSilently();
    if (googleUser == null) {
      // check the user is not already signed in
      return false;
    }
    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );
    await FirebaseAuth.instance.signInWithCredential(credential);
    return true; // Sign-in successful
  } catch (e) {
    print('Error during Google sign in: $e');
    return false; // Sign-in failed
  }
}
```

Figure 34 - Google Sign In [Code]

The Firebase package also includes methods for Google Sign in, this is even more secure as it redirects to Googles' own verification site.

```dart
Center(
  child: Padding(
    padding: const EdgeInsets.all(0.0),
    child: IconButton(
      icon: Image.asset('/Users/hcampbell/Documents/Uni/Year3/FYProject/Flutter/my_first_flutt
      onPressed: () async {
        bool success = await signInWithGoogle();
        if (success) {
          Navigator.pushReplacementNamed(context, '/home'); // Navigate to Home Screen
        } else {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text('Google sign-in failed. Please try again.')),
          );
        }
      },
    ), // IconButton
  ), // Padding
), // Center
```

Figure 35 - Sign In Button [Code]

```dart
Future<void> _sendPasswordResetEmail(BuildContext context, email) async {
  try {
    await FirebaseAuth.instance.sendPasswordResetEmail(
      email: email,
    );
    // Show a success message or navigate to a success screen
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Password reset email sent. Check your email inbox.'),
      ), // SnackBar
    );
  } catch (e) {
    // Show an error message if there's an issue
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Error sending password reset email: $e'),
      ), // SnackBar
    );
  }
}
```

Figure 36 - Forgotten Password Button [Code]

Adding and automated password reset is a simple implementation which drastically increases the platforms usability.

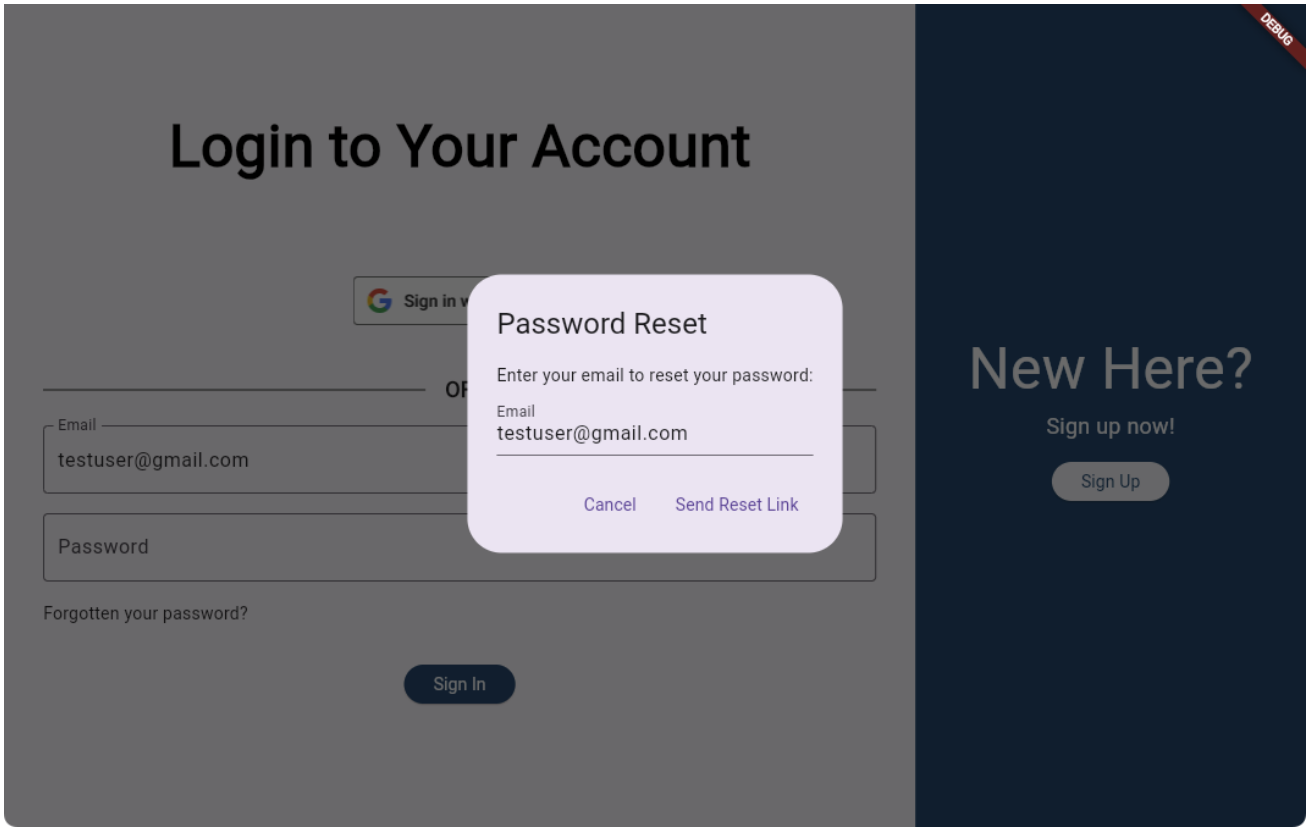

Figure 37 - Password Reset [UI]

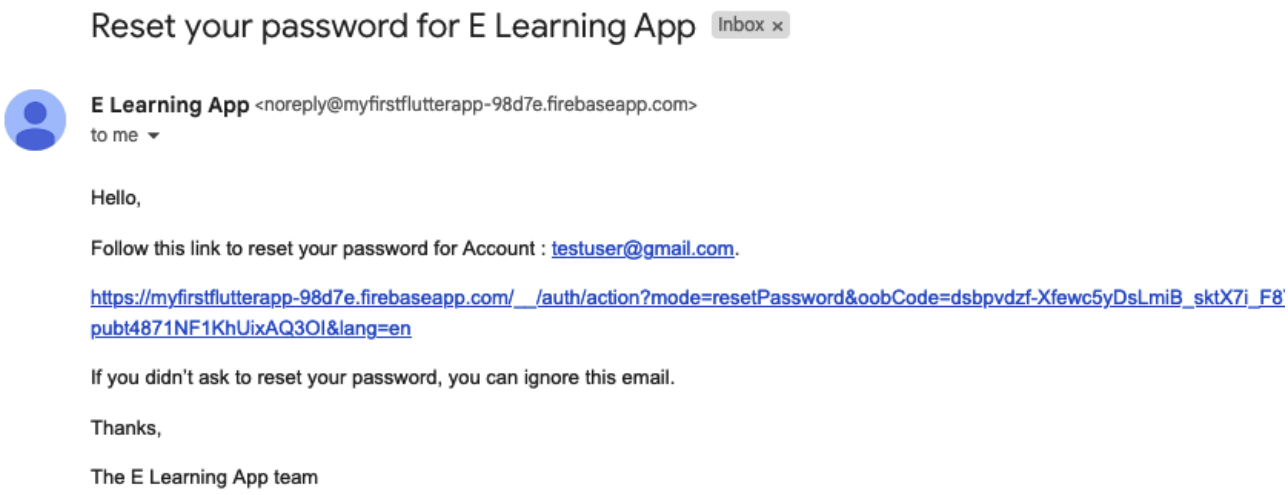

Figure 38 - Password Reset Email

The password reset will send an email like above to the original email for the account, only if said email address belongs to a user account.

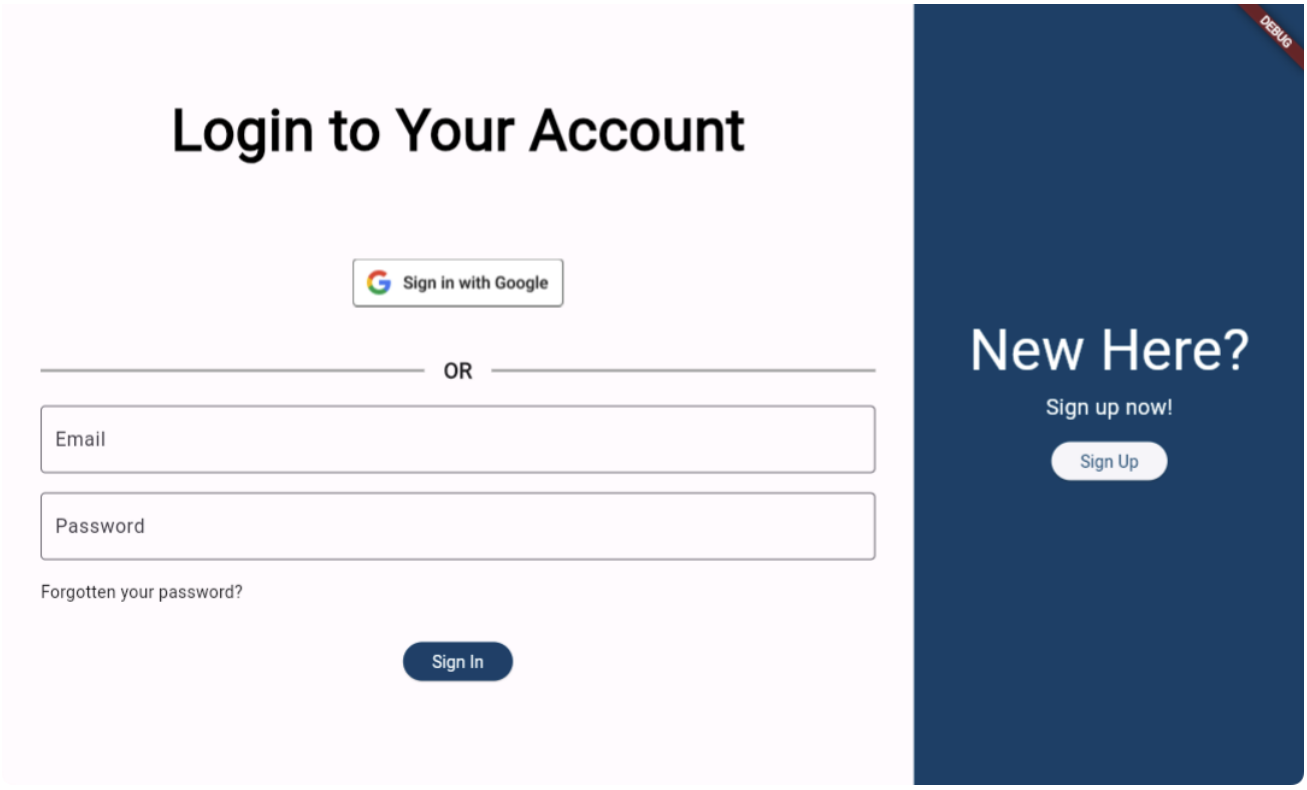

Figure 39 - Sign In [UI]

This is the final result for the login page, with password reset, sign in and Google sign in functionality working.

Account Creation

For users to sign in, they must already have an existing account - for this there is a separate page for a user to either sign up with Google or enter their name, email, and password to sign up. If the credentials are allowed and the user does not already exist, the account will be created with Firebase, and they will be sent to the home screen.

```
Future<bool> signUpWithEmailAndPassword(String email, String password, String name, String surname) async {
  try {
    UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    User? user = userCredential.user;
    if (user != null) {
      // Update user's display name
      await user.updateDisplayName("$name $surname");

    return true;
```

Figure 40 - Sign Up [Code]

Using a Firebase package to sign users up with email and password, they will automatically be given UID's and added to the user authentication panel.

```
ElevatedButton(
  onPressed: () async {
    bool success = await signUpWithEmailAndPassword(
      emailController.text,
      passwordController.text,
      nameController.text,
      surnameController.text,
    );
    if (success) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => HomeScreen()),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Sign up failed. Please try again.')),
      );
    }
  },
  style: ElevatedButton.styleFrom(
    backgroundColor: Color.fromARGB(255, 15, 75, 115),
    foregroundColor: Colors.white,
  ),
  child: Text('Sign Up'),
```

Figure 41 - Sign Up Button [Code]

The email, name, and password are pulled from the webpage and used to sign up the user then navigate to the home page.

```dart
Future<bool> signUpWithGoogle() async {
try {
  final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();

  if (googleUser == null) {
    // Google sign-in was canceled
    return false;
  }

  final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
  final AuthCredential credential = GoogleAuthProvider.credential(
    accessToken: googleAuth.accessToken,
    idToken: googleAuth.idToken,
  );

  await FirebaseAuth.instance.signInWithCredential(credential);
  return true; // Sign-up successful
} catch (e) {
  print('Error during Google sign-up: $e');
  return false; // Sign-up failed
}
}
```

Figure 42 - Google Sign Up [Code]

Firebase includes a method for signing up with Google as well as sign in, the functionality is almost identical, taking you to Googles own verification site. For users signing up with Google, their name must be taken from their Google account.

```dart
Center(
  child: Padding(
    padding: const EdgeInsets.all(0.0),
    child: IconButton(
      icon: Image.asset('/Users/hcampbell/Documents/Uni/Year3/FYProject/Flutter/my_first_flutter_app/web/icons/Google
      onPressed: () async {
        bool success = await signUpWithGoogle();
        if (success) {
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) => HomeScreen()),
          );
        } else {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content: Text('Google sign-up failed. Please try again.')),
          );
        }
      },
    ), // IconButton
  ), // Padding
), // Center
```

Figure 43 - Google Sign Up Button [Code]

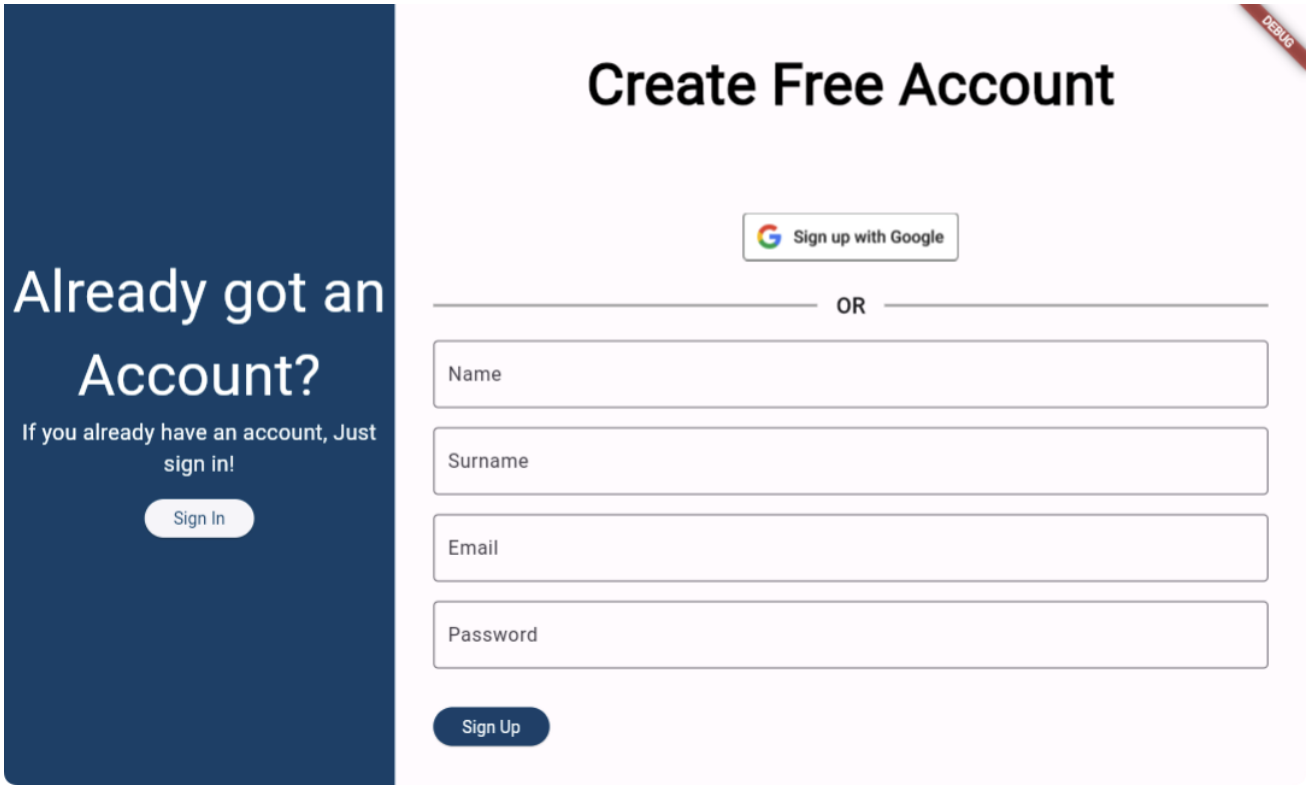

Figure 44 - Sign Up [UI]

The finished Sign-up account, with regular sign in and Google sign in options.

Firestore User Database

Firebase authentication assigns each user a unique ID and stores their email and password - it does not however store any other personal details such as name, age or date account was created. For this, we must implement a Firestore Cloud Database - Cloud Firestore is a NoSQL database built on Google infrastructure which allows collections of data such as users to be stored and served across various apps in real-time.

```dart
Future<bool> signUpWithEmailAndPassword(String email, String password, String name, String surname) async {
  try {
    UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    User? user = userCredential.user;
    if (user != null) {
      // Update user's display name
      await user.updateDisplayName("$name $surname");

      // Store user data in Firestore
      CollectionReference usersCollection = FirebaseFirestore.instance.collection('users');
      await usersCollection.doc(user.uid).set({
        'UID' : user.uid,
        'email': email,
        'Name': name,
        'Surname': surname,
        'DateCreated': FieldValue.serverTimestamp(),
      });
    }

    return true;
  } catch (e) {
    print('Error during sign up: $e');
    return false;
  }
}
```

Figure 45 - Save To Firestore [Code]

Creating a collection using the information provided by the user, this will also check they do not already have an account.

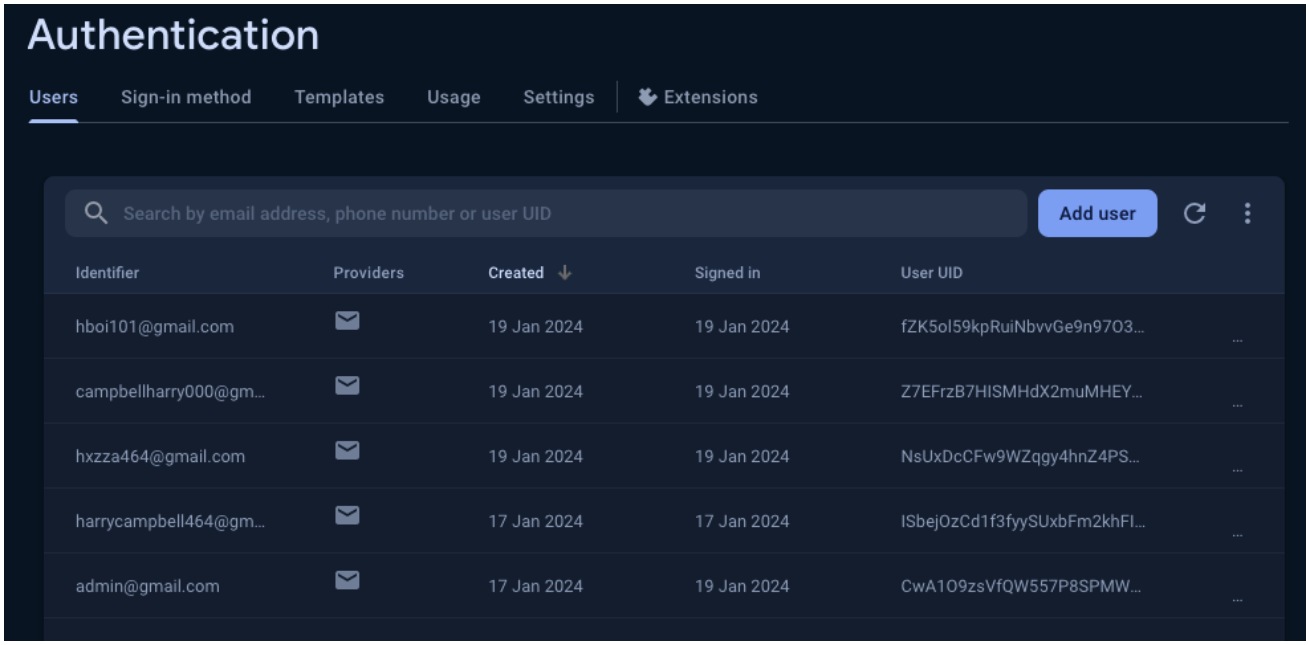

Figure 46 - Firestore User Panel

When user accounts are created, they consist of just a UID, Last Sign in, Creation date and Identifier. The Firestore collection 'Users' will store information which is used by the program itself such as name, courses etc. This collection can grow and expand to have more fields as the program grows in complexity.

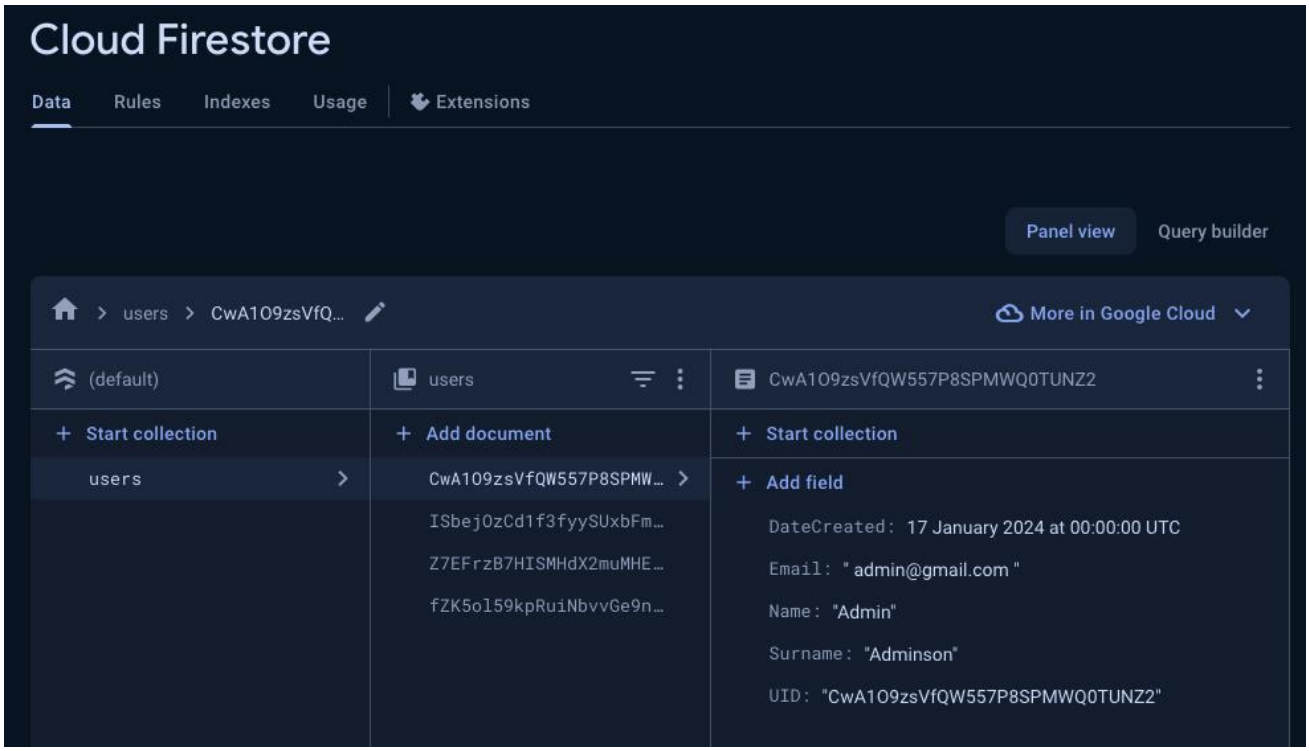

Figure 47 - Firestore Users Database

Navigation Pane

Once signed in, or signed up, users are met with a home page which will be able to take them to other pages of the platform to complete other tasks such as viewing courses, adding courses, and viewing their dashboard. In order to navigate these pages, a custom flutter widget is used. This will exist across all pages and allow swift navigation from page to page.

```
// Handle navigation based on the index here
switch (index) {
  case 0:
    //Navigator.push(context, MaterialPageRoute(builder: (context) => DashboardScreen()));
    Navigator.pushReplacement(context,MaterialPageRoute(builder: (context) => HomeScreen()));
    // Handle Dashboard item tap
  case 1:
    // Handle Search item tap s
    break;
  case 2:
    // Handle Course Home item tap
    // (already on home)
    break;
  case 3:
    // Handle Saved item tap
    break;
  case 4:
    // Handle Settings item tap
    break;
```

Figure 48 - Navigation Pane [Code]

Using a case statement to navigate between pages and highlight the current page. Creating the Appdrawer which will act as a navigation pane.

```
class AppDrawer extends StatelessWidget {
  final List<String> drawerItems = [
    'Dashboard',
    'Search',
    'Course Home',
    'Saved',
    'Settings',
  ];

  final List<IconData> drawerIcons = [
    Icons.dashboard,
    Icons.search,
    Icons.home,
    Icons.bookmark,
    Icons.settings,
  ];

  final int selectedIndex;
  final Function(int) onNavItemTap;
  final User? currentUser; // Add User object for the current user
```

Figure 49 - App Drawer [Code]

```dart
@override
Widget build(BuildContext context) {
  return Drawer(
    child: ListView(
      padding: EdgeInsets.zero,
      children: <Widget>[
        UserAccountsDrawerHeader(
          accountName: currentUser != null ? Text(currentUser!.displayName ?? 'Your Name') : Text('Your Name'),
          accountEmail: currentUser != null ? Text(currentUser!.email ?? 'your.email@example.com') : Text('your.email@example.com'),
          currentAccountPicture: CircleAvatar(
            backgroundColor: Colors.white,
            child: Icon(Icons.person),
          ), // CircleAvatar
        ), // UserAccountsDrawerHeader
        // Build your drawer items using 'drawerItems' list
        for (var index = 0; index < drawerItems.length; index++)
          ListTile(
            leading: Icon(drawerIcons[index]),
            title: Text(drawerItems[index]),
            selected: index == selectedIndex,
            onTap: () {
              onNavItemTap(index);
              Navigator.of(context).pop(); // Close the drawer
            },
          ), // ListTile
        Divider(), // Add a divider
        ListTile(
          leading: Icon(Icons.logout),
          title: Text('Logout'),
          onTap: () async {
            try {
              await FirebaseAuth.instance.signOut();
              Navigator.pushReplacementNamed(context, '/'); // Navigate to Home Screen
              // Redirect to the login screen or perform any other actions after logout
            } catch (e) {
              print('Error during logout: $e');
              // Handle any logout errors here
            }
          },
        ), // ListTile
      ], // <Widget>[]
    ), // ListView
  ); // Drawer
 }
}
```

Figure 50 - Navigation Widget [Code]

This code forms the navigation pane widget, which will be linked to from every page. The widget fetches ‘currentUser’ from Firebase on initialisation to show the name and email address of currently signed in user. It shows all the available page options such as dashboard, search, home, and settings. The navigation pane also allows users to sign out.

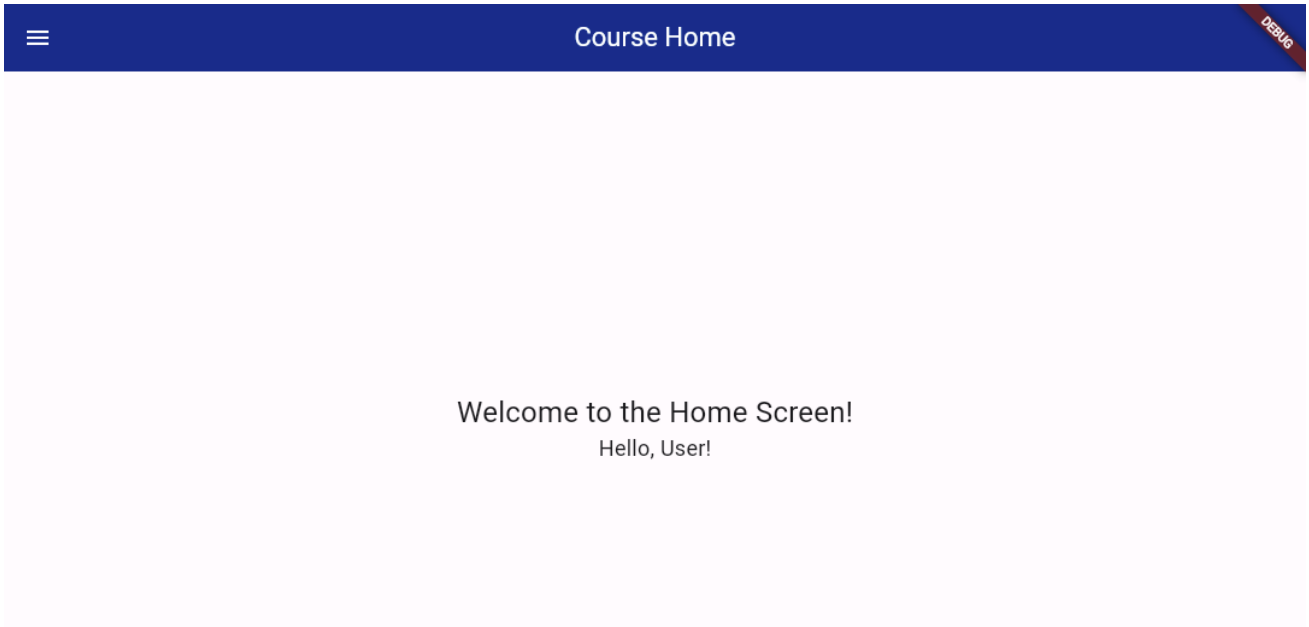

Figure 51 - Home Screen [UI]

The home screen will look like this, with a menu icon in the top right to open the navigation panel.

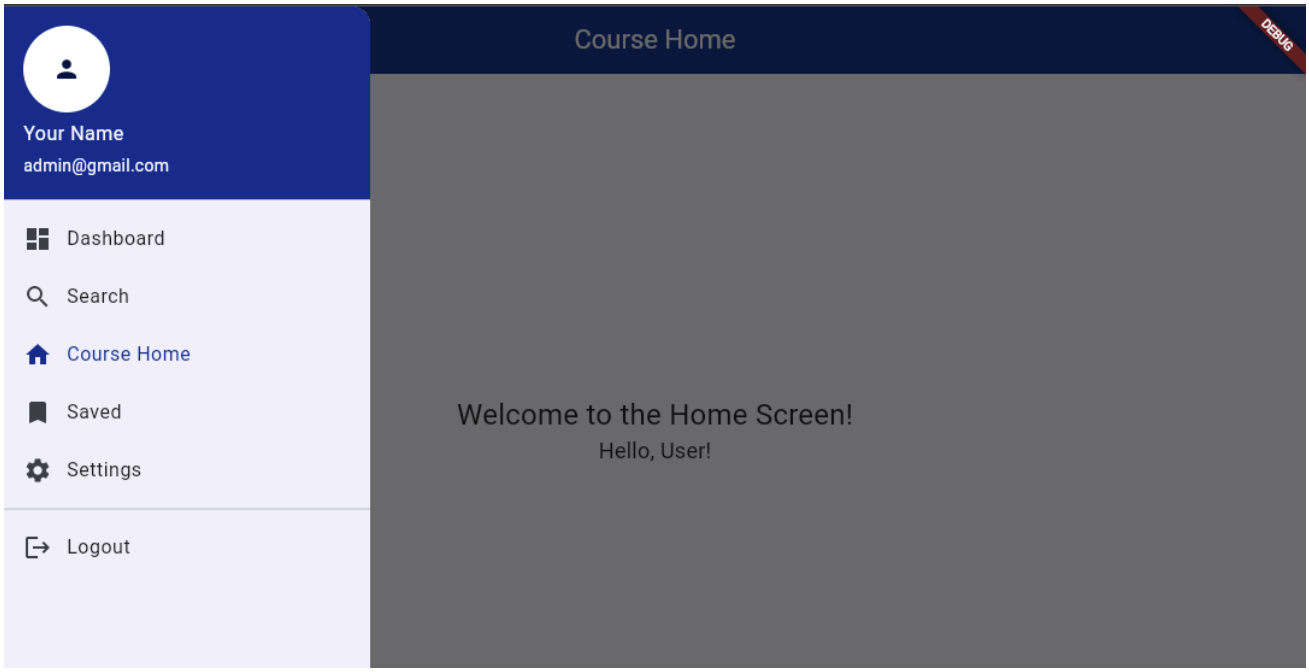

Figure 52 - Navigation Pane [UI]

When opened, the navigation panel shows all available page options, the current user, and their email address, as well as the option to sign out.

Account Customisation

Users' details can be change at any time. Allowing user's name and surname to be changed enhances the platform's usability and the ability to change password increases security.

```
Future<void> fetchUserDetails() async {
  final userDocRef = FirebaseFirestore.instance.collection('users').doc(_currentUser?.uid);

  final userDoc = await userDocRef.get();
  if (userDoc.exists) {
    final userData = userDoc.data() as Map<String, dynamic>;
    setState(() {
      _firstName = userData['Name'];
      _lastName = userData['Surname'];
    });
  }
}
```

Figure 53 - Firebase Fetch [Code]

Fetching the user's details from Firestore.

```
// Reference to the user's Firestore document
final userDocRef = FirebaseFirestore.instance.collection('users').doc(_currentUser?.uid);

// Create a map to update the Firestore document
final updatedData = {
  'Name': newFirstName,
  'Surname': newLastName
  // You can add other fields if needed
};

// Update the Firestore document
userDocRef.update(updatedData).then((_) {
  // Handle success
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content: Text('Changes saved successfully!'),
  )); // SnackBar
}).catchError((error) {
  // Handle error
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content: Text('Error saving changes: $error'),
  )); // SnackBar
});
```

Figure 54 - Firebase Update [Code]

Updating Firestore with the new name or surname.

```
  // Update the password if a new password is provided
  if (newPassword.isNotEmpty) {
    _currentUser?.updatePassword(newPassword).then((_) {
      // Handle password update success
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Password updated successfully!'),
      )); // SnackBar
    }).catchError((error) {
      // Handle password update error
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Error updating password: $error'),
      )); // SnackBar
    });
  }
}
```

Figure 55 - Firebase Update Password [Code]

Updating the password using a Firebase method which is secure.

```
    if (confirmDelete != true) return; // If user cancels, do nothing

    try {
      // Delete Firestore user data
      await userDocRef.delete();

      // Delete the user's account
      await _currentUser?.delete();
      // Redirect to a safe screen, e.g., login screen
      Navigator.pushReplacementNamed(context, '/');
    } catch (error) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Error deleting account: $error'),
      )); // SnackBar
    }
  }
```

Figure 56 - Firebase Delete [Code]

Deleting a user and all relating documents.

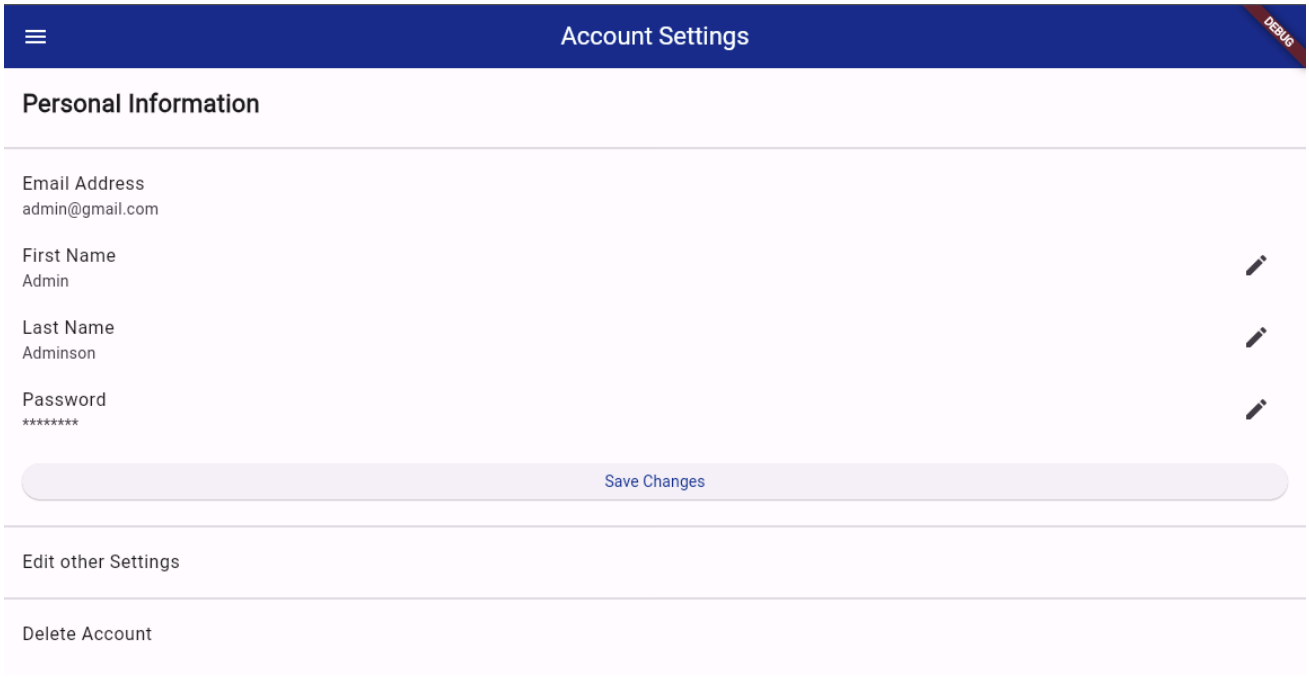

Figure 57 - Edit Information [UI]

View User details, click the pencil icon to make any edits.

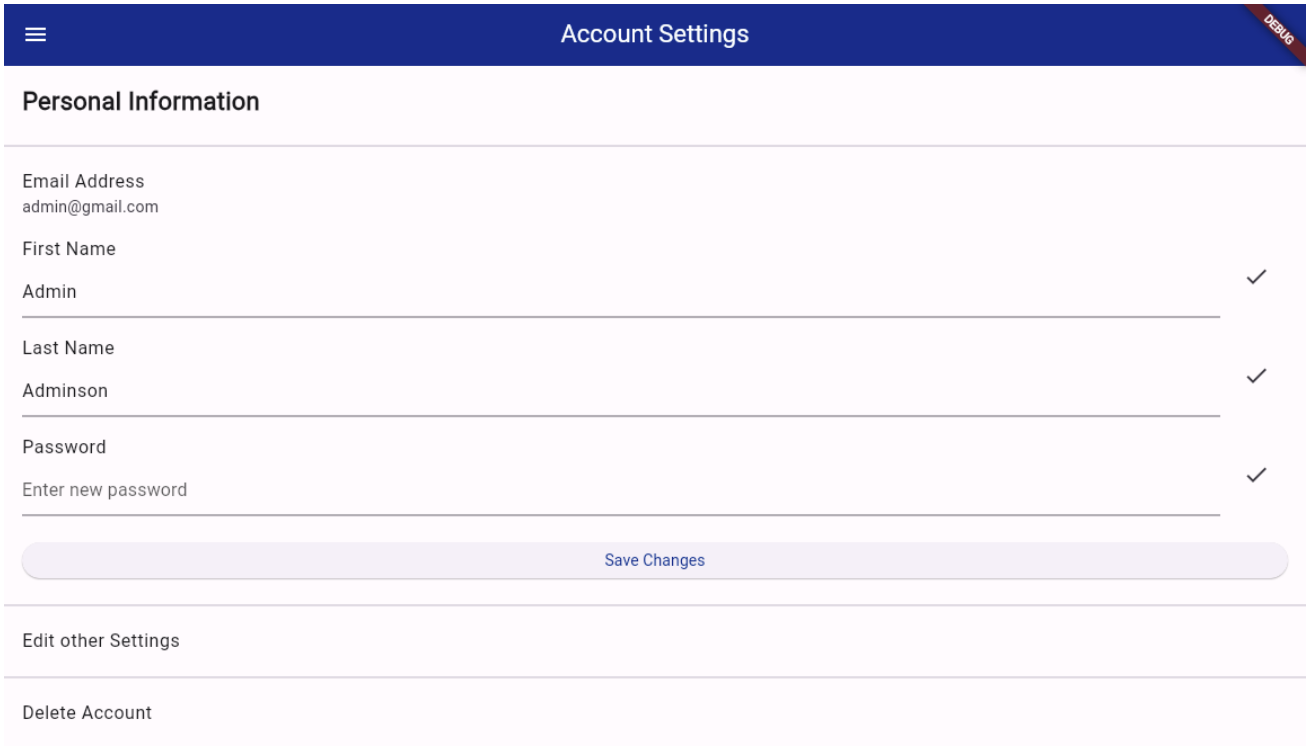

Figure 58 - Edit Information [UI]

From here the name and password can be edited and then changes saved.

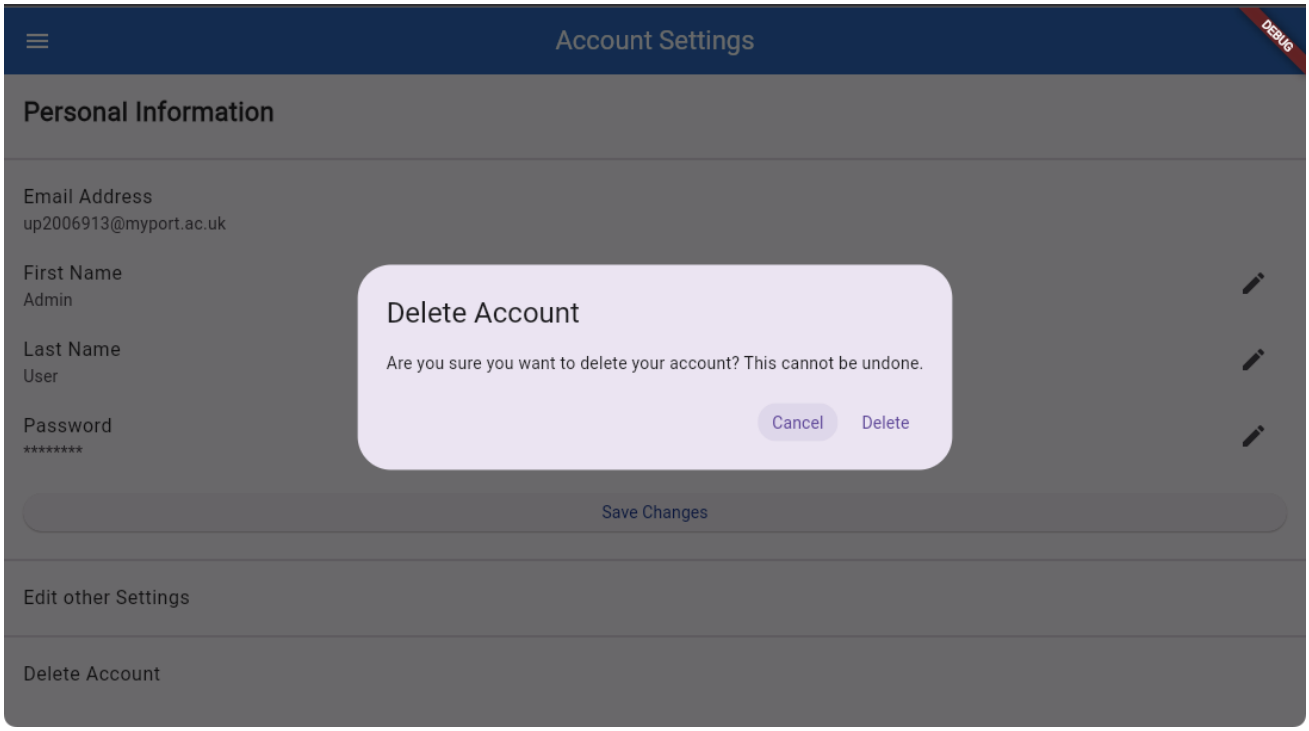

Figure 59 - Delete Account [UI]

Deleting a user account requires an additional confirmation, this ensures that the risk of accidental deletion is as low as possible. This will become especially relevant later on in development as a user's account can hold progress values for courses.

6.1.5 Testing

Feature	Test Description	Result
Login Authentication	Login credentials should provide access to the home page.	Success
	Only valid login credentials should provide access.	Success
	“Snackbar” should prompt users with any error messages.	Success
Google Sign-In	Sign in using a Google account.	Fail
	Sign up using a Google account.	Fail
	Sign up with Google account should provide a field for first name and surname.	Fail
Account Creation	Accounts created should be added to user list and given access to home page.	Success
	Users with existing accounts should not be able to create new ones.	Success
	Users should be set up with name, surname, email, password, and creation time.	Success
Cloud Firestore Integration	New users should be generated in a document within the users’ collection.	Success
	Users store with name, surname, email, password, and creation time.	Success
	User data such as name should be able to be extracted from Firestore.	Success
Navigation Pane	Navigation pane should be hidden unless clicked.	Success
	Navigation pane should hide when new page option is clicked.	Success

	Navigation pane should allow users to navigate between all pages independently	Success
Account Customisation	Account details should be pulled from Firestore	Success
	Account details should be updated in Firestore when edited	Success
	Accounts should be removed from Firestore when deleted	Success

Table 5 - Iteration 1 Testing

Fixing Google Authentication

Google sign in / authentication threw several errors through testing, mostly due to strict security rules on unpublished software. To prevent any further errors, the Google People API had to be enabled and my URL localhost: had to be added to the projects' domain.

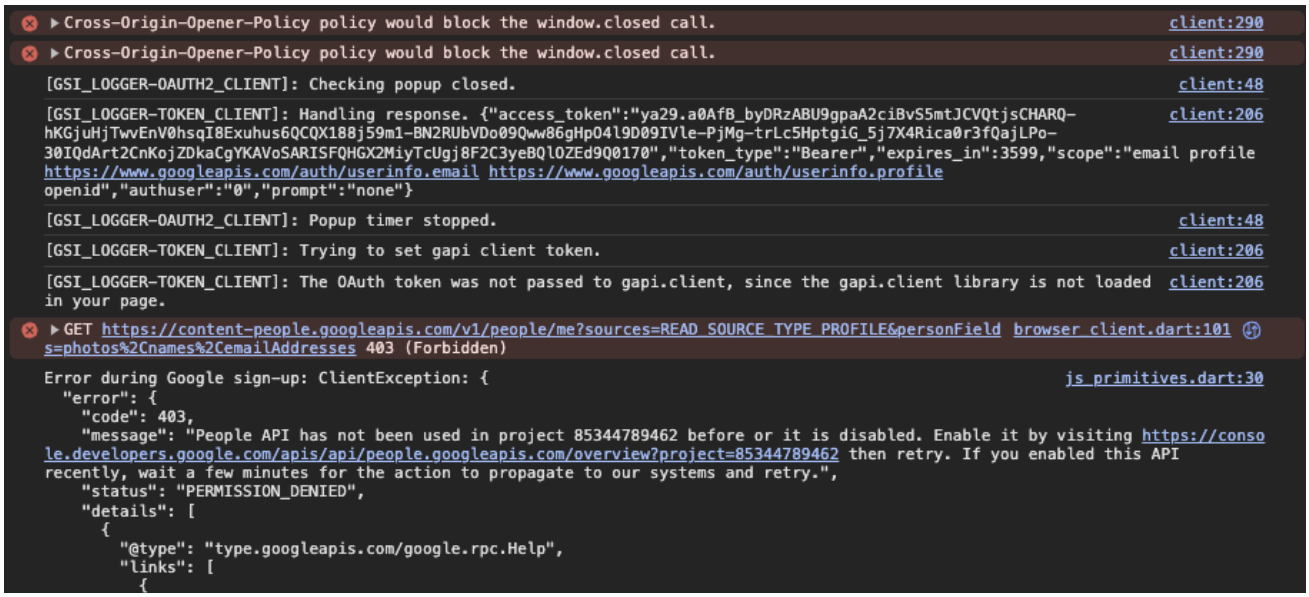

Figure 60 - API Error Message

Google Chrome's debugger suggests the error is due to a failed GET, likely being blocked by Google since the People API is not enabled.

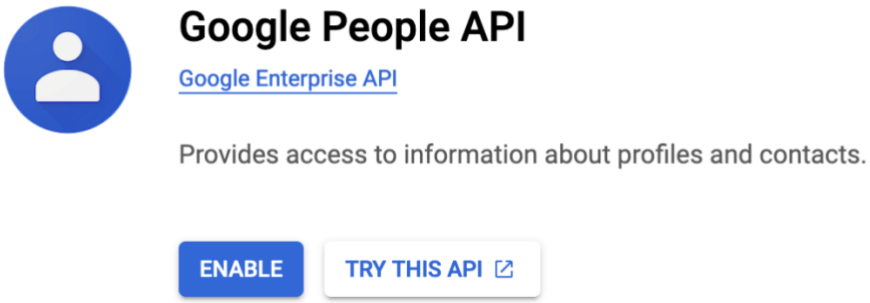

Figure 61 - Google People API

Enable the API from the Google developer dashboard.

```dart
Future<bool> signInWithGoogle() async {
  try {
    final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();
    if (googleUser == null) {
      // check the user is not already signed in
      return false;
    }
    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
    final AuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    final UserCredential authResult = await FirebaseAuth.instance.signInWithCredential(credential);
    final User? user = authResult.user;

    if (user != null) {
    updateUserLastLogin(user); // Update last login timestamp
  }
    return true; // Sign-in successful
  } catch (e) {
    print('Error during Google sign in: $e');
    return false; // Sign-in failed
  }
}
```

Figure 62 - Updated Sign In [Code]

Update the ''signInWithGoogle code section to utilise the API and Firebase methods correctly, this now allows users to sign in and sign up effortlessly using their Google accounts.

6.1.6 Review

This first iteration focussed on establishing core functionalities essential for users' interaction and navigation within the app. This included the implementation of login authentication, account creation, Google OAuth 2.0 authentication and a navigation pane. Establishing the core functionalities necessary for effective user interaction and navigation sets a solid foundation for the e-learning platform's future development.

6.2 Iteration 2

6.2.1 Features to Implement

- User account roles (Teacher, Student, Admin) (FR1,FR4) (NFR2)
- Access management (FR1) (NFR2)
- User Role management (FR4) (NFR2)
- Course Creation (FR6)
    - Creating curriculum (video/text).
    - Creating Quizzes.
    - Publishing Course.
    - Course Home with tiles and Course Enrol button.
- Course Management (FR7)
- Database System (FR8)

6.2.2 Task Breakdown

The feature implementation has been broken down into actionable tasks to be used throughout the development process.

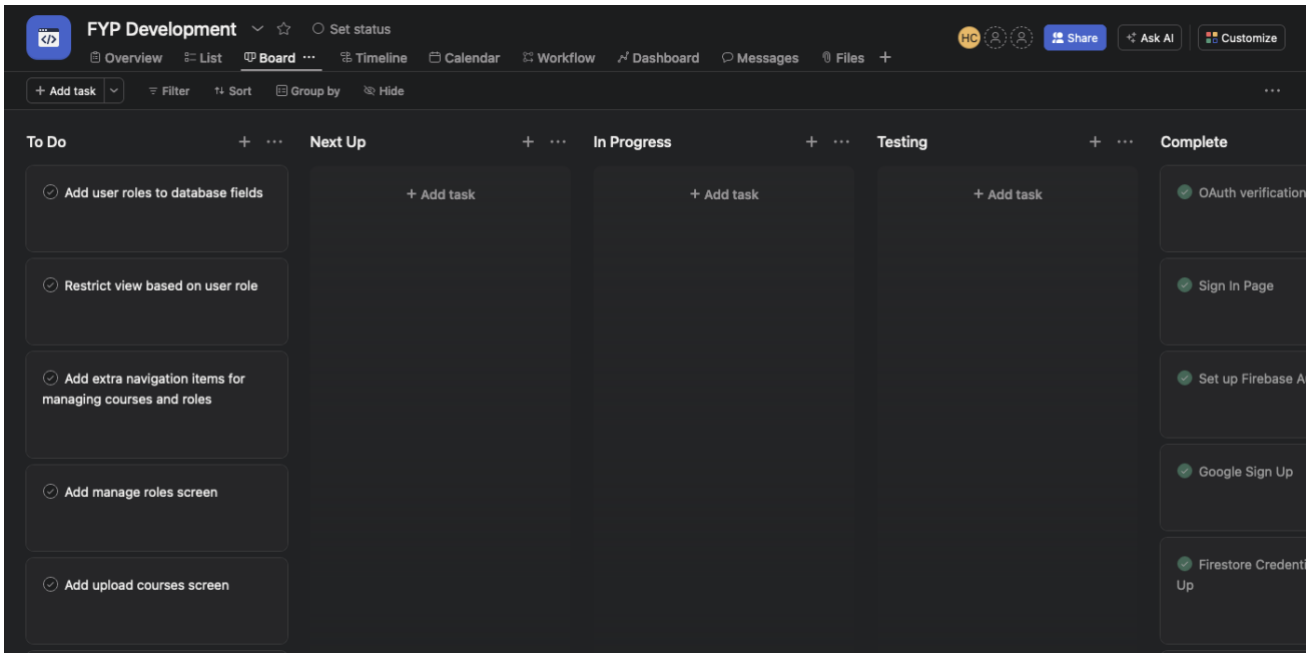

Figure 63 - Iteration 2 Task Breakdown

6.2.3 Design

The design of these features will incorporate features from Figures 21, 25, 26, 27, 28 and 29 in the design section, for the course creation screens. These features will require updates to the users collection and a new courses and curriculum/quizzes collection in Firestore.

6.2.4 Implementation

User Roles

For the next implementation, it is important that users are defined with specific roles: 'Admin', 'Teacher', and 'Student' - these roles are stored in each users' database collection and dictate what they are able to do with the system and ensuring that only users such as Admins have access to critical information and functionality such as deleting accounts or courses.

Figure 64 - Firebase User Collection

The 'userRole' field has been added to the Users collection.

User Access Rights (RBAC Implementation)

Once the users are given roles, the system can be adjusted to show only information which is necessary for each user. In this instance, the navigation menu options for 'upload course' is saved for teachers and administrators, and 'manage roles' is only accessible to administrators.

```dart
Future<String> getUserRole(User? user) async {
  if (user == null) return 'Guest'; // Return 'Guest' if not signed in

  final doc = await FirebaseFirestore.instance.collection('users').doc(user.uid).get();
  if (!doc.exists) return 'Student'; // Default to 'Student' if role not set
  print(doc.data()?['UserRole'] ?? 'Student');
  return doc.data()?['UserRole'] ?? 'Student';
}
```

Figure 65 - Get User Role [Code]

Fetching the 'userRole' field from Firebase.

```dart
if (userRole == 'Teacher') {
  menuItems.add(
    ListTile(
      leading: Icon(Icons.upload_file),
      title: Text('Course Upload'),
      onTap: () {
        // Add navigation logic for Course Upload
      },
    ), // ListTile
  );
}

if (userRole == 'Admin') {
  menuItems.add(
    ListTile(
      leading: Icon(Icons.upload_file),
      title: Text('Course Upload'),
      onTap: () {
        // Add navigation logic for Course Upload
      },
    ), // ListTile
  );
  menuItems.add(
    ListTile(
      leading: Icon(Icons.manage_accounts),
      title: Text('Manage Roles'),
      onTap: () {
        Navigator.pushReplacementNamed(context, '/manage_roles');
      },
    ), // ListTile
  );
}
```

Figure 66 - Hide Admin Options [Code]

Once the ‘userRole’ value is fetched, depending on whether the user is a student, teacher, or admin they should have different access rights. Here the widget for the navigation panel is adapted to hide the course upload and manage roles pages from students. And the manage roles page is hidden from teachers.

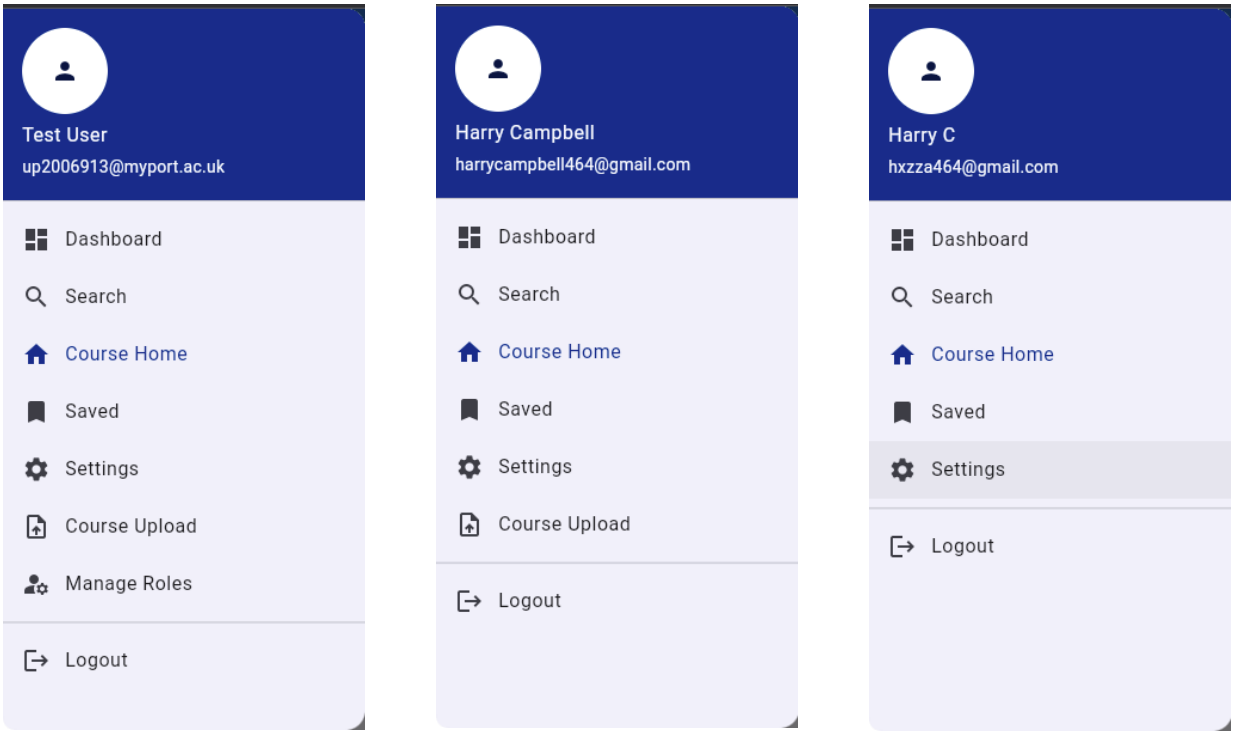

Figure 67 - Hidden Menu Options [UI]

The result is a successful implementation of role-based access control, this simple mechanism protects the platform from malicious users attempting to delete courses or even users.

User Role Management

In order to manage roles within the system, the administrator accounts are given access to a section titled manage roles. In this section they can choose who is assigned to the roles of 'Student', 'Teacher' or 'Admin'. This also gives the ability for admins or teachers to be demoted when access is no longer needed.

```
Future<String> getUserRole(User? user) async {
  if (user == null) return 'Guest'; // Return 'Guest' if not signed in

  final doc = await FirebaseFirestore.instance.collection('users').doc(user.uid).get();
  if (!doc.exists) return 'Student'; // Default to 'Student' if role not set
  print(doc.data()?['UserRole'] ?? 'Student');
  return doc.data()?['UserRole'] ?? 'Student';
}

void updateUserRole(String userId, String newRole) {
  FirebaseFirestore.instance.collection('users').doc(userId).update({
    'UserRole': newRole,
  }).then((_) {
    print('User role updated');
  }).catchError((error) {
    print('Error updating user role: $error');
  });
```

Figure 68 - Get User Roles [Code]

Fetching user roles from Firebase and displaying users and their roles on the manage roles screen. From here any user role can be edited.

```
return ListView.builder(
  itemCount: users.length,
  itemBuilder: (context, index) {
    var user = users[index];
    String userRole = user['UserRole'] ?? 'Student';
    String userEmail = user['email'] ?? 'No Email';

    return ListTile(
      title: Text(userEmail),
      subtitle: Text('UID: ${user.id}'),
      trailing: Row(
        mainAxisSize: MainAxisSize.min,
        children: [
          DropdownButton<String>(
            value: userRole,
            items: <String>['Student', 'Teacher', 'Admin'].map((String value) {
              return DropdownMenuItem<String>(
                value: value,
                child: Text(value),
              ); // DropdownMenuItem
            }).toList(),
            onChanged: (String? newValue) {
              if (newValue != null) {
                updateUserRole(user.id, newValue);
              }
```

Figure 69 - Display User Roles [Code]

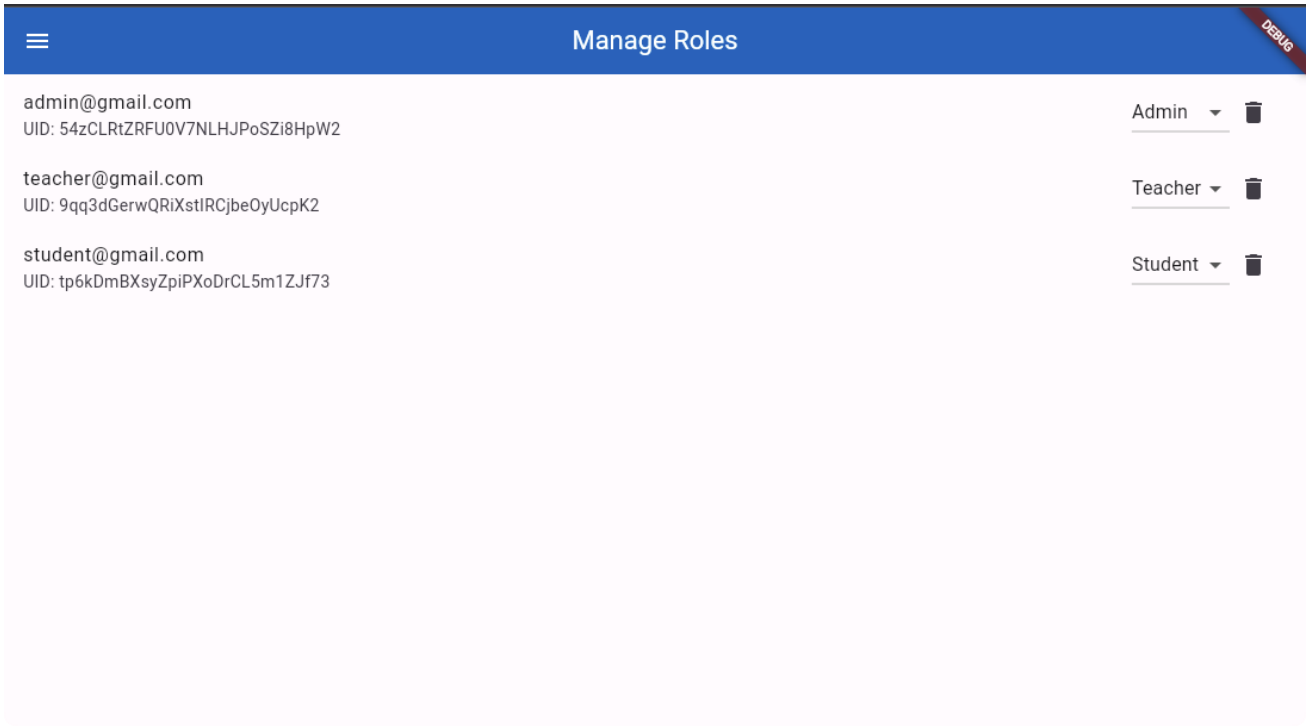

Figure 70 - Manage Roles Screen [UI]

The users are all listed, with their email address, UID, and role. Clicking on the role opens a dropdown from which roles can be altered, users can even be deleted by clicking on the bin icon next to them.

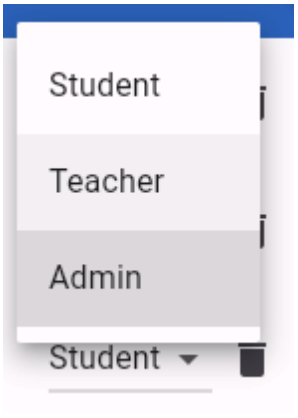

Figure 71 - Roles [UI]

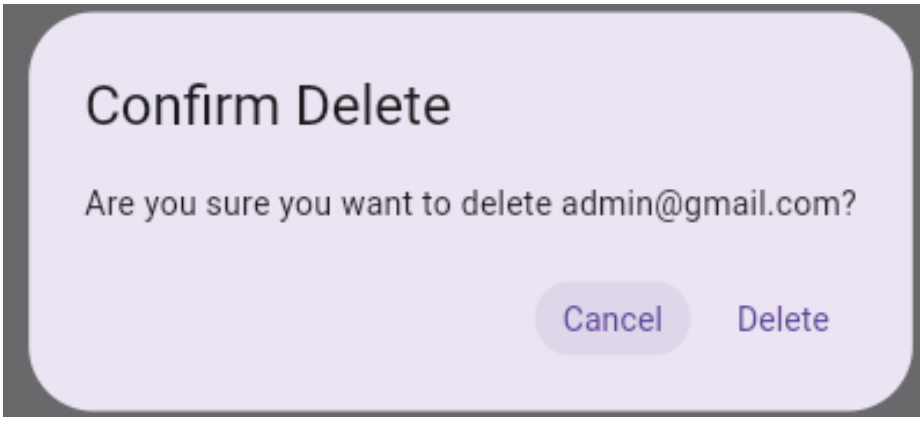

Figure 72 - Delete User [UI]

In order to delete a user, the same as if they were to delete themselves it requires a second confirmation. This protects against accidental account deletion.

Course Creation

The next feature to implement is course creation, a large aspect of the application itself, only available to Admins and Teachers through the 'Upload Course' menu item. The page will allow for a course to have basic details added, with a widget dynamically updating to show these changes how a student will see them. Then the add curriculum button will open a new page which breaks down the curriculum into sections containing text, video lessons and quizzes.

```
Future<void> _saveCourse() async {
  if (_formKey.currentState!.validate()) {
    final courseData = {
      'Category': _courseCategoryController.text,
      'Creator': _currentUser?.email ?? '',
      'Duration': int.tryParse(_courseDurationController.text) ?? 0,
      'ImageUrl': imageUrl,
      'Title': _courseTitleController.text,
      'Rating': 0,
    };

    final courseDoc = await FirebaseFirestore.instance.collection('courses').add(courseData);
    final courseID = courseDoc.id;

    await courseDoc.update({'CourseID': courseID});

    _courseTitleController.clear();
    _courseDescriptionController.clear();
    _courseCategoryController.clear();
    _courseDurationController.clear();

    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => CurriculumCreationScreen(courseID: courseID)),
    );
  }
```

Figure 73 - Saving Course [Code]

From the course creation screen, course details are uploaded to a Firebase collection called 'Courses', this collection contains many courses with unique ID's.

```dart
// Function to refresh the widget preview
void refreshPreview() {
  setState(() {
    previewTitle = _courseTitleController.text;
    previewDuration = _courseDurationController.text;
    previewRating = 0.0; // Initialize with a default rating of 0
    previewImageUrl = imageUrl;
    showPreview = true; // Show the preview after refreshing
  });
}
```

Figure 74 - Widget Preview [Code]

```dart
Future<void> _pickImage() async {
  try {
    final FilePickerResult? pickedFile = await FilePicker.platform.pickFiles(
      type: FileType.image,
      allowMultiple: false,
    );

    if (pickedFile != null && pickedFile.files.isNotEmpty) {
      final PlatformFile file = pickedFile.files.first;
      final Uint8List bytes = file.bytes!;

      final Reference storageRef = FirebaseStorage.instance.ref().child('CourseImages/${DateTime.now()}.jpg');
      final UploadTask uploadTask = storageRef.putData(bytes);

      uploadTask.snapshotEvents.listen((TaskSnapshot snapshot) {
        if (snapshot.state == TaskState.success) {
          // Image uploaded successfully, get the download URL
          storageRef.getDownloadURL().then((downloadURL) {
            setState(() {
              imageUrl = downloadURL;
            });
          });
        }
      });
    }
  } catch (e) {
    print("Error picking image: $e");
  }
}
```

Figure 75 - Image Picker [Code]

Using an image picker to allow teachers to upload a thumbnail for the course tile widget.

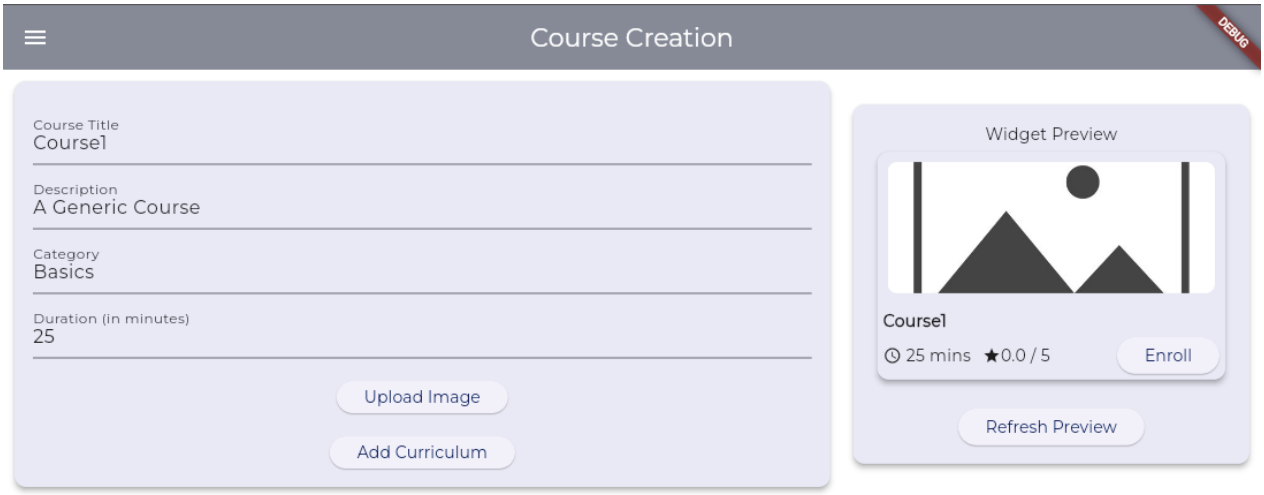

Figure 76 - Create Course Screen [UI]

The course creation page result, allowing for a basic course to be created with a given title, description, category, duration, and image.

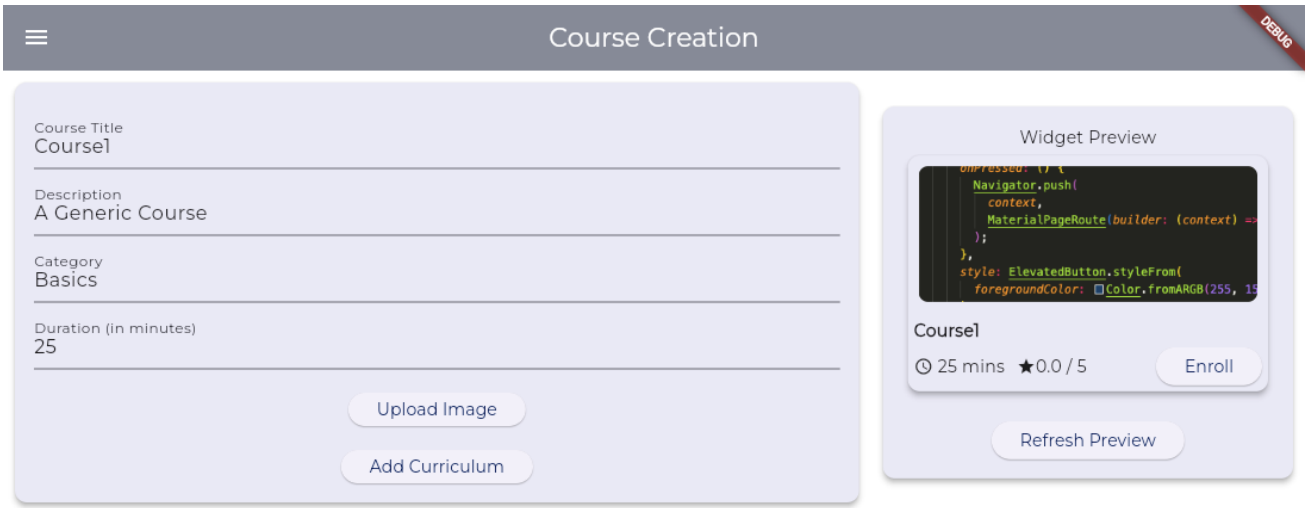

Figure 77 - Widget Refresh [UI]

Refreshing the preview will show exactly what the course tile looks like on the home page where students see it.

Curriculum Creation

Once a user has created a skeleton course with necessary basic details such as title, duration, category, and description, they are redirected to the 'course builder' page which allows for multiple text, video lessons and quizzes to be added to a course.

```
void _saveLessonToFirestore(Lesson lesson) {
  curriculumCollection().add({
    'Content': lesson.content,
    'Creator': _currentUser?.email ?? '',
    'Description': lesson.description,
    'LessonType': lesson.type,
    'Section': (_selectedSectionIndex + 1).toString(),
    'Title': lesson.title,
  });
}
```

Figure 78 - Saving Lesson [Code]

Saving a lesson object as a subcollection within a course, lessons have a few basic attributes such as content, creator, description, section, title, and lesson type.

```
void _updateLessonInFirestore(Lesson lesson) {
  curriculumCollection()
      .where('Section', isEqualTo: (_selectedSectionIndex + 1).toString())
      .where('Title', isEqualTo: _editingLesson!.title)
      .get()
      .then((querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      doc.reference.update({
        'Content': lesson.content,
        'Description': lesson.description,
        'LessonType': lesson.type,
        'Title': lesson.title,
      });
    });
  });
}
```

Figure 79 - Updating Lesson [Code]

Updating fields of the lesson when changed.

```dart
void _deleteLesson(int sectionIndex, int lessonIndex) {
  final lessonToDelete = sectionLessons[sectionIndex][lessonIndex];

  setState(() {
    sectionLessons[sectionIndex].removeAt(lessonIndex);
  });

  curriculumCollection()
      .where('Section', isEqualTo: (_selectedSectionIndex + 1).toString())
      .where('Title', isEqualTo: lessonToDelete.title)
      .get()
      .then((querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      doc.reference.delete();
    });
  });
}
```

Figure 80 - Deleting Lesson [Code]

Deleting a lesson.

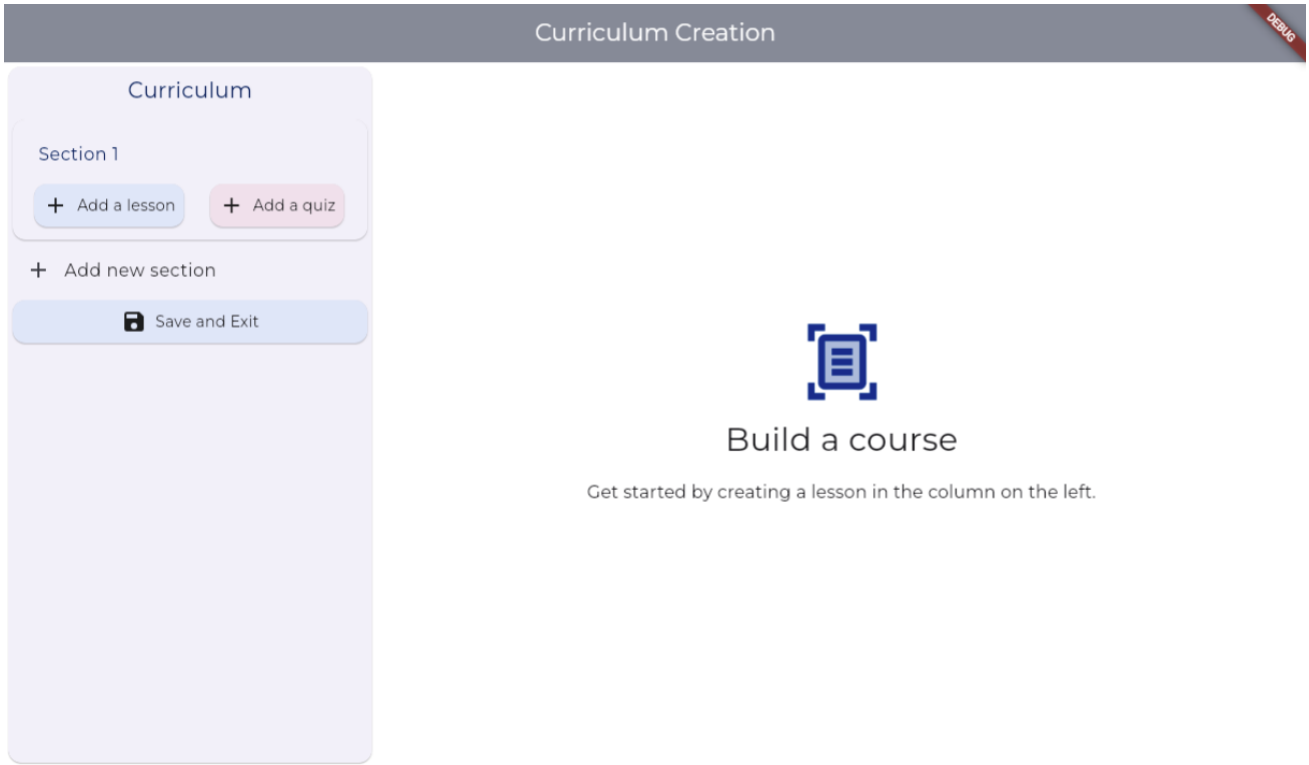

Figure 81 - Curriculum Creation Screen [UI]

This is the view when you first add curriculum, the column on the left holds all of the lessons, adding a lesson will open up a card on the right with text entry fields.

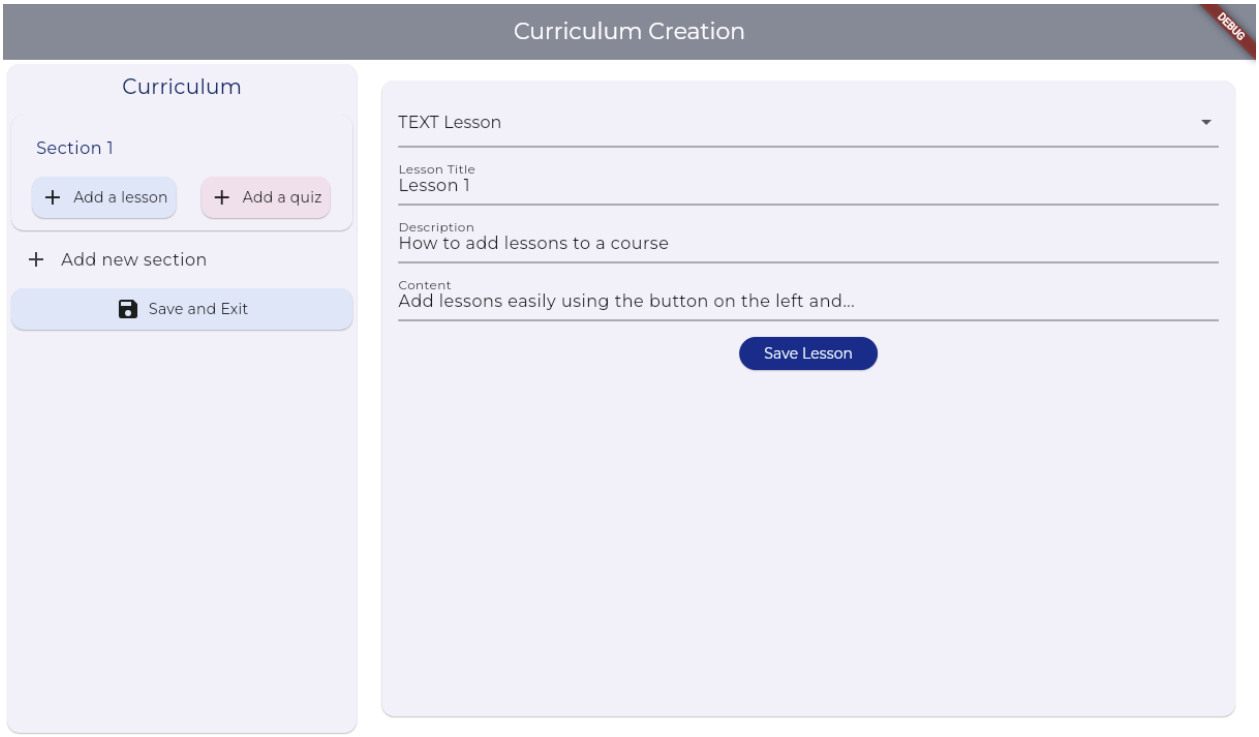

Figure 82 - Adding Lesson [UI]

The labels in the entry fields will change based on the type of lesson the user selects, since the content will either be text or a YouTube URL.

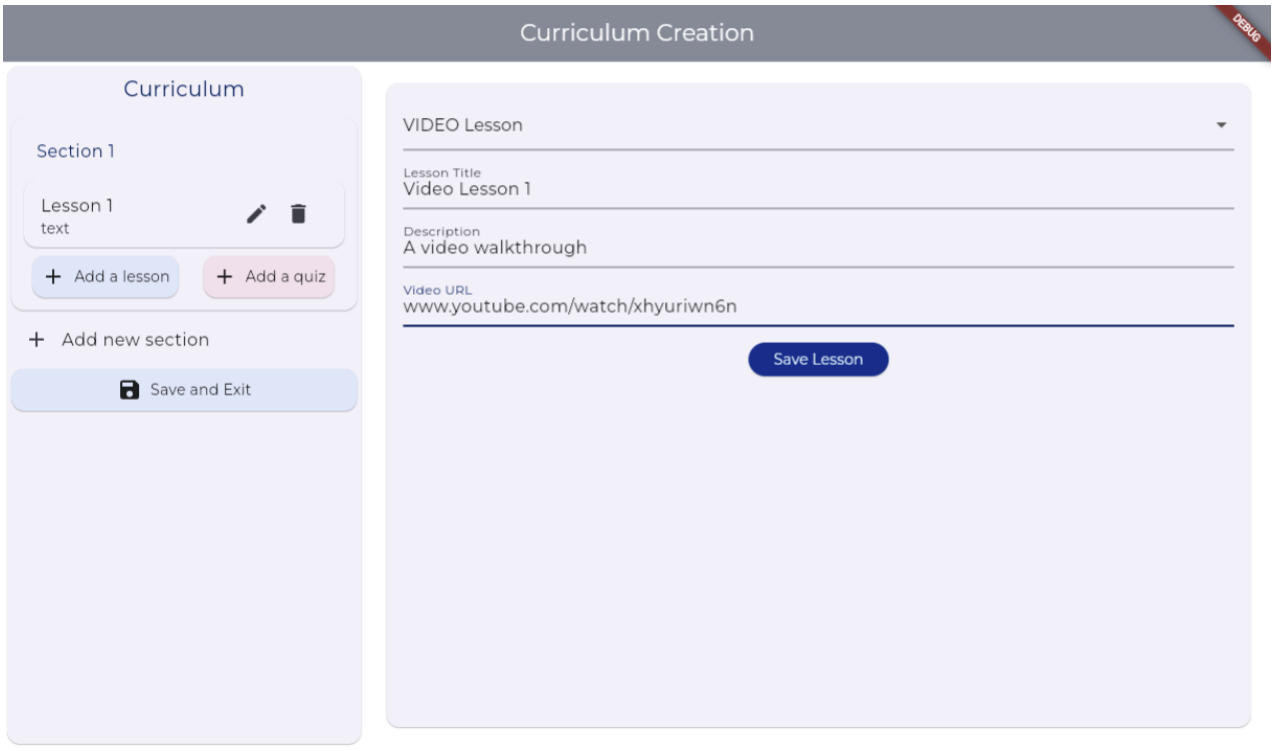

Figure 83 - Adding Video [UI]

```
void _addQuizToSection() {
  if (_quizTitleController.text.isNotEmpty && quizQuestions.isNotEmpty) {
    List<QuizQuestion> tempQuizQuestions = List.from(quizQuestions);
    final quiz = Quiz(
      title: _quizTitleController.text,
      description: _quizDescriptionController.text,
      questions: tempQuizQuestions,
    );
    setState(() {
      sectionQuizzes[_selectedSectionIndex].add(quiz);
      _isAddingQuiz = false;
      _clearQuizForm();
    });

    _saveQuizToFirestore(quiz);
  }
}
```

Figure 84 - Add Quiz [Code]

This code allows a quiz to be added in line with the lessons in each section.

```
void _saveQuizToFirestore(Quiz quiz) {
  print(quiz);
  quizCollection().add({
    'Title': quiz.title,
    'Description': quiz.description,
    'Questions': quiz.questions.map((q) => q.toJson()).toList(),
    'Creator': _currentUser?.email ?? '',
    'Section': (_selectedSectionIndex + 1).toString(),
  }).then((docRef) {
    print("Quiz saved successfully");
  }).catchError((error) {
    print("Error saving quiz: $error");
  });
}

void _editQuiz(int sectionIndex, int quizIndex) {
  final quizToEdit = sectionQuizzes[sectionIndex][quizIndex];

  setState(() {
    _isAddingQuiz = true;
    _quizTitleController.text = quizToEdit.title;
    _quizDescriptionController.text = quizToEdit.description;
    quizQuestions = List.from(quizToEdit.questions);
    sectionQuizzes[sectionIndex].removeAt(quizIndex);
  });
}
```

Figure 85 - Save Quiz [Code]

Saving a quiz to Firestore and updating the quiz.

```
void _deleteQuiz(int sectionIndex, int quizIndex) {
  final quizToDelete = sectionQuizzes[sectionIndex][quizIndex];

  setState(() {
    sectionQuizzes[sectionIndex].removeAt(quizIndex);
  });

  quizCollection()
      .where('Section', isEqualTo: (_selectedSectionIndex + 1).toString())
      .where('Title', isEqualTo: quizToDelete.title)
      .get()
      .then((querySnapshot) {
    querySnapshot.docs.forEach((doc) {
      doc.reference.delete();
    });
  });
}
```

Figure 86 - Delete Quiz [Code]

Deleting a quiz.

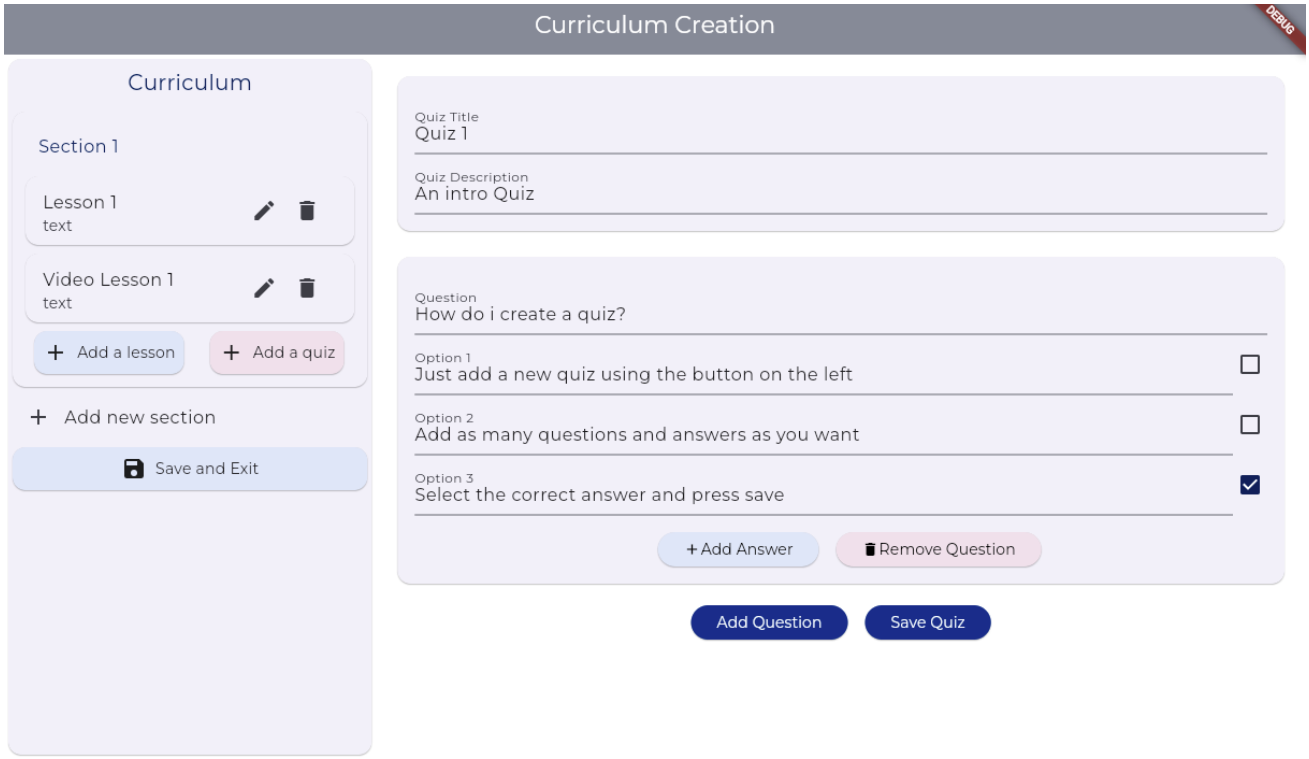

Figure 87 - Quiz Creation [UI]

This is the UI result for creating a quiz, the user must choose a title and description for the whole quiz, then they can add as many questions as they like which can each have as many answers as they like.

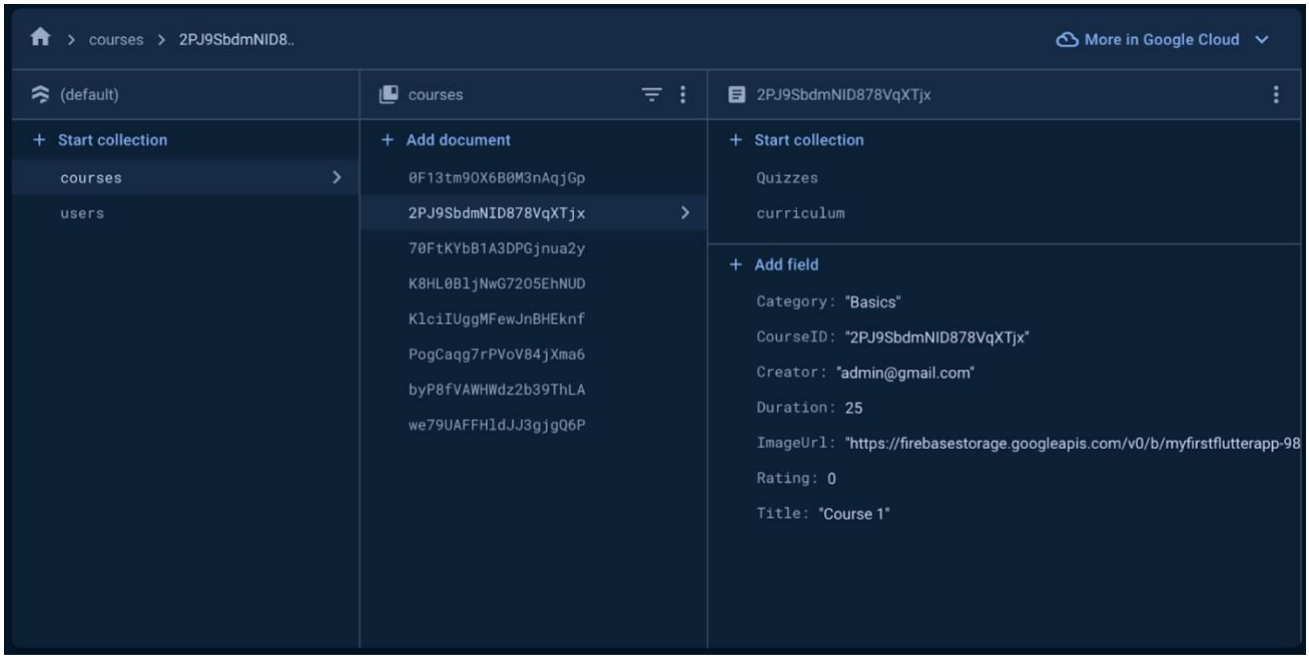

Figure 88 - Firebase Course Collection

Firestore showing Quiz and Curriculum linked to a course.

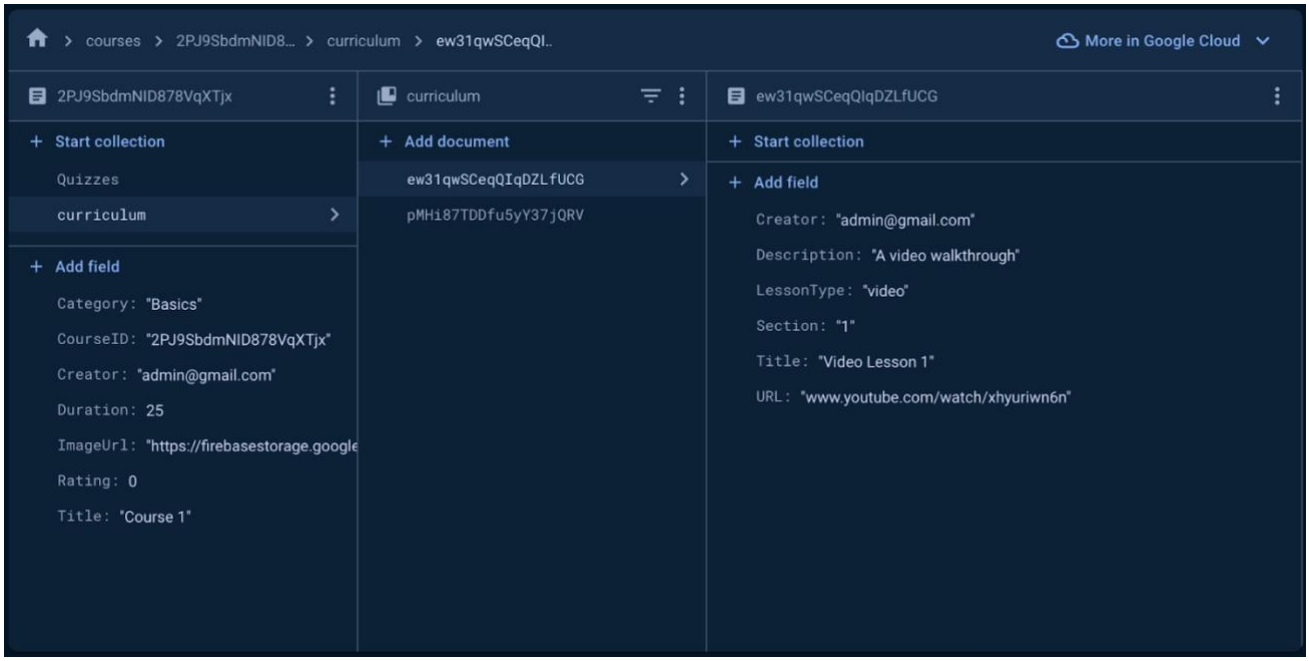

Figure 89 - Firebase Curriculum Documents

Curriculum contains the standard lessons, text, and video.

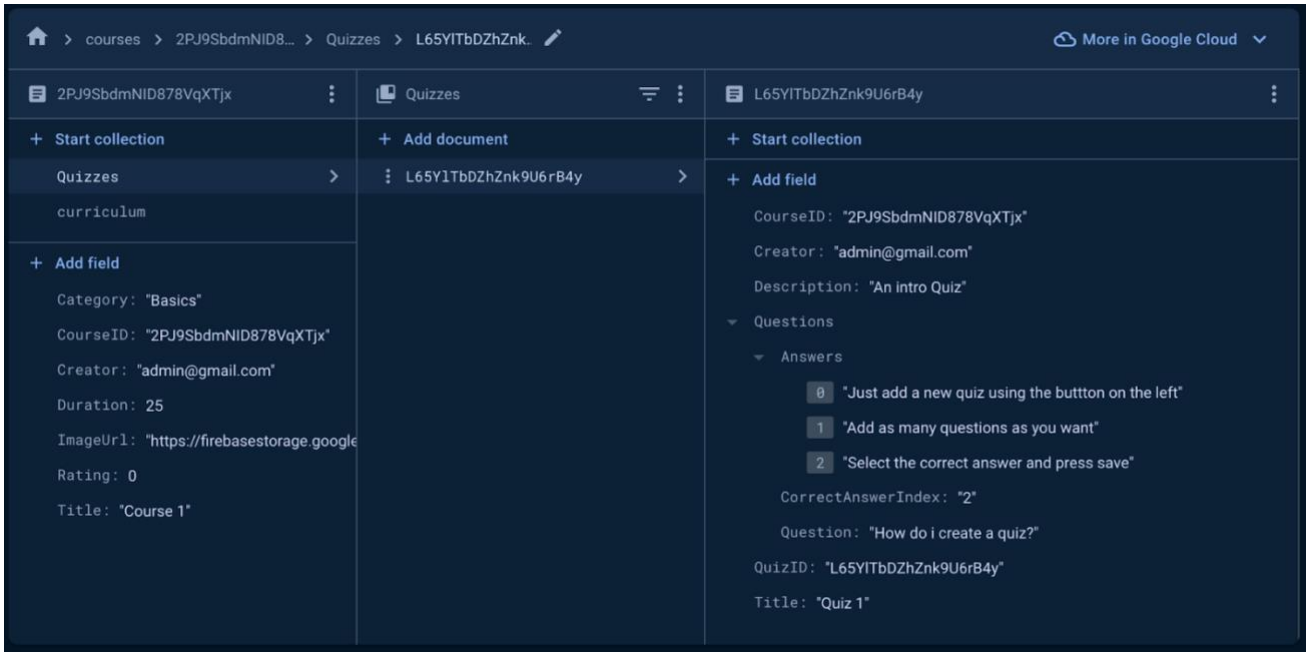

Figure 90 - Firebase Quiz Documents

'Quizzes' stores all the quizzes, using an array map of questions with an array of answers for each question.

View Courses

Once courses are created and instantiated in the database, they must be able to be seen by prospective students. This page allows students to easily view all courses and the enrol button will eventually have the functionality to add them to the course for completion.

```
  factory Course.fromFirestore(DocumentSnapshot doc) {
    Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
    return Course(
      courseId: data['CourseID'] ?? '',
      title: data['Title'] ?? '',
      imageUrl: data['ImageUrl'] ?? '',
      duration: data['Duration'] ?? 0,
      rating: (data['Rating'] ?? 0).toDouble(),
      category: data['Category'] ?? '',
    );
  }

  String formatDuration(int duration) {
    final hours = duration ~/ 60;
    final minutes = duration % 60;
    return '${hours}h ${minutes}m';
  }

  Future<String> getImageUrl() async {
    try {
      final ref = FirebaseStorage.instance.ref().child(imageUrl); // Specify the child reference
      final url = await ref.getDownloadURL();
      return url;
    } catch (e) {
      print('Error getting image URL: $e');
      return ''; // Return an empty string or handle the error as needed
    }
  }
}
```

Figure 91 - Firebase Fetch Courses [Code]

Fetching the essential data for the course widget, such as: title, ID, rating, category, and image.

```dart
// CourseTile widget
class CourseTile extends StatelessWidget {
  final Course course;

  const CourseTile({required this.course});

  @override
  Widget build(BuildContext context) {
    final imageUrl = course.imageUrl.isNotEmpty
        ? course.imageUrl
        : '/Users/hcampbell/Documents/Uni/Year3/FYProject/Flutter/e_learning_app/web/icons/background.png';

    return Card(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: const EdgeInsets.all(4.0),
            child: Center(
              child: LayoutBuilder(
                builder: (context, constraints) {
                  final width = constraints.maxWidth * 1.0;
                  final height = width * 0.4;
                  return ClipRRect(
                    borderRadius: BorderRadius.circular(8.0),
                    child: Image.network(
                      imageUrl,
                      width: width,
                      height: height,
                      fit: BoxFit.cover,
```

Figure 92 - Course Tile Widget [Code]

The widget takes all of the fetched information and builds a course tile to show each course.

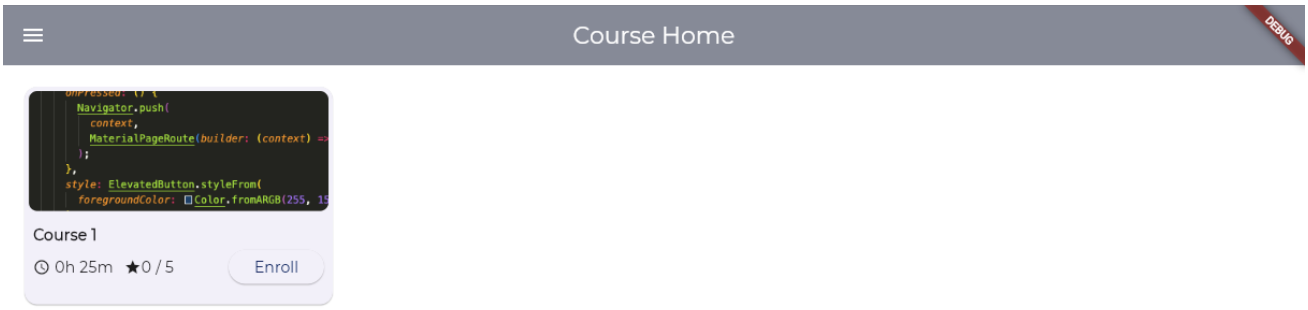

Figure 93 - Course Tile Widget [UI]

The course tile showing on the home screen.

Course Management

To allow teachers and admins to make edits to courses and delete them, the upload course navigation item will become manage courses and have three options within: Create, Edit, Delete.

```
if (userRole == 'Teacher') {
  menuItems.add(
    ListTile(
      leading: Icon(Icons.upload_file),
      title: Text('Manage Courses'),
      selected: selectedIndex == drawerItems.indexOf('Manage Courses'), // Highlight when selected
      onTap: () {
        Navigator.pushReplacementNamed(context, '/manage_courses');
      },
    ), // ListTile
  );
```

Figure 94 - Manage Courses [Code]

Updating the navigator and name of the section to manage courses.

```
body: Center(
  child: Column(
    mainAxisSize: MainAxisSize.max,
    children: [
      CourseCard(
        iconData: Icons.add,
        title: 'Create New Course',
        color: backgroundColor3,
        onTap: () {
          Navigator.pushReplacementNamed(context, '/course_create');
        },
      ), // CourseCard
      CourseCard(
        iconData: Icons.edit,
        title: 'Edit Existing Course',
        color: backgroundColor,
        onTap: () {
          Navigator.pushReplacementNamed(context, '/course_edit');
        },
      ), // CourseCard
      CourseCard(
        iconData: Icons.delete,
        title: 'Delete Existing Course',
        color: backgroundColor2,
        onTap: () {
          Navigator.pushReplacementNamed(context, '/course_delete');
        },
```

Figure 95 - Navigation Pane [Code]

Adding three new pages for the new course management options.

```
child: IconButton(
  icon: Icon(Icons.delete, color: Colors.white),
  onPressed: () {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text("Confirm Delete"),
          content: Text("Are you sure you want to delete this course?"),
          actions: <Widget>[
            TextButton(
              child: Text("Delete"),
              onPressed: () {
                FirebaseFirestore.instance.collection('courses').doc(course.courseId).delete()
                  .then((_) => Navigator.of(context).pop())
                  .catchError((error) => print("Delete failed: $error"));
              },
            ), // TextButton
            TextButton(
              child: Text("Cancel"),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ), // TextButton
```

Figure 96 - Delete Courses [Code]

Code for deleting a course.

```
Future<void> _loadCourseDetails() async {
  FirebaseFirestore.instance.collection('courses').doc(widget.courseID).get().then((courseDoc) {
    if (courseDoc.exists) {
      var data = courseDoc.data();
      setState(() {
        _courseTitleController.text = data?['Title'] ?? '';
        _courseDescriptionController.text = data?['Description'] ?? '';
        _courseCategoryController.text = data?['Category'] ?? '';
        _courseDurationController.text = data?['Duration']?.toString() ?? '';
        imageUrl = data?['ImageUrl'] ?? '';
        previewImageUrl = imageUrl ?? '';
      });
    }
  });
}

Future<void> _updateCourseDetails() async {
  if (_formKey.currentState!.validate()) {
    FirebaseFirestore.instance.collection('courses').doc(widget.courseID).update({
      'Title': _courseTitleController.text,
      'Description': _courseDescriptionController.text,
      'Category': _courseCategoryController.text,
      'Duration': int.tryParse(_courseDurationController.text) ?? 0,
      'ImageUrl': imageUrl,
```

Figure 97 - Edit Courses [Code]

Code for editing a course.

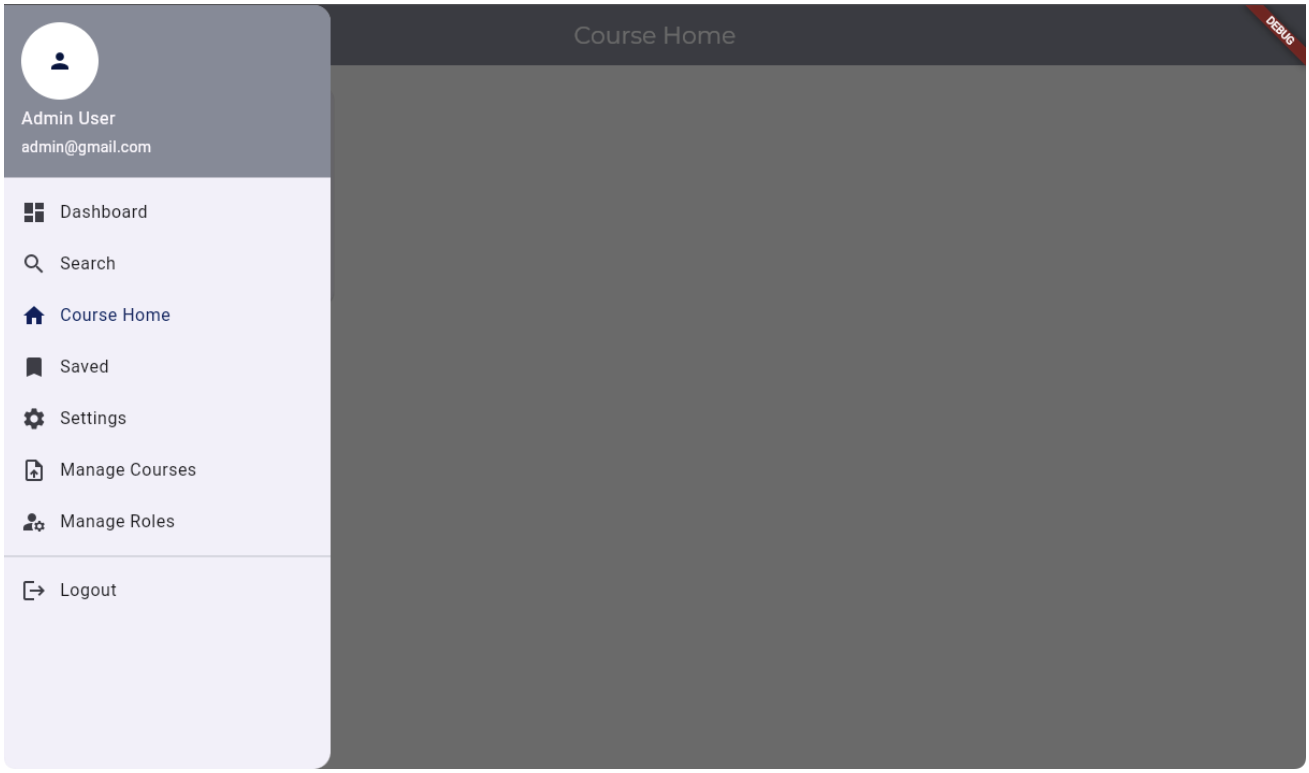

Figure 98 - New Navigation Pane [UI]

The new menu item ‘manage courses’ shows on the navigation pane.

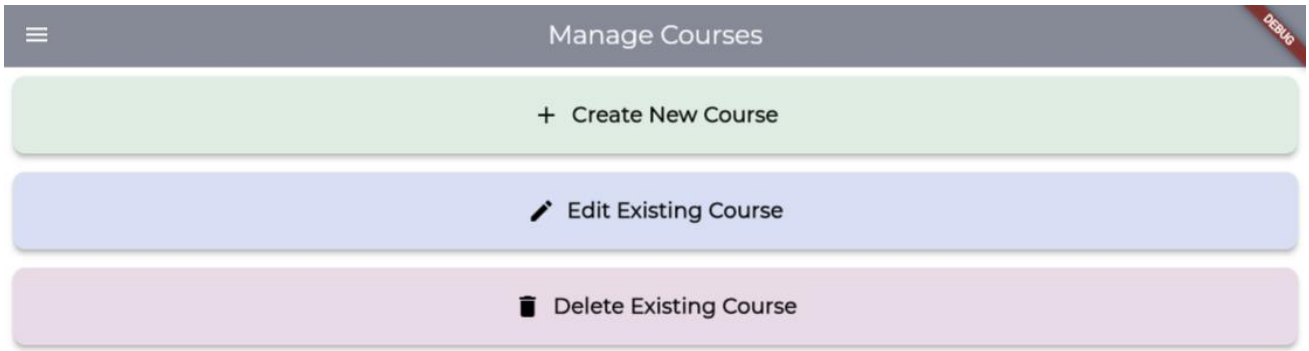

Figure 99 - Manage Courses Screen [UI]

Within manage courses there are three options, to create, edit, and delete courses.

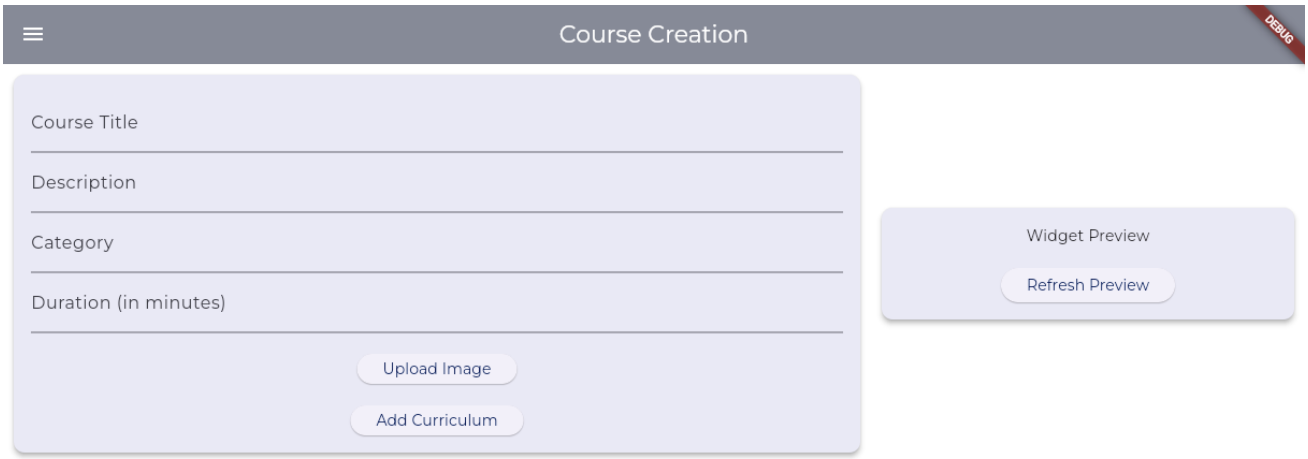

Figure 100 - Course Creation [UI]

Creating a new course, looks the same essentially as before

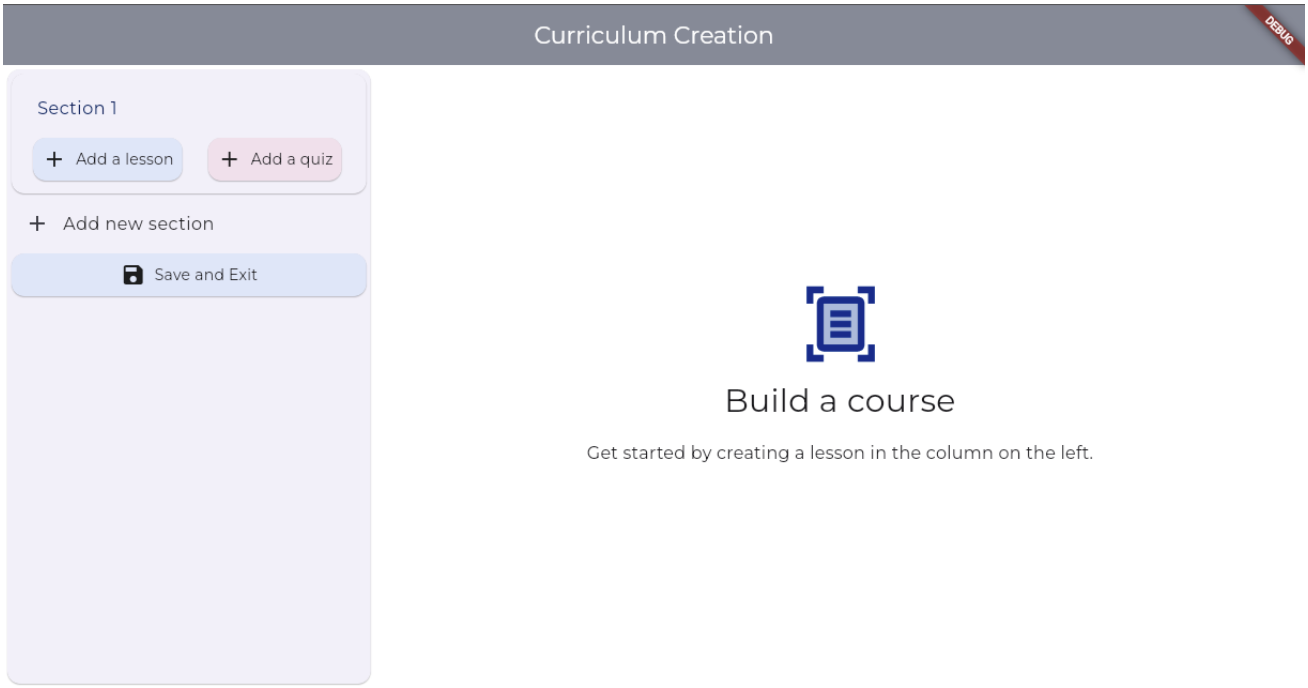

Figure 101 - Curriculum Creation [UI]

Adding curriculum to new course, also unchanged.

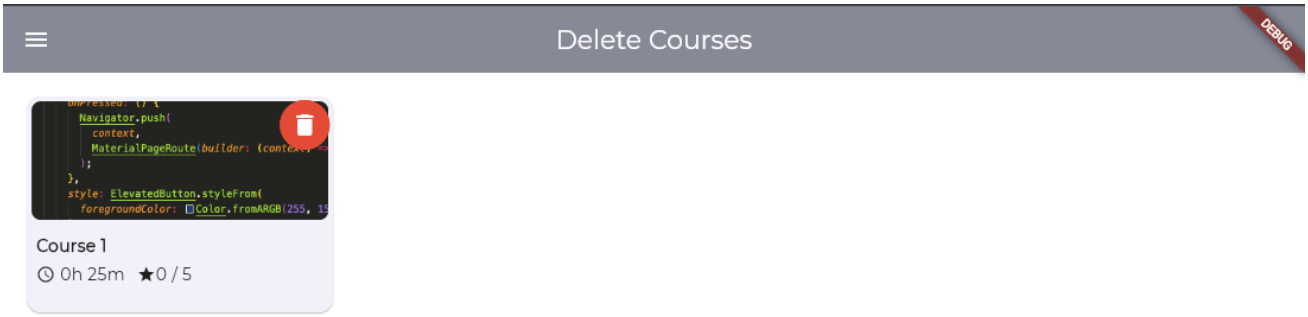

Figure 102 - Delete Courses [UI]

List all courses to delete any course.

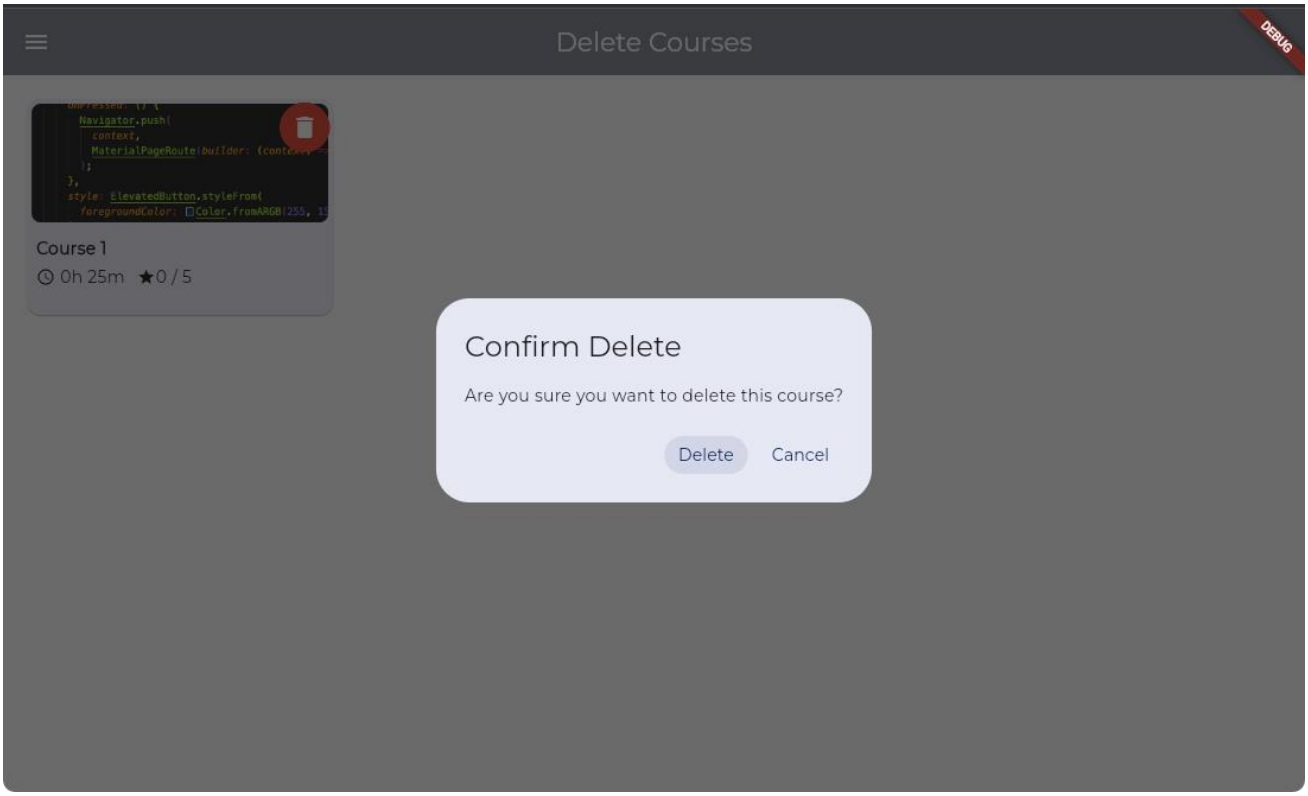

Figure 103 - Delete Courses Confirm [UI]

Similarly to users, it will require a confirmation to make sure it is deleted.

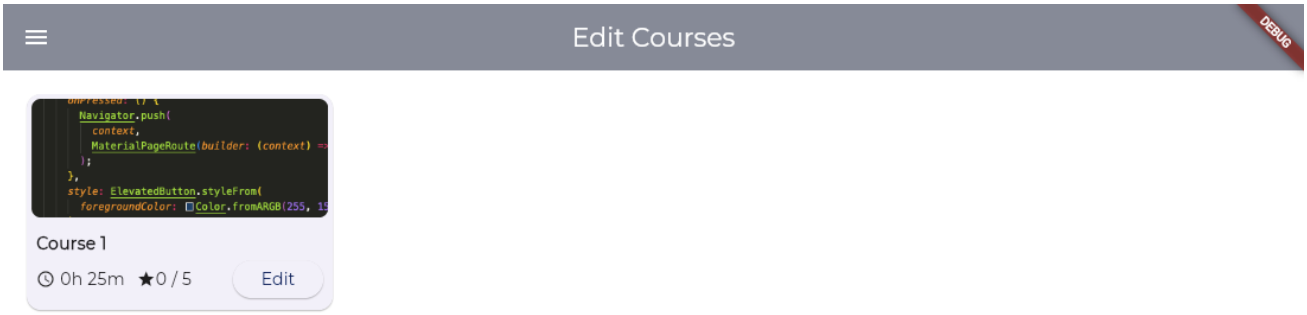

Figure 104 - Edit Courses [UI]

Display all courses, with an edit button.

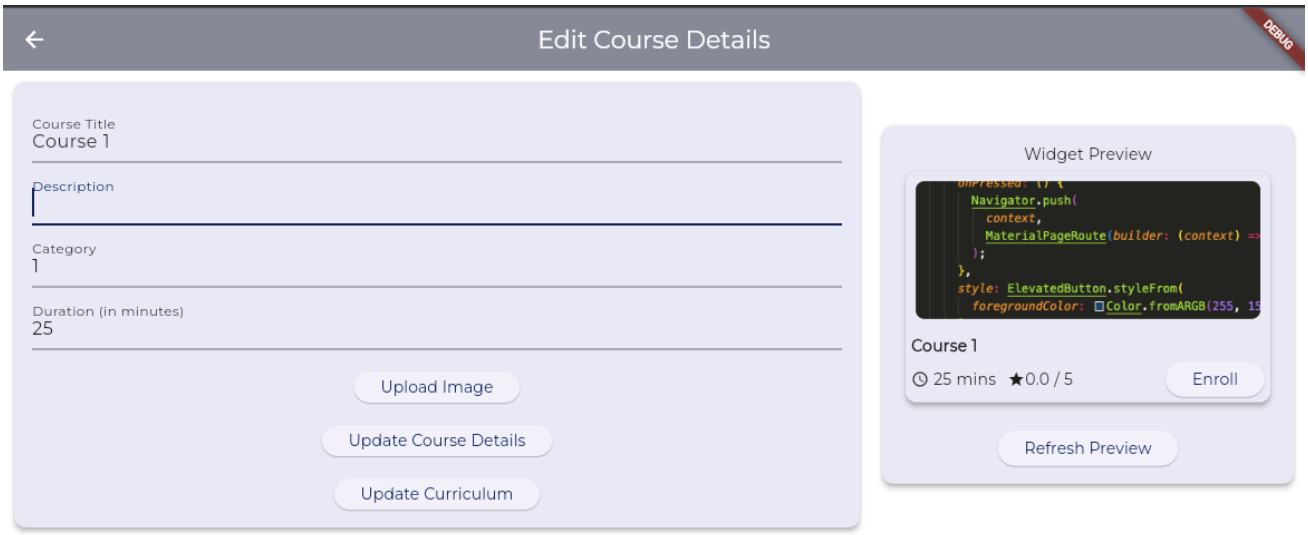

Figure 105 - Update Course Details [UI]

Editing a course opens up a pre-filled page where any details can be changed and saved, or curriculum can be updated.

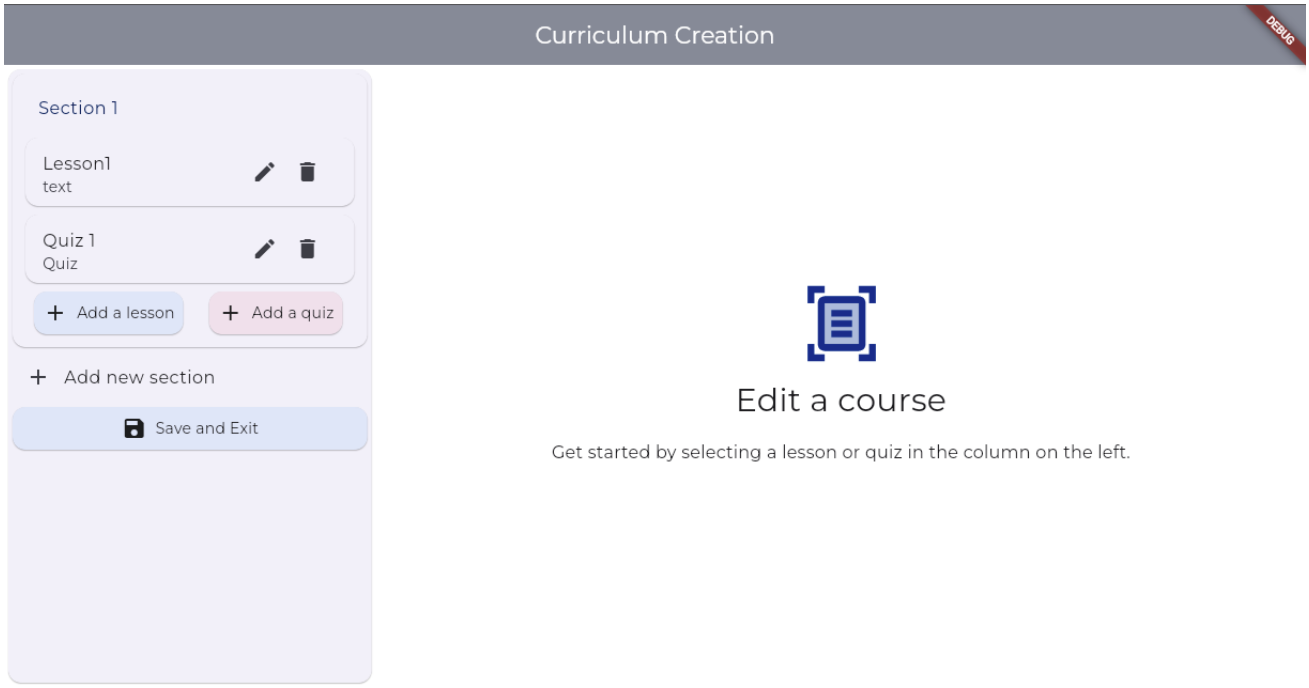

Figure 106 - Edit Curriculum [UI]

Updating the curriculum will load in all the existing lessons and quizzes, the user is then able to edit any lesson or quiz, as well as add more.

6.2.5 Testing

Feature	Test Description	Result
User Roles / Access Permissions	Students should only see navigation items 1-4 for accessing learning content.	Success
	Teachers should have access to items 1-4 for learning content as well as manage courses.	Success
	Admins should have access to items 1-4, manage courses and manage roles.	Success
User Role Management	Only admins should have access to change roles.	Success
	The page should load in all existing users with their correct permissions.	Success
	Changed roles should take immediate effect, allowing or removing access to functions of the app.	Success
	Changed roles should be updated in the Firestore accurately.	Success
Course Creation	Courses can be created with basic information (Title, Duration, Image, Description).	Success
	Preview Widget dynamically updates with content.	Success
	Add curriculum takes you to the course builder page to add Text, Video Lessons, and Quizzes.	Success
	Lessons are able to be created by giving a type; title; description and content.	Success
	Quizzes are able to be created with as many questions and as many answers as wanted.	Success
	Lessons and Quizzes add into sections.	Success
	Lessons can be removed.	Fail

	Lessons can be Edited from the panel on the right to change title, description etc	Success
	Lessons and Quizzes upload to Firestore when saved.	Success
Course View	Courses upload to the home page as they are created	Success
	Courses have the correct information from Firebase	Success
	Deleted courses no longer show on home page	Success
Manage Courses	Course manage option should only show for teachers, admins.	Success
	Course manage should present the user with 3 options: delete, edit, create.	Success
	The delete course page should show all courses existing to the current teacher only unless an admin	Success
	Courses should have a red delete icon which when pressed will confirm delete.	Success
	The edit course page should show all courses existing to the current teacher only unless an admin	Success
	The edit course page should work as the creation one does	Fail
	All changes to courses should be immediately reflected in the Firebase storage	Success

Table 6 - Iteration 2 Testing

Fix for Lesson/Quiz Indexing Issue

When attempting to edit a lesson, some tests found an indexing error being thrown suggesting that the lesson being edited no longer existed. Through the debugger and breakpoints, it became clear that the existing method for indexing the sections was not effective. Giving Quizzes their own index and list to display underneath each other has resolved the issue.

```
List<List<Lesson>> sectionLessons = [[]];
List<List<Quiz>> sectionQuizzes = [[]];
List<QuizQuestion> quizQuestions = [];
```

Figure 107 - Questions Array [Code]

Creating empty arrays for questions

```
void _addNewSection() {
  setState(() {
    sectionLessons.add([]);
    sectionQuizzes.add([]);
    _selectedSectionIndex = sectionLessons.length - 1;
  });
}
```

Figure 108 - New Add Section [Code]

Updating the array sizes to add new questions.

6.2.6 Review

Iteration 2 of the project incorporates more advanced features to develop the e-learning platforms key functionality and user experience. Key aspects of the iteration were user role differentiation, course creation and management, curriculum development, and quiz integration - all built from a robust database system using Firestore. These features are highest on the requirements and form the core of the application, allowing the platform to meet the needs of users, including teachers, students, and admins.

6.3 Iteration 3

6.3.1 Features to Implement

- Course Completion (FR9, FR10)
  - Text lessons
  - Video lessons
  - Quizzes/Assessments
- Course Ratings/feedback (FR12)
- Progress Tracking (FR11)
- Search Courses (FR5) (NFR5)
- Filter Courses (FR5) (NFR5)

6.3.2 Task Breakdown

The feature implementation has been broken down into actionable tasks to be used throughout the development process.

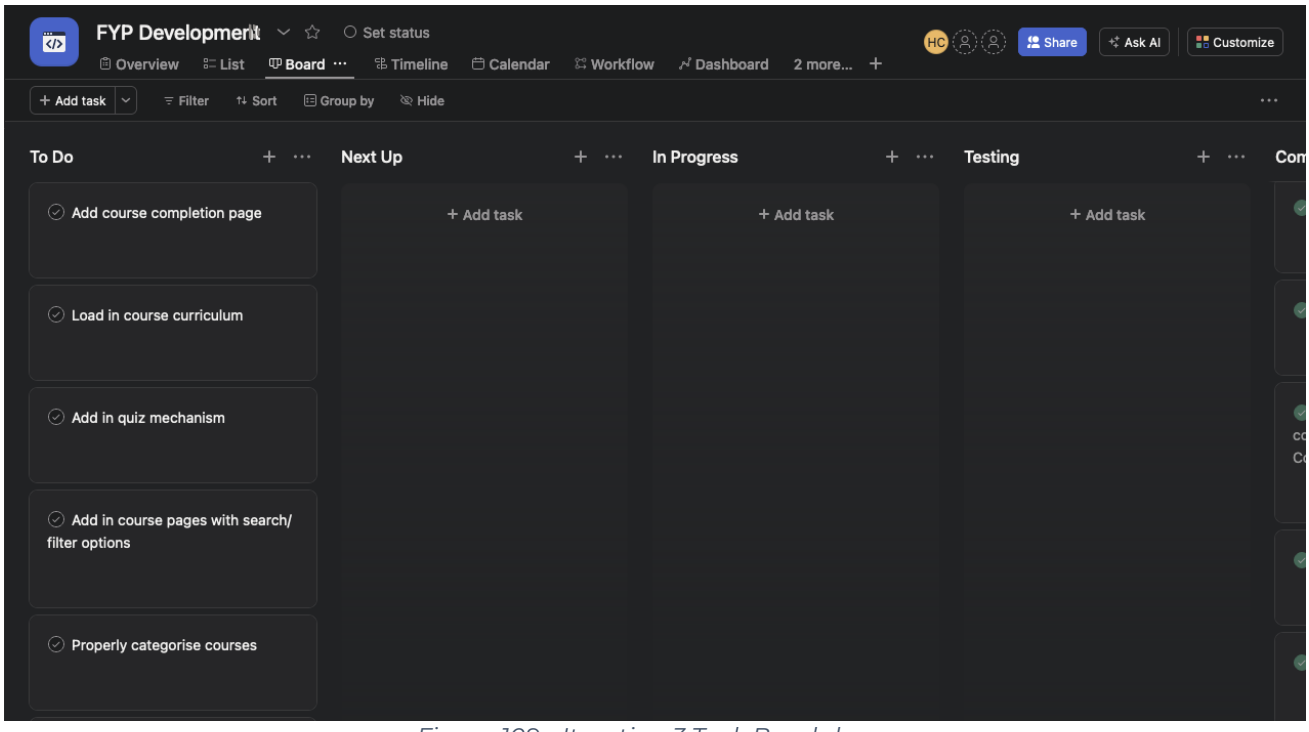

Figure 109 - Iteration 3 Task Breakdown

6.3.3 Design

The design of these features will incorporate features from Figures 20, 21, 22, 23 and 24 in the design section, for the completion of courses and quizzes. This iteration will require a heavy focus on design elements to ensure usability and effectiveness as a learning environment.

6.3.4 Implementation

Course Completion Page

This page is one of the most important in the whole application. From here, users will have the ability to view course content such as videos, text lessons and quizzes. To link users to courses, a new attribute will be added to each collection - users will gain the coursesEnrolled field and courses gain the usersEnrolled field, these will link to create userProgress which contains information about each user's progress on each course.

```dart
static Future<void> enrollUserInCourse(String courseID) async {
  try {
    // Get the current user ID
    User? currentUser = FirebaseAuth.instance.currentUser;
    String? userID = currentUser?.uid;

    print('Enrolling user $userID in course: $courseID');

    if (userID != null) {

      // Update user document
      await FirebaseFirestore.instance.collection('users').doc(userID).update({
        'coursesEnrolled': FieldValue.arrayUnion([courseID]),
      });

      // Create a sub-collection for user progress within the course
      await FirebaseFirestore.instance.collection('courses').doc(courseID)
          .collection('userProgress').doc(userID).set({
        'progressPercentage': 0,
        'completedLessons': [],
        'quizScores': {},
      });

      // Update course document
      await FirebaseFirestore.instance.collection('courses').doc(courseID).update({
        'enrolledUsers': FieldValue.arrayUnion([userID]),
      });
    }
```

Figure 110 - Firebase Enrol User [Code]

Adding sub collections for user progress to Firebase, courses enrolled to each user and users enrolled to each course.

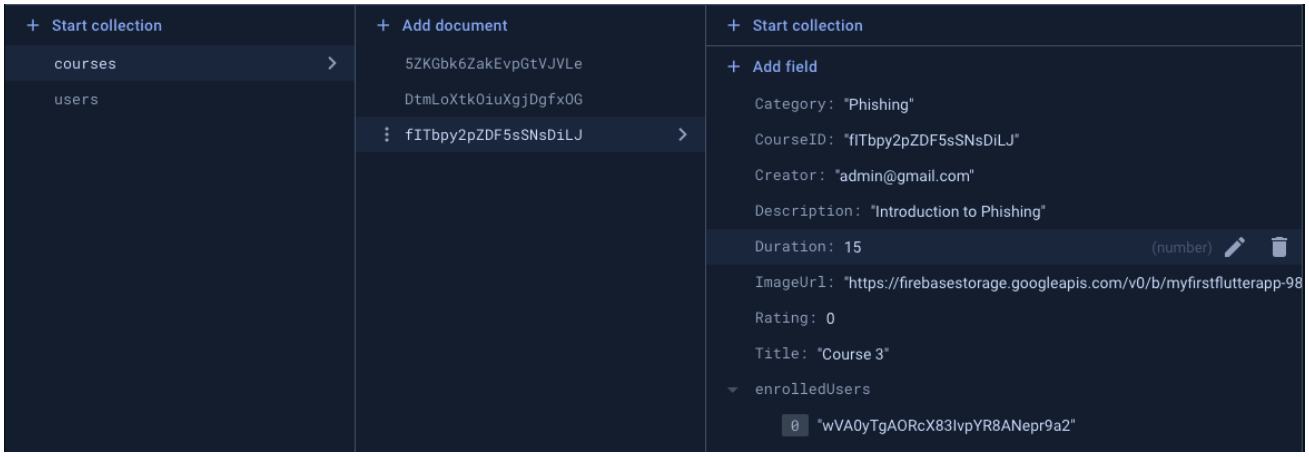

Figure 111 - Firebase 'enrolledUsers'.

Enrolled users in course.

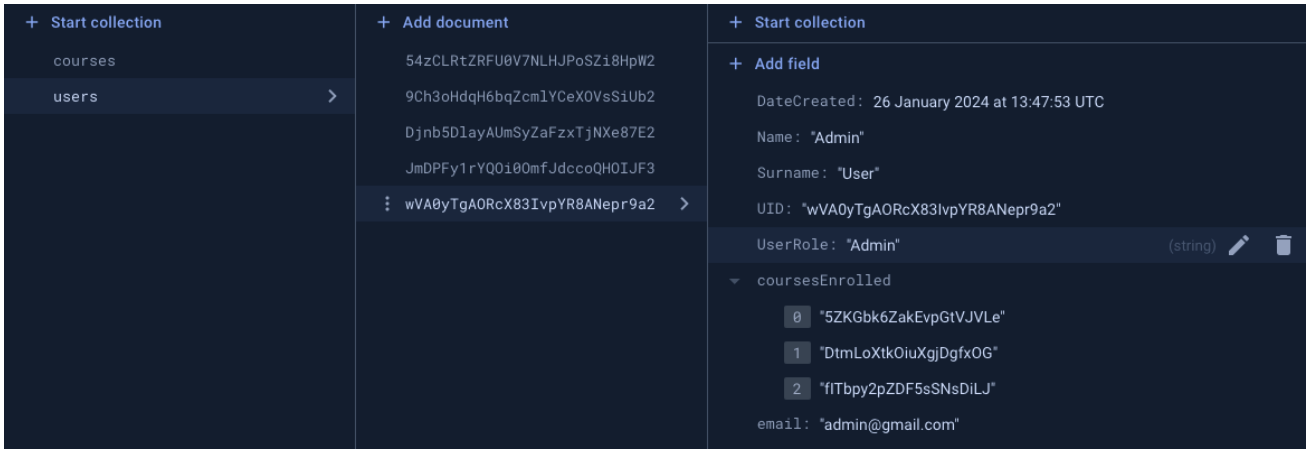

Figure 112 - Firebase 'coursesEnrolled'.

Courses enrolled for user.

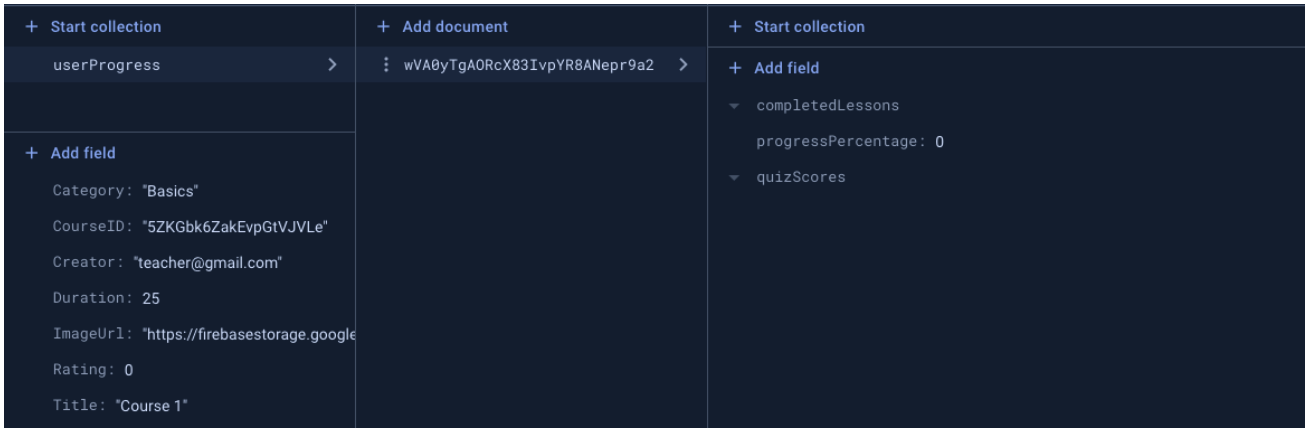

Figure 113 - Firebase 'userProgress'.

Linked course progress for every user on each course.

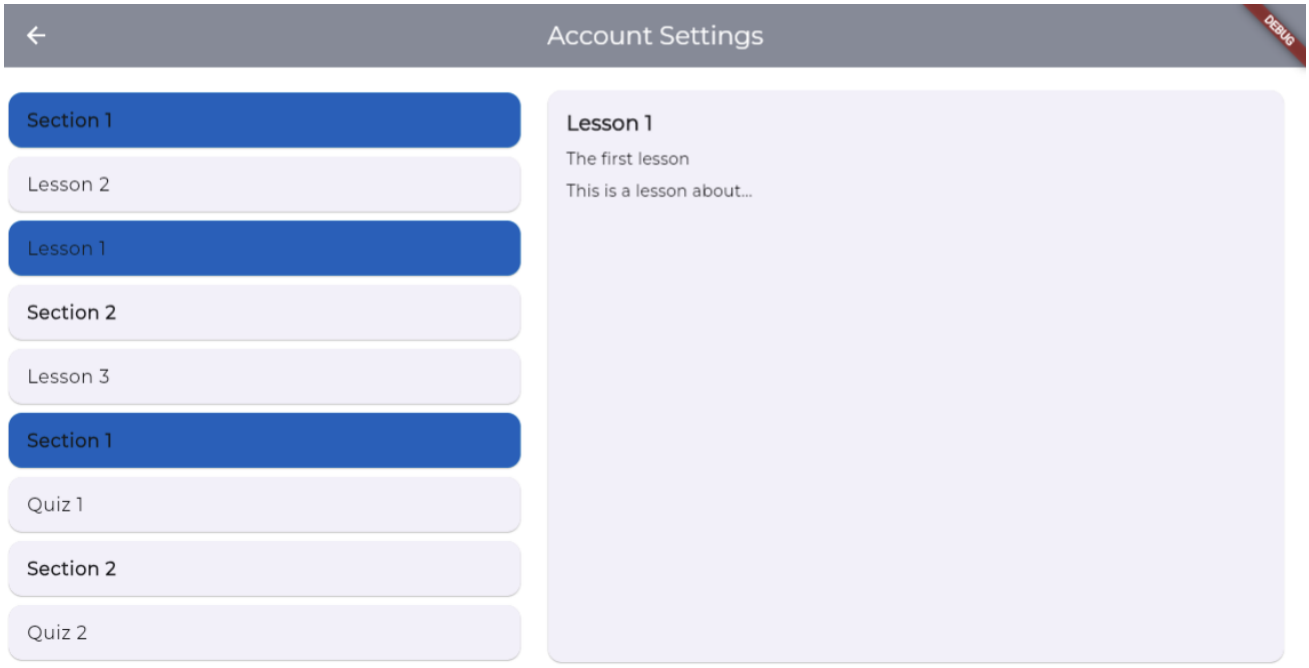

Figure 114 - Course Completion Screen [UI]

When completing a course, lessons show up out of order and quizzes are separated.

```
class Lesson {
  String type;
  String title;
  String description;
  String content;
  int order;

  Lesson({
    required this.type,
    required this.title,
    required this.description,
    required this.content,
    required this.order,
  });
}
```

Figure 115 - Lesson Class [Code]

Adding order field to the lesson class, this will enable them to be saved in the order they are made and displayed the same way.

```dart
List<Widget> _buildSectionCards(List<dynamic> items) {
  // Sort items based on section and order
  items.sort((a, b) {
    // First, compare sections
    int sectionComparison = a.section.compareTo(b.section);
    if (sectionComparison != 0) {
      return sectionComparison;
    } else {
      // If sections are equal, compare order
      if (a is Lesson && b is Lesson) {
        return a.order.compareTo(b.order);
      } else if (a is Quiz && b is Quiz) {
        return a.order.compareTo(b.order);
      }
      return 0;
    }
  });
```

Figure 116 - Adding Order [Code]

Finding the correct order for each lesson when building the section card.

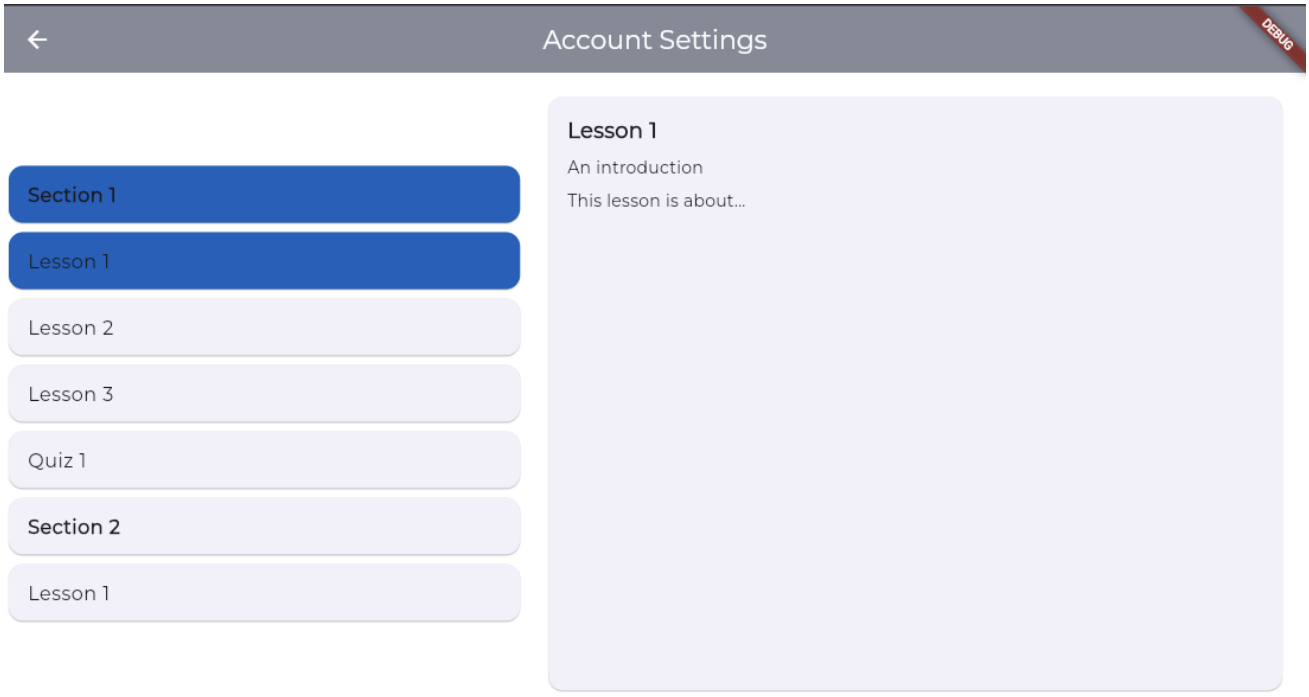

Figure 117 - Course Completion Screen [UI]

Now lessons are in order and quizzes show at the bottom of the section.

```
Text(selectedLesson!.content),
if (selectedLesson!.lessonType == 'Video')
  YoutubePlayer(
    controller: _controller,
    aspectRatio: 16/9,
  ), // YoutubePlayer
SizedBox(height: 20),
```

Figure 118 - YouTube Player [Code]

```
void updateVideoId(String url) {
  print('Updating video id: $url');
  final videoId = url.split('=').last;
  _controller.loadVideoById(videoId: videoId);
}
```

Figure 119 - YouTube Player [Code]

Adding in YouTube support for videos

```
@override
void initState() {
  super.initState();
  fetchCourseData();
  _controller.loadVideoById(videoId: 'iLnmTe5Q2Qw'); //default value
}

void updateVideoId(String url) {
  print('Updating video id: $url');
  final videoId = url.split('=').last;
  _controller.loadVideoById(videoId: videoId);
}
```

Figure 120 - YouTube Player [Code]

Default video id and a function to change it based on current lesson.

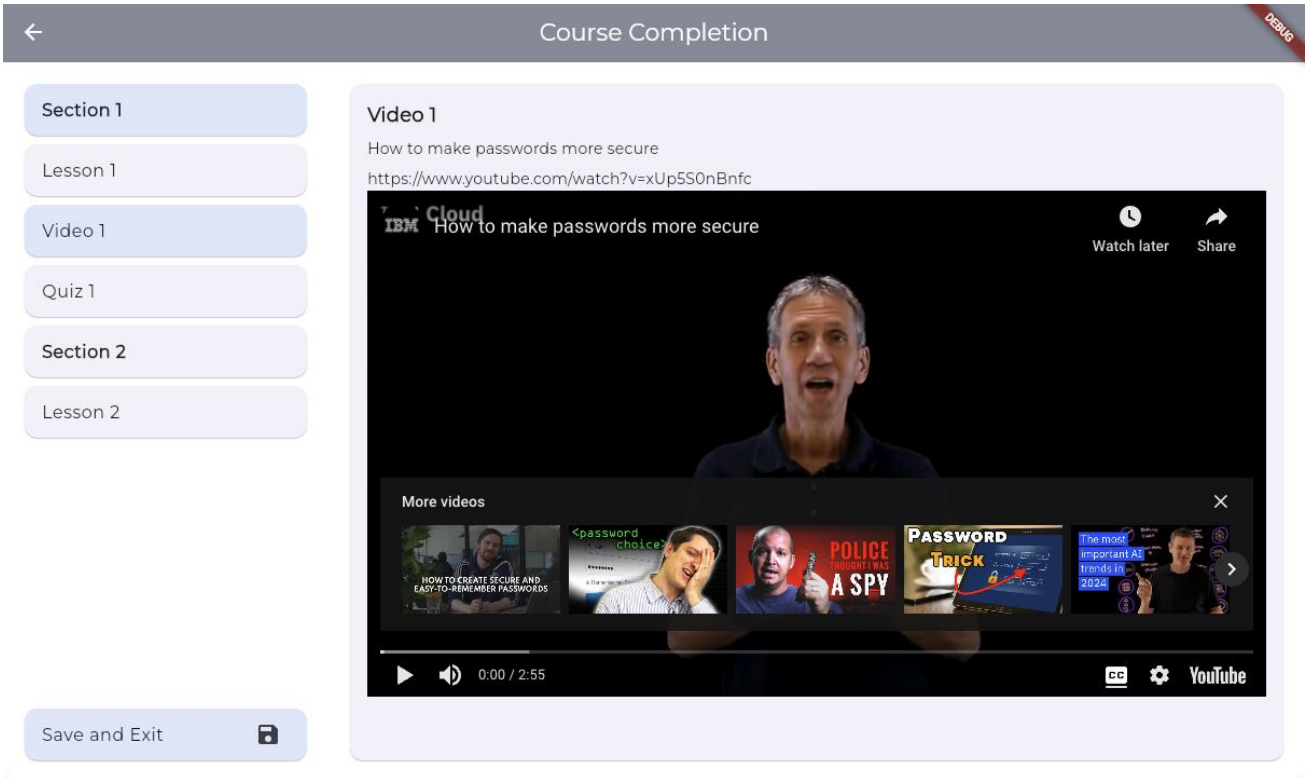

Figure 121 - YouTube Player [UI]

The YouTube player integrates into the card and will auto play whatever video is the 'content' of the lesson.

```
else if (selectedQuiz != null)
  Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        selectedQuiz!.title,
        style: TextStyle(
          fontSize: 18,
          fontWeight: FontWeight.bold,
        ), // TextStyle
      ), // Text
      SizedBox(height: 8),
      Text(selectedQuiz!.description),
      SizedBox(height: 8),
      _quizInteractionView(),
      SizedBox(height: 20),
```

Figure 122 - Quiz Section [Code]

Building the quiz section and calling build methods.

```
Widget _quizInteractionView() {
  if (selectedQuiz == null) {
    return Center(child: Text("Please select a quiz.", style: TextStyle(fontSize: 20)));
  }

  if (showResults) {
    int correctAnswers = 0;
    for (int i = 0; i < userAnswers.length; i++) {
      if (userAnswers[i] == selectedQuiz!.questions[i]['correctAnswerIndex']) {
        correctAnswers++;
      }
    }
    selectedQuiz!.completed = true;
    return Center(child: Text("Your score is $correctAnswers/${selectedQuiz!.questions.length}", style: TextStyle(fontSize: 20)));
  }

  var currentQuestion = selectedQuiz!.questions[currentQuestionIndex];
  bool hasAnsweredCurrentQuestion = userAnswers.length > currentQuestionIndex;

  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Padding(
        padding: const EdgeInsets.symmetric(vertical: 8.0),
        child: Text(currentQuestion['questionText'], style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
      ), // Padding
      ...currentQuestion['options'].asMap().entries.map((option) {
        Color? buttonColor = Colors.grey.shade200; // Default color for unselected options
        if (hasAnsweredCurrentQuestion && userAnswers[currentQuestionIndex] == option.key) {
          buttonColor = option.key == currentQuestion['correctAnswerIndex'] ? Colors.green : Colors.red;
        }
```

Figure 123 - Quiz Scoring [Code]

Building the quiz buttons and scoring mechanism.

```
void _answerQuestion(int index) {
  if (userAnswers.length == currentQuestionIndex) {
    setState(() {
      userAnswers.add(index);
    });
  }
}

void _nextQuestion() {
  if (currentQuestionIndex < selectedQuiz!.questions.length - 1) {
    setState(() {
      currentQuestionIndex++;
    });
  } else {
    setState(() {
      showResults = true;
    });
  }
}
```

Figure 124 - Quiz Buttons [Code]

Adding the quiz buttons, which will take you to the next question and show the result.

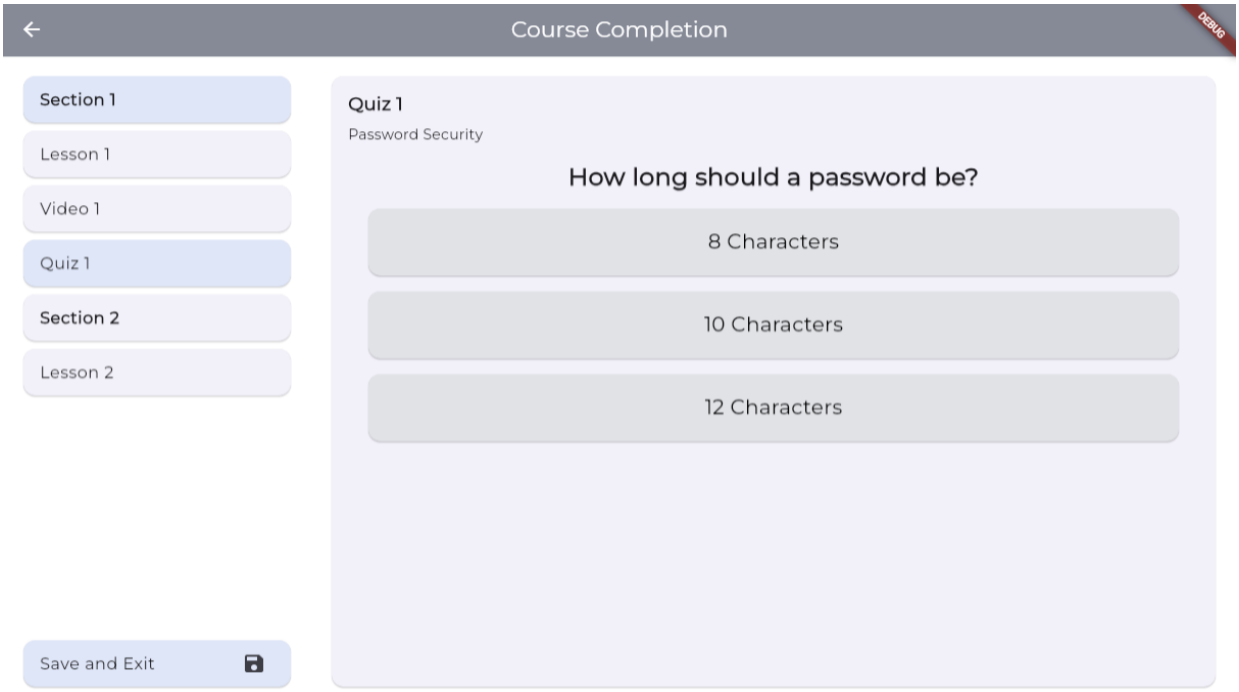

Figure 125 - Quiz Completion [UI]

The final UI for quiz completion.

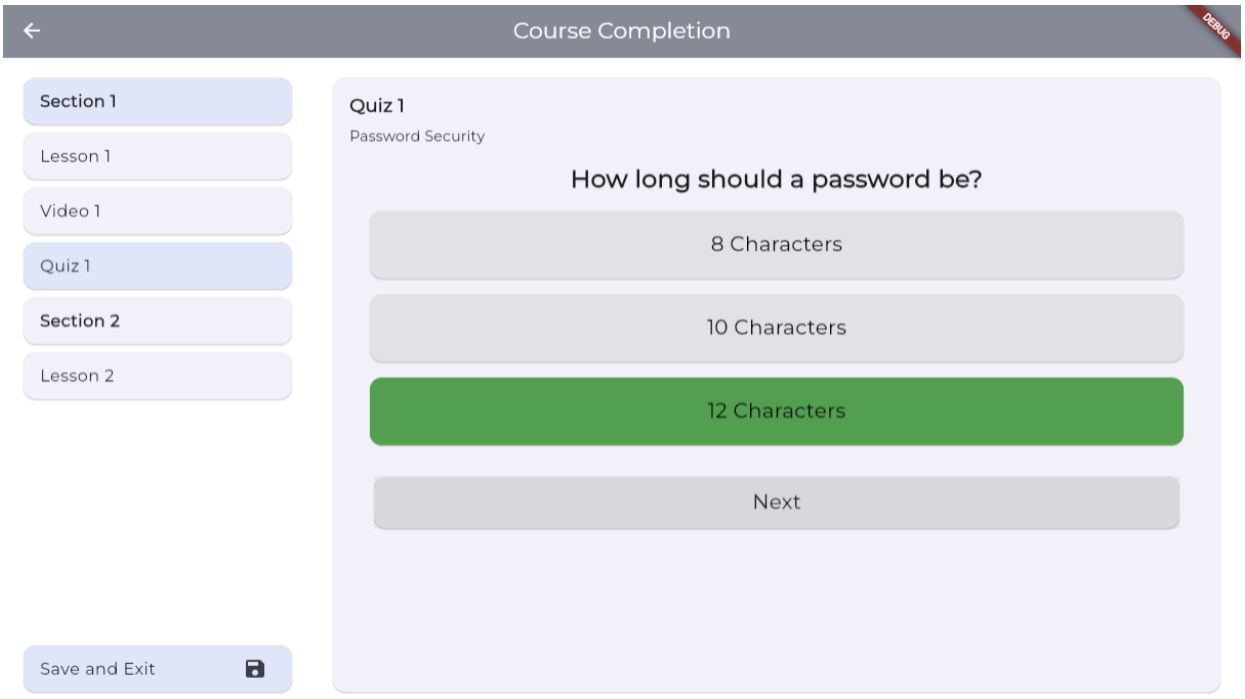

Figure 126 - Quiz Correct Answer [UI]

Correct answers show in green.

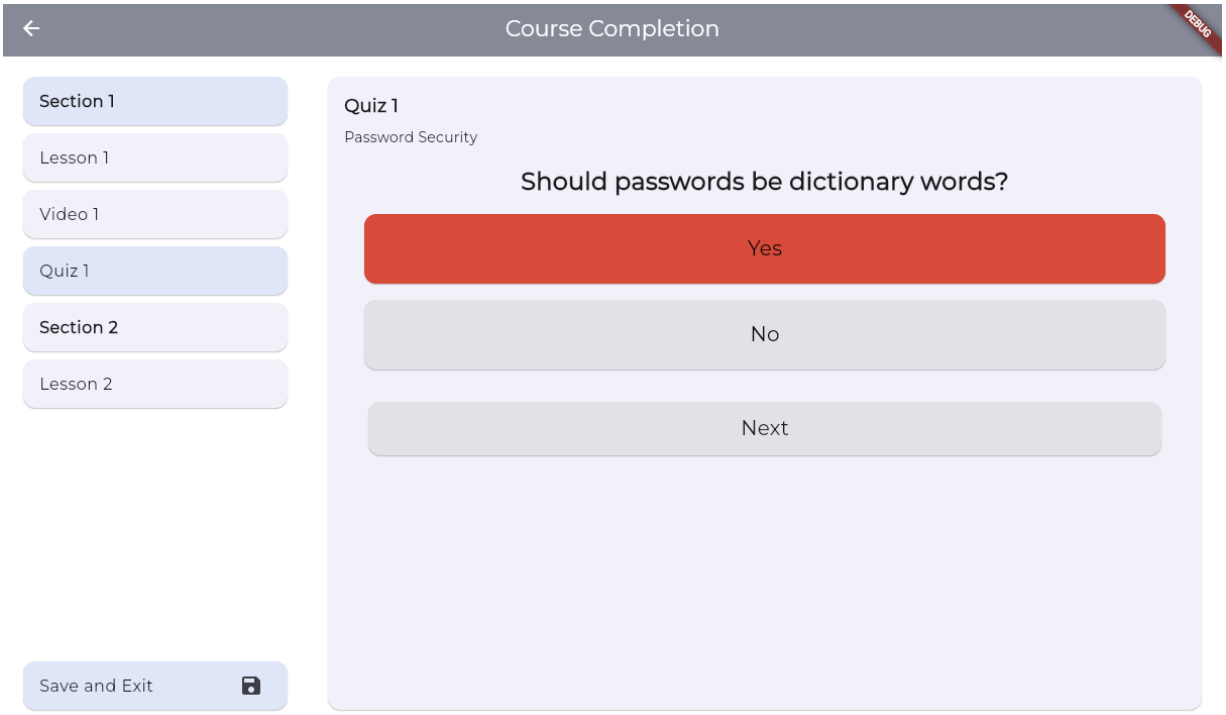

Figure 127 - Quiz Incorrect Answer [UI]

Incorrect answers highlight in red.

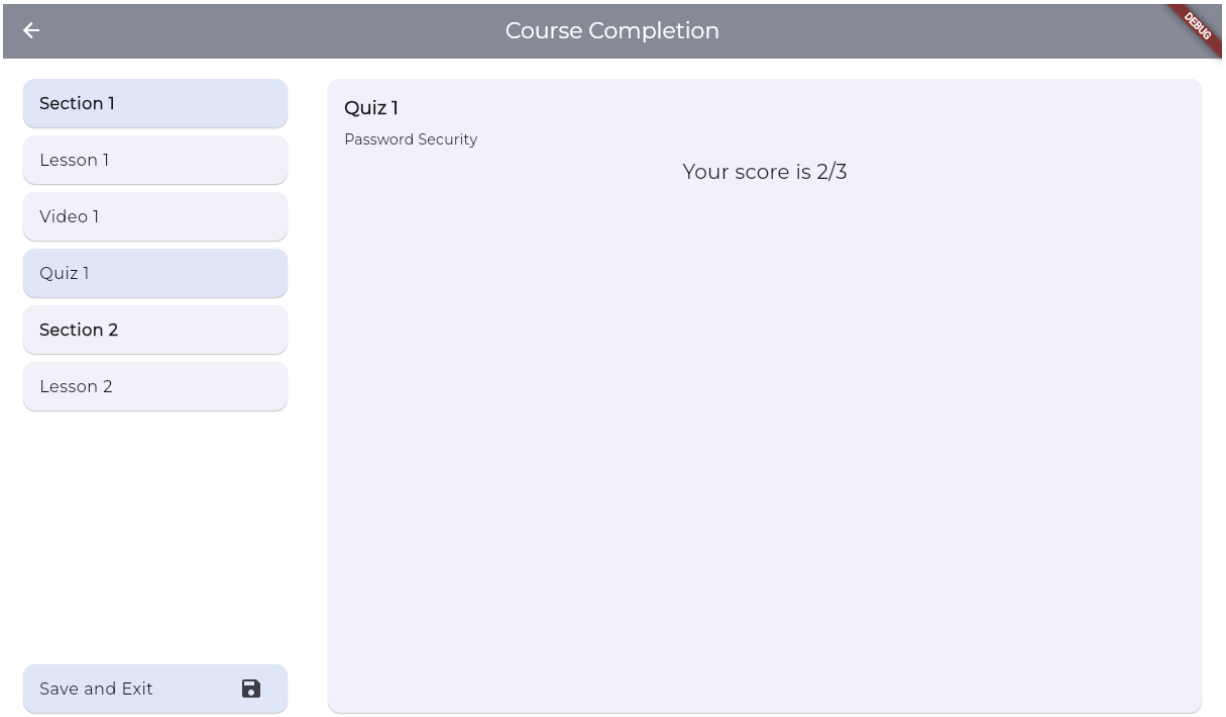

Figure 128 - Quiz Scoring [UI]

When the quiz is finished, a final score is given.

```
    _quizInteractionView(),
    SizedBox(height: 20),
    _buildLessonNavigationButton(items, selectedQuiz, isNext: true),
  ],
```

Figure 129 - Course Navigation [Code]

```
  SizedBox(height: 20),
  _buildLessonNavigationButton(items, selectedLesson, isNext: true)
],
```

Figure 130 - Course Navigation [Code]

Navigating through the course, using next and back buttons. Depending on what is next, a lesson or quiz, different build methods need to be called.

```
Widget _buildLessonNavigationButton(List<dynamic> items, dynamic currentItem,
    {required bool isNext}) {
  int index = items.indexOf(currentItem);
  bool hasNext = index < items.length - 1;

  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      if (isNext && index > 0)
        TextButton(
          onPressed: () {
            setState(() {
              selectedLesson = items[index - 1] is Lesson
                  ? items[index - 1] as Lesson
                  : null;
              selectedQuiz = items[index - 1] is Quiz
                  ? items[index - 1] as Quiz
                  : null;
              // Mark the previous lesson/quiz as completed and change its color
              if (items[index - 1] is Lesson) {
                (items[index - 1] as Lesson).completed = true;
              } else if (items[index - 1] is Quiz) {
                (items[index - 1] as Quiz).completed = true;
              }
            });
          },
          child: Text('Previous'),
        ), // TextButton
```

Figure 131 - Course Navigation Buttons [Code]

This widget is what adds in the next and previous button, it also controls which lesson is next up.

```
TextButton(
  onPressed: () {
    if (isNext && hasNext) {
      setState(() {
        selectedLesson = items[index + 1] is Lesson
            ? items[index + 1] as Lesson
            : null;
        selectedQuiz = items[index + 1] is Quiz
            ? items[index + 1] as Quiz
            : null;
        // Mark the current lesson/quiz as completed and change its color
        if (items[index] is Lesson) {
          (items[index] as Lesson).completed = true;
        } else if (items[index] is Quiz) {
          (items[index] as Quiz).completed = true;
        }
      });
    } else {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: Text('Congratulations!'),
            content: Text('You have completed the course.'),
            actions: <Widget>[
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                  Navigator.of(context).pop();
                },
                child: Text('Finish'),
              ), // TextButton
            ], // <Widget>[]
          ); // AlertDialog
```

Figure 132 - Find Next Lesson [Code]

Logic to find next lesson to complete.

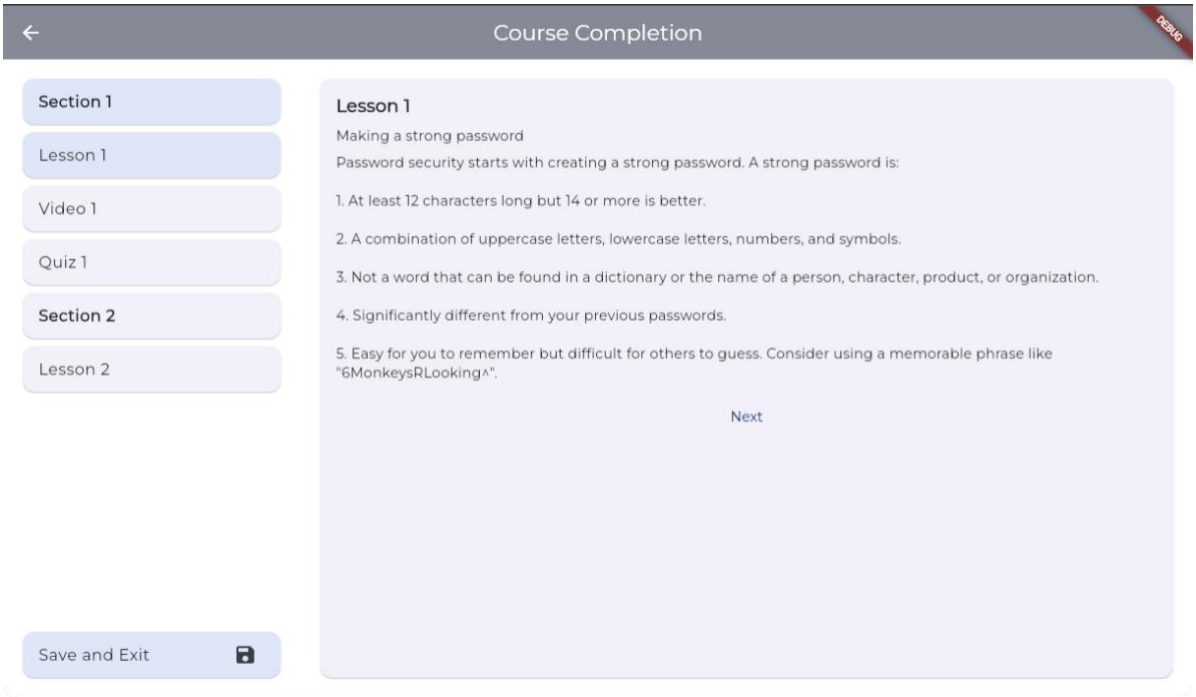

Figure 133 - Course Navigation [UI]

The final result is a simple next button which will lead you through the course until the finish.

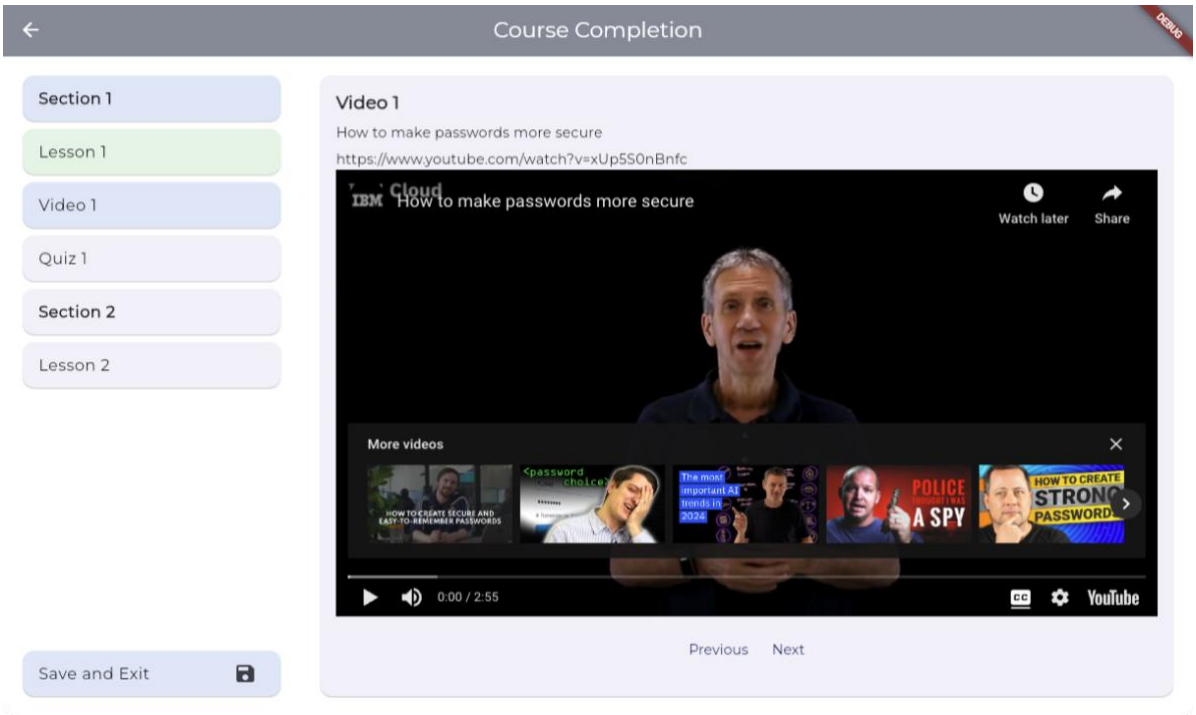

Figure 134 - Progress Tracking [UI]

As lessons are completed, they will start to highlight in green.

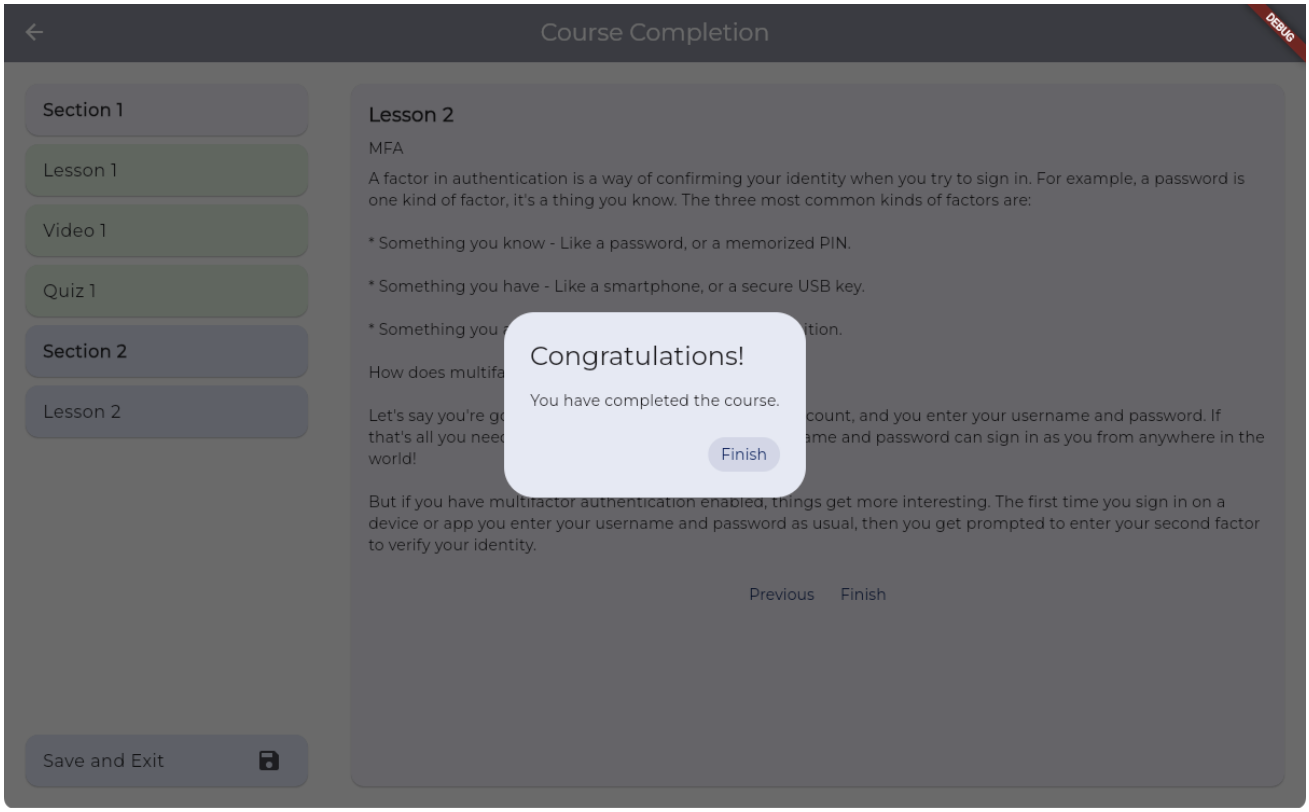

Figure 135 - Course Completed [UI]

Once you are at the last lesson, the button will change to ‘Finish’, pressing it takes you back to home screen.

Course Progress / Save States

As outlined in the literature review, a critical aspect of andragogical learning is the ability to complete lessons whenever the learner has free time - for this to be effective my application must store user's progress within courses and allow them to return to the spot they left from. This will use the linked database field previously mentioned, userProgress.

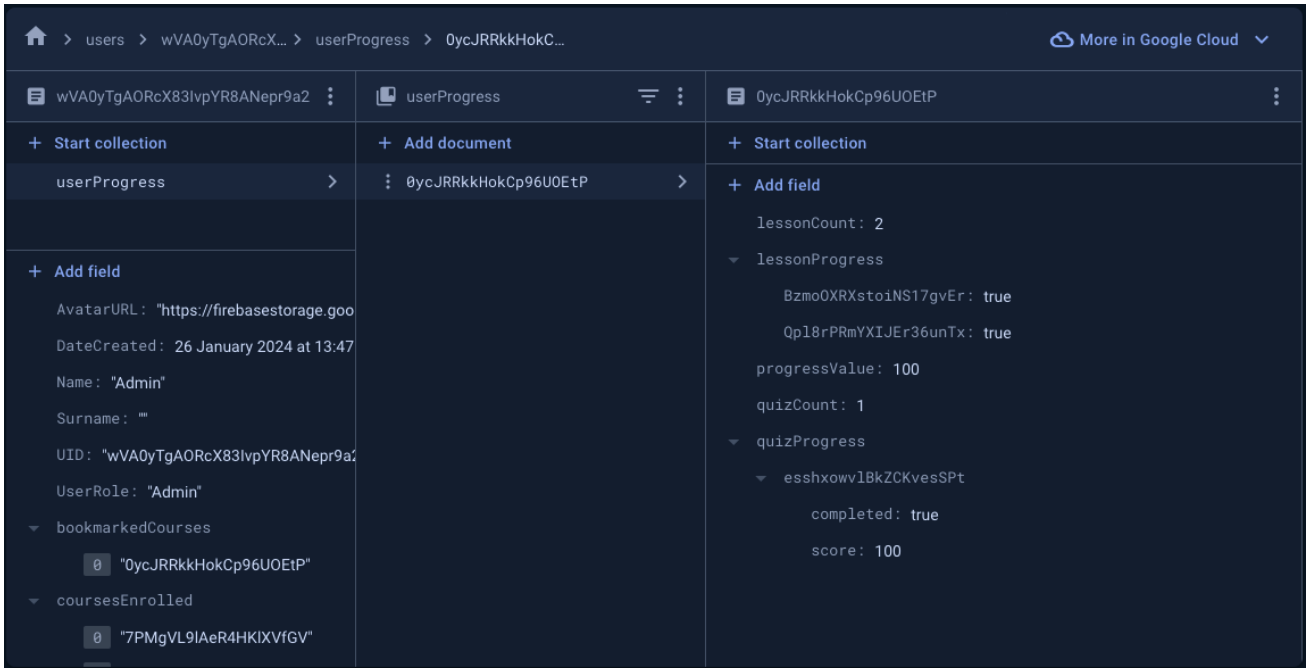

Figure 136 - Updated Firebase 'UserProgress'.

Adding in Firebase fields for course progress, 'quizProgress' is an array of quizzes which contains a score and completed value; 'quizCount' and 'lessonCount' find out how many lessons there are and are used to derive the 'progressValue' which is a percentage out of 100.

```
void checkOrCreateUserProgress() async {
  try {
    User? user = FirebaseAuth.instance.currentUser;

    if (user != null) {
      String userId = user.uid;

      var userProgress = await _firestore
          .collection('users')
          .doc(userId)
          .collection('userProgress')
          .doc(widget.courseId)
          .get();

      if (!userProgress.exists) {
        // User progress collection doesn't exist, create it
        // Fetch course data to get lesson and quiz IDs
        final courseDoc = await _firestore.collection('courses').doc(widget.courseId).get();
        final lessonSnapshot = await courseDoc.reference.collection('curriculum').get();
        final quizSnapshot = await courseDoc.reference.collection('quizzes').get();
```

Figure 137 - Check For Existing Progress Document [Code]

Checking for an existing progress document.

```
    // Create user progress document
    await _firestore
        .collection('users')
        .doc(userId)
        .collection('userProgress')
        .doc(widget.courseId)
        .set({
      'lessonProgress': lessonProgressMap,
      'quizProgress': quizProgressMap,
      'lessonCount': totalLessonCount,
      'quizCount': totalQuizCount,
      'progressValue': progressValue,
    });

    print('User progress collection created for user: $userId');
  } else {
    print("User progress collection already exists for user: $userId");
  }
} else {
  print('User is not signed in.');
```

Figure 138 - Create New Progress Document [Code]

Creating a progress document if there is not one already.

```dart
void updateLessonProgress(String lessonId) async {
try {
  User? user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    String userId = user.uid;

    // Reference to the user progress document for the current course
    DocumentReference userProgressRef = _firestore
        .collection('users')
        .doc(userId)
        .collection('userProgress')
        .doc(widget.courseId);

    // Update the lesson progress in the user progress document
    await userProgressRef.update({
      'lessonProgress.$lessonId': true,
    });

    print('Lesson progress updated for lesson: $lessonId');
  } else {
    print('User is not signed in.');
  }
```

Figure 139 - Save Progress State [Code]

Saving the users progress state.

```dart
void startCourse() {
  Lesson? firstLesson = items.whereType<Lesson>().reduce((a, b) => a.order < b.order ? a : b);
  setState(() {
    selectedLesson = firstLesson;
    selectedQuiz = null;
  });
}
```

Figure 140 - Find Previous Progress [Code]

```dart
Future<Map<String, dynamic>> fetchCourseDetailsAndProgress(String courseId) async {
final courseDetails = await _firestore.collection('courses').doc(courseId).get();
User? user = FirebaseAuth.instance.currentUser;
Map<String, dynamic> userProgress = {};
double progressValue = 0.0;

if (user != null) {
  final progressDoc = await _firestore
      .collection('users')
      .doc(user.uid)
      .collection('userProgress')
      .doc(courseId)
      .get();

if (progressDoc.exists) {
      userProgress = progressDoc.data()!;
      // Safely cast progressValue to double
      final progress = userProgress['progressValue'];
      if (progress is num) {
        progressValue = progress.toDouble();
      }
    }
  }
```

Figure 141 - Fetch Previous Progress [Code]

Fetching progress document from Firebase to continue a course.

```dart
  return {
    'title': courseDetails.data()?['Title'] ?? 'Course Title',
    'description': courseDetails.data()?['Description'] ?? 'Course Description',
    'progressValue': progressValue, // Pass the converted double value
  };
}
```

Figure 142 - Fetch Previous Progress [Code]

```dart
void continueCourse() async {
  final user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    final userId = user.uid;
    final userProgressDoc = await FirebaseFirestore.instance
        .collection('users')
        .doc(userId)
        .collection('userProgress')
        .doc(widget.courseId)
        .get();

    if (userProgressDoc.exists) {
      final userProgressData = userProgressDoc.data()!;
      final lessonProgress = userProgressData['lessonProgress'] as Map<String, dynamic>? ?? {};
      final quizProgress = userProgressData['quizProgress'] as Map<String, dynamic>? ?? {};

      // Find the first item (lesson or quiz) that hasn't been completed
      final nextItem = items.firstWhere((item) {
        if (item is Lesson) {
          return !(lessonProgress[item.id] as bool? ?? false);
        } else if (item is Quiz) {
          return !(quizProgress[item.id]?['completed'] as bool? ?? false);
        }
        return false;
      }, orElse: () => null);
```

Figure 143 - Continue From Progress [Code]

Logic to continue where user left off.

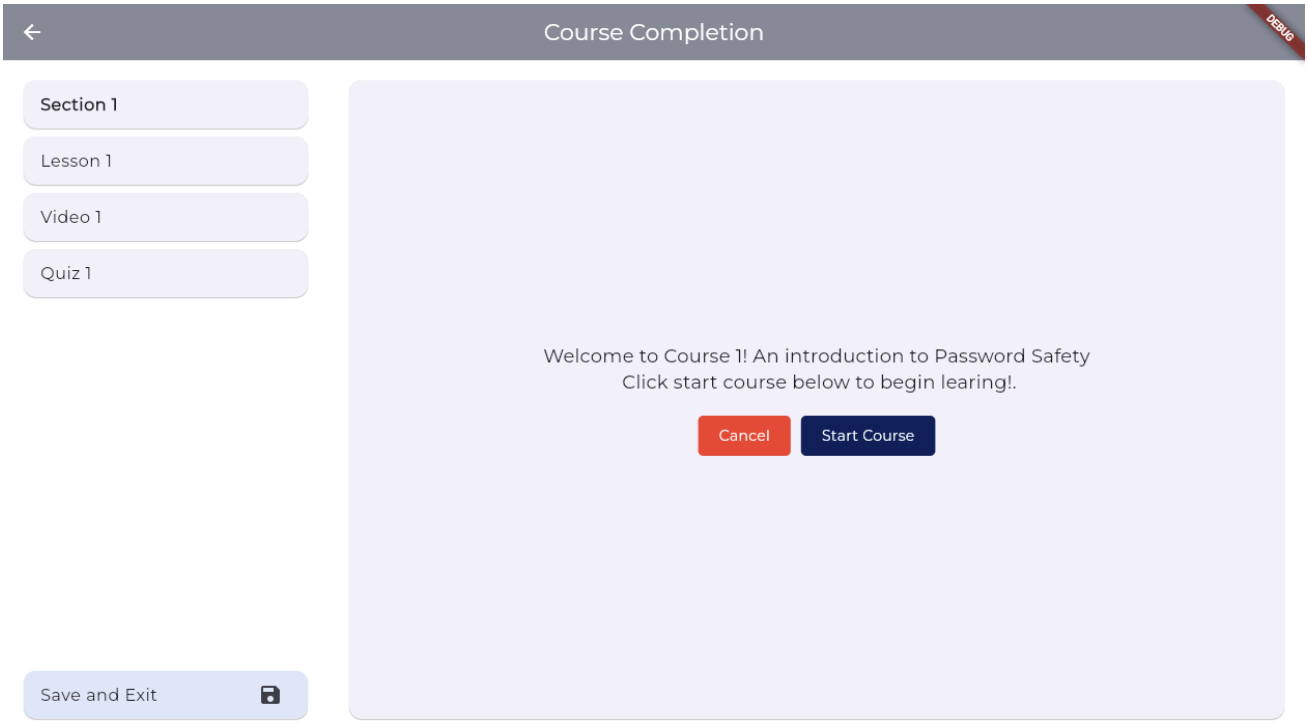

Figure 144 - Progress Tracking [UI]

If a user has already started the course before, they will be offered to restart or continue from where they previously left off.

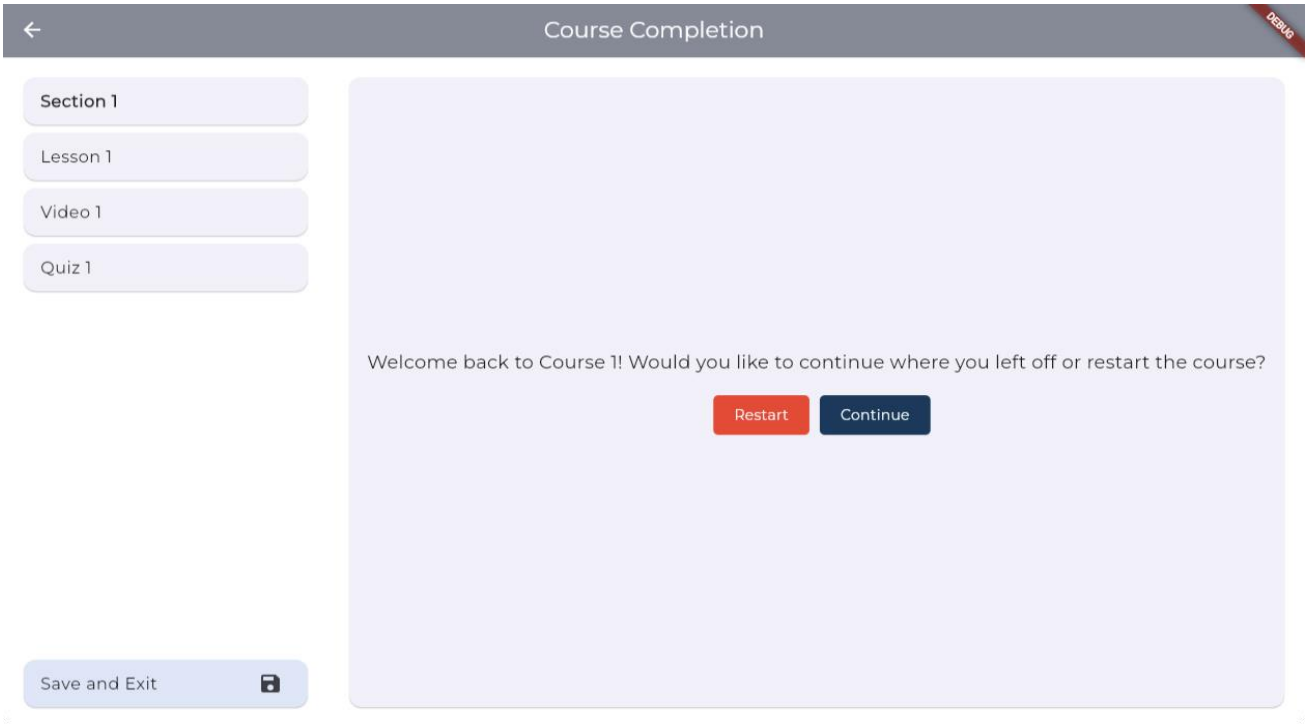

Figure 145 - Continue Where You Left Off [Code]

Course Feedback / Rating

Course satisfaction and user enjoyment is a large factor of this application's success, to effectively measure the quality of content on the platform - especially as it evolves past the scope of this project - there must be an effective mechanism for collective rating of courses. This will be done using a star rating system, from which an average rating is derived.

```
      Text('Please leave a rating:'),
      SizedBox(height: 10),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: List.generate(5, (index) {
          return GestureDetector(
            onTap: () {
              setState(() {
                selectedRating = index + 1;
              });
            },
            child: Icon(
              Icons.star,
              color: index < selectedRating ? Color.fromARGB(255, 255, 215, 0) : Colors.grey,
            ), // Icon
          ); // GestureDetector
        }), // List.generate
      ), // Row
    ],
  ), // Column
  actions: <Widget>[
    TextButton(
      onPressed: () {
        submitRating((widget.courseId), selectedRating);
        Navigator.of(context).pop();
        Navigator.of(context).pop();
```

Figure 146 - Course Rating Mechanism [Code]

Code for showing rating stars, for a user to submit a 1-5 rating for courses.

```
Future<void> submitRating(String courseId, int rating) async {
  try {
    final courseRef = FirebaseFirestore.instance.collection('courses').doc(courseId);
    await courseRef.update({'Rating': FieldValue.arrayUnion([rating])});

    print('Rating submitted successfully!');
  } catch (e) {
    print('Error submitting rating: $e');
  }
}
```

Figure 147 - Submit Rating [Code]

Submitting array of ratings to Firestore.

```dart
// Course model class
class Course {
  final String courseId;
  final String title;
  final String imageUrl;
  final int duration; // Duration in minutes
  final List<int> rating;
  final String category;
```

Figure 148 - Add Rating To Course Class [Code]

Updating class datatypes to include a list for rating.

```dart
Row(
  children: [
    Icon(Icons.access_time, size: 16),
    SizedBox(width: 4),
    Text('${course.duration} mins'),
    SizedBox(width: 10),
    Icon(Icons.star, size: 16),
    Text('${course.averageRating} / 5'),
    Spacer(),
```

Figure 149 - Course Rating On CourseTile Widget [Code]

```dart
double get averageRating {
  if (rating.isEmpty) return 0; // Handle case when there are no ratings
  int sum = rating.reduce((value, element) => value + element);
  return sum / rating.length; // Convert to double here
}
```

Figure 150 - Calculate Average Rating [Code]

Calculating and displaying average rating, all items in array added and / by final index.

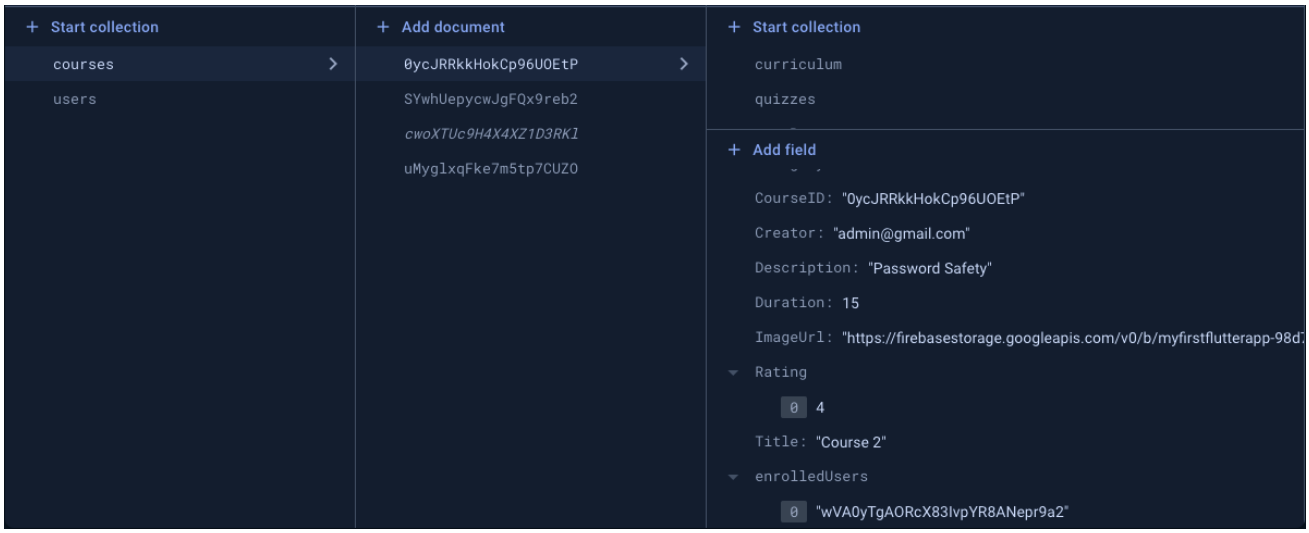

Figure 151 - Ratings Array In Firestore

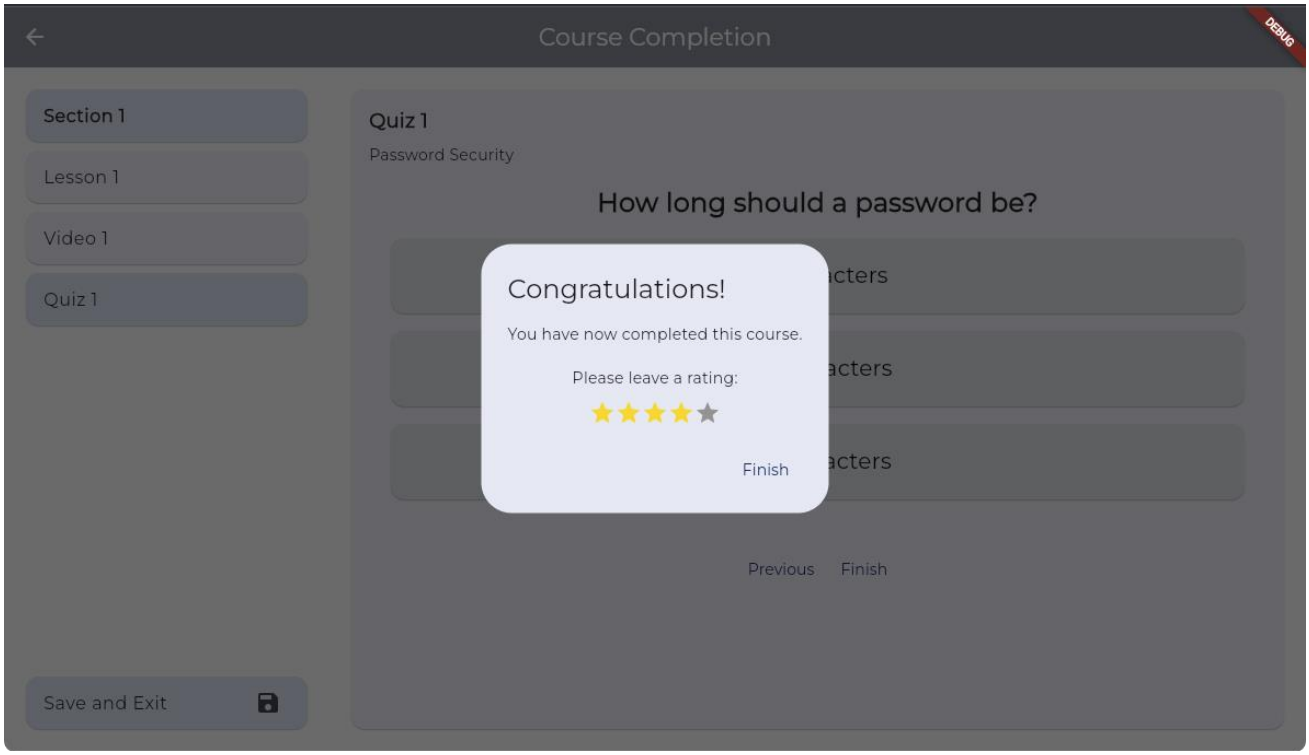

Figure 152 - Course Rating [UI]

The rating shows up as an intuitive and user-friendly card with 5 stars in

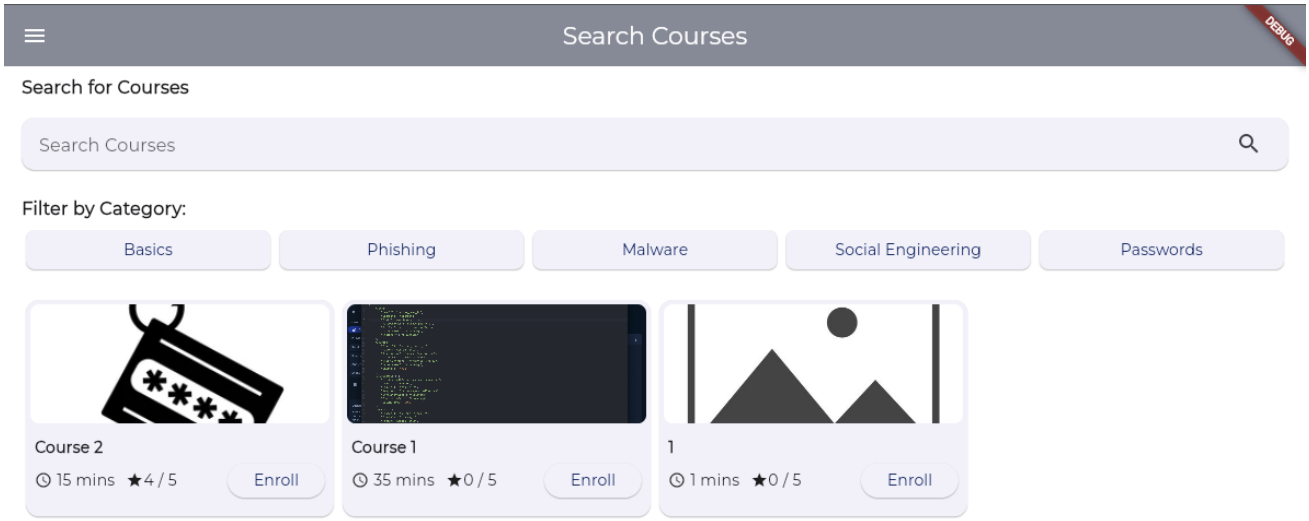

Figure 153 - Course Rating on CourseTile [UI]

Ratings update instantly on all of the course pages.

Course Searching / Filtering

In the application's future, it is not unfeasible to suggest that there could be hundreds of courses. For users to find a course which fits their needs, there should be a search and filter option to narrow down courses based on category or name.

```dart
final List<String> _categories = [
  'Phishing',
  'Malware',
  'Online Safety',
  'Social Engineering',
  'Passwords',
];
```

Figure 154 - Categories List [Code]

Creating a list of categories for a dropdown selector in the course creation page, this will allow courses to be one of 5 categories, this also allows courses to be filtered based on category.

```dart
DropdownButtonFormField<String?>(
  value: _selectedCategory,
  onChanged: (String? newValue) {
    setState(() {
      _selectedCategory = newValue!;
    });
  },
  items: [
    DropdownMenuItem<String?>(
      value: null,
      child: Text("Select a category..."), // Placeholder
    ), // DropdownMenuItem
    ..._categories.map<DropdownMenuItem<String?>>((String value) {
      return DropdownMenuItem<String?>(
        value: value,
        child: Text(value),
      ); // DropdownMenuItem
    }).toList(),
  ],
   decoration: InputDecoration(
    labelText: 'Category',
  ), // InputDecoration
  validator: (value) => value == null || value.isEmpty ? 'Please select a course category' : null,
), // DropdownButtonFormField
```

Figure 155 - Category Dropdown [Code]

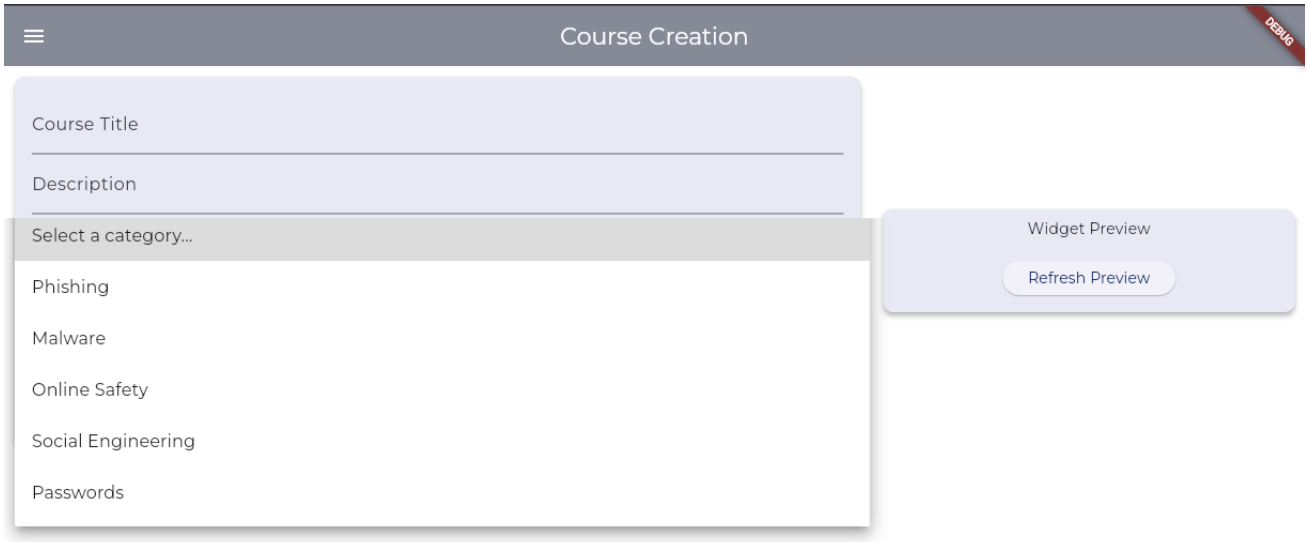

Figure 156 - Category Dropdown [UI]

The text form is now a drop-down selector.

```dart
if (snapshot.connectionState == ConnectionState.waiting) {
  return Center(child: CircularProgressIndicator());
}

if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
  return Center(child: Text('No courses available'));
}

List<Course> courses = snapshot.data!.docs
    .map((doc) => Course.fromFirestore(doc))
    .where((course) =>
        course.title.toLowerCase().contains(_searchController.text.toLowerCase()) &&
        (_selectedCategory == null || _selectedCategory!.isEmpty || course.category == _selectedCategory))
    .toList();

return GridView.builder(
  padding: const EdgeInsets.all(16.0),
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 4,
    childAspectRatio: 1.4,
  ), // SliverGridDelegateWithFixedCrossAxisCount
  itemCount: courses.length,
  itemBuilder: (context, index) {
    return CourseTile(course: courses[index]);
  },
```

Figure 157 - Filter Courses [Code]

Code to show only courses which match search criteria or filter selection.

```dart
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 16.0),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      Expanded(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 4.0),
          child: ElevatedButton(
            onPressed: () => _onCategorySelected('Basics'),
            style: ElevatedButton.styleFrom(
              padding: EdgeInsets.symmetric(vertical: 16.0), // Adjust padding as needed
              backgroundColor: _selectedCategory == 'Basics' ? textColor : null,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(8.0),
              ), // RoundedRectangleBorder
            ),
            child: Text('Basics'),
          ), // ElevatedButton
        ), // Padding
      ), // Expanded
```

Figure 158 - Filter Buttons [Code]

Button code for each category, add more options to have extra filters.

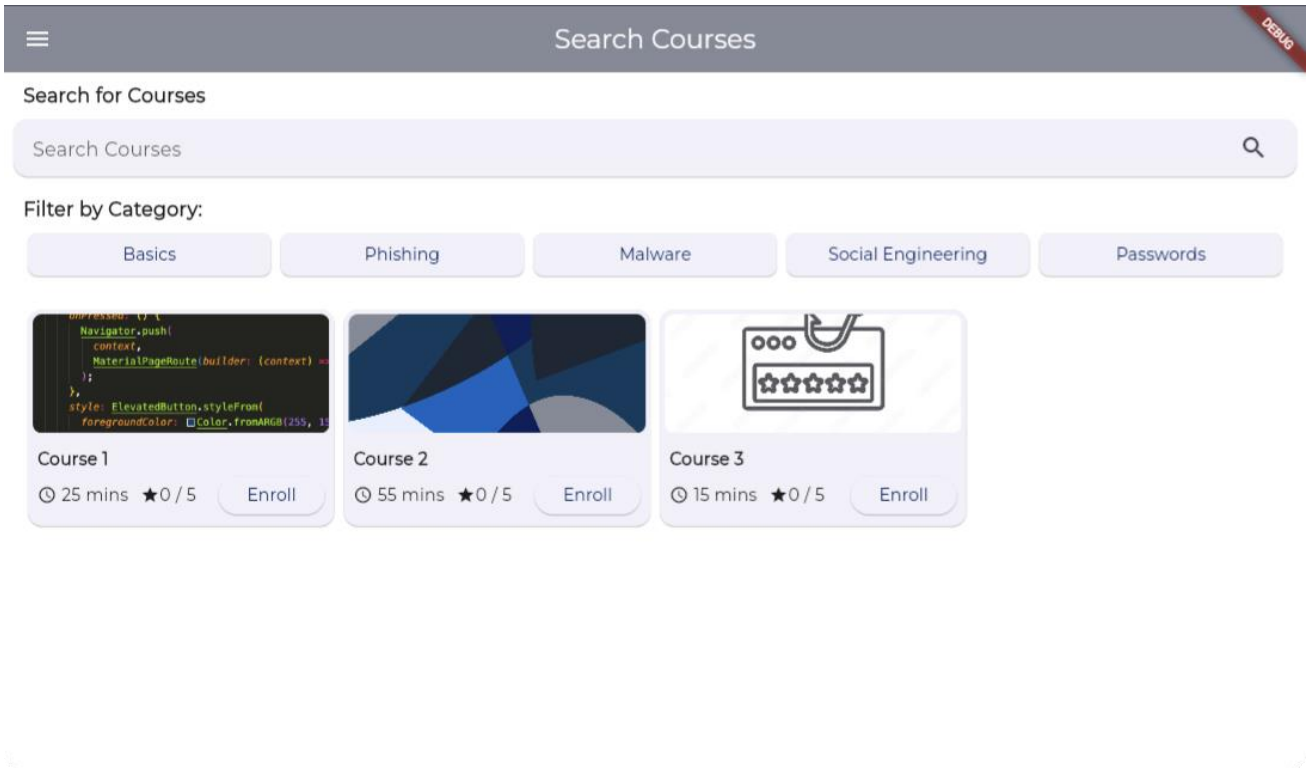

Figure 159 - Filter Courses [UI]

Final UI result, with 5 buttons for course filters, and a search bar for course titles.

Figure 160 - Search Courses [UI]

Searching for ‘course 1’.

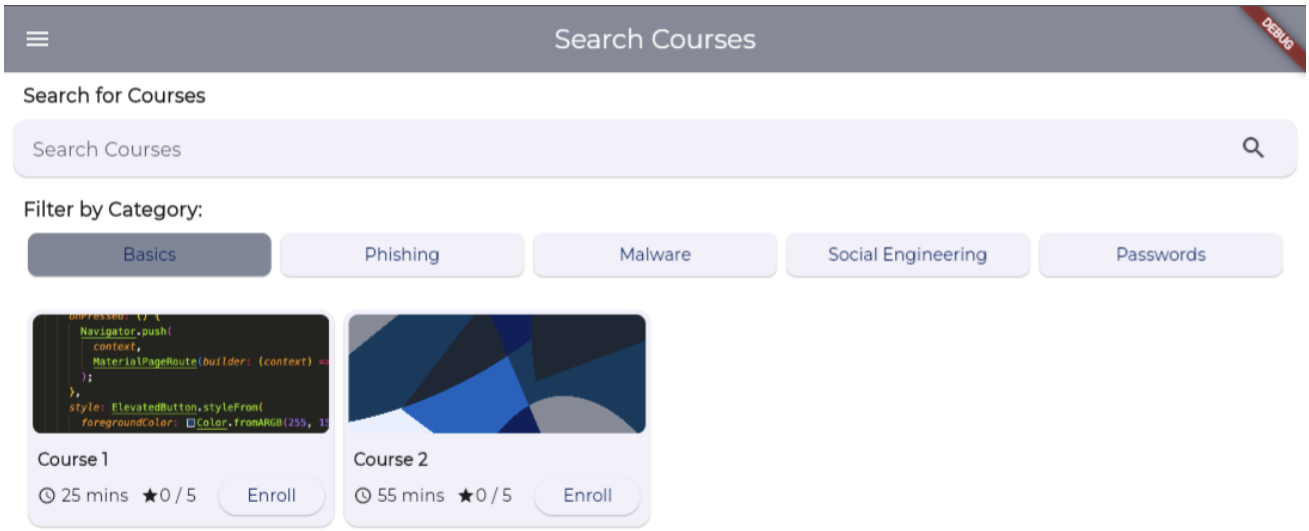

Figure 161 - Filter Courses [UI]

Filter by the category ‘basics’.

6.3.5 Testing

Feature	Test Description	Result
Enrolling in course	Enrolling in a course should open up the course completion page for said course.	Success
	All lessons and quizzes should be displayed.	Success
	Firebase should update to show users as enrolled in the course.	Success
	Users should be able to start a course from all pages where course tiles are shown.	Success
Saving Progress	userProgress subcollection should be created for each user for each course.	Success
	userProgress should correctly identify the quantity of lessons and quizzes in the course.	Success
	When a user finished a lesson or quiz their progress should be updated in the progress map.	Success
	When a user exits the course, a completion % should be generated.	Success
	When a user returns to a course they should continue from where they were.	Success
Feedback / Ratings	Users can leave a star rating when finishing a course.	Success
	Users should only be able to give one rating per course.	Success
	Ratings should be collated as an array and used to find an average value.	Fail
Completing Lessons	Lessons should show up in the order they were created in.	Success
	All content from a lesson should be pulled from Firebase and loaded into the application.	Success

	Based on lesson type the content shown should either be text based or a YouTube player	Success
	Video lessons should auto play the correct YouTube video when the lesson is moved to	Success
Completing Quizzes	Quizzes should load in all questions from Firestore and display them one at a time	Success
	The correct answer index should be correctly pulled from Firestore to validate the users' selection	Success
	When an answer is correct it should be highlighted green.	Success
	When an answer is incorrect it should be highlighted red, and an explanation given for the correct answer	Success
	Correct answers should be tracked to give a score fraction and % to be uploaded to Firestore	Success
	Completing a question should take the user into the next, unless it is the last question in which case it says 'finish'	Success
Searching for / filtering courses	The search page should allow users to see all courses	Success
	Filtering by category should show only courses in said category on the page, hiding others	Success
	Searching for courses should match the entered search string to any courses containing the string.	Success

Table 7 - Iteration 3 Testing

Fix for the Rating System

Due to a Firebase quirk, no more than 4 users can give ratings on each course before it maxes out. This is because the 'arrayUnion' which is used to add values to the ratings array cannot accept non-unique values – therefore the second time someone gives a rating of '4', it is not uploaded.

```
Future<void> submitRating(String courseId, int rating) async {
  try {
    final courseRef = FirebaseFirestore.instance.collection('courses').doc(courseId);
    await courseRef.update({'Rating': FieldValue.arrayUnion([rating])});

    print('Rating submitted successfully!');
  } catch (e) {
    print('Error submitting rating: $e');
  }
}
```

Figure 162 - Non-Working Array Union [Code]

```
Future<void> submitRating(String courseId, int rating) async {
  try {
    final courseRef = FirebaseFirestore.instance.collection('courses').doc(courseId);
    final courseDoc = await courseRef.get();

    if (courseDoc.exists) {
      final List<dynamic>? currentRatings = courseDoc.data()?['Rating']; //Fetch all ratings from fires

      List<int> updatedRatings = [];
      if (currentRatings != null) {
        updatedRatings.addAll(currentRatings.cast<int>());
      }
      updatedRatings.add(rating); //Add new rating to the list

      await courseRef.update({'Rating': updatedRatings}); //Re-send the updated list to firestore
```

Figure 163 - Working Array Splice [Code]

Solution: Pull out the full array, add the value and reupload said array to Firebase.

Figure 164 - Firebase Accepting Non-Unique Values

6.3.6 Review

In iteration 3, the e-learning platform meets all the functional requirements; integration of a course completion mechanism and several other new features and improvements have refined the learning experience allowing users to complete courses, track progress, provide feedback and easily search for courses which suit their learning needs. All these features work together to create a more engaging, accessible, and user-friendly e-learning environment.

6.4 Iteration 4

6.4.1 Features to Implement

- Dashboard stats (FR11, FR13)
- Saved Courses (FR6) (NFR5)
- Streak Feature (FR16)
- Profile Picture (FR3)

6.4.2 Task Breakdown

The feature implementation has been broken down into actionable tasks to be used throughout the development process.

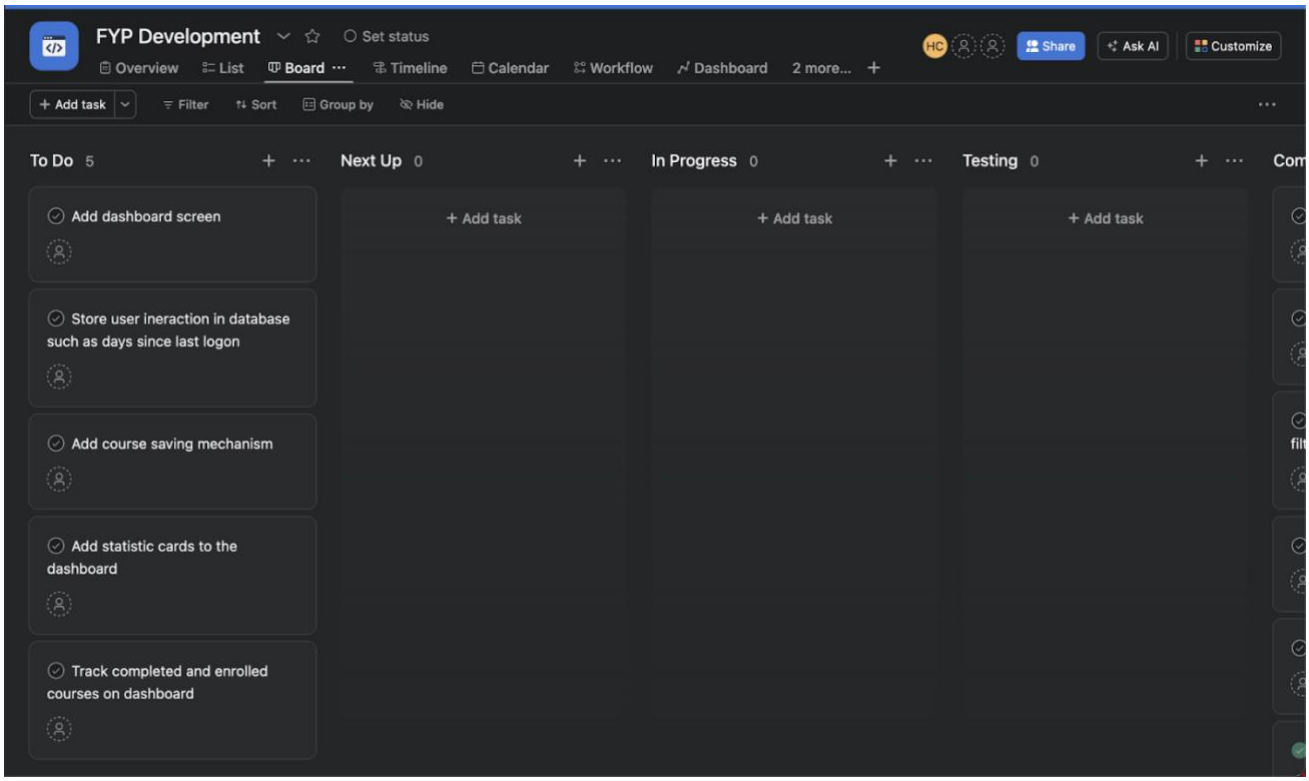

Figure 165 - Iteration 4 Task Breakdown

6.4.3 Design

The design of these features will incorporate features from Figures 19 and 30 in the design section, however, will mostly focus on the less critical requirements identified as ‘should’ or ‘could’ haves in chapter 4.

6.4.4 Implementation

Save Courses

As mentioned in the literature review, with the andragogical approach having a strong focus on learning when you can fit it into busy schedules, there should be a feature which allows users to save courses they think they will benefit from, to come back to later when they are able to complete it. This will be done using a bookmark icon on each course which moves the courses to a different section where they can be easily accessed.

```
      Positioned(
        top: 7,
        right: 7,
        child: Container(
          decoration: BoxDecoration(
            color: Color.fromARGB(255, 224, 220, 228),
            shape: BoxShape.circle,
          ), // BoxDecoration
          child: IconButton(
            icon: Icon(
              _isBookmarked ? Icons.bookmark : Icons.bookmark_border,
              color: _isBookmarked ? highlightColor : null,
            ), // Icon
            onPressed: () {
              _toggleBookmark(context);
            },
          ), // IconButton
        ), // Container
      ), // Positioned
    ],
  ), // Stack
); // Card
```

Figure 166 - Bookmark Icon [Code]

Adding the bookmark icon as a button to the course tile widget.

```dart
void _toggleBookmark(BuildContext context) async {
  try {
    final user = FirebaseAuth.instance.currentUser;
    if (user != null) {
      final userDoc = FirebaseFirestore.instance.collection('users').doc(user.uid);

      DocumentSnapshot userSnapshot = await userDoc.get();
      Map<String, dynamic>? userData = userSnapshot.data() as Map<String, dynamic>?;
      if (userData != null) {
        List<String> bookmarkedCourses = List<String>.from(userData['bookmarkedCourses'] ?? []);
        print (bookmarkedCourses);

        // Check if the course ID is already bookmarked
        if (bookmarkedCourses.contains(widget.course.courseId)) {
          // If the course is already bookmarked, remove it
          bookmarkedCourses.remove(widget.course.courseId);
        } else {
          // If the course is not bookmarked, add it
          bookmarkedCourses.add(widget.course.courseId);
        }

        await userDoc.update({'bookmarkedCourses': bookmarkedCourses});

        setState(() {
          _isBookmarked = !_isBookmarked;
        });
      }
```

Figure 167 - Add or Remove Bookmark [Code]

This code allows the toggling of bookmark status.

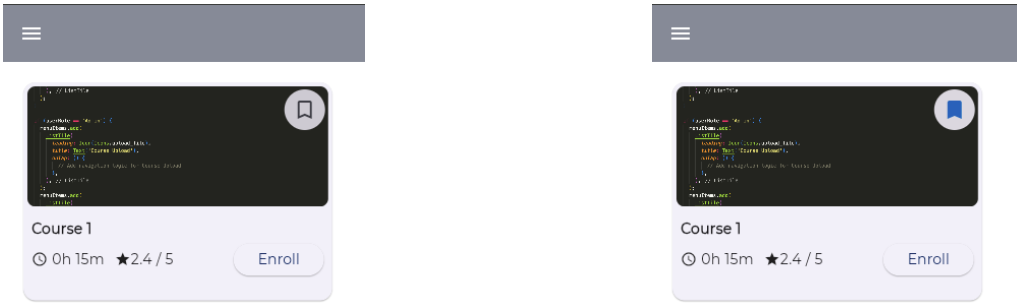

Figure 168 - Bookmark Toggle [UI]

The result is a little save icon which can be toggled on (blue), or off (grey).

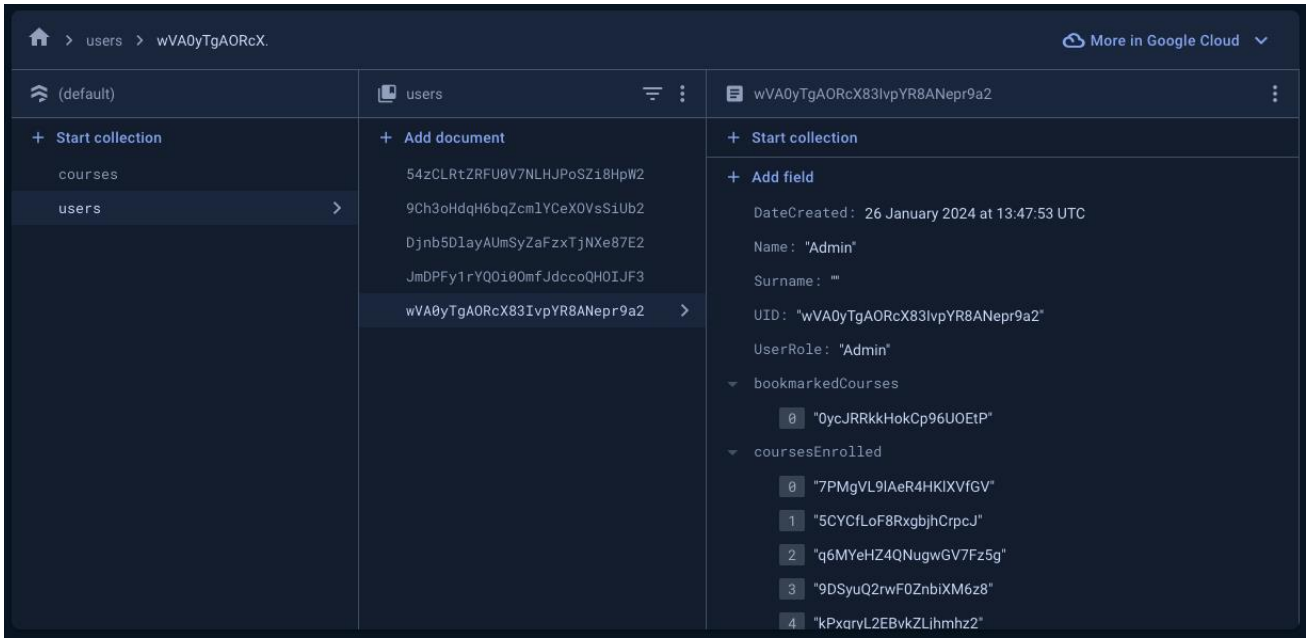

Figure 169 - Firebase User Bookmarked Field

Firestore updates bookmarked courses, saving an array for each user of bookmarked courses.

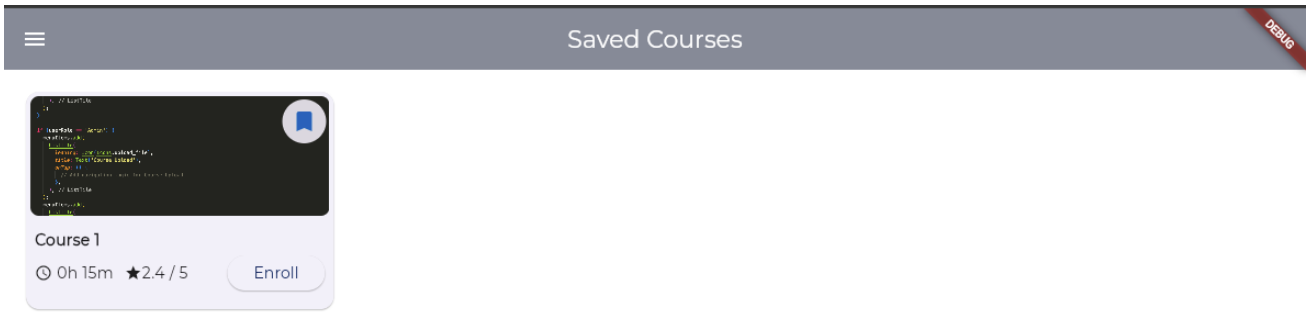

Figure 170 - Displaying Saved Courses [UI]

A users bookmarked courses will display in the saved section.

Dashboard

The dashboard screen will streamline the user experience by providing immediate access to relevant and critical information, improving efficiency and user satisfaction. In the future, the dashboard can be expanded to include additional statistics or features such as awards.

```
FutureBuilder<DocumentSnapshot>(
  future: FirebaseFirestore.instance.collection('users').doc(_currentUser!.uid).get(),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return CircularProgressIndicator();
    }
    if (!snapshot.hasData || !snapshot.data!.exists) {
      return Text('User data not found.');
    }
    final inactiveTime = snapshot.data!['inactiveTime'] ?? 0;
    return Card(
      child: Padding(
        padding: EdgeInsets.all(16),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Text(
              'Days Since Last Sign-In:',
              style: TextStyle(fontSize: 18),
            ), // Text
            Text(
              '$inactiveTime',
```

Figure 171 - Dashboard Statistics [Code]

```
void updateUserLastLogin(User? user) async {
  if (user != null) {
    final userRef = FirebaseFirestore.instance.collection('users').doc(user.uid);

    // Fetch the lastLogin timestamp from Firestore
    final userDoc = await userRef.get();
    DateTime lastLogin = userDoc['lastLogin'].toDate();

    // Calculate the inactive time (time since the last login)
    Duration inactiveTime = DateTime.now().difference(lastLogin);

    // Update the lastLogin field to today's date
    await userRef.update({
      'lastLogin': DateTime.now(),
      'inactiveTime': inactiveTime.inDays, // Store inactive time in days
    }).then((_) {
      print('Last login updated successfully');
    }).catchError((error) {
      print('Error updating last login: $error');
    });
```

Figure 172 - Updating Statistics [Code]

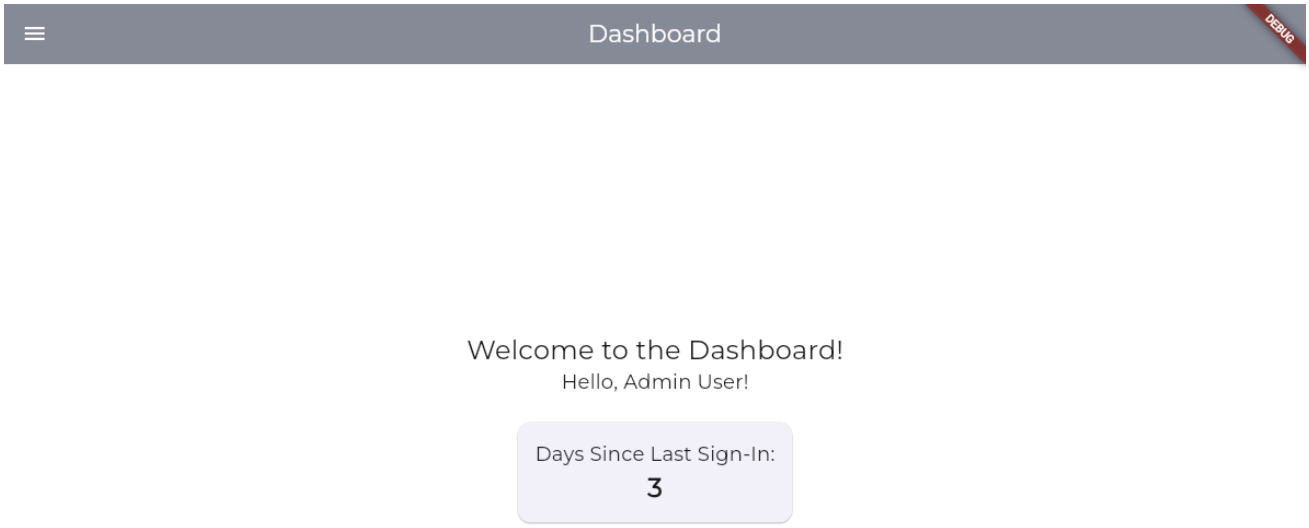

Figure 173 – Dashboard Last Sign In [UI]

Show time since last sign in.

```dart
SizedBox(height: 20),
Card(
  child: Padding(
    padding: EdgeInsets.all(16),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Text(
          'Courses Currently Enrolled In:',
          style: TextStyle(fontSize: 18),
        ), // Text
        FutureBuilder<List<String>>(
          future: _fetchCourseTitles(coursesEnrolled),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return CircularProgressIndicator();
            }
            final courses = snapshot.data ?? [];
            return Column(
              children: courses.map((title) => Text(title)).toList(),
            ); // Column
          },
```

Figure 174 - Display Enrolled Courses [Code]

Show currently enrolled courses.

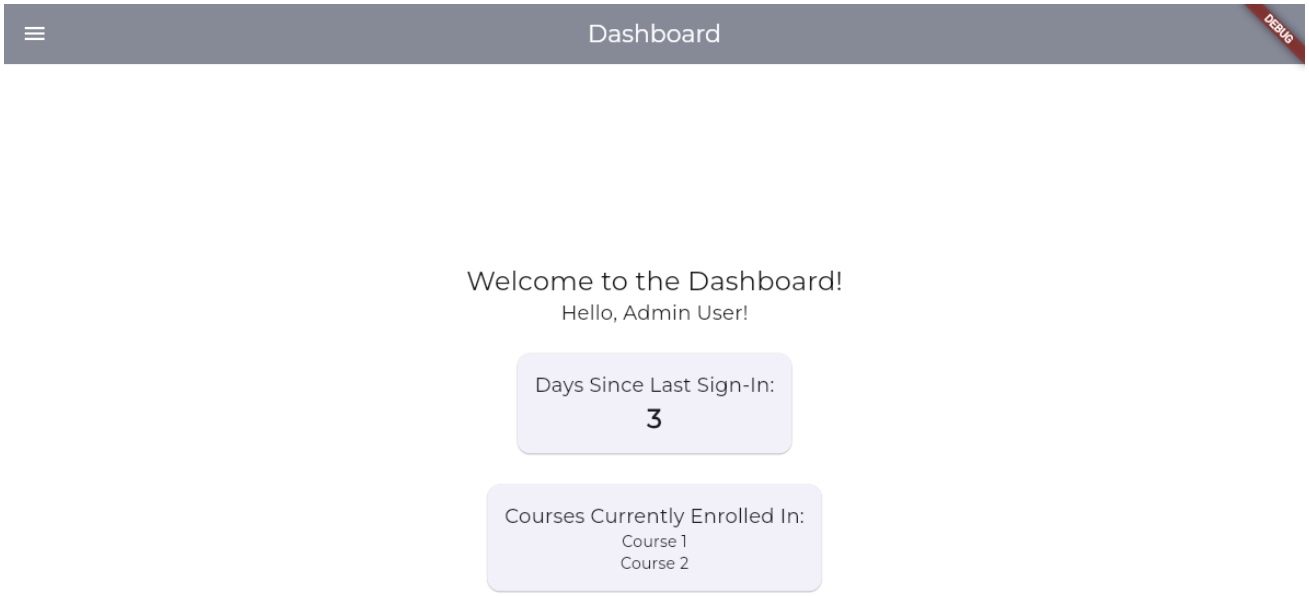

Figure 175 – Dashboard Enrolled Courses [UI]

Show courses currently enrolled in.

```dart
// Get the last login timestamp and calculate the inactive time
DateTime lastLogin = userDoc['lastLogin'].toDate();
Duration inactiveTime = DateTime.now().difference(lastLogin);

// Check if the streak field exists in the document
int streak = userDoc['streak'] ?? 0;

// Check if the last login was yesterday or today
DateTime today = DateTime.now();
DateTime yesterday = DateTime.now().subtract(Duration(days: 1));

if (lastLogin.year == today.year &&
    lastLogin.month == today.month &&
    lastLogin.day == today.day) {
  // Last login was today, so streak remains the same
} else if (lastLogin.year == yesterday.year &&
    lastLogin.month == yesterday.month &&
    lastLogin.day == yesterday.day) {
  // Last login was yesterday, so increase streak by 1
  streak += 1;
} else {
  // Last login was neither yesterday nor today, reset streak to 1
  streak = 1;
}
```

Figure 176 - Calculate Streak [Code]

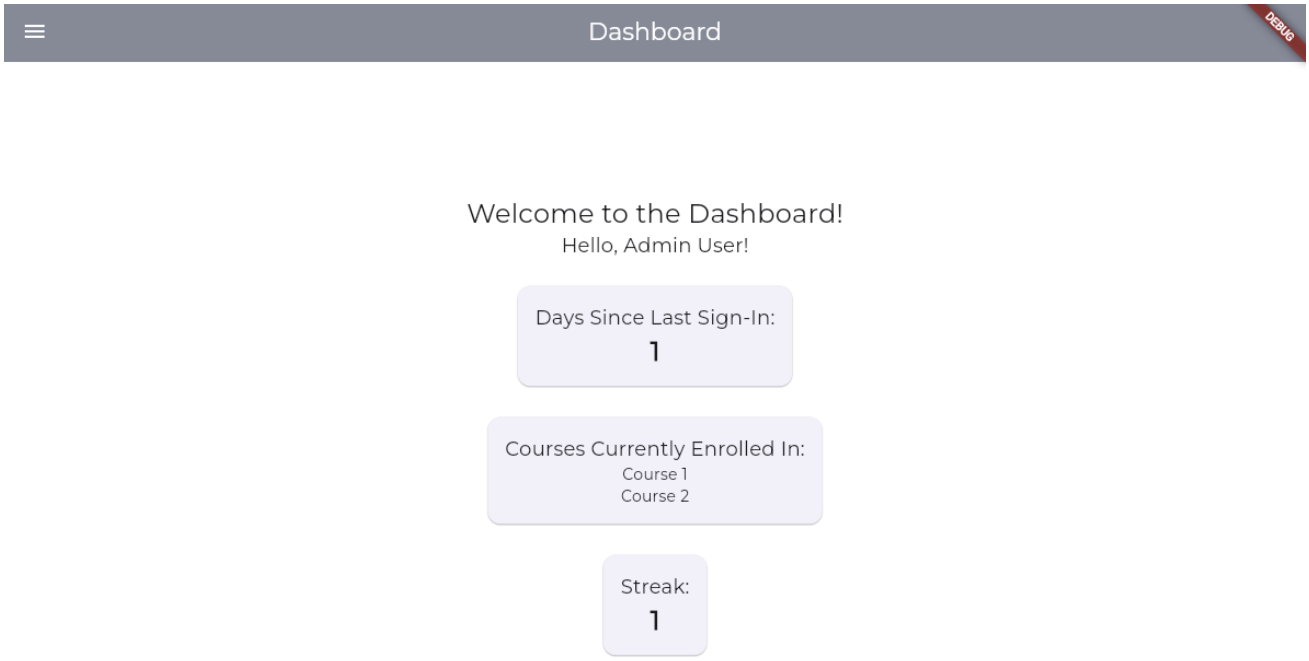

Figure 177 - Dashboard Streak [UI]

Show streak on dashboard.

```dart
Future<List<String>> _fetchCompletedCourses() async {
  final completedCourses = <String>[];
  final userDoc = FirebaseFirestore.instance.collection('users').doc(_currentUser!.uid);
  final progressDoc = userDoc.collection('userProgress');
  final querySnapshot = await progressDoc.where('progressValue', isEqualTo: 100).get();

  for (final doc in querySnapshot.docs) {
    final courseId = doc.id; // Assuming course ID is the document ID
    final courseSnapshot = await FirebaseFirestore.instance.collection('courses').doc(courseId).get();
    if (courseSnapshot.exists) {
      final title = courseSnapshot.data()?['Title'] ?? 'Unknown Course';
      completedCourses.add(title);
    }
  }
  return completedCourses;
}
}
```

Figure 178 - Fetch Completed Courses [Code]

Fetch completed courses.

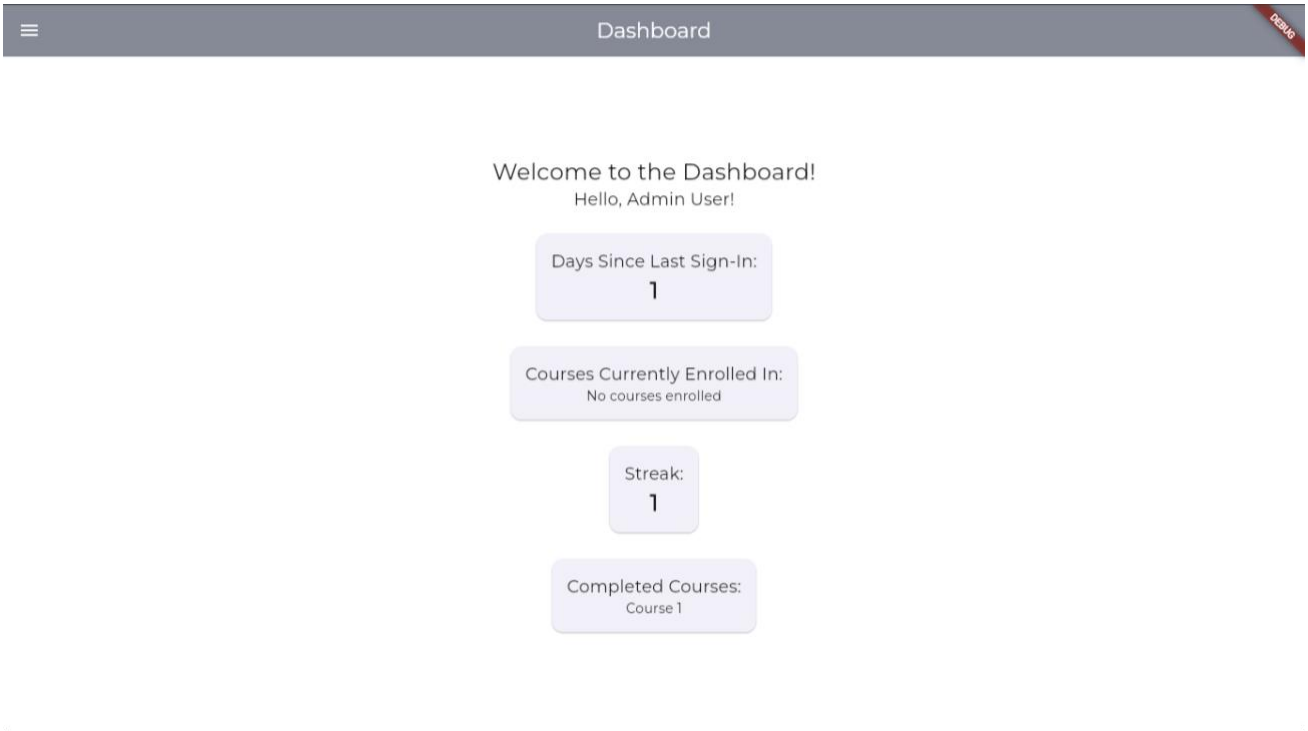

Figure 179 - Dashboard Completed Courses [UI]

Display completed courses on dashboard.

```
InkWell(
  onTap: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => CourseCompletionScreen(courseId: course['id'])),
    );
  },
  child: Text('Continue Course'),
), // InkWell
```

Figure 180 - Continue Course Button [Code]

Adding continue button, this will allow users to continue courses they have already started from the dashboard.

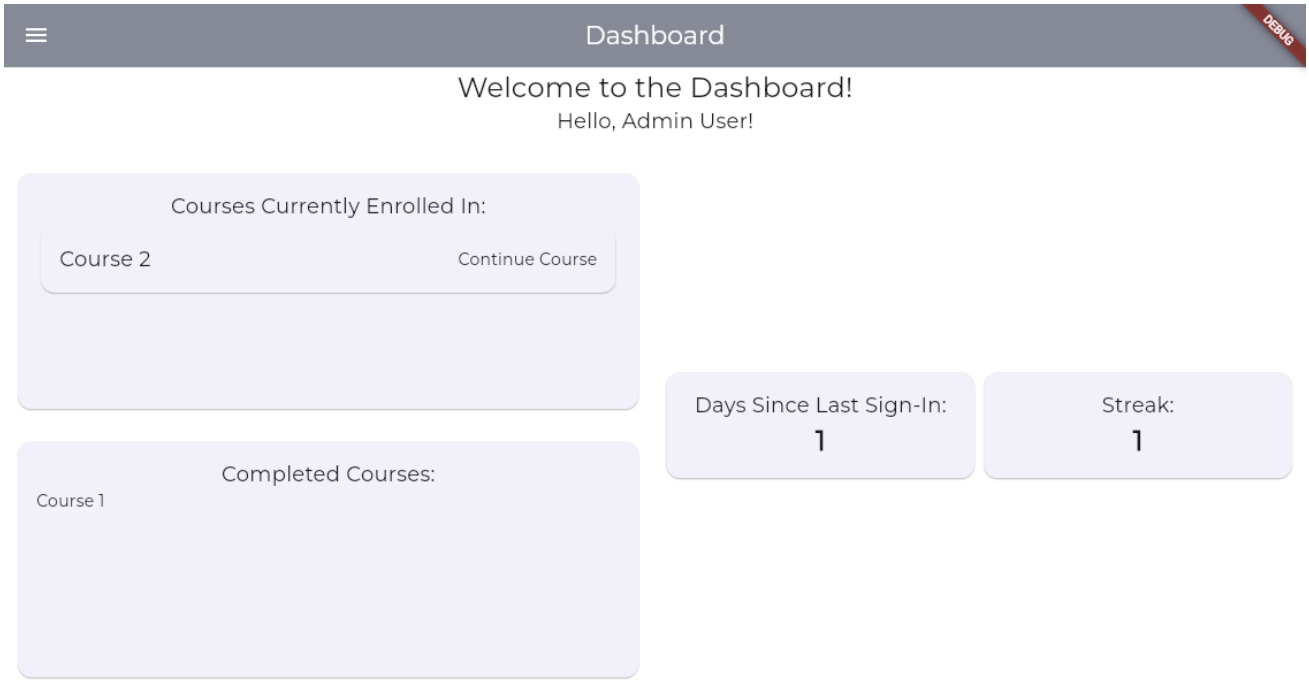

Figure 181 - New Dashboard With Continue Course [UI]

Updated UI with the ability to continue a course from dashboard.

Streak Feature

A strong element of enforcing learning in adults lies in continuous development and reinforcement; implementing an incentive to return daily, such as a streak, has proven successful in other applications. In this application the users' streak will be stored in their file and viewed in the dashboard panel. In the future however, an automated notification system could be set up to alert users that they have not accessed the platform today.

```
Future<bool> signUpWithEmailAndPassword(String email, String password, String name, String surname) async
  try {
    UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );

    User? user = userCredential.user;
    if (user != null) {
      // Update user's display name
      await user.updateDisplayName("$name $surname");

      // Store user data in Firestore
      CollectionReference usersCollection = FirebaseFirestore.instance.collection('users');
      await usersCollection.doc(user.uid).set({
        'UID' : user.uid,
        'email': email,
        'Name': name,
        'Surname': surname,
        'DateCreated': FieldValue.serverTimestamp(),
        'UserRole': 'Student', // Default to student user role
        'lastLogin' : DateTime.now(),
      });
    }
```

Figure 182 - Adding LastLogin to User [Code]

Adding the lastLogin value to the user collection on signup.

```dart
void updateUserLastLogin(User? user) async {
  if (user != null) {
    final userRef = FirebaseFirestore.instance.collection('users').doc(user.uid);

    // Fetch the lastLogin timestamp from Firestore
    final userDoc = await userRef.get();
    DateTime lastLogin = userDoc['lastLogin'].toDate();

    // Calculate the inactive time (time since the last login)
    Duration inactiveTime = DateTime.now().difference(lastLogin);

    // Update the lastLogin field to today's date
    await userRef.update({
      'lastLogin': DateTime.now(),
      'inactiveTime': inactiveTime.inDays, // Store inactive time in days
    }).then((_) {
      print('Last login updated successfully');
    }).catchError((error) {
      print('Error updating last login: $error');
    });
```

Figure 183 - Update LastLogin [Code]

Update lastlogin on sign in, and track days since last sign in (existing lastlogin value - todays date).

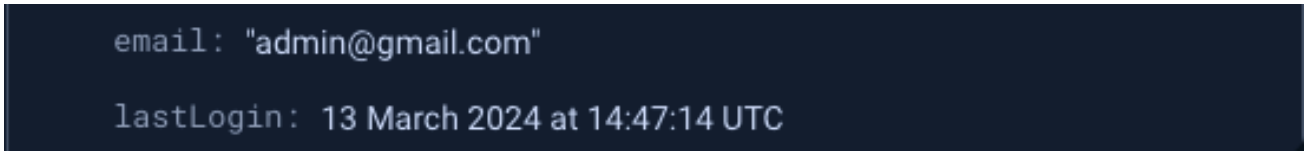

Figure 184 – Firebase LastLogin

Result in Firebase.

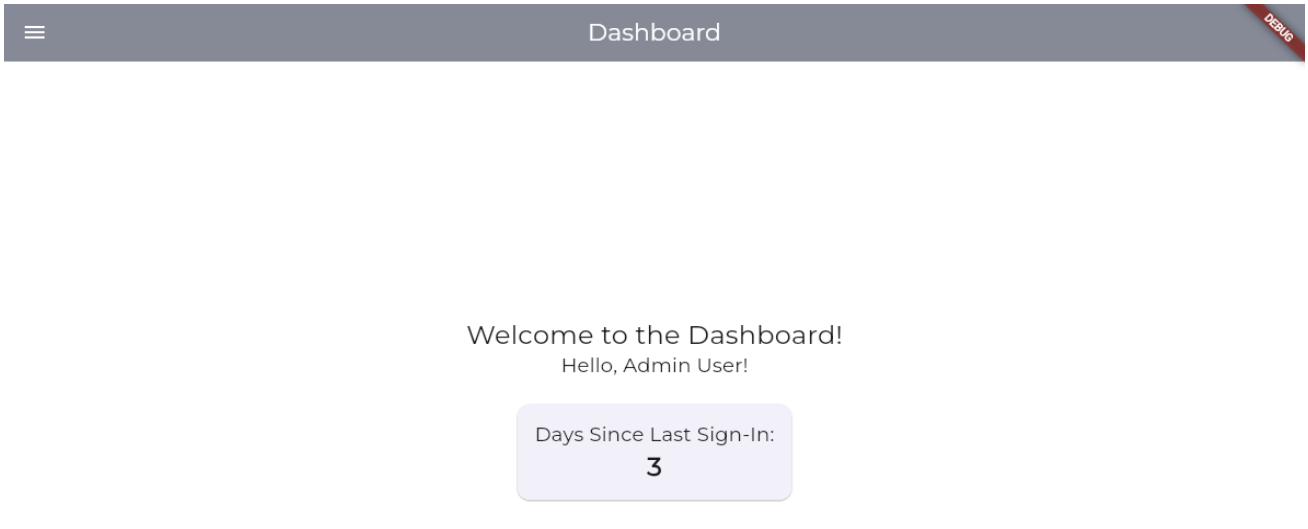

Figure 185 - Streak On Dashboard [UI]

Result on the dashboard.

```dart
// Get the last login timestamp and calculate the inactive time
DateTime lastLogin = userDoc['lastLogin'].toDate();
Duration inactiveTime = DateTime.now().difference(lastLogin);

// Check if the streak field exists in the document
int streak = userDoc['streak'] ?? 0;

// Check if the last login was yesterday or today
DateTime today = DateTime.now();
DateTime yesterday = DateTime.now().subtract(Duration(days: 1));

if (lastLogin.year == today.year &&
    lastLogin.month == today.month &&
    lastLogin.day == today.day) {
  // Last login was today, so streak remains the same
} else if (lastLogin.year == yesterday.year &&
    lastLogin.month == yesterday.month &&
    lastLogin.day == yesterday.day) {
  // Last login was yesterday, so increase streak by 1
  streak += 1;
} else {
  // Last login was neither yesterday nor today, reset streak to 1
  streak = 1;
}
```

Figure 186 - Calculating Streak [Code]

Calculate users' streak.

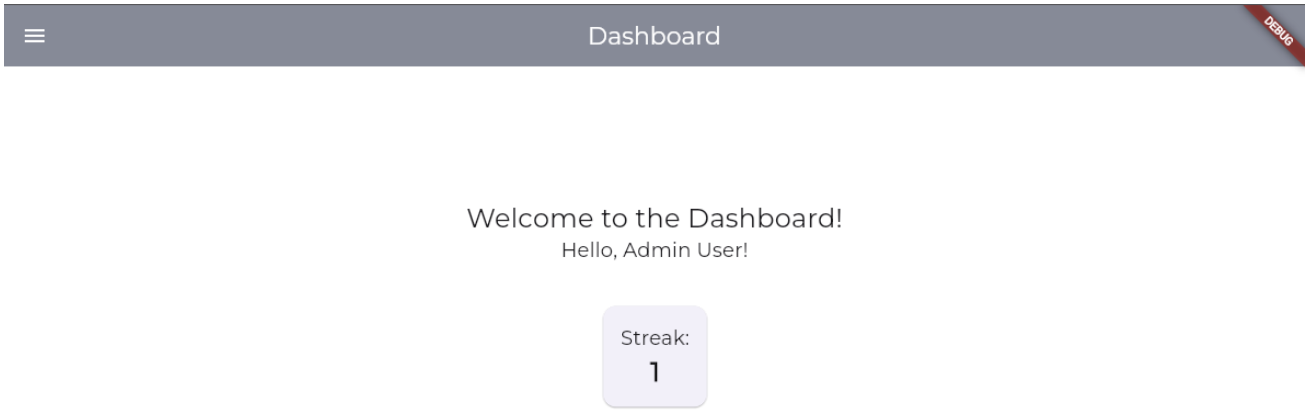

Figure 187 - Dashboard Streak [UI]

Added streak to dashboard.

Profile Pictures

Users will be able to select one of 6 avatars, as well as a background colour to make up their profile picture. The ability to customise user profiles enhances user satisfaction and connection to the platform. In the future it could also allow users to connect with each other.

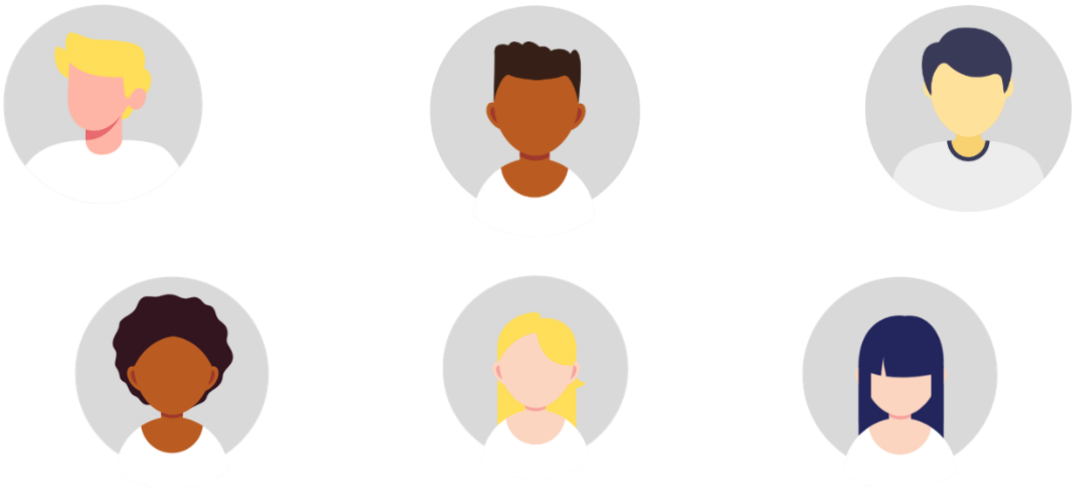

Figure 188 - Avatar Options

These are the 6 standard options which will be given for user avatars, a user can choose a combination of these 6 avatars and the 6 colour backgrounds below. Adding a sense of customisation and identity to the platform is hoped to increase engagement, this could be expanded on in future work to add other students.

Figure 189 - Avatar Background Colours

```dart
void _saveProfileSelection(int avatarIndex, int colorIndex) {

  String userId = _currentUser!.uid;
  String avatarUrl = '';

  print('Selected Avatar: $avatarIndex, Selected Color: $colorIndex');
  //Each avatar has 6 colors, so we can determine the avatar and color based on the index
  switch (avatarIndex) {
    case 0:
      switch (colorIndex) {
        case 0: // Avatar 1, Color 1
          avatarUrl = 'https://firebasestorage.googleapis.com/v0/b/myfirstflutterapp-98d7e.appspot.com
          break;
        case 1: // Avatar 1, Color 2
          avatarUrl = 'https://firebasestorage.googleapis.com/v0/b/myfirstflutterapp-98d7e.appspot.com
          break;
        case 2: // Avatar 1, Color 3
          avatarUrl = 'https://firebasestorage.googleapis.com/v0/b/myfirstflutterapp-98d7e.appspot.com
          break;
        case 3: // Avatar 1, Color 4
          avatarUrl = 'https://firebasestorage.googleapis.com/v0/b/myfirstflutterapp-98d7e.appspot.com
          break;
```

Figure 190 - Firebase Storage URL Links [Code]

URLs to Firebase storage of different images.

```dart
Future<void> uploadAvatarUrlToFirestore(String userId, String avatarUrl) async {
  try {
    DocumentReference userRef = FirebaseFirestore.instance.collection('users').doc(userId);
      await userRef.update({
        'AvatarURL': avatarUrl
      });
      print('Avatar URL uploaded to Firestore successfully.');
    } catch (error) {
      print('Error uploading avatar URL to Firestore: $error');
  }
```

Figure 191 - Firebase Upload Avatar Choice

```dart
if (snapshot.connectionState == ConnectionState.waiting) {
  return CircleAvatar(
    backgroundColor: Colors.white,
    child: CircularProgressIndicator(),
  ); // CircleAvatar
} else if (snapshot.hasData) {
return ClipOval(
  child: Image.network(
    snapshot.data!,
    fit: BoxFit.cover,
    width: 60, // Adjust size as needed
    height: 60, // Adjust size as needed
  ), // Image.network
); // ClipOval
```

Figure 192 - Navigation Pane Avatar [Code]

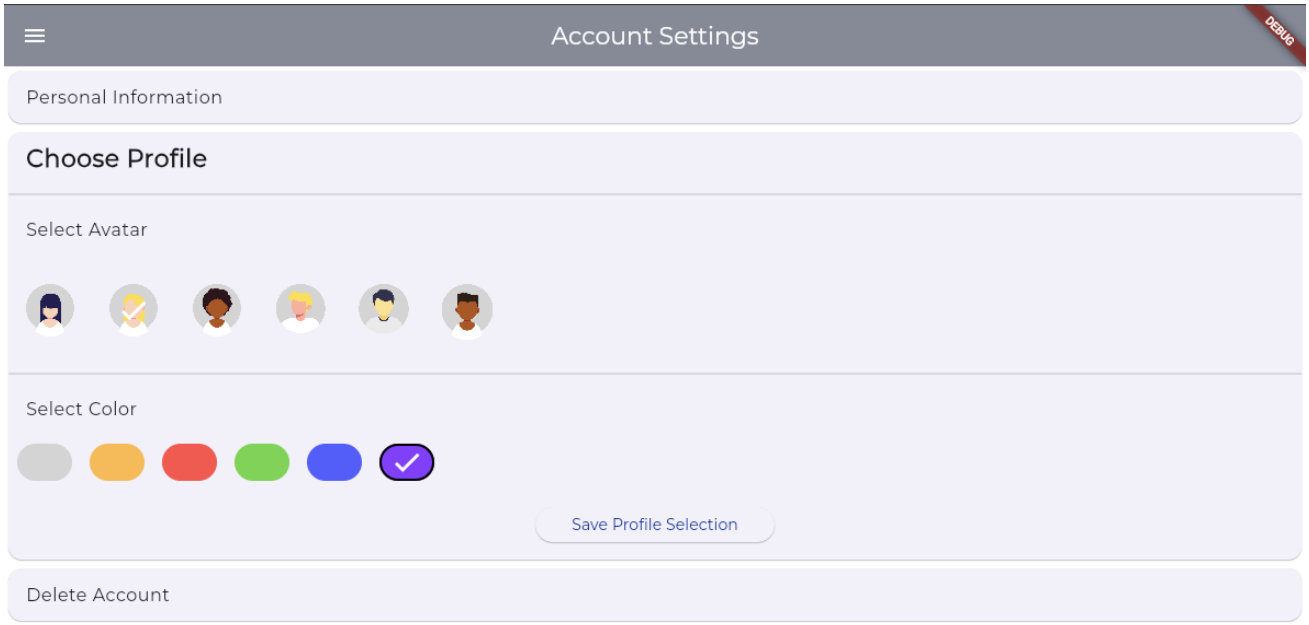

Figure 193 - Avatar Selection [UI]

UI for avatar selection

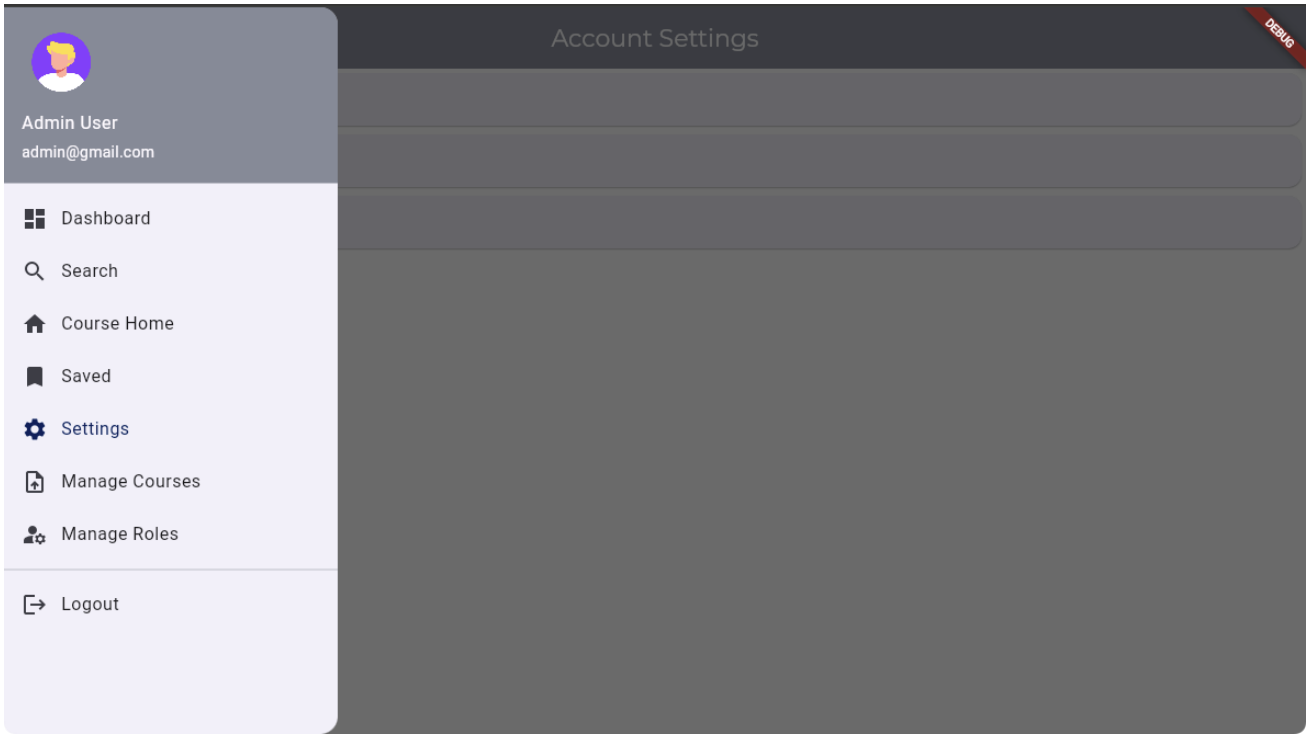

Figure 194 - Avatar on Navigation Pane [UI]

Result on the navigation pane.

Demo Content
For further testing, and demonstration purposes, a collection of courses will be added to the platform, one for each category. These courses will create the foundation for the e-learning platform, and it can be continued from them.

Course topics:

- Basics - Dark Web
- Online Safety - Keeping Safe Online
- Passwords - Creating Strong Passwords
- Malware - Malware Basics
- Phishing - Phishing for gold
- Social Engineering - How attackers steal from you.

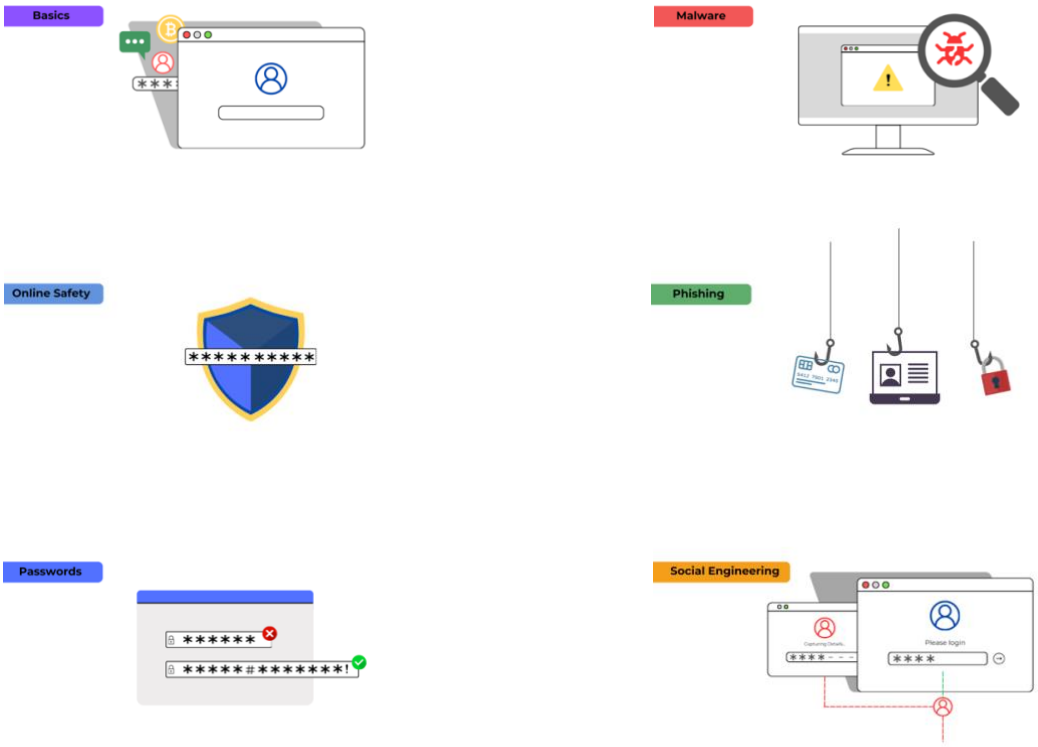

Figure 195 - Course Icons

These are the 6 icons, made specifically for each course. The icon will be the first impression a user gets of the quality of the course as well as the content. Utilising interesting graphics will hopefully lead to a higher rate of participation.

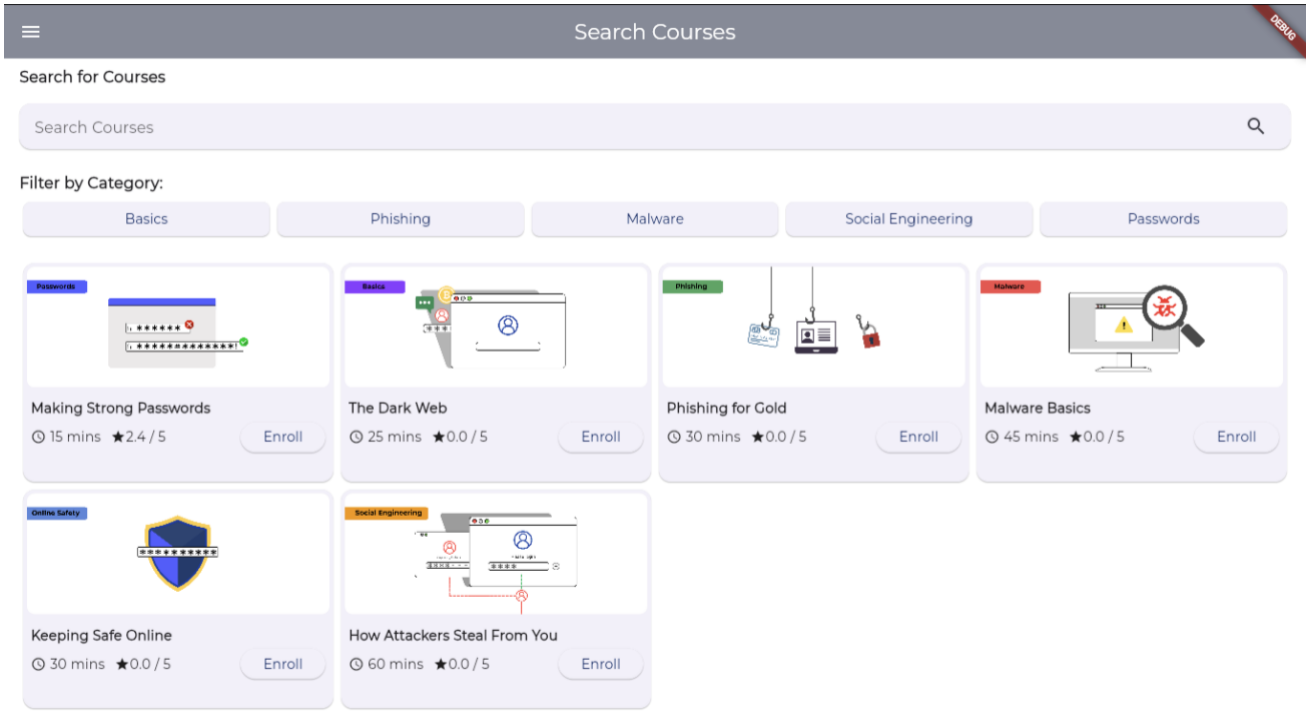

Figure 196 - All Courses Showing [UI]

The final result, the search page showing all courses.

6.4.5 Testing

Feature	Test Description	Result
Saving Courses	Bookmarking a course should add the courseID to an array of bookmarked courses in the User document.	Success
	Saved courses should appear on the saved page.	Success
	Un-bookmarking a course should remove it from the database.	Success
	Un-bookmarking a course should remove it from the saved courses screen.	Success
Dashboard	Courses users enrol in show up on the dashboard.	Success
	Once a course is completed it shows up on the dashboard as completed.	Success
	Courses can be continued from the dashboard.	Success
	Streaks and inactive days show up on dashboard.	Success
Streak feature	When a user signs in multiple days in a row, a database value 'streak' is increased by 1.	Success
	If a user doesn't sign in for 24 hours, the streak is reset to 0.	Success
	Every time a user logs in, lastLogin is updated in Firebase.	Success
	Streak and inactive days show up on the dashboard.	Success
Profile / Avatar selection	Users can choose between 6 avatars and colours.	Success
	Selecting a combination of avatar and colour will set that as your profile image.	Success
	Profile image shows on the app bar on the left of the screen.	Success

Table 8 - Iteration 4 Testing

6.4.6 Review

Iteration 4, the final iteration in this stage of development, has focussed on implementing functionalities which, while less critical, significantly contribute to a more personalised and engaging learning experience. The newly added features include dashboard statistics, saved courses, a streak feature for continuous learning encouragement, and customizable profile pictures. By focusing on the learners' experience outside of the core educational content, this iteration contributes to building a more interactive and user-friendly platform that encourages regular participation and lifelong learning.

6.5 Summary

This chapter has provided an in-depth overview of the implementation process across the application's development lifecycle, along with the outcomes observed during testing. It highlights the application of methodology, adherence to requirements, and integration of design principles, while also delving into the decision-making process throughout development. The primary aim of the implementation phase was to address all features identified as 'must have' and 'should have' in the requirements chapter which was largely successful. Despite some complexities and errors, the overall execution of the implementation phase followed closely to the agile methodology and allowed for timely completion. With implementation complete, the project is now in its final form to be analysed for its completeness against the original objectives, requirements, and existing solutions.

Chapter 7
Testing

7.1 Unit Test results

During the implementation and development of the project, regular testing was performed at the end of each iteration stage, adhering to the principles of an Agile project methodology. This iterative testing strategy ensured that every new feature and modification underwent testing and evaluation for functionality, usability, performance, and scalability before being integrated into the main codebase.

This approach to testing stems from Agiles' emphasis on adaptability, continuous improvement, and early detection of issues, which aligns with this project's goals for a high-quality output in a shorter than usual time frame. By having unit tests as an integral part of each iteration, errors could easily be rectified before they propagate to other iterations. This has not only streamlined the development process, by allowing for quick adjustments and preventing the accumulation of unresolved issues, but also significantly reduced the risk of major setbacks in later stages of the project.

- Iteration 1 15/18 (Google sign on fails) 83%.
- Iteration 2 24/26 (Indexing issues) 92%.
- Iteration 3 24/25 (Rating system) 96%.
- Iteration 4 16/16 (no issues) 100%.

Overall, 92.75% pass rate on tests.

7.2 Whole Test Results

Following the iterative testing completed during implementation, the whole application should now undergo comprehensive system-wide testing against a series of predefined use cases. This phase of testing will be crucial to validate the platform's functionality, usability, and overall user experience in real-world scenarios. For instance, testing the end-to-end process of a new user signing up, enrolling in, and completing a course will assess not only the individual components of account creation, course enrolment and content interaction but also how seamlessly these features integrate.

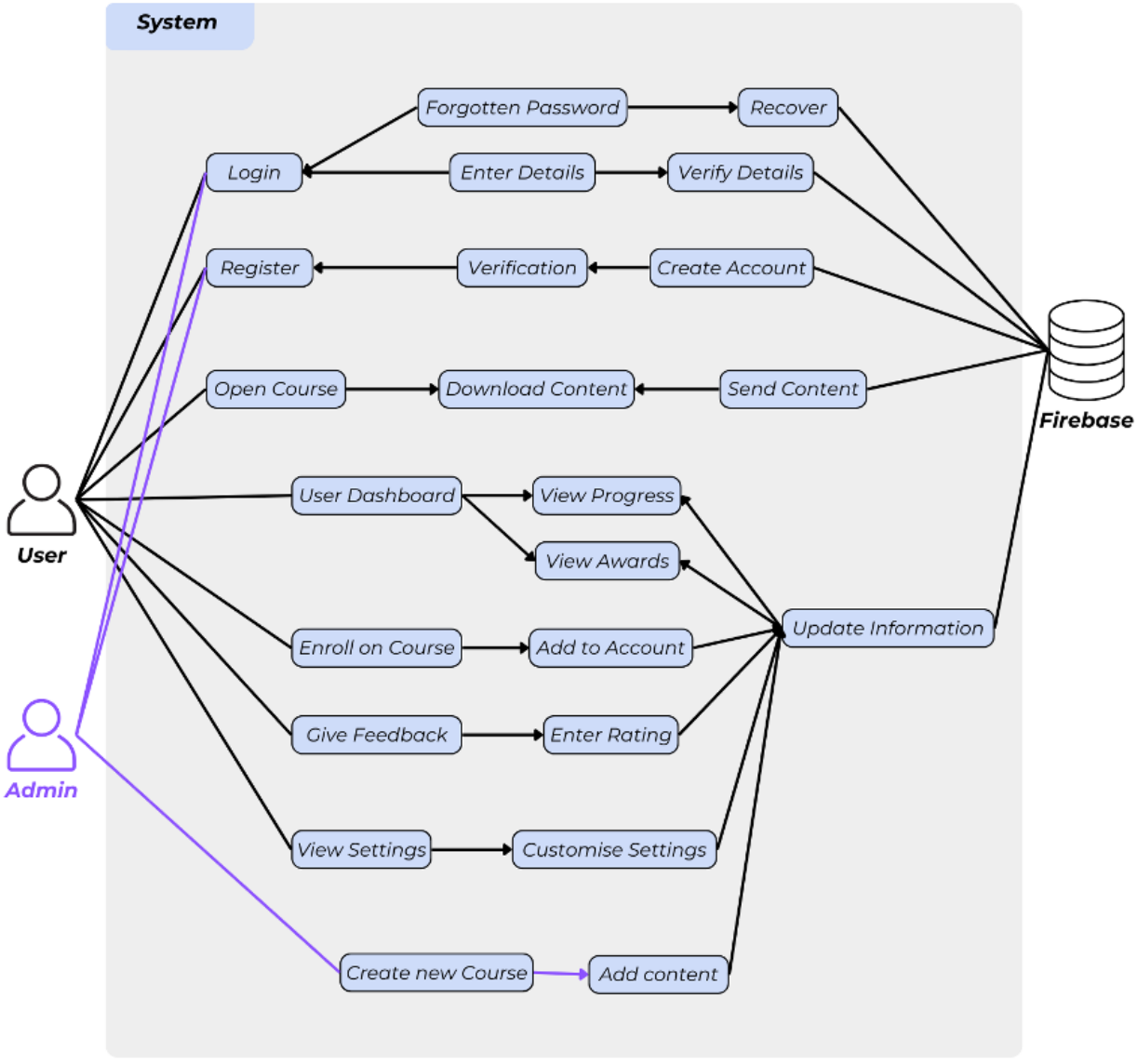

Figure 197 - Use Case Diagram

Student

Use Case	Description	Preconditions	Results	Pass / Fail
UC-01	Student logs in	Student account exists. Student provides valid credentials.	Student is logged in.	Pass
UC-02	Student signs up	Student is not already registered. Student provides valid sign-up credentials.	New student is signed up.	Pass
UC-03	Student enrols in course	Student is logged in.	Student is enrolled in the course.	Pass
UC-04	Student reads a text lecture	Student is logged in. Student is enrolled in course.	Student is shown text lecture.	Pass
UC-05	Student watches a video lecture	Student is logged in. Student is enrolled in course.	Video lecture starts playing.	Pass
UC-06	Student takes a quiz	Student is logged in. Student is enrolled in course.	Quiz details are displayed. Quiz answers are marked and stored. Quiz progress is stored.	Pass
UC-07	Student tracks progress	Student is logged in. Student is enrolled in course.	Course progress is displayed. Completion status is displayed.	Pass
UC-08	Student leaves feedback	Student is logged in. Student is enrolled on course. Student has completed course.	Feedback mechanism is displayed. Result is stored in database.	Pass

Table 9 - Student Use Case Tests

Teacher

Use Case	Description	Preconditions	Results	Pass / Fail
UC-01	Teacher Logs in	Teacher account exists. Teacher provides valid credentials.	Teacher is logged in.	Pass
UC-02	Teacher Signs up	Teacher is not already registered. Teacher provides valid sign-up credentials. Teacher requests rights.	New Teacher is signed up.	Pass
UC-03	Teacher Creates a course	Teacher is signed in.	New course is created.	Pass
UC-04	Teacher creates curriculum	Teacher is signed in. Course Exists.	Curriculum is added to the course.	Pass
UC-05	Teacher creates a quiz	Teacher is signed in. Course Exists.	Quiz is added to the course.	Pass
UC-06	Teacher edits a course	Teacher is signed in. Course Exists. Teacher created the course.	Course edit screen opens.	Pass
UC-07	Teacher deletes a course	Teacher is signed in. Course Exists. Teacher created the course.	Course is deleted.	Pass
UC-08	Teacher changes profile information	Teacher is logged in.	Teacher information is updated in database.	Pass

Table 10 - Teacher Use Case Tests

Admin

Use Case	Description	Preconditions	Results	Pass / Fail
UC-01	Admin Logs in	Admin account exists. Admin provides valid credentials.	Admin is logged in.	Pass
UC-02	Admin Signs up	Admin is not already registered. Admin provides valid sign-up credentials. Admin requests rights.	New Admin is signed up.	Pass
UC-03	Admin Creates a course	Admin is signed in.	New course is created.	Pass
UC-04	Admin creates curriculum	Admin is signed in. Course Exists.	Curriculum is added to the course.	Pass
UC-05	Admin creates a quiz	Admin is signed in. Course Exists.	Quiz is added to the course.	Pass
UC-06	Admin edits a course	Admin is signed in. Course Exists.	Course edit screen opens.	Pass
UC-07	Admin deletes a course	Admin is signed in. Course Exists.	Course is deleted.	Pass
UC-08	Admin changes profile information	Admin is logged in.	Admin information is updated in database.	Pass
UC-09	Admin changes a user's roles	Admin is signed in. User exists.	User role is changed.	Pass

Table 11 - Admin Use Case Tests

As shown in the test results from individual tests done throughout implementation, and the use case testing completed above, the application meets its success criteria.

7.3 Vulnerability Test Results

In addition to testing functionality and usability, especially paramount for a cyber security platform, conducting vulnerability tests will ensure the integrity of the e-learning platform. The application will undergo several tests to ensure it is not susceptible to any major vulnerabilities or common attacks such as SQL injection, Cross Site Scripting, Directory enumeration or bashing. Conducting these tests will help to identify any potential security flaws which can be evaluated and fixed.

Test	Description	Expected Outcome	Actual Outcome	Pass /Fail
Directory Enumeration Tests				
DE-001	Attempt to access sensitive directories.	Access should be restricted.	Access restricted.	Pass
DE-002	Attempt to access sensitive files.	Access should be restricted.	Access restricted.	Pass
DE-003	Enumerate directory structure.	Directory structure should not be disclosed.	Structure not disclosed.	Pass
SQL Injection Tests				
SQLI-001	Attempt SQL injection in login form.	Login should be denied.	Login denied.	Pass
SQLI-002	Attempt SQL injection in search functionality.	Search should return valid results.	Valid results returned.	Pass
SQLI-003	Attempt SQL injection in URL parameters.	URL parameters should be sanitised.	Parameters sanitised.	Pass

Cross Site Scripting Tests				
XSS-001	Attempt reflected XSS attack.	Script should not be executed.	Script not executed.	Pass
XSS-002	Attempt stored XSS attack.	Script should not be stored.	Script not stored.	Pass
XSS-003	Attempt DOM-based XSS attack.	Script should not be executed.	Script not executed.	Pass
Authorisation Tests				
AUTH-001	Weak Password Policy Test.	Enforce strong password policies.	Enforce strong password policies.	Pass
AUTH-002	Session Management Test.	Secure session token management.	Tokens managed securely.	Pass
AUTH-003	Role-Based Access Control (RBAC) Test.	Proper enforcement of role permissions.	Permissions enforced properly.	Pass
AUTH-004	Insecure Direct Object References (IDOR) Test.	Prevention of unauthorised access.	Unauthorised access prevented.	Pass
Firebase Tests				
FIRE-001	Firebase Security Rules Test.	Properly configured security rules.	Rules configured properly.	Pass
FIRE-002	Real-Time Database Security Test.	Prevention of unauthorised access.	Access restricted.	Pass
FIRE-003	Cloud Firestore Security Test.	Properly restricted access to data.	Access restricted.	Pass

Table 12 - Vulnerability Tests

7.5 Summary

This section has outlined the testing procedures used within the project's development and continued to provide a holistic view of use case testing and vulnerability testing. The platform has throughout only had minor errors which were caught inside of each implementation cycle and corrected immediately. While automated testing would have been beneficial to optimise the workflow, and ensure no regressions occur from bug fixes - the scale and complexity of this program did not justify the time expenditure on automation.

Chapter 8
Evaluation

8.1 Evaluation against Objectives

The original objectives, as outlined in the first chapter of this document are as follows:

1. To develop a platform for the dissemination of cyber-security knowledge.

2. To curate content which will comprehensively cover the key aspects of current cyber threats.

3. To incorporate real-world scenarios and quizzes to reinforce learning and assess users' understanding of the content.

4. To evaluate the platform's effectiveness at improving users’ awareness and online safety practices.

From these objectives outlined in Chapter 1, extensive research was carried out in Chapter 2 to outline the problems scope and existing solutions, then methodologies and production approaches were considered in Chapter 3 and requirements for success set in Chapter 4. From all this, a platform has been developed to disseminate cyber security knowledge effectively - utilising specifically curated content which addresses the key aspects of contemporary cyber security threats. Real world scenarios and quizzes have been incorporated to solidify learning efforts and assess users’ comprehension. With the first three objectives met, it can be guaranteed that the platform will improve users’ general knowledge and awareness on cyber security issues and online safety practices.

8.2 Evaluation against Requirements

8.2.1 Functional Requirements

The following table will evaluate the implementation of functional requirements set in chapter 4, where they were given priorities of ‘Must have’, ‘Should have’, ‘Could have’ and ‘Won't Have’.

ID	Requirement Name	Priority	Result	Evaluation
FR1	Login Authentication	Must Have	Yes	Authentication implemented successfully using Firebase Authentication.
FR2	Account Creation	Must Have	Yes	Users are able to register by using an email or a Google account.
FR3	Account Customisation	Must Have	Yes	Users can change their display name and select one of 6 avatars to show.
FR4	User Role Management	Must Have	Yes	Users are split into 3 roles: student, admin, and teacher. All with different access levels.
FR5	Course Enrolment	Must Have	Yes	Users can enrol in courses, creating a unique database table for userProgress.
FR6	Course Creation	Must Have	Yes	Teachers and admins can create courses. Teachers and admins can add content through text lessons, video lessons and quizzes.
FR7	Course Management	Must Have	Yes	Teachers and admins can modify content within courses such as lessons or quizzes freely. Teachers and admins can delete courses freely.

FR8	Database System	Must Have	Yes	All data is stored with Firebase, allowing for instant read and writes with multiple tables dynamically updating in real time.
FR9	Content Delivery Mechanisms	Must Have	Yes	Students can access all courses by enrolling, at which point they can complete parts of a course and leave whenever they like.
FR10	Assessment Mechanisms	Must Have	Yes	Quizzes are scored automatically, and a value is stored in the userProgress table for their scores.
FR11	Progress Tracking	Should Have	Yes	Progress is stored as an array of completed lessons and as a completion % in Firestore. Updated every time a lesson or quiz is finished.
FR12	Course Feedback	Should Have	Yes	At the end of a course, users can give a star rating, which is added to an array to give an average course rating out of 5.
FR13	Course Statistics tool	Could Have	No	There was not enough data available to provide meaningful statistics.
FR14	Notifications and Alerts	Could Have	No	Notifications were set up with firestore, to message users when they are inactive. However, without 24/7 hosting this system will not work.
FR15	Real-Time Communications	Could Have	No	Real time communication was possible with Firebase, however unfeasible to set up and not necessary for the platforms core functionality.
FR16	Streak Feature	Could Have	Yes	A streak feature was implemented to track a user's logins, giving

				them a streak increase every consecutive day they sign in.
FR17	Certificates / Badges	Could Have	No	Completing a course does show on the dashboard, however there is currently no form of certificate or badge given for completion.
FR18	Time Tracking	Could Have	No	Tracking user time would provide useful statistics, however it would require a live server to track user interactions.
FR19	Game-ified Learning	Won't Have	No	The scope of the project and the complexity of this feature meant it was not possible to implement.
FR20	Live Tutorials	Won't Have	No	Since the platform is not live hosted, and there are no active teachers, live tutorials were not possible.

Table 13 - Functional Requirements Evaluation

As shown in the table above, all requirements identified as 'Must have' or 'Should have' were successfully implemented. Even some 'Could haves' such as the Streak feature were included. Due to constraints on budget and production time, which meant not hosting the application 24/7 as a live platform, some of the features identified were not able to be successfully implemented - there does exist however strong foundations for their implementation in the future.

8.2.2 Non-Functional Requirements

The following table will evaluate the implementation of all Non-functional requirements set in chapter 4, where they were also given priorities of 'Must have', 'Should have' and 'Could have'.

ID	Name	Priority	Result	Evaluation
NFR1	Compatibility	Should Have	Yes	The application uses contextual media queries to shrink and grow widgets to fit different sizes. The application runs best in the aspect ratio 16:9.
NFR2	Security	Must Have	Yes	Utilising Firebase for authentication and data storage ensures strong data security practices. Mitigating any potential risks by allowing Google to take the responsibility.
NFR3	Availability	Must Have	Partially	The platform is currently not hosted on a web server; however, it is designed in a way to allow for multiple users to access it at once when it is hosted.
NFR4	Performance	Should Have	Yes	Throughout testing, several use case tests were performed across the whole application with little to no errors occurring.
NFR5	Usability	Must Have	Yes	From the beginning, usability best practices were incorporated into the design. The final product uses a clean sans serif font, blue accent colours and clear button/form names for users to use the application with little help or confusion.
NFR6	Scalability	Could Have	Yes	Utilising Firebase for authentication and database management, there is theoretically no maximum user base so long as you have the budget. The codebase itself is also prepared for future enhancements.

Table 14 - Non-Functional Requirements Evaluation

As shown above, all the non-functional requirements have been met and, in many cases, exceeded beyond the original descriptions except for availability which cannot be fully met until the server is hosted on a web server continuously.

8.3 Evaluation of Project Methodology

The use of Agile methodology in this project proved to be highly effective and well suited to the scope and circumstances. Employing Gantt charts and Kanban boards allowed activities and tasks to be tracked throughout the iteration sprints - streamlining the development process. The approach taken allowed for adaptability as the requirements changed and ensured the project stayed on schedule, especially important given the short development timeline. Conducting iteration tests proved very successful, allowing several small errors to be identified and corrected before propagating into the codebase and inflating in size, overall allowing the final product to be of high quality. The iterative nature of agile development provided the flexibility needed to prioritise tasks effectively and adjust mid-production. Overall, the use of agile methodology was instrumental in achieving the projects' requirements and objectives efficiently and effectively.

8.4 Evaluation against Existing Solutions

The following table compares the features of the application developed in this project against those identified during the analysis of existing solutions in the literature review.

Feature	Project Application	Certified Secure	GCF Global Internet Safety	Phishing Box attack Simulation
Login System & Progress Tracking	Yes	Yes	Yes	No
Free Online Safety Module	Yes	Yes	Yes	Limited (Only free quiz)
Online Quiz Activities	Yes	Yes	Yes, end of module quiz	Yes
Feedback on quizzes	No	No feedback provided	No feedback provided	Limited feedback
Video Content	Yes	Provided linked below but not contextualised	Informative videos provided, backing up the text content.	No
Text Content	Yes	Provided linked below but not contextualised	Modules are mostly made of text content	No
Website Usability	Yes	Old, hard to navigate, lacks structure	Structured and staged learning	N/A
Learning Approach	Courses on specific topics, containing sections and linear lessons	Focus on certifications and examination	Module by module, not entirely linear	N/A
Use of Graphics	No	Used in some of the PDF content and videos	Used throughout to help users understand information	Quiz includes graphics of phishing emails

Feedback Mechanisms	Feedback via course rating	No feedback on quizzes	No feedback in final quiz	Limited feedback
Overall Content Accessibility	Easy to find all content by search or filter.	Hard to find content and navigate website	Well-structured web pages easy to navigate	N/A

Table 15 - Existing Solutions Evaluation

From the table, it is clear that the final application produced contains most of the features identified across all of the alternative solutions - as well as containing the most overall features. The only parts where it lacks are in the use of graphics and diagrams to support learning, and feedback mechanisms for quizzes. Both features could easily be implemented in the project's future development.

Compared to Phishing Box, the project application provides a much fuller learning experience with several more features being included. Certified secure and GCF globals applications provide a similar learning experience, but the linear nature and progress tracking abilities of the project application contribute to a more streamlined, enjoyable, and interactive learning experience - which results in high user satisfaction, user retention and successfully lowers users' vulnerability to cyber threats.

8.5 Summary

This chapter has evaluated the project's outcomes against predefined objectives, requirements and existing solutions, highlighting the strengths and areas for improvement in the final application, and offering a straightforward assessment of its performance in the cyber-security education landscape.

This chapter has also evaluated the project's development process, its alignment with the objectives and the impact which the chosen methodology has had on the project's outcome.

Chapter 9
Conclusion

9.1 Project Conclusion

This project set out to tackle the pressing issue of cybersecurity awareness and education, aiming to equip users with the knowledge and skills needed to navigate the digital landscape safely. The development of an e-learning platform served as a solution to motivate users to actively engage in cybersecurity best practices. The final product largely accomplishes this, prioritising educational content whilst incorporating interactive elements to enhance learning experiences.

By fulfilling the objectives outlined in Chapter 1 and implementing a significant portion of the requirements detailed in Chapter 4, the application effectively promotes cybersecurity literacy and encourages users to enhance their online security habits. Through features such as course enrolment, curriculum development, and interactive quizzes, the platform fosters a culture of continuous learning and skill-building in cybersecurity.

Throughout the project journey, invaluable skills were acquired, encompassing effective project management, software development expertise, strategic planning, and research acumen. The endeavour shed light on the complexities inherent in developing an e-learning platform tailored to cybersecurity education. Notably, features like course management and interactive quizzes underscored the intricacies and challenges involved in delivering engaging educational content in the cybersecurity domain.

Despite constraints in time and resources, the project successfully implemented core features, whilst highlighting the potential for further refinement and expansion with additional resources and time investment. As the project concludes, it stands as a testament to the importance of cybersecurity education and the role of technology in empowering users to navigate the digital landscape securely.

9.2 Future work

The application itself was designed to continue beyond the scope of this project, including mechanisms for admins to moderate roles and users, and for teachers to upload, edit and delete content within the application itself. If hosted online, the platform could be left to grow indefinitely with the only constraints being the willingness of educators to curate and add content to the platform.

The following sections include some of the functional requirements set during chapter 4 which were assigned a low priority due to complexity and time constraints. All of these features could be considered for future work and the foundations have been set for them to be built from.

9.2.1 Real Time Communication

As the platform grows and becomes hosted, there is potential for real-time communication integration, say through a messages/inbox section, which would allow both student-student communication as well as student-teacher communications. This feature would require a more complex web server, however as the platform grows it would have its merits, allowing teachers and students to communicate and clarify complex learning components.

9.2.2 Game-based learning

A feature far too complex for the scope of this project, the addition of game-based learning activities showed in the literature review to be an invaluable teaching tool. It allows for a far more interactive learning experience and can be a great mechanism to keep users engaged. It would require significant alterations to the codebase and expertise in programming languages specific to game development such as C++ and Java.

9.2.3 Certificates and Badges

This feature would not require a lot to implement in the future of this project as content which is uploaded to the platform increases in quality and depth, users could receive certificates which prove the completion of certain courses. This practice has already been seen in other learning platforms such as ‘tryhackme’ or ‘LinkedIn learning’, however it requires the issuing platform to have a much greater credibility than the project currently has.

9.2.4 Course statistic tools

As the user base grows, and there becomes a need to increase the quality of content, statistics from courses will become a lot more important. It would require a significant amount more data collection such as completion times, scores, or number of attempts - however this data could be used to streamline future courses and improve the overall quality of the content on the platform.

9.3 Summary

This final chapter concludes the cyber security e-learning platform project, highlighting achievements and lessons learned as well as discussing the potential for future enhancements such as game-based learning, certificates, and statistical monitoring. Despite facing constraints, the project has not only managed to promote and encourage cyber security literacy, but also has laid a strong foundation for future growth and refinement.

Chapter 10
References

Kaspersky Lab. (2016, October 3). *Elderly people online: habits and concerns*. Kaspersky.co.uk; Kaspersky. https://www.kaspersky.co.uk/blog/older-people-internet/7736/

Adenowo, A., & Adenowo, B. (2013, July). *Software Engineering Methodologies: A Review of the Waterfall Model and Object- Oriented Approach*. Researchgate; International Journal of Scientific & Engineering Research. https://www.researchgate.net/profile/Adetokunbo_Adenowo/publication/344194737_Software_Engineering_Methodologies_A_Review_of_the_Waterfall_Model_and_Object-_Oriented_Approach/links/5f5a803292851c07895d2ce8/Software-Engineering-Methodologies-A-Review-of-the-Waterfall-Model-and-Object-Oriented-Approach.pdf

Agile Manifesto. (2019). *Manifesto for Agile Software Development*. Agilemanifesto.org. https://agilemanifesto.org/iso/en/manifesto.html

Ahmad, M. O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. https://doi.org/10.1109/seaa.2013.28

Alhaddad, M., Mohd, M., Qamar, F., & Imam, M. (2023). *Study of Student Personality Trait on Spear-Ing Susceptibility Behavior*. *14*(5). https://doi.org/10.14569/ijacsa.2023.0140571

Andaluri, N. B., Srivastava, A., & Andaluri, R. (2023). A Critical Study of Social Engineering vis-a-vis Phishing Attacks. In *Research Gate*.

https://www.researchgate.net/publication/374259835_A_Critical_Study_of_Social_Engineering_vis-a-vis_Phishing_Attacks

Andreassen, C. S., Torsheim, T., & Pallesen, S. (2014). Use of online social network sites for personal purposes at work: does it impair self-reported performance?1. *Comprehensive Psychology*, *3*(1), Article 18. https://doi.org/10.2466/01.21.cp.3.18

Arora, H., Manglani, T., Bakshi, G., & Choudhary, S. (2022, March 1). *Cyber Security Challenges and Trends on Recent Technologies*. IEEE Xplore. https://doi.org/10.1109/ICCMC53470.2022.9753967

Brown, A. R., & Voltz, B. D. (2005). Elements of Effective e-Learning Design. *The International Review of Research in Open and Distributed Learning*, *6*(1). https://doi.org/10.19173/irrodl.v6i1.217

Cercone, K. (2008). Characteristics of Adult Learners with Implications for Online Learning Design. *AACE Journal*, *16*(2), 137–159. https://www.learntechlib.org/primary/p/24286/

Cernat, M. (2013). ETHICAL CHALLENGES OF THE E-LEARNING SYSTEM. In *Challenges of the Knowledge Society. Education and Sociology*.

Certified Secure. (2010). *Certified Secure*. Www.certifiedsecure.com. https://www.certifiedsecure.com/frontpage

Chinyio, E., & Morton, N. (2006). The Effectiveness of E-learning. *Architectural Engineering and Design Management*, *2*(1-2), 73–86. https://doi.org/10.1080/17452007.2006.9684606

Esfahani, S., & Mohit, H. (2023). *The Effect of Cyberloafing of Remote Working Employees on The Effect of Cyberloafing of Remote Working Employees on Noncompliance*

Behavior Noncompliance Behavior. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1005&context=irais2023

Falade, P. (2023). Polra Victor Falade Int. *J. Sci. Res. Comput. Sci. Eng. Inf. Technol*, *9*(5), 185–198. https://doi.org/10.32628/CSEIT2390533

Fornaciari, C. J., & Lund Dean, K. (2013). The 21st-Century Syllabus. *Journal of Management Education*, *38*(5), 701–723. https://doi.org/10.1177/1052562913504763

Fry, K. (2001). E-learning markets and providers: some issues and prospects. *Education + Training*, *43*(4/5), 233–239. https://doi.org/10.1108/EUM0000000005484

GCF Global. (2014). *Free Internet Safety Tutorial at GCFGlobal*. GCFGlobal.org. https://edu.gcfglobal.org/en/internetsafety/

Gorbachenko, P. (2021, April 9). *Functional vs Non-Functional Requirements [Updated 2021]*. Enkonix.com. https://enkonix.com/blog/functional-requirements-vs-non-functional/

Gordon, N. (1998). Colour blindness. *Public Health*, *112*(2), 81–84. https://doi.org/10.1038/sj.ph.1900446

Holmes, G., & Abington-Cooper, M. (2018). *JOTS v26n2 - Pedagogy vs. Andragogy: A False Dichotomy?* Virginia Tech Scholarly Communication University Libraries. https://scholar.lib.vt.edu/ejournals/JOTS/Summer-Fall-2000/holmes.html

Horton, W. (2011). e-Learning by Design. In *Google Books*. John Wiley & Sons. https://books.google.co.uk/books?hl=en&lr=&id=3EXIYh-2TXYC&oi=fnd&pg=PA1&dq=best+practices+for+e+learning+design&ots=BO_i0QvlyT&sig=VrF_23147VjjA-

J25AMUbXlV398&redir_esc=y#v=onepage&q=best%20practices%20for%20e%20learning%20design&f=false

Huang, C.-C., & Kusiak, A. (1996). Overview of Kanban systems. *International Journal of Computer Integrated Manufacturing*, *9*(3), 169–189. https://doi.org/10.1080/095119296131643

IT Governance UK. (2023, June 1). *List of Data Breaches and Cyber Attacks in 2023*. IT Governance UK Blog. https://www.itgovernance.co.uk/blog/list-of-data-breaches-and-cyber-attacks-in-2023

Jenkins, B. D., Golding, J. M., Le Grand, A. M., Levi, M. M., & Pals, A. M. (2022). When Opportunity Knocks: College Students' Cheating Amid the COVID-19 Pandemic. *Teaching of Psychology*, 407–419. https://doi.org/10.1177/00986283211059067

Knowles, M. (1984). *Knowles, M. S. (1984). The Adult Learner A Neglected Species (3rd Edition). Houston, TX Gulf Publishing. - References - Scientific Research Publishing*. Www.scirp.org. https://www.scirp.org/reference/ReferencesPapers?ReferenceID=2326729

Kumar, G., & Kumar, P. (2012, August). *Impact of Agile Methodology on Software Development Process*. International Journal of Computer Technology and Electronics Engineering (IJCTEE). https://www.researchgate.net/profile/Gaurav-Kumar-175/publication/255707851_Impact_of_Agile_Methodology_on_Software_Development_Process/links/00b49520489442e12d000000/Impact-of-Agile-Methodology-on-Software-Development-Process.pdf

Lallie, H. S., Shepherd, L. A., Nurse, J. R. C., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021). Cyber Security in the Age of COVID-19: A Timeline and Analysis of Cyber-Crime and Cyber-Attacks during the Pandemic. *Computers & Security*, *105*(1), 1–20. https://doi.org/10.1016/j.cose.2021.102248

Lau, R. W. H., Yen, N. Y., Li, F., & Wah, B. (2013). Recent development in multimedia e-learning technologies. *World Wide Web*, *17*(2), 189–198. https://doi.org/10.1007/s11280-013-0206-8

Mccormick, M. (2012). *Waterfall vs. Agile Methodology*. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

Moran, M., & Hart, A. (2023). An Experiment on Why You Are Vulnerable To Online Phishing Scams. *Undergraduate Research and Engagement Symposium*. https://ecommons.luc.edu/ures/2023/2023/126/

Muirhead, R. J. (2007). E-learning: Is This Teaching at Students or Teaching With Students? *Nursing Forum*, *42*(4), 178–184. https://doi.org/10.1111/j.1744-6198.2007.00085.x

Muresan, M. (2014). Using Cybergogy and Andragogy Paradigms in Lifelong Learning. *Procedia - Social and Behavioral Sciences*, *116*, 4722–4726. https://doi.org/10.1016/j.sbspro.2014.01.1015

Norton. (2022). *2022 Norton Cyber Safety Insights Report: Special Release – Online Creeping | NortonLifeLock*. Www.nortonlifelock.com. https://www.nortonlifelock.com/us/en/newsroom/press-kits/2022-norton-cyber-safety-insights-report-special-release-online-creeping/

O'Connor, Z. (2011). Colour Psychology and Colour Therapy: Caveat Emptor. *Color Research & Application*, *36*(3), 229–234. https://doi.org/10.1002/col.20597

Odhaib, M. (2018). Does E-Learning Give a Better Result than Traditional Learning? *International Journal of Computer Science and Mobile Computing*, *7*(9), 29–36.

Office for National Statistics. (2022, September 26). *Nature of fraud and computer misuse in England and Wales - Office for National Statistics*. Www.ons.gov.uk. https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/articles/natureoffraudandcomputermisuseinenglandandwales/yearendingmarch2022

Pande, J. (2017). *Introduction to Cyber Security*. Uttarakhand Open University. https://www.academia.edu/download/59862224/FCS20190625-18244-nzm6g9.pdf

Phishing Box. (2022). *Phishing Test - Free Phishing Security Test by PhishingBox*. Www.phishingbox.com. https://www.phishingbox.com/phishing-iq-test/quiz.php?reset=1

Ricci, J., Breitinger, F., & Baggili, I. (2018). Survey results on adults and cybersecurity education. *Education and Information Technologies*, *24*(1), 231–249. https://doi.org/10.1007/s10639-018-9765-8

Ringström, S. (2023). *http://his.diva-portal.org/smash/record.jsf?pid=diva2%3A1802013*. DIVA. https://his.diva-portal.org/smash/record.jsf?pid=diva2%3A1802013&dswid=-7732

Shank, P. (2005). The Value of Multimedia in learning. *Adobe Motion Design Center (2005)*.

Sikra, J., Renaud, K. V., & Thomas, D. R. (2023). UK cybercrime, victims and reporting : a systematic review. *Commonwealth Cybercrime Journal*, *1*(1), 28–59. https://strathprints.strath.ac.uk/84979/

Symantec. (2016). *symantec_1115_01*. Www.nortonlifelock.com. https://www.nortonlifelock.com/us/en/newsroom/press-releases/2016/symantec_1115_01/

Uma, M., & Padmavathi, G. (2013). A Survey on Various Cyber Attacks and Their Classification. *International Journal of Network Security*, *15*(5), 390–396. http://ijns.jalaxy.com.tw/contents/ijns-v15-n5/ijns-2013-v15-n5-p390-396.pdf

Wakode, R. B., Raut, L. P., & Talmale, P. (2015). Overview on Kanban Methodology and its Implementation. *International Journal for Scientific Research & Development*|, *3*(02), 2518–2521.

Zhang, D., Zhou, L., Briggs, R. O., & Nunamaker, J. F. (2006). Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. *Information & Management*, *43*(1), 15–27. https://doi.org/10.1016/j.im.2005.01.004

Zhang, Y., Xiao, Y., Ghaboosi, K., Zhang, J., & Deng, H. (2011). A survey of cyber crimes. *Security and Communication Networks*, *5*(4), 422–437. https://doi.org/10.1002/sec.331

Appendix

PID

Ethics Certificate